

---

# **Enterprise DBA Part 2: Performance and Tuning**

**Volume 2 • Instructor Guide**

---

30052GC10

Production 1.0

September 1999

M09216

**ORACLE®**

**Author**

Dominique Jeunot

**Technical Contributors  
and Reviewers**

Bruce Ernst

Richard Foote

Antonio Florindo

Steven George

Joel Goodman

Scott Gossett

Lex de Haan

Donna Hamby

Scott Heisey

John Hough Jr.

Peter Kilpatrick

Kurt Lysy

Michael Moller

Howard Ostrow

Thomas Raes

Shankar Raman

S. Roo

Ulrike Schwinn

Roger Simon

Anthony Woodell

**Publisher**

Kelly Lee

Copyright © Oracle Corporation, 1999. All rights reserved.

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

**Restricted Rights Legend**

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

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

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

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

Oracle is a registered trademark and Oracle and all Oracle products are trademarks or registered trademarks of Oracle Corporation.

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

## **Lesson 1: Course Introduction**

Objectives 1-2

## **Lesson 2: Tuning Overview**

Objectives 2-2

System Tuning Overview 2-3

Tuning Goals 2-5

Tuning Steps 2-7

Summary 2-8

## **Lesson 3: Oracle Alert and Trace Files**

Objectives 3-2

Diagnostic Information 3-3

The Alert Log File 3-4

Controlling the Alert Log File 3-7

Controlling the Background Processes Trace Files 3-8

User Trace Files 3-11

Controlling the User Trace Files 3-12

Summary 3-14

Quick Reference 3-15

## **Lesson 4: Utilities and Dynamic Performance Views**

Objectives 4-2

Views, Utilities, and Tools 4-3

Dictionary and Special Views 4-5

Dynamic Troubleshooting and Performance Views 4-6

Topics for Troubleshooting and Tuning 4-7

Collecting System-Wide Statistics 4-9

Collecting Session-Related Statistics 4-11

UTLBSTAT and UTLESTAT Utilities 4-14

Examining the Statistics Report 4-17

Library Cache Statistics Section 4-21

I/O Statistics Section 4-22

Latches 4-23

Latch Types	4-25
Oracle Wait Events	4-26
Statistics Event Views	4-29
Event Management System	4-34
Predefined Event Tests	4-36
Event Frequency and Parameters	4-43
Fix the Problem Detected by the Event	4-45
DBA-Developed Tools	4-47
Oracle Packs	4-48
Performance Manager	4-50
TopSessions	4-52
Oracle Tablespace Manager	4-57
Oracle Trace Manager	4-58
Oracle Expert	4-60
Tuning Categories	4-61
Tuning Recommendations	4-62
Summary	4-64
Quick Reference	4-65

## **Lesson 5: Tuning the Shared Pool**

Objectives	5-2
The Shared Global Area	5-3
The Shared Pool	5-4
The Library Cache	5-5
Tuning the Library Cache	5-7
Terminology	5-9
Diagnostic Tools for Tuning the Library Cache	5-10
Shared Cursors	5-11
Guidelines	5-12
Invalidations	5-14
Sizing the Library Cache	5-15
Global Space Allocation	5-16
Large Memory Requirements	5-18

Tuning the Shared Pool Reserved Space	5-20
Keeping Large Objects	5-22
Anonymous PL/SQL Blocks	5-23
Other Parameters That Affect the Library Cache	5-24
The Data Dictionary Cache	5-26
Diagnostic Tools	5-27
Tuning the Data Dictionary Cache	5-28
Guidelines	5-29
User Global Area and Multithreaded Server	5-30
Sizing the User Global Area	5-31
The Large Pool	5-32
Summary	5-35
Quick Reference	5-36

## **Lesson 6: Tuning the Buffer Cache**

Objectives	6-2
Buffer Cache Overview	6-3
Managing the Buffer Cache	6-5
Tuning Goals and Techniques	6-8
Diagnostic Tools for Tuning the Buffer Cache	6-10
Cache Hit Ratio	6-11
Guidelines for Using the Cache Hit Ratio	6-12
Using Multiple Buffer Pools	6-15
Defining Multiple Buffer Pools	6-16
Enabling Multiple Buffer Pools	6-19
Sizing Buffer Pools	6-20
Recycle Buffer Pool Guidelines	6-21
Calculating the Buffer Pool Hit Ratio	6-24
Segments for the Keep and Recycle Buffer Pools	6-26
Dictionary Views with Buffer Pools	6-27
Other Performance Indicators	6-28
Caching Tables	6-29
LRU Latches	6-30

LRU Latch Tuning Goals	6-31
Diagnosing LRU Latch Contention	6-32
Resolving LRU Latch Contention	6-33
Free Lists	6-34
Diagnosing Free List Contention	6-35
Resolving Free List Contention	6-37
Summary	6-38
Quick Reference	6-39

## **Lesson 7: Tuning the Redo Log Buffer**

Objectives	7-2
The Redo Log Buffer	7-3
Sizing the Redo Log Buffer	7-4
Tuning the Redo Log Buffer	7-5
Diagnostic Tools for Tuning the Redo Log Buffer	7-6
Guidelines for Tuning the Redo Log Buffer	7-8
Reducing Redo Operations	7-11
Summary	7-13
Quick Reference	7-14

## **Lesson 8: Database Configuration and I/O Issues**

Objectives	8-2
Overview	8-3
Tablespace Usage	8-4
Distributing Files Across Devices	8-6
Oracle File Striping	8-8
Full Table Scans	8-10
Diagnostic Tools	8-13
Using I/O Statistics in report.txt	8-15
Online Redo Log File Configuration	8-16
Archive Log File Configuration	8-19
Tuning Checkpoint	8-22
Checkpoint Tuning Guidelines	8-23
Multiple I/O Slaves	8-25

Initialization Parameters	8-27
Multiple DBWn Processes	8-28
Tuning DBWn I/O	8-29
Summary	8-30
Quick Reference	8-31

## **Lesson 9: Using Oracle Blocks Efficiently**

Objectives	9-2
Database Storage Hierarchy	9-3
Allocating an Extent	9-4
Avoiding Dynamic Space Management	9-5
Large Extents	9-7
Database Block Size	9-9
Oracle Block Size	9-10
Block Size Advantages and Disadvantages	9-11
Block Packing factors	9-13
Guidelines for Setting the Packing Factor	9-15
Migration and Chaining	9-16
Detecting Chaining and Migration	9-17
Selecting Migrated and Chained Rows	9-18
Eliminating Migrated Rows	9-19
The High-Water Mark	9-21
Table Statistics	9-22
The DBMS_SPACE Package	9-23
Index Reorganization	9-26
Monitoring and Rebuilding Indexes	9-27
Summary	9-30
Quick Reference	9-31

## **Lesson 10: Optimizing Sort Operations**

Objectives	10-2
Sort Operations	10-3
Sort Process	10-5
Sort Area and Parameters	10-7

Sort Process and Temporary Space	10-11
Tuning Sort Operations	10-13
Avoiding Sort Operations	10-14
Diagnostic Tools for Tuning Sort Operations	10-16
Diagnostics and Guidelines	10-18
Monitoring Temporary Tablespaces	10-19
Configuring Temporary Tablespaces	10-20
Summary	10-22
Quick Reference	10-23

## **Lesson 11: Tuning Rollback Segments**

Objectives	11-2
Rollback Segment Usage	11-3
Rollback Segment Activity	11-4
Rollback Segment Header Activity	11-5
Growth of Rollback Segments	11-6
Transaction Types	11-7
Tuning the Rollback Segments	11-9
Diagnostic Tools for Tuning Rollback Segments	11-10
Diagnosing Rollback Segment Header Contention	11-12
Guidelines: How Many Rollback Segments?	11-15
Guidelines: Sizing Rollback Segments	11-17
Guidelines: Sizing Transaction Rollback Data	11-18
Sizing Transaction Rollback Data Volume	11-19
Guidelines: Using Less Rollback	11-21
Possible Problems	11-23
Summary	11-24
Quick Reference	11-25

## **Lesson 12: Monitoring and Detecting Lock Contention**

Objectives	12-2
Locking Mechanism	12-3
Types of Locks	12-6
DML Locks	12-8



Table Lock Modes	12-10
Manual Table Lock Modes	12-12
Row-Level Lock in Block	12-16
DDL Locks	12-17
Possible Causes of Lock Contention	12-19
Diagnostic Tools for Monitoring Locking Activity	12-20
TopSessions (Diagnostic Pack)	12-22
Guidelines: Resolve Contention	12-24
Deadlocks	12-26
Summary	12-29
Quick Reference	12-30

## **Lesson 13: SQL Issues and Tuning Considerations for Different Applications**

Objectives	13-2
The Role of the DBA	13-4
Diagnostic Tools Overview	13-5
The EXPLAIN PLAN Statement	13-6
SQL Trace and TKPROF	13-7
Enabling and Disabling SQL Trace	13-9
Formatting the Trace File with TKPROF	13-10
TKPROF Options	13-11
TKPROF Statistics	13-12
SQL*Plus AUTOTRACE	13-13
Optimizer Modes	13-14
Setting the Optimizer Mode	13-16
Managing Statistics	13-18
Table Statistics	13-20
Index Statistics	13-22
Column Statistics	13-23
Histograms	13-24
Copying Statistics Between Databases	13-26
Plan Equivalence	13-29

Creating Stored Outlines	13-30
Using Stored Outlines	13-31
Maintaining Stored Outlines	13-33
Data Access Methods	13-34
B-Tree Indexes	13-35
Bitmap Indexes	13-37
Reverse Key Indexes	13-42
Index-Organized Tables	13-44
Clusters	13-49
Materialized Views	13-52
Query Rewrites	13-55
Materialized Views and Query Rewrites: Example	13-57
Enabling and Controlling Query Rewrites	13-59
OLTP Systems	13-62
DSS Systems	13-67
Multipurpose Applications	13-71
Summary	13-76
Quick Reference	13-79

## **Lesson 14: Managing a Mixed Workload**

Objectives	14-2
Overview	14-3
Resource Management Concepts	14-4
Resource Consumer Groups	14-6
Resource Plan Directives	14-7
Database Resource Management Example	14-9
Steps in Database Resource Management	14-10
Assigning the Resource Manager Privilege	14-11
Creating Database Resource Manager Objects	14-13
Assigning Users to Consumer Groups	14-16
Setting the Resource Plan for an Instance	14-17
Changing a Consumer Group Within a Session	14-18
Changing Consumer Groups for Sessions	14-19

Database Resource Manager Information	14-20
Current Database Resource Manager Settings	14-23
Summary	14-24
Quick Reference	14-25

## **Lesson 15: Tuning with Oracle Expert**

Objectives	15-2
Overview	15-3
Types of Tuning	15-5
Starting Oracle Expert	15-7
Tuning Session Scope	15-9
Data Collection	15-12
Collected Data	15-21
Attributes	15-22
Rules	15-24
Analysis	15-26
Recommendations	15-27
Reports	15-28
Implementation	15-32
Summary	15-33

## **Lesson 16: Multithreaded Server Tuning Issues**

Objectives	16-2
Overview	16-3
Multithreaded Server Characteristics	16-4
Configuring the Multithreaded Server	16-6
Monitoring Dispatchers	16-7
Monitoring Shared Server Processes	16-9
Monitoring Process Usage	16-11
Shared Servers and Memory Usage	16-12
Possible Problems	16-13
Obtaining Dictionary Information	16-14
Summary	16-15
Quick Reference	16-16

## **Lesson 17: Workshop**

Objectives	17-2
Workshop Methodology	17-3
Troubleshooting Scope	17-4
Directories Configuration	17-5
Workshop Database Configuration	17-7
Information Gathering	17-8
Statistics	17-10
Review	17-11
Presentation	17-14
Analysis	17-15
New Statistics	17-17
Results	17-19
Post-Tuning Actions	17-20
Pending Performance Tuning Issues	17-22
Summary	17-23

## **Appendix A: Practices**

Practice 3-1	A-2
Practice 4-1	A-3
Practice 5-1	A-4
Practice 6-1	A-5
Practice 7-1	A-6
Practice 8-1	A-7
Practice 9-1	A-8
Practice 10-1	A-9
Practice 11-1	A-10
Practice 12-1	A-11
Practice 13-1	A-12
Practice 14-1	A-13
Guided Practice 17-1	A-14

## **Appendix B: Practice Hints**

Practice 3-1 Hint	B-2
-------------------	-----

Practice 4-1 Hint	B-3
Practice 5-1 Hint	B-5
Practice 6-1 Hint	B-6
Practice 7-1 Hint	B-7
Practice 8-1 Hint	B-8
Practice 9-1 Hint	B-9
Practice 10-1 Hint	B-10
Practice 11-1 Hint	B-11
Practice 12-1 Hint	B-13
Practice 13-1 Hint	B-14
Practice 14-1 Hint	B-16

## **Appendix C: Practice Solutions**

Practice 3-1 Solutions	C-2
Practice 4-1 Solutions	C-3
Practice 5-1 Solutions	C-8
Practice 6-1 Solutions	C-13
Practice 7-1 Solutions	C-17
Practice 8-1 Solutions	C-18
Practice 9-1 Solutions	C-21
Practice 10-1 Solutions	C-24
Practice 11-1 Solutions	C-27
Practice 12-1 Solutions	C-33
Practice 13-1 Solutions	C-37
Practice 14-1 Solutions	C-44

## **Appendix D: Redundant Arrays of Inexpensive Disks Technology (RAID)**

System Hardware Configuration	D-2
RAID Level 0, Nonredundant Striping	D-5
RAID Level 1, Mirroring	D-6
RAID Level 0+1, Striping and Mirroring	D-8
RAID Level 3, Bit Interleaved Parity	D-9
RAID Level 5, Block-Interleaved with Distributed Parity	D-10
Ranking of RAID Levels Against Oracle File Types	D-12

**Appendix E: Dictionary and Dynamic Performance Views**

Dictionary and Dynamic Performance Views E-2

Data Dictionary Views E-3

Dynamic Performance Views E-21

**Appendix F: Initialization Parameters**

Initialization Parameters F-2

Parameters Definition F-4

## **Managing a Mixed Workload**

## Objectives

### Objectives

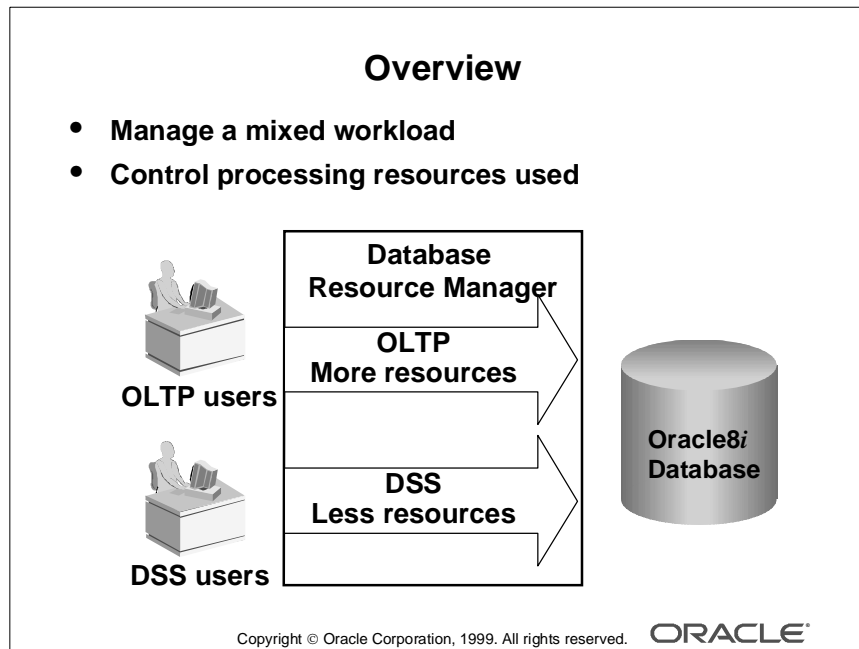
**After completing this lesson, you should be able to do the following:**

- **List the features of Database Resource Manager**
- **Limit the use of resources using Database Resource Manager**

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®



## Overview



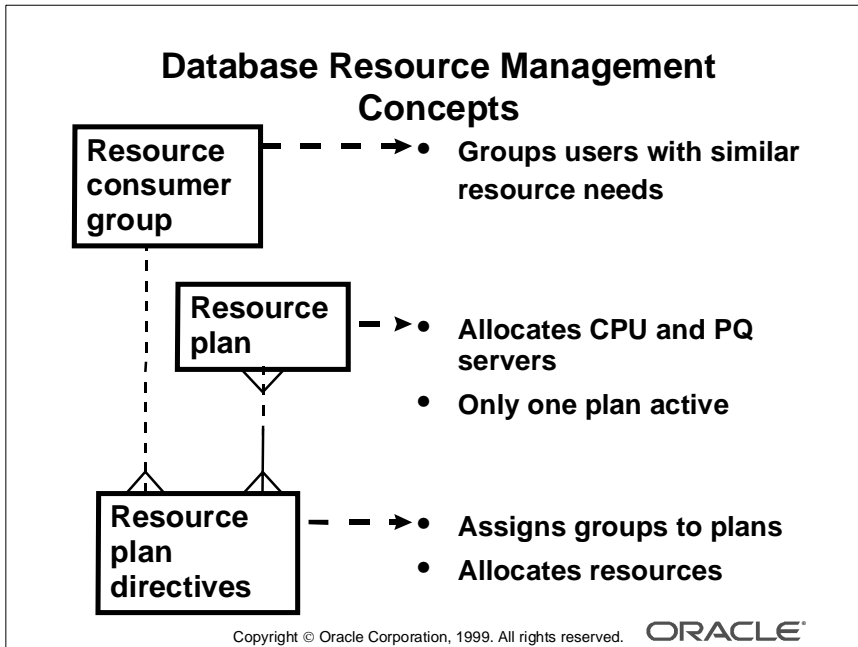
### Lesson Overview

The Database Resource Manager enables the database administrator to have more control over resource management than would typically be possible through operating system resource management alone. Using this facility, the database administrator can:

- Guarantee certain users or groups of users a minimum amount of processing resources regardless of the load on the system and the number of users
- Distribute available processing resources by allocating percentages of CPU time to different users (In an OLTP environment, a higher priority may be given to OLTP applications than to DSS applications during normal business hours)
- Limit the number of parallel query (PQ) servers available to a set of users
- Configure an instance to use a particular method of allocating resources (For example, a DBA can dynamically change the method from a daytime setup to a nighttime setup without having to shut down and restart the instance.)

This is a new feature introduced in Oracle8i.

## Resource Management Concepts



### Database Resource Manager Concepts

To use the Database Resource Manager, a database administrator defines:

- Resource plans
- Resource consumer groups
- Resource plan directives

Resource consumer groups, plans, and directives provide a method for specifying how to partition processing resources among different users. Resource plans currently support control of two resources: CPU and degree of parallelism.

### Resource Consumer Group

A resource consumer group defines a set of users who have similar resource usage requirements.

## **Resource Plan**

Resource plans contain resource plan directives, which specify the resources that are to be allocated to each resource consumer group.

You can have multiple resource plans defined in the database, each allocating resources to resource consumer groups in different ways, making resource assignment flexible. However, only one plan can be active in one instance. You can dynamically switch the top-level active plan while an instance is running.

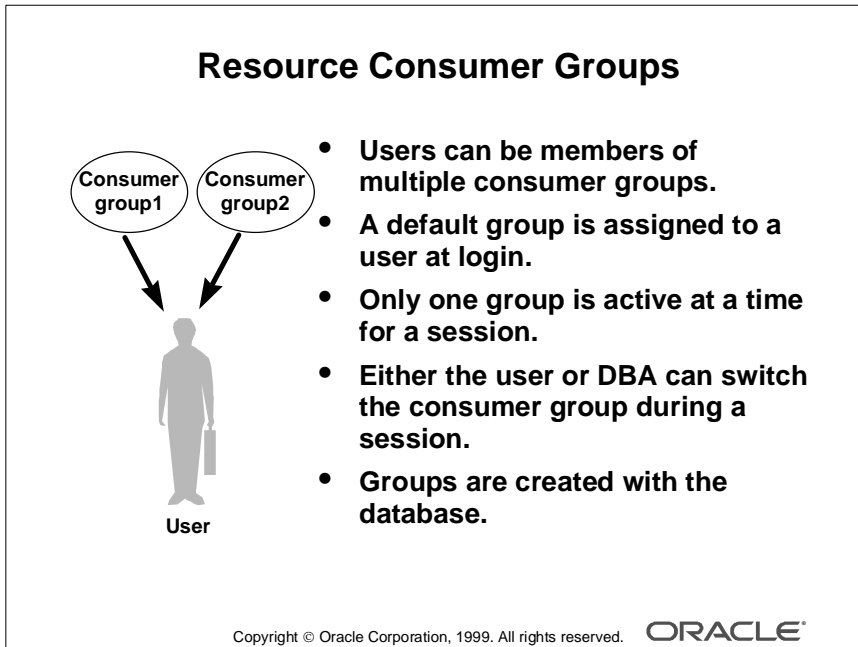
You can specify resource plans in a hierarchical fashion using subplans. Activating a plan also activates all of its subplans.

## **Resource Plan Directives**

Resource plan directives are used to:

- Assign consumer groups or subplans to resource plans
- Allocate resources among consumer groups or subplans in the plan

## Resource Consumer Groups



### Resource Consumer Groups

The following consumer groups are defined when the database is created:

- **SYS\_GROUP:**
  - Given high priority in the plan SYSTEM\_PLAN
  - Assigned to the users SYS and SYSTEM
  - Enables the database administrators to correct problems without contending for resources with other users
- **LOW\_GROUP:** A low-priority group in the plan SYSTEM\_PLAN
- **DEFAULT\_CONSUMER\_GROUP:** Applies to all sessions that do not explicitly belong to any group (A user is assigned to this group by default.)
- **OTHER\_GROUPS:** Applies to all sessions that belong to a consumer group that is not part of the current resource plan (OTHER\_GROUPS must exist somewhere in the plan schema of any active plan.)

The resource plan named SYSTEM\_PLAN is also defined when the database is created.

## Resource Plan Directives

### Resource Plan Directives

- **Manage parallelism:**
  - **Method: Absolute**
  - **Allocate PQ servers for an operation**
  - **Limit degree of parallelism**
- **Manage CPU usage:**
  - **Method: Emphasis**
  - **Allocate based on percentages at different levels**
  - **Delay work that exceeds CPU limits**

Copyright © Oracle Corporation, 1999. All rights reserved.

ORACLE®

### Resource Allocation Methods

Oracle8i provides resource allocation methods for controlling each managed resource. Resource allocation methods determine what method or policy the Database Resource Manager uses when allocating for a particular resource, and are used by both resource consumer groups and resource plans. There are currently two methods defined in Oracle:

- **Absolute method:** Used to define the degree of parallelism
- **Emphasis method:** Used to allocate CPU resources

### Degree of Parallelism

The parallel degree limit resource directive uses the absolute method, because the default resource allocation method for the maximum degree of parallelism is an absolute number.

If there are multiple plan directives referring to the same subplan/consumer group, the parallel degree limit for that subplan/consumer group will be the minimum of all the incoming values.

## CPU Usage

The CPU resources are allocated using the emphasis resource allocation method. This method allocates resources to various consumer groups or subplans within a plan using:

- A maximum of eight levels, numbered 1 through 8
- Percentages specifying how to partition CPU at each level

The following rules apply for the emphasis resource allocation method:

- Sessions in resource consumer groups with non-zero percentages at lower levels always get the first opportunity to run. If 100% of the resources are used at a lower level, then there are no resources available at the higher levels.
- CPU resources are distributed at a given level based on the specified percentages. The percentage of resources specified for a resource consumer group is a maximum for how much that consumer group can use.
- If any CPU resources are left after all resource consumer groups at a given level have been given an opportunity to run, the remaining CPU resources fall through to the next higher level. If a consumer group does not consume its allotted resources, then the resources are passed to the next level, not given to the other consumer groups at the same level.
- The sum of percentages at any given level must be less than or equal to 100.
- Any unused CPU time is recycled; in other words, if no consumer groups are immediately interested in a quantum (due to percentages), the consumer groups get another opportunity to use the quantum, starting at level one.
- Any levels that have no plan directives explicitly specified are implied to have 0% resources for all subplans or consumer groups.

## Database Resource Management Example

Database Resource Management Example				
Plan	Level	Consumer Group	CPU	Parallelism Degree
DAY	1	SYS_GROUP	100%	20
	2	OLTP	100%	0
	3	DSS	100%	20
NIGHT	1	SYS_GROUP	100%	20
	2	OLTP	25%	0
	2	DSS	75%	20
	3	OLTP	100%	0

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE

### DAY Plan

The DAY plan is used during high OLTP usage. It allocates CPU resources using the following logic:

- Because it is the only group defined at level 1, the SYS\_GROUP uses the CPU resources it needs.
- If the SYS\_GROUP does not use all of the CPU resources, then the OLTP group can use the remaining CPU resources. The OLTP group can use these resources, because it is the only consumer group defined at level 2.
- If the OLTP group does not use the rest of the CPU resources, then there are no resources for the DSS group. Finally, any left-over CPU resources can be used by the DSS group.

### NIGHT Plan

The NIGHT plan allows DSS processing as well as OLTP, using the following logic:

- The SYS\_GROUP uses the CPU resources it needs.
- Remaining CPU resources are divided between the OLTP and DSS groups:
  - The OLTP group uses 25% of the remaining CPU resources.
  - The DSS group uses 75% of the remaining CPU resources.
- Any left-over CPU resources can be used by the OLTP group.

**Note:** Because all users will be in one of these three groups, the OTHER\_GROUPS group is not included in the description of these plans; however, the OTHER\_GROUPS groups must be included in an active plan.

## Steps in Database Resource Management

### Steps in Database Resource Management

1. **Assign the resource manager system privileges to the administrator.**
2. **Create resource objects with the package DBMS\_RESOURCE\_MANAGER:**
  - Resource consumer groups
  - Resource plans
  - Resource plan directives
3. **Assign users to groups with the package DBMS\_RESOURCE\_MANAGER\_PRIVS.**
4. **Specify the plan to be used by the instance.**

Copyright © Oracle Corporation, 1999. All rights reserved.

ORACLE®

### Resource Manager Administration Privilege

DBAs have been granted the privileges to administer Database Resource Manager with the ADMIN OPTION. To permit other users to manage database resources, the DBA grants them system privileges on Database Resource Manager.

### Database Resource Manager PL/SQL Packages

Two packages are provided to administer and use Database Resource Manager:

- **DBMS\_RESOURCE\_MANAGER:** Used by a DBA to:
  - Maintain plans, consumer groups, and plan directives
  - Define the initial consumer group for a user
  - Dynamically alter the consumer group for a currently connected user
- **DBMS\_RESOURCE\_MANAGER\_PRIVS:** Used by the DBA to maintain privileges associated with resource consumer groups:
  - Grant and revoke resource management privileges to users
  - Grant users the capability to switch the resource consumer group for their current session, using the PL/SQL package DBMS\_SESSION

The steps to create a resource plan and implement it using these packages are illustrated in the following sections.



## Assigning the Resource Manager Privilege

### Assigning the Resource Manager Privilege

1. Assign the resource manager system privileges to the administrator.

```
DBMS_RESOURCE_MANAGER_PRIVS.  
GRANT_SYSTEM_PRIVILEGE (  
    grantee_name => 'SCOTT',  
    privilege_name  
        => 'ADMINISTER_RESOURCE_MANAGER',  
    admin_option => FALSE );
```

Copyright © Oracle Corporation, 1999. All rights reserved.

ORACLE

### Privilege Needed to Administer Resources

A new privilege, ADMINISTER\_RESOURCE\_MANAGER, is used to control who can use the DBMS\_RESOURCE\_MANAGER package to administer resource plans and consumer groups. The privilege is granted and revoked with the package DBMS\_RESOURCE\_MANAGER\_PRIVS.

### Granting Privileges Needed to Administer Resources

The procedure

DBMS\_RESOURCE\_MANAGER\_PRIVS.GRANT\_SYSTEM\_PRIVILEGE is used to grant the privilege to a user. For example, to permit the user SCOTT to manage database resources:

```
DBMS_RESOURCE_MANAGER_PRIVS.GRANT_SYSTEM_PRIVILEGE (  
    grantee_name => 'SCOTT',  
    privilege_name => 'ADMINISTER_RESOURCE_MANAGER',  
    admin_option => FALSE );
```

The privilege\_name parameter defaults to ADMINISTER\_RESOURCE\_MANAGER.

The third parameter specifies that SCOTT has been granted the privilege without the ADMIN OPTION.

## Revoking Privileges Needed to Administer Resources

To revoke this privilege, use the procedure `DBMS_RESOURCE_MANAGER_PRIVS.REVOKE_SYSTEM_PRIVILEGE`. For example, to revoke the privilege from the user `SCOTT`:

```
DBMS_RESOURCE_MANAGER_PRIVS.REVOKE_SYSTEM_PRIVILEGE (  
    revokee_name => 'SCOTT',  
    privilege_name => 'ADMINISTER_RESOURCE_MANAGER', );
```

The `privilege_name` parameter defaults to `ADMINISTER_RESOURCE_MANAGER`.

## Creating Database Resource Manager Objects

### Creating Database Resource Manager Objects

#### 2. Create resource objects with the package DBMS\_RESOURCE\_MANAGER.

##### 2.1. Create a pending area.

```
DBMS_RESOURCE_MANAGER.CREATE_PENDING_AREA( );
```

##### 2.2. Create resource consumer groups.

```
DBMS_RESOURCE_MANAGER.CREATE_CONSUMER_GROUP (
    consumer_group => 'OLTP',
    comment => 'Online users' );
```

Copyright © Oracle Corporation, 1999. All rights reserved.

ORACLE

### Creating a Pending Area

A pending area is used to store new Database Resource Manager plans, consumer groups and resource plan directives, and changes before they are committed. This area is used to facilitate validation before the changes are accepted. Only one pending area can be created in an instance at a given point in time. It is created using the procedure, CREATE\_PENDING\_AREA.

### Creating Resource Consumer Groups

When a consumer group is defined, it is stored in the pending area. A DBA can specify the name and a comment while defining a consumer group.

Procedures are available to modify or delete a consumer group.

The following consumer groups are defined when the database is created:

- **SYS\_GROUP:** A group given high priority in the plan SYSTEM\_PLAN, and assigned to the users SYS and SYSTEM
- **LOW\_GROUP:** A low-priority group in the plan SYSTEM\_PLAN
- **DEFAULT\_CONSUMER\_GROUP:** A group that is assigned to all users (PUBLIC) by default
- **OTHER\_GROUPS:** A special group for users that are not in a consumer group in the active plan

## Creating Database Resource Manager Objects

### 2.3 Create resource plans.

```
DBMS_RESOURCE_MANAGER.CREATE_PLAN (  
    plan =>      'NIGHT',  
    comment =>    'DSS/Batch priority, ...' );
```

### 2.4 Create resource plan directives.

```
DBMS_RESOURCE_MANAGER.CREATE_PLAN_DIRECTIVE (  
    plan =>      'NIGHT',  
    group_or_subplan =>    'SYS_GROUP',  
    comment =>    '...',  
    cpu_p1 =>      100,  
    parallel_degree_limit_p1 => 20);
```

Copyright © Oracle Corporation, 1999. All rights reserved. **ORACLE**

## Creating Resource Plans

When a resource plan is defined, it is stored in the pending area. A DBA can specify the name and a comment while defining a resource plan.

Procedures are also available to modify or delete a resource plan.

A plan `SYSTEM_PLAN` is created by default when the database is created.

## Creating Resource Plan Directives

A consumer group or a subplan is associated with a resource plan using the `CREATE_PLAN_DIRECTIVE` procedure.

Every plan must have a directive for the consumer group `OTHER_GROUPS`. The resources allocated to this special consumer group will be shared by all user sessions that do not have an explicitly assigned directive in the active plan.

A DBA uses the arguments `cpu_p1`, `cpu_p2`, ..., `cpu_p8` to partition CPU resources up to eight levels. CPU resources are used by the consumer groups in `cpu_p1` before `cpu_p2`, and so on.

## Creating Database Resource Manager Objects

### 2.5. Validate the pending area.

```
DBMS_RESOURCE_MANAGER.VALIDATE_PENDING_AREA( );
```

### 2.6. Commit the pending area.

```
DBMS_RESOURCE_MANAGER.SUBMIT_PENDING_AREA( );
```

Copyright © Oracle Corporation, 1999. All rights reserved. **ORACLE**

### 2.5. Validating the Pending Area

The procedure, `VALIDATE_PENDING_AREA`, can be used to validate the changes made. When this procedure is invoked, the Oracle server verifies whether rules such as the following are adhered to:

- All consumer groups referred to in the plans created or changed exist.
- Plans have directives that reference other plans or consumer groups.
- Plans do not reference themselves either directly or indirectly.
- The sum of CPU percentages defined for a specific level within a plan does not exceed 100.
- A plan that is currently used is not deleted.

If any of the rules are violated, the user receives an error. At this point the user may clear all changes using `CLEAR_PENDING_AREA` and reissue the commands, or use the appropriate procedure to correct the errors.

### 2.6. Committing the Pending Area

The changes made can be committed using the `SUBMIT_PENDING_AREA` procedure. Invoking this procedure implicitly validates the pending area before commit. If successful, this command clears the pending area. If there is a validation error, an exception is raised and the user can make corrections before invoking this procedure again.

## Assigning Users to Consumer Groups

### Assigning Users to Consumer Groups

#### 3. Assign users to groups.

```
DBMS_RESOURCE_MANAGER_PRIVS.  
GRANT_SWITCH_CONSUMER_GROUP (  
    grantee_name =>    'MOIRA',  
    consumer_group =>  'OLTP',  
    grant_option  =>    FALSE );
```

#### Set the initial consumer group for users

```
DBMS_RESOURCE_MANAGER.  
SET_INITIAL_CONSUMER_GROUP (  
    user =>            'MOIRA',  
    consumer_group =>  'OLTP' );
```

Copyright © Oracle Corporation, 1999. All rights reserved.

ORACLE®

### Assigning Users to Groups

The DBMS\_RESOURCE\_MANAGER\_PRIVS package includes procedures to grant and revoke consumer groups to and from users or roles. When a consumer group is granted to a user, the user can be given the privilege to grant the consumer group to other users.

### Setting the Initial Consumer Group for Users

Where a user has been granted several consumer groups, an initial consumer group can be assigned to the user. The consumer group specified while defining the initial consumer group must have been granted to the user directly or through PUBLIC, and not through a role. If a user has not been assigned an initial consumer group, it defaults to DEFAULT\_CONSUMER\_GROUP, which is a group that is assigned to PUBLIC.

## Setting the Resource Plan for an Instance

### Setting the Resource Plan for an Instance

#### 4. Specify the plan to be used by the instance.

- Specify the `RESOURCE_MANAGER_PLAN` initialization parameter.

```
RESOURCE_MANAGER_PLAN=day
```

- Change the resource plan without shutting down and restarting the instance.

```
ALTER SYSTEM  
SET RESOURCE_MANAGER_PLAN=night;
```

Copyright © Oracle Corporation, 1999. All rights reserved.

ORACLE®

### Using the `RESOURCE_MANAGER_PLAN` Initialization Parameter

The plan for an instance can be defined using the `RESOURCE_MANAGER_PLAN` parameter. If this parameter is not set, resource management is disabled for the instance. If the plan specified in the parameter file is not defined in the database, the database cannot be opened and the following error is returned:

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

If this error is encountered, the instance must be shut down and restarted after the parameter is modified to show a correct value.

If the resource plan is changed using the `ALTER SYSTEM` command, it takes effect immediately.

## Changing a Consumer Group Within a Session

### Changing a Consumer Group Within a Session

The user or the application can switch the current consumer group.

```
DBMS_SESSION.  
  SWITCH_CURRENT_CONSUMER_GROUP (  
    new_consumer_group => 'DSS',  
    old_consumer_group => v_old_group,  
    initial_group_on_error => FALSE );
```

Copyright © Oracle Corporation, 1999. All rights reserved.

ORACLE®

### Changing the Current Consumer Group

For example, an online application that wants to generate a report at the end of a user session could execute the command shown in the slide so that the report runs at a different priority than the rest of the application.

The old value is returned to the calling application. If necessary, the consumer group can be switched back to the user's initial group within the application.

The third argument, if TRUE, sets the current consumer group of the invoker to the initial consumer group in the event of an error.



## Changing Consumer Groups for Sessions

### Changing Consumer Groups for Sessions

- Can be set by DBA for a session

```
DBMS_RESOURCE_MANAGER.  
  SWITCH_CONSUMER_GROUP_FOR_SESS (  
    session_id => 7,  
    session_serial => 13,  
    consumer_group => 'OLTP');
```

- Can be set by DBA for all sessions for a user

```
DBMS_RESOURCE_MANAGER.  
  SWITCH_CONSUMER_GROUP_FOR_USER (  
    user => 'MOIRA',  
    consumer_group => 'OLTP');
```

Copyright © Oracle Corporation, 1999. All rights reserved.

ORACLE

### Changing Consumer Groups

A database administrator can switch the consumer group for a session or for a user. These changes take effect immediately.

To switch the consumer group for a session, use the SWITCH\_CONSUMER\_GROUP\_FOR\_SESS procedure and specify the session ID, serial number, and new consumer group for the session. The user must have been assigned the consumer group that is specified for this operation to succeed. To determine the session ID and serial number, query v\$session:

```
SQL> SELECT sid, serial#  
2 FROM v$session  
3 WHERE USERNAME = 'MOIRA';
```

```
      SID      SERIAL#  
-----  
          7          13
```

The SWITCH\_CONSUMER\_GROUP\_FOR\_USER provides a convenient method for the database administrator to switch all sessions for a given user that are to be switched into a new consumer group. The username and the new consumer group are the parameters passed to this procedure.

## Database Resource Manager Information

### Database Resource Manager Information

- **DBA\_RSRC\_PLANS**  
Resource plans and status
- **DBA\_RSRC\_PLAN\_DIRECTIVES**  
Resource plan directives and status
- **DBA\_RSRC\_CONSUMER\_GROUPS**  
Consumer groups and status
- **DBA\_RSRC\_CONSUMER\_GROUP\_PRIVS**  
Users granted consumer groups
- **DBA\_USERS**  
Column:  
**INITIAL\_RSRC\_CONSUMER\_GROUP**

Copyright © Oracle Corporation, 1999. All rights reserved.

ORACLE®

### Data Dictionary Views

Several new data dictionary views are available to check the resource plans, consumer groups, and plan directives declared in the instance. This section discusses some useful information that can be obtained from these views.

### Resource Plans

Use the following query to get information on resource plans defined in the database:

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

PLAN	NUM_PLAN_DIRECTIVES	STATUS	MAN
NIGHT_PLAN	3	ACTIVE	NO
SYSTEM_PLAN	3	ACTIVE	NO

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

## Resource Plan Directives

Use the following query to retrieve information on resource plan directives defined in the database:

```
SQL> SELECT plan, group_or_subplan, cpu_p1, cpu_p2, cpu_p3,
2    parallel_degree_limit_p1, status
3    FROM dba_rsrc_plan_directives;
```

PLAN	GROUP_OR_SUBPLA	CPU_P1	CPU_P2	CPU_P3	PARALL	STATUS
NIGHT_PLAN	BATCH	70	0	0	20	ACTIVE
NIGHT_PLAN	OLTP	25	0	0	2	ACTIVE
NIGHT_PLAN	OTHER_GROUPS	5	0	0	2	ACTIVE
SYSTEM_PLAN	SYS_GROUP	100	0	0	0	ACTIVE
SYSTEM_PLAN	OTHER_GROUPS	0	100	0	0	ACTIVE
SYSTEM_PLAN	LOW_GROUP	0	0	100	0	ACTIVE

Note that the SYSTEM\_PLAN specifies SYS\_GROUP at level-1, OTHER\_GROUPS at level-2 and LOW\_GROUP at level-3 for CPU usage. This setting will simulate a priority system.

## Resource Consumer Groups and Privileges

Resource consumer groups and associated privileges can be obtained using the following queries:

```
SQL> SELECT *
2    FROM dba_rsrc_consumer_group_privs;
```

GRANTEE	GRANTED_GROUP	GRA	INI
BRUCE	BATCH	NO	NO
DAVID	BATCH	NO	YES
DAVID	OLTP	YES	NO
PUBLIC	DEFAULT_CONSUMER_GROUP	YES	YES
PUBLIC	LOW_GROUP	NO	NO
SYSTEM	SYS_GROUP	NO	YES

## Resource Consumer Groups and Privileges (continued)

```
SQL> SELECT consumer_group, status, mandatory
       2     FROM dba_rsrc_consumer_groups;
```

CONSUMER_GROUP	STATUS	MAN
-----	-----	---
BATCH	ACTIVE	NO
OLTP	ACTIVE	NO
OTHER_GROUPS	ACTIVE	YES
DEFAULT_CONSUMER_GROUP	ACTIVE	YES
SYS_GROUP	ACTIVE	NO
LOW_GROUP	ACTIVE	NO

## Current Database Resource Manager Settings

### Current Database Resource Manager Settings

- **V\$SESSION:** Contains the **RESOURCE\_CONSUMER\_GROUP** column that shows the current group for a session
- **V\$RSRC\_PLAN:** A view that show the active resource plan
- **V\$RSRC\_CONSUMER\_GROUP:** A view that contains statistics by consumer group

Copyright © Oracle Corporation, 1999. All rights reserved.

ORACLE®

### Dynamic Performance Views

These dynamic performance views contain information on the current setting for the Database Resource Manager.

## Summary

### Summary

**In this lesson, you should have learned how to control the use of CPUs and the degree of parallelism using Database Resource Manager.**

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

## Quick Reference

Context	Reference
Parameters	RESOURCE_MANAGER_PLAN
Dynamic performance views	V\$RSRC_PLAN V\$RSRC_CONSUMER_GROUP V\$SESSION
Data dictionary views	DBA_RSRC_PLANS DBA_RSRC_PLAN_DIRECTIVES DBA_RSRC_CONSUMER_GROUPS DBA_RSRC_CONSUMER_GROUP_PRIVS DBA_USERS
Commands	ALTER SYSTEM SET RESOURCE_MANAGER_PLAN
Packages and Procedures	CLEAR_PENDING_AREA CREATE_PENDING_AREA CREATE_PLAN CREATE_PLAN_DIRECTIVE DBMS_RESOURCE_MANAGER DBMS_RESOURCE_MANAGER_PRIVS DBMS_SESSION GRANT_SWITCH_CONSUMER_GROUP GRANT_SYSTEM_PRIVILEGE REVOKE_SYSTEM_PRIVILEGE SET_INITIAL_CONSUMER_GROUP SUBMIT_PENDING_AREA SWITCH_CONSUMER_GROUP_FOR_SESS SWITCH_CONSUMER_GROUP_FOR_USER SWITCH_CURRENT_CONSUMER_GROUP VALIDATE_PENDING_AREA





## **Tuning with Oracle Expert**

## Objectives

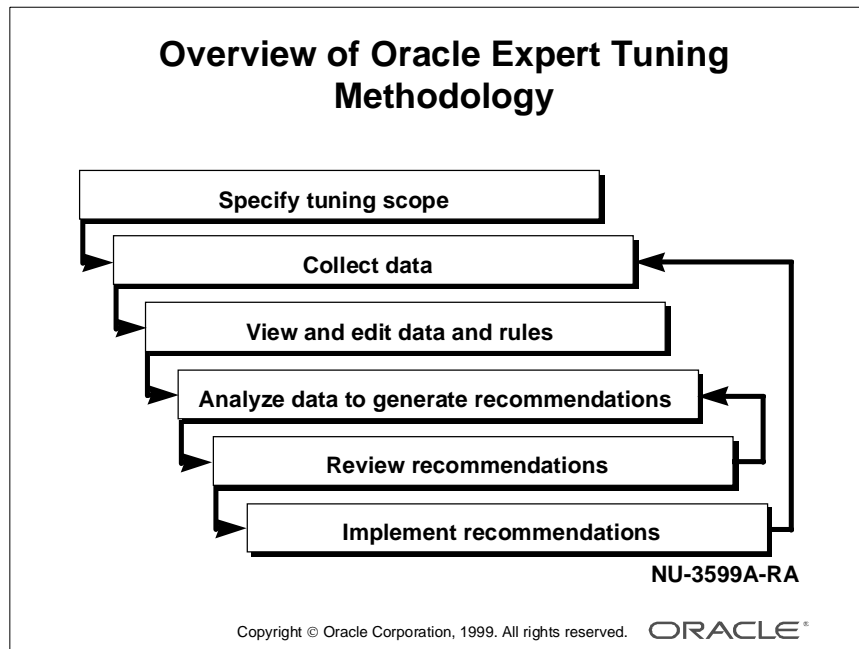
### Objectives

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

- **Describe the features of Oracle Expert**
- **Create a tuning session**
- **Gather, view, and edit the input data**
- **Analyze the collected data, using rules**
- **Review tuning recommendations**
- **Implement tuning recommendations**

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

## Overview



### Characteristics

Oracle Expert provides automated performance tuning with integrated rules. The Oracle Expert tuning methodology involves the following steps:

- 1** Setting the scope of the tuning session: Set the scope of the tuning session to tell Oracle Expert what aspects of your database you want to tune.
- 2** Collecting the data: To get comprehensive performance recommendations, Oracle Expert collects the following classes of data: database, instance, schema, environment, and workload.
- 3** Viewing and editing the collected data and rules: Once you have collected the various pieces of tuning data, you can view and edit that data. The data is organized in a hierarchical format. You can also view and edit the rules and attributes that Oracle Expert uses to make its recommendations.
- 4** Analyzing the data to generate recommendations: When you have collected and edited the data as needed, Oracle Expert performs an analysis to generate tuning recommendations.

### **Characteristics (continued)**

- 5** Reviewing the Oracle Expert recommendations: After Oracle Expert has analyzed the data, you can review the recommendations and decide which to accept. If you do not want to accept all the recommendations, have Oracle Expert generate a new recommendation. The new analysis takes into account the interdependencies of your preferences.
- 6** Generating scripts for implementing the recommendations: When you are satisfied with all of your inputs and the resulting recommendations, Oracle Expert can generate parameter files and implementation scripts to implement at your convenience.

## Types of Tuning

### Types of Tuning

- **Routine tuning**
- **Focused tuning**
- **What-If tuning**

Copyright © Oracle Corporation, 1999. All rights reserved. **ORACLE®**

### Routine Tuning

Routine tuning of your database will help to identify and solve potential problems before they occur. Tuning is especially recommended after changes have been made in the hardware capacity, transaction volume, and the users or applications that access the database.

### Focused Tuning

Focused tuning deals with the resolution of a known performance problem. This resolution occurs by choosing the appropriate tuning categories and focusing on the particular problem.

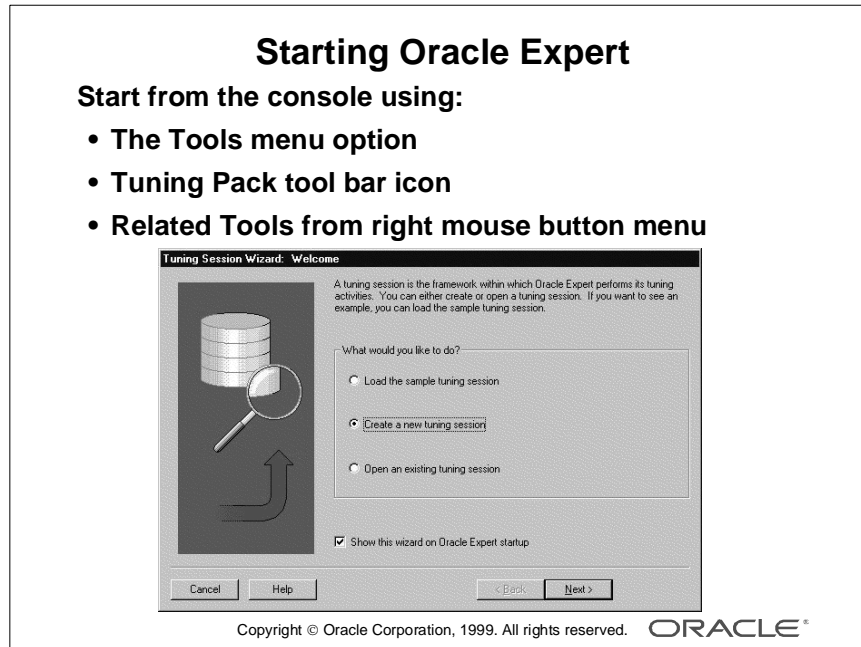
Use the Performance Manager to monitor your database. Use its advanced drilldown features to help identify areas of your database operations that require tuning.

## **What-If Tuning**

Once you have gathered all the data required for a tuning session, you can edit most of the parameters and test different configurations. For example, you might want to know how your system would perform with extra memory, or what might happen if the number of users doubled. The Oracle Expert Analysis report would explain any changes that your database would require.

**Note:** If you can make good predictions about what your new database will be like, Oracle Expert can provide an initial configuration that is more appropriate than the Oracle server default instance parameters would be.

## Starting Oracle Expert



### Starting Oracle Expert

To start identifying high impact SQL statements, the Oracle SQL Analyze tool can be started by one of the following methods:

- Select Tools—>Tuning Pack—>Oracle Expert from the console menu.
- Click the Tuning Pack tool bar icon from the console window, and then select Oracle Expert.
- Click the right mouse button on the database that requires monitoring, then select Related Tools—>Oracle Expert.
- On NT, select Programs—>Oracle Enterprise Manager—>Tuning Pack—>Oracle Expert from the Start menu.

### Connecting to a Tuning Repository

After starting Oracle Expert, the Login dialog box may prompt for a connection to an Oracle Expert repository (an Oracle Expert repository is created if one does not exist):

- Connect to an OMS when sharing Oracle Expert information among administrators or when access to all databases in the repository is required. Supply a username, password, and machine name containing an OMS for the repository.

### Connecting to a Tuning Repository (continued)

- Connect as a database user (with DBA privileges) where an Oracle Expert repository is created. If an Oracle Expert repository does not already exist, one is created for storing and retrieving Oracle Expert session details.

**Note:** Oracle Expert requires the SELECT ANY TABLE privilege for the database in which the repository is stored. When you connect for the first time, there will be a pause as Oracle Expert loads its default rules base into the repository.

### Working from a Tuning Session

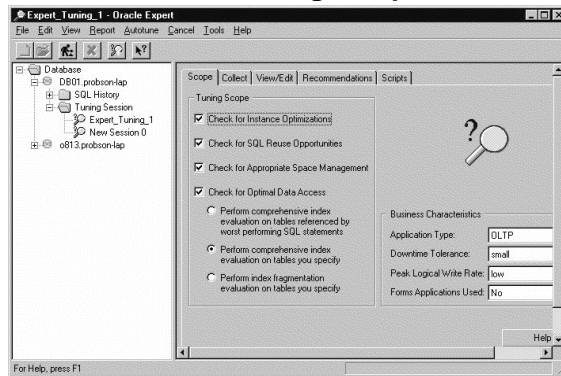
After opening Oracle Expert, the first task in the tuning process is starting a tuning session. This is performed by selecting one of the following:

- From the opening screen (or Tools—>Tuning Session Wizard), use the Oracle Expert wizard to open the Sample tuning session, a new tuning session, or an existing tuning session (see slide).
- From the Navigator window, expand the appropriate database and tuning folders to select an existing tuning session.
- From the Oracle Expert menu select File—>New (or the New icon in the toolbar) to create a new tuning session. If no database is selected in the Navigator window, supply login information to create the new database tuning session.



## Tuning Session Scope

- Instance parameter tuning
- Application tuning
- Database structure sizing and placement



Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

### Setting the Tuning Session Scope

Setting the scope for the tuning session directs Oracle Expert into specific problem areas. This reduces both the amount of information to be collected and the time required to complete the analysis.

**Instance Optimizations** This gathers information for tuning database instances, such as SGA size, I/O distribution, and sort operation performance:

- **SGA (system global area) parameters:** These instance parameters affect the total size of the instance's System Global Area (SGA). The appropriate setting of these parameters results in the efficient use of memory and prevents reparsing SQL statements except when necessary. Examples of these parameters include the `DB_BLOCK_BUFFERS` and `SHARED_POOL_RESERVED_SIZE` parameters.
- **I/O parameters:** These instance parameters affect the throughput or distribution of I/O for the instance. Examples of these parameters include the `DB_FILE_MULTIBLOCK_READ_COUNT` parameter.
- **Sort parameters:** These parameters influence how the Oracle server performs sort operations on behalf of the user. Examples of these parameters include the `SORT_AREA_RETAINED_SIZE` parameter.

### **Setting the Tuning Session Scope (continued)**

- **Parallel query parameters:** These instance parameters are specific to the parallel query behavior for the instance. Examples of these parameters include the `PARALLEL_MIN_SERVERS` and `PARALLEL_MAX_SERVERS` parameters.
- **Parallel Server parameters:** These parameters influence the performance or configuration of the parallel server environment. Selecting the Oracle Parallel Server parameters category for tuning is relevant only if the Oracle Parallel Server option is installed.
- **OS-Specific:** These instance parameters affect performance and are specific to certain hardware platforms. Examples of these parameters include the `DISK_ASYNCH_IO` and the `DB_WRITER_PROCESSES` parameters.

**SQL Reuse Opportunities** This analyzes the shared pool to identify SQL statements inefficiently using the shared pool.

The Oracle server maintains only one copy of a distinct SQL statement within the library cache to maximize memory and minimize redundant parsing and validating. To effectively use this feature, you must write identical SQL statements that use the same structure and form. Oracle Expert compares your workload statements to determine if any can be rewritten to take advantage of the cache behavior, and reports its findings.

**Optimal Data Access** This determines which indexes are required or redundant for tables. You can choose to:

- Evaluate table and index access by the most inefficient SQL statements (during analysis, each statement is ranked by physical reads per execution)
- Which tables to monitor (also located inefficient indexes)
- Determine which indexes are inefficient and should be rebuilt

**Appropriate Space Management** This analyzes tablespaces, users, and other database structures to determine efficient storage methods, such as placement of database files or storage parameters for segments.

**Business Characteristics Section** This section helps Oracle Expert make more accurate tuning recommendations about the database environment:

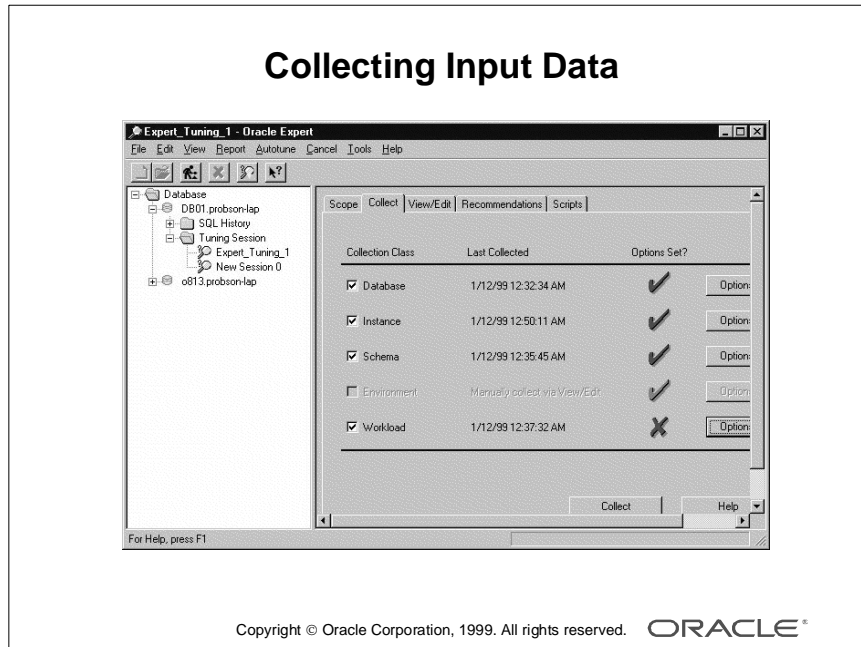
- **Application Type:** Notifies Oracle Expert that database is primarily used for fast concurrent database access (online transaction processing—OLTP), long-running, resource-intensive jobs (data warehousing, also known as decision support), or a multipurpose (hybrid) mixture between the two.

### **Setting the Tuning Session Scope (continued)**

- Downtime Tolerance: Indicates whether the database is strictly a 24 x 7 database (field value of None) or an insignificant database (field value of Large).
- Peak Logical Write Rate: Indicates that there will be fewer than 5 write transactions per second (field value of Low) or more than 500 write transactions per second (field value of Huge)
- Forms Applications Used: Informs Oracle Expert whether forms applications, such as Oracle Forms or Reports, are being used

**Note:** You can use Oracle Expert for comprehensive tuning of all aspects of the database at once. However, for a large and complex database, this could take many hours to process.

## Data Collection



### Collecting Data

For each tuning session, data must be collected and stored in the Oracle Expert repository. Then Oracle Expert can analyze the data and generate tuning recommendations.

When you select the Collect page, one or more of the collection classes is enabled and selected based on the tuning categories you selected. After you select or clear the classes you want to collect for this tuning session, and before you start collecting by clicking the Collect button, you can edit the options for each collection class.

All of the classes except the Environment class can be collected automatically by Oracle Expert.

You can reduce the time you spend collecting tuning session data by collecting the minimum amount of data Oracle Expert requires. Reducing the amount of data also shortens the analysis and recommendation review cycle.

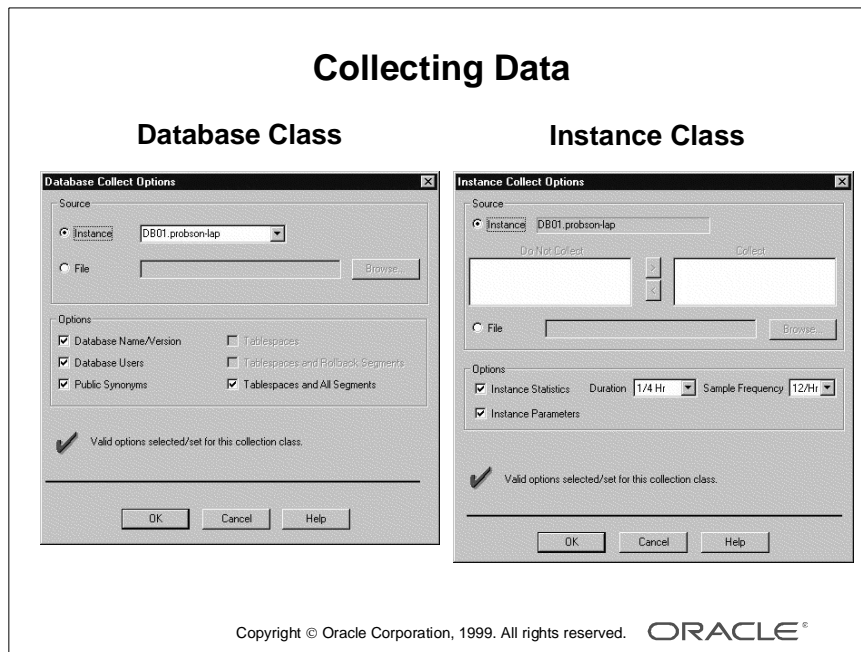
### Tuning Scope and Collections

The tuning categories that you select for a tuning session determine the classes you should collect. If you are tuning multiple categories, any common classes need only be collected once.

### Tuning Scope and Collections (continued)

The following table shows the required collection class options for each tuning category:

<b>Collection Class Options</b>	<b>Instance Optimization</b>	<b>SQL Reuse Opportunities</b>	<b>Space Management</b>	<b>Optimal Data Access</b>
<b>Database Class:</b>				
Name/Version	Y	Y	Y	Y
Database Users	N	N	N	Y
Public Synonyms	N	N	Y	Y
Tablespaces	N	N	Y	N
Tablespaces and Rollback Segments	Y	N	N	N
Tablespaces and All Segments	N	N	N	Y
<b>Instance Class:</b>				
Instance Statistics	Y	N/A	N	Y
Instance Parameters	Y	N/A	Y	Y
<b>Schema Class:</b>				
Schema	N/A	N/A	Y	Y
Statistics	N/A	N/A	Y	Y
<b>Environment Class:</b>	Y	N/A	N/A	N/A
<b>Workload Class:</b>	N/A	Y		Y



### Database Class Data

- Oracle Expert uses the Database class to recommend using Oracle server features available for the database version and distributing data among tablespaces.
- Oracle Expert is generally more accurate with its tuning recommendations once it has compiled a significant amount of historical data.
- Oracle Expert primarily collects Database class statistics from the data dictionary views (DBA views) and dynamic performance views (V\$ views).

### Instance Class Data

The Source section has two fields for selecting the data collection source:

- Instance: The Instance list box displays only one instance. To select multiple instances, move instances from the Do Not Collect box into the Collect box.
- File: Enter the name or locate a file that has previously exported class data from Oracle Expert (\*.XDL file).

The Options section determines what statistics to collect, such as instance statistics or instance parameters. When Instance is selected as the source:

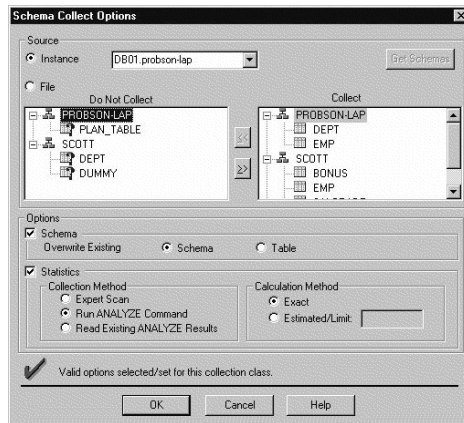
- The Duration list box specifies the statistics collection time (default is 1/4 hour).
- The Sample Frequency list box specifies the number of statistics samples to collect per hour (default is 12 per hour, or 1 every 5 minutes).

**Instance Class Data (continued)**

**Note:** Oracle Expert maintains a history of all the instance statistics samples for a database. If your samples are collected during different peak periods, Oracle Expert gains insight into the performance of the instance in a variety of situations and can generate better tuning recommendations over time.

For a more comprehensive tuning analysis, collect multiple instance statistic samples for a tuning session (a minimum of ten samples is suggested).

## Collecting Schema Class Data



Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

### The Schema Class Data

Schema information is used by Oracle Expert to determine optimal access for schema objects, such as tables and indexes. To specify the options to collect:

- 1** Populate the Do Not Collect box by clicking the Get Schemas button to retrieve from the database all schemas (except SYS and SYSTEM) that contain tables.
- 2** Select Instance as the source (see point 3) and use these buttons move tables and schemas from the Do Not Collect box to the Collect box (and vice versa). To specify what Schema class data to collect, follow these steps:
  - For the entire schema, double click the schema or select the schema from the Do Not Collect box and click the right-arrow (>) button to move it.
  - For an individual table in a schema, click the plus sign next to the required schema, then double click the table or select the table and click the right-arrow (>) button.
  - To remove schemas or tables from collection, select a table or schema in the Collect box, and click the left-arrow (<) button.
- 3** Select one of the two data collection sources from the Source section:
  - Instance: Select an instance from the Instance list box, then click the Get Schemas button (see point 1) and specify which schemas or tables to collect (see point 2).
  - File: Enter the name or locate a file that has previously exported class data from Oracle Expert (\*.XDL file).

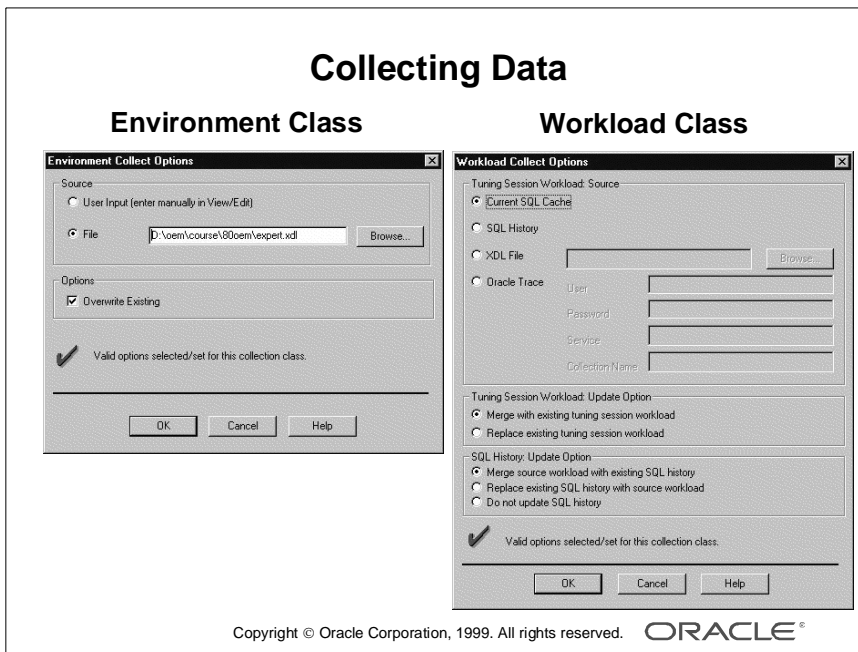


**The Schema Class Options (continued)**

- 4 Display all schemas and tables collected from the database in the Do Not Collect box by clicking the Get Schemas button (see point 1).
- 5 Select the Options check box if previously collected data should be overwritten. There are two methods available:
  - Schema: If a selected schema has previously been collected, Oracle Expert deletes all schema data before collecting the new data.
  - Table: If a selected table has previously been collected, Oracle Expert deletes the all table data before collecting the new data.
- 6 Select the Statistics check box to include statistics as part of the collection. There are three methods available:
  - Expert Scan: Oracle Expert scans the tables during collection and stores cardinality values in the repository. Estimated or exact values can be specified by selecting either the Estimated/Limit or Exact check boxes.
  - Run ANALYZE Command: The ANALYZE command with the STATISTICS option collects statistics and stores column, index, and cluster statistics in the data dictionary. Estimated or exact values can be specified by selecting either the Estimated/Limit or Exact check boxes.
  - Read Existing ANALYZE Results. Reads results of a previous SQL ANALYZE.
- 7 Click OK to accept the options after the green check mark is displayed.

**Note:**

- Oracle Expert uses the data in the Schema class to make recommendations about:
  - Storage optimization of tables, indexes, and clusters
  - Index structuring
  - Database file I/O distribution
  - Table, index, and cluster placement
- Schema class data should be used in conjunction with workload data. Import workload data that references tables selected in the Schema class options, since Oracle Expert removes requests that refer to tables that have not been collected.
- To reduce collection time, specify only schemas and tables required for tuning.
- Do not use the Expert Scan option for very large tables.



## Environment Class Data

Oracle Expert expects to be provided with system data (including memory, CPU, and operating system page size data):

- Total memory in megabytes
- Average memory utilization (percent of total used by databases)
- Maximum memory utilization
- Average CPU utilization (percent of total used by database applications)
- Number of CPUs
- Operating system page size in bytes

Oracle Expert cannot collect environment data automatically. You must provide Environment class data to Oracle Expert in one of the following two ways:

- Entering it manually in the View/Edit page
- Importing it from an .XDL file

## Workload Class Data

Workload class data describes to Oracle Expert how your database is used in daily operations. It describes the nature, frequency, and relative importance of applications, business units, transactions, and requests that access a database.

Gather representative workload data by thinking about your database and what you want to accomplish by tuning it. Do you want to minimize bottlenecks that occur regularly? Is there a certain combination of applications that adversely affects performance? Collect your workload data while these events are occurring.

## Four Ways to Input Workload Data

- **Collect SQL cache data:** The SQL cache of a database contains the statements that are currently the most frequently executed against the database. Oracle Expert can read these statements directly from the cache. Note that these statements are likely to be different at different times, depending on which applications are running.
- **SQL History:** Oracle Expert uses workload data stored in the database being tuned (The SQL History section is disabled when this option is selected.)
- **Import XDL data from a previous tuning session:** Because good workload data so accurately describes the current performance of your database, you might want to reuse this data for additional tuning sessions. With a good workload, you could generate different “what-if” scenarios by changing other inputs.
- **Import Oracle Trace data:** Oracle Trace is a product in the Diagnostics Pack. Oracle Trace is designed to collect workload data directly from an Oracle database. You can collect a representative workload during a problem to find bottlenecks, or collect on an average day for routine tuning.

Oracle Trace provides the most complete workload input, which also means that it might take the longest to import and analyze. See the lesson on Oracle Trace in the Oracle Enterprise Manager course or refer to the Oracle Trace documentation for information on collecting workload data.

The Tuning Session Workload Update Option section (only available when specific tables or indexes are selected from the tuning scope) determines whether the workload should be:

- Merged with existing workload data
- Replaced by the new workload data

When the Worst performing SQL statements option is selected from the tuning scope, the above section is replaced by the Tables Referenced by Worst Performing SQL section. Set a number of SQL statements to consider (default is 25) or select All source statements for a complete evaluation.

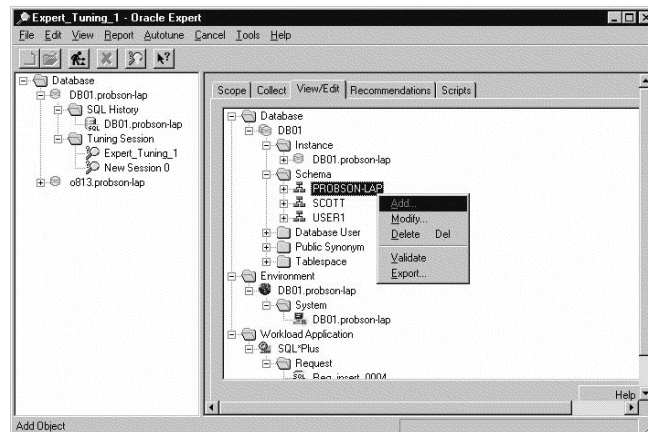
## **Workload Class Data (continued)**

### **Note**

- The Table Statistics Collection Method section is also displayed when the Worst Performing SQL statements option is selected from the tuning scope. It displays the Expert Scan, Run ANALYZE Command, and Read Existing ANALYZE Results options (see “The Schema Class Options” earlier in this lesson for more information).
- Workload data should be collected to assist tuning goals, such as collecting data during peak periods to identify possible reasons for bottlenecks, or while database performance is poor to identify possible inefficient applications.

## Collected Data

### Viewing and Editing the Collected Data



Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

### Viewing and Editing Collected Data

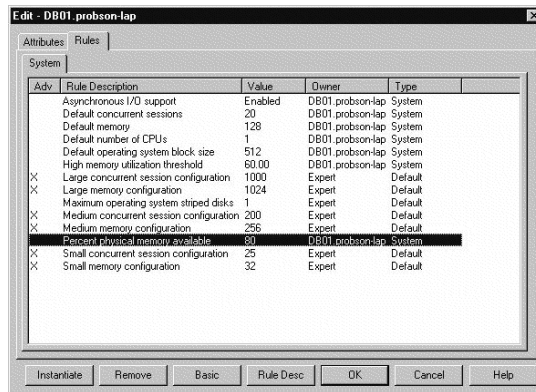
Oracle Expert gives you the ability to view, edit, add to, and delete from the data you collected.

The hierarchical data on the View/Edit page consists of the following:

- The Workload Application folder contains all applications that were running during the collection or obtained from the SQL History. Each application contains statistics that can be viewed and modified (for manually changing workload data).
- The Environment folder contains the system data (such as memory, CPU, and operating system page size) which is used by Oracle Expert to perform instance tuning. Each object under the System folder should be manually edited to provide accurate information for the system.
- The Database folder contains all the statistics collected for the instance, schemas, database users, public synonyms, and tablespaces. Each folder can be expanded to view the statistics and change rules for specific items collected. Modifying the data affects the results of the analysis. You might want to do this to run different scenarios and see what changes are most desirable for your circumstances.

## Attributes

### Editing Basic Rules Before Analysis



Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

### Viewing and Editing Attributes

When you add or modify data, Oracle Expert displays a property sheet for that object that lists the existing attributes and rule values.

Attributes are the characteristics of a data object. Changing their values enables you to test different database and tuning scenarios.

The buttons at the bottom of the window are:

- **Instantiate:** Copies a default rule and changes the owner (displayed in the owner column: see point 7) so that values can be changed and used during analysis. The letter R is also placed on the object icon in the View/Edit page. This button is only available in the Rule page. These values should only be changed by experienced DBAs.
- **Remove:** Removes the instantiation for the object (default rules cannot be removed). This button is only available in the Rule page.
- **Basic (or Advanced):** Displays or hides the advanced rules that are used by Oracle Expert in calculating algorithms during analysis. This button is only available in the Rule page.
- **Rule Desc:** Displays a description of the selected rule.

**Viewing and Editing Attributes (continued)**

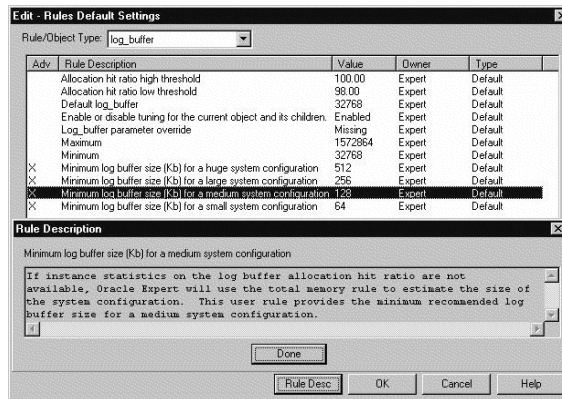
- OK: Applies any changes made to attributes or rules and closes the window.
- Cancel: Cancels any changes made and closes the window.
- Help: Provides context-sensitive help.

To modify the value and attributes for a rule:

- 1 Select it from the Rule page and click Instantiate. Then click in the Value field and enter the required value.
- 2 The X indicates an advanced rule (changing this value is not recommended)
- 3 Select the Attributes page to view or change the values for various attributes, such as the amount of memory, number of CPUs, and memory utilization values.
- 4 Select the Rules page to view or change rules associated with the object.
- 5 By selecting the Value field for a specific rule or attribute, a new value can be entered. Only some attributes and instantiated rules can be modified.
- 6 The Owner and Type fields change from Expert and Default when an object is instantiated.

## Rules

### Analyzing Using Default Rules



Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

## Rules

Oracle Expert uses rules or pieces of knowledge to analyze collected data for a tuning session.

Click the Rule Desc button to display an explanation of the selected rule.

## Default Rules

The default rules make up the extensive Oracle Expert knowledge base. They have default values.

Oracle Expert enables you to view some default rules. You can change the default value of any default rule that Oracle Expert allows you to view.

## Modified Rules

Oracle Expert allows you to view some of its rules, and you can modify the value of any rule. Modifying rule values may change the behavior of Oracle Expert during the analysis, and also the recommendations. You can also instantiate a default rule at an object level.



### Modified Rules (continued)

The modified rules may:

- Override an Oracle Expert recommendation
- Place a limit on the value of a recommendation
- Adjust the factors involved in calculating the recommendation

**Note:** Viewing and editing the collected data and the default Oracle Expert rules are advanced topics that only an experienced DBA will be able to do correctly. On the other hand, an experienced DBA probably would not use Oracle Expert unless it allowed some customization.

### Advanced Rules

By default, the Rules page displays only the basic rules for an object. Click the Advanced button to see the advanced rules for the object. Oracle Expert displays the advanced rules with the basic rules. When the advanced rules are displayed, an X appears in the Adv column of each advanced rule and the Advanced button is renamed the Basic button.

You will rarely want to view or edit these advanced rules. An example is low-level constants that are factored into the algorithms used by the Oracle Expert rules.

## Analysis

### Analyzing the Collected Data

To generate tuning suggestions:

- **Select the Recommendations page**
- **After the Generate button is clicked:**
  - **Stored rules are applied**
  - **Recommendations are created**
  - **Information is stored in repository**
- **Expand the required recommendations**
- **Analyze again if any recommendations are declined**

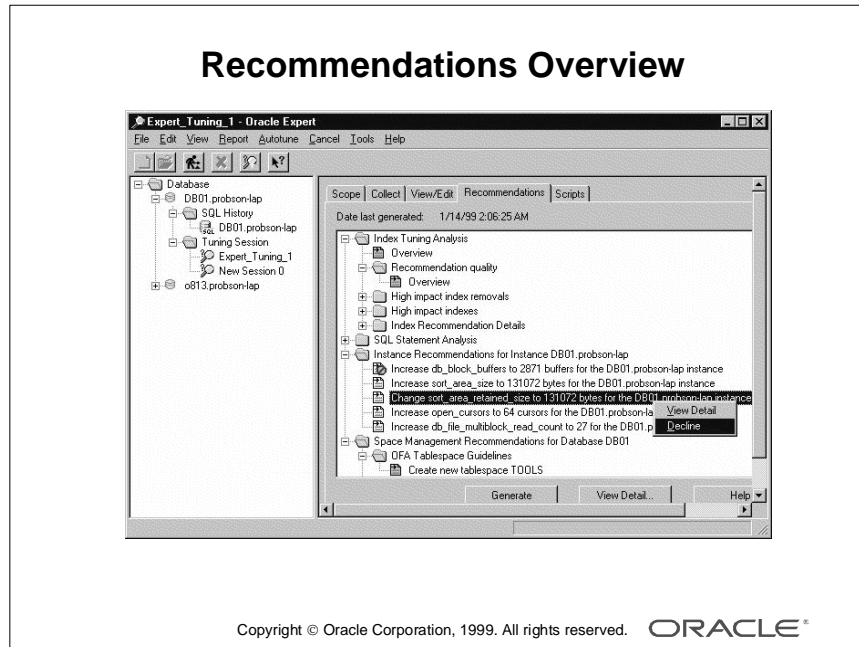
Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

### Analyzing the Collected Data

Oracle Expert can automatically generate tuning suggestions from the collected data, known as recommendations, by performing the following functions:

- Select the Recommendations page of the Tuning Session window
- Click the Generate button. The analysis could take minutes or hours depending on the amount of data collected and machine performance, because Oracle Expert:
  - Applies stored rules to collected data
  - Generates the tuning recommendations
  - Stores the information in the repository (until another analysis is performed for the tuning session)
- Expand the required recommendations in the Recommendations page when the analysis has completed (the recommendations appear in the order they were generated by Oracle Expert).
- Analyze again if any recommendations are declined (since some recommendations are dependent on others).

## Recommendations



### Overview

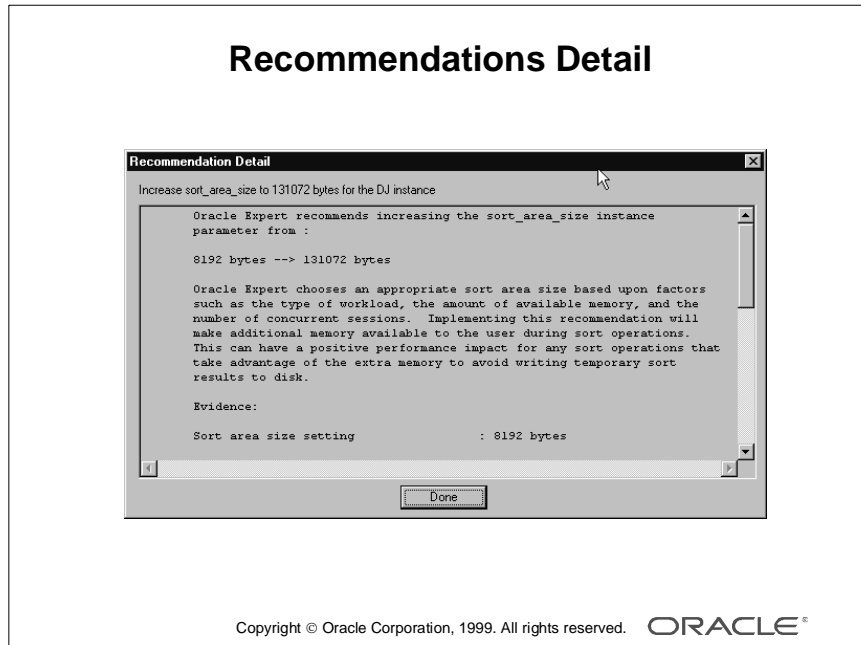
The recommendations appear in the order in which they were generated by Oracle Expert.

Many of the recommendations are based on interdependencies with the other recommendations and if you decline some of them, you may not get the expected results. If you reanalyze the data, Oracle Expert will take your preferences into account and will work around the recommendations you did not accept. Continue analyzing and reviewing the data until you are satisfied with all of the recommendations.

### Four Information Categories

- Index Tuning Analysis
- SQL Statement Analysis
- Instance
- Space Management

## Reports



### Recommendations Detail

For information about why Oracle Expert made a particular recommendation, select the recommendation and either click the View Detail button or double-click the recommendation. Details of the rationale for the recommendation are then displayed.

You should always review the recommendations generated by Oracle Expert before implementing any of them. You can generate two types of reports, the Session Data report and the Analysis report.

## Reports

- **The Analysis report lists and explains the Oracle Expert recommendations**
- **The Session Data report displays the collected data and the generated statistics**
- **The Recommendation summary provides a recommendation overview**
- **The Workload Cross Reference report displays tables with workload requests**

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

### Session Data Report

This report provides detailed information about the collected data in the repository for a tuning session; use it to make sure that all the data you planned to collect has been made available to Oracle Expert.

You can select one or more categories to be included in the report:

- Database
- Instance
- Schema
- Tablespace
- Database users
- Public synonyms
- Environment
- Workload
- Rules

## Session Data Report (continued)

### Example

Collected on 14/04/99 17:17:32

Instance Statistic	Value
-----	-----
Max Concurrent Users	6 Sessions
Buffer Cache Hit Ratio	53.39%
In Memory Sort Ratio	97.22%

Collected on 14/04/98 17:25:24

Instance Statistic	Value
-----	-----
Concurrent Users	5 Sessions
Buffer Cache Hit Ratio	15.07%
In Memory Sort Ratio	87.50%

### Analysis Report

This report describes the rationale for each recommendation made by Oracle Expert stored in the repository. The Analysis report information stays in the repository until you generate another analysis.

Use this report to understand how and why Oracle Expert made its decisions.

### Example

Structure Analysis for database DJ

Tablespace INDX

- o Alter tablespace INDX to specify PCTINCREASE of 0

Tablespace RBS

- o Alter tablespace RBS to specify MINEXTENTS value of 20
- o Alter tablespace RBS to specify MAXEXTENTS value of 90

...

Oracle Expert recommends increasing the log\_buffer instance parameter from :

8192 bytes --> 32768 bytes

The current value is below the threshold established as the recommended minimum for the size of a redo log buffer. Undersizing the redo log buffer will result in performance degradation while users wait for space in the redo log buffer to become available.

Evidence:

Log buffer setting : 8192 bytes

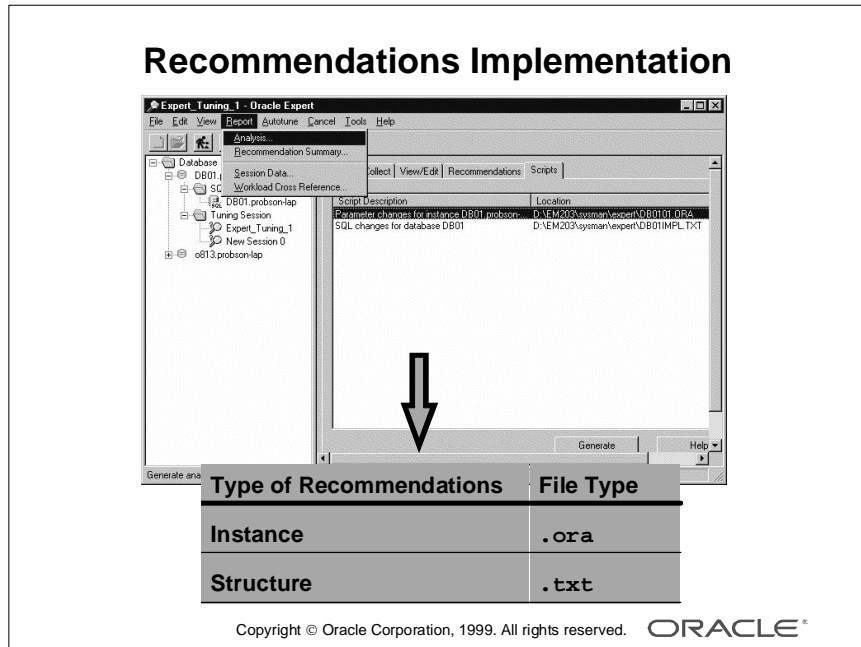
\*\*\*\*\* User Rule Settings \*\*\*\*\*

Minimum log buffer size : 32768 bytes

**Workload Cross Reference Report**

This report reviews tables and their associated requests (statements) or a request and its associated table, based on tuning session workload.

## Implementation



### Implementing the Recommendations

After Oracle Expert has analyzed the collected data and generated recommendations, you can have it create implementation files to help you make changes to your database.

- To implement recommendations, replace the instance parameter values in your `init.ora` file for the instance with the instance parameter values in the `.ora` file generated by Oracle Expert.
- To implement SQL statements and structure recommendations, examine the `.txt` file generated by Oracle Expert. This file contains SQL statements that include the string TBS in places where you must provide the appropriate information. After you have entered the correct information, you can execute the SQL statements in this file.



## Summary

### Summary

**In this lesson, you should have learned how to:**

- **Tune Oracle databases using the Oracle Expert system**
- **Store all tuning inputs and recommendations in the repository**
- **View and edit the tuning rules**
- **Generate the analysis report for all recommendations made by Oracle Expert**

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®



## **Multithreaded Server Tuning Issues**

## Objectives

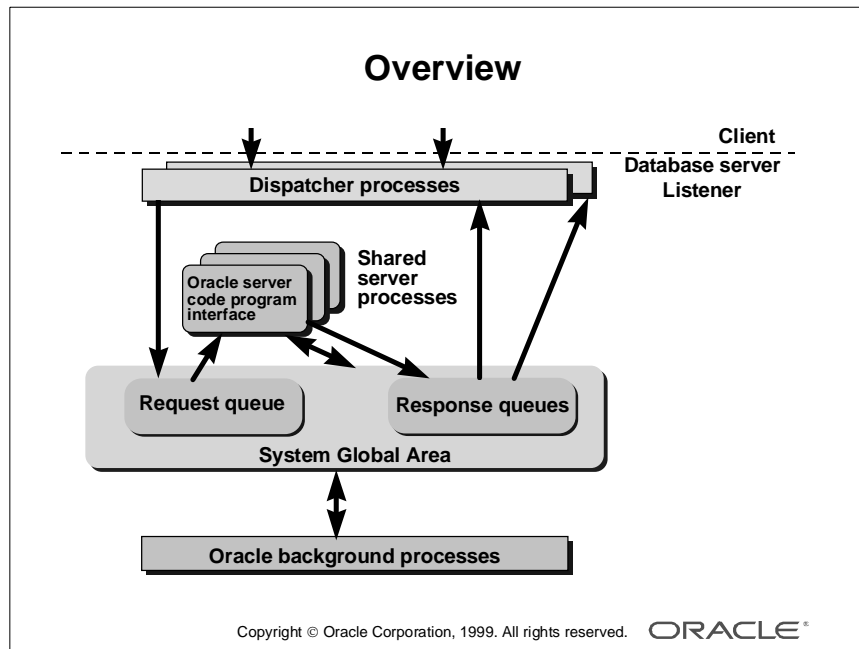
### Objectives

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

- **Identify issues associated with managing users in a multithreaded server environment**
- **Diagnose and resolve performance issues with multithreaded server processes**
- **Configure the multithreaded server environment to optimize performance**

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

## Overview



### Multithreaded Server

The fundamental architecture of the multithreaded server is to enable multiple user processes to share a limited number of servers. In a dedicated server environment, each user process is allocated a server process.

This server process, in turn, may not be fully used by a user process, due to idle time and inactivity. However, the allocation of this server process consumes both memory and CPU resources.

When using the multithreaded server, user processes are dynamically allocated to a server process that may be shared across many user processes. The dispatcher process receives a request from a user process and places it in the request queue, so that a shared server can process it and return the results to the response queue for the dispatcher. The dispatcher process will return the results to the user after the results are placed in the response queue.

A useful example is the reservations process for Oracle courses. A customer calls to inquire about booking a course. The reservations clerk has a screen to query course availability, but needs to talk to the customer for a few minutes first. The query is then sent to the database. After a bit more conversation, the customer decides which course to book, and the clerk commits the booking. In a ten-minute conversation, the dedicated server has been idle 99% of the time.

## Multithreaded Server Characteristics

### Multithreaded Server Characteristics

- **Users can share processes**
- **Supports NET8i functionality**
- **Increases number of concurrent users**
- **Is most useful on:**
  - **UNIX systems**
  - **Other servers with remote clients**
- **Incurs some CPU overhead**

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

### Characteristics

The multithreaded server (MTS) provides a way for users to share processes. It does not speed up performance, but enables a much greater number of concurrent users to connect to the database.

In Oracle8i, the multithreaded server supports both multiplexing and pooling for user connections in the form of Connection Manager and Connection Pooling.

Without MTS, each user on a standard UNIX system or who connects from a remote client, needs a dedicated server process to access Oracle8i Server files and memory structures. In an interactive application, where users spend much of their time talking to customers, these dedicated servers are mainly idle. They only have work to do when the user sends a query or a change to the database.

With MTS, multiple users can share dispatcher processes, which access the Oracle8i Server for them. Oracle8i uses shared servers to process the SQL statements passed in by the dispatchers.

### **Characteristics (continued)**

MTS is useful on:

- UNIX systems, where it saves the overhead of having a dedicated server for each user
- Other servers that have many remote clients
- Machines that are approaching limits on resources such as process slots or semaphores

There is no advantage in using shared servers for database-intensive work. Heavy or batch users should have dedicated servers.

## Configuring the Multithreaded Server

### Configuring the Multithreaded Server

- **NET8i**
  - listener.ora
  - tnsnames.ora
- **MTS instance parameters:**

```
mts_servers = 4
mts_dispatchers = "(PROTOCOL=ipc)(DISPATCHERS=4)"
mts_max_servers = 20
mts_max_dispatchers = 20
```

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

### Using MTS

You cannot use MTS unless you have installed and configured Net8i. Configuration is beyond the scope of this course (it is covered in the *Oracle8i: Network Administration* course), but means that you should have at least the following two files for your server machine.

Filename	Contents
listener.ora	The listener address and the instances to which it can connect
tnsnames.ora	Aliases for each instance, specifying usable connections

### Initialization Parameters

Parameter	Value
MTS_SERVERS	Initial number of shared servers
MTS_DISPATCHERS	protocol, number_of_dispatchers
MTS_MAX_SERVERS	Optional: maximum number of server processes
MTS_MAX_DISPATCHERS	Optional: maximum number of dispatcher processes

### Example of Parameter Values

```
MTS_SERVERS = 4
MTS_DISPATCHERS = "(protocol=ipc)(dispatchers=4)"
MTS_MAX_SERVERS = 20
MTS_MAX_DISPATCHERS = 20
```



## Monitoring Dispatchers

### Monitoring Dispatchers

Identify contention for dispatchers by checking:

- Busy rates
- Dispatcher waiting time

```
SQL> SELECT network
2  "Protocol",
3  SUM(busy) / ( SUM(busy) + SUM(idle) )
4  "Total Busy Rate"
5  FROM v$dispatcher
6  GROUP BY network;
```

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

### Identifying Contention for Dispatcher Usage

You can find the maximum number of users that can connect through a single dispatcher from the `MAXIMUM_CONNECTIONS` column of the `V$MTS` view. (This value defaults to the value of the `SESSIONS` parameter, if `SESSIONS` is set lower than the actual limit for a dispatcher).

You can also query the `V$DISPATCHER` view to determine the usage for selected dispatcher processes. You identify contention for dispatchers by checking:

- Busy rates for existing dispatchers
- Dispatcher waiting time

The query in the example returns the total busy rate for the dispatcher processes of each protocol; that is, the percentage of time the dispatcher processes of each protocol are busy. The result of this query might look like this:

Protocol	Total Busy Rate
-----	-----
decnet	.004589828
tcp	.029111042

A value of 0.5 means that a dispatcher has been busy 50% of the time. This is too high, and you would need to add more dispatchers.

## Monitoring Dispatchers

- Check for dispatcher contention
- Dynamically add or remove dispatchers
- Performance Manager predefined charts: Dispatcher and Queue

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

## Checking if Users Wait for Dispatchers

You can check if users are waiting for dispatchers by executing the following query:

```
SELECT DECODE(SUM(totalq),0,'No Responses',
              SUM(wait)/SUM(totalq)) "Average wait time"
FROM   v$queue q, v$dispatcher d
WHERE  q.type   = 'DISPATCHER'
AND    q.paddr  = d.paddr;
```

The answer is expressed in hundredths of a second. A steadily increasing value indicates a problem. You may want to start up more dispatchers at once or if you run the query twice or more and wait times seem to be increasing.

**Example** To add or remove dispatchers, use the command:

```
ALTER SYSTEM SET mts_dispatchers = 'protocol, number';
```

Allocating more dispatchers will not have any immediate effect, because users are bound to the same dispatcher until they log off. Only new connections can make use of the new dispatchers.

## Performance Manager

When diagnosing performance issues associated with dispatchers and queues, you can use the predefined charts, Dispatcher and Queue.

Oracle Diagnostics Pack—>Performance Manager—>Contention—>Dispatcher

Oracle Diagnostics Pack—>Performance Manager—>Contention—>Queue

## Monitoring Shared Server Processes

### Monitoring Shared Servers

Oracle8i starts up shared servers dynamically.

- Check for shared server process contention
- Dynamically add or remove shared servers
- Use Performance Manager charts:
  - Shared Server
  - Queue

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

### Monitoring Services

If you underestimate the number of shared servers you need, the Oracle8i Server starts up more shared servers dynamically. The extra servers are removed again when they are idle. This means that there is less need to monitor servers than dispatchers.

However, you may have started up more shared servers than you need. If you specify a value for MTS\_SERVERS in the parameter file, the Oracle8i Server does not kill these processes even if they are idle, so you may want to check the loading.

You may also want to find out whether the number of servers is approaching the value of MTS\_MAX\_SERVERS or (even worse) is bringing the number of processes close to the value of the PROCESSES parameter.

You remove shared servers by using the following command:

```
ALTER SYSTEM SET MTS_SERVERS = number;
```

## Monitoring Services (continued)

You determine how many shared server processes are currently running by issuing this query:

```
SELECT COUNT(*) "Shared Server Processes"
FROM v$shared_server
WHERE status != 'QUIT';
```

The result of this query might look like this:

```
Shared Server Processes
-----
10
```

You can determine if there is contention for shared server processes by querying the WAIT and TOTALQ columns of the V\$QUEUE dynamic performance view.

Monitor these statistics occasionally while your application is running:

```
SELECT DECODE( totalq, 0, 'No Requests',
wait/totalq || ' hundredths of seconds')
"Average Wait Time Per Requests"
FROM v$queue
WHERE type = 'COMMON';
```

This query returns the total wait time for all requests and total number of requests for the request queue. The result of this query might look like this:

```
Average Wait Time per Request
-----
.090909 hundredths of seconds
```

## Performance Manager

When diagnosing performance issues associated with a user process, you can use the predefined chart, Shared Server.

Oracle Diagnostics Pack—>Performance Manager—>Contention—>Shared Server

Oracle Diagnostics Pack—>Performance Manager—>Contention—>Queue

---

## Monitoring Process Usage

### Monitoring Process Usage

- The V\$CIRCUIT view:
  - Server address
  - Dispatcher address
  - User session address
- Performance Manager charts: predefined charts  
Process, Circuits

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

### Checking Shared Connections

If a user has a problem or a process seems to be doing too much work, you may need to check on current users with shared connections.

You can query current dispatcher and server use with the V\$CIRCUIT view that gives you server and dispatcher addresses, and the session address for the user.

You need to check with V\$DISPATCHER, V\$SHARED\_SERVER, and V\$SESSION for the corresponding values in the NAME and USERNAME columns.

### Performance Manager

When diagnosing performance issues associated with a user process, you can use the predefined chart, Circuit.

Oracle Diagnostics Pack—>Performance Manager—>Contention—>Circuit

## Shared Servers and Memory Usage

### Shared Servers and Memory Usage

- **Some user information goes into the shared pool**
- **Overall memory demand should still decrease**
- **Shared servers use UGA for sorts**
- **UGA stored in large pool if configured**

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

### Using MTS for Search Servers

If you decide to use MTS, some of the information that the Oracle8i Server would otherwise keep in the user's memory area must go into the shared pool, instead. This User Global Area was described in a previous lesson, "Tuning the Shared Pool."

The overall memory demand on the system should still decrease using MTS, because the increase in the shared pool is compensated by a decrease in the user's own memory demands.

Shared servers use the UGA for sorts, so if you are using MTS, you may consider setting `SORT_AREA_RETAINED_SIZE` smaller than `SORT_AREA_SIZE`, so that memory can be released back for other users as quickly as possible.

---

## Possible Problems

### Possible Problems

- **Net8*i* listener is not running.**
- **The MTS initialization parameters are set incorrectly.**
- **The dispatcher process has been terminated.**
- **The DBA does not have a dedicated connection.**
- **The PROCESSES parameter is too low.**

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

## Troubleshooting

Troubleshooting the multithreaded server environment is a key DBA function. Some common problems include:

- If the Net8*i* listener is not running, all attempts at shared connections fail. You need to bring it up whenever the machine is booted.
- Any Net8*i* configuration error gives a TNS\_ error message when you try to establish a shared connection.
- It is always bad practice to terminate a user's server process at the operating system level (use KILL SESSION inside Oracle8*i* instead). But if a user is connected through a dispatcher, it is even more dangerous, because killing the dispatcher may affect many other users as well.
- You cannot perform privileged operations such as STARTUP and SHUTDOWN using a shared server. Make sure you have a dedicated connection for yourself as DBA.
- Your servers and dispatchers count as background processes for the instance. Be careful that your setting of PROCESSES allows for all possible servers and dispatchers, or new users may not be able to log in. Setting MTS\_MAX\_SERVERS and MTS\_MAX\_DISPATCHERS can act as a useful ceiling.

## Obtaining Dictionary Information

### Obtaining Dictionary Information

Dynamic performance views:



**V\$CIRCUIT**  
**V\$DISPATCHER**  
**V\$DISPATCHER\_RATE**  
**V\$QUEUE**  
**V\$MTS**  
**V\$SESSION**  
**V\$SHARED\_SERVER**

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

### Using Dynamic Performance Views

You can use different dynamic performance views to obtain information about the multithreaded server environment. Use the Oracle8i reference manual to obtain details about each view.

- **V\$CIRCUIT**: Contains information about virtual circuits, which are user connections to the database through dispatchers and servers
- **V\$DISPATCHER**: Provides information on the dispatcher processes
- **V\$DISPATCHER\_RATE**: Provides rate statistics for the dispatcher processes
- **V\$QUEUE**: Contains information on the multithread message queues
- **V\$MTS**: Contains information for tuning the multithreaded server
- **V\$SESSION**: Lists session information for each current session
- **V\$SHARED\_SERVER**: Contains information about the shared server processes



## Summary

### Summary

**In this lesson, you should have learned that:**

- **MTS is a resource-sharing configuration.**
- **MTS is not intended for batch processing or decision support.**
- **MTS requires a Net8i listener.**
- **You can monitor dispatcher and server usage.**
- **The Oracle8i Server manages shared servers dynamically.**

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

## Quick Reference

Context	Reference
Parameters	MTS_DISPATCHERS MTS_SERVERS MTS_MAX_SERVERS MTS_MAX_DISPATCHERS PROCESSES LARGE_POOL_SIZE
Dynamic performance views	V\$CIRCUIT V\$DISPATCHER V\$DISPATCHER_RATE V\$QUEUE V\$MTS V\$SESSION V\$SHARED_SERVER V\$SGASTAT
Data dictionary views	None
Commands	ALTER SYSTEM SET MTS_SERVERS ALTER SYSTEM SET MTS_DISPATCHERS
Procedures	None
Oracle Diagnostics Pack	Performance Manager

---

17

---

## Workshop

## Objectives

### Objectives

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

- **Use a tuning methodology for diagnosing and resolving performance issues**
- **Use Oracle tools for diagnosing performance problems**
- **Tune memory structures, file I/O, and contention**

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

## Workshop Methodology

### Workshop Methodology

- **Group-oriented and interactive**
- **Intensive hands-on diagnosis and problem resolution**
- **Instructor-led discussions on findings and actions**
- **Proactive participant involvement**

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

### Group-Oriented and Interactive

The workshop is structured to allow groups of individuals to work together to perform tuning diagnostics and resolution. Each group is encouraged to share its tuning diagnostic and resolution approach with other groups in the class.

### Intensive Hands-On Diagnosis and Problem Resolution

The intent is to provide students with as much hands-on experience as possible to diagnose and work through a performance tuning methodology, diagnosis, and resolution. The experience and knowledge gained from the first three days of the Oracle8: Performance Tuning Workshop will play a major role toward successfully completing the workshop.

## Troubleshooting Scope

### Tuning Scope

Use Oracle tools to tune the following areas:

- Memory
- I/O
- Resource contention

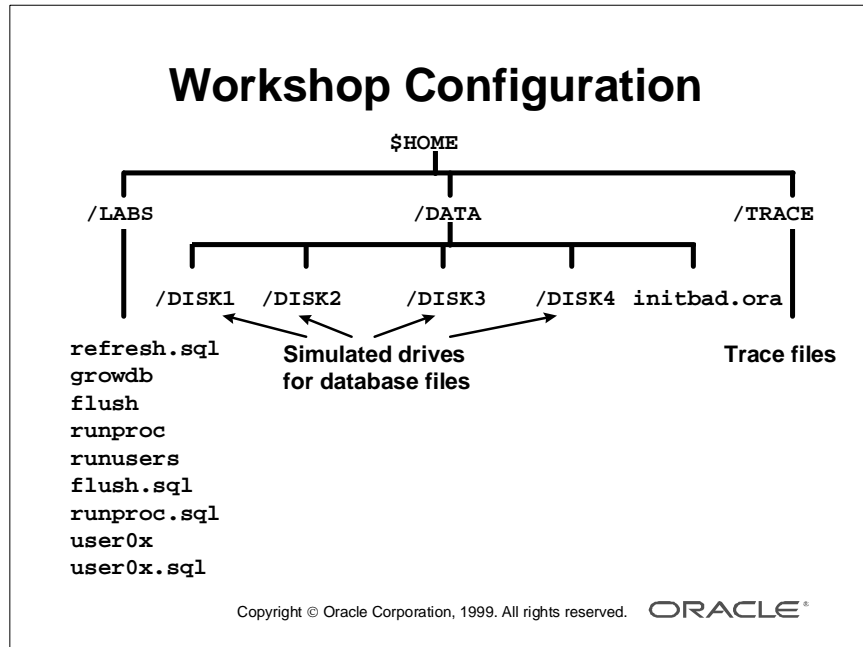
Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

### Tuning Areas

This workshop addresses four main tuning areas and the tools that are appropriate for each. These include:

- Memory: Focus is on memory structures in the SGA
- I/O: Focus is on file I/O and the cause and effect that improperly tuned memory structures may have upon file I/O
- Resource contention: Focus is on contention for resources, which also includes latches and locks

## Directories Configuration



### Directories

Each lab account is set up in the following manner:

- Each user account has its own `$HOME` directory.
- Four directories under `$HOME/DATA` are used to simulate multiple physical devices (`DISK1`, `DISK2`, `DISK3`, and `DISK4`).
- The `LABS` directory contains an `initbad.ora` parameter file. It is important to use the `initbad.ora` parameter file because it has been set up to result in specific performance tuning problems. Where appropriate, rename any instance-specific parameters in the `initbad.ora` file, such as `DB_NAME`, control files, path names, and so on.
- The `TRACE` directory is the default for instance-specific trace files.
- The `LABS` directory contains lab files.

## Scripts

- `refresh.sql`: This SQL script simulates an instance that has been running for some time.
- `growdb`: This shell script executes other shell scripts, such as `flush`, `runproc`, and `runusers`, that each runs a SQL script several times
  - `flush.sql` runs `ALTER SYSTEM` and `ALTER TABLE` commands
  - `runproc.sql` executes procedures
  - `runusers` executes a different shell script (`user0x`) that runs a different SQL script (`user0x.sql`) under different users (`user01`, `user02`, `user03`, `user04`, and `user05`) to insert, update new rows in SCOTT's tables.



## Workshop Database Configuration

### Workshop Database Configuration

- One schema is created under `scott/tiger`.
- There are six end users (`user01-05`, `scott`).
- End users have access to `scott`'s objects.
- Four tablespaces: `system`, `rbs`, `user_data`, `temp`
- The DBA account is `system/manager`.
- The `sys` account is `sys/change_on_install`.

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

### Database Configuration

Use the Oracle data dictionary views to get a complete picture of the database configuration.

## Information Gathering

### Information Gathering

- **Ask the instructor questions regarding performance tuning issues as they apply to the simulated database environment.**
- **Formulate questions that will enable you to familiarize yourself with external factors that may affect performance and will aid you in establishing a tuning methodology.**

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

### Objectives of the Discussion

Take the opportunity to ask the instructor questions regarding performance tuning issues as they apply to the simulated database environment. Formulate questions that will familiarize you with external factors that may affect performance and will aid in establishing a tuning methodology.

Such factors include:

- Number of users
- Type of performance problem: length of time to commit, retrieval of information, and so forth
- Nature of application: OLTP, batch, data warehouse, DSS, hybrid
- Time-of-day performance issues encountered; resource contention
- Any unusual activity, load on system
- Whether changes have been made to the functionality of the applications
- What statistics are being captured
- Transactions over a selected period of time
- Peak usage times

**Objectives of the Discussion (continued)**

- Dedicated server architecture
- Availability
- Backup/recovery requirements

At the end of this discussion, you should be able to:

- Formulate a strategy for gathering information that will aid in performance tuning
- Develop methodical performance tuning skills
- Consider the variety of possible issues that may affect database performance

**Example** In a real-world environment, an end user may call the DBA and report that an update is taking an unusually long time to complete. Questions to ask may include the following:

- How long does the update usually take?
- When was the problem first noticed?
- What is an acceptable amount of time?
- Does the problem appear to be more noticeable during certain times of the day?
- Has the amount of data that is being updated increased significantly over time?

Questions such as these can direct the DBA toward specific areas of the `report.txt` output, when to run the `utlbstat/utlestat` utilities, and what areas demand attention, such as application, memory, I/O, or contention.

**Considerations**

Another area of concern to the DBA is the ability to be proactive in the performance tuning activity. Answers to questions in this area allow the DBA to establish a long-term performance tuning and maintenance strategy as opposed to performing short-term fixes. Questions to ask may include the following:

- How much will the user base grow over the next six months to one year?
- What will be the primary types of database-level activity—OLTP, reporting, data warehouse?
- What are the long-term backup and recovery requirements?
- Are there any planned hardware or software upgrades?
- Is there any intent to install additional applications on the same database server?

## Statistics

### Generate Statistics

To perform a physical investigation of your workshop database environment, generate statistics using:

- V\$ dynamic performance views
- Data dictionary views
- Table statistics
- Various hit ratios
- utlbstat/utlestat report.txt output

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

### Physical Investigation

Perform a physical investigation of your workshop database environment. Some areas that you may want to focus on are listed below as a guide. Remember to use the tools that are available to you within the Oracle environment, such as the V\$ dynamic performance views, data dictionary views, table statistics, various hit ratios, and so forth. Use the statistics and ratios presented during the first three days of the class to analyze the baseline `report.txt`. A number of statistics in the `report.txt` file are contrary to a well-tuned database.

### How to Start

- 1 Connect to the UNIX server and log in to SQL\*Plus as `sys`.
- 2 Start your instance using the `initbad.ora` in the `$HOME/DATA` directory. Do not change the settings of any of the initialization parameters.
- 3 Run the `refresh.sql` script to simulate instance activity.
- 4 Run `$ORACLE_HOME/rdbms/admin/utlbstat.sql` as `sys`.
- 5 Execute the `growdb` UNIX shell script to simulate database activity.
- 6 Wait approximately 15 minutes and run `$ORACLE_HOME/rdbms/admin/utlestat.sql` as `sys`.
- 7 Once the `utlestat.sql` script has completed, shut down your instance.

**Note:** While the `growdb` script is executing, you can use available tools to monitor database performance. Also ensure that you are in a writeable directory when you run the `utlestat.sql` script.

---

## Review

### Review Statistics

Review statistics regarding specific areas:

- Shared pool diagnostics, errors, and sizing
- Rollback segments placement, sizing, and numbering
- Buffer cache diagnostics and sizing
- Redo log buffer contention
- Files organization, sizing, and I/O distribution

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

### Instance Memory Structure Use

- Free memory in the shared pool
- Shared pool usage and hit ratio
- Database buffer cache usage and hit ratio
- Redo log buffer contention

### Rollback Segments

- Number of rollback segments
- Tablespaces for rollback segments
- Rollback segments online
- Sizing parameters adequate

Contention for rollback segment headers

## **Review Statistics**

- **Segment differentiation**
- **Storage management issues, diagnostics, and resolution**
- **Row migration and chaining diagnostics and resolution**
- **Sort operation diagnostics and configuration**
- **Lock, latch, free list, and rollback segments contention and configuration**

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

## **Organization of Data and Data Files, I/O Distribution**

- Have files been distributed across all physical devices?
- Is there contention for system and rollback segment tablespace files?
- Default and temporary tablespaces for users properly set
- Non-data dictionary objects in the system tablespace
- Excessive checkpointing

## **Segment Differentiation**

- Data and index segments separated
- Independent rollback segment tablespace

## **Redo Log Files**

- Files mirrored
- Sized correctly (A high number of log switch events should be seen in the `alert.log` because of the small size of the online redo log files.)
- Excessive log switches

## **Block Usage**

- Row chaining
- Parameter settings

### **Latch Contention**

- Freelist contention
- Redo allocation and copy latch contention

### **User Session Information**

- Session waits
- Memory use
- Transaction information

## Presentation

### Example

#### Physical investigation:

- **Memory structures:**
  - Buffer cache hit ratio low
  - Library cache hit ratio low
  - V\$SYSSTAT
- **Contention:**
  - Rollback segment contention
  - Latch contention—>V\$LATCH

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

### Presenting the Facts

This discussion provides each group with the opportunity to present its conclusions and the list of facts issued from the statistics review to other members in the workshop. The intent is to discuss how each group interpreted the information provided by the instructor during the information-gathering step and the data collected during the physical investigation to formulate a tuning strategy. Tailor your presentation to the specific tuning area assigned to you by the instructor.



---

## Analysis

### Application Analysis

**Application analysis:**

- **Online transaction processing separated from batch**
- **Explain plan indicates no use of indexes**
- **High number of disk sorts**
- **Trace files**

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

## Analysis

According to the facts, list the actions to modify the instance and database configuration to enhance database performance.

Instance and database modifications should be based on statistics analyzed in the `report.txt` baseline report and the physical investigation. Remember that the intent is to establish a tuning methodology, not to follow a “hunt-and-peck” approach. The type of items that can be changed are:

- **Initialization file parameters:** Change parameters as necessary to increase the size of the shared pool, database buffer cache, and so on.
- **Additional tablespaces:** Evaluate the need to add tablespaces for index and rollback segments.
- **Database object placement:** Separate user objects from system objects and possibly change the default and temporary tablespaces for users. Also review the possibility of separating segment types such as index, temporary, and rollback segments, and placing objects in locally-managed tablespaces.
- **Batch jobs:** Review and alter existing batch jobs to change execution times and intervals to minimize contention with online transaction-processing activities.

### **Analysis (continued)**

- Log contention: Modify the size of redo log files, the number of redo log files, the number of redo log file groups, archive destination, and appropriate `init.ora` parameters.
- Parallelization: Review settings on operations that could be better served using parallelization.
- Application: Are packages, procedures, functions, and triggers pinned in the shared pool? Are indexes being used? Check object storage parameters.

---

## New Statistics

### Regenerate Statistics

- Restart the instance with the new `initbad.ora` parameters.
- Run the `refresh.sql` script to simulate an instance that has been running for some time.
- Rerun the `utlbstat` script.
- Run the `growdb` shell script.
- Rerun the `utlestat` script.
- Shut down the instance.
- Review the new statistics.

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

### Simulate Database Activity

Before you can determine the results of your tuning strategy, you must run the `refresh.sql` script that will populate the database memory structures to simulate an instance that is representative of one that has been running for some time.

### Regenerate New Statistics

- 1 Make a copy of your original `report.txt` by renaming it `report.old`.
- 2 Make a copy of your original `initbad.ora` by renaming it `initbad.old`.
- 3 Modify the `initbad.ora` with the new parameters you have chosen as part of your tuning strategy.
- 4 Connect to the UNIX server and log in to SQL\*Plus as `sys`.
- 5 Start your instance using the modified `initbad.ora`.
- 6 Run the `refresh.sql` script.
- 7 Run `$ORACLE_HOME/rdbms/admin/utlbstat.sql` as `sys`.
- 8 Execute the `growdb` UNIX shell script.
- 9 Wait approximately 20 minutes and run the `$ORACLE_HOME/rdbms/admin/utlestat.sql` script as `sys`.
- 10 Once the `utlestat` script has completed, shut down your instance.

### **Regenerate New Statistics (continued)**

**Note:** While the `growdb` script is executing, you can use available tools to monitor database performance. Ensure that you are in a writeable directory when you run the `utlstat.sql` script.

Once the scripts are completed, conduct another physical investigation by reviewing the new `report.txt` output, database dictionary views, and trace files. Compare your findings with the original `report.txt` output to determine the overall effectiveness of your tuning strategy by walking through the checklist in Practice 16 Overview in Appendix A. Use these findings as the foundation for your presentation.

### **Example Presentation Areas of Interest**

These areas may include the following:

- Shared SQL area
- Library cache/dictionary cache statistics
- Database buffer cache
- Database writers/checkpoint performance
- Redo log buffer/redo log file
- Sorts/table scans/segment usage
- Rollback segment contention/issues/configuration
- File I/O, distribution, contention

---

## Results

### Results

- **Present your conclusions and findings.**
- **Demonstrate the effectiveness of the tuning strategy, and what effect the changes to the instance and database parameters had on overall performance.**
  - **What was done and why?**
  - **What were the results?**
  - **Are there still any issues pending?**
  - **What would you do differently?**

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

### Conclusions

This discussion should be conducted in a manner similar to the previous presentation. Each group will present its conclusions and findings to the rest of class. The intent is to demonstrate the effectiveness of the tuning strategy and show what effect the changes over the baseline the modifications to the instance and database parameters had on overall performance. Include the following items in your presentation:

- What was done?
- What was the justification?
- What were the results?
- Are there any pending issues?
- What would you do differently?

## Post-Tuning Actions

### Example

- **Library cache hit ratio increased from 53% to 81%.**
- **Database buffer cache hit ratio increased from 67% to 92%.**
- **Sorts (disk) decreased, sorts (memory) increased.**
- **Undo header waits decreased.**
- **Distribution of hot files evened out.**

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

### Post-Tuning Actions

You must think about additional actions that need to be performed later or regularly due to:

- Current changes that impact other areas (A larger SGA decreases the OS system memory usage by other applications.)
- Changes that require a long-term window, such as:
  - Increasing the database block size (This operation requires that the database be rebuilt, and therefore a full database export/import.)
  - Regular export/import to reduce fragmentation to be done during offline work

The necessity of measuring new results by periodic monitoring, using `utlbstat/`  
`utlestat`.

### **Additional Concerns**

- **Monitor paging and swapping using OS utilities because of larger SGA.**
- **Consider increasing the size of database blocks.**
- **Export/import to reduce fragmentation.**
- **Continue monitoring using `utlbstat`/`utlstat` to measure results against the baseline.**
- **Separate index segments from data segments.**

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

## Pending Performance Tuning Issues

### Pending Performance Tuning Issues

- Importance of good database and application design
- Architecture configurations:
  - Multithreaded server
  - Parallel query
  - Partitioning
- Recommendations for proactive performance tuning

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

### Configurations

Some of the available configurations can offer better performance results:

- Multithreaded server
  - Memory usage
  - Process and network resource consumption
- Parallel query: response time
- Partitioning:
  - I/O distribution
  - Response time
  - Fragmentation



## Summary

### Summary

In this lesson, you should have learned how to:

- Follow a tuning methodology:
  1. Collect and review statistics.
  2. List the objectives for enhanced performance before modifications.
  3. Modify the instance and the database.
  4. Recollect and review new statistics.
  5. Compare the new results with the objectives.
- Implement Oracle architectural options for enhancing performance.

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®



---

A

---

**Practices**

## Practice 3-1

The goal of this practice is to familiarize yourself with the machine setup and to have a look at the diagnostic tools, such as trace files, waiting event views, and EM event management.

- 1** Log on as directed by your instructor.
- 2** View your parameter file and check that the appropriate parameters locate the alert log file and user trace files, respectively, in the `BDUMP` and `UDUMP` sub-directories in your home directory. If necessary shut down, then start up, your instance.
- 3** Examine your `alert.log` file to find any internal errors (ORA-600).
- 4** As `SYSTEM`, connect to `SQL*Plus` and issue a command that will create a user trace file after querying against the `DBA_TABLES` data dictionary view. Do not try to interpret the content of the trace file. You will learn how to do this in Lesson 13, “SQL Issues and Tuning Considerations for Different Applications.”
- 5** View the resulting trace file without trying to interpret the content of the trace file. You will learn how to do it in Lesson 13, “SQL Issues and Tuning Considerations for Different Applications.”

## Practice 4-1

The objective of this practice is to familiarize yourself with different ways of retrieving statistics information: run the `utlbstat.sql` and `utlestat.sql` scripts, examine the resulting `report.txt` file, display the major performance dynamic views appropriate for tuning, and know how to launch one of the Diagnostics Pack applications.

During the labs, use the `$HOME/LABS/initUn.ora` parameter file to start up the database.

- 1 Before running the `utlbstat.sql` script, verify that the required `init.ora` parameter has been set to true.
- 2 Run the `utlbstat.sql` script connected to SQL\*PLUS as SYSDBA.
- 3 Start the following three sessions as the user SCOTT:
  - The first session executes the `lab04_1.sql` script (it creates and loads data into SCOTT's tables). Wait until the script has finished.
  - The second session then executes the `lab04_2.sql` script.
  - At the same time, the third session executes the `lab04_3.sql` script.Do not wait until the last two scripts have finished loading data. Go to step 4.
- 4 Query the appropriate dynamic view to see:
  - The library cache performance
  - Among the system statistics, those concerning sort activity
  - The sessions consuming more than 10,000 bytes of PGA memory
  - The rollback segments statistics
  - The file I/O statistics
- 5 Run the `utlestat.sql` script so that the resulting report is stored in your UDUMP directory.
- 6 Examine the resulting `report.txt` file and compare it to what you displayed in:
  - The library cache dynamic view with the library cache section of the report
  - File I/O dynamic view with the I/O statistics section of the report
- 7 Display the list of Wait event names.
- 8 Are there any sessions actually waiting for resources?

## Practice 5-1

The objective of this practice is to use available diagnostics tools to monitor and tune the shared pool.

- 1** Check the size of the shared pool.
- 2** From SQL\*Plus, connected as SYSDBA, run the `utlbstat.sql` script to start collecting statistics.
- 3** To simulate user activity against the database, connect as SCOTT and run the `lab05_1.sql` script and in another session `lab05_2.sql`.
- 4** Measure the pin hit ratio for the library cache. Determine if it is a good ratio or not.
- 5** Measure the hit ratio for the data dictionary cache. Determine if it is a good ration or not.
- 6** When both scripts have finished running, from SQL\*Plus and connected as SYSDBA, run the `utlestat.sql` script to collect ending statistics.
- 7** Use the `utlestat.sql report.txt` output to get both hit ratios.
- 8** Which solutions would you apply if any of the hit ratios were bad?
- 9** Determine which packages, procedures, and triggers are pinned in the shared pool.
- 10** Pin one of the Oracle supplied packages that needs to be kept in memory.
- 11** Determine the amount of session memory used in the shared pool for your session.
- 12** Determine the amount of session memory used in the shared pool for all sessions.

## Practice 6-1

The objective of this practice is to use available diagnostics tools to monitor and tune the database buffer cache.

- 1** From SQL\*Plus, connected as SYSDBA, and run the `utlbstat.sql` script to start collecting statistics.
- 2** To simulate user activity against the database, connect as SCOTT and run the `lab06_1.sql` script.
- 3** Connect as SYSTEM and measure the hit ratio for the database buffer cache. Determine if it is a good ratio or not.
- 4** When the script has finished, from SQL\*Plus and connected as SYSDBA, run the `utlestat.sql` script to collect ending statistics.
- 5** Use the `report.txt` output to get the cache hit ratio. What is the ratio?  
If the ratio is bad, add new buffers, run steps 2 to 5, and examine the new ratio to verify that the ratio has improved. If the ratio is good, remove buffers, run steps 2 to 5, and verify if the ratio is still good.
- 6** Size the keep buffer pool to hold the SCOTT.S\_EMP and SCOTT.S\_DEPT tables.
- 7** Modify the `init.ora` file to set up the keep buffer pool.
- 8** Enable the SCOTT.S\_EMP and SCOTT.S\_DEPT tables for caching in the keep pool.
- 9** Set up both of these tables so that their blocks are kept in memory, even during full table scans.
- 10** Determine how many LRU latches have been allocated for your instance.
- 11** Connect as SYSTEM and display the hit percentage for the LRU latches.
- 12** Determine what the overall hit percentage is for free lists in your instance.

## Practice 7-1

The objective of this practice is to use available diagnostics tools to monitor and tune the redo log buffer.

- 1 Connect as **SYSTEM** and measure the redo buffer allocation retries for the redo entries. Determine if it is a good ratio or not.
- 2 Which solution would you apply if the ratio were bad?
- 3 Change the size of the appropriate parameter.



## Practice 8-1

- 1** Have a look at your database. Find out if there are deficiencies concerning the database configuration. If necessary, change the database configuration. Consider the following:
  - a** How many tablespaces do you have? Add more tablespaces if necessary, spreading them across the subdirectories that represent devices.
  - b** Are there any non-data-dictionary objects in the SYSTEM tablespace? Objects can be moved by creating one table as a copy of another, dropping the original object, and then renaming the new object to have the original name.
  - c** Check to see if you have the data files on different disks from your online redo log files. Also, the archive logs should be on different disk from both the data files and redo log files.
  - d** Are the online redo log files mirrored?
- 2** Find out how often checkpoints are occurring. Is this timing optimal? If not, correct it.
- 3** To simulate user activity against the database, connect as SCOTT and run the `lab08_1.sql` script. Monitor the I/O statistics.

## Practice 9-1

The objective of this practice is to use available diagnostics tools to monitor and tune block space usage.

- 1 Connect as **SYSTEM** and find the **PCTFREE**, **CHAIN\_CNT** and high-water mark for **SCOTT's S\_ORD** table.
- 2 Run the `lab09_1.sql` script to update rows in **SCOTT.S\_ORD**
- 3 Find the **CHAIN\_CNT** and high water mark for **SCOTT's S\_ORD** table.
- 4 Eliminate any migrated rows.
- 5 Find the **CHAIN\_CNT** and high-water mark for **SCOTT's S\_ORD** table.
- 6 Remove the rows from table **CHAINED\_ROWS** without deleting them.

### Optional Practice

Find the high-water mark for **SCOTT's S\_ORD** table without querying the data dictionary. You can use the `lab09_2.sql` script.

## Practice 10-1

The objective of this practice is to use available diagnostics tools to monitor and tune sorts.

- 1** Connect as **SYSTEM** and execute the `lab10_1.sql` script. It forces sorts to write to disk.
- 2** Measure the sorts (disk) to the sorts (memory) ratio.
- 3** Configure the **`SORT_AREA_SIZE`** parameter to result in sorts (memory) only.
- 4** Connect as **SCOTT** and execute the `lab10_2.sql` script and recalculate the ratio.
- 5** Create a new **TEMPORARY** tablespace; and when **SCOTT** runs sort operations, the sorts to disk occur on this tablespace.
- 6** Connect as **SCOTT** and reexecute the `lab10_1.sql` script. Monitor the **TEMPORARY** tablespace.

## Practice 11-1

The objective of this practice is to use available diagnostics tools to monitor and tune the rollback segments. You will use the `report.txt` output, so run `utlbstat.sql` first.

- 1 Run the `utlbstat.sql` script. If you have not already done so, create a new rollback segment tablespace at least 2 MB in size. Connect as SYSTEM.
- 2 For the purposes of this practice, create a new rollback segment called RBSX. Give it an overall size of 200 KB, divided into 20 extents. It should shrink back to 200 KB if expanded.
- 3 Bring your new rollback segment online and take any others (except the SYSTEM rollback segment) offline.
- 4 Before executing a new transaction, find the number of bytes written in the new rollback segment.
- 5 As SCOTT, run the script `ins_ords.sql`. The script inserts 100 new rows into the S\_ORD table. Without committing, log in again as SYS (either use the HOST command or open another window).  
How many rollback segment blocks or bytes is the transaction using?
- 6 Return to the SCOTT session and commit the insert. Run the `del_ords` script, without committing. This script deletes the hundred rows you have just inserted. As SYS, check the amount of rollback space used, as in step 5. What is the difference between the space used for the original insert and the space used for the delete?
- 7 As SYS, find out if you have had any rollback segment contention since startup, using two different sources of information.
- 8 Does the V\$SYSTEM\_EVENT view show any waits related to rollback segments?
- 9 The tested transaction will be executed simultaneously by an average of ten users during the day. Every night, a batch job will load ten times the same volume as the `ins_ords.sql` script. Prepare the script to create the appropriate rollback segments with appropriate storage parameters to optimize space usage.  
*Do not run the script.*
- 10 As SCOTT, run the `lab11_1.sql` script again, allocating the transaction to a specific rollback segment, and check that the transaction is working with the defined rollback segment.

## Practice 12-1

The objective of this practice is to use available diagnostics tools to monitor lock contention.

You will need to start three sessions, in separate windows. Log in twice as SCOTT (sessions 1 and 2) and once as SYS (session 3).

- 1 In session 1, update the salary of one of the employees in the S\_EMP table:

```
SQL> UPDATE s_emp SET salary=salary*1.1
      2 WHERE id=1;
```

- 2 In session 3, check to see if any locks are being held.

- 3 In session 2, drop the S\_EMP table. Does it work?

- 4 In session 2, update a different row in the S\_EMP table.

```
SQL> UPDATE s_emp SET salary=salary*1.1
      2 WHERE id=2;
```

In session 3, check to see what kind of locks are being held.

- 5 Roll back the changes you made in session 2, and try to update the same row as in session 1.

```
SQL> UPDATE s_emp SET salary=salary*1.1
      2 WHERE id=1;
```

Check to see that the locks held in session 3, using any tool.

- 6 In session 3, terminate the session that is holding the lock.

## Practice 13-1

The objective of this practice is to familiarize you with SQL statement execution plans and to interpret the formatted output of a trace file generated using SQL Trace and the formatted output generated by TKPROF. You will also learn how to manipulate statistics using the Oracle supplied DBMS\_STATS package.

- 1 Ensure the current directory is \$HOME/LABS.
- 2 Connect as SCOTT and create the PLAN\_TABLE under the SCOTT schema.
- 3 Describe PLAN\_TABLE.
- 4 Use SQL\*Plus AUTOTRACE to look at the execution plan worked out by the optimizer for the SQL statement in the lab13\_1.sql script.
- 5 Use SQL Trace and TKPROF to look at the performance of the query executed by the lab13\_2.sql script.
  - a Set up your session in trace mode.
  - b Run the lab13\_2.sql script.
  - c Exit from SQL\*Plus and format your trace file using TKPROF. Use the options SYS=NO and EXPLAIN.
  - d You should only note down the CPU, current, and query figures for the fetch phase. You will not see a drastic change in numbers. The objective is to become familiar and comfortable with running TKPROF and SQL Trace.

Results:

Query	CPU (secs)	Query (blocks)	Current (blocks)
<pre>select e.last_name, d.name from   s_dept d, s_emp e where  d.id = e.dept_id and    d.id = 10;</pre>			

- 6 Gather statistics for all objects under the SCOTT schema while saving the current statistics, then restore the original statistics.
  - a Connect as SCOTT and create a table to hold statistics in that schema.
  - b Save the current schema statistics into your local statistics table.
  - c Analyze all objects under the SCOTT schema.
  - d Remove all schema statistics from the dictionary and restore the original statistics you saved in step b.

## Practice 14-1

- 1 Create a consumer group called ONLINE\_USERS.
- 2 Create a resource plan called DAYTIME with the following characteristics:
  - Users in the SYS\_GROUP get as much CPU time as they require.
  - Users in the ONLINE\_USERS group get 80% of the remaining resources.
  - Users in the OTHER\_GROUPS get 20% of the remaining resources.

There is no parallel query processing occurring.

This is the only plan that will be created, so complete the process of creating a plan.

- 3 Make ONLINE\_USERS the default consumer group for User01.
- 4 Give User01 the capability to switch to the SYS\_GROUP.
- 5 Activate DAYTIME as the current resource plan group.
- 6 Start a second SQL\*Plus session and log on as User01. Display the current consumer group for User01's session.
- 7 As User01, switch to the SYS\_GROUP.
- 8 Redisplay User01's current consumer group.

## Guided Practice 17-1

### Step 1: Shared SQL Areas and Private SQL Area

**Reduce Library Cache Misses** You can do this by increasing the memory allocated to library cache until the statistics RELOADS is near 0.

- RELOADS should not be more than 1% of PINS.
- Increase the number of cursors permitted for each session by increasing OPEN\_CURSORS.

**Speed Up Access to Shared SQL and PL/SQL Areas in Library Cache** You can do this by setting the value of initialization parameter CURSOR\_SPACE\_FOR\_TIME. If it is set to TRUE, this implies that a shared SQL area can only be deallocated when all applications cursors associated with its statement are closed. If there is always free memory in shared pool, increasing its size will have little or no effect. If it is always full, this does not mean that there is a problem. The main point to consider is the hit ratio.

- Determine the amount of memory and the get hit ratio in the shared pool, using the following queries:

```
SQL> select * from v$sgastat where name = 'free memory';
```

```
SQL> select gethitratio from v$librarycache  
2 where namespace = 'SQL AREA';
```

Statistic	Pre-Tuning Setting	Post-Tuning Setting
free memory		
gethitratio		

- Review the following statistics in the `report.txt` output.

Pinhitratio	Cluster	Body	Index	Object	Pipe	SQL/ Area	Table Procedure	Trigger
Pre-Tuning								
Post-Tuning								



Reloads/ Pins	Cluster	Body	Index	Object	Pipe	SQL/ Area	Table Procedure	Trigger
Pre-Tuning								
Post-Tuning								

- Pin the packages, procedures, and triggers that are frequently required and need to be kept in memory, using the `DBMS_SHARED_POOL.KEEP` supplied procedure to make sure that they are never aged out of the shared pool. This prevents a large number of small objects from being aged out of the shared pool to make room.
- Review the following `init.ora` parameters. Insert additional parameters that you think are appropriate.

Parameter	Pre-Tuning Setting	Post-Tuning Setting
<code>CURSOR_SPACE_FOR_TIME</code>		
<code>SHARED_POOL_SIZE</code>		
<code>OPEN_CURSORS</code>		
<code>SHARED_POOL_RESERVED_SIZE</code>		
<code>SESSION_CACHED_CURSORS</code>		

**Step 2: Data Dictionary Statistics**

The data dictionary data is maintained in a separate cache called the *dictionary cache*, which is stored in shared SQL area. This is accessed for each SQL statement when it is parsed, and also at run time for dynamic storage allocation, and so on. Cache hits avoid the necessity for recursive calls, and performance on SQL statements improves. If the ratio of the sum of GET\_MISS to the sum of GET\_REQ is greater than 15%, the `init.ora` parameter `SHARED_POOL_SIZE` should be increased.

Using the `report.txt`, sample six dictionary statistics that you think may indicate a tuning problem.

Dictionary Statistic	Tuning Phase	GET_MISS	GET_REQ	GET_MISS/ GET_REQ
Sum	Pre			
Sum	Post			

### Step 3: Rollback Segments

The amount of rollback generated by a transaction depends on the nature of the operation and the number of data block changes.

- A transaction that is writing rollback data must first access the transaction table stored in the rollback segment header and acquire a slot in that table. This requires momentary latching of the table to serialize the concurrent update to it. If the database is update-intensive and has a small number of rollback segments, user transactions will wait on the latch to access the transaction table.
- A large number of rollback segments on a query-intensive database will result in waste of space.

The following statistics give interesting information about the rollback segments:

- Using the `report.txt` output, if the ratio of `TRANS_TBL_WAITS` to `TRANS_TBL_GETS` is greater than 1%, additional rollback segments should be added to the database.
- In general, rollback segments should be the same size, and should be created with a large number of small extents.
- It is very important to set the `OPTIMAL` value of the rollback segments so that it does not `SHRINK` too often. Query `V$ROLLSTAT` to find the `OPTIMAL` parameter setting for each rollback segment.

## UNDO SEGMENT

Statistic	Tuning	0	1	2	3	4	5	6	7
TRANS_TBL_GETS	Pre								
	Post								
TRANS_TBL_WAITS	Pre								
	Post								
TRANS_TBL_WAITS/ TRANS_TBL_GETS	Pre								
	Post								
SHRINKS	Pre								
	Post								
WRAPS	Pre								
	Post								
OPTIMAL	Pre								
	Post								

Review the following `init.ora` parameters. Insert additional parameters that you think are appropriate.

Parameter	Pre-Tuning Setting	Post-Tuning Setting
ROLLBACK_SEGMENTS		
TRANSACTIONS		
TRANSACTIONS_PER_ ROLLBACK_SEGMENT		

## Step 4: Database-Wide Statistics

**Buffer Busy Waits** This indicates the instances in which a user process attempted to acquire a buffer and had to wait because it was already held in an incompatible mode. A nonzero value indicates block contention. Block contention waits should be compared with the logical reads to get the relative level of contention (blocks read in read-consistent mode: consistent gets). If this ratio is 1%–5%, contention can be analyzed further by block type, such as data block, segment header, undo block, undo header, and so on, by querying on the V\$WAITSTAT view.

- If total waits is high for undo block and undo header classes, additional rollback segments should be created to reduce rollback segment buffer contention.
- If the total waits is high for the data block and segment header classes, additional free lists should be created to reduce data block contention. To add free lists, re-create the tables and indexes with a higher FREELISTS storage parameter. It is not recommended that this parameter be increased higher than is necessary to avoid contention.

**Cache Hit Ratio** Hit ratio gives an indication of how often the various processes that are accessing the data buffers find the blocks in cache. This is calculated as follows:

$$\text{Hit Ratio} = (1 - (\text{Physical Reads} / \text{Logical Reads})) \times 100$$

**where:** *Logical reads* = DB block gets + consistent gets

*Physical reads* = Block requests that resulted in disk reads

The target for a hit ratio is greater than 90%. If this value is less than 70%, increase the value of DB\_BLOCK\_BUFFERS.

## Multiple Buffer Pools

- Next, you can identify segments for the KEEP and RECYCLE buffer pools that enable you to appropriately set two parameters, BUFFER\_POOL\_KEEP and BUFFER\_POOL\_RECYCLE, and re-create these segments with the appropriate storage clause, storage (BUFFER\_POOL KEEP) or (BUFFER\_POOL RECYCLE).
- The size of each buffer pool is subtracted from the total number of buffers defined for the entire buffer cache.
- The V\$BUFFER\_POOL\_STATISTICS view displays statistics (physical writes, consistent gets, FREE\_BUFFER\_WAITS) against the multiple buffer caches.

## Step 4: Database-Wide Statistics (continued)

**Cluster Key Scans and Block Gets** These are the number of cluster blocks accessed. Cluster key scans is the number of scans processed on cluster blocks. If the ratio of cluster key scan block gets to cluster key scans is greater than 1, the rows for one cluster key are stored in multiple data blocks and the cluster should be analyzed for row chaining.

### Example

If cluster key scan block gets is 8414 and cluster key scans is 3223, the ratio is:

$$(8414/3223) = 2.61$$

which indicates that on an average, three Oracle blocks are accessed for each cluster key.

The SIZE parameter specified during the CREATE CLUSTER command determines the number of cluster keys per block; the default is one. If this parameter is not specified correctly, rows for one cluster key may not fit into one data block or there may be wasted space in the data block. If all of the data for one cluster key does not fit in one block, additional disk access must occur to access the data.

**Migrated and Chained Rows** Migrated and chained rows in a table or a cluster could also be found by using the ANALYZE command with the COMPUTE STATISTICS and LIST CHAINED ROWS option. It may not be possible to avoid chaining in all cases (for example, having LONG columns in the tables).

**Cumulative Opened Cursors** This parameter indicates the total number of cursors opened during the script execution interval. A cursor is opened for each SQL statement that is parsed into a context area. To improve performance, cursors should be reused and reparsing should be avoided.

If a cursor will not be reused, it is best to close the cursor when the SQL statement completes, because this will free up all the resources used by the cursor.

A per-transaction average of cursors opened is more relevant than the total number. This value should be analyzed in the context of the composition of the database workload during the interval. Generally, this value should not exceed five to seven cursors per transaction for online processing intervals. For batch processing, it can be higher than ten depending on the processing being done. If the values are higher than this, the application should be analyzed for optimal cursor usage.

**Step 4: Database-Wide Statistics (continued)**

Review the following statistics in the `report.txt` output.

Statistic	buffer_ busy_waits	cache_ hit_ratio	cumulative_ _opened _cursors	cumulative_ opened _cursors/ transactions	cluster_ key_scans	cluster_ key_scan_ block_gets
Pre-Tuning						
Post-Tuning						

Review the following `init.ora` parameters. Insert additional parameters that you think are appropriate.

Parameter	Pre-Tuning Setting	Post-Tuning Setting
DB_BLOCK_BUFFERS		
DB_BLOCK_SIZE		
DB_BLOCK_LRU_LATCHES		

## Step 5: Redo Log Writer Performance

### Log Buffer Space Requests

This value reflects the number of times a user process waits for space in the redo log buffer. Ideally, this value should be 0; if it is not zero, increase the value of the initialization parameter LOG\_BUFFER in 5% increments.

Check V\$SYSTEM\_EVENT to see if there have been waits for the event Log Buffer Space requests:

```
select * from v$system_event where event = 'log buffer space';
```

Statistic	Pre-Tuning Setting	Post-Tuning Setting
Log Buffer Space		

**Redo Buffer Allocation Retries** This value indicates the number of repeated attempts to allocate space in the redo buffer. A value indicates that the redo writer is falling behind, possibly because of a log switch.

**Redo Entries/Redo Size** This statistic is useful in sizing the redo log files and planning the checkpointing frequency. Log switches also post the archiver, and the file copy is done to archive the log file. Redo size indicates the number of bytes of redo generated, and redo entries is the number of redo records created.

These statistics should be analyzed for the given time interval. For example:

Redo size: N bytes

Time interval:  $t_{end} - t_{start} = t$  (minutes, for example)

Redo generation per minute =  $(N/t)$

Log switch and desired interval can be determined by considering the impact of archiver file copy and checkpoints on the system. Log switches force checkpoints to occur. If archiving is enabled, the archiver performs a copy of the file. These two operations involve activity for the background processes, and can affect the database response if the sizing is not done properly. Large redo log files reduce the number of log switches but increase the archiver copy time, degrading the user response on single-CPU systems. Frequent checkpointing may cause the users to wait on buffers pinned by the DBWn.

A good time interval for log switch is when the CPU time used and the I/Os on the devices involved are not large enough for users to be blocked. For example, on a busy VAX 6420, a file copy of 30,000 VMS blocks takes 30 seconds and several thousand I/Os on a system with 30 users. Once the time interval is known (for example, in minutes), the log file size can be calculated by simple multiplication with the redo generated per minute.



### Step 5: Redo Log Writer Performance (continued)

**Redo Writes** This value indicates total number of redo writes to the redo buffer. It may be useful to determine whether or not it is too large compared with redo entries statistics.

- Check V\$LATCH. The IMMEDIATE\_GETS and IMMEDIATE\_MISSES columns give information about IMMEDIATE requests. If the ratio of IMMEDIATE\_MISSES to the sum of IMMEDIATE\_GETS plus IMMEDIATE\_MISSES is more than 1% of the redo copy latches, you may need more redo copy latches.

	IMMEDIATE_GETS	IMMEDIATE_MISSES	IMMEDIATE_MISSES/ IMMEDIATE_GETS
Pre-Tuning			
Post-Tuning			

- Check V\$LATCH. The GETS, MISSES, and SLEEPS columns give information about WILLING\_TO\_WAIT requests. SLEEPS shows the number of times the process waited. If the ratio of MISSES to GETS is more than 1% for the redo allocation latch, you should try to reduce contention for it.

	GETS	MISSES	SLEEPS	MISSES/GETS
Pre-Tuning				
Post-Tuning				

- Review the following `init.ora` parameters. Insert additional parameters that you think are appropriate.

Parameter	Pre-Tuning Setting	Post-Tuning Setting
DB_BLOCK_CHECKSUM		
LOG_BUFFER		

## Step 6: Minimize I/O

- Check the high-water mark for tables to ensure it is not wastefully high. Full table scan time is degraded by a wastefully high high-water mark and a low block fill factor. Although chaining must be avoided, it is still essential to maintain a reasonably high blocking factor, minimizing the need for costly disk I/O.

You should truncate and reload a table with a high-water mark, or issue the command:

```
ALTER TABLE <table_name> deallocate unused;
```

Finding the high-water mark is probably the most difficult chore in analyzing a table. A decent estimate can be obtained by selecting a count of the unique FILEID/BLOCKID combinations for the table. Both the FILEID and BLOCKID are components of the ROWID pseudo-column. A precise answer is obtained by counting consistent mode reads in a SQL trace analysis of a full table scan.

- PCTUSED and PCTFREE determine the blocking factor for the table. PCTUSED, PCTFREE, and storage clauses should be considered mandatory for all segment creation. Even estimated starting values are likely to be better than the default values.
- File I/O should be spread evenly across multiple disk drives. Further analysis needs to be done to determine which tables and indexes are used most in the accessed files. A high number of physical blocks reads to physical reads indicates a high number of full table scans.

**Step 6: Minimize I/O (continued)**

Review the following statistics in the `report.txt` output.

Tablespace Name	Tuning	PHYS_READS	PHYS_BKLS_RD
SYSTEM	Pre		
	Post		
USER_DATA	Pre		
	Post		
TEMP	Pre		
	Post		
RBS	Pre		
	Post		
	Pre		
	Post		
	Pre		
	Post		
	Pre		
	Post		
	Pre		
	Post		
	Pre		
	Post		

**Distribute File I/O**

- Locate redo logs on disks that do not contain database files, because the data file I/O is scattered, whereas redo log files are always written sequentially.
- Tables should be located on different disks than their associated indexes, because the two can be accessed concurrently in a multiuser environment.
- Stripe large tables and indexes across several disks if the nature of access is highly distributed. Active database files should not be located at opposite ends of the disk, and the most active database files should be located on the highest throughput disks.
- Place rollback segments in a separate tablespace and spread across disks on different files. Take care to ensure that the creation order of rollback segments is consecutively numbered segments on different disks.
- The `init.ora` parameter `DB_FILE_MULTIBLOCK_READ_COUNT` can be set to increase the number of blocks read during a single read. Increasing this parameter reduces I/O when full table scans are being performed.

**Step 6: Minimize I/O (continued)**

- Review the following statistics in the `report.txt` output (write in the file name and place an X in the column for the disk on which it is located).

File Name	Tuning	DISK 1	DISK 2	DISK 3	DISK 4
	Pre				
	Post				
	Pre				
	Post				
	Pre				
	Post				
	Pre				
	Post				
	Pre				
	Post				
	Pre				
	Post				
	Pre				
	Post				
	Pre				
	Post				
	Pre				
	Post				
	Pre				
	Post				
	Pre				
	Post				
	Pre				
	Post				

**Step 6: Minimize I/O (continued)**

Review the following `init.ora` parameters. Insert additional parameters that you think are appropriate.

Parameter	Pre-Tuning Setting	Post-Tuning Setting
DB_FILE_MULTIBLOCK_READ_COUNT		
DB_WRITER_PROCESSES		

## Step 7: DBWn Performance

### DBWn Checkpoints

- This is the number of checkpoint messages that were sent to DBWn. This does not mean, however, that this number of checkpoints was actually performed, because if the second checkpoint is issued while the first is active, the first stops and the second starts. During checkpoint processing, the log writer gives the DBWn a list of modified blocks that are to be written to disk. This causes a slight degradation in performance as buffers are pinned and physical I/O occurs. The performance of normal database operations can be improved if checkpointing frequency is reduced, although this may increase the crash recovery time.
- To reduce the number of checkpoints, increase the `init.ora` parameter `LOG_CHECKPOINT_INTERVAL`. If this parameter is set to a size (bytes) larger than the size of the redo log, a checkpoint is performed during each log switch. To increase the number of checkpoints and decrease database recovery time, use both the `init.ora` parameters `FAST_START_IO_TARGET` and `LOG_CHECKPOINT_INTERVAL`.  
`LOG_CHECKPOINT_TIMEOUT` can also be used to reduce or increase the number of checkpoints. A value of 0 disables the time-based checkpoint.

**Free Buffers Inspected** The number of buffers skipped in the buffer cache by user processes in order to find a free buffer. If this value is high compared to the free buffer scans, it means that the buffer cache has too many modified blocks and the cache size should be increased.

**Enqueue Timeouts** These indicate the number of times that an enqueue lock was requested and was not immediately granted. If this parameter is greater than zero, increase the `init.ora` parameter `ENQUEUE_RESOURCES` to reduce waits.

**Recursive Calls** These occur because of cache misses and dynamic storage extension. If the dictionary data is found in cache, a recursive call is not made and the data is read from cache directly. In general, if recursive calls are greater than four per process, the data dictionary cache should be optimized and segments should be rebuilt with storage clauses to have a few large extents. Segments include tables, indexes, and rollback segments.

Recursive calls should be fewer than user calls (less than one-tenth). Where there is an imbalance, the aim should be to reduce parsing. High levels of recursive SQL may also be attributable to significant use of PL/SQL. For each SQL statement in a PL/SQL block, on each iteration, there are recursive calls to do the equivalent of bind and define.

## Step 8: Table Statistics

**Table Scans (Long Tables)** This is the total number of full table scans performed on tables with more than five DB\_BLOCKS. If the number of full table scans is greater than 0 per transaction, the SQL statements in the application typically needs tuning. Index use by statements should be based on selectivity of the keys, clustering of rows by the key values, the amount of data retrieved, and several other considerations.

**Table Scans (Short Tables)** This is the number of full table scans performed on tables with less than five DB\_BLOCKS. It is optimal to perform full table scans on short tables rather than use indexes. Table Scans (Long Tables) plus Table Scans (Short Tables) is equal to the number of full table scans. It is not recommended to build indexes on small tables.

**Table Scan Blocks Gotten and Table Scan Rows Gotten** These are respectively, the number of blocks and rows scanned during all full table scans.

- Determine on average the number of rows gotten per block for all full table scans:

`Table Scan Rows Gotten / Table Scan Blocks Gotten`

- Determine the approximate number of rows gotten for short and long table scans:

`Table Scans (short) * 4 blocks = Blocks Scanned (short)`

`Table Scan Blocks Gotten - Blocks Scanned (short) = Blocks Scanned (long)`

**Table Fetch by ROWID** This denotes the rows that were accessed using an index and rows that were accessed using the statement where ROWID = xxx.xxxxxx.xxx.xxxxxx. ROWID is the fastest path to a row and should be used whenever applicable. If Table Fetch by ROWID is low compared with the total number of rows retrieved, SQL statements should be tuned to use indexes on long tables.

**Table Fetch by Continued Row** This is the number of rows that are continued or chained to another block. If this number is high, additional I/O must be performed in order to read the entire row. Row chaining cannot be avoided for tables with long columns. The ratio of Table Fetch by Continued Row to Table Fetch by ROWID is a good indication of the number of chained rows.

The ANALYZE command provides the number of chained rows in a table.

## Step 8: Table Statistics (continued)

**User Calls and Parse Statistics** User Calls is the number of times a call is made to the kernel. Parse Count indicates the number of times a SQL statement was parsed. The number of calls to the kernel should be reduced if possible. Calculate the number of calls to the kernel per parse, using the following calculation:

$$\text{Parse Count} / \text{User Calls} = \text{Avg. calls per parse}$$

- 3** Review the following statistics in the `report.txt` output. Insert additional parameters that you think are appropriate.

Statistic	Pre-Tuning Value	Post-Tuning Value
Enqueue Timeouts		
Recursive Calls		
User Calls		
Parse Count		
Parse count/user calls = avg.		
Sorts(Disk)/Sorts(Memory)		
Table Scans (Long Tables)		
Table Scans (Short Tables)		
Table Scan Rows Gotten		
Table Scan Blocks Gotten		
Table Scan Rows Gotten/Table Scan Blocks Gotten		
Table Scans (Short) $\times$ 4 blocks = Blocks Scanned (Short)		
Table Scan Blocks Gotten – Blocks Scanned (Short) = Blocks Scanned (Long)		
Table Fetch By ROWID		
Table Fetch by Continued Row		



**Step 8: Table Statistics (continued)**

Review the following `init.ora` parameters. Insert additional parameters that you think are appropriate.

Parameter	Pre-Tuning Setting	Post-Tuning Setting
LOG_CHECKPOINT_INTERVAL		
LOG_CHECKPOINT_TIMEOUT		
SORT_AREA_SIZE		
SORT_AREA_RETAINED_SIZE		

## Step 9: Sorts

**Sorts (Disk)** These are sorts that require creation of temporary segments on disk to store the intermediate sort results. This occurs if the data being sorted cannot be fit into memory block specified by the `SORT_AREA_SIZE` in the `init.ora` file.

**Sorts (Memory)** These are sorts that are completed in memory. If the ratio of disk sorts to memory sorts is greater than 5%, increase the `SORT_AREA_SIZE`, depending on memory availability.

**Temporary Tablespaces** Because sorts are done in memory if they are smaller than `SORT_AREA_SIZE`, you should consider this value when setting extent sizes:

- Choose `INITIAL` and `NEXT` values as integer multiples of `SORT_AREA_SIZE`, allowing an extra block for the segment header.
- Set `PCTINCREASE` to 0.
- The `V$SORT_SEGMENT` view contains information about every sort segment of the `TEMPORARY` tablespaces in the instance.

Remember that the success of your tuning strategy also depends on your knowledge of the Oracle database architecture, `V$` dynamic performance views, Oracle tools, and so on. List two additional areas that you investigated using these tools. Refer to the technical manuals and three-day course notes as guidelines.

---

B

---

**Practice Hints**

## Practice 3-1 Hint

The goal of this practice is to familiarize yourself with the machine setup and to have a look at the diagnostic tools, such as trace files, waiting event views, and OEM event management.

- 1 Log on as directed by your instructor.
- 2 View your parameter file to check that the appropriate parameters locate the alert log file and user trace files, respectively, in the BDUMP and UDUMP directories in your home directory. If necessary, shut down, then start up, your instance.

**Hint:**

- BACKGROUND\_DUMP\_DEST is the one for the alert log file destination.
- USER\_DUMP\_DEST is the one for the user trace file destination.

- 3 Examine your alert.log file to find any internal errors (ORA-600).

**Hint:** Look in the \$HOME/BDUMP directory.

- 4 As SYSTEM, connect to SQL\*Plus and issue a command that will create a user trace file after querying against the DBA\_TABLES data dictionary view. Do not try to interpret the content of the trace file. You will learn how to do this in Lesson 13, “SQL Issues and Tuning Considerations for Different Applications.”

**Hint:** Find the command to set SQL\_TRACE to TRUE;

- 5 Examine the resulting trace file without trying to interpret its content. You will learn how to do this in Lesson 13, “SQL Issues and Tuning Considerations for Different Applications.”

**Hint:** Look in the \$HOME/UDUMP directory.

## Practice 4-1 Hint

The objective of this practice is to familiarize yourself with different ways of retrieving statistics information: run the `utlbstat.sql` and `utlestat.sql` scripts, examine the resulting `report.txt` file, display the major performance dynamic views appropriate for tuning, and learn how to launch one of the Diagnostics Pack applications.

During the labs, use the `$HOME/LABS/init.ora` parameter file to start up the database.

- 1 Before running the `utlbstat.sql` script, verify that the required `init.ora` parameter has been set to true.

**Hint:** The appropriate parameter is `TIMED_STATISTICS`.

- 2 Run the `utlbstat.sql` script connected to SQL\*Plus as SYSDBA.

**Hint:** The script `$ORACLE_HOME/rdbms/admin/utlbstat` must be run under SYS.

- 3 Start the following three sessions as the user SCOTT:

- The first session executes the `lab04_1.sql` script (it creates and loads data into SCOTT's tables). Wait until the script has finished.
- The second session then executes the `lab04_2.sql` script.
- At the same time, the third session executes `lab04_3.sql` script.

Do not wait until the last two scripts have finished loading data. Go to step 4.

- 4 Query the appropriate dynamic view to see:

- The library cache performance

**Hint:** Query the `V$LIBRARYCACHE` view.

- Among the system statistics, those concerning sort activity

**Hint:** Query the `V$SYSSTAT` view.

- The sessions consuming more than 10,000 bytes of PGA memory

**Hint:** Query the `V$SESSTAT`, `V$SESSION`, and `V$STATNAME` views.

- The rollback segments statistics

**Hint:** Query the `V$ROLLSTAT` view.

- The file I/O statistics

**Hint:** Query the `V$DATAFILE`, `V$FILESTAT` view.

- 5** Run the `utlestat.sql` script so that the resulting report is stored in your `UDUMP` directory.

**Hint:** Before running `SQL*Plus`, go to the `UDUMP` directory.

- 6** Examine the resulting `report.txt` file and compare what you displayed in: Library cache dynamic view with the library cache section of the report

**Hint:**

```
$more report.txt (read 1st section)
```

File I/O dynamic view with the I/O statistics section of the report.

**Hint**

```
$more report.txt (read the last section)
```

- 7** Display the list of Wait event names.

**Hint:** Display the `V$SEVENT_NAME` view.

- 8** Are there any sessions actually waiting for resources?

**Hint:** Be careful to set the `init.ora` parameter `TIMED_STATISTICS` to true and display the `V$SESSION_WAIT` view.

---

## Practice 5-1 Hint

The objective of this practice is to use available diagnostics tools to monitor and tune the shared pool.

- 1 Check the size of the shared pool.

**Hint:** Look at the `SHARED_POOL_SIZE` parameter in the `init.ora` parameter file.

- 2 From SQL\*Plus, connected as SYSDBA, run the `utlbstat.sql` script to start collecting statistics.
- 3 To simulate user activity against the database, connect as SCOTT and run the `lab05_1.sql` script and in another session `lab05_2.sql`.
- 4 Measure the pin hit ratio for the library cache. Determine if it is a good ratio or not.

**Hint:** You can choose between dynamic views or Oracle Diagnostics Pack:

- Query the `V$LIBRARYCACHE` view.
- Launch the Performance Manager (Library Cache Hit% chart).

- 5 Measure the hit ratio for the data dictionary cache. Determine if it is a good ratio or not.

**Hint:** You can choose between dynamic views or Oracle Diagnostics Pack:

- Query the `V$ROWCACHE` view.
- Launch the Performance Manager (Data Dictionary Cache Hit% chart).

- 6 When both scripts have finished, from SQL\*Plus, and connected as SYSDBA, run the `utlestat.sql` script to collect ending statistics.

- **Hint:** You will detect values different from those in the `V$LIBRARYCACHE` view and the values in the report section, because the last one covers a period of time, whereas `V$LIBRARYCACHE` displays the current values of the time.

- 7 Use the `report.txt` output to get both hit ratios.

**Hint:** On one hand, figures show ratios at a point in time; on the other hand, they display ratios for a period of activity.

- 8 Which solutions would you apply if any of the hit ratios were bad?

**Hint:** Change the `SHARED_POOL_SIZE` parameter.

- 9 Determine which packages, procedures, and triggers are pinned in the shared pool.

**Hint:** Query the `V$DB_OBJECT_CACHE` view.

- 10 Pin one of the Oracle supplied packages that needs to be kept in memory.

**Hint:** Pin the `STANDARD` package using the `DBMS_SHARED_POOL` package.

- 11 Determine the amount of session memory used in the shared pool for your session.

**Hint:** Query the “session uga memory” statistics from `V$MYSTAT` and `V$STATNAME` views.

- 12 Determine the amount of session memory used in the shared pool for all sessions.

**Hint:** You can choose between dynamic views or Oracle Diagnostics Pack:

- Query the “session uga memory” statistics from `V$SESSTAT` and `V$STATNAME` views.

## Practice 6-1 Hint

The objective of this practice is to use available diagnostics tools to monitor and tune the database buffer cache.

- 1** From SQL\*Plus, connected as SYSDBA and run the `utlbstat.sql` script to start collecting statistics.
- 2** To simulate user activity against the database, connect as SCOTT and run the `lab06_1.sql` script.
- 3** Connect as SYSTEM and measure the hit ratio for the database buffer cache. Determine if it is a good ratio or not.  
**Hint:** You can choose between dynamic views or the Oracle Diagnostics Pack.
  - To query the V\$SYSSTAT view.
- 4** When the script has finished, from SQL\*Plus, and connected as SYSDBA, run the `utlestat.sql` script to collect ending statistics.
- 5** Use the `report.txt` output to get the cache hit ratio. What is the ratio?  
**Hint:** Use statistics from STATS\$STATS to compute the hit ratio.  
If the ratio is bad, add new buffers, run steps 2 to 5, and examine the new ratio to verify that the ratio has improved. If the ratio is good, remove buffers, run steps 2 to 5, and verify if the ratio is still good.
- 6** Size the keep buffer pool to hold the SCOTT.S\_EMP and SCOTT.S\_DEPT tables.  
**Hint:** Analyze both tables.  
Query the DBA\_TABLES.BLOCKS to get their size in blocks.
- 7** Modify the `init.ora` file to set up the keep buffer pool.  
**Hint:** Edit DB\_BLOCK\_BUFFERS, BUFFER\_POOL\_KEEP and DB\_BLOCK\_LRU\_LATCHES parameters.
- 8** Enable the SCOTT.S\_EMP and SCOTT.S\_DEPT tables for caching in the keep pool.  
**Hint:** Execute ALTER TABLE ... STORAGE commands against both tables.
- 9** Set up both of these tables so that their blocks are kept in memory, even during full table scans.  
**Hint:** Execute ALTER TABLE ... CACHE commands against both tables.
- 10** Determine how many LRU latches have been allocated for your instance.  
**Hint:** Use the V\$BUFFER\_POOL view.
- 11** Connect as SYSTEM and display the hit percentage for the LRU latches.  
**Hint:** Use the V\$LATCH view.
- 12** Determine what the overall hit percentage is for free lists in your instance.  
**Hint:** Use the V\$WAITSTAT and V\$SYSTEM\_EVENT views.



## Practice 7-1 Hint

The objective of this practice is to use available diagnostics tools to monitor and tune the redo log buffer.

- 1 Connect as **SYSTEM** and measure the redo buffer allocation retries for the redo entries. Determine if it is a good ratio or not.

**Hint:** Query the V\$SYSSTAT view to get the two required statistics.

- 2 Which solution would you apply if the ratio were bad?

**Hint:** Modify the value of the parameter LOG\_BUFFER.

- 3 Change the size of the appropriate parameter.

## Practice 8-1 Hint

- 1 Have a look at your database. Find out if there are deficiencies concerning the database configuration. If necessary, change the database configuration. Consider the following:

- a How many tablespaces do you have? Add more tablespaces, if necessary, spreading them across the subdirectories that represent devices.

**Hint:** Query `DBA_TABLESPACES` or `V$TABLESPACE`, `DBA_DATA_FILES`, or use Performance Manager application.

- b Are there any non-data-dictionary objects in the `SYSTEM` tablespace? Objects can be moved by creating one table as a copy of another, dropping the original object, and then renaming the new object to have the original name.

**Hint:** Query `DBA_SEGMENTS` or use the Performance Manager application.

- c Check to see if you have the data files on different disks from your online redo log files. The archive logs should be on different disk from both the data files and redo log files.

**Hint:** Query `V$LOGFILE` and `V$DATAFILE` or use the Performance Manager application.

`LOG_ARCHIVE_DEST` and `LOG_ARCHIVE_DUPLEX_DEST` specify the archive destinations.

- d Are the online redo log files mirrored?

**Hint:** Query `V$LOG` and `V$LOGFILE` or use the EM Backup Manager application.

- 2 Find out how often checkpoints are occurring. Is this timing optimal? If not, correct it.

**Hint:** Investigate the alert file. If necessary change the values of the initialization parameters `LOG_CHECKPOINT_INTERVAL` and/or `LOG_CHECKPOINT_TIMEOUT` and/or `FAST_START_IO_TARGET`.

- 3 To simulate user activity against the database, connect as `SCOTT` and run the `lab08_1.sql` script. Monitor the I/O statistics.

**Hint:** Query `V$FILESTAT` or use the Performance Manager.

## Practice 9-1 Hint

The objective of this practice is to use available diagnostics tools to monitor and tune block space usage.

- 1 Connect as **SYSTEM** and find the **PCTFREE**, **CHAIN\_CNT** and high-water mark for **SCOTT's S\_ORD** table.

**Hint:** Analyze the table and select the statistics from **DBA\_TABLES**.

- 2 Run the `lab09_1.sql` script to update rows in **SCOTT.S\_ORD**
- 3 Find the **CHAIN\_CNT** and high-water mark for **SCOTT's S\_ORD** table.
- 4 Eliminate any migrated rows.

**Hint:**

- Run `utlchain.sql`
- Run `ANALYZE TABLE ... LIST CHAINED ROWS`.
- Copy, delete, and reinsert the migrated rows.

- 5 Find the **CHAIN\_CNT** and high water mark for **SCOTT's S\_ORD** table.
- 6 Remove the rows from table **CHAINED\_ROWS** without deleting them.

**Hint:** Use the **TRUNCATE** command.

## Optional Practice

Find the high-water mark for **SCOTT's S\_ORD** table without querying the data dictionary. You can use the `lab09_2.sql` script.

**Hint:** Use the **DBMS\_SPACE** package and **UNUSED\_SPACE** procedure.

## Practice 10-1 Hint

The objective of this practice is to use available diagnostics tools to monitor and tune sorts.

**1** Connect as **SYSTEM** and execute the `lab10_1.sql` script. It forces sorts to write to disk.

**2** Measure the sorts (disk) to the sorts (memory) ratio.

**Hint:** Query the **V\$SYSSTAT** view or the Performance Manager application.

**3** Configure the **SORT\_AREA\_SIZE** parameter to result in sorts (memory) only.

**Hint:** Increase the value of **SORT\_AREA\_SIZE**.

**4** Connect as **SCOTT** and execute the `lab10_2.sql` script and recalculate the ratio.

**5** Create a new **TEMPORARY** tablespace; when **SCOTT** runs sort operations, the sorts to disk occur on this tablespace.

**Hint:** Use the **TEMPORARY** clause in the **CREATE TABLESPACE** command.

Assign this new tablespace to the user **SCOTT** through the **ALTER USER** command.

**6** Connect as **SCOTT** and reexecute the `lab09_2.sql` script. Monitor the **TEMPORARY** tablespace.

**Hint:** Query the **V\$SORT\_SEGMENT** view before and after the execution of `lab10_2.sql`.

## Practice 11-1 Hint

The objective of this practice is to use available diagnostics tools to monitor and tune the rollback segments. You will use the `report.txt` output, so run `utlbstat.sql` now.

- 1 Run the `utlbstat.sql` script.

**Hint:** The script `$ORACLE_HOME/rdbms/admin/utlbstat` must be run under SYS.

If you have not already done so, create a new rollback segment tablespace, at least 2 MB in size. Connect as SYSTEM.

- 2 For the purposes of this practice, create a new rollback segment called RBSX. Give it an overall size of 200 KB, divided into 20 extents. It should shrink back to 200 KB if expanded.

**Hint:** For the storage parameters, use 10 KB for the INITIAL and NEXT extent sizes with MINEXTENTS value set to 20. Set the OPTIMAL value so that the segment shrinks back to 200 KB automatically.

- 3 Rollback segment created. Bring your new rollback segment online and take any others (except the SYSTEM rollback segment) offline.

**Hint:** Query the DBA\_ROLLBACK\_SEGS view to get the names of the rollback segments to be taken offline by the ALTER ROLLBACK SEGMENT command.

- 4 Before executing a new transaction, find the number of bytes written in the new rollback segment.

**Hint:** In the V\$ROLLSTAT view, query the WRITES column to get the number of bytes written in the RBSX rollback segment until now.

- 5 As SCOTT, run the `ins_ords.sql` script. The script inserts 100 new rows into the S\_ORD table. Without committing, log in again as SYS (either use the HOST command or open another window).

How many rollback segment blocks or bytes is the transaction using?

**Hint:**

- Join the V\$TRANSACTION and V\$SESSION views to find, in the USED\_UBLK column, for your session, how many blocks the current transaction is using.
- Query in the V\$ROLLSTAT view the WRITES column to get the number of bytes written in the RBSX rollback segment since question 3.

- 6 Return to the SCOTT session and commit the insert. Run the `del_ords` script, without committing. This script deletes the hundred rows you have just inserted. As SYS, check the amount of rollback space used, as in step 5. What is the

difference between the space used for the original insert and the space used for the delete?

**Hint:** Because the ROWID is the only information kept for INSERTs, it requires much less space than the DELETES that copy the image of the rows deleted.

- 7 As SYS, find out if you have had any rollback segment contention since startup, using two different sources of information.

**Hint:**

- In the V\$ROLLSTAT view, query the WAITS and GETS column values to compute the contention ratio.
- A better source of information is the `report.txt` output.
- Use Performance Manager Rollback Nowait Hit%.
- Query the V\$WAITSTAT view for the following classes: “undo header,” “undo block,” “system undo header,” and “system undo block.”

- 8 Does the V\$SYSTEM\_EVENT view show any waits related to rollback segments?

**Hint:** Query in V\$SYSTEM\_EVENT view for the “undo segment tx slot” entry.

- 9 The tested transaction will be executed simultaneously by an average of ten users during the day. Every night, a batch job will load ten times the same volume as the `ins_ords.sql` script. Prepare the script to create the appropriate rollback segments with appropriate storage parameters to optimize space usage.

*Do not run the script.*

**Hint:**

- Create three rollback segments for the daytime activity and another one for batch activity.
- If there is too much extension, recreate the rollback RBSX with larger INITIAL and NEXT values.
- Increase the OPTIMAL value if too many shrinks occurred.
- Resize all rollback segments so that they are equally sized, except for the one dedicated for large batch transactions.
- Alter the batch execution script by inserting the command that forces the batch transaction to use the appropriate rollback segment.

- 10 As SCOTT, run the `lab11_1.sql` script again, allocating the transaction to a specific rollback segment, and check that the transaction is working with the defined rollback segment.

**Hint:** Before running the `lab10_1.sql` script, execute the command `SET TRANSACTION USE ROLLBACK SEGMENT`.

Then join the V\$ROLLSTAT, V\$SESSION, and V\$TRANSACTION views to check that the transaction is using the defined rollback segment.

## Practice 12-1 Hint

The objective of this practice is to use available diagnostics tools to monitor lock contention.

You will need to start three sessions, in separate windows. Log in twice as SCOTT (sessions 1 and 2) and once as SYS (session 3).

- 1 In session 1, update the salary of one of the employees in the S\_EMP table:

```
SQL> UPDATE s_emp SET salary=salary*1.1
      2 WHERE id=1;
```

- 2 In Session 3, check to see if any locks are being held.

**Hint:** Use the V\$LOCK or V\$LOCKED\_OBJECT view.

- 3 In session 2, drop the S\_EMP table. Does it work?

- 4 In session 2, update a different row in the S\_EMP table.

```
SQL> UPDATE s_emp SET salary=salary*1.1
      2 WHERE id=2;
```

In Session 3, check to see what kind of locks are being held.

**Hint:**

– Use the V\$LOCK or V\$LOCKED\_OBJECT view.

- 5 Roll back the changes you made in session 2, and try to update the same row as session 1.

```
SQL> UPDATE s_emp SET salary=salary*1.1
      2 WHERE id=1;
```

Check to see that the locks held in session 3, using any tool.

**Hint:** Use V\$LOCK view.

- 6 In session 3, kill the session that is holding the lock.

**Hint:** Use the ALTER SYSTEM KILL SESSION command.

## Practice 13-1 Hint

The objective of this practice is to familiarize you with SQL statement execution plans and to interpret the formatted output of a trace file generated using SQL Trace and the formatted output generated by TKPROF. You will also learn how to manipulate statistics using the Oracle supplied DBMS\_STATS package.

- 1 Ensure the current directory is \$HOME/LABS.
- 2 Connect as SCOTT and create the PLAN\_TABLE under the SCOTT schema.  
**Hint:** Use the \$ORACLE\_HOME/rdbms/admin/utlxplan.sql script.
- 3 Describe PLAN\_TABLE.
- 4 Use SQL\*Plus AUTOTRACE to look at the execution plan worked out by the optimizer for the SQL statement in the lab13\_1.sql script.
- 5 Use SQL Trace and TKPROF to look at the performance of the query executed by the lab13\_2.sql script.
  - a Set up your session in trace mode.
  - b Run the lab13\_2.sql script.
  - c Exit from SQL\*Plus and format your trace file using TKPROF. Use the options SYS=NO and EXPLAIN.  
**Hint:** Find the trace file in the UDUMP directory and use it as an input file for TKPROF.
  - d You should only note down the CPU, current, and query figures for the fetch phase. You will not see a drastic change in numbers. The objective is to become familiar and comfortable with running TKPROF and SQL Trace.  
**Hint:** Find the output file in the current working directory. By default, the name of the file is report.prf. Use the text file to find the statistics for the user statement below.

Results:

Query	CPU (secs)	Query (blocks)	Current (blocks)
<pre>select e.last_name, d.name from   s_dept d, s_emp e where  d.id = e.dept_id and    d.id = 10;</pre>			



- 
- 6** Gather statistics for all objects under the SCOTT schema while saving the current statistics, then restore the original statistics.
- a** Connect as SCOTT and create a table to hold statistics in that schema.  
**Hint:** Use the `dbms_stats.create_stat_table` procedure.
  - b** Save the current schema statistics into your local statistics table.  
**Hint:** Use the `dbms_stats.export_schema_stats` procedure.
  - c** Analyze all objects under the SCOTT schema.  
**Hint:** Use the `dbms_stats.gather_schema_stats` procedure.
  - d** Remove all schema statistics from the dictionary, and restore the original statistics you saved in step b.  
**Hint:** Use the `dbms_stats.delete_schema_stats` and `dbms_stats.import_schema_stats` procedures.

## Practice 14-1 Hint

- 1 Create a consumer group called ONLINE\_USERS.

**Hint:** Remember to create the pending area first.

- 2 Create a resource plan called DAYTIME with the following characteristics:
  - Users in the SYS\_GROUP get as much CPU time as they require.
  - Users in the ONLINE\_USERS group get 80% of the remaining resources.
  - Users in the OTHER\_GROUPS get 20% of the remaining resources.

There is no parallel query processing occurring.

This is the only plan that will be created, so complete the process of creating a plan.

**Hint:** Remember to validate and submit the pending area when updates are complete.

- 3 Make ONLINE\_USERS the default consumer group for User01.

**Hint:** First, grant User01 the privilege to switch to this group.

- 4 Give User01 the capability to switch to the SYS\_GROUP.

- 5 Activate DAYTIME as the current resource plan group.

**Hint:** There is no hint for this question.

- 6 Start a second SQL\*Plus session and log on as User01. Display the current consumer group for User01's session.

**Hint:** You may have to log on as SYS to display the data from V\$SESSION.

- 7 As User01, switch to the SYS\_GROUP.

**Hint:** Use a PL/SQL block, because the procedure has an OUT mode variable.

- 8 Redisplay User01's current consumer group.

---

C

---

## **Practice Solutions**

## Practice 3-1 Solutions

The goal of this practice is to familiarize yourself with the machine setup and to have a look at the diagnostic tools, such as trace files, waiting event views, and EM event management.

- 1 Log on as directed by your instructor.
- 2 View your parameter file and check that the appropriate parameters locate the alert log file and user trace files in the BDUMP and UDUMP sub-directories respectively in your home directory. If necessary, shut down and then start up your instance.

```
BACKGROUND_DUMP_DEST=$HOME/BDUMP
```

```
USER_DUMP_DEST=$HOME/UDUMP
```

- 3 Examine your alert.log file to find any internal errors (ORA-600).

```
$ cd $HOME/BDUMP
```

```
$ more alert_U01.log
```

- 4 As SYSTEM, connect to SQL\*Plus and issue a command that will create a user trace file after querying against the DBA\_TABLES data dictionary view. Do not try to interpret the content of the trace file. You will learn how to do this in Lesson 13, “SQL Issues and Tuning Considerations for Different Applications.”

```
SQL> ALTER SESSION SET SQL_TRACE=TRUE;
```

```
Session altered.
```

```
SQL> select * from sys.dba_tables;
```

```
SQL> exit;
```

- 5 View the resulting trace file without trying to interpret its content. You will learn how to do this in Lesson 13, “SQL Issues and Tuning Considerations for Different Applications.”

```
$cd $HOME/UDUMP
```

```
$ls -l
```

```
-rw-r----- 1 ora803 dba 7904 Feb  6 17:29 o804_ora_14628.trc
```

```
$more o804_ora_14628.trc
```

## Practice 4-1 Solutions

The objective of this practice is to familiarize yourself with different ways of retrieving statistics information: run the `utlbstat.sql` and `utlestat.sql` scripts, examine the resulting `report.txt` file, display the major performance dynamic views appropriate for tuning.

During the labs, use the `$HOME/LABS/initUn.ora` parameter file to start up the database.

- 1 Before running the `utlbstat.sql` script, verify that the parameter `TIMED_STATISTICS` has been set to true.

```
TIMED_STATISTICS=true (In the init.ora)
```

or

```
SQL> alter system set TIMED_STATISTICS=true;
```

- 2 Run the `utlbstat.sql` script connected to `SQL*PLUS` as a `sysdba`.

```
SQL> @$ORACLE_HOME/rdbms/admin/utlbstat
```

- 3 Start the following three sessions under `SCOTT` user:

- The first session executes the `lab04_1.sql` script (it creates and loads data into `SCOTT`'s tables). Wait until the script has finished.

```
$sqlplus scott/tiger
```

```
SQL> @lab04_1
```

- The second session then executes the `lab04_2.sql` script.

```
$sqlplus scott/tiger
```

```
SQL> @lab04_2
```

- At the same time, the third session executes the `lab04_3.sql` script.

```
$sqlplus scott/tiger
```

```
SQL> @lab04_3
```

Do not wait until the last two scripts have finished loading data. Go to step 4.

**4 Query the appropriate dynamic view to see:**

- The library cache performance

```
$sqlplus system/manager
```

```
SQL> select namespace, gethitratio, pinhitratio, reloads,
2         invalidations
3 from v$librarycache;
```

NAMESPACE	GETHITRATIO	PINHITRATIO	RELOADS	INVALIDATIONS
SQL AREA	.976229896	.993471792	23	30
TABLE/PROCEDURE	.799808429	.988957654	21	0
BODY	.666666667	.666666667	0	0
TRIGGER	.5	0	0	0
INDEX	0	0	0	0
CLUSTER	.971153846	.976271186	0	0
OBJECT	1	1	0	0
PIPE	1	1	0	0

8 rows selected.

- Among the system statistics, those concerning sort activity

```
SQL> select * from v$sysstat
2 where name like '%sort%';
```

STATISTIC#	NAME	CLASS	VALUE
173	sorts (memory)	64	3502
174	sorts (disk)	64	0
175	sorts (rows)	64	4425

- The sessions consuming more than 10,000 bytes of PGA memory

```
SQL> select username,name,value
2 from v$statname n, v$session s, v$sesstat t
3 where s.SID=t.SID
4 and n.statistic#=t.statistic#
5 and s.type='USER'
6 and s.username is not null
7 and n.name='session pga memory'
8*and t.value > 10000;
```

USERNAME	NAME	VALUE
SYSTEM	session pga memory	468816

– The rollback segments statistics

```
SQL> select usn, gets, waits
       2 from v$rollstat;
```

USN	GETS	WAITS
0	96	0
1	457	0
2	459	0
3	19925	20

4 rows selected.

– The file I/O statistics

```
SQL> select name, phydrds, phywrts, phyblkrd, phyblkwrt, readtim,
       2 writetim
       3 from v$datafile d, v$filestat f
       4 where d.file#=f.file#;
```

NAME	PHYDRDS	PHYWRTS	PHYBLKRD	PHYBLKWRT	READTIM	WRITETIM
/DATA/DISK1/sys01.dbf	957231	1	210	231	302	516
/DATA/DISK1/temp01.dbf	6	4	6	4	0	0
/DATA/DISK2/rbs01.dbf	21	165	21	165	3	323
/DATA/DISK3/user01.dbf	6	4	6	4	0	0
/DATA/scott_dat.dbf	327120	4	2601311	4	17539	0
DATA/scott_ind.dbf	6	4	6	4	0	0

6 rows selected.

**5** Run the utlestat.sql script so that the resulting report is stored in your UDUMP directory.

```
$cd UDUMP
$ sqlplus "/ as sysdba"
SQL> @$ORACLE_HOME/rdbms/admin/utlestat
```

**6** Examine the resulting report.txt file and compare what you displayed in:

- The library cache dynamic view with the library cache section of the report.

\$more report.txt (read 1st section)

LIBRARY	GETS	GETHITRATI	PINS	PINHITRATI	RELOADS	SINVALIDATI
BODY	105	.65	105	.65	0	0
CLUSTER	10	.97	9	.97	0	0
INDEX	0	1	0	1	0	0
OBJECT	0	1	0	1	0	0
PIPE	0	1	0	1	0	0
SQL AREA	2036	.92	12822	.932	95	30
TABLE/PROCED	553	.69	3714	.90	81	0
TRIGGER	917	.5	917	.997	3	0

8 rows selected.

- File I/O dynamic view with the I/O statistics section of the report

\$more report.txt (read the last section)

TABLE_SPACE	FILE_NAME	READS	BLKS_READ	READ_TIME
WRITES	BLKS_WRT	WRITE_TIME		
RBS	/DATA/DISK2/rbs01.dbf	26	26	50
257	257	2411		
SCOTT_DATA	/DATA/scott_dat.dbf	465012	3416752	18420
5	5	0		
SCOTT_INDEX	/DATA/scott_ind.dbf	8	8	0
8	8	0		
SYSTEM	/DATA/DISK1/sys01.dbf	1206	15	198
320	321	612		
TEMP	/DATA/DISK1/temp01.dbf	168	666	483
675	675	0		
USER_DATA	/DATA/DISK3/user01.dbf	8	8	0
8	8	0		

6 rows selected.

You will detect values different from those in the V\$FILESTAT view and the values in the report section, because the last one covers a period of time, whereas V\$FILESTAT displays the current values of the time.



**7** Display the list of Wait event names.

```
SQL> select name,parameter1 from v$event_name;
```

NAME	PARAMETER1
free buffer waits	file#
checkpoint completed	
buffer for checkpoint	buffer#
write complete waits	file#
buffer read retry	file#
buffer busy waits	file#
log switch/archive	thread#
...	

**8** Are there any sessions actually waiting for resources?

Be careful to set the `init.ora` parameter `TIMED_STATISTICS` to true.

```
SQL> select SID,event,pltext,wait_time,state
       2 from v$session_wait;
```

SID	EVENT	PLTEXT	WAIT_TIME	STATE
1	pmon timer	duration	0	WAITING
2	rdbms ipc message	timeout	0	WAITING
3	rdbms ipc message	timeout	0	WAITING
6	rdbms ipc message	timeout	0	WAITING
4	rdbms ipc message	timeout	0	WAITING
5	smon timer	sleep time	0	WAITING
7	SQL*Net message to client	driver id	2	WAITED KNOWN TIME

## Practice 5-1 Solutions

The objective of this practice is to use diagnostics tools to monitor and tune the shared pool.

- 1 Check the size of the shared pool.

```
SHARED_POOL_SIZE = 3 000 000
```

- 2 From SQL\*Plus, connected as SYSDBA, run the `utlbstat.sql` script to start collecting statistics.

```
$sqlplus
```

```
SQL> @$ORACLE_HOME/rdbms/admin/utlbstat
```

- 3 To simulate user activity against the database, connect as SCOTT and run the `lab05_1.sql` script and in another session `lab05_2.sql`.

```
$sqlplus scott/tiger
```

```
SQL> @lab05_1
```

```
$sqlplus scott/tiger
```

```
SQL> @lab05_2
```

- 4 Measure the pin hit ratio for the library cache. Determine if it is a good ratio or not.

Using the dynamic view:

```
$sqlplus system/manager
```

```
SQL> select sum(pins), sum(reloads),
2          sum(reloads)/ sum(pins)*100
3          from v$llibrarycache;
```

```
SUM(PINS) SUM(RELOADS) SUM(RELOADS)/SUM(PINS)*100
```

```
-----
149096          11          .007378
```

- 5** Measure the hit ratio for the data dictionary cache. Determine if it is a good ratio or not.

Using the dynamic view:

```
$sqlplus system/manager
SQL> select sum(getmisses), sum(gets),
2         sum(getmisses)/sum(gets)*100
3         from v$rowcache;
SUM(GETMISSES) SUM(GETS) SUM(GETMISSES)/SUM(GETS)*100
-----
```

4768	88138	5.409698
------	-------	----------

**Because GETMISSES are much less than 15% of the GETS, it is a good ratio.**

- 6** When both scripts have finished running, from SQL\*Plus and connected as SYSDBA, run the utlestat.sql script to collect ending statistics.

```
$sqlplus "/as sysdba"
SQL> @$ORACLE_HOME/rdbms/admin/utlestat
```

- 7** Use the report.txt output to get both hit ratios.

LIBRARY	GETS	GETHITRATI	PINS	PINHITRATI	RELOADS
BODY	105	1	105	1	0
CLUSTER	10	1	9	1	0
INDEX	1	0	1	0	0
OBJECT	0	1	0	1	0
PIPE	0	1	0	1	0
SQL AREA	2036	.987	12822	.982	.95
TABLE/PROCED	553	.98	3714	.969	81
TRIGGER	917	1	917	.997	3

176 RELOADS for 17577 PINS: the hit ratio is now 1.001%. It is worse than in 5. This is the result after a longer period of activity. This figure is more relevant than the one in 5.

NAME	GET_REQS	GET_MISS
dc_tablespace	311	0
dc_free_extents	338	195
dc_segments	330	9
dc_rollback_seg	171	0
dc_used_extents	376	142
dc_users	350	0
dc_user_grants	227	0
dc_objects	279	10
dc_synonyms	9	1
dc_usernames	73	0
dc_object_ids	86	0
dc_sequences	96	0
dc_profiles	19	0

**357 GETMISSES for 2665 GETS: the ratio is now 13.39%, worse than in step 6. This is the result after a longer period of activity; this figure is more relevant than the one in step 6.**

**8** Which solutions would you apply if any of the hit ratios were bad?

**Increase the SHARED\_POOL\_SIZE parameter.**

**9 Determine which packages, procedures, and triggers are pinned in the shared pool.**

```
SQL> select name,type,kept
2> from v$db_object_cache
3> where type = 'PACKAGE'
4> or      type = 'PROCEDURE'
5> or      type = 'TRIGGER'
6> or      type = 'PACKAGE BODY';
```

NAME	TYPE	KEP
-----	-----	---
DBMS_SHARED_POOL	PACKAGE	NO
DBMS_SHARED_POOL	PACKAGE BODY	NO
DBMS_APPLICATION_INFO	PACKAGE	NO
DBMS_APPLICATION_INFO	PACKAGE BODY	NO
DBMS_STANDARD	PACKAGE	NO
S_EMP_TRIGGER	TRIGGER	NO
STANDARD	PACKAGE	NO
STANDARD	PACKAGE BODY	NO
DBMS_OUTPUT	PACKAGE	NO
DBMS_OUTPUT	PACKAGE BODY	NO
DBMS_UTILITY	PACKAGE	NO
DBMS_UTILITY	PACKAGE BODY	NO

**10 Pin one of the Oracle supplied packages that needs to be kept in memory.**

```
SQL> CONNECT / AS SYSDBA
SQL> @?/rdbms/admin/dbmspool
SQL> execute DBMS_SHARED_POOL.KEEP('SYS.STANDARD');
PL/SQL procedure successfully completed.
SQL> select * from v$db_object_cache where kept='YES';
```

**11 Determine the amount of session memory used in the shared pool for your session.**

```
SQL> select a.name, b.value
2> from v$statname a, v$mystat b
3> where a.statistic# = b.statistic#
4> and name = 'session uga memory';
```

NAME	VALUE
-----	-----
session uga memory	43824

**12 Determine the amount of session memory used in the shared pool for all sessions.**

- Using V\$SESSTAT and V\$STATNAME views:

```
SQL> select SUM(value)||' bytes' "Total session memory"
  2   from v$sesstat, v$statname
  3   where name = 'session uga memory'
  4   and v$sesstat.statistic# = v$statname.statistic#;
Total session memory
-----
173476 bytes
```

## Practice 6-1 Solutions

The objective of this practice is to use available diagnostics tools to monitor and tune the database buffer cache.

- 1 From SQL\*Plus, connected as SYSDBA, and run the `utlbstat.sql` script to start collecting statistics.

```
$ sqlplus "/as sysdba"
SQL> @$ORACLE_HOME/rdbms/admin/utlbstat
```

- 2 To simulate user activity against the database, connect as SCOTT and run the `lab06_1.sql` script.

```
SQL> connect scott/tiger
Connected.
SQL> @lab06_1
PL/SQL procedure successfully completed.
```

- 3 Connect as SYSTEM and measure the hit ratio for the database buffer cache. Determine if it is a good ratio or not.

To query the V\$SYSSTAT view:

```
SQL> SELECT 1 - (phy.value / (cur.value + con.value))
2  "CACHE HIT RATIO"
3  FROM v$sysstat cur, v$sysstat con, v$sysstat phy
4  WHERE cur.name = 'db block gets'
5  AND   con.name = 'consistent gets'
6  AND   phy.name = 'physical reads';
CACHE HIT RATIO
-----
.797827453
```

- 4 When the script has finished, from SQL\*Plus, and connected as SYSDBA, run the `utlestat.sql` script to collect ending statistics.

```
$ cd $HOME/UDUMP
$ sqlplus "/ as sysdba"
SQL> @$ORACLE_HOME/rdbms/admin/utlestat
```

- 5** Use the `report.txt` output to get the cache hit ratio.

Use statistics from `stats$stats` to compute the hit ratio.

Statistic	Per Transaction	Per Logon	Per Second	Total
consistent gets	2613	435.5	522.6	68.76
db block gets	524	87.33	104.8	13.79
physical reads	658	109.67	131.6	17.32

If the ratio is bad, add new buffers, run steps 2 to 5, and examine the new ratio to verify that the ratio has improved. If the ratio is good, remove buffers, run steps 2 to 5, and verify if the ratio is still good.

- 6** Size the keep buffer pool to hold the `SCOTT.S_EMP` and `SCOTT.S_DEPT` tables. Analyze both tables.

```
SQL> analyze table scott.s_emp compute statistics;
```

Table analyzed.

```
SQL> analyze table scott.s_dept compute statistics;
```

Table analyzed.

**Query the `DBA_TABLES.BLOCKS` to get their size in blocks.**

```
SQL> select table_name , blocks
```

```
2 from dba_tables
```

```
3 where table_name in ('S_EMP','S_DEPT');
```

TABLE_NAME	BLOCKS
S_DEPT	1
S_EMP	65

- 7** Modify the `init.ora` file to set up the keep buffer pool.

```
BUFFER_POOL_KEEP=(buffers:50,lru_latches:1)
```

**The minimum number of buffers that you must allocate to each buffer pool is 50 times the number of LRU latches.**



**Because DB\_BLOCK\_BUFFERS was 60, it is not enough for both the DEFAULT and KEEP pools.**

```
DB_BLOCK_BUFFERS=100
DB_BLOCK_LRU_LATCHES=5
SQL> SELECT name, buffers, set_count
       2 FROM v$dbuffer_pool;
```

NAME	BUFFERS	SET_COUNT
-----	-----	-----
	0	0
KEEP	50	1
RECYCLE	0	1
DEFAULT	50	1

- 8** Enable the SCOTT.S\_EMP and SCOTT.S\_DEPT tables for caching in the keep pool.

Execute ALTER TABLE ... STORAGE commands against both tables.

```
SQL> alter table scott.s_emp storage (buffer_pool keep);
Table altered.
SQL> alter table scott.s_dept storage (buffer_pool keep);
Table altered.
SQL> SELECT table_name,buffer_POOL
       2 FROM dba_tables
       3 WHERE table_name in ('S_EMP','S_DEPT');
```

TABLE_NAME	BUFFER_
-----	-----
S_DEPT	KEEP
S_EMP	KEEP

- 9** Set up both of these tables so that their blocks are kept in memory, even during full table scans.

Execute ALTER TABLE ... CACHE commands against both tables.

```
SQL> alter table scott.s_emp cache;
Table altered.
SQL> alter table scott.s_dept cache;
Table altered.
SQL> SELECT table_name, cache
       2 FROM dba_tables
```

```

3 WHERE table_name in ('S_EMP','S_DEPT');
TABLE_NAME                                CACHE
-----
S_DEPT                                    Y
S_EMP                                    Y

```

- 10** Determine how many LRU latches have been allocated for your instance.

Use the V\$BUFFER\_POOL view:

```

SQL> select name, set_count, buffers
2 from v$buffer_pool;
NAME                                SET_COUNT    BUFFERS
-----
0                                0            0
KEEP                              1            50
RECYCLE                          1            0
DEFAULT                          1            50

```

- 11** Connect as SYSTEM and display the hit percentage for the LRU latches.

Use the V\$LATCH view:

```

SQL> select name, sleeps/gets "LRU Hits"
2 from v$latch
3* where name = 'cache buffers lru chain';
NAME                                LRU Hits
-----
cache buffers lru chain 0

```

- 12** Determine what the overall hit percentage is for free lists in your instance.

Use the V\$WAITSTAT and V\$SYSTEM\_EVENT views:

```

SQL> select class, count, time
2 from v$waitstat
3* where class='data block'
CLASS                                COUNT        TIME
-----
data block                          1            0
SQL> select event, total_waits
2 from v$system_event
3 where event = 'buffer busy waits';
EVENT                                TOTAL_WAITS
-----
buffer busy waits                    1

```

## Practice 7-1 Solutions

The objective of this practice is to use available diagnostics tools to monitor and tune the redo log buffer.

- 1 Connect as SYSTEM and measure the redo buffer allocation retries for the redo entries. Determine if it is a good ratio or not.

```
SQL> SELECT RBAR.name, RBAR.value, RE.name, RE.value,
2          (RBAR.value*100)/RE.value||'%' "ratio"
3 FROM    v$sysstat RBAR, v$sysstat RE
4 WHERE   RBAR.name = 'redo buffer allocation retries'
5 AND     RE.name   = 'redo entries';
```

NAME	VALUE	NAME	VALUE	ratio
redo buffer allocation retries	6	redo entries	317	1.26%

**It is not a good ratio, because it should not be greater than 1.**

- 2 Which solution would you apply if the ratio were bad?  
**Increase the value of the parameter LOG\_BUFFER.**
- 3 Change the size of the appropriate parameter.

```
LOG_BUFFER=131072
```

## Practice 8-1 Solutions

- 1 Have a look at your database. Find out if there are deficiencies concerning the database configuration. If necessary, change the database configuration. Consider the following:

- a How many tablespaces do you have? Add more tablespaces, if necessary, spreading them across the subdirectories that represent devices.

```
SQL> SELECT * FROM v$tablespace;
```

```
TS#          NAME
```

```
-----  
      0  SYSTEM  
      1  TEMP  
      2  RBS  
      3  USER_DATA
```

```
4 rows selected.
```

```
SQL> SELECT file_name, tablespace_name, bytes
```

```
2> FROM dba_data_files;
```

FILE_NAME	TABLESPACE_NAME	BYTES
/users/tun8/DATA/DISK2/user01.dbf	USER_DATA	104857600
/users/tun8/DATA/DISK2/rbs01.dbf	RBS	31457280
/users/tun8/DATA/DISK1/temp01.dbf	TEMP	20971520
/users/tun8/DATA/DISK1/system01.dbf	SYSTEM	31457280

```
4rows selected.
```

**The tablespace for the indexes is missing.**

**If another application is created, a new tablespace should be created: if the application objects are constantly growing in an unpredictable way, create a locally managed tablespace with a uniform extents.**

- b Are there any non-data-dictionary objects in the SYSTEM tablespace? Objects can be moved by creating one table as a copy of another, dropping the original object, and then renaming the new object to have the original name.

```
SQL> SELECT owner, segment_name, segment_type, tablespace_name
```

```
2 FROM dba_segments
```

```
3 WHERE tablespace_name = 'SYSTEM'
```

```
4 AND owner != 'SYS' AND owner != 'SYSTEM' AND owner != 'OUTLN';
```

```
no rows selected
```

- c** Check to see if you have the data files on different disks from your online redo log files. Also, the archive logs should be on different disk from both the data files and redo log files.

```
SQL> SELECT name FROM v$datafile
2 UNION
3 SELECT member from v$logfile
4 UNION
5 SELECT value from v$parameter
6 WHERE name IN ('log_archive_dest','log_archive_duplex_dest');
NAME
```

```
-----
/users/tun8/DATA/DISK1/system01.dbf
/users/tun8/DATA/DISK1/temp01.dbf
/users/tun8/DATA/DISK2/rbs01.dbf
/users/tun8/DATA/DISK2/user01.dbf
/users/tun8/DATA/DISK3/log1a.rdo
/users/tun8/DATA/DISK3/log2a.rdo
/users/tun8/archive
```

8 rows selected.

**The redo log, data, and archived redo log files are on different disks.**

- d** Are the online redo log files mirrored?

```
SQL> SELECT * FROM v$log;
GROUP#    THREAD#    SEQUENCE#  BYTES    MEMBERS    ARC STATUS
-----
FIRS
T_CHAN FIRST_TIM
-----
-----
-----
1          1          532      204800      1 NO    CURRENT
34388 26-APR-98
2          1          530      204800      1 NO    INACTIVE
34326 26-APR-98
2 rows selected.
```

**Yes.**

- 2** Find out how often checkpoints are occurring. Is this timing optimal? If not, correct it.

**View the initialization parameter file or run the following query:**

```
SQL> SELECT name, value FROM V$PARAMETER
       2 WHERE name like 'log_check%' OR name like 'fast_start%';
```

NAME	VALUE
log_checkpoint_interval	10000
log_checkpoint_timeout	1800
fast_start_io_target	60
fast_start_parallel_rollback	LOW
log_checkpoints_to_alert	FALSE

If you find that the timing is not appropriate, edit the initialization parameter file and restart the instance.

- 3** To simulate user activity against the database, connect as SCOTT and run the lab08\_1.sql script. Monitor the I/O statistics.

```
SQL> @lab08_1.sql
SQL> SELECT phydds,phywrts,d.name
       2> FROM v$datafile d, v$filestat f
       3> WHERE d.file#=f.file#;
```

PHYRDS	PHYWRTS	NAME
2023	11	/users/tun8/DATA/DISK1/system01.dbf
3	1	/users/tun8/DATA/DISK1/temp01.dbf
35	23	/users/tun8/DATA/DISK2/rbs01.dbf
3	1	/users/tun8/DATA/DISK2/user01.dbf
3	1	/users/tun8/DATA/DISK1/temp02.dbf
3	1	/users/tun8/DATA/DISK2/rbs_test.dbf

6 rows selected.

## Practice 9-1 Solutions

The objective of this practice is to use available diagnostics tools to monitor and tune block space usage.

- 1 Connect as SYSTEM and find the PCTFREE, CHAIN\_CNT and high-water mark for SCOTT's S\_ORD table.

Analyze the table and select the statistics from DBA\_TABLES.

```
SQL> analyze table scott.s_ord compute statistics;
```

Table analyzed.

```
SQL> select table_name,
2         pct_free,
3         chain_cnt,
4         blocks,
5         empty_blocks
6 from dba_tables
```

```
7 where table_name='S_ORD' and owner='SCOTT';
```

TABLE_NAME	PCT_FREE	CHAIN_CNT	BLOCKS	EMPTY_BLOCKS
S_ORD	5	0	1	48

- 2 Run the lab09\_1.sql script to update rows in SCOTT.S\_ORD

```
SQL> @lab09_1
```

Table altered.

4 rows updated.

- 3 Find the CHAIN\_CNT and high-water mark for SCOTT's S\_ORD table.

```
SQL> analyze table scott.s_ord compute statistics;
```

Table analyzed.

```
SQL> select table_name, chain_cnt, blocks, empty_blocks
2  from dba_tables
3  where table_name = 'S_ORD' and owner='SCOTT';
```

TABLE_NAME	CHAIN_CNT	BLOCKS	EMPTY_BLOCKS
S_ORD	3	2	47

#### 4 Eliminate any migrated rows.

```

- Run utlchain.sql
SQL> @$ORACLE_HOME/rdbms/admin/utlchain
Table created.
- Run ANALYZE TABLE ... LIST CHAINED ROWS.
SQL> analyze table scott.s_ord list chained rows;
Table analyzed.
- Copy, delete, and reinsert the migrated rows.
SQL> create table scott.temp
  2 as select * from scott.s_ord
  3 where rowid in (select head_rowid
  4 from chained_rows);
Table created.
SQL> alter table scott.s_ord disable primary key cascade;
SQL> delete from scott.s_ord
  2 where rowid in (select head_rowid
  3                  from chained_rows);
3 rows deleted.
SQL> insert into scott.s_ord
  2 select * from scott.temp;
3 rows created.
SQL> commit;
Commit complete.
SQL> alter table scott.s_ord enable primary key;
Table altered.
SQL> alter table scott.s_item enable constraint S_ITEM_ORD_ID_FK;
Table altered.

```

#### 5 Find the CHAIN\_CNT and high-water mark for SCOTT's S\_ORD table.

```

SQL> analyze table scott.s_ord compute statistics;
Table analyzed.
SQL> select table_name, chain_cnt, blocks, empty_blocks
  2 from dba_tables
  3 where table_name = 'S_ORD' and owner = 'SCOTT';

```

TABLE_NAME	CHAIN_CNT	BLOCKS	EMPTY_BLOCKS
S_ORD	0	2	47



**6** Remove the rows from table CHAINED\_ROWS without deleting them.

Use the TRUNCATE command.

```
SQL> truncate table chained_rows;
Table truncated.
SQL> drop table scott.temp;
Table dropped.
```

**Optional Practice** Find the high-water mark for SCOTT's S\_ORD table without querying the data dictionary. You can use the lab09\_2.sql script.

Use the DBMS\_SPACE package and UNUSED\_SPACE procedure.

```
SQL> @lab09_2.sql
SQL> variable a number
SQL> variable b number
SQL> variable c number
SQL> variable d number
SQL> variable e number
SQL> variable f number
SQL> variable g number
SQL> execute dbms_space.unused_space('SCOTT', 'S_ORD', 'TABLE',
:a, :b, :c, :d, :e, :f, :g);
PL/SQL procedure successfully completed.
SQL> set feedback off
SQL> set serveroutput on
SQL> exec dbms_output.put_line('TOTAL_BLOCKS = '||:a)
TOTAL_BLOCKS = 50
SQL> exec dbms_output.put_line('TOTAL_BYTES = '||:b)
TOTAL_BYTES = 102400
SQL> exec dbms_output.put_line('UNUSED_BLOCKS = '||:c)
UNUSED_BLOCKS = 47
SQL> exec dbms_output.put_line('UNUSED_BYTES = '||:d)
UNUSED_BYTES = 96256
SQL> exec dbms_output.put_line('LAST_USED_EXTENT_FILE_ID = '||:e)
LAST_USED_EXTENT_FILE_ID = 4
SQL> exec dbms_output.put_line('LAST_USED_EXTENT_BLOCK_ID = '||:f)
LAST_USED_EXTENT_BLOCK_ID = 11822
SQL> exec dbms_output.put_line('LAST_USED_BLOCK = '||:g)
LAST_USED_BLOCK = 3
```

## Practice 10-1 Solutions

The objective of this practice is to use available diagnostics tools to monitor and tune sorts.

- 1 Connect as SYSTEM and execute the lab10\_1.sql script. It forces sorts to write to disk.

```
SQL> CONNECT SYSTEM/MANAGER
```

```
Connected.
```

```
SQL> @lab10_1
```

```
Session altered.
```

```
TABLE_NAME
```

```
-----
```

```
ACCESS$
```

```
ACCESS$
```

```
ACCESS$
```

```
ACCESS$
```

```
...
```

- 2 Measure the sorts (disk) to the sorts (memory) ratio.

**Query the V\$SYSSTAT view:**

```
SQL> select disk.value "Disk", mem.value "Mem",
2         (disk.value/mem.value)*100 "Ratio"
3   from   v$sysstat mem, v$sysstat disk
4  where   mem.name = 'sorts (memory)'
5  and     disk.name = 'sorts (disk)';
```

```
      Disk      Mem      Ratio
-----
      44      2135    2.0608899
```

- 3 Configure the SORT\_AREA\_SIZE parameter to result in sorts (memory) only.

```
SQL> alter system set SORT_AREA_SIZE=1048576 deferred;
```

- 4** Connect as SCOTT and execute the lab10\_2.sql script and recalculate the ratio.

```
SQL> @lab10_2
TABLE_NAME
-----
ACCESS$
ACCESS$
ACCESS$
ACCESS$
...
SQL> select disk.value "Disk", mem.value "Mem",
2          (disk.value/mem.value)*100 "Ratio"
3  from    v$sysstat mem, v$sysstat disk
4  where   mem.name = 'sorts (memory)'
5  and     disk.name = 'sorts (disk)';
      Disk      Mem      Ratio
-----
          47      3238      2.007219
```

- 5** Create a new TEMPORARY tablespace; when SCOTT runs sort operations, the sorts to disk occur on this tablespace.

```
SQL> create tablespace temp2 datafile '$HOME/DATA/temp2.dbf'
2  size 2m TEMPORARY;
Tablespace created.
SQL> alter user scott temporary tablespace temp2;
User altered.
```

- 6** Connect as SCOTT and reexecute the lab10\_1.sql script. Monitor the TEMPORARY tablespace. After that set the SORT\_AREA\_SIZE back to 64000.

```
SQL> select tablespace_name, current_users, total_extents,
2          used_extents, extent_hits, max_used_blocks,
3          max_sort_blocks
4  from v$sort_segment;
TABLESPACE_NAME CURRENT_USERS TOTAL_EXTENTS USED_EXTENTS
EXTENT_HITS MAX_USED_BLOCKS MAX_SORT_BLOCKS
-----
0 rows selected.
SQL> @lab10_1
Session altered.
```

## Appendix C: Practice Solutions

---

TABLE\_NAME

-----

ACCESS\$

ACCESS\$

...

```
SQL> select tablespace_name, current_users, total_extents,
2  used_extents, extent_hits, max_used_blocks,
3  max_sort_blocks
4  from v$sort_segment;
```

TABLESPACE_NAME	CURRENT_USERS	TOTAL_EXTENTS	USED_EXTENTS
EXTENT_HITS	MAX_USED_BLOCKS	MAX_SORT_BLOCKS	

-----

-----

TEMP2	0	810	0
1	4050	4050	

## Practice 11-1 Solutions

The objective of this practice is to use available diagnostics tools to monitor and tune the rollback segments. You will use the `report.txt` output, so run `utlbstat.sql` now.

- 1 Run the `utlbstat.sql` script.

```
$sqlplus
```

```
SQL> @$ORACLE_HOME/rdbms/admin/utlbstat
```

If you have not already done so, create a new rollback segment tablespace at least 2 MB in size. Connect as `SYSTEM`.

```
SQL> CREATE TABLESPACE rbs_test
```

```
2> DATAFILE '$ORACLE_HOME/DATA/DISK2/rbs_test.dbf' SIZE 2m;
```

- 2 For the purposes of this practice, create a new rollback segment called `RBSX`. Give it an overall size of 200 KB, divided into 20 extents. It should shrink back to 200 KB if expanded.

For the storage parameters, use 10 KB for the `INITIAL` and `NEXT` extent sizes with `MINEXTENTS` value set to 20. Set the `OPTIMAL` value so that the segment shrinks back to 200 KB automatically.

```
SQL> create rollback segment rbsx
```

```
2 tablespace rbs_test
```

```
3 storage (initial 10k
```

```
4          next 10k
```

```
5          minextents 20
```

```
6          optimal 200k);
```

Rollback segment created.

- 3** Bring your new rollback segment online and take any others (except the SYSTEM rollback segment) offline.

Query the DBA\_ROLLBACK\_SEGS view to get the names of the rollback segments to be taken offline by the ALTER ROLLBACK SEGMENT command.

```
SQL> select segment_id,segment_name,status
       2  from dba_rollback_segs;
```

SEGMENT_ID	SEGMENT_NAME	STATUS
0	SYSTEM	ONLINE
1	RBS01	ONLINE
2	RBS02	ONLINE
6	RBSX	OFFLINE

```
SQL> alter rollback segment rbsx online;
```

Rollback segment altered.

```
SQL> alter rollback segment rbs01 offline;
```

Rollback segment altered.

```
SQL> alter rollback segment rbs02 offline;
```

Rollback segment altered.

- 4** Before executing a new transaction, find the number of bytes written in the new rollback segment.

**Hint:** In the V\$ROLLSTAT view, query the WRITES column to get the number of bytes written in the RBSX rollback segment until now.

```
SQL> select usn, extents, rssize, writes, optsize, shrinks,
       2          wraps, extends
       3  from v$rollstat
       4  where usn=6
```

USN	EXTENTS	RSSIZE	WRITES	OPTSIZE	SHRINKS
WRAPS	EXTENDS				
6	3	28672	96528	102400	0
0	0				

- 5** As SCOTT, run the script ins\_ords.sql. The script inserts 100 new rows into the S\_ORD table. Without committing, log in again as SYS (either use the HOST command or open another window).

How many rollback segment blocks or bytes is the transaction using?

- Join the V\$TRANSACTION and V\$SESSION views to find, in the USED\_UBLK column, for your session, how many blocks the current transaction is using.

```
SQL> SELECT s.username, t.used_ublk, t.start_time
       2 FROM v$transaction t, v$session s
       3 WHERE t.addr = s.taddr;
```

USERNAME	USED_UBLK	START_TIME
SCOTT	7	04/06/98 16:24:59

- In the V\$ROLLSTAT view, query the WRITES column to get the number of bytes written in the RBSX rollback segment since step 3.

```
SQL> select usn, extents, rssize, writes, optsize, shrinks,
       2 wraps, extends
       3 from v$rollstat where usn=6;
```

USN	EXTENTS	RSSIZE	WRITES	OPTSIZE	SHRINKS
6	3	28672	97350	102400	0

- 6** Return to the SCOTT session and commit the insert. Run the del\_ordr script, without committing. This script deletes the hundred rows you have just inserted. As SYS, check the amount of rollback space used, as in step 5. Note the difference between the two runs.

```
SQL> commit;
```

Commit complete.

```
SQL> @del_ordr
```

600 rows deleted.

```
SQL> SELECT s.username, t.used_ublk, t.start_time
       2 FROM v$transaction t, v$session s
       3 WHERE t.addr = s.taddr;
```

USERNAME	USED_UBLK	START_TIME
SCOTT	68	04/06/98 16:32:57

```
SQL> select usn, extents, rssize, writes, optsize, shrinks,
       2 wraps, extends
       3 from v$rollstat where usn=6;
```

USN	EXTENTS	RSSIZE	WRITES	OPTSIZE	SHRINKS
6	17	186368	251076	102400	0

**7** As SYS, find out if you have had any rollback segment contention since startup, using two different sources of information.

- In V\$ROLLSTAT view, query the WAITS and GETS column values to compute the contention ratio.

```
SQL> select sum(waits)/sum(gets) "Ratio",
2         sum(waits) "Waits", sum(gets) "Gets"
3   from v$rollstat;
      Ratio      Waits      Gets
-----
          0          0      1272
```

- A better source of information is the `report.txt` output.

\$sqlplus

```
SQL> @$ORACLE_HOME/rdbms/admin/utlestat
```

.....

```
SQL> select * from stats$waitstat
2 where count != 0
3 order by count desc;
```

CLASS	COUNT	TIME
undo header	29	966
undo block	2	0
data block	1	3

3 rows selected.

.....

- Query the V\$WAITSTAT view for the following classes: undo header, undo block, system undo header, and system undo block.

```
SQL> select class, count
2   from v$waitstat
3  where class in ('system undo header', 'system undo block', 'undo
header', 'undo block');
```

CLASS	COUNT
system undo header	0
system undo block	0
undo header	0
undo block	0



- 8** Does the V\$SYSTEM\_EVENT view show any waits related to rollback segments?

Query in V\$SYSTEM\_EVENT view for the “undo segment tx slot” entry.

```
SQL> select * from v$system_event
      2  where event = 'undo segment tx slot';
no rows selected
```

- 9** The tested transaction will be executed simultaneously by an average of ten users during the day. Every night, a batch job will load ten times the same volume as the ins\_ords.sql script. Prepare the script to create the appropriate rollback segments with appropriate storage parameters to optimize space usage.

*Do not run the script.*

- Create three rollback segments for the daytime activity and another one for batch activity.
- If there is too much extension, recreate the rollback RBSX with larger INITIAL and NEXT values.
- Increase the OPTIMAL value if too many shrinks occurred.
- Resize all rollback segments so that they are equally sized, except for the one dedicated for large batch transactions.
- Alter the batch execution script by inserting the command that forces the batch transaction to use the appropriate rollback segment.

```
alter rollback segment rbsx offline;
drop rollback segment rbsx;
create rollback segment rbsx_1
  tablespace rbs_test
  storage (initial 20k
           next 20k
           minextents 20
           optimal 400k);
alter rollback segment rbsx_1 online;
create rollback segment rbsx_2
  tablespace rbs_test
  storage (initial 20k
           next 20k
           minextents 20
           optimal 400k);
alter rollback segment rbsx_2 online;
create rollback segment rbsx_3
  tablespace rbs_test
  storage (initial 20k
           next 20k
```

```

        minextents 20
        optimal 400k);
alter rollback segment rbsx_3 online;
create rollback segment rbsx_big
    tablespace rbs_test
    storage (initial 200k
    next 200k
    minextents 20);
alter rollback segment rbsx_big online;

```

– Alter the batch execution by inserting the following command:

```
ALTER ROLLBACK SEGMENT rbsx_big SHRINK TO 400k;
```

- 10** As SCOTT, run the lab11\_1.sql script again, allocating the transaction to a specific rollback segment, and check that the transaction is working with the defined rollback segment.

Before running the lab11\_1.sql script, execute the following command:

```
SQL> SET TRANSACTION USE ROLLBACK SEGMENT rbsx;
```

Then join the V\$ROLLSTAT, V\$SESSION, and V\$TRANSACTION views to check that the transaction is using the defined rollback segment.

```
SQL> set transaction use rollback segment rbsx;
```

Transaction set.

```
SQL> @lab11_1
```

```
SQL> select s.username, rn.name
2  from v$session s, v$transaction t, v$rollstat r, v$rollname
rn
3  where s.saddr = t.ses_addr
4  and    t.xidusn = r.usn
5  and    r.usn = rn.usn;
```

USERNAME	NAME
SCOTT	RBSX

## Practice 12-1 Solutions

The objective of this practice is to use available diagnostics tools to monitor lock contention.

You will need to start three sessions, in separate windows. Log in twice as SCOTT (sessions 1 and 2) and once as SYS (session 3).

- 1 In session 1, update the salary of one of the employees in the S\_EMP table:

```
SQL> connect scott/tiger
Connected.
SQL> update s_emp
  2  set salary=salary*1.1
  3  where id=1;
1 row updated.
```

- 2 In session 3, check to see if any locks are being held.

Use the V\$LOCK or V\$LOCKED\_OBJECT view.

```
SQL> select SID, TYPE, ID1, ID2, LMODE, REQUEST from v$lock;
```

SID	TY	ID1	ID2	LMODE	REQUEST
2	MR	7	0	4	0
2	MR	1	0	4	0
2	MR	2	0	4	0
2	MR	3	0	4	0
2	MR	4	0	4	0
2	MR	5	0	4	0
2	MR	6	0	4	0
3	RT	1	0	6	0
5	TS	6	29360130	3	0
8	TX	65536	3856	6	0
8	TM	1821	0	3	0

Session 1 is the SID 8. It holds a TM table lock (LMODE = 3) and a TX row level lock (LMODE = 6) on object 1821.

```
SQL> select object_name
       2 from user_objects
       3 where object_id=1821;
```

OBJECT\_NAME

-----

S\_EMP

```
SQL> select * from v$locked_object;
```

XIDUSN	XIDSLOT	XIDSQN	OBJECT_ID	SESSION_ID
3	0	1605	1821	8
SCOTT	djeunot	93:208		3

**3** In session 2, drop the S\_EMP table. Does it work?

```
SQL> drop table s_emp;
drop table s_emp
```

\*

ERROR at line 1:

ORA-00054: resource busy and acquire with NOWAIT specified

**The DDL statement requires an exclusive table lock. It cannot obtain it, because session 1 already holds a row exclusive table lock on the S\_EMP table.**

**4** In session 2, update a different row in the S\_EMP table.

```
SQL> update s_emp
       2 set salary=salary*1.1
       3 where id=2;
```

1 row updated.

In session 3, check to see what kind of locks are being held.

Use the V\$LOCK or V\$LOCKED\_OBJECT view:

```
SQL> select SID, TYPE, ID1, LMODE, REQUEST from v$lock
       2 where request <> 0;
```

no rows selected.

This means that there are no waiting nor blocking sessions.

```
SQL> select * from v$locked_object;
```

XIDUSN	XIDSLOT	XIDSQN	OBJECT_ID	SESSION_ID
ORACLE_USERNAME	OS_USER_NAME	PROCESS	LOCKED_MODE	
3	0	1605	1821	8
SCOTT		djeunot	93:208	3
1	19	1948	1821	10
SCOTT		djeunot	74:416	3

- 5** Roll back the changes you made in session 2, and try to update the same row as session 1.

**In session 2:**

```
SQL> rollback;
Rollback complete.
SQL> update s_emp
2   set salary=salary*1.1
3   where id=1;
```

Check to see that the locks held in session 3, using any tool.

**Hint**

– Use V\$LOCK view:

```
SQL> select SID, TYPE, ID1, LMODE, REQUEST from v$lock
2   where SID in (8,12);
```

SID	TY	ID1	ID2	LMODE	REQUEST
8	TX	65536	3856	6	0
8	TM	1821	0	3	0
12	TM	1821	0	3	0
12	TX	65536	3856	0	6

The last row identified session SID 12 requesting a TX row level lock, held by session with SID 8 on resource 65536 and on object 1821.

```
SQL> select object_name
2   from dba_objects
3   where object_id=1821;
```

OBJECT\_NAME

S\_EMP

- 6** In session 3, terminate the session that is holding the lock and check the lock is released.

Use the ALTER SYSTEM KILL SESSION command:

```
SQL> select SID,serial#,username
```

```
2  from v$session
```

```
3  where type='USER';
```

```
      SID    SERIAL#  USERNAME
```

```
-----
```

```
        7         58    SYS
```

```
        8        124    SCOTT
```

```
       11        156    SYSTEM
```

```
       12         29    SCOTT
```

```
SQL> alter system kill session '8,124';
```

```
System altered.
```

## Practice 13-1 Solutions

The objective of this practice is to familiarize you with SQL statement execution plans and to interpret the formatted output of a trace file generated using SQL Trace and the formatted output generated by TKPROF. You will also learn how to manipulate statistics using the Oracle supplied DBMS\_STATS package.

- 1 Ensure the current directory is \$HOME/LABS.
- 2 Connect as SCOTT and create the PLAN\_TABLE under the SCOTT schema.

```
SQL> @$ORACLE_HOME/rdbms/admin/utlxplan
Table created.
```

- 3 Describe PLAN\_TABLE.

```
SQL> describe plan_table
```

Name	Null?	Type
STATEMENT_ID		VARCHAR2(30)
TIMESTAMP		DATE
REMARKS		VARCHAR2(80)
OPERATION		VARCHAR2(30)
OPTIONS		VARCHAR2(30)
OBJECT_NODE		VARCHAR2(128)
OBJECT_OWNER		VARCHAR2(30)
OBJECT_NAME		VARCHAR2(30)
OBJECT_INSTANCE		NUMBER(38)
OBJECT_TYPE		VARCHAR2(30)
OPTIMIZER		VARCHAR2(255)
SEARCH_COLUMNS		NUMBER
ID		NUMBER(38)
PARENT_ID		NUMBER(38)
POSITION		NUMBER(38)
COST		NUMBER(38)
CARDINALITY		NUMBER(38)
BYTES		NUMBER(38)
OTHER_TAG		VARCHAR2(255)
PARTITION_START		VARCHAR2(255)
PARTITION_STOP		VARCHAR2(255)
PARTITION_ID		NUMBER(38)
OTHER		LONG
DISTRIBUTION		VARCHAR2(30)

- 4** Use SQL\*Plus AUTOTRACE to look at the execution plan worked out by the optimizer for the SQL statement in the lab13\_1.sql script.

```
SQL> set autotrace traceonly explain
```

```
SQL> @lab13_1
```

```
SQL> select e.last_name
       2  from   s_emp e
       3  where  e.dept_id = 10;
```

Execution Plan

```
-----
0      SELECT STATEMENT Optimizer=CHOOSE
1  0    TABLE ACCESS (FULL) OF 'S_EMP'
```

```
SQL> set autotrace off
```

- 5** Use SQL Trace and TKPROF to look at the performance of the query executed by the lab13\_2.sql script.

- a** Set up your session in trace mode.

```
SQL> alter session set sql_trace = true;
```

- b** Run the lab13\_2.sql script.

```
SQL> @lab13_2
```

```
SQL> select e.last_name
       2  ,      d.name
       3  from   s_dept d
       4  ,      s_emp e
       5  where  d.id = e.dept_id
       6  and    d.id = 10;
```

```
LAST_NAME      NAME
```

```
-----
```

Quick-To-See	Finance
Velasquez	Finance
Ngao	Finance
...	



- c** Exit from SQL\*Plus and format your trace file using TKPROF. Use the options SYS=NO and EXPLAIN.

```
$ cd $HOME/UDUMP
$ ls -l
total 20
-rw-rw---- 1 oracle dba 9669 July 1 17:05 ora_tun8_08_20389.trc
$ tkprof ora_tun8_08_20389.trc myfile.txt explain=scott/tiger sys=no
TKPROF: Release 8.1.5.0.0 - Production on Thu Jul 1 17:10:56 1999
(c) Copyright 1999 Oracle Corporation. All rights reserved.
$ ls
myfile.txt          ora_tun8_08_20389.trc
$ more myfile.txt
TKPROF: Release 8.1.5.0.0 - Production on Thu Jul 1 17:10:56 1999
(c) Copyright 1999 Oracle Corporation. All rights reserved.
Trace file: ora_tun8_08_20389.trc
Sort options: default
*****
count      = number of times OCI procedure was executed
cpu        = cpu time in seconds executing
elapsed    = elapsed time in seconds executing
disk       = number of physical reads of buffers from disk
query      = number of buffers gotten for consistent read
current    = number of buffers gotten in current mode (usually for
update)
rows       = number of rows processed by the fetch or execute call
*****
alter session set sql_trace=true
call      count  cpu    elapsed    disk  query  current  rows
-----
Parse          0  0.00      0.00      0      0        0      0
Execute        1  0.00      0.00      0      0        0      0
Fetch          0  0.00      0.00      0      0        0      0
-----
total          1  0.00      0.00      0      0        0      0

Misses in library cache during parse: 0
Misses in library cache during execute: 1
Optimizer goal: CHOOSE
Parsing user id: 26 (SCOTT)
```

## Appendix C: Practice Solutions

\*\*\*\*\*

```
BEGIN DBMS_APPLICATION_INFO.SET_MODULE(:1,NULL); END;
```

call	count	cpu	elapsed	disk	query	current	rows
Parse	2	0.00	0.00	0	0	0	0
Execute	2	0.00	0.00	0	0	0	2
Fetch	0	0.00	0.00	0	0	0	0
total	4	0.00	0.00	0	0	0	2

Misses in library cache during parse: 2

Misses in library cache during execute: 2

Optimizer goal: CHOOSE

Parsing user id: 26 (SCOTT)

\*\*\*\*\*

```
select e.last_name
,      d.name
from   s_dept d
,      s_emp e
where  d.id = e.dept_id
and    d.id = 10
```

call	count	cpu	elapsed	disk	query	current	rows
Parse	1	0.00	0.00	0	0	0	0
Execute	1	0.00	0.00	0	0	0	0
Fetch	3	0.00	0.00	0	14	4	21
total	5	0.00	0.00	0	14	4	21

Misses in library cache during parse: 1

Optimizer goal: CHOOSE

Parsing user id: 26 (SCOTT)

Rows Row Source Operation

21	MERGE JOIN
2	TABLE ACCESS BY INDEX ROWID S_DEPT
2	INDEX UNIQUE SCAN (object id 12389)
21	FILTER
265	TABLE ACCESS FULL S_EMP

Rows Execution Plan

```

-----
      0  SELECT STATEMENT      GOAL: CHOOSE
    21  MERGE JOIN
      2    TABLE ACCESS (BY INDEX ROWID) OF 'S_DEPT'
      2      INDEX (UNIQUE SCAN) OF 'S_DEPT_ID_PK' (UNIQUE)
    21  FILTER
    265    TABLE ACCESS (FULL) OF 'S_EMP'

*****
OVERALL TOTALS FOR ALL NON-RECURSIVE STATEMENTS

call      count  cpu   elapsed   disk  query  current  rows
-----
Parse          3  0.00     0.00      0     0         0     0
Execute        4  0.00     0.00      0     0         0     2
Fetch          3  0.00     0.00      0    14         4    21
-----
total         10  0.00     0.00      0    14         4    23

Misses in library cache during parse: 3
Misses in library cache during execute: 3

OVERALL TOTALS FOR ALL RECURSIVE STATEMENTS

call      count  cpu   elapsed   disk  query  current  rows
-----
Parse          0  0.00     0.00      0     0         0     0
Execute        0  0.00     0.00      0     0         0     0
Fetch          0  0.00     0.00      0     0         0     0
-----
total          0  0.00     0.00      0     0         0     0

Misses in library cache during parse: 0

      4  user  SQL statements in session.
      0  internal SQL statements in session.
      4  SQL statements in session.
      1  statement EXPLAINED in this session.

*****
Trace file: ora_tun8_08_20389.trc
Trace file compatibility: 7.03.02
Sort options: default

```

```

1 session in tracefile.
4 user SQL statements in trace file.
0 internal SQL statements in trace file.
4 SQL statements in trace file.
3 unique SQL statements in trace file.
1 SQL statements EXPLAINED using schema:
  SCOTT.prof$plan_table
    Default table was used.
    Table was created.
    Table was dropped.
61 lines in trace file.

```

- d** You should only note down the CPU, current, and query figures for the fetch phase. You will not see a drastic change in numbers. The objective is to become familiar and comfortable with running TKPROF and SQL Trace.

Results:

Query	CPU (secs)	Query (blocks)	Current (blocks)
select e.last_name, d.name from s_dept d, s_emp e where d.id = e.dept_id and d.id = 10;			

- 6** Gather statistics for all objects under the SCOTT schema while saving the current statistics then restore the original statistics.

- a** Connect as SCOTT and create a table to hold statistics in that schema.

```

$ sqlplus scott/tiger
SQL> execute dbms_stats.create_stat_table('SCOTT','MY_STATS');
PL/SQL procedure successfully completed.

```

- b** Save the current schema statistics into your local statistics table.

```

SQL> execute dbms_stats.export_schema_stats('SCOTT','MY_STATS');
PL/SQL procedure successfully completed.

```

- c** Analyze all objects under the SCOTT schema.

```

SQL> execute dbms_stats.gather_schema_stats('SCOTT');
PL/SQL procedure successfully completed.

```

- d** Remove all schema statistics from the dictionary and restore the original statistics you saved in step b.

```

SQL> execute dbms_stats.delete_schema_stats('SCOTT');
PL/SQL procedure successfully completed.
SQL> execute dbms_stats.import_schema_stats('SCOTT','MY_STATS');

```

PL/SQL procedure successfully completed.

## Practice 14-1 Solutions

- 1** Connect as user SYS and create a consumer group called ONLINE\_USERS.

```
SQL> EXEC DBMS_RESOURCE_MANAGER.CREATE_PENDING_AREA();
```

PL/SQL procedure successfully completed.

```
SQL> EXEC DBMS_RESOURCE_MANAGER. -  
> CREATE_CONSUMER_GROUP('ONLINE_USERS', 'Online users');
```

PL/SQL procedure successfully completed.

- 2** Create a resource plan called DAYTIME with the following characteristics:

- Users in the SYS\_GROUP get as much CPU time as they require.
- Users in the ONLINE\_USERS group get 80% of the remaining resources.
- Users in the OTHER\_GROUPS get 20% of the remaining resources.

There is no parallel query processing occurring.

This is the only plan that will be created, so complete the process of creating a plan.

```
SQL> EXEC DBMS_RESOURCE_MANAGER. -  
> CREATE_PLAN('DAYTIME', 'Daytime plan');
```

PL/SQL procedure successfully completed.

```
SQL> EXEC DBMS_RESOURCE_MANAGER. -  
> CREATE_PLAN_DIRECTIVE -  
> ('DAYTIME', 'SYS_GROUP', 'Day rules for DBAs', 100);
```

PL/SQL procedure successfully completed.

```
SQL> EXEC DBMS_RESOURCE_MANAGER. -  
> CREATE_PLAN_DIRECTIVE -  
> ('DAYTIME', 'ONLINE_USERS', 'Day rules for online', NULL, 80);
```

PL/SQL procedure successfully completed.

```
SQL> EXEC DBMS_RESOURCE_MANAGER. -  
> CREATE_PLAN_DIRECTIVE -  
> ('DAYTIME', 'OTHER_GROUPS', 'Day rules for others', NULL, 20);
```

PL/SQL procedure successfully completed.

```
SQL> EXEC DBMS_RESOURCE_MANAGER.VALIDATE_PENDING_AREA();
```

PL/SQL procedure successfully completed.

```
SQL> EXEC DBMS_RESOURCE_MANAGER.SUBMIT_PENDING_AREA();
```

PL/SQL procedure successfully completed.

### 3 Make ONLINE\_USERS the default consumer group for User01.

First, grant User01 the privilege to switch to this group. Also, User01 needs the EXECUTE privilege on package DBMS\_SESSION.

```
SQL> EXEC DBMS_RESOURCE_MANAGER_PRIVS. -  
> GRANT_SWITCH_CONSUMER_GROUP('USER01', 'ONLINE_USERS', FALSE);
```

PL/SQL procedure successfully completed.

```
SQL> EXEC DBMS_RESOURCE_MANAGER. -  
> SET_INITIAL_CONSUMER_GROUP('USER01', 'ONLINE_USERS');
```

PL/SQL procedure successfully completed.

```
SQL> GRANT EXECUTE ON DBMS_SESSION TO USER01;
```

Grant succeeded.

**4** Give User01 the capability to switch to the SYS\_GROUP.

```
SQL> EXEC DBMS_RESOURCE_MANAGER_PRIVS. -  
> GRANT_SWITCH_CONSUMER_GROUP('USER01', 'SYS_GROUP', FALSE);
```

PL/SQL procedure successfully completed.

**5** Activate DAYTIME as the current resource plan group.

```
SQL> ALTER SYSTEM SET RESOURCE_MANAGER_PLAN=daytime;
```

System altered.

**6** Start a second SQL\*Plus session and log on as User01. Display the current consumer group for User01's session.

You may have to log on as SYS to display the data from V\$SESSION.

```
SQL> SELECT resource_consumer_group  
2      FROM v$session  
3      WHERE username = 'USER01';
```

RESOURCE\_CONSUMER\_GROUP

-----

ONLINE\_USERS

**7** As User01, switch to the SYS\_GROUP.

Use a PL/SQL block, because the procedure has an OUT mode variable.

```
SQL> DECLARE  
2      v_old_group VARCHAR2(30);  
3  BEGIN  
4      DBMS_SESSION.SWITCH_CURRENT_CONSUMER_GROUP  
5          ('SYS_GROUP', v_old_group, FALSE);  
6  END;  
7  /
```

PL/SQL procedure successfully completed.



**8** Redisplay User01's current consumer group.

```
SQL> SELECT resource_consumer_group
      2     FROM v$session
      3     WHERE username = 'USER01';
```

```
RESOURCE_CONSUMER_GROUP
```

```
-----
```

```
SYS_GROUP
```



---

D

---

**Redundant Arrays of  
Inexpensive Disks  
Technology (RAID)**

## System Hardware Configuration

### Storage Subsystem Detail

Storage subsystem performance is one of the most important aspects of tuning an Oracle database server for optimal performance. The architecture and design of the storage subsystem must therefore be considered early in the system design process. In performing the storage subsystem design, system requirements such as the required volume of online transaction data, peak transactions per second load, and system availability are transformed into specific storage subsystem design requirements for storage capacity, peak sustainable I/Os per second, and fault tolerance. Values for design parameters in the selected technology are then chosen to meet these specific requirements.

Modern storage systems offer great flexibility in meeting a wide range of design criteria; technologies such as striping, mirroring, and other fault-tolerant RAID configurations provide the ability to meet these design requirements. Matching the right technology with application I/O characteristics is key to achieving the promised performance and fault tolerance levels; conversely, using the wrong technology for a specific I/O characteristic can lead to I/O bottlenecks and degraded response times. In this section, key parameters in the design of the storage subsystem are described, as well as how they relate to the performance of an Oracle database.

### Storage Subsystem Design Parameters and Oracle

When designing a storage subsystem, the available design parameters are weighed against each other until a design solution is achieved that meets or exceeds all design requirements. In the context of an Oracle database server, certain measures are available to specify these requirements. These measures can be categorized under performance, availability, and cost.

#### Performance

- Random read performance: Important for Oracle indexed or hash-based queries and rollback segment reads
- Random write performance: Important for Oracle DBW $n$  writes; heavy in an OLTP environment, light in a data warehouse
- Sequential read performance: Backups, Oracle full table scans, index creations, parallel queries, temporary segment reads, and recovery from archived redo log files
- Sequential write performance: Oracle LGWR writes, temporary segment writes, direct-path loader writes, tablespace creations
- Impact of concurrency

## Storage Subsystem Design Parameters and Oracle (continued)

### Availability

- Outage frequency: Expected number of occurrences of a possible outage per unit of time specified in mean time to failure (MTTF)
- Outage duration: The mean time to repair (MTTR) for a given outage event
- Performance degradation during outage: Whether a disk configuration provides service during a fault, and if so, at what level

### Cost

- Acquisition cost: The cost of purchasing, installing, and configuring the storage subsystem
- Operational cost: The cost of running and maintaining the system to meet the system availability and service level requirements

The Redundant Arrays of Inexpensive Disks (RAID) technologies have been developed for nonmainframe, open system solutions. RAID provides low-cost fault tolerance and improved performance. Several levels of RAID are available and may be mixed within one storage subsystem design. Each level of RAID can be categorized against the measures listed above and its impact on Oracle database performance. Some levels of RAID configurations have been available for many years under different names. Key parameters when configuring a RAID system are:

- Array size: The number of drives in the array
- Disk size: The size of each disk
- Stripe size: The size of an I/O chunk, written to or read from a contiguous location on a disk (*Striping* allows data files to be interleaved and spread across the disks in an array in an attempt to parallelize file I/O.)

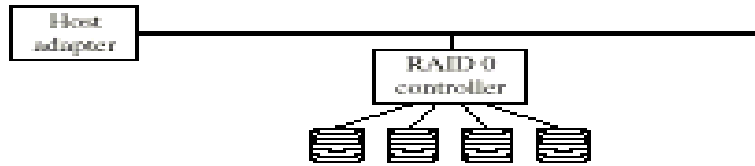
## Storage Subsystem Design Parameters and Oracle (continued)

**Cost (continued)** The throughput of a storage subsystem, expressed in I/Os per second, determines how many transactions can be processed by the subsystem before queuing delays begin to occur. If the I/Os-per-second requirement of the application is known, broken down into reads per transaction, writes per transaction, and transactions per second, you can determine whether a particular RAID configuration will support the transaction rate applied by an application. To calculate throughput, or total sustainable I/Os-per-second load that a RAID array can support, simply multiply the number of drives in the array times the sustainable I/Os per second of one drive, currently in the approximate range of 35 to 50 I/Os per second. Different RAID configurations, however, add to the I/O load on a disk array applied by the application in order to provide the fault tolerance function. Once the total I/O load on the array is known, the number of drives required to support the load can be found. Simply divide the total

I/Os per second load by the random I/O throughput rating for the selected drive.

The sections below briefly describe each RAID level and some of its characteristics. For each level, an equation is provided to calculate the total I/Os per second load on a RAID array, to illustrate the impact of the RAID configuration on the array's capacity. Also provided is an equation for calculating the size of the disk drives required for an array.

## RAID Level 0, Nonredundant Striping



RAID 0 refers to simple data striping of multiple disks into a single logical volume, and has no fault tolerance. When properly configured, it provides excellent response times for high concurrency random I/O and excellent throughput for low concurrency sequential I/O. Selection of the array and stripe sizes requires careful consideration in order to achieve the promised throughput. For RAID 0, the total I/Os-per-second load generated against the array is calculated directly from the application load, because there is no fault tolerance in this configuration:

```
total I/O per second load on array = (reads/transaction + writes/transaction) * transactions/second
```

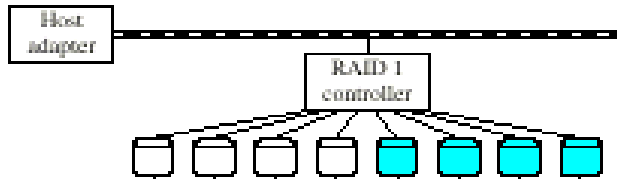
The size of each drive in the array can be calculated from the online volume requirements as follows:

```
drive size = [total space required by application / number of drives in array] Rounded up to next drive size.
```

Below is a summary of RAID 0 characteristics:

- Random read performance: Excellent under all concurrency levels if each I/O request fits within a single striping segment
- Random write performance: Same as random read performance
- Sequential read performance: Excellent with fine-grained striping at low concurrency levels
- Sequential write performance: Same as sequential read performance
- Outage frequency: Poor; any single disk failure will cause application outage
- Outage duration: Poor; the duration of a RAID 0 outage is the time required to detect the failure, replace the disk drive, and perform Oracle media recovery
- Performance degradation during outage: Poor; any disk failure causes all applications requiring use of the array to crash
- Acquisition cost: Excellent, because there is no redundancy; you buy only enough for storage and I/Os per second requirements
- Operational cost: Fair to poor; frequent media recoveries increase operational costs and may outweigh the acquisition cost advantage

## RAID Level 1, Mirroring



RAID Level 1, or disk mirroring, provides the best fault tolerance of any of the RAID configurations. Each disk drive is backed up by an exact copy of itself on an identical drive. A storage subsystem of mirrored drives can continue at full performance with a multiple disk failure as long as no two drives in a mirrored pair have failed. The total I/Os per load applied to a mirrored pair is calculated as follows:

```
total I/O per second load on array = (reads/transaction + 2*writes/transaction) * transactions/second
```

Note the two multiplier of the writes/transaction factor. This is due to the fact that each write request by an application to a mirrored pair actually results in two writes, one to the primary disk and one to the backup disk. The size of the drive required is:

```
drive size = [total space required by application / number of drives in array/2] Rounded up to next drive size.
```

In the simplest RAID 1 configuration, the number of drives in the array is two: the primary drive and its backup. The definition of RAID 1, however, includes the ability to expand the array in units of two drives to achieve a striped and mirrored configuration. Striping occurs in an array of four or more disks. Some industry literature (for example, Millsap, 1996) refers to striped and mirrored configurations as RAID 0 + 1. The Compaq hardware used as an example configuration in this document supports both configurations. Compaq uses only the RAID 1 term to describe all 100% mirrored configurations in arrays of even-numbered disks. Because the performance of a simple two-drive RAID 1 pair is somewhat different from a striped and mirrored array, the figures for striped and mirrored are presented separately under the RAID 0 + 1 section.

Below is a summary of characteristics of the two-disk array RAID configuration:

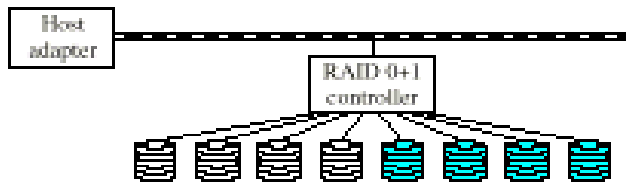
- Random read performance: Good; if the implementation uses read-optimized RAID 1 controllers, which read from the drive with the smallest I/O setup cost, then slightly better than an independent disk
- Random write performance: Good (Application write requests are multiplied by two, because the data must be written to two disks. Thus, some of the I/Os-per-second capacity of the two drives is used up by the mirroring function.)



**RAID Level 1, Mirroring (continued)**

- Sequential read performance: Fair; throughput is limited to the speed of one disk
- Sequential write performance: Fair; same factors as are influencing the random write performance
- Outage frequency: Excellent
- Outage duration: Excellent; for “hot swappable” drives, no application outage is encountered by a single failure
- Performance degradation during outage: Excellent; there is no degradation during a disk outage (After replacing of the failed drive, the resilvering operation that takes place when the failed disk is replaced will consume some of the available I/Os per second capacity.)
- Acquisition cost: Poor; each RAID 1 pair requires two drives to achieve the storage capacity of one
- Operational cost: Fair; increased complexity of the configuration leads to higher training costs and costs to develop custom software to integrate the mirroring procedures into scheduled maintenance operations

## RAID Level 0+1, Striping and Mirroring

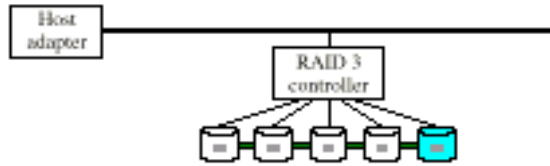


As noted in the previous section, the striped and mirrored configuration is an expansion of the RAID 1 configuration from a simple mirrored pair to an array of even-numbered drives. This configuration offers the performance benefits of RAID 0 striping and the fault tolerance of simple RAID 1 mirroring. The striped and mirrored configuration is especially valuable for Oracle data files holding files with high write rates, such as table data files and online and archived redo log files. Unfortunately, it also presents the high costs of simple RAID 1. The equations for the total I/Os per second and disk drive size calculations for RAID 0 + 1 are identical to those presented for RAID 1 above. The Compaq SMART array controller used in the example configuration supports RAID 0 + 1 (RAID 1 in Compaq terminology) in arrays up to 14 drives, providing the effective storage of 7 drives.

Below is a summary of characteristics of RAID 0 + 1 storage arrays:

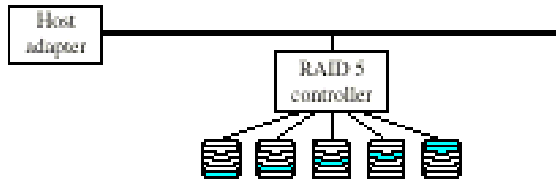
- Random read performance: Excellent under all concurrency levels if each I/O requests fits within a single striping segment (Using a stripe size that is too small can cause dramatic performance breakdown at high concurrency levels.)
- Random write performance: Good (Application write requests are multiplied by two because the data must be written to two disks. Thus, some of the I/Os-per-second capacity of the two drives is used up by the mirroring function.)
- Sequential read performance: Excellent under all concurrency levels if each I/O request fits within a single striping segment
- Sequential write performance: Good
- Outage frequency: Excellent; same as RAID 1
- Outage duration: Excellent; same as RAID 1
- Performance degradation during outage: Excellent; there is no degradation during a disk outage (The resilvering operation that takes place when the failed disk is replaced will consume a significant amount of the available I/Os-per-second capacity.)
- Acquisition cost: Poor; same as RAID 1
- Operational cost: Fair; same as RAID 1

## RAID Level 3, Bit Interleaved Parity



In the RAID 3 configuration, disks are organized into arrays in which one disk is dedicated to storage of parity data for the other drives in the array. The stripe size in RAID 3 is 1 bit. This enables recovery time to be minimized, because data can be reconstructed with a simple exclusive-OR operation. However, using a stripe size of 1 bit reduces I/O performance. RAID 3 is not recommended for storing any Oracle database files. Also, RAID 3 is not supported by the Compaq SMART array controllers.

## RAID Level 5, Block-Interleaved with Distributed Parity



RAID 5 is similar to RAID 3, except that RAID 5 striping segment sizes are configurable, and RAID 5 distributes parity across all the disks in an array. A RAID 5 striping segment contains either data or parity.

Battery-backed cache greatly reduces the impact of this overhead for write calls, but its effectiveness is implementation-dependent. Large write-intensive batch jobs generally fill the cache quickly, reducing its ability to offset the write-performance penalty inherent in the RAID 5 definition.

The total I/Os per second load applied to a RAID 5 array is calculated as follows:

```
total I/O per second load on array = (reads/transaction + 4*writes/transaction) * transactions/second
```

The writes/transaction figure is multiplied by four because the parity data must be written in a six-step process:

- 1 Read the data drive containing the old value of the data to be overwritten. This requires one I/O.
- 2 Read the parity drive. This requires one I/O.
- 3 Subtract the contribution of the old data from the parity value.
- 4 Add the contribution of the new data to the parity value.
- 5 Write the new value of the parity requiring one I/O.
- 6 Write the new data value to the data drive. This requires one I/O.

Summing up all I/Os in this process yields four I/Os required for each write requested by the application. This is the main reason that RAID 5 is not recommended for storing files with a high I/O performance requirement; the 4 multiplier reduces the effective I/Os-per-second capacity of the array.

The size of the drive required is:

```
drive size = [total space required by application / (total number drives - number of arrays)] Rounded up to next drive size.
```

**RAID Level 5, Block-Interleaved with Distributed Parity (continued)**

Note that the figure “number of arrays” is used to account for the space of one drive per array consumed by the parity data. If it is necessary to exceed the maximum recommended array size to meet the I/Os-per-second performance requirement, then multiple arrays are required.

Below is a summary of the characteristics of RAID 5 storage arrays:

- Random read performance: Excellent under all concurrency levels if each I/O requests fits within a single striping segment (Using a stripe size that is too small can cause dramatic performance breakdown at high concurrency levels.)
- Random write performance: Poor; worst at high concurrency levels (The read-modify-write cycle requirement of RAID 5 parity implementation reduces the effective throughput or I/Os-per-second capacity of the array, especially for heavy write I/O files. It should be noted, however, that under light load, response time is not degraded from that provided by a faster array configuration such as RAID 0. This is due to the asynchronous write capabilities provided by most array controllers. With asynchronous write, the application does not have to wait until the storage subsystem has completed the read-modify-write cycle before continuing. Instead, the controller buffers the data in battery-backed RAM, signals the application that the write has completed, then completes the write to disk. (This buffering does nothing to increase throughput, however.)
- Sequential read performance: Excellent under high concurrency levels if each I/O request fits within a single striping segment; also excellent with fine grain striping under low concurrency levels
- Sequential write performance: Fair for low concurrency levels, poor for high concurrency levels (See random write performance.)
- Outage frequency: Good; can withstand the loss of any single disk in a given array without incurring an application outage (Multiple simultaneous disk failures causes application outage. The possibility of multiple simultaneous failures increase as the size of the array increases.)
- Outage duration: Good; a single disk failure causes no application outage
- Performance degradation during outage: Fair; there is no degradation for reads and writes to or from surviving drives in the array (Reads and writes to a failed drive incur a high performance penalty, requiring data from all surviving drives in the array to be read. Reconstruction of the failed drive’s data also degrades performance.)
- Acquisition cost: Fair (If storage capacity were the only factor, the cost would be  $g/(g-1)$  times the cost of the equivalent RAID 0 capacity, where  $g$  is the number of disks in the array. However, when factoring I/Os-per-second performance requirement, the cost can meet or exceed the cost of a RAID 0 + 1 implementation.)
- Operational cost: Fair (Training is required to configure striped disk arrays for optimal performance.)

## Ranking of RAID Levels Against Oracle File Types

The following table provides relative rankings for RAID configurations for specific Oracle file types. The rankings range from 1 (Best) to 5 (Worst). Adapted from Millsap, 1996, page 13.

	<b>None</b>	<b>0</b>	<b>1</b>	<b>0 + 1</b>	<b>3</b>	<b>5</b>
Control file performance	2	1	2	1	5	3
Redo log file performance	4	1	5	1	2	3
System tablespace performance	2	1	2	1	5	3
Sort segment performance	4	1	5	1	2	3
Rollback segment performance	2	1	2	1	5	5
Indexed read-only data files	2	1	2	1	5	1
Sequential read-only data files	4	1	5	1	2	3
DBWn intensive data files	1	1	2	1	5	5
Direct load-intensive data files	4	5	1	1	2	2
Data protection	4	5	1	1	2	2
Acquisition and operating costs	1	1	5	5	3	3

---

E

---

## **Dictionary and Dynamic Performance Views**

## Dictionary and Dynamic Performance Views

This appendix contains descriptions of data dictionary and dynamic views. To see the data dictionary and dynamic views available to you, query the view `DICTIONARY`.

The views that exist only in release 8.0 and not in release 8.1, and vice versa, will be mentioned. The views that have different columns in release 8.0 and release 8.1 will also be mentioned.

Refer to the *Oracle8i Reference Release 8.1.5 A67790-01* to get the full list of the dictionary and dynamic views and to the *Oracle8i Migration Release 8.1.5 A67774-01* to get the full list of obsolete views and views whose name or structure changed between release 8.0 and 8.1.



## Data Dictionary Views

The following is an alphabetical reference of the data dictionary views accessible to all users of an Oracle server. Most views can be accessed by any user with the CREATE SESSION privilege.

The data dictionary views that begin with DBA\_ are restricted. These views can be accessed only by users with the SELECT ANY TABLE privilege. This privilege is assigned to the DBA role when the system is initially installed.

### DBA\_ALL\_TABLES

This view displays descriptions of all tables (object tables and relational tables) in the database.

Column	Description
OWNER	Owner of the table
TABLE_NAME	Name of the table
TABLESPACE_NAME	Name of the tablespace containing the table
CLUSTER_NAME	Name of the cluster, if any, to which the table belongs
IOT_NAME	Name of the index organized table, if any, to which the overflow entry belongs
PCT_FREE	Minimum percentage of free space in a block
PCT_USED	Minimum percentage of used space in a block
INI_TRANS	Initial number of transactions
MAX_TRANS	Maximum number of transactions
INITIAL_EXTENT	Size of the initial extent in bytes
NEXT_EXTENT	Size of secondary extents in bytes
MIN_EXTENTS	Minimum number of extents allowed in the segment
MAX_EXTENTS	Maximum number of extents allowed in the segment
PCT_INCREASE	Percentage increase in extent size
FREELISTS	Number of process free lists allocated in this segment
FREELIST_GROUPS	Number of freelist groups allocated to this segment
LOGGING	Logging attribute
BACKED_UP	Has table been backed up since last modification?
NUM_ROWS	Number of rows in the table
BLOCKS	Number of used blocks in the table
EMPTY_BLOCKS	Number of empty (never used) blocks in the table
AVG_SPACE	Average available free space in the table
CHAIN_CNT	Number of chained rows in the table
AVG_ROW_LEN	Average row length, including row overhead

Column	Description
AVG_SPACE_FREELIST_BLOCKS	Average free space of all blocks on a freelist
NUM_FREELIST_BLOCKS	Number of blocks on the freelist
DEGREE	Number of threads per instance for scanning the table
INSTANCES	Number of instances across which the table is to be scanned
CACHE	Whether the table is to be cached in the buffer cache
TABLE_LOCK	Whether table locking is enabled or disabled
SAMPLE_SIZE	Sample size used in analyzing this table
LAST_ANALYZED	Date of the most recent time this table was analyzed
PARTITIONED	Is this table partitioned? YES or NO
IOT_TYPE	If an index organized table, then IOT_TYPE is IOT or IOT_OVERFLOW else NULL
TABLE_TYPE_OWNER	Owner of the type of the table if the table is a typed table
TABLE_TYPE	Type of the table if the table is a typed table
TEMPORARY	Can the current session only see data that it place in this object itself?
SECONDARY only in release 8.1	Is the table object created as part of icreate for domain indexes?
NESTED	Is the table a nested table?
BUFFER_POOL	The default buffer pool to be used for table blocks
ROW_MOVEMENT only in release 8.1	The movement of the row
GLOBAL_STATS only in release 8.1	Are the statistics calculated without merging underlying partitions?
USER_STATS only in release 8.1	Were the statistics entered directly by the user?
DURATION only in release 8.1	If temporary table, then duration is SYS\$SESSION or SYS\$TRANSACTION else NULL
SKIP_CORRUPT only in release 8.1	Whether skip corrupt blocks is enabled or disabled

**DBA\_CLUSTERS**

This view contains description of all clusters in the database.

Column	Description
OWNER	Owner of the cluster
CLUSTER_NAME	Name of the cluster
TABLESPACE_NAME	Name of the tablespace containing the cluster
PCT_FREE	Minimum percentage of free space in a block
PCT_USED	Minimum percentage of used space in a block
KEY_SIZE	Estimated size of cluster key plus associated rows
INI_TRANS	Initial number of transactions
MAX_TRANS	Maximum number of transactions
INITIAL_EXTENT	Size of the initial extent in bytes
NEXT_EXTENT	Size of secondary extents in bytes
MIN_EXTENTS	Minimum number of extents allowed in the segment
MAX_EXTENTS	Maximum number of extents allowed in the segment
PCT_INCREASE	Percentage increase in extent size
FREELISTS	Number of process free lists allocated in this segment
FREELIST_GROUPS	Number of free list groups allocated to this segment
AVG_BLOCKS_PER_KEY	Average number of blocks containing rows with a given cluster key
CLUSTER_TYPE	Type of cluster: B-tree index or hash
FUNCTION	If a hash cluster, the hash function
HASHKEYS	If a hash cluster, the number of hash keys (hash buckets)
DEGREE	Number of threads per instance for scanning the table
INSTANCES	Number of instances across which the table is to be scanned
CACHE	Whether the table is to be cached in the buffer cache
BUFFER_POOL	Name of the default buffer pool for the appropriate object
SINGLE_TABLE	Y if the cluster is single table; N if not only in release 8.1

**DBA\_EXTENTS**

This view lists the extents comprising all segments in the database.

Column	Description
OWNER	Owner of the segment associated with the extent
SEGMENT_NAME	Name of the segment associated with the extent
SEGMENT_TYPE	Type of the segment: INDEX PARTITION, TABLE PARTITION
TABLESPACE_NAME	Name of the tablespace containing the extent
EXTENT_ID	Extent number in the segment
FILE_ID	Name of the file containing the extent
BLOCK_ID	Starting block number of the extent
BYTES	Size of the extent in bytes
BLOCKS	Size of the extent in Oracle blocks
RELATIVE_FNO	Relative file number of the first extent block
PARTITION_NAME	Object Partition Name (Set to NULL for nonpartitioned objects)

**DBA\_INDEXES**

This view contains descriptions for all indexes in the database. To gather statistics for this view, use the SQL command ANALYZE. This view supports parallel partitioned index scans.

Column	Description
OWNER	Username of the owner of the index
INDEX_NAME	Name of the index
INDEX_TYPE	Type of index
TABLE_OWNER	Owner of the indexed object
TABLE_NAME	Name of the indexed object
TABLE_TYPE	Type of the indexed object
UNIQUENESS	Uniqueness status of the index: UNIQUE or NONUNIQUE
COMPRESSION only in release 8.1	Enabled or disabled
PREFIX_LENGTH only in release 8.1	Number of columns in the prefix of the key used for compression
TABLESPACE_NAME	Name of the tablespace containing the index
INI_TRANS	Initial number of transactions
MAX_TRANS	Maximum number of transactions

Column	Description
INITIAL_EXTENT	Size of initial extent
NEXT_EXTENT	Size of secondary extents
MIN_EXTENTS	Minimum number of extents allowed in the segment
MAX_EXTENTS	Maximum number of extents allowed in the segment
PCT_INCREASE	Percentage increase in extent size
PCT_THRESHOLD	Threshold percentage of block space allowed per index entry
INCLUDE_COLUMN	User column ID for last column to be included in index organized table top index
FREELISTS	Number of process freelists allocated to this segment
FREELIST_GROUPS	Number of free list groups allocated to this segment
PCT_FREE	Minimum percentage of free space in a block
LOGGING	Logging attribute
BLEVEL	B-Tree level: depth of the index from its root block to its leaf blocks (A depth of 0 indicates that the root block and leaf block are the same)
LEAF_BLOCKS	The number of leaf blocks in the index
DISTINCT_KEYS	The number of distinct keys in the index
AVG_LEAF_BLOCKS_PER_KEY	The average number of leaf blocks per key
AVG_DATA_BLOCKS_PER_KEY	The average number of data blocks per key
CLUSTERING_FACTOR	A measurement of the amount of (dis)order of the table that this index is for
STATUS	Whether index is in Direct Load State
DOMIDX_STATUS only in release 8.1	Reflects the status of the domain index (A NULL value means that the specified index is not a domain index. A Value of VALID means that the index is a domain index and the index does not have any errors. If the value of this column is IDXTYP_INVLD it means that the index type corresponding to this domain index is invalid)
DOMIDX_OPSTATUS only in release 8.1	Reflects the status of an operation that was performed on the domain index (A value of NULL indicates that the specified index is not a domain index. A value of VALID specifies that the index does not have any errors. A value of FAILED indicates that the operation that was performed on the domain index failed with an error.)

Column	Description
FUNCIDX_STATUS only in release 8.1	A value of NULL indicates a non function-based index (ENABLED indicates the function-based index is enabled. DISABLED indicates the function-based index is disabled.)
NUM_ROWS	Number of rows in this index
SAMPLE_SIZE	Size of the sample used to analyze this index
LAST_ANALYZED	Time stamp indicating when this index was last analyzed
DEGREE	Number of threads per instance for scanning the index. NULL: (if PARTITIONED=NO)
INSTANCES	Number of instances across which the indexes are to be scanned. NULL if PARTITIONED=NO.
PARTITIONED	Indicates whether this index is partitioned. Set to YES if it is partitioned
TEMPORARY	Can the current session only see data that it place in this object itself?
GENERATED	Was the name of this index system generated?
SECONDARY only in 8.1	Is the index object created as part of icreate for domain indexes?
BUFFER_POOL	Name of the default buffer pool for the appropriate object
USER_STATS only in 8.1	Were the statistics entered directly by the user?
DURATION only in 8.1	The duration
PCT_DIRECT_ACCESS only in 8.1	If index is on IOT, then this is the percentage of rows with valid estimate

## DBA\_OBJECT\_TABLES

This view displays descriptions of all object tables in the database.

Column	Description
OWNER	Owner of the table
TABLE_NAME	Name of the table
TABLESPACE_NAME	Name of the tablespace containing the table
CLUSTER_NAME	Name of the cluster, if any, to which the table belongs
IOT_NAME	Name of the index organized table, if any, to which the overflow entry belongs
PCT_FREE	Minimum percentage of free space in a block
PCT_USED	Minimum percentage of used space in a block

Column	Description
INI_TRANS	Initial number of transactions
MAX_TRANS	Maximum number of transactions
INITIAL_EXTENT	Size of the initial extent in bytes
NEXT_EXTENT	Size of secondary extents in bytes
MIN_EXTENTS	Minimum number of extents allowed in the segment
MAX_EXTENTS	Maximum number of extents allowed in the segment
PCT_INCREASE	Percentage increase in extent size
FREELISTS	Number of process freelists allocated in this segment
FREELIST_GROUPS	Number of freelist groups allocated to this segment
LOGGING	Logging attribute
BACKED_UP	Has table been backed up since last modification?
NUM_ROWS	Number of rows in the table
BLOCKS	Number of used blocks in the table
EMPTY_BLOCKS	Number of empty (never used) blocks in the table
AVG_SPACE	Average available free space in the table
CHAIN_CNT	Number of chained rows in the table
AVG_ROW_LEN	Average row length, including row bverhead
AVG_SPACE_FREELIST_BLOCKS	Average freespace of all blocks on a freelist
NUM_FREELIST_BLOCKS	Number of blocks on the freelist
DEGREE	Number of threads per instance for scanning the table
INSTANCES	Number of instances across which the table is to be scanned
CACHE	Whether the table is to be cached in the buffer cache
TABLE_LOCK	Whether table locking is enabled or disabled
SAMPLE_SIZE	Sample size used in analyzing this table
LAST_ANALYZED	Date of the most recent time this table was analyzed
PARTITIONED	Is this table partitioned? YES or NO
IOT_TYPE	If an index-organized table, then IOT_TYPE is IOT or IOT_OVERFLOW; otherwise NULL
OBJECT_ID_TYPE only in release 8.1	If user-defined OID, then USER-DEFINED; if system-generated OID, then SYSTEM GENERATED
TABLE_TYPE_OWNER	Owner of the type of the table if the table is a typed table
TABLE_TYPE	Type of the table if the table is a typed table
TEMPORARY	Can the current session only see data that it place in this object itself?

Column	Description
NESTED	Is the table a nested table?
BUFFER_POOL	Default buffer pool to be used for table blocks
ROW_MOVEMENT only in release 8.1	Movement of the row
GLOBAL_STATS only in release 8.1	Are the statistics calculated without merging underlying partitions?
USER_STATS only in release 8.1	Were the statistics entered directly by the user?
DURATION only in release 8.1	If temporary table, then duration is SYS\$SESSION or SYS\$TRANSACTION; otherwise NULL
SKIP_CORRUPT only in release 8.1	Whether skip corrupt blocks is enabled or disabled

## DBA\_OUTLINE\_HINTS

This release 8.1 view lists the set of hints that make up the outlines.

Column	Description
NAME	Name of the outline
OWNER	The name of the user who created the outline
NODE	I.D. of the query or subquery to which the hint applies (The top-level query is labelled 1. Subqueries are assigned sequentially numbered labels, starting with 2)
JOIN_POS	Position of the table in the join order (The JOIN_POS column is 0 for all hints except the access method hints. The access method hints identify a table to which the hint and the join position apply)
HINT	Text of the hint



---

**DBA\_OUTLINES**

This release 8.1 view lists information about outlines.

Column	Description
NAME	User-specified or generated name of the stored outline (The name must be of a form that can be expressed in SQL.)
OWNER	Name of the user who created the outline
CATEGORY	A user-defined name used to group outlines into collections
USED	Flag indicating whether the outline has ever been used
TIMESTAMP	Time stamp indicating when the outline was created
VERSION	Oracle Version that created the outline
SQL_TEXT	SQL text of the query-including any hints that were a part of the original statement (If bind variables are included, the variable names are stored as SQL text, not the values that are assigned to the variables)

### DBA\_RSRC\_CONSUMER\_GROUP\_PRIVS

This release 8.1 view lists all resource consumer groups and the users and roles to which they have been granted.

Column	Description
GRANTEE	User or role receiving the grant
GRANTED_GROUP	Granted consumer group name
GRANT_OPTION	Whether grant was with the GRANT option
INITIAL_GROUP	Whether consumer group is designated as the default

### DBA\_RSRC\_CONSUMER\_GROUPS

This release 8.1 view lists all resource consumer groups which exist in the database.

Column	Description
CONSUMER_GROUP	Consumer group name
CPU_METHOD	CPU resource allocation method for the consumer group
COMMENTS	A text comment on the consumer group
STATUS	PENDING if it is part of the pending area, otherwise ACTIVE
MANDATORY	Whether consumer group is designated as the default

### DBA\_RSRC\_MANAGER\_SYSTEM\_PRIVS

This release 8.1 view lists all the users and roles that have been granted system privileges pertaining to the resource manager.

Column	Description
GRANTEE	User or role receiving the grant
PRIVILEGE	The name of the system privilege
ADMIN_OPTION	Whether the grant was with the ADMIN option

### DBA\_RSRC\_PLANS

This release 8.1 view lists all resource plans that exist in the database.

Column	Description
PLAN	Plan name
NUM_PLAN_DIRECTIVES	Number of plan directives for the plan
CPU_METHOD	CPU resource allocation method for the plan
MAX_ACTIVE_SESS_TARGET_MTH	Reserved for future use

Column	Description
PARALLEL_DEGREE_LIMIT_MTH	Parallel degree limit resource allocation method for the plan
COMMENTS	A text comment on the plan
STATUS	PENDING if it is part of the pending area, otherwise ACTIVE
MANDATORY	Whether the plan is mandatory

### DBA\_RSRC\_PLAN\_DIRECTIVES

This release 8.1 view lists all resource plans that exist in the database.

Column	Description
PLAN	Plan name
GROUP_OR_SUBPLAN	Name of the consumer group/subplan referred to
TYPE	Whether GROUP_OR_SUBPLAN refers to a consumer group or plan
CPU_P1	First parameter for the CPU resource allocation method
CPU_P2	Second parameter for the CPU resource allocation method
CPU_P3	Third parameter for the CPU resource allocation method
CPU_P4	Fourth parameter for the CPU resource allocation method
CPU_P5	Fifth parameter for the CPU resource allocation method
CPU_P6	Sixth parameter for the CPU resource allocation method
CPU_P7	Seventh parameter for the CPU resource allocation method
CPU_P8	Eighth parameter for the CPU resource allocation method
MAX_ACTIVE_SESS_TARGET_P1	Reserved for future use
PARALLEL_DEGREE_LIMIT_P1	First parameter for the parallel degree limit resource allocation method
COMMENTS	A text comment on the plan directive
STATUS	PENDING if it is part of the pending area, otherwise ACTIVE
MANDATORY	Whether the plan is mandatory

## DBA\_SEGMENTS

This view contains information about storage allocated for all database segments.

Column	Description
OWNER	Username of the segment owner
SEGMENT_NAME	Name, if any, of the segment
PARTITION_NAME	Object Partition Name (Set to NULL for non-partitioned objects).
SEGMENT_TYPE	Type of segment: INDEX PARTITION, TABLE PARTITION, TABLE, CLUSTER, INDEX, ROLLBACK, DEFERRED ROLLBACK, TEMPORARY, or CACHE
TABLESPACE_NAME	Name of the tablespace containing the segment
HEADER_FILE	ID of the file containing the segment header
HEADER_BLOCK	ID of the block containing the segment header
BYTES	Size in bytes, of the segment
BLOCKS	Size, in Oracle blocks, of the segment
EXTENTS	Number of extents allocated to the segment
INITIAL_EXTENT	Size in bytes of the initial extent of the segment
NEXT_EXTENT	Size in bytes of the next extent to be allocated to the segment
MIN_EXTENTS	Minimum number of extents allowed in the segment
MAX_EXTENTS	Maximum number of extents allowed in the segment
PCT_INCREASE	Percent by which to increase the size of the next extent to be allocated
FREELISTS	Number of process free lists allocated to this segment
FREELIST_GROUPS	Number of free list groups allocated to this segment
RELATIVE_FNO	Relative file number of the segment header
BUFFER_POOL	Name of the default buffer pool for the appropriate object

## DBA\_TAB\_COL\_STATISTICS

This view contains column statistics and histogram information that is in the DBA\_TAB\_COLUMNS view.

Column	Description
TABLE_NAME	Table, view, or cluster name
COLUMN_NAME	Column name
NUM_DISTINCT	Number of distinct values in the column
LOW_VALUE	Low value in the column
HIGH_VALUE	High value in the column
DENSITY	High value in the column

Column	Description
NUM_NULLS	Number of nulls in the column
NUM_BUCKETS	Number of buckets in histogram for the column
SAMPLE_SIZE	Sample size used in analyzing this column
LAST_ANALYZED	Date of the most recent time this column was analyzed
GLOBAL_STATS only in release 8.1	Are the statistics calculated without merging underlying partitions?
USER_STATS only in release 8.1	Were the statistics entered directly by the user?
AVG_COL_LEN only in release 8.1	Average column length in bytes

### DBA\_TAB\_COLUMNS

This view contains information that describes columns of all tables, views, and clusters. To gather statistics for this view, use the SQL command ANALYZE.

Column	Description
OWNER	Owner of the table, view, or cluster
TABLE_NAME	Table, view, or cluster name
COLUMN_NAME	Column name
DATA_TYPE	Datatype of the column
DATA_TYPE_MOD	Datatype modifier of the column
DATA_TYPE_OWNER	Owner of the datatype of the column
DATA_LENGTH	Length of the column in bytes
DATA_PRECISION	Decimal precision for NUMBER datatype; binary precision for FLOAT datatype; NULL for all other datatypes
DATA_SCALE	Digits to right of decimal point in a number
NULLABLE	Does column allow NULL values?
COLUMN_ID	Sequence number of the column as created
DEFAULT_LENGTH	Length of default value for the column
DATA_DEFAULT	Default value for the column
NUM_DISTINCT	This column remains for backward compatibility with Oracle7; this information is now in the {TAB PART}_COL_STATISTICS views
LOW_VALUE	This column remains for backward compatibility with Oracle7; this information is now in the {TAB PART}_COL_STATISTICS views

Column	Description
HIGH_VALUE	This column remains for backward compatibility with Oracle7; this information is now in the {TAB PART}_COL_STATISTICS views
DENSITY	This column remains for backward compatibility with Oracle7; this information is now in the {TAB PART}_COL_STATISTICS views
NUM_NULLS	This column remains for backward compatibility with Oracle7; this information is now in the {TAB PART}_COL_STATISTICS views
NUM_BUCKETS	Number of buckets in histogram for the column
LAST_ANALYZED	Date of the most recent time this column was analyzed
SAMPLE_SIZE	Sample size used in analyzing this column
CHARACTER_SET_NAME	Name of the character set: CHAR_CS, NCHAR_CS
CHAR_COL_DECL_LENGTH	Declaration length of character type column
GLOBAL_STATS only in release 8.1	Are the statistics calculated without merging underlying partitions?
USER_STATS only in release 8.1	Were the statistics entered directly by the user?
AVG_COL_LEN only in release 8.1	Average column length in bytes

## DBA\_TABLES

This view contains descriptions of all relational tables in the database. To gather statistics for this view, use the SQL command ANALYZE.

Column	Description
OWNER	Owner of the table
TABLE_NAME	Name of the table
TABLESPACE_NAME	Name of the tablespace containing the table
CLUSTER_NAME	Name of the cluster, if any, to which the table belongs
IOT_NAME	Name of the index organized table, if any, to which the overflow entry belongs
PCT_FREE	Minimum percentage of free space in a block
PCT_USED	Minimum percentage of used space in a block
INI_TRANS	Initial number of transactions
MAX_TRANS	Maximum number of transactions

Column	Description
INITIAL_EXTENT	Size of the initial extent in bytes
NEXT_EXTENT	Size of secondary extents in bytes
MIN_EXTENTS	Minimum number of extents allowed in the segment
MAX_EXTENTS	Maximum number of extents allowed in the segment
PCT_INCREASE	Percentage increase in extent size
FREELISTS	Number of process free lists allocated in this segment
FREELIST_GROUPS	Number of freelist groups allocated to this segment
LOGGING	Logging attribute
BACKED_UP	Has table been backed up since last modification?
NUM_ROWS	Number of rows in the table
BLOCKS	Number of used blocks in the table
EMPTY_BLOCKS	Number of empty (never used) blocks in the table
AVG_SPACE	Average available free space in the table
CHAIN_CNT	Number of chained rows in the table
AVG_ROW_LEN	Average row length, including row overhead
AVG_SPACE_FREELIST_BLOCKS	Average freespace of all blocks on a freelist
NUM_FREELIST_BLOCKS	Number of blocks on the freelist
DEGREE	Number of threads per instance for scanning the table
INSTANCES	Number of instances across which the table is to be scanned
CACHE	Whether the table is to be cached in the buffer cache
TABLE_LOCK	Whether table locking is enabled or disabled
SAMPLE_SIZE	Sample size used in analyzing this table
LAST_ANALYZED	Date of the most recent time this table was analyzed
PARTITIONED	Is this table partitioned? YES or NO
IOT_TYPE	If an index organized table, then IOT_TYPE is IOT or IOT_OVERFLOW else NULL
TABLE_TYPE_OWNER only in release 8.0	Owner of the type of the table if the table is a typed table
TABLE_TYPE only in release 8.0	Type of the table if the table is a typed table
PACKED only in release 8.0	If the table is a typed table, does it store objects in packed format?
TEMPORARY	Can the current session only see data that it place in this object itself?

Column	Description
SECONDARY only in 8.1	Is the table object created as part of icreate for domain indexes?
NESTED	Is the table a nested table?
BUFFER_POOL	Default buffer pool to be used for table blocks
ROW_MOVEMENT only in release 8.1	Whether partitioned row movement is enabled or disabled
GLOBAL_STATS only in release 8.1	Are the statistics calculated without merging underlying partitions?
USER_STATS only in release 8.1	Were the statistics entered directly by the user?
DURATION only in release 8.1	If temporary table, then duration is sys\$session or sys\$transaction else NULL
SKIP_CORRUPT only in release 8.1	Whether skip corrupt blocks is enabled or disabled
MONITORING only in release 8.1	Should we keep track of the amount of modification?

## DBA\_TEMP\_FILES

This release 8.1 view contains information about database temp files.

Column	Description
FILE_NAME	Name of the database temp file
FILE_ID	ID of the database temp file
TABLESPACE_NAME	Name of the tablespace to which the file belongs
BYTES	Size of the file in bytes
BLOCKS	Size of the file in ORACLE blocks
STATUS	File status: AVAILABLE
RELATIVE_FNO	Tablespace-relative file number
AUTOEXTENSIBLE	Autoextensible indicator: YES or NO
MAXBYTES	maximum size of the file in bytes
MAXBLOCKS	Maximum size of the file in ORACLE blocks
INCREMENT_BY	Default increment for autoextension
USER_BYTES	Size of the useful portion of file in bytes
USER_BLOCKS	Size of the useful portion of file in ORACLE blocks



## DBA\_USERS

This view lists information about all users of the database.

Column	Description
USERNAME	Name of the user
USER_ID	ID number of the user
PASSWORD	Encrypted password
ACCOUNT_STATUS	Indicates if the account is locked, expired, or unlocked
LOCK_DATE	Date the account was locked if account status was locked
EXPIRY_DATE	Date of expiration of the account
DEFAULT_TABLESPACE	Default tablespace for data
TEMPORARY_TABLESPACE	Default tablespace for temporary table
CREATED	User creation date
PROFILE	User resource profile name
INITIAL_RSRC_CONSUMER_GROUP	Initial resource consumer group for the user
EXTERNAL_NAME	User external name

## INDEX\_HISTOGRAM

This view contains information from the ANALYZE INDEX ... VALIDATE STRUCTURE command.

Column	Description
REPEAT_COUNT	Number of times that one or more index keys is repeated in the table
KEYS_WITH_REPEAT_COUNT	Number of index keys that are repeated that many times

## INDEX\_STATS

This view stores information from the last ANALYZE INDEX ... VALIDATE STRUCTURE command.

Column	Description
HEIGHT	Height of the B-tree
BLOCKS	Blocks allocated to the segment
NAME	Name of the index
PARTITION_NAME	Name of the partition of the index which was analyzed (if the index is not partitioned, a NULL is returned.)
LF_ROWS	Number of leaf rows (values in the index)

Column	Description
LF_BLKs	Number of leaf blocks in the B-Tree
LF_ROWS_LEN	Sum of the lengths of all the leaf rows
LF_BLK_LEN	Usable space in a leaf block
BR_ROWS	Number of branch rows in the B-tree
BR_BLKs	Number of branch blocks in the B-tree
BR_ROWS_LEN	Sum of the lengths of all the branch blocks in the B-tree
BR_BLK_LEN	Usable space in a branch block
DEL_LF_ROWS	Number of deleted leaf rows in the index
DEL_LF_ROWS_LEN	Total length of all deleted rows in the index
DISTINCT_KEYS	Number of distinct keys in the index (may include rows that have been deleted)
MOST_REPEATED_KEY	How many times the most repeated key is repeated (may include rows that have been deleted)
BTREE_SPACE	Total space currently allocated in the B-tree
USED_SPACE	Total space that is currently being used in the B-tree
PCT_USED	Percent of space allocated in the B-tree that is being used
ROWS_PER_KEY	Average number of rows per distinct key (this figure is calculated without consideration of deleted rows)
BLKS_GETS_PER_ACCESS	Expected number of consistent mode block reads per row, assuming that a randomly chosen row is accessed using the index; used to calculate the number of consistent reads that will occur during an index scan
PRE_ROWS only in release 8.1	Number of prefix rows (values in the index)
PRE_ROWS_LEN only in release 8.1	Sum of lengths of all prefix rows

## Dynamic Performance Views

The Oracle server contains a set of underlying views that are maintained by the server and accessible to the database administrator user `sys`. These views are called dynamic performance views because they are continuously updated while a database is open and in use, and their contents relate primarily to performance.

Although these views appear to be regular database tables, they are not. These views provide data on internal disk structures and memory structures. These views can be selected from, but never updated or altered by the user.

The file `CATALOG.SQL` contains definitions of the views and public synonyms for the dynamic performance views. You must run `CATALOG.SQL` to create these views and synonyms.

### V\$ Views

Dynamic performance views are identified by the prefix `V_`. Public synonyms for these views have the prefix `V$`. Database administrators or users should only access the `V$` objects, not the `V_` objects.

The dynamic performance views are used by Enterprise Manager and Oracle Trace, which is the primary interface for accessing information about system performance.

Suggestion: Once the instance is started, the `V$` views that read from memory are accessible. Views that read data from disk require that the database be mounted.

### Access to the Dynamic Performance Tables

After installation, only users with the username `sys` or anyone with `SYSDBA` role has access to the dynamic performance tables.

**V\$ARCHIVE\_DEST**

This view describes, for the current instance, all the archive log destinations and their current value, mode, and status.

Column	Description
ARCMode only in release 8.0	Archiving mode: MUST SUCCEED: This is a must-succeed destination BEST-EFFORT: This is a best-effort destination
DEST_ID only in release 8.1	ID (1-5)
STATUS	Status: Values in 8.0 NORMAL: This destination is normal DISABLED: This destination has been disabled Status: Values in 8.1 VALID: Initialized and available INACTIVE: No destination information DEFERRED: Manually disabled by the user ERROR: Error during open or copy DISABLED: Disabled after error; BAD PARAM: Parameter has errors
BINDING only in release 8.1	Requirement for success: MANDATORY (must succeed) or OPTIONAL (need not succeed) (depends on LOG_ARCHIVE_MIN_SUCCEED_DEST)
NAME_SPACE only in release 8.1	Definition scope: SYSTEM: System definition SESSION: Session definition
TARGET only in release 8.1	Target: PRIMARY: Copy to primary STANDBY: Copy to standby
REOPEN_SECS only in release 8.1	Retry time in seconds (after error)
DESTINATION	Meaning in 8.0: Destination text string Meaning in 8.1: Destination text string (translated primary location or standby service name)

Column	Description
FAIL_DATE only in release 8.1	Date and time of any last error
FAIL_SEQUENCE only in release 8.1	Any log sequence number at last error
FAIL_BLOCK only in release 8.1	Any block number at last error
ERROR only in release 8.1	Text of any last error

### V\$ARCHIVED\_LOG

This view displays archived log information from the control file including archive log names. An archive log record is inserted after the online redo log is successfully archived or cleared (name column is NULL if the log was cleared). If the log is archived twice, there will be two archived log records with the same THREAD#, SEQUENCE#, and FIRST\_CHANGE#, but with a different name. An archive log record is also inserted when an archive log is restored from a backup set or a copy.

Column	Description
RECID	Archived log record ID
STAMP	Archived log record stamp
NAME	Archived log file name. If set to NULL, the log file was cleared before it was archived
THREAD#	Redo thread number
SEQUENCE#	Redo log sequence number
RESETLOGS_CHANGE#	Resetlogs change# of the database when this log was written
RESETLOGS_TIME	Resetlogs time of the database when this log was written
FIRST_CHANGE#	First change# in the archived log
FIRST_TIME	Timestamp of the first change
NEXT_CHANGE#	First change in the next log
NEXT_TIME	Timestamp of the next change
BLOCKS	Size of the archived log in blocks
BLOCK_SIZE	Redo log block size
COMPLETION_TIME	Time when the archiving completed
DELETED	YES/NO

## V\$ARCHIVE\_PROCESSES

This new release 8.1 view provides information about the state of the various ARCH processes for the instance.

Column	Description
PROCESS	The identifier for the ARCH process for the instance, numbered from 0-9
STATUS	The status of the ARCH process, displayed as a keyword; possible values are STOPPED, SCHEDULED, STARTING, ACTIVE, STOPPING, and TERMINATED
LOG_SEQUENCE	This is the online redo log sequence number currently being archived, if STATE=BUSY
STATE	This is the current state of the ARCH process, displayed as a keyword; possible keywords are IDLE or BUSY

## V\$BUFFER\_POOL

This view displays information about all buffer pools available for the instance. The “sets” pertain to the number of LRU latch sets.

Column	Description
INST_ID	Instance ID
ID	Buffer pool ID number
NAME	Buffer pool name
LO_SETID	Low set ID number
HI_SETID	High set ID number
SET_COUNT	Number of sets in this buffer pool; This is HI_SETID–LO_SETID + 1
SIZE	Number of buffers allocated to the buffer pool
LO_BNUM	Low buffer number for this pool
HI_BNUM	High buffer number for this pool

## V\$BUFFER\_POOL\_STATISTICS

This view displays information about all buffer pools available for the instance. The “sets” pertain to the number of LRU latch sets.

Column	Description
ID	Buffer pool ID number
NAME	Buffer pool name
SET_MSIZE	Buffer pool ID number
CNUM_REPL	Number of buffers on replacement list
CNUM_WRITE	Number of buffers on write list
CNUM_SET	Number of buffers in set
BUF_GOT	Number of buffers gotten by the set
SUM_WRITE	Number of buffers written by the set
SUM_SCAN	Number of buffers scanned in the set
FREE_BUFFER_WAIT	Free buffer wait statistic
WRITE_COMPLETE_WAIT	Write complete wait statistic
BUFFER_BUSY_WAIT	Buffer busy wait statistic
FREE_BUFFER_INSPECTED	Free buffer inspected statistic
DIRTY_BUFFERS_INSPECTED	Dirty buffers inspected statistic
DB_BLOCK_CHANGE	Database blocks changed statistic
DB_BLOCK_GETS	Database blocks gotten statistic
CONSISTENT_GETS	Consistent gets statistic
PHYSICAL_READS	Physical reads statistic
PHYSICAL_WRITES	Physical writes statistic

## V\$CACHE

This is a Parallel Server view. This view contains information from the block header of each block in the SGA of the current instance as related to particular database objects.

Column	Description
FILE#	Datafile identifier number (to find filename, query DBA_DATA_FILES or V\$DBFILES)
BLOCK#	Block number
STATUS	Status of block: FREE = not currently in use XCUR = exclusive SCUR = shared current CR = consistent read READ = being read from disk MREC = in media recovery mode IREC = in instance recovery mode
XNC	Number of PCM x to null lock conversions due to contention with another instance. This column is obsolete but is retained for historical compatibility.
NAME	Name of the database object containing the block
KIND	Type of database object.
OWNER#	Owner number
LOCK_ELEMENT_ADDR	The address of the lock element that contains the PCM lock that is covering the buffer (If more than one buffer has the same address, then these buffers are covered by the same PCM lock.)
LOCK_ELEMENT_NAME	The address of the lock element that contains the PCM lock that is covering the buffer (If more than one buffer has the same address, then these buffers are covered by the same PCM lock.)
PARTITION_NAME	NULL for non-partitioned objects



## V\$CIRCUIT

This view contains information about virtual circuits, which are user connections to the database through dispatchers and servers.

Column	Description
CIRCUIT	Circuit address
DISPATCHER	Current dispatcher process address
SERVER	Current server process address
WAITER	Address of server process that is waiting for the (currently busy) circuit to become available
SADDR	Address of session bound to the circuit
STATUS	Status of the circuit: BREAK (currently interrupted), EOF (about to be removed), OUTBOUND (an outward link to a remote database), NORMAL (normal circuit into the local database)
QUEUE	Queue the circuit is currently on: COMMON (on the common queue, waiting to be picked up by a server process), DISPATCHER (waiting for the dispatcher), SERVER (currently being serviced), OUTBOUND (waiting to establish an outbound connection: only release 8.0), NONE (idle circuit)
MESSAGE0	Size in bytes of the messages in the first message buffer
MESSAGE1	Size in bytes of the messages in the second message buffer
MESSAGE2 only in release 8.1	Size in bytes of the messages in the third message buffer
MESSAGE3 only in release 8.1	Size in bytes of the messages in the fourth message buffer
MESSAGES	Total number of messages that have gone through this circuit
BYTES	Total number of bytes that have gone through this circuit
BREAKS	Total number of breaks (interruptions) for this circuit
PRESENTATION only in release 8.1	The presentation protocol used by the client and server

## V\$CURRENT\_BUCKET

This release 8.0 view displays information useful for predicting the number of additional cache misses that would occur if the number of buffers in the cache were reduced. This view becomes obsolete in release 8.1.

Column	Description
COUNT	The count

## V\$DATAFILE

This view contains data file information from the control file. The order of the column has changed from release 8.0 and release 8.1.

Column	Description
FILE#	File identification number
CREATION_CHANGE#	Change number at which the data file was created
CREATION_TIME	Time stamp of the datafile creation
TS#	Tablespace number
RFILE#	Tablespace relative data file number
STATUS	Type of file (system or user) and its status; values: OFFLINE, ONLINE, SYSTEM, RECOVER, SYSOFF (an offline file from the SYSTEM tablespace)
ENABLED	Describes how accessible the file is from SQL
CHECKPOINT_CHANGE#	SCN at last checkpoint
CHECKPOINT_TIME	Time stamp of the checkpoint#
UNRECOVERABLE_CHANGE#	Last unrecoverable change# made to this data file (This column is always updated when an unrecoverable operation completes.)
UNRECOVERABLE_TIME	Time stamp of the last unrecoverable change
LAST_CHANGE#	Last change# made to this data file (Set to NULL if the datafile is being changed.)
LAST_TIME	Time stamp of the last change
OFFLINE_CHANGE#	Offline change# of the last offline range (This column is updated only when the datafile is brought online.)
ONLINE_CHANGE#	Online change# of the last offline range
ONLINE_TIME	Online time stamp of the last offline range
BYTES	Current size in bytes; 0 if inaccessible
BLOCKS	Current datafile size in blocks; 0 if inaccessible
CREATE_BYTES	Size when created, in bytes
BLOCK_SIZE	Block size of the data file
NAME	Data file name
PLUGGED_IN only in release 8.1	Describes whether the tablespace is plugged in (The value is 1 if the tablespace is plugged in and has not been made read-write, 0 if not.)

## V\$DBFILE

This view lists all data files making up the database. This view is retained for historical compatibility. Using V\$DATAFILE is recommended instead.

Column	Description
FILE#	File identifier
NAME	Name of file

## V\$DB\_OBJECT\_CACHE

This view displays database objects that are cached in the library cache. Objects include tables, indexes, clusters, synonym definitions, PL/SQL procedures and packages, and triggers.

Column	Description
OWNER	Owner of the object
NAME	Name of the object
DB_LINK	Database link name, if any
NAMESPACE	Library cache namespace of the object: TABLE/PROCEDURE, BODY, TRIGGER, INDEX, CLUSTER, OBJECT
TYPE	Type of the object: INDEX, TABLE, CLUSTER, VIEW, SET, SYNONYM, SEQUENCE, PROCEDURE, FUNCTION, PACKAGE, PACKAGE BODY, TRIGGER, CLASS, OBJECT, USER, DBLINK
SHARABLE_MEM	Amount of sharable memory in the shared pool consumed by the object
LOADS	Number of times the object has been loaded. This count also increases when an object has been invalidated
EXECUTIONS	Not used. To see actual execution counts, see “V\$SQLAREA”
LOCKS	Number of users currently locking this object
PINS	Number of users currently pinning this object
KEPT	YES or NO, depending on whether this object has been “kept” (permanently pinned in memory) with the PL/SQL procedure DBMS_SHARED_POOL.KEEP

**V\$DISPATCHER**

This view provides information on the dispatcher processes.

Column	Description
NAME	Name of the dispatcher process
NETWORK	Network protocol supported by this dispatcher; for example, TCP or DECNET
PADDR	Process address
STATUS	Dispatcher status: WAIT (idle), SEND (sending a message connection), RECEIVE (receiving a message), CONNECT (establishing a connection), DISCONNECT (handling a disconnect request), BREAK (handling a break), OUTBOUND (establishing an outbound connection)
ACCEPT	Whether this dispatcher is accepting new connections: YES, NO
MESSAGES	Number of messages processed by this dispatcher
BYTES	Size in bytes of messages processed by this dispatcher
BREAKS	Number of breaks occurring in this connection
OWNED	Number of circuits owned by this dispatcher
CREATED	Number of circuits created by this dispatcher
IDLE	Total idle time for this dispatcher in hundredths of a second
BUSY	Total busy time for this dispatcher in hundredths of a second
LISTENER	The most recent Oracle error number the dispatcher received from the listener
CONF_INDX only in release 8.1	Zero-based index of the MTS_DISPATCHERS configuration used by this dispatcher

**V\$EVENT\_NAME**

This view contains information about Wait events.

Column	Description
EVENT#	The number of the Wait event
NAME	The name of the Wait event
PARAMETER1	The description of the first parameter for the Wait event
PARAMETER2	The description of the second parameter for the Wait event
PARAMETER3	The description of the third parameter for the Wait event

**V\$FILESTAT**

This view contains information about file read/write statistics.

Column	Description
FILE#	Number of the file
PHYRDS	Number of physical reads done
PHYWRTS	Number of times DBWR is required to write
PHYBLKRD	Number of physical blocks read
PHYBLKWRT	Number of blocks written to disk, which may be the same as PHYWRTS if all writes are single blocks
READTIM	Time (in hundredths of a second) spent doing reads if the TIMED_STATISTICS parameter is TRUE; 0 if FALSE
WRITETIM	Time (in hundredths of a second) spent doing writes if the TIMED_STATISTICS parameter is TRUE; 0 if FALSE
AVGIOTIM	The average time (in hundredths of a second) spent on I/O, if the TIMED_STATISTICS parameter is TRUE; 0 if FALSE
LSTIOTIM	The time (in hundredths of a second) spent doing the last I/O, if the TIMED_STATISTICS parameter is TRUE; 0 if FALSE
MINIOTIM	The minimum time (in hundredths of a second) spent on a single I/O, if the TIMED_STATISTICS parameter is TRUE; 0 if FALSE
MAXIOWTM	The maximum time (in hundredths of a second) spent doing a single write, if the TIMED_STATISTICS parameter is TRUE; 0 if FALSE
MAXIORTM	The maximum time (in hundredths of a second) spent doing a single read, if the TIMED_STATISTICS parameter is TRUE; 0 if FALSE

**V\$FIXED\_TABLE**

This view displays all dynamic performance tables, views, and derived tables in the database. Some V\$ tables refer to real tables and are therefore not listed.

Column	Description
NAME	Name of the object
OBJECT_ID	Identifier of the fixed object
TYPE	Object type: TABLE, VIEW
TABLE_NUM	Number that identifies the dynamic performance table if it is of type TABLE

**V\$LATCH**

This view lists statistics for nonparent latches and summary statistics for parent latches; that is, the statistics for a parent latch include counts from each of its children.

**Note:** Columns SLEEP5, SLEEP6,... SLEEP11 are present for compatibility with previous versions of Oracle. No data are accumulated for these columns.

Column	Description
ADDR	Address of latch object
LATCH#	Latch level
NAME	Latch name
GETS	Number of times obtained a wait
MISSES	Number of times obtained a wait but failed on the first try
SLEEPS	Number of times slept when wanted a wait
IMMEDIATE_GETS	Number of times obtained without a wait
IMMEDIATE_MISSES	Number of times failed to get without a wait
WAITERS_WOKEN	How many times a wait was awakened
WAITS_HOLDING_LATCH	Number of waits while holding a different latch
SPIN_GETS	Gets that missed first try but succeeded on spin
SLEEP1	Waits that slept 1 time
SLEEP2	Waits that slept 2 times
SLEEP3	Waits that slept 3 times
SLEEP4	Waits that slept 4 times
SLEEP5	Waits that slept 5 times
SLEEP6	Waits that slept 6 times
SLEEP7	Waits that slept 7 times
SLEEP8	Waits that slept 8 times
SLEEP9	Waits that slept 9 times
SLEEP10	Waits that slept 10 times
SLEEP11	Waits that slept 11 times

## V\$LATCHNAME

This view contains information about decoded latch names for the latches shown in V\$LATCH. The rows of V\$LATCHNAME have a one-to-one correspondence to the rows of V\$LATCH.

Column	Description
LATCH#	Latch number
NAME	Latch name

## V\$LIBRARYCACHE

This view contains statistics about library cache performance and activity.

Column	Description
NAMESPACE	The library cache namespace
GETS	The number of times a lock was requested for objects of this namespace
GETHITS	The number of times an object's handle was found in memory
GETHITRATIO	The ratio of GETHITS to GETS
PINS	The number of times a PIN was requested for objects of this namespace
PINHITS	The number of times all of the meta data pieces of the library object were found in memory
PINHITRATIO	The ratio of PINHITS to PINS
RELOADS	Any PIN of an object that is not the first PIN performed since the object handle was created, and which requires loading the object from disk
INVALIDATIONS	The total number of times objects in this namespace were marked invalid because a dependent object was modified
DLM_LOCK_REQUESTS	The number of GET requests lock instance locks
DLM_PIN_REQUESTS	The number of PIN requests lock instance locks
DLM_PIN_RELEASES	The number of release requests PIN instance locks
DLM_INVALIDATION_REQUESTS	The number of GET requests for invalidation instance locks
DLM_INVALIDATIONS	The number of invalidation pings received from other instances

**V\$LOCK**

This view lists the locks currently held by the Oracle server and outstanding requests for a lock or latch.

Column	Description
ADDR	Address of lock state object
KADDR	Address of lock
SID	Identifier for session holding or acquiring the lock
TYPE	Type of user lock: TM: DML enqueue TX: Transaction enqueue UL: User supplied
ID1	Lock identifier #1 (depends on type)
ID2	Lock mode in which the process requests the lock: 0, None 1, Null (NULL) 2, Row-S (SS) 3, Row-X (SX) 4, Share (S) 5, S/Row-X (SSX) 6, Exclusive (X)
CTIME	Time since current mode was granted
BLOCK	The lock is blocking another lock



**Note:** Types of system locks:

BL: Buffer hash table instance

CF: Control file schema global enqueue

CI: Cross-instance function invocation instance

CU: Cursor bind

DF: Data file instance

DL: Direct loader parallel index create

DM: Mount/startup db primary/secondary instance

DR: Distributed recovery process

DX: Distributed transaction entry

FS: File set

HW: Space management operations on a specific segment

IN: Instance number

IR: Instance recovery serialization global enqueue

IS: Instance state

IV: Library cache invalidation instance

JQ: Job queue

KK: Thread kick

LA..LP: Library cache lock instance lock (A..P = namespace)

MM: Mount definition global enqueue

MR: Media recovery

NA..NZ: Library cache pin instance (A..Z = namespace)

PF: Password File

PI, PS: Parallel operation

PR: Process startup

QA..QZ: Row cache instance (A..Z = cache) RT: Redo thread global enqueue

SC: System commit number instance

SM: SMON

SN: Sequence number instance

SQ: Sequence number enqueue

SS: Sort segment

ST: Space transaction enqueue

SV: Sequence number value

TA: Generic enqueue

TS: Temporary segment enqueue (ID2=0)

TS: New block allocation enqueue (ID2=1)

TT: Temporary table enqueue

UN: User name

US: Undo segment DDL

WL: Being-written redo log instance

## V\$LOCKED\_OBJECT

This view lists all locks acquired by every transaction on the system.

Column	Description
XIDUSN	Undo segment number
XIDSLOT	Slot number
XIDSQN	Sequence number
OBJECT_ID	Object ID being locked
SESSION_ID	Session ID
ORACLE_USERNAME	Oracle username
OS_USER_NAME	OS username
PROCESS	OS process ID
LOCKED_MODE	Lock mode

## V\$LOG

This view contains log file information from the control files.

Column	Description
GROUP#	Log group number
THREAD#	Log thread number
SEQUENCE#	Log sequence number
BYTES	Size of the log in bytes
MEMBERS	Number of members in the log group

Column	Description
ARCHIVED	Archive status: YES, NO
STATUS	Log status. The STATUS column can have the following values: UNUSED: Indicates that the online redo log has never been written to (This is the state of a redo log that was just added, or just after a RESETLOGS, when it is not the current redo log.) CURRENT: Indicates that this is the current redo log (This implies that the redo log is active. The redo log could be open or closed.) ACTIVE: Indicates that the log is active but is not the current log (It is needed for crash recovery. It may be in use for block recovery. It might or might not be archived.) CLEARING: Indicates that the log is being recreated as an empty log after an ALTER DATABASE CLEAR LOGFILE command. (After the log is cleared, the status changes to UNUSED.) CLEARING_CURRENT: Indicates that the current log is being cleared of a closed thread (The log can stay in this status if there is some failure in the switch such as an I/O error writing the new log header.) INACTIVE: Indicates that the log is no longer needed for instance recovery (It may be in use for media recovery. It might or might not be archived.)
FIRST_CHANGE#	Lowest SCN in the log
FIRST_TIME	Time of first SCN in the log

## V\$LOGFILE

This view contains information about redo log files.

Column	Description
GROUP#	Redo log group identifier number
STATUS	Status of this log member: INVALID: File is inaccessible STALE: File's contents are incomplete DELETED: File is no longer used blank: File is in use
MEMBER	Redo log member name

**V\$MTS**

This view contains information for tuning the multithreaded server.

Column	Description
MAXIMUM _CONNECTIONS	Maximum number of connections each dispatcher can support (This value is determined at startup time using Net8 constants and other port-specific information, or can be lowered using the MLS_DISPATCHERS parameter.)
SERVERS _STARTED	Total number of multi-threaded servers started since the instance started (but not including those started during startup)
SERVERS _TERMINATED	Total number of multithreaded servers stopped by Oracle since the instance started
SERVERS _HIGHWATER	Highest number of servers running at one time since the instance started (If this value reaches the value set for the MTS_MAX_SERVERS initialization parameter, consider raising the value of MTS_SERVERS.)

**V\$MYSTAT**

This view contains statistics on the current session.

Column	Description
SID	The ID of the current session
STATISTIC#	The number of the statistic
VALUE	The value of the statistic

**V\$OBsolete\_PARAMETER**

This new release 8.1 view lists obsolete parameters. If any value is true, you should examine why.

Column	Description
NAME	The name of the parameter
ISSPECIFIED	Whether the parameter was specified in the config file

**V\$PARAMETER**

This view lists information about initialization parameters.

Column	Description
NUM	Parameter number
NAME	Parameter name
TYPE	Parameter type: 1 = Boolean, 2 = string, 3 = integer
VALUE	Parameter value
ISDEFAULT	Whether the parameter value is the default
ISSES_MODIFIABLE	TRUE: the parameter can be changed with ALTER SESSION FALSE: the parameter cannot be changed with ALTER SESSION
ISSYS_MODIFIABLE	IMMEDIATE: the parameter can be changed with ALTER SYSTEM DEFERRED: the parameter cannot be changed until the next session FALSE: the parameter cannot be changed with ALTER SYSTEM
ISMODIFIED	Indicates how the parameter was modified (If an ALTER SESSION was performed, the value will be MODIFIED. If an ALTER SYSTEM (which will cause all the currently logged in sessions' values to be modified) was performed the value will be SYS_MODIFIED.)
ISADJUSTED	Indicates that the RDBMS adjusted the input value to a more suitable value (for example, the parameter value should be prime, but the user input a nonprime number, so the rdbms adjusted the value to the next prime number)
DESCRIPTION	A descriptive comment about the parameter

**V\$PX\_PROCESS**

This release 8.1 view contains information about the sessions running parallel execution.

Column	Description
SERVER_NAME	The name of the parallel server (P000, P001, etc)
STATUS	The state of the parallel server. Either In Use or Available
PID	The process identifier
SPID	The OS process ID
SID	The session ID of slave, if in use
SERIAL#	The session serial number of slave, if in use

## V\$PX\_PROCESS\_SYSSTAT

This release 8.1 view contains information about the sessions running parallel execution at the system level.

Column	Description
STATISTIC	The name of the statistic
VALUE	The value of the statistic

## V\$PX\_SESSION

This release 8.1 view contains information about the sessions running parallel execution at the session level.

Column	Description
SADDR	Session address
SID	Session identifier
SERIAL#	Session serial number
QCSID	Session identifier of the parallel coordinator
QCSERIAL#	Session serial number of the parallel coordinator
QCINST_ID	Instance number on which the parallel coordinator is running
SERVER_GROUP	Logical group of servers to which this parallel server process belongs
SERVER_SET	Logical set of servers to which this parallel server process belongs to (A single server group will have at most two server sets.)
SERVER#	The logical number of a parallel server process within a server set
DEGREE	Degree of parallelism being used by the server set
REQ_DEGREE	Degree of parallelism that was requested by the user when the statement was issued and prior to any resource, multiuser, or load-balancing reductions

## V\$PX\_SESSTAT

This release 8.1 view contains information about the sessions running parallel execution.

Column	Description
SADDR	Session address
SID	Session identifier
SERIAL#	Session serial number
QCSID	Session identifier of the parallel coordinator

Column	Description
QCSERIAL#	Session serial number of the parallel coordinator
QCINST_ID	Instance number on which the parallel coordinator is running
SERVER_GROUP	Logical group of servers to which this parallel server process belongs
SERVER_SET	Logical set of servers to which this parallel server process belongs (A single server group will have at most two server sets.)
SERVER#	Logical number of a parallel server process within a server set
DEGREE	Degree of parallelism being used by the server set
REQ_DEGREE	Degree of parallelism that was requested by the user when the statement was issued and prior to any resource, multiuser, or load balancing reductions
STATISTIC#	Statistic number (identifier)
VALUE	Statistic value

## V\$QUEUE

This view contains information on the multithread message queues.

Column	Description
PADDR	Address of the process that owns the queue
TYPE	Type of queue: COMMON (processed by servers), OUTBOUND (used by remote servers), DISPATCHER
QUEUED	Number of items in the queue
WAIT	Total time that all items in this queue have waited (Divide by TOTALQ for average wait per item.)
TOTALQ	Total number of items that have ever been in the queue

## V\$RECENT\_BUCKET

This release 8.0 view displays information useful for estimating the performance of a large cache. This view becomes obsolete in release 8.1.

Column	Description
COUNT	The count

## V\$ROLLNAME

This view lists the names of all online rollback segments. It can only be accessed when the database is open.

Column	Description
USN	Rollback (undo) segment number
NAME	Rollback segment name

## V\$ROLLSTAT

This view contains rollback segment statistics.

Column	Description
USN	Rollback segment number
EXTENTS	Number of extents in rollback segment
RSSIZE	Size in bytes of rollback segment
WRITES	Number of bytes written to rollback segment
XACTS	Number of active transactions
GETS	Number of header gets
WAITS	Number of header waits
OPTSIZE	Optimal size of rollback segment
HWMSIZE	High water mark of rollback segment size
SHRINKS	Number of times the size of a rollback segment decreases
WRAPS	Number of times rollback segment is wrapped
EXTENDS	Number of times rollback segment size is extended
AVESHRINK	Average shrink size
AVEACTIVE	Current size of active extents, averaged over time
STATUS	Rollback segment status
CUREXT	Current extent
CURBLK	Current block

## V\$ROWCACHE

This view displays statistics for data dictionary activity. Each row contains statistics for one data dictionary cache.

Column	Description
CACHE#	Row cache ID number
TYPE	Parent or subordinate row cache type
SUBORDINATE#	Subordinate set number
PARAMETER	Name of the initialization parameter that determines the number of entries in the data dictionary cache



Column	Description
COUNT	Total number of entries in the cache
USAGE	Number of cache entries that contain valid data
FIXED	Number of fixed entries in the cache
GETS	Total number of requests for information on the data object
GETMISSES	Number of data requests resulting in cache misses
SCANS	Number of scan requests
SCANMISSES	Number of times a scan failed to find the data in the cache
SCANCOMPLETES	For a list of subordinate entries, the number of times the list was scanned completely
MODIFICATIONS	Number of inserts, updates, and deletions
FLUSHES	Number of times flushed to disk

### V\$RSRC\_CONSUMER\_GROUP

This release 8.1 view displays data related to the currently active resource consumer groups.

Column	Description
NAME	Name of the consumer group
ACTIVE_SESSIONS	Number of currently active sessions in this consumer group
EXECUTION_WAITERS	Number of currently active sessions waiting for an execution quantum
REQUESTS	Total number of requests that were executed in this consumer group
CPU_WAIT_TIME	Total amount of time that sessions waited for CPU
CPU_WAITS	Number of times all sessions in this consumer group had to wait for CPU
CONSUMED_CPU_TIME	Total amount of CPU time consumed by all sessions in this consumer group
YIELDS	Number of times sessions in this consumer group had to yield the CPU
SESSIONS_QUEUED	Count of currently queued sessions waiting to become active

### V\$RSRC\_CONSUMER\_GROUP\_CPU\_MTH

This release 8.1 view shows all available resource allocation methods for resource consumer groups.

Column	Description
NAME	Name of the CPU resource allocation method

## V\$RSRC\_PLAN

This release 8.1 view displays the names of all currently active resource plans.

Column	Description
NAME	Name of the resource plan

## V\$RSRC\_PLAN\_CPU\_MTH

This release 8.1 view shows all available CPU resource allocation methods for resource plans.

Column	Description
NAME	Name of the resource allocation method

## V\$SESSION

This view lists session information for each current session. The order of the columns has changed from release 8.0 to release 8.1.

Column	Description
SADDR	Session address
SID	Session identifier
SERIAL#	Session serial number (Used to identify uniquely a session's objects. Guarantees that session-level commands are applied to the correct session objects if the session ends and another session begins with the same session ID.)
AUDSID	Auditing session ID
PADDR	Address of the process that owns this session
USER#	Oracle user identifier
USERNAME	Oracle username
COMMAND	Command in progress (last statement parsed)
OWNERID	Column contents are invalid if the value is 2147483644; Otherwise, this column contains the identifier of the user who owns the migratable session.)
TADDR	Address of transaction state object
LOCKWAIT	Address of lock waiting for; NULL if none
STATUS	Status of the session: ACTIVE (currently executing SQL), INACTIVE, KILLED (marked to be killed), CACHED (temporarily cached for use by Oracle*XA), SNIPED (session inactive, waiting on the client)
SERVER	Server type: DEDICATED, SHARED, PSEUDO, NONE
SCHEMA#	Schema user identifier
SCHEMANAME	Schema user name

Column	Description
OSUSER	Operating system client user name
PROCESS	Operating system client process ID
MACHINE	Operating system machine name
TERMINAL	Operating system terminal name
PROGRAM	Operating system program name
TYPE	Session type
SQL_ADDRESS	Used with SQL_HASH_VALUE to identify the SQL statement that is currently being executed
SQL_HASH_VALUE	Used with SQL_ADDRESS to identify the SQL statement that is currently being executed
PREV_SQL_ADDR only in release 8.1	
PREV_HASH_VALUE only in release 8.1	
MODULE	Contains the name of the currently executing module as set by calling the DBMS_APPLICATION_INFO.SET_MODULE procedure
MODULE_HASH	Hash value of the above MODULE
ACTION	Contains the name of the currently executing action as set by calling the DBMS_APPLICATION_INFO.SET_ACTION procedure
ACTION_HASH	Hash value of the above action name
CLIENT_INFO	Information set by the DBMS_APPLICATION_INFO.SET_CLIENT_INFO procedure
FIXED_TABLE_SEQUENCE	This contains a number that increases every time the session completes a call to the database and there has been an intervening select from a dynamic performance table (This column can be used by performance monitors to monitor statistics in the database. Each time the performance monitor looks at the database, it only needs to look at sessions that are currently active or have a higher value in this column than the highest value that the performance monitor saw the last time. All the other sessions have been idle since the last time the performance monitor looked at the database.)
ROW_WAIT_OBJ#	Object ID for the table containing the ROWID specified in ROW_WAIT_ROW#

Column	Description
ROW_WAIT_FILE#	Identifier for the datafile containing the ROWID specified in ROW_WAIT_ROW# (This column is valid only if the session is currently waiting for another transaction to commit and the value of ROW_WAIT_OBJ# is non-zero.)
ROW_WAIT_BLOCK#	Identifier for the block containing the ROWID specified in ROW_WAIT_ROW# (This column is valid only if the session is currently waiting for another transaction to commit and the value of ROW_WAIT_OBJ# is non-zero.)
ROW_WAIT_ROW#	The current ROWID being locked (This column is valid only if the session is currently waiting for another transaction to commit and the value of ROW_WAIT_OBJ# is non-zero.)
LOGON_TIME	Time of logon.
LAST_CALL_ET	The last call
PDML_ENABLED only in release 8.0	If set to YES, the session is in a PARALLEL DML enabled mode; otherwise set to NO
FAILOVER_TYPE	NONE if failover is disabled for this session; SESSION if client is able to failover its session following a disconnect; SELECT if client is able to fail over selects in progress as well
FAILOVER_METHOD	NONE if failover is disabled for this session, BASIC if client reconnects following a disconnect, PRECONNECT if the backup instance is able to support all connections from every instance that it is backup for
FAILED_OVER	TRUE if running in failover mode and have failed over, otherwise FALSE
RESOURCE_CONSUMER_GROUP only in release 8.1	Name of the session's current resource consumer group
PDML_STATUS only in release 8.1	If ENABLED, the session is in a PARALLEL DML enabled mode; if DISABLED, PARALLEL DML enabled mode is not supported for the session; if FORCED, the session has been altered to force PARALLEL DML
PDDL_STATUS only in release 8.1	If ENABLED, the session is in a PARALLEL DDL enabled mode; if DISABLED, PARALLEL DDL enabled mode is not supported for the session; if FORCED, the session has been altered to force PARALLEL DDL

List of values for the COMMAND column:

0 No command in progress. Occurs when process is in a transitory state, usually when terminating.

1 CREATE TABLE  
2 INSERT  
3 SELECT  
4 CREATE CLUSTER  
5 ALTER CLUSTER  
6 UPDATE  
7 DELETE  
8 DROP CLUSTER  
9 CREATE INDEX  
10 DROP INDEX  
11 ALTER INDEX  
12 DROP TABLE  
13 CREATE SEQUENCE  
14 ALTER SEQUENCE  
15 ALTER TABLE  
16 DROP SEQUENCE  
17 GRANT  
18 REVOKE  
19 CREATE SYNONYM  
20 DROP SYNONYM  
21 CREATE VIEW  
22 DROP VIEW  
23 VALIDATE INDEX  
24 CREATE PROCEDURE  
25 ALTER PROCEDURE  
26 LOCK TABLE  
27 NO OPERATION  
28 RENAME  
29 COMMENT  
30 AUDIT  
31 NOAUDIT  
32 CREATE DATABASE LINK  
33 DROP DATABASE LINK

34 CREATE DATABASE  
35 ALTER DATABASE  
36 CREATE ROLLBACK SEGMENT  
37 ALTER ROLLBACK SEGMENT  
38 DROP ROLLBACK SEGMENT  
39 CREATE TABLESPACE  
40 ALTER TABLESPACE  
41 DROP TABLESPACE  
42 ALTER SESSION  
43 ALTER USER  
44 COMMIT  
45 ROLLBACK  
46 SAVEPOINT  
47 PL/SQL EXECUTE  
48 SET TRANSACTION  
49 ALTER SYSTEM SWITCH LOG  
50 EXPLAIN  
51 CREATE USER  
52 CREATE ROLE  
53 DROP USER  
54 DROP ROLE  
55 SET ROLE  
56 CREATE SCHEMA  
57 CREATE CONTROL FILE  
58 ALTER TRACING  
59 CREATE TRIGGER  
60 ALTER TRIGGER  
61 DROP TRIGGER  
62 ANALYZE TABLE  
63 ANALYZE INDEX  
64 ANALYZE CLUSTER  
65 CREATE PROFILE  
66 DROP PROFILE

67 ALTER PROFILE  
68 DROP PROCEDURE  
69 DROP PROCEDURE  
70 ALTER RESOURCE COST  
71 CREATE SNAPSHOT LOG  
72 ALTER SNAPSHOT LOG  
73 DROP SNAPSHOT LOG  
74 CREATE SNAPSHOT  
75 ALTER SNAPSHOT  
76 DROP SNAPSHOT  
79 ALTER ROLE  
85 TRUNCATE TABLE  
86 TRUNCATE CLUSTER  
88 ALTER VIEW  
91 CREATE FUNCTION  
92 ALTER FUNCTION  
93 DROP FUNCTION  
94 CREATE PACKAGE  
95 ALTER PACKAGE  
96 DROP PACKAGE  
97 CREATE PACKAGE BODY  
98 ALTER PACKAGE BODY  
99 DROP PACKAGE BODY

## V\$SESSION\_EVENT

This view lists information on waits for an event by a session. Note that the `TIME_WAITED` and `AVERAGE_WAIT` columns will contain a value of zero on those platforms that do not support a fast timing mechanism. If you are running on one of these platforms and you want this column to reflect true wait times, you must set `TIMED_STATISTICS` to `TRUE` in the parameter file. Please remember that doing this will have a small negative effect on system performance.

Column	Description
SID	The ID of the session
EVENT	The name of the Wait event
TOTAL_WAITS	Total number of waits for this event by this session
TOTAL_TIMEOUTS	Total number of timeouts for this event by this session
TIME_WAITED	Total amount of time waited for this event by this session, in hundredths of a second
AVERAGE_WAIT	Average amount of time waited for this event by this session, in hundredths of a second
MAX_WAIT	Maximum time (in hundredths of a second) waited for this event by this session

## V\$SESSION\_LONGOPS

This view displays the status of certain long-running operations. It provides progression reports on operations using the columns `SO FAR` and `TOTALWORK`. For example, the operational status for the following components can be monitored:

- Hash cluster creations
- Backup operations
- Recovery operations

Column	Description
SID	Session identifier
SERIAL#	Session serial number
OPNAME	The operation name
TARGET	Object on which the operation is carried out
TARGET_DESC	Description of the target
SO FAR	Units of work done so far
TOTALWORK	Total units of work
UNITS	Units of measurement
START_TIME	Starting time of operation



Column	Description
LAST_UPDATE_TIME	Time when statistics last updated
ELAPSED_SECONDS	Number of elapsed seconds from the start of operations
CONTEXT	Context
MESSAGE	Statistics summary message

### V\$SESSION\_WAIT

This view lists the resources or events for which active sessions are waiting.

Column	Description
SID	Session identifier
SEQ#	Sequence number that uniquely identifies this wait. Incremented for each wait
EVENT	Resource or event for which the session is waiting
P1TEXT	Description of first parameter
P1	First additional parameter
P1RAW	First additional parameter
P2TEXT	Description of second parameter
P2	Second additional parameter
P2RAW	Second additional parameter
P3TEXT	Description of third parameter
P3	Third additional parameter
P3RAW	Third additional parameter
WAIT_TIME	A non-zero value is the session's last wait time; a zero value means the session is currently waiting
STATE	WAITING: 0 The session is currently waiting WAITED UNKNOWN TIME : -2 Duration of last wait is unknown WAITED SHORT TIME : -1 Last wait < 1/100th of a second WAITED KNOWN TIME: >0 WAIT_TIME = duration of last wait

### V\$SESSTAT

This view lists user session statistics.

Column	Description
SID	Session identifier
STATISTIC#	Statistic number (identifier)
VALUE	Statistic value

**V\$SESS\_IO**

This view lists I/O statistics for each user session.

Column	Description
SID	Session identifier
BLOCK_GETS	Block gets for this session
CONSISTENT_GETS	Consistent gets for this session
PHYSICAL_READS	Physical reads for this session
BLOCK_CHANGES	Block changes for this session
CONSISTENT_CHANGES	Consistent changes for this session

**V\$SGASTAT**

This view contains detailed information on the System Global Area.

Column	Description
NAME	SGA component name
BYTES	Memory size in bytes
POOL	Designates the pool in which the memory in NAME resides. Value can be: LARGE POOL: Memory is allocated from the large pool SHARED POOL: Memory is allocated from the shared pool

**V\$SHARED\_POOL\_RESERVED**

This fixed view lists statistics that help you tune the reserved pool and space within the shared pool.

Column	Description
FREE_SPACE	Total amount of free space on the reserved list
AVG_FREE_SIZE	Average size of the free memory on the reserved list
FREE_COUNT	Number of free pieces of memory on the reserved list
MAX_FREE_SIZE	Size of the largest free piece of memory on the reserved list
USED_SPACE	Total amount of used memory on the reserved list
AVG_USED_SIZE	Average size of the used memory on the reserved list
USED_COUNT	Number of used pieces of memory on the reserved list
MAX_USED_SIZE	Size of the largest used piece of memory on the reserved list
REQUESTS	Number of times that the reserved list was searched for a free piece of memory

Column	Description
REQUEST_MISSES	Number of times the reserved list did not have a free piece of memory to satisfy the request, and started flushing objects from the LRU list
LAST_MISS_SIZE	Request size of the last request miss, when the reserved list did not have a free piece of memory to satisfy the request and started flushing objects from the LRU list
MAX_MISS_SIZE	Request size of the largest request miss, when the reserved list did not have a free piece of memory to satisfy the request and started flushing objects from the LRU list
REQUEST_FAILURES	Number of times that no memory was found to satisfy a request (that is, the number of times the error ORA-4031 occurred)
LAST_FAILURE_SIZE	Request size of the last failed request (that is, the request size for the last ORA-4031 error)
ABORTED_REQUEST_THRESHOLD	Minimum size of a request which signals an ORA-4031 error without flushing objects
ABORTED_REQUESTS	Number of requests that signalled an ORA-4031 error without flushing objects
LAST_ABORTED_SIZE	Last size of the request that returned an ORA-4031 error without flushing objects from the LRU list

### Note

The following columns of the V\$SHARED\_POOL\_RESERVED view contain values that are valid even if SHARED\_POOL\_RESERVED\_SIZE is not set:

- REQUEST\_FAILURES
- LAST\_FAILURE\_SIZE
- ABORTED\_REQUEST\_THRESHOLD
- ABORTED\_REQUESTS
- LAST\_ABORTED\_SIZE

## V\$SHARED\_SERVER

This view contains information on the shared server processes.

Column	Description
NAME	Name of the server
PADDR	Server's process address
STATUS	Server status: EXEC (executing SQL) WAIT (ENQ) (waiting for a lock), WAIT (SEND) (waiting to send data to user) WAIT (COMMON) (idle; waiting for a user request) WAIT (RESET) (waiting for a circuit to reset after a break) QUIT (terminating)
MESSAGES	Number of messages processed
BYTES	Total number of bytes in all messages
BREAKS	Number of breaks
CIRCUIT	Address of circuit currently being serviced
IDLE	Total idle time in hundredths of a second
BUSY	Total busy time in hundredths of a second
REQUESTS	Total number of requests taken from the common queue in this server's lifetime

## V\$SORT\_SEGMENT

This view contains information about every sort segment in a given instance. The view is only updated when the tablespace is of the TEMPORARY type.

Column	Description
TABLESPACE_NAME	Name of tablespace
SEGMENT_FILE	File number of the first extent
SEGMENT_BLOCK	Block number of the first extent
EXTENT_SIZE	Extent size
CURRENT_USERS	Number of active users of the segment
TOTAL_EXTENTS	Total number of extents in the segment
TOTAL_BLOCKS	Total number of blocks in the segment
RELATIVE_FNO	Relative file number of the sort segment header
USED_EXTENTS	Extents allocated to active sorts
USED_BLOCKS	Blocks allocated to active sorts
FREE_EXTENTS	Extents not allocated to any sort

Column	Description
FREE_BLOCKS	Blocks not allocated to any sort
ADDED_EXTENTS	Number of extent allocations
EXTENT_HITS	Number of times an unused extent was found in the pool
FREED_EXTENTS	Number of deallocated extents
FREE_REQUESTS	Number of requests to deallocate
MAX_SIZE	Maximum number of extents ever used
MAX_BLOCKS	Maximum number of blocks ever used
MAX_USED_SIZE	Maximum number of extents used by all sorts
MAX_USED_BLOCKS	Maximum number of blocks used by all sorts
MAX_SORT_SIZE	Maximum number of extents used by an individual sort
MAX_SORT_BLOCKS	Maximum number of blocks used by an individual sort

### V\$SORT\_USAGE

This view describes sort usage.

Column	Description
USER	User who requested temporary space
SESSION_ADDR	Address of shared SQL cursor
SESSION_NUM	Serial number of session
SQLADDR	Address of SQL statement
SQLHASH	Hash value of SQL statement
TABLESPACE	Tablespace in which space is allocated
CONTENTS	Indicates whether tablespace is TEMPORARY/PERMANENT
SEGFILE#	File number of initial extent
SEGBLK#	Block number of the initial extent
EXTENTS	Extents allocated to the sort
BLOCKS	Extents in blocks allocated to the sort
SEGRFNO#	Relative file number of initial extent

**V\$SQLAREA**

This view lists statistics on shared SQL area and contains one row per SQL string. It provides statistics on SQL statements that are in memory, parsed, and ready for execution.

Column	Description
SQL_TEXT	First eighty characters of the SQL text for the current cursor
SHARABLE_MEM	Sum of all sharable memory, in bytes, of all the child cursors under this parent
PERSISTENT_MEM	sum of all persistent memory, in bytes, of all the child cursors under this parent
RUNTIME_MEM	sum of all the ephemeral frame sizes of all the children
SORTS	Sum of the number of sorts that was done for all the children
VERSION_COUNT	Number of children that are present in the cache under this parent
LOADED_VERSIONS	Number of children that are present in the cache AND have their context heap (KGL heap 6) loaded
OPEN_VERSIONS	Number of child cursors that are currently open under this current parent
USERS_OPENING	Number of users that have any of the child cursors open
EXECUTIONS	Total number of executions, totalled over all the children
USERS_EXECUTING	Total number of users executing the statement over all children
LOADS	Number of times the object was loaded or reloaded
FIRST_LOAD_TIME	Time stamp of the parent creation time
INVALIDATIONS	Total number of invalidations over all the children
PARSE_CALLS	Sum of all parse calls to all the child cursors under this parent
DISK_READS	Sum of the number of disk reads over all child cursors
BUFFER_GETS	Sum of buffer gets over all child cursors
ROWS_PROCESSED	Total number of rows processed on behalf of this SQL statement
COMMAND_TYPE	Oracle command type definition
OPTIMIZER_MODE	Mode under which the SQL statement is executed
PARSING_USER_ID	User ID of the user that has parsed the very first cursor under this parent
PARSING_SCHEMA_ID	Schema ID that was used to parse this child cursor
KEPT_VERSIONS	The number of child cursors that have been marked to be kept using the DBMS_SHARED_POOL package

Column	Description
ADDRESS	Address of the handle to the parent for this cursor
HASH_VALUE	Hash value of the parent statement in the library cache
MODULE	Contains the name of the module that was executing at the time that the SQL statement was first parsed as set by calling DBMS_APPLICATION_INFO.SET_MODULE
MODULE_HASH	Hash value of the module that is named in the MODULE column
ACTION	Contains the name of the action that was executing at the time that the SQL statement was first parsed as set by calling DBMS_APPLICATION_INFO.SET_ACTION
ACTION_HASH	Hash value of the action that is named in the ACTION column
SERIALIZABLE_ABORTS	Number of times the transaction fails to serialize, producing ORA-8177 errors, totaled over all the children

### V\$SQLTEXT

This view contains the text of SQL statements belonging to shared SQL cursors in the SGA.

Column	Description
ADDRESS	Used with HASH_VALUE to identify uniquely a cached cursor
HASH_VALUE	Used with ADDRESS to identify uniquely a cached cursor
PIECE	Number used to order the pieces of SQL text
SQL_TEXT	A column containing one piece of the SQL text
COMMAND_TYPE	Code for the type of SQL statement (SELECT, INSERT, and so on)

## V\$STATNAME

This view displays decoded statistic names for the statistics shown in the V\$SESSTAT and V\$SYSSTAT tables.

Column	Description
STATISTIC#	Statistic number
NAME	Statistic name
CLASS	Statistic class: 1 (User) 2 (Redo) 4 (Enqueue) 8 (Cache) 16 (OS) 32 (Parallel Server) 64 (SQL) 128 (Debug)

## V\$SYSSTAT

This view lists system statistics. To find the name of the statistic associated with each statistic number (STATISTIC#).

Column	Description
STATISTIC#	Statistic number
NAME	Statistic name
CLASS	Statistic class: 1 (User) 2 (Redo) 4 (Enqueue) 8 (Cache) 16 (OS) 32 (Parallel Server) 64 (SQL) 128 (Debug)
VALUE	Statistic value



**V\$SYSTEM\_EVENT**

This view contains information on total waits for an event. Note that the `TIME_WAITED` and `AVERAGE_WAIT` columns will contain a value of zero on those platforms that do not support a fast timing mechanism. If you are running on one of these platforms and you want this column to reflect true wait times, you must set `TIMED_STATISTICS` to `TRUE` in the parameter file. Please remember that doing this will have a small negative effect on system performance.

Column	Description
EVENT	Name of the Wait event
TOTAL_WAITS	Total number of waits for this event
TOTAL_TIMEOUTS	Total number of timeouts for this event
TIME_WAITED	Total amount of time waited for this event, in hundredths of a second
AVERAGE_WAIT	Average amount of time waited for this event, in hundredths of a second

**V\$SYSTEM\_PARAMETER**

This view contains information on system parameters.

Column	Description
NUM	Parameter number
NAME	Parameter name
TYPE	Parameter type: 1 = Boolean, 2 = string, 3 = integer
VALUE	Parameter value
ISDEFAULT	Whether the parameter value is the default
ISSES_MODIFIABLE	TRUE: The parameter can be changed with ALTER SESSION FALSE: The parameter cannot be changed with ALTER SESSION
ISSYS_MODIFIABLE	IMMEDIATE: The parameter can be changed with ALTER SYSTEM DEFERRED: The parameter cannot be changed until the next session FALSE: The parameter cannot be changed with ALTER SYSTEM

Column	Description
ISMODIFIED	Indicates how the parameter was modified (If an ALTER SESSION was performed, the value will be MODIFIED. If an ALTER SYSTEM (which will cause all the currently logged in sessions' values to be modified) was performed the value will be SYS_MODIFIED.)
ISADJUSTED	Indicates that the RDBMS adjusted the input value to a more suitable value (for example, the parameter value should be prime, but the user input a non-prime number, so the RDBMS adjusted the value to the next prime number)
DESCRIPTION	A descriptive comment about the parameter
ISSYS_MODIFIABLE	IMMEDIATE: The parameter can be changed with ALTER SYSTEM DEFERRED: The parameter cannot be changed until the next session FALSE: The parameter cannot be changed with ALTER SYSTEM
ISMODIFIED	Indicates how the parameter was modified (If an ALTER SESSION was performed, the value will be MODIFIED. If an ALTER SYSTEM (which will cause all the currently logged in sessions' values to be modified) was performed the value will be SYS_MODIFIED.)
ISADJUSTED	Indicates that the RDBMS adjusted the input value to a more suitable value (for example, the parameter value should be prime, but the user input a non-prime number, so the RDBMS adjusted the value to the next prime number)
DESCRIPTION	A descriptive comment about the parameter

## V\$TEMPFILE

This release 8.1 view displays tempfile information.

Column	Description
FILE#	The absolute file number
CREATION_CHANGE#	The creation System Change Number
CREATION_TIME	The creation time
TS#	The tablespace number
RFILE#	The relative file number in tablespace
STATUS	The status of the file (offline/online)
ENABLED	Enabled for read and/or write
BYTES	Size of the file in bytes (from file header)

Column	Description
BLOCKS	Size of the file in blocks (from =file header)
CREATE_BYTES	Creation size of the file (in bytes)
BLOCK_SIZE	Block size for the file
NAME	Name of the file

### V\$TEMP\_EXTENT\_MAP

This release 8.1 view displays the status of each unit for all temporary tablespaces.

Column	Description
TABLESPACE_NAME	Name of tablespace to which this unit belongs
FILE_ID	Absolute file number
BLOCK_ID	Begin block number for this unit
BYTES	Bytes in extent
BLOCKS	Blocks in extent
OWNER	Which instance owns this unit (string)
RELATIVE_FNO	The relative file number

### V\$TEMPSTAT

This 8.1 view contains information about file read/write statistics.

Column	Description
FILE#	Number of the file
PHYRDS	Number of physical reads done
PHYWRTS	Number of times DBWR is required to write
PHYBLKRD	Number of physical blocks read
PHYBLKWRT	Number of blocks written to disk, which may be the same as PHYWRTS if all writes are single blocks
READTIM	Time (in hundredths of a second) spent doing reads if the TIMED_STATISTICS parameter is TRUE; 0 if FALSE
WRITETIM	Time (in hundredths of a second) spent doing writes if the TIMED_STATISTICS parameter is TRUE; 0 if FALSE
AVGIOTIM	Average time (in hundredths of a second) spent on I/O, if the TIMED_STATISTICS parameter is TRUE; 0 if FALSE

Column	Description
LSTIOTIM	Time (in hundredths of a second) spent doing the last I/O, if the TIMED_STATISTICS parameter is TRUE; 0 if FALSE
MINIOTIM	Minimum time (in hundredths of a second) spent on a single I/O, if the TIMED_STATISTICS parameter is TRUE; 0 if FALSE
MAXIOWTM	Maximum time (in hundredths of a second) spent doing a single write, if the TIMED_STATISTICS parameter is TRUE; 0 if FALSE
MAXIORTM	Maximum time (in hundredths of a second) spent doing a single read, if the TIMED_STATISTICS parameter is TRUE; 0 if FALSE

### V\$WAITSTAT

This view lists block contention statistics. This table is only updated when timed statistics are enabled.

Column	Description
CLASS	Class of block
COUNT	Number of waits by this OPERATION for this CLASS of block
TIME	Sum of all wait times for all the waits by this OPERATION for this CLASS of block

---

F

---

## Initialization Parameters

## Initialization Parameters

Refer to the Oracle8i Reference Release 8.1.5 A67790-011 to get the full list of the initialization parameters and to the Oracle8i Migration Release 8.1.5 A67774-01 for a list of new, obsolete, and changed parameters.

The following topics are included in this appendix:

- Initialization parameter file
- Specifying values in the parameter file
- Reading the parameter descriptions
- Parameter descriptions

### Initialization Parameter File

The initialization parameter file is a text file that contains a list of parameters and a value for each parameter. The file should be written in the client's default character set. Specify values in the parameter file which reflect your installation.

See your Oracle operating system-specific documentation for the default locations and filenames for these parameter files. The `init.ora` file is what the Oracle Server reads for its parameter information upon startup.

### Changing Parameter Values

To change a parameter's value, edit the parameter file. The next time the instance starts, it uses the new parameter values in the updated parameter file. Note that the change does not take effect until the instance is shut down and restarted.

**Dynamic Parameters** Some initialization parameters are dynamic; that is, they can be modified using the `ALTER SESSION`, `ALTER SYSTEM`, or `ALTER SYSTEM DEFERRED` commands while an instance is running.

Use this syntax for dynamically altering the initialization parameters:

```
ALTER SESSION SET parameter_name = value
ALTER SYSTEM SET parameter_name = value
ALTER SYSTEM SET parameter_name = value DEFERRED
```

Whenever a dynamic parameter is modified using the `ALTER SYSTEM`, or `ALTER SYSTEM DEFERRED` command, then the command that modifies the parameter is also recorded in the alert log.

The ALTER SESSION command changes the value of the parameter specific to the session that invokes this command. The value of this parameter does not change for other sessions in the instance.

The ALTER SYSTEM command modifies the global value of the parameter until the database is shut down. The ALTER SYSTEM command does not always change the parameter value for the current session. Use the ALTER SESSION command to change the parameter value for the current session.

The ALTER SYSTEM DEFERRED command does not modify the global value of the parameter for existing sessions, but the value will be modified for future sessions that connect to the database.

**Displaying Current Parameter Values** To see the current settings for initialization parameters, use the following SQL\*Plus command:

```
SQL> SHOW PARAMETERS
```

This displays all parameters in alphabetical order, with their current values.

Enter the following text string to see a display for all parameters having BLOCK in their name.:

```
SQL> SHOW PARAMETERS BLOCK
```

If you display all the parameters, you might want to use the SPOOL command to write the output to a file.

## Parameters Definition

### ARCH\_IO\_SLAVES (Obsolete in 8.1.3)

ARCH\_IO\_SLAVES specifies the number of I/O slaves used by the ARCH process to archive redo logfiles. The ARCH process and its slaves always write to disk. By default the value is 0 and I/O slaves are not used.

This parameter is normally adjusted when an I/O bottleneck has been detected in the ARCH process. Typically, I/O bottlenecks in this process will occur on platforms that do not support asynchronous I/O or implement it inefficiently.

<b>Parameter Type</b>	integer
<b>Parameter Class</b>	static
<b>Default Value</b>	0
<b>Range of Values</b>	0-15

### BACKGROUND\_DUMP\_DEST

BACKGROUND\_DUMP\_DEST specifies the pathname for a directory where debugging trace files for the background processes (LGWR, DBWR, and so on) are written during Oracle operations.

An ALERT file in the directory specified by BACKGROUND\_DUMP\_DEST logs significant database events and messages. Anything that affects the database instance-wide or globally is recorded here. This file records all instance startups and shutdowns, messages to the operator console, and errors that cause trace files to be written. It also records every CREATE, ALTER, or DROP operation on a database, tablespace, or rollback segment. The ALERT file is a normal text file. Its filename is operating system-dependent. For platforms that support multiple instances, it takes the form ALERT\_sid.LOG. This file grows slowly, but without limit, so the database administrator might want to delete it periodically. The file can be deleted even when the database is running.

<b>Parameter Type</b>	string
<b>Parameter Class</b>	static
<b>Default Value</b>	operating system-dependent
<b>Range of Values</b>	valid local pathname, directory, or disk

### BACKUP\_DISK\_IO\_SLAVES (Obsolete in 8.1.3)

BACKUP\_DISK\_IO\_SLAVES specifies the number of I/O slaves used by the Recovery Manager to backup, copy, or restore. Note that every Recovery Manager channel can get the specified number of I/O slave processes. By default, the value is 0 and I/O slaves are not used.



Typically I/O slaves are used to “simulate” asynchronous I/O on platforms that either do not support asynchronous I/O or implement it inefficiently. However, I/O slaves can be used even when asynchronous I/O is being used. In this case, the I/O slaves will use asynchronous I/O.

<b>Parameter Type</b>	Integer
<b>Parameter Class</b>	Dynamic, scope = ALTER SYSTEM DEFERRED
<b>Default Value</b>	0
<b>Range of Values</b>	0–15, although a value under 7 is recommended

## BACKUP\_TAPE\_IO\_SLAVES

BACKUP\_TAPE\_IO\_SLAVES specifies whether I/O slaves are used by the Recovery Manager to backup, copy, or restore data to tape. When

BACKUP\_TAPE\_IO\_SLAVES = TRUE, an I/O slave process is used to write to or read from a tape device. If this parameter is FALSE (the default), then I/O slaves are not used for backups; instead, the shadow process engaged in the backup will access the tape device.

Note, as a tape device can only be accessed by one process at any given time, this parameter is a boolean, that allows or disallows deployment of an I/O slave process to access a tape device.

Typically I/O slaves are used to “simulate” asynchronous I/O on platforms that either do not support asynchronous I/O or implement it inefficiently. However, I/O slaves can be used even when asynchronous I/O is being used. In that case the I/O slaves will use asynchronous I/O.

<b>Parameter Type</b>	Boolean
<b>Parameter Class</b>	Dynamic, scope = ALTER SYSTEM DEFERRED
<b>Default Value</b>	FALSE
<b>Range of Values</b>	TRUE/FALSE

## BUFFER\_POOL\_KEEP

BUFFER\_POOL\_KEEP is used to improve buffer cache performance. It enables you to keep an object in the buffer cache.

<b>Parameter Type</b>	String
<b>Parameter Class</b>	Static
<b>Default Value</b>	None
<b>Range of Values</b>	-

**BUFFER\_POOL\_RECYCLE**

BUFFER\_POOL\_RECYCLE is used to improve buffer cache performance. It allows you to limit the size of an object in the buffer cache.

<b>Parameter Type</b>	String
<b>Parameter Class</b>	Static
<b>Default Value</b>	None
<b>Range of Values</b>	-

**CACHE\_SIZE\_THRESHOLD (Obsolete in 8.1.3)****CLOSE\_CACHED\_OPEN\_CURSORS (Obsolete in 8.1.3)**

CLOSE\_CACHED\_OPEN\_CURSORS specifies whether cursors opened and cached in memory by PL/SQL are automatically closed at each COMMIT. A value of FALSE signifies that cursors opened by PL/SQL are held open so that subsequent executions need not open a new cursor. If PL/SQL cursors are reused frequently, setting the parameter to FALSE can cause subsequent executions to be faster.

A value of TRUE causes open cursors to be closed at each COMMIT or ROLLBACK. The cursor can then be reopened as needed. If cursors are rarely reused, setting the parameter to TRUE frees memory used by the cursor when the cursor is no longer in use.

<b>Parameter Type</b>	Boolean
<b>Parameter Class</b>	Static
<b>Default Value</b>	FALSE
<b>Range of Values</b>	TRUE/FALSE

**CURSOR\_SPACE\_FOR\_TIME**

If CURSOR\_SPACE\_FOR\_TIME is set to TRUE, the database uses more space for cursors to save time. It affects both the shared SQL area and the client's private SQL area.

Shared SQL areas are kept pinned in the shared pool when this parameter's value is TRUE. As a result, shared SQL areas are not aged out of the pool as long as there is an open cursor that references them. Because each active cursor's SQL area is present in memory, execution is faster. Because the shared SQL areas never leave memory while they are in use, however, you should set this parameter to TRUE only when the shared pool is large enough to hold all open cursors simultaneously.

Setting this parameter to TRUE also retains the private SQL area allocated for each cursor between executes instead of discarding it after cursor execution. This saves cursor allocation and initialization time.

<b>Parameter Type</b>	Boolean
<b>Parameter Class</b>	Static
<b>Default Value</b>	FALSE
<b>Range of Values</b>	TRUE/FALSE

## DB\_BLOCK\_BUFFERS

DB\_BLOCK\_BUFFERS specifies the number of database buffers available in the buffer cache. It is one of the primary parameters which contribute to the total memory requirements of the SGA on the instance. The DB\_BLOCK\_BUFFERS parameter, together with the DB\_BLOCK\_SIZE parameter, determines the total size of the buffer cache. Effective use of the buffer cache can greatly reduce the I/O load on the database. Since DB\_BLOCK\_SIZE can be specified only when the database is first created, use DB\_BLOCK\_BUFFERS to control the size of the buffer cache.

This parameter affects the probability that a data block will be pinged when Parallel Server is enabled: the more buffers, the more chance of pings.

<b>Parameter Type</b>	Integer
<b>Parameter Class</b>	Static
<b>Default Value</b>	50
<b>Range of Values</b>	4–operating system–specific

## DB\_BLOCK\_CHECKPOINT\_BATCH (Obsolete in 8.1.3)

DB\_BLOCK\_CHECKPOINT\_BATCH specifies the number of buffers that will be added to each batch of buffers that DBWR writes in order to advance checkpoint processing.

Reducing DB\_BLOCK\_CHECKPOINT\_BATCH prevents the I/O system from being flooded with checkpoint writes and allows other modified blocks to be written to disk. Setting it to a higher value allows checkpoints to complete more quickly.

In general, DB\_BLOCK\_CHECKPOINT\_BATCH should be set to a value that allows the checkpoint to complete before the next log switch takes place. If a log switch takes place every 20 minutes, then this parameter should be set to a value that allows checkpointing to complete within 20 minutes.

Setting `DB_BLOCK_CHECKPOINT_BATCH` to zero causes the default value to be used. If an overly large value is specified for this parameter, Oracle (silently) limits it to the number of blocks that can be written in a database writer write batch.

<b>Parameter Type</b>	Integer
<b>Parameter Class</b>	Dynamic, scope = ALTER SYSTEM IMMEDIATE
<b>Default Value</b>	8
<b>Range of Values</b>	0–derived

### **DB\_BLOCK\_CHECKING (New in 8.1)**

`DB_BLOCK_CHECKING` is used to control whether block checking is done from transaction managed blocks. Because early detection of corruptions is useful and has minimal, if any, performance impact, it is recommended that you use the default setting. The `FALSE` setting is provided for compatibility with earlier releases where block checking is disabled as a default. As the parameter is dynamic, it provides more flexibility than events 10210 and 10211, which it will ultimately replace. Note that the setting of `DB_BLOCK_CHECKING` overrides any setting of events 10210 and 10211.

<b>Parameter Type</b>	Boolean
<b>Parameter Class</b>	Dynamic, scope = ALTER SESSION, ALTER SYSTEM DEFERRED
<b>Default Value</b>	FALSE
<b>Range of Values</b>	FALSE/TRUE

### **DB\_BLOCK\_CHECKSUM**

If `DB_BLOCK_CHECKSUM` is set to `TRUE`, `DBWn` and the direct loader will calculate a checksum and store it in the cache header of every data block when writing it to disk. Checksums will be verified when a block is read only if this parameter is `TRUE` and the last write of the block stored a checksum.

**Warning:** Setting `DB_BLOCK_CHECKSUM` to `TRUE` can cause performance overhead.

<b>Parameter Type</b>	Boolean
<b>Parameter Class</b>	Dynamic, scope = ALTER SYSTEM
<b>Default Value</b>	FALSE
<b>Range of Values</b>	FALSE/TRUE

**DB\_BLOCK\_LRU\_EXTENDED\_STATISTICS (Obsolete in 8.1.3)**

DB\_BLOCK\_LRU\_EXTENDED\_STATISTICS disables or enables compilation of statistics which measures the effects of increasing the number of buffers in the buffer cache in the SGA. When this facility is enabled, it keeps track of the number of disk accesses that would be saved if additional buffers were allocated. A value greater than zero specifies the additional number of buffers (over DB\_BLOCK\_BUFFERS) for which statistics are kept. This tuning tool should be turned off during normal operation.

When compiling statistics, set this parameter to the maximum size you want to use to evaluate the buffer cache. It should be set to zero otherwise. (Although you can set this value very high, it is not practical to set it to a size beyond your system's memory capacity.)

Setting this parameter can cause a large performance loss, so it should only be set when the system is lightly loaded.

<b>Parameter Type</b>	Integer
<b>Parameter Class</b>	Static
<b>Default Value</b>	0
<b>Range of Values</b>	0–dependent on system memory capacity

**DB\_BLOCK\_LRU\_LATCHES**

DB\_BLOCK\_LRU\_LATCHES specifies the upper bound of the number of LRU latch sets. Set this parameter to a value equal to the desired number of LRU latch sets. Oracle decides whether to use this value or reduce it based on a number of internal checks. If the parameter is not set, Oracle calculates a value for the number of sets. The value calculated by Oracle is usually adequate. Increase this only if misses are higher than 3% in V\$LATCH.

<b>Parameter Type</b>	Integer
<b>Parameter Class</b>	Static
<b>Default Value</b>	CPU_COUNT/2
<b>Range of Values</b>	1–the number of CPUs

**DB\_BLOCK\_LRU\_STATISTICS (Obsolete in 8.1.3)**

DB\_BLOCK\_LRU\_STATISTICS disables or enables compilation of statistics in the V\$CURRENT\_BUCKET table, which measures the effect of fewer buffers in the SGA buffer cache.

Set this parameter to TRUE when you want to compile statistics for the V\$CURRENT\_BUCKET table; otherwise, leave it set to FALSE. This parameter is a tuning tool and should be set to FALSE during normal operation.

Setting this parameter can cause a large performance loss, so it should only be set when the system is lightly loaded.

<b>Parameter Type</b>	Boolean
<b>Parameter Class</b>	Static
<b>Default Value</b>	FALSE
<b>Range of Values</b>	FALSE/TRUE

## DB\_BLOCK\_SIZE

DB\_BLOCK\_SIZE specifies the size in bytes of Oracle database blocks. Typical values are 2048 and 4096. The value for DB\_BLOCK\_SIZE in effect at CREATE DATABASE time determines the size of the blocks; at all other times the value must be set to the original value.

This parameter affects the maximum value of the FREELISTS storage parameter for tables and indexes. DSS (data warehouse) database environments tend to benefit from larger block size values.

<b>Parameter Type</b>	Integer
<b>Parameter Class</b>	static
<b>Default Value</b>	Operating system-dependent
<b>Range of Values</b>	Operating system-dependent (2048 - 32768)

## DB\_FILE\_MULTIBLOCK\_READ\_COUNT

DB\_FILE\_MULTIBLOCK\_READ\_COUNT is used for multi-block I/O and specifies the maximum number of blocks read in one I/O operation during a sequential scan. The total number of I/Os needed to perform a full table scan depends on factors such as these:

- The size of the table
- The multi-block read count
- Whether parallel query is being utilized for the operation

The default is 8. OLTP and batch environments typically have values for this parameter in the range of 4 to 16. DSS (data warehouse) database environments tend to get the most benefit from maximizing the value for this parameter.

The actual maximums vary by operating system; they are always less than the operating system's maximum I/O size expressed as Oracle blocks ( $\text{max\_IO\_size} / \text{DB\_BLOCK\_SIZE}$ ). Attempts to set this parameter to a value greater than the maximum will cause the maximum to be used.

<b>Parameter Type</b>	Integer
<b>Parameter Class</b>	Dynamic, scope = ALTER SYSTEM, ALTER SESSION
<b>Default Value</b>	8
<b>Range of Values</b>	Operating system-dependent

**DB\_FILE\_SIMULTANEOUS\_WRITES (Obsolete in 8.1.3)**

DB\_FILE\_SIMULTANEOUS\_WRITES specifies the maximum number of simultaneous writes that can be made to a given database file. Oracle also uses the value of this parameter in computing various internal parameters that affect read and write operations to database files.

If you specify an excessively large value for this parameter, significant delays in performing read and write operations to a given database file might occur. This is because I/O requests get queued in the disk. If you set a value that is too small, the number of I/Os that can be issued to a given database file will be limited.

In environments where the database files reside on RAM devices or which use disk striping at the operating system level, it is beneficial to increase the value of this parameter. If striped files are used, Oracle recommends that you set the value of this parameter to 4 times the maximum number of disks in the file that is striped the most.

This parameter is also used to determine the number of reads per file in the redo read-ahead when reading redo during recovery.

<b>Parameter Type</b>	Integer
<b>Parameter Class</b>	Static
<b>Default Value</b>	4
<b>Range of Values</b>	Minimum: 1 Maximum: <ul style="list-style-type: none"><li>• When striping is used: 4 times the number of disks in the file that is striped the most</li><li>• When striping is not used: 4</li></ul>

**DB\_WRITER\_PROCESSES**

DB\_WRITER\_PROCESSES specifies the initial number of database writer processes for an instance.

If you use DBWR\_IO\_SLAVES, only one database writer process will be used, regardless of the setting for DB\_WRITER\_PROCESSES.

<b>Parameter Type</b>	Integer
<b>Parameter Class</b>	Static
<b>Default Value</b>	1
<b>Range of Values</b>	1–10



## DBWR\_IO\_SLAVES

DBWR\_IO\_SLAVES specifies the number of I/O slaves used by the DBWR process. The DBWR process and its slaves always write to disk. By default, the value is 0 and I/O slaves are not used.

Typically I/O slaves are used to “simulate” asynchronous I/O on platforms that do not support asynchronous I/O or implement it inefficiently. However, I/O slaves can be used even when asynchronous I/O is being used. In that case the I/O slaves will use asynchronous I/O.

I/O slaves are also useful in database environments with very large I/O throughput, even if asynchronous I/O is enabled.

<b>Parameter Type</b>	Integer
<b>Parameter Class</b>	Static
<b>Default Value</b>	0
<b>Range of Values</b>	0 to system-dependent value

## DISK\_ASYNC\_IO

DISK\_ASYNC\_IO can be used to control whether I/O to datafiles, controlfiles and logfiles are asynchronous. If a platform supports asynchronous I/O to disk, it is recommended that you leave this parameter set to its default. However, if the asynchronous I/O implementation is not stable, this parameter can be set to FALSE to disable asynchronous I/O. If a platform does not support asynchronous I/O to disk, this parameter has no effect.

If DISK\_ASYNC\_IO is set to FALSE, then DBWR\_IO\_SLAVES should also be set.

<b>Parameter Type</b>	Boolean
<b>Parameter Class</b>	Static
<b>Default Value</b>	TRUE
<b>Range of Values</b>	TRUE, FALSE

## DML\_LOCKS

DML\_LOCKS specifies the maximum number of DML locks—one for each table modified in a transaction. The value should equal the grand total of locks on tables currently referenced by all users. For example, if three users are modifying data in one table, then three entries would be required. If three users are modifying data in two tables, then six entries would be required.

The default value assumes an average of four tables referenced per transaction. For some systems, this value may not be enough.

If the value is set to 0, enqueues are disabled and performance is slightly increased. However, you cannot use DROP TABLE, CREATE INDEX, or explicit lock statements such as LOCK TABLE IN EXCLUSIVE MODE. If the value is set to 0 on one instance, it must be set to 0 on all instances of an Oracle Parallel Server.

**PDML Restrictions** DML\_LOCKS has the following PDML restrictions regarding locks acquired by a parallel UPDATE/DELETE/INSERT statement.

- The coordinator acquires one Table lock SX where there is one Partition lock X per partition.
- For parallel UPDATE/DELETE, unless the UPDATE/DELETE's WHERE clause specifies the partitions involved, the coordinator will acquire partition locks for all partitions.
- For parallel INSERT, the coordinator will acquire partition locks for all partitions.
- Each slave acquires one Table lock SX where there is one Partition lock NULL per partition, and where there is one Partition-Wait lock X per partition.

A slave can work on one or more partitions but a partition can only be worked on by one slave. So for a table with 600 partitions, running with parallel degree 100, assuming all partitions are involved in the parallel UPDATE/DELETE statement.

There are the following requirements:

- The coordinator acquires one Table lock SX and 600 Partition locks X
- Total slaves acquire 100 Table locks SX, 600 Partition locks NULL, and 600 Partition-Wait locks X.

<b>Parameter Type</b>	Integer
<b>Parameter Class</b>	Static
<b>Default Value</b>	Derived (4 * TRANSACTIONS)
<b>Range of Values</b>	20–unlimited, 0 Multiple instances must all have positive values or must all be 0

## ENQUEUE\_RESOURCES

An enqueue is a sophisticated locking mechanism that permits several concurrent processes to share known resources to varying degrees. Any object that can be used concurrently can be protected with enqueues. For example, Oracle allows varying levels of sharing on tables: two processes can lock a table in share mode or in share update mode.

One difference between enqueues and latches is that in latches there is no ordered queue of waiting processes as there are in enqueues. Processes waiting for latches can either use timers to wake up and retry or spin (only in multiprocessors).

ENQUEUE\_RESOURCES sets the number of resources that can be concurrently locked by the lock manager. The default value of ENQUEUE\_RESOURCES is derived from the SESSIONS parameter and should be adequate, as long as  $DML\_LOCKS + 20$  is less than ENQUEUE\_RESOURCES. For three or fewer sessions, the default value is 20. For four to ten sessions, the default value is  $((SESSIONS - 3) * 5) + 20$ ; and for more than 10 sessions, it is  $((SESSIONS - 10) * 2) + 55$ .

If you explicitly set ENQUEUE\_RESOURCES to a value higher than  $DML\_LOCKS + 20$ , then the value you provide is used.

If there are many tables, the value may be increased. Allow one per resource (regardless of the number of sessions or cursors using that resource), not one per lock. Only increase this parameter if Oracle returns an error specifying that enqueues are exhausted.

<b>Parameter Type</b>	Integer
<b>Parameter Class</b>	Static
<b>Default Value</b>	Derived
<b>Range of Values</b>	10: Unlimited

## EVENT

EVENT is used to debug the system. This parameter should not usually be altered except at the direction of Oracle technical support personnel.

<b>Parameter Type</b>	String
<b>Parameter Class</b>	Static
<b>Default Value</b>	NULL
<b>Range of Values</b>	-

**FAST\_FULL\_SCAN\_ENABLED (Obsolete in 8.1.3)****FAST\_START\_IO\_TARGET (New in 8.1)**

FAST\_START\_IO\_TARGET (available only with the Oracle Enterprise Edition) specifies the number of IOs that should be needed during crash or instance recovery. It imposes a more accurate bound on the number of recovery IOs than DB\_BLOCK\_MAX\_DIRTY\_TARGET.

When this parameter is set, DBWn writes dirty buffers out more aggressively to keep the number of blocks that must be processed during recovery below the value specified in the parameter. Note that this parameter does not impose a hard limit on the number of recovery IOs. There may be transient workload situations in which the number of IOs needed during recovery is greater than the value specified in this parameter, but if this occurs, DBWn will not slow down database activity.

Smaller values for this parameter result in faster recovery times. This improvement in recovery performance is achieved at the expense of additional writing activity during normal processing.

Setting this parameter's value to 0 disables the mechanism that limits the number of IOs that need to be performed during recovery. All other writing activity is unaffected.

<b>Parameter Type</b>	Integer
<b>Parameter Class</b>	Dynamic, scope= ALTER SYSTEM set at run time
<b>Default Value</b>	All the buffers in the cache
<b>Range of Values</b>	1000 to all buffers in the cache, setting to 0 disables limiting recovery IOs

**FAST\_START\_PARALLEL\_ROLLBACK (New in 8.1)**

FAST\_START\_PARALLEL\_ROLLBACK helps to determine the maximum number of processes which may exist for performing parallel rollback. If the value is false, parallel rollback is disabled. If the value is low, 2 \* CPU\_COUNT number of processes may be used. If the value is high, at most 4 \* CPU\_COUNT number of rollback servers are used for parallel rollback.

<b>Parameter Type</b>	String
<b>Parameter Class</b>	Dynamic, scope= ALTER SYSTEM
<b>Default Value</b>	Low
<b>Range of Values</b>	False, low, high

**HASH\_AREA\_SIZE**

HASH\_AREA\_SIZE specifies the maximum amount of memory, in bytes, to be used for hash joins. If this parameter is not set, its value defaults to twice the value of the SORT\_AREA\_SIZE parameter.

<b>Parameter Type</b>	Integer
<b>Parameter Class</b>	Dynamic, scope= ALTER SESSION
<b>Default Value</b>	2 * SORT_AREA_SIZE
<b>Range of Values</b>	0–system-dependent value

**HASH\_JOIN\_ENABLED**

HASH\_JOIN\_ENABLED specifies whether the optimizer should consider using a hash join as a join method. When set to FALSE, hash join is turned off; that is, it is not available as a join method that the optimizer can consider choosing. When set to TRUE, the optimizer will compare the cost of a hash join to other types of joins, and choose it if it gives the best cost.

<b>Parameter Type</b>	Boolean
<b>Parameter Class</b>	Dynamic, scope= ALTER SESSION
<b>Default Value</b>	TRUE
<b>Range of Values</b>	TRUE, FALSE

**HASH\_MULTIBLOCK\_IO\_COUNT**

HASH\_MULTIBLOCK\_IO\_COUNT specifies how many sequential blocks a hash join reads and writes in one IO. When operating in multithreaded server mode, however, this parameter is ignored (a value of 1 is used even if you set the parameter to another value).

The maximum value for HASH\_MULTIBLOCK\_IO\_COUNT varies by operating system. It is always less than the operating system's maximum I/O size expressed as Oracle blocks (max\_IO\_size/DB\_BLOCK\_SIZE).

This parameter strongly affects performance because it controls the number of partitions into which the input is divided. If you change the parameter value, try to make sure that the following formula remains true:

$$R / M \leq \text{Po2}(M/C)$$

where:

R = size of(left input to the join)

M = HASH\_AREA\_SIZE \* 0.9

Po2(n) = largest power of 2 that is smaller than n

C = HASH\_MULTIBLOCK\_IO\_COUNT \* DB\_BLOCK\_SIZE

<b>Parameter Type</b>	Integer
<b>Parameter Class</b>	Dynamic, scope= ALTER SESSION, ALTER SYSTEM
<b>Default Value</b>	1
<b>Range of Values</b>	Operating system–dependent

### INSTANCE\_NAME (New in 8.1)

INSTANCE\_NAME is a string value representing the name of the instance and is used to uniquely identify a specific instance when multiple instances share common services names. INSTANCE\_NAME should not be confused with the SID, which actually uniquely identifies the instances shared memory on a host.

<b>Parameter Type</b>	String
<b>Parameter Class</b>	Static
<b>Default Value</b>	SID
<b>Range of Values</b>	Any alphanumeric characters

**LARGE\_POOL\_MIN\_ALLOC (Obsolete in 8.1.3)**

LARGE\_POOL\_MIN\_ALLOC specifies the minimum allocation size from the large pool. The value of the parameter can be specified in megabytes or kilobytes.

LARGE\_POOL\_MIN\_ALLOC can accept a numerical value or a number followed by the suffix “K” or “M” where “K” means “multiply by 1000” and “M” means “multiply by 1000000.”

<b>Parameter Type</b>	String
<b>Parameter Class</b>	Static
<b>Default Value</b>	16K
<b>Range of Values</b>	Minimum: 16 KB Maximum: ~64M

**LARGE\_POOL\_SIZE**

The parameter LARGE\_POOL\_SIZE lets you specify the size of the large pool allocation heap. The default size is 0, and the minimum size is 300K or LARGE\_POOL\_MIN\_ALLOC, whichever is larger. The value of the parameter can be specified in megabytes or kilobytes. If specified, the large pool is used for session memory if running with the multithreaded server. It is also used for IO buffers during backup operations.

LARGE\_POOL\_SIZE can accept a numerical value or a number followed by the suffix “K” or “M” where “K” means “multiply by 1000” and “M” means “multiply by 1,000,000.”

<b>Parameter Type</b>	String
<b>Parameter Class</b>	sStatic
<b>Default Value</b>	0
<b>Range of Values</b>	Minimum: 300 KB or LARGE_POOL_MIN_ALLOC, whichever is larger Maximum: At least 2 GB (maximum is operating system-specific)

**LGWR\_IO\_SLAVES (Obsolete in 8.1.3)**

LGWR\_IO\_SLAVES specifies the number of I/O slaves used by the LGWR process. The LGWR process and its slaves always write to disk. By default the value is 0 and I/O slaves are not used.

Typically I/O slaves are used to “simulate” asynchronous I/O on platforms that do not support asynchronous I/O or implement it inefficiently. However, I/O slaves can be used even when asynchronous I/O is being used. In that case the I/O slaves will use asynchronous I/O.

The default value is almost always adequate.

<b>Parameter Type</b>	Integer
<b>Parameter Class</b>	Static
<b>Default Value</b>	0
<b>Range of Values</b>	0–system-dependent value

**LOG\_ARCHIVE\_BUFFER\_SIZE (Obsolete in 8.1.3)**

LOG\_ARCHIVE\_BUFFER\_SIZE specifies the size of each archival buffer, in redo log blocks (operating system blocks). The default should be adequate for most applications. This parameter, with LOG\_ARCHIVE\_BUFFERS, can be used to tune archiving.

<b>Parameter Type</b>	Integer
<b>Parameter Class</b>	Static
<b>Default Value</b>	Operating system-dependent
<b>Range of Values</b>	1–operating system–dependent (in operating system blocks) Multiple instances: Can have different values



**LOG\_ARCHIVE\_BUFFERS (Obsolete in 8.1.3)**

LOG\_ARCHIVE\_BUFFERS specifies the number of buffers to allocate for archiving. The default should be adequate for most applications.

This parameter, with LOG\_ARCHIVE\_BUFFER\_SIZE, can tune archiving so that it runs as fast as necessary, but not so fast that it reduces system performance.

<b>Parameter Type</b>	Integer
<b>Parameter Class</b>	Static
<b>Default Value</b>	Operating system–dependent
<b>Range of Values</b>	Operating system–dependent Multiple instances: Can have different values

**LOG\_ARCHIVE\_DEST (Modified in 8.1)**

In 8.0, LOG\_ARCHIVE\_DEST is applicable only if you are using the redo log in ARCHIVELOG mode. Use a text string to specify the default location and root of the disk file or tape device when archiving redo log files. (Archiving to tape is not supported on all operating systems.) The value cannot be a raw partition.

In release 8.1, deprecated in favor of LOG\_ARCHIVE\_DEST\_n when Oracle Enterprise Edition is installed. If Oracle Enterprise Edition is not installed or it is installed but you have not specified LOG\_ARCHIVE\_DEST\_n, this parameter can be used as described below. LOG\_ARCHIVE\_DEST is incompatible with the LOG\_ARCHIVE\_DEST\_n parameters, and must be defined as the NULL string (") or (") when any LOG\_ARCHIVE\_DEST\_n parameter has a non\_NULL string value. Use a text string to specify the default location and root of the disk file or tape device when archiving redo log files. (Archiving to tape is not supported on all operating systems.) The value cannot be a raw partition. If LOG\_ARCHIVE\_DEST is not explicitly defined and all the LOG\_ARCHIVE\_DEST\_n parameters have NULL string values, LOG\_ARCHIVE\_DEST is set to an operating system-specific default value on instance startup.

To override the destination that this parameter specifies, either specify a different destination for manual archiving or use the SQL\*Plus command **ARCHIVE LOG START filespec** for automatic archiving, where **filespec** is the new archive destination. To permanently change the destination, use the command **ALTER SYSTEM SET LOG\_ARCHIVE\_DEST = filespec**, where **filespec** is the new archive destination.

<b>Parameter Type</b>	String
<b>Parameter Class</b>	Static (in 8.0), dynamic, scope = ALTER SYSTEM (in 8.1)
<b>Default Value</b>	Operating system–dependent (in 8.0), NULL string default (in 8.1)
<b>Range of Values</b>	Any valid path or device name, except raw partitions (or NULL in 8.1)

### **LOG\_ARCHIVE\_DEST\_n (New in 8.1)**

This parameter is valid only if you have installed the Oracle Enterprise Edition. You can continue to use **LOG\_ARCHIVE\_DEST** if you have installed the Oracle Enterprise Edition; however, you cannot use both **LOG\_ARCHIVE\_DEST\_n** and **LOG\_ARCHIVE\_DEST**, because they are not compatible.

**LOG\_ARCHIVE\_DEST\_n** defines a destination and attributes for the archive redo log file group. The parameter suffix (1 through 5) specifies one of the 5 corresponding **LOG\_ARCHIVE\_DEST\_n** destination parameters. The parameter number suffix is defined as the “handle” displayed by the fixed-table queries.

**LOG\_ARCHIVE\_DEST\_n** = ("null\_string" | **SERVICE**=tnsnames-service | **LOCATION**=directory-spec)[**MANDATORY** | **OPTIONAL**] [**REOPEN**=integer]

**SERVICE** specifies a standby destination. Net8 (IPC or TCP) will be used to transmit the archivelog. There must be a standby instance associated with the destination.

tnsnames-service corresponds to an appropriate service name in **TNSNAMES.ORA**.

**LOCATION** specifies a local file-system destination.

**MANDATORY** specifies that archiving to the destination must succeed before the **REDO** logfile can be made available for reuse.

OPTIONAL specifies that successful archiving to the destination is not required before the REDO log file can be made available for re-use. If the “must succeed count” (LOG\_ARCHIVE\_MIN\_SUCCEED\_DEST) is met the REDO logfile is marked for re-use. This is the default REOPEN specifies an interval of time (in seconds) that must pass after an error has been encountered during archiving to the destination before future archives to the destination can be attempted. Future attempts are made when the next REDO logfile is archived. If a destination is MANDATORY Oracle recommend that you specify a REOPEN time that reduces the possibility of primary database shutdown due to lack of available online REDO logfiles.

Following is an example scenario:

```
LOG_ARCHIVE_MIN_SUCCEED_DEST=2
LOG_ARCHIVE_DEST_1='SERVICE=standby1 OPTIONAL REOPEN=120'
LOG_ARCHIVE_DEST_2='LOCATION=filespec MANDATORY REOPEN=5'
LOG_ARCHIVE_DEST_3='SERVICE=standby2 OPTIONAL REOPEN=60'
LOG_ARCHIVE_DEST_STATE_1=enable
LOG_ARCHIVE_DEST_STATE_2=enable
LOG_ARCHIVE_DEST_STATE_3=enable
```

In the above example, destination 1 and 3 are standby destinations, both are optional. Destination 1 has a reopen interval of 2 minutes, destination 3 has a reopen interval of 1 minute. Destination 2 is a local destination, completion is mandatory and a five second reopen interval is specified.

All three destinations are enabled and therefore available to ARCHIVELOG operations as target destinations.

The minimum number of destinations that must archive successfully for the redo log to be marked for reuse is set to two. This means, in addition to destination 2 (the mandatory destination) either destination 1 and/or destination 3 must also archive successfully.

<b>Parameter Type</b>	String
<b>Parameter Class</b>	Dynamic, scope = ALTER SYSTEM, ALTER SESSION
<b>Default Value</b>	NULL
<b>Range of Values</b>	Valid keyword definitions

**LOG\_ARCHIVE\_DEST\_STATE\_n (New in 8.1)**

LOG\_ARCHIVE\_DEST\_STATE\_n specifies the availability state of the corresponding destination. The parameter suffix (1 through 5) specifies one of the 5 corresponding LOG\_ARCHIVE\_DEST\_n destination parameters.

If enabled, a valid log archive destination can be used for a subsequent archiving operation (automatic or manual). If deferred, any valid destination information and attributes are preserved, but the destination is excluded from archiving operations until re-enabled.

The LOG\_ARCHIVE\_DEST\_STATE\_n parameters have no effect on the Enable state for the LOG\_ARCHIVE\_DEST or LOG\_ARCHIVE\_DUPLEX\_DEST parameters.

Changed by ALTER SYSTEM SET LOG\_ARCHIVE\_DEST\_STATE\_n = value or ALTER SESSION SET LOG\_ARCHIVE\_DEST\_STATE\_n = value. For example: ALTER SESSION SET LOG\_ARCHIVE\_DEST\_STATE\_n = enable ALTER SESSION effectively hides the system level value for the session issuing the command. The system level value can only be reestablished by an ALTER SESSION using the current system level value. Always refer to V\$ARCHIVE\_DEST for values in use for the current session.

Corresponds to V\$ARCHIVE\_DEST index n.

<b>Parameter Type</b>	String
<b>Parameter Class</b>	Dynamic, scope = ALTER SYSTEM, ALTER SESSION
<b>Default Value</b>	ENABLE
<b>Range of Values</b>	ENABLE, DEFER

**LOG\_ARCHIVE\_DUPLEX\_DEST (Modified in 8.1)**

In 8.0, LOG\_ARCHIVE\_DUPLEX\_DEST is similar to the initialization parameter LOG\_ARCHIVE\_DEST. This parameter specifies a second archive destination: the duplex archive destination. This duplex archive destination can be either a must-succeed or a best-effort archive destination, depending on how many archive destinations must succeed.

If LOG\_ARCHIVE\_DUPLEX\_DEST is set to be a NULL string (" ") or ( ' ' ), it means there is no duplex archive destination. The default of this parameter is a NULL string.

In release 8.1, this is deprecated in favor of LOG\_ARCHIVE\_DEST\_n when Oracle Enterprise Edition is installed. If Oracle Enterprise Edition is not installed, this parameter is valid.

LOG\_ARCHIVE\_DUPLEX\_DEST is similar to the initialization parameter LOG\_ARCHIVE\_DEST.

<b>Parameter Type</b>	String
<b>Parameter Class</b>	Dynamic, scope = ALTER SYSTEM
<b>Default Value</b>	a NULL string
<b>Range of Values</b>	Either a NULL string or any valid path or device name, except raw partitions

## LOG\_ARCHIVE\_FORMAT

LOG\_ARCHIVE\_FORMAT is applicable only if you are using the redo log in ARCHIVELOG mode. Use a text string and variables to specify the default filename format when archiving redo log files. The string generated from this format is appended to the string specified in the LOG\_ARCHIVE\_DEST parameter. The following variables can be used in the format:

- %s log sequence number
- %t thread number

Using uppercase letters (for example, %S) for the variables causes the value to be a fixed length padded to the left with zeros.

The following is an example of specifying the archive redo log filename format:

LOG\_ARCHIVE\_FORMAT = "LOG%s\_%t.ARC".

<b>Parameter Type</b>	String
<b>Parameter Class</b>	Static
<b>Default Value</b>	Operating system–dependent (length for uppercase variables is also operating system-dependent)
<b>Range of Values</b>	Any valid filename Multiple instances: Can have different values, but identical values are recommended

**LOG\_ARCHIVE\_MAX\_PROCESSES (New in 8.1)**

LOG\_ARCHIVE\_MAX\_PROCESSES specifies the number of ARCH processes to be invoked. This value is evaluated at instance startup if the LOG\_ARCHIVE\_START initialization parameter has the value TRUE; otherwise, this parameter is evaluated when the ARCH process is invoked via SQL\*Plus or SQL syntax.

<b>Parameter Type</b>	Integer
<b>Parameter Class</b>	Dynamic, scope = ALTER SYSTEM
<b>Default Value</b>	1
<b>Range of Values</b>	Any integer from 1 to 10 inclusive

**LOG\_ARCHIVE\_MIN\_SUCCEED\_DEST (Modified in 8.1)**

LOG\_ARCHIVE\_MIN\_SUCCEED\_DEST specifies the minimum number of archive log destinations that must succeed in order for the online logfile to be available for reuse. When automatic archiving is enabled, the allowable values are 1 and 2 in 8.0, cannot be more than the total number of destinations, and the number of enabled, valid MANDATORY destinations plus the number of enabled, valid non-standby OPTIONAL destinations in 8.1.

In 8.0, if this parameter is 1, LOG\_ARCHIVE\_DEST is a must-succeed destination and LOG\_ARCHIVE\_DUPLEX\_DEST is a best-effort destination. If this parameter is 2, both LOG\_ARCHIVE\_DEST and LOG\_ARCHIVE\_DUPLEX\_DEST are must-succeed destinations.

In 8.1, If the value is less than the number of enabled, valid MANDATORY destinations, it will be ignored in favor of the MANDATORY destination count. If the value is more than the number of enabled, valid MANDATORY destinations, some of the enabled, valid OPTIONAL non-standby destinations will essentially be treated as MANDATORY.

ALTER SESSION SET LOG\_ARCHIVE\_MIN\_SUCCEED\_DEST = n cannot be used when LOG\_ARCHIVE\_DEST or LOG\_ARCHIVE\_DUPLEX\_DEST are in use.

Changed by ALTER SYSTEM SET LOG\_ARCHIVE\_MIN\_SUCCEED\_DEST = number or ALTER SESSION SET LOG\_ARCHIVE\_MIN\_SUCCEED\_DEST = number. ALTER SESSION effectively hides the system level value for the session issuing the command. The system level value can only be re-established by an ALTER SESSION using the current system level value.

LOG\_ARCHIVE\_MIN\_SUCCEED\_DEST can be set to 1, and all destinations can be set to the null string as a necessary transitory state used to dynamically switch between the destinations specified by the LOG\_ARCHIVE\_DEST, LOG\_ARCHIVE\_DUPLEX\_DEST parameters and the destinations specified by the LOG\_ARCHIVE\_DEST\_n parameters.

<b>Parameter Type</b>	Integer
<b>Parameter Class</b>	Dynamic, scope = ALTER SESSION (in 8.1), ALTER SYSTEM
<b>Default Value</b>	1
<b>Range of Values</b>	1–2 (in 8.0) 1–5 (restricted to 1–2 when used with LOG_ARCHIVE_DEST and LOG_ARCHIVE_DUPLEX_DEST) (in 8.1)

### LOG\_ARCHIVE\_START

LOG\_ARCHIVE\_START is applicable only when you use the redo log in ARCHIVELOG mode, LOG\_ARCHIVE\_START indicates whether archiving should be automatic or manual when the instance starts up. TRUE indicates that archiving is automatic. FALSE indicates that the database administrator will archive filled redo log files manually. (The SQL\*Plus command ARCHIVE LOG START or STOP overrides this parameter.)

In ARCHIVELOG mode, if all online redo log files fill without being archived, an error message is issued, and instance operations are suspended until the necessary archiving is performed. This delay is more likely if you use manual archiving. You can reduce its likelihood by increasing the number of online redo log files.

To use ARCHIVELOG mode while creating a database, set this parameter to TRUE. Normally, a database is created in NOARCHIVELOG mode and then altered to ARCHIVELOG mode after creation.

<b>Parameter Type</b>	Boolean
<b>Parameter Class</b>	Static
<b>Default Value</b>	FALSE
<b>Range of Values</b>	TRUE/FALSE Multiple instances: Can have different values

### LOG\_BLOCK\_CHECKSUM (Obsolete in 8.1.3)

If LOG\_BLOCK\_CHECKSUM is TRUE, then every log block will be given a checksum before it is written to the current log.

Warning: Setting LOG\_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.

<b>Parameter Type</b>	Boolean
<b>Parameter Class</b>	Static
<b>Default Value</b>	FALSE
<b>Range of Values</b>	TRUE/FALSE

## LOG\_BUFFER

LOG\_BUFFER specifies the amount of memory, in bytes, that is used when buffering redo entries to a redo log file. Redo log entries contain a record of the changes that have been made to the database block buffers. The LGWR process writes redo log entries from the log buffer to a redo log file.

In general, larger values for LOG\_BUFFER reduce redo log file I/O, particularly if transactions are long or numerous. In a busy system, the value 65536 or higher would not be unreasonable.

<b>Parameter Type</b>	Integer
<b>Parameter Class</b>	Static
<b>Default Value</b>	Operating system-dependent
<b>Range of Values</b>	Operating system-dependent

## LOG\_CHECKPOINT\_INTERVAL (Modified in 8.1)

In 8.0, LOG\_CHECKPOINT\_INTERVAL specifies the frequency of checkpoints in terms of the number of redo log file blocks that are written between consecutive checkpoints.

Regardless of this value, a checkpoint always occurs when switching from one online redo log file to another. If the value exceeds the actual redo log file size, checkpoints occur only when switching logs. The checkpoint frequency is one of the factors which impacts the time required for the database to recover from an unexpected failure.

Extremely frequent checkpointing can cause excessive writes to disk, possibly impacting transaction performance. In addition, if the intervals are so close together that the interval checkpoint requests are arriving at a rate faster than the rate at which Oracle can satisfy these requests, Oracle can choose to ignore some of these requests in order to avoid excessive interval checkpointing activity.

The number of times DBW $n$  has been notified to do a checkpoint for a given instance is shown in the cache statistic DBW $n$  checkpoints, which is displayed in the System Statistics Monitor of the Enterprise Manager.



Note that specifying a value of 0 (zero) for the interval might cause interval checkpoints to be initiated very frequently since a new request will be started even if a single redo log buffer is written since the last request was initiated. Hence, setting the value to 0 is not recommended.

In 8.1, when LOG\_CHECKPOINT\_INTERVAL is set, the target for checkpoint position cannot lag the end of the log by more than the number of redo log blocks specified by this parameter. This ensures that no more than a fixed number of redo blocks will need to be read during instance recovery.

<b>Parameter Type</b>	Integer
<b>Parameter Class</b>	Dynamic, scope = ALTER SYSTEM
<b>Default Value</b>	Operating system–dependent
<b>Range of Values</b>	Unlimited (operating-system blocks, not database blocks) Multiple instances: Can have different values

### LOG\_CHECKPOINT\_TIMEOUT (Modified in 8.1)

In 8.0, LOG\_CHECKPOINT\_TIMEOUT specifies the maximum amount of time before another checkpoint occurs. The value is specified in seconds. The time begins at the start of the previous checkpoint, then a checkpoint occurs after the amount of time specified by this parameter.

Specifying a value of 0 for the timeout disables time-based checkpoints. Hence, setting the value to 0 is not recommended.

**Note:** A checkpoint scheduled to occur because of this parameter is delayed until the completion of the previous checkpoint if the previous checkpoint has not yet completed.

In 8.1, when LOG\_CHECKPOINT\_TIMEOUT is set, this parameter sets the target for checkpoint position to a location in the log file where the end of the log was this many seconds ago. This ensures that no more than the specified number of seconds worth of redo log blocks need to be read during instance recovery.

<b>Parameter Type</b>	Integer
<b>Parameter Class</b>	Dynamic, scope = ALTER SYSTEM
<b>Default Value</b>	0 seconds (in 8.0) 900 seconds. Enterprise Edition: 1800 seconds (in 8.1)
<b>Range of Values</b>	0–unlimited Multiple instances: Can have different values

**LOG\_CHECKPOINTS\_TO\_ALERT**

LOG\_CHECKPOINTS\_TO\_ALERT enables you to log your checkpoints to the alert file. This parameter is useful to determine if checkpoints are occurring at the desired frequency.

<b>Parameter Type</b>	Boolean
<b>Parameter Class</b>	Static
<b>Default Value</b>	FALSE
<b>Range of Values</b>	TRUE/FALSE

**LOG\_FILES (Obsolete in 8.1.3)****LOG\_SIMULTANEOUS\_COPIES (Obsolete in 8.1.3)**

LOG\_SIMULTANEOUS\_COPIES specifies the maximum number of redo buffer copy latches available to write log entries simultaneously. For good performance, you can have up to twice as many redo copy latches as CPUs. For a single-processor system, set to zero so that all log entries are copied on the redo allocation latch.

.....

If this parameter is set to 0, redo copy latches are turned off, and the parameters LOG\_ENTRY\_PREBUILD\_THRESHOLD and LOG\_SMALL\_ENTRY\_MAX\_SIZE are ignored.

<b>Parameter Type</b>	Integer
<b>Parameter Class</b>	Static
<b>Default Value</b>	CPU_COUNT
<b>Range of Values</b>	0–unlimited

### **LOG\_SMALL\_ENTRY\_MAX\_SIZE (Obsolete in 8.1.3)**

LOG\_SMALL\_ENTRY\_MAX\_SIZE specifies the size in bytes of the largest copy to the log buffers that can occur under the redo allocation latch without obtaining the redo buffer copy latch. If the value for LOG\_SIMULTANEOUS\_COPIES is 0, this parameter is ignored (all writes are “small” and are made without the copy latch).

If the redo entry is copied on the redo allocation latch, the user process releases the latch after the copy. If the redo entry is larger than this parameter, the user process releases the latch after allocating space in the buffer and getting a redo copy latch.

<b>Parameter Type</b>	String
<b>Parameter Class</b>	Static
<b>Default Value</b>	NULL
<b>Range of Values</b>	-

**MAX\_DUMP\_FILE\_SIZE**

MAX\_DUMP\_FILE\_SIZE specifies the maximum size of trace files to be written. Change this limit if you are concerned that trace files may take up too much space.

MAX\_DUMP\_FILE\_SIZE can accept a numerical value or a number followed by the suffix “K” or “M” where “K” means “multiply by 1000” and “M” means “multiply by 1000000.” A numerical value for MAX\_DUMP\_FILE\_SIZE specifies the maximum size in operating system blocks, whereas a number followed by a “K” or “M” suffix specifies the file size in number of bytes. MAX\_DUMP\_FILE\_SIZE can also assume the special value string UNLIMITED. UNLIMITED means that there is no upper limit on trace file size, thus dump files can be as large as the operating system permits.

<b>Parameter Type</b>	String
<b>Parameter Class</b>	Dynamic, scope = ALTER SYSTEM, ALTER SYSTEM DEFERRED, ALTER SESSION
<b>Default Value</b>	10000 blocks
<b>Range of Values</b>	0-unlimited

**MTS\_DISPATCHERS (Modified in 8.1)**

MTS\_DISPATCHERS lets the database administrator enable various attributes for each dispatcher.

<b>Parameter Type</b>	String
<b>Parameter Class</b>	Dynamic, scope = ALTER SYSTEM
<b>Default Value</b>	NULL

In Oracle 7.3, the database administrator could specify a protocol and an initial number of dispatchers. These attributes are specified in a position-dependent, comma-separated string assigned to MTS\_DISPATCHERS.

For example:

```
MTS_DISPATCHERS = "TCP, 3"
```

While remaining backwardly compatible with this format, the parsing software in Oracle8 supports a name-value syntax (similar to the syntax used by Net8) to enable the specification of the existing and additional attributes in a position-independent case-insensitive manner. For example:

```
MTS_DISPATCHERS = "(PROTOCOL=TCP)(DISPATCHERS=3)"
```

One and only one of the following attributes is required: ADDRESS, DESCRIPTION, or PROTOCOL.

Attribute	Description
ADDRESS (ADD or ADDR)	The network address (in Net8 syntax) of the end point on which the dispatchers will listen (Includes the protocol.)
DESCRIPTION (DES or DESC)	The network description (in Net8 syntax) of the end point which the dispatchers will listen (Includes the protocol.)
PROTOCOL (PRO or PROT)	The network protocol for which the dispatchers will generate a listening end point

The ADDRESS and DESCRIPTION attributes provides support for the specification of additional network attributes. (This enables support of multi-homed hosts.)

The attributes CONNECTIONS, DISPATCHERS, LISTENER, MULTIPLEX, POOL, SERVICE, and TICKS are optional:

Attribute	Description
CONNECTIONS (CON or CONN)	The maximum number of network connections to allow for each dispatcher  Default is set by Net8 and is platform-specific
DISPATCHERS (DIS or DISP)	The initial number of dispatchers to start; default is 1
LISTENER (LIS, LIST)	The network name of an address or address list of the Net8 listeners with which the dispatchers will register  The LISTENER attribute makes it easier to administer multi-homed hosts. This attribute specifies the appropriate listeners with which the dispatchers will register. The LISTENER attribute overrides the LOCAL_LISTENER parameter.
MULTIPLEX (MUL or MULT)	Used to enable the Net8 Network Session Multiplex feature  If 1, ON, Yes, TRUE, or BOTH is specified, then Network Session Multiplex is enabled for both incoming and outgoing network connections.  If IN is specified, then Network Session Multiplex is enabled for incoming network connections.  If OUT is specified, then Network Session Multiplexing is enabled for outgoing network connections.  If 0, NO, OFF, or FALSE is specified, then Network Session Multiplexing is disabled for both incoming and outgoing network connections.  The default “Network Session Multiplex” is disabled on both incoming and outgoing network connections.

Attribute	Description
PRESENTATION (PRE or PRES) only in release 8.1	Used to enable support of specific presentation protocols (If GIOP is specified, the dispatcher will listen on the specified protocol for GIOP messages. For IIOP support, the protocol specified must be TCP. The default presentation is “TTC”.)
POOL (POO)	<p>Used to enable the Net8 Connection Pooling feature.</p> <p>If a number is specified, then Connection Pooling is enabled for both incoming and outgoing network connections and the number specified is the timeout in ticks for both incoming and outgoing network connections.</p> <p>If ON, YES, TRUE, or BOTH is specified, then Connection Pooling is enabled for both incoming and outgoing network connections and the default timeout (set by Net8) will be used for both incoming and outgoing network connections.</p> <p>If IN is specified, then Connection Pooling is enabled for incoming network connections and the default timeout (set by Net8) will be used for incoming network connections.</p> <p>If OUT is specified, then Connection Pooling is enabled for outgoing network connections and the default timeout (set by Net8) will be used for outgoing network connections.</p> <p>If NO, OFF, or FALSE is specified, then Connection Pooling is disabled for both incoming and outgoing network connections.</p> <p>POOL can also be assigned a name-value string such as: “(IN=10)”, “(OUT=20)”, or “(IN=10)(OUT=20)”, in which case, if an IN numeric value is specified, then Connection Pooling is enabled for incoming connections and the number specified is the timeout in ticks for incoming network connections. If an “OUT” numeric value is specified, then “Connection Pooling” is enabled for outgoing network connections and the number specified is the timeout in ticks for outgoing network connections.</p> <p>If the numeric value of a specified timeout is 0, then the default value (set by Net8) will be used.</p> <p>The default Connection Pooling is disabled on both incoming and outgoing network connections.</p>
SERVICE (SER, SERV)	<p>The service name which the dispatchers register with the Net8 listeners.</p> <p>The SERVICE attribute overrides the MTS_SERVICE parameter. This attribute specifies a service name that the dispatchers will use to register.</p>

Attribute	Description
SESSIONS (SES or SESS)	The maximum number of network sessions to allow for each dispatcher The default is set by Net8 and is platform-specific.
TICKS (TIC or TICK)	The size of a network tick in seconds (The default is set by Net8 and is platform-specific.)

### MTS\_LISTENER\_ADDRESS (Obsolete in 8.1.3)

MTS\_LISTENER\_ADDRESS specifies the configuration of the Listener process. The Listener process requires an address to listen for connection requests for each network protocol that is used on your system. Addresses are specified as the Net8 description of the connection address.

Warning: Each address must be specified with its own parameter. (This differs from the Net8 syntax.) For example, if you use TCP/IP as well as DECNet, you would provide specifications similar to the following in your initialization file:

```
MTS_LISTENER_ADDRESS = \
    " ( ADDRESS= ( PROTOCOL=tcp ) ( HOST=myhost ) ( PORT=7002 ) ) "
MTS_LISTENER_ADDRESS = \
    " ( ADDRESS= ( PROTOCOL=decnet ) ( NODE=name ) ( OBJECT=mts ) ) "
```

**Note:** If you have multiple MTS\_LISTENER\_ADDRESS parameters, they must be adjacent to each other in your initialization file.

Address specifications for the Listener process are operating system-specific and network protocol-specific.

MTS\_LISTENER\_ADDRESS is obsolete but is supported for backward compatibility. The functionality of MTS\_LISTENER\_ADDRESS has been replaced with the LOCAL\_LISTENER parameter and LISTENER attribute of the MTS\_DISPATCHERS parameter.

<b>Parameter Type</b>	String
<b>Parameter Class</b>	Static
<b>Default Value</b>	NULL
<b>Range of Values</b>	-

**MTS\_MAX\_DISPATCHERS**

MTS\_MAX\_DISPATCHERS specifies the maximum number of dispatcher processes allowed to be running simultaneously.

<b>Parameter Type</b>	Integer
<b>Parameter Class</b>	Static
<b>Default Value</b>	If dispatchers are configured, then defaults to whichever is greater: 5 or the number of dispatchers configured
<b>Range of Values</b>	Operating system-dependent

**MTS\_MAX\_SERVERS**

MTS\_MAX\_SERVERS specifies the maximum number of shared server processes allowed to be running simultaneously. EVENT

<b>Parameter Type</b>	Integer
<b>Parameter Class</b>	Static
<b>Default Value</b>	Defaults to whichever is greater: 20 or 2 times the value of MAX_SERVERS
<b>Range of Values</b>	Operating system-dependent

**MTS\_MULTIPLE\_LISTENERS (Obsolete in 8.1.3)****MTS\_SERVICE (Obsolete in 8.1.3)**

MTS\_SERVICE specifies the name of the service you want to be associated with the dispatcher. Using this name in the CONNECT string allows users to connect to an instance through a dispatcher. Oracle always checks for such a service before establishing a normal database connection.

The name you specify must be unique. It should not be enclosed in quotation marks. It is a good idea for this name to be the same as the instance name. That way, if the dispatcher is unavailable for any reason, the CONNECT string will still connect the user to the database.

If not specified, MTS\_SERVICE defaults to the value specified by DB\_NAME. If DB\_NAME also is not specified, the Oracle Server returns an error at startup indicating that the value for this parameter is missing.

<b>Parameter Type</b>	String
<b>Parameter Class</b>	Static
<b>Default Value</b>	NULL
<b>Range of Values</b>	-



## OPEN\_CURSORS

OPEN\_CURSORS specifies the maximum number of open cursors (context areas) a session can have at once. This constrains a session from opening an excessive number of cursors. Assuming that a session does not open the number of cursors specified by OPEN\_CURSORS, there is no added overhead by setting this value too high.

It is important to have the value of OPEN\_CURSORS set high enough to prevent your application from running out of open cursors.

The number varies from one application to another.

This parameter also constrains the size of the PL/SQL cursor cache which PL/SQL uses to avoid having to reparse as statements are reexecuted by a user.

<b>Parameter Type</b>	Integer
<b>Parameter Class</b>	Static
<b>Default Value</b>	50
<b>Range of Values</b>	1–operating system limit

## OPTIMIZER\_MODE

OPTIMIZER\_MODE specifies the behavior of the optimizer. When set to RULE, this parameter causes rule-based optimization to be used unless hints are specified in the query. When set to CHOOSE, the optimizer uses the cost-based approach for a SQL statement if there are statistics in the dictionary for at least one table accessed in the statement. (Otherwise, the rule-based approach is used.)

You can set the goal for cost-based optimization by setting this parameter to FIRST\_ROWS or ALL\_ROWS. FIRST\_ROWS causes the optimizer to choose execution plans that minimize response time. ALL\_ROWS causes the optimizer to choose execution plans that minimize total execution time.

<b>Parameter Type</b>	Integer
<b>Parameter Class</b>	Dynamic, scope=ALTER SESSION
<b>Default Value</b>	CHOOSE
<b>Range of Values</b>	RULE/CHOOSE/FIRST_ROWS/ALL_ROWS

## PARALLEL\_AUTOMATIC\_TUNING (New in 8.1)

**Note:** This parameter applies to Parallel Execution, not the Oracle8i Parallel Server Option.

Enable `PARALLEL_AUTOMATIC_TUNING` when you want Oracle to determine the default values for parameters that control Parallel Execution. In addition to setting this parameter, you must enable `PARALLEL`, for the target tables in the system. All subsequent tuning will be done by the system.

If you used Parallel Execution in a previous release and are now enabling `PARALLEL_AUTOMATIC_TUNING`, you should reduce the amount of memory allocated from the Shared Pool to account for the decreased demand on that pool. This memory will be allocated from the Large Pool, and will be computed automatically if `LARGE_POOL_SIZE` is left unset.

This will include setting the `PARALLEL_ADAPTIVE_MULTI_USER` parameter which will override user-provided hints in favor of maintaining the load on the system within acceptable ranges. The database administrator can override any of the system-provided defaults if desired.

<b>Parameter Type</b>	Boolean
<b>Parameter Class</b>	Static
<b>Default Value</b>	FALSE
<b>Range of Values</b>	TRUE, FALSE

**PARALLEL\_THREADS\_PER\_CPU (New in 8.1)**

**Note:** This parameter applies to Parallel Execution, not the Oracle8i Parallel Server option.

PARALLEL\_THREADS\_PER\_CPU is used to set the default degree of parallelism, and to tune the parallel adaptive and load balancing algorithms. The parameter describes the number of processes or threads that a CPU can handle during parallel execution. The default is platform-dependent. The default provided by the system should be adequate for most cases. This number should be decreased from the default provided if the machine appears to be overloaded when a representative query is executed. The value for this parameter should be increased if the system is I/O bound.

<b>Parameter Type</b>	Integer
<b>Parameter Class</b>	Dynamic, scope = ALTER SYSTEM
<b>Default Value</b>	OS-dependent, usually 2
<b>Range of Values</b>	Any non-zero number

**PROCESSES**

For a multiple-process operation, PROCESSES specifies the maximum number of operating system user processes that can simultaneously connect to an Oracle Server. This value should allow for all background processes such as LCK processes, Job Queue processes, and Parallel Query processes.

The default values of SESSIONS is derived from PROCESSES. If you alter the value of PROCESSES, you may want to adjust the values of this derived parameters.

<b>Parameter Type</b>	Integer
<b>Parameter Class</b>	Static
<b>Default Value</b>	30
<b>Range of Values</b>	6–operating system–dependent

**QUERY\_REWRITE\_ENABLED**

QUERY\_REWRITE\_ENABLED allows you to enable or disable query rewriting. Query rewriting is enabled for a particular materialized view only if both the session parameter and the individual materialized view are enabled and when cost-based optimization is enabled.

<b>Parameter Type</b>	bBoolean
<b>Parameter Class</b>	Dynamic, scope = ALTER SYSTEM, ALTER SESSION
<b>Default Value</b>	FALSE
<b>Range of Values</b>	TRUE, FALSE

**QUERY\_REWRITE\_INTEGRITY**

QUERY\_REWRITE\_INTEGRITY determines the degree to which query rewriting must be enforced by the Oracle server. In the safest level, query rewrite transformations that rely on unenforced relationships are not used.

With ENFORCE, consistency and integrity are enforced and guaranteed by Oracle. With NO\_ENFORCE, rewrites are allowed using relationships that have been declared, but that are not enforced by Oracle. With USE\_STALE, rewrites are allowed using unenforced relationships, and materialized views are eligible for rewrite even if they are known to be inconsistent with the underlying detail data.

<b>Parameter Type</b>	String
<b>Parameter Class</b>	Dynamic, scope = ALTER SYSTEM, ALTER SESSION
<b>Default Value</b>	ENFORCE
<b>Range of Values</b>	ENFORCE, NO_ENFORCE, USE_STALE

**RESOURCE\_MANAGER\_PLAN (New in 8.1)**

This parameter dictates which top plan to use for this instance. The resource manager will load this top plan as well as all its descendants (subplans, directives and consumer groups). If the parameter is not specified, the resource manager is, by default, off.

The administrator may use the ALTER SYSTEM command on the parameter to turn on the resource manager (if it was previously off), turn off the resource manager or change the current plan schema (if it was previously on). If a plan is specified that does not exist in the data dictionary, an error message will be returned.

<b>Parameter Type</b>	String
<b>Parameter Class</b>	Dynamic, scope = ALTER SYSTEM
<b>Default Value</b>	NULL
<b>Range of Values</b>	Any valid character string

## SERVICE\_NAMES (New in 8.1)

SERVICE\_NAMES specifies the service names supported by the instance.

SERVICE\_NAMES is one or more strings which represent the names of the database on the network. Net8 8.1 wants a service name, rather than a SID, to identify a database. It is possible to provide multiple services names so that different usages of a single database can be identified separately. Service names can also be used to identify a single service that is available from two different databases through the use of replication.

Example:

SERVICE\_NAMES = sales.acme.com, widgetsales.acme.com

This value/values will be sent to the TNS Listeners as the service name(s) that this instance belongs to. If the names in this parameter are not domain qualified, they will be qualified with the value of the DB\_DOMAIN parameter before being sent to the listener. If this parameter is not specified, as of 8.1, the default value that will be registered is <db\_name>.<db\_domain>.

Using the above example, the client CONNECT\_DATA should read as follows:  
(CONNECT\_DATA=(SERVICE\_NAMES=widgetsales.acme.com)).

<b>Parameter Type</b>	String
<b>Parameter Class</b>	Static
<b>Default Value</b>	DB_NAME.DB_DOMAIN if defined
<b>Range of Values</b>	Any ASCII string, or comma-separated list of string names

## SESSION\_CACHED\_CURSORS

SESSION\_CACHED\_CURSORS lets you specify the number of session cursors to cache. Repeated parse calls of the same SQL statement cause the session cursor for that statement to be moved into the session cursor cache. Subsequent parse calls will find the cursor in the cache and need not reopen the cursor. The value of this parameter is the maximum number of session cursors to keep in the session cursor cache.

<b>Parameter Type</b>	Integer
<b>Parameter Class</b>	Dynamic, scope=ALTER SESSION
<b>Default Value</b>	0
<b>Range of Values</b>	0–operating system–dependent

**SHARED\_POOL\_RESERVED\_MIN\_ALLOC (Obsolete in 8.1.3)**

The value of SHARED\_POOL\_RESERVED\_MIN\_ALLOC controls allocation of reserved memory. Memory allocations larger than this value can allocate space from the reserved list if a chunk of memory of sufficient size is not found on the shared pool free lists.

The default value is adequate for most systems. If you increase the value, then the Oracle Server will allow fewer allocations from the reserved list and will request more memory from the shared pool list.

SHARED\_POOL\_RESERVED\_MIN\_ALLOC can accept a numerical value or a number followed by the suffix “K” or “M” where “K” means “multiply by 1000” and “M” means “multiply by 1000000.”

<b>Parameter Type</b>	String
<b>Parameter Class</b>	Static
<b>Default Value</b>	5000
<b>Range of Values</b>	5000–SHARED_POOL_RESERVED_SIZE (in bytes)

**SHARED\_POOL\_RESERVED\_SIZE**

SHARED\_POOL\_RESERVED\_SIZE specifies the shared pool space which is reserved for large contiguous requests for shared pool memory. This parameter, along with the SHARED\_POOL\_RESERVED\_MIN\_ALLOC parameter, can be used to avoid performance degradation in the shared pool from situations where pool fragmentation forces Oracle to search for and free chunks of unused pool to satisfy the current request.

The shared pool contains the library cache of shared SQL requests, the dictionary cache, stored procedures, and other cache structures that are specific to a particular instance configuration. For example, in an MTS configuration, the session and private SQL area for each client process is included in the shared pool. When the instance is configured for parallel query, the shared pool includes the parallel query message buffers.

Proper sizing of the shared pool can reduce resource consumption in at least three ways:

- Parse time is avoided if the SQL statement is already in the shared pool. This saves CPU resources.
- Application memory overhead is reduced, since all applications use the same pool of shared SQL statements and dictionary resources.
- I/O resources are saved, since dictionary elements which are in the shared pool do not require disk access.

Default value for SHARED\_POOL\_RESERVED\_SIZE is 5% of the SHARED\_POOL\_SIZE. This means that, by default, the reserved list will always be configured.

If SHARED\_POOL\_RESERVED\_SIZE > 1/2 SHARED\_POOL\_SIZE, Oracle signals an error.

Ideally, this parameter should be large enough to satisfy any request scanning for memory on the reserved list without flushing objects from the shared pool. The amount of operating system memory, however, may constrain the size of the shared pool. In general, you should set shared\_pool\_reserved\_size to 10% of shared\_pool\_size. For most systems, this value will be sufficient if you have already tuned the shared pool.

SHARED\_POOL\_RESERVED\_SIZE can accept a numerical value or a number followed by the suffix “K” or “M,” where “K” means “multiply by 1000” and “M” means “multiply by 1000000.”

<b>Parameter Type</b>	String
<b>Parameter Class</b>	Static
<b>Default Value</b>	5% of the value of SHARED_POOL_SIZE
<b>Range of Values</b>	From SHARED_POOL_RESERVED_MIN_ALLOC to one-half of SHARED_POOL_SIZE (in bytes)

## SHARED\_POOL\_SIZE

SHARED\_POOL\_SIZE specifies the size of the shared pool in bytes. The shared pool contains shared cursors and stored procedures. Larger values improve performance in multi-user systems. Smaller values use less memory.

SHARED\_POOL\_SIZE can accept a numerical value or a number followed by the suffix “K” or “M” where “K” means “multiply by 1000” and “M” means “multiply by 1000000.”

<b>Parameter Type</b>	String
<b>Parameter Class</b>	Static
<b>Default Value</b>	3,500,000 bytes
<b>Range of Values</b>	300 Kbytes–operating system–dependent

## SORT\_AREA\_RETAINED\_SIZE

SORT\_AREA\_RETAINED\_SIZE specifies the maximum amount, in bytes, of User Global Area (UGA) memory retained after a sort run completes. The retained size controls the size of the read buffer which is used to maintain a portion of the sort in memory. This memory is released back to the UGA, not to the operating system, after the last row is fetched from the sort space.

If a sort requires more memory, a temporary segment is allocated and the sort becomes an external (disk) sort. The maximum amount of memory to use for the sort is then specified by `SORT_AREA_SIZE` instead of by this parameter.

Larger values permit more sorts to be performed in memory. However, multiple sort spaces of this size may be allocated. Usually, only one or two sorts occur at one time, even for complex queries. In some cases, though, additional concurrent sorts are required. Each sort occurs in its own memory area, as specified by `SORT_AREA_RETAINED_SIZE`. If the value is set too high, it will be converted to a usable value.

<b>Parameter Type</b>	Integer
<b>Parameter Class</b>	Dynamic, scope= ALTER SESSION, ALTER SYSTEM DEFERRED
<b>Default Value</b>	The value of <code>SORT_AREA_SIZE</code>
<b>Range of Values</b>	From the value equivalent of one database block to the value of <code>SORT_AREA_SIZE</code>

## **`SORT_AREA_SIZE`**

`SORT_AREA_SIZE` specifies the maximum amount, in bytes, of Program Global Area (PGA) memory to use for a sort. If MTS is enabled, the sort area is allocated from the SGA. After the sort is complete and all that remains to do is to fetch the rows, the memory is released down to the size specified by `SORT_AREA_RETAINED_SIZE`. After the last row is fetched, all memory is freed. The memory is released back to the PGA, not to the operating system.

Increasing `SORT_AREA_SIZE` size improves the efficiency of large sorts. Multiple allocations never exist; there is only one memory area of `SORT_AREA_SIZE` for each user process at any time.

If more space is required to complete the sort than will fit into the memory provided, then temporary segments on disk hold the intermediate sort runs.

The default is usually adequate for most OLTP operations. You might want to adjust this parameter for decision support systems, batch jobs, or large `CREATE INDEX` operations.

<b>Parameter Type</b>	Integer
<b>Parameter Class</b>	Dynamic, scope= ALTER SESSION, ALTER SYSTEM DEFERRED
<b>Default Value</b>	Operating system-dependent
<b>Range of Values</b>	0–system-dependent value



**SORT\_DIRECT\_WRITES (Obsolete in 8.1.3)**

SORT\_DIRECT\_WRITES can improve sort performance if memory and temporary space are abundant on your system. This parameter controls whether sort data will bypass the buffer cache to write intermediate sort results to disk. When set to the default of AUTO, and the value of the sort area size is greater than ten times the block size, memory is allocated from the sort area to do this. When SORT\_DIRECT\_WRITES is TRUE, additional buffers are allocated from memory during each sort.

Additional temporary segment space can be required when SORT\_DIRECT\_WRITES is enabled. The sort allocation mechanism allocates temporary space using fixed-size chunks which are based on the SORT\_WRITE\_BUFFER\_SIZE parameter. Since the values for this parameter are typically an order of magnitude larger than the DB\_BLOCK\_SIZE chunks used when SORT\_DIRECT\_WRITES is disabled, unused temporary space in the final sort segment increases the overall space requirements. When SORT\_DIRECT\_WRITES is set to FALSE, the sorts that write to disk write through the buffer cache.

<b>Parameter Type</b>	String
<b>Parameter Class</b>	Dynamic, scope= ALTER SESSION, ALTER SYSTEM DEFERRED
<b>Default Value</b>	AUTO
<b>Range of Values</b>	AUTO/TRUE/FALSE

**SORT\_MULTIBLOCK\_READ\_COUNT (New in 8.1)**

SORT\_MULTIBLOCK\_READ\_COUNT specifies the number of database blocks to read each time a sort performs a read from a temporary segment. Temporary segments are used by a sort when the data does not fit in SORT\_AREA\_SIZE of memory. In these situations, sort writes out sections of data to temporary segments in the form of sorted runs. Once all the data has been partially sorted to these runs, sort merges the runs by reading pieces of them from the temporary segment into memory to produce the final sorted output. If SORT\_AREA\_SIZE is not large enough to merge all the runs at once, subsets of the runs are merged in a number of merge passes.

Increasing the SORT\_MULTIBLOCK\_READ\_COUNT parameter forces sort to read a larger section of each run into memory during a merge pass. This reduces the merge width, or number of runs that can be merged in one merge pass, and may increase the number of merge passes. Each merge pass produces an intermediate run on disk, a run that contains all the data that was part of the runs that were just merged. Any increase

in I/O throughput obtained by increasing SORT\_MULTIBLOCK\_READ\_COUNT needs to be balanced with a possible increase in total amount of I/O performed due to an increase in the number of merge passes.

Sort may read more blocks at a time than what is specified by SORT\_MULTIBLOCK\_READ\_COUNT in cases where the number of runs, and therefore the merge width is small relative to SORT\_AREA\_SIZE.

<b>Parameter Type</b>	Integer
<b>Parameter Class</b>	Dynamic, scope= ALTER SESSION, ALTER SYSTEM DEFERRED
<b>Default Value</b>	2
<b>Range of Values</b>	1–system-dependent value

### **SORT\_WRITE\_BUFFER\_SIZE (Obsolete in 8.1.3)**

SORT\_WRITE\_BUFFER\_SIZE sets the size of the sort IO buffer when the SORT\_DIRECT\_WRITES parameter is set to TRUE.

SORT\_WRITE\_BUFFER\_SIZE is recommended for use with symmetric replication.

<b>Parameter Type</b>	Integer
<b>Parameter Class</b>	Dynamic, scope= ALTER SESSION, ALTER SYSTEM DEFERRED
<b>Default Value</b>	32768
<b>Range of Values</b>	32 Kb, 64 Kb

### **SORT\_WRITE\_BUFFERS (Obsolete in 8.1.3)**

SORT\_WRITE\_BUFFERS specifies the number of sort buffers when the SORT\_DIRECT\_WRITES parameter is set to TRUE. SORT\_WRITE\_BUFFERS is recommended for use with symmetric replication.

<b>Parameter Type</b>	Integer
<b>Parameter Class</b>	Dynamic, scope= ALTER SESSION, ALTER SYSTEM DEFERRED
<b>Default Value</b>	1
<b>Range of Values</b>	2–8

## SQL\_TRACE

The value of SQL\_TRACE disables or enables the SQL trace facility. Setting this parameter to TRUE provides information on tuning that you can use to improve performance. Because the SQL trace facility causes system overhead, you should run the database with the value TRUE only for the purpose of collecting statistics. The value can also be changed using the DBMS\_SYSTEM package.

<b>Parameter Type</b>	Boolean
<b>Parameter Class</b>	Dynamic, scope = ALTER SESSION
<b>Default Value</b>	FALSE
<b>Range of Values</b>	TRUE/FALSE

## STAR\_TRANSFORMATION\_ENABLED

The value of STAR\_TRANSFORMATION\_ENABLED determines whether a cost-based query transformation will be applied to star queries. If set to TRUE, the optimizer will consider performing a cost-based query transformation on the star query. If set to FALSE, the transformation will not be applied.

<b>Parameter Type</b>	Boolean
<b>Parameter Class</b>	Dynamic, scope = ALTER SESSION
<b>Default Value</b>	FALSE
<b>Range of Values</b>	TRUE/FALSE

## TAPE\_ASYNC\_IO

TAPE\_ASYNC\_IO can be used to control whether I/O to sequential devices (for example, BACKUP/RESTORE of Oracle data TO/FROM tape) is asynchronous. If a platform supports asynchronous I/O to sequential devices, it is recommended that this parameter is left to its default. However, if the asynchronous I/O implementation is not stable, TAPE\_ASYNC\_IO can be used to disable its use. If a platform does not support asynchronous I/O to sequential devices, this parameter has no effect.

<b>Parameter Type</b>	Boolean
<b>Parameter Class</b>	Static
<b>Default Value</b>	TRUE
<b>Range of Values</b>	TRUE, FALSE

## TIMED\_STATISTICS

If TIMED\_STATISTICS is FALSE, the statistics related to time are always zero and the server can avoid the overhead of requesting the time from the operating system. To turn on statistics, set the value to TRUE. Normally, TIMED\_STATISTICS should be

FALSE. On some systems with very fast timer access, timing might be enabled even when the parameter is set to FALSE. On these systems, setting the parameter to TRUE might produce more accurate statistics for long-running operations.

<b>Parameter Type</b>	Boolean
<b>Parameter Class</b>	Dynamic, scope = ALTER SYSTEM, ALTER SESSION
<b>Default Value</b>	FALSE
<b>Range of Values</b>	TRUE/FALSE

## **USER\_DUMP\_DEST**

USER\_DUMP\_DEST specifies the pathname for a directory where the server will write debugging trace files on behalf of a user process.

For example, this directory might be set to C:\ORACLE\UTRC on MS-DOS; to /oracle/utrc on UNIX; or to DISK\$UR3:[ORACLE.UTRC] on VMS.

<b>Parameter Type</b>	String
<b>Parameter Class</b>	Dynamic, scope = ALTER SYSTEM
<b>Default Value</b>	Operating system-dependent
<b>Range of Values</b>	Valid local pathname, directory, or disk