

---

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

**Volume 1 • Instructor Guide**

---

30052GC10

Production 1.0

September 1999

M09215

**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

---

# 1

---

## **Course Introduction**

## Objectives

### Course Objectives

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

- List the important steps for outlining a tuning methodology
- Use Oracle tools to diagnose performance problems
- Configure memory resources to optimize cache operations
- Reconfigure file structures to enhance performance

Copyright © Oracle Corporation, 1999. All rights reserved.

ORACLE®



## **Course Objectives**

- **Identify and resolve I/O, storage, and database configuration problems**
- **Detect and resolve latch and lock contention problems**
- **Configure memory and disk resources to optimize sort operations**
- **Diagnose and resolve performance issues associated with the multithreaded server**
- **List options to enhance performance across differing application environments**

Copyright © Oracle Corporation, 1999. All rights reserved.

**ORACLE**



## **Tuning Overview**

## Objectives

### Objectives

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

- **List the roles associated with the database tuning process**
- **Define the steps associated with the tuning process**
- **Identify tuning goals**

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

## System Tuning Overview

### Tuning Questions

- **Who tunes?**
  - **Application designers**
  - **Application developers**
  - **Database administrators**
  - **System administrators**
- **Why tune?**
- **How much tuning?**

Copyright © Oracle Corporation, 1999. All rights reserved.

ORACLE®

### Who Tunes?

Everyone involved with the Oracle8i system—system architects, designers, developers, and database administrators (DBAs)—should think about performance and tuning in doing their work.

If problems develop, the DBA usually makes the first attempt at solving them.

### Why Tune?

The best practice of database tuning is to carefully design the system and application.

The majority of performance gains are realized by tuning the application.

Your system is much less likely to run into performance problems if:

- The hardware can meet user demands.
- Your Oracle8i database was carefully designed.
- Your application developers write efficient SQL programs.

If wrong decisions were made early in the system development process, or if users now expect much more from the system than they did previously, you may need to seriously consider improving performance. The longer you delay in addressing the tuning process, the more it costs in time and resources.

## How Much Tuning?

You should begin tuning with a clear idea of what you are trying to achieve. Try to quantify your goals as precisely as possible, in real-world terms. For example:

- Process 10,000 orders each day
- Produce 250,000 billing statements overnight at the end of the month

This course frequently describes Oracle8i tuning objectives and methods, but ultimately these tuning steps must benefit the users. There is no point in making a small improvement in the use of the data buffer cache if all users are accessing the data over a slow network on aging PCs: your tuning efforts will not be noticed.

Tuning is an iterative process—it is not an activity that is performed only once.

## Tuning Goals

### Examples of Measurable Tuning Goals

- **Response time**
- **Database availability**
- **Database hit percentages**
- **Memory utilization**

Copyright © Oracle Corporation, 1999. All rights reserved.

ORACLE®

### Measurable Tuning Goals

When tuning an Oracle8i database environment, the DBA should establish measurable tuning goals. Without them, it is difficult to determine when you have performed enough tuning.

- *Response time* addresses how long it takes for a user to receive data from a request—for example, the result set of a query—or the time it takes to update a table, or to generate a report.
- *Database availability* is also a good measure for tuning goals. Availability can be backup and recovery procedures, or by shutting down and starting the instance to tune parameters.
- *Database hit percentages* provide a good baseline from which to determine if performance is increasing or decreasing over time.
- *Memory utilization* is also a valid measure for tuning, because excessive paging and swapping can impact database and operating system performance. Memory utilization can also impact database hit percentages.

## **Tuning Goals**

- **Access the least number of blocks**
- **Cache blocks in memory**
- **Share application code**
- **Read and write data as quickly as possible**
- **Ensure that users do not wait for resources**
- **Perform backups and housekeeping while minimizing impact**

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

## **Establishing Tuning Goals**

Your primary goals in tuning an Oracle8i server are to make sure that:

- SQL statements access the smallest possible number of Oracle *blocks*.
- If a block is needed, it is cached in memory.
- Users share the same code.
- When code is needed, it is cached in memory.
- Where reads and writes are unavoidable, they are done as quickly as possible.
- Users never have to wait for resources held by others.
- Backups and other necessary housekeeping can be done as quickly as possible.



---

## Tuning Steps

### Tuning Steps

1. Tune the design.
2. Tune the application.
3. Tune memory.
4. Tune I/O.
5. Tune contention.
6. Tune the operating system.

Copyright © Oracle Corporation, 1999. All rights reserved.

ORACLE®

### Tuning Steps

The recommended order in which to implement tuning is as follows:

- 1 The design (if it is possible)
- 2 The application
- 3 The memory
- 4 Input/output (I/O)
- 5 Contention
- 6 Operating system

Repeat the tuning process if your goals have not yet been achieved.

The rationale for this structure is that improvements early in the sequence can save you from encountering problems later. For example, if your applications are using a lot of full table scans, this may show up as excessive I/O. However, there is no point in resizing the buffer cache or redistributing disk files if you can rewrite the queries so that they access only 4 blocks instead of 4,000.

The first two steps are typically the responsibility of the system architects and application developers; however, the DBA may also be involved in application tuning.

## Summary

### Summary

**In this lesson, you should have learned that it is important to:**

- **Create a good initial design**
- **Define roles clearly**
- **Perform application tuning**
- **Establish quantifiable goals**

Copyright © Oracle Corporation, 1999. All rights reserved.

**ORACLE**

## **Oracle Alert and Trace Files**

## Objectives

### Objectives

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

- **Describe the location and usefulness of the Alert log file**
- **Describe the location and usefulness of the background and user process trace files**

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

## Diagnostic Information

### Diagnostic Information

**Trace files:**

- **Alert log file**
- **Background process trace files**
- **User trace files**

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

### Alert Log File

If an error occurs while your Oracle instance is running, the error messages are written to the Alert log file. During startup of the database, if the Alert log file does not exist, the Oracle server creates one.

The database Alert log file is a chronological log of messages and errors. Oracle server uses the Alert log file as an alternative to displaying such information.

### Background Process Trace Files

If an error is detected by a background process, the information is dumped into a trace file.

### User Trace Files

Trace files can also be generated by server processes, at the user's request, to display resource consumption during statement processing.

## The Alert Log File

### Alert Log File

- The Alert log file consists of a chronological log of messages and errors.
- Check the Alert log file regularly to:
  - Detect internal errors (ORA-600) and block corruption errors
  - Monitor database operations
  - View the nondefault initialization parameters
- Remove or trim the Alert log file regularly after checking.

Copyright © Oracle Corporation, 1999. All rights reserved.

ORACLE®

### The Alert Log File

It is important for the database administrator to check the Alert log file regularly to detect problems before they become serious.

The following information is logged in the Alert log file:

- Internal errors (ORA-600), and block corruption errors (ORA-1578)
- Operations that affect database structures and parameters, and statements such as CREATE DATABASE, STARTUP, SHUTDOWN, ARCHIVE LOG, and RECOVER
- The values of all nondefault initialization parameters at the time the instance starts

### Example of Information Found in the Alert Log File

```
Starting up ORACLE RDBMS Version: 8.1.5.0.0.
```

```
System parameters with non-default values:
```

```
processes                = 50
timed_statistics          = TRUE
shared_pool_size          = 3500000
control_files              = /home/disk3/user30/DATA/DISK1/
                           control01.con, /home/disk3/
                           user30/DATA/DISK2/control02.con
```

**Example of Information Found in the Alert Log File (continued)**

```
db_block_buffers= 200
compatible = 8.1.5
log_buffer= 8192
log_checkpoint_interval= 10000
db_files= 20
db_file_multiblock_read_count= 8
dml_locks= 100
db_name= U30
utl_file_dir = /home/disk3/user30/UDUMP
background_dump_dest= /home/disk3/user30/BDUMP
user_dump_dest= /home/disk3/user30/UDUMP
max_dump_file_size= 10240
core_dump_dest= /home/disk3/user30/CDUMP
PMON started with pid=2
DBW0 started with pid=3
LGWR started with pid=4
CKPT started with pid=5
SMON started with pid=6
RECO started with pid=7
Fri Jun 4 02:51:55 1999
create database ptw01
controlfile reuse
datafile '/home/disk3/user30/DATA/DISK1/sys01.dbf' size 20m
logfile  '/home/disk3/user30/DATA/DISK3/log1.rdo' size 300k,
        '/home/disk3/user30/DATA/DISK3/log2.rdo' size 300k
Fri Jun 4 02:51:55 1999
ORA-1501 signalled during: create database ptw01
controlfile reuse
datafile '...'
Fri Jun 4 02:51:55 1999
alter database open
ORA-1507 signalled during: alter database open...
Fri Jun 4 02:54:20 1999
Shutting down instance (normal)
License high water mark = 1
Fri Jun 4 02:54:20 1999
```

### Example of Information Found in the Alert Log File (continued)

```
ALTER DATABASE CLOSE NORMAL
```

```
ORA-1507 signalled during: ALTER DATABASE CLOSE NORMAL...
```

- Control file and online tablespace backups

```
Fri Jun 4 10:54:20 1999
```

```
alter tablespace user_data begin backup
```

```
Fri Jun 4 10:54:21 1999
```

```
ORA-1123 signalled during: alter tablespace user_data begin backup
```

```
...
```

- Noncompletion of checkpoints due to spinning the redo log files so rapidly

```
Thread 1 advanced to log sequence 1597
```

```
Current log# 2 seq# 1597 mem# 0: /users/cours/tun8_08/DATA/DISK3/  
log2a.rdo
```

```
Thread 1 cannot allocate new log, sequence 1598
```

```
Checkpoint not complete
```

- Creation of tablespaces

```
Fri Jun 4 10:57:20 1999
```

```
create tablespace SYSTEM datafile '/home/disk3/user30/DATA/DISK1/  
sys01.dbf' size 20m
```

```
default storage (initial 10K next 10K) online
```

```
Fri Jun 4 10:57:30 1999
```

```
Completed: create tablespace SYSTEM datafile '/home/disk3/user30/  
DATA/DISK1/sys01.dbf'
```

```
create tablespace rbs
```

```
datafile '/home/disk3/user30/DATA/DISK2/rbs01.dbf' size 30m
```

```
Fri Jun 4 10:58:48 1999
```

```
Completed: create tablespace rbs datafile '/home/disk3/user30/  
DATA/DISK2/rbs01.dbf'
```

- Creation and modification of rollback segments

```
Fri Jun 4 11:57:48 1999
```

```
create rollback segment SYSTEM tablespace SYSTEM
```

```
storage (initial 50K next 50K)
```

```
Completed: create rollback segment SYSTEM tablespace SYSTEM
```

```
Fri Jun 4 12:07:58 1999
```

```
alter rollback segment rbs01 online
```

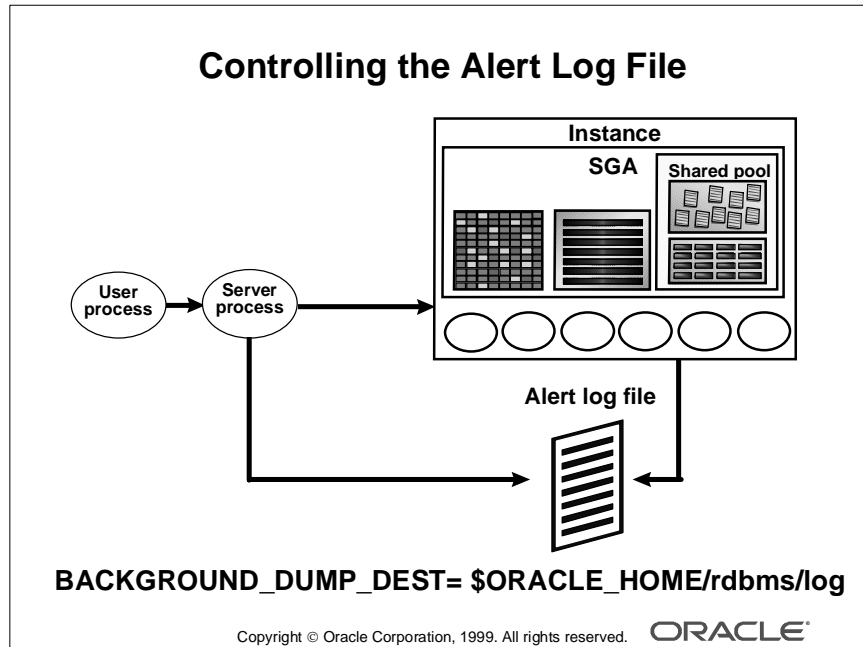
```
Completed: alter rollback segment rbs01 online
```

```
Fri Jun 4 12:58:48 1999
```

Because the Alert log file grows and therefore uses an ever-increasing amount of disk space, archive and remove it often, or trim it periodically.



## Controlling the Alert Log File



## Controlling the Alert Log File

The `init.ora` parameter that controls the location of the Alert log file is `BACKGROUND_DUMP_DEST`.

The default directory for different operating systems is:

- On a UNIX operating system: `$ORACLE_HOME/rdbms/log`, and the name of the file is `alert_SID.log`
- On NT: `%ORACLE_HOME%\Admin\SID\bdump`, and the name of the file is `SIDALRT.LOG`

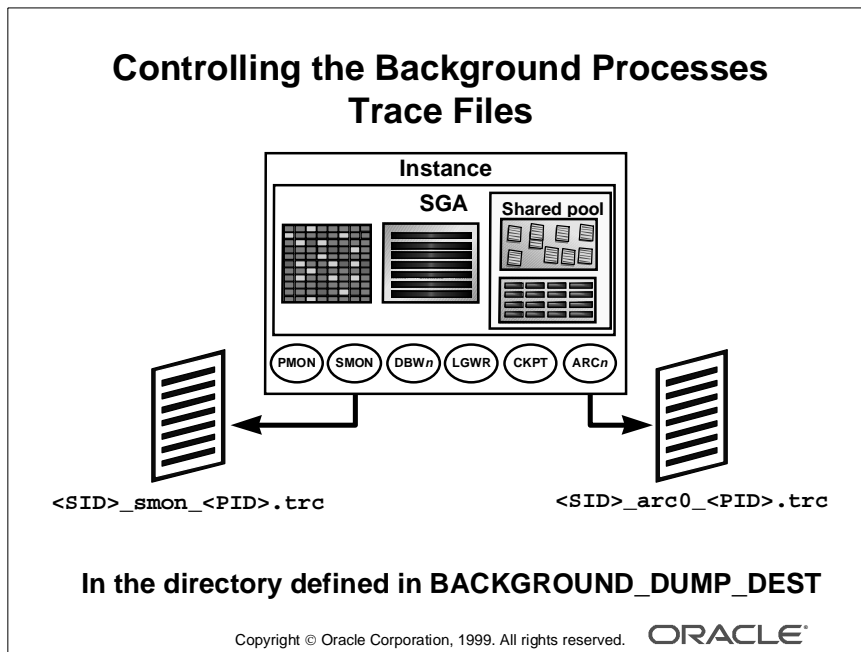
## Controlling the Background Processes Trace Files

### Background Processes Trace Files

- The Oracle server dumps information about errors detected by any background process in trace files.
- Oracle support uses these trace files to diagnose and troubleshoot problems.

Copyright © Oracle Corporation, 1999. All rights reserved.

ORACLE®



## Controlling the Background Processes Trace Files

The `init.ora` parameter that controls the location of the background processes trace file is `BACKGROUND_DUMP_DEST`.

The default directory for different operating systems is:

- On UNIX: `$ORACLE_HOME/rdbms/log`, and the name of the file is `SID_processname_PID.trc`
- On NT: `%ORACLE_HOME%\Admin\SID\bdump`, and the name of the file is `SIDprocessname.TRC`

**Example** Partial output of the UNIX `ls` command:

```
$ ls -l /ora/ora815/rdbms/log
-rw-r----- 1 ora815 dba 682 Jun 5 8:39 ora8_ckpt_27742.trc
-rw-r----- 1 ora815 dba 682 Jun 5 8:39 ora8_lgwr_27736.trc
```

## Controlling the Background Processes Trace Files (continued)

**Example** Partial output of an NT background trace file `orc8LGWR.TRC`:

```
Dump file D:\ORANT8i\admin\ORC8\bdump\orc8LGWR.TRC
Mon Mar 22 18:15:05 1999
ORACLE V8.1.5.0.0 - Production vsnsta=0
Windows NT V4.0, OS V5.101, CPU type 586
Oracle8i Enterprise Edition Release 8.1.5.0.0 - Production
Windows NT V4.0, OS V5.101, CPU type 586
Instance name: orc8
Redo thread mounted by this instance: 1
Oracle process number: 4
Windows thread id: 148, image: ORACLE.EXE
*** 1999.03.22.18.15.05.533
*** SESSION ID:(3.1) 1999.03.22.18.15.05.483
ORA-00313: open failed for members of log group 2 of thread 1
```

## User Trace Files

### User Trace Files

- Server process tracing is enabled or disabled at the session or instance level by:
  - The ALTER SESSION command
  - The SET\_SQL\_TRACE\_IN\_SESSION procedure
  - The initialization parameter SQL\_TRACE
- A user trace file contains statistics for traced SQL statements for that session.
- A user trace file is useful for SQL tuning.
- The Oracle database creates user trace files on a per-server-process basis.

Copyright © Oracle Corporation, 1999. All rights reserved.

ORACLE®

### User Trace Files

A user or DBA can also request that user trace files be generated by server processes.

### Instance-Level Tracing

You can use the initialization parameter SQL\_TRACE to enable or disable tracing. The default value of the parameter setting is FALSE (disabled).

### Session-Level Tracing

The following statement enables tracing for a particular session:

```
SQL> EXECUTE dbms_system.set_sql_trace_in_session(8,12,TRUE);
```

where 8 and 12 are the SID and SERIAL# of the connected user.

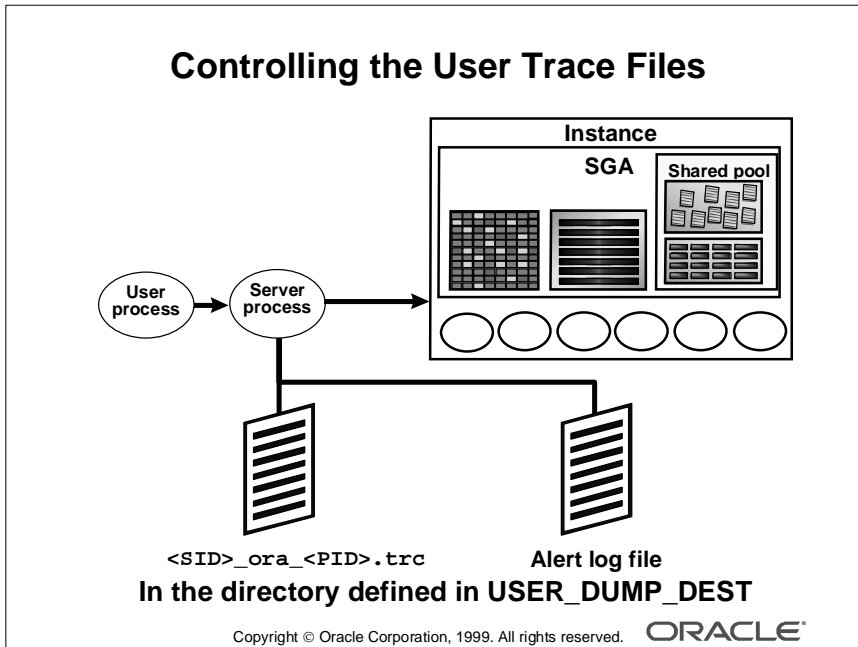
The DBMS\_SYSTEM package is automatically created when catproc.sql is run. If needed, you can re-create it by running the prvtutil.plb script stored in the directory:

- \$ORACLE\_HOME/rdbms/admin script on a UNIX operating system
- %ORACLE\_HOME%\Rdbms\Admin on an NT operating system

The following statement enables tracing for the session of the connected user:

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

## Controlling the User Trace Files



### Controlling the User Trace Files

The following initialization parameters control the location and size of the user trace files:

- `USER_DUMP_DEST`: Defines where trace files will be created at the request of the users or the DBA
- `MAX_DUMP_FILE_SIZE`: Specified in O/S blocks; limits the size of user trace files

The default directory for `USER_DUMP_DEST` is:

- On a UNIX operating system: `$ORACLE_HOME/rdbms/log`
- On an NT operating system: `%ORACLE_HOME%\Admin\SID\udump`

The name of this file will be:

- On a UNIX operating system: `SID_ora_PID.trc`
- On an NT operating system: `OraPID.trc`

## Controlling the User Trace Files (continued)

**Example** Enabling tracing in a session:

```
SQL> alter session set sql_trace=true;
Session altered.
SQL> select id,last_name from scott.s_emp
  2* where id=40;
      ID  LAST_NAME
-----
      40  Dumas
SQL>
```

**Example** Partial output of the resulting user trace file:

```
$ cd $ORACLE_HOME/rdbms/log
$ ls -l
-rw-r--r--  1 oracle  dba 12307 Jun 4 11:32 u30_ora_19280.trc
$ more u30_ora_19280.trc
=====
PARSING IN CURSOR#1 len=49 dep=0 uid=25 oct=3 lid=25 tim=31496536
hv=4182141062 ad='801a0394'
select id,last_name from scott.s_emp
where id=40
END OF STMT
PARSE
#1:c=11,e=31,p=5,cr=42,cu=1,mis=1,r=0,dep=0,og=4,tim=31496536
EXEC #1:c=0,e=0,p=0,cr=0,cu=0,mis=0,r=0,dep=0,og=4,tim=31496536
FETCH #1:c=0,e=4,p=4,cr=4,cu=0,mis=0,r=1,dep=0,og=4,tim=31496542
FETCH #1:c=0,e=0,p=0,cr=0,cu=0,mis=0,r=0,dep=0,og=0,tim=31496567
```

**Example** Partial output of the UNIX `ls` command:

```
$ ls -l /ora/ora815/rdbms/log
-rw-r-----  1 ora815 dba 2616 Jun 4 11:57  ora8_ora_1454.trc
-rw-r-----  1 ora815 dba 373310 Jun 4 10:12 ora8_ora_15819.trc
-rw-r-----  1 ora815 dba 356178 Jun 4 15:35 ora8_ora_1918.trc
```

**Note:** The `MAX_DUMP_FILE_SIZE` and `USER_DUMP_DEST` parameters are dynamic initialization parameters.

## Summary

### Summary

In this lesson, you should have learned how to:

- Set, retrieve, and use the Alert log file
- Use background processes trace files
- Trace user SQL statements

Copyright © Oracle Corporation, 1999. All rights reserved.

ORACLE®



---

## Quick Reference

Context	Reference
Initialization parameters	BACKGROUND_DUMP_DEST USER_DUMP_DEST MAX_DUMP_FILE_SIZE SQL_TRACE
Dynamic performance views	V\$PARAMETER
Data dictionary views	None
Commands	ALTER SESSION SET SQL_TRACE
Packaged procedures and functions	DBMS_SYSTEM.SET_SQL_TRACE_IN_SESSION
Scripts	prvtutil.plb



---

# 4

---

## **Utilities and Dynamic Performance Views**

## Objectives

### Objectives

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

- **Collect statistics through:**
  - Available dynamic troubleshooting and performance views
  - The UTLBSTAT/UTLESTAT report output
  - Oracle wait events
  - Appropriate Enterprise Manager (EM) tuning tools
- **Define the latch types**
- **Use EM to set events for predefined situations**

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

## Views, Utilities, and Tools

### Views, Utilities, and Tools

- **Dynamic troubleshooting, performance and dictionary views**
  - **V\$xxx dynamic troubleshooting and performance views**
  - **DBA\_ xxx dictionary views**
- **UTLBSTAT.SQL and UTLESTAT.SQL scripts**
- **Oracle Wait events**
- **Enterprise Manager event service**
- **Oracle Diagnostics and Tuning packs**

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

### Dictionary and Dynamic Views

The Oracle server displays all system statistics in the V\$SYSSTAT view, and uses many other views for performance and troubleshooting information. You can query these views to find cumulative totals since the instance started, but this may not be helpful; if your instance is rarely shut down, then the statistics may cover a long period and have little meaning.

The Oracle server displays data storage statistics in DBA\_xxx views that help you troubleshoot the segments storage (tables, clusters, and indexes).

### UTLBSTAT and UTLESTAT Utilities

You should gather performance figures over a defined period, probably your busiest time of day or at the end of the month, and produce a hard-copy report.

You can do this by using the UTLBSTAT.SQL and UTLESTAT.SQL scripts.

Experienced consultants usually begin a tuning project by using this utility to capture data.

### **Oracle Wait Events**

If you are troubleshooting a database system, you need to know when a process has waited for any resource. The Oracle server has a list of wait events.

Some dictionary views display the events for which sessions had to wait.

### **Oracle Enterprise Manager Events**

The Oracle Enterprise Manager event service enables you to detect systematic problems by registering event tests.

### **Oracle Diagnostics and Tuning Packs Applications**

You can also use the Oracle GUI tools provided with the Oracle Diagnostics and Tuning packs, a set of Windows-based applications that address many Oracle performance management areas, such as graphical monitoring, analysis, and automated tuning of Oracle databases.

## Dictionary and Special Views

### Dictionary and Special Views

Dictionary and special views provide useful statistics after you run the ANALYZE command:

- DBA\_TABLES, DBA\_TAB\_COLUMNS
- DBA\_CLUSTERS
- DBA\_INDEXES, INDEX\_STATS
- INDEX\_HISTOGRAM, DBA\_HISTOGRAMS

This statistics information is static until you reexecute the ANALYZE command.

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

### Dictionary and Special Views

When you want to look at data storage in detail, you need to use the ANALYZE command, which collects statistics and populates DBA\_xxx and special view columns.

ANALYZE populates columns in the views concerned with:

- Table data storage within extents and blocks:
  - DBA\_TABLES
  - DBA\_TAB\_COLUMNS
- Cluster data storage within extents and blocks:
  - DBA\_CLUSTERS
- Index data storage within extents and blocks, and indexation usefulness:
  - DBA\_INDEXES
  - INDEX\_STATS
- Nonindexed and indexed columns data distribution:
  - DBA\_HISTOGRAMS
  - INDEX\_HISTOGRAM

This command is described in more detail in Lesson 9, “Using Oracle Blocks Efficiently.”

## Dynamic Troubleshooting and Performance Views

### Dynamic Troubleshooting and Performance Views

- **V\$ views**
  - Based on X\$ tables
  - Listed in V\$FIXED\_TABLE
- **X\$ tables**
  - Not usually queried directly
  - Dynamic and constantly changing
  - Names abbreviated and obscure

Populated at startup and cleared at shutdown

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

### V\$Views

- These views are based on X\$ tables—memory structures that hold instance information and are therefore available when the instance is in a NOMOUNT or MOUNT state.
- V\$ views are listed in V\$FIXED\_TABLE.
- The V\$ views (actually synonyms for V\_\$ views) belong to the `sys` user.

### X\$ Tables

- These tables are not usually queried directly; not all the information is necessarily useful, and column names tend to be abbreviated and obscure.
- The X\$ tables are dynamic and their contents are constantly changing.
- The V\$ views, and the underlying X\$ tables, are populated at instance startup and cleared at shutdown.
- X\$ tables hold timing information if you set the `init.ora` parameter `TIMED_STATISTICS` to `TRUE` or if you execute the SQL command

```
SQL> ALTER SYSTEM SET timed_statistics = TRUE;
```



## Topics for Troubleshooting and Tuning

Topics for Troubleshooting and Tuning	
System-Wide Statistics	Session-Related Statistics
<b>Instance/Database</b> V\$DATABASE T V\$INSTANCE T V\$OPTION V\$PARAMETER T/P V\$BACKUP T V\$PX_PROCESS_SYSSTAT T/P V\$PROCESS T V\$WAITSTAT T/P V\$SYSTEM_EVENT T/P	<b>User/Session</b> V\$LOCK P V\$OPEN_CURSOR T V\$PROCESS T V\$SORT_USAGE T/P V\$SESSION T/P V\$SESSTAT T/P V\$TRANSACTION T V\$SESSION_EVENT T/P V\$SESSION_WAIT T/P V\$PX_SESSTAT P V\$PX_SESSION P V\$SESSION_OBJECT_CACHE P
<b>Memory</b> V\$BUFFER_POOL_STATISTICS T/P V\$DB_OBJECT_CACHE T V\$LIBRARYCACHE P V\$ROWCACHE P V\$SYSSTAT T/P V\$SGASTAT P	<b>Contention</b> V\$LOCK T/P V\$ROLLNAME T/P V\$ROLLSTAT T/P V\$WAITSTAT T/P V\$LATCH T/P
T for Troubleshooting T/P for Troubleshooting/Performance Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®	

### System-Wide Statistics

#### Views Pertaining to the Instance/Database

- V\$PX\_PROCESS\_SYSSTAT: Parallel query system statistics
- V\$PROCESS: Information about currently active processes
- V\$WAITSTAT: Contention statistics
- V\$SYSTEM\_EVENT: Total waits for particular events

#### Views Pertaining to Memory

- V\$BUFFER\_POOL\_STATISTICS: Buffer pools allocation on the instance.  
(created by \$ORACLE\_HOME/rdbms/admin/catperf.sql script)
- V\$DB\_OBJECT\_CACHE: Database objects cached in the library cache
- V\$LIBRARYCACHE: Library cache performance and activity statistics
- V\$ROWCACHE: Data dictionary hits and misses activity
- V\$SYSSTAT: Basic instance statistics

## **System-Wide Statistics (continued)**

### **Views Pertaining to Disk Performance**

- V\$FILESTAT: Data file read/write statistics
- V\$TEMPSTAT: Information about file read/write statistics on temporary tablespace data files

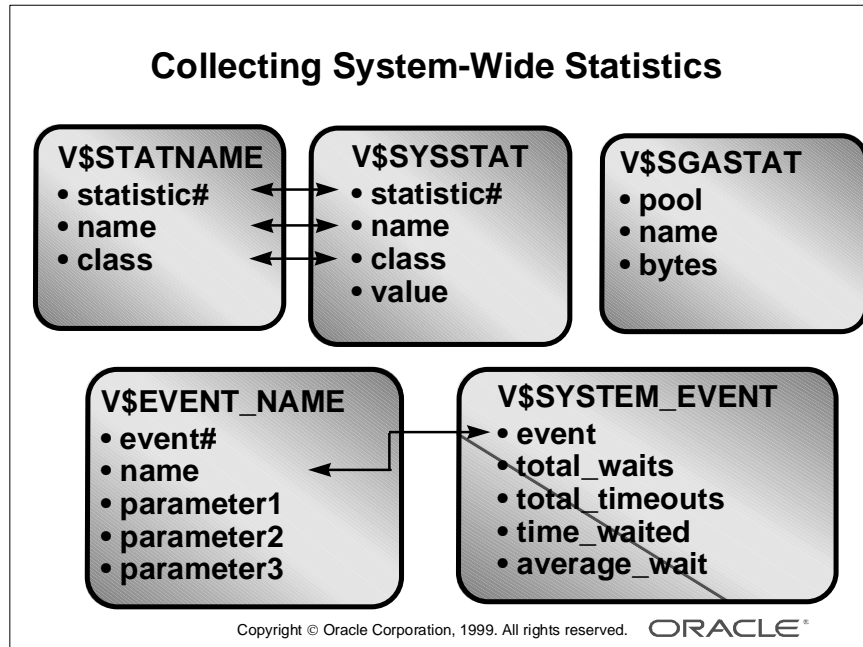
### **Views Pertaining to Contention**

- V\$LATCH: Statistics for each type of latch
- V\$ROLLSTAT: Statistics for all online rollback segments
- V\$WAITSTAT: Block contention statistics (The `init.ora` parameter `TIMED_STATISTICS` should be set to `TRUE`.)

### **Session-Related Statistics**

- V\$LOCK: Locks currently held by the server and outstanding requests for a lock or latch
- V\$OPEN\_CURSOR: Cursors currently opened and parsed by each session
- V\$SORT\_USAGE: Size of temporary segments and sessions creating them; identification of processes doing disk sorts
- V\$SESSTAT: User session statistics
- V\$SESSION\_EVENT: Information on waits for an event by a session
- V\$SESSION\_WAIT: Resources or events for which active sessions are waiting
- V\$PX\_SESSTAT: Information about the sessions running parallel execution

## Collecting System-Wide Statistics



### General System-Wide Statistics

Many system-wide statistics are catalogued in the V\$STATNAME view: about 180 statistics are available.

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

### Example

```
SQL> SELECT name,class,value FROM v$sysstat;
```

NAME	CLASS	VALUE
db block gets	8	7687
consistent gets	8	17547
physical reads	8	763
session uga memory	1	1312236
session pga memory	1	57634176
sorts (memory)	64	497
sorts (disk)	64	0
sorts (rows)	64	605

## General System-Wide Statistics (continued)

System statistics are classified by topic:

- Class 1 refers to general instance activity.
- Class 2 refers to redo log buffer activity.
- Class 4 refers to locking.
- Class 8 refers to database buffer cache activity.
- Class 16 refers to operating system activity.
- Class 32 refers to parallelization.
- Class 64 refers to tables access.
- Class 128 refers to debugging purposes.

## SGA Global Statistics

The Oracle server displays all calculated memory statistics in the V\$SGASTAT view. You can query this view to find cumulative totals of detailed System Global Area (SGA) usage since the instance started.

### Example

```
SQL> SELECT * FROM v$sgastat;
POOL          NAME                                BYTES
-----
fixed_sga                                46136
db_block_buffers                        409600
log_buffer                              524288
shared pool free memory                 8341616
shared pool table columns                15248
shared pool SYSTEM PARAMETERS           42496
shared pool transactions                 64800
shared pool dictionary cache            156524
shared pool library cache               358660
shared pool sql area                    551488
```

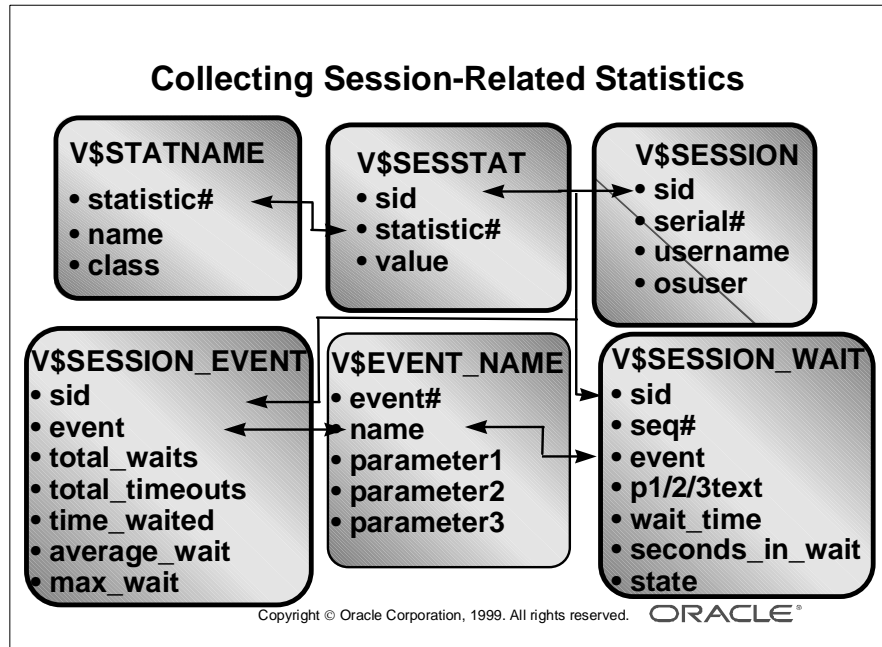
## Waiting Events Statistics

Many waiting events are catalogued in V\$EVENT\_NAME view: about 136 events are available.

Cumulated statistics for all sessions are stored in the V\$SYSTEM\_EVENT view: it shows the total waits for a particular event since instance startup.

If you are troubleshooting, you need to know when a process has waited for any resource.

## Collecting Session-Related Statistics



### General Session-Related Statistics

Session data is cumulative from connect time.

You can display current session information for each user that is logged on.

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

**Example** Determine the type of connection the users have.

```
SQL> SELECT sid, username, type, server FROM v$sqlsession;
```

SID	USERNAME	TYPE	SERVER
1		BACKGROUND	DEDICATED
2		BACKGROUND	DEDICATED
3		BACKGROUND	DEDICATED
4		BACKGROUND	DEDICATED
5		BACKGROUND	DEDICATED
9	SYSTEM	USER	DEDICATED
10	SCOTT	USER	NONE
11	JIM	USER	SHARED

**General Session-Related Statistics (continued)**

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

**Example** Determine the sessions that consume more than 30,000 bytes of Program Global Area (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 > 30000;
```

USERNAME	NAME	VALUE
SYSTEM	session pga memory	468816

**Session Waiting Events Statistics**

The V\$SESSION\_EVENT view shows, by session, the total waits for a particular event since instance startup.

The V\$SESSION\_WAIT view lists the resources or events for which active sessions are waiting.

If you are troubleshooting the Oracle server, you need to know when a process has waited for any resource. The structure of the V\$SESSION\_WAIT view enables you to easily check in real time whether any sessions are waiting, and if so, why.

**Example**

```
SQL> select sid, event
2      from V$SESSION_WAIT
3*  where wait_time = 0;
```

SID	EVENT
1	pmon timer
2	rdbms ipc message
3	rdbms ipc message
9	rdbms ipc message
16	rdbms ipc message
10	rdbms ipc message
4	rdbms ipc message
5	smon timer

9 rows selected.

### **Session Waiting Events Statistics (continued)**

You can then investigate further to see whether such waits occur frequently and whether they can be correlated with other phenomena, such as the use of particular modules.

## UTLBSTAT and UTLESTAT Utilities

### UTLBSTAT and UTLESTAT Scripts

- **Gather performance figures over a defined period**
- **Produce a hard-copy report**
- **Use UTLBSTAT.SQL and UTLESTAT.SQL scripts**
- **Run the scripts from SQL\*Plus connected as SYSDBA**
- **Set TIMED\_STATISTICS to TRUE**

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

### Using the Begin and End Scripts to Gather Statistics

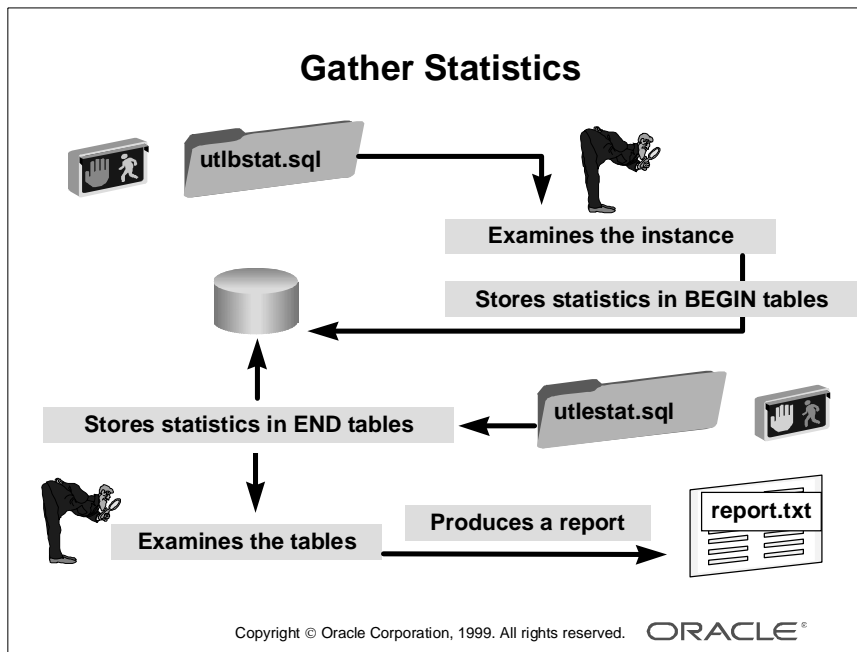
The dynamic views display cumulative totals since the instance started, but this is not helpful; if your instance is rarely shut down, the statistics may cover a long period and have little meaning.

You should gather performance figures over a defined period, probably your busiest time of day or the end of the month, and produce a hard-copy report.

You can do this with the UTLBSTAT.SQL and UTLESTAT.SQL scripts, stored in \$ORACLE\_HOME/rdbms/admin directory on a UNIX operating system and in %ORACLE\_HOME%\Rdbms\Admin on an NT operating environment.

The script connects as SYSDBA and creates tables in the SYS default tablespace, SYSTEM. Before running the script, create a new tablespace for that purpose and change the SYS default tablespace to this new one. When the two scripts have finished running, change the SYS default tablespace back to SYSTEM.





### Using the Begin and End Scripts to Gather Statistics (continued)

To begin gathering statistics, use the following script:

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

- 1 This script first creates the BEGIN and END tables.

```
SQL> create table stats$begin_stats as select * from
2> v$sysstat where l=0;
```

```
SQL> create table stats$end_stats as select * from
2> stats$begin_stats;
```

- 2 Then the script takes a snapshot of data from the dynamic performance tables (V\$xxx) to collect initial statistics and stores these in the BEGIN tables.

```
insert into stats$begin_stats select * from v$sysstat;
```

The following table shows the link between stats tables and performance views.

Stats Table Name	Based on
stats\$begin_latch	v\$latch
stats\$begin_roll	v\$rollstat
stats\$begin_lib	v\$librarycache
stats\$begin_dc	v\$rowcache
stats\$begin_event	v\$system_event
stats\$file_view	v\$filestat, ts\$, v\$datafile, file\$
stats\$begin_file	stats\$file_view
stats\$begin_waitstat	v\$waitstat

## Using the Begin and End Scripts to Gather Statistics (continued)

To terminate statistics gathering, use the following script:

```
SQL> @$ORACLE_HOME/rdbms/admin/utlestat.sql
```

- 1** The script takes a new snapshot of data from the dynamic performance tables (V\$xxx) to collect final statistics and stores these in the END tables.

```
SQL> insert into stats$end_latch select * from v$latch;
```

- 2** The script creates DIFFERENCE tables, where it stores the values of the subtraction of the results of the initial statistics from the final statistics.

```
SQL> create table stats$stats as select  e.value-b.value
2>    change , n.name
3>    from v$statname n, stats$begin_stats b,stats$end_stats e
4>    where n.statistic# = b.statistic#
5>    and   n.statistic# = e.statistic#;
```

- 3** The script generates a report by selecting data from these DIFFERENCE tables.
- 4** The script drops all the temporary views and tables.

The following table shows the link between stats tables and performance views.

Stats Table Name	Based on
stats\$end_stats	v\$sysstat
stats\$end_lib	v\$librarycache
stats\$end_event	v\$system_event
stats\$end_waitstat	v\$waitstat
stats\$end_roll	v\$rollstat
stats\$end_file	stats\$file_view
stats\$end_dc	v\$rowcache

## Examining the Statistics Report

### The Statistics Report

- Library cache statistics
- System statistics
- Wait events statistics
- Latch statistics
- Rollback contention statistics
- Buffer Busy Wait Statistics
- Dictionary cache statistics
- I/O statistics per data file and tablespace
- Period of measurement

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

### Examining the Statistics Report

You can generate the statistics report by using the `UTLESTAT.SQL` script. This report contains a sequence of `SELECT` statements on the `DIFFERENCE` tables.

```
SQL> spool report.txt;  
SQL> select ... from stats$lib;
```

**Note:** The subsequent sections are not explained in detail in this lesson. They are covered in other lessons. This list presents all the types of statistics available in this report to familiarize you with these areas of tuning.

### Library Cache Statistics

The library cache contains shared SQL and PL/SQL areas.

You can tune this area to reduce misses on either the parsing or execution of a SQL or PL/SQL statement. Any library cache misses will adversely affect the performance of the Oracle database.

## System Statistics

This section of the report provides, for each system-wide statistic, a total number of operations and the total number of operations per user commit and per logon. This helps you in tuning several areas.

### Example 1 Database Writer checkpoints:

Statistic	Total	Per Trans	Per Logon
-----	-----	-----	-----
DBWR checkpoint buffers written	1	1	1
DBWR free buffers found	11	11	11

*DBWR checkpoints* indicates the number of checkpoint messages that were sent to the Database Writer.

The increase in I/O of data during a checkpoint can result in a decrease in system performance.

You can reduce the number and frequency of checkpoints by increasing the `init.ora` parameters `LOG_CHECKPOINT_INTERVAL` and `LOG_CHECKPOINT_TIMEOUT`. However, be aware that infrequent checkpoints increase database recovery time.

### Example 2 consistent gets, db block gets, physical reads:

Statistic	Total	Per Transact	Per Logon
-----	-----	-----	-----
consistent gets	559715	1168.51	25441.59
db block gets	9949	20.77	452.23
physical reads	419280	875.32	19058.18

*consistent gets* is the number of blocks accessed in the buffer cache for queries without the `FOR UPDATE` clause.

*db block gets* is the number of blocks accessed in the buffer cache for `INSERT`, `UPDATE`, and `SELECT FOR UPDATE` statements.

Logical read is the sum of *consistent gets* and *db block gets* database buffer cache.

*physical reads* is the number of requests for a block that caused a physical I/O.

You calculate the hit ratio to detect if the size of the database buffer cache is large enough or if it cannot keep often-read blocks in memory.

### **Wait Events Statistics**

Each system Wait event is a context switch that costs CPU time. By looking at the total time for each statistic, you can often determine where the bottleneck is that processes are waiting for.

### **Latch Statistics**

The Oracle server uses latches to protect access to internal structures, such as the library cache for shared cursors, or the least recently used (LRU) list for data buffers in the buffer cache.

Tuning latch areas consists of reducing contention for latch allocation.

### **Rollback Contention Statistics**

If transactions need to wait to receive a slot in rollback headers before continuing to be processed, performance decreases.

This section helps you determine contention for the undo header.

### **Buffer Busy Wait Statistics**

If the Buffer Busy Wait event is high, this small section indicates which class of blocks has high contention: data block, segment header, or undo header.

### **Dictionary Cache Statistics**

Every SQL or PL/SQL statement implies access to dictionary objects and therefore to the dictionary cache. Misses in the dictionary cache cause an increase in data I/O and a corresponding decrease in performance. This section displays the gets and misses for each type of item cached.

Tuning this area means that you reduce the dictionary cache misses.

### **I/O Statistics per Data File/Tablespace**

This section displays how file I/O is spread across multiple disk drives by counting the number of physical read/writes, physical block read/writes, and the amount of time spent for these operations for each data file and tablespace.

### **Period of Measurement**

This section displays the time when the UTLBSTAT utility started to collect begin statistics and when UTLESTAT started to collect end statistics.

## Library Cache Statistics Section

### Library Cache Statistics

```
SQL> Rem Select Library cache statistics.The pinhitrate should be high.
SQL> select namespace library, gets,
3>         round(decode(gethits,0,1,gethits)/decode(gets,0,1,gets),3)
4>         gethitratio, pins,
6>         round(decode(pinhits,0,1,pinhits)/decode(pins,0,1,pins),3)
7>         pinhitratio, reloads, invalidations
9>   from stats$lib;
```

LIBRARY	GETS	GETHITRATI	PINS	PINHITRATI	RELOADS	INVALIDAT
BODY	105	1	105	1	0	0
CLUSTER	10	1	9	1	0	0
INDEX	0	1	0	1	0	0
OBJE	0	1	0	1	0	0
PIPE	0	1	0	1	0	0
SQL AREA	2036	.987	12822	.982	95	0
TABLE/PROCED	553	.98	3714	.969	81	0
TRIGGER	917	1	917	.997	3	0

8 rows selected.

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

### report.txt Output

This slide shows an example of the first section of the `report.txt`.

## I/O Statistics Section

### I/O Statistics

```
SQL> Rem I/O should be spread evenly accross drives. A big difference
between phys_reads and phys_blks_rd implies table scans are going on.
SQL> select table_space, file_name, phys_reads reads, phys_blks_rd
2> blks_read, phys_rd_time read_time, phys_writes writes, phys_blks_wr
3> blks_wrt, phys_wrt_tim write_time
4> from stats$files order by table_space, file_name;
```

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	65012	416752	38420	564	564	8860
SCOTT_INDEX	/DATA/scott_ind.dbf	8	8	0	8	8	0
SYSTEM	/DATA/DISK1/sys01.dbf	806	1538	1985	116	116	1721
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.

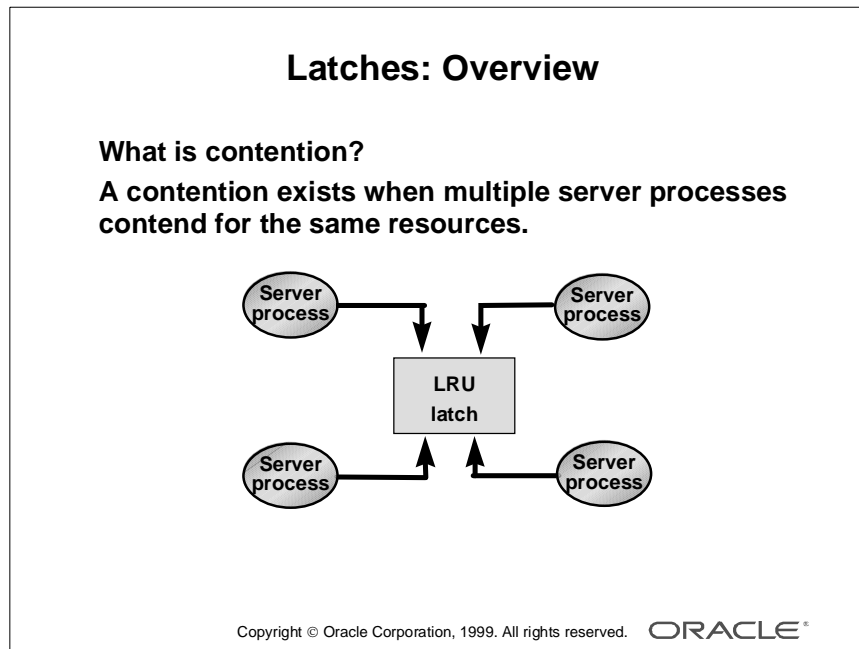
Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

### report.txt Output

This slide shows an example of the last section of the report.txt.



## Latches



### Definition

In an Oracle environment, memory structures are held in a consistent state for a short period while a process is accessing them. This is necessary to ensure that the structure does not change while it is being accessed.

Latches are used to ensure that these structures do not change. When multiple processes attempt to acquire these latches, contention exists.

The goal in tuning for latch contention is to minimize the contention between processes when latches are required.

## Latches

**Contention areas that the DBA can tune:**

- Redo allocation latch
- Redo copy latch
- LRU latch

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

### Categories

Latches are areas of contention for which the DBA can tune. However, there are very few latches over which the DBA has direct control. The three main areas that the DBA has direct control over for tuning are:

- Redo allocation latch
- Redo copy latch
- LRU latch

**Note:** Typically, you should have addressed buffer sizing and file I/O prior to tuning latches. Latch tuning will realize fewer performance gains than properly sizing memory structures or ensuring acceptable file I/O.

---

## Latch Types

### Latch Types

- **Willing-To-Wait:**
  - **GETS**
  - **MISSES**
  - **SLEEPS**
- **Immediate:**
  - **IMMEDIATE GETS**
  - **IMMEDIATE MISSES**

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

### Types

Different types of latches exist in the Oracle environment. Statistics for each category of latch, Willing-To-Wait and Immediate, can be found in the V\$LATCH view.

- **Willing-To-Wait:** If the latch requested with a Willing-To-Wait request is not available, the requesting process waits a short time and requests the latch again. The process continues waiting and requesting until the latch is available.
  - **GETS:** Shows the number of successful Willing-To-Wait requests for a latch
  - **MISSES:** Shows the number of times an initial Willing-To-Wait request was unsuccessful
  - **SLEEPS:** Shows the number of times a process waited and requested a latch after an initial Willing-To-Wait request
- **Immediate:** If the latch requested with an Immediate request is not available, the requesting process does not wait, but continues processing.
  - **IMMEDIATE GETS:** This column shows the number of successful immediate requests for each latch.
  - **IMMEDIATE MISSES:** This column shows the number of unsuccessful immediate requests for each latch.

## Oracle Wait Events

### Oracle Wait Events

The `V$EVENT_NAME` view lists a collection of Wait events that provide information on the sessions that had to wait to be processed:

- `EVENT#`
- `NAME`
- `PARAMETER1`
- `PARAMETER2`
- `PARAMETER3`

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

### List of Events

There are more than 100 Wait events in the Oracle server.

These events are listed in the `V$EVENT_NAME` view.

## V\$EVENT\_NAME View

```
SQL> SELECT name, parameter1, parameter2, parameter3
2 FROM v$event_name;
```

NAME	PARAMETER1	PARAMETER2	PARAMETER3
PL/SQL lock timer	duration		
alter system set mts_dispatcher	waited		
buffer busy waits	file#	block#	id
library cache pin	handle	addr	pin address 0*mode+name
log buffer space			
log file switch			
(checkpoint incomplete)			
transaction	undo seg#	wrap#	count
...			
136 rows selected.			

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

## Parameters Describing a Wait Event

**Example 1** The Buffer Busy Waits event waits until a buffer becomes available. This event occurs when a buffer is either being read into the buffer cache by another session (and the session is waiting for that read to complete), or is in the buffer cache but in an incompatible mode (that is, some other session is changing the buffer).

This event is completed with three parameters:

- **FILE# and BLOCK#:** These parameters identify the block number in the data file that is identified by the file number for the block for which the Oracle server needs to wait.
- **ID:** The Buffer Busy Waits event is called from different places in the session. Each place in the kernel points to a different reason. ID refers to the place in the session calling this event.

**Example 2** The Log File Switch (Checkpoint Incomplete) waits for a log switch because the session cannot wrap into the next log. Wrapping cannot be performed because the checkpoint for that log has not completed.

This event has no parameter.

### **Common Wait Events**

- Free Buffer Wait
- Latch Free
- Buffer Busy Waits
- Db File Sequential Read
- Db File Scattered Read
- Db File Parallel Write
- Undo Segment Tx Slot
- Undo Segment Extension

For the complete list, refer to the *Oracle8i Reference, Release 8.1.5*, Appendix A (A67790-01).

## Statistics Event Views

### Statistics Event Views

- **V\$SYSTEM\_EVENT**: Total waits for an event, all sessions together
- **V\$SESSION\_EVENT**: Waits for an event for each session that had to wait
- **V\$SESSION\_WAIT**: Waits for an event for current active sessions that are waiting

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

### Statistics Results of Waiting Sessions

The statistics results of the sessions that had to wait or are currently waiting for a resource can be viewed using the **V\$SESSION\_EVENT** and **V\$SESSION\_WAIT** views.

Cumulated statistics for all sessions can be viewed using the **V\$SYSTEM\_EVENT** view.

## V\$SYSTEM\_EVENT View

```
SQL> SELECT event, total_waits, total_timeouts,  
2 time_waited, average_wait  
3 FROM v$system_event;
```

EVENT	TOTAL_ WAITS	TOTAL_ TIMEOUTS	TIME_ WAITED	AVERAGE_ WAIT
-----	-----	-----	-----	-----
latch free	5	5	5	1
pmon timer	932	535	254430	272.993562
process startup	3		8	2.66666667
buffer busy waits	12	0	5	5
...				
23 rows selected.				

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

## V\$SYSTEM\_EVENT View

The V\$SYSTEM\_EVENT view shows the total waits for a particular event since instance startup.

If you are troubleshooting, you need to know when a process has waited for any resource. Therefore, it is useful to query this view each time the system slows down.

The V\$SYSTEM\_EVENT view contains the following columns:

- **EVENT:** Name of the wait event
- **TOTAL\_WAITS:** Total number of waits for the event
- **TOTAL\_TIMEOUTS:** Total number of timeouts for the event
- **TIME\_WAITED:** Total amount of time waited for this event, in hundredths of a second
- **AVERAGE\_WAIT:** The average amount of time waited for this event, in hundredths of a second



## V\$SESSION\_EVENT View

```
SQL> select sid, event, total_waits, average_wait
2> from v$session_event where sid=10;
```

SID	EVENT	TOTAL_WAITS	AVERAGE_WAIT
10	buffer busy waits	12	5
10	db file sequential read	129	0
10	file open	1	0
10	SQL*Net message to client	77	0
10	SQL*Net more data to client	2	0
10	SQL*Net message from client	76	0

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

## V\$SESSION\_EVENT View

The V\$SESSION\_EVENT view shows the same information as the V\$SYSTEM\_EVENT view, but by session. It includes the columns listed on the previous page, with an extra column, SID, to identify the session. You can join the SID column to V\$SESSION\_SID to find user details.

**Note:** You can query these views directly to find out about all system waits that have occurred since startup.

## V\$SESSION\_WAIT View

```
SQL> SELECT sid, seq#, event, wait_time, state
2      FROM v$session_wait;
```

SID	SEQ#	EVENT	WAIT TIME	STATE
1	1284	pmon timer	0	WAITING
2	1697	rdbms ipc message	0	WAITING
3	183	rdbms ipc message	0	WAITING
4	4688	rdbms ipc message	0	WAITING
5	114	smon timer	0	WAITING
6	14	SQL*Net message from client	-1	WAITED SHORT TIME

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

## V\$SESSION\_WAIT View

This view lists the resources or events for which active sessions are waiting.

### Columns

- SID: Session identifier
- SEQ#: Sequence number identifying the wait
- EVENT: Resource or event waited for
- P1TEXT: Description of the first additional parameter, which corresponds to the PARAMETER1 described for the V\$EVENT\_NAME view
- P1: First additional parameter value
- P1RAW: First additional parameter value in hexadecimal
- P2TEXT: Description of the second additional parameter, which corresponds to the PARAMETER2 described for the V\$EVENT\_NAME view
- P2: Second additional parameter value
- P2RAW: Second additional parameter value in hexadecimal
- P3TEXT: Description of the third additional parameter, which corresponds to the PARAMETER3 described for the V\$EVENT\_NAME view
- P3: Third additional parameter value

---

**V\$SESSION\_WAIT View (continued)****Columns (continued)**

- P3RAW: Third additional parameter value in hexadecimal
- WAIT\_TIME:

Value	Explanation
> 0	The session's last wait time
= 0	The session is currently waiting
= -1	The value was less than 1/100 of a second
= -2	The system cannot provide timing information

- SECONDS\_IN\_WAIT: Number of seconds the event waited
- STATE: WAITING, WAITED UNKNOWN TIME, WAITED SHORT TIME (less than one hundredth of a second), WAITED KNOWN TIME (the value is stored in the WAIT\_TIME column)

**Note:** Not all of the parameter columns are used for all events.

**TIMED\_STATISTICS Initialization Parameter**

Set the TIMED\_STATISTICS parameter to TRUE to retrieve values in the WAIT\_TIME column. It is a dynamic initialization parameter.

## Event Management System

### Event Management System

- **Monitors for unusual conditions in databases, nodes, and network by creating events**
- **Automates problem detection by registering events**
- **Automates problem correction by applying fixit jobs**
- **Shares events and notifies administrators of event occurrences**
- **Has five predefined event test categories: Space, Fault, Resource, Performance, and Audit Management**

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

### Event Management System

The Event Management (EM) system is useful to:

- Monitor for unusual conditions in databases, nodes, and networks by creating events; for example:
  - When a database shuts down, it can generate an alert (such as an alert message sent to the console, your pager number, or your e-mail).
  - When a tablespace is about to run out of free space, it can generate an alert.
  - When a particular table approaches or reaches its maximum limits, it can generate a warning or an alert.
- Automate problem detection by registering events
- Automate problem correction by applying fixit jobs that run automatically when the event occurs; for example, you can also ask EM to automatically add a data file to the tablespace when the event occurs

**Event Management System (continued)**

- Determine which administrators will be notified when an event occurs and during which period of the week they can be alerted  
(A single administrator, responsible for 50 or 100 databases, can monitor a large system but cannot connect to each database to check on its availability and performance. Rather than monitor all sites, administrators can pinpoint only those services they want to monitor.)
- Specify the way the administrators should be notified when an event occurs (by e-mail, pager, or logged messages on the console)

The EM Repository tables contain:

- Predefined events and events sets, and those created by the user
- The parameters and frequency for each event
- The fixit jobs for each event
- The registered events and their registration status
- The occurred events and their degree of alert
- The acknowledged events moved to history
- The list of administrators to be notified when events occur
- Information on how to notify administrators on duty when events occur
- The schedule for notifying the administrator on duty when events occur

**Predefined Event Tests Categories**

The next four slides detail the event tests for the Fault, Space, Resource, and Performance tuning areas. The Audit Management test category does not belong to the tuning scope.

## Predefined Event Tests

### Predefined Event Tests

#### Fault management event tests:

- Database Alert (database)
- Database UpDown (database)
- Archiver Hung (database)
- Database Probe (database)
- Data Block Corruption (database)
- Node UpDown (node)
- Session Terminated (database)

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

### Fault Management Events

These tests primarily monitor for faults that can occur at the database or node level. The fault management tests are:

- Database Alert (database): This test monitors when new errors have been encountered in the database alert log.
- Database UpDown (database): This test monitors database status. An event occurrence is issued if the database fails.
- Archiver Hung (database): This test monitors when database archiving has become suspended (only for databases in ARCHIVELOG mode). This can happen due to lack of disk space or disabling automatic archiving.
- Database Probe (database): Ensures that database connections can be made. If the listener fails or session limits are exceeded, an event occurrence is issued.
- Data Block Corruption (database): This test monitors integrity of blocks in the database. If an ORA-01578 error occurred in the last sample, an event occurrence is issued.
- Sqlnet UpDown (listener): This test monitors listener status. An event occurrence is issued if the listener fails.

## Predefined Event Tests

### Space management events:

- **Alert File Large (database)**
- **Chunk Small (database)**
- **Disk Full (node)**
- **Dump Full (database)**
- **Fast Segment Growth (database)**
- **Maximum Extents (database)**
- **Tablespace Full (database)**

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

## Space Management Events

These tests primarily monitor disk space requirements on machines or in the database. Some of the space management tests are:

- **Alert File Large (database):** This test monitors when the database alert log file has grown beyond specified warning and critical thresholds.
- **Chunk Small (database):** This test monitors certain tablespaces, segment names, or segment types to ensure more extents can be allocated into the free space. If the number of extents specified by the warning or critical threshold cannot be allocated in an object's tablespace, an event occurrence is issued.
- **Disk Full (node):** This test monitors the amount of free space on a specified disk. If the amount of free disk space falls below the specified warning or critical thresholds, an event occurrence is issued.
- **Dump Full (database):** This test monitors the amount of free space on the disk required by BACKGROUND\_DUMP\_DEST and USER\_DUMP\_DEST. If the amount of free disk space falls below the specified warning or critical thresholds, an event occurrence is issued.
- **Fast Segment Growth (database):** This test monitors specified segments that have allocated many extents since the last sample. If the number of extents recently allocated exceeds the specified warning or critical thresholds, an event occurrence is issued.

### **Space Management Events (continued)**

- **Maximum Extents (database):** This test causes an event occurrence if specific segments cannot allocate the number of extents specified in the warning or critical thresholds, because a MAXEXTENTS error would be encountered.
- **Tablespace Full (database):** This test monitors the percentage of used space in chosen tablespaces. If the specified warning or critical thresholds are exceeded, an event occurrence is issued.
- **Multiple Extents (database):** This test causes an event occurrence if specific segments contain more extents than allowed by the warning or critical thresholds.
- **Snapshot Log Full (database):** This test causes an event occurrence if the number of rows in all snapshot log tables is greater than the snapshot log's table size parameter.
- **Swap Full (node):** Monitors the percentage of swap space available. If the amount of swap space falls below the specified warning or critical thresholds, an event occurrence is issued.



## Predefined Event Tests

### Resource management events:

- **Datafile Limit (database)**
- **Lock Limit (database)**
- **Process Limit (database)**
- **User Limit (database)**
- **Session Limit (database)**

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

## Resource Management Events

These tests monitor database resource usage. The resource management tests are:

- **Datafile Limit (database):** Monitors the percentage of datafiles created in a database compared with the `DB_FILES` `init.ora` parameter.
- **Lock Limit (database):** Monitors the percentage of DDL or DML locks in a database compared with the `DML_LOCKS` `init.ora` parameter.
- **Process Limit (database):** Monitors the percentage of connected processes in a database compared with the `PROCESSES` `init.ora` parameter.
- **User Limit (database):** Monitors the percentage of concurrent user connections in a database compared with the `LICENSE_MAX_USERS` `init.ora` parameter.
- **Session Limit (database):** Monitors the percentage of database sessions (including background processes) compared with the `SESSIONS` `init.ora` parameter.

## Predefined Event Tests

### Performance management events:

- **Buffer Cache (database)**
- **Chain Row (database)**
- **CPU Utilization (node)**
- **Disk I/O (node)**
- **In Memory Sorts (database)**
- **Library Cache (database)**
- **Rollback Contention (database)**

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

## Performance Management Events

These tests primarily monitor database tuning statistics. The performance management tests are:

- **Buffer Cache (database):** This test monitors database buffer cache efficiency by calculating the hit ratio (a high percentage indicates that most data blocks requested were already located in the cache). An occurrence is issued when the hit ratio exceeds the warning or critical thresholds sequentially a certain number of times.
- **Chain Row (database):** This test monitors chained or migrated rows for all segments specified in the Segment Name, Segment Owner, or Segment Type parameters. An occurrence is issued when a segment contains a chained row.
- **CPU Paging (node):** This test monitors the amount of paging performed by the CPU (in KBytes/Second). An occurrence is issued when the paging rate exceeds the specified thresholds sequentially a certain number of times.
- **CPU Utilization (node):** This test monitors the percentage of CPU activity. An occurrence is issued when the percentage exceeds the specified thresholds sequentially a certain number of times.
- **Data Dictionary Cache (database):** This test monitors data dictionary cache efficiency by calculating the miss ratio (a low percentage indicates most system information was already located in the cache). An occurrence is issued when the miss ratio exceeds the thresholds sequentially a certain number of times.

**Performance Management Events (continued)**

- **Disk I/O (node):** This test monitors the amount of disk activity (in I/O per second). An occurrence is issued when the percentage exceeds the specified thresholds sequentially a certain number of times.
- **Free Buffer (database):** This test monitors the percentage of waits caused by DBWR needing to create a free buffer in the buffer cache. An occurrence is issued when the percentage of all samples exceeds the threshold.
- **Index Rebuild (database):** This test monitors when indexes need to be re-created. Choose indexes either by index name or owner, or by object (table) name or owner. An occurrence is issued when a monitored index needs to be rebuilt.
- **In Memory Sorts (database):** Monitors the percentage of data sorts performed in memory (a high percentage is good). An occurrence is issued when thresholds are continuously exceeded.
- **Library Cache (database):** This test monitors library cache efficiency by calculating the Miss Ratio (a low percentage indicates most SQL statements are shared in the cache reducing parse time). An occurrence is issued when the Miss Ratio continuously exceeds the thresholds a certain number of times.
- **Net I/O (database):** This test monitors the amount of network activity (in I/O per second). An occurrence is issued when the percentage exceeds the specified thresholds sequentially a certain number of times.
- **Redo Log Allocation (database):** This test monitors the percentage of log buffer hits by users (LGWR did not need to free space in the log buffer). An occurrence is issued when thresholds are continuously exceeded.
- **Rollback Contention (database):** This test monitors header contention in rollback segments (a low number is good). An occurrence is issued when thresholds are continuously exceeded.
- **SysStat Table (database):** This test monitors a named event statistic from the V\$SYSSTAT fixed view. An occurrence is issued when the parameter value of the statistic exceeds the specified thresholds a certain number of times.
- **SysStat Table Delta (database):** This test monitors a named event statistic from the V\$SYSSTAT fixed view. An occurrence is issued when the difference between two parameter values of the statistic exceed the specified thresholds a certain number of times.

**Note:** For more information on predefined event tests:

- Refer to the Help menu in Oracle Enterprise Manager.
- Read the appropriate script files on the server.

## How to Use a Predefined Event

An event can be created on any managed node through the following process:

- 1 From the console, select Event—>Create Event from the menu bar or the console toolbar.
- 2 Complete the four pages for the Create Event window:
  - General: The event name, type, description, frequency, and destination
  - Tests: Event tests, such as detecting chained rows or excessive CPU usage
  - Parameters: The parameters required for each chosen test
  - Permissions: The permissions (privileges) that other administrators in the repository have for this event
- 3 Decide whether this event needs to be registered, saved in the library, or both.
- 4 When all pages have been completed, specify whether the event will be run immediately (registered) or saved to the event library.

---

## Event Frequency and Parameters

### Event Parameters

**Parameters:**

- **Warning and alert thresholds**
- **Number of occurrences**
- **Focused monitoring:**
  - **SEGMENT\_OWNER**
  - **SEGMENT\_TYPE**
  - **SEGMENT\_NAME**
  - **Any criteria related to the area**

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

### Frequency

For each event, the frequency determines how often the event condition is to be checked by the intelligent agents. The administrator can adjust the frequency by modifying the existing events or creating new events.

Select the frequency for monitoring the event (from the pull-down lists) to determine how often the event condition is checked. If the threshold of the event remains above the specified level longer than the specified frequency, a new notification is not sent. But if the condition changes from a warning to an alert (or from an alert to a warning), a new notification is sent.

### Parameters

- **Warning and alert thresholds:** Some events have default values for alert and warning thresholds, which determine the values above which the administrator should be notified by a warning or an alert.

**Example:** The Process Limit event has an alert threshold of 90 and a warning threshold of 80.

- **Number of occurrences:** Some events have a default value for the NO\_OCCURENCE parameter that counts the number of times that a condition has exceeded given thresholds before an alert is generated.

### Parameters (continued)

- Focused monitoring: Some event test have default values for SEGMENT\_NAME, SEGMENT\_TYPE, SEGMENT\_OWNER parameters (\* value means all possible values) when these can restrict the selection of segments on which the event will monitor certain conditions.

**Example:** The Continued Row event has a default value set to \* for the segment name, segment owner, and segment type parameters.

- Depending on the event selected, there may or may not be parameters to set.

## Fix the Problem Detected by the Event

### Fix the Problem Detected by the Event

- **Manually**
- **Automatically by fixit jobs**

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

### Fixit Jobs

Fixit jobs are the automated or manual jobs that you would create to correct a problem when an event occurs. You can correct the problem detected by an event using one of two methods:

- When an event detects a problem, administrators may prefer to correct it manually because they must check the environment before applying any correction.  
If you did not supply their events with a fixit job when you defined it, you must correct the problem manually.

### Examples

- The Database UpDown event has detected that the database instance was shut down. The administrator must start it up again.
- The Sqlnet UpDown event has detected that the listener process on the server was down. The administrator must start it up again.
- Administrators may prefer to have a job running automatically when the alert or the warning is generated because they believe there are no preliminary operations to be performed before applying the correction. This can be performed with a fixit job applied to the event.

**Note**

- For more information on predefined job tasks, refer to the lesson on the Jobs System in *Oracle Enterprise Manager* or the Help menu (Predefined Job Tasks), or read the appropriate script files on the server.
- For detailed information on how to create, register, and display events status, refer to the lesson on Manager Events in *Oracle Enterprise Manager* or to the Oracle Enterprise Manager documentation.



## DBA-Developed Tools

### DBA-Developed Tools

- **Develop your own scripts.**
- **Use the Supplied Packages for tuning.**
- **Schedule periodic performance checking.**
- **Take advantage of the EM Job service to automate the regular execution of these administrative tasks.**
- **Take advantage of the EM Event service to track specific situations.**
- **Take advantage of the EM Job service to apply tasks that automatically solve problems detected by EM event service.**

3-31

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

### In-House Scripts

The utilities and Oracle tools may not provide all the statistics you need. Therefore, you might create your own scripts to do the following:

- Check for free space left or not left at all, in every data file
- Determine whether segments have enough space to be able to be extended.
- Describe schema structures to show tables and associated indexes
- Use Oracle Supplied Packages (created by `$ORACLE_HOME/rdbms/admin/dbms*.sql` scripts) (Refer to the *Oracle Database Administration* course.)

## Oracle Packs

### Oracle Packs

- **Oracle Diagnostics Pack:**
  - Performance Manager
  - TopSessions
  - Oracle Trace Manager
  - Trace Data Viewer
  - Capacity Planner
- **Oracle Tuning Pack:**
  - Tablespace Manager
  - SQL Analyze
  - Oracle Expert

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

### Oracle Diagnostics and Tuning Packs

Besides the standard suite of applications of EM, the Diagnostics and Tuning packs provide an optional set of windows-based integrated monitoring and tuning applications that address many Oracle performance management areas, such as graphical monitoring, analysis, and automated tuning of Oracle databases, providing real-time graphical performance information.

#### Performance Manager

This application provides the ability to monitor database performance in real time. It provides dozens of predefined charts for displaying a wide variety of database performance statistics regarding users, throughput, tablespaces, redo logs, buffers, caches, and I/O.

#### TopSessions

This application monitors how connected sessions use database-instance resources in real time. You can obtain an overview of session activity, by displaying the top *n* sessions sorted by a statistic of your choice.

The application also allows you to monitor locks, which are mechanisms that prevent destructive interaction between users accessing the same resources.

**Oracle Trace Manager**

This application enables you to monitor performance by collecting data about events that happen in applications. For this data collection to take place, the application must contain calls to Oracle Trace routines.

**Oracle Trace Data Viewer**

This application enables you to view the trace formatted output and make appropriate tuning or configuration decisions.

**Capacity Planner**

This application enables you to plan for system resources based on the current workload.

**Tablespace Manager**

This application allows you to monitor and manage database storage. You can display an overview of tablespace usage information and use the coalescing feature to join adjacent free blocks.

**SQL Analyze**

This application enables you to tune the SQL application.

**Oracle Expert**

This application enables you to optimize the performance of your database environment. Oracle Expert assists you with the initial configuration of the database and with the collection and evaluation of the performance characteristics of existing databases. The tool provides tuning recommendations that you can implement immediately.

## Performance Manager

**Performance Manager**

**Predefined scopes of statistics:**

- I/O
- Contention
- Database instance
- Load
- Memory
- Top resource consumers
- Overview of performance
  - Overview of cache utilization
  - Overview of user activity
  - Overview of throughput
  - Overview of performance default chart

**User-defined charts**

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

### Performance Manager Characteristics

The Performance Manager application captures, computes, and presents performance data in a graphical, real-time view that allows you to monitor key metrics required to:

- Use memory effectively
- Minimize disk I/O
- Avoid resource contention

The data displayed in real-time mode can be recorded for replay.

You can define new charts and display windows containing charts in many categories. Seven different categories of predefined charts are available for display. Each category has a set of specific charts that focus on the parent subject.

### Lock Manager

These charts include Locks chart, Blocking / Waiting Locks chart and User Type Locks chart.

### I/O

These charts include File I/O Rate, File I/O Rate Details, Network I/O Rate, and System I/O Rate.

## Contention

These charts include Circuit, Dispatcher, Free List Hit %, Latch, Lock, Queue, Redo Allocation Hit %, Rollback NoWait Hit %, and Shared Server.

## Database Instance

These charts include Process, Session, System Statistics, Table Access, Tablespace, Tablespace Free Space, # Users Active, # Users Logged on, # Users Waiting, # Users Waiting for Locks, and # Users Running.

## Load

These charts include Buffer Gets Rate, Network Bytes Rate, Redo Statistics Rate, Sort Rows Rate, Table Scan Rows Rate, and Throughput Rate.

## Memory

These charts include Buffer Cache Hit %, Data Dictionary Cache Hit %, Library Cache Hit %, Library Cache Details, SQL Area, Memory Allocated, Memory Sort Hit %, Parse Ratio, and Read Consistency Hit %.

## Top Resource Consumers

Top Resource Consumers is one of the predefined charts for database services.

## Overview of Performance

The overview displays a composite of the most commonly used charts from the other categories. Use the overview charts to get an overall picture of your database activity, then drill down to the more specific reports if you need to. The set includes the following:

**Overview of Throughput** This is a group chart for the Overview of Performance class. The icon depicts four small bar graphs.

**Overview of Cache Utilization** This group of charts includes Buffer Cache Hit %, Library Cache Hit %, Data dictionary Cache Hit %, Memory Sort Hit %, and Rollback (Nowait) Hit %.

## User-Defined

If you have defined any charts of your own, you can select from among them by using this category.

## TopSessions

**TopSessions**

USERNAME	SID	OSUSER	BYTES	COMMAND	STATUS	MACHINE	PROGRAM
PROBSON-LAP	17	probson	4139148	UNKNOWN	INACTIVE	WVVED-DOMAIN-PROBSON-LAP	re -noit -D0mx1hrPolType=MThr
PROBSON-LAP	18	probson	1933628	UNKNOWN	INACTIVE	WVVED-DOMAIN-PROBSON-LAP	re -noit -D0mx1hrPolType=MThr
PROBSON-LAP	24	probson	643669	UNKNOWN	IN	WVVED-DOMAIN-PROBSON-LAP	re -noit -D0mx1hrPolType=MThr
PROBSON-LAP	9	probson	390254	UNKNOWN	IN	WVVED-DOMAIN-PROBSON-LAP	re -noit -D0mx1hrPolType=MThr
PROBSON-LAP	22	probson	108405	UNKNOWN	IN	WVVED-DOMAIN-PROBSON-LAP	re -noit -D0mx1hrPolType=MThr
PROBSON-LAP	16	probson	63196	UNKNOWN	IN	WVVED-DOMAIN-PROBSON-LAP	re -noit -D0mx1hrPolType=MThr
PROBSON-LAP	25	probson	51796	UNKNOWN	IN	WVVED-DOMAIN-PROBSON-LAP	re -noit -D0mx1hrPolType=MThr
PROBSON-LAP	21	probson	41807	UNKNOWN	IN	WVVED-DOMAIN-PROBSON-LAP	re -noit -D0mx1hrPolType=MThr
PROBSON-LAP	20	probson	28460	UNKNOWN	IN	WVVED-DOMAIN-PROBSON-LAP	re -noit -D0mx1hrPolType=MThr
SYSTEM	23	probson	11117	UNKNOWN	IN	WVVED-DOMAIN-PROBSON-LAP	re -noit -D0mx1hrPolType=MThr
PROBSON-LAP	15	probson	4777	UNKNOWN	IN	WVVED-DOMAIN-PROBSON-LAP	re -noit -D0mx1hrPolType=MThr
SYSTEM	19	o813	1177	UNKNOWN	IN	WVVED-DOMAIN-PROBSON-LAP	re -noit -D0mx1hrPolType=MThr

Statistic	Value
bytes received via SQL*Net from client	1277
bytes received via SQL*Net from dblink	0
bytes sent via SQL*Net to client	1310
bytes sent via SQL*Net to dblink	0
CPU used by this session	0
logons cumulative	1
logons current	1
opened cursors cumulative	9
opened cursors current	1
recursive calls	25
recursive cpu usage	0
serializable aborts	0
session connect time	0
session logical reads	79
session pga memory	124508
session pga memory max	124508
session stored procedure space	0
session uga memory	26364
session uga memory max	26364
SQL*Net roundtrips to/from client	28
SQL*Net roundtrips to/from dblink	0
user calls	27
user commits	0
user rollbacks	0

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

### TopSessions Characteristics

Often a DBA needs more information than is provided by general database monitoring. For example, he or she may detect a file I/O problem through Performance Manager. To solve the problem quickly, it would be helpful to know which particular user sessions are causing the greatest I/O activity.

TopSessions is the tool for monitoring:

- How connected sessions use database instance resources in real time. You can display the top  $n$  sessions sorted by the statistic of your choice. You can also identify specific SQL statements being executed by these sessions.
- Sessions (users) currently holding or waiting for locks in the database from each session.
- The disconnection of users that are causing database problems, such as consuming too many resources or have appeared to hang.

## Overview Display

The TopSessions window displays a certain number of user sessions connected to the database in a specific sort order (configurable). The window contains the following features:

- 1 Information contained in the window varies depending on which options are selected. It shows the following columns for each user who is currently connected to the monitored database:
  - Username: Name of the Oracle database user currently connected
  - SID (Session Identifier): Uniquely identifies the current session
  - OSUser: Name of the operating system user from the machine establishing the connection
  - Sort Statistic Value: Displays values based on the statistic filter and sort statistic value chosen from the Option window
  - Command: The SQL command currently being issued by that session
  - Status: Whether the session is currently ACTIVE, INACTIVE, or KILLED
  - Machine: The machine from which the user requested the connection
  - Program: The program that initiated the session connection, such as SQL\*Plus
- 2 The Session menu option provides access to:  
Details: If a session is selected, this option will view the session statistics and general session information.
- 3 The username and service name of the database being monitored is displayed in the title bar of the TopSessions window.
- 4 To sort the TopSessions window contents by a column, click the column heading.

## Details Window (Statistics Filter)

You can drill down on any session to get the TopSessions Details window.

You can specify which group of statistics needs to be monitored by selecting a value from the Statistics Filter field. This populates the Sort Statistics field with all statistics that belong to the group. Each statistic filter is described below with some of the statistics that belong to the group (from V\$STATNAME):

- Predefined: Statistics that monitor CPU usage, File IO, Memory, Open cursors, and User transactions.
- User: Statistics, such as bytes sent using SQL\*Net, CPU used, number of logons, number of commits or rollbacks, and PGA memory used, are monitored.

### **Details Window (Statistics Filter) (continued)**

- **Redo:** This group monitors redo log and log buffer statistics, such as number of blocks written to logs, number of redo entries created, amount of wastage per redo block, and time wasted on redo log space.
- **Enqueue:** This group monitors locking statistics, such as the number of DML locks requested, deadlocks encountered, and locks waited for.
- **Cache:** This group contains statistics to monitor the database buffer cache, such as number of consistent reads, database block gets, physical reads, DBWR checkpoints, and DBWR LRU scans.
- **Operating System:** Information contained in this group is operating system specific.
- **Parallel Server:** This group monitors statistics related to Oracle Parallel Server performance, such as DDL or DML statements parallelized, global lock synchronization, and amount of information sent between instances.
- **SQL:** Statistics for SQL statements, such as number of parses and executions, sorts in memory and disk, and full table scans on tables, are monitored.
- **Other:** This group contains other statistics that may be considered useful.
- **All:** This group contains all statistics available.



## TopSessions: Locks

User Name	Session ID	Lock Type	Mode	Object Name	Object Owner	Object Type
SYSTEM	23	TM	Exclusive	None	PETER	SYSTEM TABLE
SYSTEM	23	TX	Exclusive	None	RBS_01	SYS ROLLBACK...
SYSTEM	24	TX	None	Exclusive	RBS_01	SYS ROLLBACK...

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

### TopSessions Details (TSD): Locks Page

The Locks page lists information primarily contained from within the V\$LOCK fixed view. The important information located on this page includes:

- This line displays a user waiting on a row lock. The lock type and mode requested can be viewed to determine the severity of the lock.
- Every lock held in each user session is displayed. Any row with a + can be expanded to reveal other sessions waiting for this lock to be released.
- All locks can be displayed for the session, or only the locks that are waiting due to other blocking sessions.
- The object name displays the object causing the locking problems. For TX locks, the rollback segment name is displayed. For TM locks, the table name is displayed.

### Locks Page Fields

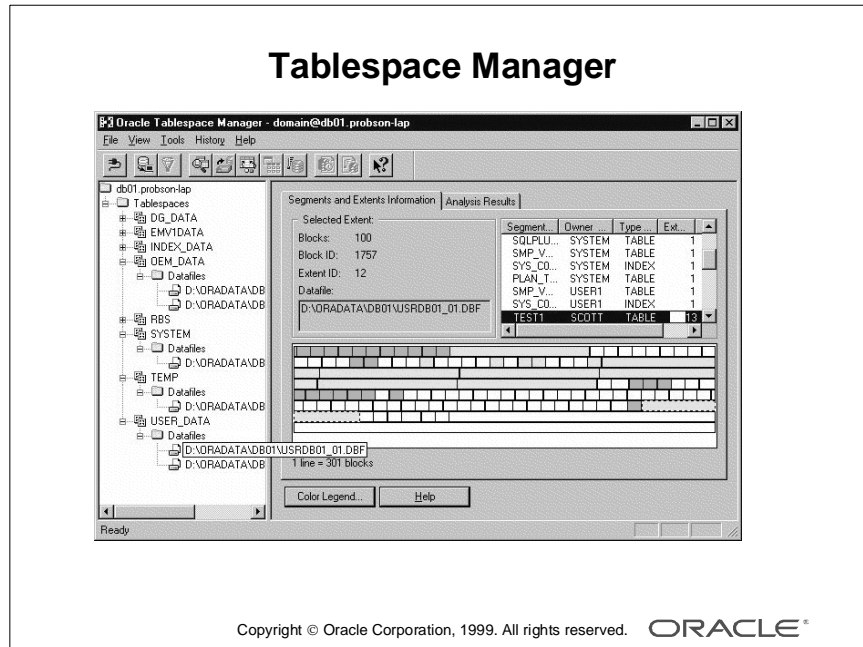
The fields for each row located on the Locks page are:

- User Name: Name of the database user who established this connection
- Session ID: A unique number assigned to the session during connection

### **Locks Page Fields (continued)**

- Lock Type: Some of the commonly acquired lock types are:
  - TM: Table locks provide concurrent access on tables
  - TX: Transaction (or row) locks provide concurrent access rows
  - MR: Media Recovery locks indicate that database files are online
  - ST: Space Transaction locks indicate that SMON is coalescing adjacent free extents because of extent growth or shrinkage in poorly tuned databases
- Mode Held: The level of lock currently held on the table or row acquired through a successful SQL statement; the five main table-level lock modes are:
  - Row Share: A SELECT... FOR UPDATE statement obtains this lock
  - Row Exclusive: Any DML statement obtains this table lock
  - Shared Row Exclusive: A lock usually obtained from a statement using the ON DELETE CASCADE option of a foreign key constraint
  - Shared: A lock sometimes obtained to prevent DML on a parent table in a foreign key relationship
  - Exclusive: Can only be obtained through a LOCK TABLE command (It is also the only type of row level lock obtained through DML commands.)
- Mode Requested: The level of lock waiting to be acquired on the table or row requested by a SQL statement (It cannot obtain the lock due to an existing lock on the table or row. The table-level lock modes are the same as Mode Held.)
- Object Name: Name of the object that the lock is acting on
- Object Owner: Name of the database user who owns the object
- Object Type: Type of object, such as TABLE or ROLLBACK
- Resource ID1: For TM lock types, this number identifies the table using OBJECT\_ID from the DBA\_OBJECTS view; for TX lock types, this number divided by 65535 identifies the rollback segment number from UNDO\$
- Resource ID2: This value is rarely used

## Oracle Tablespace Manager



### Tablespace Manager Characteristics

Suppose the I/O section of the `report.txt` output makes you suspect database performance problems due to tablespace disorganization. In such cases, you can use Oracle Tablespace Manager to monitor and manage database storage.

You can drill down to:

- Graphically display how storage has been allocated for database segments
- Analyze selected objects
- Deallocate unused space for selected segments
- Defragment fragmented segments (This functionality uses Export/Import utilities.)
- Coalesce free adjacent blocks to make them more useful space for future extents

Many of the other tools deal with the logical side of a database. The Tablespace Manager deals with the physical side.

## Oracle Trace Manager

### Common Uses of Oracle Trace Manager

- Resource-usage information collection
- Performance analysis
- Database tuning
- Application tuning
- Input to expert systems

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

### Trace Manager Characteristics

Oracle Trace Manager is an application used to gather performance and resource-usage data from other software products and application programs.

Most data used in performance monitoring applications is collected based on sampling methodologies. For example, Performance Manager and TopSessions use this technique by periodically collecting data from the dynamic performance views.

Oracle Trace Manager provides a new data collection methodology: it collects data for each and every occurrence of key events in a software application or database event that is being monitored.

### Common Uses of Oracle Trace Manager

- Resource-usage information is collected through a set of predefined items that Oracle Trace Manager provides to application developers. This set includes most of the system characteristics you typically look at, including CPU time, direct I/O statistics, page faults, and so on.
- Performance analysis can be accomplished either through your own interpretation of the collected data or the use of a secondary application that imports the Oracle Trace data.

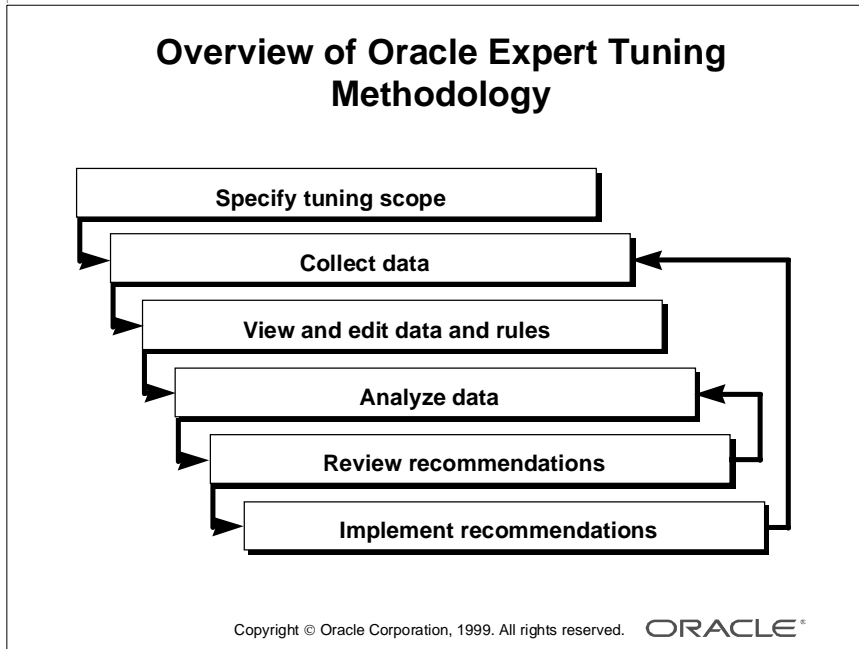
**Common Uses of Oracle Trace Manager (continued)**

- Database tuning is possible because the Oracle server has been instrumented with Oracle Trace Manager routine calls. The relevant data items have already been selected for you.
- Application tuning is possible for any application that has been instrumented with the Oracle Trace service routine calls. The software kit includes a sample bank automated teller machine (ATM) application, which was instrumented to illustrate application tuning.
- Input to expert systems is a key reason for the existence of Oracle Trace Manager. Oracle Trace Manager does not provide any interpretation of its own; it is just a data collector. Other applications, such as Oracle Expert, can apply heuristic rules to this data and develop optimized suggestions based on actual workloads collected by Oracle Trace.

**Examples of Events**

- Connection: Records each connection to a database (point event)
- Disconnect: Records each disconnection from a database (point event)
- SQL Segment: Text of a SQL statement (point event)
- Parse: Start and end of event that contains SQL query information (actual text of query) (duration event)
- Execute: Analyzes the access methods used to retrieve data (duration event)
- Fetch: Describes the output returned by the statement (duration event)

## Oracle Expert



### Characteristics

Oracle Expert provides automated performance tuning with integrated rules. It automates:

- Data collection
- Collection analysis
- Recommendations generation
- Script creation for implementing recommendations

---

## Tuning Categories

### Tuning Session Scope

Four major tuning categories:

- Instance optimizations
- SQL reuse opportunities
- Appropriate space management
- Optimal data access

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

### Tuning Scopes

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

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

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

**Optimal Data Access** Determines what indexes are required or redundant for tables. The check for Optimal Data Access field can be customized by choosing 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

## Tuning Recommendations

### Tuning Recommendations

1. Collect the data.
2. Review the recommendations:
  - Session data report
  - Analysis report
3. Implement recommendations:

Type of Recommendations	File Type
Instance	.ora
Structure	.txt

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

### Collection

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.

The five collection classes are chosen according to the selected tuning category:

- The data collected for *Database class* recommends using available features for your release of Oracle8i and distributing database information among the database tablespaces.
- The data collected for the *Instance class* recommends changing SGA, sort, parallel query, I/O parameters.
- The data collected for *Schema class* provides instance tuning, index structuring, and sizing recommendations.
- The data collected for *Environment class* concerns logical device data and system 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). It recommends making instance modifications.



**Collection (continued)**

- *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.

There are three ways to input workload data:

- Import data from an Oracle Trace collection
- Collect data from the SQL cache of your database
- Reuse 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.

**Session Data Report**

This report provides detailed information about the data collected.

**Analysis Report**

Oracle Expert sifts through the collected data in this report, and uses its rules to generate tuning recommendations.

**Instance Recommendations Implementation**

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.

**Index Recommendations Implementation**

Execute the `.sql` file generated by Oracle Expert. Note that when Oracle Expert recommends modifying an existing index, the `.sql` file contains SQL statements that first delete the existing index, then create it again with the recommended modifications.

**Structure Recommendations Implementation**

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:

- Collect statistics from dictionary and dynamic performance troubleshooting views
- Collect statistics from `report.txt` output of UTLBSTAT and UTLESTAT scripts
- Define latch types
- Retrieve Oracle Wait events information
- Set alerts through EM events
- Collect statistics using the GUI tools of Oracle Enterprise Manager, such as the Diagnostics Pack and Tuning Pack

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

## Quick Reference

Context	Reference
Initialization parameters	TIMED_STATISTICS
Dynamic performance views	V\$FIXED_TABLE V\$SYSSTAT V\$SGASTAT V\$EVENT_NAME V\$SYSTEM_EVENT V\$SESSION V\$SESSTAT V\$SESSION_EVENT V\$SESSION_WAIT V\$LIBRARYCACHE V\$ROWCACHE V\$LATCH V\$ROLLSTAT V\$WAITSTAT V\$FILESTAT V\$DATAFILE
Data dictionary views	DBA_HISTOGRAMS DBA_TABLES DBA_TAB_COLUMNS DBA_CLUSTERS DBA_INDEXES INDEX_STATS INDEX_HISTOGRAM
Commands	ANALYZE ALTER SYSTEM SET TIMED_STATISTICS=true
Packaged procedures and functions	None
Scripts	utlbstat.sql utlestat.sql catperf.sql dbms*.sql



## **Tuning the Shared Pool**

## Objectives

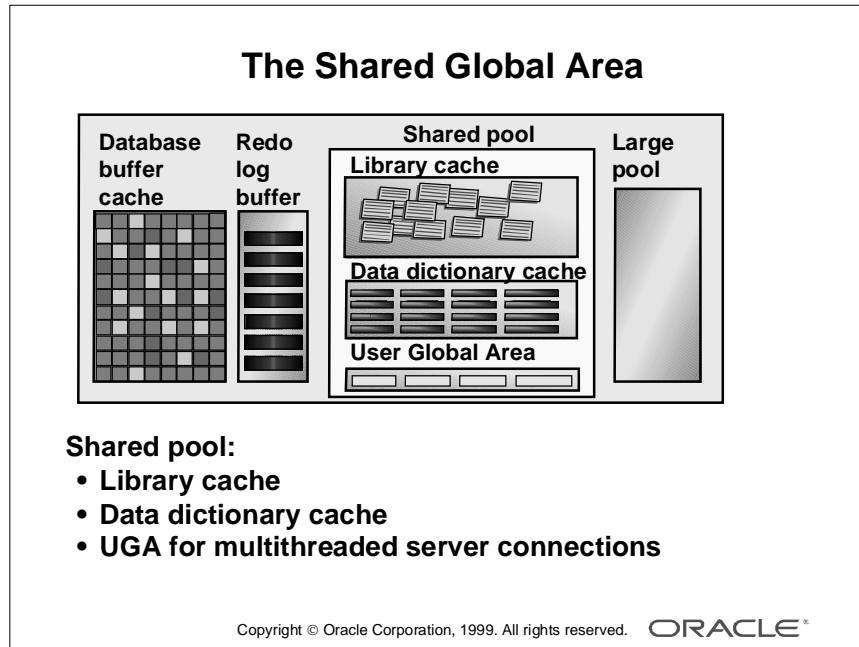
### Objectives

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

- **Tune the library cache and the data dictionary cache**
- **Measure the shared pool hit ratio**
- **Size the shared pool appropriately**
- **Pin objects in the shared pool**
- **Tune the shared pool reserved space**
- **Describe the User Global Area (UGA) and session memory considerations**
- **Configure the large pool**

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

## The Shared Global Area



### Shared Pool Contents

The *shared pool* contains two main structures plus a third:

- The *library cache*, which stores shared SQL and PL/SQL areas
- The *data dictionary cache*, which keeps information about dictionary objects
- The *User Global Area (UGA)*, which keeps information about multithreaded connections

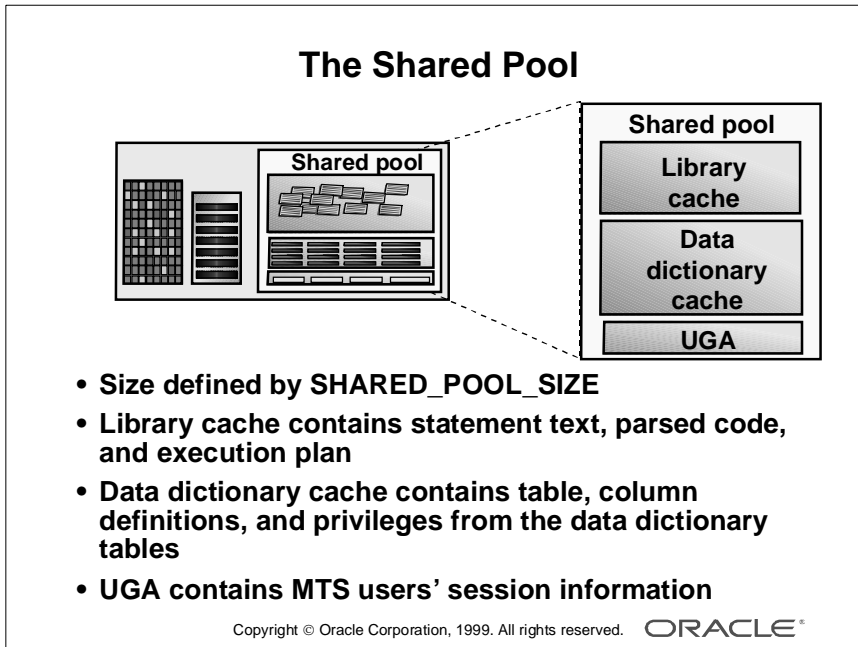
### Tuning the Shared Global Area

A cache miss on the data dictionary cache or library cache is more expensive than a miss on the database buffer cache. Tuning the shared pool is a priority.

When you tune the shared pool, you are likely to be mainly concerned with the library cache, since the Oracle algorithm tends to hold dictionary data in memory longer than library cache data. Therefore, tuning the library cache to an acceptable cache hit ratio ensures that the data dictionary cache hit ratio is also acceptable.

If the shared pool is too small, the server must dedicate resources to managing the limited space available. This consumes CPU resources and causes contention.

## The Shared Pool



### Size of the Shared Pool

You set the size of the shared pool with the `init.ora` parameter `SHARED_POOL_SIZE`. It defaults to 3,500,000 bytes.

### The Library Cache

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

PL/SQL blocks include:

- Procedures
- Functions
- Packages
- Triggers
- Anonymous PL/SQL blocks

### The Data Dictionary Cache

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

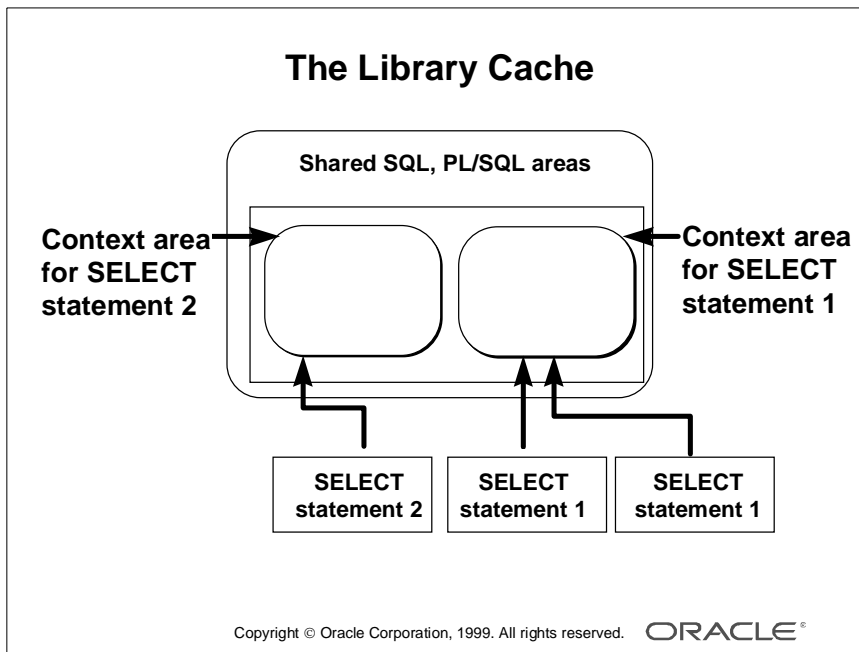


## The Library Cache

### The Library Cache

- Used to store SQL statements and PL/SQL blocks to be shared by users
- Managed by an LRU algorithm
- Used to prevent statements reparsing

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



## SQL and PL/SQL Storage

The Oracle server uses the library cache to store SQL statements and PL/SQL blocks. A least recently used (LRU) algorithm is used to manage the cache.

If a user issues a statement that is already in the cache, the Oracle server can use the cached version without having to reparse it.

To determine whether a statement is already cached, the Oracle server:

- 1 Reduces the statement to the numeric value of the ASCII text
- 2 Uses a hash function of this number

## Tuning the Library Cache

### Tuning the Library Cache

**Reduce misses by keeping parsing to a minimum:**

- **Make sure that users can share statements**
- **Prevent statements from being aged out by allocating enough space**
- **Avoid invalidations that induce reparsing**

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

### First Tuning Goal

Reduce misses by keeping parsing to a minimum:

- If an application makes a parse call for a SQL statement and the parsed representation of the statement does not already exist in a shared SQL area in the library cache, the Oracle server parses the statement and allocates a shared SQL area.  
Ensure that SQL statements can share a shared SQL area whenever possible by using:
  - As much generic code as possible
  - Bind variables rather than constants
- If an application calls for a SQL statement, and the shared SQL area containing the parsed representation of the statement has been deallocated from the library cache to make room for another statement, the Oracle server implicitly reparses the statement, allocates a new shared SQL area for it, and executes it. Reduce library cache misses on execution calls by allocating more memory to the library cache.
- If a schema object is referenced in a SQL statement and that object is later modified in any way, the shared SQL area becomes invalidated.

## **Tuning the Library Cache**

**Avoid fragmentation by:**

- **Reserving space for large memory requirements**
- **Pinning frequently required large objects**
- **Eliminating large anonymous PL/SQL blocks**
- **Reducing UGA consumption of MTS connections**

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

## **Second Tuning Goal**

Avoid fragmentation by:

- Ensuring enough contiguous space for large memory requirements through the allocation of reserved space in the shared pool area
- Pinning frequently required large objects such as SQL and PL/SQL areas in memory, instead of aging them out with the normal LRU mechanism
- Using small PL/SQL packaged functions instead of large anonymous blocks
- Measuring session memory use by the shared server processes in multithreaded connections

---

## Terminology

### Terminology

- **GETS:** The number of lookups for objects of the namespace
- **PINS:** The number of reads or executions of the objects of the namespace
- **RELOADS:** The number of library cache misses on the execution step, causing implicit reparsing of the statement and block

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

### Three Keywords

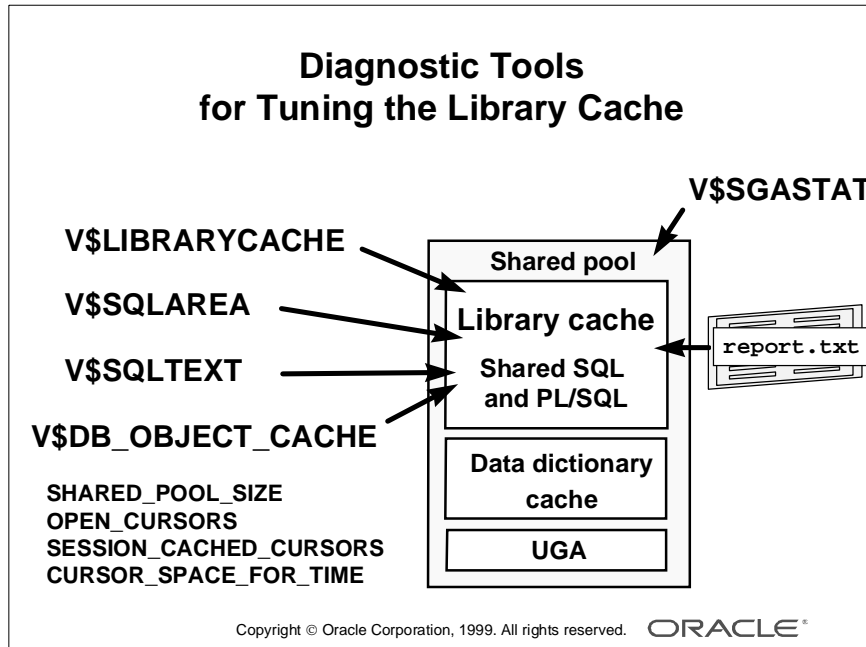
Each row in the V\$LIBRARYCACHE view contains statistics for one type of item kept in the library cache. The item described by each row is identified by the value of the NAMESPACE column. Rows of the table with the following NAMESPACE values reflect library cache activity for SQL statements and PL/SQL blocks: SQL AREA, TABLE/PROCEDURE, BODY, TRIGGER.

Rows with other NAMESPACE values reflect library cache activity for object definitions that Oracle uses for dependency maintenance: INDEX, CLUSTER, OBJECT, PIPE.

Three keywords related to the namespace are:

- **GETS:** This keyword shows the total number of requests for information on the corresponding item.
- **PINS:** For each of these areas, PINS shows the number of executions of SQL statements or procedures.
- **RELOADS:** If an execute call for a SQL statement is done and the shared SQL area containing the parsed representation of the statement has been deallocated from the library cache to make room for another statement or because the objects the statement refers to have been invalidated, the Oracle server implicitly reloads the statement and therefore reparses it. The number of reloads is counted for each of these namespaces.

## Diagnostic Tools for Tuning the Library Cache



### Description of the Views

- V\$SGASTAT: Sizes of all SGA structures:

```
SQL> select * from v$sgastat;
```

POOL	NAME	BYTES
shared pool	free memory	6447072
shared pool	dictionary cache	160932
shared pool	library cache	463100
shared pool	sql area	1220580
shared pool	sessions	242880
...		

(Because the shared pool acts as a cache, nothing will ever be aged out until all the free memory has been used up. Free memory is more properly thought of as wasted space. A high value of free memory is a symptom of fragmentation.)

- V\$LIBRARYCACHE: Statistics on library cache management
- V\$SQLAREA: Full statistics about all shared cursors, and the first 1,000 characters of the SQL statement
- V\$SQLTEXT: The full SQL text without truncation, in multiple rows
- V\$DB\_OBJECT\_CACHE: Database objects cached, including packages; also objects such as tables and synonyms, where these are referenced in SQL statements

## Shared Cursors

### Are Cursors Being Shared?

Check GETHITRATIO in V\$LIBRARYCACHE:

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

Find out which statements users are running:

```
SQL> select sql_text, users_executing,
2         executions, loads
3   from v$sqlarea;
```

```
SQL> select * from v$sqltext
2  where sql_text like
3  'select * from scott.s_dept where id =%';
```

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

### Are Cursors Being Shared?

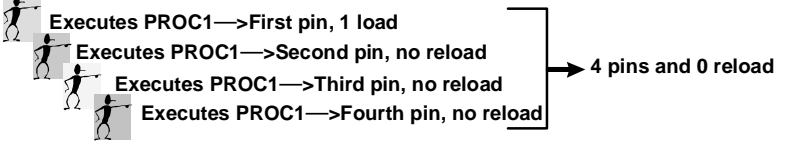
The GETHITRATIO in the V\$LIBRARYCACHE view determines the percentage of parse calls that find a cursor to share (GETHITS/GETS). This ratio should be in the high 90s. If not, there is probably room to improve the efficiency of your application code.

```
SQL> select namespace, gethitratio
2   from v$librarycache;
NAMESPACE          GETHITRATIO
-----
SQL AREA            .86928702
TABLE/PROCEDURE     .80073801
BODY                .6
```

**Note:** Oracle Enterprise Manager Diagnostics Pack application:  
Performance Manager—>Memory—>Library Cache Details

## Guidelines

### Guidelines: Library Cache Reloads



**Reloads should:**

- Ideally be 0
- Never be more than 1% of the pins

```
SQL> select sum(pins) "Executions", sum(reloads)
2         "Cache Misses", sum(reloads)/sum(pins)
3   from v$sqlibrarycache;
Executions Cache Misses sum(reloads)/sum(pins)
-----
2641          10          .00378644
```

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

### How to Get the Reloads-to-Pins Ratio

- The V\$LIBRARYCACHE view shows whether statements that have already been parsed have been aged out of the cache. The number of reloads should not be more than 1% of the number of pins.
- The report.txt output: this section indicates the same ratio for the period when UTLBSTAT and UTLESTAT ran.

LIBRARY	PINS	PINHITRATI	RELOADS
-----			
BODY			1051 0
SQL	AREA		12822.98295
TABLE/PROCED3714	.969		81
TRIGGER917	.997		3

**Note:** Oracle Enterprise Manager Diagnostics Pack application:  
Performance Manager—>Memory—>Library Cache Hit%



## Guidelines: Library Cache Reloads

report.txt output::

LIBRARY RELOADS	GETS INVALIDATI	GETHITRATI	PINS	PINHITRATI
SQL AREA	2036	.987	12822	.982
95	0			

**If the reloads-to-pins ratio is greater than 1%,  
increase the SHARED\_POOL\_SIZE parameter.**

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

### Guidelines

If the reloads-to-pins ratio is greater than 1 percent, there are two possible reasons:

- Shared parsed areas have been aged out, though required by successive reexecutions, because of a lack of space
- Shared parsed areas are invalidated

To avoid these frequent reloads, increase the `init.ora` parameter `SHARED_POOL_SIZE`.

## Invalidations

### Invalidations

This column represents the number of times objects of the namespace were marked invalid, causing reloads.

```
SQL> select namespace,pins,reloads,invalidations
2  from v$llibrarycache;
NAMESPACE          PINS    RELOADS  INVALIDATIONS
-----
SQL AREA           1793         10           0
SQL> ANALYZE TABLE scott.s_dept COMPUTE STATISTICS;
SQL> select * from scott.s_dept;
```

NAMESPACE	PINS	RELOADS	INVALIDATIONS
SQL AREA	1797	11	4

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

### When Invalidations Occur

If a schema object is referenced in a SQL statement and that object is later modified in any way, the shared SQL area is invalidated (marked invalid) and the statement must be reparsed the next time it is executed, and therefore reloaded.

For example, when a table, sequence, synonym, or view is re-created, altered, or dropped, or a procedure or package specification is recompiled, all dependent shared SQL areas are invalidated.

**Note:** Oracle Enterprise Manager Diagnostics Pack application:  
Performance Manager—>Memory—>Library Cache Details

## Sizing the Library Cache

### Sizing the Library Cache

- Define the global space necessary for stored objects (packages, views, and so on).
- Define the amount of memory used by the usual SQL statements.
- Reserve space for large memory requirements, to avoid misses and fragmentation.
- Keep frequently used objects.
- Convert large anonymous PL blocks into small anonymous blocks that call packaged functions.

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

## Global Space Allocation

### Global Space Allocation

Stored objects such as packages and views:

```
SQL> select sum(sharable_mem)
2   from V$DB_OBJECT_CACHE;
SUM(SHARABLE_MEM)
-----
379600
```

SQL statements:

```
SQL> select sum(sharable_mem)
2   from V$SQLAREA where executions > 5;
SUM(SHARABLE_MEM)
-----
381067
```

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

### Testing Your Applications

For an existing application, you can set up a test and use the dynamic views to find out how much memory is used. Begin by setting `SHARED_POOL_SIZE` to a very large value (at the expense of other structures, if necessary), then run the application.

### Computation of Shareable Memory Used

- For stored objects such as packages and views, use the following query:  
SQL>SELECT SUM(sharable\_mem)  
2 FROM v\$db\_object\_cache  
3 WHERE type = 'PACKAGE' or type = 'PACKAGE BODY' or  
4 type = 'FUNCTION' or type = 'PROCEDURE';
- For SQL statements, you need to query `V$SQLAREA` after the application has been running for a while. For frequently issued statements, you can use the following query to estimate the amount of memory being used, though this will not include dynamic SQL:

```
SQL> SELECT SUM(sharable_mem)
2   FROM v$sqlarea
3   WHERE executions > 5;
```

**Computation of Shareable Memory Used (continued)**

- You should also allow about 250 bytes in the shared pool per user per open cursor. This can be tested during peak times with the following query:

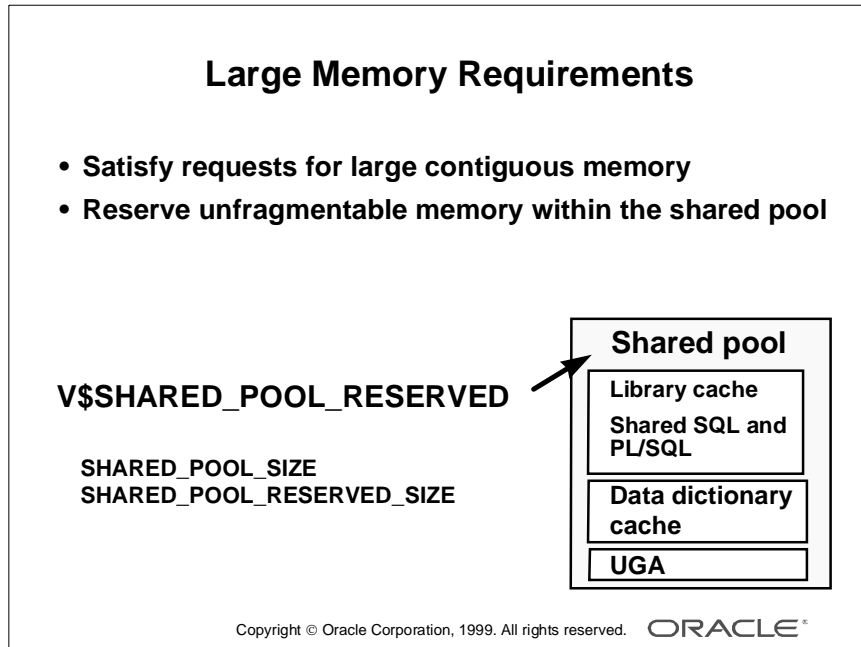
```
SQL> SELECT SUM(250 * users_opening)
       2 FROM v$sqlarea;
```

In a test environment, you can measure shareable memory by selecting the number of open cursors for a test user. You multiply the resulting value by the total number of users:

```
SQL> SELECT 250 * value bytes_per_user
       2 FROM v$sesstat s, v$statname n
       3 WHERE s.statistic# = n.statistic#
       4 AND n.name = 'opened cursors current'
       5 AND s.sid = 15;
```

Your application should ideally have a library cache as large as the sum of the numbers above, plus a small allowance for dynamic SQL.

## Large Memory Requirements



### Why Reserve Space in the Shared Pool?

The DBA can reserve memory within the shared pool to satisfy large allocations during operations such as PL/SQL compilation and trigger compilation. Smaller objects will not fragment the reserved list, helping to ensure that the reserved list will have large contiguous chunks of memory. Once the memory allocated from the reserved list is freed, it returns to the reserved list.

### Initialization Parameter

The size of the reserved list, as well as the minimum size of the objects that can be allocated from the reserved list, are controlled by the following initialization parameter:

**SHARED\_POOL\_RESERVED\_SIZE:** Controls the amount of **SHARED\_POOL\_SIZE** reserved for large allocations (Set the initial value to 10% of the **SHARED\_POOL\_SIZE**). If **SHARED\_POOL\_RESERVED\_SIZE** is greater than half of the **SHARED\_POOL\_SIZE**, the Oracle server signals an error.

## V\$SHARED\_POOL\_RESERVED View

This view helps in tuning the reserved pool and space within the shared pool.

The columns of the view are only valid if the parameter `SHARED_POOL_RESERVED_SIZE` is set to a valid value:

```
SQL> desc V$SHARED_POOL_RESERVED
```

Name	Null?	Type
-----	-----	-----
FREE_SPACE		NUMBER
AVG_FREE_SIZE		NUMBER
FREE_COUNT		NUMBER
MAX_FREE_SIZE		NUMBER
USED_SPACE		NUMBER
AVG_USED_SIZE		NUMBER
USED_COUNT		NUMBER
MAX_USED_SIZE		NUMBER
REQUESTS		NUMBER
REQUEST_MISSES		NUMBER
LAST_MISS_SIZE		NUMBER
MAX_MISS_SIZE		NUMBER

<b>where:</b>	FREE_SPACE	is the total of free space in the reserved list
	AVG_FREE_SIZE	is the average size of the free memory on the reserved list
	MAX_FREE_SIZE	is the size of the largest free piece of memory on the reserved list
	REQUEST_MISSES	is the number of times the served list did not have a free piece of memory to satisfy the request, and proceeded to start flushing objects from the LRU list

The following columns in the view contain values that are valid even if the parameter is not set:

- REQUEST\_FAILURES
- LAST\_FAILURE\_SIZE
- ABORTED\_REQUEST\_THRESHOLD
- ABORTED\_REQUESTS
- LAST\_ABORTED\_SIZE

<b>where:</b>	REQUEST_FAILURES	is the number of times that no memory was found to satisfy a request
	LAST_FAILURE_SIZE	is the size of the last failed request

## Tuning the Shared Pool Reserved Space

### Tuning the Shared Pool Reserved Space

Diagnostic tools for tuning:

- The `V$SHARED_POOL_RESERVED` dictionary view
- The supplied package and procedure:
  - `DBMS_SHARED_POOL`
  - `ABORTED_REQUEST_THRESHOLD`

Guidelines: Set the `SHARED_POOL_RESERVED_SIZE` parameter

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

### Diagnostics with `V$SHARED_POOL_RESERVED` View

Statistics from the `V$SHARED_POOL_RESERVED` view can help you tune the parameters. On a system with ample free memory to increase the SGA, the goal is to have `REQUEST_MISSES` equal 0 or not to have any `REQUEST_FAILURES` or at least prevent this value from increasing.

### Diagnostics with `ABORTED_REQUEST_THRESHOLD` Procedure

The `ABORTED_REQUEST_THRESHOLD` procedure, in the package `DBMS_SHARED_POOL`, enables you to limit the amount of shared pool to flush prior to reporting an ORA-4031 error, in order to limit the extent of a flush that could occur because of a large object.

### Guidelines When `SHARED_POOL_RESERVED_SIZE` Is Too Small

The reserved pool is too small when the value for `REQUEST_FAILURES` is more than zero and increasing. To resolve this, you can increase the value for the `SHARED_POOL_RESERVED_SIZE` and `SHARED_POOL_SIZE` accordingly. The settings you select for these depend on your system's SGA size constraints.

This option increases the amount of memory available on the reserved list without having an effect on users who do not allocate memory from the reserved list. As a second option, reduce the number of allocations allowed to use memory from the reserved list; however, doing so increases the normal shared pool, which may affect other users on the system.



### **Guidelines When SHARED\_POOL\_RESERVED\_SIZE Is Too Large**

Too much memory may have been allocated to the reserved list if:

- REQUEST\_MISS = 0 or not increasing
- FREE\_MEMORY = > 50% of SHARED\_POOL\_RESERVED\_SIZE minimum

If either of these is true, decrease the value for SHARED\_POOL\_RESERVED\_SIZE

### **Guidelines When SHARED\_POOL\_SIZE Is Too Small**

The V\$SHARED\_POOL\_RESERVED fixed table can also indicate when the value for SHARED\_POOL\_SIZE is too small. This may be the case if REQUEST\_FAILURES is greater than 0 and increasing.

If you have enabled the reserved list, decrease the value for SHARED\_POOL\_RESERVED\_SIZE. If you have not enabled the reserved list, you could increase SHARED\_POOL\_SIZE.

## Keeping Large Objects

### Keeping Large Objects

- Find those PL/SQL objects that are not kept in the library cache:

```
SQL> select * from v$db_object_cache
2  where sharable_mem > 10000
3  and (type='PACKAGE' or type='PACKAGE BODY' or
4       type='FUNCTION' or type='PROCEDURE')
5  and KEPT='NO';
```

- Pin large packages in the library cache:

```
SQL> EXECUTE dbms_shared_pool.keep('package_name');
```

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

### Why and When to Keep Objects

Loading large objects is the primary source of fragmentation. Users' response time is affected because of the large number of small objects that need to be aged out from the shared pool to make room. To prevent these situations, keep large or frequently required objects in the shared pool to make sure that they are never aged out of the shared pool.

- Which objects to keep:
  - Frequently required large procedural objects such as STANDARD, DBUTIL packages, and those for which shareable memory exceeds a defined threshold
  - Compiled triggers that are often executed on frequently used tables
  - Sequences, because sequence numbers are lost when the sequence is aged out of the shared pool
- When to keep them: Startup time is the best, because it prevents further fragmentation.
- Flushing the shared pool using the command ALTER SYSTEM FLUSH SHARED\_POOL does not flush kept objects.

### How to Keep Objects

Use the supplied DBMS\_SHARED\_POOL package and the KEEP procedure.

To create the package, run the dbmspool.sql script. The prvtpool.plb script is automatically executed at the end of the previous one. These scripts are not run by catproc.sql.

Use the UNKEEP procedure to remove pinned objects from the shared pool.

## Anonymous PL/SQL Blocks

### Anonymous PL/SQL Blocks

Find the anonymous PL/SQL blocks and convert them into small anonymous PL/SQL blocks that call packaged functions.

```
SQL> select sql_text from v$sqlarea
2  where command_type = 47
3  and length(sql_text) > 500;
```

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

### Two Solutions to Eliminate Large Anonymous PL/SQL Blocks

- Find them and convert them into small anonymous PL/SQL blocks that call packaged functions.
- If an anonymous PL/SQL block cannot be turned into a package, it can be identified in the V\$SQLAREA view and marked KEPT.

You can then keep these blocks in memory, using the appropriate supplied procedure.

```
SQL> declare x number;
2  begin x := 5;
3  end;
```

would become:

```
SQL> declare /* KEEP_ME */ x number;
2  begin x := 5;
3  end;
```

```
SQL> select address, hash_value
2  from v$sqlarea
3  where command_type = 47 and sql_text like '% KEEP_ME %';
```

Execute the KEEP procedure on the anonymous PL/SQL block identified by the address and hash\_value retrieved from the previous statement.

```
SQL> execute dbms_shared_pool.keep('address,hash_value');
```

## Other Parameters That Affect the Library Cache

### Other Parameters Affecting the Library Cache

- **OPEN\_CURSORS**
- **CURSOR\_SPACE\_FOR\_TIME**
- **SESSION\_CACHED\_CURSORS**

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

### Initialization Parameters

The following parameters also affect the library cache:

- **OPEN\_CURSORS**: This parameter defines the number of cursors referencing private SQL areas allocated to the user's process. A private SQL area continues to exist until the cursor is closed and therefore still exists after the completion of the statement.

To take advantage of additional memory available for shared SQL areas, you may need to increase the number of cursors permitted for a session. Application developers should close unneeded opened cursors to conserve system memory. The default value is 50.

- **CURSOR\_SPACE\_FOR\_TIME**: This is a Boolean parameter that defaults to FALSE. If you set it to TRUE, you choose to use space to gain time; shared SQL areas are not aged out until the cursor referencing them is closed. Therefore, make sure that there is free memory and no cache misses. Do not change this parameter unless the value of RELOADS in V\$LIBRARYCACHE is consistently 0. If your application uses Forms, or any dynamic SQL, leave the setting at FALSE.

### **Initialization Parameters (continued)**

- **SESSION\_CACHED\_CURSORS:** This parameter helps in situations in which a user repeatedly parses the same statements. This occurs in Forms applications when users often switch between forms; all the SQL statements opened for a form will be closed when you switch to another one. The parameter causes closed cursors to be cached within the session. Therefore, any subsequent call to parse the statement will bypass the parse phase. (This is similar to **HOLD\_CURSORS** in the precompilers.)

To check that your setting is efficient, compare the session statistics “session cursor cache hits” and “parse count” in the **V\$SESSTAT** view for a typical user session. If few parses result in hits, you might increase the number. Remember that this will increase overall demands on memory.

The default is 0, which means no caching.

## The Data Dictionary Cache

### The Data Dictionary Cache, Terminology, and Tuning

- **Content: Definitions of dictionary objects**
- **Terminology:**
  - **GETS: Number of requests on objects**
  - **GETMISSES: Number of requests resulting in cache misses**
- **Tuning: Avoid dictionary cache misses**

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

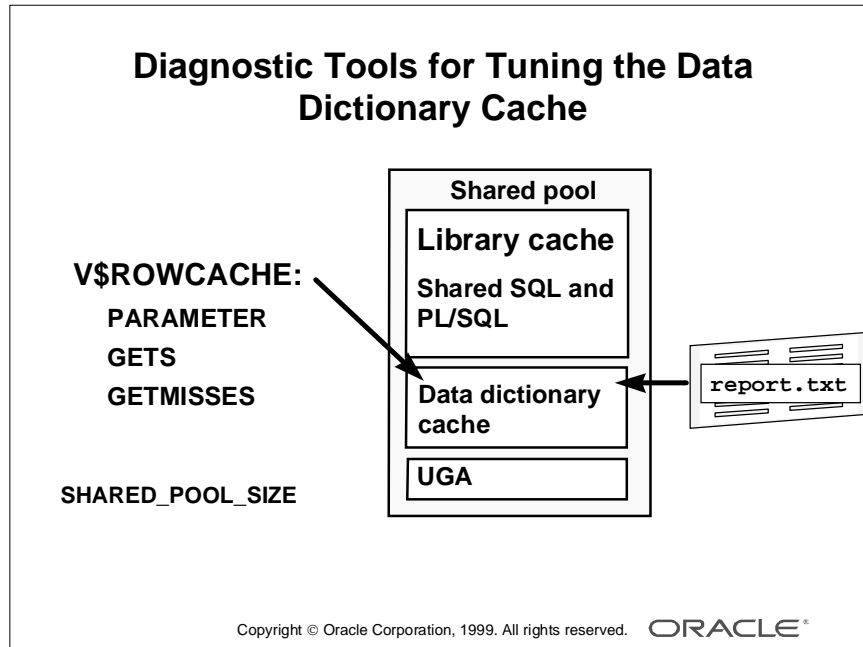
### Two Keywords

- **GETS:** Shows the total number of requests for information on the corresponding item (For example, in the row that contains statistics for file descriptions, this column has the total number of requests for file descriptions data.)
- **GETMISSES:** Shows the number of data requests resulting in cache misses

### Tuning Goal

Misses on the data dictionary cache are to be expected in some cases. Upon instance startup, the data dictionary cache contains no data, so any SQL statement issued is likely to result in cache misses. As more data is read into the cache, the likelihood of cache misses should decrease. Eventually, the database should reach a “steady state” in which the most frequently used dictionary data is in the cache. At this point, very few cache misses should occur. To tune the cache, examine its activity only after your application has been running for some time.

## Diagnostic Tools



### Monitoring the Dictionary Cache

Use the V\$ROWCACHE view. The columns of interest in monitoring the dictionary cache are shown in the following table:.

Column	Description
PARAMETER	Categories of data dictionary items
GETS	Requests for information on that category
GETMISSES	Requests resulting in cache misses

### Sizing

You can size the dictionary cache only indirectly with the SHARED\_POOL\_SIZE parameter. The algorithm for allocating shared pool space gives preference to the dictionary cache.

## Tuning the Data Dictionary Cache

### Tuning Data Dictionary Cache

**Keep the ratio of the sum of GETMISSES to the sum of GETS less than 15%:**

```
SQL> select parameter, gets, getmisses  
2      from v$rowcache;
```

PARAMETER	GETS	GETMISSES
dc_objects	143434	171
dc_synonyms	140432	127

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

### Goal for a Good Ratio

The ratio of the sum of all GETMISSES to the sum of all GETS should be less than 15% during normal running. If it is higher, consider increasing SHARED\_POOL\_SIZE.

You cannot hope to achieve a zero value for GETMISSES, because an object definition must be loaded into the cache the first time after startup that a server needs the object definition.



## Guidelines

### Guidelines: Dictionary Cache Misses

`report.txt` output:

NAME	GET_REQS	GET_MISS
-----	-----	-----
dc_objects	143434	171
dc_synonyms	140432	127

If there are too many cache misses, increase the parameter `SHARED_POOL_SIZE`.

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

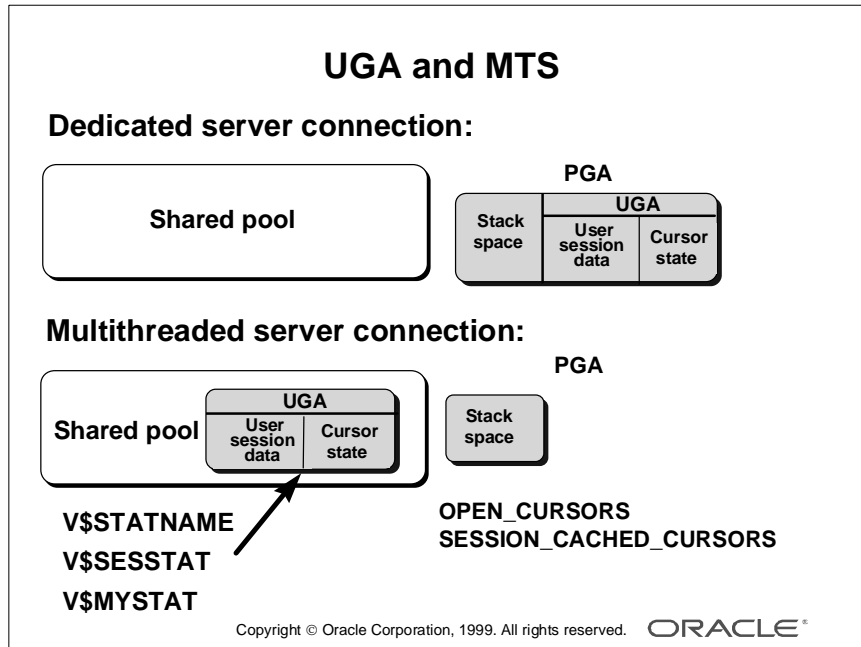
### Ratio from the `report.txt` Output

If the `report.txt` output indicates a high `GET_MISS/GET_REQ` ratio for a number of items, this indicates that `SHARED_POOL_SIZE` should be increased.

**Note:** Oracle Enterprise Manager Diagnostics Pack application:

Performance Manager—>Memory—>Data Dictionary Cache Hit%

## User Global Area and Multithreaded Server



### The User Global Area

If you use the multithreaded server, user session and cursor state information is stored in the shared pool instead of in private user memory. Sort areas and private SQL areas are included in the session information. This is because shared servers work on a per-statement basis, so any server may need access to any user's information. This part of the shared pool is called the User Global Area (UGA).

The total memory requirement for the multithreaded server is no larger than if you use dedicated servers. You may need to increase `SHARED_POOL_SIZE`, but your private user memory is less.

## Sizing the User Global Area

### Sizing the User Global Area

#### UGA space used by your test connection:

```
SQL> select SUM(value) || 'bytes' "Total session memory"
2   from V$MYSTAT, V$STATNAME
3   where name = 'session uga memory'
4   and v$mystat.statistic# = v$statname.statistic#;
```

#### UGA space used by all MTS users:

```
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#;
```

#### Maximum UGA space used by all MTS users:

```
SQL> select SUM(value) || 'bytes' "Total max memory"
2   from V$SESSTAT, V$STATNAME
3   where name = 'session uga memory max'
4   and v$sesstat.statistic# = v$statname.statistic#;
```

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

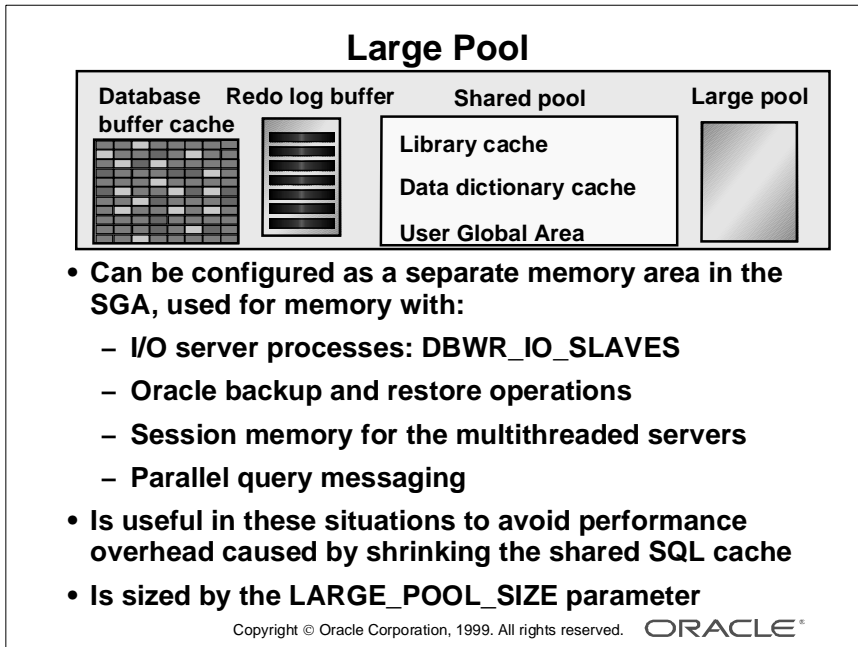
### Required Space Measurement

For all multithreaded connections, you need to compute the amount of space required for all shared server users to put their session memory in the shared pool.

**Note:** Oracle Enterprise Manager Diagnostics Pack application:

Performance Manager—>Memory—>Memory Allocated

## The Large Pool



### Existence of the Large Pool

Oracle8 introduced a new area in the SGA similar to the shared pool. There is no large pool allocated: it must be explicitly configured when needed. The memory of the large pool does not come out of the shared pool, but directly out of the SGA, thus adding to the amount of shared memory the Oracle server needs for an instance at startup.

### Advantages

The large pool is used to provide large memory allocations for session memory for the:

- I/O server processes
- Oracle backup and restore operations

The memory for Oracle server backup and restore operations and for I/O server processes is allocated in buffers of a few hundred kilobytes. The large pool is better able to satisfy such requests than the shared pool.
- Multithreaded server: By allocating session memory from the large pool for the multithreaded server, the Oracle server can use the shared pool primarily for caching shared SQL and avoid the performance overhead caused by shrinking the shared SQL cache.

**Advantages (continued)**

- Parallel execution: When the `PARALLEL_AUTOMATIC_TUNING` parameter is set to `TRUE`, the Oracle Server allocates parallel execution buffers from the large pool. When this parameter is set to `FALSE`, Oracle allocates parallel execution buffers from the shared pool. The Oracle server automatically computes the `LARGE_POOL_SIZE` parameter value if the `PARALLEL_AUTOMATIC_TUNING` parameter value is set to `TRUE`. To manually set a value for `LARGE_POOL_SIZE`, query the `V$SGASTAT` view and increase or decrease the value for `LARGE_POOL_SIZE` depending on your needs.

**Behavior**

If the `LARGE_POOL_SIZE` initialization parameter is not set, then the Oracle server attempts to allocate shared memory buffers from the shared pool in the SGA. If `LARGE_POOL_SIZE` is set but is not large enough, the allocation fails and the Oracle server component requesting the buffers does the following:

- The log archiving fails and returns an error.
- The `RMAN` command writes a message to the alert file and does not use I/O slaves for that operation.
- When an MTS session is started, a small amount of memory for the fixed UGA is allocated in the shared pool and the rest of the UGA is taken from the large pool. If there is insufficient space left in the large pool, an ORA-4031 error is returned:

```
ORA-4031: unable to allocate 636 bytes of shared memory (  
"large pool","EMP","session heap","define var info")
```

The large pool does not have an LRU list. It is different from reserved space in the shared pool, which uses the same LRU list as other memory allocated from the shared pool.

**How to Configure the Large Pool**

If the parameter `LARGE_POOL_SIZE` is not set, then there is no large pool. The specified size of memory is allocated from the SGA.

- Description: Size of the large pool, in bytes (can specify values in KB or MB)
- Minimum: 600 KB
- Maximum: At least 2 GB (the maximum is operating system specific)

## How to Configure the Large Pool (continued)

- To determine how the large pool is being used, query the V\$SGASTAT view:

```
SQL> SELECT *
      2 FROM v$sgastat
      3 WHERE pool = 'large pool';
```

POOL	NAME	BYTES
large pool	free memory	4194304

The Oracle server automatically computes LARGE\_POOL\_SIZE if PARALLEL\_AUTOMATIC\_TUNING is set to TRUE. To manually set a value for LARGE\_POOL\_SIZE, query the V\$SGASTAT view and increase or decrease the value for LARGE\_POOL\_SIZE depending on your needs.

For example, if the Oracle server displays the following error on startup:

```
ORA-27102: out of memory
SVR4 Error: 12: Not enough space
```

Consider reducing the value for LARGE\_POOL\_SIZE low enough so that your database starts. If, after lowering the value of LARGE\_POOL\_SIZE, you see the error:

```
ORA-04031: unable to allocate 16084 bytes of shared memory
("large pool","unknown object","large pool hea","PX msg pool")
```

execute the following query to determine why the Oracle Server could not allocate the 16,084 bytes:

```
SQL> SELECT NAME, SUM(BYTES) FROM V$SGASTAT
      2 WHERE POOL='LARGE POOL' GROUP BY ROLLUP (NAME);
```

Oracle should respond with output similar to:

NAME	SUM(BYTES)
PX msg pool	1474572
free memory	562132
	2036704

3 rows selected.

To resolve this, increase the value for LARGE\_POOL\_SIZE. This example shows the LARGE\_POOL\_SIZE to be about 2 MB. Depending on the amount of memory available, you could increase the value of LARGE\_POOL\_SIZE to 4 MB and attempt to start your database.

## Summary

### Summary

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

- **The shared pool, which is made up of:**
  - **The shared SQL and PL/SQL areas (library cache)**
  - **The data dictionary cache or row cache**
  - **The User Global Area, if connections are multithreaded server connections, unless the large pool is configured**
- **Tuning the library cache**
- **Configuring the large pool**

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

## Quick Reference

Context	Reference
Initialization parameters	LARGE_POOL_SIZE PARALLEL_AUTOMATIC_TUNING SHARED_POOL_SIZE OPEN_CURSORS SESSION_CACHED_CURSORS CURSOR_SPACE_FOR_TIME SHARED_POOL_RESERVED_SPACE
Dynamic performance views	V\$SGASTAT V\$LIBRARYCACHE V\$SQLAREA V\$SQLTEXT V\$DB_OBJECT_CACHE V\$ROWCACHE V\$SHARED_POOL_RESERVED V\$STATNAME V\$SESSTAT V\$MYSTAT
Data dictionary views	None
Commands	None
Packaged procedures and functions	DBMS_SHARED_POOL.KEEP DBMS_SHARED_POOL.UNKEEP DBMS_SHARED_POOL.ABORTED_REQUEST_THRESHOLD
Scripts	dbmspool.sql prvtpool.plb
Oracle Diagnostics Pack	Performance Manager



## **Tuning the Buffer Cache**

## Objectives

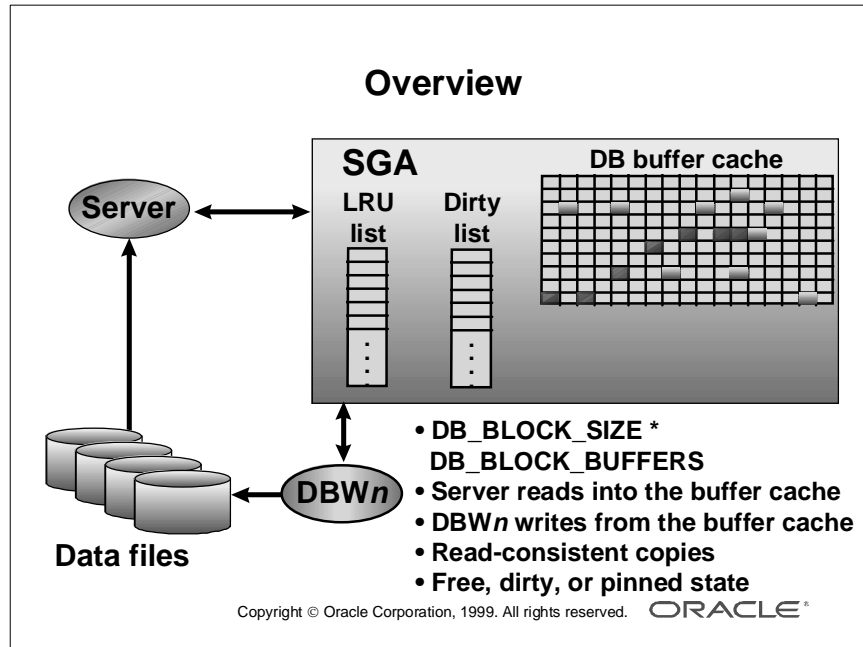
### Objectives

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

- **Describe how the buffer cache is managed**
- **Calculate and tune the buffer cache hit ratio**
- **Tune the buffer cache hit ratio by adding or removing buffers**
- **Create multiple buffer pools**
- **Size multiple pools**
- **Monitor buffer cache usage**
- **Make appropriate use of table caching**
- **Diagnose LRU latch contention**
- **Avoid free list contention**

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

## Buffer Cache Overview



### Buffer Cache Characteristics

The buffer cache holds copies of the data blocks from the data files. Because the buffer cache is a part of the SGA, these blocks can be shared by all users. The buffer cache has the following characteristics:

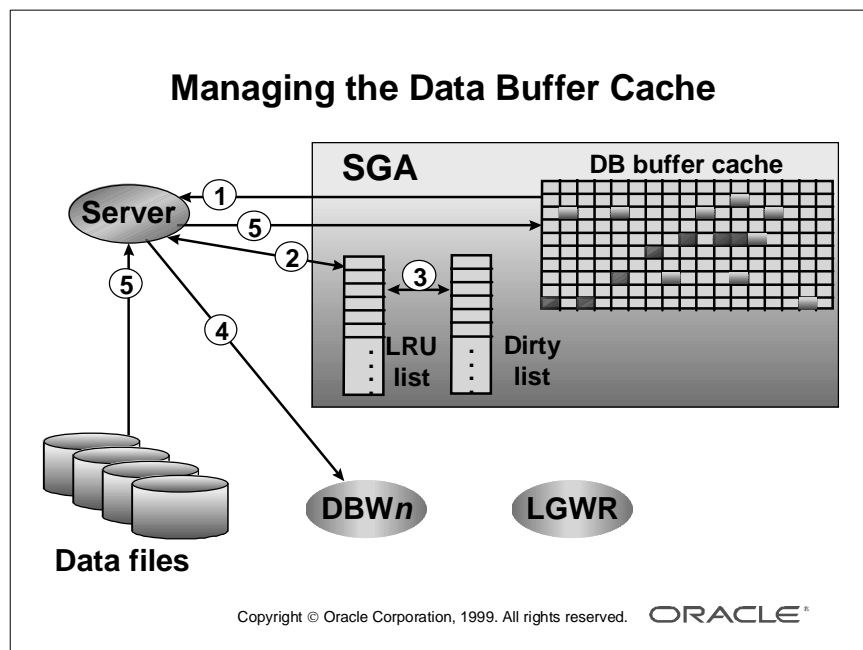
- It is sized using the **DB\_BLOCK\_BUFFERS** parameter. This parameter specifies the number of blocks in the buffer cache. To find the size of the cache in bytes, multiply **DB\_BLOCK\_BUFFERS** by **DB\_BLOCK\_SIZE**.
- The server processes read data from the data files into the buffer cache. To improve performance, the server process sometimes reads multiple blocks in a single read.
- The **DBWn** process writes data from the buffer cache into the data files. To improve performance, **DBWn** writes multiple blocks in a single write.
- Each buffer in the cache holds a single database block.
- At any given time, the buffer cache may hold multiple copies of a single database block. Only one current copy of the block exists, but server processes may need to construct read-consistent copies, using rollback information, to satisfy queries.

### **Buffer Cache Characteristics (continued)**

- The blocks in the buffer cache are managed using two lists:
  - The least recently used (LRU) list is used to keep the most recently accessed blocks in memory. The blocks on the list are organized from the most recently used (MRU) to the least recently used.
  - The dirty list points to blocks in the buffer cache that have been modified but not written to disk.
- Blocks in the buffer cache can be in one of three states:
  - Free buffers are blocks that have the same image on disk and in memory. These blocks are available for reuse.
  - Dirty blocks are blocks with a different image in memory than the image on disk. These blocks must be written to disk before they can be reused.
  - Pinned buffers are memory blocks that are currently being accessed.

Server processes use the blocks in the buffer cache, but the DBW<sub>n</sub> process makes blocks in the cache available by writing changed blocks back to the data files.

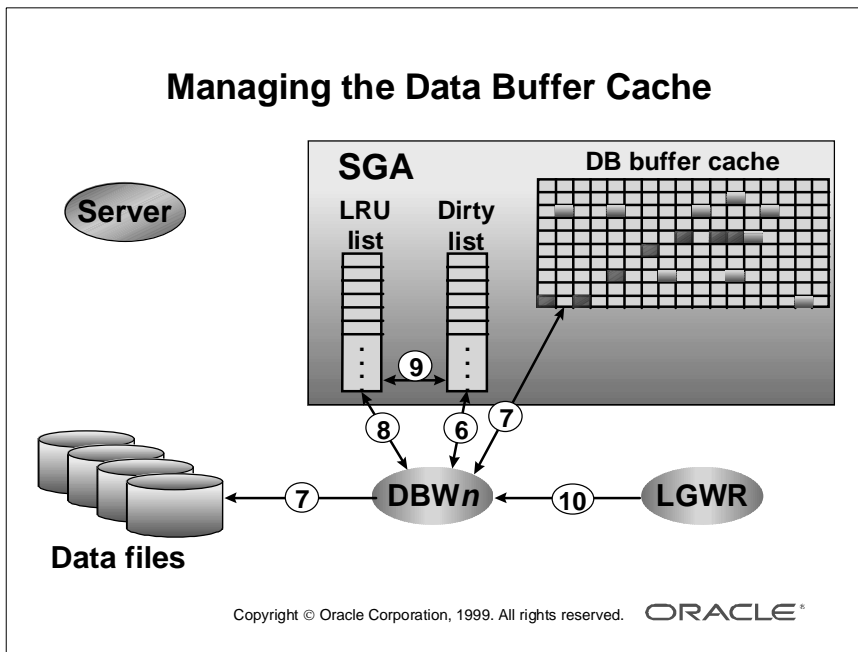
## Managing the Buffer Cache



### The Server Process and the DB Buffer Cache

When a server needs a block, it follows these steps to read the block:

- First, the server looks for the block in the buffer cache using a hash function (step 1). If the block is there, it is moved to the MRU end of the LRU list. This is a logical read, because no actual I/O took place.
- If the block is not found in the buffer cache, the server process reads the block from the data file. First, the server searches the LRU list for a free block (step 2). Then:
  - While searching the LRU list, the server moves dirty blocks to the dirty list (step 3).
  - If the dirty list exceeds its size threshold, the server signals DBWn to flush dirty blocks from the data buffer cache (step 4).
  - If the server cannot find a free block within a search threshold, it signals DBWn to flush (step 4).
  - After a free block is found, the server reads the block from the data file into the free block in the database buffer cache (step 5). The server moves the block from the LRU end to the MRU end of the LRU List.
- Finally, if the block is not consistent, the server rebuilds an earlier version of the block from the current block and rollback segments.



### The DBWn Process and the DB Buffer Cache

DBWn manages the buffer cache by writing dirty blocks to the data files to ensure that there are free blocks for servers. DBWn responds to different events in the Oracle instance.

- **Dirty List Exceeds its Size Threshold:** A server process finds that the dirty list has exceeded its size threshold, so it signals DBWn to flush (step 4). DBWn writes out the blocks on the dirty list (steps 6 and 7).
- **Search Threshold Exceeded:** A server process that cannot find a free block on the LRU list within the search threshold signals DBWn to flush dirty blocks (step 4). DBWn writes out dirty blocks directly from the LRU list (steps 7 and 8).
- **Three-Second Time-Out:** Every 3 seconds, DBWn wakes up to check the dirty list for blocks to write. DBWn moves dirty blocks from the LRU list to the dirty list (step 9), so that it has enough blocks for a full write buffer. Then DBWn writes blocks on the dirty list from the buffer cache to the data files (steps 6 and 7). If there is no update activity for extended periods of time, DBWn will eventually write out all of the dirty blocks during the 3-second time-outs.
- **LGWR Signals a Checkpoint:** When LGWR signals that a checkpoint has occurred (step 10), DBWn copies dirty blocks from the LRU to the dirty list (step 8) and writes out all of the blocks on the dirty list (steps 6 and 7).

**The DBWn Process and the DB Buffer Cache (continued)**

- **Alter Tablespace Offline Temporary or Alter Tablespace Begin Backup:** When a tablespace is altered using the option Offline Temporary or it is set in backup mode, DBWn copies the dirty blocks for that tablespace from the LRU to the dirty list (step 8) and writes out the blocks on the dirty list (steps 6 and 7).
- **Drop Object:** When an object is dropped, DBWn first flushes the objects dirty blocks to disk (steps 8 and 7).
- **Pinged Block:** In Parallel Server, the LCKn processes also signal DBWn to write blocks. This happens when a block is pinged for another instance in the parallel server database.
- **Clean Shutdown:** The database is shutdown using Normal, Immediate, or Transactional option.

## Tuning Goals and Techniques

### Tuning Goals and Techniques

- **Tuning goals:**
  - Servers find data in memory
  - 90% hit ratio for OLTP
- **Tuning techniques:**
  - Increase buffer cache size
  - Use multiple buffer pools
  - Cache tables
  - Bypass the buffer cache for sorting and parallel reads

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

### Tuning Goals

Because physical I/O takes significant time and increases CPU demand, Oracle server performance can be improved when the servers find most of the blocks that they need in memory. The statistic that measures the performance of the database buffer cache is the cache hit ratio. This statistic is the ratio of the number of blocks found in memory to the number of blocks accessed. When the database buffer cache is too small, the system is slower because it is performing too many I/Os.

### Tuning Techniques

The DBA monitors the buffer cache by calculating the cache hit ratio from statistics collected by the Oracle server.

To improve the cache hit ratio, the DBA:

- Increases the parameter `DB_BLOCK_BUFFERS` to add blocks to the buffer cache
- Uses multiple buffer pools to separate blocks by access characteristics
- Caches table in memory
- Bypass the buffer cache for sorting and parallel reads if possible.



**Tuning Techniques (continued)**

The DBA should determine the change in the hit ratio as buffers are added or removed. As a general rule, increase the `DB_BLOCK_BUFFERS` parameter when:

- The cache hit ratio is less than 90%.
- There is adequate memory for other processes, as measured by the amount of page faults.
- The previous increase of `DB_BLOCK_BUFFERS` was effective. Increasing the size of the data buffer cache does not always improve performance, because the characteristics of the application may prevent further improvement of the ratio. For example, in large data warehouse or decision support systems, which routinely use many scans of large tables, most of the data is read from disk. For these systems, tuning the buffer cache is less important and tuning I/O is vital.

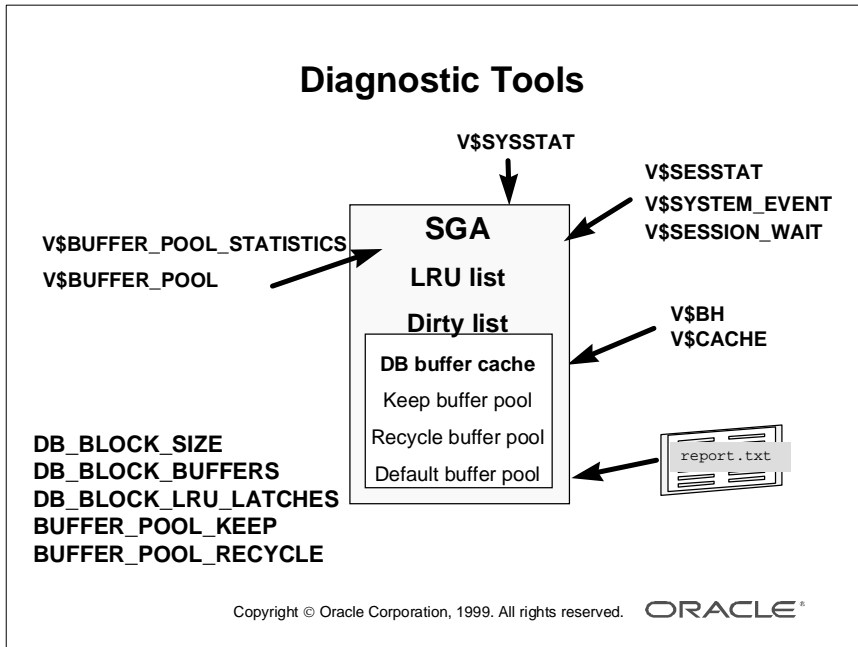
If data access characteristics are causing the low cache hit ratio, the DBA might be able to improve the ratio by defining multiple pools or caching tables.

**Technical Note**

You need to consider the impact of operating system caching. For example, the Oracle server may show a high rate of physical I/O that does not appear at the operating system level. This could mean that Oracle blocks, aged out of the buffer cache, are kept in the operating system cache and can be accessed very quickly. However, as a general rule it is best to bypass the operating system cache, because:

- More memory may be required, because there are duplicate Oracle blocks in memory: one in the operating system cache and one in the database buffer cache.
- The Oracle blocks kept in the operating system cache use memory that could be used by other operating system processes.
- There is the CPU overhead of copying blocks from the operating system cache to the database buffer cache.

## Diagnostic Tools for Tuning the Buffer Cache



### Description of the Views

- V\$SYSSTAT and V\$SESSTAT: Contain the statistics used to calculate the cache hit ratio.

```
SQL> SELECT name, value
2 FROM v$sysstat
3 WHERE name in ('db block gets', 'consistent gets',
4 'physical reads');
```

NAME	VALUE
db block gets	2241341
consistent gets	130303
physical reads	103434

- V\$BUFFER\_POOL: Describes multiple buffer pools
- V\$BH: Describes blocks held in the buffer cache

## Cache Hit Ratio

### Measuring the Cache Hit Ratio

From V\$SYSSTAT:

```
SQL> SELECT 1 - (phy.value / (cur.value + con.value))
2          "CACHE HIT RATIO"
3 FROM   v$sysstat cur, v$sysstat con, v$sysstat phy
6 WHERE  cur.name = 'db block gets'
7 AND    con.name = 'consistent gets'
8 AND    phy.name = 'physical reads';
```

```
CACHE HIT RATIO
-----
.908160337
```

From report.txt:

Statistic	Total	Per Transact	Per Logon	Per Second
consistent gets	121754	1117.07	6764.11	50.73
db block gets	20628	189.25	1146	8.6
physical reads	104695	960.5	5816.94	43.62

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

### Measuring the Cache Hit Ratio

Oracle collects statistics on data access and stores them in the dynamic performance table V\$SYSSTAT. You measure the cache hit ratio using three system statistics:

- `db block gets`: Accesses to the current image of a block
- `consistent gets`: Accesses to a read-consistent image of a block
- `physical reads`: Number of blocks read from disk

Calculate the hit ratio for the buffer cache with this formula:

Hit Ratio =  $1 - (\text{physical reads} / (\text{db block gets} + \text{consistent gets}))$

Adding `db_blocks_gets` and `consistent_gets` gives the total number of requests for data. This value includes requests satisfied by access to buffers in memory and requests that cause a physical I/O. You can multiply the ratio by 100 to convert to a percentage.

Because these statistics are collected only from instance startup time, query them during normal working loads but not immediately after startup. Because the buffer cache is empty when the instance starts, there are more physical reads after startup.

## Guidelines for Using the Cache Hit Ratio

### Guidelines for Using the Cache Hit Ratio

Hit ratio peaks because of data access methods:

- Full table scans
- Data or application design
- Large table with random access
- Uneven distribution of cache hits

Guidelines for increasing the cache size:

- Cache hit ratio is less than 0.9
- No undue page faulting
- Previous increase was effective

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

### **Increasing the Cache Hit Ratio by Reducing Buffer Cache Misses**

- 1 If your hit ratio is low, or less than 60% or 70%, then you might want to increase the number of buffers in the cache to improve performance.
- 2 To make the buffer cache larger, increase the value of the initialization parameter `DB_BLOCK_BUFFERS`.
- 3 The DBA recollects new statistics that estimate the performance gain that would result from increasing the size of the buffer cache. With these statistics, you can estimate how many buffers to add to your cache if necessary.

### **Removing Unnecessary Buffers When the Cache Hit Ratio Is High**

- 1 If your hit ratio is high, your cache is probably large enough to hold your most frequently accessed data. In this case, you may be able to reduce the cache size and still maintain good performance.
- 2 To make the buffer cache smaller, reduce the value of the initialization parameter `DB_BLOCK_BUFFERS`. The minimum value for this parameter is 4. You can use any leftover memory for other Oracle memory structures.
- 3 The DBA recollects new statistics to calculate buffer cache performance based on a smaller cache size. Examining these statistics can help you determine how small you can afford to make your buffer cache without adversely affecting performance.

### **Evaluating the Cache Hit Ratio**

- Do not continue increasing `DB_BLOCK_BUFFERS` when the last increase made no significant difference in the cache hit ratio. This might be caused by the way that you are accessing your data, or there might be other operations that do not even use the buffer pool. For example, the Oracle server bypasses the buffer cache for sorting and parallel reads.
- Also, when looking at the cache hit ratio, bear in mind that blocks encountered during a full table scan are not placed at the top of the LRU list; therefore, repeated scanning does not cause the blocks to be cached.
- The solution lies at the design or implementation level, in that repeated scanning of the same large table is rarely the most efficient solution to the problem. It might be better to perform all of the processing in a single pass or add appropriate indexes to the table.
- In large databases running an online transaction processing (OLTP) application, in any given unit of time, most rows are accessed either one or zero times. On this basis there is little point in keeping the row (or the block that contains it) in memory for very long after its use.

### **Evaluating the Cache Hit Ratio (continued)**

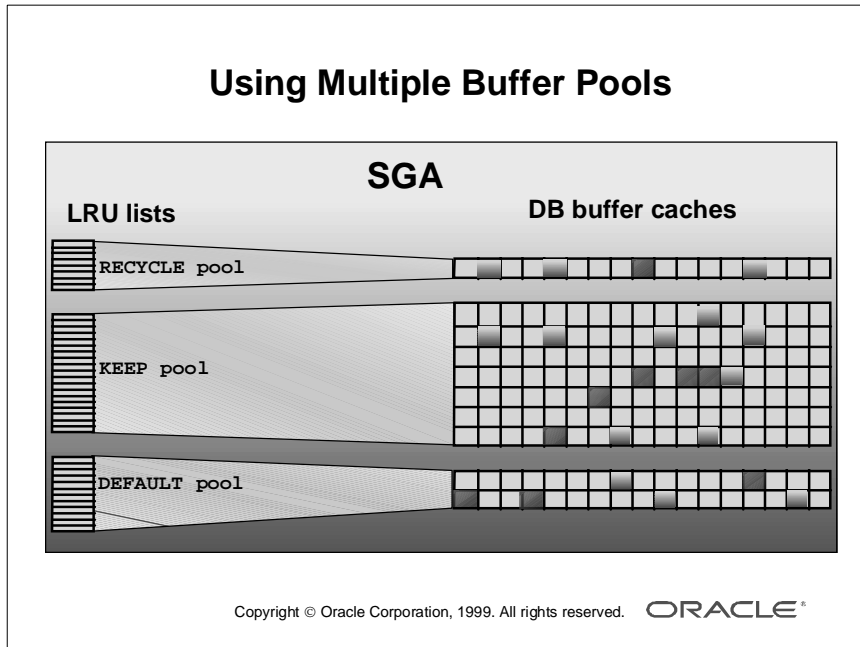
- Finally, the relationship between cache hit ratio and number of buffers is far from a smooth distribution. When tuning the buffer pool, avoid the use of additional buffers that contribute little or nothing to the cache hit ratio.

### **Increasing DB\_BLOCK\_BUFFERS**

As a general rule, increase DB\_BLOCK\_BUFFERS under the following conditions:

- The cache hit ratio is less than 90%.
- There is adequate memory for other processes, as measured by the amount of page faults.
- The previous increase of DB\_BLOCK\_BUFFERS was effective.

## Using Multiple Buffer Pools



### Multiple Buffer Pools

The DBA may be able to improve the performance of the database buffer cache by creating multiple buffer pools. Objects are assigned to a buffer pool depending on how the objects are accessed. Oracle8 has three buffer pools:

- **KEEP:** This pool is used to retain objects in memory that are likely to be reused. Keeping these objects in memory reduces I/O operations.
- **RECYCLE:** This pool is used to eliminate blocks from memory that have little chance of being reused. Flushing these blocks from memory enables you to allocate the space that would be used by their cache buffers to other objects.
- **DEFAULT:** The pool always exists. It is equivalent to the single buffer cache.

You can define each buffer pool using the `BUFFER_POOL_name` initialization parameter. You specify two attributes for each buffer pool:

- The number of buffers in the buffer pool
- The number of LRU latches allocated to the buffer pool

**Note:** This feature is new in Oracle8.

## Defining Multiple Buffer Pools

### Defining Multiple Buffer Pools

```
...  
DB_BLOCK_BUFFERS = 20000  
DB_BLOCK_LRU_LATCHES = 6  
BUFFER_POOL_KEEP=(BUFFERS:14000,LRU_LATCHES:1)  
BUFFER_POOL_RECYCLE=(BUFFERS:2000,LRU_LATCHES:3)  
...
```

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



## Defining Multiple Buffer Pools

- Pool blocks are taken from **DB\_BLOCK\_BUFFERS**
- Latches are taken from **DB\_BLOCK\_LRU\_LATCHES**
- There are at least 50 blocks per latch
- DBA can define one, two, or three pools

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

## Defining Multiple Buffer Pools

You can use the following buffer pools to define multiple buffer pools.

- **DB\_BLOCK\_BUFFERS**: Defines the number of buffers for the instance (Each individual buffer pool is created from this total amount. The remainder is allocated to the default buffer pool.)
- **DB\_BLOCK\_LRU\_LATCHES**: The number of LRU latches for the entire database instance (Each buffer pool defined takes a latch from this total in a fashion similar to **DB\_BLOCK\_BUFFERS**.)
- **BUFFER\_POOL\_KEEP**: Used to define the buffer pool for retaining blocks (The first parameter is the number of blocks and the second is the number of latches.)
- **BUFFER\_POOL\_RECYCLE**: Used to define the buffer pool for discarding blocks (The first parameter is the number of blocks and the second is the number of latches.)

You can subtract size of each pool from the total (that is, the value of the **DB\_BLOCK\_BUFFERS** parameter) to get the size of **DEFAULT** pool. Therefore, the aggregate number of buffers in all of the buffer pools cannot exceed the value **DB\_BLOCK\_BUFFERS**.

Similarly, the number of LRU latches allocated to each buffer pool is taken from the total number allocated to the instance using the **DB\_BLOCK\_LRU\_LATCHES** parameter. If either constraint is violated, then an error occurs when the database is mounted.

### **Defining Multiple Buffer Pools (continued)**

The minimum number of buffers that must be allocated to each buffer pool is 50 times the number of LRU latches. For example, if a buffer pool has 3 LRU latches, it must have at least 150 buffers. There is no requirement that any buffer pool be defined for another buffer pool to be used.

## Enabling Multiple Buffer Pools

### Enabling Multiple Buffer Pools

```
CREATE INDEX cust_idx ...  
  STORAGE (BUFFER_POOL KEEP ...);  
  
ALTER TABLE customer  
  STORAGE (BUFFER_POOL RECYCLE);  
  
ALTER INDEX cust_name_idx  
  REBUILD  
  STORAGE (BUFFER_POOL KEEP);
```

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

### The BUFFER\_POOL Clause

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

The syntax is BUFFER\_POOL { KEEP | RECYCLE | DEFAULT }.

### Examples

- BUFFER\_POOL KEEP
- BUFFER\_POOL RECYCLE

When the default buffer pool of an object is changed using the ALTER statement, all buffers that currently contain blocks of the altered segment remain in the buffer pool that they were in before the ALTER statement. Newly loaded blocks and any blocks that have aged out and are reloaded will go into the new buffer pool.

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

## Sizing Buffer Pools

### Keep Buffer Pool Guidelines

- **Tuning goal: Keep blocks in memory**
- **Size: Hold all or nearly all blocks**
- **Tool: ANALYZE ... ESTIMATE STATISTICS**

```
SQL> ANALYZE TABLE codes ESTIMATE STATISTICS;

Table analyzed.

SQL> SELECT  table_name, blocks
      2     FROM    dba_tables
      3     WHERE   owner = 'HR' AND table_name = 'CODES';

TABLE_NAME      BLOCKS
-----
CODES           14
```

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

### Tuning Goal

The goal of the keep buffer pool is to retain objects in memory, thus avoiding I/O operations. The size of the keep buffer pool is computed by adding together the sizes of all objects dedicated to this pool.

### Sizing

Use ANALYZE ... ESTIMATE STATISTICS to obtain the size of each object. The high water mark is always exact, even if you estimate statistics. Sum the BLOCKS column from DBA\_TABLES, DBA\_INDEXES, and DBA\_CLUSTERS to determine the total blocks required.

Depending on the data access characteristics and the amount of available memory, you may not want to keep all of the blocks from all of these objects in the buffer pool. Often you can decrease the size of your keep buffer pool by quite a bit and still maintain a high hit ratio. Those blocks can be allocated to other buffer pools.

The DBA must monitor objects in the KEEP pool that grow in size. An object may no longer fit in the keep buffer pool, and then you will begin to lose blocks out of the cache.

## Recycle Buffer Pool Guidelines

### Recycle Buffer Pool Guidelines

- **Tuning goal:** Eliminate blocks from memory when transactions have completed
- **Size:** Hold only active blocks
- **Tool:** V\$CACHE

```
SQL> @catparr
...

SQL> SELECT  owner#, name, count(*) blocks
      2      FROM    v$cache
      3      GROUP BY owner#, name;

OWNER# NAME          BLOCKS
-----
5 CUSTOMER          147
...
```

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

### Tuning Goal

The goal of the recycle buffer pool is to eliminate blocks from memory as soon as they are no longer needed. Be careful, however, not to discard blocks from memory too quickly. If the buffer pool is too small it is possible for a block to age out of the cache before the transaction or SQL statement has completed execution. For example, an application may select a value from a table, use the value to process some data, and then update the row selected. If the block is removed from the cache after the SELECT statement, it must be read from disk again to perform the update. The block needs to be retained for the duration of the transaction.

### Sizing

You can size the recycle pool by using the physical reads statistic from a tracing tool or by totaling the buffer cache blocks used by the object.

### Using V\$CACHE to Find Blocks in the Buffer Pool

The DBA can also monitor the number of buffer pool blocks by object using the V\$CACHE view. V\$CACHE is created by the script `catparr.sql` V\$CACHE:

- Is really intended for use with Oracle Parallel Server (OPS)
- Creates a number of other views that are useful only for OPS

### Using V\$CACHE to Find Blocks in the Buffer Pool (continued)

- Maps extents in the data files to database objects
- Needs to be rerun after new objects have been created

To determine the number of blocks required for objects in the RECYCLE pool:

- Tune the buffer cache with the recycle pool disabled.
- Run `catparr.sql` to set up and populate V\$CACHE.
- During peak running times, calculate how many blocks are used by each object with the following query:

```
SQL> SELECT owner#, name, count(*) blocks
2    FROM v$cache
3    GROUP BY owner#, name;
```

Sum the blocks for all objects that will be used in the recycle buffer pool and divide by 4 to obtain the recycle pool size. You divide by 4 because it is assumed that one-fourth of the blocks targeted for the recycle pool are active. The other three-fourths are waiting to be aged out of the cache.

## Recycle Buffer Pool Guidelines

### Tool: V\$SESS\_IO

```
SQL> SELECT io.block_gets,  
2         io.consistent_gets,  
3         io.physical_reads  
4 FROM   v$sess_io      io,  
5        v$session      s  
6 WHERE  s.audsid = USERENV('SESSIONID')  
7* AND   io.sid = s.sid ;
```

BLOCK_GETS	CONSISTENT_GETS	PHYSICAL_READS
2187	23271	1344

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

### Tracing Physical Reads

By executing statements with a SQL statement tuning tool such as Oracle Trace Manager, SQL\*Plus Autotrace, or SQL trace with TKPROF, you can obtain a listing of the total number of data blocks physically read from disk. Also, the dynamic performance table V\$SESS\_IO provides I/O statistics by session. Because you always expect to physically read blocks from the objects in this cache, the number of physical reads for the SQL statement is greater than or equal to the number of blocks read from the object.

## Calculating the Buffer Pool Hit Ratio

### Calculating the Hit Ratio for Multiple Pools

```
SQL> @catperf
...

SQL> SELECT name,
           1 - (physical_reads / (db_block_gets +
                                consistent_gets)) "HIT_RATIO"
2  FROM   sys.v$dbuffer_pool_statistics
3  WHERE  db_block_gets + consistent_gets > 0;
```

NAME	HIT_RATIO
KEEP	.983520845
RECYCLE	.503866235
DEFAULT	.790350047

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



**V\$BUFFER\_POOL\_STATISTICS**

This view displays statistics (physical writes, consistent gets, free buffer waits) against the multiple buffer caches (if allocated):

Column Name	Description
NAME	Name of the buffer pool (KEEP, RECYCLE, DEFAULT)
SET_MSIZE	Maximum buffer size allowed
CNUM_REPL	Current number of buffers in replacement
CNUM_WRITE	Current number of buffers in write list
CNUM_SET	Current total number of buffers in this pool
BUF_GOT	Number of buffers that foreground process got for this pool
SUM_WRITE	Number of buffers written by DBWn in this pool
SUM_SCAN	Number of buffers scanned by DBWn in this pool
FREE_BUFFER_WAIT	Free buffer waits for this pool
WRITE_COMPLETE_WAIT	Write complete waits for this pool
BUFFER_BUSY_WAIT	Buffer busy waits for this pool
FREE_BUFFER_INSPECTED	Free buffers inspected for this pool
DIRTY_BUFFERS_INSPECTED	Dirty buffers inspected for this pool
DB_BLOCK_CHANGE	DB block changes for this pool
DB_BLOCK_GETS	DB block gets for this pool
CONSISTENT_GETS	Consistent gets for this pool
PHYSICAL_READS	Physical reads for this pool
PHYSICAL_WRITES	Physical writes for this pool

## Segments for the Keep and Recycle Buffer Pools

### Identifying Candidate Pool Segments

- **KEEP Pool**
  - **Blocks repeatedly accessed**
  - **Segment size is less than 10% of default buffer pool size**
- **RECYCLE Pool**
  - **Blocks not reused outside of transaction**
  - **Segment size is more than twice the default buffer pool size**

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

### Trade-Offs

Remember that each object kept in memory results in a trade-off. Although it is beneficial to keep frequently accessed blocks in the cache, retaining infrequently used blocks results in less available space for other, more active, blocks.

## Dictionary Views with Buffer Pools

### Dictionary Views with Buffer Pools

```
SQL> SELECT *
      2 FROM v$buffer_pool
      3 WHERE id <> 0;
```

ID	NAME	LO_SETID	HI_SETID	SET_COUNT	BUFFERS	LO_BNUM	HI_BNUM
1	KEEP	3	3	1	14000	0	13999
2	RECYCLE	4	6	3	2000	14000	15999
3	DEFAULT	1	2	2	4000	16000	19999

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

### Dictionary Views

These dictionary views have a BUFFER\_POOL column that indicates the default buffer pool for the given object:

- USER\_, DBA\_SEGMENTS
- USER\_, ALL\_, DBA\_CLUSTERS
- USER\_, ALL\_, DBA\_INDEXES
- USER\_, ALL\_, DBA\_TABLES
- USER\_, ALL\_, DBA\_OBJECT\_TABLES
- USER\_, ALL\_, DBA\_ALL\_TABLES

The V\$BUFFER\_POOL view describes the buffer pools allocated.

The columns from V\$BUFFER\_POOL show:

- The number and range of LRU latches allocated to the buffer (sets)
- The number and range of blocks allocated to the buffer

## Other Performance Indicators

### Other Performance Indicators

**From V\$SYSSTAT:**

```
SQL> SELECT name, value
2 FROM v$sysstat
3 WHERE name = 'free buffer inspected';
```

NAME	VALUE
-----	-----
free buffer inspected	183

**From V\$SYSTEM\_EVENT:**

```
SQL> SELECT event, total_waits
2 FROM v$system_event
3 WHERE event in
4 ('free buffer waits', 'buffer busy waits');
```

EVENT	TOTAL_WAITS
-----	-----
free buffer waits	337
buffer busy waits	3466

Copyright © Oracle Corporation, 1999. All rights reserved. **ORACLE**<sup>®</sup>

### Other Performance Indicators

The buffer cache hit ratio is by far the most useful measure of buffer cache performance. However, there are a few other indicators.

#### Wait Statistics

You should consider increasing the buffer cache size if there are high or increasing values for the system statistic Free Buffer Inspected. This statistic is the number of buffers skipped to find a free buffer. Buffers are skipped because they are dirty or pinned.

#### Wait Events

You can find out whether there have been waits for buffers from the V\$SYSTEM\_EVENT or V\$SESSION\_WAIT view.

- One event to look for is Buffer Busy Waits, which means that a process has been waiting for a buffer to become available.
- A second event to examine is Free Buffer Waits, which occurs after a server cannot find a free buffer or when the dirty queue is full.

There are no waits if the event does not occur.

#### Technical Note

These statistics and events could also indicate that the DBWn process needs tuning.

## Caching Tables

### Caching Tables

Enable caching during full table scans by:

- Creating the table with the **CACHE** clause
- Altering the table with the **CACHE** clause
- Using the **CACHE** hint in a query

**Guidelines: Do not overcrowd the cache.**

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

### Caching Tables

When the server retrieves blocks using a full table scan, the blocks go to the least recently used end of the LRU list. The blocks will be used the next time a free block is needed, so they are not available for other processes. You can choose to cache whole tables at the most recently used (MRU) end of the list.

You alter this behavior if you do the following:

- Create a table using the **CACHE** clause
- Alter a table using the **CACHE** clause
- Code the **CACHE** hint clause into a query

If you use one of these methods, the Oracle server places the table blocks at the most recently used end of the LRU list. Use the **CACHE** clause when you create small lookup tables used by many users. You may overcrowd the buffer cache if you have too many cached tables.

## LRU Latches

### LRU Latches

- LRU latches regulate the least recently used (LRU) lists used by the buffer cache.
- By default, the Oracle server sets the number of LRU latches to one-half the number of CPUs, with a minimum of one.
- Each latch controls a minimum of 50 buffers.

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

### Using LRU Latches

Space must be available in the buffer cache when new blocks are read into the buffer cache. Determining which blocks to flush from the buffer cache to make room for new blocks is regulated by a least recently used (LRU) list. Blocks that have been not been used recently are candidates for being flushed from the cache.

## LRU Latch Tuning Goals

### LRU Latch Tuning Goals

- **Ensure there are a sufficient number of LRU latches for the data buffer cache, so that contention between server processes is minimized.**
- **Balance the number of latches with the number of CPUs.**
- **Set one DBW $n$  process for each latch.**

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

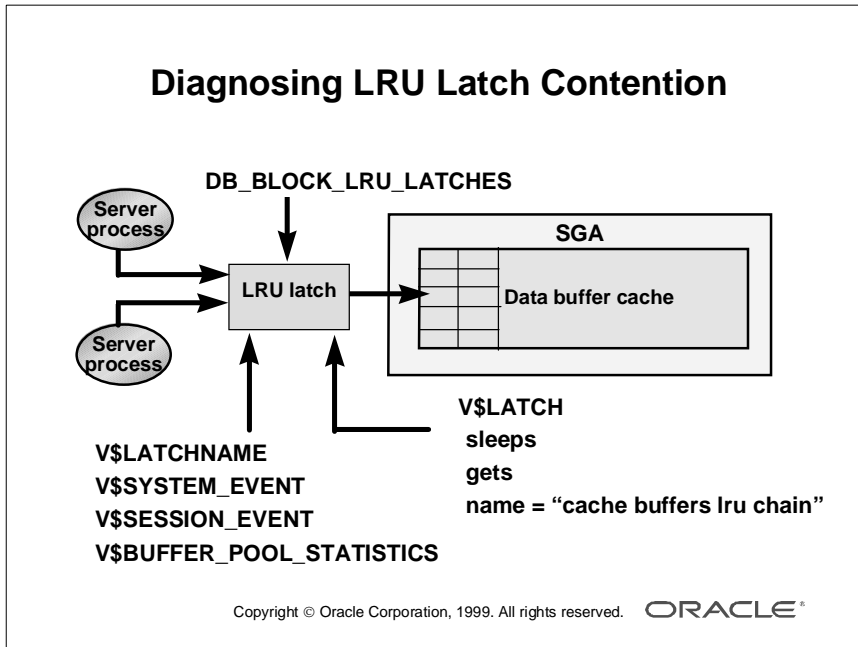
### Tuning Goals

The tuning goals for the LRU latch is the same as that for all other latches:

- Minimize contention among server processes that are requesting the latch.
- Balance the number of latches with the number of CPUs.

Adding LRU latches in a single-CPU system may not be beneficial, because the limitation still exists with the number of processes the CPU can manage at any given time. Therefore, the number of LRU latches should be balanced with the number of available CPUs.

## Diagnosing LRU Latch Contention



### Dynamic Performance Views

The **V\$LATCH** and **V\$LATCHNAME** views provide information regarding the latches. This information includes the values used to derive the hit percentage for the LRU latch.

In this case, the latch name is “cache buffers lru chain.”

### Initialization Parameter

The **DB\_BLOCK\_LRU\_LATCHES** parameter determines the number of LRU latches to allocated per instance. Typically, having more LRU latches than CPUs does not benefit performance unless you are using multiple buffer pools (keep, recycle, default).

### Query Latch Statistics

When querying LRU latch statistics, you are looking for the number of times a process tries to acquire the latch, “gets” it, and then must “sleep,” or wait for the latch. Ideally, a process should be able to acquire a latch when needed. Use the following query to determine the get percentage for the LRU latch:

```
SELECT name, sleeps/gets "LRU Hit%"
FROM v$latch
WHERE name = 'cache buffers lru chain';
```



## Resolving LRU Latch Contention

### Resolving LRU Latch Contention

If the hit percentage for the LRU latch is less than 99%:

- Increase the number of LRU latches by setting the parameter `DB_BLOCK_LRU_LATCHES`
- The maximum number of latches is the lower of:
  - Number of CPUs x 2 x 3
  - Number of buffers/50

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

### LRU Latch Limits

If the hit percentage on the LRU latch is less than 99%, consider increasing the number of latches by modifying the `DB_BLOCK_LRU_LATCHES` parameter.

Two methods for determining the number of LRU latches are to multiply the number of CPUs by six or to take the total number of block buffers and divide by 50. The second method uses the number 50 because this is the minimum number of buffers that will be allocated to each latch.

Remember that CPU use is also a factor when setting LRU latches.

## Free Lists

### Free Lists

- A free list for an object maintains a list of blocks that are available for inserts.
- The number of free lists for an object cannot be set dynamically.
- Single-CPU systems do not benefit greatly from multiple free lists.
- The tuning goal is to ensure that an object has sufficient free lists to minimize contention.

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

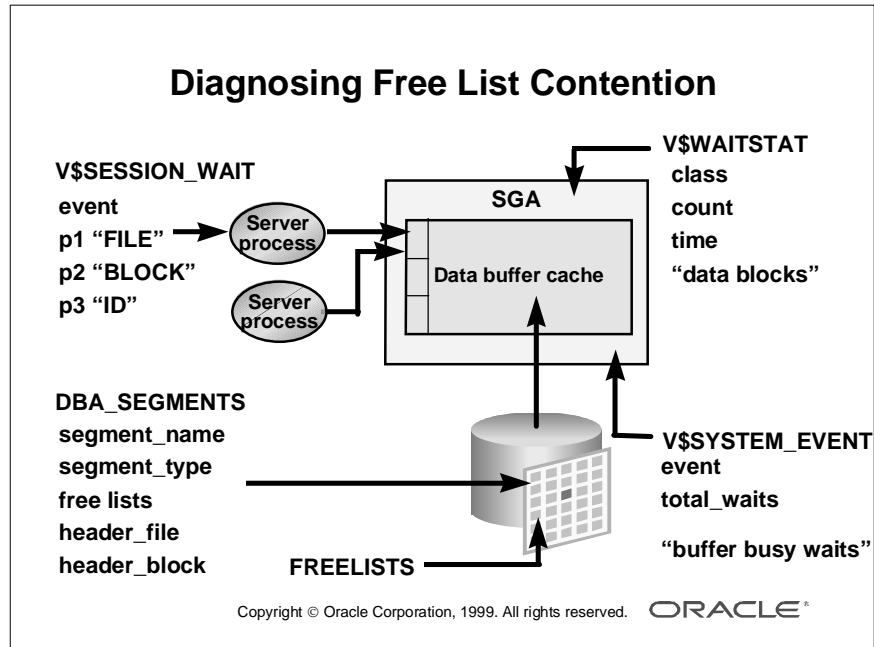
### Using Free Lists

When an insert operation on an object occurs, the free list is used to determine which blocks are available for inserts. Many server processes can contend for the same free list if many inserts are occurring. This results in free list contention as server processes incur waits.

Because the number of free lists for an object cannot be set dynamically, you must ensure that the object is initially created with a sufficient number of free lists. Single-CPU systems do not benefit greatly from multiple free lists, because the CPU manages one process at a time. Even in a single-CPU system, adding free lists may ensure that the processor is used more effectively, but care should still be taken when adding free lists.

The overall tuning goal for free lists is to ensure that there are a sufficient number to minimize contention among many server processes.

## Diagnosing Free List Contention



### Dynamic Performance Views

The dynamic performance views, V\$SESSION\_WAIT, V\$WAITSTAT, and V\$SYSTEM\_EVENT, are used to diagnose free list contention problems.

The data dictionary view DBA\_SEGMENTS is used to identify the objects that need to be modified to increase the number of free lists.

### Initialization Parameters

There are no initialization parameters to set for minimizing free list contention. The FREELISTS keyword is used at the segment level. This value cannot be set dynamically and requires that an object be dropped and re-created in order to change it.

**Note:** Oracle Enterprise Manager Diagnostics Pack application:

Performance Manager—>Contention—>Free List Hit%

## Diagnostic Criteria

You can query the V\$WAITSTAT and V\$SYSTEM\_EVENT views to determine if there is free list contention. If high numbers are returned, you must identify the object or objects.

- Query V\$WAITSTAT:

```
SELECT class, count, time
FROM v$waitstat
WHERE class = 'segment header';
```

- Query V\$SYSTEM\_EVENT:

```
SELECT event, total_waits
FROM v$system_event
WHERE event = 'buffer busy waits';
```

To reduce buffer busy waits on:

- Data blocks: Change PCTFREE and/or PCTUSED storage parameters; check for right-hand indexes (indexes that are inserted into at the same point by many processes); increase the storage parameter INITRANS; reduce the number of rows per block
- Segment header: Use free lists or increase the number of free lists; use free list groups (even in a single instance environment, this can make a difference)
- Free list blocks: Add more free lists (In the case of the Parallel Server, make sure that each instance has its own free list group.)

## Resolving Free List Contention

### Resolving Free List Contention

1. Query the V\$SESSION\_WAIT view.
2. Identify the object and get free lists for the segment from DBA\_SEGMENTS.
3. Re-create the object in question.

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

### Resolving Free List Contention

To resolve free list contention, you should identify the object whose free list is in contention.

#### To Identify the Object

- 1 Determine the FILE, BLOCK, and ID for which free list contention is occurring by querying V\$SESSION\_WAIT.
- 2 Identify the segment and determine the number of free lists that currently exist for the segment identified by querying DBA\_SEGMENTS:

```
SELECT s.segment_name, s.segment_type, s.freelists, w.wait_time,  
       w.seconds_in_wait, w.state  
FROM dba_segments s, v$session_wait w  
WHERE w.event = 'buffer busy waits'  
AND    w.p1      = s.header_file  
AND    w.p2      = s.header_block;
```

- 3 Re-create the object.

To increase the number of free lists for the object, you must drop the object and re-create it using a higher value for the FREELISTS keyword.

## Summary

### Summary

In this lesson, you should have learned how to:

- Get a high cache hit ratio
- Adjust `DB_BLOCK_BUFFERS` as necessary
- Separate objects into multiple buffer pools
- Use multiple buffer pools
- Cache tables
- Resolve LRU latch contention
- Avoid free list contention

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

## Quick Reference

Context	Reference
Initialization parameters	DB_BLOCK_SIZE DB_BLOCK_BUFFERS DB_BLOCK_LRU_LATCHES BUFFER_POOL_KEEP BUFFER_POOL_RECYCLE
Dynamic performance views	V\$BUFFER_POOL V\$BUFFER_POOL_STATISTICS V\$CACHE V\$SYSSTAT V\$SESSTAT V\$SESS_IO V\$SYSTEM_EVENT V\$SESSION_WAIT
Data dictionary views	USER_, DBA_SEGMENTS USER_, ALL_, DBA_CLUSTERS USER_, ALL_, DBA_INDEXES USER_, ALL_, DBA_TABLES USER_, ALL_, DBA_OBJECT_TABLES USER_, ALL_, DBA_ALL_TABLES
Commands	ALTER/CREATE INDEX/TABLE/CLUSTER
Packaged procedures and functions	None
Scripts	catparr.sql, catperf.sql
Oracle Diagnostics Pack	Performance Manager





## **Tuning the Redo Log Buffer**

## Objectives

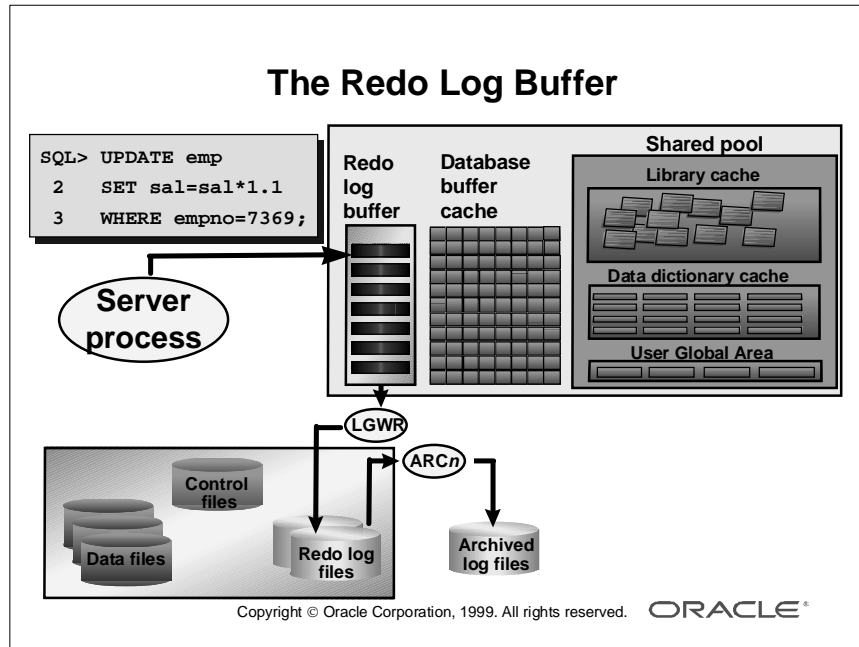
### Objectives

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

- **Determine if processes are waiting for space in the redo log buffer**
- **Size the redo log buffer appropriately**
- **Reduce redo operations**

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

## The Redo Log Buffer



### Redo Log Buffer Content

The Oracle server processes copy redo entries from the user's memory space to the redo log buffer for each data manipulation language (DML) or data definition language (DDL) statement.

The redo entries contain the information necessary to reconstruct or redo changes made to the database by INSERT, UPDATE, DELETE, CREATE, ALTER, or DROP operations. They are used for database recovery. They take up continuous, sequential space in the buffer.

### Redo Entries and LGWR

The LGWR process writes the redo log buffer to the active online redo log file (or members of the active group) on disk. It writes all redo entries that have been copied into the buffer since the last time it wrote.

The redo log buffer is a circular buffer. Thus, server processes can copy new entries over the entries in the redo log buffer that have been written to disk. LGWR normally writes fast enough to ensure that space is always available in the buffer for new entries.

## Sizing the Redo Log Buffer

### Sizing the Redo Log Buffer

- **LOG\_BUFFER** parameter
- **Default value: OS-specific, generally four times the maximum block size**

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

### Size of the Redo Log Buffer

- Larger values reduce log file I/O, particularly if transactions are long or numerous.
- Frequent COMMIT statements will clear out the buffer, leading to a smaller buffer size.

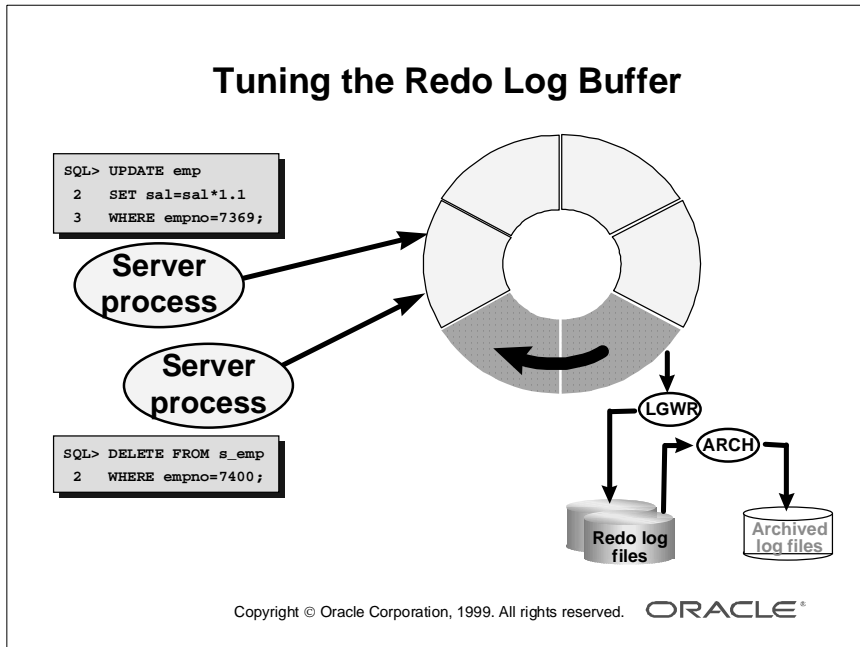
**Note:** The LOG\_BUFFER parameter can be set smaller than the platform-specific value (for example, 512 KB on UNIX systems), but if the LOG\_BUFFER value is less than the platform-specific value, then Oracle RDBMS increases LOG\_BUFFER value to the platform-specific value at instance startup. This can be changed on a platform basis, depending on what is determined to be the optimum value for the given port.

### Example

```
SQL> select 'V$PARAMETER' "Table name", name,  
to_number(value,'9999999') "Value"  
2  from v$parameter  
3  where name = 'log_buffer'  
4  UNION  
5  select 'V$SGASTAT' "Table name", name, bytes  
6  from v$sgastat  
7  where name = 'log_buffer';
```

Table name	NAME	VALUE
V\$PARAMETER	log_buffer	8192
V\$SGASTAT	log_buffer	65536

## Tuning the Redo Log Buffer



### Diagnosing Problems

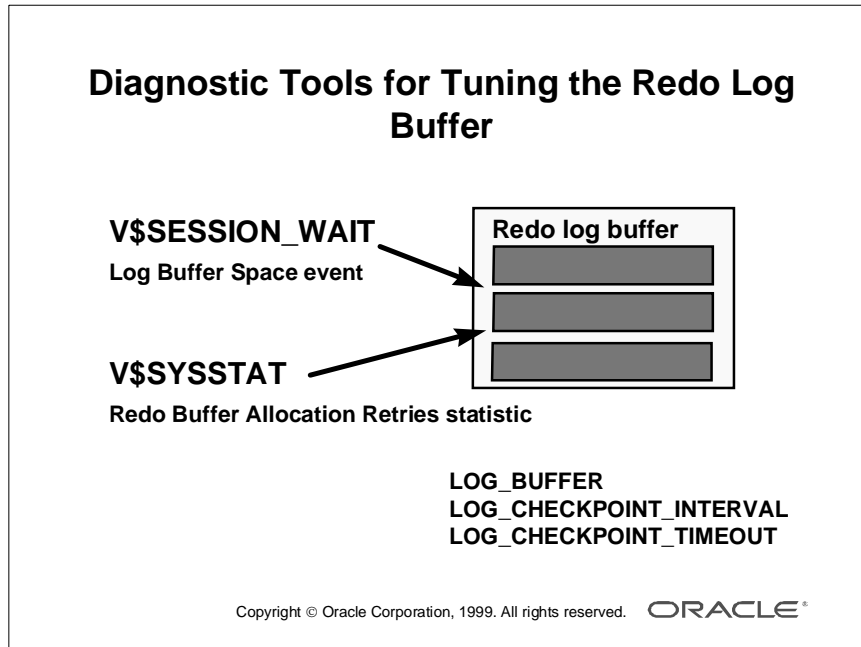
On machines with fast processors and relatively slow disks, the processors may be filling the rest of the redo log buffer in the time it takes the LGWR process to move a portion of the buffer out to disk. For this reason, a larger buffer ensures that new entries are less likely to collide with the part of the buffer still being written.

Server processes may request space from the redo log buffer to write new entries and not find any. In this case, they will have to wait for LGWR to flush the buffer to disk.

### Tuning Goal

Tuning the redo log buffer means ensuring that the space required by server processes in the redo log buffer is sufficient. However, too much space will reduce the amount of memory that can be allocated to other areas.

## Diagnostic Tools for Tuning the Redo Log Buffer



### Dynamic Views

- The V\$SESSION\_WAIT view indicates through the Log Buffer Space event if there are any waits for space in the log buffer because the session is writing data into the log buffer faster than LGWR can write it out.

```
SQL> select sid, event, seconds_in_wait, state
2   from v$session_wait
3  where event = 'log buffer space%';
```

SID	EVENT	SECONDS_IN_WAIT	STATE
5	log buffer space	110	WAITING

The SECONDS\_IN\_WAIT value of the Log Buffer Space event indicates the time spent waiting for space in the redo log buffer because the log switch does not occur. This is an indication that the buffers are being filled up faster than LGWR is writing and may also indicate disk I/O contention on the redo log files.

**Dynamic Views (continued)**

- The Redo Buffer Allocation Retries statistic in V\$SYSSTAT view reflects the number of times a user process waits for space in the redo log buffer to copy new entries over the entries that have been written to disk. LGWR normally writes fast enough to ensure that space is always available in the buffer for new entries, even when access to the redo log is heavy.

```
SQL> SELECT name, value
2   FROM   v$sysstat
3  WHERE  name = 'redo buffer allocation retries';
```

**Note:** The V\$SYSSTAT view displays another statistic, Redo Log Space Requests:

```
SQL> select name, value
2   from   v$sysstat
3  where  name='redo log space requests';
```

This statistic indicates that the active log file is full and that the Oracle server is waiting for disk space to be allocated for the redo log entries. Space is created by performing a log switch.

## Guidelines for Tuning the Redo Log Buffer

### Guidelines

**There should be no Log Buffer Space waits.**

```
SQL> SELECT sid, event, seconds_in_wait, state
2 FROM v$session_wait
3 WHERE event = 'log buffer space';
```

**The Redo Buffer Allocation Retries value should be near 0; the number should be less than 1% of Redo Entries.**

```
SQL> SELECT name, value
2 FROM v$sysstat
3 WHERE name IN ('redo buffer allocation retries',
4               'redo entries');
```

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

### SECONDS\_IN\_WAIT for Log Buffer Space Event

In the V\$SESSION\_WAIT view, if the SECONDS\_IN\_WAIT value for the Log Buffer Space event indicates that some time was spent waiting for space in the redo log buffer, consider:

- Increasing the size of the log buffer if it is small
- Moving the log files to faster disks such as striped disks

```
SQL> select sid, event, seconds_in_wait, state
2 from v$session_wait
3 where event = 'log buffer space';
```

SID	EVENT	SECONDS_IN_WAIT	STATE
5	log buffer space	110	WAITING

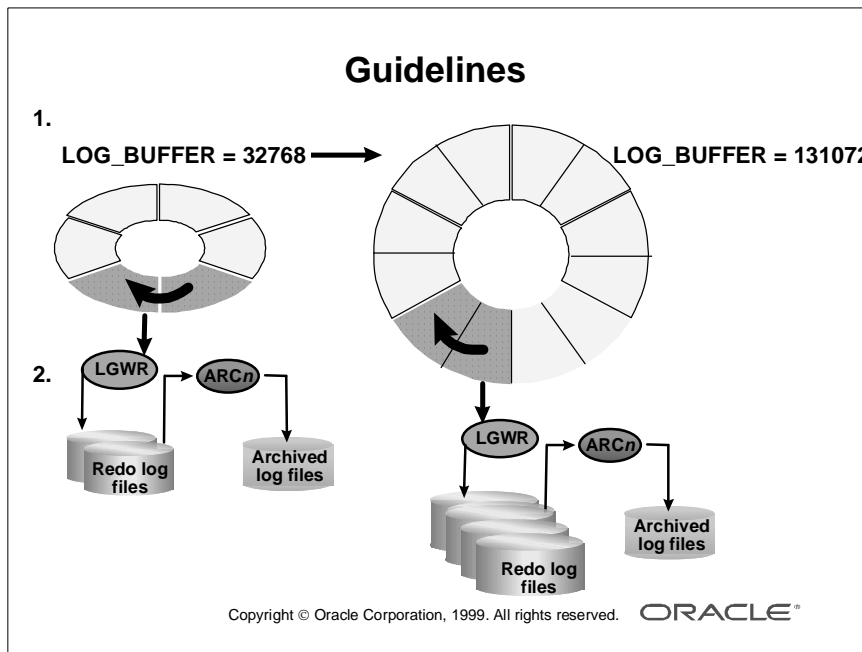
### Redo Buffer Allocation Retries Statistic Ratio

The value of Redo Buffer Allocation Retries should be near 0. The number of Redo Buffer Allocation Retries should not be greater than 1% of the Redo Entries. If this value increments consistently, processes have had to wait for space in the buffer. The wait may be caused by the log buffer being too small, by checkpointing, or by log switching.

- Increase the size of the redo log buffer, if necessary, by changing the value of the initialization parameter LOG\_BUFFER.
- Alternatively, improve the checkpointing or archiving process.

The redo log buffer is normally small compared to the Shared Global Area (SGA). A modest increase can significantly enhance throughput.





## Guidelines

Increase the value of LOG\_BUFFER until the ratio is stable.

The LOG\_BUFFER size must be a multiple of the operating system block size.

## Further Investigations

Investigate the possible reasons why the LGWR is slow in freeing buffers:

- There is disk I/O contention on the redo log files. Check that the redo log files are stored on separate, fast devices.
- In the V\$SYSTEM\_EVENT view, the Log File Switch Completion event identifies Wait events because of log switches.

```
SQL> select event, total_waits, time_waited, average_wait
2   from v$system_event
3  where event like 'log file switch completion%';
```

Increase the size of the redo log files.

### Further Investigations (continued)

- DBW<sub>n</sub> has not completed checkpointing the file when the LGWR needs the file again. LGWR has to wait.
    - In the alert.log file, check for the message “CHECKPOINT NOT COMPLETE.”
    - In the V\$SYSTEM\_EVENT view, check the number of occurrences of the event Log File Switch (Checkpoint Incomplete), which identifies the log file switch waits that are due to incomplete checkpoints.

```
SQL> select event, total_waits, time_waited, average_wait
2   from v$system_event
3  where event like 'log file switch (check%';
```
    - Check the frequency of checkpoints and set the appropriate values for the LOG\_CHECKPOINT\_INTERVAL and LOG\_CHECKPOINT\_TIMEOUT parameters.
    - Check the size and number of redo log groups.
  - The archiver cannot write to the archived redo log files or cannot achieve the archive operation fast enough. Therefore, it prevents the LGWR from writing.
    - Confirm that the archive device is not full.
    - Add redo log groups.
    - In the V\$SYSTEM\_EVENT view, check the number of occurrences of the event Log File Switch (Archiving Needed), which identifies the log file switch waits that are due to a problem in archiving the redo log.

```
SQL> select event, total_waits, time_waited, average_wait
2   from v$system_event
3  where event like 'log file switch (arch%';
```
    - Regulate archiving speed.
- The LGWR process starts a new ARC<sub>n</sub> process whenever the current number of ARC<sub>n</sub> processes is insufficient to handle the workload. If you anticipate a heavy workload for archiving, such as during bulk loading of data, you can specify multiple archiver processes with the initialization parameter LOG\_ARCHIVE\_MAX\_PROCESSES.
- DB\_BLOCK\_CHECKSUM is set to TRUE, and therefore adds performance overhead.

## Reducing Redo Operations

### Reducing Redo Operations

Fewer redo operations require fewer redo entries and thus less redo log buffer space.

Some ways of reducing the redo entries are:

- Direct Path loading without archiving
- Direct Path loading with archiving using NOLOGGING mode
- Direct Load INSERT in NOLOGGING mode
- Using NOLOGGING mode in SQL statements

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

### SQL\*Loader and NOLOGGING Mode

Conventional path loading generates redo log entries just as any DML statement. When using Direct Path loading, redo log entries are not generated if:

- The database is in NOARCHIVELOG mode
- The database is in ARCHIVELOG mode, but logging is disabled (Logging can be disabled by setting the NOLOGGING attribute for the table or by using the UNRECOVERABLE clause in the control file.)

### Direct Load Insert and NOLOGGING Mode

The NOLOGGING option:

- Applies to tables, tablespaces, and indexes
- Does not record changes to data in the redo log buffer (Some minimal logging is still carried out, for operations such as extent allocation.)
- Is not specified as an attribute at the INSERT statement level, but is instead specified when using the ALTER or CREATE command for the table, index, or tablespace
- The mode is set before the load and is reset to LOGGING once the load completes (If a media failure occurs before a backup is taken, then all tables, and indexes that have been modified, may be corrupted.)

### SQL Statements That Can Use NOLOGGING Mode

Although you can set the NOLOGGING attribute for a table, index, or tablespace, NOLOGGING mode does not apply to every operation on the object for which the attribute is set.

The following SQL statements can use NOLOGGING mode:

- CREATE TABLE ... AS SELECT
- CREATE INDEX
- ALTER INDEX ... REBUILD

The following statements are nevertheless unaffected by the NOLOGGING attribute: UPDATE, DELETE, conventional path INSERT, and various DDL statements not listed above.

**Note:** For backward compatibility, UNRECOVERABLE is still supported as an alternate keyword with the CREATE TABLE statement in Oracle8i Server release 8.1. This alternate keyword may not be supported in future releases.

## Summary

### Summary

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

- **Tune log buffer space**
- **Redo buffer allocation retries**
- **Size the redo log buffer appropriately**
- **Reduce redo operations with the NOLOGGING attribute**

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

## Quick Reference

Context	Reference
Initialization parameters	DB_BLOCK_CHECKSUM LOG_ARCHIVE_MAX_PROCESSES LOG_BUFFER LOG_CHECKPOINT_INTERVAL LOG_CHECKPOINT_TIMEOUT
Dynamic performance views	V\$SYSSTAT V\$SESSION_WAIT V\$SYSTEM_EVENT
Data dictionary views	None
Commands	None
Packaged procedures and functions	None
Scripts	None
Oracle Diagnostics Pack	None

## **Database Configuration and I/O Issues**

## Objectives

### Objectives

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

- Diagnose inappropriate use of SYSTEM, RBS, TEMP, DATA, and INDEX tablespaces
- Use locally managed tablespaces to avoid space management issues
- Detect I/O problems
- Ensure that files are distributed to minimize I/O contention and use appropriate type of devices
- Use striping where appropriate
- Tune checkpoints
- Tune DBWn process I/O

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



## Overview

### I/O Statistic for Different Oracle File Types

Process	Oracle File I/O			
	Data Files	Log	Archive	Control
CKPT	Write			Write
DBW <sub>n</sub>	Write			
LGWR		Write		
ARC <sub>n</sub>		Read	Write	Read/write
SERVER	Read			

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

### Performance Guidelines

You can improve database performance by setting up the disks in accordance with the following basic performance rules:

- Keep disk I/O to the minimum
- Spread your disk load across disk devices and controllers

The table on the slide shows the basic disk components and the I/O throughput of the background processes.

## Tablespace Usage

### Tablespace Usage

- Reserve the **SYSTEM** tablespace for data dictionary objects
- Create locally managed tablespaces to avoid space management issues
- Split tables and indexes into separate tablespaces
- Create separate rollback tablespaces
- Store very large database objects in their own tablespace
- Create one or more temporary tablespaces

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

### Guidelines for Tablespaces

Each database should have tablespaces specified for:

- Data dictionary objects
- Rollback segments
- Temporary segments
- Tables
- Indexes
- Very large objects

Most production databases have many more tablespaces than these, but the important principle is to separate data of different types and with different uses for housekeeping and backup purposes.

The **SYSTEM** tablespace contains only data dictionary objects owned by SYS. No other users should have the ability to create objects in it.

Remember that stored objects such as packages and database triggers form part of the data dictionary.

Rollback segments should use rollback segment tablespaces exclusively.

The lesson on rollback segments will cover in detail rollback segment configuration and tuning.

**Guidelines for Tablespaces (continued)**

There are concurrency problems for very heavy online transactions processing (OLTP) types of applications, because the dictionary needs to be accessed for space management operations during extent allocation. With locally managed tablespaces, there is no dictionary intervention, and therefore fewer concurrency problems.

Tables created in locally managed tablespaces can have thousands of extents with no performance implications, so reorganization of tables is unnecessary.

In a locally managed tablespace, free space does not need to be coalesced to remove “beehive” fragmentation because bitmaps track free space and allocate it effectively.

When you create users, you allocate a temporary tablespace for any disk sorts that they need. These sort areas should be separate from other database objects. If users do not have a temporary tablespace, any sort areas use the SYSTEM tablespace. The lesson on tuning sorts deals with configuration of temporary tablespaces in more detail.

Tables and indexes should be split into separate tablespaces, because indexes and tables are often inserted into and read from simultaneously.

All tables that contain LONG or LOB datatypes, for example, BLOB and CLOB, should be placed into a separate tablespace.

## Distributing Files Across Devices

### Distributing Files Across Devices

- **Separate data files and redo log files**
- **Stripe table data**
- **Reduce disk I/O**
- **Evaluate the use of raw devices**

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

### Guidelines for Distributing Files

- In general, to reduce the activity on an overloaded disk, move one or more of its heavily accessed files to a less active disk.
  - Redo log files are written sequentially by the LGWR process. Put the redo log files on a disk with no other activity or a low incidence of reads and writes. The background process LGWR can write much faster if there is no concurrent activity.
  - If users concurrently access a large table, striping across separate data files and disks can help to reduce contention. This subject will be discussed in a subsequent section.
- Try to eliminate I/O unrelated to the Oracle server on disks that contain database files. This is also helpful in optimizing access to redo log files and enables you to monitor all data file activities on such disks with the dynamic performance view V\$FILESTAT.

**Guidelines for Distributing Files(continued)**

- Make the appropriate choice between raw and block devices by testing the performance of your disks:
  - On raw devices, reads and writes are done at the character level.
  - On block devices associated with file systems, these operations are done at the block level. (Many concurrent processes may generate overhead due to head and arm movement of the disk drives.)

Knowing the types of operations that predominate in your application and the speed with which your system can process the corresponding I/Os, you can choose the disk layout that will maximize performance.

For example, with the following application test results, the UNIX file system would be a good choice:

- Random reads predominate (50% of all I/O operations), then 8 KB would be a good block size.
- Raw devices with UNIX file systems provide comparable performance of random reads at this block size.
- Furthermore, the UNIX file system in this example processes sequential reads (25% of all I/O operations) almost twice as fast as raw devices, given an 8 KB block size.

**Note:** Refer to the documentation *Oracle8i Server Tuning Release 8.1.5 A67775-01* for an overview of the tests.

## Oracle File Striping

### Oracle File Striping

#### Operating system striping:

- Use operating system striping software or RAID
- Decide on the right stripe size

#### Manual striping:

- Use the **CREATE TABLE** or **ALTER TABLE ALLOCATE** command
- Is worthwhile with parallel query usage

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

### File Striping Using the Operating System

Your operating system may allow striping, so that what appears to be a single contiguous file is actually spread among devices. For example, you can use operating system striping software, such as a logical volume manager.

Choose the right stripe size, which depends on the Oracle block size and the `DB_FILE_MULTIBLOCK_READ_COUNT` initialization parameter.

The most common type of striped file system in UNIX is RAID (redundant array of inexpensive disks). Different levels of RAID striping have varying degrees of safety checks built in.

### File Striping Manually

You can create tablespaces so that they are made up of multiple files, each one on a separate disk. You then create tables and indexes so that they are spread across these multiple files.

You can stripe by:

- Creating objects with `MINEXTENTS` greater than 1, where each extent is slightly smaller than the striped data files
- Allocating extents to a file explicitly

```
ALTER TABLE tablename  
  ALLOCATE EXTENT (DATAFILE 'filename' SIZE 10 M);
```

## Using Striping

If your operating system offers striping, typically you should take advantage of it. You should consider what the size of the stripe should be. The stripe size should usually be a multiple of the value `DB_FILE_MULTIBLOCK_READ_COUNT` (this will be discussed in a subsequent section).

For OS-striped database files with a stripe width close to (or less than) `DB_FILE_MULTIBLOCK_READ_COUNT * DB_BLOCK_SIZE`, you may get two or more physical reads for each Oracle server read, because you have to access two or more disks.

Keep in mind that striping by hand is a labor-intensive task. The Oracle server fills the extents that you have created one after another. At any given time, one extent is likely to be “hot” and the others less active. If you are using parallel query and doing many full table scans, then striping by hand may be worthwhile.

As with many other tuning issues, you can make the right choice only when you have thorough knowledge of how the data is used.

## Full Table Scans

### Tuning Full Table Scan Operations

- Investigate the need for full table scans
- Specify the initialization parameter **DB\_FILE\_MULTIBLOCK\_READ\_COUNT**:
  - To determine the number of database blocks the server reads at once
  - To influence the execution plan of the cost-based optimizer
- Monitor long-running full table scans with **V\$SESSION\_LONGOPS** view

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

### Investigating Full Table Scans

If there is high activity on one disk, it is often an untuned query causing the damage. The goal of performance tests is to increase the effectiveness with which data is accessed.

The following query provides an overview of how many full table scans are taking place:

```
SQL> SELECT name, value FROM v$sysstat
      2 WHERE name LIKE '%table scans%';
```

NAME	VALUE
table scans (short tables)	125
table scans (long tables)	30
table scans (rowid ranges)	0
table scans (cache partitions)	0
table scans (direct read)	0
table scan rows gotten	21224
table scan blocks gotten	804

7 rows selected.



### **Investigating Full Table Scans (continued)**

The values for “table scans (long tables)” and “table scans (short tables)” relate to the full table scans.

If the number of “table scans (long tables)” is high, then a large percentage of the tables accessed were not indexed lookups. Your application might need tuning, or indexes might need to be added. Ensure that the appropriate indexes are in place.

### **Tuning Full Table Scans**

The `DB_FILE_MULTIBLOCK_READ_COUNT` initialization parameter determines the maximum number of database blocks read in one I/O operation during a full table scan.

The setting of this parameter can reduce the number of I/O calls required for a full table scan, thus improving performance.

I/O is a function of the operating system, so there are operating system–specific limits imposed on the setting of this parameter. The Oracle server’s ability to read multiple blocks is limited by the operating system upper limit on the number of bytes that can be read in a single I/O call.

On most platforms (before 7.3) the maximum “read” memory chunk is 64 KB, so setting the `DB_FILE_MULTIBLOCK_READ_COUNT` parameter noted above to 64 KB per `DB_BLOCK_SIZE` will give no extra performance benefit.

Most platforms (7.3 or later) no longer place an upper limit on the `DB_FILE_MULTIBLOCK_READ_COUNT` parameter. In addition, this parameter is dynamic, so individual sessions can use `ALTER SESSION SET` command to set a larger size for batch type work.

Setting the `DB_FILE_MULTIBLOCK_READ_COUNT` parameter dictates how many I/O calls will be required to complete a table scan. For example, if `DB_FILE_MULTIBLOCK_READ_COUNT` is set to 16, and the Oracle block size equals 4 KB, then a sequential scan of a 64 KB table can be read in one pass. This improves the speed of the table scan and overall query performance. The goal of setting the `DB_FILE_MULTIBLOCK_READ_COUNT` parameter is that table scans are performed in fewer, larger I/Os. This is done by evaluating the number of blocks required to complete each table scan over time, then adjusting the parameter so that most scans can be performed in one I/O.

## Tuning Full Table Scans (continued)

The total number of I/Os actually required to perform a full table scan depends on other factors, such as the size of the table and whether parallel query is being used.

The cost-based optimizer uses all of these factors, including the `DB_FILE_MULTIBLOCK_READ_COUNT` parameter, to determine the cost of full table scans. The cost-based optimizer favors full table scans when the cost is lower than index scans.

## Monitoring Full Table Scan Operations

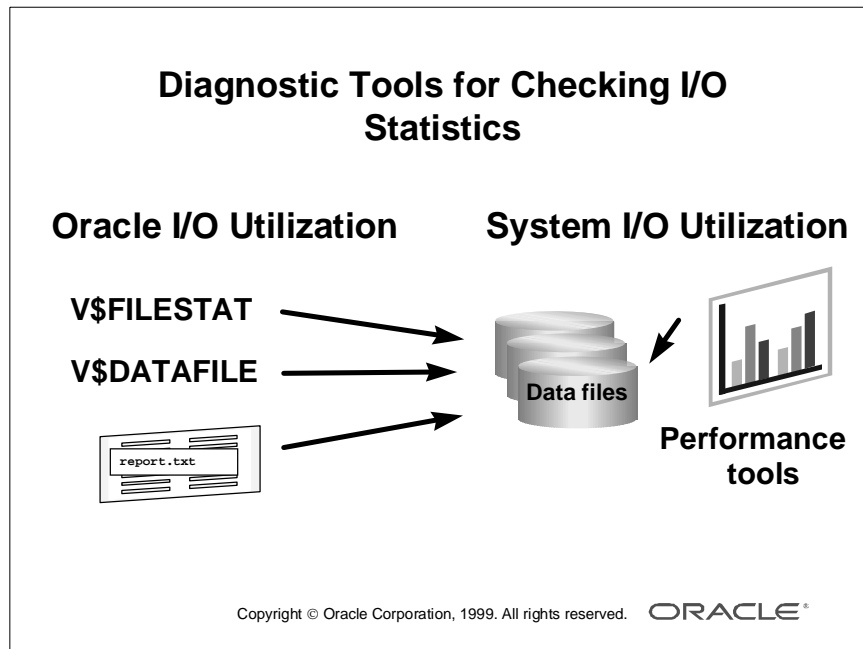
Some mechanism is available for the users and DBAs to monitor the progress of the full table scans and to get an idea of the estimated completion time. To provide such a mechanism, Oracle server maintains statistics tracking the progress of such operations and makes it available to the users via the dynamic performance view, `V$SESSION_LONGOPS`.

**Note:** The package `DBMS_APPLICATION_INFO` contains a procedure `SET_SESSION_LONGOPS` to populate the view from an application.

```
SQL> SELECT sid, serial#, opname,
2         TO_CHAR(start_time, 'HH24:MI:SS') AS START,
3         (sofar/totalwork)*100 AS PERCENT_COMPLETE
4 FROM v$session_longops;
```

SID	SERIAL#	OPNAME	START	PERCENT_COMPLETE
---	-----	-----	-----	-----
8	219	TABLE SCAN	13:00:09	48.98098

## Diagnostic Tools



### Monitoring File Use

To monitor which files are subject to the most I/O in an existing database, you can query the following:

- `V$FILESTAT` view
- FILE I/O monitor using EM
- File statistics in `report.txt`

## Using the V\$FILESTAT Dynamic Performance View

You can query V\$FILESTAT to find out the number of disk I/Os for each disk file.

Summarize all of the I/Os on data files on a per-disk basis to find the data files most likely to cause a disk bottleneck.

V\$FILESTAT contains the following columns:

Column	Description
FILE#	File number (join to FILE# in V\$DATAFILE for the name)
PHYRDS	Number of physical reads done
PHYWRTS	Number of physical writes done
PHYBLKRD	Number of physical blocks read
PHYBLKWRT	Number of physical blocks written
READTIM	Time spent doing reads
WRITETIM	Time spent doing writes

**Note:** The last two columns contain 0 unless the TIMED\_STATISTICS parameter is set to TRUE.

Use the following query to monitor these values:

```
SQL> SELECT phyrd,phywrts,d.name
2  FROM v$datafile d, v$filestat f
3  WHERE d.file#=f.file# order by d.name;
PHYRDS      PHYWRTS      NAME
-----
806116  /DISK1/sys01.dbf
168675  /DISK1/temp01.dbf
26257   /DISK2/rbs01.dbf
88      /DISK3/user01.dbf
65012564 /DISK4/scott_dat.dbf
88      /DISK4/scott_ind.dbf
6 rows selected
```

**Note:** In Oracle Enterprise Manager, you can use:

Oracle Diagnostics Pack—>Performance Manager—>I/O—>File I/O Rate

## Using I/O Statistics in report.txt

### I/O Statistics

```
SQL> Rem I/O should be spread evenly across drives. A big difference
between phys_reads and phys_blks_rd implies table scans are going on.
SQL> select table_space, file_name, phys_reads reads, phys_blks_rd
2> blks_read, phys_rd_time read_time, phys_writes writes, phys_blks_wr
3> blks_wrt, phys_wrt_tim write_time
4> from stats$files order by table_space, file_name;
```

TABLE_SPACE	FILE_NAME	READS	BLKS_ READ	READ_ TIME	WRITES	BLKS_ WRT	WRITE_ TIME
RBS	/DISK2/rbs01.dbf	26	26	50	257	257	411
SCOTT_DATA	/DISK4/scott_dat.dbf	65012	416752	38420	564	564	8860
SCOTT_INDEX	/DISK4/scott_ind.dbf	8	8	0	8	8	0
SYSTEM	/DISK1/sys01.dbf	806	1538	1985	116	116	1721
TEMP	/DISK1/temp01.dbf	168	666	483	675	675	0
USER_DATA	/DISK3/user01.dbf	8	8	0	8	8	0

6 rows selected.

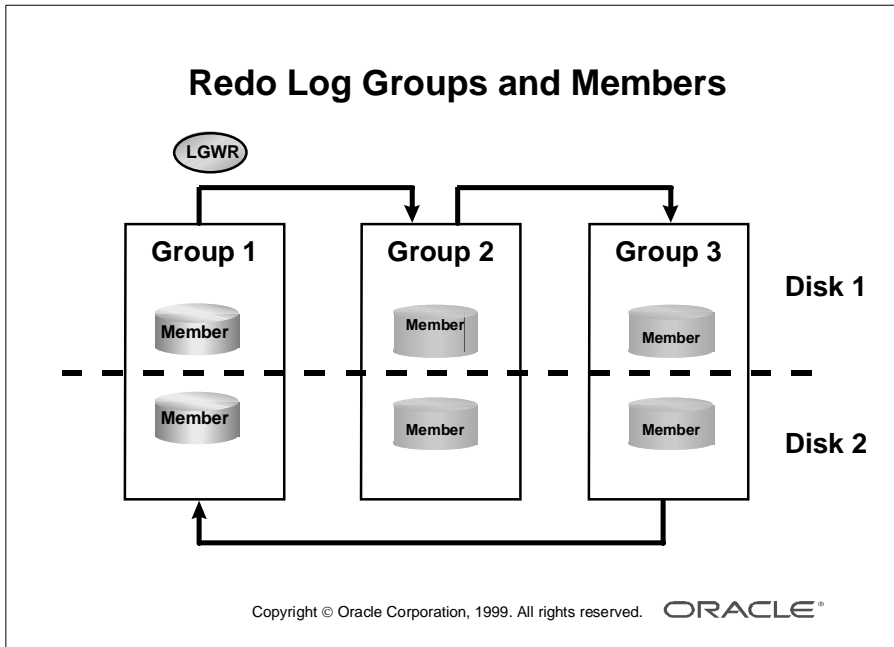
Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

### Using I/O Statistics

Check the output in report.txt to observe how well the I/O load is distributed across the disk devices. The output shows which files are most active. In the example shown on the slide, the tablespace SCOTT\_DATA has a large number of hits. About 98% of the reads are being performed on the data file that contains tables. The index data file is having 0.01% of the disk reads performed against it.

You should investigate how many full table scans are being performed because of missing indexes and ineffective use of the cost-based optimizer.

## Online Redo Log File Configuration



## Online Redo Log File Configuration

You can configure redo log files as follows:

- Size redo log files to minimize contention
- Have enough groups to prevent waiting
- Store redo log files on separate, fast devices
- Query the dynamic performance views V\$LOGFILE and V\$LOG

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

## Online Redo Log File Configuration

Online redo log files are organized in groups. A group must have one or more members. All members of a group have identical contents.

You should have two or more members in each group for safety, unless you are mirroring all files at a hardware level.

Redo log files in the same group should ideally be on separate, fast devices, because LGWR writes to them almost continuously.

**Note:** Redo log files can be created on raw devices.

Properly size redo log files to minimize contention and frequent log switches. You can use the Redo Size statistics in `report.txt`.

**Note:** In Oracle8i, the number of I/O slaves used by the LGWR process to improve redo log throughput is automatically set to 4 when the initialization parameter `DBWR_IO_SLAVES` is set to a value other than 0.

## Monitoring Redo Log File Information

You can query the `V$LOGFILE` and `V$LOG` dynamic performance views to obtain information about the name, location, size, and status of the online redo log file.

Any waits for Log File Parallel Write in `V$SYSTEM_EVENT` indicate a possible I/O problem with the log files.

The Oracle server does not provide a means of monitoring redo disk I/Os, so you must use operating system disk monitoring commands.



## Archive Log File Configuration

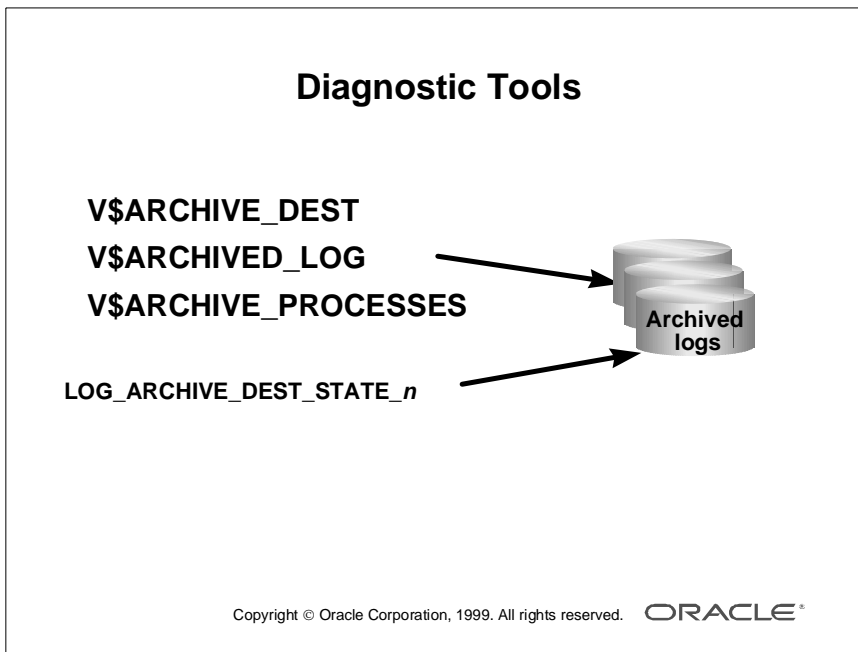
### Archive Log File Configuration

- Allow the LGWR to write to a different disk from the one the ARC $n$  process is reading
- Share the archiving work:

```
ALTER SYSTEM ARCHIVE LOG ALL  
TO <log_archive_dest>
```

- Change archiving speed:  
LOG\_ARCHIVE\_MAX\_PROCESSES,  
LOG\_ARCHIVE\_DEST\_ $n$ ,  
(LOG\_ARCHIVE\_DUPLEX\_DEST,  
LOG\_ARCHIVE\_MIN\_SUCCEED\_DEST)

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



## Archive Log File Configuration

If you archive, it is important to have more than two redo log groups. When a group switches to another group, the DBW $n$  process must checkpoint as usual, and one file must be archived. You need to allow time for both of these operations before the LGWR process needs to overwrite the file again.

## Regulating Archiving Speed

- Occasionally, in very busy databases, a single ARC0 process cannot keep up with the volume of information written to the redo logs. Oracle8i allows the database administrator to define multiple archive processes by using the LOG\_ARCHIVE\_MAX\_PROCESSES parameter.
- The LGWR process starts a new ARC $n$  process whenever the current number of ARC $n$  processes is insufficient to handle the workload. If you anticipate a heavy workload for archiving, you can manually start server processes to share the work by running the command:

```
SQL>ALTER SYSTEM ARCHIVE LOG ALL TO 'directory_name';
```

### Regulating Archiving Speed (continued)

- Monitor V\$ARCHIVE\_PROCESSES to alert and spawn extra archiver processes whenever more than two logs need archiving.

```
SQL> select * from v$archive_processes;
  PROCESS STATUS      LOG_SEQUENCE STAT
-----
0 ACTIVE              122 BUSY
1 ACTIVE              0 IDLE
2 STOPPED             0 IDLE
```

### Note

- A number of I/O slaves used by the ARC0 process to archive redo log files is automatically set to 4 when the initialization parameter DBWR\_IO\_SLAVES is set to a value greater than 0.
- Archive redo log files cannot be created on raw devices.
- Refer to Lesson 2 of the *Oracle8i Backup and Recovery* course for a full explanation of the LOG\_ARCHIVE\_MAX\_PROCESSES, LOG\_ARCHIVE\_DEST\_1, and LOG\_ARCHIVE\_DEST\_STATE\_1 parameters.
- The LOG\_ARCHIVE\_DEST\_1 parameter is valid only if you have installed the Oracle Enterprise Edition. You may continue to use LOG\_ARCHIVE\_DEST if you have installed the Oracle Enterprise Edition, however you cannot use both LOG\_ARCHIVE\_DEST\_1 and LOG\_ARCHIVE\_DEST as they are not compatible.

### Obtaining Information About Archived Log Files and Their Locations

You can query the V\$ARCHIVED\_LOG dynamic performance view to display 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 (the 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.

The dynamic performance view V\$ARCHIVE\_DEST describes, for the current instance, all the archive log destinations, their current value, mode, and status.

## Tuning Checkpoint

### Checkpoints

- **Checkpoints cause:**
  - **DBW $n$  to perform I/O**
  - **CKPT to update the data file header and control file**
- **Frequent checkpoints:**
  - **Reduce instance recovery time**
  - **Decrease run-time performance**

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

### Checkpoints

The LGWR background process writes to redo log groups in a circle, one after the other.

When a group fills, the Oracle server must perform a checkpoint. This means that:

- DBW $n$  writes all or part of the dirty blocks covered by the log being checkpointed to the data files.
- CKPT updates the data file headers and the control files.

Checkpoints normally allow other work to continue at the same time, although the checkpoint generates many disk writes. If the DBW $n$  process has not finished checkpointing a file and LGWR needs the file again, LGWR must wait.

Frequent checkpoints mean shorter instance recovery times but more writes by DBW $n$  (to the data files) and CKPT (to the data file headers). Your choice depends on your priorities regarding recovery time and run-time performance.

## Checkpoint Tuning Guidelines

### Guidelines

- **Size the online redo log files to cut down the number of checkpoints.**
- **Add online redo log groups to increase the time before LGWR starts to overwrite.**
- **Regulate checkpoints with the initialization parameters:**
  - **FAST\_START\_IO\_TARGET**
  - **LOG\_CHECKPOINT\_INTERVAL**
  - **LOG\_CHECKPOINT\_TIMEOUT**
  - **DB\_BLOCK\_MAX\_DIRTY\_TARGET**

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

### Monitoring Checkpoint Frequency

You can check the times of log switches in the `alert.log` file.

Also, check the file for error messages saying “Checkpoint not complete; unable to allocate file.” These messages mean that LGWR has waited for the checkpoint to finish.

If the system statistics Background Checkpoints Started and Background Checkpoints Completed have values that differ by more than 1, the checkpoints are not completing between log switches. You need larger log files.

You can set the `LOG_CHECKPOINTS_TO_ALERT` parameter so that the beginning and end of checkpoints are recorded in the `alert.log` file.

The statistic `DBWn Checkpoint Write Requests` in `report.txt` indicates the number of times that checkpoints were sent to the `DBWn`. A high number of checkpoints per transaction indicates that too many checkpoints are occurring.

## Regulating Checkpoints

DBWn must always perform a checkpoint at the end of each redo log group. You can also set checkpointing with initialization parameters.

**Note:** Refer to Lesson 2 of the *Oracle8i Backup and Recovery* course for a complete explanation of checkpoints and fast-start checkpointing, as well as all parameters involved in these concepts.

If efficient performance is your priority, choose a redo log file size so that checkpoints happen frequently enough not to cause a noticeable slowdown in response, but no more often.

For many sites, this frequency is roughly every 30 minutes, but your database may have checkpoint frequencies of anything between 2 seconds and 8 hours, depending on business needs.

You can experiment with different checkpoint frequencies. For instance, OLTP systems may experience disk contention during checkpoints if the SGA is very large and checkpoints are infrequent. In this case, more frequent checkpoints involve fewer dirty blocks.

**Note:** The DB\_BLOCK\_MAX\_DIRTY\_TARGET parameter is explained in the section on “Tuning DBWn I/Os.”

## Multiple I/O Slaves

### Multiple I/O Slaves

- Provide nonblocking asynchronous I/O requests
- Are deployed by DBW $n$ , LGWR, ARC $n$ , and backup processes
- Are typically not recommended if asynchronous I/O is available
- Follow the naming convention ora\_iNnn\_SID

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

### Multiple I/O Slaves

To share the I/O operations, you can use multiple I/O slaves.

If asynchronous I/O behavior is not natively available, you can simulate it by deploying I/O slave processes. I/O slaves are specialized processes whose only function is to perform I/O.

In Oracle8i, the file I/O interface was reworked to provide nonblocking I/O calls for its clients.

Many platforms that do support asynchronous I/O for disk devices may not do so for tape devices. Even in this case, I/O slaves can be used to do nonblocking I/O to tape devices.

Under heavy loads, the DBW $n$  process may need processes in addition to the I/O slaves to do some of the I/O.

## The I/O Slave Mechanism

I/O slaves can be deployed by the DBW0 process and used by LGWR, ARC0, and backup processes.

I/O slaves for the DBW0 process are allocated immediately after the database is opened, when the first I/O request is made. For example, if the parameter DBWR\_IO\_SLAVES=4, the following I/O slaves are spawned for the DBW0:

```
ora_i101_SID ora_i102_SID ora_i103_SID ora_i104_SID
```

Other I/O slaves (for LGWR, ARC0, and backup processes) are dynamically allocated on an as-needed basis; that is, the I/O issuing process looks for an idle I/O slave. If one is available, that I/O slave will get a post. If there are no idle slaves, the I/O issuer will spawn one. If the allowed number of slaves have been spawned, the issuer waits and tries again to find an idle slave.

The DBW0 continues to do all the DBW0-related work; for example, gathering dirty buffers into a batch. Further, the DBWn process initiates the I/O. The DBW0 I/O slave simply does the I/O on behalf of DBW0. That is the writing of the batch is parallelized between the I/O slaves.

This is beneficial in write-intensive environments, because the CPU time associated with issuing I/Os can be divided between the I/O slaves.



## Initialization Parameters

### Initialization Parameters

**Deploy I/O slaves for DBWR<sub>n</sub>, LGWR, ARC<sub>n</sub>, and BACKUP processes with:**

- DBWR\_IO\_SLAVES
- BACKUP\_TAPE\_IO\_SLAVES

**Turn on or off the asynchronous I/O with:**

- DISK\_ASYNCH\_IO
- TAPE\_ASYNCH\_IO

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

### Parameters for I/O Slave Deployment

The initialization parameters DBWR\_IO\_SLAVES and BACKUP\_TAPE\_IO\_SLAVES control I/O slave deployment.

You can turn on and off the use of asynchronous I/O with the DISK\_ASYNCH\_IO and TAPE\_ASYNCH\_IO parameters. It may be necessary to turn off the asynchronous I/O facility provided by the operating system. For example, if the asynchronous I/O code of the platform has bugs or is not efficient, asynchronous I/O can be disabled on a “per-device type” basis.

Usually the parameter should be left at the default value of TRUE.

## Multiple DBW $n$ Processes

### Multiple DBW $n$ Processes

- Deploy multiple DBW $n$  processes with DB\_WRITER\_PROCESSES (DBW0 to DBW9)
- Useful for SMP systems with large numbers of CPUs
- Cannot concurrently be used with multiple I/O slaves

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

### The Multiple DBW $n$ Mechanism

Multiple DBW $n$  processes can be specified by the DB\_WRITER\_PROCESSES parameter. Up to ten processes (DBW0 to DBW9) can be used. In contrast to the multiple I/O slaves, which only make parallel the writing of the batch between the DBWR I/O slaves, you can make parallel the gathering as well as the writing buffers with the multiple DBW $n$  feature.

From the throughput standpoint,  $n$  DBW $n$  processes should deliver more than the throughput of one DBW0 process with the same number  $n$  of I/O slaves.

## Tuning DBWn I/O

### Tuning DBWn I/O

- Influence the DBWn to write dirty buffers more often with the parameter `DB_BLOCK_MAX_DIRTY_TARGET`.
- If the number of dirty buffers is under the computed low limit, DBWn is not aggressive for writing checkpoints buffers.
- If the number is between the low and high computed limits, DBWn writes from the checkpoint queue until the number of checkpoint buffers drops under low.
- If the number is greater than the high limit, DBWn writes out checkpoint buffers.
- The default value is  $(2^{32}) - 1$ .

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

### Guidelines for Tuning DBWn I/O

If this parameter is set too high, the DBWn process might not write dirty buffers often enough, which can cause sessions to wait for free buffers.

A excessive high value for the `DB_BLOCK_MAX_DIRTY_TARGET` parameter causes extra memory to be used by the DBWn process. The extra memory is allocated in the master process. Also, delays in read and write operations may be seen if this value exceeds the operating system capacity to process all requests, because these requests are queued waiting for the disk to become available.

When setting this parameter, analysis of database statistics related to DBWn, such as the wait event `WRITE COMPLETE WAITS` in the `V$SYSTEM_EVENT`, is needed to determine whether you can benefit from a decreased value for `DB_BLOCK_MAX_DIRTY_TARGET`.

### Predefined Low and High Limits of the Number of Dirty Buffers

- 1 The low limit of dirty buffers is set to  $\min(\max(\text{DB\_BLOCK\_MAX\_DIRTY\_TARGET}, 100), \text{buffers})$ .
- 2 The high limit of dirty buffers is set to  $\min((\text{low limit} * 12) / 10, \text{buffers})$ .

## Summary

### Summary

In this lesson, you should have learned how to:

- Monitor I/O contention
- Stripe Oracle files
- Configure tablespaces, online redo log files, and archived log files
- Configure checkpoint frequency
- Deploy I/O slaves
- Influence DBWn I/Os

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

## Quick Reference

Context	Reference
Initialization parameters	DB_WRITER_PROCESSES DBWR_IO_SLAVES DISK_ASYNC_IO LOG_CHECKPOINTS_TO_ALERT TAPE_ASYNC_IO
Dynamic initialization parameters	BACKUP_TAPE_IO_SLAVES DB_BLOCK_MAX_DIRTY_TARGET DB_FILE_MULTIBLOCK_READ_COUNT FAST_START_IO_TARGET LOG_ARCHIVE_DEST_ <i>n</i> LOG_ARCHIVE_DEST_STATE_ <i>n</i> LOG_ARCHIVE_DUPLEX_DEST LOG_ARCHIVE_MAX_PROCESSES LOG_ARCHIVE_MIN_SUCCEED_DEST LOG_CHECKPOINT_TIMEOUT LOG_CHECKPOINT_INTERVAL
Dynamic performance views	V\$FILESTAT V\$DATAFILE V\$SYSTEM_EVENT V\$LOGFILE V\$LOG V\$ARCHIVE_DEST V\$ARCHIVED_LOG V\$ARCHIVE_PROCESSES V\$SESSION_LONGOPS
Data dictionary views	None
Commands	ALTER TABLE ALLOCATE EXTENT ALTER SYSTEM ARCHIVE LOG ALL
Packaged procedures and functions	None
Scripts	None
Oracle Diagnostics Pack	Performance Manager



## **Using Oracle Blocks Efficiently**

## Objectives

### Objectives

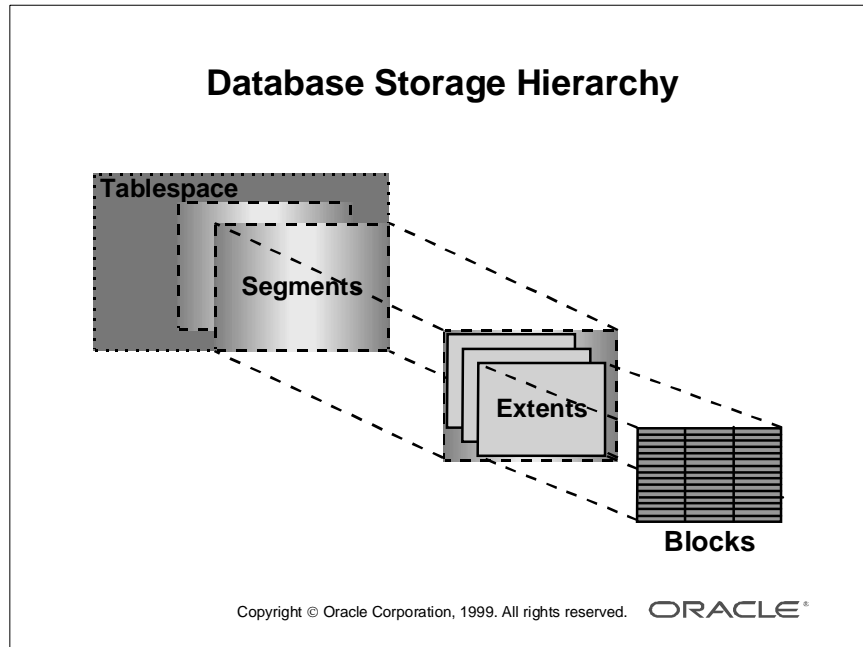
**After completing this lesson, you should be able to do the following:**

- **Determine an appropriate block size**
- **Optimize space usage within blocks**
- **Detect and resolve row migration**
- **Monitor and tune indexes**

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



## Database Storage Hierarchy



### Space Management

The efficient management of space in the database is important to its performance. This lesson examines how to manage extents and blocks in the database.

### Blocks

In an Oracle database, the block is the smallest unit of data file I/O and the smallest unit of space that can be allocated. An Oracle block consists of one or more contiguous operating system blocks.

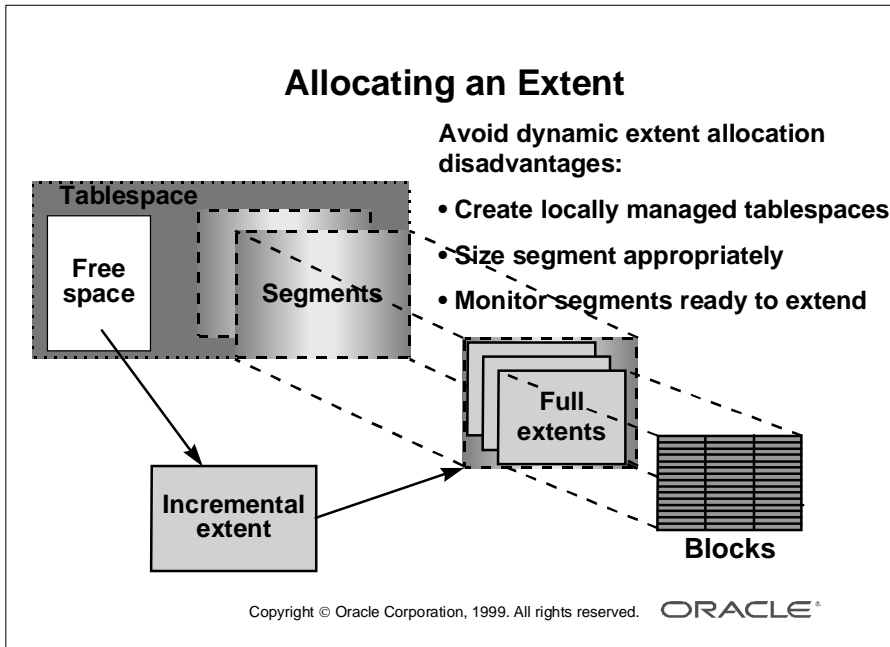
### Extents

An extent is a logical unit of database storage space allocation made up of a number of contiguous data blocks. One or more extents in turn make up a segment. When the existing space in a segment is completely used, the Oracle server allocates a new extent for the segment.

### Segments

A segment is a set of extents that contains all the data for a specific logical storage structure within a tablespace. For example, for each table, the Oracle server allocates one or more extents to form data segments for that table. For indexes, the Oracle server allocates one or more extents to form its index segment.

## Allocating an Extent



### Allocating an Extent

When database operations cause the data to grow and exceed the space allocated, the Oracle server extends the segment. The Oracle server executes several recursive SQL statements to find free space and allocate to the segment when extending the segment. This process, called *dynamic extension*, reduces performance. You can avoid dynamic extension by:

- Creating locally managed tablespaces
- Sizing segments appropriately
- Monitoring and preallocating extents.

## Avoiding Dynamic Space Management

### Avoiding Dynamic Allocation

To display segments with less than 10% free blocks:

```
SQL> SELECT owner, table_name, blocks, empty_blocks
2 FROM dba_tables
3 WHERE empty_blocks / (blocks+empty_blocks) < .1;
```

OWNER	TABLE_NAME	BLOCKS	EMPTY_BLOCKS
HR	EMP	1450	50
HR	REGION	460	40

To avoid dynamic allocation:

```
SQL> ALTER TABLE hr.emp ALLOCATE EXTENT;
Table altered.
```

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

### Avoiding Dynamic Extension

- Size the segment appropriately by:
  - Determining the maximum size of your object
  - Choosing storage parameters that allocate extents large enough to accommodate all of your data when you create the object

When determining the segment size, you should allow for data growth. For example, allocate enough space for the current data and any data that will be inserted into the segment over the next year. For formulas to predict how much space to allow for a table, see the *Oracle8i Server Administrator's Guide*.

- Monitor the database for segments that are about to dynamically extend, and extend them with an ALTER TABLE/INDEX/CLUSTER command.

## Avoiding Dynamic Allocation Disadvantages

Create a locally managed tablespace:

```
CREATE TABLESPACE user_data_1
DATAFILE 'oracle8/oradata/db1/lm_1.dbf'
SIZE 100M
EXTENT MANAGEMENT LOCAL
UNIFORM SIZE 2M;
```

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

## Avoid Dynamic Extension Disadvantages

Create locally managed tablespaces for the objects that extend continuously.

A locally managed tablespace manages its own extents and maintains a bitmap in each data file to keep track of the free or used status of blocks in that data file. Each bit in the bitmap corresponds to a block or a group of blocks. When an extent is allocated or freed for reuse, the Oracle server changes the bitmap values to show the new status of the blocks. These changes do not generate rollback information because they do not update tables in the data dictionary.

---

## Large Extents

### Pros and Cons of Large Extents

- **Pros:**
  - Are less likely to dynamically extend
  - Deliver small performance benefit
  - Can overcome OS limitations on file size
  - Single read against extent map
- **Cons:**
  - Free space may not be available
  - Unused space

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

### Large Extents

To ease space management, the DBA creates objects with appropriate size segments and extents. As a general rule, larger extents are preferred over smaller extents.

### Advantages of Large Extents

- Large extents avoid dynamic extents, because segments with larger extents are less likely to need to be extended.
- Larger extents can improve performance slightly because the Oracle server can read one large extent from disk with fewer multiblock reads than are required to read many small extents. To avoid partial multiblock reads, set the extent size to a multiple of  $5 \times \text{DB\_FILE\_MULTIBLOCK\_READ\_COUNT}$ . Multiply by five because the Oracle server tries to allocate extents on five-block boundaries. By matching extent sizes to the I/O and space allocation sizes, the performance cost of having many extents in a segment is minimized. However, for a table that never has a full table scan operation, it makes no difference in terms of query performance whether the table has one extent or multiple extents.
- For very large tables, operating system limitations on file size may force the DBA to allocate the object with multiple extents.
- The performance of searches using an index is not affected by the index having one extent or multiple extents.

### **Advantages of Large Extents (continued)**

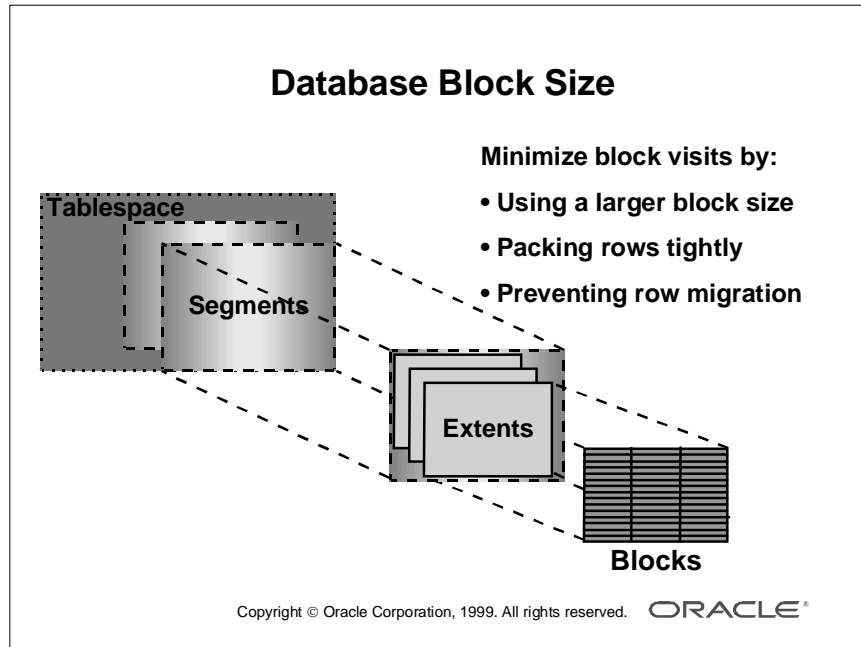
- Extent maps list all the extents for a certain segment. For MAXEXTENTS UNLIMITED, these maps are in multiple blocks. For best performance, you should be able to read the extent map with a single I/O. Performance degrades if multiple I/Os are necessary for a full table scan to get the extent map. Also, a large number of extents can degrade data dictionary performance, because each extent uses space in the dictionary cache.

### **Disadvantages of Large Extents**

- Because large extents require more contiguous blocks, the Oracle server may have difficulty finding enough contiguous space to store them.
- Because the DBA sizes the segment to allow for growth, some of the space allocated to the segment is not used initially.

To determine whether to allocate few large extents or many small extents, consider how the benefits and drawbacks of each would affect your plans for the growth and use of your tables.

## Database Block Size



### Minimizing Block Visits

One of the database tuning goals is to minimize the number of blocks visited. The developer contributes to this goal by tuning the application and SQL statements. The DBA reduces block visits by:

- Using a larger block size
- Packing rows as closely as possible into blocks
- Preventing row migration

Unfortunately for the DBA, the last two goals conflict: as more data is packed into a block, the likelihood of migration increases.

## Oracle Block Size

### **DB\_BLOCK\_SIZE**

- Is set when the database is created
- Is the minimum I/O unit for data file reads
- Default is 2 KB or 4 KB, but up to 64 KB is allowed
- Cannot be easily changed
- Should be a multiple of the OS block size
- OS I/O size is equal to or greater than DB\_BLOCK\_SIZE

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

### Oracle Block Characteristics

The Oracle data blocks have the following characteristics:

- When the database is created, its block size is set to the value of DB\_BLOCK\_SIZE parameter.
- The data block is the minimum unit for data file I/O operation.
- The default block size on most Oracle platforms is either 2 or 4 KB.
- Some operating systems now allow block sizes of up to 64 KB. To configure the block size, check the documentation specific to your operating system, and particularly the Oracle installation and configuration guide for your platform.
- The block size cannot be changed without re-creating or duplicating the database. This makes it difficult to test applications with different block sizes.
- The database block size should be a multiple of the operating system block size.
- If your operating system reads the next block during sequential reads, and your application performs many full table scans, the operating system I/O size should be equal to or greater than the database block size.

The block size chosen does affect performance. Use these general guidelines:

- Random access to a large object, as in an OLTP environment, favors small blocks.
- Sequential access to large amounts of data, as in a DSS environment, prefers large blocks.



## Block Size Advantages and Disadvantages

### Small Block Size Pros and Cons

- **Pros:**
  - Reduces block contention
  - Is good for small rows
  - Is good for random access
- **Cons:**
  - Has relatively large overhead
  - Has small number of rows per block
  - Can cause more index blocks to be read

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

### Small Oracle Blocks

#### Advantages

- Small blocks reduce block contention because there are fewer rows per block.
- Small blocks are good for small rows.
- Small blocks are good for random access. Because it is unlikely that a block will be reused after it is read into memory, a smaller block size makes more efficient use of the buffer cache. This is especially important when memory resources are scarce, because the size of the database buffer cache is limited.

#### Disadvantages

- Small blocks have relatively large overhead.
- You may end up storing only a small number of rows per block, depending on the size of the row. This can cause additional I/Os.
- Small blocks can cause more index blocks to be read.

## Large Block Size Pros and Cons

- **Pros:**
  - **Less overhead**
  - **Good for sequential access**
  - **Good for very large rows**
  - **Better performance of index reads**
- **Cons:**
  - **Increases block contention**
  - **Uses more space in the buffer cache**

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

## Large Oracle Blocks

### Advantages

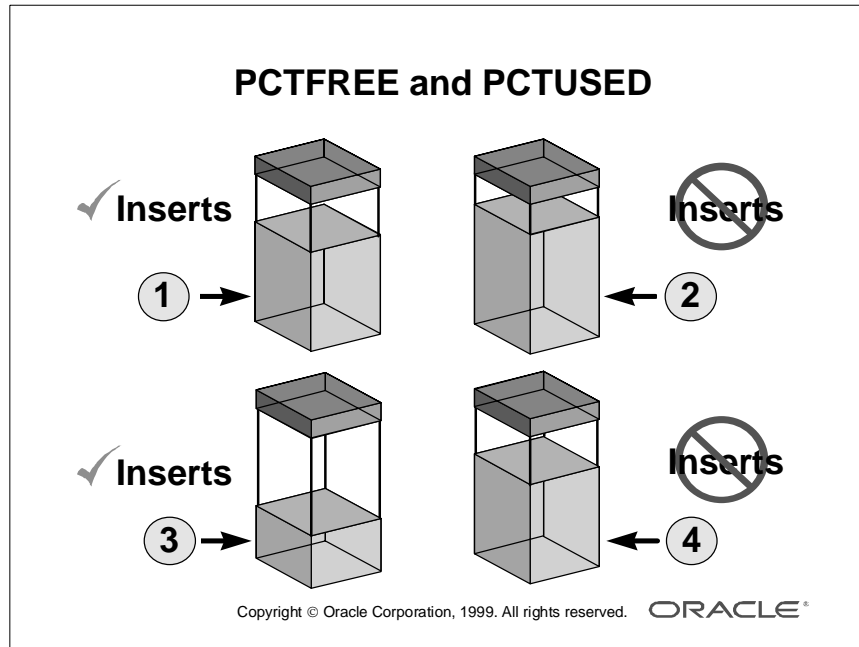
- There is relatively less overhead and thus more room to store useful data.
- Large blocks are good for sequential reads.
- Large blocks are good for very large rows.
- Larger blocks improve performance of index reads. The larger blocks can hold more index entries in each block, which reduces the number of levels in large indexes. Fewer index levels mean fewer I/Os when traversing the index branches.

### Disadvantages

- A large index block size is not good when used in an OLTP type of environment, because it increases block contention on the index leaf blocks.
- Space in the buffer cache is wasted if there are random accesses to small rows and in a large block. For example, with an 8 KB block size and a 50-byte row size, you would be wasting 7,950 bytes in the buffer cache if there were random accesses.

If you resize database blocks and there is no additional memory, you also need to reset DB\_BLOCK\_BUFFERS. This affects the cache hit percentage.

## Block Packing Factors



### Setting the Packing Factor

Two space management parameters, PCTFREE and PCTUSED, enable you to control the use of free space within all the data blocks of a segment. These parameters are also called the packing factors. You specify these parameters when creating or altering a table or cluster (which has its own data segment). You can also specify the storage parameter PCTFREE when creating or altering an index (which has its own index segment).

### The PCTFREE Parameter

The PCTFREE parameter sets the minimum percentage of a data block to be reserved as free space for possible updates to rows that already exist in that block.

### The PCTUSED Parameter

The PCTUSED parameter sets the minimum percentage of a block that can be used for row data plus overhead before new rows will be added to the block.

### How PCTFREE and PCTUSED Work Together

As an example, if you issue a CREATE TABLE statement with PCTFREE 20, the Oracle server reserves 20% of each data block in the data segment of this table for updates to the existing rows in each block. The used space in the block can grow (step 1 on the slide) until the row data and overhead total 80% of the total block size. Then the block is removed from the free list to prevent additional inserts (step 2 on the slide).

### **How PCTFREE and PCTUSED Work Together (continued)**

After you issue a DELETE or UPDATE statement, the Oracle server processes the statement and checks to see if the space being used in the block is now less than PCTUSED. If it is, the block is placed in the free list. When the transaction commits, free space in the block becomes available for other transactions (step 3).

After a data block is filled to the PCTFREE limit again (step 4), the Oracle server considers the block unavailable for the insertion of new rows until the percentage of that block falls below the PCTUSED parameter.

### **DML and PCTFREE and PCTUSED**

Two types of statements can increase the free space of one or more data blocks: DELETE statements and UPDATE statements that update existing values to values that use less space.

Released space in a block may not be contiguous. For example, a row in the middle of the block is deleted. The Oracle server coalesces the free space of a data block only when:

- An INSERT or UPDATE statement attempts to use a block that contains enough free space to contain a new row piece
- The free space is fragmented so that the row piece cannot be inserted in a contiguous section of the block

The Oracle server does this compression only in such situations, because otherwise the database system performance would decrease due to the continuous compression of the free space in data blocks.

## Guidelines for Setting the Packing Factor

### Guidelines

- **PCTFREE**
  - Default 10
  - Zero if no UPDATE activity
  - $PCTFREE = 100 \times upd / (upd + ins)$
- **PCTUSED**
  - Default 40
  - Set if rows deleted
  - $PCTUSED = 100 - PCTFREE - 100 \times rows \times (ins + upd) / blocksize$

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

### Guidelines for Setting PCTUSED and PCTFREE

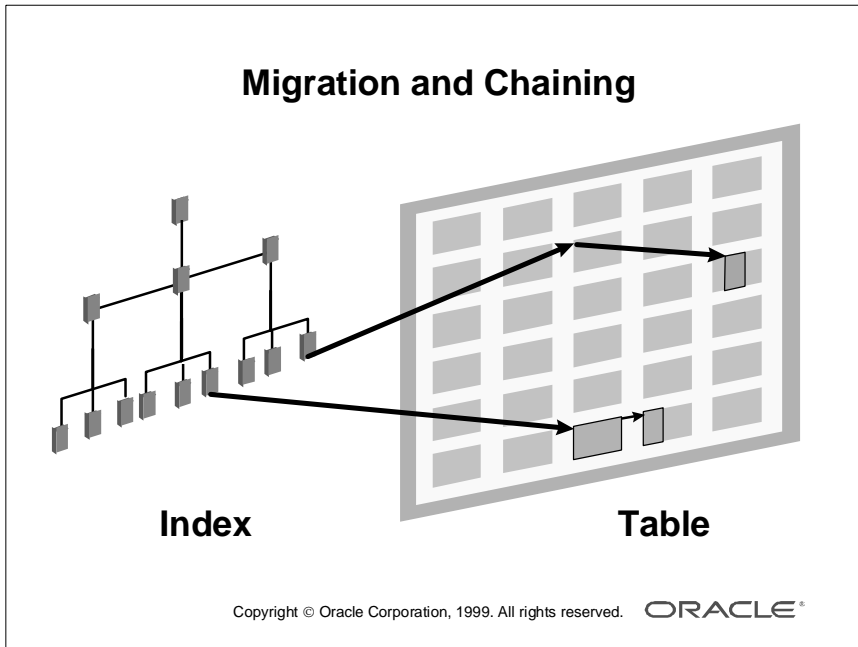
- To set PCTFREE and PCTUSED, use the following formulas:
  - $PCTFREE = 100 \times upd / (upd + ins)$
  - $PCTUSED = 100 - PCTFREE - 100 \times rows \times (ins + upd) / blocksize$

**where:**

upd	is the average amount added by updates, in bytes
ins	is the average initial row length at insert
rows	is the number of rows to be deleted before free list maintenance occurs

- The formula for PCTUSED allows inserts into the table when there is enough space in the block for row updates and for at least one more row.
- PCTUSED is relevant only in tables that undergo deletes, but in many tables you may be able to pack rows into blocks more tightly by setting PCTUSED to be higher than the default (40%).
- For tables with many inserts, changing PCTUSED may improve block storage performance. Blocks that are densely populated can cause free list contention, but fewer blocks are required, tables are smaller, and read operations are faster.
- If you change PCTFREE and PCTUSED for existing tables, there is no immediate impact on blocks. However, future DML activity uses the new rules for tables.

## Migration and Chaining



### Migration and Chaining of Rows

When the data for a row in a table is too large to fit into a single data block, then one of the following would happen.

- The Oracle server stores the data for the row in a chain of data blocks if the data is too large to fit an empty block. This process is called *row chaining*. Chaining can occur when the row is inserted or updated. Row chaining usually occurs with large rows, such as rows that contain an LOB. Row chaining in these cases is unavoidable.
- However, when an UPDATE statement increases the amount of data in a row so that the row no longer fits in its data block, the Oracle server tries to find another block with enough free space to hold the entire row. If such a block is available, the Oracle server moves the entire row to the new block. The Oracle server keeps the original row piece of a migrated row to point to the new block containing the actual row; the ROWID of a migrated row does not change. Indexes are not updated, so they point to the original row location. This process is called *row migration*.

Migration and chaining have a negative effect on performance:

- INSERT and UPDATE statements that cause migration and chaining perform poorly, because they perform additional processing.
- Queries that use an index to select migrated or chained rows must perform additional I/Os.

Migration is caused when PCTFREE is set too low and there is not enough room in the block for updates. To avoid migration, all tables that are updated should have their PCTFREE set so that there is enough space within the block for updates.

## Detecting Chaining and Migration

### Detecting Migration and Chaining

Detect migration and chaining using the ANALYZE command:

```
SQL> ANALYZE TABLE sales.order_hist COMPUTE STATISTICS;
Table analyzed.
```

```
SQL> SELECT num_rows, chain_cnt FROM dba_tables
2  WHERE table_name='ORDER_HIST';
NUM_ROWS CHAIN_CNT
-----
168      102
```

Detect migration and chaining from report.txt:

Statistic	Total	Per transaction ...
table fetch continued row	495	.02 ...

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

### Using the ANALYZE Command to Detect Row Chaining and Migration

You can detect the existence of migrated and chained rows in a table or cluster by using the ANALYZE command with the COMPUTE STATISTICS option. This command counts the number of migrated and chained rows and places this information in the CHAIN\_CNT column of DBA\_TABLES.

The NUM\_ROWS column provides the number of rows stored in the analyzed table or cluster. Compute the ratio of chained and migrated rows to the number of rows to decide if you need to eliminate migrated rows.

### Table Fetch Continued Row Statistic

You can also detect migrated or chained rows by checking the Table Fetch Continued Row statistic in the V\$SYSSTAT view or in report.txt.

### Guidelines

Increase PCTFREE to avoid migrated rows. If you leave more free space available in the block for updates, the row will have room to grow. You can also reorganize (re-create) tables and indexes with a high deletion rate.

## Selecting Migrated and Chained Rows

### Selecting Migrated Rows

```
SQL> ANALYZE TABLE sales.order_hist LIST CHAINED ROWS;
Table analyzed.

SQL> SELECT  owner_name, table_name, head_rowid
2    FROM    chained_rows
3    WHERE table_name = 'ORDER_HIST';
OWNER_NAME  TABLE_NAME  HEAD_ROWID
-----
SALES       ORDER_HIST   AAAA1uAAHAAAAA1AAA
SALES       ORDER_HIST   AAAA1uAAHAAAAA1AAB
...
```

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

### Using ANALYZE ... LIST CHAINED ROWS

You can identify migrated and chained rows in a table or cluster by using the ANALYZE command with the LIST CHAINED ROWS option. This command collects information about each migrated or chained row and places this information into a specified output table. To create the table that holds the chained rows, execute the script UTLCHAIN.SQL:

```
create table CHAINED_ROWS (
  owner_name      varchar2(30),
  table_name      varchar2(30),
  cluster_name    varchar2(30),
  partition_name  varchar2(30),
  head_rowid      rowid,
  analyze_timestampdate );
```

If you create this table manually, your table must have the same column names, data types, and sizes as the CHAINED\_ROWS table.



## Eliminating Migrated Rows

### Eliminating Migrated Rows

1. Run **ANALYZE TABLE ... LIST CHAINED ROWS;**
2. Copy the rows to another table.
3. Delete the rows from the original table.
4. Insert the rows from step 2 back into the original table.

**Step 4 eliminates migrated rows because migration only occurs during an UPDATE operation.**

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

### Eliminating Migrated Rows

You can eliminate migrated rows using this SQL\*Plus script:

```
/* Get the name of the table with migrated rows */
accept table_name prompt 'Enter the name of the table with migrated rows: '
/* Clean up from last execution */
set echo off
drop table migrated_rows;
drop table chained_rows;

/* Create the CHAINED_ROWS table */
@?/rdbms/admin/utlchain
set echo on
spool fix_mig
/* List the chained & migrated rows */
analyze table &table_name list chained rows;
```

### Eliminating Migrated Rows (continued)

```
/* Copy the chained/migrated rows to another table */
create table migrated_rows as
  select orig.* from &table_name orig, chained_rows cr
     where orig.rowid = cr.head_rowid
     and   cr.table_name = upper('&table_name');

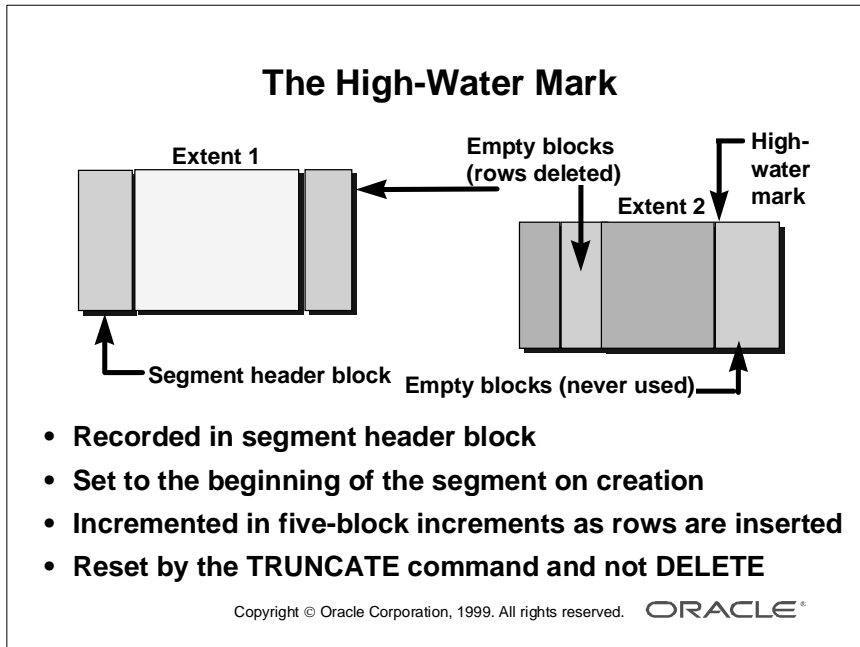
/* Delete the chained/migrated rows from the original table */
delete from &table_name
where rowid in ( select head_rowid from chained_rows );

/* Copy the chained/migrated rows back into the original table */
insert into &table_name select * from migrated_rows;

spool off
```

When using this script, you must disable any foreign key constraints that will be violated when the rows are deleted.

## The High-Water Mark



### High-Water Mark

Space above the high-water mark can be reclaimed at the table level by using the command:

```
ALTER TABLE <table_name> DEALLOCATE UNUSED...
```

In a full table scan, the Oracle server reads in all blocks below the high-water mark. Empty blocks above the high-water mark might waste space, but should not degrade performance; however, underused blocks below the high-water mark can degrade performance.

## Table Statistics

### Table Statistics

Query table statistics from the ANALYZE command:

```
SQL> ANALYZE TABLE hr.emp COMPUTE STATISTICS;
Table analyzed.

SQL> SELECT  num_rows, blocks, empty_blocks as empty,
2           avg_space, chain_cnt, avg_row_len
3 FROM      dba_tables
4 WHERE     owner = 'HR'
5 AND       table_name = 'EMP';
NUM_ROWS BLOCKS EMPTY AVG_SPACE CHAIN_CNT AVG_ROW_LEN
-----
13214    615    35    1753          0        184
```

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

### Statistics Table

You can analyze the storage characteristics of tables, indexes, and clusters to gather statistics, which are then stored in the data dictionary. You can use these statistics to determine whether a table or index has unused space.

Query the DBA\_TABLES view to see the resulting statistics:

Columns	Description
NUM_ROWS	Number of rows in the table
BLOCKS	Number of blocks below the table high-water mark
EMPTY_BLOCKS	Number of blocks above the table high-water mark
AVG_SPACE	Average free space in bytes in the blocks below high-water mark
AVG_ROW_LEN	Average row length, including row overhead
CHAIN_CNT	Number of chained or migrated rows in the table

EMPTY\_BLOCKS represents blocks that have not yet been used, rather than blocks that were full and now are empty.

## The DBMS\_SPACE Package

### The DBMS\_SPACE Package

```

declare
    owner      VARCHAR2(30);
    name       VARCHAR2(30);
    seg_type   VARCHAR2(30);
    tblock     NUMBER;
    ...
BEGIN
    dbms_space.unused_space
        ('&owner', '&table_name', 'TABLE',
         tblock, tbyte, ublock, ubyte, lue_fid, lue_bid, lublock);

    dbms_output.put_line(...
END;
/

```

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

### Using the DBMS\_SPACE Package

You can also use the supplied package DBMS\_SPACE to get information about space use in segments. It contains two procedures:

- **UNUSED\_SPACE** returns information about unused space in an object (table, index, or cluster). Its specification is:

unused_space(segment_owner IN	varchar2,
segment_name IN	varchar2,
segment_type IN	varchar2,
total_blocks OUT	number,
total_bytes OUT	number,
unused_blocks OUT	number,
unused_bytes OUT	number,
last_used_extent_file_id OUT	number,
last_used_extent_block_id OUT	number,
last_used_block OUT	number);

### Using the DBMS\_SPACE Package (continued)

- `FREE_BLOCKS` returns information about free blocks in an object (table, index, or cluster). Its specification is:

```
free_blocks(segment_owner IN varchar2
            segment_name IN      varchar2,
            segment_type IN      varchar2,
            freelist_group_id IN number,
            free_blks OUT        number
            scan_limit IN        number DEFAULT NULL);
```

These procedures are created by and documented in the `dbmsutil.sql` script that is run by `catproc.sql`. When running this package, you must provide a value for the `FREE_LIST_GROUP_ID`. Use a value of 1, unless you are using Oracle Parallel Server.

This script prompts the user for the table owner and table name, executes `DBMS_SPACE.UNUSED_SPACE`, and displays the space statistics.

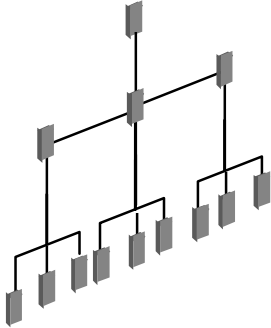
```
declare
    owner      varchar2(30);
    name       varchar2(30);
    seg_type   varchar2(30);
    tblock     number;
    tbyte      number;
    uBlock     number;
    ubyte      number;
    lue_fid    number;
    lue_bid    number;
    lublock    number;
```

**Using the DBMS\_SPACE Package (continued)**

```
BEGIN
    dbms_space.unused_space('&owner','&table_name','TABLE',
        tblock,tbyte,ublock,ubyte,lue_fid,lue_bid,lublock);
    dbms_output.put_line('Total blocks allocated to table      = '
        || to_char(tblock));
    dbms_output.put_line('Total bytes allocated to table      = '
        || to_char(tbyte));
    dbms_output.put_line('Unused blocks(above HWM)           = '
        || to_char(ublock));
    dbms_output.put_line('Unused bytes(above HWM)           = '
        || to_char(ubyte));
    dbms_output.put_line('Last extent used file id       = '
        || to_char(lue_fid));
    dbms_output.put_line('Last extent used begining block id = '
        || to_char(lue_bid));
    dbms_output.put_line('Last used block in last extent    = '
        || to_char(lublock));
END;
```

## Index Reorganization

### Index Reorganization



- Indexes on volatile tables cause a performance problem.
- Empty index blocks go to the free list.
- Even if a block contains only one entry, it must be maintained.
- You may need to rebuild indexes.

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

### Managing Index Blocks

Indexes on volatile tables can also present a performance problem.

In data blocks, the Oracle server replaces deleted rows with inserted ones; however, in index blocks, the Oracle server orders entries sequentially. Values must go in the correct block, together with others in the same range.

Many applications insert rows (hence indexes) in ascending order and delete older values. The insert and delete operations fragment the data segments and index segments. To overcome the fragmentation, you may need to rebuild your indexes regularly.

If you delete all the entries from an index block, the Oracle server puts the block back on the free list.



## Monitoring and Rebuilding Indexes

### Monitoring Indexes

```
SQL> ANALYZE INDEX acct_no_idx VALIDATE STRUCTURE;
Index analyzed.

SQL> SELECT (DEL_LF_ROWS_LEN/LF_ROWS_LEN) * 100
2         AS index_usage
3 FROM   index_stats;
INDEX_USAGE
-----
24

SQL> ALTER INDEX acct_no_idx REBUILD;
Index altered.
```

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

### Monitoring Index Space

You can monitor the space used by indexes by using the command:

```
SQL> ANALYZE INDEX index_name VALIDATE STRUCTURE;
```

Then query the INDEX\_STATS view:

Column	Description
LF_ROWS	Number of values currently in the index
LF_ROWS_LEN	Sum in bytes of the length of all values
DEL_LF_ROWS	Number of values deleted from the index
DEL_LF_ROWS_LEN	Length of all deleted values

**Note:** The INDEX\_STATS view contains the last analyzed index. The view is session-specific. If you connect to another session under the same user, the view is populated with the index that is analyzed by the session that is querying the view.

## Rebuilding Indexes

You may decide to rebuild the indexes if deleted entries represent 20% or more of current entries, although again this depends on your application and priorities. You can use a query to find the ratio. Use the `ALTER INDEX REBUILD` statement to reorganize or compact an existing index or to change its storage characteristics. The `REBUILD` statement uses the existing index as the basis for the new index. All index storage commands are supported, such as `STORAGE` (for extent allocation), `TABLESPACE` (to move the index to a new tablespace), and `INITTRANS` (to change the initial number of entries). When building an index, you can also use the following keywords to reduce the time it takes to rebuild:

- `PARALLEL` or `NOPARALLEL` (`NOPARALLEL` is the default)
- `RECOVERABLE` or `UNRECOVERABLE` (`RECOVERABLE` is the default)
  - `UNRECOVERABLE` is faster because it does not write redo log entries when the index is created or rebuilt.
  - This clause does not set a permanent attribute of the object, but is only effective when the index is created; thus, it does not appear in the data dictionary.
  - This attribute cannot be updated.
  - This attribute can only be used when the object is created or rebuilt.
  - This attribute implies `LOGGING` by default, which means that additional inserts are logged.
  - The index is not recoverable if it needs to be re-created during a recovery operation.
- `LOGGING` or `NOLOGGING`
  - `NOLOGGING` is faster because it does not write any redo log entries during the index life until `NOLOGGING` is changed to `LOGGING`.
  - This attribute is permanent and thus appears in the dictionary.
  - This attribute can be updated (`ALTER INDEX NOLOGGING/LOGGING`) at any time.

`UNRECOVERABLE` and `LOGGING` are not compatible.

**Note:** The `UNRECOVERABLE` option can be used in early versions of Oracle8, but will eventually be replaced by the `NOLOGGING` option.

To duplicate the semantics of the `UNRECOVERABLE` clause, create an object with the `NOLOGGING` option and then use the `ALTER` command, specifying `LOGGING`. To duplicate the semantics of a `RECOVERABLE` clause, create an object with the `LOGGING` option.

**Rebuilding Indexes (continued)**

Using ALTER INDEX REBUILD is usually faster than dropping and re-creating an index because it uses the fast full scan feature. It thus reads all the index blocks, using multiblock I/O, then discards the branch blocks. Oracle8i introduces a method of creating an index or re-creating an existing index while allowing concurrent operations on the base table. This helps achieve demanding availability goals by allowing maintenance operations to be performed online with no downtime.

## Summary

### Summary

**In this lesson, you should have learned how to store blocks as economically as possible by:**

- **Using a larger block size**
- **Setting PCTFREE and PCTUSED**
- **Rebuilding tables with many empty blocks**
- **Rebuilding tables with migrated rows**
- **Rebuilding volatile indexes**
- **Using locally managed tablespaces**

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

## Quick Reference

Context	Reference
Initialization parameters	None
Dynamic performance views	V\$SYSSTAT V\$SESSTAT V\$MYSTAT
Data dictionary views	USER_, ALL_, DBA_CLUSTERS USER_, ALL_, DBA_INDEXES USER_, ALL_, DBA_TABLES INDEX_STATS
Commands	CREATE TABLESPACE ... EXTENT MANAGEMENT LOCAL ALTER/CREATE INDEX/TABLE/CLUSTER ALTER INDEX ... REBUILD TRUNCATE ANALYZE ... COMPUTE STATISTICS ANALYZE ... LIST CHAINED ROWS
Packaged procedures and functions	DBMS_SPACE
Scripts	dbmsutil.sql, utlchain.sql
Oracle Diagnostics Pack	Performance Manager



---

# 10

---

## **Optimizing Sort Operations**

## Objectives

### Objectives

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

- **Identify the SQL operations that require sorting**
- **Ensure that sorting is done in memory where possible**
- **Reduce the number of I/Os required for the sort runs**
- **Allocate temporary space appropriately**

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



## Sort Operations

### Operations Requiring Sorting

- Index creation
- Parallel insert operation involving index maintenance
- ORDER BY or GROUP BY clauses
- DISTINCT values selection
- UNION, INTERSECT, or MINUS operators
- Sort-merge joins
- ANALYZE command execution

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

### Operations That Require Sorting

The following are some of the operations that require sorting of data.

**Index Creation** The server process (or processes, if the index is being created in parallel) has to sort the indexed values before building the B-tree.

**ORDER BY or GROUP BY Clauses** The server process must sort on the values in the ORDER BY or GROUP BY clauses.

**DISTINCT Values** For the DISTINCT keyword, the sort must eliminate duplicates.

**UNION, INTERSECT, or MINUS Operator** Servers need to sort the tables they are working on to eliminate duplicates.

## Operations That Require Sorting (continued)

**Sort-Merge Joins** Sort-merge joins are performed for queries that involve joining two tables on columns that have no indexes; for example a query such as the following:

```
SQL> select dept_id, id from s_emp, s_dept
2   where s_emp.dept_id = s_dept.id;
```

If there are no indexes available, an equijoin request needs to:

- 1 Perform full table scans of S\_EMP and S\_DEPT tables.
- 2 Sort each row source separately.
- 3 Merge the sorted sources together, combining each row from one source with each matching row of the other source.

Before the sort operation:

```
SQL> select name, value
2   from v$sysstat where name = 'sorts (rows)';
NAME                                VALUE
-----
sorts (rows)                        639330
```

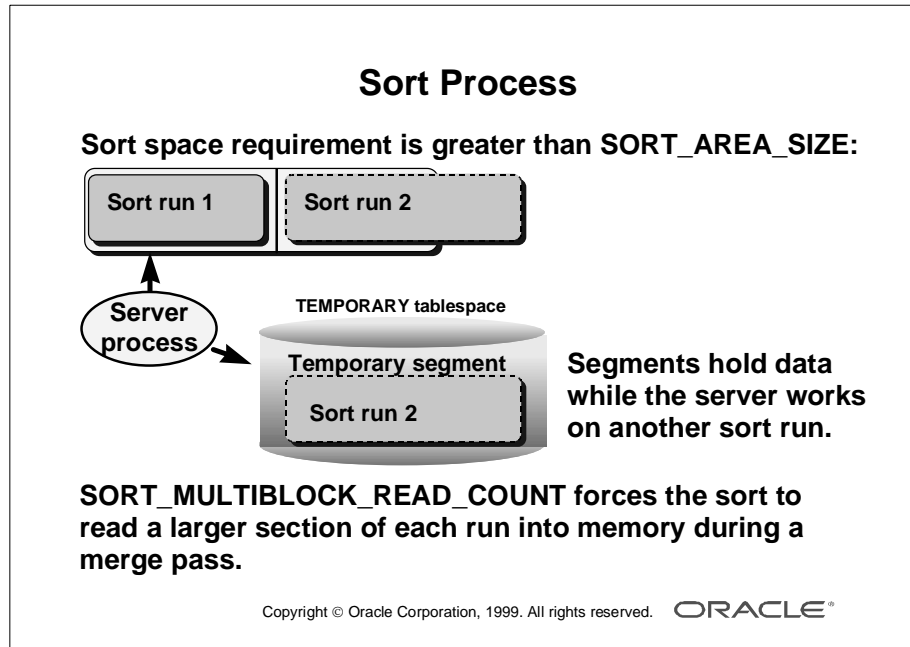
After the sort operation:

```
SQL> select name, value
2   from v$sysstat where name = 'sorts (rows)';
NAME                                VALUE
-----
sorts (rows)                        655714
```

**ANALYZE Execution** The ANALYZE command is useful for collecting statistics on tables, indexes, and clusters to help the CBO define the best execution plans. It sorts the data to provide summarized information.

```
SQL> select name, value
2   from v$sysstat where name = 'sorts (rows)';
NAME                                VALUE
-----
sorts (rows)                        61751
SQL> execute sys.dbms_utility.analyze_schema('SCOTT','COMPUTE');
PL/SQL procedure successfully completed.
SQL> select name, value
2   from v$sysstat where name = 'sorts (rows)';
NAME                                VALUE
-----
sorts (rows)                        622930
```

## Sort Process



### The Sort Process

The Oracle server sorts in memory if the work can be done within an area smaller than the value (in bytes) of the parameter SORT\_AREA\_SIZE.

If the sort needs more space than this value:

- 1** The data is split into smaller pieces, called sort runs; and each piece is sorted individually.
- 2** The server process writes pieces to temporary segments on disk; these segments hold intermediate sort runs data while the server works on another sort run.
- 3** The sorted pieces are merged to produce the final result. 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.

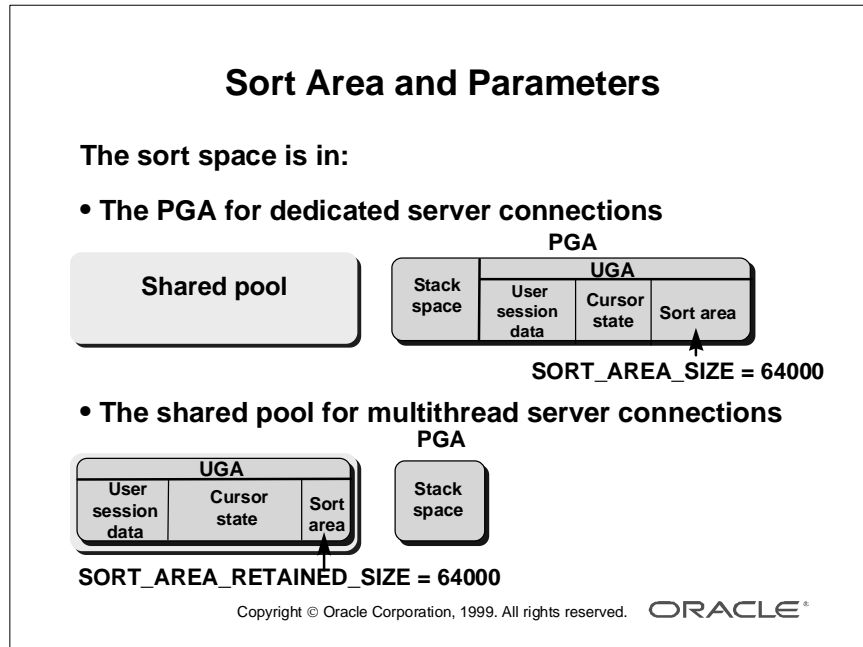
### **The SORT\_MULTIBLOCK\_READ\_COUNT Parameter**

The sort process can be forced to read a larger section of each sort run from disk to memory during each merge pass by increasing the number of blocks to read each time a sort performs a read from a temporary segment. This number is set through the SORT\_MULTIBLOCK\_READ\_COUNT. If the system is performing too many I/Os per second during sort operations and the CPUs are relatively idle during that time, consider increasing the SORT\_MULTIBLOCK\_READ\_COUNT parameter to force the sort operations to perform fewer, larger I/Os. This also forces the sort process to reduce 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 because of an increase in the number of merge passes.

**Note:** The sort process 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.

## Sort Area and Parameters



### Sort Area

The sort space is a part of:

- The PGA when connected with a dedicated server
- The shared pool or large pool when connected with MTS

### Parameters

The parameters that affect the size of sort space are SORT\_AREA\_SIZE and SORT\_AREA\_RETAINED\_SIZE.

#### SORT\_AREA\_SIZE

- The sort area is sized with the `init.ora` parameter SORT\_AREA\_SIZE.
- It can be set dynamically, using the ALTER SESSION or ALTER SYSTEM DEFERRED command.
- The default value depends on the operating system.
- The default is generally adequate for most OLTP operations. Adjust it for DSS applications, batch jobs, or large operations.

## Parameters (continued)

### **`SORT_AREA_RETAINED_SIZE`**

- When the sort completes and the sort area still contains sorted rows to be fetched, the sort area can shrink to the size specified by the parameter `SORT_AREA_RETAINED_SIZE`.
- The memory is released back to the User Global Area (UGA) for use by the same Oracle server process (not to the operating system) after the last row is fetched from the sort space.
- The default value for this parameter is equal to the value of the `SORT_AREA_SIZE` parameter.

## Sort Area and Parameters

Each parallel query server needs **`SORT_AREA_SIZE`**.

Two sets of servers can write at once, so:

- Calculate **`SORT_AREA_SIZE × 2 × degree of parallelism`**.
- Add **`SORT_AREA_RETAINED_SIZE × degree of parallelism × number of sorts above two`**.

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

## Memory Requirements

**Single Server Process** An execution plan can contain multiple sorts. For example, a sort-merge join of two tables can be followed by a sort for an **`ORDER BY`** clause. This constitutes three sorts altogether.

If a single server is working on the sort process, while it performs the **`ORDER BY`** sort it uses:

- An area of **`SORT_AREA_SIZE`**, in bytes, for the active sort
- Two areas of the size specified by **`SORT_AREA_RETAINED_SIZE`**, for the join sorts

**Parallel Query Processes** If you parallelize the statement, each query server needs an amount of memory equal to **`SORT_AREA_SIZE`**.

With parallel query, two sets of servers can be working at once, so you should:

- Calculate **`SORT_AREA_SIZE × 2 × degree of parallelism`**
- If necessary, add **`SORT_AREA_RETAINED_SIZE × degree of parallelism × number of sorts above two`**

If you can still afford the memory, the optimal value for **`SORT_AREA_SIZE`** and **`SORT_AREA_RETAINED_SIZE`** for parallel query is 1 MB. In testing, larger values than this have not improved performance significantly.

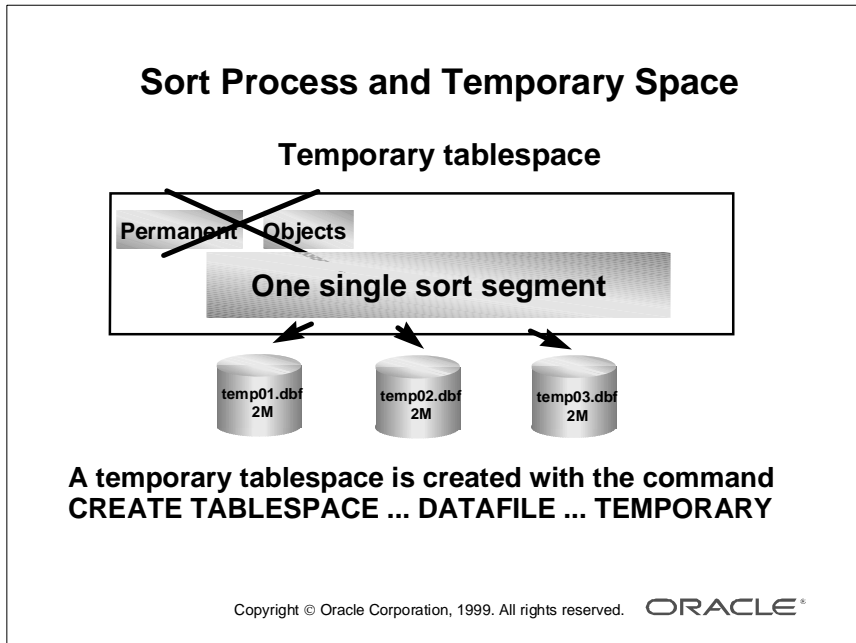
## **Sizes**

Usually, `SORT_AREA_SIZE` and `SORT_AREA_RETAINED_SIZE` should be set to the same value, unless:

- You are very short of memory
- You are using a multithreaded server



## Sort Process and Temporary Space



### Advantage of Using Temporary Tablespaces

Designating temporary tablespaces exclusively for sorts effectively eliminates serialization of space management operations involved in the allocation and deallocation of sort space.

## Temporary Space Segment

- Is created by the first sort operation
- Extends as demands are made on it
- Comprises extents, which can be used by different sort operations
- Is described in the SGA in the sort extent pool (SEP)

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

## Syntax

```
CREATE TABLESPACE name... DATAFILE ... TEMPORARY;  
ALTER TABLESPACE name... TEMPORARY;
```

A tablespace created or altered in this way:

- Cannot contain any permanent objects
- Is exclusively designated for sorts
- Contains a single sort segment per instance for Oracle Parallel Server environments

## The Sort Segment

- Created during the first sort operation that uses the tablespace
- Dropped when the database is closed
- Grows as demands are made on it
- Made up of extents, each of which can be used by different sort operations
- Described in an SGA structure called the sort extent pool (SEP) (When a process needs sort space, it looks for free extents in the SEP.)

## Tuning Sort Operations

### Tuning Sort Operations

- Avoid sort operations whenever possible
- Reduce swapping and paging by ensuring that sorting is done in memory where possible
- Reduce space allocation calls: allocate temporary space appropriately

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

### Diagnosing Problems

- You may be able to avoid sort operations if the processed data has previously been sorted.
- When sort operations are not too large, a sort area that is too small results in performance overheads by swapping to disk; ensure that the sort operations occur in memory.
- Using large chunks of memory for sorting may result in paging and swapping, and reduce overall system performance.
- Allocation and deallocation of temporary segments occur frequently on permanent tablespaces.

### Tuning Goals

- Avoid sort operations that are not necessary
- Optimize memory sort and disk overhead
- Eliminate space allocation calls to allocate and deallocate temporary segments

## Avoiding Sort Operations

### Avoiding Sort Operations

Avoid sort operations whenever possible by:

- Using NOSORT to create indexes
- Using UNION ALL instead of UNION
- Using index access for table joins
- Creating indexes on columns referenced in the ORDER BY clause
- Selecting the columns for analysis
- Using ESTIMATE rather than COMPUTE for large objects

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

### NOSORT Clause

Use the NOSORT clause when creating indexes for presorted data on a single-CPU machine using SQL\*Loader. This clause is only valid for the data inserted into a table:

```
SQL> create index S_EMP_DEPT_ID_FK on s_emp(dept_id) NOSORT;  
ORA-01409: NOSORT option may not be used; rows are not in  
ascending order
```

On a multi-CPU machine, it is probably quicker to load data in parallel, even though it is not loaded in order. Then you can use parallel index creation to speed up sorting.

### UNION ALL

Use UNION ALL instead of UNION; this clause does not eliminate duplicates, so does not need to sort.

### Nested Loop Joins

Use index access for equijoin requests:

```
SQL> select dept_id, id  
2   from s_emp, s_dept  
3  where s_emp.dept_id = s_dept.id;
```

### **Nested Loops Joins (continued)**

The Oracle optimizer chooses a nested loop join instead of a sort-merge join. A nested loop join does not require any sort operations. The steps required to influence the Oracle server to use the nested loop join are:

- 1** Perform a full table scan of the S\_EMP table.
- 2** Use the DEPT\_ID value for each row returned to perform a unique scan on the PK\_ID index.
- 3** Use the ROWID retrieved from the index scan to locate the matching row in the S\_DEPT table.
- 4** Combine each row returned from S\_EMP with the matching row returned from S\_DEPT.

### **INDEXES and ORDER BY**

Create indexes on columns that are frequently referenced with ORDER BY statements. Since the index is sorted in ascending order and is doubly linked, the Oracle server will use the index rather than a sort operation.

### **ANALYZE FOR COLUMNS**

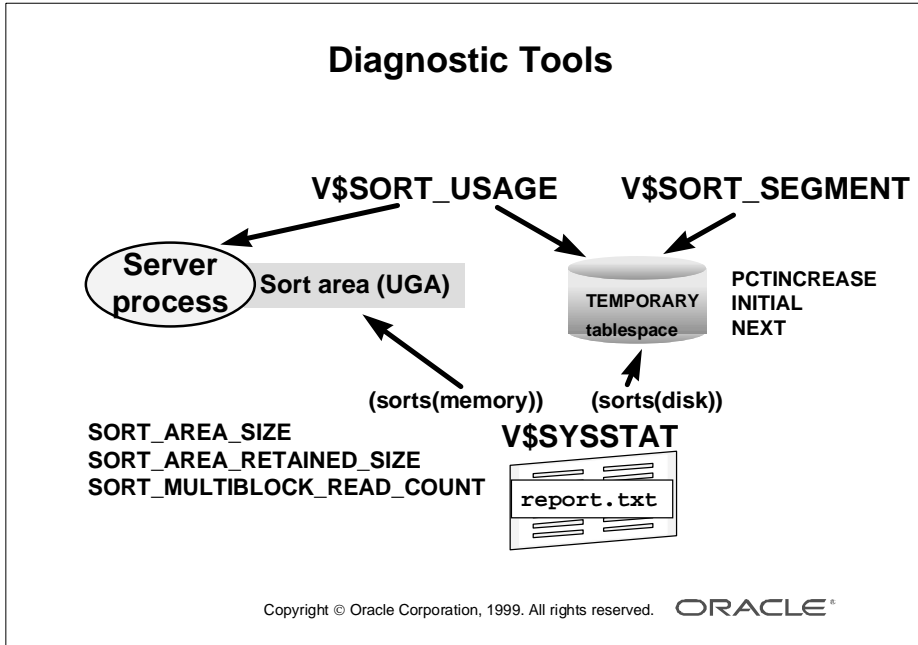
Reduce the amount of statistics collected for the columns of interest only; for example, those involved in join conditions, ANALYZE... FOR COLUMNS, or ANALYZE...FOR ALL INDEXED COLUMNS.

**Note:** The ANALYZE... SIZE *n* command creates histograms for the columns involved. Combining this clause with the clause FOR ALL INDEXED COLUMNS generates unnecessary histograms for primary keys and unique constraints.

### **ANALYZE ESTIMATE**

The COMPUTE clause is more precise for the optimizer. However, it demands a large amount of sort space. The ESTIMATE clause is preferable for large tables and clusters.

## Diagnostic Tools for Tuning Sort Operations



### Dynamic Views and report.txt Output

- The V\$SYSSTAT view displays the number of all sorts (memory), all sorts (disk), and all sorts (rows).
  - sorts (disk): Number of sorts requiring I/O to temporary segments
  - sorts (memory): Number of sorts performed entirely in memory
  - sorts (rows): Total rows sorted in the period being monitored

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

STATISTIC#	NAME	CLASS	VALUE
161	sorts (memory)	64	154
162	sorts (disk)	64	4
163	sorts (rows)	64	571768

**Dynamic Views and report.txt Output (continued)**

- The `report.txt` output provides the same information. Moreover, these figures pertain to the the period between when the `UTLBSTAT` and `UTLESTAT` scripts were run.

Statistic	Total	Per Transact	Per Logon	Per Second
sorts (disk)	4	.02	.41	.01
sorts (memory)	154	.27	5.77	.12
sorts (rows)	571768	39.62	862.59	18.19

- The `V$SORT_SEGMENT` and `V$SORT_USAGE` views display information about the temporary segments used and about the users who used them.

**Note:** Using the Oracle Enterprise Manager Diagnostics Pack:

Performance Manager—>Database\_Instance—>System Statistics

Performance Manager—>Load—>Sort Rows Rate

## Diagnostics and Guidelines

### Diagnostics and Guidelines

```
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
23	206	11.165049

- The ratio of disk sorts to memory sorts should be less than 5%.
- Increase the size of SORT\_AREA\_SIZE if the ratio is greater than 5%.

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

### Ratio

The ratio of disk sorts to memory sorts should be less than 5%.

### Guidelines

If the ratio indicates a high number of sorts going to disk, increase the size of SORT\_AREA\_SIZE. It increases the size of each run and decreases the total number of runs and merges.

### Performance Trade-Offs for Large Sort Areas

Increasing the size of the sort area causes each server process that sorts to allocate more memory. It may affect operating system memory allocation and induce paging and swapping.

If you increase sort area size, consider decreasing the retained size of the sort area, or the size to which Oracle reduces the sort area if its data is not expected to be referenced soon. A smaller retained sort area reduces memory usage but causes additional I/O to write and read data to and from temporary segments on disk.

**Note:** Oracle Enterprise Manager Diagnostics Pack application:

Performance Manager—>Memory—>Memory Sort Hit%



## Monitoring Temporary Tablespaces

### Monitoring Temporary Tablespaces

```
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
-----
TEMP                2           4              3
20                200          200
```

- Default storage parameters apply to sort segments.
- Sort segments have unlimited extents.

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

### The V\$SORT\_SEGMENT View

This view contains information about every sort segment of the TEMPORARY tablespaces in the instance.

Column	Description
CURRENT_USERS	Number of active users
TOTAL_EXTENTS	Total number of extents
USED_EXTENTS	Extents currently allocated to sorts
EXTENT_HITS	Number of times an unused extent was found in the pool
MAX_USED_BLOCKS	Maximum number of used blocks
MAX_SORT_BLOCKS	Maximum number of blocks used by an individual sort

### Temporary Tablespace Configuration

Default storage parameters for the temporary tablespace apply to sort segments, except that they have unlimited extents—whatever value MAXEXTENTS is set to.

## Configuring Temporary Tablespaces

### Temporary Tablespace Configuration

- Set appropriate storage values.
- Set up different TEMPORARY tablespaces based on sorting needs.

```
SQL> SELECT session_num, tablespace, extents, blocks
2 FROM v$sort_usage;
SESSION_NUM TABLESPACE          EXTENTS    BLOCKS
-----
16 TEMP                      4          200
```

- Stripe temporary tablespaces.

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

### Guidelines

**Storage Parameters** Because sorts are done in memory if they are smaller than SORT\_AREA\_SIZE, you should consider this value when setting extent sizes.

- Select INITIAL and NEXT values as integer multiples of SORT\_AREA\_SIZE, allowing an extra block for the segment header.
- Set PCTINCREASE to 0.

**Different Temporary Tablespaces** To define the sort space needed by the users, join the V\$SESSION and V\$SORT\_USAGE views to obtain information on the currently active disk sorts in the instance:

```
SQL> SELECT s.username, u."USER", u.tablespace, u.contents,
2          u.extents, u.blocks
3 FROM v$session s,v$sort_usage u
4 WHERE s.saddr=u.session_addr
5 AND    u.contents='TEMPORARY' ;
```

**Guidelines (continued)****Different Temporary Tablespaces (continued)**

USERNAME	USER	TABLESPACE	CONTENTS	EXTENTS	BLOCKS
-----	-----	-----	-----	-----	-----
STAT	SYSTEM	TEMP	TEMPORARY	20	1000
SCOTT	SYSTEM	TEMP	TEMPORARY	2	100

The USER column in the V\$SORT\_USAGE view always shows the user who is querying this view. The user performing the sort is the username from the V\$SESSION view.

The user STAT requiring a large amount of disk space should be assigned another temporary tablespace with larger extent size.

**Striping** The temporary tablespace should be striped over many disks. If the temporary tablespace is only striped over two disks with a maximum of 50 I/Os per second each, then you can only perform 100 I/Os per second. This restriction could become a problem, making sort operations take a very long time. You can speed up sort operations fivefold if you stripe the temporary tablespace over ten disks. This enables 500 I/Os per second.

## Summary

### Summary

In this lesson, you should have learned how to:

- Avoid sort operations
- Size `SORT_AREA_SIZE` for sorting in memory
- Size `SORT_MULTIBLOCK_READ_COUNT` to reduce the number of I/Os
- Configure `TEMPORARY` tablespaces

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

---

## Quick Reference

Context	Reference
Initialization parameters	<code>SORT_AREA_SIZE</code> <code>SORT_AREA_RETAINED_SIZE</code> <code>SORT_MULTIBLOCK_READ_COUNT</code>
Dynamic performance views	<code>V\$SYSSTAT</code> <code>V\$SORT_SEGMENT</code> <code>V\$SORT_USAGE</code> <code>V\$SESSION</code>
Data dictionary views	None
Commands	None
Packaged procedures and functions	None
Scripts	None
Oracle Diagnostics Pack	Performance Manager



## **Tuning Rollback Segments**

## Objectives

### Objectives

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

- **Use the dynamic performance views to check rollback segment performance**
- **Reconfigure and monitor rollback segments**
- **Define the number and sizes of rollback segments**
- **Appropriately allocate rollback segments to transactions**

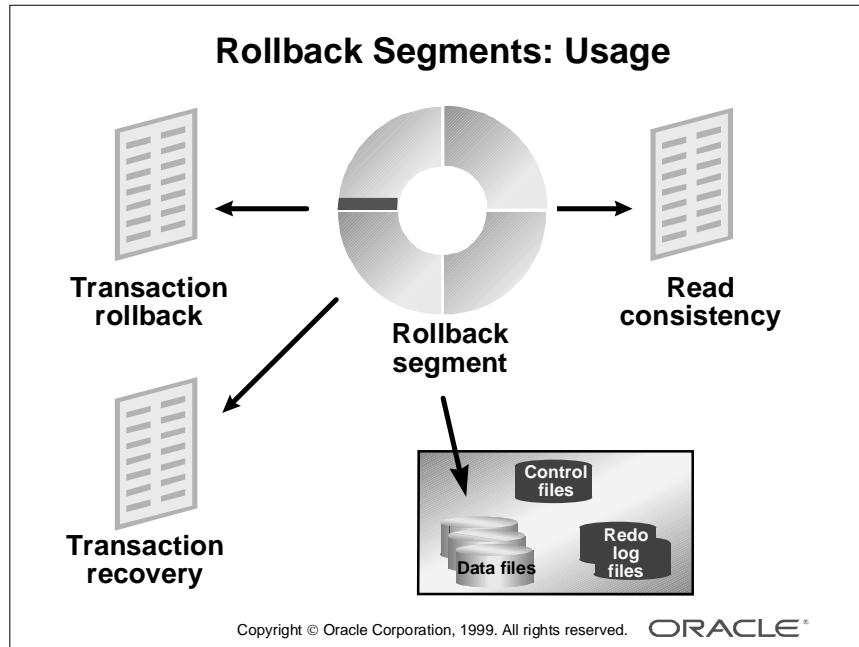
Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

### Objectives

Good rollback segment configuration is crucial to a well-tuned Oracle database. This lesson helps you to recognize and solve problems arising from inappropriate numbers or sizes of rollback segments.



## Rollback Segment Usage



### Transaction Rollback

When a transaction makes changes to a row in a table, the old image is saved in the rollback segment. If the transaction is rolled back, the value in the rollback segment is written back to the row, restoring the original value.

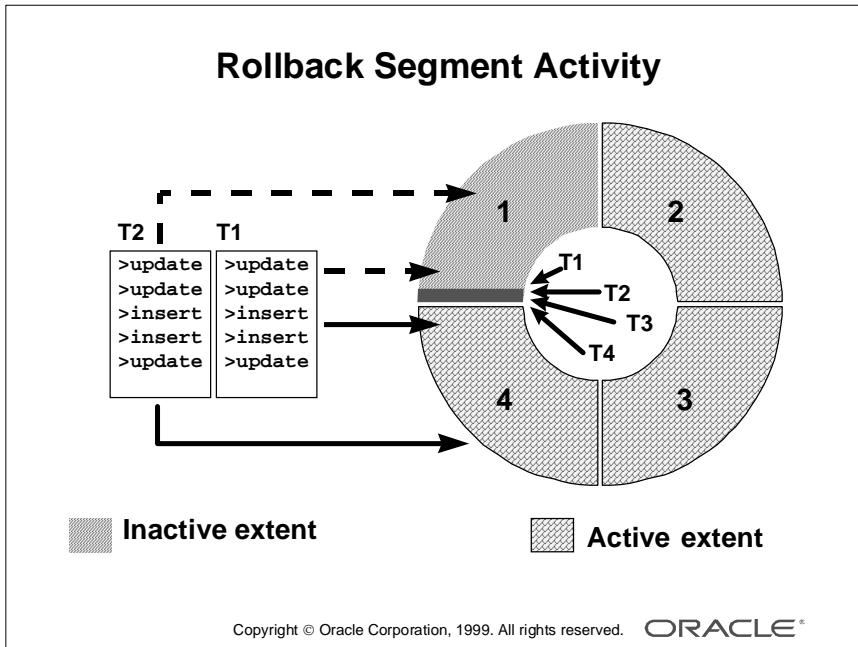
### Transaction Recovery

If the instance fails when transactions are in progress, the Oracle server needs to roll back the uncommitted changes when the database is opened again. This rollback is known as *transaction recovery*, and is only possible if changes made to the rollback segment are also protected by the redo log files.

### Read Consistency

When transactions are in progress, other users in the database should not see any uncommitted changes made by these transactions. In addition, a statement should not see any changes that were committed by others after the statement commenced execution. The old values in the rollback segments are also used to provide the readers with a consistent image for a given statement.

## Rollback Segment Activity



### Active and Inactive Extents

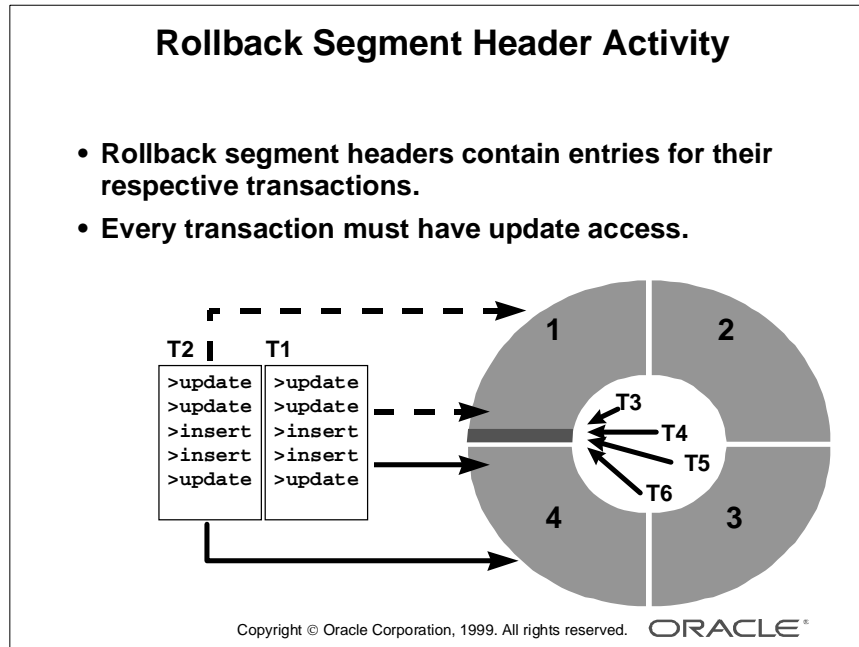
Transactions use extents of a rollback segment in an ordered, circular fashion, moving from one to the next after the current extent is full. A transaction writes a record to the current location in the rollback segment and advances the current pointer by the size of the record.

**Note:** More than one transaction can write to the same extent of a rollback segment. Each rollback segment block contains information from only one transaction.

### Rollback Segment Activity

The rollback activity can impact the hit ratio, because writing to rollback segments requires buffers in the cache (and that no direct writes exist), and this can overlay buffers containing blocks of data, thereby requiring more physical I/Os. Queries that require only data segments perform faster than those that access both the data segments and rollback segments.

## Rollback Segment Header Activity



### Rollback Segment Header Activity

The Oracle server keeps a transaction table in the header of every rollback segment. The rollback segment header activity controls the writing of changed data blocks to the rollback segments. The rollback segment header is a data block, and it is frequently modified; therefore, the rollback segment header block will remain in the data block buffer cache for long periods of time. Access to the rollback segment header block will thus increase the hit ratio for the application, even though it is not related to the data blocks.

### The Impact of Rollback Segment Header Activity

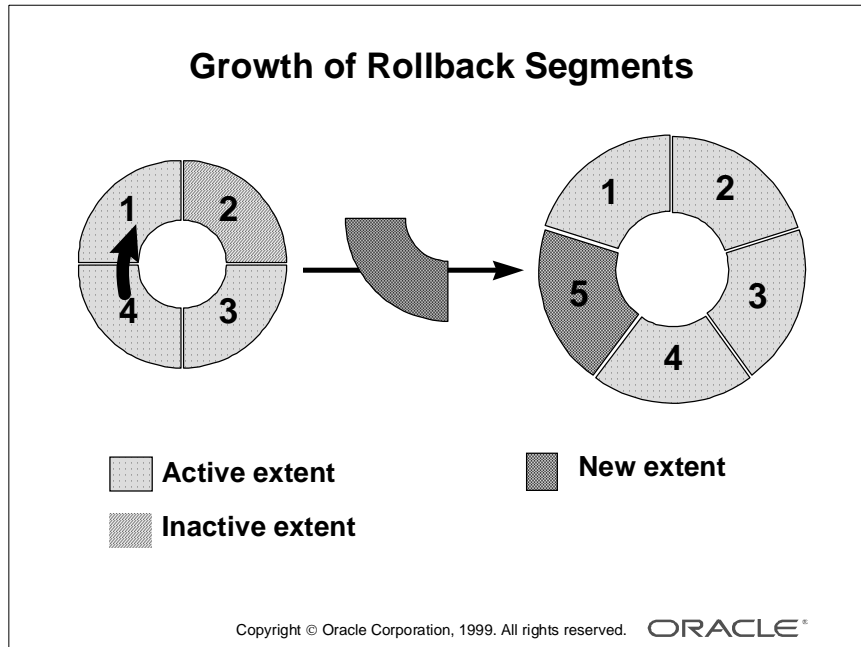
The impact of the rollback segment header activity on the cache hit ratio is important for OLTP systems that feature many small transactions.

Every transaction must have update access to the transaction table for its rollback segment. You need enough rollback segments to prevent transactions contending for the transaction table.

If you underestimate the number of rollback segments you need, performance degrades, and transactions may generate errors.

Overestimating numbers requires additional space.

## Growth of Rollback Segments



### Growth of Rollback Segments

The pointer or the head of the rollback segment moves to the next extent when the current extent is full. When the last extent that is currently available is full, the pointer can only move back to the beginning of the first extent if that extent is free. The pointer cannot skip over an extent and move to the second or any other extent. If the first extent is being used, the transaction allocates an additional extent for the rollback segment. This is called *extending*. Similarly, if the head tries to move into an active extent, the rollback segment allocates an additional extent.

### The Impact of Rollback Segment Extending

Avoid frequent extent allocation for rollback segments during normal running. Size the rollback segments so that they are large enough to hold the rollback entries for the transactions.

As with other objects, you should avoid dynamic space management.

If you underestimate the size of rollback segments, performance degrades, and transactions may generate errors.

Overestimating sizes could result in wasted space.

## Transaction Types

Read-only	Serializable
<pre>&gt;SET TRANSACTION   READ ONLY;  &gt;SELECT ... &gt;SELECT ... &gt;UPDATE ... ORA-01456: may not perform I/D/U operation inside a READ ONLY transaction</pre>	<pre>&gt;SET TRANSACTION   ISOLATION LEVEL   SERIALIZABLE; &gt;SELECT ... &gt;SELECT ... &gt;UPDATE ... &gt;UPDATE ... ORA-08177: can't serialize access for this transaction</pre>

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

### Read-Only and Read-Write Transactions

- The default state for all transactions is *statement-level read consistency*, or read-write transactions.
- You can establish *transaction-level read consistency*, or read-only transactions. All subsequent queries in a read-only transaction see only changes committed before the transaction began.

Export with `CONSISTENT = Y` sets the transaction to read-only.

No DML statements are allowed.

### Serializable and Read-Committed Isolation Level Transactions

- The default Oracle behavior for all transactions is *read-committed isolation level*. If the transaction contains DML that requires row locks that are held by another transaction, the DML statement waits until the row locks are released. Read-committed isolation provides more concurrency.
- The *serializable isolation level* provides the consistency that a read-only transaction provides, but also allows DML statements.

### **Serializable and Read-Committed Isolation Level Transactions (continued)**

If a transaction with a serializable isolation level attempts to execute a DML statement that updates rows that have been updated in a transaction that was uncommitted at the start of the serializable transaction, the DML statement fails.

This isolation level is suitable for environments where there is a relatively low chance that two concurrent transactions will modify the same rows, and the relatively long-running transactions are primarily read-only (large databases and short transactions updating a few rows).

### **Impact of Read-Only and Serializable Transactions**

All committed changes are kept in rollback segments until the read-only or serializable transactions are ended by a COMMIT. Application developers should take into account the cost of rolling back data.

## Tuning the Rollback Segments

### Tuning the Rollback Segments

- Transactions should never wait for access to rollback segments.
- Rollback segments should not extend during normal running.
- Users and utilities should try to use less rollback.
- No transaction should ever run out of rollback space.
- Readers should always see the read-consistent images they need.

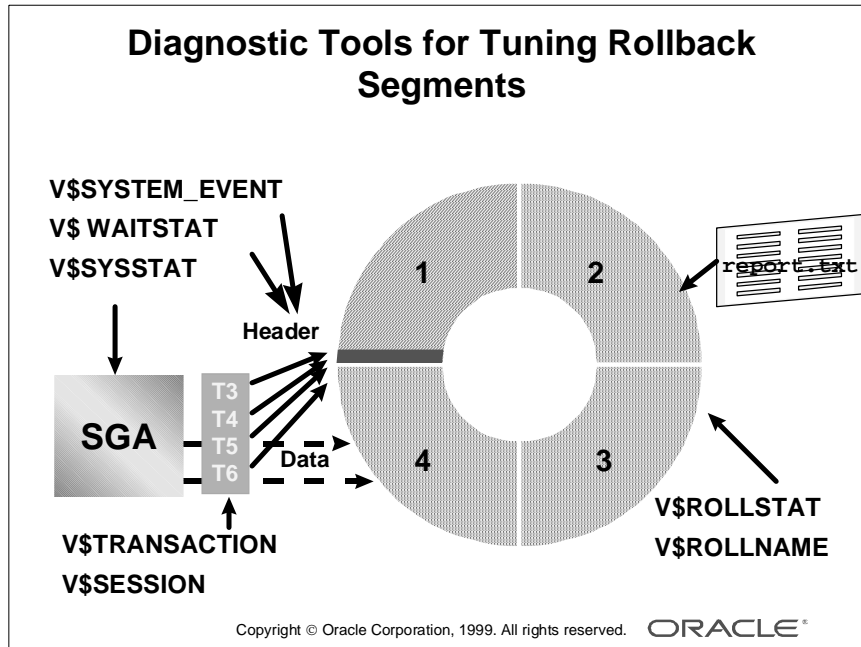
Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

### Tuning Goals

The goals in tuning the rollback segments are that:

- Transactions should never wait for access to rollback segments. This depends on having enough rollback segments to accommodate your database.
- Rollback segments should not extend during normal running. This depends on:
  - The number of extents per segment
  - The correct sizing of the extents
  - The appropriate number of rollback segments
  - A better use of utilities that can use less rollback
- No transaction, however large or exceptional, should ever run out of rollback space. This means that:
  - Rollback segments should be resized correctly.
  - Large transactions should be split into smaller transactions.
- Readers should always be able to see the read-consistent images they need; this depends on:
  - The number of rollback segments
  - The size of rollback segments

## Diagnostic Tools for Tuning Rollback Segments



### Dynamic Views and `report.txt` Output

Some of the tools you can use for tuning rollback segments are:

- The `V$ROLLNAME` view: Displays the name and number of the online rollback segments
- The `V$ROLLSTAT` view: Displays statistics for the activity for each online rollback segment:
  - The number of waits on the header transaction table
  - The volume of bytes written by the transactions
- The `V$SYSTEM_EVENT` view: The Undo Segment Tx Slot event shows waits for transaction slots and therefore contention on rollback segment headers
- The `V$WAITSTAT` view: Displays the cumulative statistics of waits on header blocks and data blocks of system and nonsystem rollback segments
- The file `report.txt` from `utlestat.sql`: Indicates the time spent waiting on the header transaction tables (These figures cover the period during which `UTLBSTAT` and `UTLESTAT` ran.)



### **Dynamic Views and report.txt Output (continued)**

- The V\$SYSSTAT view: Displays the number of consistent and data block gets (You can calculate the number of waits with the total number of requests for data.)
- The V\$TRANSACTION view: Displays the current transactions using rollback segments

**Note:** Oracle Enterprise Manager Diagnostics Pack application:

Performance Manager—>Contention—>Rollback (Nowait) Hit%

Performance Manager—>Database\_Instance—>System Statistics (Observe the details for rollback changes and rollback only (read-consistent).)

## Diagnosing Rollback Segment Header Contention

### Diagnosing Rollback Segment Header Contention

```
SQL> select sum(waits)* 100 /sum(gets) "Ratio",
2          sum(waits) "Waits", sum(gets) "Gets"
3  from v$rollstat;
```

Ratio	Waits	Gets
0.296736	5	1685

The ratio of the sum of waits to the sum of gets should be less than 1%.

If not, create more rollback segments.

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

### Checking Contention for Rollback Header Blocks

Any nonzero value in the WAITS column of the V\$ROLLSTAT view, the UNDO HEADER column of the V\$WAITSTAT view, or the Undo Segment Tx Slot event of the V\$\$SYSTEM\_EVENT view, indicates contention for rollback segment header blocks.

```
SQL> select class, count from v$waitstat
+6 where class like '%undo%';
```

CLASS	COUNT
system undo header	0
system undo block	0
undo header	7
undo block	0

The ratio of the sum of waits to the sum of gets should be nearly 0.

### Checking Contention for Rollback Header Blocks (continued)

- This section of the `report.txt` output displays the same information as the `WAITS` and `GETS` columns of the `V$ROLLSTAT` view, but the statistics cover a longer period of time:

UNDO_SEGMENT	TRANS_TBL_GETS	TRANS_TBL_WAITS
-----	-----	-----
0	8	0
1	749	118
2	878	95

- The Undo Segment Tx Slot event in the `V$SYSTEM_EVENT` view shows waits for transaction slots. Also note that contention does not necessarily mean header contention; the buffer cache must also be sized correctly.

### Guideline

If the ratio of `gets` to `waits` is greater than 5%, consider creating more rollback segments.

## Diagnosing Rollback Segment Contention

```
SQL> select value from v$sysstat
2  where name = 'consistent gets';
VALUE
-----
71563
```

The number of `waits` for any class should be less than 1% of the total number of requests.  
If not, create more rollback segments.

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

## Diagnosing Contention for Rollback Segments

Compare the number of `waits` for each class of block with the total number of requests for data over the same period of time.

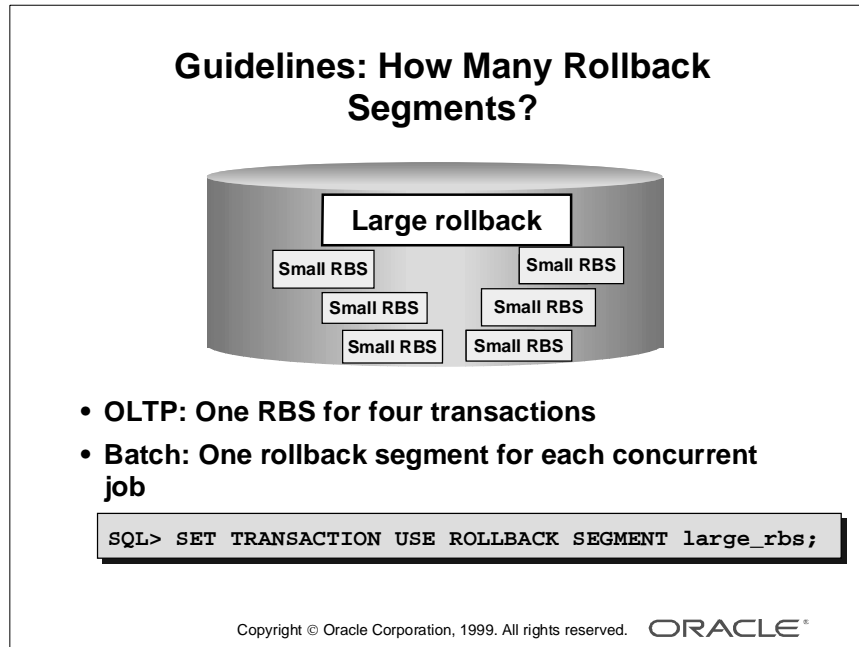
```
SQL> select class, count from v$waitstat
2  where class like '%undo%';
CLASS                                COUNT
-----
system undo header                   0
system undo block                    0
undo header                          7
undo block                           0
SQL> select value from v$sysstat
2  where name = 'consistent gets';
VALUE
-----
47774
```

The ratio of `waits` for any class should be less than 1% of the total number of requests.

### Guideline

If the ratio is greater than 1%, consider creating more rollback segments.

## Guidelines: How Many Rollback Segments?



### OLTP Transactions

OLTP applications are characterized by frequent concurrent transactions, each of which modifies a small amount of data. Assign small rollback segments to OLTP transactions.

The reasonable rule of thumb is one rollback segment for four transactions up to 200 users.

### Long Batch Transactions

Assign large rollback segments to transactions that modify large amounts of data. Such transactions generate large rollback entries. If a rollback entry does not fit into a rollback segment, the Oracle server extends the segment. Dynamic extension reduces performance and should be avoided whenever possible. Allow for the growth of the rollback segments by creating them in large or autoextensible tablespaces, with MAXEXTENTS set to UNLIMITED.

### **Long Batch Transactions (continued)**

For exceptionally long transactions, you may want to assign a large rollback segment using the following syntax:

```
SQL> SET TRANSACTION USE ROLLBACK SEGMENT large_rbs;
```

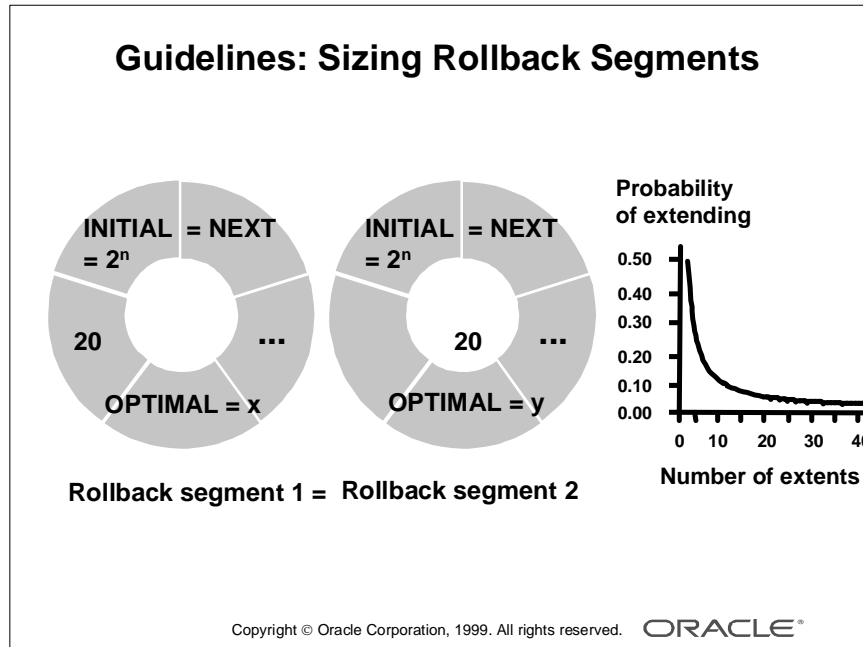
You can also use a supplied procedure:

```
SQL> EXECUTE dbms_transaction.use_rollback_segment('large_rbs');
```

Remember that any commit, explicit or implicit, ends the transaction. This means that the command may need to be included repeatedly.

**Note:** The SET TRANSACTION USE rollback segment must be the first one in the transaction.

## Guidelines: Sizing Rollback Segments



### Storage Parameters

Setting the right size for the rollback segments is significant for performance. It reduces dynamic extension and increases the chances that rollbacks will be in the buffer cache when needed.

- Choose the **INITIAL** storage parameter from the list: 8 KB, 16 KB, 32 KB, and 64 KB for small transactions; 128 KB, 256 KB, 512 KB, 1 MB, 2 MB, 4 MB, and so on for larger transactions. Choose a size that is large enough to prevent wrapping (an entry wraps into the next extent when it cannot find enough space in the current extent).
- Use the same value for **NEXT** as for **INITIAL**. Because **PCTINCREASE** is 0, all the other extents will have the same size as the **NEXT**.
- Make all your rollback segments the same size. If you do not, they are likely to become the same size over time, anyway. The smaller ones extend as they are used by large transactions.
- Set **MINEXTENTS** to 20. This makes it unlikely that the rollback segment needs to grab another extent, because the extent that it should move into is still being used by an active transaction.

### Tablespace Size

Leave enough free space in the rollback segments tablespace for a larger-than-usual transaction to be able to extend the rollback segment it is using. The **OPTIMAL** setting will later cause that rollback segment to shrink.

## Guidelines: Sizing Transaction Rollback Data

### Sizing Transaction Rollback Data

- Deletes are expensive.
- Inserts use minimal rollback space.
- Updates use rollback space depending on the number of columns.
- Index maintenance adds rollback.

```
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	2	11/16/95 10:26:39

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

### Transaction Statements

The number of bytes required to store information that is needed in case of rollback depends on the type of transaction being performed. Following are some guidelines to be considered in sizing the rollback data of transactions.

- Deletes are expensive for rollback segments; they need to store the actual row itself. If you can use TRUNCATE instead, performance improves.
- Inserts use little rollback space; only the ROWID is kept.
- The rollback space used for updates depends on how many columns are being updated.
- Indexed values generate more rollback, because the server process must change values in the index as well as in the table. For updates on indexed columns, the Oracle server records in the rollback segment the old data value, the old index value, and the new index value.

**Note:** Columns of LOB data type do not use rollback segment space for changes. They use their own segment space defined by the PCTVERSION clause setting: the actual data being processed.



## Sizing Transaction Rollback Data Volume

### Sizing Transaction Rollback Data

The number of bytes in rollback segments before execution of statements:

```
SQL> select usn,writes from v$rollstat;
      USN      WRITES
-----
       1      4738
SQL> @upd
```

After execution of statements:

```
SQL> select usn,writes from v$rollstat;
      USN      WRITES
-----
       1     1102686
```

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

### Sizing Transaction Rollback Data Volume

Estimate the size of the rollback segment by running the longest transaction expected and checking the size of the rollback segment. For the current transaction, get the number of blocks used in a rollback segment:

```
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
-----
SYSTEM                1075    04/01/98 07:07:14
```

Another way to estimate the volume of rollback data for a test transaction is to perform the following steps.

- 1 Before you execute the test transaction, display the current number of writes in the rollback segments:

```
SQL> select usn,writes from v$rollstat;
      USN      WRITES
-----
       0      1962
       1     1102686
       2      32538
       3     1226096
```

## Sizing Transaction Rollback Data Volume (continued)

### 2 Execute the test transaction:

```
SQL> update scott.s_emp set salary=1000;  
6560 rows updated.
```

### 3 Display the new number of writes in the rollback segments:

```
SQL> select usn,writes from v$rollstat;
```

USN	WRITES
0	1962
1	2232270
2	32538
3	1226096

You calculate the difference between the new and the old number of writes in rollback segment USN 1 to determine the amount of rollback data used for this test transaction.

## Guidelines: Using Less Rollback

### Using Less Rollback

- The design of the application should allow users to commit regularly.
- Developers should not code long transactions.

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

### Transactions

You may be able to reduce some rollback segment wastage by training users and developers to do the following:

- Users should commit work regularly so that their transactions do not lock others out of rollback segment extents.
- Developers should not code unnecessarily long transactions.

### Using Less Rollback

- **Import**
  - Set **COMMIT = Y**
  - Size the set of rows with **BUFFER**
- **Export: Set CONSISTENT=N**
- **SQL\*Loader: Set the COMMIT intervals with ROWS**

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

### Using Less Rollback in Data Loading Operations

In data loading operations that use import or SQL\*LOADER, you can control the size of the rollback segment used by using appropriate options.

#### Import

- Set **COMMIT = Y** to make sure that each set of inserted rows is committed as the import goes on.
- Size the set of rows with the **BUFFER\_SIZE** keyword.

#### Export

Set **CONSISTENT = N** prevents the transaction from being set as read-only, and thus using too much rollback segment space.

#### SQL\*Loader

For conventional path loading, set the **COMMIT** intervals with the **ROWS** keyword.

---

## Possible Problems

### Possible Problems

- Transaction fails for lack of rollback space
- “Snapshot too old” error occurs if:
  - The Interested Transaction List in the block being queried has been reused, and the SCN in the block is newer than the SCN at the start of the query.
  - The transaction slot in the rollback segment header has been reused.
  - The undo data in the rollback segment has been overlaid after a commit.

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

### Large Transactions

Large transactions may fail because the rollback segment cannot extend. Such a situation can occur when:

- The rollback segment has reached its MAXEXTENTS
- There is no room left in the tablespace for the rollback segment to expand

To resolve this problem you need bigger rollback segments or more space in the tablespace.

### Snapshot Too Old

If a query fails with the following error message, the rollback image needed for read consistency has probably been overwritten by an active transaction:

ORA-01555: snapshot too old (rollback segment too small)

Occasionally, you may get this error even if there are no other transactions active.

To resolve this error, you need more or bigger rollback segments.

## Summary

### Summary

In this lesson, you should have learned how to:

- Avoid contention for rollback segment headers
- Work out numbers and sizes of rollback segments
- Monitor the rollback space used by transactions
- Monitor the accurate value of the OPTIMAL storage parameter
- Identify possible rollback segment problems

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

---

## Quick Reference

Context	Reference
Initialization parameters	TRANSACTIONS
Dynamic performance views	V\$ROLLNAME V\$ROLLSTAT V\$WAITSTAT V\$SYSSTAT V\$SYSTEM_EVENT V\$TRANSACTION V\$SESSION
Data dictionary views	None
Commands	SET TRANSACTION READ ONLY; SET TRANSACTION ISOLATION LEVEL SERIALIZABLE; SET TRANSACTION USE ROLLBACK SEGMENT;
Packaged procedures and functions	DBMS_TRANSACTION.USE_ROLLBACK_SEGMENT
Scripts	None
Oracle Diagnostics Pack	Performance Manager





## **Monitoring and Detecting Lock Contention**

## Objectives

### Objectives

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

- **Define types and modes of locking**
- **List possible causes of contention**
- **Use Oracle utilities to detect lock contention**
- **Resolve contention in an emergency**
- **Prevent locking problems**
- **Recognize Oracle errors arising from deadlocks**

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

## Locking Mechanism

### Locking Mechanism

- Automatic management
- High level of data concurrency
  - Row-level locks for DML transactions
  - No locks required for queries
- Varying levels of data consistency
- Exclusive and share lock modes
- Locks held until commit or rollback occurs

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

### Locking Management

The Oracle server automatically manages locking. The Oracle server's default locking mechanisms lock data at the lowest level of restrictiveness to guarantee data consistency while allowing the highest degree of data concurrency.

You can modify the default mechanism by using the `init.ora` parameter `ROW_LOCKING`. The default value is `ALWAYS`. This leads the Oracle server to always lock at the lowest and least restrictive level—the row level—and not at the table level during DML statements. You can set the value to `INTENT`, which leads the Oracle server to lock at a more constraining level—the table level— except for a `SELECT FOR UPDATE` statement, for which a row-level lock is used.

## Data Concurrency

Locks are designed to allow a high level of *data concurrency*; that is, many users can safely access the same data at the same time. Data concurrency can involve two levels of locking: row level or none.

- Data Manipulation Language (DML) locking is at row level.

### Example

Transaction 1	Transaction 2
<pre>SQL&gt; update scott.s_emp   2  set salary=salary*1.1   3  where id= 24877; 1 row updated.</pre>	<pre>SQL&gt; update scott.s_emp   2  set salary=salary*1.1   3  where id= 24878; 1 row updated.</pre>

- A query holds no locks unless the user specifies that it should.

### Example

Transaction 1	Transaction 2
<pre>SQL&gt; UPDATE s_emp   2  SET salary = salary*1.1; 13120 rows updated.</pre>	<pre>SQL&gt; select salary   2  from s_emp   3  where id = 10;       SALARY -----       1000</pre>

## Data Consistency

The Oracle server also provides varying levels of *data consistency*; that is, the user sees a static picture of the data even if other users are changing it.

## Duration

Locks are held until a COMMIT or ROLLBACK occurs, or a transaction is terminated. If a transaction terminates abnormally, PMON cleans up the locks.

## Locking Modes

*Exclusive lock mode* prevents the associated resource from being shared with other transactions until the exclusive lock is released.

**Example:** Exclusive locks are set at row level for DML transactions:

Transaction 1	Transaction 2
<pre>SQL&gt; update scott.s_emp   2  set salary=salary*1.1   3  where id= 24877; 1 row updated.</pre>	<pre>SQL&gt; update scott.s_emp   2  set salary=salary*1.1   3  where id= 24877; Transaction 2 waits.</pre>

In *share lock mode*, several transactions can acquire share locks on the same resource.

**Example:** Shared locks are set at table level for DML transactions:

Transaction 1	Transaction 2
<pre>SQL&gt; update scott.s_emp   2  set salary=salary*1.1   3  where id= 24877; 1 row updated.</pre>	<pre>SQL&gt; update scott.s_emp   2  set salary=salary*1.1   3  where id= 24878; 1 row updated.</pre>

The two transactions update rows in the same table.

## Lock Duration

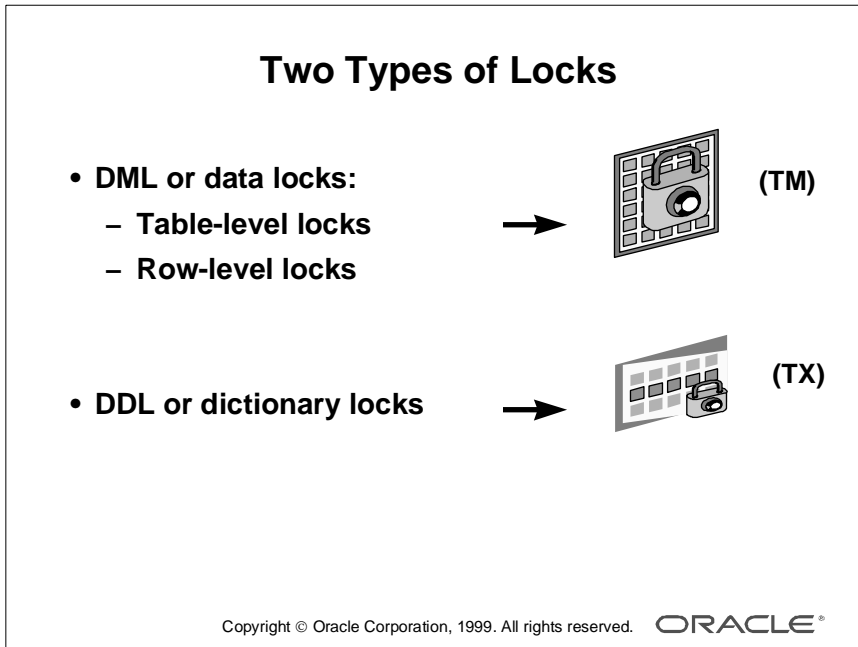
Transactions hold locks until they commit or roll back.

**Example**

Transaction 1	Transaction 2
<pre>SQL&gt; update scott.s_emp   2  set salary=salary*1.1   3  where id= 24877; 1 row updated. SQL&gt; commit; Commit complete.</pre>	<pre>SQL&gt; update scott.s_emp   2  set salary=salary*1.1   3  where id= 24877; <b>Transaction 2 waits.</b> 1 row updated.</pre>

As soon as Transaction 1 committed, Transaction 2 could update the row, because the transaction acquired the requested lock.

## Types of Locks



### DML Locks

You can use DML locks to guarantee the integrity of data that is accessed concurrently by multiple users. Locks prevent destructive interference by simultaneous, conflicting DML and DDL operations. DML locks provide two levels of locks.

- A *table-level lock* (TM type) is set for any DML transaction that modifies a table: INSERT, UPDATE, DELETE, SELECT...FOR UPDATE, or LOCK TABLE. The table lock prevents DDL operations that conflict with the transaction.

### Example

Transaction 1	Transaction 2
<pre>SQL&gt; UPDATE s_emp   2  SET salary=salary*1.1; 13120 rows updated.</pre>	<pre>SQL&gt; DROP TABLE s_emp; ERROR at line 1: ORA-00054: resource busy and acquire with NOWAIT specified</pre>

**DML Locks (continued)**

- A *row-level* lock (TX type) is automatically acquired for each row that is modified by INSERT, UPDATE, DELETE, or SELECT...FOR UPDATE. The row level lock assures that no other user can modify the same row at the same time. Therefore, there is no risk of a user modifying a row that is being modified and has not yet been committed by another user.

**Example**

Transaction 1	Transaction 2
SQL> update scott.s_emp 2 set salary=salary*1.1 3 where id= 24877; 1 row updated.	SQL> update scott.s_emp 2 set salary=salary*1.1 3 where id= 24877; Transaction 2 waits.

**DDL Locks**

A DDL lock protects the definition of a schema object while that object is acted upon or referred to by an ongoing DDL operation. The Oracle server automatically acquires a DDL lock to prevent any destructive interference with other DDL operations that might modify or reference the same schema object.

## DML Locks

### DML Locks

- A DML transaction acquires at least two locks:
  - A shared table lock
  - An exclusive row lock
- The enqueue mechanism keeps track of:
  - Users waiting for locks
  - The requested lock mode
  - The order in which users requested the lock

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

### DML Transactions Acquire at Least Two Locks

Two kinds of lock structures are used for DML statements (INSERT, UPDATE, DELETE, SELECT...FOR UPDATE):

- The transaction acquires a shared lock on the table, called *TM* lock, no matter what shared lock mode it is.
- The transaction acquires an exclusive lock, called *TX* lock, on the rows it is changing. Each row gets a lock byte turned on in the row header pointing to the Interested Transaction List (ITL) slot used by the transaction. The lock mode at row level can only be exclusive.



## Enqueue Mechanism

The Oracle server maintains all locks as *enqueues*. The enqueue mechanism can keep track of:

- Users waiting for locks held by other users
- The lock mode these users require
- The order in which users requested the lock

If three users want to update the same row at the same time, all of them get the shared table lock, but only one (the first) gets the row lock. The table locking mechanism keeps track of who holds the row lock and who waits for it.

You can increase the overall number of locks available for an instance by increasing the parameters `DML_LOCKS` and `ENQUEUE_RESOURCES`. This may be necessary in a parallel server configuration.

## Table Lock Modes

### Table Lock Modes

**Automatically acquired:**

- **Row Exclusive (RX): INSERT, UPDATE, DELETE**
- **Row Share (RS): SELECT... FOR UPDATE**

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

### Automatic Table Lock Modes

You often see the two TM table lock modes held by DML transactions, RX and RS. These are the table lock modes automatically assigned by the Oracle server for DML transactions.

The restrictiveness of a table lock's mode determines the modes in which other table locks on the same table can be obtained and held.

### Row Exclusive (RX)

- Permits other transactions to query, insert, update, delete, or lock other rows concurrently in the same table.
- Prevents other transactions from manually locking the table for exclusive reading or writing.

### Example

Transaction 1 (RX table lock held)	Transaction 2 (RX table lock held)
SQL> update s_emp 2 set salary=salary*1.1 3 where id= 24877; 1 row updated.	SQL> update s_emp 2 set salary=salary*1.1 3 where id= 24878; 1 row updated.

## Row Share (RS)

You can choose to lock rows during a query by using the `SELECT ... FOR UPDATE` statement. This prevents other transactions from manually locking the table for exclusive write access.

### Example

Transaction 1 (RS table lock held)	Transaction 2 (X table lock requested)
<pre>SQL&gt; select id,salary   2  from s_emp   3  where id=24877   4  for update;       ID      SALARY -----       24877      1100 SQL&gt; commit; Commit complete.</pre>	<pre>SQL&gt; lock table s_emp       2  in exclusive mode; Transaction 2 waits.  Table(s) Locked.</pre>

## Manual Table Lock Modes

### Table Lock Modes

**Manually acquired in LOCK statement:**

```
SQL> LOCK TABLE table_name IN mode_name MODE;
```

**Share (S)**

- No DML allowed
- Implicitly used for referential integrity

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

### Manual Lock Modes

The three other table lock modes are assigned manually by an explicit LOCK TABLE command:

```
SQL> LOCK TABLE s_emp IN exclusive MODE;  
Table(s) Locked.
```

Often there are good application reasons for explicit locking, but if you get lock contention, you may want to check with the developers.

Developers from backgrounds other than Oracle sometimes use unnecessarily high locking levels.

#### Share (S)

- Permits other transactions only to query the table (SELECT ... FOR UPDATE)
- Prevents any modification on the table

The SQL statements that implicitly get a Share lock involve referential integrity constraints. If there is no index on the foreign key column in the child table:

- Any DELETE on the parent holds a Share lock on the child
- Any UPDATE on referenced columns in the parent holds a Share lock on the child

### Share (S) (continued)

The reason for this behavior is that the parent row must not be deleted (or have its primary key updated) while child rows remain in any dependent table. The child table is locked to prevent updates and insertions that violate this rule.

#### Example

Transaction 1 (RX table lock held on s_dept, S table lock held on s_emp)	Transaction 2 (RX table lock requested)
<pre>SQL&gt; delete from s_dept   2  where id=60; 1 row deleted.  SQL&gt; commit; Commit complete.</pre>	<pre>SQL&gt; update s_emp   2  set salary=salary*1.1   3  where id=24877;  Transaction 2 waits.  1 row updated.</pre>

To avoid this behavior, set up indexes on foreign key columns in the referencing (child) table.

If you index the foreign key in the child table, however, the Oracle server can prevent changes to the child rows by locking the altered values in the index.

#### Example

Transaction 1 (RX table lock held)	Transaction 2 (RX table lock held)
<pre>SQL&gt; delete from s_dept   2  where id=60; 1 row deleted.</pre>	<pre>SQL&gt; create index i on s_emp   2  (dept_id); Index created.  SQL&gt; update s_emp   2  set salary=salary*1.1   3  where id=24877;  1 row updated.</pre>

## Table Lock Modes

**Manually acquired in LOCK statement:**

- **Share Row Exclusive (SRX)**
  - No DML or Share mode allowed
  - Implicitly used for referential integrity
- **Exclusive (X)**

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

### Share Row Exclusive (SRX)

This mode of lock prevents other statements from acquiring DML or manual share locks.

The SQL statement that implicitly gets a Share Row Exclusive lock again involves referential integrity. You need a Share Row Exclusive lock on the child when you DELETE from the parent table, under the following circumstances:

- A foreign key constraint includes ON DELETE CASCADE.
- There is no index on the foreign key column in the child table.

### Example

Transaction 1 (RX table lock held on s_dept, SRX table lock held on s_emp)	Transaction 2 (RX table lock requested)
<pre>SQL&gt; delete from s_dept       2  where id=60; 1 row deleted.  SQL&gt; commit; Commit complete.</pre>	<pre>SQL&gt; update s_emp       2  set salary=salary*1.1       3  where id=24877; Transaction 2 waits.  1 row updated.</pre>

Again, the solution is to index the foreign key column on the child table.

**Exclusive (X)**

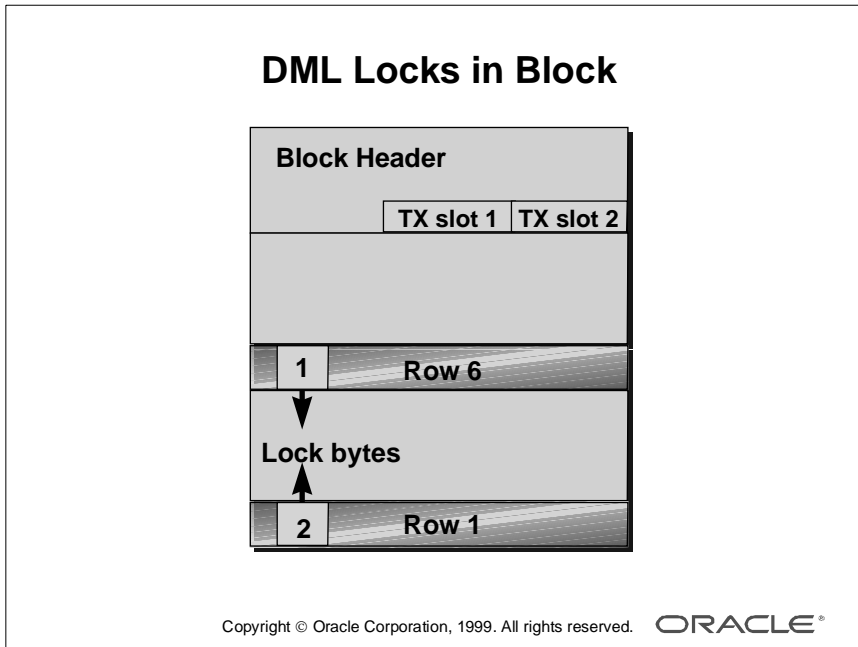
This is the highest level of table lock, thus the most restrictive mode. The exclusive lock:

- Permits other transactions only to query the table
- Prevents any type of DML statements and any manual lock mode

**Example**

Transaction 1 (X table lock held)	Transaction 2 (RS table lock requested)
SQL> lock table s_dept in exclusive mode; Table(s) Locked.	SQL> select * from s_dept 2 for update; Transaction 2 waits.

## Row-Level Lock in Block



### Row-Level Lock in a Block

The locking information for a row-level lock in a block is not cleared out when transactions commit. It is cleared when the next query reads the block. This is known as *delayed block cleanout*.

The query that does the cleaning must check the status of the transaction and the system change number (SCN) in the transaction table held in the rollback segment header.

Within blocks, the Oracle server keeps an identifier for each active transaction in the block header. At row level, the lock byte stores an identifier for the slot containing the transaction.

**Example** In the diagram shown on the slide, the transaction using slot 1 is locking row 6, and the transaction in slot 2 is locking row 1.



## DDL Locks

### DDL Locks

- **Exclusive DDL locks:**
  - DROP TABLE statement
  - ALTER TABLE statement
- **Shared DDL locks:**
  - CREATE PROCEDURE statement
  - AUDIT statement
- **Breakable parse locks:** Invalidating shared SQL area

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

### DDL Locks

You are unlikely to see contention for DDL locks, because they are held only briefly and are requested in NOWAIT mode. There are three types of DDL locks.

#### Exclusive DDL Locks

Some DDL statements, such as CREATE, ALTER, and DROP, must acquire an *exclusive lock* on the object they are working on.

Users cannot acquire an exclusive lock on the table if any other user holds any level of lock, so an ALTER TABLE statement fails if there are users with uncommitted transactions on that table.

#### Example

Transaction 1	Transaction 2
<pre>SQL&gt; UPDATE s_emp       2 SET salary=salary*1.1; 3120 rows updated.</pre>	<pre>SQL&gt; ALTER TABLE s_emp       2 DISABLE primary key; ORA-00054: resource busy and acquire with NOWAIT specified</pre>

### **Exclusive DDL Locks (continued)**

**Note:** Using locally managed tablespaces instead of dictionary managed tablespaces eliminates the contention for the ST (space transaction) lock because locally managed tablespaces do not update tables in the data dictionary when asking for space allocation.

### **Shared DDL Locks**

A *shared DDL lock* does not prevent similar DDL statements, or any DML, but it prevents another user from altering or dropping the referenced object.

Some statements (such as GRANT and CREATE PACKAGE) need a shared DDL lock on the objects they reference.

### **Breakable Parse Locks**

The *breakable parse lock* is used to check whether the statement should be invalidated if the object changes.

A statement or PL/SQL object in the library cache holds one of these locks for every object it references, until the statement is aged out of the shared pool.

You can think of this lock as a pointer. It will never cause waits or contention.

## Possible Causes of Lock Contention

### Possible Causes of Lock Contention

- Unnecessarily high locking levels
- Uncommitted changes
- Other products imposing higher-level locks

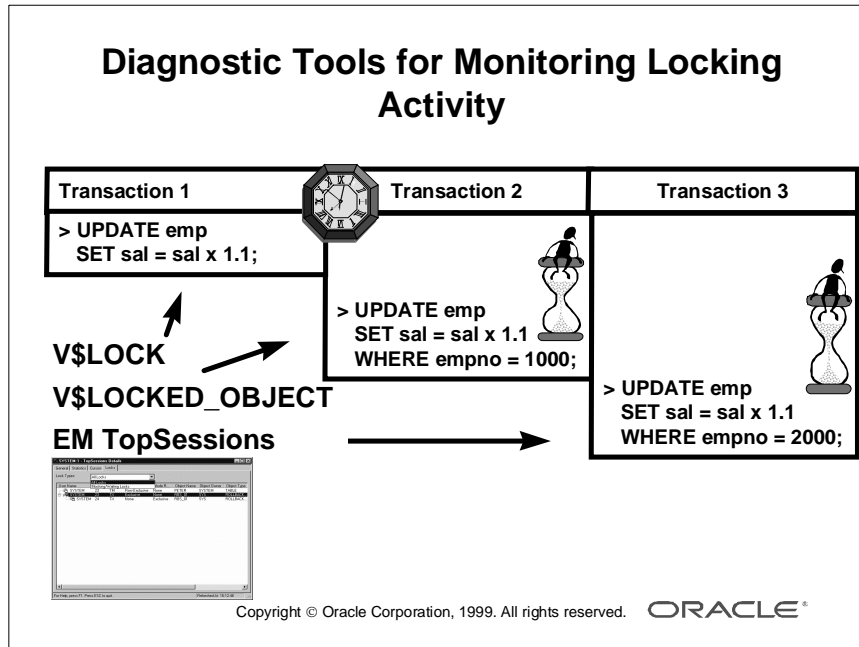
Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE®

### Lock Contention

The Oracle server locks are inexpensive and efficient, and most sites do not have problems with locking. If locks do cause contention, it is often because:

- Developers have coded in unnecessarily high locking levels
- Developers have coded in unnecessarily long transactions
- Users are not committing changes when they should
- The application uses the Oracle server in conjunction with other products that impose higher locking levels

## Diagnostic Tools for Monitoring Locking Activity



### The V\$LOCK View

The V\$LOCK view provides information about the locks currently held in the instance

Lock Type	ID1
TX	Rollback segment number and slot number
TM	ID of the table being modified

**Example** To find the table name that corresponds to a particular Resource ID 1 of the V\$LOCK view:

```
SQL> SELECT owner, object_id, object_name, object_type,
2         v$lock.type
3 FROM dba_objects, v$lock
4 WHERE object_id = v$lock.id1 and object_name = table_name;
```

Any process that is blocking others is likely to be holding a lock obtained by a user application. The locks acquired by user applications are:

- Table locks (TM)
- Row-level locks (TX)

Other locks not listed here are system locks that are held only briefly.

**The V\$LOCKED\_OBJECT View**

Lock Type	ID1
XIDUSN	Rollback segment number
OBJECT_ID	ID of the object being modified
SESSION_ID	ID of the session locking the object
ORACLE_USERNAME	
LOCKED_MODE	

**Example** To find the table name that corresponds to a particular OBJECT\_ID of the V\$LOCKED\_OBJECT view:

```
SQL> select xidusn, object_id, session_id, locked_mode
```

```
2 from v$locked_object;
```

```
      XIDUSN OBJECT_ID SESSION_ID LOCKED_MODE
```

```
-----
```

```
          3      2711          9          3
```

```
          0      2711          7          3
```

```
SQL> select object_name
```

```
2 from dba_objects
```

```
3 where object_id = 2711;
```

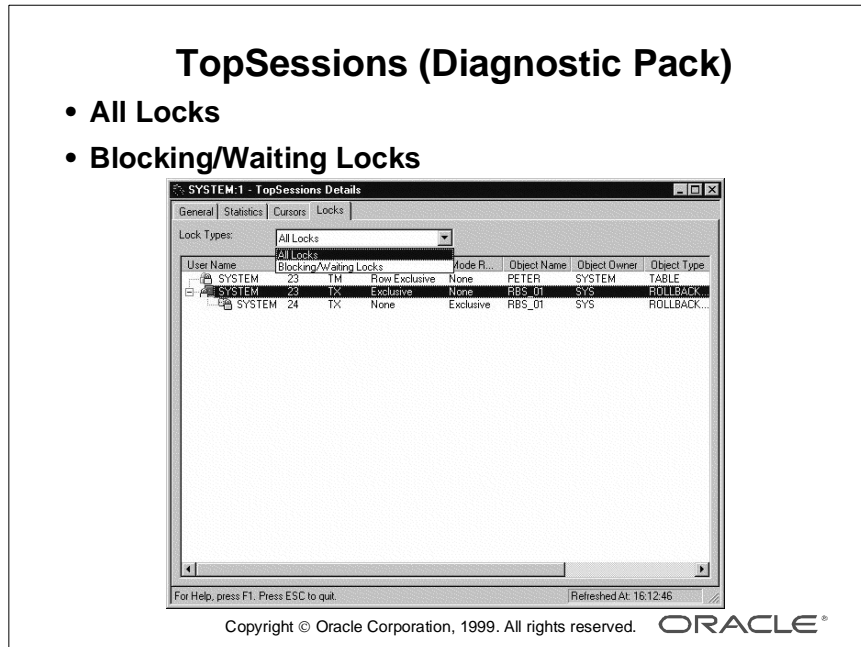
```
OBJECT_NAME
```

```
-----
```

```
S_EMP
```

If the value of XIDUSN is 0, the corresponding SESSION\_ID is requesting and waiting for the lock being held by the SESSION\_ID, for which the XIDUSN has a value different from 0.

## TopSessions (Diagnostic Pack)



### All Locks Display Option

In the All Locks display option, all locks held or requested by background and user transactions are displayed, even if some of them are not requested by others.

The columns of the list are:

- User Name: Oracle username for this session
- Session ID: Oracle session ID for this session
- Lock Type:
  - MR (Media Recovery), RT (Redo Thread), UN (User Name), UL (PL/SQL User Lock), DX (Distributed Xaction), CF (Control File), IS (Instance State), FS (File Set), IR (Instance Recovery), ST (Disk Space Transaction), TS (Temp Segment), IV (Library Cache Invalidation), LS (Log Start or Switch), RW (Row Wait), SQ (Sequence Number), TE (Extend Table), TT (Temp Table)
  - TX (Transaction: row-level lock)
  - TM (Table-level lock: DML)
- Mode Held: Mode in which the lock is currently held by the session: None, Row Exclusive, Share, Share Row Exclusive, Exclusive
- Mode Requested: Mode in which the lock is being requested by the process: None, Row Exclusive, Share, Share Row Exclusive, Exclusive

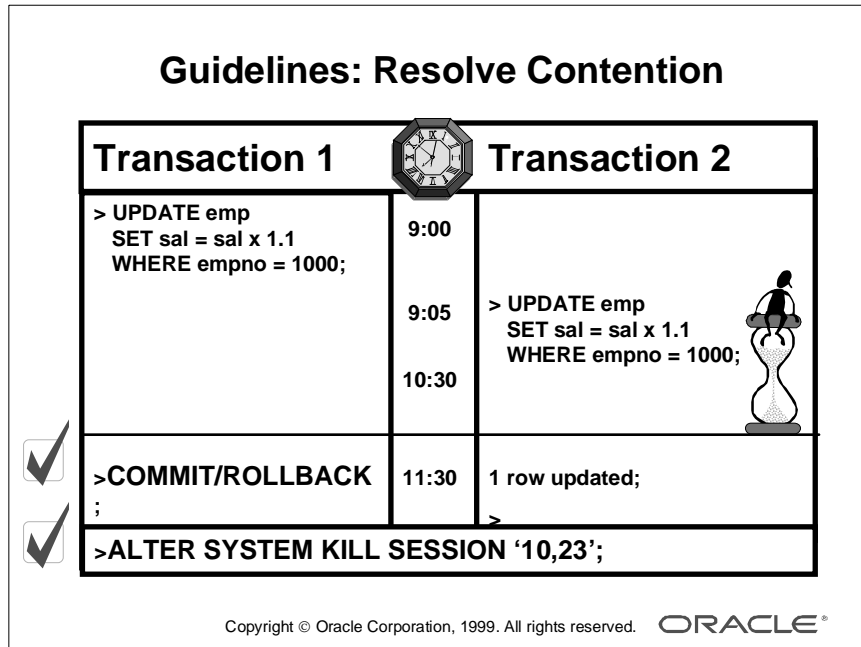
**All Locks Display Option (continued)**

- Object Name: Name of the object, such as table or view, that is being locked (If the lock type is TM, the object is a table or view. If the lock type is TX, the object is a rollback segment.)
- Object Owner: Owner of the object that has been locked by the session (The session user, listed in the Username field of the General page in the Session Details window, may be different from the owner of the object that is being locked.)
- Object Type
- Resource ID: For certain types of locks, this value is the object ID or rollback segment number
- Resource ID 2: Undocumented

**Note:** The same information is also displayed in the Oracle Diagnostics Pack TopSessions application. You can drill down on individual sessions to get the details view.

You can use the Blocking/Waiting Locks option in Locks tab in the detail view to see only those transactions that hold locks requested by others.

## Guidelines: Resolve Contention



### Terminating Sessions

If a user is holding a lock required by another user, you can:

- Contact the holder and ask this user to commit or roll back the transaction.
- As a last resort, terminate the Oracle user session; this will roll back the transaction and release locks

Any of the monitoring methods detailed above will give you the session identifier for the user.

You can terminate user sessions with:

- The ALTER SYSTEM KILL SESSION SQL command:

```
SQL> select sid,serial#,username
2  from v$session
3  where type='USER';
```

SID	SERIAL#	USERNAME
8	122	SYSTEM
10	23	SCOTT

```
SQL> alter system kill session '10,23';
System altered.
```

- Oracle TopSessions:
  - 1 Select the session to be killed.
  - 2 Select Locks—>Kill Session—>OK.

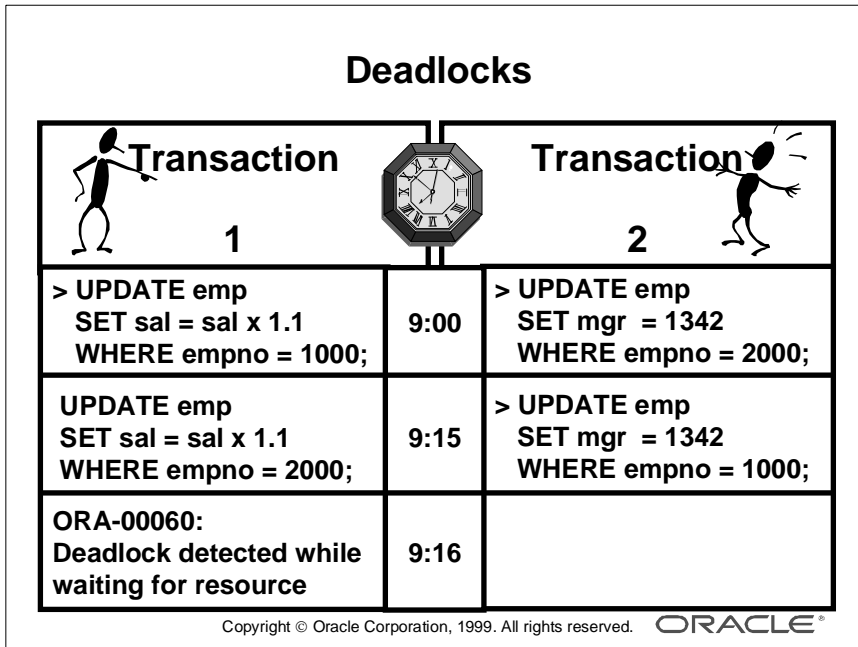


### **Which Row Is Causing Contention?**

If you need to know which row is causing contention, the V\$SESSION view contains the following columns:

- ROW\_WAIT\_BLOCK#
- ROW\_WAIT\_ROW#
- ROW\_WAIT\_FILE#
- ROW\_WAIT\_OBJ#

## Deadlocks



### Resolving Deadlocks

A *deadlock* can arise when two or more users wait for data locked by each other.

The Oracle server automatically detects and resolves deadlocks by rolling back the statement that detected the deadlock.

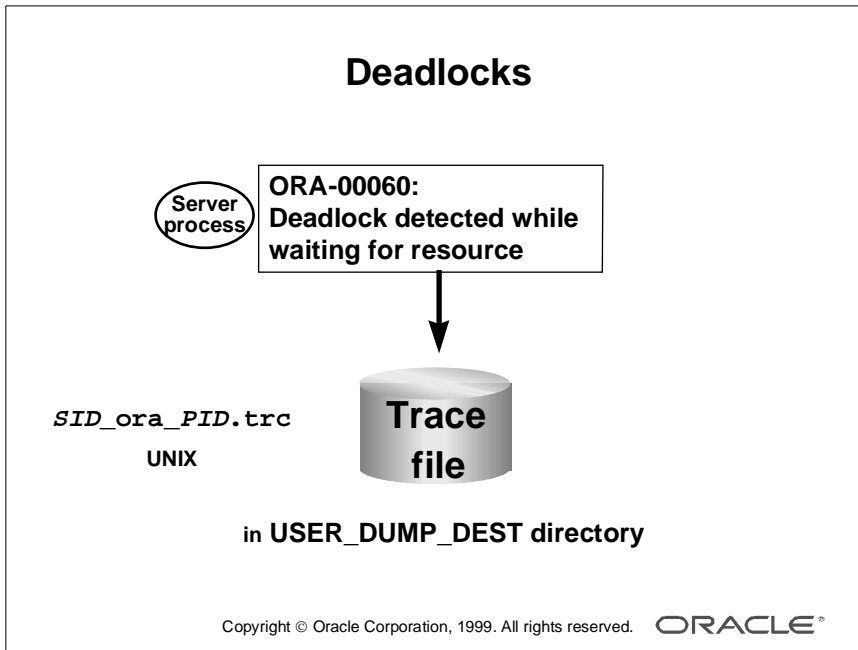
Transaction 1	Time	Transaction 2
SQL> update scott.s_emp 2 set salary=salary*1.1 3 where id= 24877; 1 row updated.	1	SQL> update scott.s_emp 2 set salary=salary*1.1 3 where id= 24876; 1 row updated.
SQL> update scott.s_emp 2 set salary=salary*1.1 3 where id= 24876; Transaction 1 waits.	2	SQL> update scott.s_emp 2 set salary=salary*1.1 3 where id= 24877; Transaction 2 waits.
ORA-00060: deadlock detected while waiting for resource	3	

**Resolving Deadlocks (continued)**

Assuming the second update in Transaction 1 detects the deadlock, the Oracle server rolls back that statement and returns the message. Although the statement that caused the deadlock is rolled back, the transaction is not, and you should receive an ORA-00060 error. Your next action should be to roll back the remainder of the transaction.

**Technical Note**

Deadlocks most often occur when transactions explicitly override the default locking of the Oracle server. Once detected, distributed deadlocks are handled in the same way as nondistributed deadlocks.



### Trace File

A deadlock situation is recorded in a trace file in the USER\_DUMP\_DEST directory. You should monitor trace files for deadlock errors to determine if there are problems with the application. The trace file contains the row IDs of the locking rows.

## Summary

### Summary

In this lesson, you should have learned that:

- Queries do not lock data, unless specified in the query
- DML statements use row-level and table-level locks on tables
- Exclusive locks are rarely used
- You can monitor locks using:
  - V\$LOCK, V\$LOCKED\_OBJECT
  - Oracle Enterprise Manager TopSessions

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

## Quick Reference

Context	Reference
Initialization parameters	DML_LOCKS ENQUEUE_RESOURCES USER_DUMP_DEST
Dynamic performance views	V\$LOCK V\$LOCKED_OBJECT V\$SESSION
Data dictionary views	None
Commands	LOCK TABLE IN lock MODE; ALTER TABLE table_name DISABLE TABLE LOCK; ALTER SYSTEM KILL SESSION 'sid,serial#';
Packaged procedures and functions	None
Scripts	None
Oracle Diagnostics Pack	TopSessions Performance Manager

## Lock Matrix

Type of Request	Lock Mode	Lock Target	Conflicts/Notes
Select	None	None	No locks on reads
Lock table in Row Share mode	Mode 2	TM(RS) lock on table	Mode 6, so no exclusive DDL (This is the least restrictive lock.)
Lock table partition in Row Share mode	Mode 2 Mode 2	TM(RS) lock on table TM(RS) lock on table partition	Mode 6, so no exclusive DDL (This is the least restrictive lock.)
Select for update	Mode 2 Mode 2  Mode 6	TM(RS) lock on table TM(RS) lock on each table partition TX lock on RBS TX slot	Mode 6 and any selects for update or DML on same rows No exclusive DDL

Type of Request	Lock Mode	Lock Target	Conflicts/Notes
Lock table in Row Exclusive mode	Mode 3	TM(RX) lock on table	Mode 4,5,6 (Updates allowed, as Mode 3 does not conflict with mode 3.) No share locks and no referential integrity locks
Lock table partition in Row Exclusive mode	Mode 3 Mode 3	TM(RX) lock on table TM(RX) lock on table partition	Mode 4,5,6 on the same partition Updates allowed, as Mode 3 does not conflict with mode 3 No share locks and no referential integrity locks
DML (up/ins/del)	Mode 3 Mode 6	TM(RX) lock on table TX lock on RBS TX slot	Mode 4,5,6 Select for update or DML on same rows No share locks and no referential integrity locks
DML (up/ins/del) on a partitioned table	Mode 3 Mode 3 Mode 6	TM(RX) lock on table TM(RX) lock on each table partition owning the updated rows TX lock on RBS TX slot	Mode 4,5,6 Select for update or DML on same rows No share locks and no referential integrity locks
Lock table in Share mode	Mode 4	TM(S) lock on table	Mode 3,5,6 Allows Select for Update and other Share locks No possible ORA 1555 on locked table

Type of Request	Lock Mode	Lock Target	Conflicts/Notes
Lock table partition in Share mode	Mode 2 Mode 4	TM(RS) lock on table TM(S) lock on table partition	Mode 3,5,6 on the same partition Allows Select for Update and other Share locks No possible ORA 1555 on locked table
Delete from/update to parent table with referential integrity constraint on child table, <i>no index on FK column in child table</i> , and no ON DELETE CASCADE in the FK constraint	Mode 4 Mode 3 Mode 6	TM(S) lock on child TM(RX) lock on parent TX lock on RBS TX slot	Mode 3,5,6 on child Allows Select for Update and Share lock on child No possible ORA 1555 on child Mode 4,5,6 on parent Select for update or DML on same rows against parent



**SQL Issues and Tuning  
Considerations for  
Different Applications**

## Objectives

### Objectives

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

- **Identify the role of the DBA in application tuning**
- **Use optimizer modes to enhance SQL statement performance**
- **Manage stored outlines to store execution paths as a series of hints**

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

## Objectives

- Use the available data access methods to tune the physical design of the database
- Identify the demands of online transaction processing (OLTP) systems
- Identify the demands of decision support systems (DSS)
- Reconfigure systems on a temporary basis for particular needs

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

## The Role of the DBA

### The Role of the DBA

- **Application tuning is the most important part of tuning.**
- **DBAs may not be directly involved in application tuning.**
- **DBAs must be familiar with the impact that poorly written SQL statements can have upon database performance.**

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

### Importance of Application Tuning

Application design and application tuning provides the greatest performance benefits. The method in which data is selected and the amount of data that is selected has serious implications on application performance if the statements that execute those operations are not properly written.

### Database Administrator Tasks

As a database administrator (DBA), you may not be directly involved in the tuning of an application, because application developers are usually responsible for developing applications and writing SQL statements. However, as a DBA you need to be aware of the impact that poorly written SQL statements can have upon the database environment. You should be able to provide assistance with application tuning and to readily identify inefficient SQL statements. You should also be aware of optimizer plan stability, implemented through stored outlines, and query rewrites using materialized views and dimensions.

## Diagnostic Tools Overview

### Diagnostic Tools Overview

- **EXPLAIN PLAN**
- **SQL Trace and TKPROF**
- **SQL\*Plus AUTOTRACE**
- **Oracle SQL Analyze**

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

### Overview of Diagnostic Tools

Numerous diagnostic tools are available for evaluating the performance of SQL statements and PL/SQL modules. Each provides a developer or DBA with a varying degree of information.

- **EXPLAIN PLAN:** This is executed within a session for a SQL statement.
- **SQL Trace:** This utility provides detailed information regarding the execution of SQL statements.
- **TKPROF:** This is an operating system utility that takes the output from a SQLTRACE session and formats it into a readable format.
- **AUTOTRACE:** This is a SQL\*Plus feature. Autotrace generates an execution plan for a SQL statement and provides statistics relative to the processing of that statement.
- **Oracle SQL Analyze:** This is part of the Oracle Enterprise Manager Tuning Pack, that provides a powerful GUI interface for tuning SQL statements.

## The EXPLAIN PLAN Statement

### Explain Plan

- Can be used without tracing
- To use the explain plan:
  1. Create `PLAN_TABLE` with `utlxplan.sql`.

```
SQL> @$ORACLE_HOME/rdbms/admin/utlxplan
```
  2. Run the `EXPLAIN PLAN SQL` command.
  3. Query `PLAN_TABLE` to display the execution plans.

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

### Using the Plan Table to get the Execution Plan

You can use the `EXPLAIN PLAN` statement in SQL\*Plus without using SQL Trace.

To use `EXPLAIN PLAN`, run the script `utlxplan.sql` (in the `ORACLE_HOME/rdbms/admin` directory) and create a table called `PLAN_TABLE`. The most useful columns in this table are `OPERATION`, `OPTIONS`, and `OBJECT_NAME`.

To explain the plan for a query, use the following syntax:

```
EXPLAIN PLAN
[SET STATEMENT_ID = '...'] [INTO my_plan_table]
FOR
SELECT ...
```

Then query `PLAN_TABLE` to check the execution plan.

`PLAN_TABLE` shows you how the statement would be executed if you chose to run it at that moment. Remember that if you make changes before running the statement (creating an index, for example), the actual execution may be different.

Also, if you do not use a `STATEMENT_ID` in the explain plan, you may want to truncate `PLAN_TABLE` prior to generating another execution plan.

**Note:** For additional information on the columns in `PLAN_TABLE`, see the *Oracle8i Tuning, Release 8.1* manual.

## SQL Trace and TKPROF

### SQL Trace and TKPROF

1. Set the initialization parameters.
2. Invoke SQL Trace.
3. Run the application.
4. Turn off SQL Trace.
5. Format the trace file with TKPROF.
6. Interpret the output.

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

### Using SQL Trace and TKPROF

A particular sequence of steps is necessary to properly diagnose SQL statement performance with SQL Trace and TKPROF:

- 1 The first step is to ensure appropriate initialization parameters. These may be set at the instance level; some parameters may also be set at the session level.
- 2 You must invoke SQL Trace at either the instance or session level. Generally, it is better if it is invoked at a session level.
- 3 Run the application or SQL statement you want to diagnose.
- 4 Turn off SQL Trace. This is necessary to properly close the trace file at the operating system level.
- 5 Use TKPROF to format the trace file generated during the trace session. Unless the output file is formatted, it will be very difficult to interpret the results.
- 6 Use the output from TKPROF to diagnose the performance of the SQL statement.

## Initialization Parameters

Two parameters in the `init.ora` file control the size and destination of the output file from the SQL trace facility:

```
max_dump_file_size = n
```

This parameter is measured in bytes if K or M is specified, otherwise the number represents operating system blocks. Default value: 10,000 operating system blocks.

When a trace file exceeds the size defined by the parameter value, the following message appears at the end of the file: `*** Trace file full ***`

The following parameter determines the trace file destination:

```
user_dump_dest = directory
```

You must set a third parameter to get timing information:

```
timed_statistics = TRUE
```

The timing statistics have a resolution of one hundredth of a second.

You can set the `TIMED_STATISTICS` parameter dynamically at the session level, using the `ALTER SESSION` command.



## Enabling and Disabling SQL Trace

### Enabling and Disabling SQL Trace

- **Instance level:**  
`SQL_TRACE = {TRUE|FALSE}`
- **Session level:**

```
SQL> alter session set SQL_TRACE = {true|false};

SQL> execute DBMS_SESSION.SET_SQL_TRACE
2      ({true|false});

SQL> execute DBMS_SYSTEM.SET_SQL_TRACE_IN_SESSION
2      (session_id, serial_id, {true|false});
```

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

### SQL Trace

SQL Trace can be enabled or disabled using different methods at either the instance or the session level.

#### Instance Level

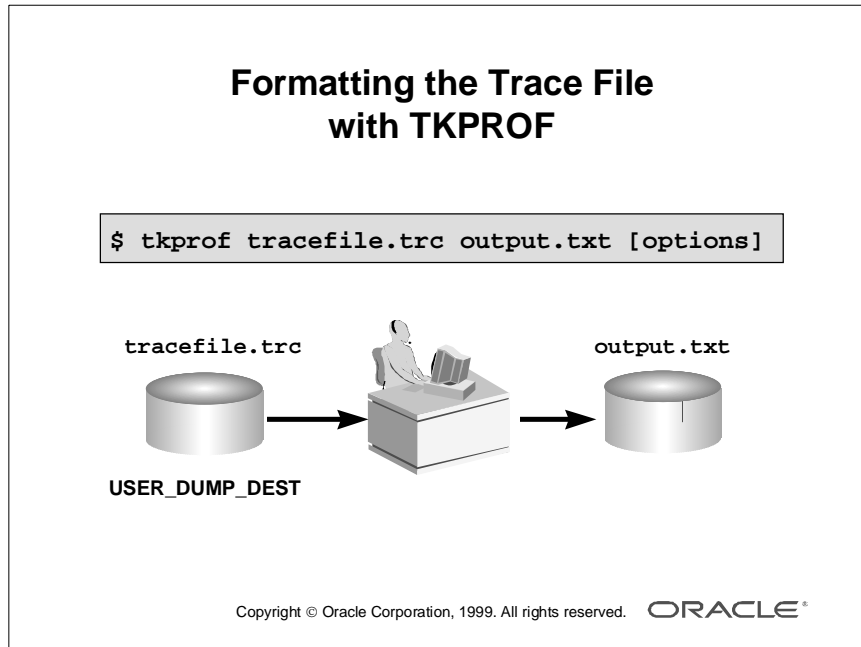
Setting the `SQL_TRACE` parameter at the instance level is one method to enable tracing. However, doing this requires that the instance be shut down, then restarted when tracing is no longer needed. This also imposes a significant performance hit, because all sessions for the instance will be traced.

#### Session Level

Session-level tracing results in less of an overall performance hit, because specific sessions can be traced. Three methods for enabling or disabling SQL Trace are:

- Using the `ALTER SESSION` command, which results in tracing for the duration of the session or until the value is set to `FALSE`
- Using the `DBMS_SESSION.SET_SQL_TRACE` procedure for the session
- Using the `DBMS_SYSTEM.SET_SQL_TRACE_IN_SESSION` procedure to enable tracing for a user other than the current user at the session level

## Formatting the Trace File with TKPROF



### Using TKPROF Command

Use TKPROF to format the trace file into a readable output:

```
tkprof tracefile outputfile [sort=option] [print=n]  
[explain=username/password] [insert=filename] [sys=NO]  
[record=filename] [table=schema.tablename]
```

The trace file is created in the directory specified by the `USER_DUMP_DEST` parameter and the output is placed in the directory specified by the output file name.

SQL Trace also gathers statistics for recursive SQL statements. You cannot directly affect the amount of recursive SQL that the server carries out, so these figures are not particularly useful in themselves. Use the TKPROF command option `SYS=NO` to suppress the output relating to the recursive SQL.

When you specify the `EXPLAIN` parameter, TKPROF logs on to the database with the username and password provided. It then works out the access path for each SQL statement traced and includes that in the output. Because TKPROF logs on to the database, it uses the information available at the time that TKPROF is run, not at the time the trace statistics were produced. This could make a difference if, for example, an index has been created or dropped since tracing the statement.

TKPROF also reports Library Cache Misses: This indicates the number of times that the statement was not found in the library cache.

## TKPROF Options

Option	Description
TRACEFILE	The name of the trace output file
OUTPUTFILE	The name of the formatted file
SORT= <i>option</i>	The order in which to sort the statements
PRINT= <i>n</i>	Print the first <i>n</i> statements
EXPLAIN= <i>user/password</i>	Run EXPLAIN PLAN in the specified username
INSERT= <i>filename</i>	Generate INSERT statements
SYS=NO	Ignore recursive SQL statements run as user <i>sys</i>
AGGREGATE=[Y N]	If you specify AGGREGATE = NO, TKPROF does not aggregate multiple users of the same SQL text
RECORD= <i>filename</i>	Record statements found in the trace file
TABLE= <i>schema.tablename</i>	Put execution plan into specified table (rather than the default PLAN_TABLE)

You can type `tkprof` at the operating system to get a listing of all the available options and output.

**Note:** The sort options are the following:

Sort Option	Description
prscnt, execnt, fchcnt	Number of times parse, execute, fetch were called
prscpu, execpu, fchcpu	CPU time parsing, executing, fetching
prselat, exelat, fchelat	Elapsed time parsing, executing, fetching
prsdsk, exedsk, fchdsk	Number of disk reads during parse, execute, fetch
prsqry, exeqry, fchqry	Number of buffers for consistent read during parse, execute, fetch
prscu, execu, fchcu	Number of buffers for current read during parse, execute, fetch
prsmis, exemis	Number of misses in library cache during parse, execute
exerow, fchrow	Number of rows processed during execute, fetch
userid	User ID of user that parsed the cursor

## TKPROF Statistics

### TKPROF Statistics

- **Count:** Number of execution calls
- **CPU:** CPU seconds used
- **Elapsed:** Total elapsed time
- **Disk:** Physical reads
- **Query:** Logical reads for consistent read
- **Current:** Logical reads in current mode
- **Rows:** Rows processed

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

### TKPROF Statistics Info

Statistic	Meaning
Count	The times the statement was parsed or executed and the number of fetch calls issued for the statement
CPU	Processing time for each phase, in seconds (If the statement was found in the shared pool, this is 0 for the parse phase.)
Elapsed	Elapsed time, in seconds (This is not usually very helpful, because other processes affect elapsed time.)
Disk	Physical data blocks read from the database files (This statistic may be quite low if the data was buffered.)
Query	Logical buffers retrieved for consistent read (usually for SELECT statements)
Current	Logical buffers retrieved in current mode (usually for DML statements)
Rows	Rows processed by the outer statement (For SELECT statements, this is shown for the fetch phase; for DML statements it is shown for the execute phase.)

The sum of Query and Current is the total number of logical buffers accessed.

## SQL\*Plus AUTOTRACE

### SQL\*Plus AUTOTRACE

- Create `PLAN_TABLE`
- Run `plustrce.sql` from the `ORACLE_HOME/sqlplus/admin` directory

```
SQL> @ORACLE_HOME/sqlplus/admin/plustrce.sql  
SQL> grant plustrace to scott;
```

- AUTOTRACE syntax:

```
set autotrace [ off | on | traceonly ]  
               [ explain | statistics ]
```

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

### The AUTOTRACE Feature

SQL\*Plus AUTOTRACE can be used instead of SQL Trace. The advantage in using AUTOTRACE is that you do not have to format a trace file and it will automatically display the execution plan for the SQL statement.

However, AUTOTRACE does parse and execute the statement, while the explain plan only parses the statement.

The steps for using AUTOTRACE are:

- 1 Create `PLAN_TABLE` using the `utlxplan.sql` script.
- 2 Create the `plustrace` role by executing the `plustrce.sql` script. This grants select privileges on V\$ views to the role, and also grants the `plustrace` role to the DBA role. Grant the `plustrace` role to users that do not have the DBA role.
- 3 Use SQL Trace.
  - ON: Produces the result set and the explain plan, and lists statistics.
  - TRACEONLY: Displays the explain plan and the statistics; you will not see the result set, although the statement is executed.
  - ON EXPLAIN: Displays the result set and the explain plan, without the statistics.
  - TRACEONLY STATISTICS: Displays the statistics only.

## Optimizer Modes

### Optimizer Modes

- **Rule-based:**
  - Uses a ranking system
  - Syntax- and data dictionary-driven
- **Cost-based:**
  - Chooses least-cost path
  - Statistics-driven

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

### Using the Optimizer

Oracle RDBMS provides two modes of optimization:

- Rule-based
- Cost-based

### Rule-Based Optimization

In rule-based optimization, the server process chooses its access path to the data by examining the query. This optimizer has a complete set of rules for ranking access paths. Experienced Oracle developers often have a very good understanding of these rules, and tune their SQL accordingly.

The rule-based optimizer is syntax-driven, in that it uses the statement syntax in combination with data dictionary information about the data structures to determine which execution plan will be used. This optimizer mode is supported for backward compatibility with earlier releases of the Oracle server.

## Cost-Based Optimization

In cost-based mode, the optimizer examines each statement and identifies all possible access paths to the data. It then calculates the resource cost of each access path and chooses the least expensive one. The costing is based mainly on the number of logical reads.

The cost-based optimizer is statistics-driven in that it uses statistics generated for the objects involved in the SQL statement to determine the most effective execution plan. The cost-based optimizer is used if any object in the SQL statement has had statistics generated for it.

Using this optimizer mode is recommended for new applications, particularly if they use the parallel query.

**Note:** For additional information about the ranking used by the rule-based optimizer, refer to the *Oracle8i Server Concepts, Release 8.1* manual.

## Setting the Optimizer Mode

### Setting the Optimizer Mode

- Instance level:

```
optimizer_mode =  
{choose|rule|first_rows|all_rows}
```

- Session level:

```
alter session set optimizer_mode =  
{choose|rule|first_rows|all_rows}
```

- Statement level: Using hints

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

### How to Set the Optimizer Mode

You can set the optimizer mode at the:

- Instance level, using the parameter `OPTIMIZER_MODE`
- Session level, by using the `ALTER SESSION` command
- Statement level, by using hints

The DBA is responsible for setting the `OPTIMIZER_MODE` at the instance level, because this requires restarting the instance. Typically, application developers can set the `OPTIMIZER_MODE` at the session level, as well as use hints in SQL statements.

### The `OPTIMIZER_MODE` Parameter

The default value is `CHOOSE`. This means that the optimizer uses the cost-based mode (`ALL_ROWS`) if statistics are available for at least one of the tables involved. Otherwise, it uses rule-based optimization.

**Note:** If any table involved has a degree of parallelization greater than 1, the default behavior will be cost-based optimization as well.

The other possible values are `RULE`, `FIRST_ROWS`, and `ALL_ROWS`. The first one forces rule-based optimization regardless the existence of any statistics. The last two represent different ways of using cost-based optimization. `FIRST_ROWS` minimizes immediate response time (possibly at the expense of overall response time); `ALL_ROWS` minimizes total response time (throughput).



## The OPTIMIZER\_MODE Option at the Session Level

Developers can set this option using the ALTER SESSION command.

```
SQL> ALTER SESSION SET OPTIMIZER_MODE = value
```

**Note:** For backward compatibility reasons, the OPTIMIZER\_GOAL option of the ALTER SESSION command is still supported as an alternative for the OPTIMIZER\_MODE option.

## Optimizer Hints

You can also use hints in the SQL statements to influence the optimizer.

You can code hints into a statement, as shown below.

```
SQL> SELECT /*+ FIRST_ROWS */
2          *
3 FROM    scott.emp;
```

Optimizer hints that can influence the optimizer mode are RULE, FIRST\_ROWS, and ALL\_ROWS.

**Note:** Refer to the *Oracle8i Server Tuning, Release 8.1* manual for a listing of all available hints.

## Precedence Rules

Hints always override session-level settings, and session-level settings always override instance-level settings.

## Managing Statistics

### Managing Statistics

- Use the **ANALYZE** command to collect or delete statistics.
- Use the **DBMS\_STATS** package:
  - **GATHER\_TABLE\_STATS**
  - **GATHER\_INDEX\_STATS**
  - **GATHER\_SCHEMA\_STATS**
  - **GATHER\_DATABASE\_STATS**

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

### Collecting and Using Statistics

You collect statistics on an object with the **ANALYZE** command. Although the cost-based optimizer is not sensitive to minor changes in volume or selectivity, you may want to collect new statistics periodically on frequently modified tables to ensure that the optimizer is using recent, accurate information.

```
ANALYZE {INDEX|TABLE|CLUSTER} object_name
{COMPUTE|DELETE|ESTIMATE} STATISTICS
[FOR ... [SIZE n]] [SAMPLE n {ROWS|PERCENT}]
```

Using the **ANALYZE** command to collect new statistics overwrites any existing statistics in the data dictionary and flushes any related execution plans from the shared pool.

The **DBMS\_STATS** package contains several procedures that allow an index, table, schema or database to be analyzed. This package also enables you to gather most of the statistics with a degree of parallelism. For detailed information, refer to the *Oracle8i Supplied Packages Reference* manual.

## Statistics Accuracy

You can specify two distinct levels of accuracy when you collect statistics:

- **COMPUTE:** This option calculates exact statistics. It performs a full table scan and several calculations. For large tables, this operation can take a considerable amount of time.
- **ESTIMATE:** You estimate statistics with this option. If you use this option with a suitable sample of the data, it is almost as reliable as the **COMPUTE** option.

When you want to clear the collected statistics, you can use the **DELETE** option. You do not need to use **DELETE** option before reanalyzing an object, because existing statistics are overwritten.

## The FOR Clause

The **FOR** clause offers the following options:

- **FOR TABLE:** Restricts the statistics collected to only table statistics rather than table and column statistics.
- **FOR COLUMNS:** Restricts the statistics collected to only column statistics for the specified columns, rather than for all columns and attributes.
- **FOR ALL COLUMNS:** collects column statistics for all columns.
- **FOR ALL INDEXED COLUMNS:** Collects column statistics for all indexed columns in the table.
- **FOR ALL [LOCAL] INDEXES:** Specifies that all indexes associated with the table will be analyzed. **LOCAL** specifies that all local index partitions are analyzed.

## The SIZE Clause

The **SIZE** clause specifies the maximum number of histogram buckets. The default value is 75, and the maximum value is 254. Histograms are discussed in more detail later in this lesson.

## Table Statistics

### Table Statistics

- Number of rows
- Number of blocks and empty blocks
- Average available free space
- Number of chained or migrated rows
- Average row length
- Last ANALYZE date and sample size
- Data dictionary view: DBA\_TABLES

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

### Table Statistics

When you analyze a table, you get the following information about the table:

- Number of rows
- Number of blocks and empty blocks
- Average available free space
- Number of chained or migrated rows
- Average row length
- Last ANALYZE date and sample size

All of the table-level statistics are important, but the importance of the number of empty blocks is not so apparent, and hence is discussed here in detail.

### Number of Blocks and Empty Blocks

Each table has a high-water mark in the segment header block. The high-water mark indicates the last block that was ever used for the table. When the Oracle server performs full table scans, it reads all the blocks up to the high-water mark. Note that the high-water mark is not reset when rows are deleted from the table. This means that Oracle server could be wasting time reading empty blocks.

### Number of Blocks and Empty Blocks (continued)

The DBMS\_SPACE.UNUSED\_SPACE procedure can also be used to find the high-water mark and the number of blocks above the high-water mark, if analyzing a table is impossible or undesirable.

**Example** In the example shown below, you can see that there are about five empty blocks. You have to monitor the ratio between empty blocks and total blocks.

```
SQL> analyze table employees estimate statistics for table;
SQL> select num_rows,blocks,empty_blocks
2      ,      avg_space,avg_row_len, sample_size
3 from    dba_tables
4 where   table_name = 'EMPLOYEES' and owner = user;
NUM_ROWS BLOCKS EMPTY_BLOCKS AVG_SPACE AVG_ROW_LEN SAMPLE_SIZE
-----
15132    434          5      225          48      1064
```

When you determine that there is a problem with empty blocks, you should reorganize the table. You can use any of the different techniques, such as export and import, SQL\*Loader, and SQL\*Plus to reorganize the table and make the table accesses more efficient.

## Index Statistics

### Index Statistics

- Index level (height)
- Number of leaf blocks and distinct keys
- Average number of leaf blocks per key
- Average number of data blocks per key
- Number of index entries
- Clustering factor
- Data dictionary view: DBA\_INDEXES

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

### Clustering Factor

The index clustering factor is an important index statistic for the cost-based optimizer to estimate index scan costs. It is an indication of the number of (logical) data block visits needed to retrieve all table rows through the index. If the index entries follow the table row order, this value approaches the number of data blocks (each block will be visited only once); but, if the index entries randomly point at different data blocks, the clustering factor could approach the number of rows.

### Example

```
SQL> analyze index reg_pk compute statistics;
SQL> select blevel, leaf_blocks, distinct_keys, clustering_factor
  2  from   dba_indexes
  3  where  index_name = 'REG_PK' and owner = user;
BLEVEL LEAF_BLOCKS DISTINCT_KEYS CLUSTERING_FACTOR
-----
      2          682         56252          21349
```

## Column Statistics

### Column Statistics

- Number of distinct values
- Lowest value, highest value
- Last ANALYZE date and sample size
- Data dictionary view:  
USER\_TAB\_COL\_STATISTICS

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

### Viewing Column Statistics

Note that both the lowest and the highest values are stored in RAW (binary) format.

The USER\_TAB\_COL\_STATISTICS view offers a relevant subset of the columns displayed by the USER\_TAB\_COLUMNS view. You can also use the DBA\_TAB\_COL\_STATISTICS view; note, however, that this view does not contain an OWNER column, so you will get confusing results when your database contains multiple tables with the same name in different schemas.

The NUM\_BUCKETS column shows that regular column statistics are treated as a histogram with one bucket. Histograms are discussed on the next page.

### Example

```
SQL> select column_name, num_distinct, low_value, high_value
2    ,          num_nulls, num_buckets
3  from    user_tab_col_statistics
4  where   table_name = 'EMPLOYEES' and column_name = 'SALARY';
```

COLUMN_NAME	NUM_DIST	LOW_VALUE	HIGH_VALUE	NUM_NULLS	NUM_BUCKETS
SALARY	496	C208	C30A62	0	1

## Histograms

### Histograms

- Describe the data distribution of a particular column in more detail
- Better predicate selectivity estimates for unevenly distributed data
- Create histograms with **ANALYZE TABLE ... FOR COLUMNS ...**
- Data dictionary view: **DBA\_HISTOGRAMS**

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

### Using Histograms

When using regular column statistics, a minimum value and a maximum value are stored for each column. The cost-based optimizer uses these values to calculate predicate selectivity, assuming an even distribution of the data between those two extreme values. However, if your data is skewed, this assumption could lead to suboptimal plans. You might consider using histograms to store more detailed information about the data distribution within a column, by partitioning the column values in a number of buckets. Note however that this means additional data dictionary storage requirements. Buckets are height balanced, meaning that each bucket contains approximately the same number of values.

The default number of buckets is 75, and the maximum value is 254.



---

## Using Histograms (continued)

### Example

```
SQL> analyze table classes compute statistics for columns class_id;
```

```
SQL> select endpoint_number, endpoint_value
```

```
2  from    user_histograms
```

```
3  where   table_name = 'CLASSES' and column_name = 'CLASS_ID';
```

```
ENDPOINT  ENDPOINT
```

```
 _NUMBER   _VALUE
```

```
-----
```

```
0         50008
```

```
1         55198
```

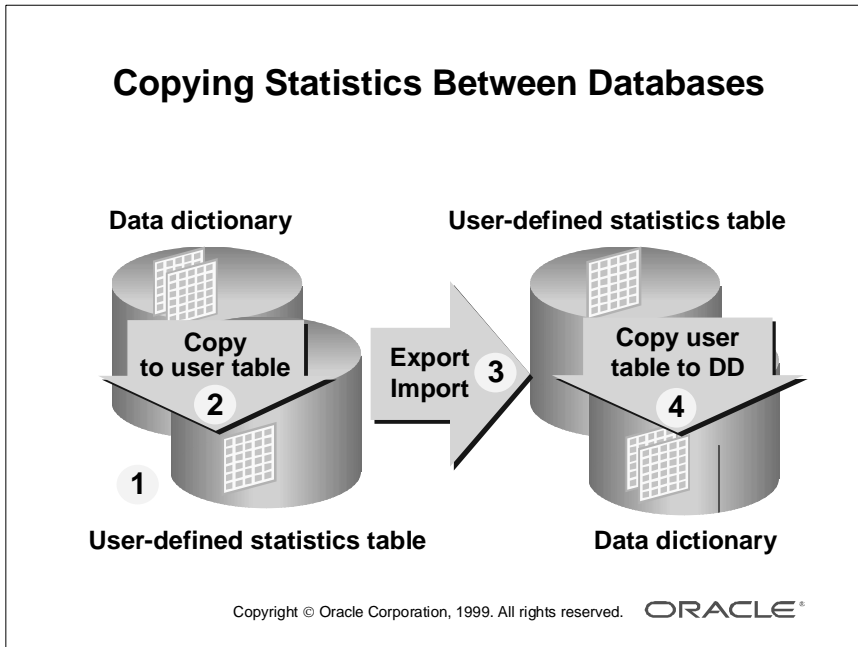
```
2         56718
```

```
3         63037
```

```
...       ...
```

```
75       155801
```

## Copying Statistics Between Databases



### Using DBMS\_STATS to Copy Statistics Between Databases

By using the DBMS\_STATS package procedures, statistics can be copied from an Oracle8i production database to a test database to facilitate tuning. For example, to copy a schema's statistics:

- 1 Use the procedure DBMS\_STATS.CREATE\_STAT\_TABLE in the production database to create a user-defined statistics table.
- 2 Use the procedure DBMS\_STATS.EXPORT\_SCHEMA\_STATS in the production database to copy statistics from the data dictionary to the user-defined statistics table from step 1.
- 3 Use the export and import utilities to transfer the statistics to a corresponding user-defined statistics table in the test database.
- 4 Use the procedure DBMS\_STATS.IMPORT\_SCHEMA\_STATS to import the statistics into the data dictionary in the test database.

You can also use the DBMS\_STATS package to back up statistics prior to analyzing objects. The backup may be used to:

- Restore old statistics
- Study changes in data characteristics over time

### Example: Copying Statistics

```
DBMS_STATS.CREATE_STAT_TABLE
('TRAIN'      /* schema name          */
,'STATS'      /* statistics table name */
,'USERS'      /* tablespace           */
);
```

```
DBMS_STATS.EXPORT_TABLE_STATS
('TRAIN'      /* schema name          */
,'COURSES'    /* table name           */
, NULL        /* no partitions        */
,'STATS'      /* statistics table name */
,'CRS990601'  /* id for statistics     */
, TRUE        /* index statistics      */
);
```

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

### Copying Statistics

To copy statistics for a table from the data dictionary to a user-defined statistics table, use the `CREATE_STAT_TABLE` procedure to create the user-defined table and the `EXPORT_TABLE_STATS` procedure to copy the statistics.

DBMS\_STATS includes the following procedures for copying statistics:

Procedure	Description
CREATE_STAT_TABLE	Creates a user-defined table capable of holding statistics
DROP_STAT_TABLE	Drops a user-defined statistics table
EXPORT_object_STATS	Exports statistics from the data dictionary to a user-defined table
IMPORT_object_STATS	Imports statistics from a user-defined table to the data dictionary
object	can be COLUMN, INDEX, TABLE, or SCHEMA

## Optimizer Plan Stability

- Allows applications to force the use of a desired SQL access path
- Maintains consistent execution path through database changes
- Is implemented using a *stored outline* consisting of hints

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

### Plan Stability

The optimizer may not always have complete information, so sometimes the best possible execution plan is not chosen. In these cases, you may find it worthwhile to influence the optimizer's plan selection by rewriting the SQL statement, by using hints or other tuning techniques. Once satisfied, you may want to ensure that the same tuned plan is generated whenever the same query is recompiled, even when factors that affect optimization have changed.

Oracle8i provides the user with a means of stabilizing execution plans across Oracle releases, database changes, or other factors that normally could cause an execution plan to change. You are able to create a stored outline containing a set of hints used by the optimizer to create an execution plan.

---

## Plan Equivalence

### Plan Equivalence

- **SQL statement text must match**
- **Plans are maintained through:**
  - **New Oracle versions**
  - **New statistics on objects**
  - **Initialization parameter changes**
  - **Database reorganization**
  - **Schema changes**

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

### SQL Statement Equivalence

Plan stability relies on exact textual matching of queries when determining whether a query has a stored outline. This is the same matching criteria used to determine if an execution plan in the shared pool can be reused.

Stored outlines rely partially on hints that the optimizer uses to achieve stable execution plans. Therefore, the degree to which plans remain equivalent is dependent on the capabilities of the hints the plans use. The execution steps included in a stored outline include row access methods, join order, join methods, distributed accesses, and view/subquery merging. Distributed access does not include the execution plan on the remote node.

### Plan Stability

These plans are maintained through many types of database and instance changes. Therefore, if you develop applications for mass distribution, you can use stored outlines to ensure that all your customers access the same execution plans. For example, if the schema is changed by adding an index, the stored outline may prevent the use of the new index.

## Creating Stored Outlines

### Creating Stored Outlines

```
SQL> alter session
      2 set CREATE_STORED_OUTLINES = train;
SQL> select ... from ... ;
SQL> select ... from ... ;
```

```
SQL> create or replace OUTLINE co_cl_join
      2 FOR CATEGORY train ON
      3 select co.crs_id, ...
      4 from   courses co
      5       ,   classes cl
      6 where  co.crs_id = cl.crs_id;
```

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

### Creating Stored Outlines

Oracle can automatically create outlines, or you can create them for specific SQL statements. Outlines use the cost-based optimizer because they rely on hints.

### Categories

Stored outlines can be grouped by categories. The same SQL statement may have a stored outline in more than one category. For example, you may want to have an OLTP category and a DSS category. If a category name is omitted, outlines are placed in the DEFAULT category.

### CREATE\_STORED\_OUTLINES Parameter

Oracle creates stored outlines automatically when you set the parameter `CREATE_STORED_OUTLINES` to `TRUE` or to a category name. When set to `TRUE`, the `DEFAULT` category is used. When activated, Oracle creates outlines for all executed SQL statements. You can deactivate the process by setting the parameter to `FALSE`. When this parameter is used, Oracle generates the outline names.

### CREATE OUTLINE Command

You can also create stored outlines for a specific statement using the `CREATE OUTLINE` command. One advantage of this approach is that you can specify a name for the stored outline.

## Using Stored Outlines

### Using Stored Outlines

- Set the **USE\_STORED\_OUTLINES** parameter to **TRUE** or to a category name

```
SQL> alter session
      2 set USE_STORED_OUTLINES = train;
SQL> select ... from ... ;
```

- Both **CREATE\_STORED\_OUTLINES** and **USE\_STORED\_OUTLINES** can be set at the instance or session level

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

### Using Stored Outlines

If **USE\_STORED\_OUTLINES** is set to **TRUE**, then outlines from the **DEFAULT** category are used.

If **USE\_STORED\_OUTLINES** is set to a category name, then the outlines from that category are used. If there is no matching outline in that category, but there is one in the **DEFAULT** category, then that outline is used.

The statement must match the text of the statement in the outline. They are compared using the method for comparing cursors in the shared pool. This means that hints in the outline must be used in the statement text to cause a match. Values in bind variables do not need to match.

To determine a SQL statement's execution plan, Oracle8i uses the following logic:

- The statement is compared to statements in the shared pool for matching text and outline category.
- If no matching statement is found, the data dictionary is queried for a matching outline.
- If a matching outline is found, Oracle8i integrates the outline into the statement and creates the execution plan.
- If no outline is found, the statement is executed using typical (nonoutline) methods.

### **Using Stored Outlines (continued)**

If an outline specifies the use of an object that cannot be used, the statement will simply not use the hint. For example, it references an index that no longer exists. To verify that a stored outline is being used, the explain plan for a statement needs to be compared when running with and without `USE_STORED_OUTLINES` set.



## Maintaining Stored Outlines

### Maintaining Stored Outlines

- Use the OUTLN\_PKG package to:
  - Drop outlines or categories of outlines
  - Rename categories
- Use the ALTER OUTLINE command to:
  - Rename an outline
  - Rebuild an outline
  - Change the category of an outline
- Outlines are stored in OUTLN schema

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

### Maintaining Stored Outlines

Use procedures in the OUTLN\_PKG package to manage stored outlines and their categories.

Procedure	Description
DROP_UNUSED	Drops outlines that have not been used since they were created
DROP_BY_CAT	Drops outlines assigned to the specified category name
UPDATE_BY_CAT	Reassigns outlines from one category to another

Outlines can also be managed using the ALTER/DROP OUTLINE commands.

You can export and import plans by exporting the OUTLN schema, where all outlines are stored. Outlines can be queried from tables in the OUTLN schema:

- OL\$: Outline name, category, create time, and the text of the statement
- OL\$HINTS: The hints for the outlines in OL\$

Equivalent data dictionary views are DBA\_OUTLINES and DBA\_OUTLINE\_HINTS.

**Note:** Because the user OUTLN is automatically created with the database, its password should be changed.

## Data Access Methods

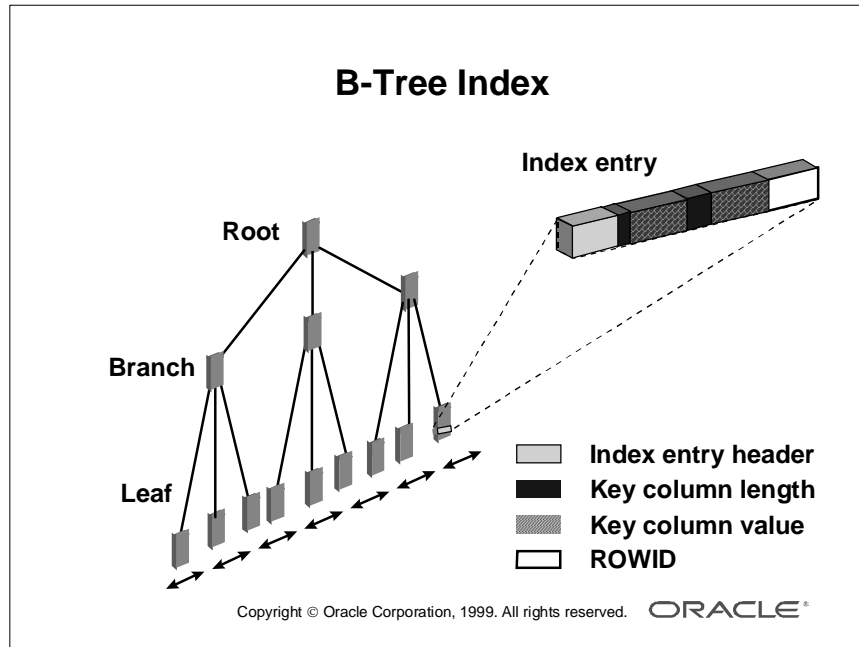
### Data Access Methods

To enhance performance, you can use the following data access methods:

- Indexes (B-tree, bitmap, reverse key)
- Index-organized tables
- Clusters
- Histograms
- Materialized views

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

## B-Tree Indexes



### When to Create B-Tree Indexes

B-tree indexes typically improve the performance of queries that select a small percentage of rows from a table. As a general guideline, you should create indexes on tables that are often queried for less than 5% of the table's rows. This value may be higher in situations where all data can be retrieved from an index, or where the indexed columns can be used for joining to other tables.

### How Indexes Grow

Indexes are always balanced, and they grow from the bottom up.

In situations where the key increases monotonically, the index level tends to increase as rows are added, the new entries are added to the right-hand side of the index. When the right-most leaf block is full, the Oracle server splits it into two blocks and puts 50% of the block in the original and 50% in the new leaf block.

This pattern is repeated until the right-most branch block is full and has to split in two. This process is repeated until the root block. Then the Oracle server adds a new level of branch blocks. The deeper an index is, the less efficient it becomes.

## How to Solve B-Tree Index Performance Degradation

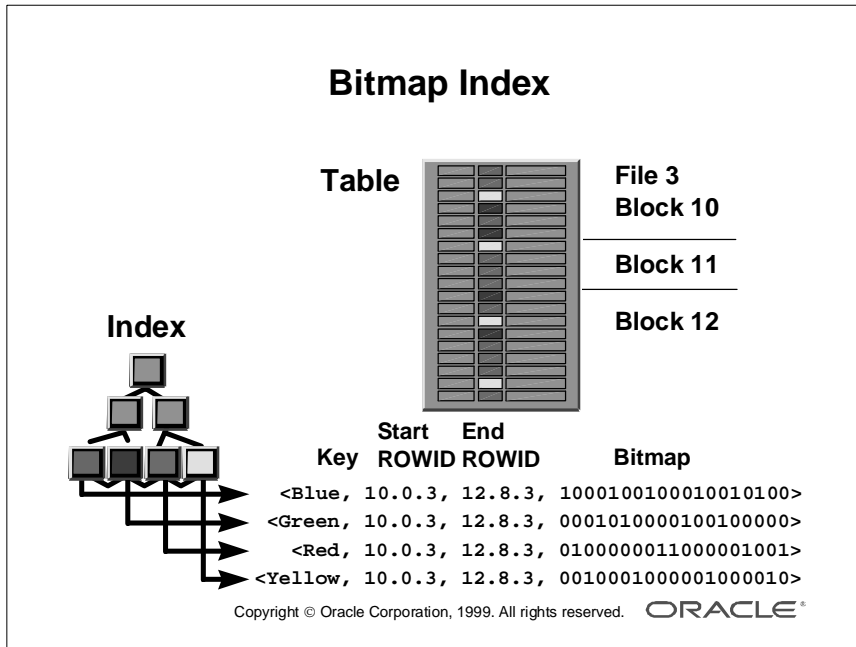
You should regularly rebuild your indexes. However, this can be a time-consuming task, especially if the base table is very large. In Oracle8i, creating and rebuilding indexes can be done online, and parallelization is possible, as well. Note that during index rebuilds the associated base table remains available for queries and DML operations. You can also compute statistics for the cost-based optimizer while rebuilding the index.

```
SQL> ALTER INDEX i_name REBUILD
      2  [PARALLEL n] ONLINE
      3  [COMPUTE STATISTICS]
      4  [NOLOGGING];
```

The keyword **ONLINE** specifies that DML operations on the table or partition are allowed during rebuilding of the index.

**Restriction:** Parallel DML is not supported during online index building. If you specify **ONLINE** and then issue parallel DML statements, the Oracle server returns an error.

## Bitmap Indexes



## Bitmap Indexes

- **Used for low-cardinality columns**
- **Good for multiple predicates**
- **Use minimal storage space**
- **Best for read-only systems**
- **Good for very large tables**

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

### When to Create Bitmapped Indexes

- Bitmap indexes are intended for *low-cardinality* columns, which contain a limited number of values.
- If you use a query with multiple WHERE conditions, the Oracle server can use logical bit-AND or bit-OR operations to combine this bitmap with bitmaps for other columns.

### Performance Considerations

- Bitmap indexes use little storage space: one entry per distinct key value, stored in a compressed form. Each bitmap is divided into bitmap segments (one-half block).
- They work very fast with multiple predicates on low-cardinality columns.
- They are particularly suited to large, read-only systems such as DSS.
- DML statements slow down performance: They are not suited for OLTP applications.
- Locking is at the bitmap segment level, not the entry level.

Parallel query, PDML (Parallel Data Manipulation Language), and parallelized CREATE statement work with bitmap indexes.

## Creating and Maintaining Bitmap Indexes

```
SQL> create BITMAP INDEX ord_region_id_idx
2          on ord(region_id)
3  storage  (initial 200k next 200k
4           pctincrease 0 maxextents 50)
5  tablespace indx01;
```

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

### Maintenance Considerations

In a DSS environment, data is usually maintained by using bulk inserts and updates. Index maintenance is deferred until the end of each DML operation. For example, if you insert 1,000 rows into a table that has a bitmapped index, the bitmap column information and the ROWID information from the inserted rows are placed into the reference to the sort buffer, and then the updates of all 1,000 index entries are batched. This is why `SORT_AREA_SIZE` must be set properly for good performance with inserts and updates on bitmap indexes. Thus, each bitmap segment is updated only once per DML operation, even if more than one row in that segment changes.

## Initialization Parameters for Bitmap Indexing

The following two initialization parameters have an impact on performance:

- **CREATE\_BITMAP\_AREA\_SIZE:**
  - Defines the amount of memory allocated for bitmap creation
  - Cannot be dynamically altered at the session level
  - Has a default value of 8 MB (A larger value may lead to faster index creation. If cardinality is very small, you can set a small value for this parameter. For example, if cardinality is only two, then the value can be on the order of kilobytes. As a general rule, the higher the cardinality, the more memory is needed for optimal performance.)
- **BITMAP\_MERGE\_AREA\_SIZE:**
  - Defines the amount of memory used to merge bitmaps retrieved from a range scan of the index
  - Cannot be dynamically altered at the session level
  - Has a default value of 1 MB (A larger value should improve performance, because the bitmap segments must be sorted before being merged into a single bitmap.)

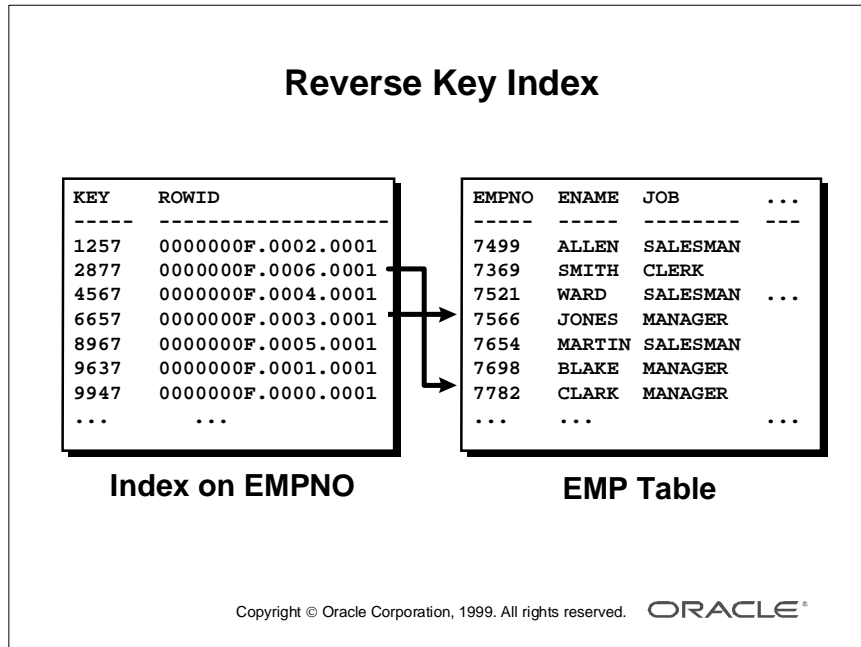


## Comparing B-Tree and Bitmap Indexes

B-Tree indexes	Bitmap indexes
Suitable for high-cardinality columns	Suitable for low-cardinality columns
Updates on keys relatively inexpensive	Updates to key columns very expensive
Inefficient for queries using OR predicates	Efficient for queries using OR predicates
Row-level locking	Bitmap segment-level locking
More storage	Less storage
Useful for OLTP	Useful for DSS

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

## Reverse Key Indexes



### Definition

Creating a reverse key index reverses the bytes of each column key value, keeping the column order in case of a composite key.

### When to Create Reverse Key Indexes

An ever-increasing key (such as sequential numbers for invoices, employees, or order numbers) will repetitively degrade the index height (BLEVEL statistic); you can avoid forced block splits by spreading the index entries more evenly.

### Disadvantage

When application statements specify ranges, the explain plan produces a full table scan, because the index is not usable for a range scan.

## Creating Reverse Key Indexes

```
SQL> create unique index i1_t1 ON t1(c1)
  2  REVERSE pctfree 30
  3  storage(initial 200k next 200k
  4          pctincrease 0 maxextents 50)
  5  tablespace indx01;
```

```
SQL> create unique index i2_t1 ON t1(c2);
SQL> alter index i2_t1 REBUILD REVERSE;
```

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

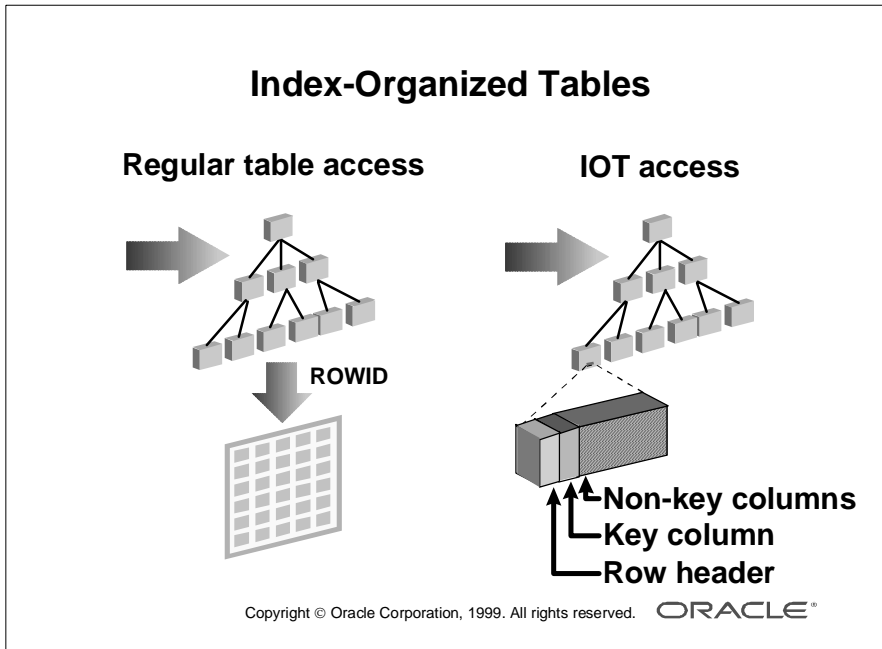
## Dictionary View

Use the following query to list the names of all reverse key indexes:

```
SQL> select index_name, index_type
  2  from   dba_indexes
  3  where  index_type like '%REV';
```

INDEX_NAME	INDEX_TYPE
ORD_ORD_NO_IDX	NORMAL/REV

## Index-Organized Tables



### Definition

An index-organized table is like a regular table with an index on one or more of its columns, but instead of maintaining two separate segments for the table and the B-tree index, the database system maintains one single B-tree structure that contains both the primary key value and the other column values for the corresponding row.

Index-organized tables are suitable for frequent data access through the primary key, or through any key that is a prefix of the primary key, such as in applications that use inverted indexes, which are used in text searches. These indexes keep the value and all its locations together. Therefore, each word has one entry, and that entry records all the places in which the word occurs.

For index-organized tables, a primary key constraint is mandatory.

### Benefits

- No duplication of the values for the primary key column (index and table columns in indexed tables); therefore, less storage is required
- Faster key-based access for queries involving exact matches or range searches or both.

## **Index-Organized Tables Compared with Regular Tables**

- **Faster key-based access to table data**
- **Reduced storage requirements**
- **Secondary indexes and logical ROWIDs**
- **Main restrictions:**
  - **Must have a primary key**
  - **Cannot use unique constraints**
  - **Cannot be clustered**

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

### **Logical ROWIDs**

Index-organized tables don't have regular (physical) ROWIDs. However, Oracle8i introduced the concept of logical ROWIDs to overcome certain restrictions caused by the lack of physical ROWIDs. Logical ROWIDs provide the fastest possible access to rows in index-organized tables (IOTs) by using two methods:

- A physical “guess” with an access time equal to that of physical ROWIDs
- Access without the guess (or after an incorrect guess); performs like a primary key access of the IOT

The guess is based on knowing of the file and block that a row resides in. The block information is accurate when the index is created, but changes if the leaf block splits. If the guess is wrong and the row no longer resides in the specified block, then the remaining portion of the logical ROWID entry, the primary key, is used to get the row. Logical ROWIDs are stored as a variable-length field, where the size depends on the primary key value being stored.

The UROWID data type makes logical ROWIDs usable by applications in the same sense that ROWIDs are used, for example, selecting ROWIDs for later update or as part of a cursor. UROWID can also be used to store ROWIDs from other databases, accessed by way of gateways. The UROWID type can also be used to reference physical ROWIDs.

## Creating Index-Organized Tables

```
SQL> create table sales
2  (office_cd      number(3)
3  ,qtr_end        date
4  ,revenue        number(10,2)
5  ,review         varchar2(1000)
6  ,constraint sales_pk
7  PRIMARY KEY (office_cd,qtr_end)
8  )
9  ORGANIZATION INDEX tablespace indx
10 PCTTHRESHOLD 20
11 INCLUDING revenue
12 OVERFLOW TABLESPACE user_data;
```

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

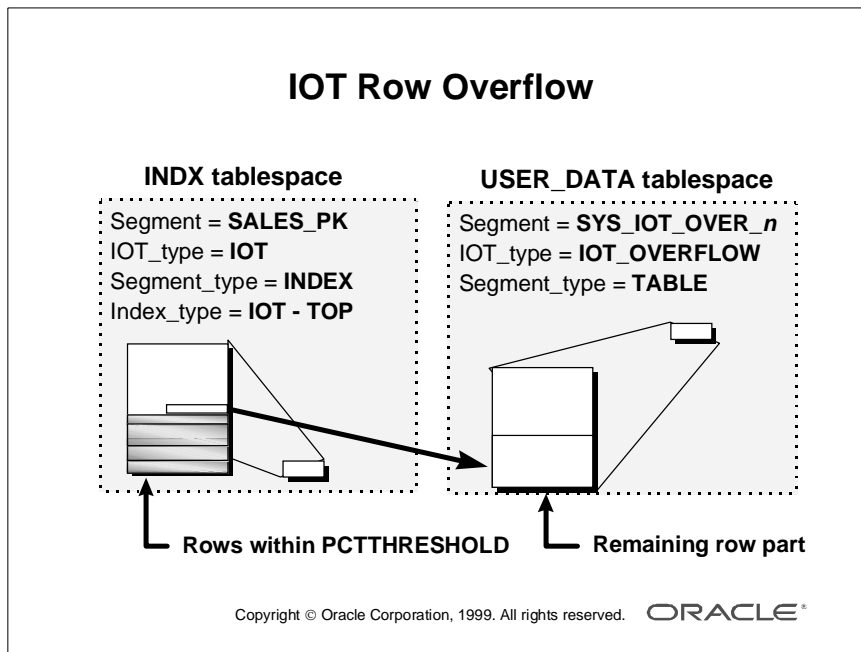
## Creating Index-Organized Tables

Regular B-tree index entries are usually small. They consist of only the primary key value and a ROWID value. Many of these small index entries can be stored in a leaf block. This is not necessarily the case for index-organized tables, because they store full rows, or at least as much as will fit without exceeding the limit set by PCTTHRESHOLD.

Storing large entries in index leaf blocks slows index searches and scans. You can specify the rows to go into an overflow area by setting a threshold value that represents a percentage of block size.

Of course, the primary key column must always be stored in the IOT index blocks as a basis for search. But, you can choose to keep nonkey values in a separate area, the row overflow area, so that the B-tree itself remains densely clustered.

The three clauses that influence IOT row overflow are discussed on the next page.



### PCTTHRESHOLD Clause

This clause specifies the percentage of space reserved in the index block for an index-organized table row. If a row exceeds the size calculated based on this value, all columns after the column named in the INCLUDING clause are moved to the overflow segment. If OVERFLOW is not specified, then rows exceeding the threshold are rejected. PCTTHRESHOLD defaults to 50 and must be a value from 0 to 50.

### INCLUDING Clause

This clause specifies a column at which to divide an index-organized table row into index and overflow portions. Oracle accommodates all non-key columns up to the column specified in the INCLUDING clause in the index leaf block, provided it does not exceed the specified threshold.

### OVERFLOW Clause and Segment

This clause specifies that index-organized table data rows exceeding the specified threshold are placed in the data segment defined by the segments attributes, which specify the tablespace, storage, and block utilization parameters.

## IOT Dictionary Views

```
SQL> select table_name,tablespace_name,iot_name,iot_type
       2 from   DBA_TABLES;
TABLE_NAME          TABLESPACE_NAME  IOT_NAME  IOT_TYPE
-----
SALES
SYS_IOT_OVER_2268  USER_DATA        SALES     IOT_OVERFLOW
```

```
SQL> select index_name,index_type,tablespace_name,table_name
       2 from   DBA_INDEXES;
INDEX_NAME          INDEX_TYPE  TABLESPACE  TABLE_NAME
-----
SALES_PK            IOT - TOP  INDX         SALES
```

```
SQL> select segment_name,tablespace_name,segment_type
       2 from   DBA_SEGMENTS;
SEGMENT_NAME        TABLESPACE  SEGMENT_TYPE
-----
SYS_IOT_OVER_2268  USER_DATA   TABLE
SALES_PK           INDX        INDEX
```

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

### DBA\_TABLES

The IOT\_TYPE and IOT\_NAME columns provide information about index-organized tables. If there is no overflow area for an IOT, then there will be only a single dictionary entry. The value of IOT\_TYPE is IOT, and the IOT\_NAME field is blank. If an overflow area has been specified, then its name appears as an additional new table in the list of table names. The overflow table is called SYS\_IOT\_OVER\_##### in DBA\_TABLES. IOT\_TYPE is set to IOT\_OVERFLOW for this table, and the IOT\_NAME is set to the name of the index-organized table to which it belongs.

### DBA\_INDEXES

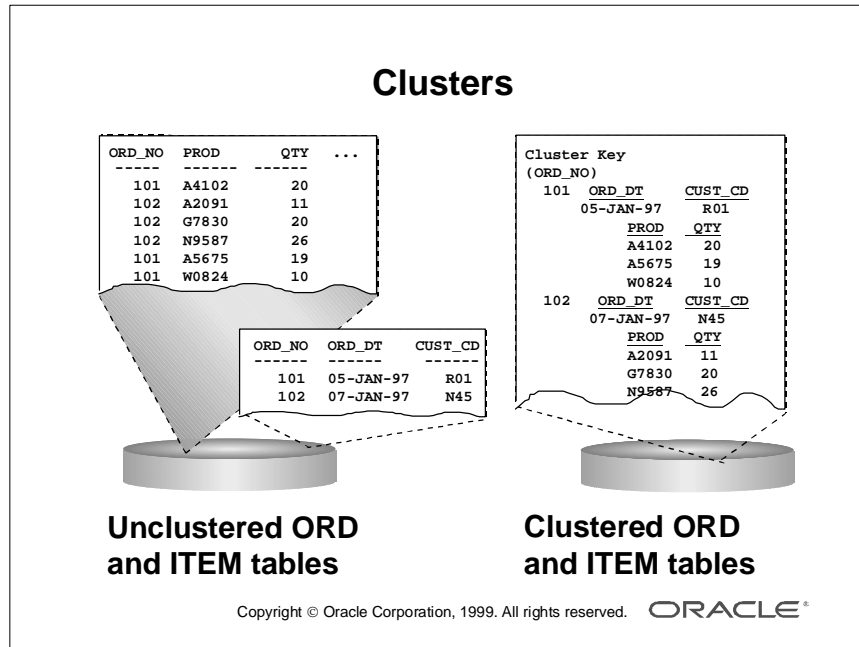
The IOT table name listed is the same as that used in the CREATE TABLE command, but an index name is also listed for the table, with a name of SYS\_IOT\_TOP\_#####.

### DBA\_SEGMENTS

The name of the overflow segment is listed here with the same name as mentioned above, SYS\_IOT\_OVER\_#####. The table itself is listed as SYS\_IOT\_TOP\_#####.



## Clusters



### Definition

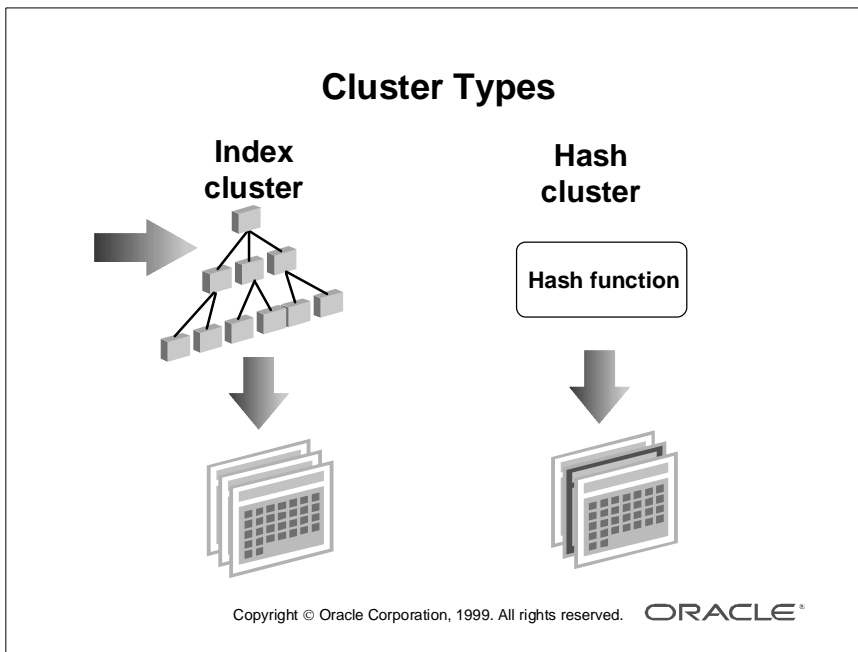
A *cluster* is a group of one or more tables that share the same data blocks because they share common columns and are often used together in join queries.

### Performance Benefits

- Disk I/O is reduced and access time improved for joins of clustered tables.
- Each cluster key value is stored only once for all the rows of the same key value; it therefore uses less storage.

### Performance Consideration

Full table scans are generally slower on clustered tables than on nonclustered tables.



## Index Clusters

An *index cluster* uses an index, known as the *cluster index*, to maintain the data within the cluster. The cluster index must be available to store, access, or maintain data in an index cluster.

The cluster index is used to point to the block that contains the rows with a given key value. The structure of a cluster index is similar to that of a normal index.

Although a normal index does not store NULL key values, cluster indexes store NULL keys. There is only one entry for each key value in the cluster index. Therefore, they are likely to be smaller than a normal index on the same set of key values.

## Hash Clusters

A *hash cluster* uses a (user-defined or system-generated) hash algorithm to calculate the location of a row, both for retrieval and for DML operations.

For equality searches that use the cluster key, a hash cluster can provide greater performance gains than an index cluster because it has only one segment to scan (no index access is needed).

### Situations in Which Clusters Are Useful

Criterion	Index	Hash
Uniform key distribution	X	X
Evenly distributed key values		X
Rarely updated key	X	X
Often joined master-detail tables	X	
Predictable number of key values		X
Queries using equality predicate on key		X

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

### When Not to Use Clusters

- If a full scan is frequently executed on one of the clustered tables, because this table is stored on more blocks than if it had been created alone.
- If the data from all tables with the same cluster key value exceeds more than one or two Oracle blocks, to access a row in a clustered table, the Oracle server reads all blocks containing rows with the same value.

### When Not to Use Hash Clusters

- If the table is constantly growing and if it is impractical to rebuild a new, larger hash cluster
- If your application often performs full table scans and you had to allocate a great deal of space to the hash cluster in anticipation of the table growing

## Materialized Views

### Materialized Views

- Are instantiations of a SQL query
- Can be used for query rewrites
- Refresh types:
  - Complete or fast
  - Force or never
- Refresh modes:
  - Manual
  - Automated (synchronous or asynchronous)

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

### Materialized View

A materialized view stores both the definition of a view plus the rows resulting from the execution of the view. Like a view, it uses a query as the basis, but the query is executed at the time the view is created and the results are stored in a table. You can define the table with the same storage parameters as any other table and place it in the tablespace of your choice. You can also index and partition the materialized view table like other tables to improve the performance of queries executed against them.

When a query can be satisfied with data in a materialized view, the server transforms the query to reference the view rather than the base tables. By using a materialized view, expensive operations such as joins and aggregations do not need to be reexecuted by rewriting the query against the materialized view.

## Materialized View Refresh

Materialized views use the same internal mechanism as snapshots, and support several refresh techniques.

A complete refresh of a materialized view involves truncating existing data and reinserting all the data based on the detail tables by reexecuting the query definition from the CREATE command.

Fast refreshes only apply the changes made since the last refresh. Two types of fast refresh are available:

- Fast refresh using materialized view logs: In this case, all changes to the base tables are captured in a log and then applied to the materialized view.
- Fast refresh using ROWID range: A materialized view can be refreshed using fast refresh after direct path loads, based on the ROWIDs of the new rows. Direct loader logs are required for this refresh type.

A view defined with a refresh type of Force refreshes with the fast mechanism if possible; if not, it uses a complete refresh. Force is the default refresh type.

The Never option suppresses all refreshes of the materialized view.

## Refresh Modes

Manual refreshes are performed using the DBMS\_MVIEW package. The DBMS\_MVIEW package provides a number of procedures and functions to manage materialized views, including the REFRESH, REFRESH\_DEPENDENT, and REFRESH\_ALL\_MVIEWS procedures.

Automatic refresh can be performed:

- ON COMMIT: When this option is specified for a materialized view, it gets updated whenever changes to one of the base tables are committed. The update to the materialized view occurs asynchronously to the user commit and does not degrade the user's perceived performance.
- At a specified time: Refresh on a materialized view can be scheduled to occur at a specified time. For example, it can be refreshed every Monday at 9:00 a.m. by using the START WITH and NEXT clauses. In order for such refreshes to occur, the instance must initiate job processes with the JOB\_QUEUE\_PROCESSES parameter.

**Note:** Not all refresh types and refresh modes are available to every materialized view. Appendix C in the *Oracle8i Replication Manual* describes the modes and when each can be used.

## Materialized Views: Manual Refresh

### Refresh-specific MVs:

```
DBMS_MVIEW.REFRESH  
( 'SF_SALES', parallelism => 10 );
```

### MVs based on one or more base tables:

```
DBMS_MVIEW.REFRESH_DEPENDENT( 'SALES' );
```

### All MVs due for refresh:

```
DBMS_MVIEW.REFRESH_ALL_MVIEWS;
```

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

## Manual Refresh of Materialized View

The following is a list of possible refresh scenarios for materialized views (MVs):

- Refresh specific materialized views using REFRESH procedure
- Refresh all materialized views that depend on a given set of base tables using REFRESH\_DEPENDENT procedure
- Refresh all materialized views that have not been refreshed since the last bulk load to one or more detail tables using REFRESH\_ALL\_MVIEWS procedure

The procedures in the package use a number of parameters to specify:

- Refresh method
- Whether to proceed if an error is encountered
- Whether to use a single transaction (consistent refresh)
- The rollback segment to use

Server job queues are used to run the refresh job. Therefore, you must set the appropriate initialization parameters, JOB\_QUEUE\_PROCESSES and JOB\_QUEUE\_INTERVAL, to enable job queue refresh processes.

## Query Rewrites

### Query Rewrites

- To use MVs instead of the base tables, a query must be rewritten.
- Query rewrites are transparent and do not require any special privileges on the MV.
- MVs can be enabled or disabled for query rewrites.

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

### Query Rewrites

Accessing a materialized view may be significantly faster than accessing the underlying base tables, so the optimizer will rewrite a query to access the view when the query allows it. The query rewrite activity is transparent to applications. In this respect, their use is similar to the use of an index.

Users do not need explicit privileges on materialized views to use them. Queries executed by any user with privileges on the underlying tables may be rewritten to access the materialized view.

A materialized view can be enabled or disabled. A materialized view that is enabled is available for query rewrites.

## Query Rewrites

- The initialization parameter **QUERY\_REWRITE\_ENABLED** must be set to **TRUE**.
- The **QUERY REWRITE** privilege allows users to enable materialized views.
- The **DBMS\_OLAP** package has options to use materialized views.

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

## Query Rewrites

The query rewrite is performed by the optimizer. The rewrites are transparent to the application.

The ability to perform rewrites must be enabled either at the session level or at the instance level using the **QUERY\_REWRITE\_ENABLED** parameter.

To enable or disable individual materialized views for query rewrites, the user must have the **GLOBAL QUERY REWRITE** or the **QUERY REWRITE** system privilege. Both versions of the privilege allow users to enable materialized views in their own schema. The **GLOBAL** version allows users to enable any materialized views they own, whereas the simple **QUERY REWRITE** privilege requires that the base tables as well as the views be in the user's schema.



## Materialized Views and Query Rewrites: Example

### Materialized Views and Query Rewrites: Example

```
SQL> create MATERIALIZED VIEW sales_summary
2      tablespace sales_ts
3      parallel (degree 4)
4      BUILD IMMEDIATE REFRESH FAST
5      ENABLE QUERY REWRITE
6  AS
7  select s.zip, p.product_type
8      ,      sum(s.amount)
9  from    sales s, product p
10 where   s.product_id = p.product_id
11 group  by s.zip, p.product_type;
```

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

### Creating a Materialized View: Syntax

The syntax for creating materialized view is similar to the CREATE SNAPSHOT command. There are some additional options. In the example, the BUILD IMMEDIATE option is chosen to cause the materialized view to be populated when the CREATE command is executed. This is the default behavior. You could choose the BUILD DEFERRED option, which creates the structure but does not populate it until the first refresh occurs.

When you want a table that was created before Oracle8i to be the source of a materialized view, use the ON PREBUILD TABLE option.

The ENABLE/DISABLE QUERY REWRITE clause determines whether query rewrites are automatically enabled for the materialized view.

## Materialized Views and Query Rewrites: Example

```
SQL> select s.zip, p.product_type, sum(s.amount)
2  from   sales s, product p
3  where  s.product_id = p.product_id
4  group  by s.zip, p.product_type;
```

OPERATION	NAME
-----	
SELECT STATEMENT	
TABLE ACCESS FULL	SALES_SUMMARY

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

### Materialized Views: Query Rewrites

The execution plan for the query—which is a formatted output from a plan table—shows that the query did not run against the two base tables, but simply scanned the materialized view table. This example embodies the power of materialized views and query rewrites. The table comprising the materialized view substitutes completely for the base tables, and all the operations on them, named in the query.

## Enabling and Controlling Query Rewrites

### Enabling and Controlling Query Rewrites

- **Initialization parameters:**
  - **OPTIMIZER\_MODE**
  - **QUERY\_REWRITE\_ENABLED**
  - **QUERY\_REWRITE\_INTEGRITY**
- **Dynamic and session-level parameters:**
  - **QUERY\_REWRITE\_ENABLED**
  - **QUERY\_REWRITE\_INTEGRITY**
- **New hints: REWRITE and NOREWRITE**

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

### Controlling Query Rewrites

The following parameters control query rewrites:

**OPTIMIZER\_MODE:** Query rewrites are only available under cost-based optimization. If rule-based optimization is used, query rewrites do not occur.

**QUERY\_REWRITE\_ENABLED:** This parameter can be set to FALSE to suppress query rewrites even while using the cost-based optimizer. This parameter can be altered dynamically for the whole instance as well as for individual sessions.

**QUERY\_REWRITE\_INTEGRITY:** This parameter can also be reset dynamically for the instance and for an individual session. It accepts the following values:

- **ENFORCED** (the default) enables query rewrites only if Oracle can guarantee consistency. Only up to date materialized views and enabled validated constraints are used for query rewrites.
- **TRUSTED** allows query rewrites based on declared, but not necessarily enforced, relationships. All updated materialized views and constraints with RELY flag are used for query rewrites.
- **STALE\_TOLERATED** allows query rewrites to use materialized views that have not been refreshed since the last DML operation and relationships that are declared.

### **Controlling Query Rewrites (continued)**

Query rewrites are treated similarly to execution plan paths. Therefore, there are no object privileges associated with them. Users who have access to the detail tables implicitly benefit from summary rewrites.

A new hint, `REWRITE`, can be used to restrict the materialized views that will be considered for query rewrite. Another hint, `NOREWRITE`, is available to suppress rewrites for a query block.

### **Dimensions**

Dimensions are Oracle8i data dictionary structures that define hierarchies based on columns in existing database tables. Although they are optional, they are highly recommended because they:

- Enable additional rewrite possibilities without the use of constraints (Implementation of constraints may not be desirable in a data warehouse for performance reasons.)
- Help document dimensions and hierarchies explicitly
- Can be used by OLAP tools

For more information about dimensions, see the *Oracle8i Server Tuning* manual.

## Disabling Query Rewrites: Example

```
SQL> select /*+ NOREWRITE */
2      s.zip, p.product_type, sum(s.amount)
3 from  sales s, product p
4 where s.product_id = p.product_id
5 group by s.zip, p.product_type;
```

OPERATION	NAME
-----	-----
SELECT STATEMENT	
SORT GROUP BY	
HASH JOIN	
TABLE ACCESS FULL	SALES
. . .	

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

### Disabling Query Rewrites

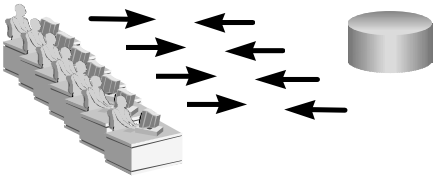
In this example, the NOREWRITE hint tells the optimizer not to rewrite the query to make use of the materialized view. The statement is otherwise identical to the previous example. The execution plan derived from the query shows that the original statement is executed, including the hash join between the two tables and the sort to obtain the groups.

A REWRITE/NOREWRITE hint overrides a materialized view's definition, set in the CREATE or ALTER MATERIALIZED VIEW command with the ENABLE QUERY REWRITE clause.

## OLTP Systems

### OLTP Systems

- High throughput, insert- and update-intensive
- Large, continuously growing data volume
- Concurrent access by many users
- Tuning goals:
  - Availability
  - Speed
  - Concurrency
  - Recoverability



The diagram illustrates an OLTP system architecture. On the left, a group of stylized human figures representing users are seated at a long desk. Arrows point from each user towards a central database cylinder on the right. From the database, arrows point back to each user, indicating a two-way flow of data and interaction. This visualizes the high concurrency and frequent data exchange characteristic of OLTP systems.

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

### Typical OLTP Applications

- Airline reservation systems
- Large order-entry applications
- Banking applications

### Requirements

- High availability (7 days, 24 hours)
- High speed
- High concurrency
- Reduced time recovery

## **OLTP Requirements**

- **Explicit space allocation**
- **Indexes:**
  - **Not too many (prefer B-tree to bitmap)**
  - **Reverse key for sequence columns**
  - **Rebuilt regularly**
- **Clusters for tables in join queries:**
  - **Index clusters for growing tables**
  - **Hash clusters for stable tables**

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

## **Space Allocation**

- To avoid the performance load of dynamic space allocation, allocate space explicitly to tables, clusters, and indexes.
- Check growth patterns regularly to find the rate at which extents are being allocated, so that you can plan extents creation.

## Indexing

- Indexing is critical to data retrieval in OLTP systems. DML statements on indexed tables require index maintenance and this is a significant performance overhead. Your indexing strategy must be closely geared to the real needs of the application.
- Indexing a foreign key helps child data to be modified without locking the parent data.
- B-tree indexing is preferred to bitmap indexing, because of locking issues affecting DML operations: when a B-tree index entry is locked, a single row is locked, whereas when a bitmap index entry is locked, a whole range of rows are locked.
- Reverse key indexes avoid frequent B-tree block splits for sequence columns.
- You need to rebuild indexes regularly, as suggested in the earlier pages.

## Hash Clustering

- Using hash clusters should speed up access on equality queries. But you should not choose this data structure for a table that is:
  - Heavily inserted into, because of the use of space and the number of blocks that need to be visited
  - Heavily updated using larger column values, because of migration probability
- If a table is growing, there are likely to be many collisions on hash keys and this leads to store rows in overflow blocks. To avoid this situation, correctly estimate the HASHKEYS value. With a large number of hash keys for a given number of rows, the likelihood of a collision decreases.



## OLTP Requirements

- **Short transactions do not require big rollback segments; multiple rollback segments prevent contention.**
- **A large MINEXTENTS value is required.**

```
SQL> create rollback segment rbs01
2 storage (initial 100k next 100k
3 minextents 20 maxextents 121
4 optimal 400k )
5 tablespace rbs;
```

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

## Rollback Segment Considerations

OLTP transactions are likely to be short, which has consequences for your rollback segment configuration. They are unlikely to run out of rollback segment space; you need enough rollback segments to prevent contention for transaction tables.

To get the correct number of rollback segments, you need to know about the transaction pattern. For example, consider the following system characteristics:

- 170 users logged on
- Each user executing three 1-second transactions per minute, on average

There is only a small probability that at any given time more than eight transactions are active, so three rollback segments are enough.

You need to set MINEXTENTS to be at least 10 for small databases and 20 for larger databases, because:

- Dynamic extension of rollback segments is just as much a performance issue as the dynamic extension of tables.
- It also reduces the risk of hitting extents in use when wrapping rollback segments.

## OLTP Application Issues

- **Use database constraints instead of application code.**
- **Make sure that code is shared.**
- **Use bind variables rather than literals for optimally shared SQL.**

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

### Integrity Constraints

If there is a choice between keeping application logic in procedural code or using declarative constraints, bear in mind that constraints are always less expensive to process. Referential integrity and CHECK constraints are the main types to consider here.

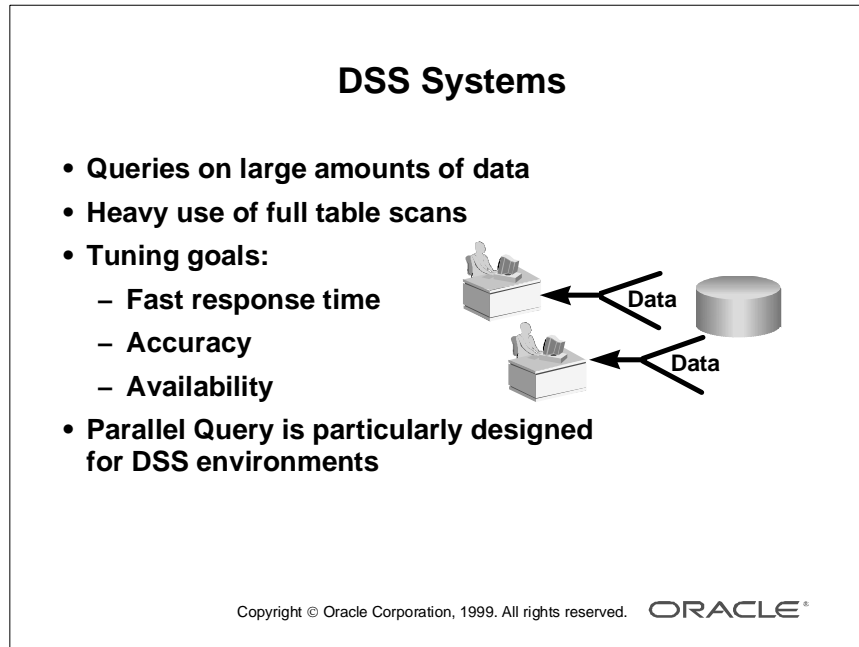
### Shared Code

You should ensure that code is shared by stored procedural objects, such as packages, procedures, and functions.

### Bind Variables

You want to keep the overhead of parsing to a minimum. Try to ensure that the application code uses bind variables rather than literals.

## DSS Systems



### Characteristics

Decision support applications distill large amounts of information into understandable reports.

They gather data from OLTP systems and use summaries and aggregates in retrieving it. They make heavy use of full table scans as an access method.

Decision makers in an organization use this data to determine what strategies the organization should take.

**Example** A marketing tool determines the buying patterns of consumers based on information gathered from demographic data to determine which items sell best in which locations.

### Requirements

- Fast response time (When you design a DSS, you must ensure that queries on large amounts of data can be performed within a reasonable time frame.)
- Accuracy
- Daytime availability

## DSS Requirements

### Storage allocation:

- Set `db_block_size` and `db_file_multiblock_read_count` carefully.
- Ensure that extent sizes are multiples of this parameter value.
- Run ANALYZE regularly.

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

### DB\_BLOCK\_SIZE and DB\_FILE\_MULTIBLOCK\_READ\_COUNT

Because DSS applications typically perform many table scans, you should consider a higher value for the `DB_BLOCK_SIZE` parameter. Even if this means re-creating a large database, it almost certainly pays off, because a larger block size facilitates read-intensive operations that are characteristic of DSS applications.

Pay particular attention to the setting of the `DB_FILE_MULTIBLOCK_READ_COUNT` parameter. During full table scans and fast full index scans it determines how many database blocks are read into the buffer cache with a single operating system read call. A larger value decreases estimated table scan costs and favors table scans over index searches.

## DSS Requirements

- **Evaluate the need for indexes:**
  - Use bitmap indexes when possible.
  - Use index-organized tables for (range) retrieval by PK.
  - Generate histograms for indexed columns that are distributed nonuniformly.
- **Clustering: Consider hash clusters for performance access.**

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

### Indexing

Consider how you can minimize the space and performance overhead of index maintenance. Because most queries use full table scans, you could:

- Dispense with indexes altogether
- Maintain indexes only for a few tables that are accessed selectively
- Regularly generate histograms for those with data distributed nonuniformly
- Choose bitmap indexes for queries on columns with few distinct values; they offer much faster retrieval access
- (For bulk inserts and updates, you must set the sorting `init.ora` parameters appropriately: `SORT_AREA_SIZE`, `BITMAP_MERGE_AREA_SIZE`, `CREATE_BITMAP_AREA_SIZE`.)
- Use index-organized tables for a faster, key-based access to table data for queries involving exact match or range search and complete row data retrieval

### Clustering

Consider both types of clusters, particularly hash clusters, for their access performance, excluding the tables growing regularly during bulk loads, except if you have the possibility to re-create the cluster.

## DSS Application Issues

- **Parse time is less important.**
- **The execution plan must be optimal.**
  - **Use the Parallel Query feature.**
  - **Tune carefully, using hints if appropriate.**
  - **Test on realistic amounts of data.**
  - **Consider using PL/SQL functions to code logic into queries.**
- **Bind variables are problematic.**

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

### Parse Time

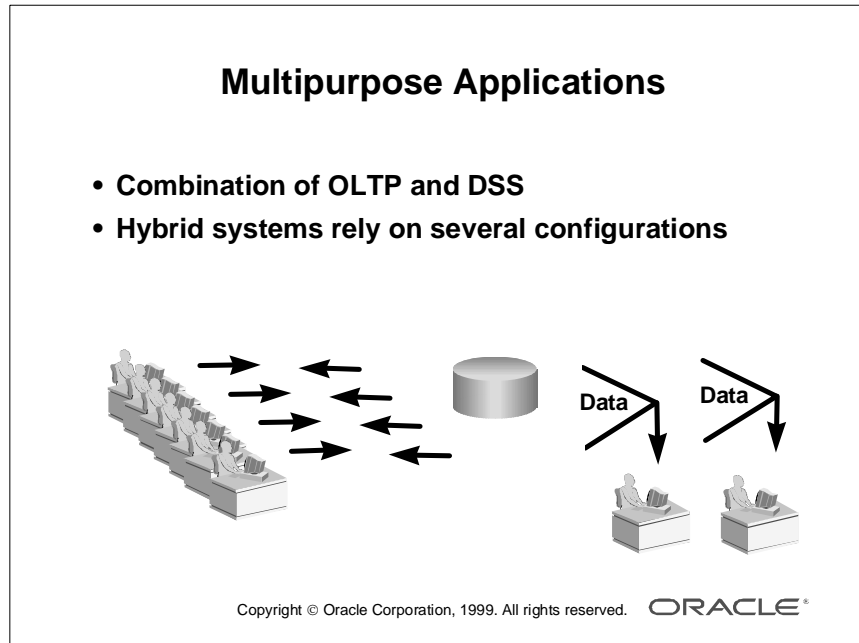
The time taken to parse SELECT statements is likely to be a very small proportion of the time taken to execute the query. Tuning the library cache is much less of an issue for DSS than for OLTP.

Your priority is an optimal access path in the execution plan; small variations can cost minutes or hours. Developers must:

- Use *parallelized queries*, which enable multiple processes to work together simultaneously to process a single SQL statement  
Symmetric multiprocessors (SMP), clustered, or massively parallel processing (MPP) configurations gain the largest performance benefits, because the operation can be effectively split among many CPUs on a single system.
- Use the EXPLAIN PLAN command to tune the SQL statements, and *hints* to control access paths

If your application logic uses bind variables, you lose the benefit of this feature: the optimizer makes a blanket assumption about the selectivity of the column, whereas with literals, the cost-based optimizer uses histograms.

## Multipurpose Applications



### Hybrid Systems

Many applications rely on several configurations and Oracle options. You must decide what type of activity your application performs and determine which features are best suited for it.

Conflicting performance goals can be solved by copying gathered data from the OLTP database into a DSS database that will only be queried. But this technique compromises the goal of accuracy for the DSS application. Therefore, you must determine which goals are most important.

Many Oracle customers who are maintaining large databases often want to use the same instance for transaction processing and for decision support. It is impossible to tune your data storage optimally for both needs, and the applications will contend for resources.

Oracle Corporation strongly advises customers to use separate databases for OLTP and DSS work, downloading data as required.

### Hybrid Systems

OLTP	DSS
Performs index searches	More full table scans
Uses B-tree indexes	Uses bitmap indexes
Uses reverse key indexes	Uses IOT tables
Needs more, small rollback segments	Fewer, large rollback segments
Should not use parallel query	Employs parallel query for large operations
PCTFREE according to expected update activity	PCTFREE can be set to 0
Shared code and bind variables	Literal variables and hints
Uses ANALYZE indexes	Histograms generation

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

### Hybrid System Issues

- OLTP needs more indexes than DSS does. They do not necessarily use the same type of indexes, because of DML/OLTP and queries/DSS issues.
- DSS needs larger rollback segments for read consistency, whereas OLTP requires many small rollback segments to avoid contention.
- OLTP should not use parallel query, whereas DSS needs it.
- DSS requires tightly packed data blocks for optimal full table scans. Individual rows will not generally be updated in a DSS environment. Therefore, PCTFREE should be set to 0 for DSS databases, where a PCTFREE of 0 for OLTP will lead to chaining, because rows are typically more dynamic.
- The cost-based optimizer takes histogram statistics into account when queries use literals and not bind variables. OLTP must use bind variables, DSS not. This means that histogram statistics collection is a loss of time and resource-consuming for OLTP transactions.



**Recommendations**

Maintaining two large databases instead of one has an obvious financial cost, however, and some organizations may still decide to run hybrid systems. For a hybrid system:

- You must keep long reports and summaries running until off-peak hours.
- You need two separate parameter files for day and night work, shutting down and starting up at the beginning and end of the working day.
- You need to configure different rollback segments for different usage.

## Parameters for Hybrid Systems

- **Memory use:**
  - **SHARED\_POOL\_SIZE**
  - **LARGE\_POOL\_SIZE**
  - **DB\_BLOCK\_BUFFERS**
  - **SORT\_AREA\_SIZE**
- **Parallel Query: Reconfigure parameters for DSS**

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

### Memory Use

The following parameters will have higher values for daytime (that is, OLTP):

- **SHARED\_POOL\_SIZE**
- **DB\_BLOCK\_BUFFERS**
- **SORT\_AREA\_SIZE**

### Parallel Query

- During the day, both **PARALLEL\_MIN\_SERVERS** and **PARALLEL\_MAX\_SERVERS** could be set to zero to prevent parallelization.
- Full table scans of large tables can be restricted: use different daytime profiles to limit **LOGICAL\_READS\_PER\_CALL** or **CPU\_PER\_CALL**, and assign these profiles to users.

In off-peak times, you can reset these parameters to the appropriate number and reassign nighttime profiles to the DSS querying users.

Note that Parallel Query processes use the **LARGE\_POOL** for passing messages between each other.

## Hybrid Systems Configuration

- **Online rollback segments:**
  - **More small ones during the day**
  - **Fewer, large ones at night**
- **Multithreaded server (MTS):**  
**For peak-time use, not for DSS**

Copyright © Oracle Corporation, 1999. All rights reserved. ORACLE<sup>®</sup>

## Rollback Segments

If you can afford the space, you may want to use:

- Many small rollback segments during the day that remain offline at night
- Fewer large ones at night that remain offline during the day

If you cannot afford that amount of space, before shutting down the database at night, schedule a script that replaces the day rollback segments with the night rollback segments, and vice versa at startup.

## Summary

### Summary

In this lesson, you should have learned that:

- Application tuning often results in the greatest performance benefits.
- You tune CBO with parameters and hints.
- You use stored outlines for plan stability.
- You should apply available data access methods appropriately.

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

## Summary

**For OLTP, try to reach the following goals:**

- **Immediate access to small amounts of data (indexing, hashing)**
- **Immediate concurrent access to transaction tables**
- **Shared code to cut down parse time**
- **No space allocation during peak hours**

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

## Summary

**For DSS, try to reach the following goals:**

- **Data tightly packed into large blocks**
- **Careful tuning of queries**
- **Histograms generation**
- **Query rewrites using materialized views and dimensions**
- **Well-configured Parallel Query support**

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

## Quick Reference

Context	Reference
Parameters	CREATE_STORED_OUTLINES MAX_DUMP_FILE_SIZE USER_DUMP_DEST TIMED_STATISTICS SQL_TRACE USE_STORED_OUTLINES OPTIMIZER_MODE QUERY_REWRITE_ENABLED QUERY_REWRITE_INTEGRITY
Dynamic performance views	
Data dictionary views	Various USER_*, ALL_*, and DBA_* views
Commands	ALTER SYSTEM ALTER SESSION ANALYZE CREATE BITMAP INDEX CREATE INDEX ... REVERSE CREATE MATERIALIZED VIEW CREATE OUTLINE CREATE TABLE ... ORGANIZATION INDEX EXPLAIN PLAN
Procedures	SYS.DBMS_MVIEW package SYS.DBMS_OLAP package SYS.DBMS_SESSION.SET_SQL_TRACE SYS.DBMS_SYSTEM.SET_SQL_TRACE_IN_SESSION SYS.DBMS_STATS package SYS.UTLNL_PKG package

