Enterprise DBA Part 2: Performance and Tuning

Slides

30052GC10

Production 1.0

September 1999

M09217

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

1

Course Introduction

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

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

2

Tuning Overview

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**

Tuning Questions

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

Examples of Measurable Tuning Goals

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

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

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.

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

3

Oracle Alert and Trace Files

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

Diagnostic Information

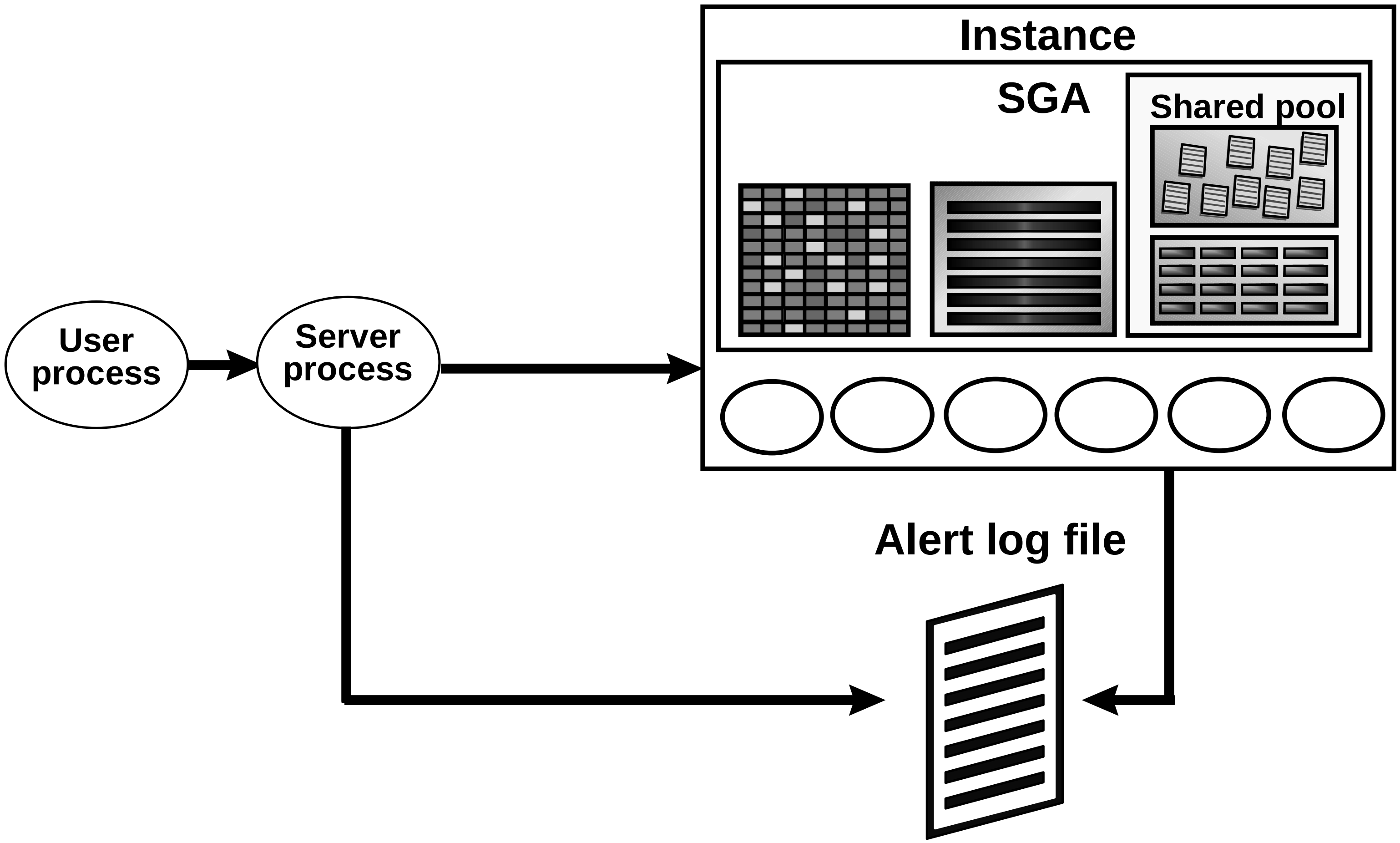
Trace files:

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

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.

Controlling the Alert Log File

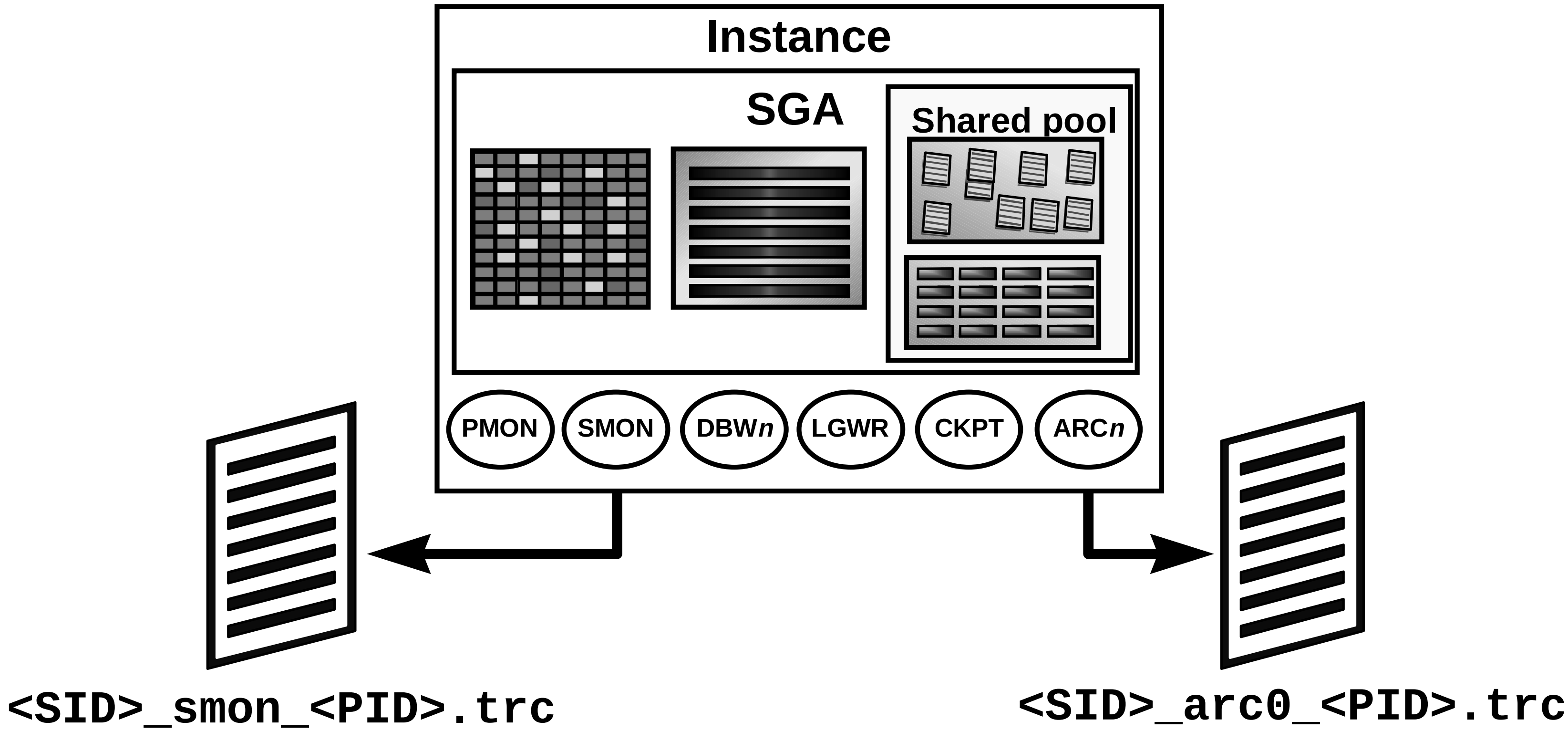


BACKGROUND_DUMP_DEST= $ORACLE_HOME/rdbms/log

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.

Controlling the Background Processes Trace Files

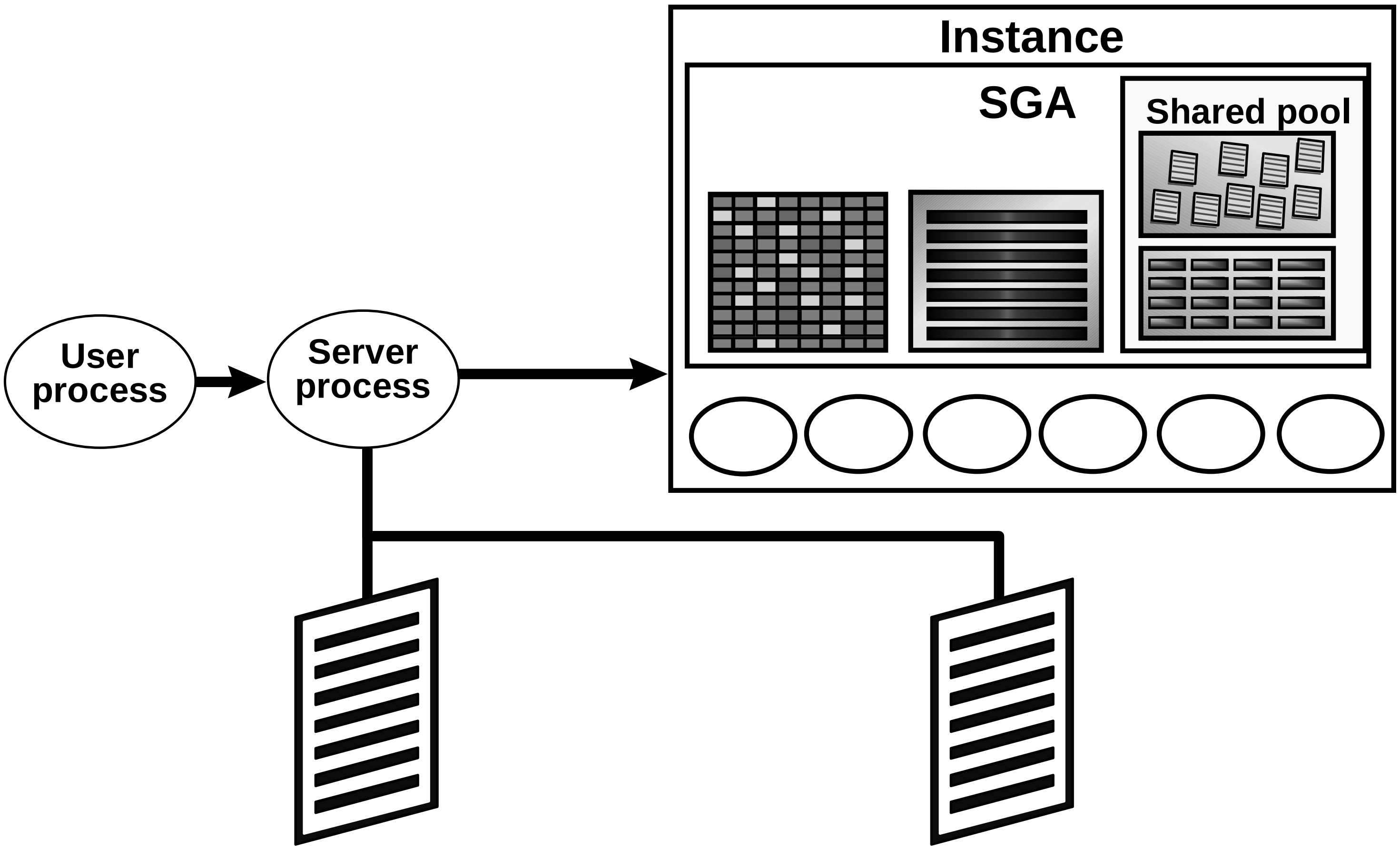


In the directory defined in BACKGROUND_DUMP_DEST

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.

Controlling the User Trace Files



`<SID>_ora_<PID>.trc`

Alert log file

In the directory defined in USER_DUMP_DEST

ORACLE®

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

4

Utilities and Dynamic Performance Views

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

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**

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.

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

Topics for Troubleshooting and Tuning

System-Wide Statistics

Instance/Database

V$DATABASE	T
V$INSTANCE	T
V$OPTION	T
V$PARAMETER	T/P
V$BACKUP	T
V$PX_PROCESS_SYSSTAT	T/P
V$PROCESS	T
V$WAITSTAT	T/P
V$SYSTEM_EVENT	T/P

Disk

V$DATAFILE	T/P
V$FILESTAT	T/P
V$LOG	T
V$LOG_HISTORY	T
V$DBFILE	T/P
V$TEMPFILE	P
V$TEMPSTAT	P

Memory

V$BUFFER_POOL_STATISTICS	T/P
V$DB_OBJECT_CACHE	T
V$LIBRARYCACHE	P
V$ROWCACHE	P
V$SYSSTAT	T/P
V$SGASTAT	P

Contention

V$LOCK	T/P
V$ROLLNAME	T/P
V$ROLLSTAT	T/P
V$WAITSTAT	T/P
V$LATCH	T/P

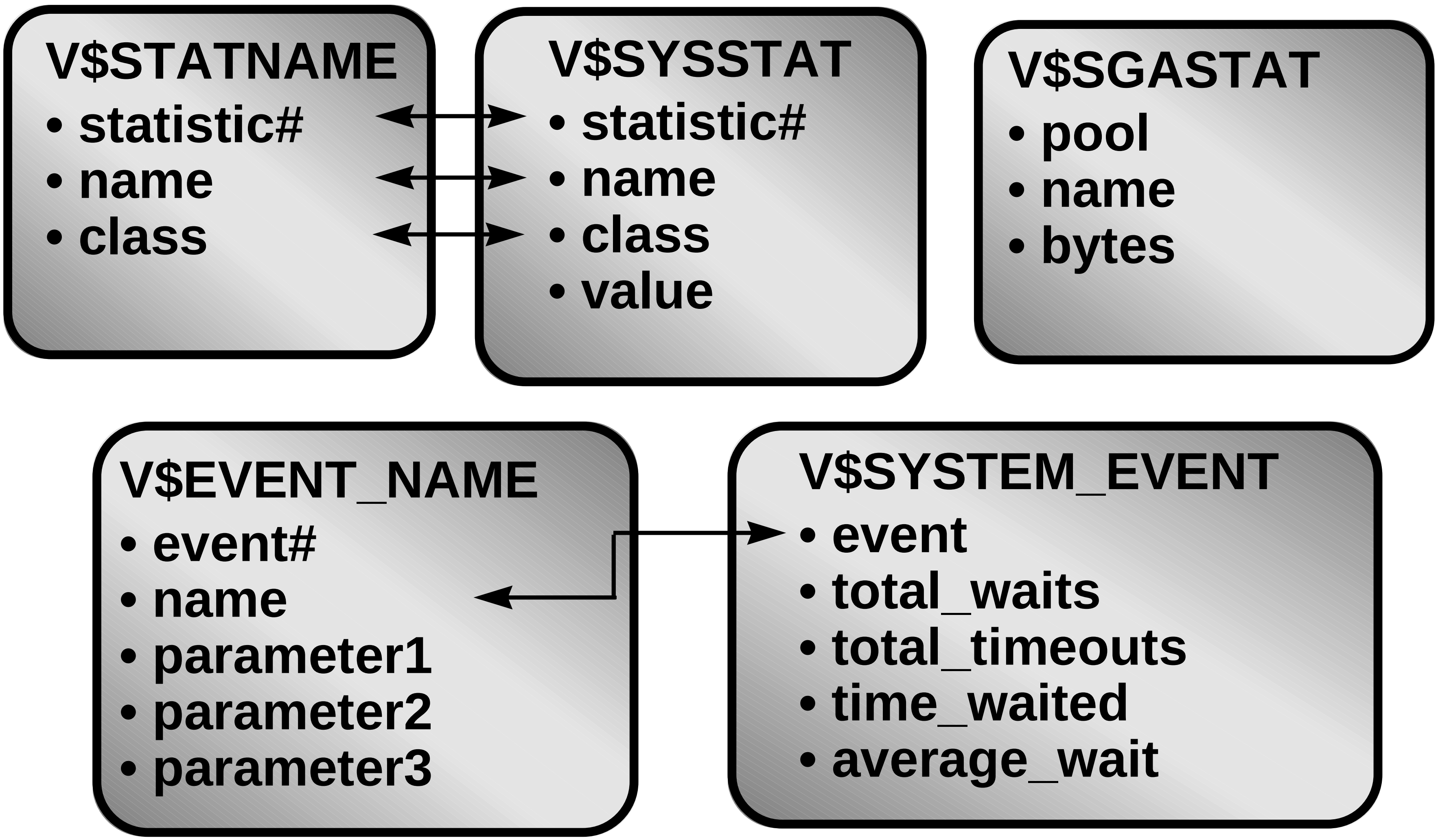
Session-Related Statistics

User/Session

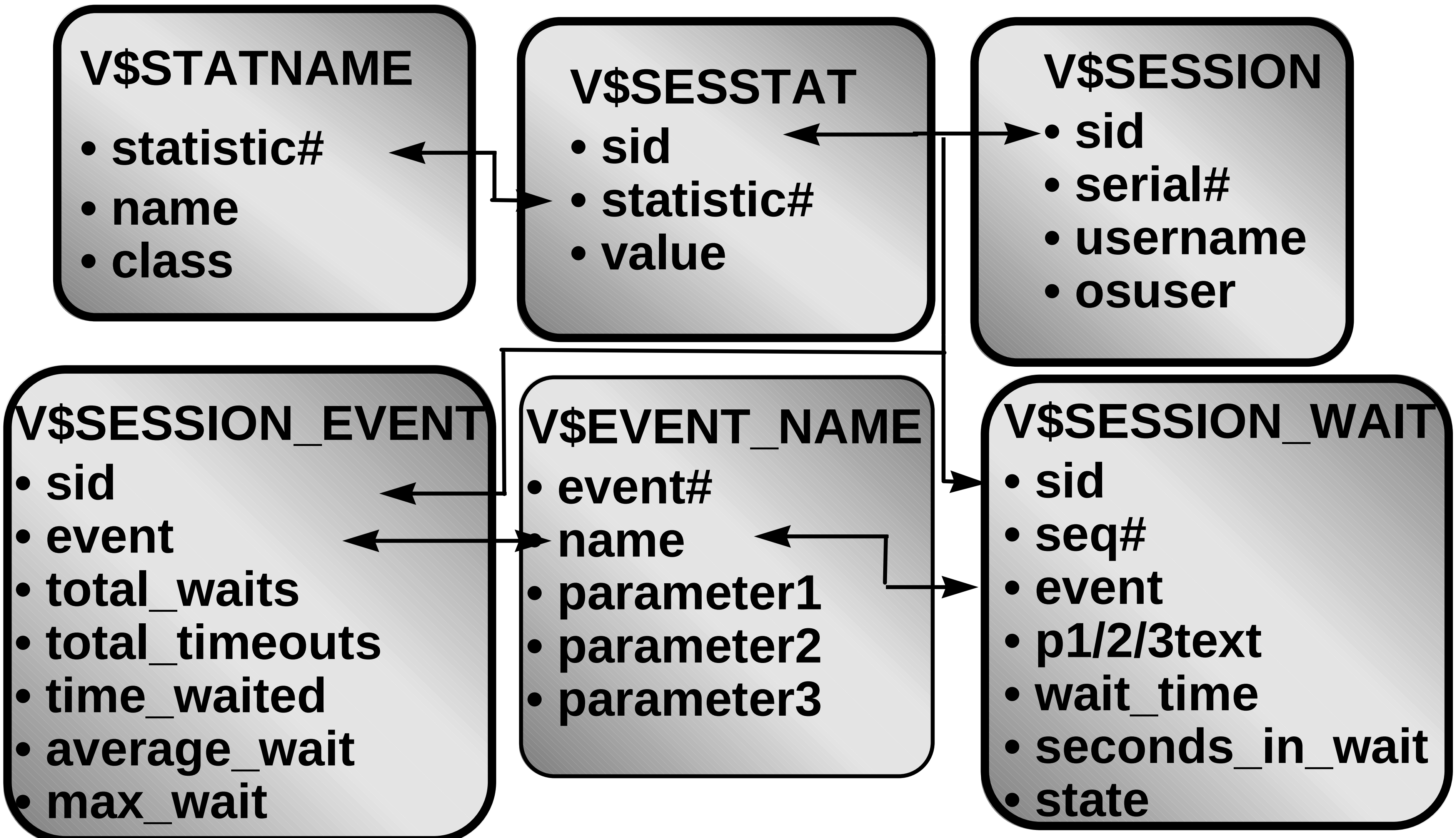
V$LOCK	P
V$OPEN_CURSOR	T
V$PROCESS	T
V$SORT_USAGE	T/P
V$SESSION	T/P
V$SESSTAT	T/P
V$TRANSACTION	T
V$SESSION_EVENT	T/P
V$SESSION_WAIT	T/P
V$PX_SESSTAT	P
V$PX_SESSION	P
V$SESSION_OBJECT_CACHE	P

T for Troubleshooting
T/P for Troubleshooting/Performance

Collecting System-Wide Statistics



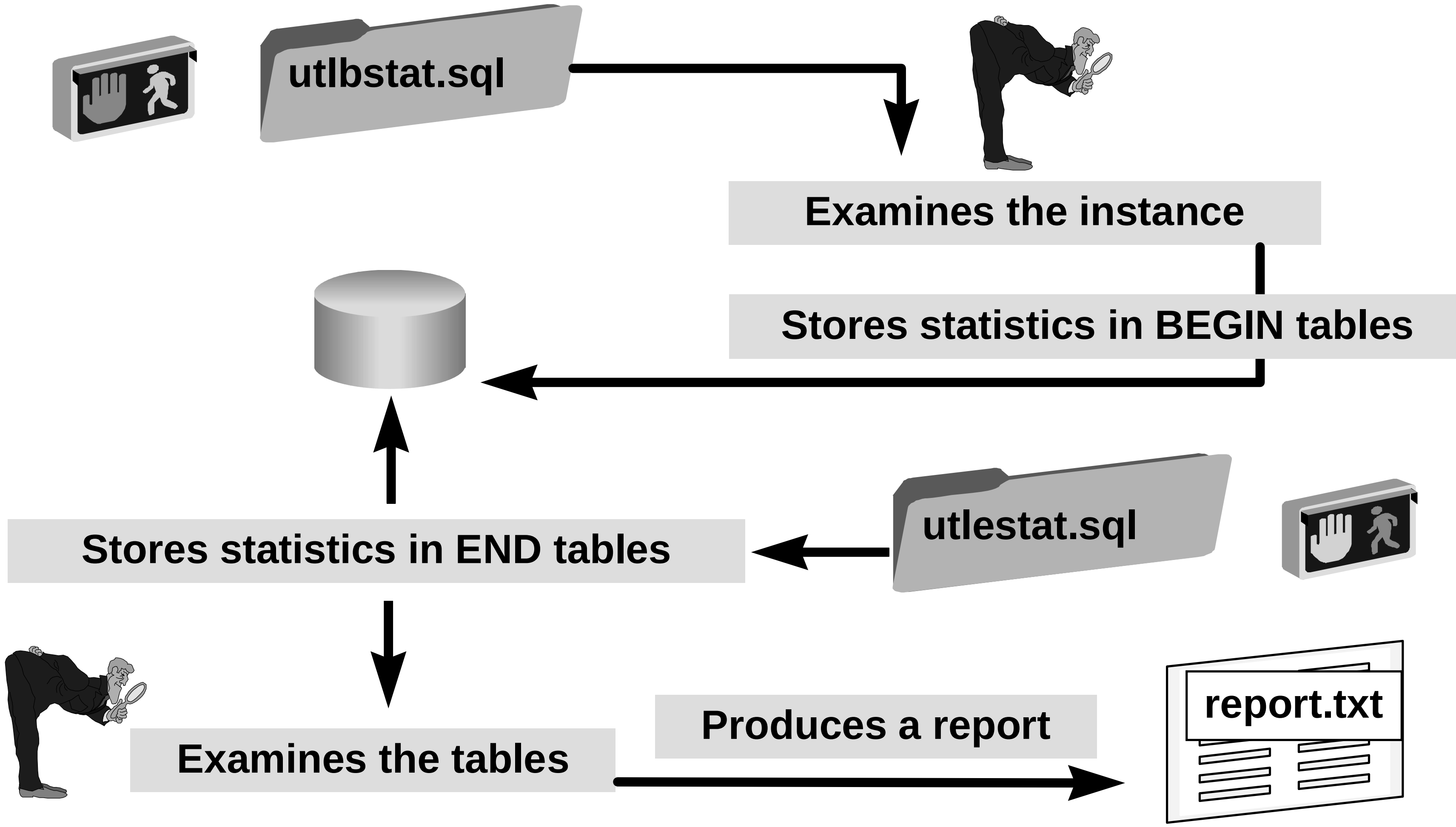
Collecting Session-Related Statistics



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

Gather Statistics



 ORACLE®

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

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.
```

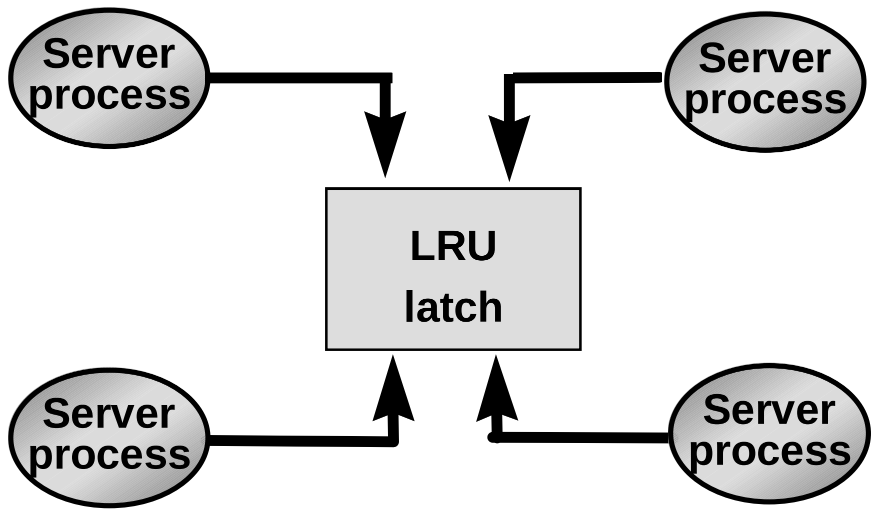
I/O Statistics

```
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_     WRITES BLKS_    WRITE_
                                              READ      TIME              WRT      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.
```

Latches: Overview

What is contention?

A contention exists when multiple server processes contend for the same resources.



ORACLE®

Latches

Contention areas that the DBA can tune:

- **Redo allocation latch**
- **Redo copy latch**
- **LRU latch**

Latch Types

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

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**

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.
```

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

V$SYSTEM_EVENT View

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

```
EVENT                   TOTAL_    TOTAL_    TIME_      AVERAGE_
                         WAITS  TIMEOUTS   WAITED          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.
```

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
```

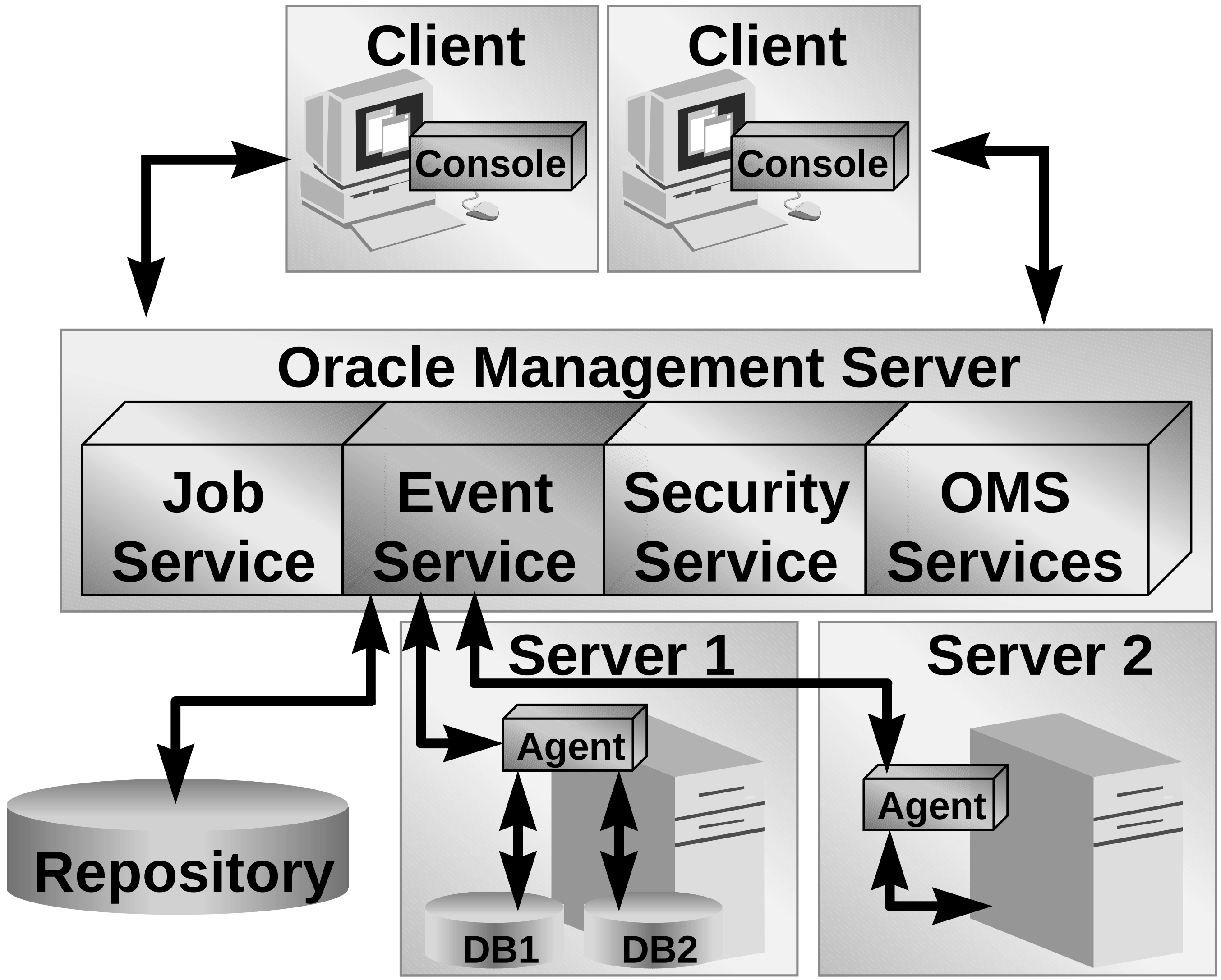
V$SESSION_WAIT View

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

```
SID   SEQ#  EVENT                               WAIT   STATE
                                                TIME
---- ------ ---------------------------      -----   -------
  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
```


ORACLE®

Enterprise Manager (EM Version 2)



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

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)**

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)**

Predefined Event Tests

Resource management events:

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

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)**

Event Parameters

Parameters:

- **Warning and alert thresholds**
- **Number of occurrences**
- **Focused monitoring:**
  - **SEGMENT_OWNER**
  - **SEGMENT_TYPE**
  - **SEGMENT_NAME**
  - **Any criteria related to the area**

Fix the Problem Detected by the Event

- Manually
- Automatically by fixit jobs

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

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

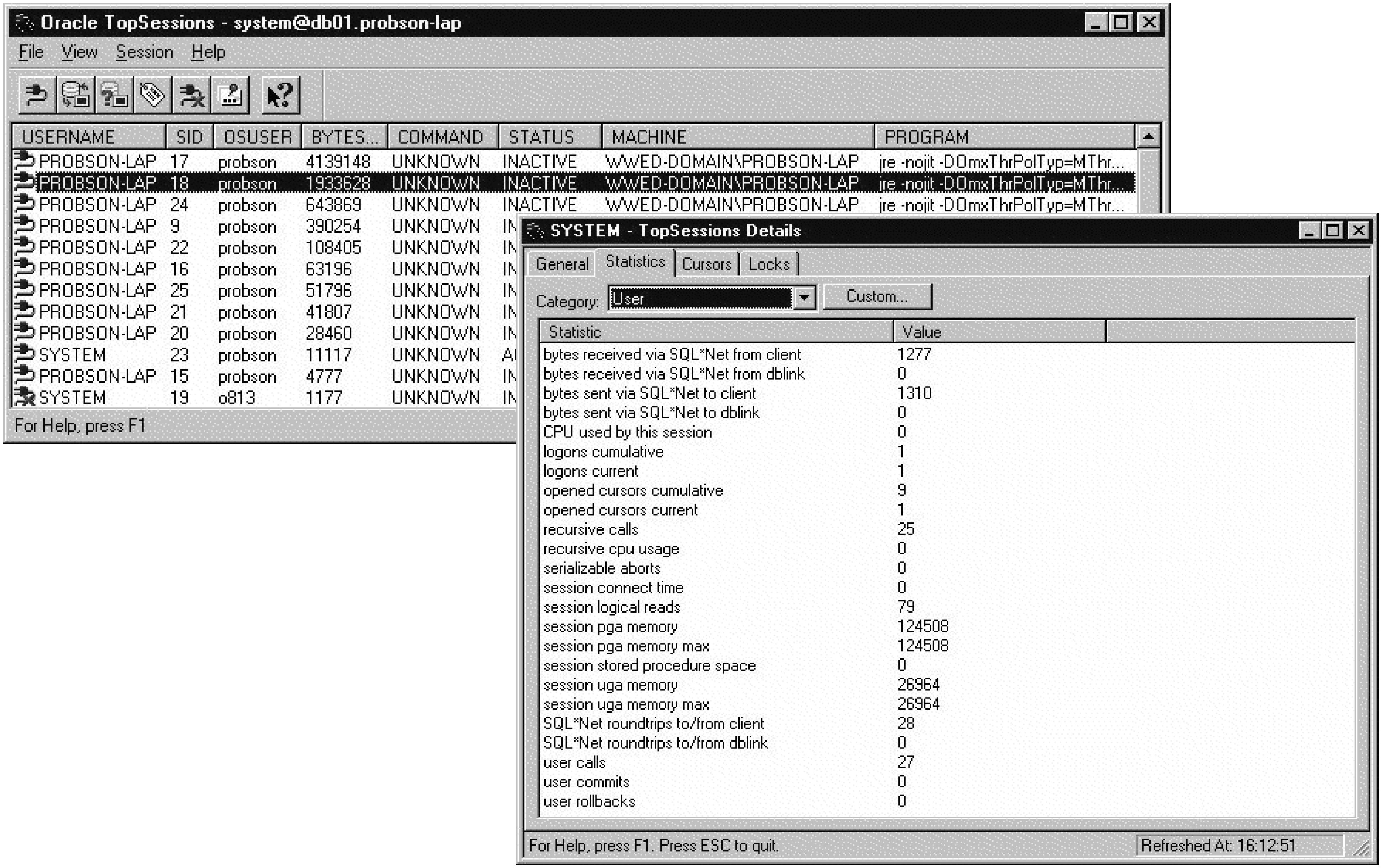
Performance Manager

Predefined scopes of statistics:

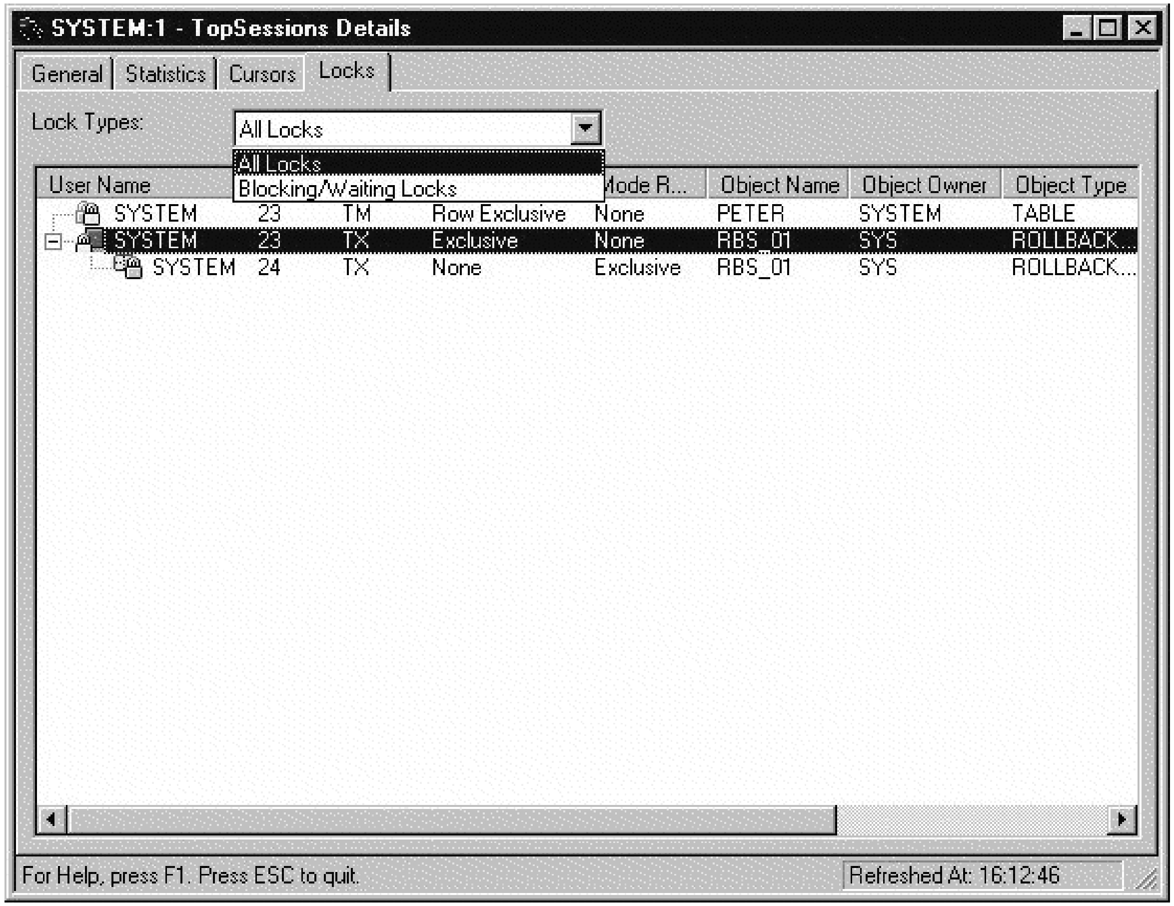
- **I/O**
- **Contention**
- **Database instance**
- **Load**
- **Memory**
- **Top resource consumers**
- **Overview of performance**
  - **Overview of cache utilization**
  - **Overview of user activity**
  - **Overview of throughput**
  - **Overview of performance default chart**

User-defined charts

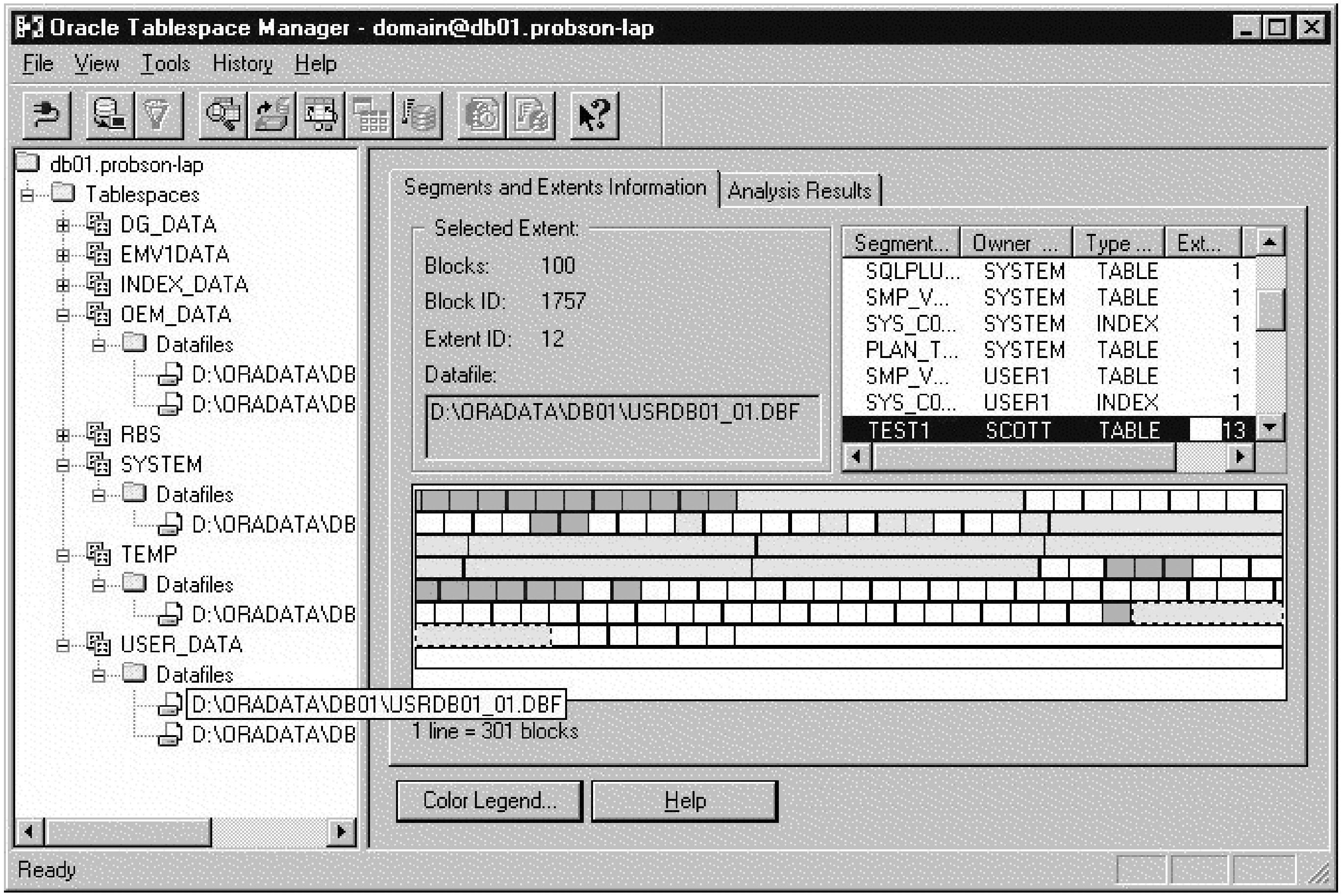
TopSessions



TopSessions: Locks



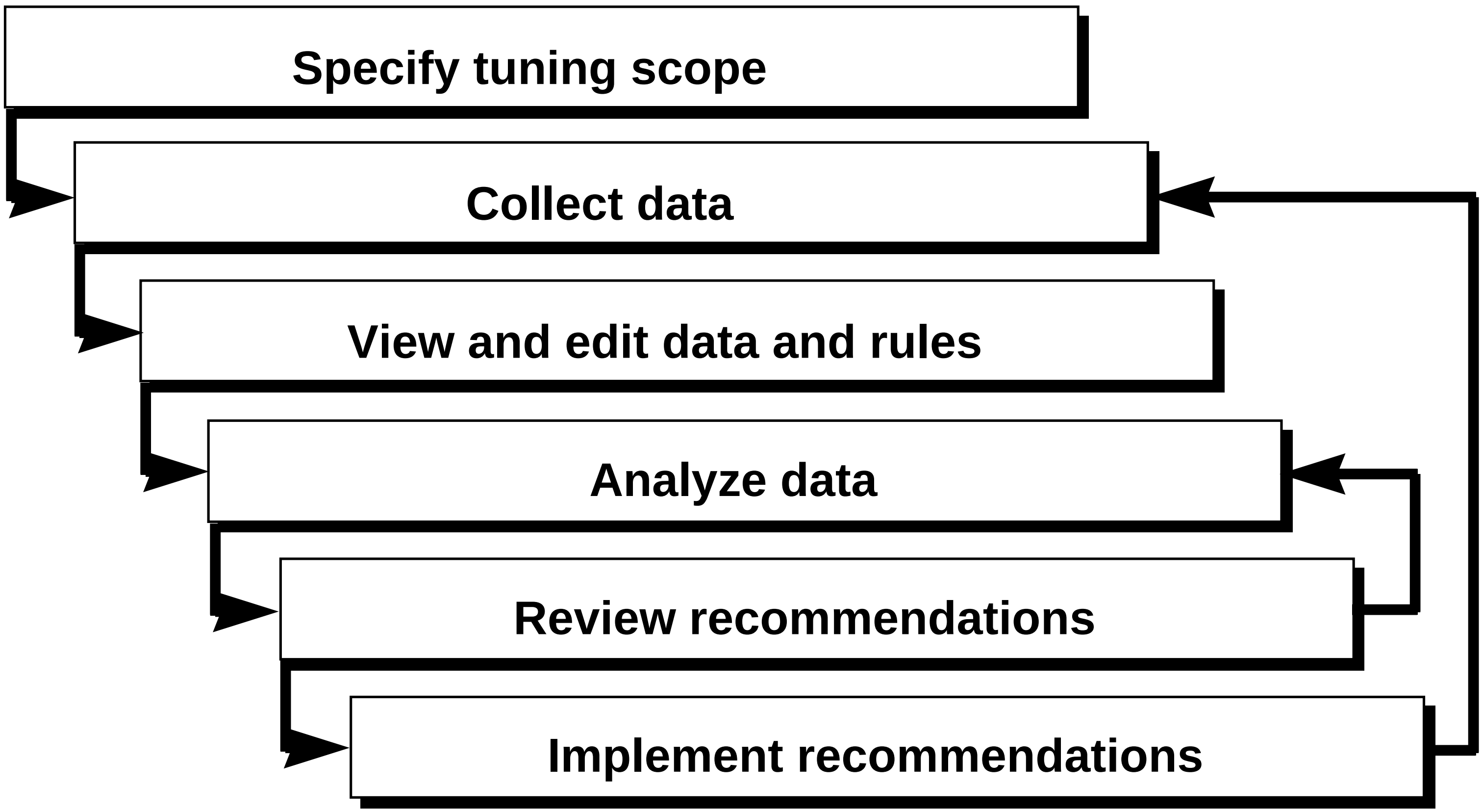
Tablespace Manager



Common Uses of Oracle Trace Manager

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

Overview of Oracle Expert Tuning Methodology



Tuning Session Scope

Four major tuning categories:

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

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`

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

5

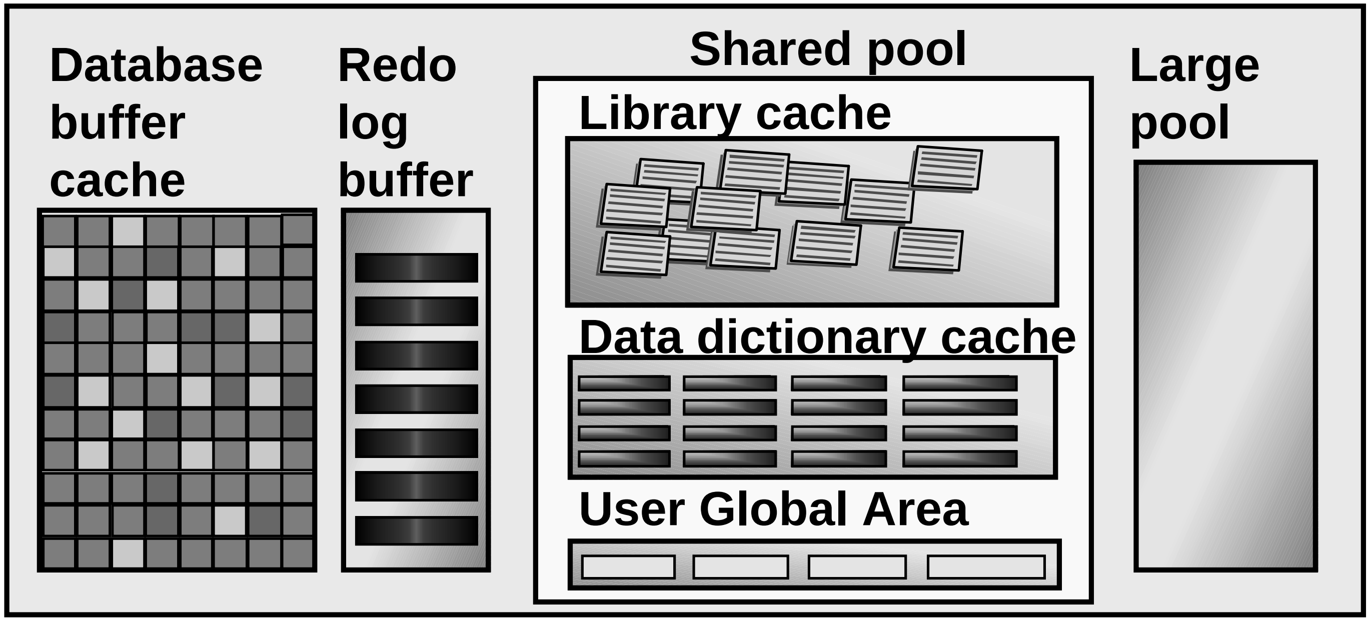
Tuning the Shared Pool

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

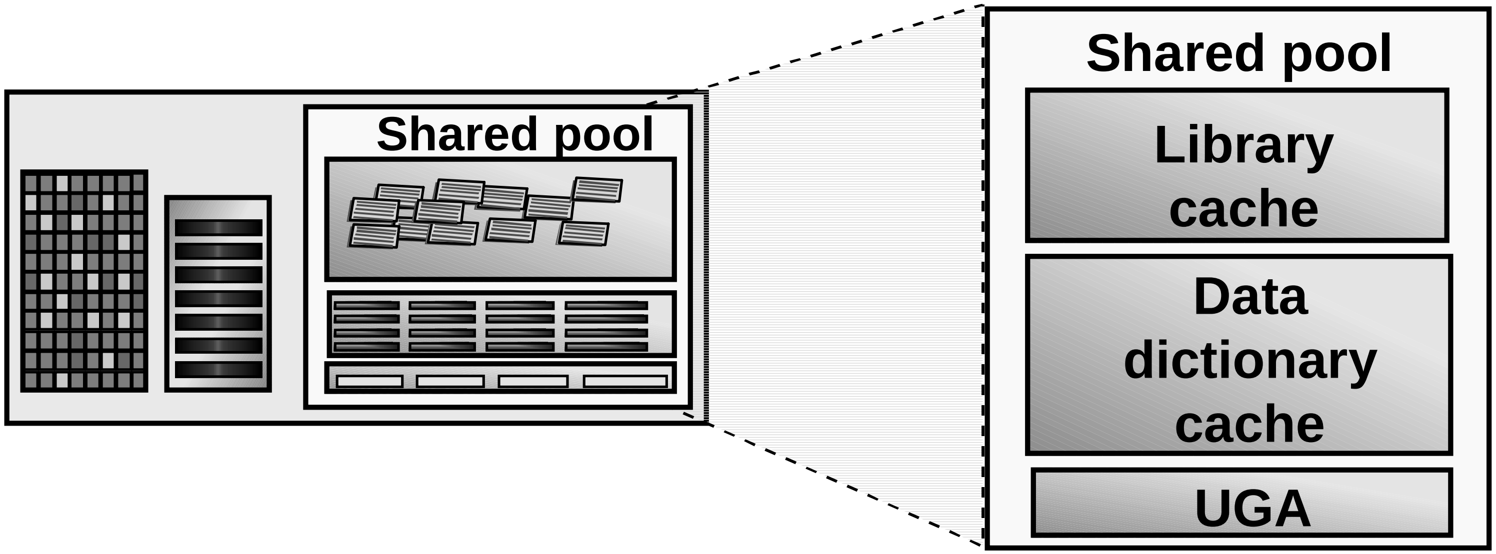
The Shared Global Area



Shared pool:

- **Library cache**
- **Data dictionary cache**
- **UGA for multithreaded server connections**

The Shared Pool

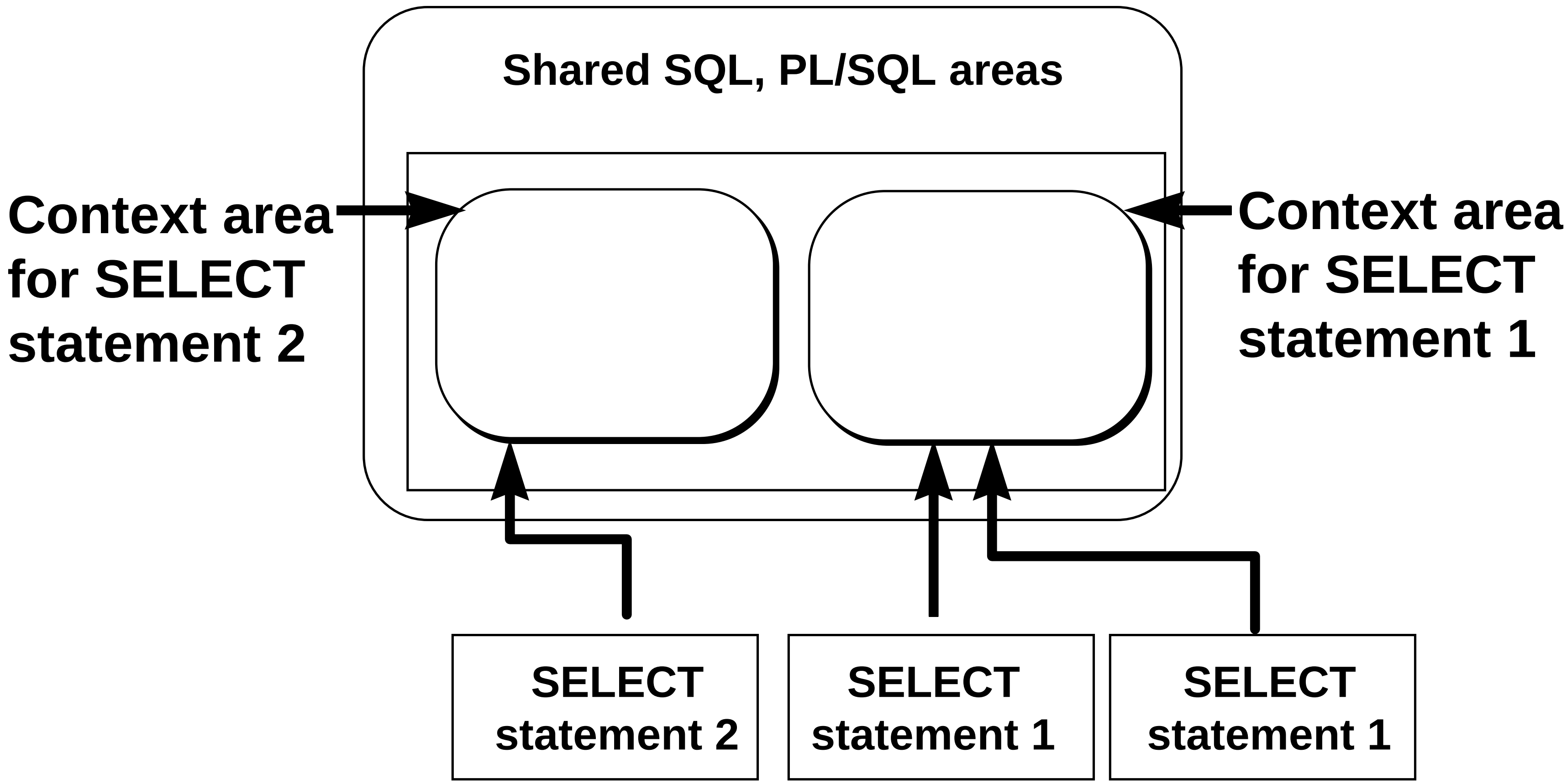


- **Size defined by SHARED_POOL_SIZE**
- **Library cache contains statement text, parsed code, and execution plan**
- **Data dictionary cache contains table, column definitions, and privileges from the data dictionary tables**
- **UGA contains MTS users' session information**

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

The Library Cache



ORACLE®

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**

Tuning the Library Cache

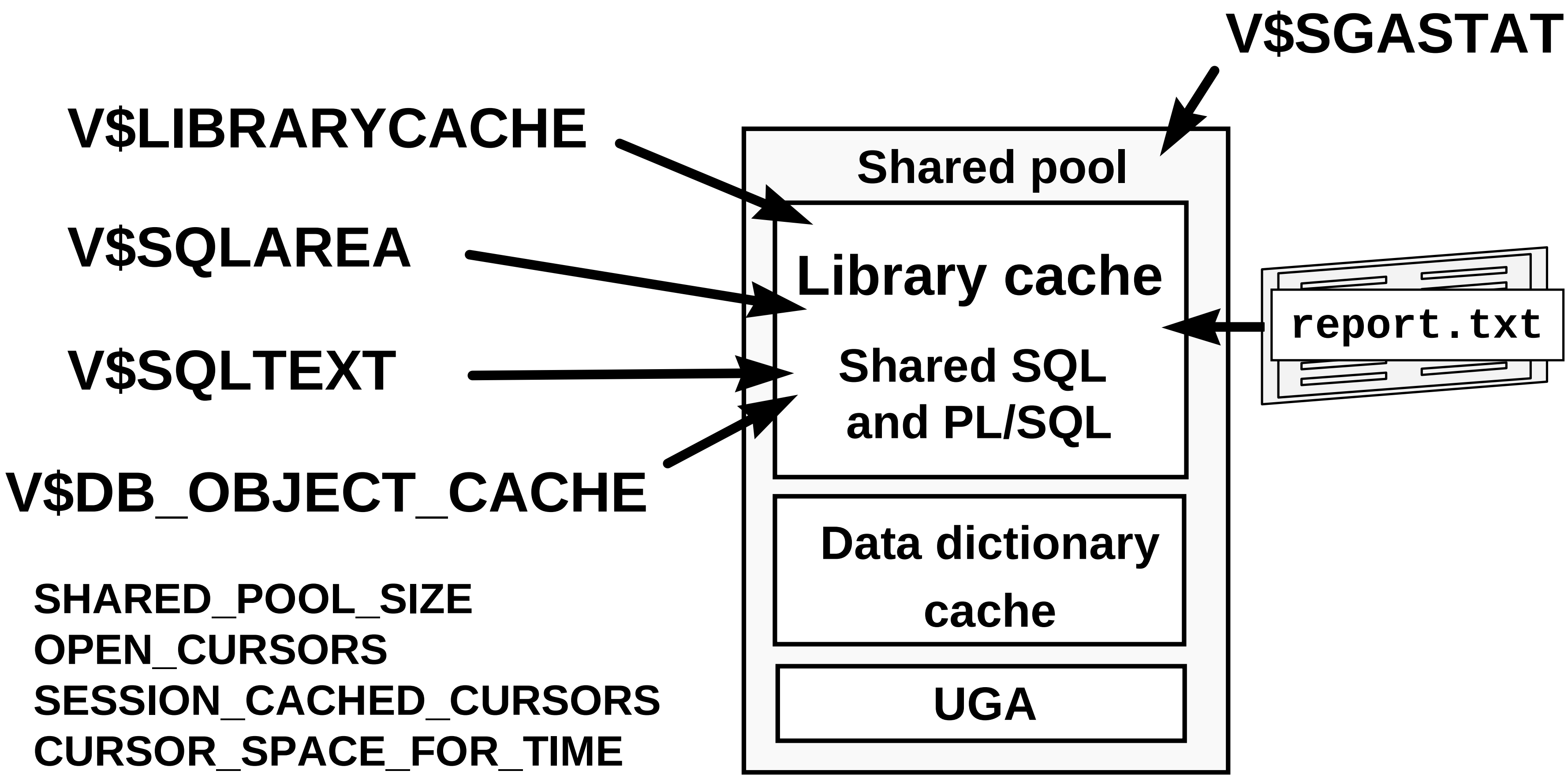
Avoid fragmentation by:

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

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**

Diagnostic Tools
for Tuning the Library Cache



SHARED_POOL_SIZE
OPEN_CURSORS
SESSION_CACHED_CURSORS
CURSOR_SPACE_FOR_TIME

Are Cursors Being Shared?

Check GETHITRATIO in V$LIBRARYCACHE:

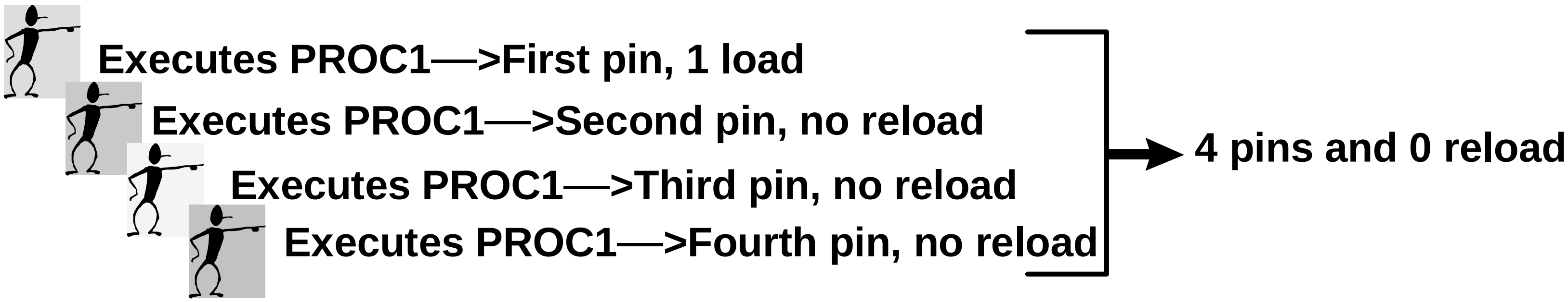
```
SQL> select gethitratio
  2  from v$librarycache
  3  where namespace = ‘SQL AREA’;
```

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 =%’;
```

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$librarycache;
Executions Cache Misses sum(reloads)/sum(pins)
---------- ------------ ----------------------
      2641           10               .00378644
```

Guidelines: Library Cache Reloads

`report.txt` **output::**

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

If the reloads-to-pins ratio is greater than 1%, increase the SHARED_POOL_SIZE parameter.

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$librarycache;
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
```

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.

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
```

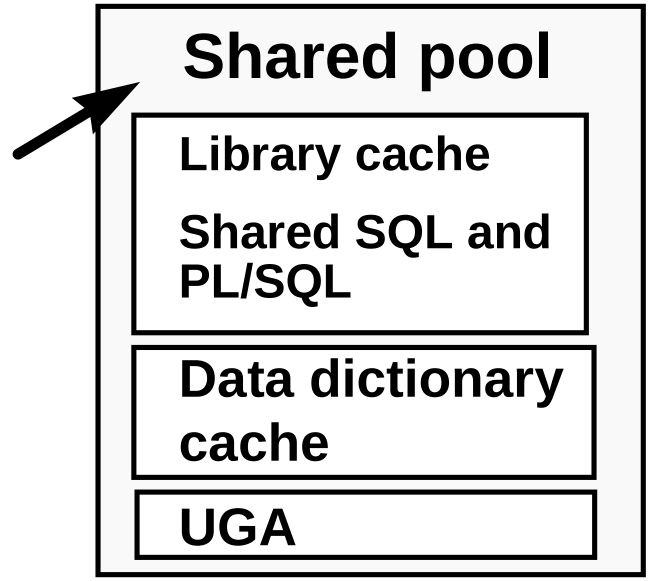
SQL statements:

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

Large Memory Requirements

- Satisfy requests for large contiguous memory
- Reserve unfragmentable memory within the shared pool

V$SHARED_POOL_RESERVED



SHARED_POOL_SIZE
SHARED_POOL_RESERVED_SIZE

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

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');
```

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;
```

Other Parameters Affecting the Library Cache

- **OPEN_CURSORS**
- **CURSOR_SPACE_FOR_TIME**
- **SESSION_CACHED_CURSORS**

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

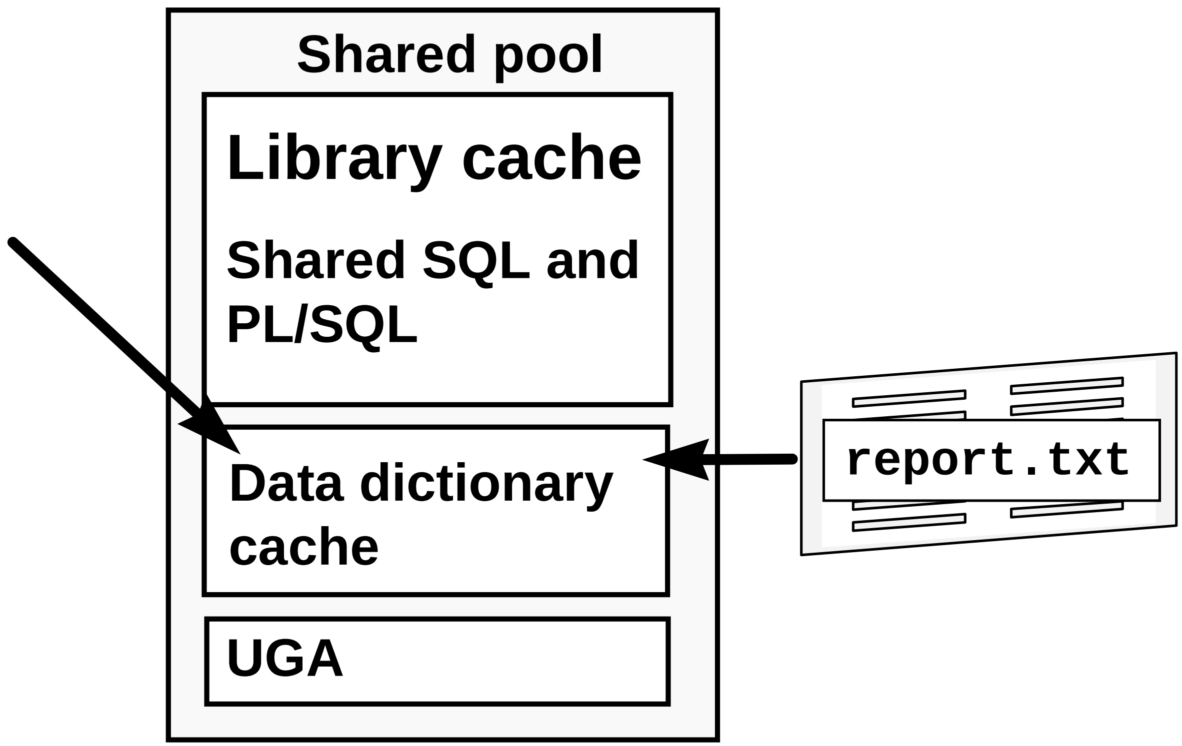
Diagnostic Tools for Tuning the Data Dictionary Cache

V$ROWCACHE:

PARAMETER

GETS

GETMISSES



SHARED_POOL_SIZE

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
```

Guidelines: Dictionary Cache Misses

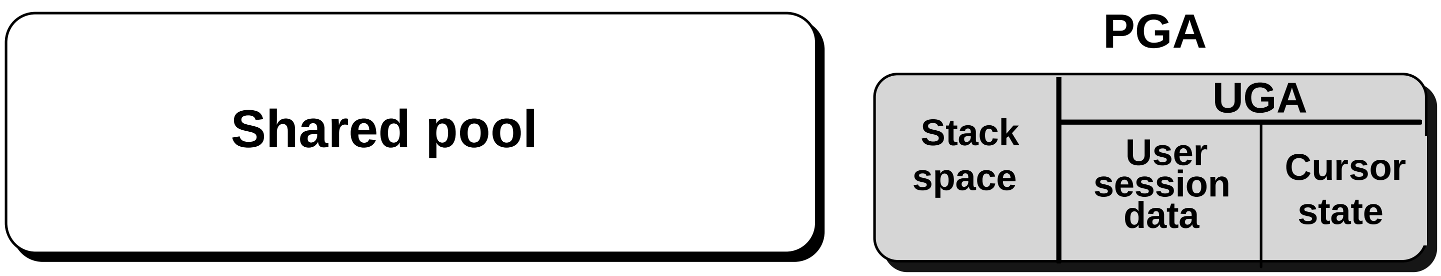
`report.txt` output:

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

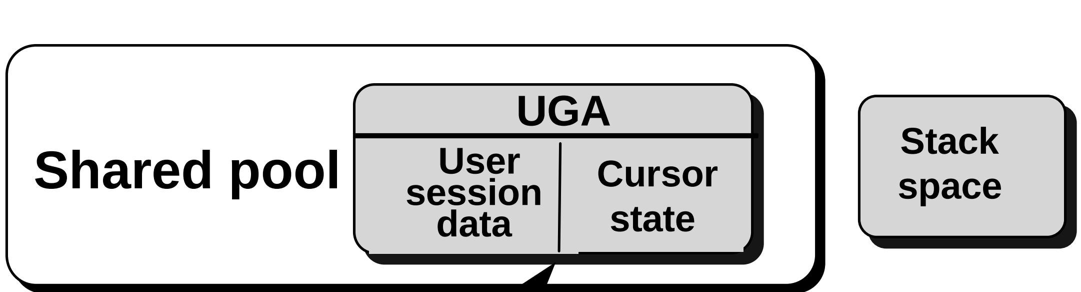
If there are too many cache misses, increase the parameter SHARED_POOL_SIZE.

UGA and MTS

Dedicated server connection:



Multithreaded server connection:



V$STATNAME

V$SESSTAT

V$MYSTAT

OPEN_CURSORS
SESSION_CACHED_CURSORS

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#;
```

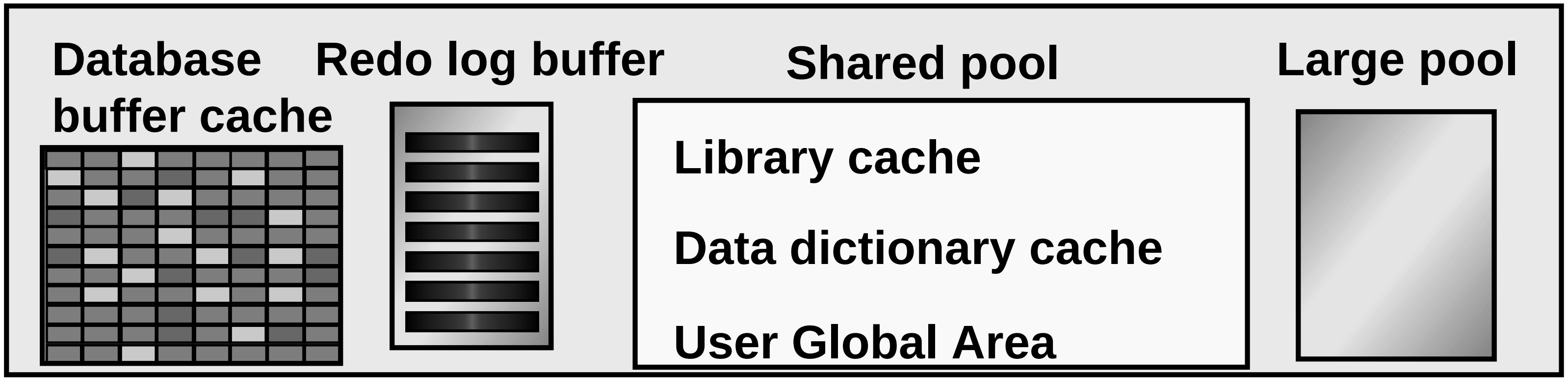
UGA space used by all MTS users:

```
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#;
```

Large Pool



- Can be configured as a separate memory area in the SGA, used for memory with:
  - I/O server processes: DBWR_IO_SLAVES
  - Oracle backup and restore operations
  - Session memory for the multithreaded servers
  - Parallel query messaging
- Is useful in these situations to avoid performance overhead caused by shrinking the shared SQL cache
- Is sized by the LARGE_POOL_SIZE parameter

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

6

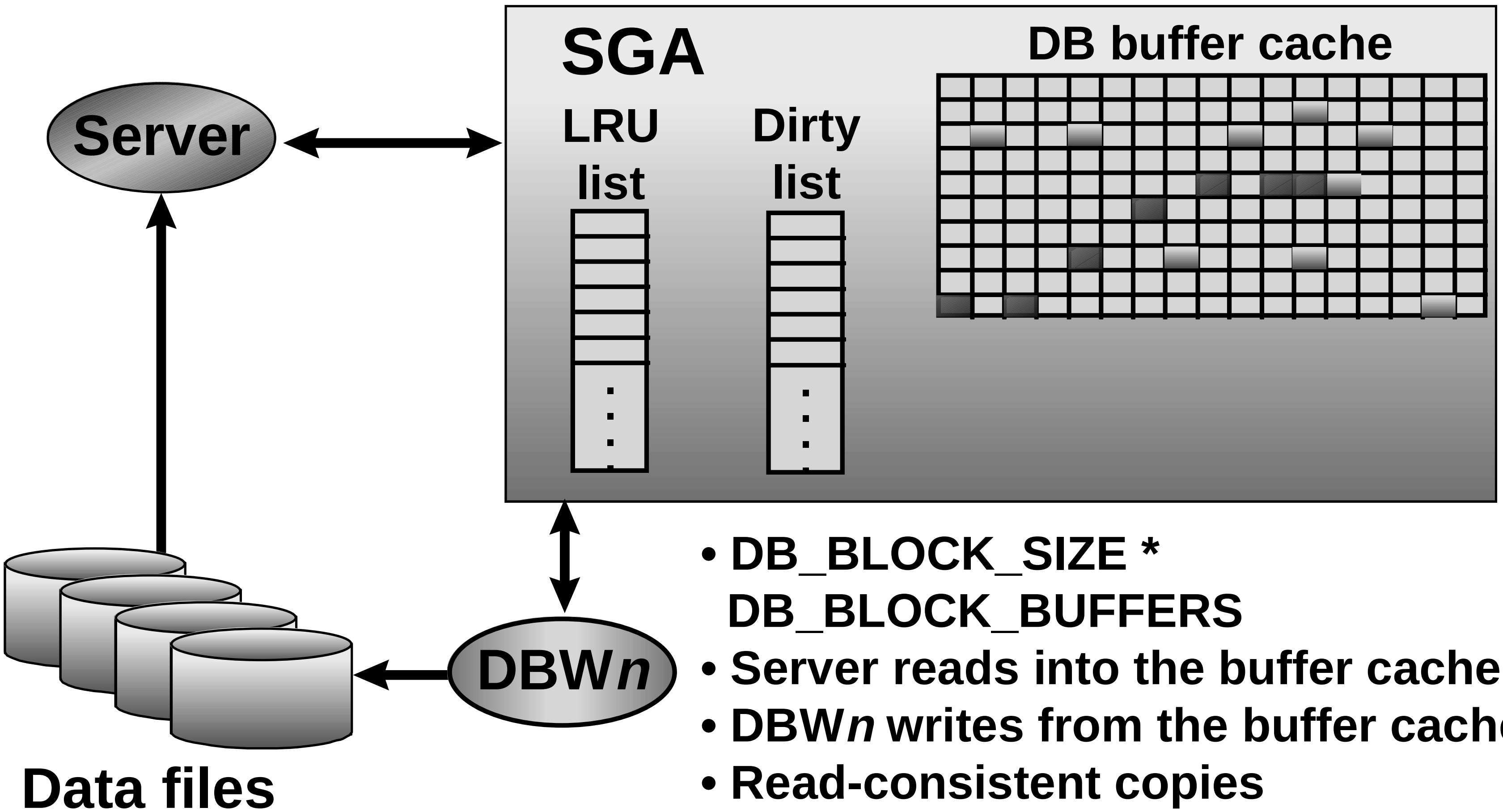
Tuning the Buffer Cache

Objectives

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

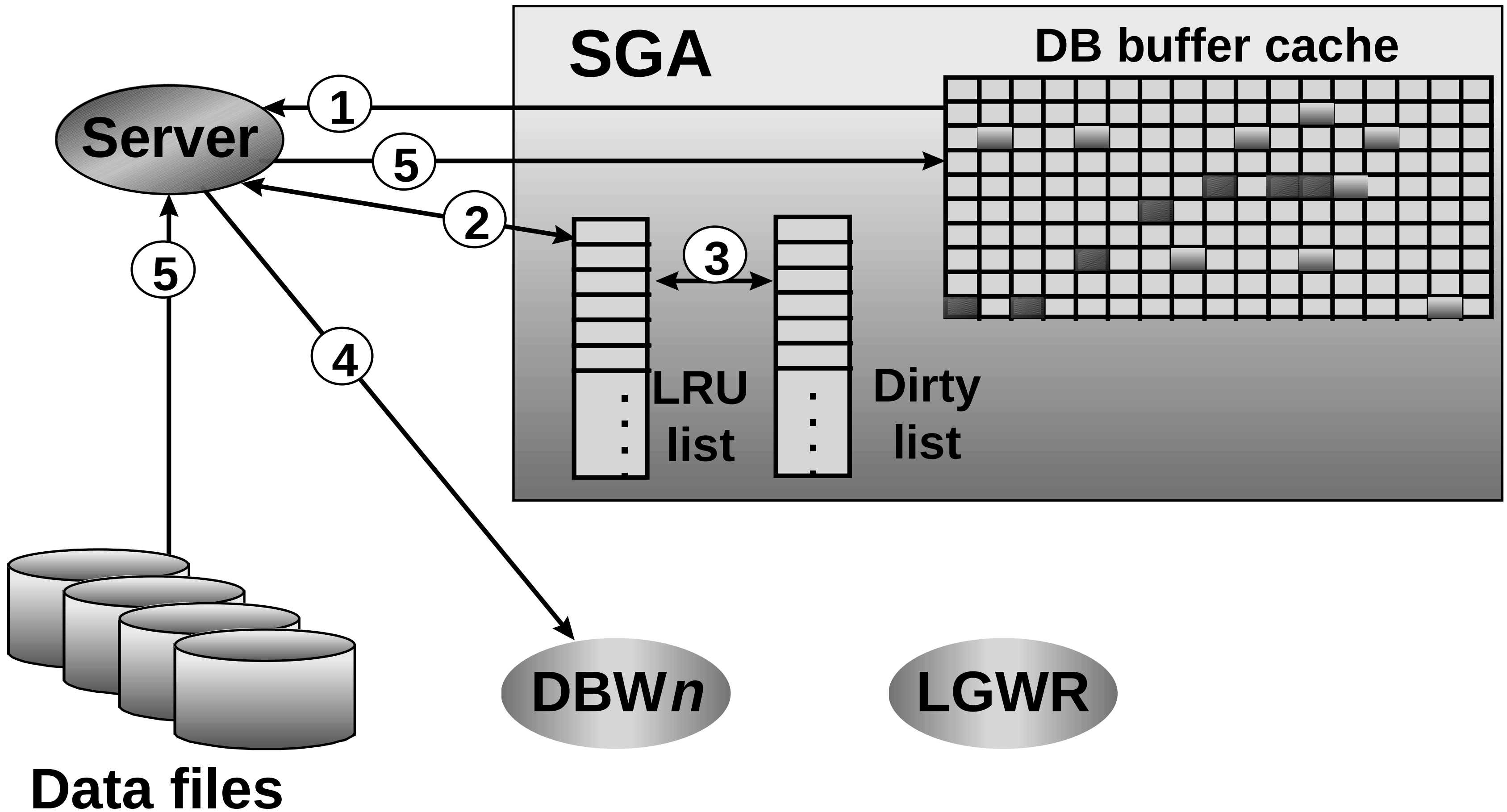
- **Describe how the buffer cache is managed**
- **Calculate and tune the buffer cache hit ratio**
- **Tune the buffer cache hit ratio by adding or removing buffers**
- **Create multiple buffer pools**
- **Size multiple pools**
- **Monitor buffer cache usage**
- **Make appropriate use of table caching**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

Overview

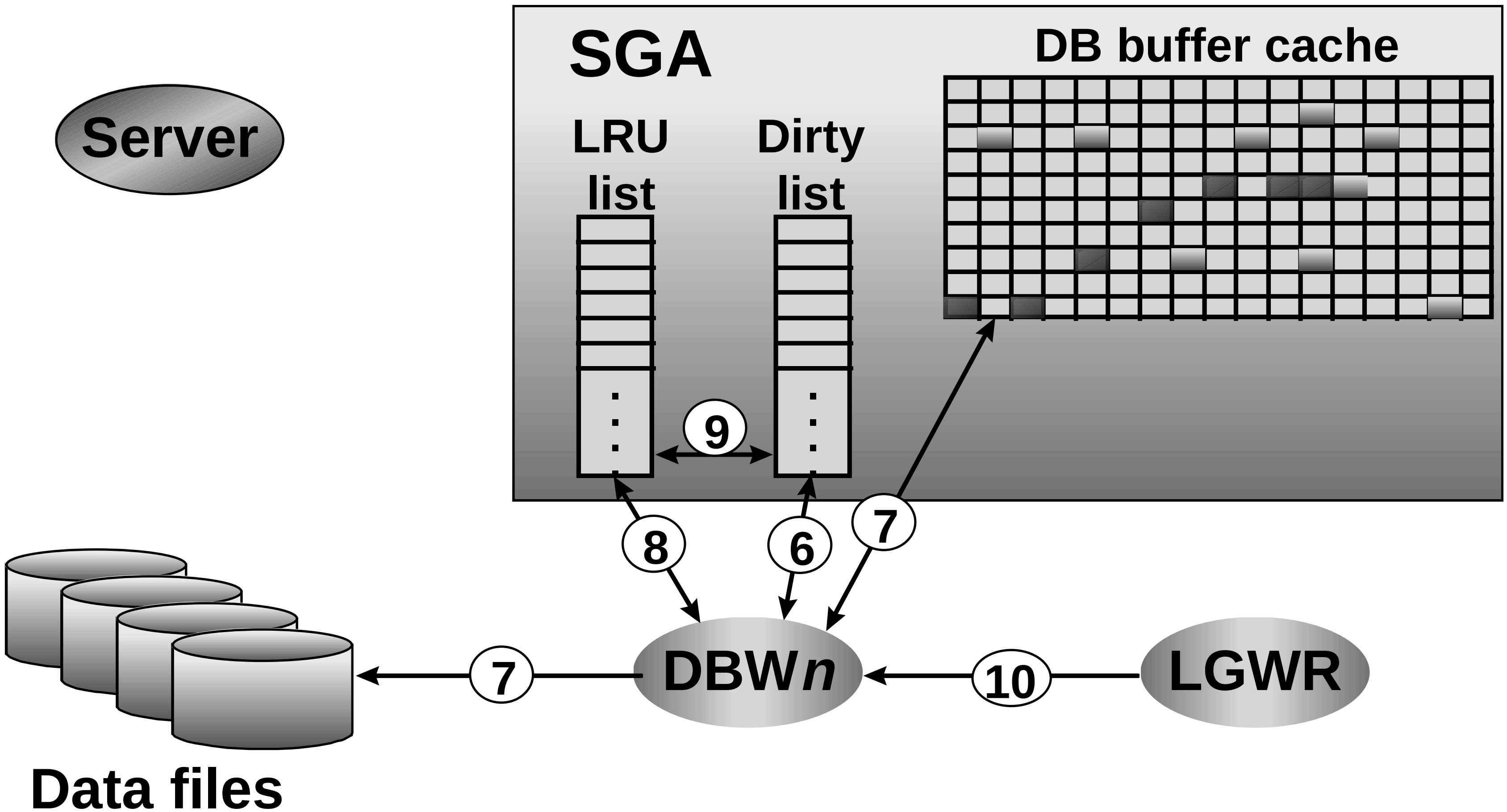


- DB_BLOCK_SIZE * DB_BLOCK_BUFFERS
- Server reads into the buffer cache
- DBW*n* writes from the buffer cache
- Read-consistent copies
- Free, dirty, or pinned state

Managing the Data Buffer Cache



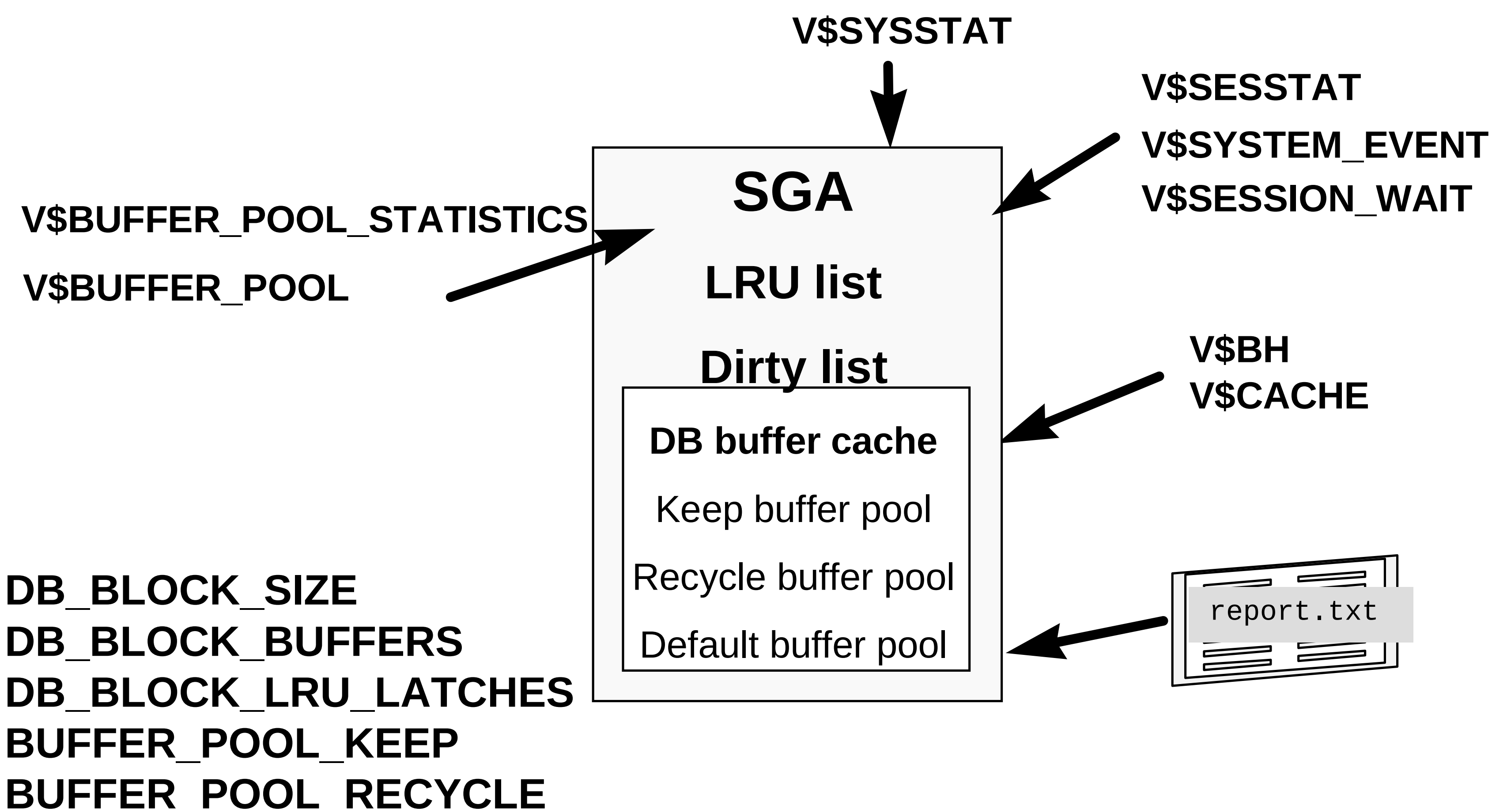
Managing the Data Buffer Cache



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

Diagnostic Tools



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
```

Guidelines for Using the Cache Hit Ratio

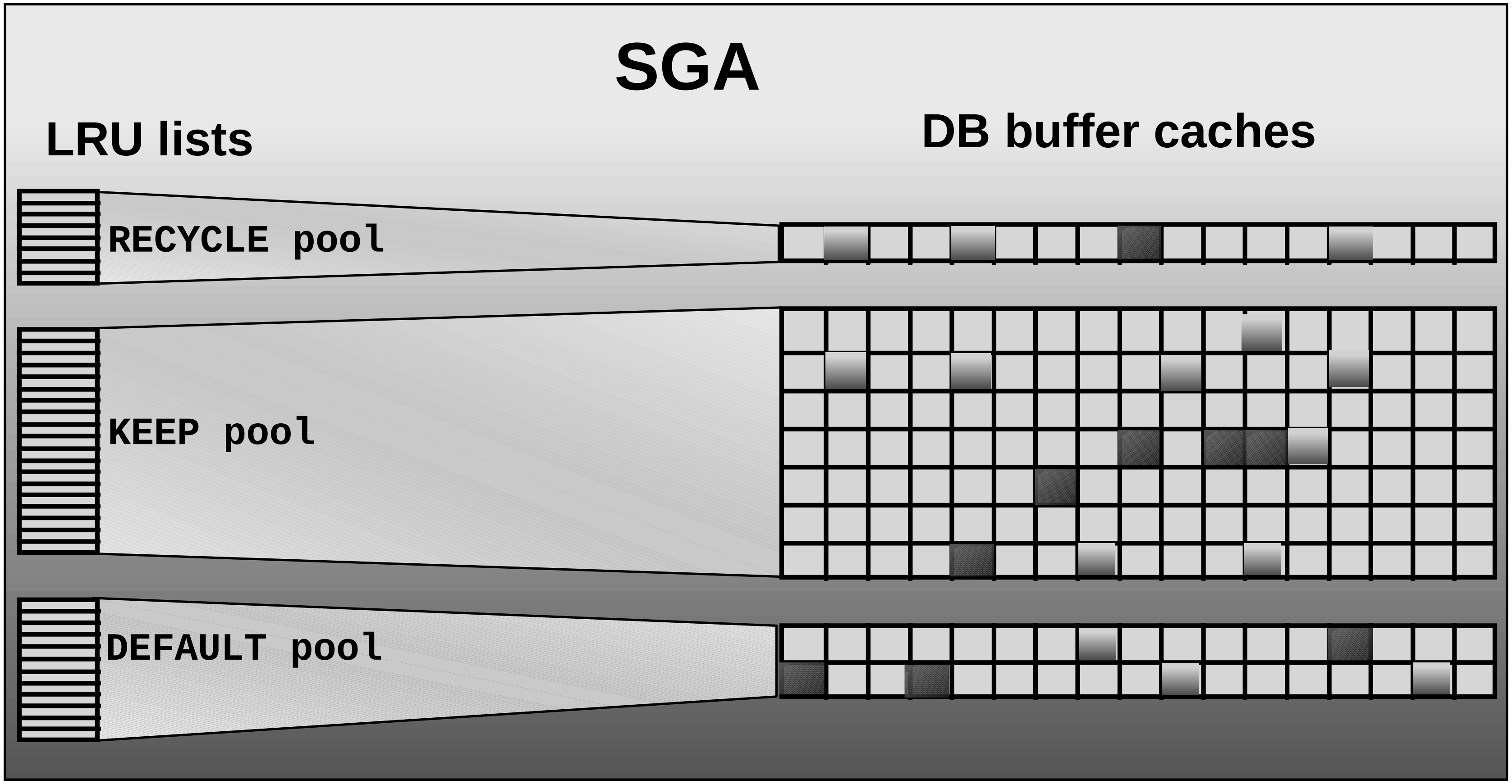
Hit ratio peaks because of data access methods:

- 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

Using Multiple Buffer Pools



ORACLE®

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)
...
```

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

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);
```

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
```

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
...
```

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
```

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$buffer_pool_statistics
 3   WHERE  db_block_gets + consistent_gets > 0;

NAME                HIT_RATIO
------------------  ----------
KEEP                .983520845
RECYCLE             .503866235
DEFAULT             .790350047
```

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**

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
```

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
```

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.

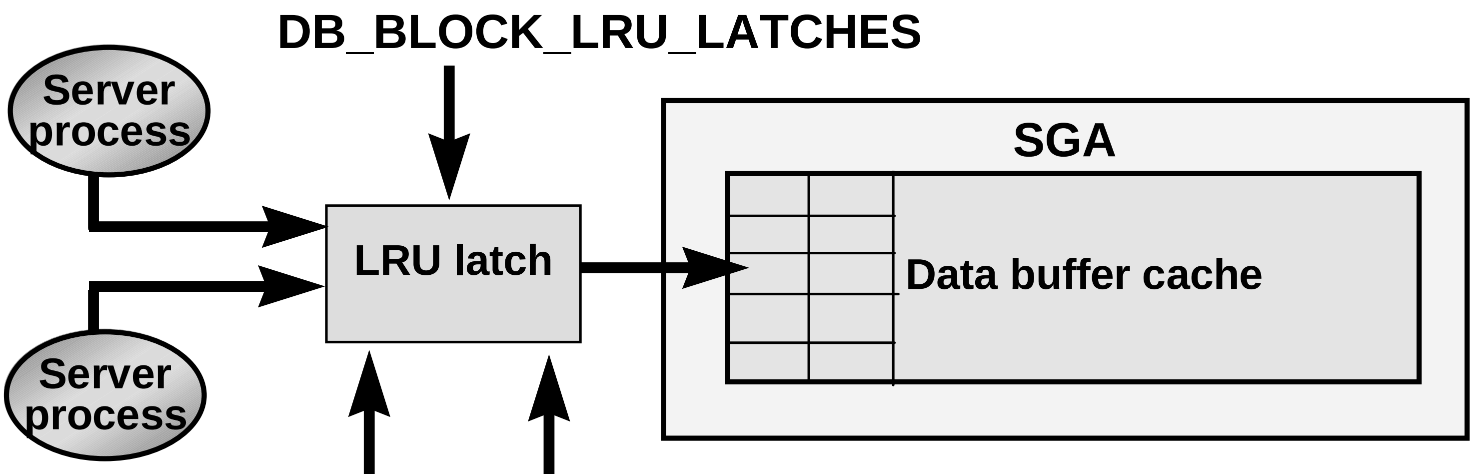
LRU Latches

- 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.

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.

Diagnosing LRU Latch Contention



V$LATCH
sleeps
gets
name = “cache buffers lru chain”

V$LATCHNAME
V$SYSTEM_EVENT
V$SESSION_EVENT
V$BUFFER_POOL_STATISTICS

Resolving LRU Latch Contention

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

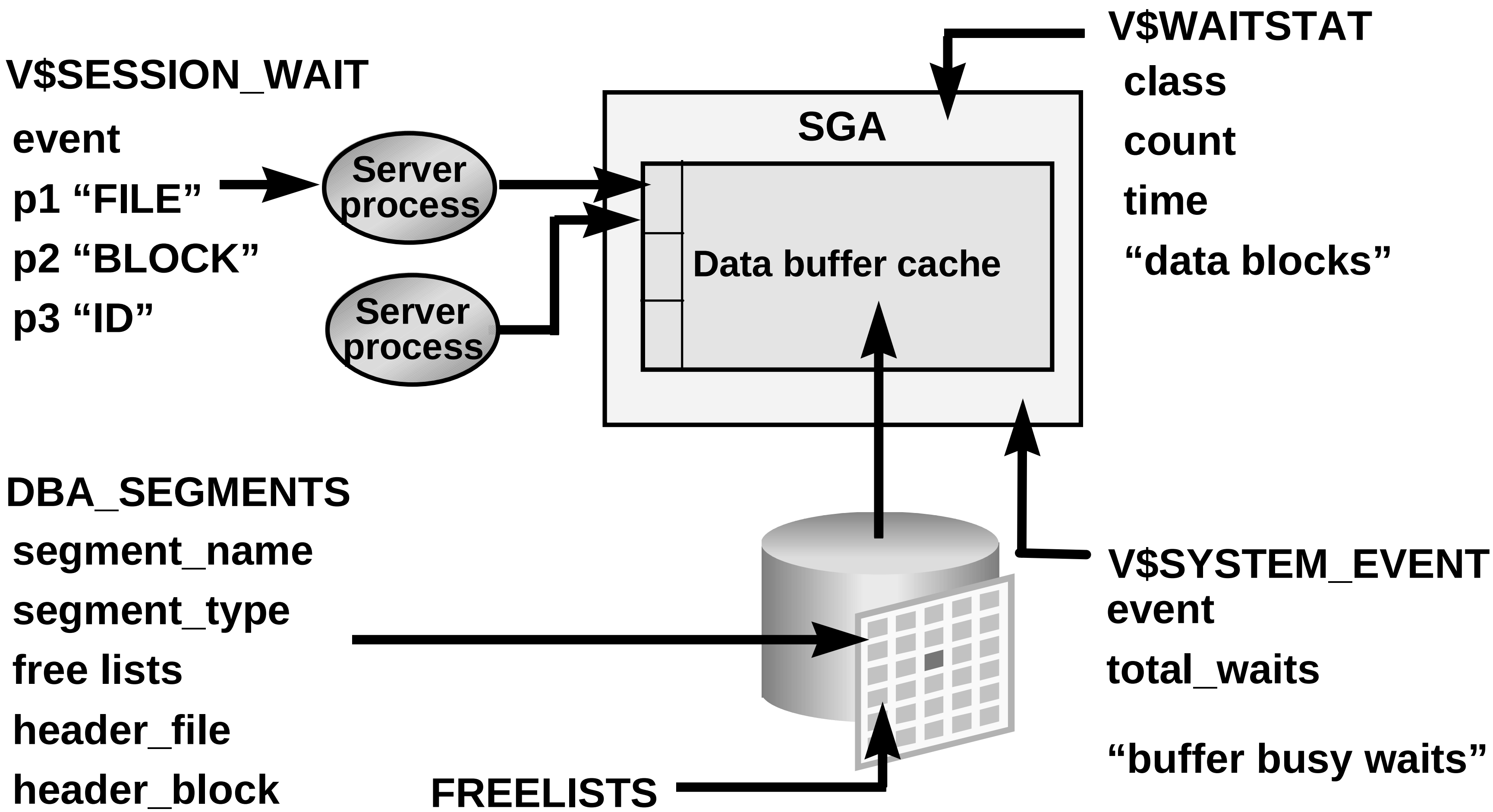
- **Increase the number of LRU latches by setting the parameter DB_BLOCK_LRU_LATCHES**
- **The maximum number of latches is the lower of:**
  - **Number of CPUs x 2 x 3**
  - **Number of buffers/50**

ORACLE®

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.

Diagnosing 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.

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

7

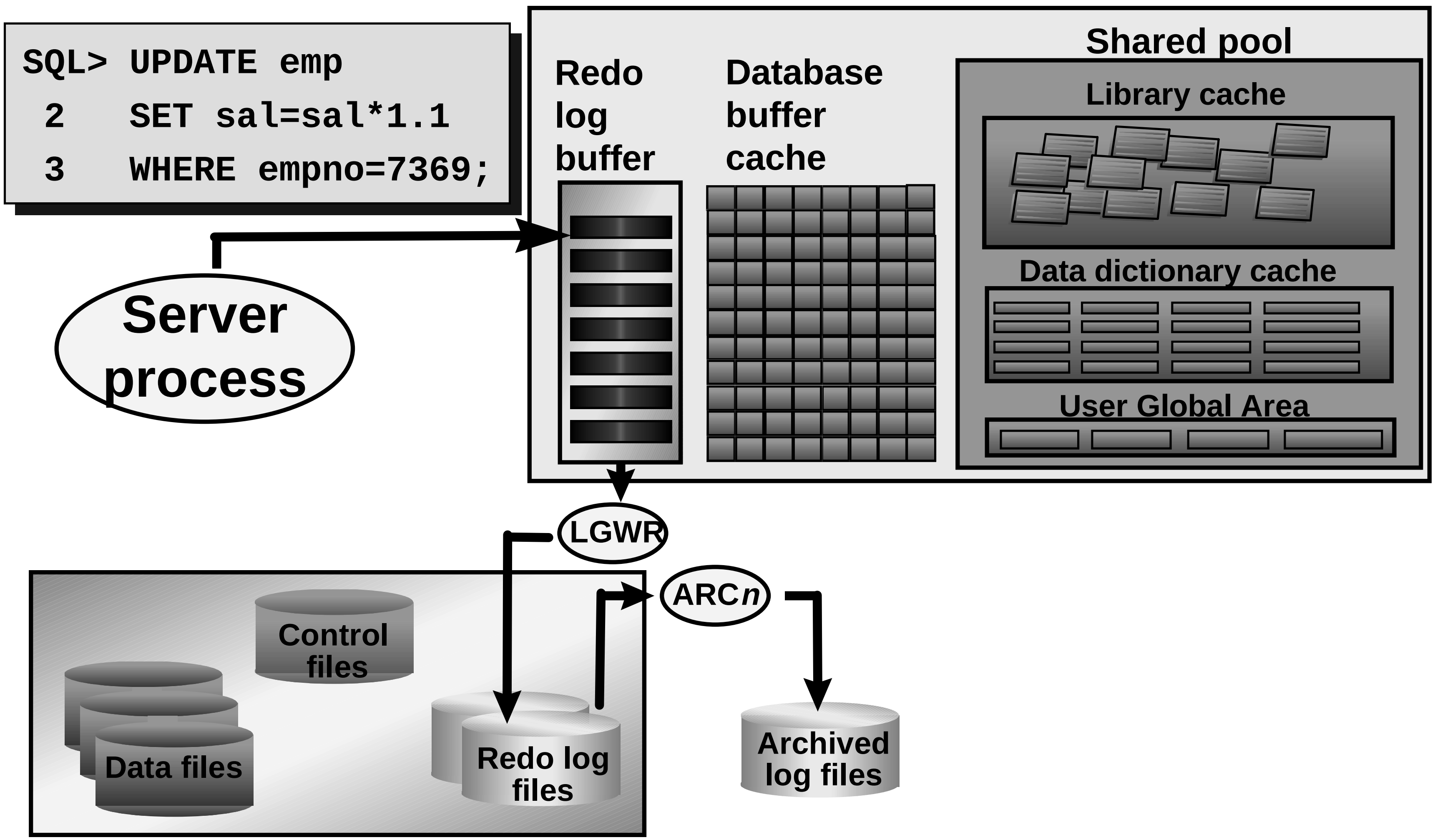
Tuning the Redo Log Buffer

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

The Redo Log Buffer

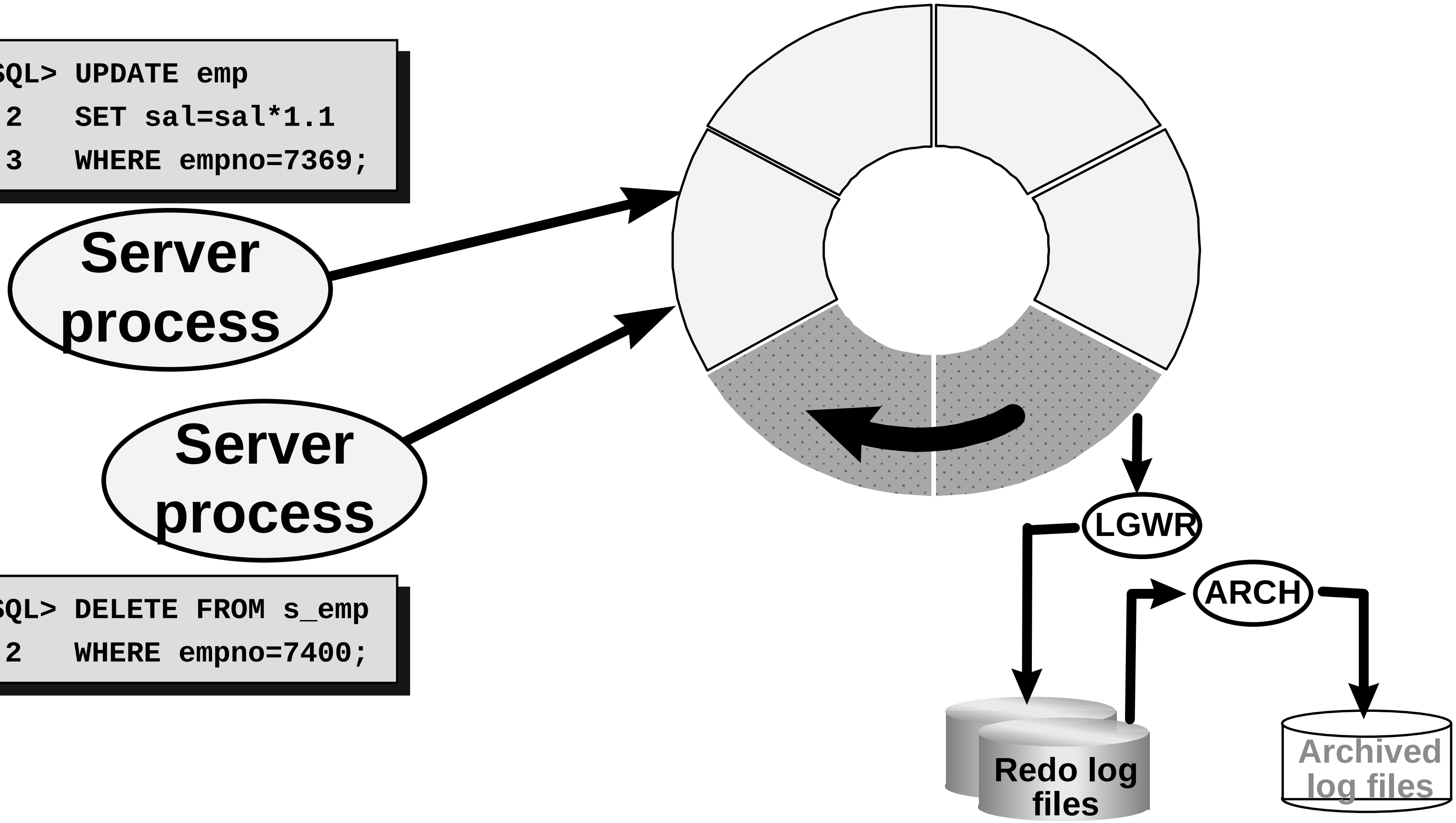


Sizing the Redo Log Buffer

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

Tuning the Redo Log Buffer

```
SQL> UPDATE emp
 2   SET sal=sal*1.1
 3   WHERE empno=7369;
```

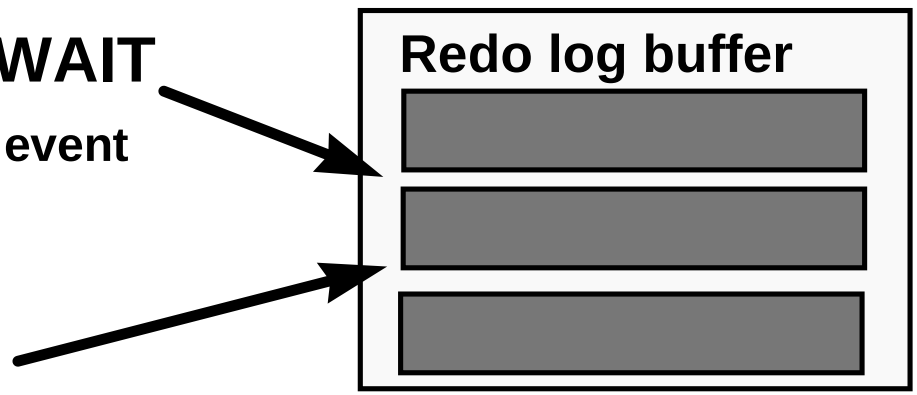



```
SQL> DELETE FROM s_emp
 2   WHERE empno=7400;
```

Diagnostic Tools for Tuning the Redo Log Buffer

V$SESSION_WAIT

Log Buffer Space event



V$SYSSTAT

Redo Buffer Allocation Retries statistic

LOG_BUFFER
LOG_CHECKPOINT_INTERVAL
LOG_CHECKPOINT_TIMEOUT

Guidelines

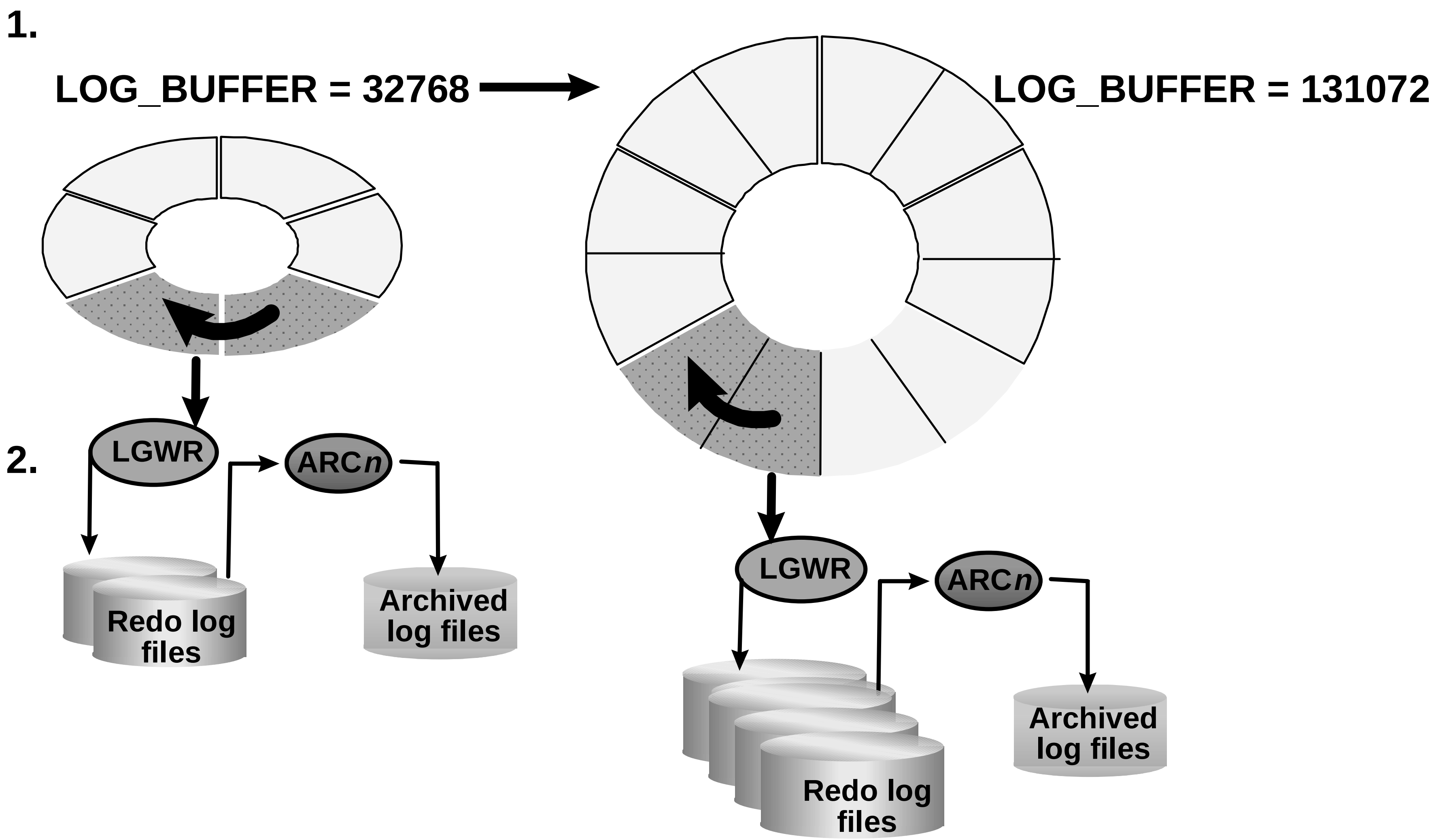
There should be no Log Buffer Space waits.

```
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’);
```

Guidelines



ORACLE®

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**

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

ORACLE®

8

Database Configuration and I/O Issues

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 DBW*n* process I/O

I/O Statistic for Different Oracle File Types

Process	Oracle File I/O			
	Data Files	Log	Archive	Control
CKPT	Write			Write
DBW*n*	Write			
LGWR		Write		
ARC*n*		Read	Write	Read/write
SERVER	Read			

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

Distributing Files Across Devices

- Separate data files and redo log files
- Stripe table data
- Reduce disk I/O
- Evaluate the use of raw devices

Oracle File Striping

Operating system striping:

- **Use operating system striping software or RAID**
- **Decide on the right stripe size**

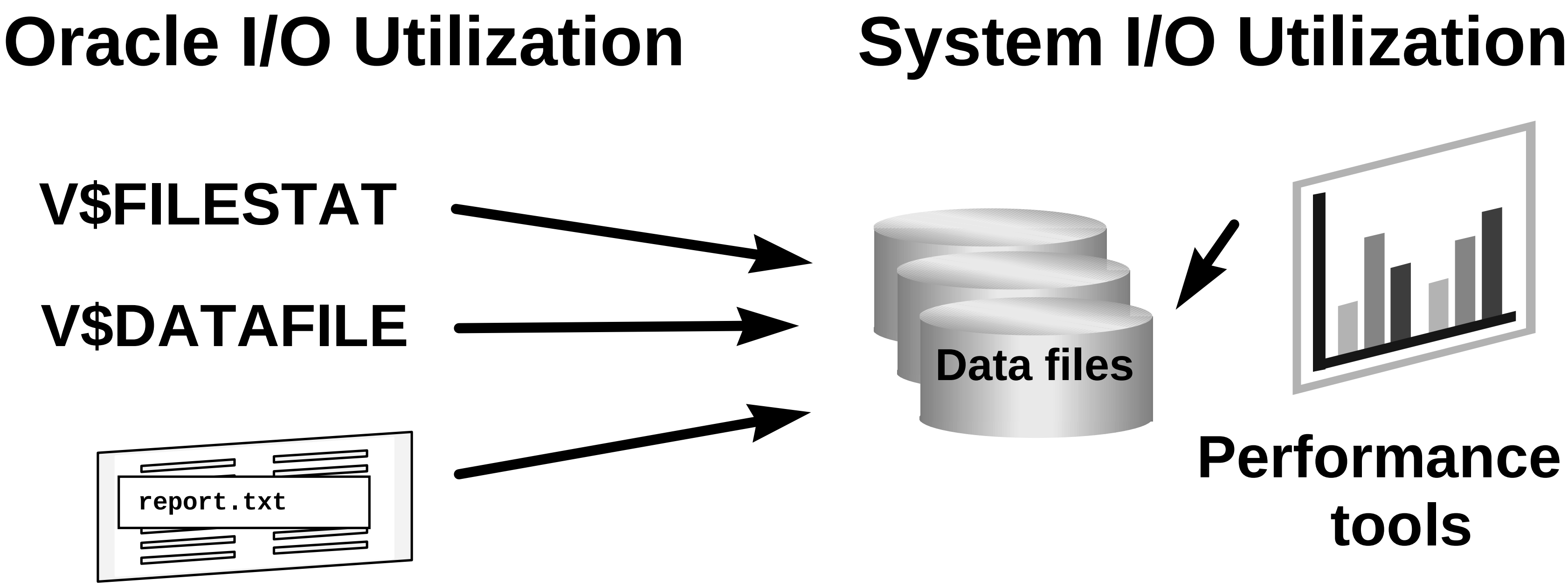
Manual striping:

- **Use the CREATE TABLE or ALTER TABLE ALLOCATE command**
- **Is worthwhile with parallel query usage**

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**

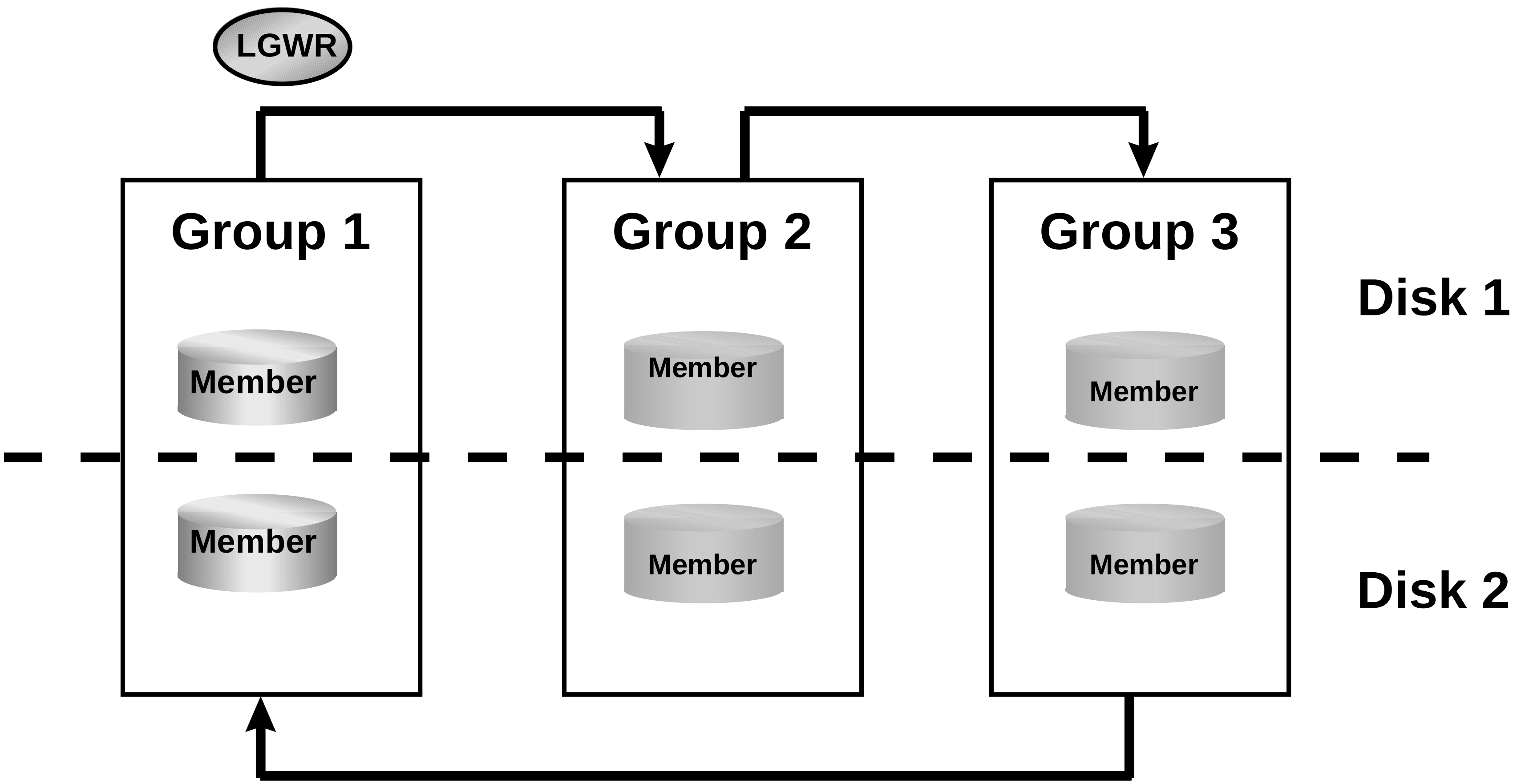
Diagnostic Tools for Checking I/O Statistics



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_     WRITES BLKS_    WRITE_
                                              READ      TIME              WRT      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.
```

Redo Log Groups and Members



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**

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)**

Diagnostic Tools

V$ARCHIVE_DEST

V$ARCHIVED_LOG

V$ARCHIVE_PROCESSES

LOG_ARCHIVE_DEST_STATE_*n*

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

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

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_i*Nnn_SID***

Initialization Parameters

Deploy I/O slaves for DBW*n*, LGWR, ARC*n*, 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`**

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**

Tuning DBW*n* I/O

- Influence the DBW*n* 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, DBW*n* is not agressive for writing checkpoints buffers.
- If the number is between the low and high computed limits, DBW*n* writes from the checkpoint queue until the number of checkpoint buffers drops under low.
- If the number is greater than the high limit, DBW*n* writes out checkpoint buffers.
- The default value is (2*32) – 1.

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 DBW*n* I/Os**

9

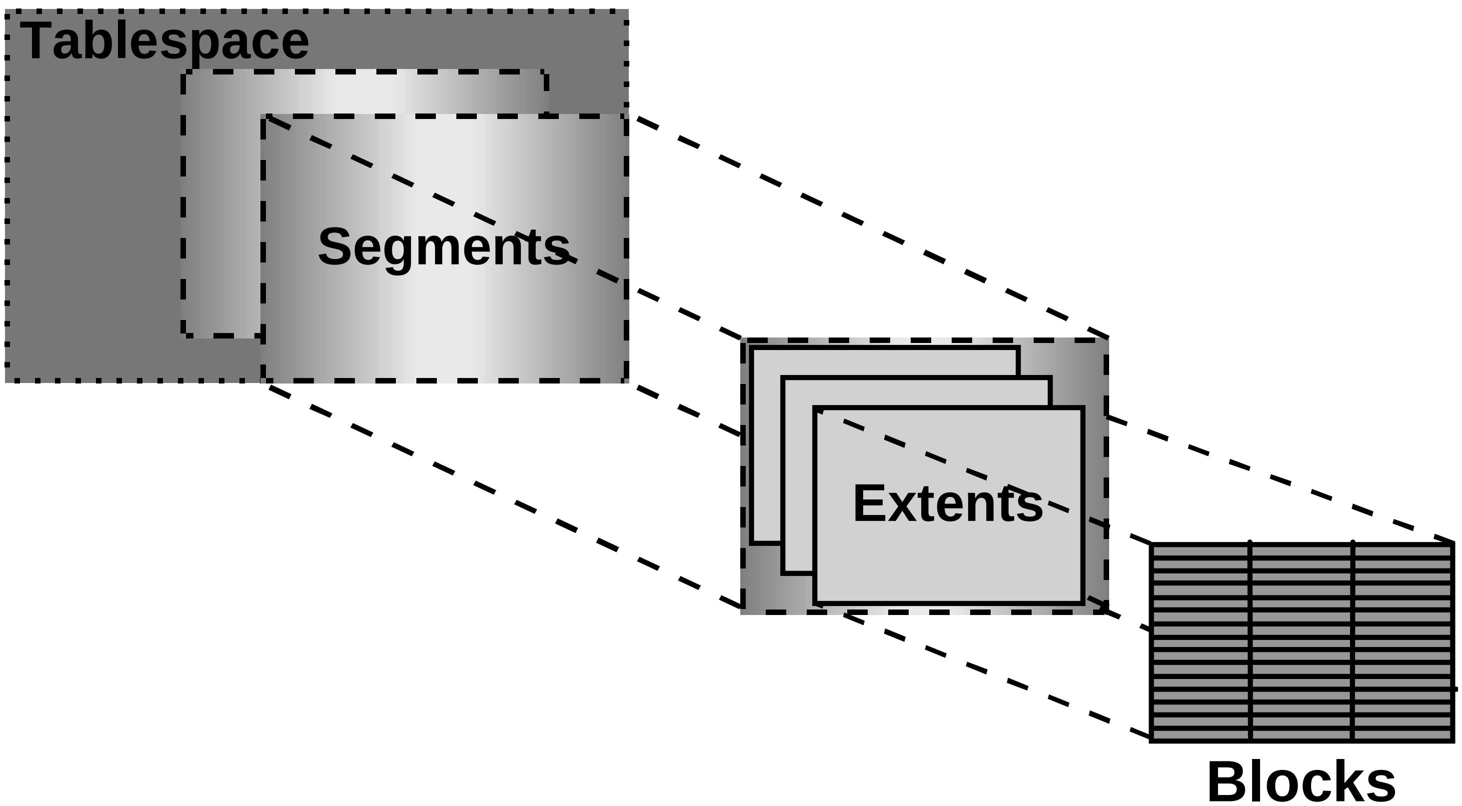
Using Oracle Blocks Efficiently

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**

Database Storage Hierarchy

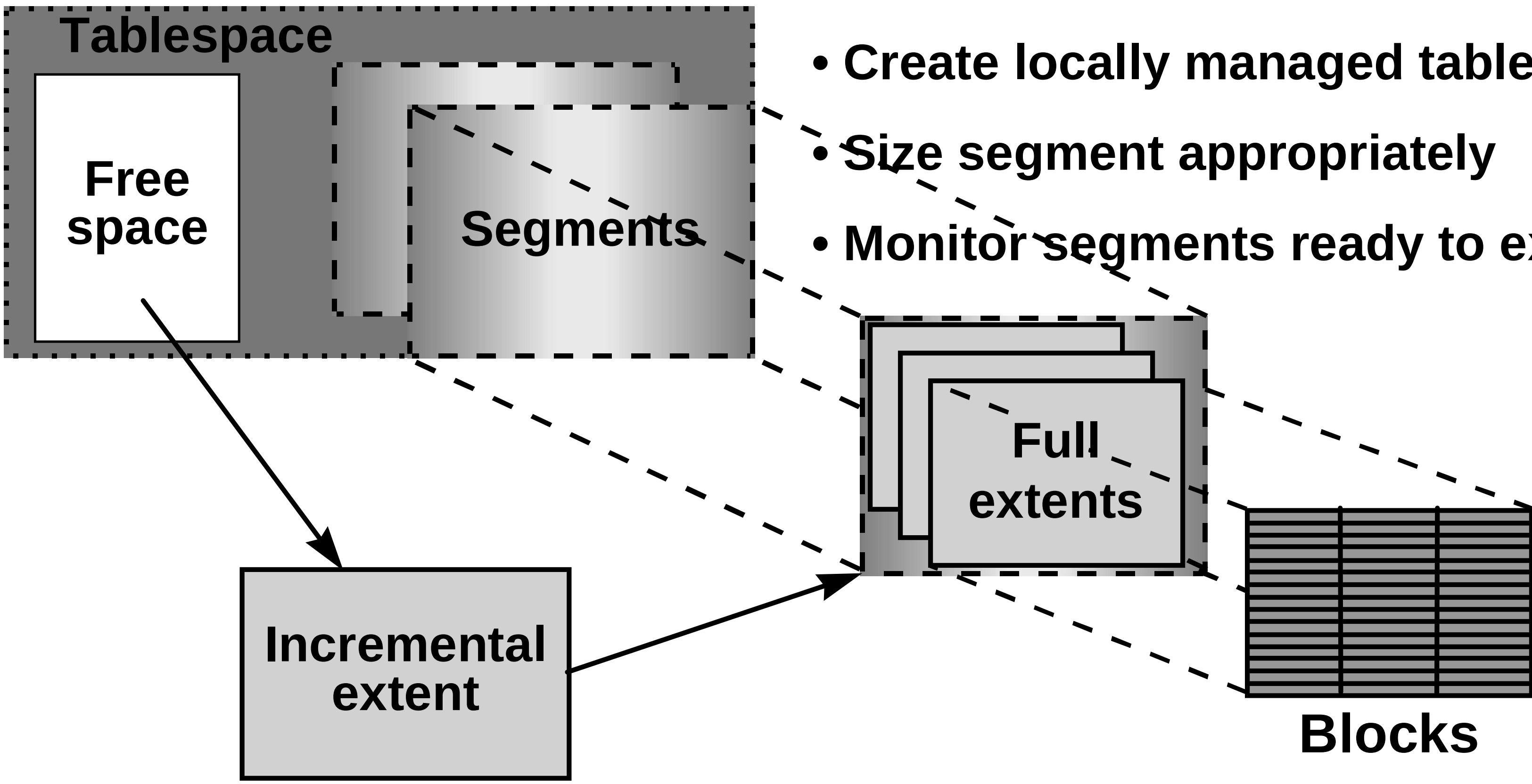


ORACLE®

Allocating an Extent

Avoid dynamic extent allocation disadvantages:

- Create locally managed tablespaces
- Size segment appropriately
- Monitor segments ready to extend



ORACLE®

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.
```

Avoiding Dynamic Allocation Disadvantages

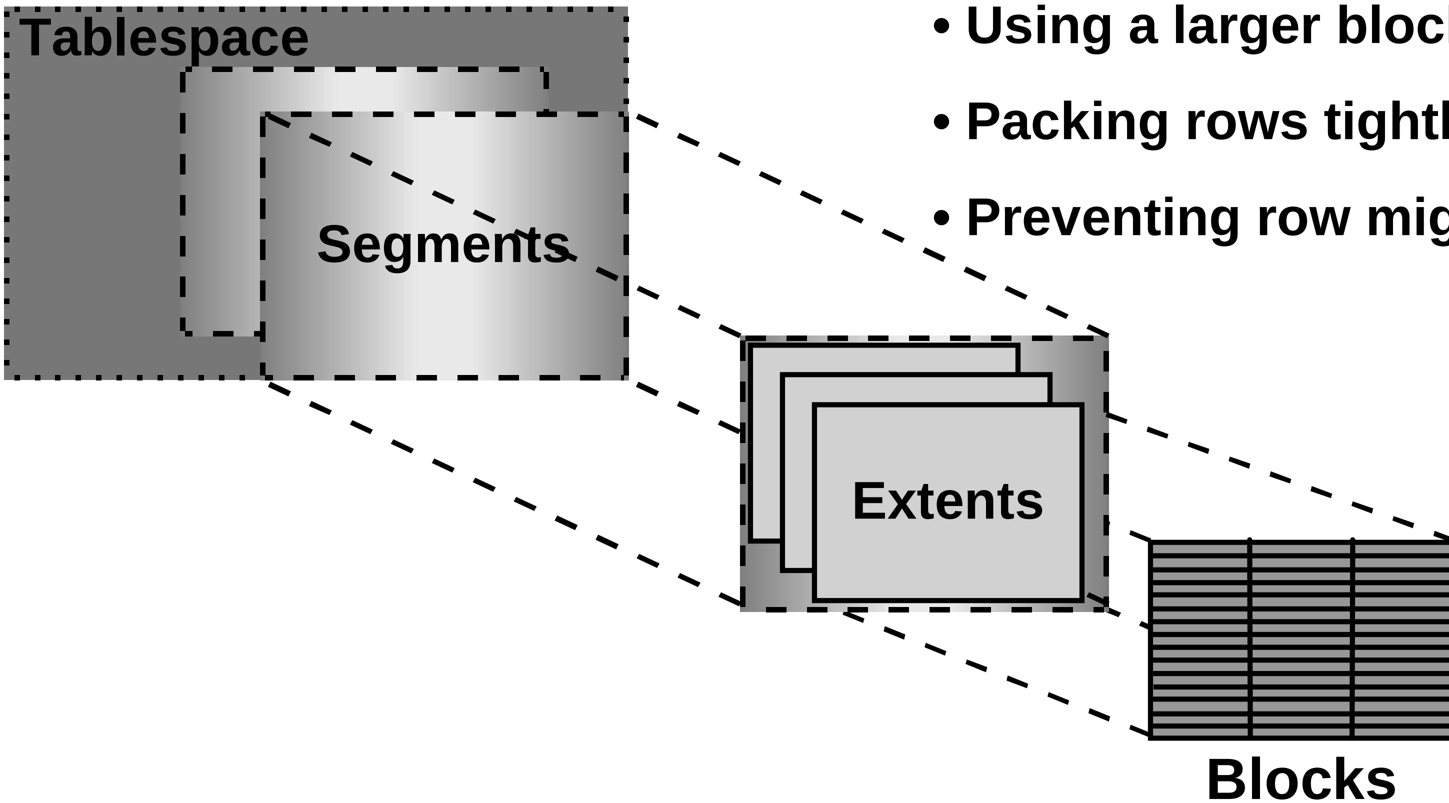
Create a locally managed tablespace:

```
CREATE TABLESPACE user_data_1
DATAFILE ‘oracle8/oradata/db1/lm_1.dbf’
SIZE 100M
EXTENT MANAGEMENT LOCAL
UNIFORM SIZE 2M;
```

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**

Database Block Size



Minimize block visits by:

- **Using a larger block size**
- **Packing rows tightly**
- **Preventing row migration**

ORACLE®

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

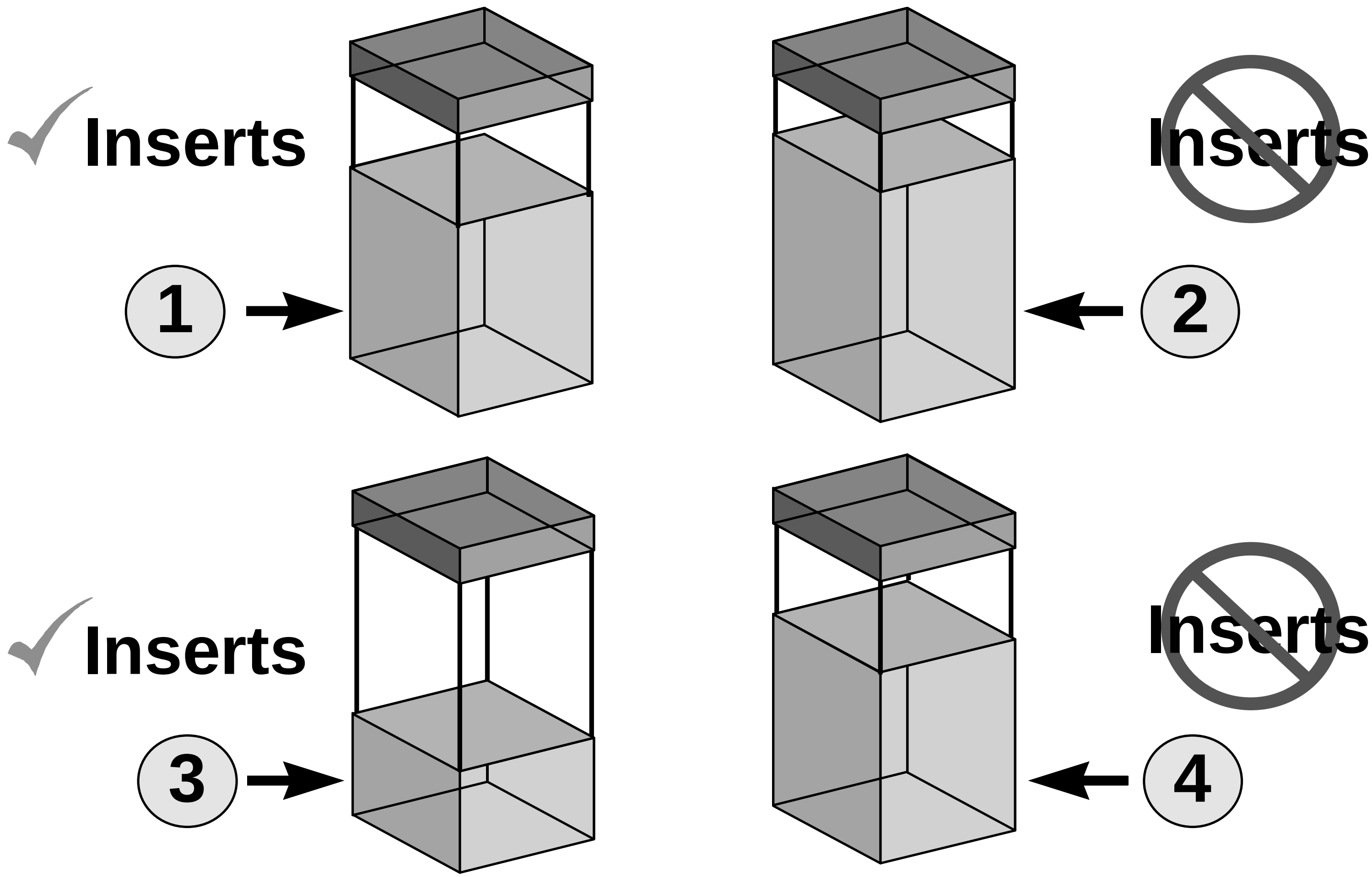
Small Block Size Pros and Cons

- **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**

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

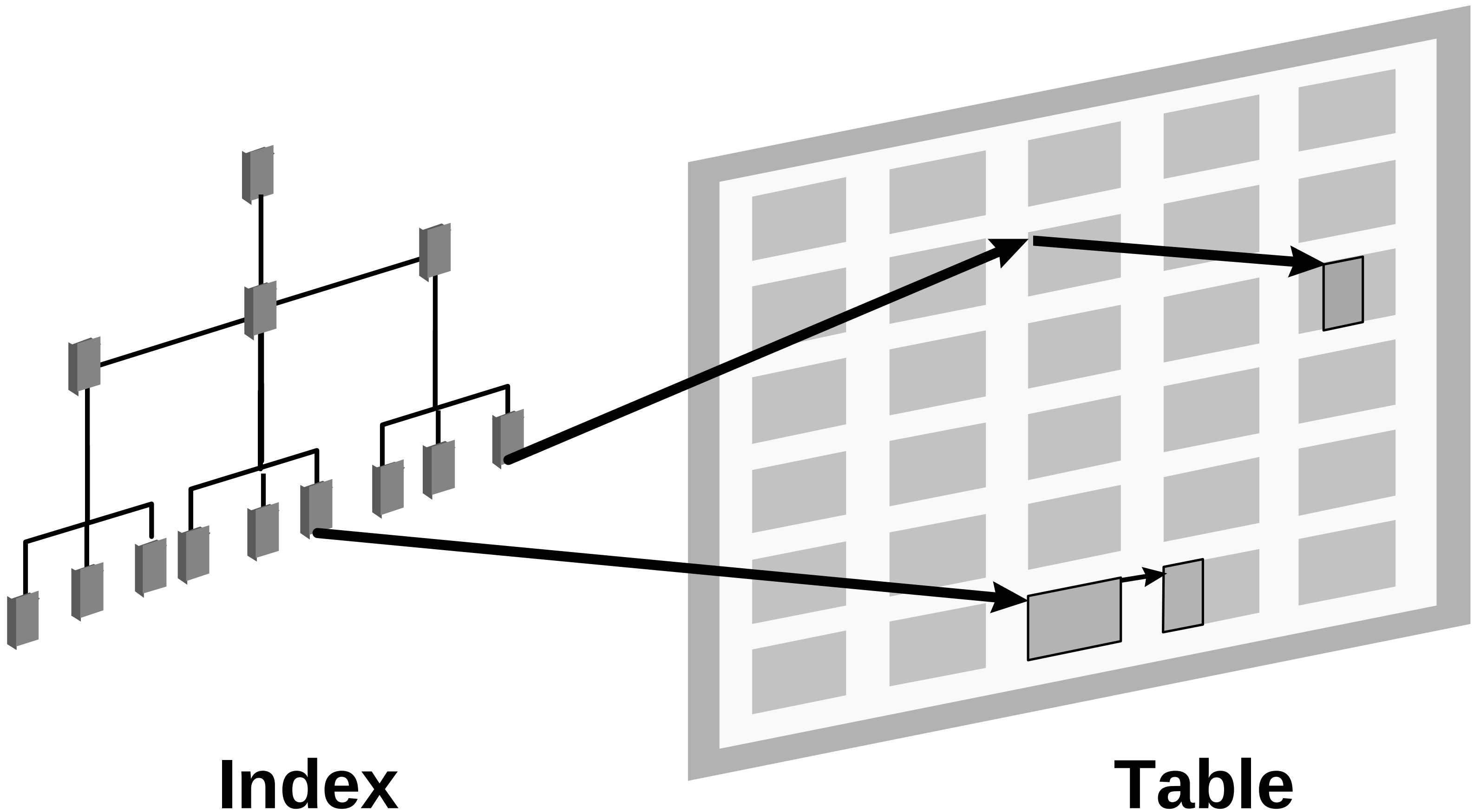
PCTFREE and PCTUSED



Guidelines

- **PCTFREE**
  - **Default 10**
  - **Zero if no UPDATE activity**
  - **PCTFREE = 100 × upd / (upd + ins)**
- **PCTUSED**
  - **Default 40**
  - **Set if rows deleted**
  - **PCTUSED = 100 – PCTFREE – 100 × rows × (ins + upd) / blocksize**

Migration and Chaining



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 …
```

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  AAAAluAAHAAAAA1AAA
SALES       ORDER_HIST  AAAAluAAHAAAAA1AAB


...
```

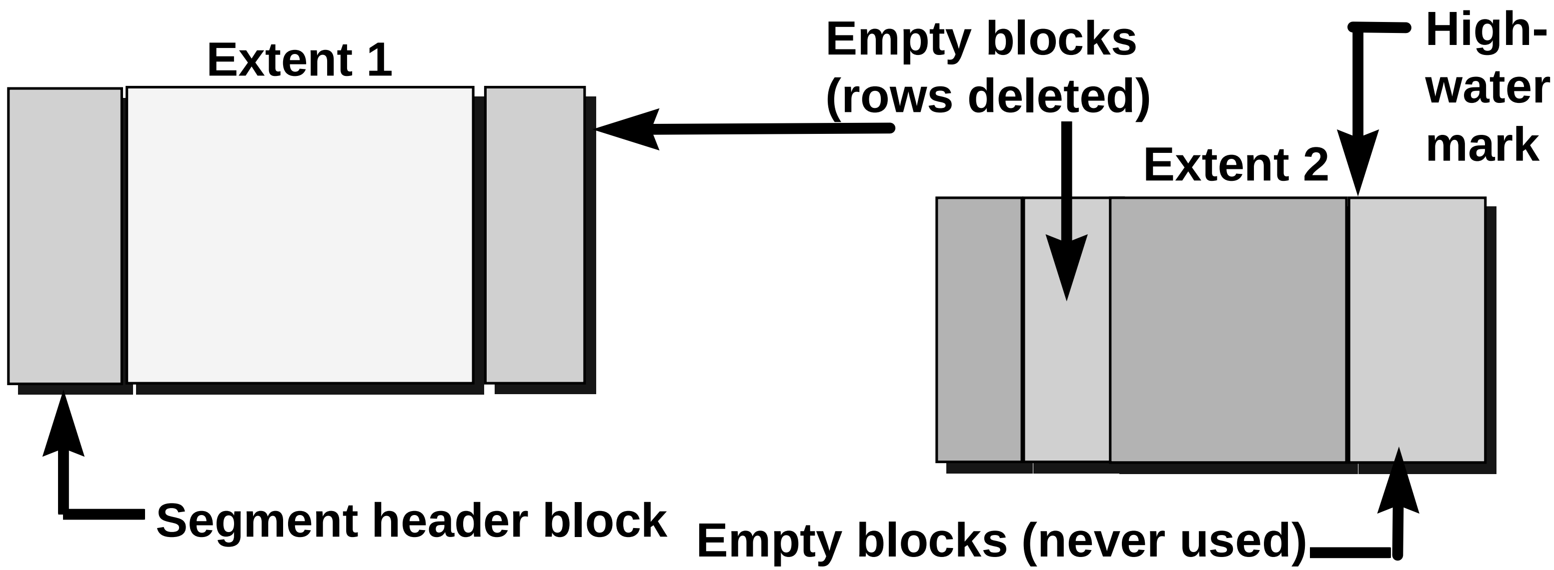
Eliminating Migrated Rows

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.

ORACLE®

The High-Water Mark



- Recorded in segment header block
- Set to the beginning of the segment on creation
- Incremented in five-block increments as rows are inserted
- Reset by the TRUNCATE command and not DELETE

Table Statistics

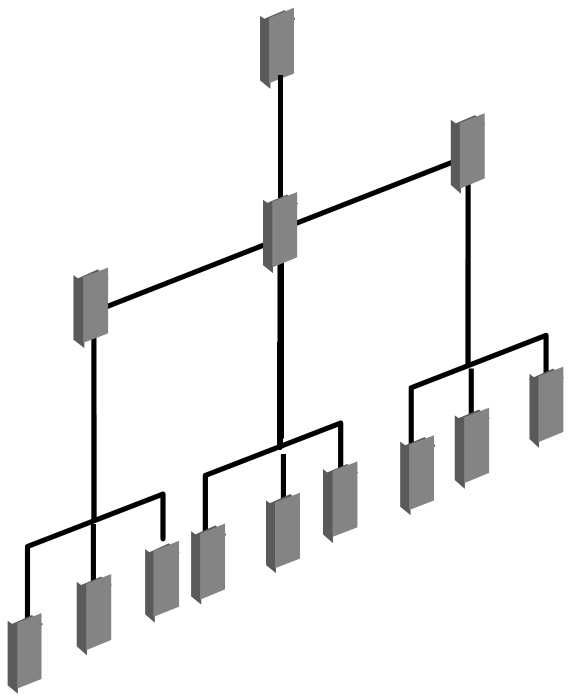
Query table statistics from the ANALYZE command:

```
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
```

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;
/
```

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.

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.
```

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

10

Optimize Sort Operations

Objectives

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

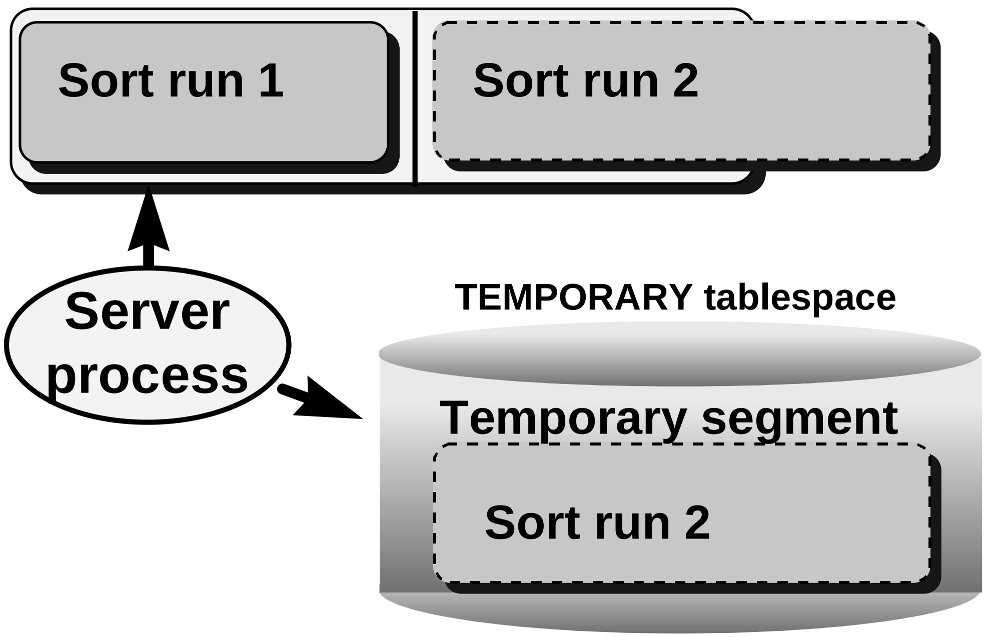
- **Identify the SQL operations that require sorting**
- **Ensure that sorting is done in memory where possible**
- **Reduce the number of I/Os required for the sort runs**
- **Allocate temporary space appropriately**

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**

Sort Process

Sort space requirement is greater than SORT_AREA_SIZE:



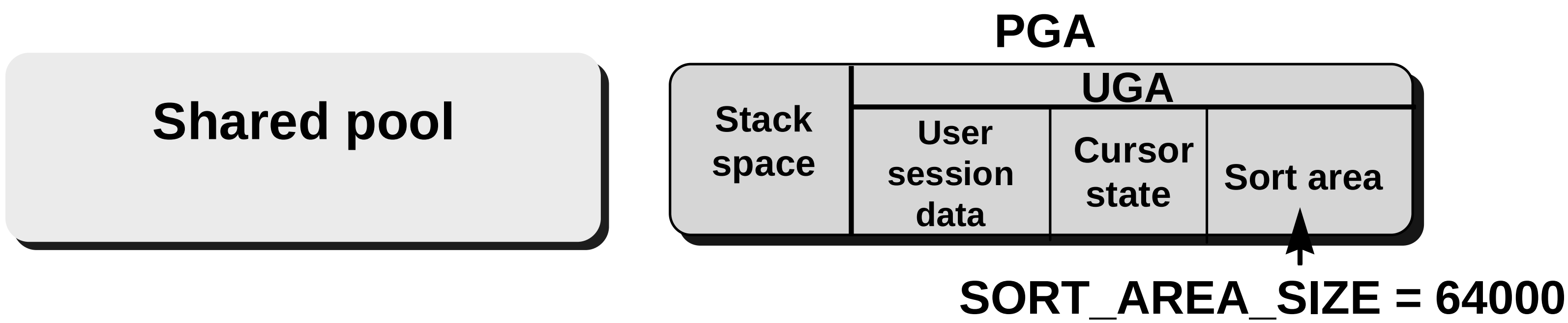
Segments hold data while the server works on another sort run.

SORT_MULTIBLOCK_READ_COUNT forces the sort to read a larger section of each run into memory during a merge pass.

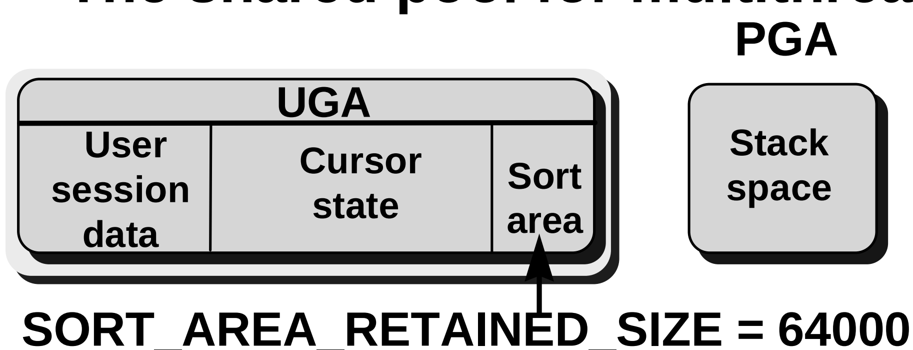
Sort Area and Parameters

The sort space is in:

- **The PGA for dedicated server connections**



- **The shared pool for multithread server connections**



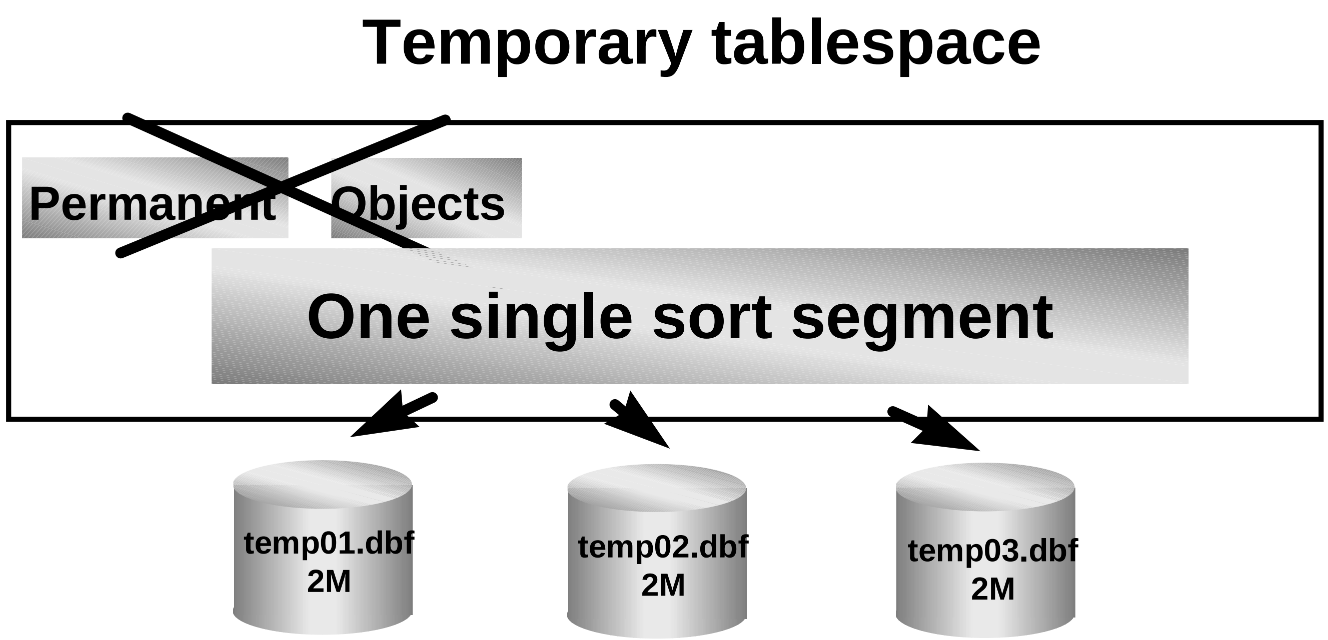
Sort Area and Parameters

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.**

Sort Process and Temporary Space



A temporary tablespace is created with the command CREATE TABLESPACE ... DATAFILE ... TEMPORARY

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)

Tuning Sort Operations

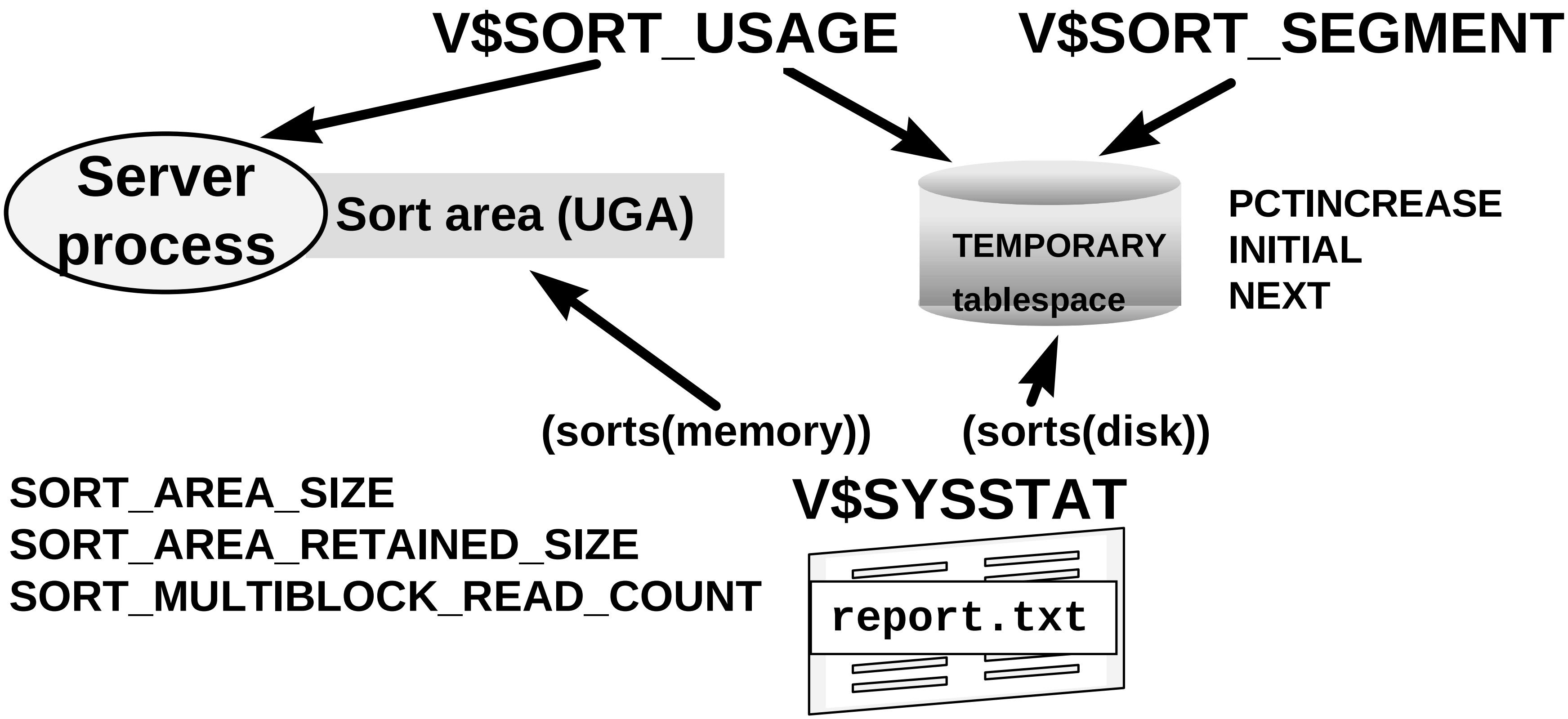
- **Avoid sort operations whenever possible**
- **Reduce swapping and paging by ensuring that sorting is done in memory where possible**
- **Reduce space allocation calls: allocate temporary space appropriately**

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**

Diagnostic Tools



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%.

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.

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.

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

11

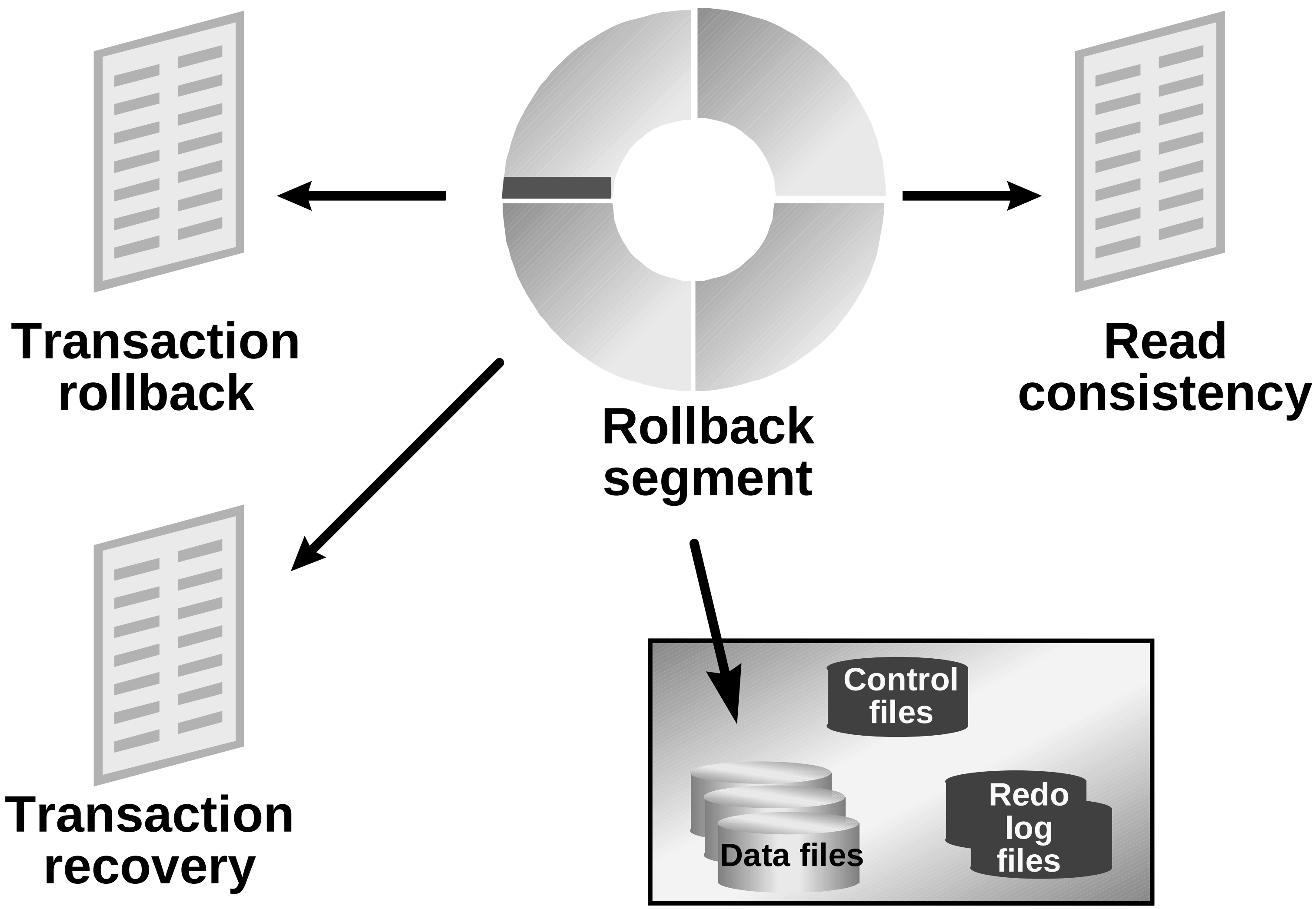
Rollback Segment Tuning

Objectives

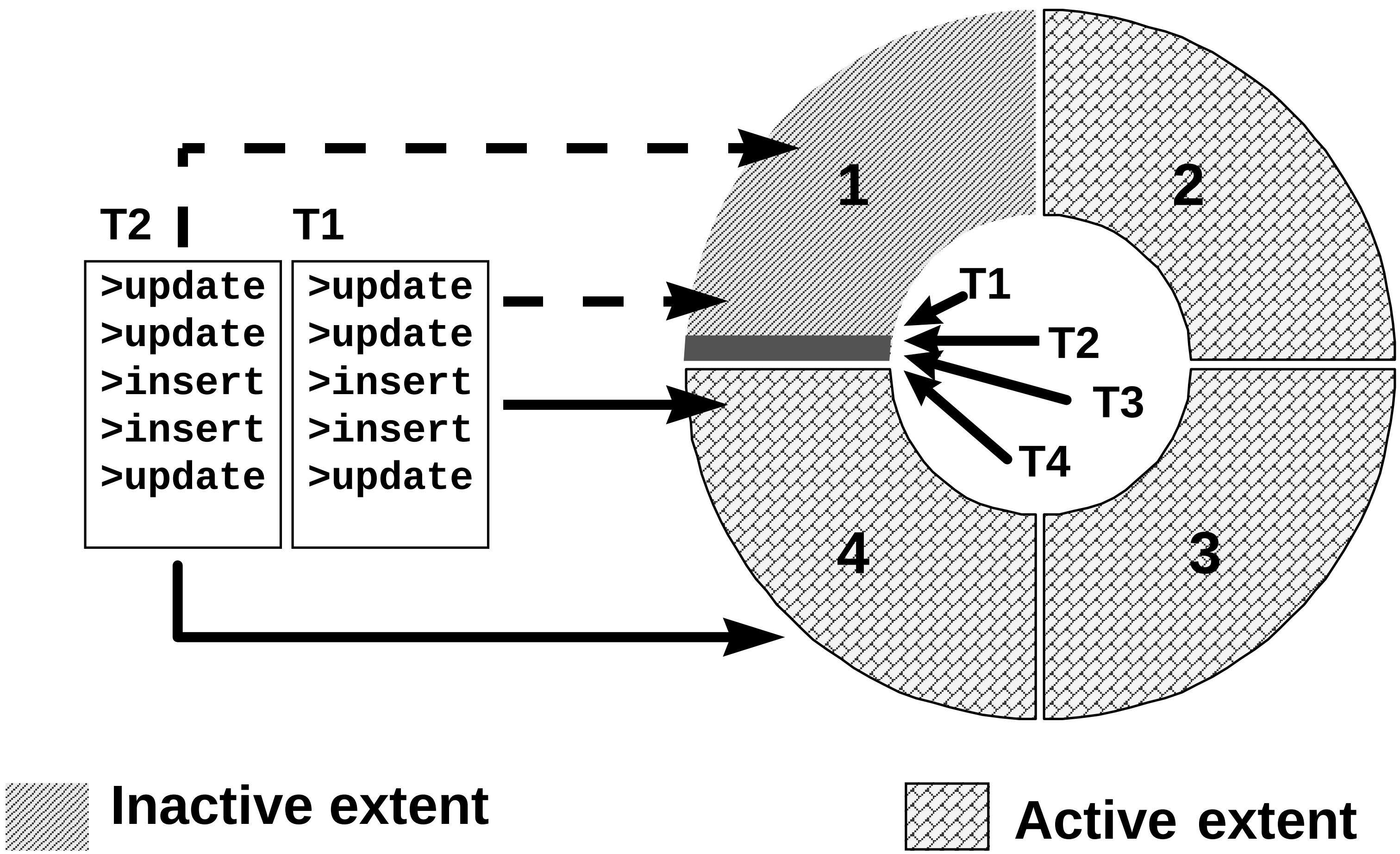
After completing this lesson, you should be able to do the following:

- **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**

Rollback Segments: Usage



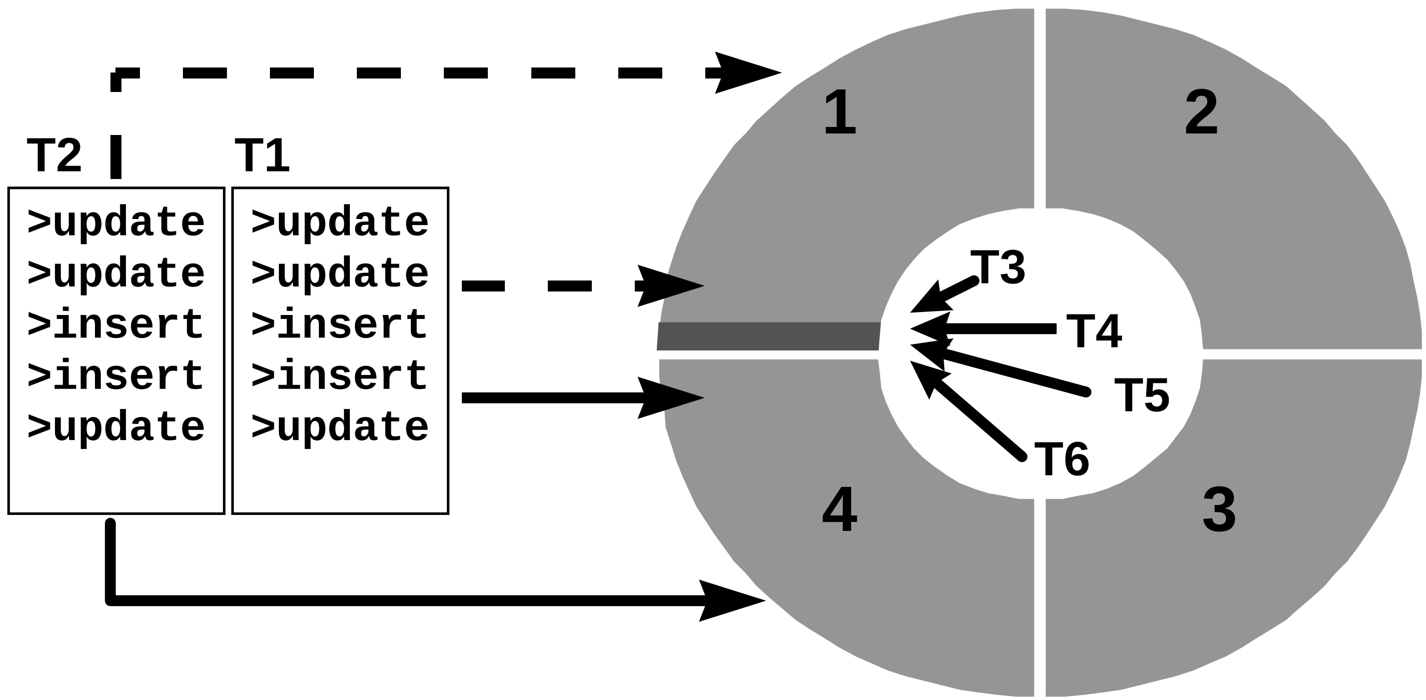
Rollback Segment Activity



ORACLE®

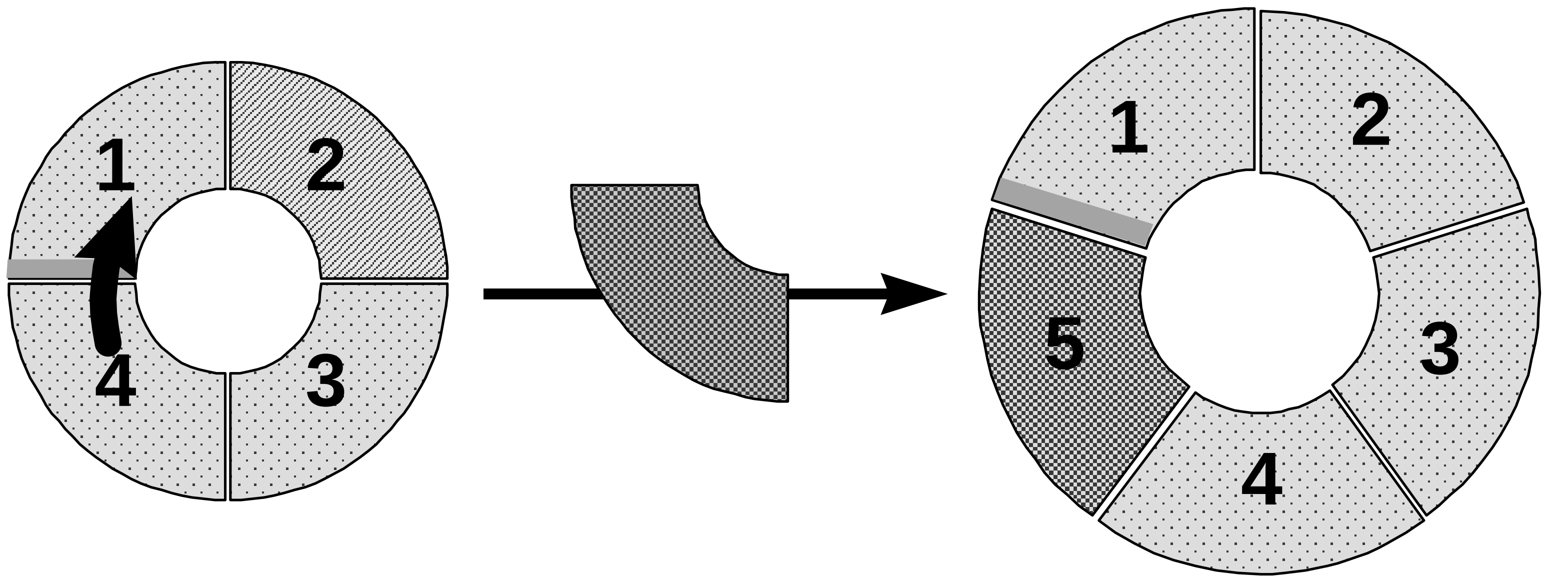
Rollback Segment Header Activity

- Rollback segment headers contain entries for their respective transactions.
- Every transaction must have update access.



ORACLE®

Growth of Rollback Segments



Active extent

Inactive extent

New extent

ORACLE®

Transaction Types

Read-only

```
>SET TRANSACTION
 READ ONLY;

>SELECT ...
>SELECT ...
>UPDATE ...
ORA-01456: may not
perform I/D/U
operation inside a
READ ONLY transaction
```

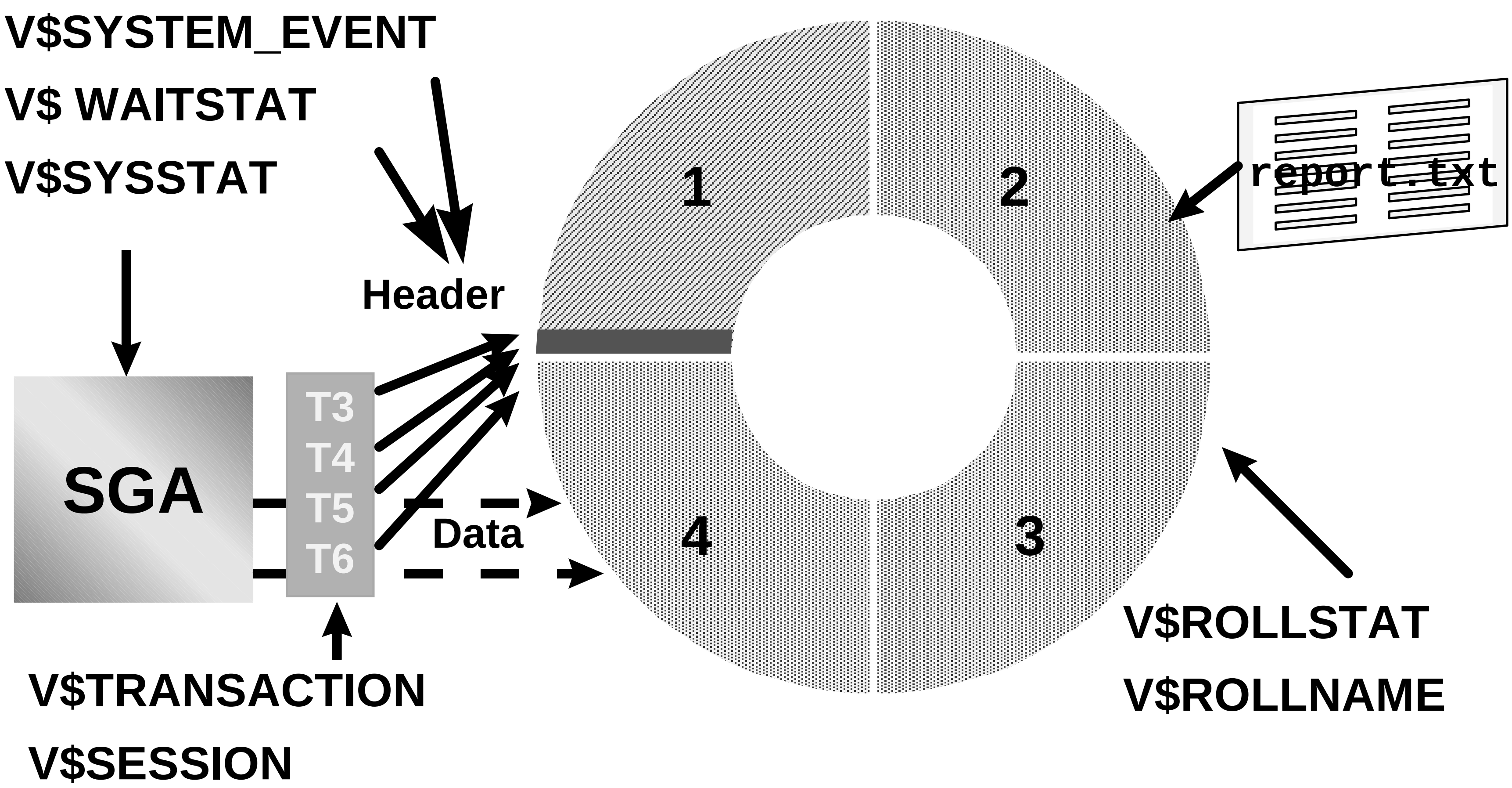
Serializable

```
>SET TRANSACTION
 ISOLATION LEVEL
 SERIALIZABLE;
>SELECT ...
>SELECT ...
>UPDATE ...
>UPDATE ...
ORA-08177: can't
serialize access for
this transaction
```

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.

Diagnostic Tools for Tuning Rollback Segments



ORACLE®

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.

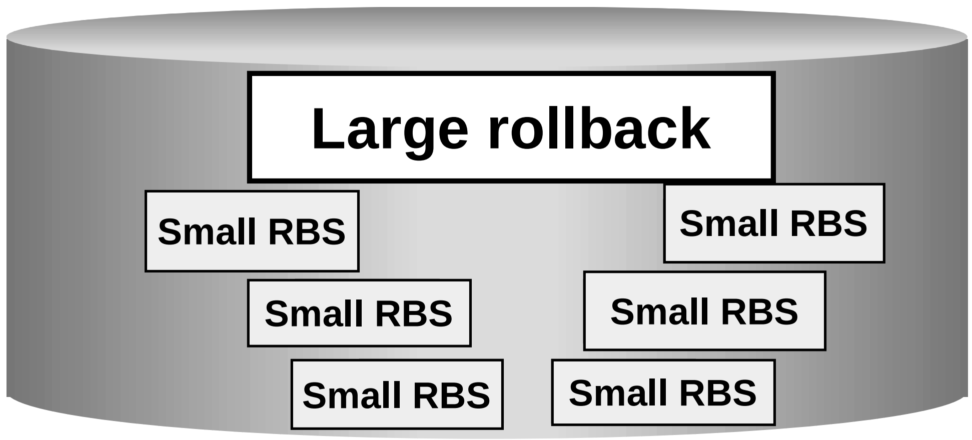
Diagnosing Rollback Segment Contention

```
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.

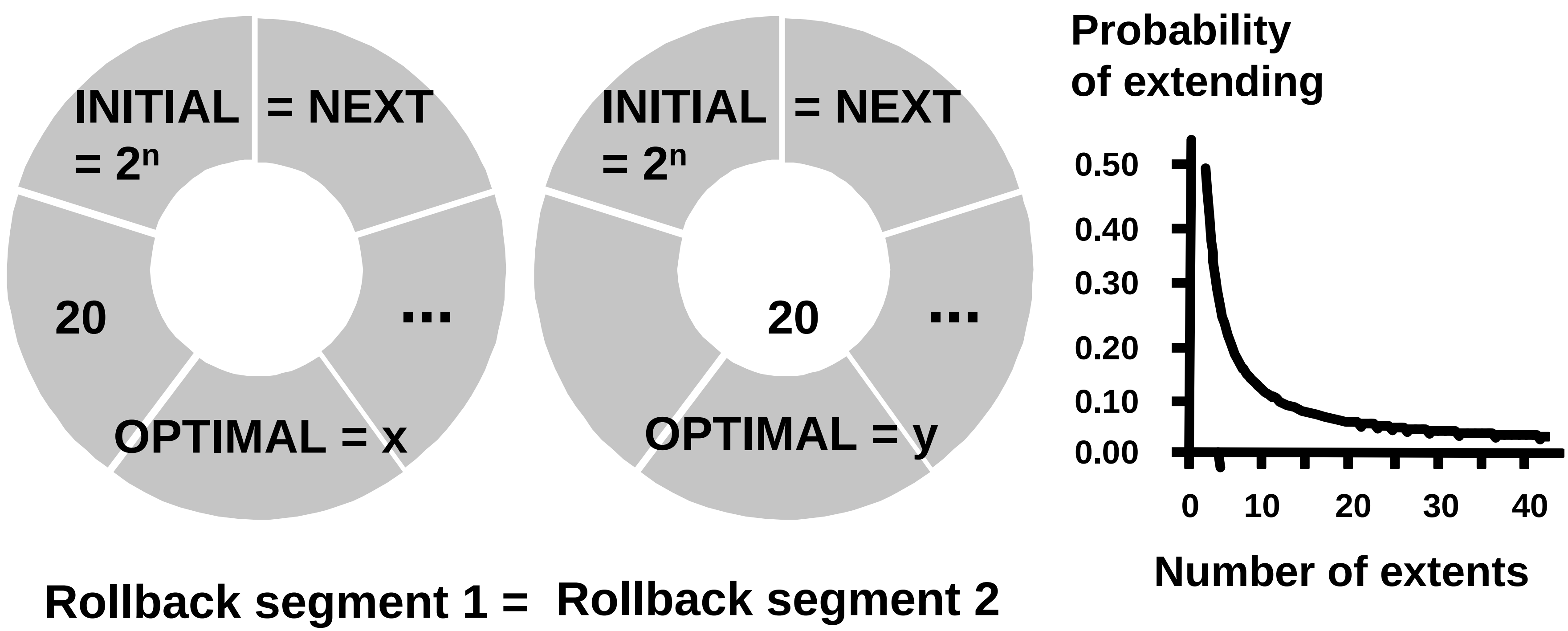
Guidelines: How Many Rollback Segments?



- OLTP: One RBS for four transactions
- Batch: One rollback segment for each concurrent job

```
SQL> SET TRANSACTION USE ROLLBACK SEGMENT large_rbs;
```

Guidelines: Sizing Rollback Segments



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
```

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
```

Using Less Rollback

- The design of the application should allow users to commit regularly.
- Developers should not code 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

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.

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

12

Monitoring and Detecting Lock Contention

Objectives

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

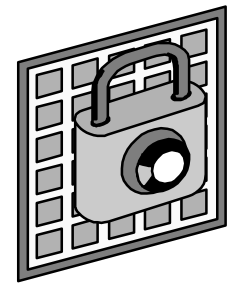
- **Define types and modes of locking**
- **List possible causes of contention**
- **Use Oracle utilities to detect lock contention**
- **Resolve contention in an emergency**
- **Prevent locking problems**
- **Recognize Oracle errors arising from deadlocks**

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**

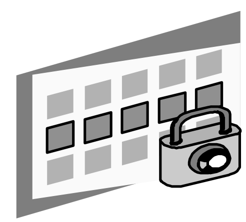
Two Types of Locks

- **DML or data locks:**
  - **Table-level locks**
  - **Row-level locks**



(TM)

- **DDL or dictionary locks**



(TX)

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

Table Lock Modes

Automatically acquired:

- **Row Exclusive (RX): INSERT, UPDATE, DELETE**
- **Row Share (RS): SELECT... FOR UPDATE**

Table Lock Modes

Manually acquired in LOCK statement:

```
SQL> LOCK TABLE table_name IN mode_name MODE;
```

Share (S)

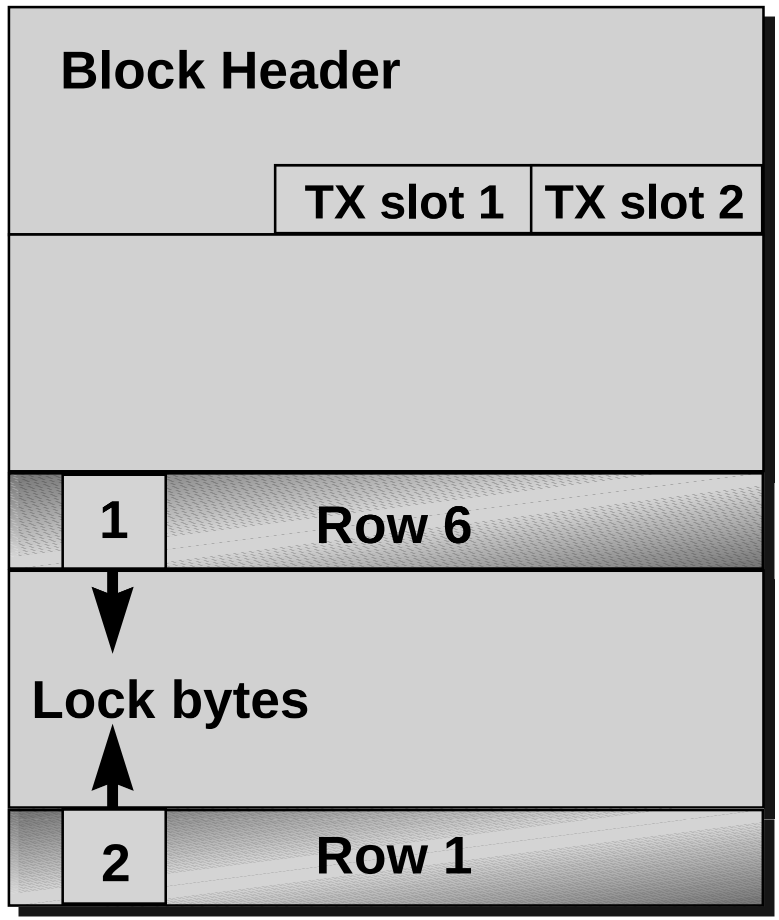
- **No DML allowed**
- **Implicitly used for referential integrity**

Table Lock Modes

Maunally acquired in LOCK statement:

- **Share Row Exclusive (SRX)**
  - **No DML or Share mode allowed**
  - **Implicitly used for referential integrity**
- **Exclusive (X)**

DML Locks in Block



DDL Locks

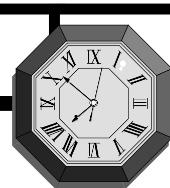
- Exclusive DDL locks:
  - DROP TABLE statement
  - ALTER TABLE statement
- Shared DDL locks:
  - CREATE PROCEDURE statement
  - AUDIT statement
- Breakable parse locks: Invalidating shared SQL area

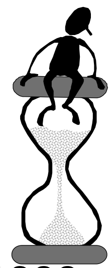
Possible Causes of Lock Contention

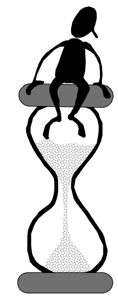
- **Unnecessarily high locking levels**
- **Uncommitted changes**
- **Other products imposing higher-level locks**

Diagnostic Tools for Monitoring Locking Activity

Transaction 1	Transaction 2	Transaction 3
> UPDATE emp SET sal = sal x 1.1;	> UPDATE emp SET sal = sal x 1.1 WHERE empno = 1000;	> UPDATE emp SET sal = sal x 1.1 WHERE empno = 2000;



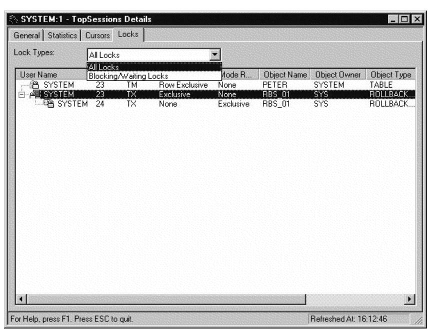




V$LOCK

V$LOCKED_OBJECT

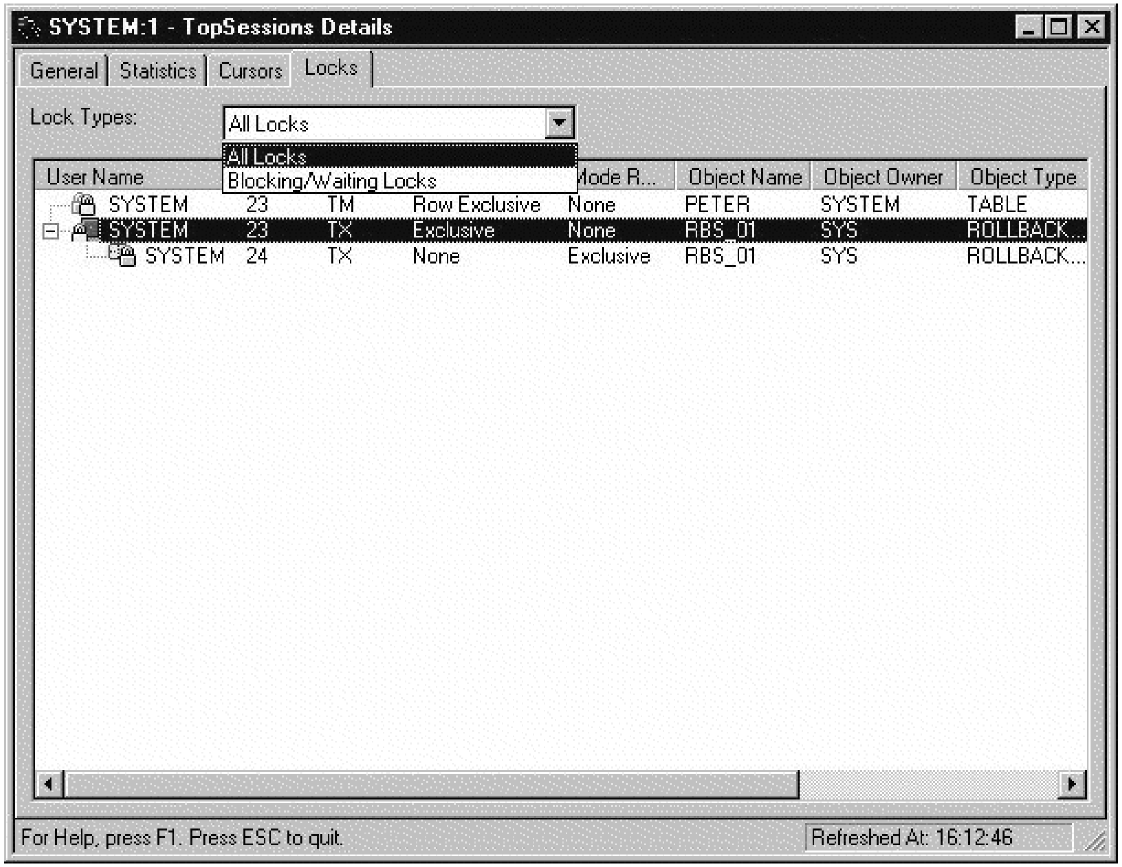
EM TopSessions



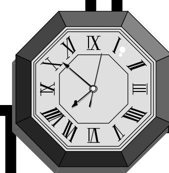
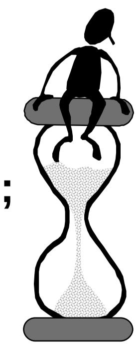
ORACLE®

TopSessions (Diagnostic Pack)

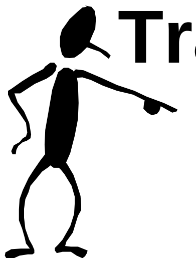
- **All Locks**
- **Blocking/Waiting Locks**

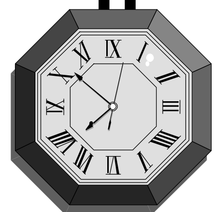


Guidelines: Resolve Contention

Transaction 1		Transaction 2
> UPDATE emp SET sal = sal x 1.1 WHERE empno = 1000;	9:00	
	9:05	> UPDATE emp SET sal = sal x 1.1 WHERE empno = 1000; 
	10:30	
>COMMIT/ROLLBACK ; >	11:30	1 row updated; >
>ALTER SYSTEM KILL SESSION ‘10,23’;		

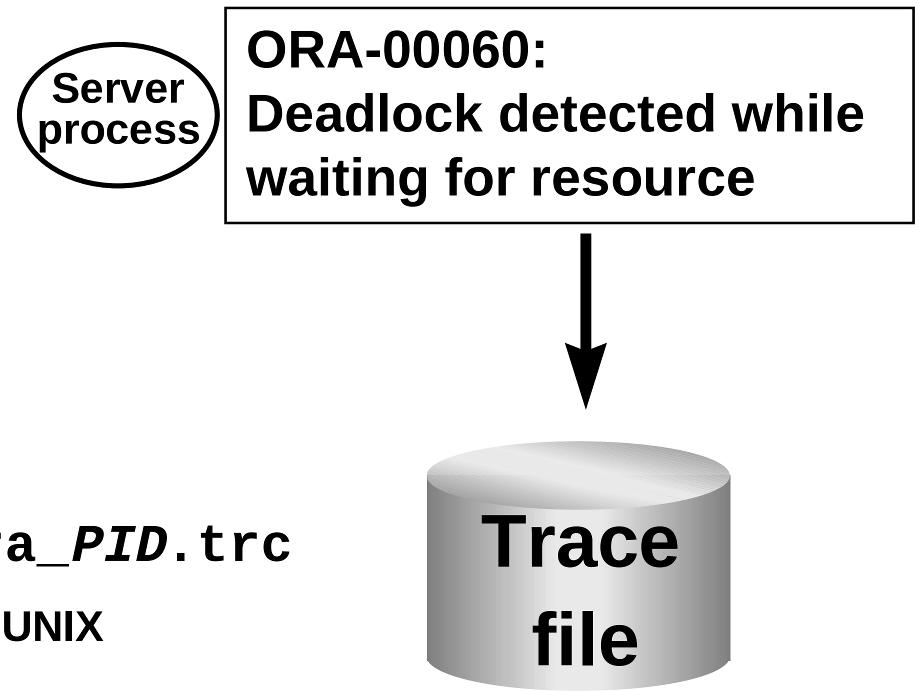
Deadlocks





Transaction 1		Transaction 2
> UPDATE emp SET sal = sal x 1.1 WHERE empno = 1000;	9:00	> UPDATE emp SET mgr = 1342 WHERE empno = 2000;
UPDATE emp SET sal = sal x 1.1 WHERE empno = 2000;	9:15	> UPDATE emp SET mgr = 1342 WHERE empno = 1000;
ORA-00060: Deadlock detected while waiting for resource	9:16	

Deadlocks



***SID*_ora_*PID*.trc**

UNIX

in USER_DUMP_DEST directory

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

13

SQL Issues and Tuning Considerations for Different Applications

ORACLE®

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

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

ORACLE®

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.**

Diagnostic Tools Overview

- EXPLAIN PLAN
- SQL Trace and TKPROF
- SQL*Plus AUTOTRACE
- Oracle SQL Analyze

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.**

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.**

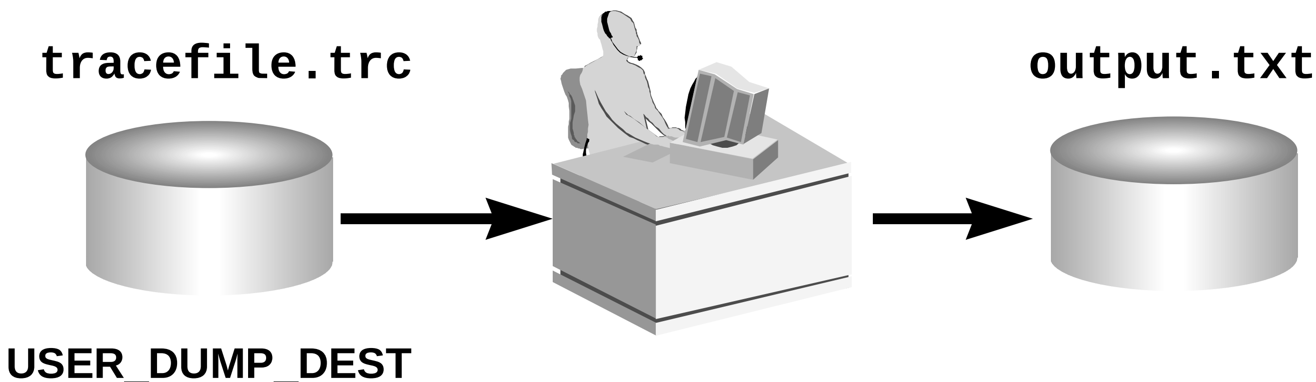
Enabling and Disabling SQL Trace

- **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});
```

Formatting the Trace File with TKPROF

```
$ tkprof tracefile.trc output.txt [options]
```



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

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 ]
```

Optimizer Modes

- **Rule-based:**
  - **Uses a ranking system**
  - **Syntax- and data dictionary–driven**
- **Cost-based:**
  - **Chooses least-cost path**
  - **Statistics-driven**

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**

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

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

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

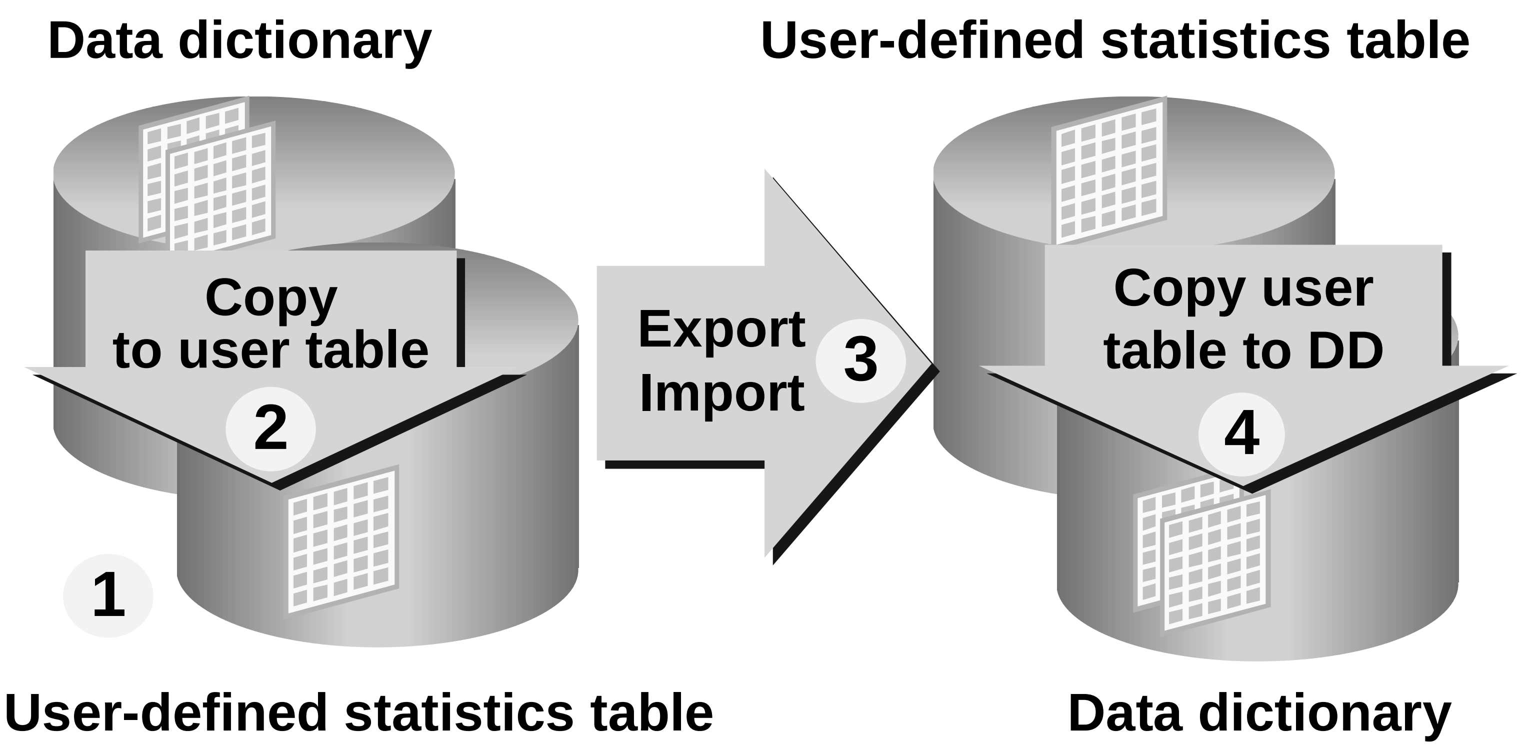
Column Statistics

- Number of distinct values
- Lowest value, highest value
- Last ANALYZE date and sample size
- Data dictionary view:
  USER_TAB_COL_STATISTICS

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

Copying Statistics Between Databases



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     */
);
```

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**

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

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;
```

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

Maintaining Stored Outlines

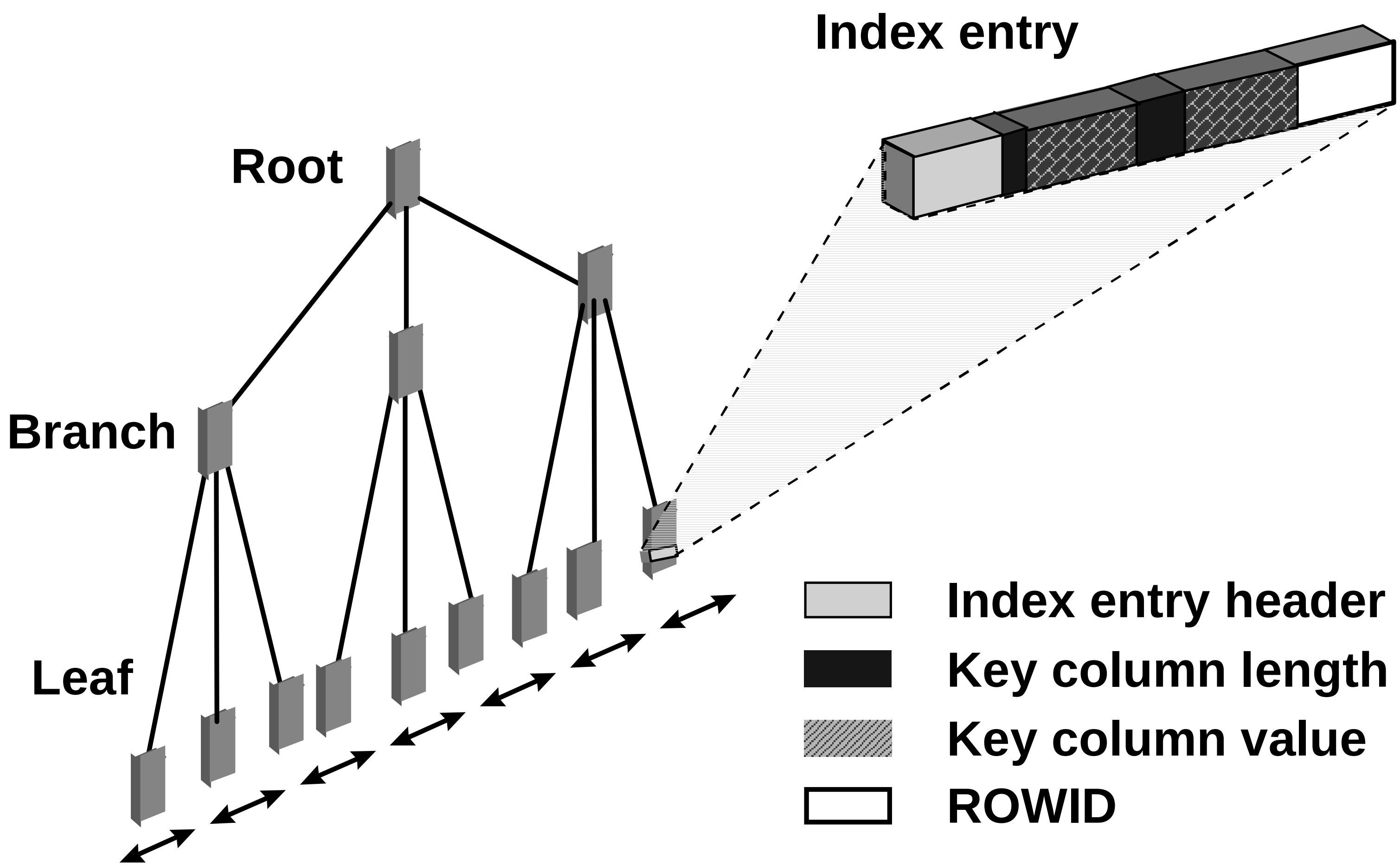
- Use the OUTLN_PKG package to:
  - Drop outlines or categories of outlines
  - Rename categories
- Use the ALTER OUTLINE command to:
  - Rename an outline
  - Rebuild an outline
  - Change the category of an outline
- Outlines are stored in OUTLN schema

Data Access Methods

To enhance performance, you can use the following data access methods:

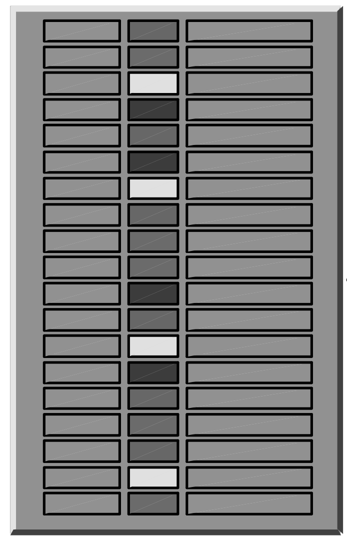
- **Indexes (B-tree, bitmap, reverse key)**
- **Index-organized tables**
- **Clusters**
- **Histograms**
- **Materialized views**

B-Tree Index


Index entry
Root
Branch
Leaf
Index entry header
Key column length
Key column value
ROWID

Bitmap Index

Table

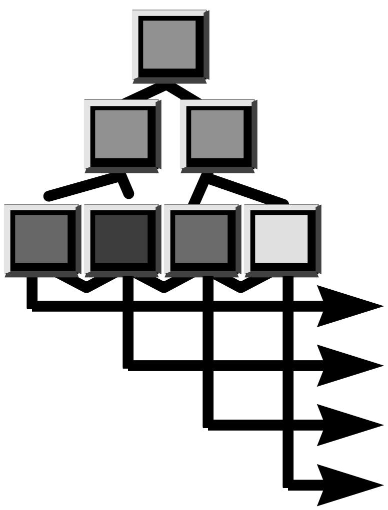


File 3
Block 10

Block 11

Block 12

Index



Key	Start ROWID	End ROWID	Bitmap
<Blue,	10.0.3,	12.8.3,	1000100100010010100>
<Green,	10.0.3,	12.8.3,	0001010000100100000>
<Red,	10.0.3,	12.8.3,	0100000011000001001>
<Yellow,	10.0.3,	12.8.3,	0010001000001000010>

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

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;
```

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

Reverse Key Index

KEY	ROWID
1257	0000000F.0002.0001
2877	0000000F.0006.0001
4567	0000000F.0004.0001
6657	0000000F.0003.0001
8967	0000000F.0005.0001
9637	0000000F.0001.0001
9947	0000000F.0000.0001
...	...

Index on EMPNO

EMPNO	ENAME	JOB	...
7499	ALLEN	SALESMAN	
7369	SMITH	CLERK	
7521	WARD	SALESMAN	...
7566	JONES	MANAGER	
7654	MARTIN	SALESMAN	
7698	BLAKE	MANAGER	
7782	CLARK	MANAGER	
...	...		...

EMP Table

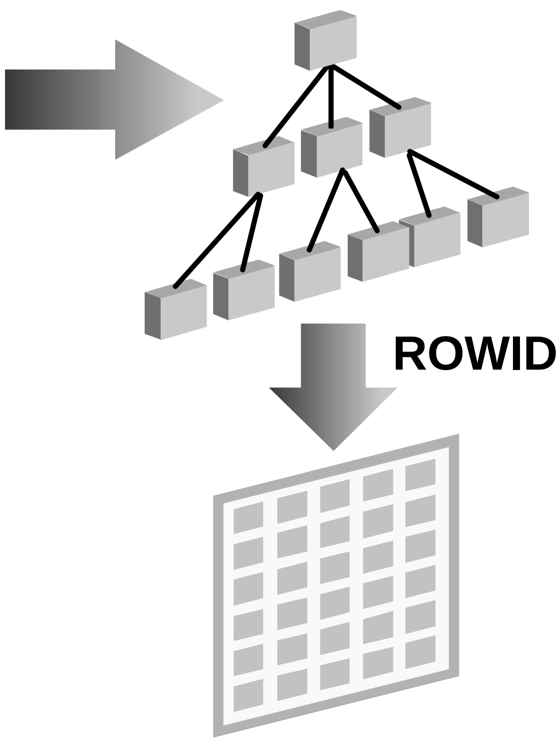
Creating Reverse Key Indexes

```
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;
```

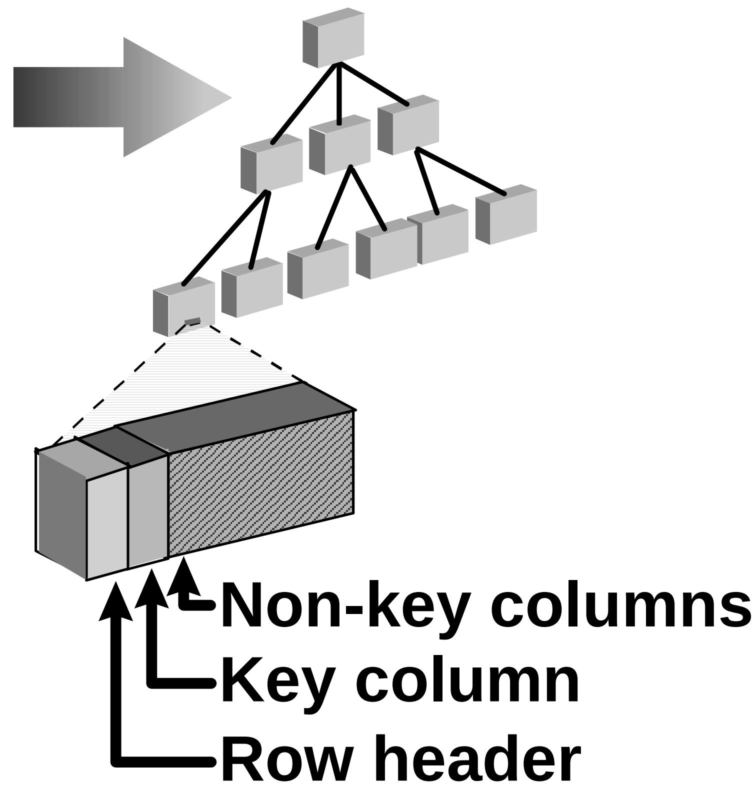
```
SQL> create unique index i2_t1 ON t1(c2);
SQL> alter index i2_t1 REBUILD REVERSE;
```

Index-Organized Tables

Regular table access



IOT access



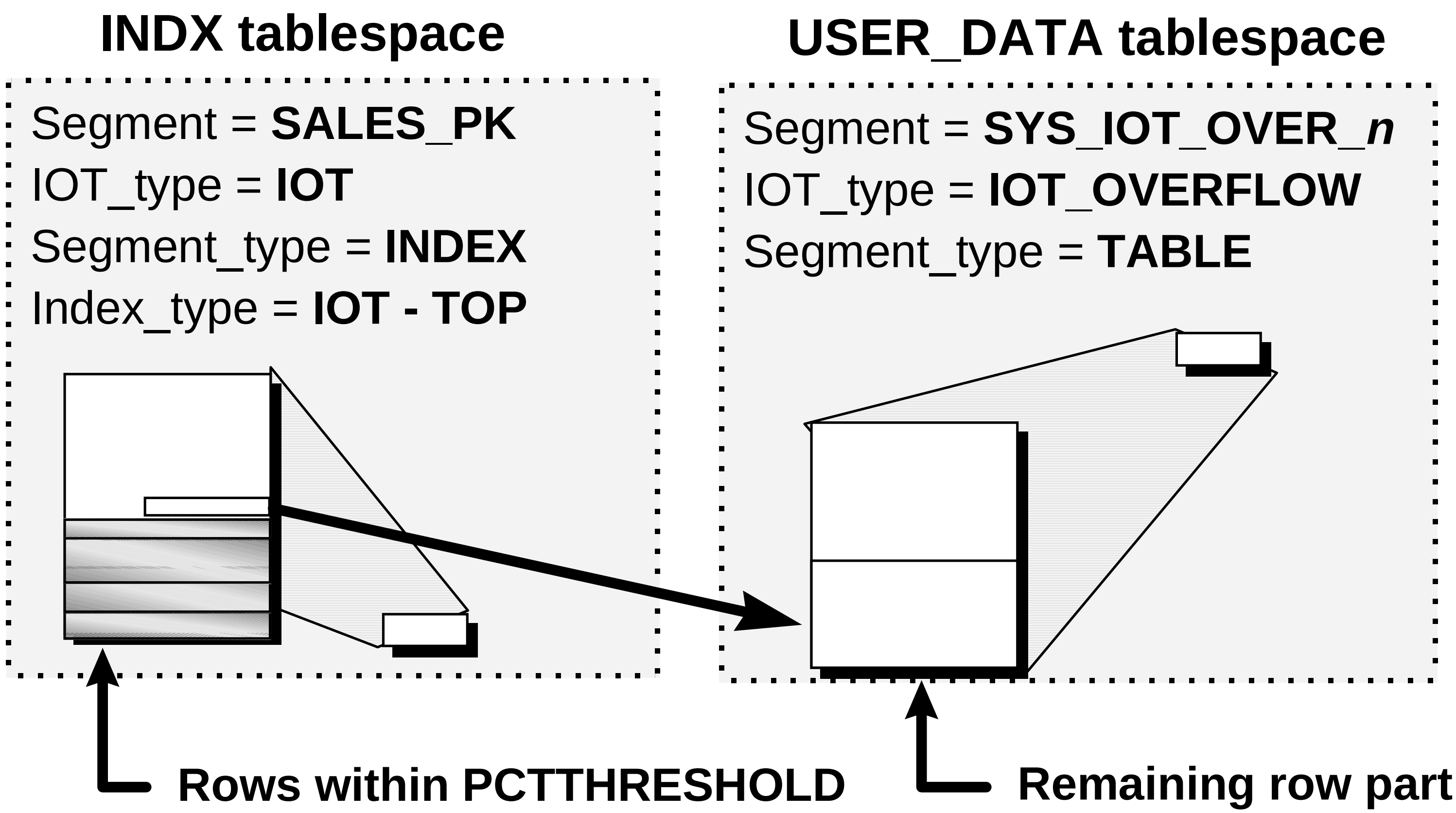
Index-Organized Tables Compared with Regular Tables

- **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**

Creating Index-Organized Tables

```
SQL> create table sales
     (office_cd      number(3)
     ,qtr_end        date
     ,revenue        number(10,2)
     ,review         varchar2(1000)
     ,constraint sales_pk
      PRIMARY KEY (office_cd,qtr_end)
     )
     ORGANIZATION INDEX tablespace indx
     PCTTHRESHOLD 20
     INCLUDING revenue
     OVERFLOW TABLESPACE user_data;
```

IOT Row Overflow



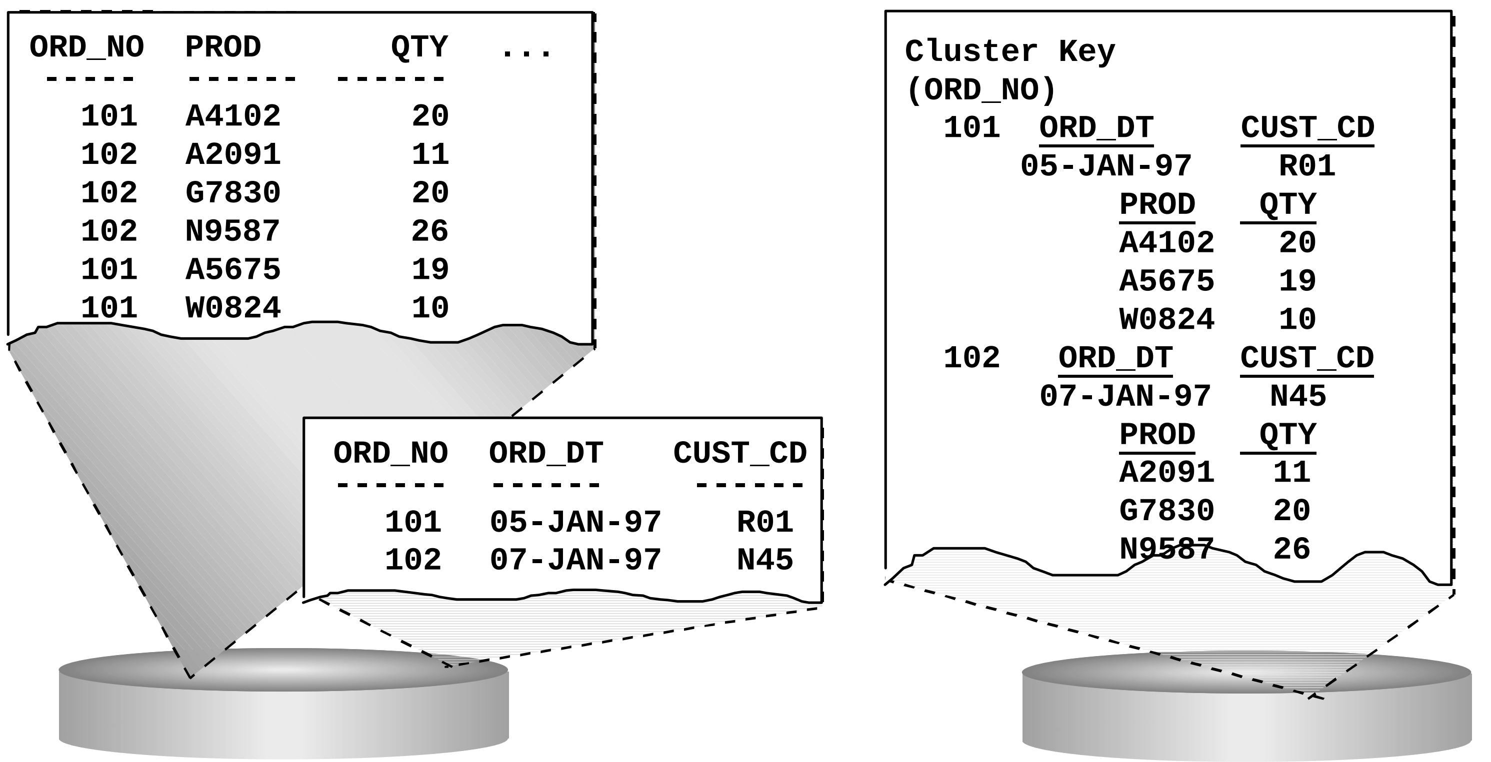
IOT Dictionary Views

```
SQL> select table_name,tablespace_name,iot_name,iot_type
  2  from   DBA_TABLES;
TABLE_NAME         TABLESPACE_NAME  IOT_NAME  IOT_TYPE
-----------------  ---------------  --------  ------------
SALES                                         IOT
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
```

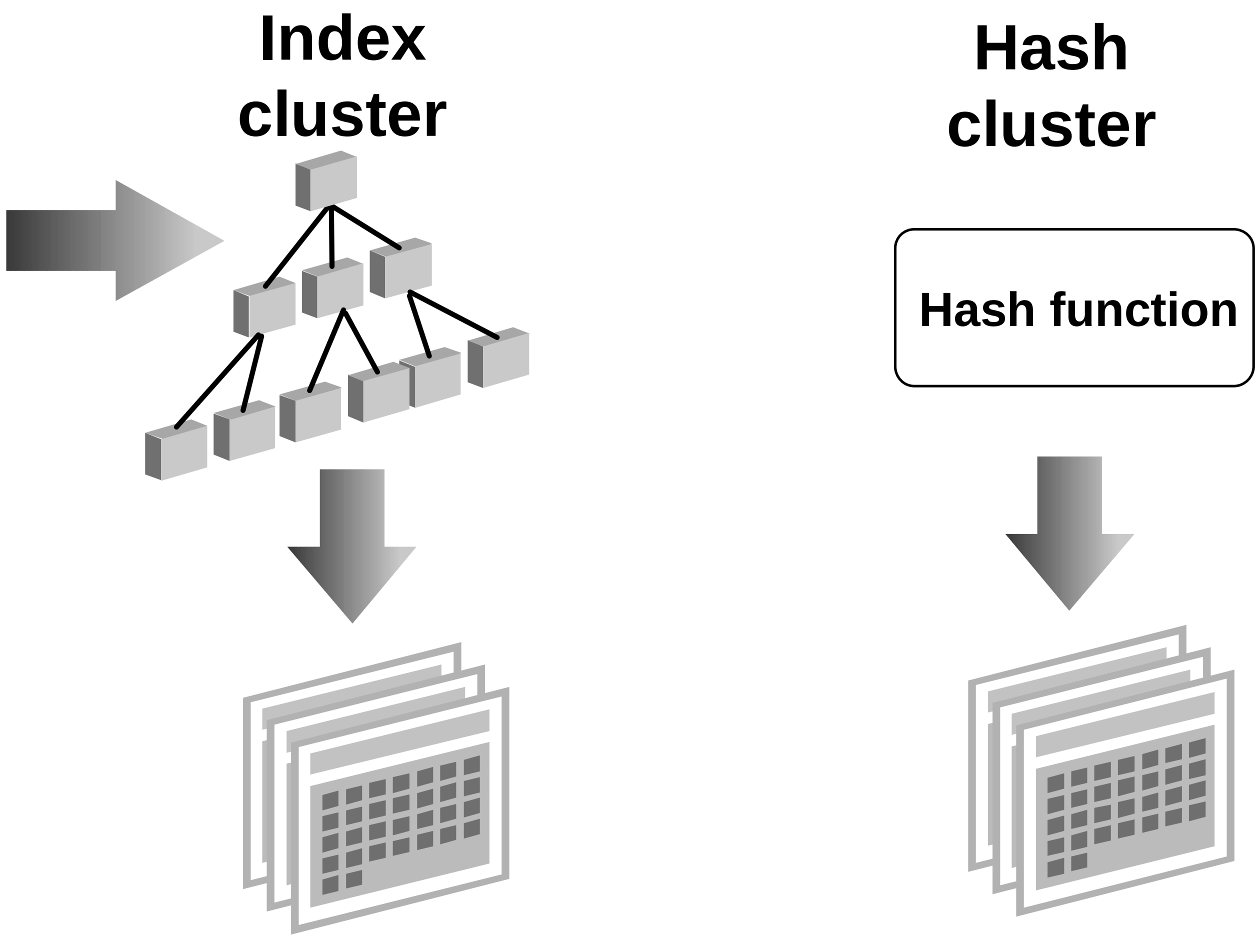
Clusters



Unclustered ORD and ITEM tables

Clustered ORD and ITEM tables

Cluster Types



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

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)

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;
```

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.

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.

Materialized Views and Query Rewrites: Example

```
SQL> create MATERIALIZED VIEW sales_summary
            tablespace sales_ts
            parallel (degree 4)
            BUILD IMMEDIATE REFRESH FAST
            ENABLE QUERY REWRITE
     AS
     select s.zip, p.product_type
     ,      sum(s.amount)
     from   sales s, product p
     where  s.product_id = p.product_id
     group  by s.zip, p.product_type;
```

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
```

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

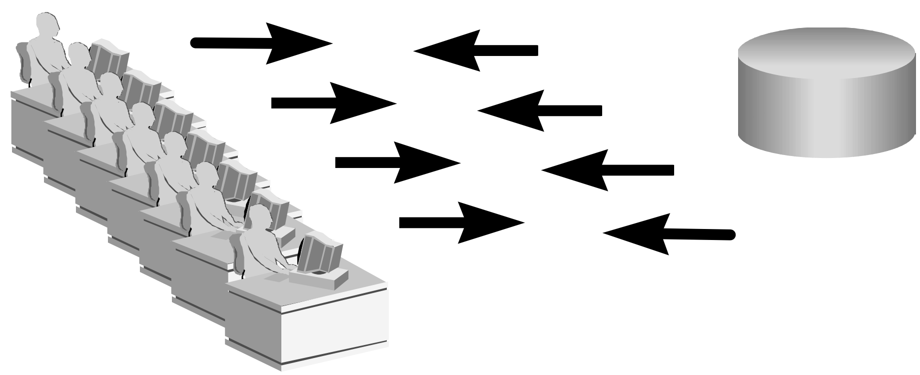
Disabling Query Rewrites: Example

```
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
      . . .
```

OLTP Systems

- High throughput, insert- and update-intensive
- Large, continuously growing data volume
- Concurrent access by many users
- Tuning goals:
  – Availability
  – Speed
  – Concurrency
  – Recoverability



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

OLTP Requirements

- **Short transactions do not require big rollback segments; multiple rollback segments prevent contention.**
- **A large MINEXTENTS value is required.**

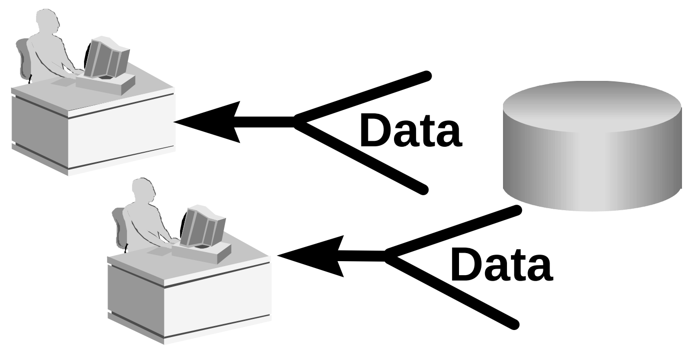
```
SQL> create rollback segment rbs01
  2  storage (initial 100k   next 100k
  3           minextents 20  maxextents 121
  4           optimal 400k )
  5  tablespace rbs;
```

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.

DSS Systems

- Queries on large amounts of data
- Heavy use of full table scans
- Tuning goals:
  - Fast response time
  - Accuracy
  - Availability
- Parallel Query is particularly designed for DSS environments



ORACLE®

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.

DSS Requirements

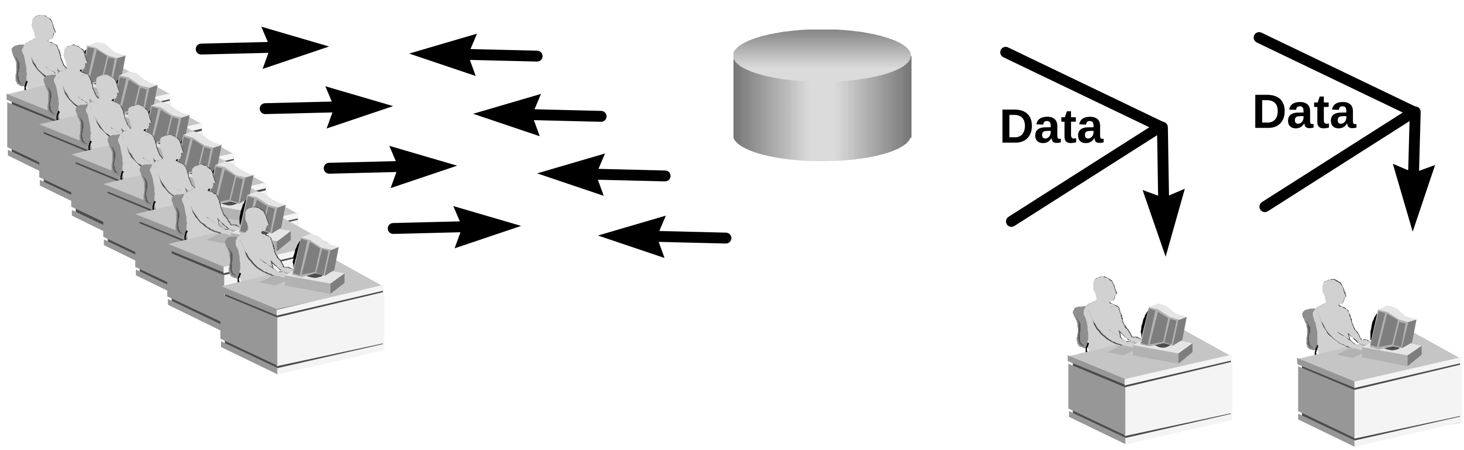
- Evaluate the need for indexes:
  - Use bitmap indexes when possible.
  - Use index-organized tables for (range) retrieval by PK.
  - Generate histograms for indexed columns that are distributed nonuniformly.
- Clustering: Consider hash clusters for performance access.

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.

Multipurpose Applications

- Combination of OLTP and DSS
- Hybrid systems rely on several configurations



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

Parameters for Hybrid Systems

- Memory use:
  - SHARED_POOL_SIZE
  - LARGE_POOL_SIZE
  - DB_BLOCK_BUFFERS
  - SORT_AREA_SIZE
- Parallel Query: Reconfigure parameters for DSS

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

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.

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

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

14

Managing a Mixed Workload

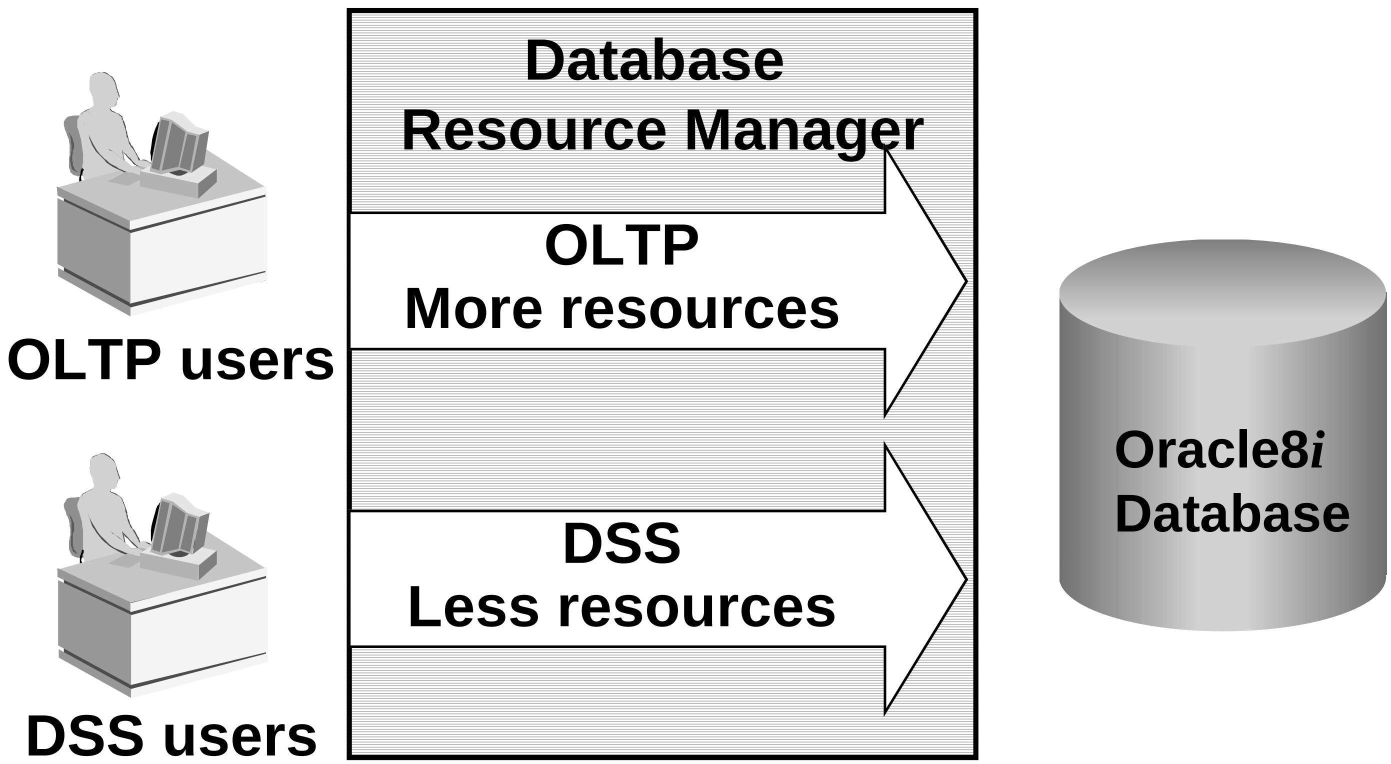
Objectives

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

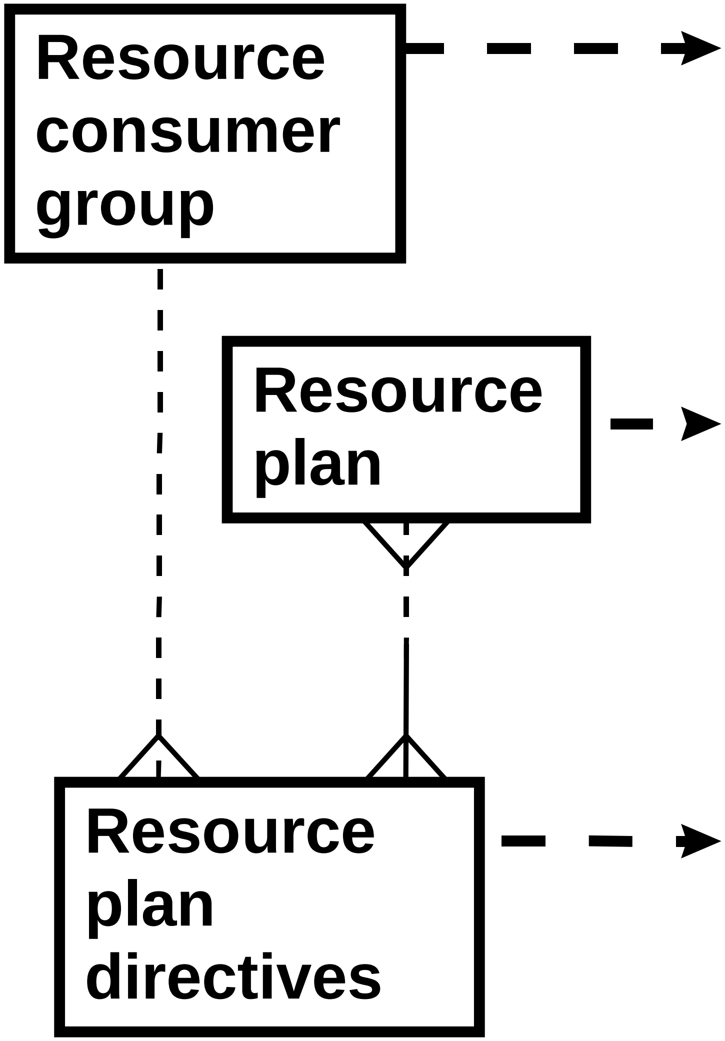
- **List the features of Database Resource Manager**
- **Limit the use of resources using Database Resource Manager**

Overview

- Manage a mixed workload
- Control processing resources used



Database Resource Management Concepts



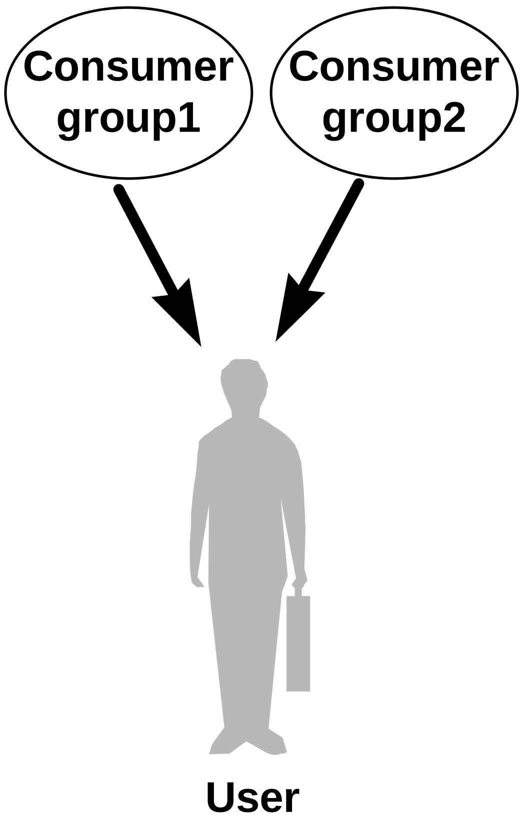
- Groups users with similar resource needs

- Allocates CPU and PQ servers
- Only one plan active

- Assigns groups to plans
- Allocates resources

ORACLE®

Resource Consumer Groups



- Users can be members of multiple consumer groups.
- A default group is assigned to a user at login.
- Only one group is active at a time for a session.
- Either the user or DBA can switch the consumer group during a session.
- Groups are created with the database.

ORACLE®

Resource Plan Directives

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

Database Resource Management Example

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

Steps in Database Resource Management

1. Assign the resource manager system privileges to the administrator.
2. Create resource objects with the package DBMS_RESOURCE_MANAGER:
   - Resource consumer groups
   - Resource plans
   - Resource plan directives
3. Assign users to groups with the package DBMS_RESOURCE_MANAGER_PRIVS.
4. Specify the plan to be used by the instance.

Assigning the Resource Manager Privilege

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

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

Creating Database Resource Manager Objects

2. **Create resource objects with the package DBMS_RESOURCE_MANAGER.**

   2.1. Create a pending area.

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

   2.2. Create resource consumer groups.

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

Creating Database Resource Manager Objects

2.3 Create resource plans.

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

2.4 Create resource plan directives.

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

Creating Database Resource Manager Objects

2.5. Validate the pending area.

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

2.6. Commit the pending area.

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

Assigning Users to Consumer Groups

3. **Assign users to groups.**

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

Set the initial consumer group for users

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

Setting the Resource Plan for an Instance

4. Specify the plan to be used by the instance.
   – Specify the RESOURCE_MANAGER_PLAN initialization parameter.

```
RESOURCE_MANAGER_PLAN=day
```

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

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

Changing a Consumer Group Within a Session

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

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

Changing Consumer Groups for Sessions

- **Can be set by DBA for a session**

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

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

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

Database Resource Manager Information

- **DBA_RSRC_PLANS**

  Resource plans and status

- **DBA_RSRC_PLAN_DIRECTIVES**

  Resource plan directives and status

- **DBA_RSRC_CONSUMER_GROUPS**

  Consumer groups and status

- **DBA_RSRC_CONSUMER_GROUP_PRIVS**

  Users granted consumer groups

- **DBA_USERS**

  Column: INITIAL_RSRC_CONSUMER_GROUP

Current Database Resource Manager Settings

- **V$SESSION: Contains the RESOURCE_CONSUMER_GROUP column that shows the current group for a session**
- **V$RSRC_PLAN: A view that show the active resource plan**
- **V$RSRC_CONSUMER_GROUP: A view that contains statistics by consumer group**

Summary

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

15

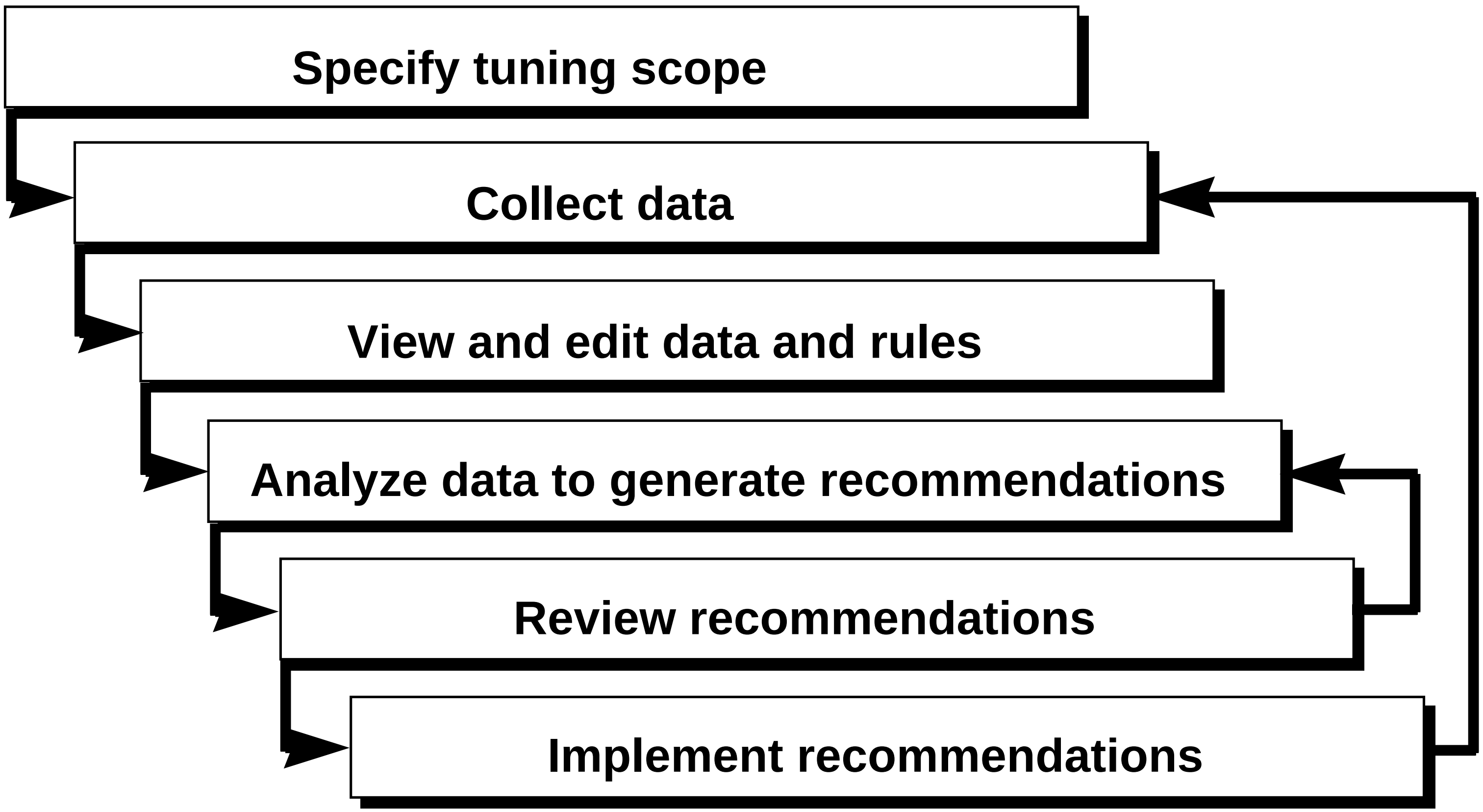
Tuning with Oracle Expert

Objectives

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

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

Overview of Oracle Expert Tuning Methodology



NU-3599A-RA

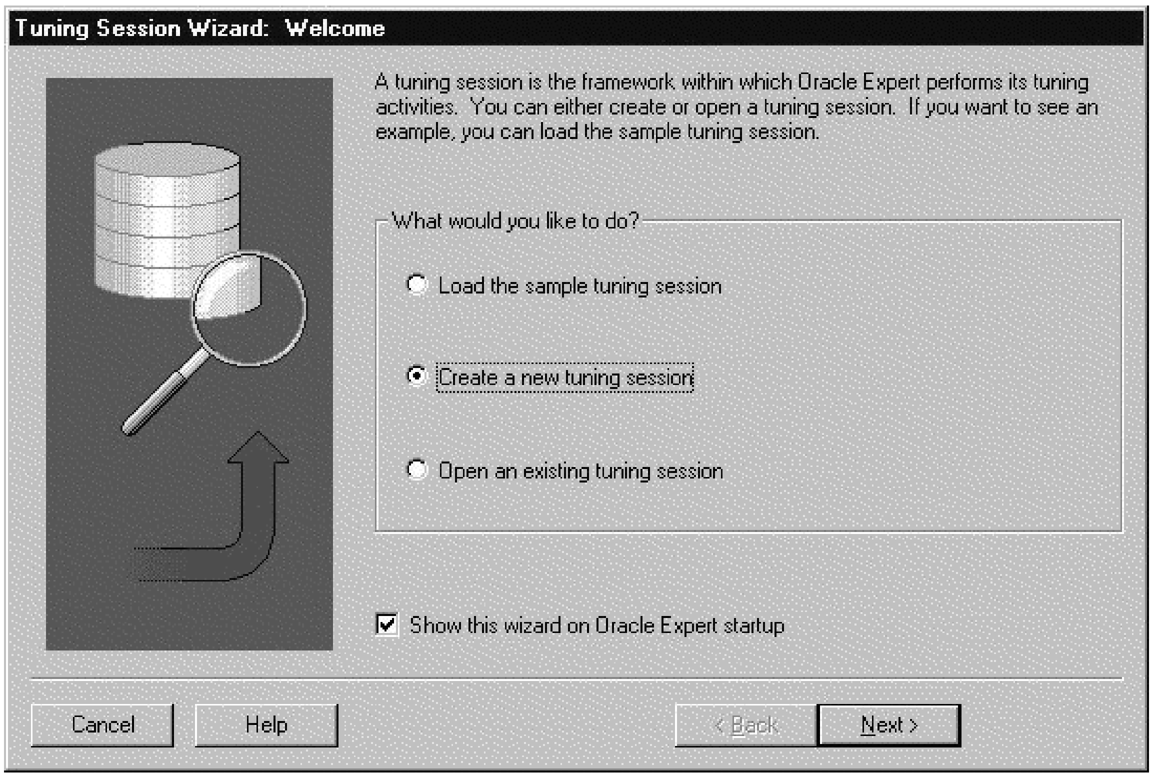
Types of Tuning

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

Starting Oracle Expert

Start from the console using:

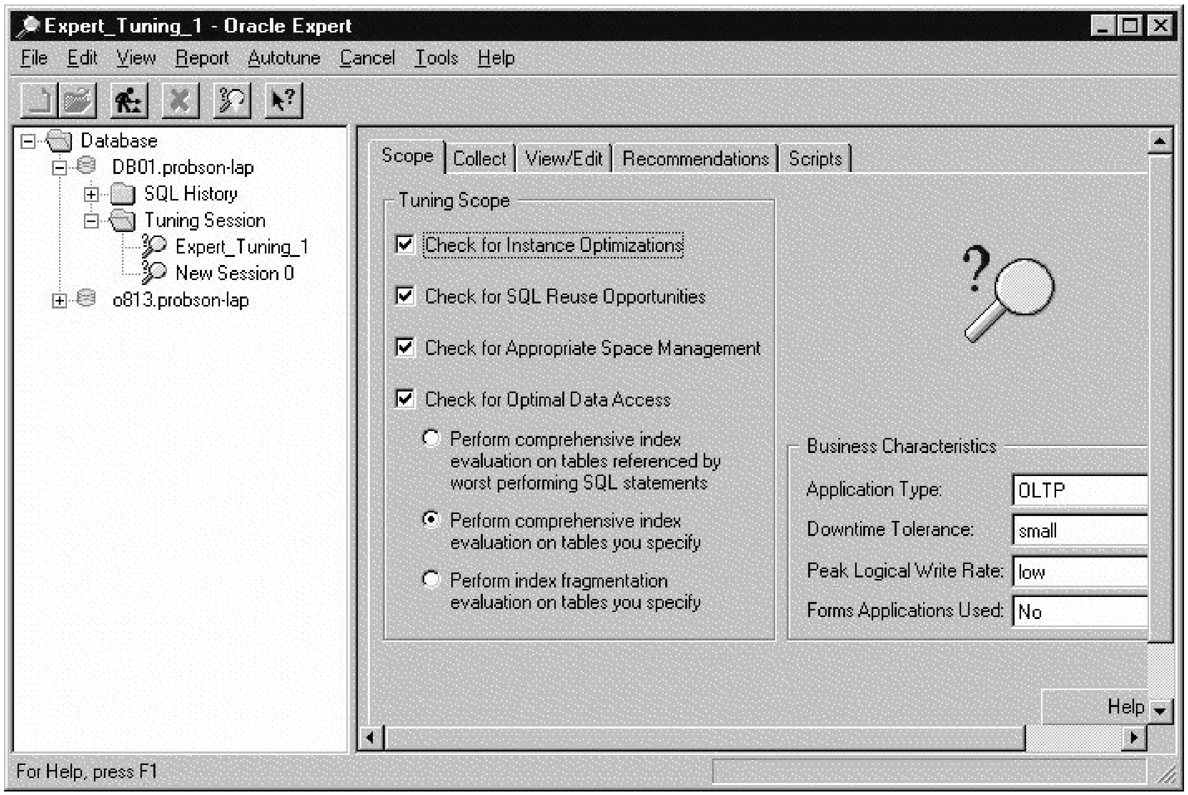
- **The Tools menu option**
- **Tuning Pack tool bar icon**
- **Related Tools from right mouse button menu**



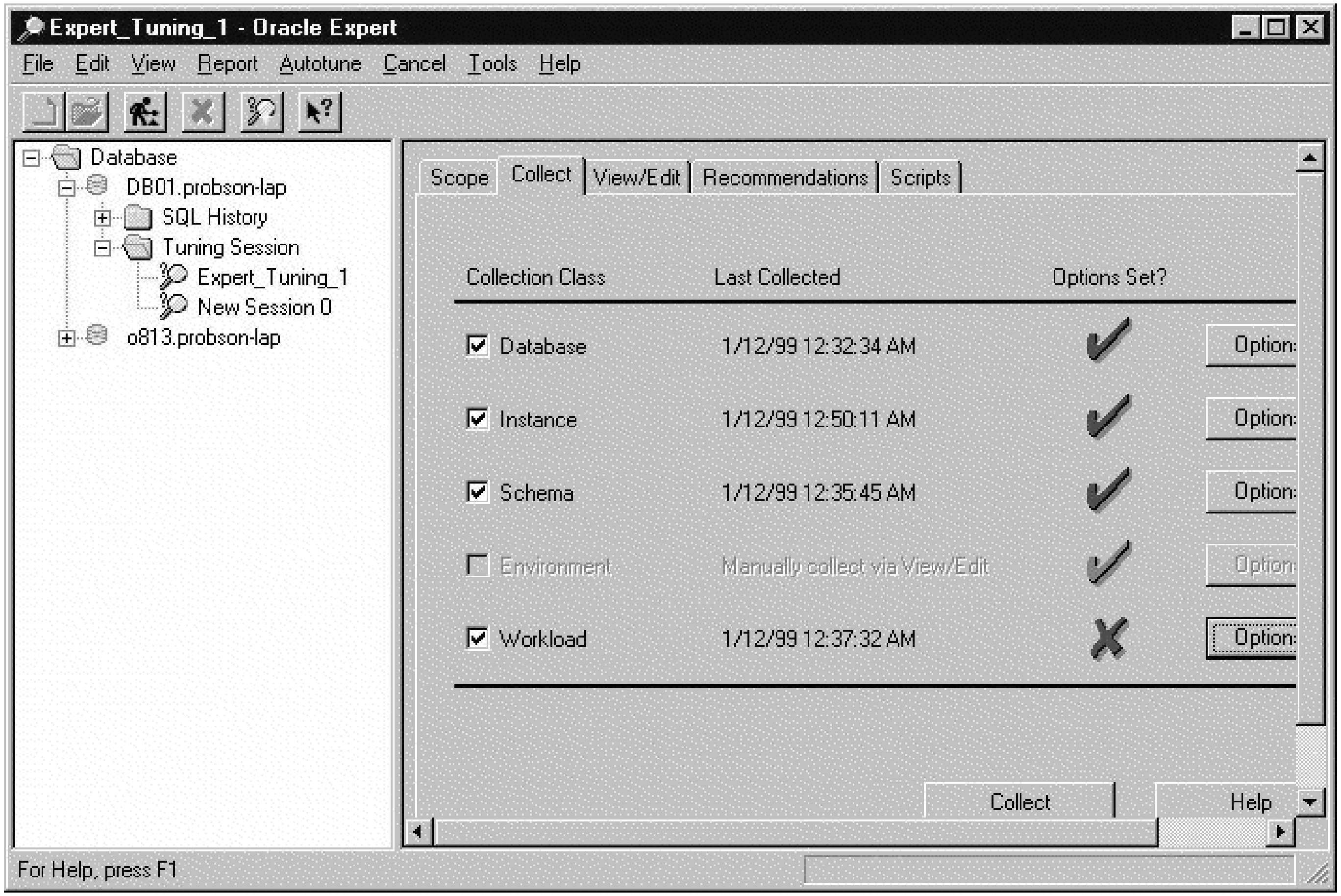
ORACLE®

Tuning Session Scope

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

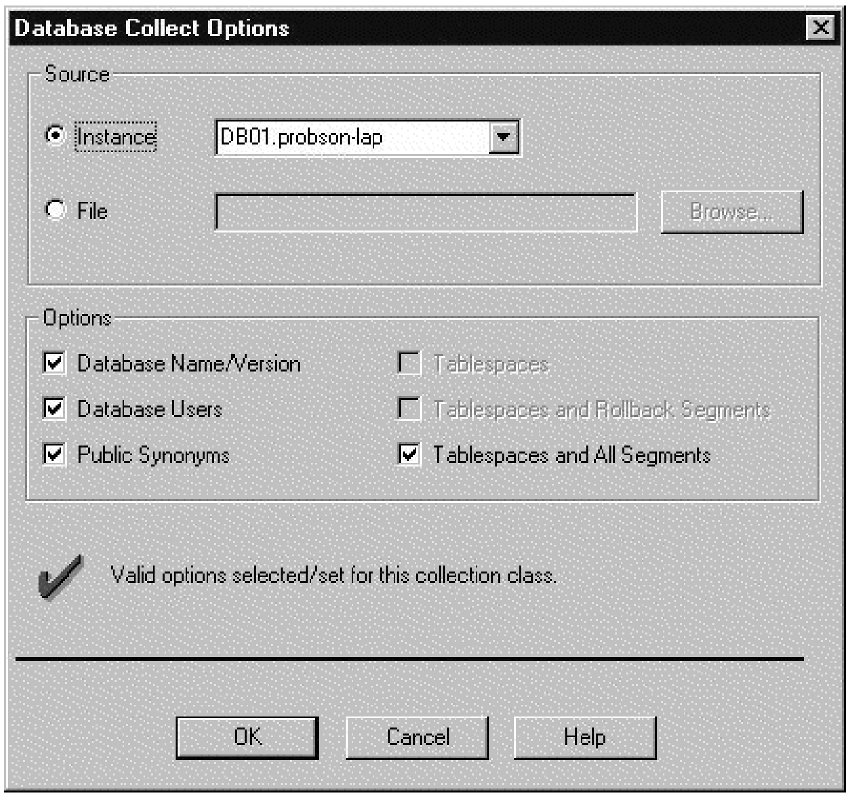


Collecting Input Data

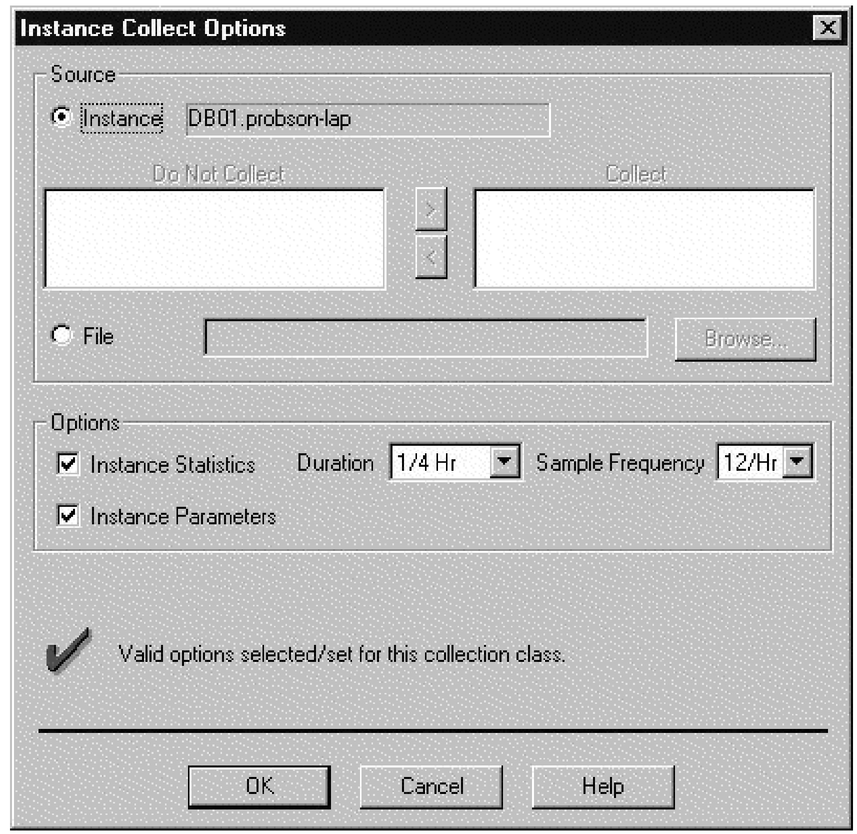


Collecting Data

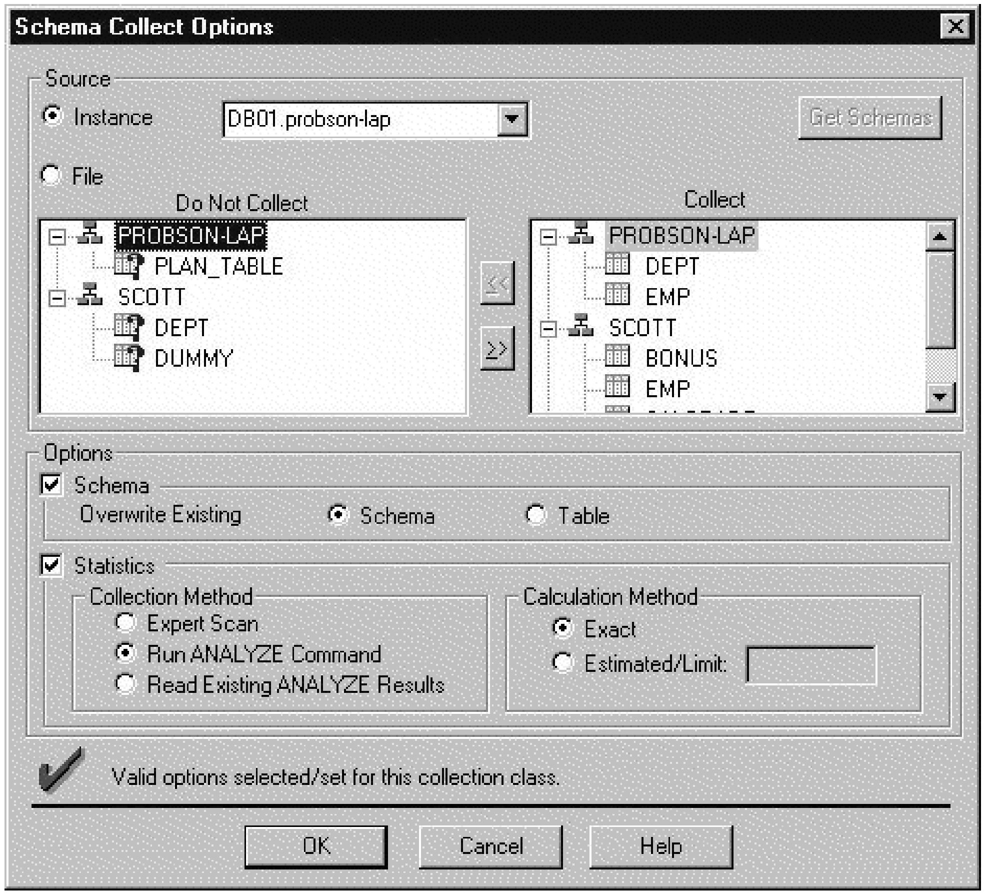
Database Class



Instance Class



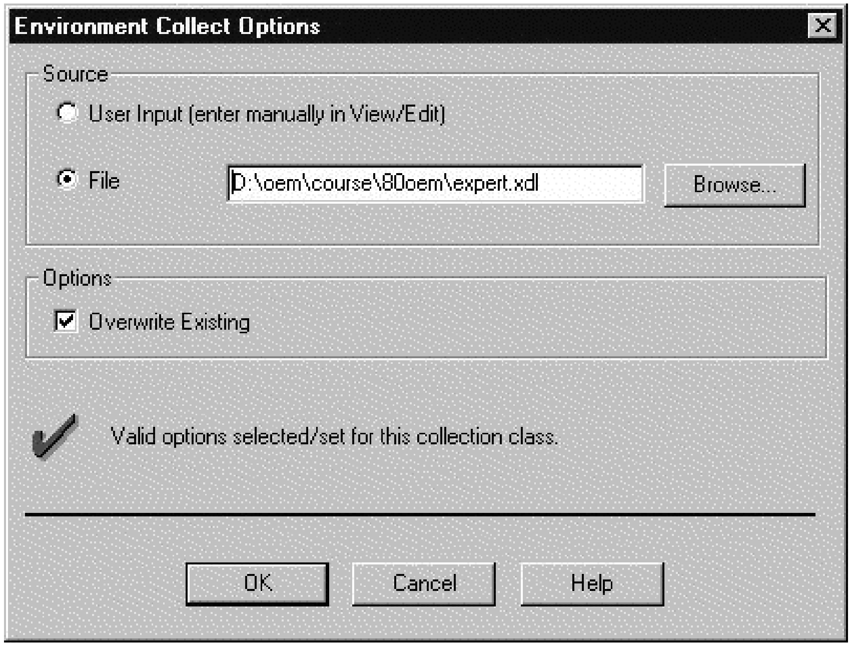
Collecting Schema Class Data



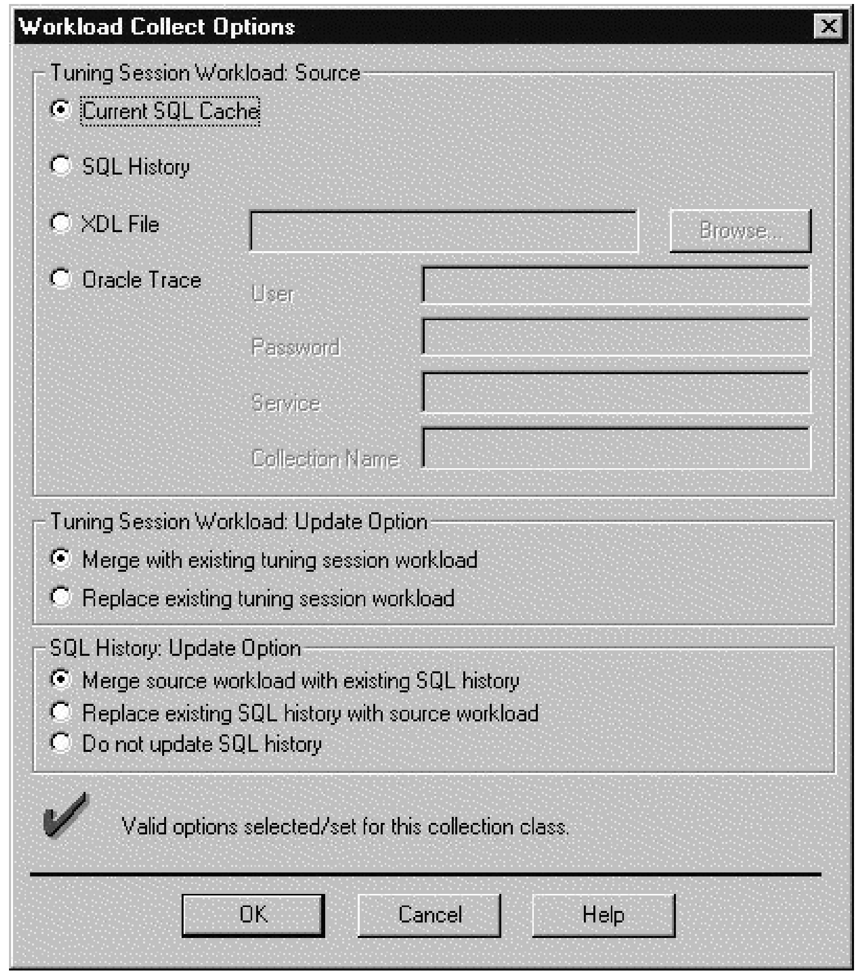
ORACLE®

Collecting Data

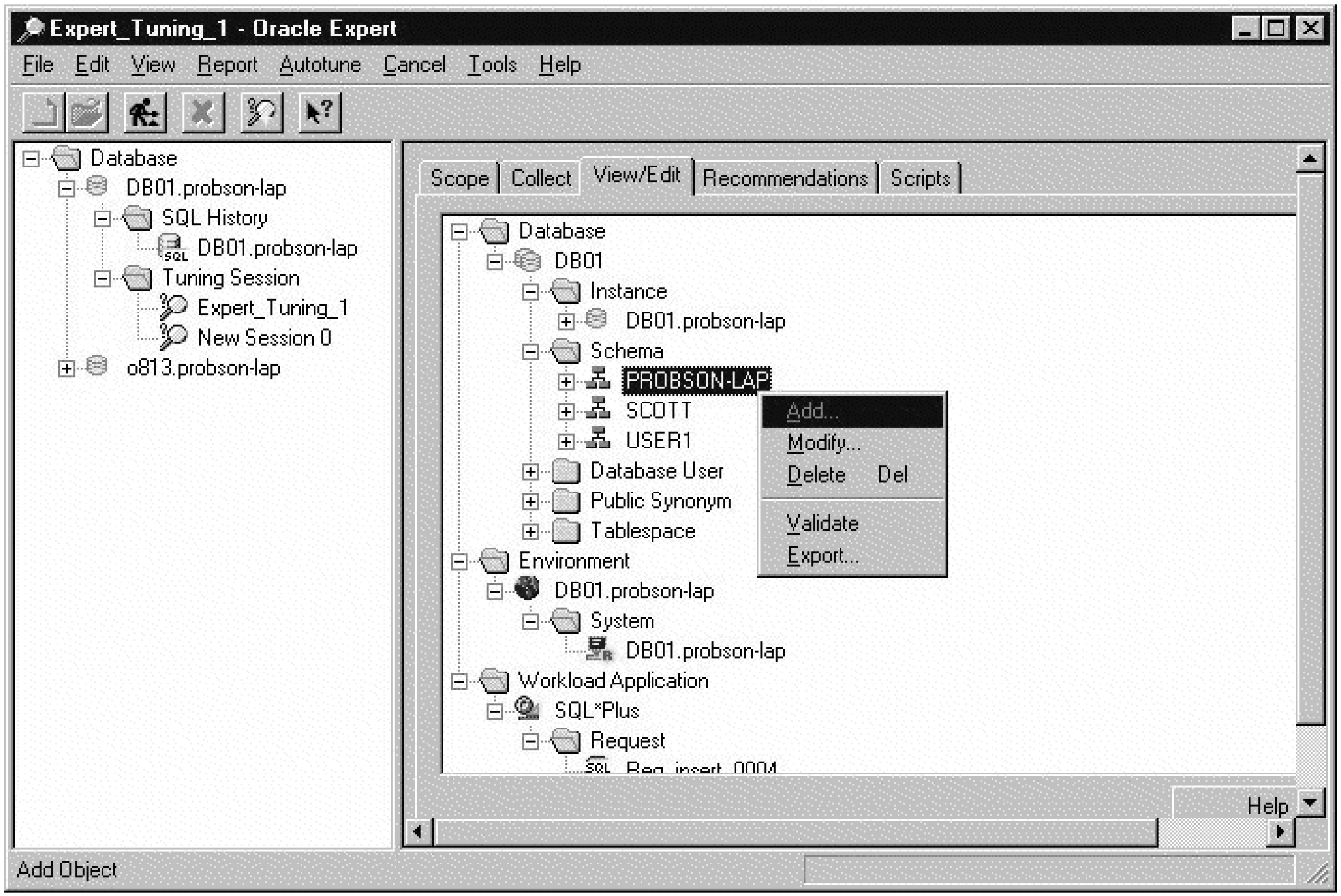
Environment Class



Workload Class

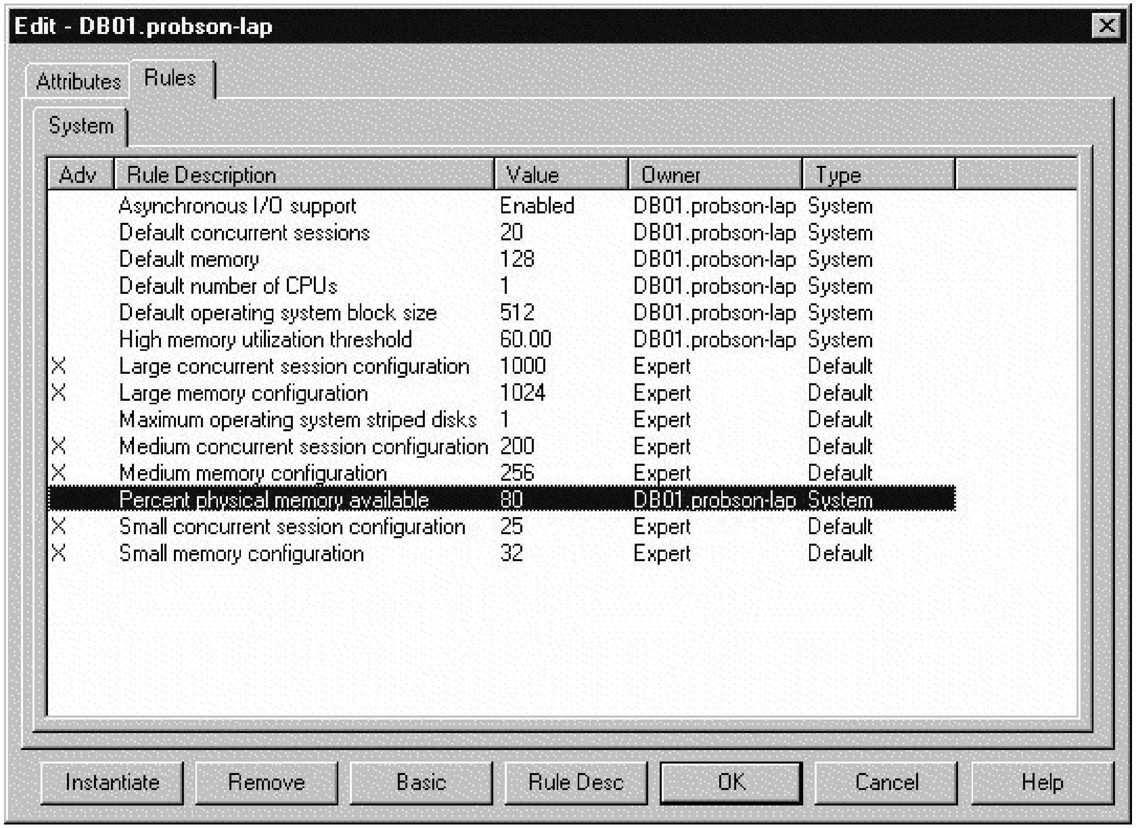


Viewing and Editing the Collected Data

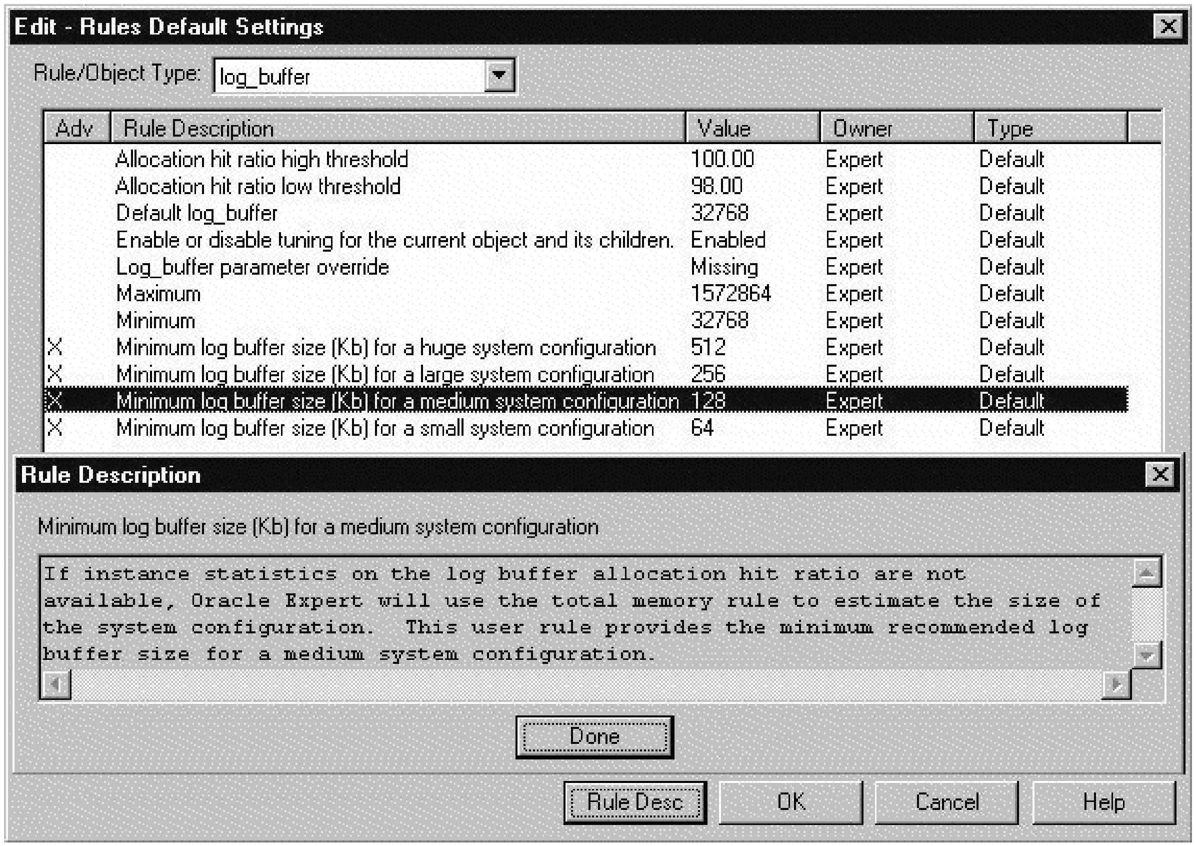


  ORACLE®

Editing Basic Rules Before Analysis



Analyzing Using Default Rules

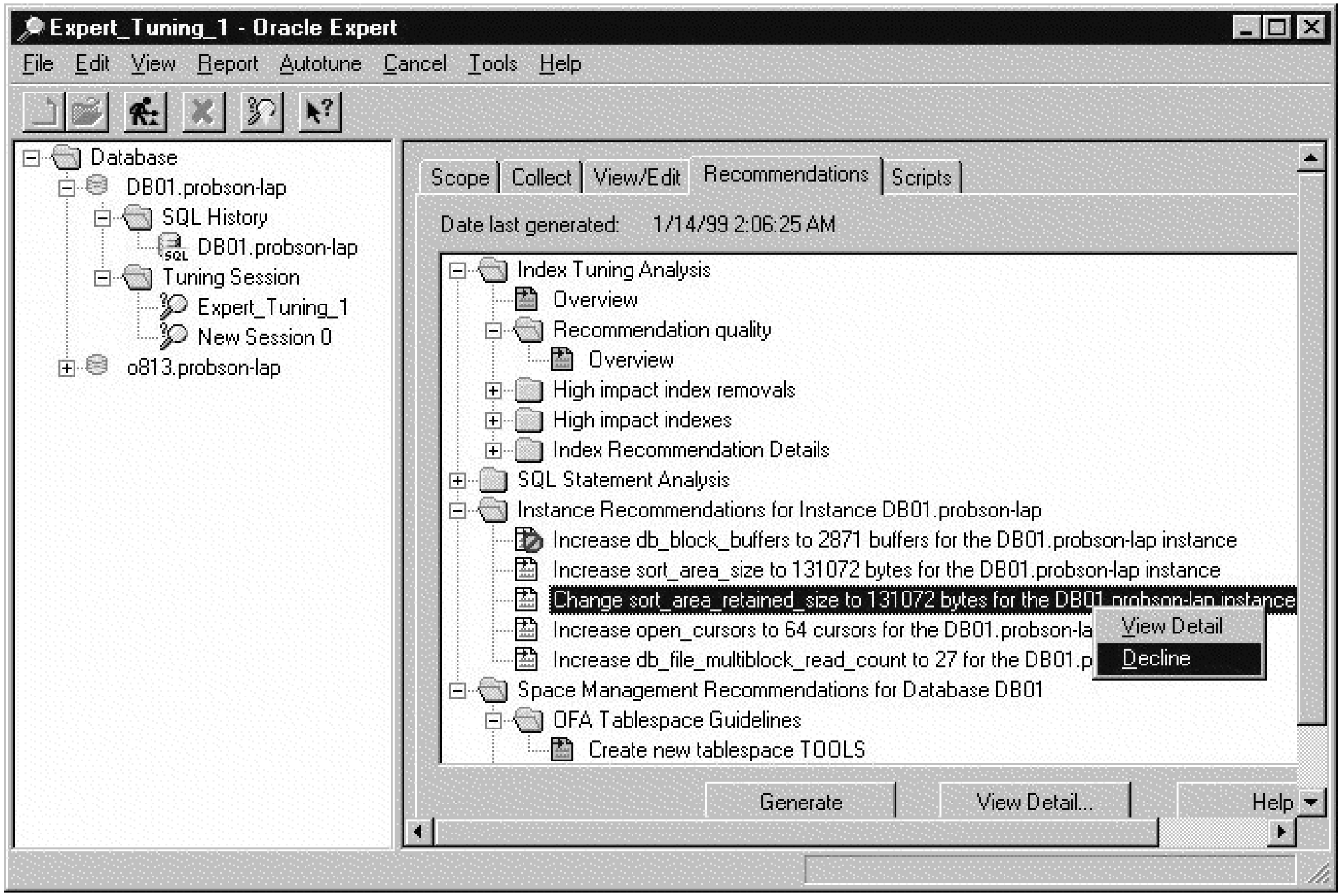


Analyzing the Collected Data

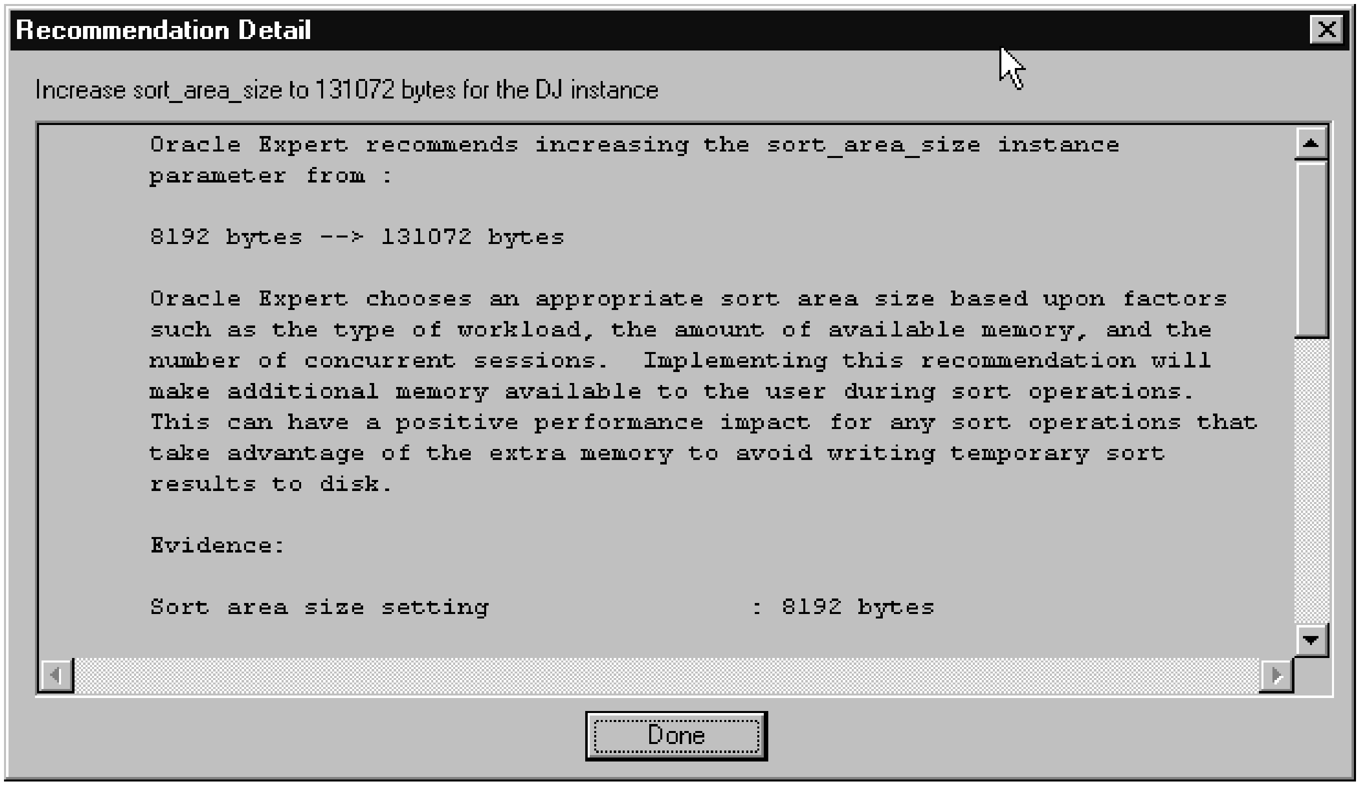
To generate tuning suggestions:

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

Recommendations Overview



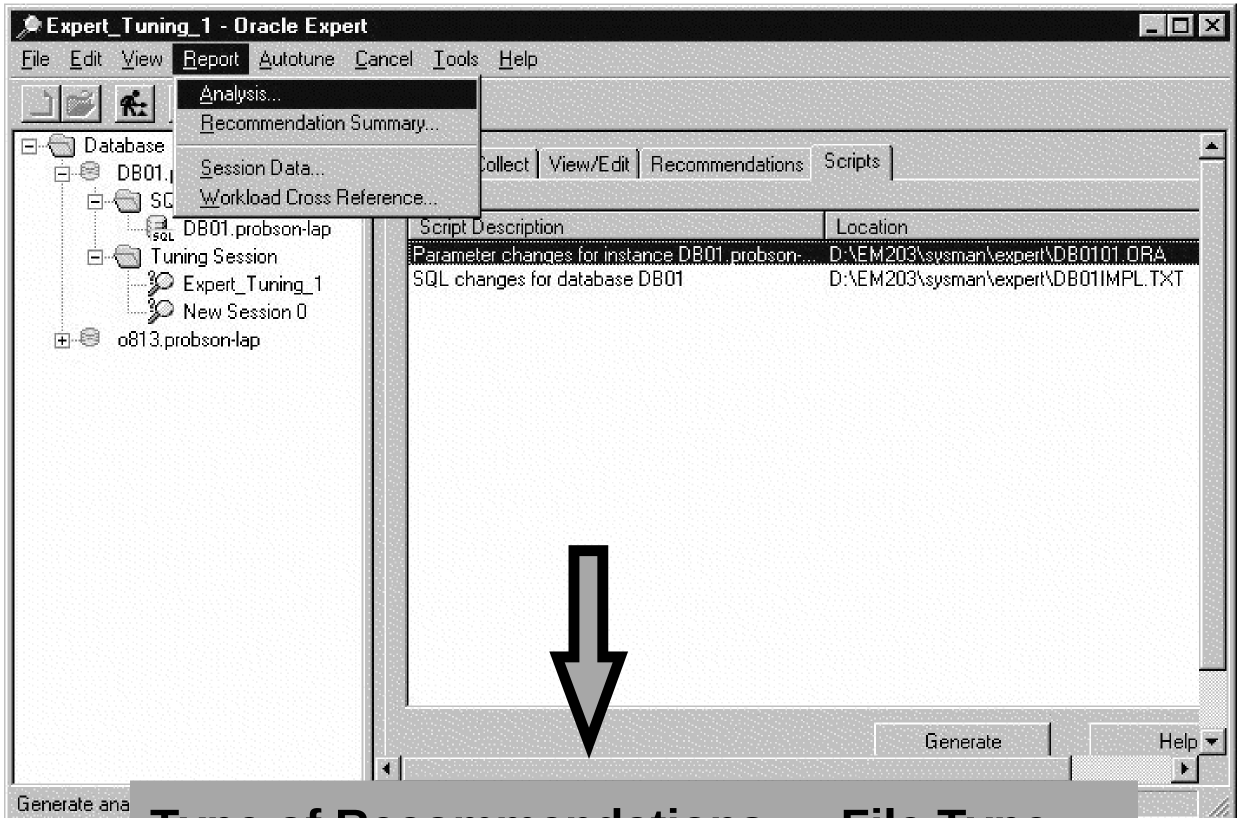
Recommendations Detail



Reports

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

Recommendations Implementation



Type of Recommendations	File Type
Instance	`.ora`
Structure	`.txt`

ORACLE®

Summary

In this lesson, you should have learned how to:

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

16

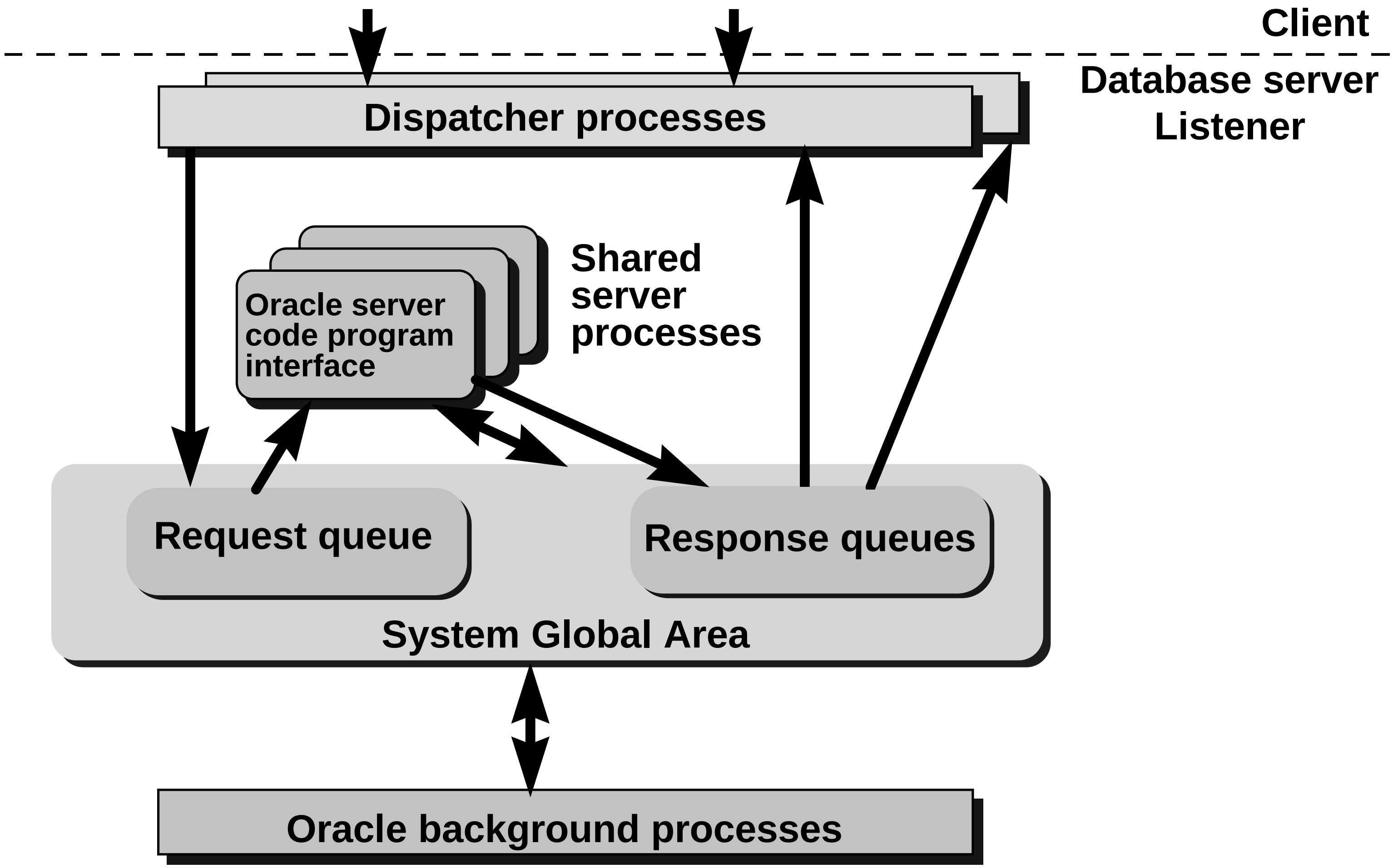
Multithreaded Server Tuning Issues

Objectives

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

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

Overview



ORACLE®

Multithreaded Server Characteristics

- Users can share processes
- Supports NET8*i* functionality
- Increases number of concurrent users
- Is most useful on:
  - UNIX systems
  - Other servers with remote clients
- Incurs some CPU overhead

Configuring the Multithreaded Server

- **NET8*i***
  - `listener.ora`
  - `tnsnames.ora`
- **MTS instance parameters:**

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

Monitoring Dispatchers

Identify contention for dispatchers by checking:

- **Busy rates**
- **Dispatcher waiting time**

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

Monitoring Dispatchers

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

Monitoring Shared Servers

Oracle8*i* starts up shared servers dynamically.

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

Monitoring Process Usage

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

Shared Servers and Memory Usage

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

Possible Problems

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

Obtaining Dictionary Information

Dynamic performance views:

V$CIRCUIT

V$DISPATCHER

V$DISPATCHER_RATE

V$QUEUE

V$MTS

V$SESSION

V$SHARED_SERVER

ORACLE®

Summary

In this lesson, you should have learned that:

- **MTS is a resource-sharing configuration.**
- **MTS is not intended for batch processing or decision support.**
- **MTS requires a Net8*i* listener.**
- **You can monitor dispatcher and server usage.**
- **The Oracle8*i* Server manages shared servers dynamically.**

17

Tuning Workshop

Objectives

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

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

Workshop Methodology

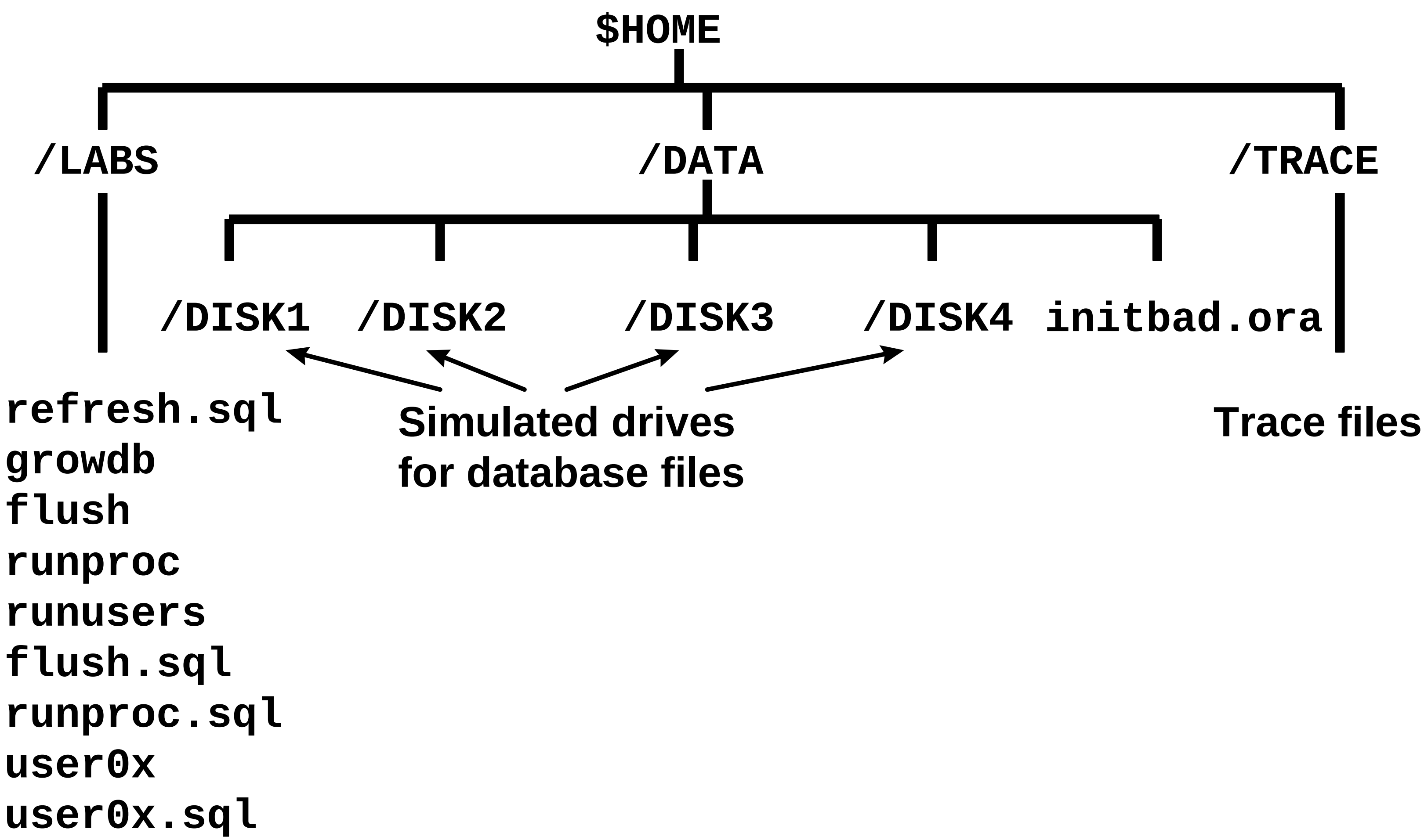
- Group-oriented and interactive
- Intensive hands-on diagnosis and problem resolution
- Instructor-led discussions on findings and actions
- Proactive participant involvement

Tuning Scope

Use Oracle tools to tune the following areas:

- **Memory**
- **I/O**
- **Resource contention**

Workshop Configuration



ORACLE®

Workshop Database Configuration

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

Information Gathering

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

Generate Statistics

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

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

Review Statistics

Review statistics regarding specific areas:

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

Review Statistics

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

Example

Physical investigation:

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

Application Analysis

Application analysis:

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

Regenerate Statistics

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

Results

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

Example

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

Additional Concerns

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

Pending Performance Tuning Issues

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

Summary

In this lesson, you should have learned how to:

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