Oracle9*i* Database Performance Tuning

Volume 2 - Student Guide

D11299GC21

Edition 2.1

June 2003

D38322

Authors

Peter Kilpatrick
Shankar Raman
Jim Womack

Technical Contributors and Reviewers

Mirza Ahmad
David Austin
Ruth Baylis
Howard Bradley
Pietro Colombo
Michele Cyran
Benoit Dagerville
Connie Dialeris
Joel Goodman
Scott Gossett
Lilian Hobbs
Alexander Hunold
Sushil Kumar
Roderick Manalac
Howard Ostrow
Darren Pelacchi
Sander Rekveld
Maria Senise
Ranbir Singh
Janet Stern
Wayne Stokes
Tracy Stollberg
Harald Van Breederode
John Watson

Publisher

Joseph Fernandez

Contents

3 Database Configuration and I/O Issues

4 Tuning the Shared Pool

17

Monitoring and Detecting Lock Contention

Objectives

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

- Define levels of locking
- Identify causes of contention
- Prevent locking problems
- Use Oracle utilities to detect lock contention
- Resolve contention in an emergency
- Resolve deadlock conditions

Locking Mechanism

- Automatic management
- High level of data concurrency
  - Row-level locks for DML transactions
  - No locks required for queries
- Multi-version consistency
- Exclusive and Share lock modes
- Locks held until commit or rollback operations are performed

Lock Management

The Oracle server automatically manages locking. The default locking mechanisms lock data at the lowest level of restriction to guarantee data consistency while allowing the highest degree of data concurrency.

Note: The default mechanism can be modified by the `ROW_LOCKING` parameter. The default value is Always, which leads the Oracle server to always lock at the lowest and least restrictive level (the row level, not the table level) during DML statements. The other possibility is to set the value to Intent, which leads the Oracle server to lock at a more constraining level (the table level), except for a `SELECT FOR UPDATE` statement, for which a row-level lock is used.

Data Concurrency

Transaction 1

```
SQL> UPDATE employees
  2  SET salary=salary*1.1
  3  WHERE id= 24877;
1 row updated.
```

```
SQL> UPDATE employees
  2  SET salary=salary+1200;
13120 rows updated.
```

Transaction 2

```
SQL> UPDATE employees
  2  SET salary=salary*1.1
  3  WHERE id= 24878;
1 row updated.
```

```
SQL> SELECT salary
 2   FROM employees
 3   WHERE id = 10;
   SALARY
---------
     1000
```


Data Concurrency

Locks are designed to allow a high level of data concurrency; that is, many users can safely access the same data at the same time.

- Data Manipulation Language (DML) locking is at row level.
- A query holds no locks, unless the user specifies that it should.

Data Consistency

The Oracle server also provides multi-version consistency; that is, the user sees a static picture of the data, even if other users are changing it.

Duration

Locks are held until the transaction is committed, rolled back, or terminated. If a transaction terminates abnormally, then the PMON process cleans up the locks.

Data Concurrency (continued)

Locking Modes

Exclusive lock mode prevents the associated resource from being shared with other transactions, until the exclusive lock is released.

Example: Exclusive locks are set at row level for a DML transaction:

Transaction 1	Transaction 2

`SQL> UPDATE employees`
`  2  SET salary=salary*1.1`
`  3  WHERE id= 24877;`
`1 row updated.`
	`SQL> UPDATE employees`
	`  2  SET salary=salary*1.1`
	`  3  WHERE id= 24877;`
	`Transaction 2 waits.`

In Share lock mode, several transactions can acquire share locks on the same resource.

Example: Shared locks are set at table level for DML transactions:

Transaction 1	Transaction 2

`SQL> UPDATE employees`
`  2  SET salary=salary*1.1`
`  3  WHERE id= 24877;`
`1 row updated.`
	`SQL> UPDATE employees`
	`  2  SET salary=salary*1.1`
	`  3  WHERE id= 24878;`
	`1 row updated.`

The two transactions update different rows in the same table.

Lock Duration

Transactions hold locks until the transactions are committed or rolled back:

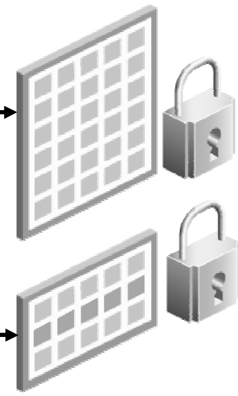
Transaction 1	Transaction 2

`SQL> UPDATE employees`
`  2  SET salary=salary*1.1`
`  3  WHERE id= 24877;`
`1 row updated.`
`SQL> commit;`
`Commit complete.`
	`SQL> UPDATE employees`
	`  2  SET salary=salary*1.1`
	`  3  WHERE id= 24877;`
	`Transaction 2 waits until`
	`transaction 1 is committed.`
	`1 row updated.`

As soon as Transaction 1 is committed, Transaction 2 can update the row, because the transaction acquired the requested lock. Transaction 2 must wait because it wants to update the same row as Transaction 1.

Two Types of Locks

- DML or data locks:
  - Table-level locks
  - Row-level locks
- DDL or dictionary locks



DML Locks

DML locks guarantee the integrity of data being accessed concurrently by multiple users for incorporating changes. They prevent destructive interference of simultaneous conflicting DML and DDL operations.

DML Levels: A table-level lock (TM type) is set for any DML transaction that modifies a table: `INSERT`, `UPDATE`, `DELETE`, `SELECT...FOR UPDATE`, or `LOCK TABLE`. The table lock prevents DDL operations that would conflict with the transaction.

Example

Transaction 1	Transaction 2

`SQL> UPDATE employees`
`  2  SET salary=salary*1.1;`
`13120 rows updated.`
	`SQL> DROP TABLE employees;`
	`ERROR at line 1:`
	`ORA-00054: resource busy and`
	`acquire with NOWAIT specified`

DML Locks (continued)

The row-level lock (TX type) is automatically acquired for each row modified by `INSERT`, `UPDATE`, `DELETE`, or `SELECT...FOR UPDATE` statements. The row-level lock ensures that no other user can modify the same row at the same time. Therefore, there is no risk that a user can modify a row that is being modified and not yet committed by another user.

Example

Transaction 1	Transaction 2

`SQL> UPDATE employees`
`  2   SET salary=salary*1.1`
`  3   WHERE id= 24877;`
`1 row updated.`
	`SQL> UPDATE employees`
	`  2   SET salary=salary*1.1`
	`  3   WHERE id= 24877;`
	`Transaction 2 waits.`

DDL Locks

A DDL lock protects the definition of a schema object while that object is acted upon or referred to by an ongoing DDL operation. The Oracle server automatically acquires a DDL lock to prevent any destructive interference from other DDL operations that might modify or reference the same schema object.

DML Locks

A DML transaction gets at least two locks:

- **A shared table lock**
- **An exclusive row lock**

DML Transactions Acquire at Least Two Locks

Two kinds of lock structures are used for DML statements (`INSERT`, `UPDATE`, `DELETE`, or `SELECT...FOR UPDATE`):

- The transaction gets a shared lock on the table that is referenced as a TM lock, no matter what shared lock mode it is.
- The transaction gets an exclusive lock on the rows it is changing, referenced as a TX lock. Each row gets a lock byte turned on in the row header pointing to the interested transaction list (ITL) slot used by the transaction. The lock mode at row level can only be exclusive.

Enqueue Mechanism

The enqueue mechanism keeps track of:

- **Users waiting for locks**
- **The requested lock mode**
- **The order in which users requested the lock**

Enqueue Mechanism

The Oracle server maintains all locks as enqueues. The enqueue mechanism keeps track of:

- Users waiting for locks held by other users
- The lock mode these users require
- The order in which users requested the lock

If three users want to update the same row at the same time, all of them get the shared table lock but only one (the first) gets the row lock. The table-locking mechanism keeps track of who holds the row lock and who waits for it.

You can increase the overall number of locks available for an instance by increasing the values of the `DML_LOCKS` and `ENQUEUE_RESOURCES` parameters. This may be necessary in a Real Application Clusters configuration.

Table Lock Modes

These table lock modes are automatically assigned by the Oracle server:

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

Automatic Table Lock Modes

You often see the two TM table lock modes held by DML transactions, RX and RS. These table lock modes are automatically assigned by the Oracle server for DML transactions.

The level of a table lock's mode determines the modes in which other table locks on the same table can be obtained and held.

Row Exclusive (RX)

- Permits other transactions to query, insert, update, delete, or lock other rows concurrently in the same table
- Prevents other transactions manually locking the table for exclusive reading or writing
- Is allocated automatically when using insert, update, delete, or lock statements

Example

Transaction 1 (RX table lock held)	Transaction 2 (RX table lock held)

`SQL> UPDATE employees`
`  2   SET salary=salary*1.1`
`  3   WHERE id= 24877;`
`1 row updated.`
	`SQL> UPDATE employees`
	`  2   SET salary=salary*1.1`
	`  3   WHERE id= 24878;`
	`1 row updated.`

Automatic Table Lock Modes (continued)

Row Share (RS)

- Permits other transactions to query, insert, update, delete, or lock other rows concurrently in the same table.
- This prevents other transactions from manually locking the table for exclusive write access.
- You can choose to lock rows during a query by using the `SELECT ... FOR UPDATE` statement.

Example

<table>
<tr><th>Transaction 1 (RS table lock held)</th><th>Transaction 2 (RX table lock held)</th></tr>
<tr><td><pre>SQL> SELECT id,salary
  2 FROM employees
  3 WHERE id=24877
  4 FOR UPDATE;
       ID   SALARY
--------- ---------
    24877   1100
SQL> COMMIT;
Commit complete.</pre></td><td><pre>SQL> LOCK TABLE employees
  2   IN EXCLUSIVE MODE;
Transaction 2 waits.

Table(s) Locked.</pre></td></tr>
</table>

Manually Locking a Table

Manually acquired in `LOCK TABLE` statement:

```
SQL> LOCK TABLE hr.employees IN share MODE;
```

- **Share (S)**
  - **No DML operations allowed**
  - **Implicitly used for referential integrity**

Manual Table Lock Modes

The table lock modes can be assigned manually by an explicit `LOCK TABLE` command. For example:

```
SQL> LOCK TABLE employees IN exclusive MODE;
Table(s) Locked.
```

Often there are good application reasons for explicit locking but if you get lock contention you may want to check with the developers. Non-Oracle developers sometimes use unnecessarily high locking levels.

The table locking modes available for manual locking include:

Share (S) Lock Mode

This lock mode permits other transactions to only query the `SELECT ... FOR UPDATE` table. It prevents any modification to the table.

Referential integrity statements implicitly get a Share lock.

Manually Locking a Table

- Share Row Exclusive (SRX)
  - No DML operations or Share mode allowed
  - Implicitly used for referential integrity
  - No index is required on the foreign key column in the child table
- Exclusive (X)
  - No DML or DDL operations allowed by other sessions
  - No manual locks allowed by other sessions
  - Queries are allowed

Manual Table Lock Modes (continued)

Share Row Exclusive (SRX) Lock Mode

This is an even higher level of table lock, which prevents DML statements and the manual share lock mode from being acquired. Referential integrity statements implicitly get a Share Row Exclusive lock.

Exclusive (X) Lock Mode

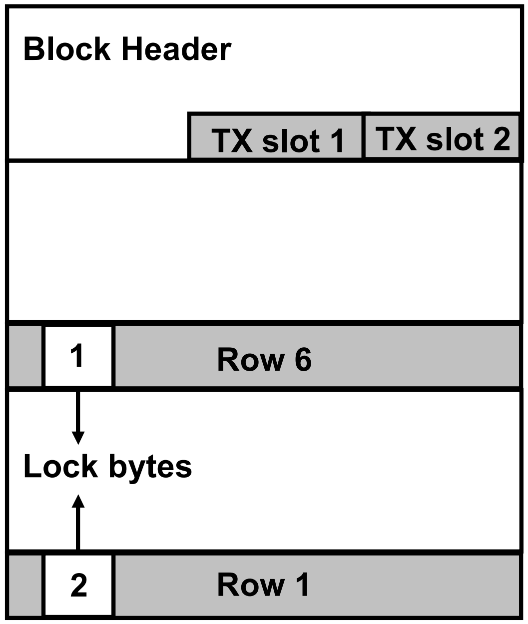
This is the highest level of table lock, thus the most restrictive mode. Exclusive table lock:

- Permits other transactions to only query the table
- Prevents any type of DML statements and any manual lock mode

Example

Transaction 1 (X table lock held)	Transaction 2 (RX table lock requested)
SQL> LOCK TABLE department IN EXCLUSIVE MODE; Table(s) Locked.	SQL> SELECT * from department 2 FOR UPDATE; Transaction 2 waits.

DML Locks in Blocks



Technical Note

This locking information is not cleared out when transactions are committed but rather when the next query reads the block. This is known as delayed block cleanout.

The query that does the cleaning must check the status of the transaction and the system change number (SCN) in the transaction table held in the rollback segment header.

Within blocks, the Oracle server keeps an identifier for each active transaction in the block header. At row level, the lock byte stores an identifier for the slot containing the transaction.

Example: In the diagram shown on the slide, the transaction using slot 1 is locking row 6 and the transaction in slot 2 is locking row 1.

DDL Locks

- **Exclusive DDL locks are required for:**
  - **DROP TABLE statements**
  - **ALTER TABLE statements**
  - **(The lock is released when the DDL statement completes.)**
- **Shared DDL locks are required for:**
  - **CREATE PROCEDURE statements**
  - **AUDIT statements**
  - **(The lock is released when the DDL parse completes.)**
- **Breakable parse locks are used for invalidating statements in the shared SQL area.**

DDL Locks

You are unlikely to see contention for DDL locks because they are held only briefly and are requested in `NOWAIT` mode. There are three types of DDL locks.

Exclusive DDL Locks

Some DDL statements, such as `CREATE`, `ALTER`, and `DROP` must get an exclusive lock on the object they are working on.

Users cannot get an exclusive lock on the table if any other user holds any level of lock, so an `ALTER TABLE` statement fails if there are users with uncommitted transactions on that table.

Example

Transaction 1	Transaction 2

`SQL> UPDATE employees`
`  2  SET salary=salary*1.1;`
`3120 rows updated.`
	`SQL> ALTER TABLE employees`
	`  2  DISABLE PRIMARY KEY;`
	`ORA-00054: resource busy and`
	`acquire with NOWAIT specified`

DDL Locks (continued)

Shared DDL Locks

Some statements, such as `GRANT` and `CREATE PACKAGE`, need a shared DDL lock on the objects they reference.

This type of lock does not prevent similar DDL statements or any DML statements but it prevents another user from altering or dropping the referenced object.

Breakable Parse Locks

A statement or PL/SQL object in the library cache holds one of these locks for every object it references, until the statement is aged out of the shared pool.

The breakable parse lock is there to check whether the statement should be invalidated if the object changes.

You could think of this lock as a pointer. It never causes waits or contention. However, this does impact the system in that when a breakable parse lock on an object is broken any objects, such as cursors and procedures, that reference that object will require parsing again. This could cause potential contention on the shared pool.

Possible Causes of Lock Contention

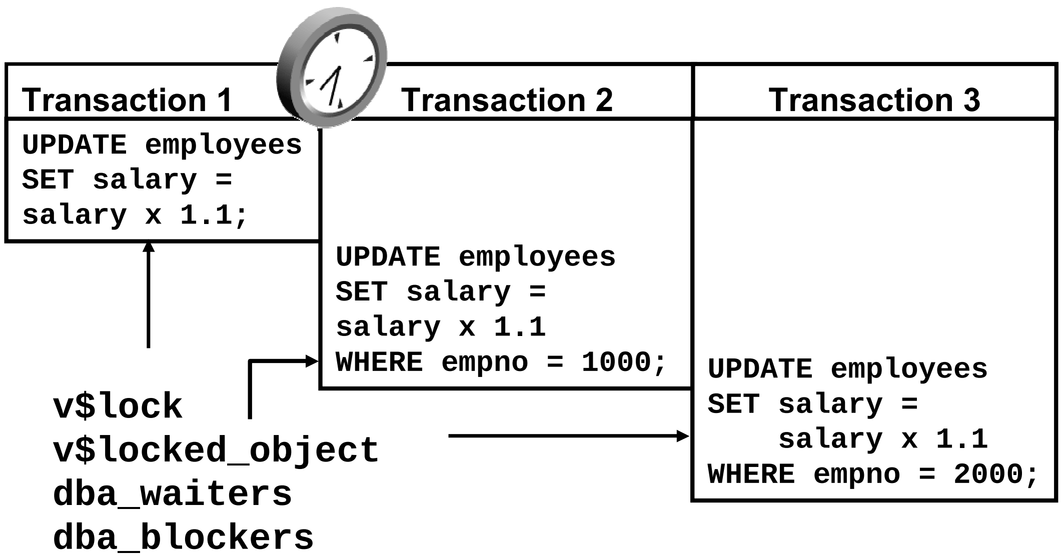
- Unnecessarily high locking levels
- Long-running transactions
- Uncommitted changes
- Other products imposing higher-level locks

Development and User Issues

The Oracle server locks are inexpensive and efficient and most sites do not have problems with locking. If locks do cause contention, it is often because:

- Developers have coded in unnecessarily high locking levels
- Developers have coded in unnecessarily long transactions
- Users are not committing changes when they should
- The application uses the Oracle server in conjunction with other products that impose higher locking levels

Diagnostic Tools for Monitoring Locking Activity



Diagnostic Tools for Monitoring Locking Activity

`dba_waiters` and `dba_blockers`

These views give further insight into who is holding or waiting for which tables. To create these views run the `catblock.sql` script. On UNIX systems this is found in the `$ORACLE_HOME/rdbms/admin` directory.

The `v$lock` View

Two of the columns in this view are `type` and `id1`. These columns have the values:

Lock type	ID1
TX	Rollback segment number and slot number
TM	Object ID of the table being modified

Any process that is blocking others is likely to be holding a lock obtained by a user application. The locks acquired by user applications are:

- Table locks (TM)
- Row-level locks (TX)

To find the table name that corresponds to a particular resource ID 1 of the `v$lock` view:

```
SQL> SELECT object_name
  2  FROM dba_objects, v$lock
  3  WHERE object_id=id1 AND type='TM';
```

Diagnostic Tools for Monitoring Locking Activity (continued)

The v$locked_object View

The columns of this view are:

- XIDUSN: Rollback segment number
- OBJECT_ID: ID of the object being modified
- SESSION_ID: ID of the session locking the object
- ORACLE_USERNAME
- LOCKED_MODE

Example

To find the table name that corresponds to a particular object ID in the v$locked_object view:

```
SQL> SELECT xidusn, object_id, session_id, locked_mode
  2  FROM v$locked_object;

   XIDUSN OBJECT_ID SESSION_ID LOCKED_MODE
--------- --------- ---------- -----------
        3      2711          9           3
        0      2711          7           3


SQL> SELECT object_name FROM dba_objects
  2  WHERE object_id = 2711;

OBJECT_NAME
-------------
EMPLOYEES
```

If the value of xidusn is 0, then the session with the corresponding session ID is requesting and waiting for the lock being held by the session, for which xidusn value is different from 0.

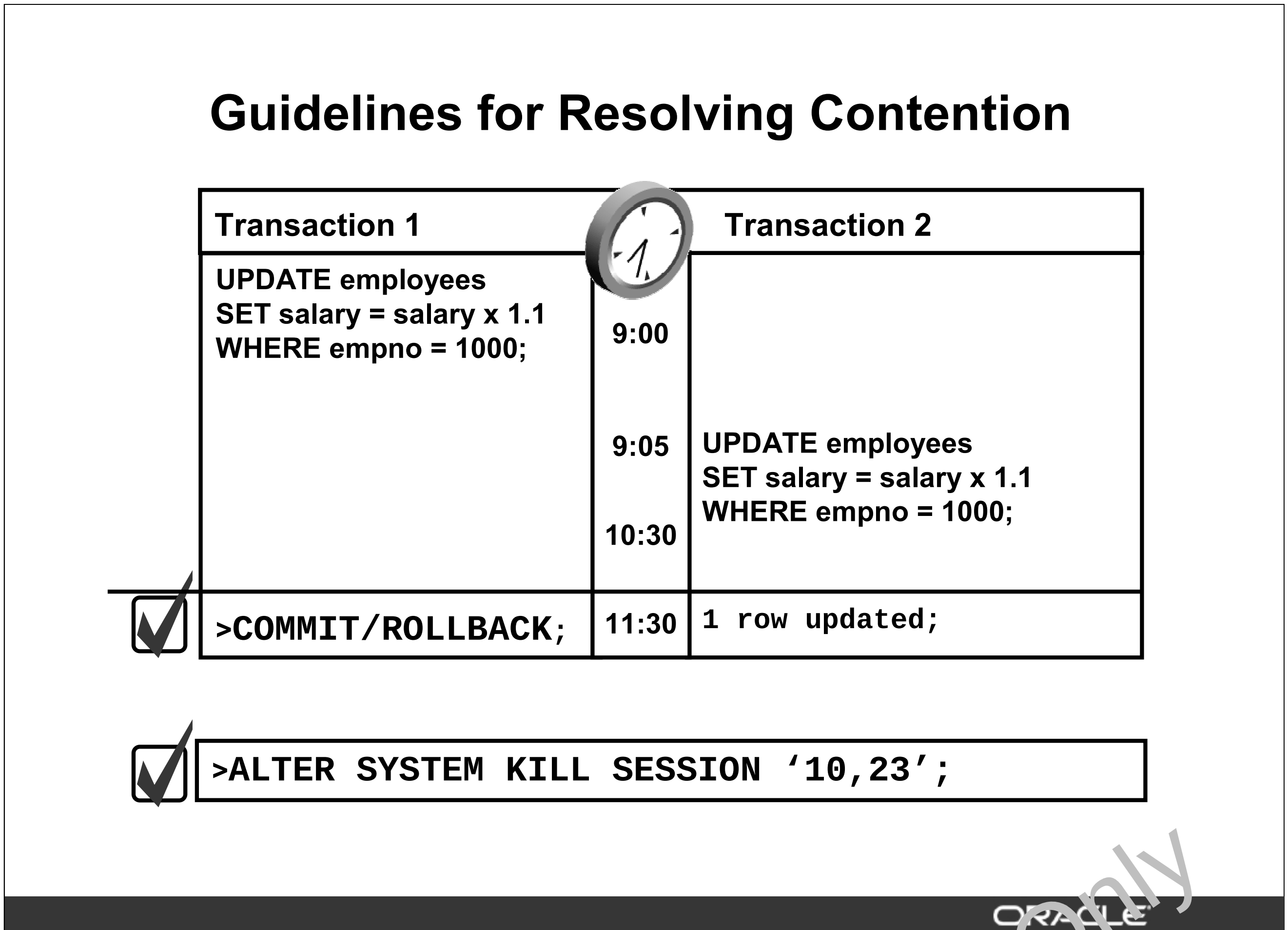
The utllockt.sql Script

You can also use the utllockt.sql script to display lock wait-for in a hierarchy. The script prints the sessions that are waiting for locks and the sessions that are blocking.

You must run the catblock.sql script (found in $ORACLE_HOME/rdbms/admin folder) as a sysdba user before using utllockt.sql. The catblock.sql script creates the dba_locks and dba_blockers views along with others that will be used by utllockt.sql.

For example, in the following output session 9 is waiting for session 8, sessions 7 and 10 are waiting for 9.

```
WAITING TYPE MODE          MODE             LOCK  LOCK
SESSION      REQUESTED     HELD             ID1   ID2
------- ---- ------------- -------------    ----- -----
      8 NONE None          None                 0   0
      9 TX   Share (S)     Exclusive (X)     65547  16
      7 RW   Exclusive (X) S/Row-X (SSX)   33554440  2
     10 RW   Exclusive (X) S/Row-X (SSX)   33554440  2
```



Guidelines for Resolving Contention

Killing Sessions

If a user is holding a lock required by another user, you can:

- Contact the holder and ask this user to commit or roll back the transaction
- As a last resort, kill the Oracle user session; this rolls back the transaction and releases locks

Any of the monitoring methods detailed previously will give you the session identifier for the user.

You can kill user sessions with the `ALTER SYSTEM KILL SESSION` command:

```
SQL> SELECT sid, serial#, username
  2  FROM v$session
  3  WHERE type='USER';

  SID   SERIAL#   USERNAME
----- ---------   --------
    8       122   SYSTEM
   10        23   SCOTT

SQL> ALTER SYSTEM KILL SESSION '10,23';
System altered.
```

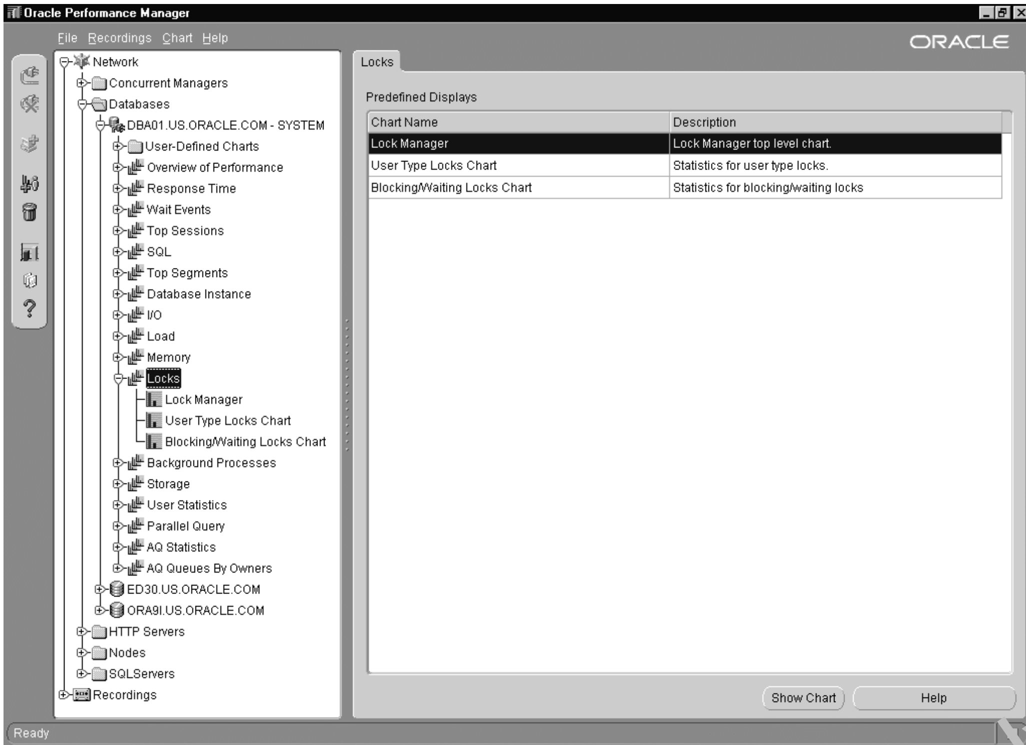
Guidelines for Resolving Contention (continued)

Which Row Is Causing Contention?

If you need to know which row is causing contention, the `v$session` view contains the following columns:

- `row_wait_block#`
- `row_wait_row#`
- `row_wait_file#`
- `row_wait_obj#`

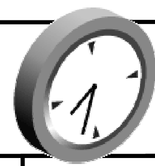
Performance Manager: Locks



Performance Manager: Locks

The Performance Manager has a set of charts labeled Locks. This set of charts can be used to determine which locks are causing other users to wait. These are termed “blocking locks” and must be resolved before the waiting transaction can proceed.

Deadlocks



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

Deadlocks

A deadlock can arise when two or more users wait for data locked by each other.

The Oracle server automatically detects and resolves deadlocks by rolling back the statement that detected the deadlock.

Transaction 1	Time	Transaction 2

`SQL> UPDATE employees`
`  2  SET salary=salary*1.1`
`  3  WHERE id= 24877;`
`1 row updated.`
	1	`SQL> UPDATE employees`
		`  2  SET salary=salary*1.1`
		`  3  WHERE id= 24876;`
		`1 row updated.`

`SQL> UPDATE employees`
`  2  SET salary=salary*1.1`
`  3  WHERE id= 24876;`
Transaction 1 waits.
	2	`SQL> UPDATE employees`
		`  2  SET salary=salary*1.1`
		`  3  WHERE id= 24877;`
		Transaction 2 waits.

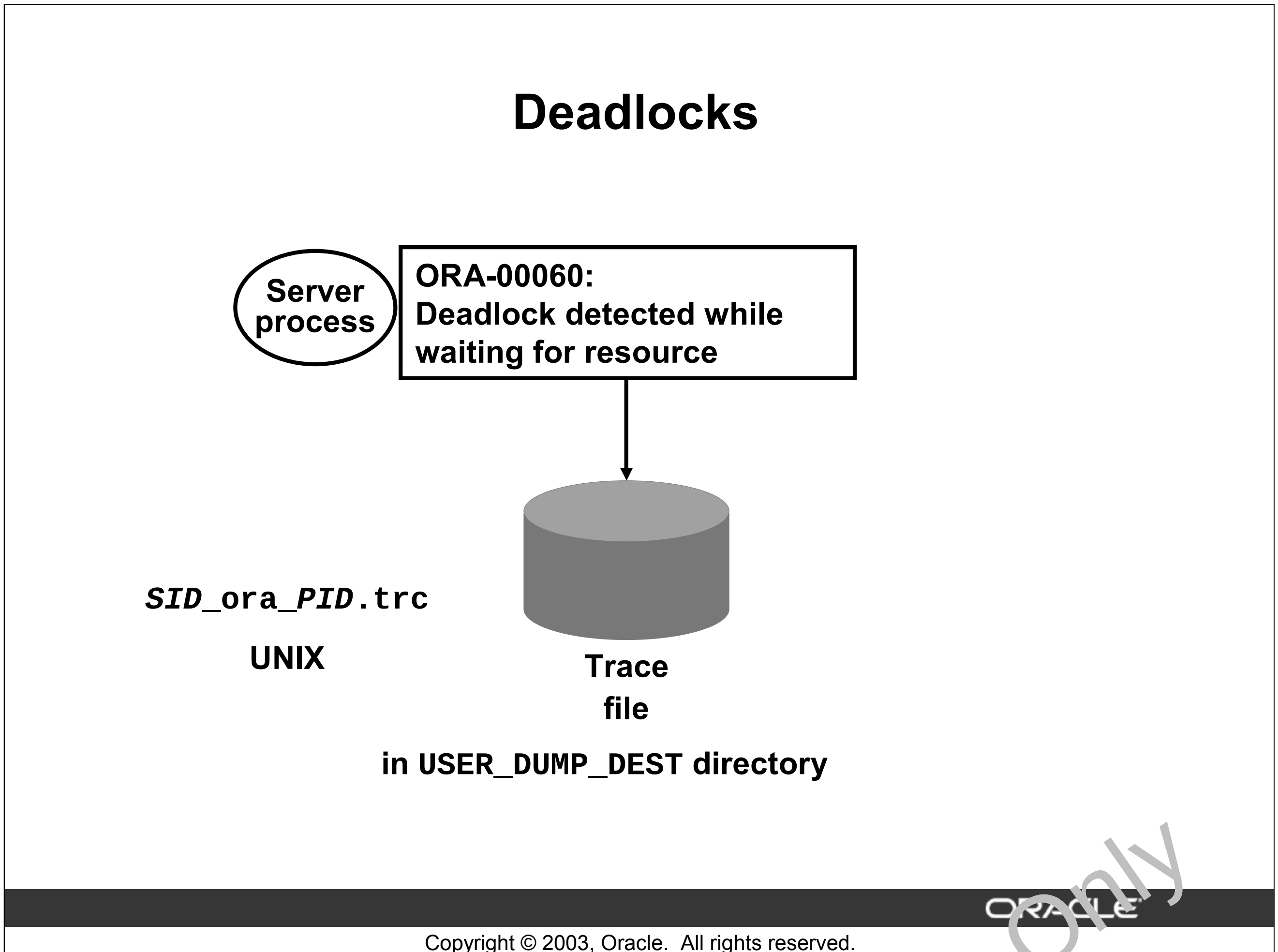
`ORA-00060: deadlock detected while waiting for resource`	3	

Deadlocks (continued)

If the second update in Transaction 1 detects the deadlock, the Oracle server rolls back that statement and returns the message. Although the statement that caused the deadlock is rolled back, the transaction is not, and you receive an `ORA-00060` error. Your next action should be to roll back the remainder of the transaction.

Technical Note

Deadlocks most often occur when transactions explicitly override the default locking of the Oracle server. Distributed deadlocks are handled in the same way as nondistributed deadlocks.



Trace File

A deadlock situation is recorded in a trace file in the `USER_DUMP_DEST` directory. It is advisable to monitor trace files for deadlock errors to determine whether there are problems with the application. The trace file contains the `rowids` of the locking rows.

In distributed transactions, local deadlocks are detected by analyzing a "waits for" graph and global deadlocks are detected by a time-out.

When detected, nondistributed and distributed deadlocks are handled by the database and application in the same way.

Summary

In this lesson, you should have learned to do the following:

- Define levels of locking
- Identify causes of contention
- Prevent locking problems
- Use Oracle utilities to detect lock contention
- Resolve contention in an emergency
- Resolve deadlock conditions

Practice 17

The objective of this practice is to use available diagnostic tools to monitor lock contention. You will need to start three sessions in separate windows. Log in as `hr/hr` in two separate sessions (sessions 1 and 3) and as `sys/oracle as sysdba` in another session (session 2). Throughout this practice Oracle Enterprise Manager can be used if desired. SQL Worksheet can be used instead of SQL*Plus and there are many uses for the Oracle Enterprise Manager console. (Solutions for Oracle Enterprise Manager can be found in Appendix B).

1. In session 1 (user `hr/hr`), update the salary by 10% for all employees with a salary < 15000 in the `temp_emps` table. Do not `COMMIT`.

2. In session 2 connect as `sys/oracle AS sysdba` and check to see whether any locks are being held by querying the `v$lock` view.

3. In session 3 (the session not yet used), connect as `hr/hr` and drop the `temp_emps` table. Does it work?

4. In session 3 (`hr/hr`), update the salary by 5% for all employees with a salary > 15000 in the `temp_emps` table. Do not `COMMIT`.

5. In session 2, check to see what kind of locks are being held on the `temp_emps` table, using the `v$lock` view.

6. In session 3, roll back the changes you made and set the `manager_id` column to 10 for all employees who have a salary < 15000.

   Note: This session will be hanging, so do not wait for the statement to complete.

7. In session 2, check to see what kind of locks are being held on the `temp_emps` table, using the `v$lock` view.

8. In session 2, run the `$ORACLE_HOME/rdbms/admin/catblock.sql` script. The script will create the `dba_waiters` view, which gives information regarding sessions holding or waiting on a lock. Use this view to determine the session ID for the session that is holding locks. Use this value to query `v$session` to obtain the serial number for the session holding the lock. Then run the `alter system kill session` command to release the session holding the lock.

Practice 17 (continued)

Lock Matrix

Type of Request	Lock Mode	Lock Target	Conflicts/Notes
Initialization parameters	None	None	No locks on reads
Lock table in Row Share mode	Mode 2	TM(RS) lock on table	Mode 6, so no exclusive DDL (this is the least restrictive lock.)
Lock table partition in Row Share mode	Mode 2 Mode 2	TM (RS)lock on table TM (RS) lock on table partition	Mode 6, so no exclusive DDL (This is the least restrictive lock.
Select for update	Mode 2 Mode 2 Mode 6	TM (RS) lock on table TM (RS) lock on each table partition TX lock on RBX TX slo	Mode 6 and any selects for update or DML on same rows No exclusive DDL
Lock table in Row Exclusive mode	Mode 3	TM (RX) lock on table	Modes 4, 5, 6 (updates allowed, because mode 3 does not conflict with mode 3.) No share locks and no referential integrity locks
Lock table partition in Row Exclusiving mode	Mode 3 Mode 3	TM (RX) lock on table TM (RX) lock on table partition	Modes 4, 5, 6 on the same partition Updates allowed, because mode 3 does not conflict with mode 3 No share locks and no referential integrity locks
DML (up/ins/del)	Mode 3 Mode 6	TM (RX) lock on table TX lock on RBS TX slot	Modes 4, 5, 6 Select for update or DML on same rows No share locks and no referential integrity locks
DML (up/ins/del) on a partioned table	Mode 3 Mode 3 Mode 6	TM (RX) lock on table TM (RX) lock on each table partition owning the updated rows TX lock on RBS TX slot	Modes 4, 5, 6 Select for update or DML on same rows No share locks and no referential integrity locks

Practice 17 (continued)

Lock Matrix (continued)

Type of Request	Lock Mode	Lock Target	Conflicts/Notes
Lock table in Share mode	Mode 4	TM (S) lock on table	Modes 3, 5, 6 Allows Select for Update and other Share Locks No possible ORA 1555 error on locked table
Lock table partition in Share mode	Mode 2 Mode 4	TM(RS) lock on table TM(S) lock on table partition	Mode 3,5,6 on the same partition Allows Select for Update and other Share locks No possible ORA 1555 on locked table
Lock table in Share Row Exclusive mode	Mode 5	TM(SRX) lock on table	Mode 3,4,5,6 Allows Select for Update only No Share locks No ORA 1555 No cascaded deletes
Lock table in partition in Share Row Exclusive mod	Mode 2 Mode 4	TM(RS) lock on table TM(S) lock on table partition	Mode 4 on the same partition Mode 3,5,6 on any partition Allows Select for Update only No ORA 1555 No cascaded deletes
Lock table in Exclusive mode	Mode 6	TM(X) lock on table	Mode 2,3,4,5,6 Selects only; no DDL Most restrictive lock mode
Lock table partition in Exclusive mode	Mode 3 Mode 6	TM(X) lock on table TM(X) lock on table partition	Mode 2,3,4,5,6 on the same partition Mode 5 on any partition No exclusive DDL Most restrictive lock mode on partition

Practice 17 (continued)

Lock Matrix (continued)

Type of Request	Lock Mode	Lock Target	Conflicts/Notes
Drop, Truncate, Create Table and Create Index DDL	Mode 6 No wait	TM(X) lock on table	Mode 2,3,4,5,6 Selects only; No DDL DDL fails if any other lock mode on table due to no wait
Drop, Truncate, ADD Partition DDL	Mode 3 Mode 6 No wait	TM(X) lock on table TM(X) lock on table partition	Mode 2,3,4,5,6 on the same partition Mode 5 on any partition DDL fails if any other lock mode on table partition due to no wait

18

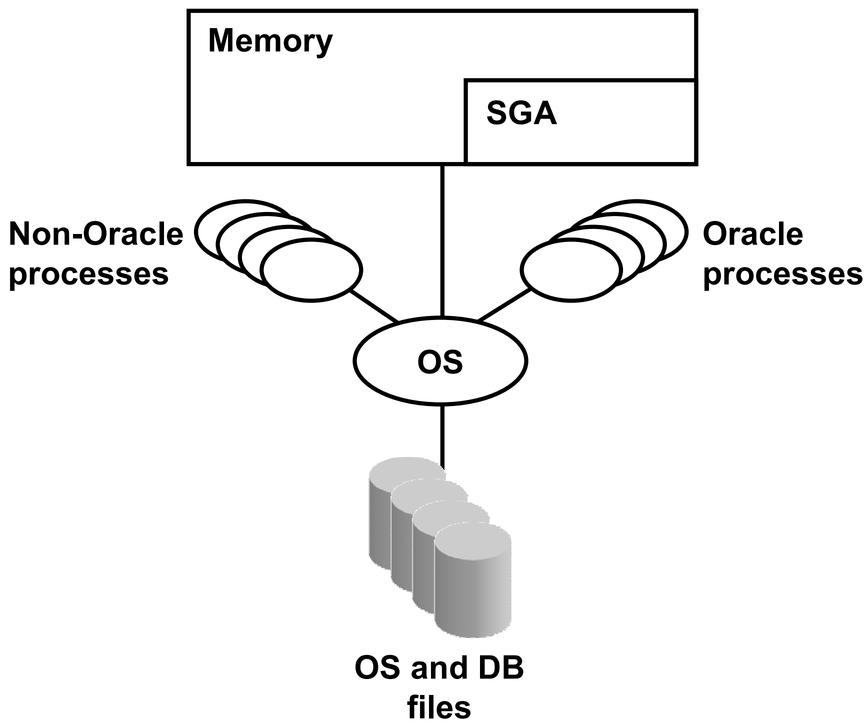
Tuning the Operating System

Objectives

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

- Describe different system architectures
- Describe the primary steps of OS tuning
- Identify similarities between OS and DB tuning
- Understand virtual memory and paging
- Explain the difference between a process and a thread

Operating System Tuning



Introduction to Operating System Tuning

The system administrator, the person responsible for tuning the operating system (OS), has tuning concerns similar to those of the database administrator. But the system administrator is also concerned with how applications other than Oracle applications affect performance. When tuning, the system administrator considers:

- Memory usage
- I/O levels
- CPU usage
- Network traffic

This lesson provides an overview of OS tuning, not specifics. This class focuses on OS tuning as it relates to the Oracle database rather than system performance tuning issues.

The operating system is tuned in a specific order because each area has its effect on the other underlying areas. If the memory usage is too high for example, an extra load is placed on the I/O layer, which in turn places an extra load on the CPU. The correct tuning order is:

1. Memory
2. I/O
3. CPU

System Architectures

The Oracle database can run on different system architectures. Some examples are:

- **Uni Processor systems**
- **Symmetric multiprocessing systems (SMP)**
- **Massively parallel processing systems (MPP)**
- **Clustered systems**
- **Nonuniform memory architecture systems (NUMA)**

System Architectures

Uni Processor Systems

Uni Processor systems have only one CPU and one memory.

Symmetric Multiprocessing (SMP) Systems

SMP systems have multiple CPUs. The number commonly ranges from two to 64. All of the CPUs in an SMP machine share the same memory, system bus, and I/O system. A single copy of the operating system controls all of the CPUs.

Massively Parallel Processing (MPP) Systems

MPP systems consist of several nodes connected together. Each node has its own CPU, memory, bus, disks, and I/O system. Each node runs its own copy of the operating system.

Clustered (Cluster) Systems

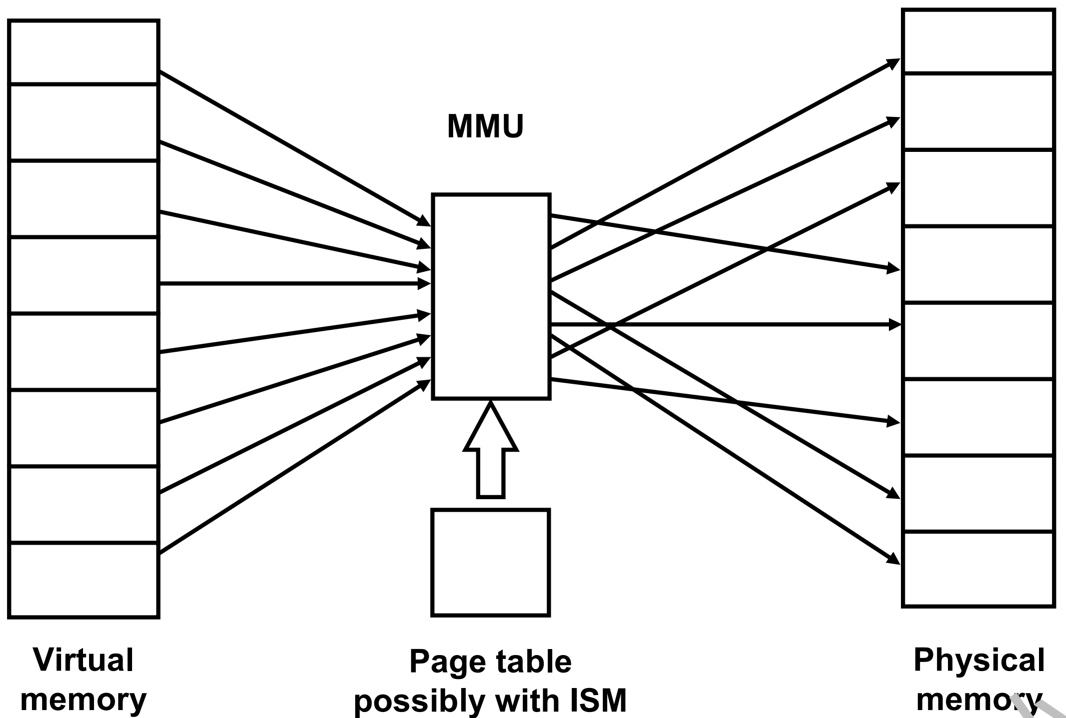
A cluster consists of several nodes loosely coupled using local area network (LAN) interconnection technology. Each of the individual nodes can contain one or more CPUs. In a cluster, system software balances the workload among the nodes and provides for high availability.

System Architectures (continued)

Nonuniform Memory Architecture (NUMA) Systems

NUMA systems consist of several SMP systems that are interconnected to form a larger system. In contrast to a clustered system all of the memory in all of the SMP systems are connected together to form a single large memory space transparent to all sub-systems. A single copy of the operating system runs across all the SMP systems.

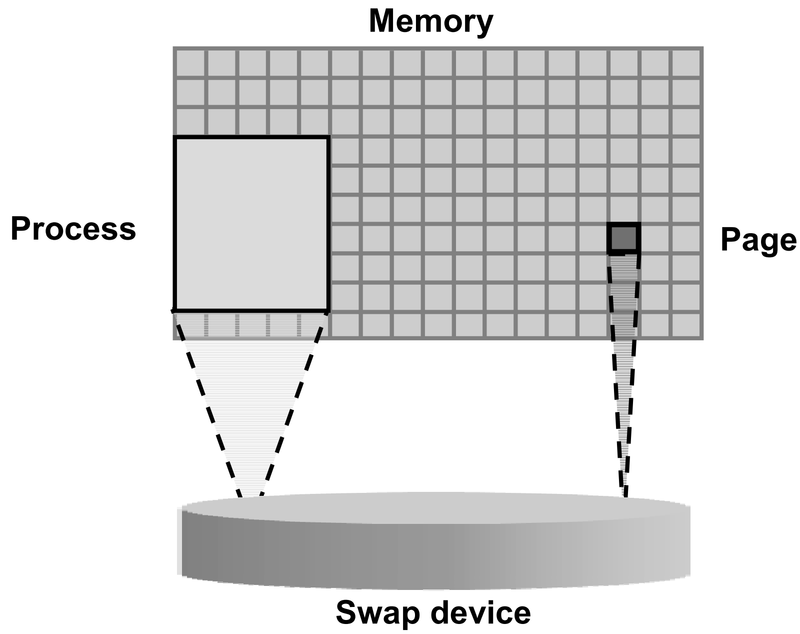
Virtual and Physical Memory



Virtual Memory

Operating systems make use of virtual memory. Virtual memory gives the application the feeling that it is the only application on the system. Each application sees a complete isolated memory area starting at address zero. This virtual memory area is divided into memory pages, which are usually 4 or 8 KB in size. The operating system maps these virtual memory pages into physical memory by the use of a memory management unit (MMU). The mapping between virtual and physical memory is under the control of a page table. On most operating systems, each process has its own page table. This can cause memory wastage if many processes need to access a very large area of shared memory. On some platforms, Solaris for example, this memory wastage can be avoided by sharing the page table entries for a shared memory area. This is called intimate shared memory (ISM). An additional benefit of using ISM is that the shared memory area gets locked into physical memory.

Paging and Swapping



Paging and Swapping

Operating systems use the same technique to manage memory as the Oracle database: keep the most recently used pages in real memory. Inadequate memory resources cause too much paging or swapping to occur. This symptom is often called thrashing, because it causes blocks to be transferred back and forth (thrashed) between memory and disk.

Paging occurs when a process needs a page (block) of memory that is no longer in real memory but in virtual memory. The block must be read (paged) in from the disk and the block in memory that it replaces may also need to be written to disk. Swapping is similar to paging, except that the memory space of the entire process is removed from memory. If there are too many processes running at a time, swapping may increase to an unacceptable level.

Both swapping and paging require adequate disk space to temporarily hold the memory blocks on disk. These files are I/O intensive, so they also need to be considered when balancing I/O. Some operating systems, such as Microsoft Windows, do not use swapping but only paging. Most other operating systems use swapping only as a last resort when the amount of free memory is getting unacceptably low.

Tuning Memory

- Database tuning can improve paging performance by locking SGA into real memory.
- The DBA should monitor real and virtual memory use.
- The DBA should use intimate shared memory, if it is available.

Tuning Memory

DB Tuning and Its Effects on Paging

Besides tuning the SGA, the DBA can also affect paging and swapping performance in another way.

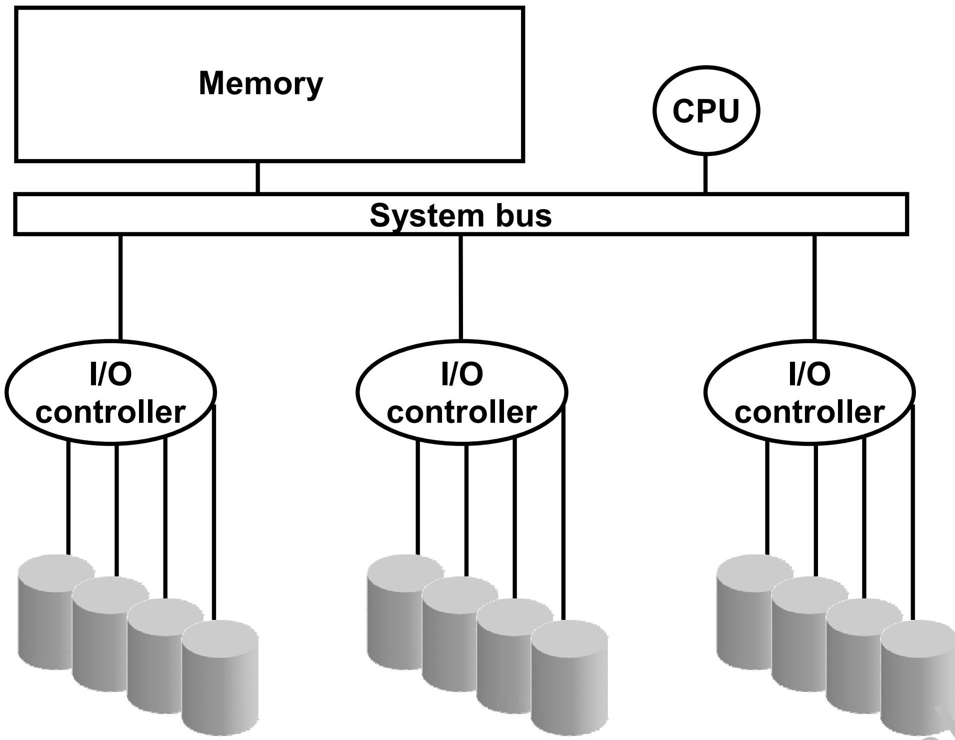
On some operating systems, the DBA can lock the SGA into real memory by setting the `LOCK_SGA` initialization parameter to True, so it is never paged out to disk. Obviously, the Oracle server performs better if the entire SGA is kept in real memory.

This should be used only on systems that have sufficient memory to hold all the SGA pages without degrading performance in other areas.

Monitor Memory Usage

Real and virtual memory usage and paging and swapping can usually be monitored by process or for the entire operating system. The amount of paging and swapping that is acceptable varies by operating system; some tolerate more than others.

Tuning I/O



Tuning I/O

The system administrator improves the performance of disk I/O by balancing the load across disks and disk controllers.

I/O-intensive systems, such as database servers, perform better with many small disks instead of a few large disks. More disks reduce the likelihood that a disk becomes a bottleneck. Parallel Query operations also benefit by distributing the I/O workload over multiple disk drives.

Raw Devices

A raw device is a disk or disk partition without a file or directory structure. They are more difficult to administer than operating system files.

Monitoring

I/O performance statistics usually include the number of reads and writes, reads and writes per second, and I/O request queue lengths. Acceptable loads vary by device and controller.

Understanding Different I/O System Calls

Operating systems can perform disk I/O in two different ways:

- **Normal (blocking) I/O**
- **Asynchronous (nonblocking) I/O is implemented on most platforms and file systems**

Understanding Different I/O System Calls

The Normal (or Blocking) I/O System Call

When the Oracle database issues an I/O request (read or write) using a normal (or blocking) I/O system call, it has to wait until the I/O operation has completed. This limits the amount of I/O that can be performed in a certain amount of time.

The Asynchronous (or Nonblocking) I/O System Call

When the Oracle database issues an I/O request (read or write) using an asynchronous (or non blocking) I/O system call, processing can continue with no waiting for the I/O operation to complete. This allows many I/O requests to be issued at the same time. By using asynchronous I/O the Oracle database can overlap several I/O requests. After the operating system completes an asynchronous I/O, the operating system notifies the Oracle database that the I/O request has been completed along with the status of this particular I/O request.

Asynchronous I/O on File Systems

Although asynchronous I/O on file systems is supported on most UNIX implementations, it is usually implemented using the user-level multithreading capabilities in the operating system. Therefore, it induces a significant CPU overhead. To avoid this CPU overhead it is preferred to use multiple database writers or use database writer I/O slaves rather than asynchronous I/O.

CPU Tuning

- **Guidelines:**
  - **Maximum CPU busy rate: 90%**
  - **Maximum OS/User processing ratio: 40/60**
  - **CPU load balanced across CPUs**
- **Monitoring:**
  - **CPU**
  - **Process**

CPU Tuning Guidelines

When tuning CPU usage, the system administrator has the following primary concerns:

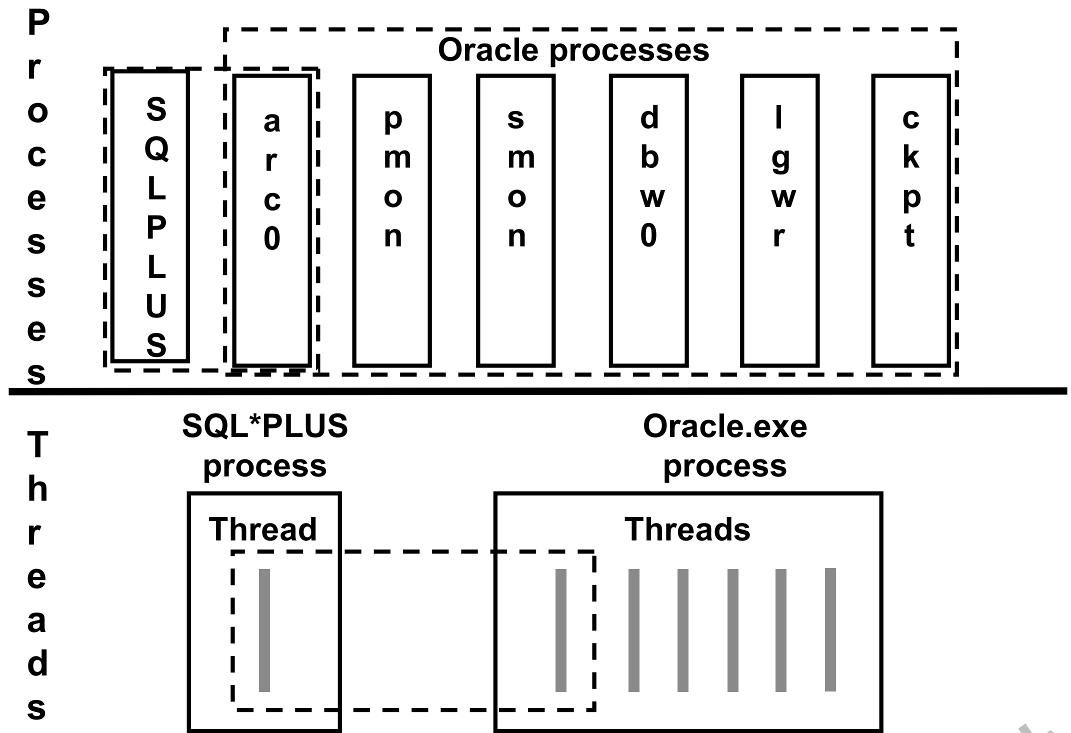
- Are there adequate CPU resources? The system administrator ensures that the CPU is not too busy. As a general rule, if the CPU is busy 90% of the time, it has probably reached its capacity.
- Is there a good ratio between operating system processing and application processing? Operating system processing includes the tasks that the operating system performs for the applications; for example, reading and writing to devices, sending messages between processes, and scheduling processes.
- The goal is to have the CPU working mostly on the application and least on operating system related tasks. Too much time spent in the operating system mode is a symptom of an underlying problem, such as:
  - Insufficient memory, which also causes swapping or paging
  - Poor application design, causing too many operating system calls
- For multiprocessing systems, the system administrator must also check that the CPU load is balanced across all of the CPUs, particularly if any of the CPUs have a very high usage rate.

CPU Tuning Guidelines (continued)

CPU Monitoring

Operating system monitors for CPU usage normally include the percentage of time the CPU is active and the time spent in operating system versus user tasks. Process monitors show the process status and statistics on the number of I/Os, operating system calls, and paging and swapping rates.

Process versus Thread



Processes and Threads

On most operating systems, each Oracle process is also an operating system process. However under some operating systems, notably Microsoft Windows, the Oracle processors are implemented as a single operating system process with multiple threads. In Microsoft Windows the Oracle process threads share the memory allocated to the process.

Each Oracle process is a thread within the operating system process. A thread is a sequence of instructions that can execute independently of other threads in the same process. This configuration makes SGA access and communication among Oracle processes more efficient at the expense of having a limit on the maximum process memory.

Summary

In this lesson, you should have learned how to:

- Describe different system architectures
- Describe the primary steps of OS tuning
- Identify similarities between OS and DB tuning
- Understand virtual memory and paging
- Explain the difference between a process and a thread

19

Workshop Overview

Objectives

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

- Follow the Oracle tuning methodology to diagnose and resolve performance problems
- Improve your tuning skills
- Use Oracle tools to diagnose performance problems

Approach to Workshop

The workshop is intended to provide:

- A group-oriented and interactive experience
- Intensive hands-on diagnosis and problem resolution
- Proactive participant involvement

Approach to Workshop

Your instructor may assign you to a group so that you can work together with your classmates to perform tuning diagnostics and resolution. During the workshop, you are encouraged to share what you have learned from the first part of this course, as well as your experiences with your group members to help diagnose and solve various tuning problems. Each group is encouraged to share its tuning diagnostic and resolution approach with other groups in the class.

You are assigned a database that is isolated from the other groups in the class to avoid contention that could distort your results. In some cases, your group may have two or more such databases with which you can work, allowing you to run different scenarios or to test different possible solutions concurrently. Your goal during this workshop is to obtain as much hands-on experience as possible as you diagnose problems and work through the performance tuning methodology, diagnosis, and resolution steps. The experience and knowledge that you had gained from the first four days of this course should play a major role in your successful completion of the workshop steps.

At certain times, your instructor may ask you, or someone in your group, to present a summary of your investigations, analysis, tuning steps, and conclusions for one or more of the scenarios on which you worked.

Workshop Background

The workshop is based on the XYZ Company, a fictional entity that has the following characteristics:

- **XYZ Company is new and still small:**
  - **Shares a database server with other companies**
  - **Currently has four employees who use the database**
- **System was set up by a part-time DBA trainee**
- **Database performance is unacceptable**
- **XYZ Company is seeking help from a consulting group**
- **Number of database users will be increasing**

Workshop Background

A fictional company, XYZ Company, provides the basis for the workshop scenarios. XYZ Company shares computing resources with other companies and is unwilling to upgrade or add new components to improve the hardware environment. One consequence of this is that the company has imposed a maximum limit of 100 MB on the size of the SGA for its single Oracle database.

A trainee DBA has designed and built the database for XYZ Company. Although it works, its performance is not acceptable. The four database users, two OLTP application end users and two DSS analysts who execute extensive queries, are complaining about response times. Meanwhile, XYZ Company is expanding and expects to have 20 concurrent database users in the near future, 10 performing OLTP work and 10 running a DSS application.

You, and your group partners, assume the role of a consulting team that is hired by XYZ Company. Your task is to resolve the current performance problems and also prepare the database for an increase in the number of database users. You must do this while staying within the SGA size restraint. For example, although setting the shared pool size to 500 MB might be a simple solution for some problems, it is not within the restriction imposed by management to keep the total SGA size below 100 MB.

Workshop Outline

You are provided with scripts and tools to:

- **Configure your database with a tuning problem based on a selected scenario**
- **Execute a simulated workload against your detuned database**
- **Collect performance statistics for your database**
- **Analyze your data and make changes to improve your database performance**
- **Confirm that your changes have been beneficial by executing the same or a more intense workload simulation**

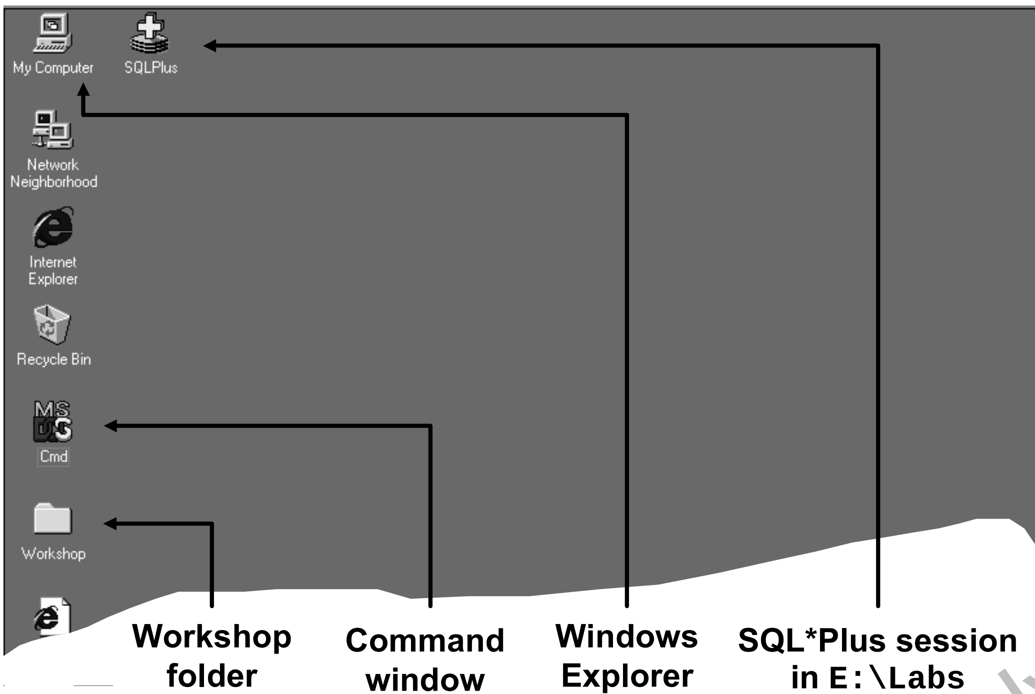
Workshop Outline

The current performance problems, and challenges for XYZ Company's planned increase in concurrent database users, are represented by six scenarios. Five of these concentrate on one element of the database that requires tuning and the sixth scenario presents various tuning problems for you to solve. To test your skills in specific tuning areas, you execute scripts that prepare your database by exaggerating the problems to be solved. Then, by executing additional scripts or running your own statements interactively, you collect statistics while a workload executes against the database, thereby simulating the activity of the current four users.

After collecting and analyzing the statistics, you implement whatever database changes your investigation determines would improve the situation. For example, you may set or change initialization parameters, create or modify database objects, and so on. With your changes in place, you rerun the simulated workload and collect new statistics to determine if your changes improved the situation. If there is no improvement, use the new statistics to refine your strategy and then test it again, repeating this process until performance improves.

When you achieve satisfactory database performance, you may continue your work on a scenario with a workload that simulates the anticipated 20 database users. Use the same data collection, collection, and tuning steps to prepare your database to support the larger user community effectively.

Workshop Configuration



Workshop Configuration

You run your workshop database, tools, and scripts on your client system only, not on the server that you used for the first part of this class. Your client desktop should contain various icons including one for the Workshop folder, as well as one for MS DOS command window (Cmd) and Windows Explorer (My Computer). Note that your desktop, as well as your Workshop folder includes a SQL*Plus icon, the former is rooted in `E:\Labs` and the latter in your Workshop directory. You may use either and do not need a host name string to connect.

Your client is configured with the following folders that you may need to access:

- `E:\orant\ora92\oradata\ORCL`: contains the database data files
- `E:\orant\ora92\admin\ORCL\bdump`: contains instance-specific trace files
- `E:\Labs\student\wkshop`: contains your workshop files

Your client also contains the Oracle database that you need for the workshop. The database is configured with:

- Five tablespaces: `system`, `undotbs1`, `temp`, `example`, and `oem_repository`
- Passwords for the `sys` and `system` users set to `oracle`
- The `hr`, `oe`, and `sh` sample schemas installed
- The `dbsnmp`, `outln`, `wmsys`, `oem_edcdr25p1`, `ordsys`, `ordplugins`, and `mdsys` default users.

Workshop Setup: Overview

Step 1: Start the setup script

Step 2: Respond to prompts

Step 3: Verify configuration

Step 4: Preserve your setup

Step 5: Open the workshop window

Workshop Setup: Overview

To prepare your database for the workshop tasks, you need to perform these activities, which you will find described in detail on the following pages:

Step 1: Execute the workshop setup script.

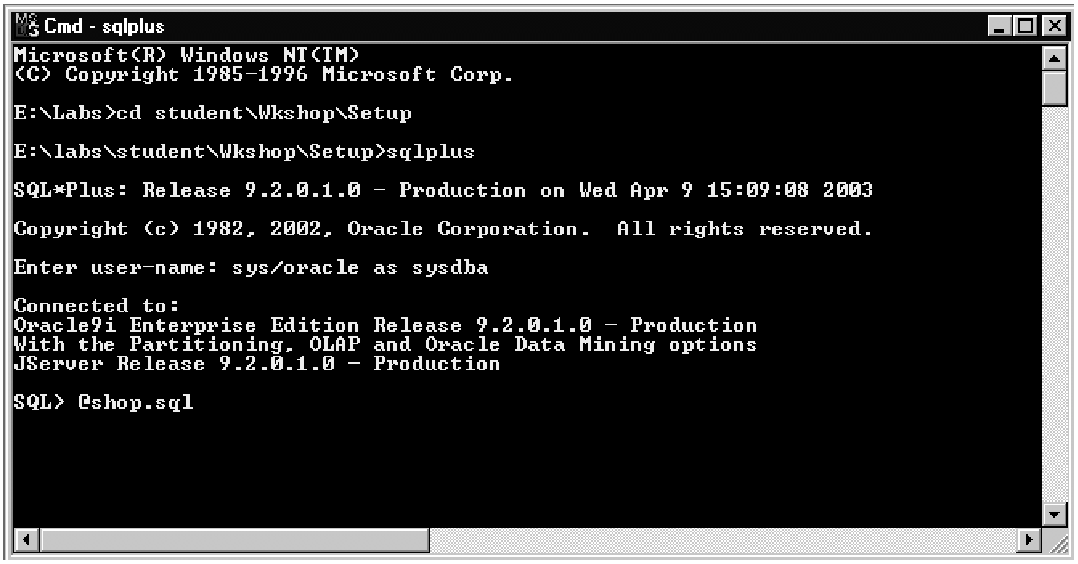
Step 2: Respond to the prompts that are generated while this script runs.

Step 3: Confirm that the database configuration is correct.

Step 4: (Optional) Make copies of your server parameter file (`SPFILE`) and sample schema tablespace (`EXAMPLE`) for later use, if needed.

Step 5: Open the workshop window to confirm that all the required scripts are available.

Workshop Setup: Step 1



Workshop Setup: Step 1, Start the Setup Script

You need to run a script, `shop.sql`, to prepare your database for the workshop. The script creates the required users, installs Statspack, and so on. To initiate the script, perform the following:

1. Open a Command window by double-clicking the MS DOS (Cmd) icon on your desktop.
2. Navigate to the Setup directory by running the command:
   `cd student\Wkshop\Setup`
3. Initiate a database session in your Command window by entering the command: `sqlplus`
4. Connect to the database with administrator privileges by responding to the Enter username prompt with: `sys/oracle as sysdba`
5. Initiate the `shop.sql` script by entering `@shop.sql` at the SQL*Plus prompt.

Your session should be similar to the one shown in the slide.

Workshop Setup: Step 2

```
... Creating PERFSTAT user ...

Choose the PERFSTAT user's password.

Not specifying a password will result in the installation FAILING

Specify PERFSTAT password
Enter value for perfstat_password: perfstat
```

...

```
Specify PERFSTAT user's default   tablespace
Enter value for default_tablespace: tools
```

...

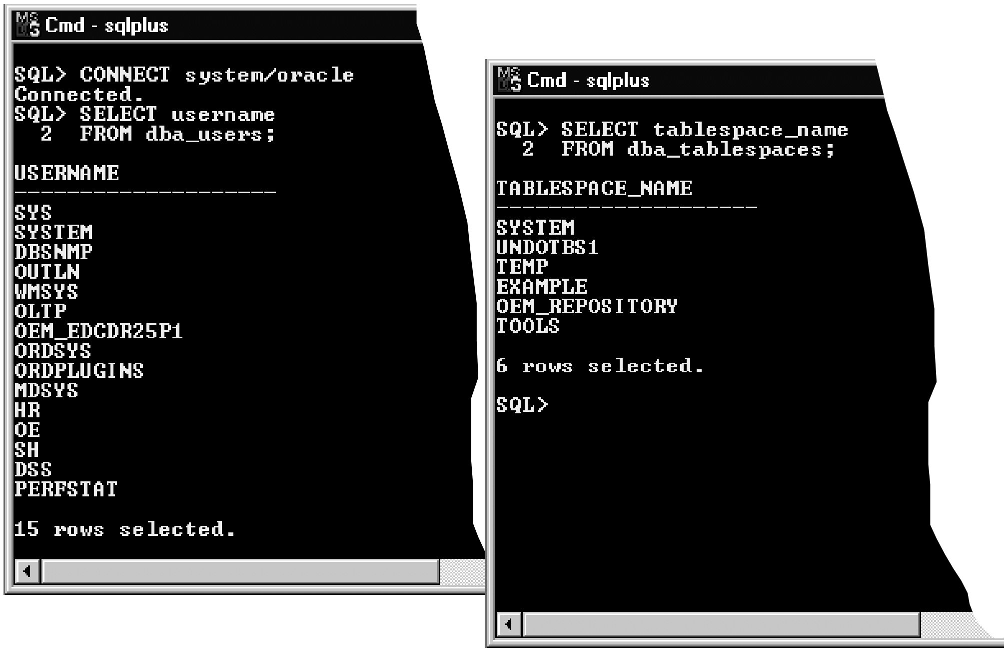
```
Choose the PERFSTAT user's temporary tablespace.

Specifying the SYSTEM tablespace will result in the installation
FAILING, as using SYSTEM for the temporary tablespace is not recommended.

Specify PERFSTAT user's temporary tablespace.
Enter value for temporary_tablespace: temp_
```


Workshop Setup: Step 2, Respond to Prompts

As the `shop.sql` script runs, it attempts to drop and recreate a number of database objects. Some of these objects may not exist, particularly if this is the first time you are running the script. Therefore, you can ignore any error messages that refer to the nonexistence of database objects. However, you will need to respond to prompts that are generated by the script as it creates the `perfstat` user. Use the responses that are shown in the graphic for the various prompts. The following is a list of the prompts and required responses:

```
Enter value for perfstat password: perfstat
. . .
Enter value for default_tablespace: tools
. . .
Enter value for temporary_tablespace: temp
```

Workshop Setup: Step 3

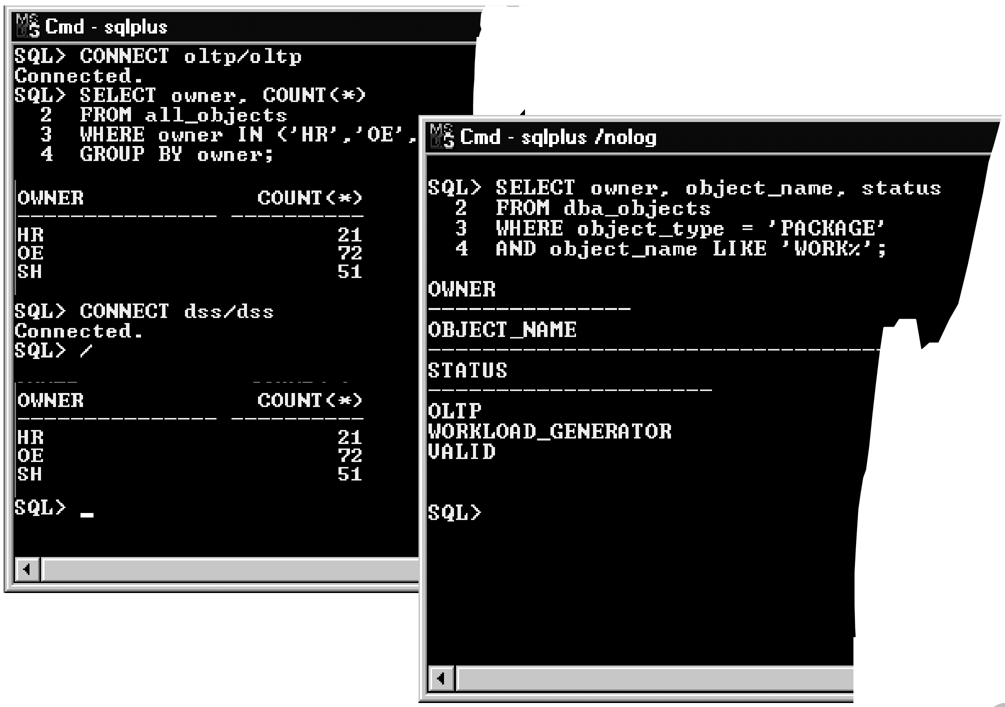


Workshop Setup: Step 3, Verify Configuration

In your SQL*Plus session, connect as `system` and confirm that your database contains the additional users, `oltp`, `dss`, and `perfstat`, as well as the new `tools` tablespace.

```
SQL> SELECT username
  2  FROM dba_users;

USERNAME
------------------------------
. . .
OLTP
DSS
PERFSTAT
. . .
SQL> SELECT tablespace_name
  2  FROM dba_tablespaces;

TABLESPACE_NAME
------------------------------
. . .
TOOLS
. . .
```

Workshop Setup: Step 3

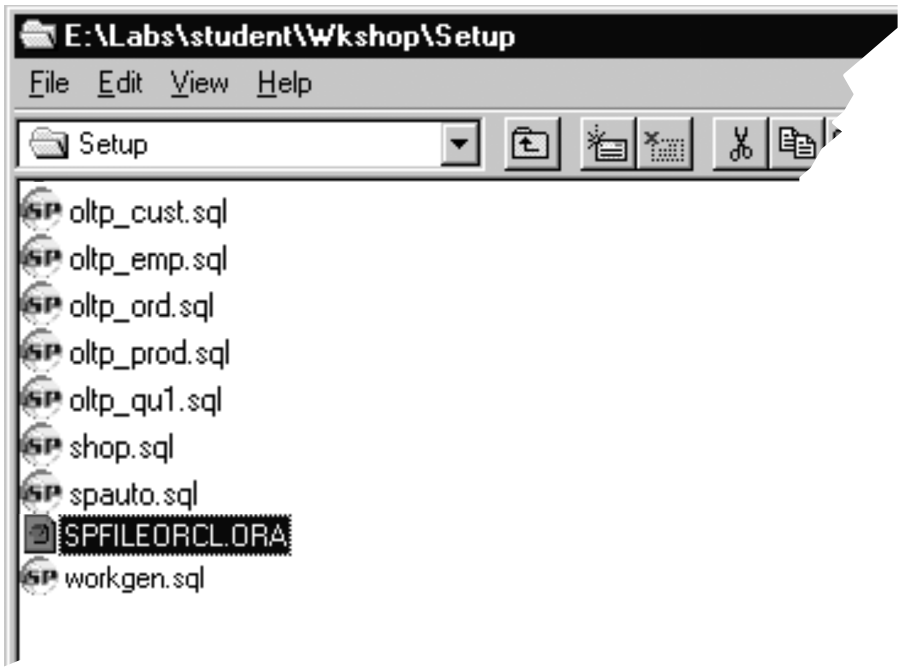


Workshop Setup: Step 3, Verify Configuration (continued)

Also validate that the `oltp` and `dss` users can access the objects in the sample schemas and that the `workload_generator` package is available with a valid status.

```
SQL> CONNECT oltp/oltp
Connected
SQL> SELECT owner, count(*)
  2  FROM all_objects
  3  WHERE owner IN ('HR','OE','SH')
  4  GROUP BY owner;

OWNER                 COUNT(*)
--------------- ----------
HR                      21
OE                      72
SH                      51
. . .
SQL> SELECT owner, object_name, status
  2  FROM dba_objects
  3  WHERE object_type = 'PACKAGE'
  4  AND object_name LIKE 'WORK%';

OWNER            OBJECT_NAME             STATUS
---------------- ----------------------- ------
OE               WORKLOAD_GENERATOR      VALID
```

Workshop Setup: Step 4



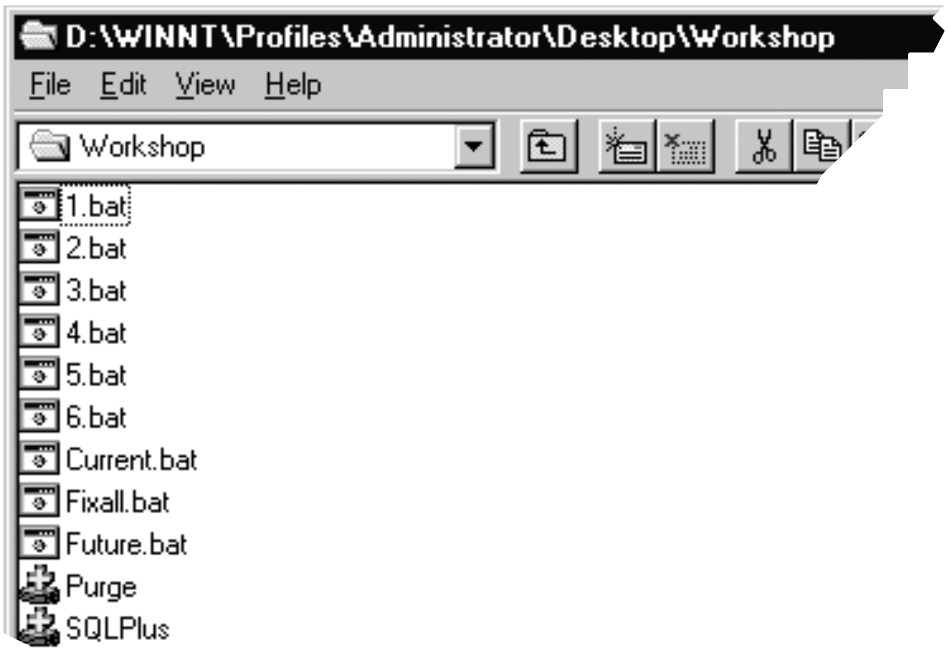
Workshop Setup: Step 4, Preserve Your Setup (Optional)

The workshop environment includes a script, `Fixall.bat`, to restore the database configuration when you have completed working on a particular scenario. This script runs automatically when you set up a new scenario and replaces the database from a backup, including a copy of the `SPFILE`, created by the `shop.sql` script. If you need to, then you can run `Fixall.bat` manually to restore your database while working on a scenario. The script resides in your Workshop folder and is available through an icon, as described in the next setup step.

If you want to control your database resources yourself, then you may want to make a database backup at this point. You can use your backup to restore your database to its current state should this be necessary while working on a scenario. To do this, shut down your database and make a cold backup. The database files that you need to copy are in the `E:\orant\ora92\oradata\ORCL` directory. If you decide to make a cold backup, then remember to restart your database when you are finished.

You may also want to make a backup of the original `SPFILE` used for this database. The Windows Explorer screenshot shows a copy of the `SPFILE`, made from the original in `E:\orant\ora92\database`, and located in the Setup subdirectory under your Workshop folder.

Workshop Setup: Step 5



Workshop Setup: Step 5, Open the Workshop Window

You may want to keep your Command window open during the remainder of this workshop or simply open a new window, if needed. However, you will need to open the Workshop folder by clicking the Workshop icon on your desktop to display the files required to run the workshop. These include:

- Setup files for each of six scenarios tuning scenarios
- Scripts to simulate a workload that is generated by the current 4 database users and by the additional 16 users, logging into the database as `oltp` and `dss`
- Utility scripts to help you run the workshop, including a connection to SQL*Plus that runs in your Workshop directory and shortcut to the `Fixall.bat` script that restores your database contents and settings

If you have any configuration problems, then you can rerun the `shop.sql` script while connected to your database as `sys`. If these actions do not solve your problems, then consult your instructor.

Note: Both of the SQL*Plus icons that are available to you will initiate a session on your local database and you do not need to provide a Host String in the dialog box. However, the icon in your Workshop folder is initiated from the Workshop folder, therefore, any unqualified file and script names use `E:\Labs\student\Wkshop` as the default directory. The icon on the desktop is rooted in `E:\Labs`.

Steps to Run a Scenario

1. Run the setup script for your selected scenario.
2. Confirm that the Statspack job is running.
3. Generate a workload for the current four users.
4. Query tables to gather statistics interactively.
5. Identify two snapshots and generate a Statspack report from them.
6. Analyze your data and determine what problems to address.
7. Make changes to improve performance.
8. Repeat these steps (except step 1) until you achieve acceptable performance.
9. Repeat steps 2 through 8 again, substituting the workload for the planned 20 users in step 2.
10. Prepare your class presentation when you are satisfied with your work.

Steps to Run a Scenario

Your instructor may assign you to work on specific scenarios or may let you pick your own. In either case, do not expect to complete every scenario, which is available, in the time allotted for this workshop. For each scenario selected, you will need to complete the steps that are listed.

Before you begin working with your first scenario, you may want to generate a baseline set of statistics. To do this, execute steps 2 through 5 and retain the Statspack report, which is generated in step 5, as a basis for comparison with the statistics generated when you work with different scenarios. The following pages provides more details about each of the workshop steps.

Run Scenario Setup Script

Scenario Description	Setup Icon
Shared pool performance	1.bat
Database buffer cache performance	2.bat
Redo log buffer performance	3.bat
Data access performance	4.bat
PGA performance	5.bat
Assorted performance areas	6.bat

Step 1: Run Scenario Setup Script

After you select a scenario, execute the setup script from your Workshop window by double-clicking its associated icon, which is listed in the table. The script first resets any parameters, which you had changed back, to their initial values. Finally, the script makes adjustments that detune the database according to the chosen scenario. While the script is running, you may see it shutting down and restarting your database at least twice.

If you have already worked on another scenario and you made a database backup, you may want to restore your data in the `Example` tablespace by restoring the database. Remember that this will effectively remove any segments that you purposely added during a previous analysis, so only restore the database when your work on the earlier scenario is complete. To restore your database, you must shut it down, copy the files from the backup location to replace the current database files, and restart your database. You must do this before running the script to set up your selected scenario.

Check Statspack Job

```
SQL> CONNECT perfstat/perfstat
```

```
SQL> SHOW PARAMETER job_queue_processes
```

```
SQL> SELECT job, log_user, what, next_date,
  2         next_sec, interval
  3  FROM user_jobs;
```


Step 2: Check Statspack Job

To be sure that your database executes a scheduled job to collect Statspack statistics during the workshop, you must connect as the `perfstat` user and check two items. First, ensure that there are at least four job processes running. Then, confirm that there is a snapshot job in the job queue. The required SQL commands are shown in the slide above, but they are available in a script called `spconfirm.sql` that is stored in your Workshop folder. Execute this script from a SQL*Plus session, if necessary, as follows:

```
SQL> @ E:\Labs\student\Wkshop\spconfirm.sql
```

The output should look similar to this:

```
NAME                    TYPE          VALUE
-------------------- ------------- --------------------
job_queue_processes  integer       4

JOB LOG_USER    WHAT            NEXT_DATE       NEXT_SEC
--- ---------- --------------- --------------- ----------
INTERVAL
------------------------------
  1 PERFSTAT   statspack.snap; 09-APR-03       22:54:00
trunc(SYSDATE+1/(24*6),'MI')
```

Check Statspack Job

```
SQL> @E:\Labs\student\Wkshop\Setup\spauto

PL/SQL procedure successfully completed.

Job number for automated statistics collection for this instance
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
. . .
     JOBNO
----------
         1

Job queue process
~~~~~~~~~~~~~~~~~
. . .
NAME                  TYPE          VALUE
--------------------- ------------- -------------
job_queue_processes   integer       4

Next scheduled run
~~~~~~~~~~~~~~~~~~
. . .
       JOB NEXT_DATE       NEXT_SEC
---------- --------------- ----------
         1 09-APR-03       23:00:00
```

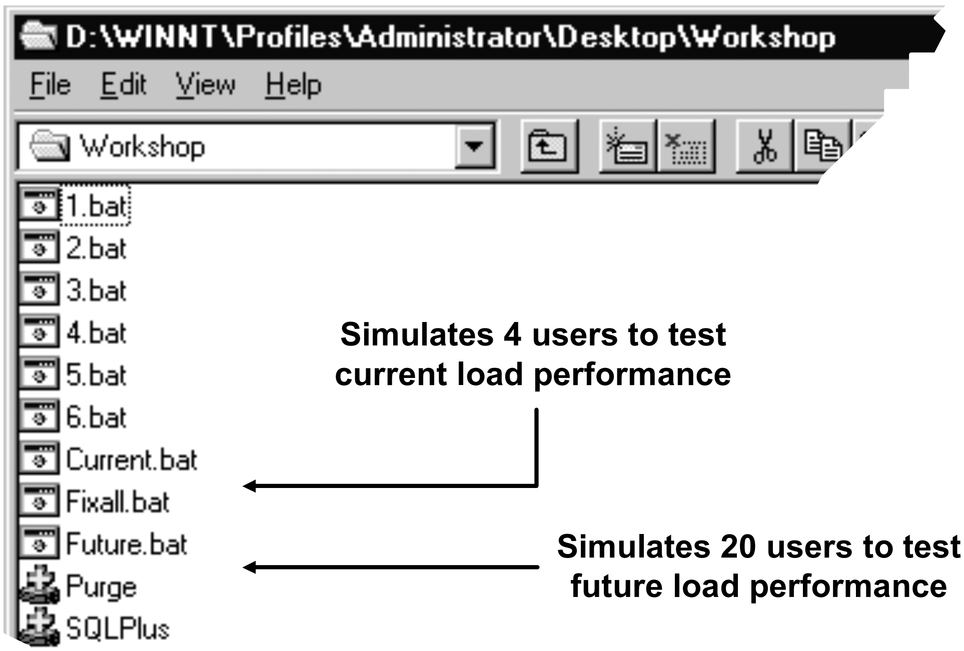

Step 2: Check Statspack Job (continued)

Check the results of your query against the `user_jobs` table, particularly the `what` and `interval` columns. The value in the `what` column should be `statspack.snap`, which is the job that collects snapshots of statistics for Statspack. This job should be scheduled to run every 10 minutes, that is, with the value `TRUNC(SYSDATE+1/(24*6),'MI')` in the `interval` column.

If you do not find a scheduled snapshot job, then connect as `perfstat/perfstat` and execute the `spauto.sql` script, which you can find in the same folder that you used to set up the workshop initially: `E:\Labs\student\Wkshop\Setup`. Then rerun your previous query to confirm that the job has started, and ask your instructor for help if it has not.

If you find the snapshot job running, then you may want to remove any existing statistics by double-clicking the Purge icon found in your Workshop folder window and entering the lowest and highest snapshot IDs when prompted. Clearing statistics is not a requirement for tuning. However, because of the snapshot frequency and the nature of this workshop, if you drop statistics before each iteration of the workshop steps, you will avoid confusion about which snapshots to use for your analysis. On a production system, you should collect snapshot statistics no more frequently than once an hour and you would want to keep a history of your snapshots for baselines and performance comparisons.

Generate a Workload



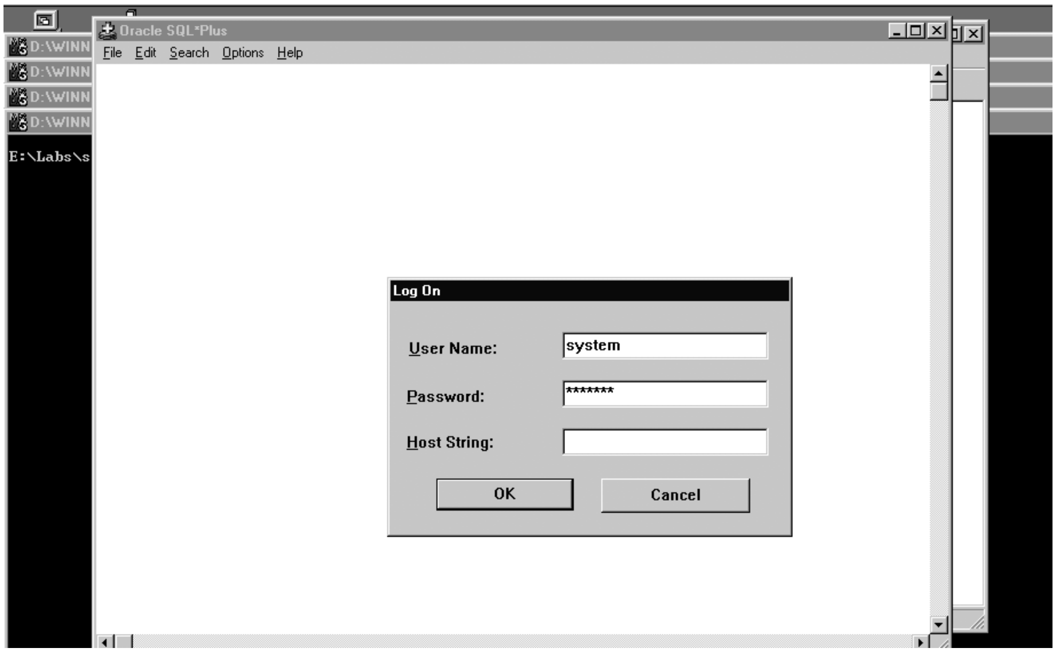
Step 3: Generate a Workload

To simulate a real world system, users need to perform work on your database. For this workshop, you simulate this work by executing a workload script. There are two different workloads to select from, one for the initial set of 4 databases users and one for the additional 16 users. If you have not yet tuned your database for the current four users, then execute the `current.bat` script by double-clicking its icon in your Workshop folder. If you are now attempting to tune your database for anticipated 20-user load, then double-click `future.bat` in this folder.

You should note the time that you start the script to ensure that you use Statspack snapshots that were taken while the database was under load when you generate a report in step 5. While the workload is running, you may also want to probe the database performance as described in step 4.

The workload script opens one Command window for each simulated user and runs a script that performs database activities appropriate to the type of user (OLTP and DSS). Close each Command window after the workload processing stops, about twenty minutes after it starts, by responding to the prompt: “Press any key to continue….” Note that, if a workload script aborts for any reason, you may have to close the related Command window manually.

Collect Statistics Interactively



Step 4: Collect Statistics Interactively

Recall that you confirmed that Statspack automatically collects statistics every ten minutes and these may be sufficient for your needs. However, you could examine some of the dynamic performance (V$) views, or other data dictionary tables, discussed during the course, particularly if you are repeating the steps to help refine your tuning strategy. For these queries, you may use an existing or new Command window to run a SQL*Plus session (or sessions), or open a SQL*Plus window by double-clicking the icon in your Workshop folder or on your desktop.

Note that all SQL statements executed during a tuning session will affect performance, so take care not to impose too much of your own load on the database. You are tuning the database for your user community, not for your own work. For all but the scenario with assorted problems, you can limit your interactive queries to those appropriate for the problem that you have selected.

Identify Statspack Snapshots

```
Oracle SQL*Plus
File  Edit  Search  Options  Help

SQL*Plus: Release 9.2.0.2.0 - Production on Wed Apr 30 16:39:51 2003

Copyright (c) 1982, 2002, Oracle Corporation.  All rights reserved.


Connected to:
Oracle9i Enterprise Edition Release 9.2.0.2.1 - Production
With the Partitioning, OLAP and Oracle Data Mining options
JServer Release 9.2.0.2.0 - Production

SQL> SELECT snap_id, TO_CHAR(snap_time,'HH24:MI') AS snap_time
  2  FROM stats$snapshot
  3  WHERE snap_time > SYSDATE - 1 / (24 * 60/20);

   SNAP_ID SNAP_TIME
---------- ---------------
        67 16:28
        68 16:38

SQL> |
```

ORACLE

Step 5: Identify Statspack Snapshots

You need a minimum of two Statspack snapshots collected while your workload was running. Your Statspack report will not provide useful information if the snapshots were taken outside of the workload window, including any shutdown and restart of the database. To prepare for your Statspack report, use a new or existing SQL*Plus session to:

1. Connect to a SQL*Plus as `perfstat/perfstat`.
2. Run a query, like the following, to find the IDs of the earliest and latest snapshots collected while the workload was running, based on how many minutes ago you began the workload process in step 3:

   ```
   SELECT snap_id, TO_CHAR(snap_time,'HH24:MI') AS snap_time
   FROM stats$snapshot
   WHERE snap_time > SYSDATE - 1 / (24 * 60/20)
   ```

   The last value in the `WHERE` clause, in this case 20, is the number of minutes since you started the workload. This query is provided in the `snapIDS.sql` script in your Workshop folder for your convenience.
3. Record, for use in the Snapshot reporting program, the IDs of the two snapshots that correspond to times while the workload was running. If you purged the old snapshots immediately before starting the workload, then these should be the two earliest snapshots reported.

Create Statspack Report

```
Oracle SQL*Plus
File  Edit  Search  Options  Help
                              67 30 Apr 2003 16:28     5
                              68 30 Apr 2003 16:38     5



Specify the Begin and End Snapshot Ids
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Enter value for begin_snap: 67
Begin Snapshot Id specified: 67

Enter value for end_snap: 68
End   Snapshot Id specified: 68


Specify the Report Name
~~~~~~~~~~~~~~~~~~~~~~~
The default report file name is sp_67_68.  To use this name,
press <return> to continue, otherwise enter an alternative.
Enter value for report_name: sp_67_68.txt

Using the report name sp_67_68.txt

STATSPACK report for
```

ORACLE

Step 5: Create Statspack Report

Assuming that you have the required two snapshot IDs, execute the `spreport.sql` script that is available in your Workshop folder. You are prompted for the IDs of the snapshots on which your report is to be based and for the name of the report file.

Respond to the first prompt, `Enter value for begin_snap`, with the ID of the earliest snapshot, which you found with the query that you just ran.

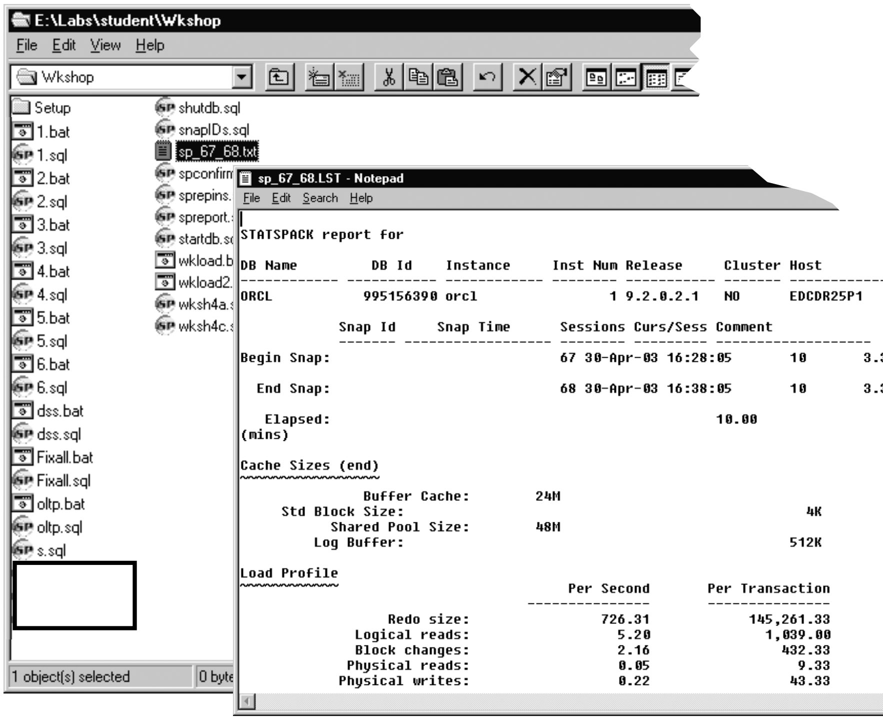
Respond to the second prompt, `Enter value for end_snap`, with the ID of the latest snapshot you found with the query you just ran.

At the prompt for the report name, repeat the default name provided and add a `.txt` extension to it. This will create a file with a name that includes the two snapshot IDs that you provided and that you can open with the Notepad editor by default.

Make a note of the file name so that you can find the correct Statspack report.

Note: The default file name has a `.lst` extension which opens, by default, with the PowerPoint viewer on your system. If you create your file with this extension, then open it either with Notepad in a Command window or with right-click > Edit in Windows Explorer.

Analyze Statistics



Step 6: Analyze Statistics

Examine the Statspack report, along with any information that you collected while the workload was running, to identify what may need to be tuned. You can use a Command window or Windows Explorer to find and open the Statspack report you created. Your report is in the default directory of the SQL*Plus session that generated the Statspack report. If you used a SQL*Plus session initiated from the Workshop folder icon, the default directory is `E:\Labs\student\Wkshop`.

If you are content with the current performance of your database, then you should skip this and the next steps and proceed to step 8. Otherwise, use the techniques that are discussed during the course to determine the tuning focus and the best method of resolving the performance problems (or problems). You may find it helpful to look at Appendix C, which contains tips for each scenario, if you need additional guidance.

In some cases, you may discover that you need to rerun the workload to collect more data interactively before you reach your conclusions. If this is the case, then skip step 7 and continue with step 8. After you have determined what changes or additional information you will need to help improve database performance, proceed to step 7.

Apply Desired Changes

```
SQL> ALTER SYSTEM
  2  SET db_cache_size = 25165824
  3  SCOPE = BOTH;

System altered.

SQL> ALTER SYSTEM
  2  SET log_buffer = 524288
  3  SCOPE = SPFILE;

System altered.
```


Step 7: Apply Desired Changes

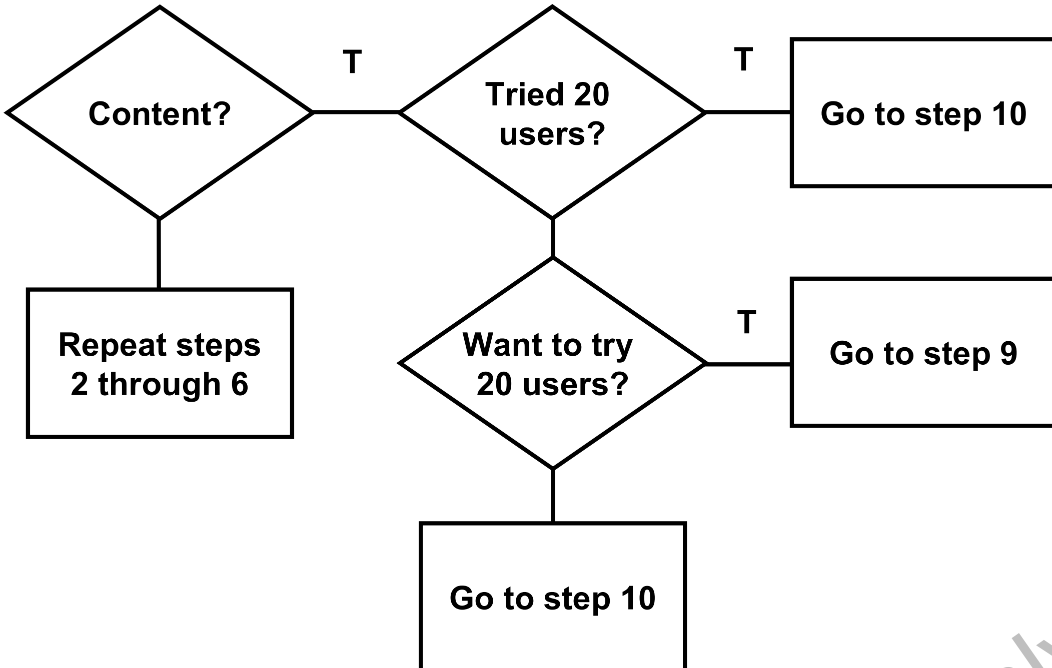
If the changes that you have identified in step 6 require modifications to the database contents, then make these while logged on as the most appropriate user. You may want to confirm that any changes that you make will benefit the users in your database community, for whom the workload scripts create sessions (`oltp` and `dss`). For example, execute a test query or DML statement (which you may want to roll back) while logged on as each of these users.

Alternatively, if your proposed improvements require changes to the initialization parameters, make these changes to your `SPFILE` by using the commands similar to those shown in the graphic in the slide above. For dynamic parameters, you can use the `BOTH` keyword to change the current instance as well as the `SPFILE` entry. If you change one or more static initialization parameters, then you will need to stop and restart your database to make the new values effective.

Your proposed solution may involve changes to database contents as well as to initialization parameters.

Note that the best method for testing the impact of multiple changes is to make one at a time and then perform a new analysis. This way, you can observe the effect of the change without having to determine which alteration is causing the performance improvement (or degradation). However, because of time constraints that are imposed on this workshop, you may need to make multiple modifications and test their impact during a single test.

Evaluate Impact of Changes



Step 8: Evaluate Impact of Changes

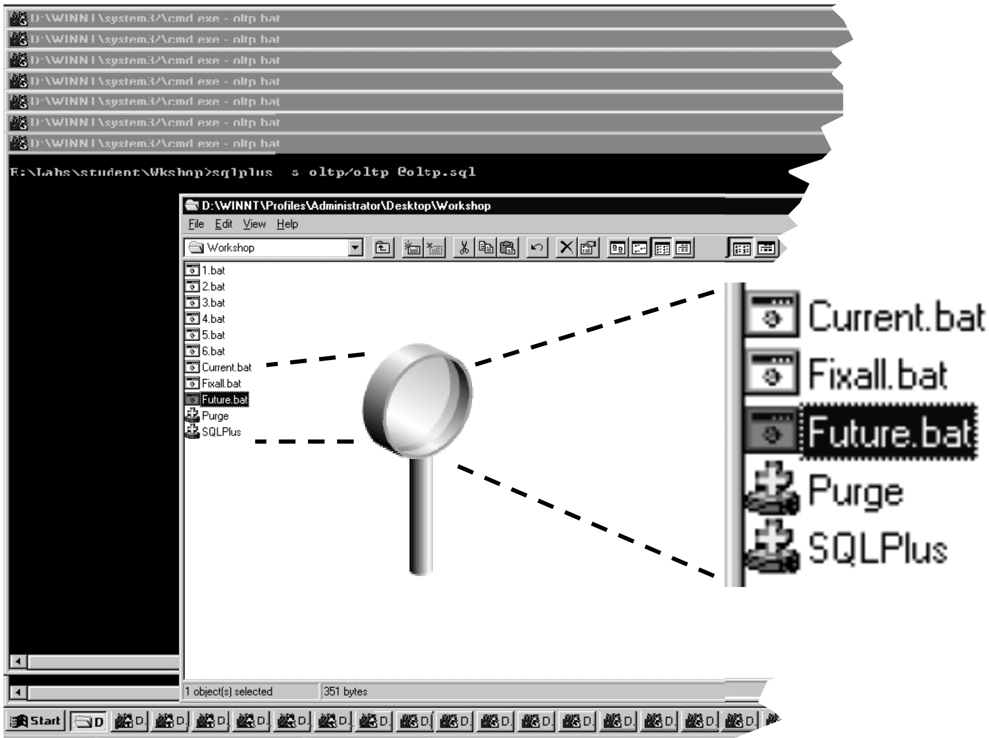
When you reach this step, your next actions depend on what steps you have already completed and on the state of your database. The flow chart explains how you determine what step to take next, depending on what work you have completed and what time and interest you have to continue. In the flow chart, the word "content" means that you are satisfied with the current performance of you database and that you are ready to try something different, either a new scenario or a different workload.

If you are content, then this might be a good time to ensure that you have not violated the 100 MB restriction on the size of the SGA, which is imposed by the constraints that are set by XYZ Company, described in the "Workshop Background" topic. You can query the `v$sga` or `v$sgastat` views to find the current SGA size of your database.

If you need to rerun the workload and you have made changes in step 7, then ensure that those changes are implemented. For example, for static parameters that you changed in the `SPFILE`, you need to stop and restart your database to instantiate the new values.

Note: Do not rerun the setup script (step 1) unless you want to repeat the whole scenario. The setup restores the database settings to their initial values for the scenario, therefore, you would lose any changes that you have made so far, if you execute the script.

Test Performance with More Users



Step 9: Test Performance with More Users

To determine if your current database is ready to support an additional user load, you should rerun steps 2 through 8, substituting the `future.bat` workload script when generating your workload in step 3. You may find that you need to repeat these steps a number of times as you make further tuning refinements in step 7, just as you may have done when performing your initial tuning for the current user community. However, you may find that your current configuration is adequate for the additional users, so you should not make any changes until you run through the steps one time. As with step 8, do not execute the scenario setup script unless you want to redo all of your changes so far.

If you are running out of time and want to test another scenario, or if your instructor advises you to, then you can skip this step, or just collect one set of statistics to see how well you database runs, but not make any further tuning adjustments. However, you must still complete step 10 by using the input from your initial tuning efforts with the simulated workload for four users.

Summarize Findings

- What changes you made
- Why you made those changes
- How your changes impacted performance
- What else you might have investigated
- How you could improve your methodology

Step 10: Summarize Findings

After you are satisfied that you have exhausted the work that you can do to improve your database for the selected scenario, review the work that you have done and try to determine the key elements that helped you. As you summarize your work, include any conclusions about your approaches and your results that may be useful for you (and other class members) in the work environments, as well as in other scenarios in this workshop. Be prepared to discuss this summary with other groups or, if your instructor prefers, with a formal presentation to the class by you or one or more other team members. Consider the following points:

- Parameters and database contents that you changed
- Evidence you used to justify the changes you made
- Differences in performance due to the changes you made
- Issues that you would like to investigate further
- Improvements to your methodology that you would consider implementing

You can now try to analyze and improve tuning problems for a different scenario by returning to step 1 and repeating the series of steps that you just completed, or ask your instructor for further instructions.

Summary

In this lesson, you should have learned how to:

- Implement the Oracle tuning methodology
- Drill down the performance problems suggested by Statspack reports
- Tune a database to support current and anticipated work loads
- Justify the changes that are made to a database configuration for tuning purposes

Practice Solutions Using SQL*Plus

Practice 2

The goal of this practice is to familiarize you with the different methods of collecting statistical information. Throughout this practice Oracle Enterprise Manager can be used if desired. SQL Worksheet can be used instead of SQL*Plus and there are many uses for the Oracle Enterprise Manager console. (Solutions for Oracle Enterprise Manager can be found in Appendix B).

1. Log on as directed by the instructor. If the database is not already started, connect to SQL*Plus using `sys/oracle as sysdba`, then start up the instance using the `STARTUP` command. Ensure that the password for the user system is set to `oracle`. Check that `TIMED_STATISTICS` has been set to True; if it has not, then set it using the `ALTER SYSTEM` statement.

```
SQL> CONNECT sys/oracle AS sysdba
SQL> ALTER USER system IDENTIFIED BY oracle;
SQL> SHOW PARAMETER TIMED_STATISTICS
```

   If a value of True is returned, then continue to question 2. If a value of False is returned, then set the `TIMED_STATISTICS` parameter to True using the command:

```
SQL> ALTER SYSTEM SET TIMED_STATISTICS = True
  2    SCOPE = both;
```

2. Connect to SQL*Plus as the `system` user and run a command that will create a trace file for this session. Run a query to count the number of rows in the `dba_tables` dictionary view. To locate your new trace file easier, if possible, delete all the trace files in the `USER_DUMP_DEST` directory before running the trace. Remember to disable the trace command after running the query.

```
SQL> CONNECT system/oracle
SQL> ALTER SESSION SET SQL_TRACE = TRUE;
SQL> SELECT COUNT(*) FROM dba_tables;
SQL> ALTER SESSION SET SQL_TRACE = FALSE;
```

3. At the operating system level view the resulting trace file located in the directory set by `USER_DUMP_DEST`. Do not try to interpret the content of the trace file, because this is the topic of a later lesson.

```
$cd $HOME/ADMIN/UDUMP
$ls –l
-rw-r----- 1 user487 dba 4444 Apr 24 22:28 U487_ora_3270.trc
```

Practice 2 (continued)

4. Open two sessions, the first as `hr/hr`, and the second as `sys/oracle as sysdba`. From the second session generate a user trace file for the first session using the `dbms_system.set_sql_trace_in_session` procedure. Get the `sid` and `serial#` from `v$session`.

   From Session 1
   ```
   $ sqlplus hr/hr
   ```
   Change to Session 2
   ```
   $ sqlplus "sys/oracle as sysdba"
   SQL> SELECT username, sid, serial#
     2  FROM v$session
     3  WHERE username = 'HR';
   SQL> BEGIN
     2  dbms_system.set_sql_trace_in_session
     3  (&SID,&SERIALNUM,TRUE);
     4  END;
     5  /
   ```
   Change to Session 1
   ```
   SQL> SELECT * FROM employees;
   ```
   Change to Session 2
   ```
   SQL> BEGIN
     2  dbms_system.set_sql_trace_in_session
     3  (&SID,&SERIALNUM,FALSE);
     4  END;
     5  /
   ```

5. Confirm that the trace file has been created in the directory set by `USER_DUMP_DEST`.

   ```
   $cd $HOME/ADMIN/UDUMP
   $ls –l
   -rw-r----- 1 dba01  dba   4444 Apr 24 22:28 dba01_ora_3270.trc
   -rw-r----- 1 dba01  dba  15443 Apr 24 22:42 dba01_ora_3281.trc
   ```

6. Connect to SQL*Plus using `sys/oracle as sysdba` and create a new tablespace (`tools`) to hold the tables and other segments required by Statspack. This tablespace must be 200 MB and be dictionary managed (this is not a requirement of Statspack, but will be used later in the course). Name the data file `tools01.dbf` and place it in the `$HOME/ORADATA/u05` directory.

   Note: Dictionary managed is not the default.
   ```
   SQL> CONNECT sys/oracle AS sysdba
   SQL> CREATE TABLESPACE tools
     2  DATAFILE '$HOME/ORADATA/u05/tools01.dbf' SIZE 200M
     3  EXTENT MANAGEMENT DICTIONARY;
   ```

Practice 2 (continued)

Note: If you want to execute the provided script, `sol02_06.sql`, from your client, you must first edit it to provide the correct path name for the file.

7. Confirm and record the amount of free space available within the `tools` tablespace by querying the `dba_free_space` view. Also check that the tablespace is dictionary managed.

```
SQL> SELECT tablespace_name, extent_management
  2  FROM dba_tablespaces
  3  WHERE tablespace_name = 'TOOLS';
SQL> SELECT tablespace_name, sum(bytes)
  2  FROM dba_free_space
  3  WHERE tablespace_name = 'TOOLS'
  4  GROUP BY tablespace_name;
```

8. Connect using `sys/oracle as sysdba`, then install Statspack using the `spcreate.sql` script located in your `$HOME/STUDENT/LABS` directory. Use the following settings when asked by the installation program:
   - User's password = perfstat
   - User's default tablespace = TOOLS
   - User's temporary tablespace = TEMP

```
SQL> CONNECT sys/oracle AS sysdba
SQL> @$HOME/STUDENT/LABS/spcreate.sql
```

9. Query `dba_free_space` to determine the amount of free space left in the `tools` tablespace. The difference between this value and the one recorded in step 7 will be the space required for the initial installation of Statspack.

   Note: The amount of storage space required will increase in proportion to the amount of information stored within the Statspack tables, that is, the number of snapshots.

   Subtract the value received now, from the value received in step 7 to get the amount of space required to install Statspack.

10. Manually collect current statistics using Statspack by running the `snap.sql` script located in `$HOME/STUDENT/LABS`. This will return the `snap_id` for the snapshot just taken, which should be recorded.

```
SQL> @$HOME/STUDENT/LABS/snap.sql
```

Practice 2 (continued)

11. To have Statspack automatically collect statistics every three minutes execute the `spauto.sql` script located in your `$HOME/STUDENT/LABS` directory. Query the database to confirm that the job has been registered using the `user_jobs` view.

```
SQL > @$HOME/STUDENT/LABS/spauto.sql
SQL > SELECT job, next_date, next_sec, last_sec
  2   FROM user_jobs;
```

Note: The `spauto.sql` script in the `$HOME/STUDENT/LABS` directory has been altered from the `spauto.sql` script shipped with the Oracle database. The alteration has changed the time between snapshots from 1 hour to 3 minutes.

12. After waiting for a period in excess of three minutes query the `stats$snapshot` view to list which snapshots have been collected. There must be at least two snapshots before moving to the next step.

```
SQL> SELECT snap_id,
  2  TO_CHAR(startup_time,'dd Mon "at" HH24:mi:ss')
  3  instart_fm,
  4  TO_CHAR(snap_time,'dd Mon YYYY HH24:mi') snap_date,
  5  snap_level “level”
  6  FROM stats$snapshot
  7  ORDER BY snap_id;
```

Note: If the job scheduler is not working, check the value of the `JOB_QUEUE_PROCESSES` parameter. The value should be greater than 0.

13. When there are at least two snapshots, start to generate a report. This is performed using the `spreport.sql` script found in the `$HOME/STUDENT/LABS` directory. The script lists the snapshot options available and then requests the beginning `snap id` and the end `snap id`. The user is then requested to give a filename for the report. It is often best left to the default.

```
SQL> @$HOME/STUDENT/LABS/spreport.sql
```

14. Locate the report file in your current directory. Use any text editor to open and examine the report. The first page shows a collection of the most queried statistics.

```
$ vi sp_X_Y.lst
```

where X is the starting snapshot and Y is the ending snapshot (this is true if the default report filename was used).

15. Connect to the database as a system administrator `sys/oracle as sysdba`.

```
SQL> CONNECT sys/oracle AS sysdba
```

Practice 2 (continued)

16. Query the database to determine what system wait events have been registered since startup using `v$system_event`.

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

Dynamic Performance Views

17. Determine whether there are any sessions actually waiting for resources, using `v$session_wait`.

```
SQL> SELECT sid, event, p1text, wait_time, state
  2  FROM v$session_wait;
```

18. Stop the automatic collection of statistics by removing the job. This is performed by connecting as `perfstat/perfstat` and querying the `user_jobs` view to get the job number. Then execute the `dbms_job.remove` procedure.

```
SQL> CONNECT perfstat/perfstat
SQL> SELECT job, log_user
  2  FROM user_jobs;
SQL> EXECUTE dbms_job.remove(&job_to_remove);
```

19. Connect to your database using Oracle Enterprise Manager. The lecturer will supply the information required to connect to the Oracle Management Server. After you have connected to the database, use Oracle Enterprise Manager to explore the database. Examine items such as the number of tablespaces, users, and tables.

Oracle Classroom Only

If you are in an Oracle classroom you must perform the following four steps that are specific to the Oracle classroom setup:

a. Click the `omsconfig` file update icon on the desktop and enter the name of the UNIX server your class is using. Your instructor will provide the server name. Please note that this is a case-sensitive entry.
b. Open an MSDOS window.
c. At the command prompt enter: `oemctl start oms`
   Wait for the message: "The Oracleoracle92_homeManagementServer service was started successfully."

Practice 2 (continued)

Oracle Classroom Only (continued)

d. Close the MSDOS window.
   Start the Oracle Enterprise Manager Console and set the Administrator to `sysman` and the password to `oem_temp`. When prompted, change the password to `oracle`. Select Discover Nodes from the Navigator and enter the host name of the server of your working database.

   i. From the Start menu > Programs > Oracle – OracleHome > Enterprise Manager Console
   ii. Make sure the Login to the Oracle Management Server is selected.
   iii. Administrator: sysman
   iv. Password: oem_temp
   v. Management server is your machine.
   vi. When prompted to change the sysman password to oracle.
   vii. Select Navigator > Discover Nodes from the console menu, or select Discover Nodes from the right mouse shortcut menu to open the Discover Nodes dialog box.
   viii. From the Discovery Wizard: Introduction page, click Next, enter the name of your UNIX database server, and click Next.
   ix. Click Next, give your regular administrator access to your database.
   x. Click Finish, then OK. If your discovery was not successful contact your instructor.

20. From Oracle Enterprise Manager load Oracle Expert and create a new tuning session. Limit the tuning scope to "Check for Instance Optimizations." This is done to reduce the time taken to collect information. Collect a new set of data.

    Note: Do not implement the changes that Oracle Expert recommends, because this will be done during the course.

Practice 3

Throughout this practice Oracle Enterprise Manager can be used if desired. SQL Worksheet can be used instead of SQL*Plus and there are many uses for the Oracle Enterprise Manager console. (Solutions for Oracle Enterprise Manager can be found in Appendix B).

1. Connect as `system/oracle` and diagnose database file configuration by querying the `v$datafile`, `v$logfile` and `v$controlfile` dynamic performance views.

```
SQL> CONNECT system/oracle
SQL> SELECT name FROM v$datafile
  2    UNION
  3  SELECT member FROM v$logfile
  4    UNION
  5  SELECT name FROM v$controlfile
  6    UNION
  7  SELECT value FROM v$parameter
  8   WHERE (name LIKE `log_archive_dest%'
  9   AND name NOT LIKE 'log_archive_dest_state%')
 10   OR name IN
 11    ('log_archive_dest','log_archive_duplex_dest');
```

2. Diagnose database file usage by querying the `v$filestat` dynamic performance view, combine with `v$datafile` to get the data file names.

```
SQL> SELECT phyrds, phywrts, d.name
  2  FROM v$datafile d, v$filestat f
  3  WHERE d.file#=f.file#;
```

3. Determine whether there are waits for redo log files by querying the `v$system_event` dynamic performance view, where the waiting event is 'log file sync' or 'log file parallel write'.

```
SQL> SELECT event, total_waits, time_waited, average_wait
  2  FROM v$system_event
  3  WHERE event = 'log file sync'
  4  OR event = 'log file parallel write';
```

Waits for log file sync are indicative of slow disks that store the online logs or unbatched commits. The log file parallel write is much less useful because this event only shows how often LGWR waits, not how often server processes wait. If LGWR waits without impacting user processes, there is no performance problem. If LGWR waits, it is likely that the log file sync event (mentioned above) will also be evident.

Practice 3 (continued)

4. Connect as `perfstat/perfstat` and diagnose file usage from Statspack.
   a. Generate a Statspack report using `$HOME/STUDENT/LABS/spreport.sql`
   b. Locate and open the report file.
   c. Examine the report and search for the "File IO Stats" string.
   Note: On a production database care should be taken in monitoring the disk and controller usage by balancing the workload across all devices. If your examination shows a distinct over-utilization of a particular data file, consider resolving the cause of the amount of I/O. For example, investigate the number of full table scans, clustering of files on a specific device and under-utilization of indexes. If after this the problem remains then look at placing the data file on a low utilization device.

5. Connect as `system/oracle` and enable checkpoints to be logged in the alert file by setting the value of the `LOG_CHECKPOINTS_TO_ALERT` parameter to True using the `ALTER SYSTEM SET` command.

```
SQL> CONNECT system/oracle
SQL> ALTER SYSTEM SET LOG_CHECKPOINTS_TO_ALERT = True;
```

6. Connect as `sh/sh` and execute the `$HOME/STUDENT/LABS/lab03_06.sql` script to provide a workload against the database.

```
SQL> CONNECT sh/sh
SQL> @$HOME/STUDENT/LABS/lab03_06.sql
```

7. At the operating system level use the editor to open the alert log file (located in the directory specified by `BACKGROUND_DUMP_DEST`). Then determine the checkpoint frequency for your instance by searching for messages containing the phrase "Completed Checkpoint." The time difference between two consecutive messages is the checkpoint interval.

   Open the alert log file using an editor and search for the line: Completed checkpoint. The line before this will be the time at which the checkpoint occurred. Search for the following checkpoint time and then subtract to get the time between checkpoints.

   Note: On a production system the checkpoint interval should range between 15 minutes to 2 hours. The actual interval is dependant on the type of application and the amount of data manipulation language (DML) activity.

Practice 4

The objective of this practice is to use diagnostic tools to monitor and tune the shared pool. Throughout this practice Oracle Enterprise Manager can be used if desired. SQL Worksheet can be used instead of SQL*Plus and there are many uses for the Oracle Enterprise Manager console. (Solutions for Oracle Enterprise Manager can be found in Appendix B).

1. Connect using `sys/oracle as sysdba` and check the size of the shared pool.

```
SQL> CONNECT sys/oracle AS sysdba
SQL> SHOW PARAMETER SHARED_POOL

NAME                          TYPE        VALUE
----------------------------- ----------- ---------------
shared_pool_reserved_size     big integer 2516582
shared_pool_size              big integer 50331648
```

2. Connect as `perfstat/perfstat`, execute the `$HOME/STUDENT/LABS/snap.sql` script to collect initial snapshot of statistics, and note the snapshot number.

```
SQL> CONNECT perfstat/perfstat
SQL> @$HOME/STUDENT/LABS/snap.sql
```

3. To simulate user activity against the database open two operating system sessions. In session 1 connect as `hr/hr` and run the `$HOME/STUDENT/LABS/lab04_03_1.sql` script. In the second session connect as `hr/hr` and run the `$HOME/STUDENT/LABS/lab04_03_2.sql` script.

In session 1:

```
SQL> CONNECT hr/hr
SQL> @$HOME/STUDENT/LABS/lab04_03_1.sql
```

In session 2:

```
SQL> CONNECT hr/hr
SQL> @$HOME/STUDENT/LABS/lab04_03_2.sql
```

4. Connect as `system/oracle` and measure the pin-to-reload ratio for the library cache by querying `v$librarycache`. Determine whether it is a good ratio or not.

Using the dynamic view:

```
SQL> CONNECT system/oracle
SQL> SELECT SUM(pins), SUM(reloads),
  2  SUM(pins) * 100 /  SUM(pins+reloads) "Pin Hit%"
  3  FROM v$librarycache;
```

Practice 4 (continued)

5. Connect as `system/oracle` and measure the get-hit ratio for the data dictionary cache by querying `v$rowcache`. Determine whether it is a good ratio or not using the dynamic view.

```
SQL> CONNECT system/oracle
SQL> SELECT SUM(getmisses), SUM(gets),
  2  SUM(getmisses )*100/SUM(gets)“MISS %”
  3  FROM v$rowcache;
```

   If `GETMISSES` are lower than 15% of the `GETS`, then it is a good ratio.

6. Connect as `perfstat/perfstat` and run the `$HOME/STUDENT/LABS/snap.sql` script to collect a statistic snapshot and obtain the snapshot number. Record this number.

```
SQL> CONNECT perfstat/perfstat
SQL> @$HOME/STUDENT/LABS/snap.sql
```

7. As `perfstat/perfstat` obtain the statistics report between the two recorded snapshot IDs (from questions 2 and 6) by running the `$HOME/STUDENT/LABS/spreport.sql` script.

```
SQL> CONNECT perfstat/perfstat
SQL> @$HOME/STUDENT/LABS/spreport.sql
```

   The following is an example of using a beginning `snapshot_id` of 3 and an ending `snapshot_id` of 5.

```
Specify the Begin and End Snapshot Ids
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Enter value for begin_snap:   3

Enter value for end_snap: 5
End   Snapshot Id specified: 5


Specify the Report Name
~~~~~~~~~~~~~~~~~~~~~~~
The default report file name is sp_3_5. To use this name,
press <return> to continue, otherwise enter an alternative.
Enter value for report_name:
```

Practice 4 (continued)

8. Analyze the generated report in the current directory. What would you consider doing if the library hit ratio (found under the heading "Instance Efficiency Percentages") is less than 98%?

   Increase the `SHARED_POOL_SIZE` parameter.

9. Connect as `system/oracle` and determine which packages, procedures, and triggers are pinned in the shared pool by querying `v$db_object_cache`.

```
SQL> CONNECT system/oracle
SQL> SELECT name, type, kept
  2  FROM v$db_object_cache
  3  WHERE type IN
  4  ('PACKAGE', 'PROCEDURE', 'TRIGGER', 'PACKAGE BODY');
```

10. Connect using `sys/oracle as sysdba` and pin one of the Oracle supplied packages that must be kept in memory, such as `sys.standard` using the `dbms_shared_pool.keep` procedure, which is created by running the `$ORACLE_HOME/rdbms/admin/dbmspool.sql` script.

```
SQL> CONNECT sys/oracle AS SYSDBA
SQL> @?/rdbms/admin/dbmspool
SQL> EXECUTE dbms_shared_pool.keep('SYS.STANDARD');
SQL> SELECT distinct owner, name
  2  FROM v$db_object_cache
  3  WHERE kept='YES'
  4  AND name LIKE '%STAND%';
```

   Note: After you complete the other steps in this practice and before you move on to the next practice, use the `unkeep` procedure to unpin the object you pinned in the shared pool. This will help avoid memory problems in later practices.

11. Determine the amount of session memory used by your session by querying the `v$mystat` view. Limit the output by including the clause:

```
WHERE name = 'session uga memory'
```

```
SQL> SELECT a.name, b.value
  2  FROM v$statname a, v$mystat b
  3  WHERE a.statistic# = b.statistic#
  4  AND name = 'session uga memory';
```

   Note: Because you are not using the Oracle Shared Server configuration this memory resides outside the SGA.

Practice 4 (continued)

12. Determine the amount of session memory used for all sessions, using `v$sesstat` and `v$statname` views:

```
SQL> SELECT SUM(value)||' bytes' "Total session memory“
  2  FROM v$sesstat, v$statname
  3  WHERE name = 'session uga memory'
  4  AND v$sesstat.statistic# = v$statname.statistic#;
```

Practice 5

The objective of this practice is to use available diagnostic tools to monitor and tune the database buffer cache. Throughout this practice Oracle Enterprise Manager can be used if desired. SQL Worksheet can be used instead of SQL*Plus and there are many uses for the Oracle Enterprise Manager console. (Solutions for Oracle Enterprise Manager can be found in Appendix B).

1. Connect as `perfstat/perfstat` and run a statistic snapshot. Make a note of the snapshot number. The snap shot can be taken by running the `$HOME/STUDENT/LABS/snap.sql` script file.

```
SQL> CONNECT perfstat/perfstat
SQL> @$HOME/STUDENT/LABS/snap.sql
```

2. To simulate user activity against the database, connect as the `hr/hr` user and run the `lab05_02.sql` script.

```
SQL> CONNECT hr/hr
SQL> @$HOME/STUDENT/LABS/lab05_02.sql
```

3. Connect as `system/oracle` and measure the hit ratio for the database buffer cache using the `v$sysstat` view. Determine whether it is a good ratio or not.

```
SQL> CONNECT system/oracle
SQL> SELECT 1 - (phy.value – lob.value – dir.value)
  2             / ses.value  "CACHE HIT RATIO“
  3  FROM   v$sysstat ses, v$sysstat lob,
  4         v$sysstat dir, v$sysstat phy
  5  WHERE  ses.name = 'session logical reads'
  6  AND    dir.name = 'physical reads direct'
  7  AND    lob.name = 'physical reads direct (lob)'
  8  AND    phy.name = 'physical reads';
```

4. Connect as `perfstat/perfstat` run a statistic snapshot. Make a note of the snapshot number. The snapshot can be taken by running the `$HOME/STUDENT/LABS/snap.sql` script file.

```
SQL> CONNECT perfstat/perfstat
SQL> @$HOME/STUDENT/LABS/snap.sql
```

5. Use the report from Statspack between the last two snapshots to check the buffer cache hit ratio, using the `$HOME/STUDENT/LABS/spreport.sql` script. Then analyze the buffer hit % in the “Instance Efficiency Percentages” section.

```
SQL> @$HOME/STUDENT/LABS/spreport.sql
```

Practice 5 (continued)

Note: On a production database if the ratio is bad, add new buffers, run steps 2 to 5, and examine the new ratio to verify that the ratio has improved. If the ratio is good, remove buffers, run steps 2 to 5, and verify if the ratio is still good.

6. Connect as `system/oracle` and determine the size of the `temp_emps` table in the `hr` schema that you want to place in the keep buffer pool. Do this by using the `dbms_stats.gather_table_stats` procedure and then query the `blocks` column of the `dba_tables` view for the `temp_emps` table.

```
SQL> CONNECT system/oracle
SQL> EXECUTE dbms_stats.gather_table_stats -
>              ('HR','TEMP_EMPS');
SQL> SELECT table_name , blocks
  2  FROM dba_tables
  3  WHERE table_name IN ('TEMP_EMPS');
```

7. Keep `temp_emps` in the keep pool. Use the `ALTER SYSTEM` command to set `DB_KEEP_CACHE_SIZE` to 4 MB for the keep pool. Limit the scope of this command to the spfile.

```
SQL> ALTER SYSTEM SET DB_KEEP_CACHE_SIZE=4M SCOPE=spfile;
```

8. For the keep pool to be allocated the database needs to be restarted. You will need to be connected as a `sysdba` user to perform this task.

```
SQL> CONNECT sys/oracle AS sysdba
SQL> SHUTDOWN IMMEDIATE
SQL> STARTUP
```

9. Connect as `system/oracle` and enable the `temp_emps` table in the `hr` schema for caching in the keep pool, using the storage clause of the `ALTER TABLE` command.

```
SQL> CONNECT system/oracle
SQL> ALTER TABLE hr.temp_emps
  2  STORAGE (BUFFER_POOL Keep);

SQL> SELECT table_name, buffer_pool
  2  FROM dba_tables
  3  WHERE buffer_pool = 'KEEP';
```

Practice 5 (continued)

10. Connect as `hr/hr` and run the `$HOME/STUDENT/LABS/lab05_10.sql` script. This will execute a query against the `temp_emps` table in the `hr` schema.

```
SQL> CONNECT hr/hr
SQL> @$HOME/STUDENT/LABS/lab05_10.sql
```

11. Connect using `sys/oracle as sysdba` and check for the hit ratio in different buffer pools, using the `v$buffer_pool_statistics` view.

```
SQL> CONNECT sys/oracle AS sysdba
SQL> SELECT name, physical_reads, db_block_gets,
  2  consistent_gets, 1 - (physical_reads
  3  / (db_block_gets + consistent_gets)) "hits"
  4  FROM v$buffer_pool_statistics;
```

Practice 6

Throughout this practice Oracle Enterprise Manager can be used if desired. SQL Worksheet can be used instead of SQL*Plus and there are many uses for the Oracle Enterprise Manager console. (Solutions for Oracle Enterprise Manager can be found in Appendix B).

1. Connect as `sys/oracle AS sysdba` and, without restarting the instance, resize the `DB_CACHE_SIZE` to 12 Mb. Limit the effect of this command to memory, so as not to modify the spfile.

```
SQL> CONNECT sys/oracle AS sysdba
SQL> ALTER SYSTEM SET DB_CACHE_SIZE = 12M
  2    SCOPE = memory;
```

   Note: This will encounter an error because the total SGA size will be bigger than `SGA_MAX_SIZE`. To overcome this you must either change the value of `SGA_MAX_SIZE` and restart the instance (which is what dynamic allocation is meant to avoid) or resize a component, thus making memory available for the increase in the buffer cache.

2. Reduce the memory used by the shared pool. Limit the effect of this command to memory, so as not to modify the spfile.

```
SQL> ALTER SYSTEM SET SHARED_POOL_SIZE = 40M
  2    SCOPE = memory;
```

3. Without restarting the instance, resize the `DB_CACHE_SIZE` to 12 Mb. Limit the effect of this command to memory, so as not to modify the spfile.

```
SQL> ALTER SYSTEM SET DB_CACHE_SIZE = 12M
  2    SCOPE = memory;
```

   Note: This time the memory is available so the command will be executed.

4. To return the SGA to the original configuration, restart the instance. You must be connected as a `sysdba` user to perform this task.

```
SQL> CONNECT sys/oracle AS sysdba
SQL> SHUTDOWN IMMEDIATE
SQL> STARTUP
```

Practice 7

Throughout this practice Oracle Enterprise Manager can be used if desired. SQL Worksheet can be used instead of SQL*Plus and there are many uses for the Oracle Enterprise Manager console. (Solutions for Oracle Enterprise Manager can be found in Appendix B).

1. Connect as `perfstat/perfstat` and collect a snapshot of the current statistics by running the `$HOME/STUDENT/LABS/snap.sql` script. Record the snapshot ID for later use.

```
SQL> CONNECT perfstat/perfstat
SQL> @$HOME/STUDENT/LABS/snap;
```

2. Connect as user `sh/sh` and run the `$HOME/STUDENT/LABS/lab07_02.sql` script in the `$HOME/STUDENT/LABS` directory to put a workload on the database.

```
SQL> CONNECT sh/sh
SQL> @$HOME/STUDENT/LABS/lab07_02.sql
```

3. Connect as `system/oracle` and query the `v$sysstat` view to determine whether there are space requests for the redo log buffer.

```
SQL> CONNECT system/oracle
SQL> SELECT rbar.name, rbar.value, re.name, re.value
  2  FROM v$sysstat rbar, v$sysstat re
  3  WHERE rbar.name = 'redo buffer allocation retries'
  4  AND re.name = 'redo entries';
```

4. Connect as `perfstat/perfstat` and collect another set of statistics using the `$HOME/STUDENT/LABS/snap.sql` script. Then use `$HOME/STUDENT/LABS/spreport.sql` to generate a report using the two snapshot IDs that you have collected. From the list of snapshots select a beginning and end value. The beginning is the value recorded in step 1 and the end value step 4. Record the name of the report file. View the generated file using an editor and locate the "log buffer space" statistic.

```
SQL> CONNECT perfstat/perfstat
SQL> @$HOME/STUDENT/LABS/snap.sql;
SQL> @$HOME/STUDENT/LABS/spreport.sql
```

Practice 7 (continued)

5. Connect as `sys/oracle AS sysdba` and increase the size of the redo log buffer in the spfile by changing the value of the `LOG_BUFFER` parameter. Because this parameter is static you must specify spfile.

```
SQL> CONNECT sys/oracle AS sysdba
SQL> ALTER SYSTEM SET LOG_BUFFER = 128000
  2    SCOPE = spfile;
```

6. To have the new value for the `LOG_BUFFER` take effect, you must restart the instance. Then confirm that the change has occurred.

```
SQL> SHUTDOWN immediate
SQL> STARTUP
SQL> SHOW PARAMETER LOG_BUFFER
```

Practice 9

Throughout this practice Oracle Enterprise Manager can be used if desired. SQL Worksheet can be used instead of SQL*Plus and there are many uses for the Oracle Enterprise Manager console. (Solutions for Oracle Enterprise Manager can be found in Appendix B).

1. Set the database to use the manual sort option by changing the value of the `WORKAREA_SIZE_POLICY` parameter to Manual. Set the `SORT_AREA_SIZE` parameter to 512 bytes.

```
SQL> CONNECT sys/oracle AS sysdba
SQL> ALTER SYSTEM SET WORKAREA_SIZE_POLICY = manual
  2    SCOPE = both;
SQL> ALTER SYSTEM SET SORT_AREA_SIZE = 512
  2    SCOPE= spfile;
```

2. For the new values of the `WORKAREA_SIZE_POLICY` and `SORT_AREA_SIZE` parameters to take effect, you must restart the instance. Then query the `v$sysstat` view and record the value for memory sorts and disk sorts.

```
SQL> SHUTDOWN immediate
SQL> STARTUP
SQL> SELECT name, value
  2  FROM v$sysstat
  3  WHERE  name LIKE 'sorts%';
```

Note: The statistics in `v$sysstat` are collected from startup. If you need to obtain accurate statistics per statement, you must record statistics before the statement runs and again afterwards. Subtracting the two values gives the statistics for the statement.

3. To perform a sort on the database that will have sorts to disk connect as `sh/sh` and execute the `$HOME/STUDENT/LABS/lab09_03.sql` script.

```
SQL> CONNECT sh/sh
SQL> @$HOME/STUDENT/LABS/lab09_03.sql;
```

Note: If this script fails due to a lack of free space in the `temp` tablespace, then connect as `system/oracle` and resize the temporary tablespace.

```
SQL> CONNECT system/oracle
SQL> ALTER DATABASE TEMPFILE
  2  '$HOME/ORADATA/u02/temp01.dbf' RESIZE 400M;
```

Note: If you wish to execute the provided script, `sol09_03_2.sql`, from your client, you must first edit it to provide the correct path name for the file.

Practice 9 (continued)

4. Connect as `system/oracle`, query the `v$sysstat` view again, and record the value for sorts (memory) and sorts (disk). Subtract the values from the recorded value in question 2. If the ratio of Disk to Memory sorts is greater than 5% then increase the sort area available.

```
SQL> CONNECT system/oracle
SQL> SELECT name, value
  2  FROM v$sysstat
  3  WHERE  name LIKE 'sorts%';
```

5. Connect as `system/oracle` and query the `tablespace_name`, `max_sort_size`, and `max_used_size` columns from the `v$sort_segment` view.

```
SQL> CONNECT system/oracle
SQL> SELECT tablespace_name, max_sort_size,
  2    max_used_size
  3  FROM v$sort_segment;
```

Note: The `used_extents` and `free_extents` columns from `v$sort_segment` are also useful in monitoring the temporary tablespace. If the view contains no rows, then it means that all sort operations since startup have completed in memory.

6. To decrease the number of sorts going to a temporary tablespace, increase the value of the `SORT_AREA_SIZE` parameter to 512000 using the `ALTER SESSION` command.

```
SQL> ALTER SESSION SET SORT_AREA_SIZE = 512000;
```

7. Connect as `system/oracle` and configure the parameters for automatic PGA memory allocation using the `ALTER SYSTEM` command. Use the values Auto for `WORKAREA_SIZE_POLICY` and 10M for `PGA_AGGREGATE_TARGET`).

```
SQL> CONNECT system/oracle
SQL> ALTER SYSTEM SET PGA_AGGREGATE_TARGET = 10M
  2    SCOPE = Both;
SQL> ALTER SYSTEM SET WORKAREA_SIZE_POLICY = Auto
  2    SCOPE = Both;
```

Practice 11

The objective of this practice is to familiarize you with SQL statement execution plans and to interpret the formatted output of a trace file generated using SQL Trace and the formatted output generated by TKPROF. Throughout this practice Oracle Enterprise Manager can be used if desired. SQL Worksheet can be used instead of SQL*Plus and there are many uses for the Oracle Enterprise Manager console. (Solutions for Oracle Enterprise Manager can be found in Appendix B).

1. Connect as `hr/hr` and create the `plan_table` table under the `hr` schema, if it is not already created, by running the `$ORACLE_HOME/rdbms/admin/utlxplan.sql` script.

```
SQL> CONNECT hr/hr
SQL> @$ORACLE_HOME/rdbms/admin/utlxplan.sql
```

   Note: If `plan_table` already exists and holds rows then truncate the table.

2. Set the optimizer mode to rule based using the `ALTER SESSION` command and generate the explain plan for the statement `$HOME/STUDENT/LABS/lab11_02.sql`. View the generated plan by querying object name, operation, option, and optimizer from the `plan_table` table.

```
SQL> ALTER SESSION SET OPTIMIZER_GOAL = Rule;
SQL> EXPLAIN PLAN FOR
  2  @$HOME/STUDENT/LABS/lab11_02.sql
SQL> SELECT object_name, operation, options, optimizer
  2  FROM plan_table;
```

3. Truncate the `plan_table` table. Change the optimizer mode to cost based by setting the value to `All_rows` and rerun the explain plan for `$HOME/STUDENT/LABS/lab11_02.sql`. Notice that the optimizer mode and the explain plan have changed.

```
SQL> TRUNCATE TABLE plan_table;
SQL> ALTER SESSION SET OPTIMIZER_GOAL = all_rows;
SQL> EXPLAIN PLAN FOR
  2  @$HOME/STUDENT/LABS/lab11_02.sql
SQL> SELECT object_name, operation, options, optimizer
  2  FROM plan_table;
```

   Note: Although exactly the same scripts are being run, due to the different optimizer settings different explain paths are found. With rule-based, one of the rules is to use any index that is on the columns in the where clause. By using cost-based optimizer mode, the server has been able to determine that it will be faster to just perform a full table scan, due to the number of rows being returned by the script.

Practice 11 (continued)

4. Truncate the `plan_table` table and set the optimizer mode to Rule by using the `ALTER SESSION` command. This time generate the explain plan for the `$HOME/STUDENT/LABS/lab11_04.sql` script. Examine the script which is a copy of `$HOME/STUDENT/LABS/lab11_02.sql` except that it changes the line "`SELECT *` " to include a hint `/*+ all_rows*/` for the optimizer. View the generated execution plan by querying object name, operation, option, and optimizer from `plan_table` table.

```
SQL> TRUNCATE TABLE plan_table;
SQL> ALTER SESSION SET OPTIMIZER_GOAL = Rule;
SQL> EXPLAIN PLAN FOR
  2  @$HOME/STUDENT/LABS/lab11_04.sql
SQL> SELECT object_name, operation, options, optimizer
  2  FROM plan_table;
```

5. Exit out of SQL*Plus, change the directory to `$HOME/ADMIN/UDUMP` and delete all the trace files already generated.

```
SQL> EXIT
$ cd $HOME/ADMIN/UDUMP
$ rm *.trc
```

   Note: this step is performed only to make it easier to find the trace file generated. It is not a requirement of SQL Trace.

6. Connect as `sh/sh` and enable SQL Trace, using the `ALTER SESSION` command, to collect statistics for the script, `$HOME/STUDENT/LABS/lab11_06.sql`. Run the script. After the script has completed, disable SQL Trace, then format your trace file using `TKPROF`. Use the options `SYS=NO` and `EXPLAIN= sh/sh`. Name the file `myfile.txt.`

```
SQL> CONNECT sh/sh
SQL> ALTER SESSION SET SQL_TRACE = True;
SQL> @$HOME/STUDENT/LABS/lab11_06.sql
SQL> ALTER SESSION SET SQL_TRACE = False;
$ cd $HOME/ADMIN/UDUMP
$ ls -l
-rw-r----- 1 user457 dba 2180 May 4 00:27 ser457_ora_10424.trc
$ tkprof user457_ora_10424.trc myfile.txt explain=sh/sh sys=no
```

7. View the output file `myfile.txt` and note the CPU, current, and query figures for the fetch phase. Do not spend time analyzing the contents of this file because the only objective here is to become familiar and comfortable with running `TKPROF` and `SQL*Trace`.

```
$ more myfile.txt
```

Practice 12

The objective of this practice is to familiarize you with the `dbms_stats` package. Throughout this practice Oracle Enterprise Manager can be used if desired. SQL Worksheet can be used instead of SQL*Plus and there are many uses for the Oracle Enterprise Manager console. (Solutions for Oracle Enterprise Manager can be found in Appendix B).

1. Connect as `hr/hr` and create a table `new_employees` as a copy of the `employees` table. Gather statistics on the `new_employees` table and determine the current number of rows in the `new_employees` table. Record the number of rows for comparison later.

```
SQL> CONNECT hr/hr
SQL> CREATE TABLE new_employees
  2    AS SELECT *
  3       FROM employees;
SQL> EXECUTE -
>    dbms_stats.gather_table_stats ('HR','NEW_EMPLOYEES');
SQL> SELECT table_name, num_rows
  2  FROM user_tables
  3  WHERE table_name = 'NEW_EMPLOYEES';
```

2. Increase the size of the `new_employees` table by using the `lab12_02.sql` script.

```
SQL> @$HOME/STUDENT/LABS/lab12_02.sql
```

3. Confirm that the statistics have not been changed in the data dictionary by re-issuing the same statement as in question 1.

```
SQL> SELECT table_name, num_rows
  2  FROM user_tables
  3  WHERE table_name = 'NEW_EMPLOYEES';
```

4. Connect `hr/hr` and gather statistics for all objects under the `hr` schema using the `dbms_stats` package. While gathering the new statistics save the current statistics in a table named `stats`.

   a. Connect as `hr/hr` and create a table to hold statistics in that schema.
   ```
   SQL> CONNECT hr/hr
   SQL> execute dbms_stats.create_stat_table('HR','STATS');
   ```
   b. Save the current schema statistics into your local statistics table.
   ```
   SQL> execute dbms_stats.export_schema_stats('HR','STATS');
   ```
   c. Analyze all objects under the `hr` schema.
   ```
   SQL> execute dbms_stats.gather_schema_stats('HR');
   ```

Practice 12 (continued)

5. Determine that the current number of rows in the employees table has been updated in the data dictionary. This should be twice the number of rows recorded in question 1.

```
SQL> SELECT table_name, num_rows
  2  FROM user_tables
  3  WHERE table_name = 'NEW_EMPLOYEES';
```

6. Remove all schema statistics from the dictionary and restore the original statistics you saved in step 4b.

```
SQL> execute dbms_stats.delete_schema_stats('HR');
SQL> execute dbms_stats.import_schema_stats('HR','STATS');
```

7. Confirm that the number of rows in the `employees` table recorded in the data dictionary has returned to the previous value collected in question 1.

```
SQL> SELECT table_name, num_rows
  2  FROM user_tables
  3  WHERE table_name = 'NEW_EMPLOYEES';
```

Practice 13

Throughout this practice Oracle Enterprise Manager can be used if desired. SQL Worksheet can be used instead of SQL*Plus and there are many uses for the Oracle Enterprise Manager console. (Solutions for Oracle Enterprise Manager can be found in Appendix B).

1. Connect using `sys/oracle AS sysdba` and query the `tablespace_name` and `extent_management` columns of `dba_tablespaces` to determine which tablespaces are locally managed and which are dictionary managed. Record which tablespaces are dictionary managed.

```
SQL> CONNECT / AS sysdba
SQL> SELECT tablespace_name, extent_management
  2  FROM dba_tablespaces;
```

2. Alter the `hr` user to have the `tools` tablespace as the default.

```
SQL> ALTER USER hr DEFAULT TABLESPACE tools;
```

3. Examine the `v$system_event` view and note the total waits for the statistic enqueue.

```
SQL> SELECT event, total_waits
  2  FROM v$system_event
  3  WHERE event = 'enqueue';
```

Note: On a production system you would be more likely to pick up the contention through the Statspack report.

4. Also examine the `v$enqueue`_stat view for `eq_type` 'ST' to determine the `total_wait#` for the ST enqueue, which is the space management enqueue.

```
SQL> SELECT *
  2  FROM v$enqueue_stat
  3  WHERE eq_type = 'ST';
```

5. Exit out of the SQL*Plus session and change the directory to `$HOME/STUDENT/LABS`. Run the `lab13_04.sh` script from the operating system prompt. This script will log five users onto the database simultaneously and then each user creates and drops tables. The tables each have many extents. The script must be run from the `$HOME/STUDENT/LABS` directory or it will fail.

```
$ cd $HOME/STUDENT/LABS
$ ./lab13_04.sh
```

6. Connect as `system/oracle` and again examine the `v$enqueue_stat` view for `eq_type` 'ST' to determine the value of `total_wait#` for the ST enqueue, which is the space management enqueue.

```
$ SQL*Plus system/oracle
SQL> SELECT *
  2  FROM v$enqueue_stat
  3  WHERE eq_type = 'ST';
```

   Note: Record the difference in the number of waits for the ST enqueue for extent anagement using a dictionary managed tablespace. This value is found by subtracting the first wait value (from practice 13-04) from the second wait value (from practice 13-06).

7. Create a new locally managed tablespace `test`, name the data file `test01.dbf`, and place it in the `$HOME/ORADATA/u06` directory. Set the size to 120 MB and a uniform extent size of 20 KB.

```
SQL> CREATE TABLESPACE test
  2  DATAFILE '$HOME/ORADATA/u06/test01.dbf' SIZE 120M
  3  UNIFORM SIZE 20k;
```

   Note: If you want to execute the provided script, `sol13_07.sql`, from your client, you must first edit it to provide the correct pathname for the file.

8. Alter the default tablespace of the `hr` user to `test`.

```
SQL> ALTER USER hr DEFAULT TABLESPACE test;
```

   Note: The same steps are covered again. This time you are looking for the number of waits for the ST enqueue caused by locally managed tablespaces.

9. Examine and record the initial `total_wait#` for 'ST' in the `v$enqueue_stat` view.

```
SQL> SELECT *
  2  FROM v$enqueue_stat
  3  WHERE eq_type = 'ST';
```

10. Exit out of the SQL*Plus session and change directory to `$HOME/STUDENT/LABS`. Run the `lab13_04.sh` script from the operating system prompt. This script will log five users onto the database simultaneously and then each user creates and drops tables. The tables each have many extents. The script must be run from the `$HOME/STUDENT/LABS` directory or it will fail.

```
$ cd $HOME/STUDENT/LABS
$ ./lab13_04.sh
```

11. Again examine and record the final `total_wait#` for 'ST' in the `v$enqueue_stat` view.

```
SQL> SELECT *
  2  FROM v$enqueue_stat
  3  WHERE eq_type = 'ST';
```

Note: Record the difference in the `total_wait#` for the ST enqueue for extent management using a locally managed tablespace. This value is found by subtracting the first wait value (from practice 13-09) from the second wait value (from practice 13-11). Compare the two results for the different tablespaces. The locally managed tablespace would be far less contentious with extent management because it is managing the space within the tablespace itself.

12. Connect as the `hr/hr` user and run the `$HOME/STUDENT/LABS/lab13_12.sql` script. This will create a similar table (`new_emp`) as the `employees` table but with PCTFREE = 0. The table is then populated with data from the `employees` table.

```
SQL> CONNECT hr/hr
SQL> @$HOME/STUDENT/LABS/lab13_12.sql;
```

13. Run `ANALYZE` on the `new_emp` table and query the `dba_tables` view to determine the value of `chain_cnt` for the `new_emp` table. Record this value.

```
SQL> ANALYZE TABLE new_emp COMPUTE STATISTICS;
SQL> SELECT table_name, chain_cnt
  2  FROM user_tables
  3  WHERE table_name = 'NEW_EMP';
```

14. Create an index called `new_emp_name_idx` on the `last_name` column of the `new_emp` table. Place the index in the tablespace `indx`. Then confirm the index's status in the `user_indexes` view.

```
SQL> CREATE INDEX new_emp_name_idx ON new_emp(last_name)
  2  TABLESPACE indx;
SQL> SELECT index_name, status
  2  FROM user_indexes
  3  WHERE index_name = 'NEW_EMP_NAME_IDX';
```

Practice 13 (continued)

15. Run the `$HOME/STUDENT/LABS/lab13_15.sql` script, which will update the rows of the `new_emp` table. Analyze the `new_emp` table again and query the `user_tables` view to get the new value of `chain_cnt` Record this value. Also check the status of the `new_emp_name_idx` index.

```
SQL> @$HOME/STUDENT/LABS/lab13_15.sql
SQL> ANALYZE TABLE new_emp COMPUTE STATISTICS;
SQL> SELECT table_name, chain_cnt
  2  FROM user_tables
  3  WHERE table_name = 'NEW_EMP';
SQL> SELECT index_name, status
  2  FROM user_indexes
  3  WHERE index_name = 'NEW_EMP_NAME_IDX';
```

16. Resolve the migration caused by the previous update, by using the `ALTER TABLE MOVE` command. This will cause the index to become unusable and should be rebuilt using the `ALTER INDEX REBUILD` command before reanalyzing the `new_emp` table. Confirm that the migration has been resolved by querying `chain_cnt` column in the `user_tables` view and confirm that the index is valid by querying the `user_indexes` view.

```
SQL> ALTER TABLE new_emp MOVE
  2  TABLESPACE users;
SQL> ALTER INDEX new_emp_name_idx REBUILD;
SQL> ANALYZE TABLE new_emp COMPUTE STATISTICS;
SQL> SELECT table_name, blocks, empty_blocks, chain_cnt
  2  FROM user_tables
  3  WHERE table_name = 'NEW_EMP';
SQL> SELECT index_name, status
  2  FROM user_indexes
  3  WHERE index_name = 'NEW_EMP_NAME_IDX';
```

Practice 15

Throughout this practice Oracle Enterprise Manager can be used if desired. SQL Worksheet can be used instead of SQL*Plus and there are many uses for the Oracle Enterprise Manager console. (Solutions for Oracle Enterprise Manager can be found in Appendix B).

1. Connect as `hr/hr`, drop the `new_employees` table, and create an IOT called `new_employees` in the `hr` schema. Give the table the same columns as the `hr.employees` table. Make the `employee_id` column the primary key and name the primary key index `new_employees_employee_id_pk`.

```
SQL> CONNECT hr/hr
SQL> DROP TABLE new_employees;
SQL> CREATE TABLE new_employees
  2    (employee_id     NUMBER(6),
  3    first_name       VARCHAR2(20),
  4    last_name        VARCHAR2(25),
  5    email            VARCHAR2(25),
  6    phone_number     VARCHAR2(20),
  7    hire_date        DATE,
  8    job_id           VARCHAR2(10),
  9    salary          NUMBER(8,2),
 10    commission_pct   NUMBER (2,2),
 11    manager_id       NUMBER(6),
 12    department_id    NUMBER(4),
 13    CONSTRAINT       new_employees_employee_id_pk
 14      PRIMARY KEY    (employee_id))
 15  ORGANIZATION INDEX;
```

2. Confirm the creation of the table by querying the `user_tables` and the `user_indexes` views

The IOT is a table and so will be found in the `user_tables` view.

```
SQL> SELECT table_name, iot_name, iot_type
  2  FROM user_tables
  3  WHERE table_name LIKE 'NEW_EMPLOYEES%';
```

The IOT is an index and so will be found in the user_indexes view.

```
SQL> SELECT index_name, index_type
  2  FROM user_indexes
  3  WHERE table_name LIKE 'NEW_EMPLOYEES%';
```

3. Populate the `new_employees` table with the rows from the `hr.employees` table.

```
SQL> INSERT INTO new_employees
  2    SELECT *
  3    FROM employees;
```

Practice 15 (continued)

4. Create a secondary B-tree index on the `last_name` column of the `new_employees` table. Place the index in the `indx` tablespace. Name the index `last_name_new_employees_idx`. Collect the statistics for the secondary index.

```
SQL> CREATE INDEX last_name_new_employees_idx
  2  ON new_employees(last_name)
  3  TABLESPACE indx;
SQL> EXECUTE dbms_stats.gather_index_stats -
>      ('HR','LAST_NAME_NEW_EMPLOYEES_IDX');
```

5. Confirm the creation of the index by using the `user_indexes` view in the data dictionary. Query the `index_name`, `index_type`, `blevel`, and `leaf_blocks`.

```
SQL> SELECT index_name, index_type, blevel, leaf_blocks
  2  FROM user_indexes
  3  WHERE index_name = 'LAST_NAME_NEW_EMPLOYEES_IDX';
```

Note: If the values for `blevel` and `leaf_blocks` are null then there were no statistics collected. Confirm that the value of `index_type` is normal.

6. Create a reverse key index on the `department_id` of the `employees_hist` table. Place the index in the `indx` tablespace. Name the index `emp_hist_dept_id_idx`.

```
SQL> CREATE INDEX emp_hist_dept_id_idx
  2  ON employees_hist (department_id)
  3  TABLESPACE indx
  4  REVERSE;
```

7. Confirm the creation of the index and that it is a reverse key index, by querying the `user_indexes` view in the data dictionary. Query the `index_name`, `index_type`, `blevel`, and `leaf_blocks`.

```
SQL> SELECT index_name, index_type, blevel, leaf_blocks
  2  FROM user_indexes
  3  WHERE index_name = 'EMP_HIST_DEPT_ID_IDX';
```

Note: This time the values of `blevel` and `leaf_blocks` should be null, because you did not collect statistics for this index while creating it. Also the value for index type should now be normal/reverse.

Practice 15 (continued)

8. Create a bitmap index on the job_id column of the employees_hist table. Place the index in the indx tablespace. Name the index bitmap_emp_hist_idx.

```
SQL> CREATE BITMAP INDEX bitmap_emp_hist_idx
  2  ON employees_hist (job_id)
  3  TABLESPACE indx;
```

9. Confirm the creation of the index and that it is a bitmapped index by querying the user_indexes view in the data dictionary. Query the index_name, index_type, blevel, and leaf_blocks.

```
SQL> SELECT index_name, index_type
  2  FROM user_indexes
  3  WHERE index_name = 'BITMAP_EMP_HIST_IDX';
```

Practice 16

In this practice you will make use of the AUTOTRACE feature and create the plan_ table table. These are covered in detail in the chapter titled "SQL Statement Tuning." Throughout this practice Oracle Enterprise Manager can be used if desired. SQL Worksheet can be used instead of SQL*Plus and there are many uses for the Oracle Enterprise Manager console. (Solutions for Oracle Enterprise Manager can be found in Appendix B).

1. Connect as sh/sh and confirm that the plan_table table exists. If the table does exist then truncate it, otherwise create the plan_table table using $ORACLE_HOME/rdbms/admin/utlxplan.sql.

```
SQL> CONNECT sh/sh
SQL> DESC plan_table
```

   If the table is found:

```
SQL> TRUNCATE table plan_table;
```

   If the table is not found then:

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

2. Create a materialized view cust_sales having two columns, cust_last_name and the total_sales for that customer. This will mean joining the sales and customers tables using cust_id and grouping the results by cust_last_name. Make sure that query rewrite is enabled on the view.

```
SQL> CREATE MATERIALIZED VIEW cust_sales
  2  ENABLE QUERY REWRITE AS
  3    SELECT c.cust_last_name, sum(s.amount_sold)
  4    FROM sales s, customers c
  5    WHERE c.cust_id = s.cust_id
  6    GROUP BY c.cust_last_name;
```

3. Confirm the creation of the materialized view cust_sales by querying the user_mviews data dictionary view, selecting the columns mview_name, rewrite_enabled and query.

```
SQL> SELECT mview_name, rewrite_enabled, query
  2  FROM user_mviews;
```

   Note: The rewrite_enabled column must have a value of Y in order for the practice on query rewrite to work.

Practice 16 (continued)

4. Set AUTOTRACE to Traceonly Explain, to generate the explain plan for the query `$HOME/STUDENT/LABS/lab16_04.sql`

```
SQL> SET AUTOTRACE Traceonly Explain
SQL> @$HOME/STUDENT/LABS/lab16_04.sql
```

5. Set the QUERY_REWRITE_ENABLED parameter to True for the session and run the same query, `$HOME/STUDENT/LABS/lab16_04.sql`, as in the previous practice. Note the change in the explain plan due to the query rewrite. Set AUTOTRACE to Off and disable query rewrite after the script has completed running.

```
SQL> ALTER SESSION SET QUERY_REWRITE_ENABLED = True;
SQL> @$HOME/STUDENT/LABS/lab16_04.sql
SQL> SET AUTOTRACE Off
SQL> ALTER SESSION SET QUERY_REWRITE_ENABLED = False;
```

Practice 17

The objective of this practice is to use available diagnostic tools to monitor lock contention. You will need to start three sessions in separate windows. Log in as `hr/hr` in two separate sessions (sessions 1 and 3) and as `sys/oracle as sysdba` in another session (session 2). Throughout this practice Oracle Enterprise Manager can be used if desired. SQL Worksheet can be used instead of SQL*Plus and there are many uses for the Oracle Enterprise Manager console. (Solutions for Oracle Enterprise Manager can be found in Appendix B).

1. In session 1 (user `hr/hr`), update the salary by 10% for all employees with a salary < 15000 in the `temp_emps` table. Do not COMMIT.

```
SQL> CONNECT hr/hr
SQL> UPDATE TEMP_EMPS
  2  SET SALARY = SALARY * 1.1
  3  WHERE salary <15000;
```

2. In session 2 connect as `sys/oracle AS sysdba` and check to see whether any locks are being held by querying the `v$lock` view.

```
SQL> CONNECT sys/oracle AS sysdba
SQL> SELECT sid, type, id1, lmode, request
  2  FROM v$lock
  3  WHERE type IN ('TX','TM');
```

3. In session 3 (the session not yet used), connect as `hr/hr` and drop the `temp_emps` table. Does it work?

```
SQL> CONNECT hr/hr
SQL> DROP TABLE hr.temp_emps;
```

Note: The DDL statement requires an exclusive table lock. It cannot obtain it because session 1 already holds a row exclusive table lock on the `temp_emps` table.

4. In session 3 (`hr/hr`), update the salary by 5% for all employees with a salary > 15000 in the `temp_emps` table. Do not COMMIT.

```
SQL> CONNECT hr/hr
SQL> UPDATE temp_emps
  2  SET salary = salary * 1.05
  3  WHERE salary > 15000;
```

Practice 17 (continued)

5. In session 2, check to see what kind of locks are being held on the `temp_emps` table, using the `v$lock` view.

```
SQL> SELECT sid, type, id1, id2, lmode, request
  2  FROM v$lock
  3  WHERE id1 =
  4    (SELECT object_id FROM dba_objects
  5     WHERE object_name = 'TEMP_EMPS'
  6     AND object_type = 'TABLE');
```

6. In session 3, roll back the changes you made and set the `manager_id` column to 10 for all employees who have a salary < 15000.

```
SQL> rollback;
SQL> UPDATE hr.temp_emps
  2  SET MANAGER_id = 10
  3  WHERE salary < 15000;
```

   Note: This session will be hanging, so do not wait for the statement to complete.

7. In session 2, check to see what kind of locks are being held on the `temp_emps` table, using the `v$lock` view.

```
SQL> SELECT sid, type, id1, id2, lmode, request
  2  FROM v$lock
  3  WHERE id1 =
  4    (SELECT object_id
  5     FROM dba_objects
  6     WHERE object_name = 'TEMP_EMPS'
  7     AND object_type = 'TABLE');
```

8. In session 2, run the `$ORACLE_HOME/rdbms/admin/catblock.sql` script. The script will create the `dba_waiters` view, which gives information regarding sessions holding or waiting on a lock. Use this view to determine the session ID for the session that is holding locks. Use this value to query `v$session` to obtain the serial number for the session holding the lock. Then run the `ALTER SYSTEM KILL SESSION` command to release the session holding the lock.

```
SQL> @$ORACLE_HOME/rdbms/admin/catblock.sql
SQL> SELECT waiting_session, holding_session
  2  FROM dba_waiters;
SQL> SELECT sid, serial#, username
  2  FROM v$session
  3  WHERE SID ='&HOLDING_SESSION';
SQL> ALTER SYSTEM KILL SESSION '&SID,&SERIAL_NUM';
```

Practice Solutions Using Enterprise Manager

Practice 2

The goal of this practice is to familiarize you with the different methods of collecting statistical information. Throughout this practice Oracle Enterprise Manager can be used if desired. SQL Worksheet can be used instead of SQL*Plus and there are many uses for the Oracle Enterprise Manager console.

1. Log on as directed by the instructor. If the database is not already started, connect to SQL*Plus using `sys/oracle as sysdba` then start up the instance using the `STARTUP` command. Ensure that the password for user system is set to `oracle`. Check that `TIMED_STATISTICS` has been set to True; if it has not, then set it using the `ALTER SYSTEM` statement.

   Use Enterprise Manager Console - Instance – Configuration
   Check 'All Initialization Parameters' Looking for '`TIMED_STATISTICS`'

   If a value of True is returned, then continue to question 2. If a value of False is returned, then set the `TIMED_STATISTICS` parameter to True using the command:

   Use Enterprise Manager – SQL Worksheet
   ```
   SQL> ALTER SYSTEM SET TIMED_STATISTICS = True
     2    SCOPE = both;
   ```

2. Connect to SQL*Plus as the `system` user and run a command that will create a trace file for this session. Run a query to count the number of rows in the `dba_tables` dictionary view. To locate your new trace file easier, if possible, delete all the trace files in the `USER_DUMP_DEST` directory before running the trace. Remember to disable the trace command after running the query.

   Use Enterprise Manager – SQL Worksheet
   ```
   SQL> CONNECT system/oracle
   SQL> ALTER SESSION SET SQL_TRACE = TRUE;
   SQL> SELECT COUNT(*) FROM dba_tables;
   SQL> ALTER SESSION SET SQL_TRACE = FALSE;
   ```

3. At the operating system level view the resulting trace file located in the directory set by `USER_DUMP_DEST`. Do not try to interpret the content of the trace file, as this is the topic of a later lesson.

   ```
   $cd $HOME/ADMIN/UDUMP
   $ls -l
   -rw-r----- 1 user487 dba 4444 Apr 24 22:28 U487_ora_3270.trc
   ```

Practice 2 (continued)

4. Open two sessions, the first as `hr/hr` and the second as `sys/oracle as sysdba`. From the second session generate a user trace file for the first session using the `dbms_system.set_sql_trace_in_session` procedure. Get the `sid` and `serial#` from `v$session`.

   Open multiple SQL Worksheets. Add the connect string for your database.

   From Session 1
   ```
   SQL> CONNECT hr/hr
   ```
   Change to Session 2
   ```
   SQL> CONNECT sys/oracle as sysdba
   SQL> SELECT username, sid, serial#
     2  FROM v$session
     3  WHERE username = 'HR';
   SQL> BEGIN
     2  dbms_system.set_sql_trace_in_session
     3  (&SID,&SERIALNUM,TRUE);
     4  END;
     5  /
   ```
   Change to Session 1
   ```
   SQL> SELECT * FROM employees;
   ```
   Change to Session 2
   ```
   SQL> BEGIN
     2  dbms_system.set_sql_trace_in_session
     3  (&SID,&SERIALNUM,FALSE);
     4  END;
     5  /
   ```

5. Confirm that the trace file has been created in the directory set by `USER_DUMP_DEST`

   ```
   $cd $HOME/ADMIN/UDUMP
   $ls –l
   -rw-r----- 1 dba01  dba   4444 Apr 24 22:28 dba01_ora_3270.trc
   -rw-r----- 1 dba01  dba  15443 Apr 24 22:42 dba01_ora_3281.trc
   ```

6. Connect to SQL*Plus using `sys/oracle as sysdba` and create a new tablespace (`tools`) to hold the tables and other segments required by Statspack. This tablespace needs to be 200 MB and be dictionary managed (this is not a requirement of Statspack, but will be used later in the course). Name the data file `tools01.dbf` and place it in the `$HOME/ORADATA/u05` directory.
   Note: Dictionary managed is not the default.

   Use Enterprise Manager Console - Storage – Tablespaces

Practice 2 (continued)

7. Confirm and record the amount of free space available within the `tools` tablespace by querying the `dba_free_space` view. Also check that the tablespace is dictionary managed.

   Use Enterprise Manager Console - Storage - Tablespaces

8. Connect using `sys/oracle as sysdba`, then install Statspack using the `spcreate.sql` script located in your `E:\LABS\LABS` directory. Use the following settings when asked by the installation program:
   - User's password = perfstat
   - User's Default Tablespace = TOOLS
   - User's Temporary Tablespace = TEMP

   ```
   SQL> CONNECT sys/oracle AS sysdba
   SQL> @E:\LABS\LABS\spcreate.sql
   ```

9. Query `dba_free_space` to determine the amount of free space left in the `tools` tablespace. The difference between this value and the one recorded in step 7 will be the space required for the initial installation of Statspack.

   Use Enterprise Manager Console - Storage - Tablespaces

   Note: The amount of storage space required will increase in proportion to the amount of information stored within the Statspack tables, that is, the number of snapshots.

   Subtract the value received now, from the value received in step 7 to get the amount of space required to install Statspack

10. Manually collect current statistics using Statspack by running the `snap.sql` script located in E:\LABS\LABS. This will return the `snap_id` for the snapshot just taken, which should be recorded.

    ```
    SQL > @E:\LABS\LABS\snap.sql
    ```

Practice 2 (continued)

11. To have Statspack automatically collect statistics every three minutes execute the `spauto.sql` script located it your `E:\LABS\LABS` directory. Query the database to confirm that the job has been registered using the `user_jobs` view.

```
SQL > @E:\LABS\LABS\spauto.sql
SQL > SELECT job, next_date, next_sec, last_sec
  2   FROM user_jobs;
```

Note: The `spauto.sql` script in the `E:\LABS\LABS` directory has been altered from the `spauto.sql` script shipped with the Oracle database. The alteration has changed the time between snapshots from 1 hour to 3 minutes.

12. After waiting for a period in excess of three minutes query the `stats$snapshot` view to list what snapshots have been collected. There must be at least two snapshots before moving to the next step.

```
SQL> SELECT snap_id,
  2  TO_CHAR(startup_time,' dd Mon "at" HH24:mi:ss')
  3  instart_fm,
  4  TO_CHAR(snap_time,'dd Mon YYYY HH24:mi') snap_date,
  5  snap_level "level"
  6  FROM stats$snapshot
  7  ORDER BY snap_id;
```

Note: If the job scheduler is not working, then check the value of the `JOB_QUEUE_PROCESSES` parameter. The value should be greater than 0.

13. When there are at least two snapshots, start to generate a report. This is performed by using the `spreport.sql` script that is found in the `E:\LABS\LABS` directory. The script lists the snapshot options that are available and then requests the beginning `snap id` and the end `snap id`. The user is then requested to give a filename for the report. It is often best left to the default.

```
SQL> @E:\LABS\LABS\spreport.sql
```

14. Locate the report file in your current directory. Use any text editor to open and examine the report. The first page shows a collection of the most queried statistics.

```
SQL> host notepad sp_X_Y.lst
```

where X is the starting snapshot and Y is the ending snapshot (if the default report filename was used).

15. Connect to the database as a system administrator `sys/oracle as sysdba`.

```
SQL> CONNECT sys/oracle AS sysdba
```

Practice 2 (continued)

16. Query the database to determine what system wait events have been registered since startup using `v$system_event`.

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

Dynamic Performance Views

17. Determine whether there are any sessions actually waiting for resources, using `v$session_wait`.

```
SQL> SELECT sid, event, p1text, wait_time, state
  2  FROM v$session_wait;
```

18. Stop the automatic collection of statistics by removing the job. This is performed by connecting as `perfstat/perfstat` and querying the `user_jobs` view to get the job number. Then execute the `dbms_job.remove` procedure.

```
SQL> CONNECT perfstat/perfstat
SQL> SELECT job, log_user
  2  FROM user_jobs;
SQL> EXECUTE dbms_job.remove(&job_to_remove);
```

19. Connect to your database using Oracle Enterprise Manager. The lecturer will supply the information required to connect to the Oracle Management Server. After you have connected to the database, use Oracle Enterprise Manager to explore the database. Examine items such as the number of tablespaces, users, and tables.

Oracle Classroom Only

If you are in an Oracle classroom you must perform the following four steps that are specific to the Oracle classroom setup:

a. Click the `omsconfig` file update icon on the desktop and enter the name of the UNIX server your class is using. Your instructor will provide the server name. Please note that this is a case-sensitive entry.
b. Open an MSDOS window.
c. At the command prompt enter: oemctl start oms
   Wait for the message: “The Oracleoracle901_homeManagementServer service was started successfully.”

Practice 2 (continued)

Oracle Classroom Only (continued)

d. Close the MSDOS window.
   Start the Oracle Enterprise Manager Console and set the Administrator to `sysman` and the password to `oem_temp`. When prompted, change the password to `oracle`. Select Discover Nodes from the Navigator and enter the host name of the server of your working database.
   i. From the Start menu > Programs > Oracle – OracleHome > Enterprise Manager Console
   ii. Make sure the Login to the Oracle Management Server is selected.
   iii. Administrator: sysman
   iv. Password: oem_temp
   v. Management server is your machine.
   vi. When prompted to change the sysman password to oracle.
   vii. Select Navigator > Discover Nodes from the console menu, or select Discover Nodes from the right mouse shortcut menu to open the Discover Nodes dialog box.
   viii. From the Discovery Wizard: Introduction page, click Next, enter the name of your UNIX database server, and click Next.
   ix. Click Next, give your regular administrator access to your database.
   x. Click Finish, then OK. If your discovery was not successful contact your instructor.

20. From Oracle Enterprise Manager load Oracle Expert and create a new tuning session. Limit the tuning scope to "Check for Instance Optimizations." This is done to reduce the time taken to collect information. Collect a new set of data.

    Note: Do not implement the changes that Oracle Expert recommends, because this will be done during the course.

Practice 3

Throughout this practice Oracle Enterprise Manager can be used if desired. SQL Worksheet can be used instead of SQL*Plus and there are many uses for the Oracle Enterprise Manager console.

1. Connect as `system/oracle` and diagnose database file configuration by querying the `v$datafile`, `v$logfile` and `v$controlfile` dynamic performance views.

   Use Enterprise Manager Console - Storage - Controlfile

   Use Enterprise Manager Console - Storage - Datafiles

   Use Enterprise Manager Console - Storage - Redo Log Groups

2. Diagnose database file usage by querying the `v$filestat` dynamic performance view, combine with `v$datafile` to get the data file names.

   Use Enterprise Manager Performance Manager - I/O - File Statistics

3. Determine whether there are waits for redo log files by querying the `v$system_event` dynamic performance view, where the waiting event is 'log file sync' or 'log file parallel write'.

   Use Enterprise Manager Performance Manager - Wait Events

   Waits for log file sync are indicative of slow disks that store the online logs or unbatched commits. The log file parallel write is much less useful. The reason it is less useful is that this event only shows how often LGWR waits, not how often server processes wait. If LGWR waits without impacting user processes, there is no performance problem. If LGWR waits, it is likely that the log file sync event (mentioned above) will also be evident.

Practice 3 (continued)

4. Connect as `perfstat/perfstat` and diagnose file usage from Statspack.

   a. Generate a Statspack report using `E:\LABS\LABS\spreport.sql`
   b. Locate and open the report file.
   c. Examine the report and search for the "File IO Stats" string.
      Note: On a production database care should be taken in monitoring the disk and controller usage by balancing the workload across all devices. If your examination shows a distinct over utilization of a particular data file, consider resolving the cause of the amount of I/O. For example, investigate the number of full table scans, clustering of files on a specific device and under utilization of indexes. If after this the problem remains then look at placing the data file on a low utilization device.

5. Connect as `system/oracle` and enable checkpoints to be logged in the alert file by setting the value of the `LOG_CHECKPOINTS_TO_ALERT` parameter to True using the `ALTER SYSTEM SET` command.

   Use Enterprise Manager Console - Instance - Configuration - All Initialization Parameters

6. Connect as `sh/sh` and execute the `E:\LABS\LABS\lab03_06.sql` script to provide a workload against the database.

   ```
   SQL> CONNECT sh/sh
   SQL> @E:\LABS\LABS\lab03_06.sql
   ```

7. At the operating system level use the editor to open the alert log file (located in the directory specified by `BACKGROUND_DUMP_DEST`). Then determine the checkpoint frequency for your instance by searching for messages containing the phrase "Completed Checkpoint." The time difference between two consecutive messages is the checkpoint interval.

   Open the alert log file using an editor and search for the line: Completed checkpoint. The line before this will be the time at which the checkpoint occurred. Search for the following checkpoint time and then subtract to get the time between checkpoints.

   Note: On a production system the checkpoint interval should range between 15 minutes to 2 hours. The actual interval is dependant on the type of application and the amount of data manipulation language (DML) activity.

Practice 4

The objective of this practice is to use diagnostic tools to monitor and tune the shared pool. Throughout this practice Oracle Enterprise Manager can be used if desired. SQL Worksheet can be used instead of SQL*Plus and there are many uses for the Oracle Enterprise Manager console.

1. Connect using `sys/oracle as sysdba` and check the size of the shared pool.

   Use Enterprise Manager Console - Instance - Configuration - All Initialization Parameters

2. Connect as `perfstat/perfstat`, execute the `E:\LABS\LABS\snap.sql` script to collect initial snapshot of statistics and note the snapshot number.

```
SQL> CONNECT perfstat/perfstat
SQL> @E:\LABS\LABS\snap.sql
```

3. To simulate user activity against the database open two operating system sessions. In session 1 connect as `hr/hr` and run the `E:\LABS\LABS\lab04_03_1.sql` script. In the second session connect as `hr/hr` and run the `E:\LABS\LABS\lab04_03_2.sql` script.

   Open multiple SQL Worksheets. Add the connect string for your database.

   In session 1

```
SQL> CONNECT hr/hr
SQL> @E:\LABS\LABS\lab04_03_1.sql
```

   In session 2

```
SQL> CONNECT hr/hr
SQL> @E:\LABS\LABS\lab04_03_2.sql
```

4. Connect as `system/oracle` and measure the pin-to-reload ratio for the library cache by querying `v$librarycache`. Determine whether it is a good ratio or not.

   Use Enterprise Manager Performance Manager - Memory

Practice 4 (continued)

5. Connect as `system/oracle` and measure the get-hit ratio for the data dictionary cache by querying `v$rowcache`. Determine whether it is a good ratio or not using the dynamic view.

   Use Enterprise Manager Performance Manager - Memory

If `GETMISSES` are lower than 15% of the `GETS`, it is a good ratio.

6. Connect as `perfstat/perfstat` and run the `E:\LABS\LABS\snap.sql` script to collect a statistic snapshot and obtain the snapshot number. Record this number.

```
SQL> CONNECT perfstat/perfstat
SQL> @E:\LABS\LABS\snap.sql
```

7. As `perfstat/perfstat` obtain the statistics report between the two recorded snapshot IDs (from questions 2 and 6) by running the `E:\LABS\LABS\spreport.sql` script.

```
SQL> CONNECT perfstat/perfstat
SQL> @E:\LABS\LABS\spreport.sql
```

The following is an example of using a beginning snapshot_id of 3 and an ending snapshot_id of 5.

```
Specify the Begin and End Snapshot Ids
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Enter value for begin_snap:   3

Enter value for end_snap: 5
End   Snapshot Id specified: 5


Specify the Report Name
~~~~~~~~~~~~~~~~~~~~~~~
The default report file name is sp_3_5. To use this name,
press <return> to continue, otherwise enter an alternative.
Enter value for report_name:
```

You can also determine an appropriate size for the Shared Pool by using:

Enterprise Manager Console - Instance - Configuration - Memory - Advice

Practice 4 (continued)

8. Analyze the generated report in the current directory. What would you consider doing if the library hit ratio (found under the heading "Instance Efficiency Percentages) is less than 98%?

   Increase the `SHARED_POOL_SIZE` parameter.

9. Connect as `system/oracle` and determine which packages, procedures and triggers are pinned in the shared pool by querying `v$db_object_cache`.

```
SQL> CONNECT system/oracle
SQL> SELECT name, type, kept
  2  FROM v$db_object_cache
  3  WHERE type IN
  4  ('PACKAGE', 'PROCEDURE', 'TRIGGER', 'PACKAGE BODY');
```

10. Connect using `sys/oracle as sysdba` and pin one of the Oracle supplied packages that needs to be kept in memory, such as sys.standard using the `dbms_shared_pool.keep` procedure , that is created by running the `$ORACLE_HOME/rdbms/admin/dbmspool.sql` script.

```
SQL> CONNECT sys/oracle AS SYSDBA
SQL> @?/rdbms/admin/dbmspool
SQL> EXECUTE dbms_shared_pool.keep('SYS.STANDARD');
SQL> SELECT distinct owner, name
  2  FROM v$db_object_cache
  3  WHERE kept='YES'
  4  AND name LIKE '%STAND%';
```

   Note: After you complete the other steps in this practice and before you move on to the next practice, use the `unkeep` procedure to unpin the object you pinned in the shared pool. This will help avoid memory problems in later practices.

11. Determine the amount of session memory used by your session by querying the `v$mystat` view. Limit the output by including the clause:

```
WHERE name = 'session uga memory'

SQL> SELECT a.name, b.value
  2  FROM v$statname a, v$mystat b
  3  WHERE a.statistic# = b.statistic#
  4  AND name = 'session uga memory';
```

   Note: Since you are not using the Oracle Shared Server configuration this memory resides outside the SGA.

Practice 4 (continued)

12. Determine the amount of session memory used for all sessions, using `v$sesstat` and `v$statname` views:

```
SQL> SELECT SUM(value)||' bytes' "Total session memory"
  2  FROM v$sesstat, v$statname
  3  WHERE name = 'session uga memory'
  4  AND v$sesstat.statistic# = v$statname.statistic#;
```

Practice 5

The objective of this practice is to use available diagnostic tools to monitor and tune the database buffer cache. Throughout this practice Oracle Enterprise Manager can be used if desired. SQL Worksheet can be used instead of SQL*Plus and there are many uses for the Oracle Enterprise Manager console.

1. Connect as `perfstat/perfstat` and run a statistic snapshot. Make a note of the snapshot number. The snap shot can be taken by running the `E:\LABS\LABS\snap.sql` script file.

```
SQL> CONNECT perfstat/perfstat
SQL> @E:\LABS\LABS\snap.sql
```

2. To simulate user activity against the database, connect as the `hr/hr` user and run the `lab05_02.sql` script.

```
SQL> CONNECT hr/hr
SQL> @E:\LABS\LABS\lab05_02.sql
```

3. Connect as `system/oracle` and measure the hit ratio for the database buffer cache using the `v$sysstat` view. Determine whether it is a good ratio or not.

   Use Enterprise Manager Performance Manager - Database Instance - Instance Efficiency Statistics

4. Connect as `perfstat/perfstat` and run a statistic snapshot. Make a note of the snapshot number. The snapshot can be taken by running the `E:\LABS\LABS\snap.sql` script file.

```
SQL> CONNECT perfstat/perfstat
SQL> @E:\LABS\LABS\snap.sql
```

5. Use the report from Statspack between the last two snapshots to check the buffer cache hit ratio, using the `E:\LABS\LABS\spreport.sql` script. Then analyze the buffer hit % in the "Instance Efficiency Percentages" section.

```
SQL> @E:\LABS\LABS\spreport.sql
```

Practice 5 (continued)

Note: On a production database if the ratio is bad, add new buffers, run steps 2 to 5 and examine the new ratio to verify that the ratio has improved. If the ratio is good, remove buffers, run steps 2 to 5 and verify if the ratio is still good.

6. Connect as `system/oracle` and determine the size of the `temp_emps` table in the `hr` schema that you want to place in the keep buffer pool. Do this by using the `dbms_stats.gather_table_stats` procedure and then query the `blocks` column of the `dba_tables` view for the `temp_emps` table.

```
SQL> CONNECT system/oracle
SQL> EXECUTE dbms_stats.gather_table_stats -
>              ('HR','TEMP_EMPS');
SQL> SELECT table_name , blocks
  2  FROM dba_tables
  3  WHERE table_name IN ('TEMP_EMPS');
```

7. Keep `temp_emps` in the keep pool. Use the `ALTER SYSTEM` command to set `DB_KEEP_CACHE_SIZE` to 4 MB for the keep pool. Limit the scope of this command to the spfile.

   Use Enterprise Manager Console - Instance - Configuration - All Initialization Parameters

8. For the keep pool to be allocated the database needs to be restarted. You will need to be connected as a `sysdba` user to perform this task.

   Shut down and start up the instance using:
   Enterprise Manager Console - Instance - Configuration

9. Connect as `system/oracle` and enable the `temp_emps` table in the `hr` schema for caching in the keep pool, using the storage clause of the `ALTER TABLE` command.

```
SQL> CONNECT system/oracle
SQL> ALTER TABLE hr.temp_emps
  2  STORAGE (BUFFER_POOL Keep);

SQL> SELECT table_name, buffer_pool
  2  FROM dba_tables
  3  WHERE buffer_pool = 'KEEP';
```

10. Connect as `hr/hr` and run the `E:\LABS\LABS\lab05_10.sql` script. This will execute a query against the `temp_emps` table in the `hr` schema.

```
SQL> CONNECT hr/hr
SQL> @E:\LABS\LABS\lab05_10.sql
```

Practice 5 (continued)

11. Connect using `sys/oracle as sysdba` and check for the hit ratio in different buffer pools, using the `v$buffer_pool_statistics` view.

    Use Enterprise Manager Performance Manager - Database Instance - Instance Efficiency Statistics

Practice 6

Throughout this practice Oracle Enterprise Manager can be used if desired. SQL Worksheet can be used instead of SQL*Plus and there are many uses for the Oracle Enterprise Manager console.

1. Connect as `sys/oracle AS sysdba` and, without restarting the instance, resize the `DB_CACHE_SIZE` to 12 Mb. Limit the effect of this command to memory, so as not to modify the spfile.

   Use Enterprise Manager Console - Instance - Configuration - All Initialization Parameters

   Note: This will encounter an error because the total SGA size will be bigger than `SGA_MAX_SIZE`. To overcome this you will must either change the value of `SGA_MAX_SIZE` and restart the instance (which is what dynamic allocation is meant to avoid) or resize a component, thus making memory available for the increase in the buffer cache.

2. Reduce the memory used by the shared pool. Limit the effect of this command to memory, so as not to modify the spfile.

   Use Enterprise Manager Console - Instance - Configuration - All Initialization Parameters

3. Without restarting the instance, resize the `DB_CACHE_SIZE` to 12 Mb. Limit the effect of this command to memory, so as not to modify the spfile.

   Use Enterprise Manager Console - Instance - Configuration - All Initialization Parameters

   Note: This time the memory is available so the command will be executed.

4. To return the SGA to the original configuration, restart the instance. You will need to be connected as a `sysdba` user to perform this task.

   Use Enterprise Manager Console - Instance - Configuration

Practice 7

Throughout this practice Oracle Enterprise Manager can be used if desired. SQL Worksheet can be used instead of SQL*Plus and there are many uses for the Oracle Enterprise Manager console.

1. Connect as `perfstat/perfstat` and collect a snapshot of the current statistics by running the script `E:\LABS\LABS\snap.sql`. Record the snapshot ID for later use.

```
SQL> CONNECT perfstat/perfstat
SQL> @E:\LABS\LABS\snap;
```

2. Connect as user `sh/sh` and run the `E:\LABS\LABS\lab07_02.sql` script in the `E:\LABS\LABS` directory to put a workload on the database.

```
SQL> CONNECT sh/sh
SQL> @E:\LABS\LABS\lab07_02.sql
```

3. Connect as `system/oracle` and query the `v$sysstat` view to determine whether there are space requests for the redo log buffer.

   Use Enterprise Manager Performance Manager - Wait Events

4. Connect as `perfstat/perfstat` and collect another set of statistics using the `E:\LABS\LABS\snap.sql` script. Then use `E:\LABS\LABS\spreport.sql` to generate a report using the two snapshot IDs that you have collected. From the list of snapshots select a beginning and end value. The beginning is the value recorded in step 1 and the end value step 4. Record the name of the report file. View the generated file using an editor and locate the "log buffer space" statistic.

```
SQL> CONNECT perfstat/perfstat
SQL> @E:\LABS\LABS\snap.sql;
SQL> @E:\LABS\LABS\spreport.sql
```

5. Connect as `sys/oracle AS sysdba` and increase the size of the redo log buffer in the spfile by changing the value of the `LOG_BUFFER` parameter. Since this parameter is static you must specify spfile.

   Use Enterprise Manager Console - Instance - Configuration - All Initialization Parameters

6. To have the new value for the `LOG_BUFFER` take effect, you must restart the instance. Then confirm that the change has occurred.

   Use Enterprise Manager Console - Instance - Configuration

Practice 9

Throughout this practice Oracle Enterprise Manager can be used if desired. SQL Worksheet can be used instead of SQL*Plus and there are many uses for the Oracle Enterprise Manager console.

1. Set the database to use the manual sort option by changing the value of the `WORKAREA_SIZE_POLICY` parameter to Manual. Set the `SORT_AREA_SIZE` parameter to 512 bytes.

   Use Enterprise Manager Console - Instance - Configuration - All Initialization Parameters

2. For the new values of the `WORKAREA_SIZE_POLICY` and `SORT_AREA_SIZE` parameters to take effect, you must restart the instance. Then query the `v$sysstat` view and record the value for memory sorts and disk sorts.

   Use Enterprise Manager Console - Instance - Configuration

   ```
   SQL> SELECT name, value
     2  FROM v$sysstat
     3  WHERE  name LIKE 'sorts%';
   ```

   Note: The statistics in `v$sysstat` are collected from startup. If you need to get accurate statistics per statement, then you must record statistics before the statement runs and again afterwards. Subtracting the two values gives the statistics for the statement.

3. To perform a sort on the database that will have sorts to disk connect as `sh/sh` and execute the `E:\LABS\LABS\lab09_03.sql` script.

   ```
   SQL> CONNECT sh/sh
   SQL> @E:\LABS\LABS\lab09_03.sql;
   ```

   Note: If this script fails due to a lack of free space in the `temp` tablespace then connect as `system/oracle` and resize the temporary tablespace.

   Use Enterprise Manager Console - Storage - Datafiles

4. Connect as `system/oracle`, query the `v$sysstat` view again, and record the value for sorts (memory) and sorts (disk). Subtract the values from the recorded value in question 2. If the ratio of Disk to Memory sorts is greater than 5% then increase the sort area available.

Practice 9 (continued)

```
SQL> CONNECT system/oracle
SQL> SELECT name, value
  2  FROM v$sysstat
  3  WHERE  name LIKE 'sorts%';
```

5. Connect as `system/oracle` and query the `tablespace_name`, `max_sort_size`, and `max_used_size` columns from the `v$sort_segment` view.

```
SQL> CONNECT system/oracle
SQL> SELECT tablespace_name, max_sort_size,
  2    max_used_size
  3  FROM v$sort_segment;
```

Note: The `used_extents` and `free_extents` columns from `v$sort_segment` are also useful in monitoring the temporary tablespace. If the view contains no rows, it means that all sort operations since startup have completed in memory.

6. To decrease the number of sorts going to a temporary tablespace, increase the value of the `SORT_AREA_SIZE` parameter to 512000 using the "`ALTER SESSION`" command.

```
SQL> ALTER SESSION SET SORT_AREA_SIZE = 512000;
```

7. Connect as `system/oracle` and configure the parameters for automatic PGA memory allocation using the `ALTER SYSTEM` command. Use the values Auto for `WORKAREA_SIZE_POLICY` and 10M for `PGA_AGGREGATE_TARGET`).

   Use Enterprise Manager Console - Instance - Configuration - All Initialization Parameters

Practice 11

The objective of this practice is to familiarize you with SQL statement execution plans and to interpret the formatted output of a trace file generated using SQL Trace and the formatted output generated by TKPROF. Throughout this practice Oracle Enterprise Manager can be used if desired. SQL Worksheet can be used instead of SQL*Plus and there are many uses for the Oracle Enterprise Manager console.

1. Connect as `hr/hr` and create the `plan_table` table under the `hr` schema, if it is not already created, by running the @%ORACLE_HOME%\rdbms\admin\`utlxplan.sql` script.

   ```
   SQL> CONNECT hr/hr
   SQL> @%ORACLE_HOME%\rdbms\admin\utlxplan.sql
   ```

   Note: If `plan_table` already exists and holds rows then truncate the table.

2. Set the optimizer mode to rule based using the `ALTER SESSION` command and generate the explain plan for the statement `E:\LABS\LABS\lab11_02.sql`. View the generated plan by querying object name, operation, option, and optimizer from the `plan_table` table.

   ```
   SQL> ALTER SESSION SET OPTIMIZER_GOAL = Rule;
   SQL> EXPLAIN PLAN FOR
     2  @E:\LABS\LABS\lab11_02.sql
   SQL> SELECT object_name, operation, options, optimizer
     2  FROM plan_table;
   ```

3. Truncate the `plan_table` table. Change the optimizer mode to cost based by setting the value to All_Rows and rerun the explain plan for `E:\LABS\LABS\lab11_02.sql`. Notice that the optimizer mode and the explain plan have changed.

   ```
   SQL> TRUNCATE TABLE plan_table;
   SQL> ALTER SESSION SET OPTIMIZER_GOAL = all_rows;
   SQL> EXPLAIN PLAN FOR
     2  @E:\LABS\LABS\lab11_02.sql
   SQL> SELECT object_name, operation, options, optimizer
     2  FROM plan_table;
   ```

   Note: Although exactly the same scripts are being run, due to the different optimizer settings, different explain paths are found. With rule based, one of the rules is to use any index that is on the columns in the where clause. By using cost based optimizer mode, the server has been able to determine that it will be faster to just perform a full table scan, due to the number of rows being returned by the script.

Practice 11 (continued)

4. Truncate the `plan_table` table and set the optimizer mode to Rule by using the `ALTER SESSION` command. This time generate the explain plan for the `E:\LABS\LABS\lab11_04.sql` script. Examine the script which is a copy of `E:\LABS\LABS\lab11_02.sql` except it changes the line "`SELECT *` " to include a hint `/*+ all_rows*/` for the optimizer. View the generated execution plan by querying object name, operation, option, and optimizer from `plan_table` table.

```
SQL> TRUNCATE TABLE plan_table;
SQL> ALTER SESSION SET OPTIMIZER_GOAL = Rule;
SQL> EXPLAIN PLAN FOR
  2  @E:\LABS\LABS\lab11_04.sql
SQL> SELECT object_name, operation, options, optimizer
  2  FROM plan_table;
```

5. Exit out of SQL*Plus, change the directory to `$HOME/ADMIN/UDUMP` and delete all the trace files already generated.

```
SQL> EXIT
$ cd $HOME/ADMIN/UDUMP
$ rm *.trc
```

Note: this step is performed only to make it easier to find the trace file generated. It is not a requirement of SQL Trace.

6. Connect as `sh/sh` and enable SQL Trace, using the `ALTER SESSION` command, to collect statistics for the script, `E:\LABS\LABS\lab11_06.sql`. Run the script. After the script has completed, disable SQL Trace, then format your trace file using `TKPROF`. Use the options `SYS=NO` and `EXPLAIN= sh/sh`. Name the file `myfile.txt`.

```
SQL> CONNECT sh/sh
SQL> ALTER SESSION SET SQL_TRACE = True;
SQL> @E:\LABS\LABS\lab11_06.sql
SQL> ALTER SESSION SET SQL_TRACE = False;
$ cd $HOME/ADMIN/UDUMP
$ ls -l
-rw-r----- 1 user457 dba 2180 May 4 00:27 ser457_ora_10424.trc
$ tkprof user457_ora_10424.trc myfile.txt explain=sh/sh sys=no
```

7. View the output file `myfile.txt` and note the CPU, current and query figures for the fetch phase. Do not spend time analyzing the contents of this file as the only objective here is to become familiar and comfortable with running `TKPROF` and `SQL*Trace`.

```
$ more myfile.txt
```

Practice 12

The objective of this practice is to familiarize you with the `dbms_stats` package. Throughout this practice Oracle Enterprise Manager can be used if desired. SQL Worksheet can be used instead of SQL*Plus and there are many uses for the Oracle Enterprise Manager console.

1. Connect as `hr/hr` and create a table `new_employees` as a copy of the `employees` table. Gather statistics on the `new_employees` table and determine the current number of rows in the `new_employees` table. Record the number of rows for comparison later.

```
SQL> CONNECT hr/hr
SQL> CREATE TABLE new_employees
  2    AS SELECT *
  3       FROM employees;
SQL> EXECUTE -
>    dbms_stats.gather_table_stats ('HR','NEW_EMPLOYEES');
SQL> SELECT table_name, num_rows
  2  FROM user_tables
  3  WHERE table_name = 'NEW_EMPLOYEES';
```

2. Increase the size of the `new_employees` table by using the `lab12_02.sql` script.

```
SQL> @E:\LABS\LABS\lab12_02.sql
```

3. Confirm that the statistics have not been changed in the data dictionary by re-issuing the same statement as in question 1.

```
SQL> SELECT table_name, num_rows
  2  FROM user_tables
  3  WHERE table_name = 'NEW_EMPLOYEES';
```

4. Connect `hr/hr` and gather statistics for all objects under the `hr` schema using the `dbms_stats` package. While gathering the new statistics save the current statistics in a table named `stats`.

   a. Connect as `hr/hr` and create a table to hold statistics in that schema.
   ```
   SQL> CONNECT hr/hr
   SQL> execute dbms_stats.create_stat_table('HR','STATS');
   ```
   b. Save the current schema statistics into your local statistics table.
   ```
   SQL> execute dbms_stats.export_schema_stats('HR','STATS');
   ```
   c. Analyze all objects under the `hr` schema.
   ```
   SQL> execute dbms_stats.gather_schema_stats('HR');
   ```

Practice 12 (continued)

5. Determine that the current number of rows in the employees table has been updated in the data dictionary. This should be twice the number of rows recorded in question 1.

```
SQL> SELECT table_name, num_rows
  2  FROM user_tables
  3  WHERE table_name = 'NEW_EMPLOYEES';
```

6. Remove all schema statistics from the dictionary and restore the original statistics you saved in step 4b.

```
SQL> execute dbms_stats.delete_schema_stats('HR');
SQL> execute dbms_stats.import_schema_stats('HR','STATS');
```

7. Confirm that the number of rows in the `employees` table recorded in the data dictionary has returned to the previous value collected in question 1.

```
SQL> SELECT table_name, num_rows
  2  FROM user_tables
  3  WHERE table_name = 'NEW_EMPLOYEES';
```

Practice 13

Throughout this practice Oracle Enterprise Manager can be used if desired. SQL Worksheet can be used instead of SQL*Plus and there are many uses for the Oracle Enterprise Manager console.

1. Connect using `sys/oracle AS sysdba` and query the `tablespace_name` and `extent_management` columns of `dba_tablespaces` to determine which tablespaces are locally managed and which are dictionary managed. Record which tablespaces are dictionary managed.

   Use Enterprise Manager Console - Storage - Tablespaces

2. Alter the `hr` user to have the `tools` tablespace as the default.

   Use Enterprise Manager Console - Security - Users - HR

3. Examine the `v$system_event` view and note the total waits for the statistic enqueue.

```
SQL> SELECT event, total_waits
  2  FROM v$system_event
  3  WHERE event = 'enqueue';
```

   Note: On a production system you would be more likely to pick up the contention through the Statspack report.

4. Also examine the `v$enqueue_`stat view for `eq_type` 'ST' to determine the `total_wait#` for the ST enqueue, which is the space management enqueue.

```
SQL> SELECT *
  2  FROM v$enqueue_stat
  3  WHERE eq_type = 'ST';
```

5. Exit out of the SQL*Plus session and change directory to `E:\LABS\LABS`. Run the `lab13_04.bat` script from the operating system prompt. This script will log five users onto the database simultaneously and then each user creates and drops tables. The tables each have many extents. The script must be run from the `E:\LABS\LABS` directory or it will fail.

```
$ cd E:\LABS\LABS
$ lab13_04.bat
```

Practice 13 (continued)

6. Connect as `system/oracle` and again examine the `v$enqueue_stat` view for `eq_type` 'ST' to determine the value of `total_wait#` for the ST enqueue, which is the space management enqueue.

```
$ SQL*Plus system/oracle
SQL> SELECT *
  2  FROM v$enqueue_stat
  3  WHERE eq_type = 'ST';
```

Note: Record the difference in the number of waits for the ST enqueue for extent management using a dictionary managed tablespace. This value is found by subtracting the first wait value (from practice 13-04) from the second wait value (from practice 13-06).

7. Create a new locally managed tablespace `test`, name the data file `test01.dbf` and place it in the directory `$HOME/ORADATA/u06`. Set the size to 120 MB and a uniform extent size of 20 KB.

   Use Enterprise Manager Console - Storage - Tablespaces

8. Alter the default tablespace of the `hr` user to `test`.

   Use Enterprise Manager Console - Security - Users - HR

Note: The same steps are covered again. This time you are looking for the number of waits for the ST enqueue caused by locally managed tablespaces.

9. Examine and record the initial `total_wait#` for 'ST' in the `v$enqueue_stat` view.

```
SQL> SELECT *
  2  FROM v$enqueue_stat
  3  WHERE eq_type = 'ST';
```

10. Exit out of the SQL*Plus session and change directory to `E:\LABS\LABS`. Run the `lab13_04.bat` script from the operating system prompt. This script will log five users onto the database simultaneously and then each user creates and drops tables. The tables each have many extents. The script must be run from the `E:\LABS\LABS` directory or it will fail.

```
$ cd E:\LABS\LABS
$ ./lab13_04.bat
```

Practice 13 (continued)

11. Again examine and record the final `total_wait#` for 'ST' in the `v$enqueue_stat` view.

```
SQL> SELECT *
  2  FROM v$enqueue_stat
  3  WHERE eq_type = 'ST';
```

Note: Record the difference in the `total_wait#` for the ST enqueue for extent management using a locally managed tablespace. This value is found by subtracting the first wait value (from practice 13-09) from the second wait value (from practice 13-11). Compare the two results for the different tablespaces. The locally managed tablespace would be far less contentious with extent management because it is managing the space within the tablespace itself.

12. Connect as the `hr/hr` user and run the `E:\LABS\LABS\lab13_12.sql` script. This will create a similar table (`new_emp`) as the `employees` table but with PCTFREE = 0. The table is then populated with data from the `employees` table.

```
SQL> CONNECT hr/hr
SQL> E:\LABS\LABS\lab13_12.sql;
```

13. Run `ANALYZE` on the `new_emp` table and query the `dba_tables` view to determine the value of `chain_cnt` for the `new_emp` table. Record this value.

```
SQL> ANALYZE TABLE new_emp COMPUTE STATISTICS;
SQL> SELECT table_name, chain_cnt
  2  FROM user_tables
  3  WHERE table_name = 'NEW_EMP';
```

14. Create an index called `new_emp_name_idx` on the `last_name` column of the `new_emp` table. Place the index in the tablespace `indx`. Then confirm the index's status in the `user_indexes` view.

```
SQL> CREATE INDEX new_emp_name_idx ON new_emp(last_name)
  2  TABLESPACE indx;
SQL> SELECT index_name, status
  2  FROM user_indexes
  3  WHERE index_name = 'NEW_EMP_NAME_IDX';
```

15. Run the `E:\LABS\LABS\lab13_15.sql` script, which will update the rows of the `new_emp` table. Analyze the `new_emp` table again and query the `user_tables` view to get the new value of `chain_cnt` Record this value. Also check the status of the `new_emp_name_idx` index.

Practice 13 (continued)

```
SQL> @E:\LABS\LABS\lab13_15.sql
SQL> ANALYZE TABLE new_emp COMPUTE STATISTICS;
SQL> SELECT table_name, chain_cnt
  2  FROM user_tables
  3  WHERE table_name = 'NEW_EMP';
SQL> SELECT index_name, status
  2  FROM user_indexes
  3  WHERE index_name = 'NEW_EMP_NAME_IDX';
```

16. Resolve the migration caused by the previous update, by using the `ALTER TABLE MOVE` command. This will cause the index to become unusable and should be rebuilt using the `ALTER INDEX REBUILD` command before reanalyzing the `new_emp` table. Confirm that the migration has been resolved by querying `chain_cnt` column in the `user_tables` view and confirm that the index is valid by querying the `user_indexes` view.

```
SQL> ALTER TABLE new_emp MOVE
  2  TABLESPACE users;
SQL> ALTER INDEX new_emp_name_idx REBUILD;
SQL> ANALYZE TABLE new_emp COMPUTE STATISTICS;
SQL> SELECT table_name, blocks, empty_blocks, chain_cnt
  2  FROM user_tables
  3  WHERE table_name = 'NEW_EMP';
SQL> SELECT index_name, status
  2  FROM user_indexes
  3  WHERE index_name = 'NEW_EMP_NAME_IDX';
```

Practice 15

Throughout this practice Oracle Enterprise Manager can be used if desired. SQL Worksheet can be used instead of SQL*Plus and there are many uses for the Oracle Enterprise Manager console.

1. Connect as `hr/hr`, drop the `new_employees` table and create an IOT called `new_employees` in the `hr` schema. Give the table the same columns as the `hr.employees` table. Make the `employee_id` column the primary key and name the primary key index `new_employees_employee_id_pk`.

```
SQL> CONNECT hr/hr
SQL> DROP TABLE new_employees;
SQL> CREATE TABLE new_employees
  2    (employee_id      NUMBER(6),
  3    first_name        VARCHAR2(20),
  4    last_name         VARCHAR2(25),
  5    email             VARCHAR2(25),
  6    phone_number      VARCHAR2(20),
  7    hire_date         DATE,
  8    job_id            VARCHAR2(10),
  9    salary           NUMBER(8,2),
 10    commission_pct    NUMBER (2,2),
 11    manager_id        NUMBER(6),
 12    department_id     NUMBER(4),
 13    CONSTRAINT        new_employees_employee_id_pk
 14      PRIMARY KEY     (employee_id))
 15  ORGANIZATION INDEX;
```

2. Confirm the creation of the table by querying the `user_tables` and the `user_indexes` views.

   The IOT is a table and so will be found in the `user_tables` view.

```
SQL> SELECT table_name, iot_name, iot_type
  2  FROM user_tables
  3  WHERE table_name LIKE 'NEW_EMPLOYEES%';
```

   The IOT is an index and so will be found in the user_indexes view.

```
SQL> SELECT index_name, index_type
  2  FROM user_indexes
  3  WHERE table_name LIKE 'NEW_EMPLOYEES%';
```

3. Populate the `new_employees` table with the rows from the `hr.employees` table.

```
SQL> INSERT INTO new_employees
  2    SELECT *
  3    FROM employees;
```

Practice 15 (continued)

4. Create a secondary B-tree index on the `last_name` column of the `new_employees` table. Place the index in the `indx` tablespace. Name the index `last_name_new_employees_idx`. Collect the statistics for the secondary index.

```
SQL> CREATE INDEX last_name_new_employees_idx
  2  ON new_employees(last_name)
  3  TABLESPACE indx;
SQL> EXECUTE dbms_stats.gather_index_stats -
>      ('HR','LAST_NAME_NEW_EMPLOYEES_IDX');
```

5. Confirm the creation of the index by using the `user_indexes` view in the data dictionary. Query the `index_name`, `index_type`, `blevel` and `leaf_blocks`.

```
SQL> SELECT index_name, index_type, blevel, leaf_blocks
  2  FROM user_indexes
  3  WHERE index_name = 'LAST_NAME_NEW_EMPLOYEES_IDX';
```

Note: If the values for `blevel` and `leaf_blocks` are null then there were no statistics collected. Confirm that the value of `index_type` is normal.

6. Create a reverse key index on the `department_id` of the `employees_hist` table. Place the index in the `indx` tablespace. Name the index `emp_hist_dept_id_idx`.

```
SQL> CREATE INDEX emp_hist_dept_id_idx
  2  ON employees_hist (department_id)
  3  TABLESPACE indx
  4  REVERSE;
```

7. Confirm the creation of the index and that it is a reverse key index, by querying the `user_indexes` view in the data dictionary. Query the `index_name`, `index_type`, `blevel` and `leaf_blocks`.

```
SQL> SELECT index_name, index_type, blevel, leaf_blocks
  2  FROM user_indexes
  3  WHERE index_name = 'EMP_HIST_DEPT_ID_IDX';
```

Note: This time the values of `blevel` and `leaf_blocks` should be null, because you did not collect statistics for this index while creating it. Also the value for index type should now be normal/reverse.

Practice 15 (continued)

8. Create a bitmap index on the `job_id` column of the `employees_hist` table. Place the index in the `indx` tablespace. Name the index `bitmap_emp_hist_idx`.

```
SQL> CREATE BITMAP INDEX bitmap_emp_hist_idx
  2  ON employees_hist (job_id)
  3  TABLESPACE indx;
```

9. Confirm the creation of the index and that it is a bitmapped index by querying the `user_indexes` view in the data dictionary. Query the `index_name`, `index_type`, `blevel`, and `leaf_blocks`.

```
SQL> SELECT index_name, index_type
  2  FROM user_indexes
  3  WHERE index_name = 'BITMAP_EMP_HIST_IDX';
```

Practice 16

In this practice you will make use of the AUTOTRACE feature and create the plan_ table table. These are covered in detail in the chapter titled "SQL Statement Tuning." Throughout this practice Oracle Enterprise Manager can be used if desired. SQL Worksheet can be used instead of SQL*Plus and there are many uses for the Oracle Enterprise Manager console.

1. Connect as sh/sh and confirm that the plan_table table exists. If the table does exist then truncate it, otherwise create the plan_table table using $ORACLE_HOME/rdbms/admin/utlxplan.sql.

   ```
   SQL> CONNECT sh/sh
   SQL> DESC plan_table
   ```

   If the table is found:

   ```
   SQL> TRUNCATE table plan_table;
   ```

   If the table is not found then:

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

2. Create a materialized view cust_sales having two columns, cust_last_name and the total_sales for that customer. This will mean joining the sales and customers tables using cust_id and grouping the results by cust_last_name. Make sure that query rewrite is enabled on the view.

   ```
   SQL> CREATE MATERIALIZED VIEW cust_sales
     2  ENABLE QUERY REWRITE AS
     3    SELECT c.cust_last_name, sum(s.amount_sold)
     4    FROM sales s, customers c
     5    WHERE c.cust_id = s.cust_id
     6    GROUP BY c.cust_last_name;
   ```

3. Confirm the creation of the materialized view cust_sales by querying the user_mviews data dictionary view, selecting the columns mview_name, rewrite_enabled and query.

   ```
   SQL> SELECT mview_name, rewrite_enabled, query
     2  FROM user_mviews;
   ```

   Note: The rewrite_enabled column must have a value of Y in order for the practice on query rewrite to work.

Practice 16 (continued)

4. Set AUTOTRACE to Traceonly Explain, to generate the explain plan for the query E:\LABS\LABS\lab16_04.sql

```
SQL> SET AUTOTRACE Traceonly Explain
SQL> @E:\LABS\LABS\lab16_04.sql
```

5. Set the QUERY_REWRITE_ENABLED parameter to True for the session and run the same query, E:\LABS\LABS\lab16_04.sql, as in the previous practice. Note the change in the explain plan due to the query rewrite. Set AUTOTRACE to Off and disable query rewrite after the script has completed running.

```
SQL> ALTER SESSION SET QUERY_REWRITE_ENABLED = True;
SQL> @E:\LABS\LABS\lab16_04.sql
SQL> SET AUTOTRACE Off
SQL> ALTER SESSION SET QUERY_REWRITE_ENABLED = False;
```

Practice 17

The objective of this practice is to use available diagnostic tools to monitor lock contention. You will need to start three sessions in separate windows. Log in as hr/hr in two separate sessions (sessions 1 and 3) and as sys/oracle as sysdba in another session (session 2). Throughout this practice Oracle Enterprise Manager can be used if desired. SQL Worksheet can be used instead of SQL*Plus and there are many uses for the Oracle Enterprise Manager console.

1. In session 1 (user `hr/hr`), update the salary by 10% for all employees with a salary < 15000 in the `temp_emps` table. Do not COMMIT.

```
SQL> CONNECT hr/hr
SQL> UPDATE TEMP_EMPS
  2  SET SALARY = SALARY * 1.1
  3  WHERE salary <15000;
```

2. In session 2 connect as `sys/oracle AS sysdba` and check to see if any locks are being held by querying the `v$lock` view.

```
SQL> CONNECT sys/oracle AS sysdba
SQL> SELECT sid, type, id1, lmode, request
  2  FROM v$lock
  3  WHERE type IN ('TX','TM');
```

3. In session 3 (the session not yet used), connect as `hr/hr` and drop the `temp_emps` table. Does it work?

```
SQL> CONNECT hr/hr
SQL> DROP TABLE hr.temp_emps;
```

Note: The DDL statement requires an exclusive table lock. It cannot obtain it, because session 1 already holds a row exclusive table lock on the `temp_emps` table.

4. In session 3 (`hr/hr`), update the salary by 5% for all employees with a salary > 15000 in the `temp_emps` table. Do not COMMIT.

```
SQL> CONNECT hr/hr
SQL> UPDATE temp_emps
  2  SET salary = salary * 1.05
  3  WHERE salary > 15000;
```

Practice 17 (continued)

5. In session 2, check to see what kind of locks are being held on the `temp_emps` table, using the `v$lock` view.

```
SQL> SELECT sid, type, id1, id2, lmode, request
  2  FROM v$lock
  3  WHERE id1 =
  4    (SELECT object_id FROM dba_objects
  5     WHERE object_name = 'TEMP_EMPS'
  6     AND object_type = 'TABLE');
```

6. In session 3, roll back the changes you made and set the `manager_id` column to 10 for all employees who have a salary < 15000.

```
SQL> rollback;
SQL> UPDATE hr.temp_emps SET MANAGER_id = 10
  2  WHERE salary < 15000;
```

Note: This session will be hanging, so do not wait for the statement to complete.

7. In session 2, check to see what kind of locks are being held on the `temp_emps` table, using the `v$lock` view.

```
SQL> SELECT sid, type, id1, id2, lmode, request
  2  FROM v$lock
  3  WHERE id1 =
  4    (SELECT object_id
  5     FROM dba_objects
  6     WHERE object_name = 'TEMP_EMPS'
  7     AND object_type = 'TABLE');
```

Or use Enterprise Manager Lock Manager

8. In session 2, run the `$ORACLE_HOME/rdbms/admin/catblock.sql` script. The script will create the `dba_waiters` view, which gives information regarding sessions holding or waiting on a lock. Use this view to determine the session ID for the session that is holding locks. Use this value to query `v$session` to obtain the serial number for the session holding the lock. Then run the `ALTER SYSTEM KILL SESSION` command to release the session holding the lock.

```
SQL> @$ORACLE_HOME/rdbms/admin/catblock.sql
SQL> SELECT waiting_session, holding_session
  2  FROM dba_waiters;
SQL> SELECT sid, serial#, username
  2  FROM v$session
  3  WHERE SID ='&HOLDING_SESSION';
SQL> ALTER SYSTEM KILL SESSION '&SID,&SERIAL_NUM';
```

C

Tuning Workshop

Workshop Scenarios

Use this Appendix for additional help in resolving the types of problem generated in the workshop scenarios that you tried to solve in the lesson titled “Workshop Overview.” There is a section devoted to each of the six scenarios that are described in the lesson titled “Workshop Overview”:

- Shared pool performance
- Database buffer cache performance
- Redo log buffer performance
- Data access performance
- PGA performance
- Assorted performance areas

As you read the hints and suggestions on the following pages, remember that you are restricted to an upper limit of 20 MB for the entire SGA.

Scenario 1: Shared Pool Performance

Waits recorded on the latch "Shared pool, library cache" can indicate an undersized shared pool. However, before increasing the SHARED_POOL_SIZE, it is advisable to determine why the shared pool is too small.

In some cases, you may find many similar SQL statements, differing only in literals in their WHERE clauses, stored in the SQL area. Often, in this situation, each statement is executed only once. You may want consider alternatives to enlarging the shared pool in such situations, such as rewriting the SQL with bind variables instead of literals or setting the CURSOR_SHARING initialization parameter to a different value.

To determine if you have SQL statements that could benefit from bind variables or greater cursor sharing, examine the Statspack report or query v$sql by using the like option of the WHERE clause, to collect information that can help identify similar SQL statements.

In other cases, the shared pool space may be poorly allocated because large packages are being swapped in and out regularly. Examine which packages are loaded by using the following query:

```
SQL> SELECT *
  2  FROM v$db_object_cache
  3  WHERE sharable_mem > 10000
  4  AND (type='PACKAGE' OR type='PACKAGE BODY'
  5  OR type='FUNCTION' OR type='PROCEDURE')
  6  AND  KEPT='NO';
```

If these packages are used frequently, then pin them in memory by using the dbms_shared_pool.keep('*Package_name*') package. Another consideration would be to reduce the size of the package. Determine whether there are large portions of the package that are not commonly used. If possible, separate the infrequently used procedures into different packages and retain the busiest, related procedures in a single package.

Examine SQL statements that are executed often using the following statement:

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

With this information, determine whether the SQL statement can be converted into a procedure and stored as a package; this can assist users in sharing the same cursor.

After you have reduced the number of SQL statements as much as possible, run the following query:

```
SQL> SELECT SUM(pins) "Executions", SUM(reloads)
  2    "Cache Misses", SUM(reloads)/SUM(pins)
  3  FROM v$librarycache;
```

Increase the shared pool to reduce cache misses. Compare the improvement you observe for each increase in the shared pool size to determine at which point the extra memory is not providing any useful benefit.

Scenario 1: Shared Pool Performance (continued)

Data Dictionary Cache

The data dictionary cache cannot be independently resized. The Oracle server automatically assigns memory from the space allocated by your SHARED_POOL_SIZE parameter for the shared SQL area and the data dictionary cache. Most of the additional memory allocated when you increase the SHARED_POOL_SIZE value is allocated to the shared SQL area.

To determine the hit ratio of the data dictionary cache, run the query:

```
SQL> SELECT parameter, gets, getmisses
  2  FROM v$rowcache;
```

The result set of this query contains a row for each segment of the data dictionary cache. You can check each area for usage. For example, if you see a large number of gets associated with sequences (dc_sequences), then this is probably because of sequence numbers not being cached. If this is the cause, then you should see a reduction in this number of gets by increasing the number of sequence numbers cached.

Scenario 2: Database Buffer Performance

The first indication that your buffer cache may be undersized is a high count for the free buffer waits event. Large values for the cache buffers LRU chain latch may also indicate that your buffer cache is too small. Waits on the latch may also signify that the DBWR process is not able to keep up with the work load.

To determine which problem is causing any latch contention you uncover, examine the number of writes in the "Tablespace IO Stats for DB" and "File IO Stats for DB" sections of your Statspack report.

On the front page of the Statspack report, the section named "Instance Efficiency Percentages" lists the important performance-related ratios of the instance. For this scenario, the value of "Buffer Hit%" is of interest.

The value of the "Buffer Hit%" depends on individual systems. A low percentage indicates that there are a lot of buffers being read into the database buffer cache. Ideally, this value should be close to 100 percent. There may be several reasons why this goal cannot be realized.

Before you increase the size of the database buffer cache, based on statistics that may point you in that direction, examine what SQL statements are being run against the database. Look for statements that cause a high number of buffer gets and how many times these statements are executed. If a statement is executed only a few times, then it may not be worth the effort to tune it. However, if a statement is executed many times and has a high number of buffer gets, then it is an ideal candidate for SQL tuning.

The Statspack report contains various lists the SQL statements that have been executed on the database. The first such list is sorted in descending order of the number of gets performed by the statements. Examine the top statements, keeping in mind that packages are also listed here, not just the SQL statements. When a candidate statement is found, examine the SQL statement to determine, if:

- Adding indexes could reduce the number of blocks accessed
- Rewriting the statement that requires fewer block accesses to obtain the same data
- Changing the application, for example, by sharing data between users, could reduce the number of executions.

After you have examined the SQL statements, and exhausted all means to reduce the number of buffers, then consider changing the size of the buffer cache. You need to answer two questions for yourself before changing the buffer cache size:

1. What is the current size?
   Determine the current size of the buffer cache with one of these methods:
   - Execute the `SHOW PARAMETER` command.
   - Query the `v$sgastat` view.
     ```
     SQL> SELECT *
       2  FROM v$sgastat
       3  WHERE name = 'db_block_buffers';
     ```
   - Find the value on the front page of your Statspack report.

Scenario 2: Database Buffer Performance (continued)

2. What is an appropriate size for the buffer cache?
   Determine the new value of the buffer cache by using the `DB_CACHE_ADVICE` initialization parameter. This parameter can have one of three values: Off, On, and Ready. Setting the value to On will start collecting the required statistics. The value can be changed by either:
   - Changing the initialization parameter and bouncing the database
   - Executing the command `ALTER SYSTEM SET DB_CACHE_ADVICE = ON`

Of course, setting the value to On will not provide you the information that you need immediately. The instance collects the necessary data as work is performed against the database. For the purpose of this Workshop, you will need to run one of the workloads against your database to collect information regarding buffer usage.

After your database has executed a typical load, query the `v$db_cache_advice` view to determine an appropriate new value for the `DB_CACHE_SIZE` parameter. When you decide on a new size, use the following command to dynamically change the size of the cache:

```
ALTER SYSTEM SET DB_CACHE_SIZE = new_value
```

To confirm if you selected an appropriate value, rerun the workload and examine the statistics again. If you intend to continue with the scenario, then you should also change the setting in your `SPFILE` to the new value with the command:

```
ALTER SYSTEM SET DB_CACHE_SIZE = new_value SCOPE = SPFILE
```

Scenario 3: Redo Log Buffer Performance

Waits on the LOG BUFFER SPACE event are an indication that your log buffer is too small.

On the first page of the Statspack report there is a section named "Instance Efficiency Percentages." Note the value of the statistic REDO NO WAIT. While this statistic's ideal value of 100 percent is seldom achieved, lower values may indicate that the redo log buffer is not sized correctly. In such cases, consider reducing the amount of redo created by the use of NO LOGGING in appropriate statements.

You may also need to change the size of your log buffer. Query the data dictionary to determine the current size of the redo log buffer and then estimate the size increase required by examining the amount of redo generated. The first page of the Statspack report has this information under the heading "Load Profile."

To set a new size for the redo log buffer, execute an ALTER SYSTEM command to modify the value of the LOG_BUFFER_SIZE initialization parameter in your SPFILE. This parameter is static, so you will need to bounce the database after making the change.

Rerun the workload generator to collect the statistics again. If your log buffer size increase has resolved the problem, then you may want to experiment with an even larger size, particularly if the improvement was substantial.

Note: The instance may not create the redo log buffer with the size stipulated by the initialization parameter value. Factors affecting the log buffer size include minimum size requirements and alignment with SGA memory boundaries. To confirm the actual redo log buffer size, query the v$sgastat view.

Scenario 4: Data Access Performance

Performance problems result from the deletion of several indexes. Statspack reports contain many indications when untuned SQL is running. For example:

- The buffer cache hit ratio is low, which can be seen on the first page of the Statspack report in the Load Profile section.
- There are many waits on the free buffer waits event.
- There are many full table scans.

Full table scans can be caused by badly written SQL statements and by missing or unused indexes.

To resolve the problem, determine which SQL statements are run on the database during a normal workload. You can use the SQL Trace utility or examine the top resource users in the Statspack report to collect representative SQL statements.

After you identify the appropriate statements, examine their `WHERE` clauses. Any columns referenced in the `WHERE` clause are good candidates for indexes.

To determine whether an index would be used by the optimizer, you must look at the expected number of rows to be returned. The more rows to be returned, the less likely an index will improve performance.

Confirm that the required indexes are present and enabled. The index might have been disabled for some reason. If the index is not present, then create the index.

Scenario 5: PGA Performance

The primary function of the DSS users is to create a series of reports for management by executing a number of scripts. Although these scripts produce the required reports, they run for a long time and do not complete the reports soon enough for the managers.

To uncover the cause for the tardy reports, your first step is to examine statistics collected during the report processing. (The workload scripts that you run for this workshop already include the appropriate queries so that you do not need to do anything differently when examining this scenario). When tuning for DSS activities, you should expect statistics to reflect contention for the buffer cache because of frequent full table scans.

You can find statistics that are related to these specific areas in various places in your Statspack reports:

- The buffer cache hit ratio, listed as "Buffer Hit %," is located in the "Instance Efficiency Percentages" section.
- The related statistic, "buffer busy waits," is in the "Wait Events for DB" section.
- Sort information is available in "Instance Activity" section of your Statspack reports.

On the front page, look at the buffer hit ratio. In a data warehouse environment you would expect this ratio to be low with a corresponding high value for buffer busy waits. To solve this situation, you may want to increase the size of the buffer cache. Because each set of blocks retrieved during full table scans reuse the same buffers, a larger cache may not significantly change the hit ratio.

Moving to the Instance Activity report, you could find that a large number of your sorts are being completed on disk. Ideally, none of your sorts should go to disk, but to accomplish this goal, you may have to devote large amounts of memory. A practical target is a ratio of disk sorts to memory sorts of less than five percent.

Your best option to tune the PGA memory, which is used for sorts, is to allow the processes to share a reserved amount of memory. Each user process acquires as much of this memory as needed to perform a specific operation, such as a sort, and allows other processes to claim it when the operation is completed. You define the overall size of the shared PGA memory with the `PGA_AGGREGATE_TARGET` initialization parameter. To enable the memory sharing, you must also set the `WORKAREA_SIZE_POLICY` initialization parameter to Auto.

If you are using shared servers, or an earlier release of Oracle, then you define the space that is available for sorts by each process with the `SORT_AREA_SIZE` initialization parameter. Increasing the value of `SORT_AREA_SIZE` parameter should allow more sorts to be performed in memory. The disadvantage of increasing the `SORT_AREA_SIZE` parameter is that more memory is consumed and it is not released after completing the sort. As more users execute sorts, they consume greater amounts of memory.

You may want to experiment by increasing the value of the `SORT_AREA_SIZE` parameter by a factor of 10 and seeing if this changes the ratio of disk to memory sorts. However, note that too much memory that is allocated this way can cause the operating system to page.

Note: With `WORKAREA_SIZE_POLICY` set to Auto, the `SORT_AREA_SIZE` initialization parameter, and other `*_AREA_SIZE` parameter values are ignored. All working memory for the user processes is taken from the aggregate pool.

Scenario 6: Assorted Performance Areas

This scenario provides a more real-world situation where you are not sure, before you begin your analysis, what may be causing poor performance. Typically, there could be one underlying reason for poor performance or a combination of related or unrelated problems. For this scenario, the database has a mix of problems that cause performance degradation, some of which are from the other workshop scenarios.

As in real life, you must apply your skills to analyze this scenario without further help.

Example of Statspack Report

```
STATSPACK report for

DB Name         DB Id    Instance     Inst Num Release     Cluster Host
------------ ----------- ------------ -------- ----------- ------- ------------
ORCL          995156390 orcl                 1 9.2.0.1.0   NO      EDT3R4P1

            Snap Id     Snap Time      Sessions Curs/Sess Comment
            ------- ------------------ -------- --------- -------------------
Begin Snap:                            1 30-Jul-02 15:14:22        7       5.0

  End Snap:                            2 30-Jul-02 16:01:23        7       5.9

   Elapsed:                                                 47.02
(mins)

Cache Sizes (end)
~~~~~~~~~~~~~~~~~
               Buffer Cache:         4M
     Std Block Size:                                              4K
           Shared Pool Size:         8M
         Log Buffer:                                             63K

Load Profile
~~~~~~~~~~~~                            Per Second       Per Transaction
                                   ---------------      ---------------
                  Redo size:                897.28             2,981.43
              Logical reads:                775.82             2,577.84
              Block changes:                  6.52                21.65
             Physical reads:                718.92             2,388.78
            Physical writes:                  5.58                18.55
                 User calls:                  0.21                 0.71
                     Parses:                  1.05                 3.47
                Hard parses:                  0.03                 0.11
                      Sorts:                  0.19                 0.65
                     Logons:                  0.01                 0.02
                   Executes:                  2.98                 9.89
               Transactions:                  0.30

  % Blocks changed per Read:    0.84
   Recursive Call %:                                          98.36
 Rollback per transaction %:    0.00
      Rows per Sort:                                        3677.05
```

```
Instance Efficiency Percentages (Target 100%)
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
            Buffer Nowait %:   98.34
      Redo NoWait %:                                          100.00
            Buffer  Hit   %:    7.59
   In-memory Sort %:                                           74.50
            Library Hit   %:   97.33
       Soft Parse %:                                           96.95
         Execute to Parse %:   64.88
        Latch Hit %:                                          100.00
Parse CPU to Parse Elapsd %:    2.11
    % Non-Parse CPU:                                           99.66

 Shared Pool Statistics        Begin   End
                               ------  ------
             Memory Usage %:   94.93   95.69
    % SQL with executions>1:   71.80   87.21
  % Memory for SQL w/exec>1:   48.25   84.48

Top 5 Timed Events
~~~~~~~~~~~~~~~~~~                                                    % Total
Event                                               Waits    Time (s) Ela Time
-------------------------------------------- ------------ ----------- --------
db file sequential read                           189,111      29,773    50.93
db file scattered read                            130,326      25,114    42.96
buffer busy waits                                  36,307       2,031     3.47
enqueue                                               724         535      .91
direct path read                                    3,085         325      .56
          -------------------------------------------------------------
```

```
Wait Events for DB: ORCL  Instance: orcl  Snaps: 1 -2
-> s  - second
-> cs - centisecond -     100th of a second
-> ms - millisecond -    1000th of a second
-> us - microsecond - 1000000th of a second
-> ordered by wait time desc, waits desc (idle events last)
```

Event	Waits	Timeouts	Total Wait Time (s)	Avg wait (ms)	Waits /txn
db file sequential read	189,111	0	29,773	157	222.7
db file scattered read	130,326	0	25,114	193	153.5
buffer busy waits	36,307	0	2,031	56	42.8
enqueue	724	19	535	738	0.9
direct path read	3,085	0	325	105	3.6
PL/SQL lock timer	217	216	222	1023	0.3
library cache load lock	177	20	86	485	0.2
log file parallel write	898	760	58	65	1.1
control file parallel write	892	0	55	61	1.1
db file parallel write	872	436	39	45	1.0
control file sequential read	406	0	24	59	0.5
library cache pin	170	0	20	117	0.2
direct path write	280	0	13	48	0.3
log file sync	174	0	5	26	0.2
row cache lock	38	0	2	41	0.0
latch free	213	18	0	1	0.3
SQL*Net break/reset to clien	38	0	0	0	0.0
SQL*Net message from client	561	0	221	394	0.7
SQL*Net message to client	561	0	0	0	0.7

```
Background Wait Events for DB: ORCL  Instance: orcl  Snaps: 1 -2
-> ordered by wait time desc, waits desc (idle events last)

                                                                   Avg
                                                     Total Wait   wait    Waits
Event                               Waits   Timeouts   Time (s)   (ms)     /txn
---------------------------- ------------ ---------- ---------- ------ --------
log file parallel write               898        760         58     65      1.1
control file parallel write           892          0         55     61      1.1
db file parallel write                872        436         39     45      1.0
control file sequential read          406          0         24     59      0.5
db file sequential read               121          0          5     42      0.1
direct path read                      172          0          0      1      0.2
db file scattered read                  2          0          0      7      0.0
latch free                              3          1          0      1      0.0
direct path write                      16          0          0      0      0.0
log file sync                           1          0          0      0      0.0
rdbms ipc message                   4,137      3,607     17,594   4253      4.9
smon timer                              9          9      4,591 ######      0.0
SQL*Net message from client             1          0        214 ######      0.0
SQL*Net message to client               1          0          0      0      0.0
          -------------------------------------------------------------
```

```
SQL ordered by Gets for DB: ORCL  Instance: orcl  Snaps: 1 -2
-> End Buffer Gets Threshold:   10000
-> Note that resources reported for PL/SQL includes the resources used by
   all SQL statements called within the PL/SQL code.  As individual SQL
   statements are also reported, it is possible and valid for the summed
   total % to exceed 100

                                                           CPU      Elapsd
  Buffer Gets    Executions  Gets per Exec  %Total Time (s)  Time (s) Hash Value
--------------- ------------ -------------- ------ -------- --------- ----------

      1,460,071           10      146,007.1   66.7    72.00 ######### 4281857347
Module: SQL*Plus
BEGIN workload_generator.oltp1; END;

        960,366          100        9,603.7   43.9    58.55 ######### 1750902811
Module: Workload Generator
DELETE from sh.customers where cust_id = :b1

        959,410          100        9,594.1   43.8    58.48 ######### 2215370455
Module: Workload Generator
 select /*+ all_rows */ count(1) from "SH"."SALES" where "CUST_I
D" = :1

        725,161           10       72,516.1   33.1    56.83 #########  899679532
Module: SQL*Plus
BEGIN workload_generator.dss1; END;

        724,775           33       21,962.9   33.1    60.44 ######### 3035176266
Module: Shipping Queries
SELECT c.cust_last_name, p.prod_name, s.amount_sold, s.quantity_
sold   from sh.customers c, sh.products p, sh.sales s   where c.
cust_id = s.cust_id   and s.prod_id = p.prod_id   and s.quantity
_sold = (select MAX(quantity_sold) from sh.sales)   and rownum =
 1

        436,669          200        2,183.3   20.0    27.36 ######### 3675510457
Module: Workload Generator
SELECT * from sh.customers             where cust_id >= :b1 and
rownum < 2 for update

          6,536          200           32.7    0.3     0.11    119.93 3247722561
Module: Workload Generator - oe-emp
SELECT employee_id from oe.employees             where employee_
id >= :b1 and rownum < 2 for update
```

```
SQL ordered by Gets for DB: ORCL  Instance: orcl  Snaps: 1 -2
-> End Buffer Gets Threshold:   10000
-> Note that resources reported for PL/SQL includes the resources used by
   all SQL statements called within the PL/SQL code.  As individual SQL
   statements are also reported, it is possible and valid for the summed
   total % to exceed 100

                                                           CPU      Elapsd
  Buffer Gets    Executions  Gets per Exec  %Total Time (s)  Time (s) Hash Value
--------------- ------------ -------------- ------ -------- --------- ----------


          5,716          200           28.6    0.3     0.13    188.80 3558587519
Module: Workload Generator - oe_ord
SELECT order_id from oe.orders             where order_id >= :b1
 and rownum < 2 for update

          4,890          200           24.5    0.2     0.09    161.84 2089710453
Module: Workload Generator - oe_prod
SELECT product_id from oe.product_information             where
product_id >= :b1 and rownum < 2 for update

          4,083          313           13.0    0.2     0.20      4.13 4168585130
Module: Workload Generator - oe_ord
INSERT into oe.order_items (order_id, line_item_id,
      product_id, unit_price, quantity)                 values
 (:b5, :b4,                   :b3, :b2, :b1)

      1,460,071           10      146,007.1   66.7    72.00 ########## 4281857347
Module: SQL*Plus
BEGIN workload_generator.oltp1; END;

        960,366          100        9,603.7   43.9    58.55 ########## 1750902811
Module: Workload Generator
DELETE from sh.customers where cust_id = :b1

        959,410          100        9,594.1   43.8    58.48 ########## 2215370455
Module: Workload Generator
 select /*+ all_rows */ count(1) from "SH"."SALES" where "CUST_I
D" = :1

        725,161           10       72,516.1   33.1    56.83 ##########  899679532
Module: SQL*Plus
BEGIN workload_generator.dss1; END;
```

```
SQL ordered by Gets for DB: ORCL  Instance: orcl  Snaps: 1 -2
-> End Buffer Gets Threshold:   10000
-> Note that resources reported for PL/SQL includes the resources used by
   all SQL statements called within the PL/SQL code.  As individual SQL
   statements are also reported, it is possible and valid for the summed
   total % to exceed 100

                                                           CPU      Elapsd
  Buffer Gets    Executions  Gets per Exec  %Total Time (s)  Time (s) Hash Value
--------------- ------------ -------------- ------ -------- --------- ----------

        724,775           33       21,962.9   33.1    60.44 ######### 3035176266
Module: Shipping Queries
SELECT c.cust_last_name, p.prod_name, s.amount_sold, s.quantity_
sold   from sh.customers c, sh.products p, sh.sales s   where c.
cust_id = s.cust_id   and s.prod_id = p.prod_id   and s.quantity
_sold = (select MAX(quantity_sold) from sh.sales)   and rownum =
 1

        436,669          200        2,183.3   20.0    27.36 ######### 3675510457
Module: Workload Generator
SELECT * from sh.customers             where cust_id >= :b1 and
rownum < 2 for update

          6,536          200           32.7    0.3     0.11    119.93 3247722561
Module: Workload Generator - oe-emp
SELECT employee_id from oe.employees             where employee_
id >= :b1 and rownum < 2 for update

          5,716          200           28.6    0.3     0.13    183.80 3558587519
Module: Workload Generator - oe_ord
SELECT order_id from oe.orders             where order_id >= :b1
 and rownum < 2 for update

          4,890          200           24.5    0.2     0.09    161.84 2089710453
Module: Workload Generator - oe_prod
SELECT product_id from oe.product_information             where
product_id >= :b1 and rownum < 2 for update

          4,083          313           13.0    0.2     0.20      4.13 4168585130
Module: Workload Generator - oe_ord
INSERT into oe.order_items (order_id, line_item_id,
      product_id, unit_price, quantity)                  values
 (:b5, :b4,                  :b3, :b2, :b1)
```

```
SQL ordered by Gets for DB: ORCL  Instance: orcl  Snaps: 1 -2
-> End Buffer Gets Threshold:   10000
-> Note that resources reported for PL/SQL includes the resources used by
   all SQL statements called within the PL/SQL code.  As individual SQL
   statements are also reported, it is possible and valid for the summed
   total % to exceed 100

                                                           CPU      Elapsd
  Buffer Gets    Executions  Gets per Exec  %Total Time (s)  Time (s) Hash Value
--------------- ------------ -------------- ------ -------- --------- ----------


          3,041          160           19.0    0.1     0.36      7.81 1915274376
Module: Workload Generator - oe-emp
INSERT into hr.employees (employee_id, last_name, first_name, em
ail,                phone_number, hire_date, job_id, salary, com
mission_pct, manager_id,                department_id)
      values (:b11, :b10, :b9, :b8,                :b7, :b6, :b5
, :b4,                :b3,:b2, :b1)

          1,992          313            6.4    0.1     0.05      0.37  467603321
Module: Workload Generator - oe_ord
SELECT (NVL(MAX(line_item_id),0)+1) FROM order_items       WHERE
 order_id = :b1

          1,790          517            3.5    0.1     0.22      0.50 3935516125
update seq$ set increment$=:2,minvalue=:3,maxvalue=:4,cycle#=:5,
order$=:6,cache=:7,highwater=:8,audit$=:9,flags=:10 where obj#=:
1

          1,350          313            4.3    0.1     0.05      0.15 4170474221
Module: Workload Generator - oe_ord
DELETE oe.order_items where  ROWID = :b1

          1,280          160            8.0    0.1     0.09      0.36 2950658496
select c.name, u.name from con$ c, cdef$ cd, user$ u  where c.co
n# = cd.con# and cd.enabled = :1 and c.owner# = u.user#

          1,259          160            7.9    0.1     0.19      2.40 2729780859
Module: Workload Generator - oe-emp
SELECT hr.employees_seq.nextval from dual

          1,076          150            7.2    0.0     0.22      0.36 1994657103
Module: Workload Generator
SELECT sh.customers_seq.nextval from dual
```

```
SQL ordered by Gets for DB: ORCL  Instance: orcl  Snaps: 1 -2
-> End Buffer Gets Threshold:   10000
-> Note that resources reported for PL/SQL includes the resources used by
   all SQL statements called within the PL/SQL code.  As individual SQL
   statements are also reported, it is possible and valid for the summed
   total % to exceed 100

                                                           CPU      Elapsd
  Buffer Gets    Executions  Gets per Exec  %Total Time (s)  Time (s) Hash Value
--------------- ------------ -------------- ------ -------- --------- ----------

            848          948            0.9    0.0     0.06      1.42 3615375148
Module: SQL*Plus
COMMIT

            575          115            5.0    0.0     0.05      0.23 1351631542
select o.name, c.name from con$ c, user$ o  where c.con# = :1 an
d owner# = user#

            554           24           23.1    0.0     0.02      1.24 1819073277
select owner#,name,namespace,remoteowner,linkname,p_timestamp,p_
obj#, d_owner#, nvl(property,0),subname from dependency$,obj$ wh
ere d_obj#=:1 and p_obj#=obj#(+) order by order#

            515          125            4.1    0.0     0.08      0.03 3863742839
Module: Workload Generator - oe_prod
DELETE oe.inventories where  ROWID = :b1

            511           26           19.7    0.0     0.05     14.45 3111103299
select /*+ index(idl_ub1$ i_idl_ub11) +*/ piece#,length,piece fr
om idl_ub1$ where obj#=:1 and part=:2 and version=:3 order by pi
ece#

            413           24           17.2    0.0     0.02      0.78 4049165760
select order#,columns,types from access$ where d_obj#=:1

            390           95            4.1    0.0     0.00      0.06 2085632044
select intcol#,nvl(pos#,0),col# from ccol$ where con#=:1

            220           55            4.0    0.0     0.02      0.86 2591785020
select obj#,type#,ctime,mtime,stime,status,dataobj#,flags,oid$,
spare1, spare2 from obj$ where owner#=:1 and name=:2 and namespa
ce=:3 and(remoteowner=:4 or remoteowner is null and :4 is null)a
nd(linkname=:5 or linkname is null and :5 is null)and(subname=:6
 or subname is null and :6 is null)
```

```
SQL ordered by Gets for DB: ORCL  Instance: orcl  Snaps: 1 -2
-> End Buffer Gets Threshold:   10000
-> Note that resources reported for PL/SQL includes the resources used by
   all SQL statements called within the PL/SQL code.  As individual SQL
   statements are also reported, it is possible and valid for the summed
   total % to exceed 100

                                                     CPU      Elapsd
  Buffer Gets    Executions  Gets per Exec  %Total Time (s)  Time (s) Hash Value
--------------- ------------ -------------- ------ -------- --------- ----------


            181          177            1.0    0.0     0.02      0.03 1375013356
Module: Workload Generator - oe_ord
UPDATE oe.order_items set unit_price = :b1                where
 ROWID = :b2

            139           26            5.3    0.0     0.00      0.76 3218356218
select /*+ index(idl_sb4$ i_idl_sb41) +*/ piece#,length,piece fr
om idl_sb4$ where obj#=:1 and part=:2 and version=:3 order by pi
ece#

            135           45            3.0    0.0     0.05      0.38 4059714361
select type#,blocks,extents,minexts,maxexts,extsize,extpct,user#
,iniexts,NVL(lists,65535),NVL(groups,65535),cachehint,hwmincr, N
VL(spare1,0) from seg$ where ts#=:1 and file#=:2 and block#=:3

             87           29            3.0    0.0     0.02      0.14  189272129
select o.owner#,o.name,o.namespace,o.remoteowner,o.linkname,o.su
bname,o.dataobj#,o.flags from obj$ o where o.obj#=:1

             72           26            2.8    0.0     0.00      0.19 1428100621
select /*+ index(idl_ub2$ i_idl_ub21) +*/ piece#,length,piece fr

          -------------------------------------------------------------
```

```
SQL ordered by Reads for DB: ORCL  Instance: orcl  Snaps: 1 -2
-> End Disk Reads Threshold:    1000

                                                       CPU      Elapsd
 Physical Reads  Executions  Reads per Exec %Total Time (s)  Time (s) Hash Value
--------------- ------------ -------------- ------ -------- --------- ----------

      1,327,038           10      132,703.8   65.4    72.00 ######### 4281857347
Module: SQL*Plus
BEGIN workload_generator.oltp1; END;

        944,747          100        9,447.5   46.6    58.55 ######### 1750902811
Module: Workload Generator
DELETE from sh.customers where cust_id = :b1

        944,693          100        9,446.9   46.6    58.48 ######### 2215370455
Module: Workload Generator
 select /*+ all_rows */ count(1) from "SH"."SALES" where "CUST_I
D" = :1

        700,549           10       70,054.9   34.5    56.83 #########  899679532
Module: SQL*Plus
BEGIN workload_generator.dss1; END;

        700,532           33       21,228.2   34.5    60.44 ######### 3035176266
Module: Shipping Queries
SELECT c.cust_last_name, p.prod_name, s.amount_sold, s.quantity_
sold   from sh.customers c, sh.products p, sh.sales s   where c.
cust_id = s.cust_id   and s.prod_id = p.prod_id   and s.quantity
_sold = (select MAX(quantity_sold) from sh.sales)   and rownum =
 1

        381,590          200        1,908.0   18.8    27.36 ######### 3675510457
Module: Workload Generator
SELECT * from sh.customers             where cust_id >= :b1 and
rownum < 2 for update

            251           26            9.7    0.0     0.05     14.45 3111103299
select /*+ index(idl_ub1$ i_idl_ub11) +*/ piece#,length,piece fr
om idl_ub1$ where obj#=:1 and part=:2 and version=:3 order by pi
ece#

             30           24            1.3    0.0     0.02      1.24 1819073277
select owner#,name,namespace,remoteowner,linkname,p_timestamp,p_
obj#, d_owner#, nvl(property,0),subname from dependency$,obj$ wh
ere d_obj#=:1 and p_obj#=obj#(+) order by order#
```

```
SQL ordered by Reads for DB: ORCL  Instance: orcl  Snaps: 1 -2
-> End Disk Reads Threshold:    1000

                                                     CPU      Elapsd
 Physical Reads  Executions  Reads per Exec %Total Time (s)  Time (s) Hash Value
--------------- ------------ -------------- ------ -------- --------- ----------


             28          200            0.1    0.0     0.09    161.84 2089710453
Module: Workload Generator - oe_prod
SELECT product_id from oe.product_information             where
product_id >= :b1 and rownum < 2 for update

             22          313            0.1    0.0     0.20      4.13 4168585130
Module: Workload Generator - oe_ord
INSERT into oe.order_items (order_id, line_item_id,
      product_id, unit_price, quantity)                   values
 (:b5, :b4,                   :b3, :b2, :b1)

             20           26            0.8    0.0     0.00      0.76 3218356218
select /*+ index(idl_sb4$ i_idl_sb41) +*/ piece#,length,piece fr
om idl_sb4$ where obj#=:1 and part=:2 and version=:3 order by pi
ece#

             19          160            0.1    0.0     0.36      7.81 1915274376
Module: Workload Generator - oe-emp
INSERT into hr.employees (employee_id, last_name, first_name, em
ail,                phone_number, hire_date, job_id, salary, com
mission_pct, manager_id,                department_id)
      values (:b11, :b10, :b9, :b8,                :b7, :b6, :b5
, :b4,                :b3,:b2, :b1)

             17           55            0.3    0.0     0.02      0.86 2591785020
select obj#,type#,ctime,mtime,stime,status,dataobj#,flags,oid$,
spare1, spare2 from obj$ where owner#=:1 and name=:2 and namespa
ce=:3 and(remoteowner=:4 or remoteowner is null and :4 is null)a
nd(linkname=:5 or linkname is null and :5 is null)and(subname=:6
 or subname is null and :6 is null)

             16           24            0.7    0.0     0.02      0.78 4049165760
select order#,columns,types from access$ where d_obj#=:1

             13           45            0.3    0.0     0.05      0.38 4059714361
select type#,blocks,extents,minexts,maxexts,extsize,extpct,user#
,iniexts,NVL(lists,65535),NVL(groups,65535),cachehint,hwmincr, N
VL(spare1,0) from seg$ where ts#=:1 and file#=:2 and block#=:3
```

```
SQL ordered by Reads for DB: ORCL  Instance: orcl  Snaps: 1 -2
-> End Disk Reads Threshold:    1000

                                                          CPU      Elapsd
 Physical Reads  Executions  Reads per Exec %Total Time (s)  Time (s) Hash Value
--------------- ------------ -------------- ------ -------- --------- ----------

              7           26            0.3    0.0     0.00      0.19 1428100621
select /*+ index(idl_ub2$ i_idl_ub21) +*/ piece#,length,piece fr
om idl_ub2$ where obj#=:1 and part=:2 and version=:3 order by pi
ece#

              7          160            0.0    0.0     0.19      2.40 2729780859
Module: Workload Generator - oe-emp
SELECT hr.employees_seq.nextval from dual

              6           26            0.2    0.0     0.02      0.34  957616262
select /*+ index(idl_char$ i_idl_char1) +*/ piece#,length,piece
from idl_char$ where obj#=:1 and part=:2 and version=:3 order by
 piece#

              6          200            0.0    0.0     0.13    188.80 3558587519
Module: Workload Generator - oe_ord
SELECT order_id from oe.orders             where order_id >= :b1
 and rownum < 2 for update

              6          517            0.0    0.0     0.22      0.50 3935516425
update seq$ set increment$=:2,minvalue=:3,maxvalue=:4,cycle#=:5,
order$=:6,cache=:7,highwater=:8,audit$=:9,flags=:10 where obj#=:
1

              3          160            0.0    0.0     0.09      0.36 2950658496
select c.name, u.name from con$ c, cdef$ cd, user$ u  where c.co
n# = cd.con# and cd.enabled = :1 and c.owner# = u.user#

              2           29            0.1    0.0     0.02      0.14  189272129

select o.owner#,o.name,o.namespace,o.remoteowner,o.linkname,o.su
bname,o.dataobj#,o.flags from obj$ o where o.obj#=:1

              2          313            0.0    0.0     0.05      0.37  467603321
Module: Workload Generator - oe_ord
SELECT (NVL(MAX(line_item_id),0)+1) FROM order_items       WHERE
 order_id = :b1
```

```
SQL ordered by Reads for DB: ORCL  Instance: orcl  Snaps: 1 -2
-> End Disk Reads Threshold:    1000

                                                     CPU      Elapsd
 Physical Reads  Executions  Reads per Exec %Total Time (s)  Time (s) Hash Value
--------------- ------------ -------------- ------ -------- --------- ----------

              2           16            0.1    0.0     0.00      0.03  931956286
select grantee#,privilege#,nvl(col#,0),max(mod(nvl(option$,0),2)
)from objauth$ where obj#=:1 group by grantee#,privilege#,nvl(co
l#,0) order by grantee#

              2          115            0.0    0.0     0.05      0.23 1351631542
select o.name, c.name from con$ c, user$ o  where c.con# = :1 an
d owner# = user#

              2           95            0.0    0.0     0.00      0.06 2085632044
select intcol#,nvl(pos#,0),col# from ccol$ where con#=:1

          -------------------------------------------------------------
```

```
SQL ordered by Executions for DB: ORCL  Instance: orcl  Snaps: 1 -2
-> End Executions Threshold:     100

                                                CPU per    Elap per
 Executions   Rows Processed   Rows per Exec    Exec (s)   Exec (s)  Hash Value
------------ --------------- ---------------- ----------- ---------- ----------


         948               0              0.0       0.00        0.00 3615375148
Module: SQL*Plus
COMMIT


         517             517              1.0       0.00        0.00 3935516425
update seq$ set increment$=:2,minvalue=:3,maxvalue=:4,cycle#=:5,
order$=:6,cache=:7,highwater=:8,audit$=:9,flags=:10 where obj#=:
1


         313             313              1.0       0.00        0.00  467603321
Module: Workload Generator - oe_ord
SELECT (NVL(MAX(line_item_id),0)+1) FROM order_items       WHERE
 order_id = :b1


         313             313              1.0       0.00        0.01 4168585130
Module: Workload Generator - oe_ord
INSERT into oe.order_items (order_id, line_item_id,
      product_id, unit_price, quantity)                   values
 (:b5, :b4,                   :b3, :b2, :b1)


         313             313              1.0       0.00        0.00 4170474221
Module: Workload Generator - oe_ord
DELETE oe.order_items where  ROWID = :b1


         200             200              1.0       0.00        0.81 2089710453
Module: Workload Generator - oe_prod
SELECT product_id from oe.product_information             where
product_id >= :b1 and rownum < 2 for update


         200             200              1.0       0.00        0.60 3247722561
Module: Workload Generator - oe_emp
SELECT employee_id from oe.employees             where employee_
id >= :b1 and rownum < 2 for update


         200             200              1.0       0.00        0.94 3558587519
Module: Workload Generator - oe_ord
SELECT order_id from oe.orders            where order_id >= :b1
 and rownum < 2 for update
```

```
SQL ordered by Executions for DB: ORCL  Instance: orcl  Snaps: 1 -2
-> End Executions Threshold:     100

                                                CPU per    Elap per
 Executions   Rows Processed   Rows per Exec    Exec (s)   Exec (s)  Hash Value
------------ --------------- ---------------- ----------- ---------- ----------


         200             200              1.0       0.14 ########### 3675510457
Module: Workload Generator
SELECT * from sh.customers             where cust_id >= :b1 and
rownum < 2 for update

         177             177              1.0       0.00        0.00 1375013356
Module: Workload Generator - oe_ord
UPDATE oe.order_items set unit_price = :b1                where
 ROWID = :b2

         160               0              0.0       0.00        0.05 1915274376
Module: Workload Generator - oe-emp
INSERT into hr.employees (employee_id, last_name, first_name, em
ail,                phone_number, hire_date, job_id, salary, com
mission_pct, manager_id,                department_id)
      values (:b11, :b10, :b9, :b8,                :b7, :b6, :b5
, :b4,                :b3,:b2, :b1)

         160             160              1.0       0.00        0.02 2729780859
Module: Workload Generator - oe-emp
SELECT hr.employees_seq.nextval from dual

         160             160              1.0       0.00        0.00 2950658496
select c.name, u.name from con$ c, cdef$ cd, user$ u  where c.co
n# = cd.con# and cd.enabled = :1 and c.owner# = u.user#

         150             150              1.0       0.00        0.00 1994657103
Module: Workload Generator
SELECT sh.customers_seq.nextval from dual

         150               0              0.0       0.00        0.00 2386552905
Module: Workload Generator
INSERT into sh.customers (cust_id, cust_first_name, cust_last_na
me,               cust_street_address, cust_city, country_id, c
ust_credit_limit,                cust_email)                valu
es (:b8, :b7, :b6,                :b5, :b4, :b3,
:b2, :b1)
```

```
SQL ordered by Executions for DB: ORCL  Instance: orcl  Snaps: 1 -2
-> End Executions Threshold:     100

                                                CPU per    Elap per
 Executions   Rows Processed   Rows per Exec    Exec (s)   Exec (s)  Hash Value
------------ --------------- ---------------- ----------- ---------- ----------

         125             125              1.0       0.00        0.00 3863742839
Module: Workload Generator - oe_prod
DELETE oe.inventories where  ROWID = :b1

         115             115              1.0       0.00        0.00 1351631542
select o.name, c.name from con$ c, user$ o  where c.con# = :1 an
d owner# = user#

         100              95              1.0       0.59 ########### 1750902811
Module: Workload Generator
DELETE from sh.customers where cust_id = :b1

         100             100              1.0       0.58 ########### 2215370455
Module: Workload Generator
 select /*+ all_rows */ count(1) from "SH"."SALES" where "CUST_I
D" = :1

          95             100              1.1       0.00        0.00 2085632044
select intcol#,nvl(pos#,0),col# from ccol$ where con#=:1

          55              47              0.9       0.00        0.02 2591785020
select obj#,type#,ctime,mtime,stime,status,dataobj#,flags,oid$,
spare1, spare2 from obj$ where owner#=:1 and name=:2 and namespa
ce=:3 and(remoteowner=:4 or remoteowner is null and :4 is null)a
nd(linkname=:5 or linkname is null and :5 is null)and(subname=:6
 or subname is null and :6 is null)

          45              45              1.0       0.00        0.01 4059714361
select type#,blocks,extents,minexts,maxexts,extsize,extpct,user#
,iniexts,NVL(lists,65535),NVL(groups,65535),cachehint,hwmincr, N
VL(spare1,0) from seg$ where ts#=:1 and file#=:2 and block#=:3

          33              33              1.0       1.83 ########### 3035176266
Module: Shipping Queries
SELECT c.cust_last_name, p.prod_name, s.amount_sold, s.quantity_
sold   from sh.customers c, sh.products p, sh.sales s   where c.
cust_id = s.cust_id   and s.prod_id = p.prod_id   and s.quantity
_sold = (select MAX(quantity_sold) from sh.sales)   and rownum =
1
```

```
SQL ordered by Executions for DB: ORCL  Instance: orcl  Snaps: 1 -2
-> End Executions Threshold:     100

                                                CPU per    Elap per
 Executions   Rows Processed   Rows per Exec    Exec (s)   Exec (s)  Hash Value
------------ --------------- ---------------- ----------- ---------- ----------

          29              29              1.0       0.00        0.00  189272129
select o.owner#,o.name,o.namespace,o.remoteowner,o.linkname,o.su
bname,o.dataobj#,o.flags from obj$ o where o.obj#=:1

          26               4              0.2       0.00        0.01  957616262
select /*+ index(idl_char$ i_idl_char1) +*/ piece#,length,piece
from idl_char$ where obj#=:1 and part=:2 and version=:3 order by
 piece#

          26               8              0.3       0.00        0.01 1428100621
select /*+ index(idl_ub2$ i_idl_ub21) +*/ piece#,length,piece fr
om idl_ub2$ where obj#=:1 and part=:2 and version=:3 order by pi
ece#

          26             102              3.9       0.00        0.56 3111103299
select /*+ index(idl_ub1$ i_idl_ub11) +*/ piece#,length,piece fr
om idl_ub1$ where obj#=:1 and part=:2 and version=:3 order by pi

          -------------------------------------------------------------
```

```
SQL ordered by Parse Calls for DB: ORCL  Instance: orcl  Snaps: 1 -2
-> End Parse Calls Threshold:      1000

                           % Total
 Parse Calls  Executions   Parses  Hash Value
------------ ------------ -------- ----------
         517          517    17.53 3935516425
update seq$ set increment$=:2,minvalue=:3,maxvalue=:4,cycle#=:5,
order$=:6,cache=:7,highwater=:8,audit$=:9,flags=:10 where obj#=:
1

         211          948     7.15 3615375148
Module: SQL*Plus
COMMIT

         160          160     5.42 2950658496
select c.name, u.name from con$ c, cdef$ cd, user$ u  where c.co
n# = cd.con# and cd.enabled = :1 and c.owner# = u.user#

         150          150     5.08 2386552905
Module: Workload Generator
INSERT into sh.customers (cust_id, cust_first_name, cust_last_na
me,                cust_street_address, cust_city, country_id, c
ust_credit_limit,               cust_email)               valu
es (:b8, :b7, :b6,               :b5, :b4, :b3,
:b2, :b1)

         115          115     3.90 1351631542
select o.name, c.name from con$ c, user$ o  where c.con# = :1 an
d owner# = user#

          45           45     1.53 4059714361
select type#,blocks,extents,minexts,maxexts,extsize,extpct,user#
,iniexts,NVL(lists,65535),NVL(groups,65535),cachehint,hwmincr, N
VL(spare1,0) from seg$ where ts#=:1 and file#=:2 and block#=:3

          32           55     1.08 2591785020
select obj#,type#,ctime,mtime,stime,status,dataobj#,flags,oid$,
spare1, spare2 from obj$ where owner#=:1 and name=:2 and namespa
ce=:3 and(remoteowner=:4 or remoteowner is null and :4 is null)a
nd(linkname=:5 or linkname is null and :5 is null)and(subname=:6
 or subname is null and :6 is null)

          26           26     0.88  957616262
select /*+ index(idl_char$ i_idl_char1) +*/ piece#,length,piece
from idl_char$ where obj#=:1 and part=:2 and version=:3 order by
 piece#
```

```
SQL ordered by Parse Calls for DB: ORCL  Instance: orcl  Snaps: 1 -2
-> End Parse Calls Threshold:      1000

                           % Total
 Parse Calls  Executions   Parses  Hash Value
------------ ------------ -------- ----------

          26           26     0.88 1428100621
select /*+ index(idl_ub2$ i_idl_ub21) +*/ piece#,length,piece fr
om idl_ub2$ where obj#=:1 and part=:2 and version=:3 order by pi
ece#

          26           26     0.88 3111103299
select /*+ index(idl_ub1$ i_idl_ub11) +*/ piece#,length,piece fr
om idl_ub1$ where obj#=:1 and part=:2 and version=:3 order by pi
ece#

          26           26     0.88 3218356218
select /*+ index(idl_sb4$ i_idl_sb41) +*/ piece#,length,piece fr
om idl_sb4$ where obj#=:1 and part=:2 and version=:3 order by pi
ece#

          24           24     0.81 1819073277
select owner#,name,namespace,remoteowner,linkname,p_timestamp,p_
obj#, d_owner#, nvl(property,0),subname from dependency$,obj$ wh
ere d_obj#=:1 and p_obj#=obj#(+) order by order#

          24           24     0.81 4049165760
select order#,columns,types from access$ where d_obj#=:1

          20          200     0.68 2089710453
Module: Workload Generator - oe_prod
SELECT product_id from oe.product_information             where
product_id >= :b1 and rownum < 2 for update

          20          200     0.68 3247722561
Module: Workload Generator - oe-emp
SELECT employee_id from oe.employees             where employee_
id >= :b1 and rownum < 2 for update

          20          200     0.68 3558587519
Module: Workload Generator - oe_ord
SELECT order_id from oe.orders             where order_id >= :b1
 and rownum < 2 for update
```

```
SQL ordered by Parse Calls for DB: ORCL  Instance: orcl  Snaps: 1 -2
-> End Parse Calls Threshold:      1000

                           % Total
 Parse Calls  Executions   Parses  Hash Value
------------ ------------ -------- ----------

          20          200     0.68 3675510457
Module: Workload Generator
SELECT * from sh.customers             where cust_id >= :b1 and
rownum < 2 for update

          14           14     0.47 1705880752
select file# from file$ where ts#=:1

          13          100     0.44 1750902811
Module: Workload Generator
DELETE from sh.customers where cust_id = :b1

          13          100     0.44 2215370455
Module: Workload Generator
 select /*+ all_rows */ count(1) from "SH"."SALES" where "CUST_I
D" = :1

          12           29     0.41  189272129
select o.owner#,o.name,o.namespace,o.remoteowner,o.linkname,o.su
bname,o.dataobj#,o.flags from obj$ o where o.obj#=:1

          12           16     0.41  931956286
select grantee#,privilege#,nvl(col#,0),max(mod(nvl(option$,0),2)
)from objauth$ where obj#=:1 group by grantee#,privilege#,nvl(co
l#,0) order by grantee#

          11           20     0.37 2385919346
select name,intcol#,segcol#,type#,length,nvl(precision#,0),decod
e(type#,2,nvl(scale,-127/*MAXSB1MINAL*/),178,scale,179,scale,180
,scale,181,scale,182,scale,183,scale,231,scale,0),null$,fixedsto
rage,nvl(deflength,0),default$,rowid,col#,property, nvl(charseti
d,0),nvl(charsetform,0),spare1,spare2,nvl(spare3,0) from col$ wh

          10           10     0.34  899679532
Module: SQL*Plus
BEGIN workload_generator.dss1; END;
```

```
SQL ordered by Parse Calls for DB: ORCL  Instance: orcl  Snaps: 1 -2
-> End Parse Calls Threshold:      1000

                           % Total
 Parse Calls  Executions   Parses  Hash Value
------------ ------------ -------- ----------

          10          177     0.34 1375013356
Module: Workload Generator - oe_ord
UPDATE oe.order_items set unit_price = :b1                where
 ROWID = :b2

          10          160     0.34 1915274376
Module: Workload Generator - oe-emp
INSERT into hr.employees (employee_id, last_name, first_name, em
ail,                phone_number, hire_date, job_id, salary, com

          -------------------------------------------------------------
```

Instance Activity Stats for DB: ORCL Instance: orcl Snaps: 1 -2

Statistic	Total	per Second	per Trans
CPU used by this session	15,305	5.4	18.0
CPU used when call started	15,305	5.4	18.0
CR blocks created	4,555	1.6	5.4
Cached Commit SCN referenced	1,556,535	551.8	1,833.4
Commit SCN cached	50	0.0	0.1
DBWR buffers scanned	1,102	0.4	1.3
DBWR checkpoint buffers written	2,049	0.7	2.4
DBWR checkpoints	0	0.0	0.0
DBWR free buffers found	1,039	0.4	1.2
DBWR lru scans	6	0.0	0.0
DBWR make free requests	6	0.0	0.0
DBWR summed scan depth	1,102	0.4	1.3
DBWR transaction table writes	430	0.2	0.5
DBWR undo block writes	723	0.3	0.9
SQL*Net roundtrips to/from client	501	0.2	0.6
active txn count during cleanout	2,806	1.0	3.3
background timeouts	2,720	1.0	3.2
buffer is not pinned count	2,130,206	755.1	2,509.1
buffer is pinned count	2,539	0.9	3.0
bytes received via SQL*Net from c	56,493	20.0	66.5
bytes sent via SQL*Net to client	65,619	23.3	77.2
calls to get snapshot scn: kcmgss	29,701	10.5	35.0
calls to kcmgas	6,251	2.2	7.4
calls to kcmgcs	160	0.1	0.2
change write time	31	0.0	0.0
cleanout - number of ktugct calls	2,253	0.8	2.7
cleanouts and rollbacks - consist	1,481	0.5	1.7
cleanouts only - consistent read	130	0.1	0.2
cluster key scan block gets	830	0.3	1.0
cluster key scans	478	0.2	0.6
commit cleanout failures: block l	2	0.0	0.0
commit cleanout failures: callbac	127	0.1	0.2
commit cleanout failures: cannot	1	0.0	0.0
commit cleanouts	2,551	0.9	3.0
commit cleanouts successfully com	2,421	0.9	2.9
commit txn count during cleanout	243	0.1	0.3
consistent changes	10,340	3.7	12.2
consistent gets	2,169,211	769.0	2,555.0
consistent gets - examination	18,811	6.7	22.2
current blocks converted for CR	6	0.0	0.0
cursor authentications	76	0.0	0.1
data blocks consistent reads - un	10,076	3.6	11.9

Instance Activity Stats for DB: ORCL Instance: orcl Snaps: 1 -2

Statistic	Total	per Second	per Trans
db block changes	18,384	6.5	21.7
db block gets	19,379	6.9	22.8
deferred (CURRENT) block cleanout	1,229	0.4	1.5
dirty buffers inspected	55	0.0	0.1
enqueue conversions	2,507	0.9	3.0
enqueue releases	8,921	3.2	10.5
enqueue requests	8,940	3.2	10.5
enqueue timeouts	19	0.0	0.0
enqueue waits	665	0.2	0.8
execute count	8,399	3.0	9.9
free buffer inspected	12,938	4.6	15.2
free buffer requested	2,027,711	718.8	2,388.4
hot buffers moved to head of LRU	695	0.3	0.8
immediate (CR) block cleanout app	1,611	0.6	1.9
immediate (CURRENT) block cleanou	195	0.1	0.2
index fast full scans (full)	547	0.2	0.6
index fetch by key	2,405	0.9	2.8
index scans kdiixs1	2,799	1.0	3.3
leaf node 90-10 splits	3	0.0	0.0
leaf node splits	18	0.0	0.0
logons cumulative	20	0.0	0.0
messages received	1,046	0.4	1.2
messages sent	1,046	0.4	1.2
no buffer to keep pinned count	31	0.0	0.0
no work - consistent read gets	2,125,242	753.4	2,503.2
opened cursors cumulative	2,601	0.9	3.1
parse count (hard)	90	0.0	0.1
parse count (total)	2,950	1.1	3.5
parse time cpu	52	0.0	0.1
parse time elapsed	2,468	0.9	2.9
physical reads	2,028,073	718.9	2,388.8
physical reads direct	5,585	2.0	6.6
physical writes	15,751	5.6	18.6
physical writes direct	13,586	4.8	16.0
physical writes non checkpoint	14,508	5.1	17.1
pinned buffers inspected	12,860	4.6	15.2
prefetched blocks	1,703,051	603.7	2,006.0
prefetched blocks aged out before	30	0.0	0.0
process last non-idle time	20,561,129,916	7,288,596.2	###########
recovery blocks read	0	0.0	0.0
recursive calls	36,164	12.8	42.6
recursive cpu usage	15,194	5.4	17.9
redo blocks written	5,545	2.0	6.5

Instance Activity Stats for DB: ORCL Instance: orcl Snaps: 1 -2

Statistic	Total	per Second	per Trans
redo entries	10,568	3.8	12.5
redo size	2,531,236	897.3	2,981.4
redo synch time	479	0.2	0.6
redo synch writes	175	0.1	0.2
redo wastage	248,460	88.1	292.7
redo write time	6,413	2.3	7.6
redo writer latching time	0	0.0	0.0
redo writes	897	0.3	1.1
rollback changes - undo records a	498	0.2	0.6
rollbacks only - consistent read	3,110	1.1	3.7
rows fetched via callback	929	0.3	1.1
session connect time	20,561,129,916	7,288,596.2	############
session logical reads	2,188,590	775.8	2,577.8
session pga memory	0	0.0	0.0
session pga memory max	0	0.0	0.0
session uga memory	295,704	104.8	348.3
session uga memory max	8,836,212	3,132.3	10,407.8
shared hash latch upgrades - no w	2,375	0.8	2.8
sorts (disk)	140	0.1	0.2
sorts (memory)	409	0.1	0.5
sorts (rows)	2,018,699	715.6	2,377.7
summed dirty queue length	134	0.1	0.2
switch current to new buffer	98	0.0	0.1
table fetch by rowid	3,109	1.1	3.7
table fetch continued row	174	0.1	0.2
table scan blocks gotten	2,123,096	752.6	2,500.7
table scan rows gotten	181,473,409	64,329.5	213,749.6
table scans (long tables)	2,480	0.9	2.9
table scans (short tables)	3,352	1.2	4.0
transaction rollbacks	162	0.1	0.2
transaction tables consistent rea	3	0.0	0.0
transaction tables consistent rea	257	0.1	0.3
user calls	602	0.2	0.7
user commits	849	0.3	1.0
workarea executions - multipass	134	0.1	0.2
workarea executions - optimal	595	0.2	0.7
write clones created in foregroun	17	0.0	0.0

```
Tablespace IO Stats for DB: ORCL  Instance: orcl  Snaps: 1 -2
->ordered by IOs (Reads + Writes) desc

Tablespace
------------------------------
                 Av      Av     Av                    Av        Buffer Av Buf
         Reads Reads/s Rd(ms) Blks/Rd       Writes Writes/s      Waits Wt(ms)
-------------- ------- ------ ------- ------------ -------- ---------- ------
EXAMPLE
       318,868     113   90.6     6.3          856        0     36,250   56.0
TEMP
         5,585       2  130.6     1.0       13,586        5          0    0.0
UNDOTBS1
            25       0   62.0     1.0        1,161        0          0    0.0
SYSTEM
           476       0   52.0     1.0           22        0         57   21.1
TOOLS
            61       0    9.3     1.0          126        0          0    0.0
          -------------------------------------------------------------
File IO Stats for DB: ORCL  Instance: orcl  Snaps: 1 -2
->ordered by Tablespace, File

Tablespace               Filename
------------------------ ----------------------------------------------------
                 Av      Av     Av                    Av        Buffer Av Buf
         Reads Reads/s Rd(ms) Blks/Rd       Writes Writes/s      Waits Wt(ms)
-------------- ------- ------ ------- ------------ -------- ---------- ------
EXAMPLE                  E:\ORANT\ORA92\ORADATA\ORCL\EXAMPLE01.DBF
       318,868     113   90.6     6.3          856        0     36,250   56.0

SYSTEM                   E:\ORANT\ORA92\ORADATA\ORCL\SYSTEM01.DBF
           476       0   52.0     1.0           22        0         57   21.1

TEMP                     E:\ORANT\ORA92\ORADATA\ORCL\TEMP01.DBF
         5,585       2  130.6     1.0       13,586        5          0

TOOLS                    E:\ORANT\ORA92\ORADATA\ORCL\TOOLS01.DBF
            61       0    9.3     1.0          126        0          0

UNDOTBS1                 E:\ORANT\ORA92\ORADATA\ORCL\UNDOTBS01.DBF
            25       0   62.0     1.0        1,161        0          0

          -------------------------------------------------------------
```

Buffer Pool Statistics for DB: ORCL Instance: orcl Snaps: 1 -2
-> Standard block size Pools D: default, K: keep, R: recycle
-> Default Pools for other block sizes: 2k, 4k, 8k, 16k, 32k

P	Number of Buffers	Cache Hit %	Buffer Gets	Physical Reads	Physical Writes	Free Buffer Waits	Write Complete Waits	Buffer Busy Waits
D	979	53.2	4,320,233	2,022,481	2,165	0	0	36,307

Instance Recovery Stats for DB: ORCL Instance: orcl Snaps: 1 -2
-> B: Begin snapshot, E: End snapshot

	Targt MTTR (s)	Estd MTTR (s)	Recovery Estd IOs	Actual Redo Blks	Target Redo Blks	Log File Size Redo Blks	Log Ckpt Timeout Redo Blks	Log Ckpt Interval Redo Blks
B	16	12	152	558	18432	18432		
E	16	12	159	0	18432	18432		

Buffer Pool Advisory for DB: ORCL Instance: orcl End Snap: 2
-> Only rows with estimated physical reads >0 are displayed
-> ordered by Block Size, Buffers For Estimate

P	Size for Estimate (M)	Size Factr	Buffers for Estimate	Est Physical Read Factor	Estimated Physical Reads
D	4	1.0	979	1.00	2,029,871
D	8	2.0	1,958	0.89	1,801,958
D	12	3.0	2,937	0.80	1,620,631
D	16	4.0	3,916	0.72	1,468,751
D	20	5.0	4,895	0.65	1,318,356
D	24	6.0	5,874	0.58	1,184,142
D	28	7.0	6,853	0.51	1,043,838
D	32	8.0	7,832	0.43	875,666
D	36	9.0	8,811	0.36	730,044
D	40	10.0	9,790	0.29	582,543
D	44	11.0	10,769	0.20	397,890
D	48	12.0	11,748	0.07	140,097
D	52	13.0	12,727	0.01	23,791
D	56	14.0	13,706	0.01	20,051
D	60	15.0	14,685	0.01	20,051
D	64	16.0	15,664	0.01	20,014
D	68	17.0	16,643	0.01	20,014
D	72	18.0	17,622	0.01	20,014
D	76	19.0	18,601	0.01	20,014
D	80	20.0	19,580	0.01	20,014

```
Buffer wait Statistics for DB: ORCL  Instance: orcl  Snaps: 1 -2
-> ordered by wait time desc, waits desc

                                                                     Tot Wait
Class                                                          Waits   Time (s)
------------------------------------------------------ ----------- ----------
   Avg
Time (ms)
---------
data block                                                    36,172      2,022
       56
2nd level bmb                                                     20          4
      224
segment header                                                    90          4
       39
1st level bmb                                                     25          1
       28
          -------------------------------------------------------------
PGA Aggr Target Stats for DB: ORCL  Instance: orcl  Snaps: 1 -2
-> B: Begin snap   E: End snap (rows dentified with B or E contain data
   which is absolute i.e. not diffed over the interval)
-> PGA cache hit % - percentage of W/A (WorkArea) data processed only in-memory
-> Auto PGA Target - actual workarea memory target
-> W/A PGA Used    - amount of memory used for all Workareas (manual + auto)
-> %PGA W/A Mem    - percentage of PGA memory allocated to workareas
-> %Auto W/A Mem   - percentage of workarea memory controlled by Auto Mem Mgmt
-> %Man W/A Mem    - percentage of workarea memory under manual control

PGA Cache Hit % W/A MB Processed Extra W/A MB Read/Written
--------------- ---------------- -------------------------
           76.4              105                        32

                                                   %PGA  %Auto   %Man
  PGA Aggr  Auto PGA   PGA Mem    W/A PGA  W/A    W/A    W/A   Global Mem
  Target(M) Target(M)  Alloc(M)   Used(M)  Mem    Mem    Mem    Bound(K)
- --------- --------- ---------- ---------- ------ ------ ------ ----------
B        24        17        8.0        0.0     .0     .0     .0      1,228
E        24        17        8.0        0.0     .0     .0     .0      1,228
          -------------------------------------------------------------
```

PGA Aggr Target Histogram for DB: ORCL Instance: orcl Snaps: 1 -2
-> Optimal Executions are purely in-memory operations

Low Optimal	High Optimal	Total Execs	Optimal Execs	1-Pass Execs	M-Pass Execs
4K	8K	456	456	0	0
8K	16K	8	8	0	0
16K	32K	2	2	0	0
32K	64K	133	1	0	132
64K	128K	1	1	0	0
128K	256K	2	0	0	2
256K	512K	66	66	0	0
1M	2M	66	66	0	0

PGA Memory Advisory for DB: ORCL Instance: orcl End Snap: 2
-> When using Auto Memory Mgmt, minimally choose a pga_aggregate_target value
   where Estd PGA Overalloc Count is 0

PGA Target Est (MB)	Size Factr	W/A MB Processed	Estd Extra W/A MB Read/ Written to Disk	Estd PGA Cache Hit %	Estd PGA Overalloc Count
12	0.5	0.0	0.0	0.0	1
18	0.8	0.0	0.0	0.0	1
24	1.0	0.0	0.0	0.0	0
29	1.2	0.0	0.0	0.0	0
34	1.4	0.0	0.0	0.0	0
38	1.6	0.0	0.0	0.0	0
43	1.8	0.0	0.0	0.0	0
48	2.0	0.0	0.0	0.0	0
72	3.0	0.0	0.0	0.0	0
96	4.0	0.0	0.0	0.0	0
144	6.0	0.0	0.0	0.0	0
192	8.0	0.0	0.0	0.0	0

Enqueue activity for DB: ORCL Instance: orcl Snaps: 1 -2
-> Enqueue stats gathered prior to 9i should not be compared with 9i data
-> ordered by Wait Time desc, Waits desc

Eq	Requests	Succ Gets	Failed Gets	Waits	Avg Wt Time (ms)	Wait Time (s)
TX	2,077	2,058	19	465	964.33	448
TM	7,814	7,814	0	200	433.54	87

```
Rollback Segment Stats for DB: ORCL  Instance: orcl  Snaps: 1 -2
->A high value for "Pct Waits" suggests more rollback segments may be required
->RBS stats may not be accurate between begin and end snaps when using Auto Undo
  managment, as RBS may be dynamically created and dropped as needed

        Trans Table       Pct   Undo Bytes
RBS No      Gets        Waits     Written        Wraps  Shrinks  Extends
------ -------------- ------- --------------- -------- -------- --------
     0           10.0    0.00               0        0        0        0
     1          428.0    0.00          73,508        2        0        2
     2          441.0    0.00          71,700        2        0        2
     3          464.0    0.00          63,328        1        0        1
     4          396.0    0.00          58,240        2        0        2
     5          437.0    0.00          67,164        0        0        0
     6          446.0    0.00          67,546        2        0        2
     7          378.0    0.00          66,198        0        0        0
     8          481.0    0.00          75,778        2        0        2
     9          379.0    0.00          54,632        0        0        0
    10          466.0    0.00         183,916        0        0        0
          -------------------------------------------------------------
Rollback Segment Storage for DB: ORCL  Instance: orcl  Snaps: 1 -2
->Optimal Size should be larger than Avg Active

RBS No    Segment Size      Avg Active    Optimal Size    Maximum Size
------ --------------- --------------- --------------- ---------------
     0         425,984           6,144                         425,984
     1         716,800          32,145                         716,800
     2         913,408          22,536                         913,408
     3       2,093,056         147,845                       2,093,056
     4         585,728          17,759                         585,728
     5       1,175,552               0                       1,175,552
     6         716,800          17,759                         716,800
     7       1,175,552               0                       1,175,552
     8         651,264          17,759                         651,264
     9       2,289,664               0                       2,289,664
    10       1,175,552               0                       1,175,552
          -------------------------------------------------------------
Undo Segment Summary for DB: ORCL  Instance: orcl  Snaps: 1 -2
-> Undo segment block stats:
-> uS - unexpired Stolen,   uR - unexpired Released,   uU - unexpired reUsed
-> eS - expired   Stolen,   eR - expired   Released,   eU - expired   reUsed

Undo           Undo        Num  Max Qry     Max Tx Snapshot Out of uS/uR/uU/
 TS#         Blocks      Trans  Len (s)   Concurcy  Too Old  Space eS/eR/eU
---- -------------- ---------- -------- ---------- -------- ------ -------------
   1            811     14,114      660         10        0      0 0/0/0/0/0/0
          -------------------------------------------------------------
```

Undo Segment Stats for DB: ORCL Instance: orcl Snaps: 1 -2
-> ordered by Time desc

End Time	Undo Blocks	Num Trans	Max Qry Len (s)	Max Tx Concy	Snap Too Old	Out of Space	uS/uR/uU/ eS/eR/eU
30-Jul 15:58	0	0	0	0	0	0	0/0/0/0/0/0
30-Jul 15:48	5	3,595	399	8	0	0	0/0/0/0/0/0
30-Jul 15:38	5	3,549	660	7	0	0	0/0/0/0/0/0
30-Jul 15:28	4	3,525	453	10	0	0	0/0/0/0/0/0
30-Jul 15:18	797	3,445	219	5	0	0	0/0/0/0/0/0

Latch Activity for DB: ORCL Instance: orcl Snaps: 1 -2
->"Get Requests", "Pct Get Miss" and "Avg Slps/Miss" are statistics for
  willing-to-wait latch get requests
->"NoWait Requests", "Pct NoWait Miss" are for no-wait latch get requests
->"Pct Misses" for both should be very close to 0.0

Latch	Get Requests	Pct Get Miss	Avg Slps /Miss	Wait Time (s)	NoWait Requests	Pct NoWait Miss
Consistent RBA	898	0.0		0	0	
FOB s.o list latch	118	0.0		0	0	
SQL memory manager latch	1	0.0		0	892	0.0
SQL memory manager worka	60,311	0.0		0	0	
active checkpoint queue	1,377	0.0		0	0	
cache buffer handles	69	0.0		0	0	
cache buffers chains	6,520,208	0.0	1.0	0	3,787,163	0.0
cache buffers lru chain	4,151,271	0.0	1.3	0	2,398	0.0
channel handle pool latc	40	0.0		0	0	
channel operations paren	1,853	0.0		0	0	
checkpoint queue latch	62,800	0.0		0	1,731	0.0
child cursor hash table	1,049	0.0		0	0	
dml lock allocation	10,660	0.0		0	0	
dummy allocation	40	0.0		0	0	
enqueue hash chains	21,119	0.0		0	0	
enqueues	5,205	0.0		0	0	
event group latch	20	0.0		0	0	
file number translation	19,558	0.0	1.0	0	0	
hash table column usage	35	0.0		0	92	0.0
hash table modification	2	0.0		0	0	
ktm global data	9	0.0		0	0	
lgwr LWN SCN	1,271	0.0		0	0	
library cache	57,587	0.0	1.0	0	476	0.0

Latch Activity for DB: ORCL Instance: orcl Snaps: 1 -2
->"Get Requests", "Pct Get Miss" and "Avg Slps/Miss" are statistics for
  willing-to-wait latch get requests
->"NoWait Requests", "Pct NoWait Miss" are for no-wait latch get requests
->"Pct Misses" for both should be very close to 0.0

Latch	Get Requests	Pct Get Miss	Avg Slps /Miss	Wait Time (s)	NoWait Requests	Pct NoWait Miss
library cache load lock	577	0.0		0	0	
library cache pin	35,875	0.0		0	0	
library cache pin alloca	14,850	0.0		0	0	
list of block allocation	38	0.0		0	0	
loader state object free	684	0.0		0	0	
longop free list parent	390	0.0		0	98	0.0
messages	9,781	0.0		0	0	
mostly latch-free SCN	1,271	0.0		0	0	
multiblock read objects	625,480	0.0	1.1	0	0	
ncodef allocation latch	46	0.0		0	0	
object stats modificatio	104	0.0		0	0	
post/wait queue	1,809	0.0		0	174	0.0
process allocation	20	0.0		0	20	0.0
process group creation	40	0.0		0	0	
redo allocation	12,734	0.0		0	0	
redo copy	4	0.0		0	10,561	0.0
redo writing	5,047	0.0		0	0	
row cache enqueue latch	13,873	0.0		0	0	
row cache objects	16,402	0.0		0	723	0.0
sequence cache	1,085	0.0		0	0	
session allocation	5,513	0.0		0	0	
session idle bit	2,409	0.0		0	0	
session switching	46	0.0		0	0	
session timer	957	0.0		0	0	
shared pool	23,536	0.0		0	0	
sim partition latch	0			0	257	0.0
simulator hash latch	269,094	0.0		0	0	

Latch Activity for DB: ORCL Instance: orcl Snaps: 1 -2
->"Get Requests", "Pct Get Miss" and "Avg Slps/Miss" are statistics for
willing-to-wait latch get requests
->"NoWait Requests", "Pct NoWait Miss" are for no-wait latch get requests
->"Pct Misses" for both should be very close to 0.0

Latch	Get Requests	Pct Get Miss	Avg Slps /Miss	Wait Time (s)	NoWait Requests	Pct NoWait Miss
simulator lru latch	144,571	0.0	1.0	0	18	0.0
sort extent pool	1,163	0.0		0	0	
trace latch	19	0.0		0	0	
transaction allocation	50	0.0		0	0	
transaction branch alloc	46	0.0		0	0	
undo global data	13,816	0.0		0	1	0.0
user lock	80	0.0		0	0	

Latch Sleep breakdown for DB: ORCL Instance: orcl Snaps: 1 -2
-> ordered by misses desc

Latch Name	Get Requests	Misses	Sleeps	Spin & Sleeps 1->4
cache buffers lru chain	4,151,271	92	117	0/67/25/0/0
multiblock read objects	625,480	58	66	0/50/8/0/0
cache buffers chains	6,520,208	18	18	0/0/0/0/0
library cache	57,587	10	10	0/10/0/0/0
file number translation ta	19,558	1	1	0/1/0/0/0
simulator lru latch	144,571	1	1	0/1/0/0/0

Latch Miss Sources for DB: ORCL Instance: orcl Snaps: 1 -2
-> only latches with sleeps are shown
-> ordered by name, sleeps desc

Latch Name	Where	NoWait Misses	Sleeps	Waiter Sleeps
cache buffers chains	kcbgtcr: kslbegin excl	0	13	12
cache buffers chains	kcbrls: kslbegin	0	2	1
cache buffers chains	kcbzib: finish free bufs	0	2	0
cache buffers chains	kcbzwb	0	1	3
cache buffers lru chain	kcbzgb: wait	0	90	3
cache buffers lru chain	kcbzar: KSLNBEGIN	0	15	112
cache buffers lru chain	kcbzgm	0	12	0
file number translation	kftts2a	0	1	1
library cache	kgldte: child 0	0	5	0
library cache	kglpnc: child	0	3	8
library cache	kglupc: child	0	2	2
multiblock read objects	kcbzib: mbr get	0	42	40
multiblock read objects	kcbzib: normal mbr free	0	24	26
simulator lru latch	kcbs_lookup_setid	0	1	1

Dictionary Cache Stats for DB: ORCL Instance: orcl Snaps: 1 -2
->"Pct Misses" should be very low (< 2% in most cases)
->"Cache Usage" is the number of cache entries being used
->"Pct SGA"   is the ratio of usage to allocated size for that cache

Cache	Get Requests	Pct Miss	Scan Reqs	Pct Miss	Mod Reqs	Final Usage
dc_free_extents	9	0.0	0		0	1
dc_histogram_defs	212	29.7	0		0	63
dc_object_ids	924	3.1	0		0	211
dc_objects	284	19.4	0		0	460
dc_profiles	20	0.0	0		0	1
dc_rollback_segments	234	0.0	0		0	14
dc_segments	273	16.5	0		0	147
dc_sequences	511	0.8	0		511	7
dc_tablespaces	980	0.0	0		0	6
dc_user_grants	200	6.0	0		0	13
dc_usernames	179	2.8	0		0	9
dc_users	3,425	0.4	0		0	19

```
Library Cache Activity for DB: ORCL  Instance: orcl  Snaps: 1 -2
->"Pct Misses"  should be very low

                                                        Get  Pct        Pin
Namespace                                          Requests  Miss     Requests
--------------------------------------------- ------------ ------ --------------
 Pct            Invali-
 Miss     Reloads  dations
------ ---------- --------
BODY                                                    160    6.3            160
  60.6          0        0
CLUSTER                                                   7    0.0             11
   0.0          0        0
SQL AREA                                              3,937    2.0         15,510
   1.3          7        0
TABLE/PROCEDURE                                       1,510    4.3          2,841
   7.0          3        0
TRIGGER                                                  50    4.0             50
   4.0          0        0
          -------------------------------------------------------------
Shared Pool Advisory for DB: ORCL  Instance: orcl  End Snap: 2
-> Note there is often a 1:Many correlation between a single logical object
   in the Library Cache, and the physical number of memory objects associated
   with it.  Therefore comparing the number of Lib Cache objects (e.g. in
   v$librarycache), with the number of Lib Cache Memory Objects is invalid

                                                                 Estd
Shared Pool    SP       Estd         Estd     Estd Lib LC Time
   Size for  Size  Lib Cache    Lib Cache   Cache Time   Saved  Estd Lib Cache
  Estim (M) Factr   Size (M)      Mem Obj    Saved (s)   Factr    Mem Obj Hits
----------- ----- ---------- ------------ ------------ ------- ---------------
          4    .5          5        1,527          171     1.0          17,801
          8   1.0          7        2,481          171     1.0          17,817
         12   1.5          7        2,481          171     1.0          17,817
         16   2.0          7        2,481          171     1.0          17,817
          -------------------------------------------------------------
SGA Memory Summary for DB: ORCL  Instance: orcl  Snaps: 1 -2

SGA regions                       Size in Bytes
------------------------------ ----------------
Database Buffers                      4,194,304
Fixed Size                              452,992
Redo Buffers                            143,360
Variable Size                        25,165,824
                               ----------------
sum                                  29,956,480
          -------------------------------------------------------------
```

```
SGA breakdown difference for DB: ORCL  Instance: orcl  Snaps: 1 -2

Pool   Name                                 Begin value        End value  % Diff
------ ------------------------------ ---------------- ---------------- -------
shared 1M buffer                              2,098,176        2,098,176    0.00
shared Checkpoint queue                         282,304          282,304    0.00
shared FileIdentificatonBlock                   323,292          323,292    0.00
shared FileOpenBlock                            695,504          695,504    0.00
shared KGK heap                                   3,756            3,756    0.00
shared KGLS heap                              1,573,468        1,102,560  -29.93
shared KQR M PO                                 504,860          534,044    5.78
shared KQR S PO                                 109,072          117,544    7.77
shared KQR S SO                                   1,280            3,340  160.94
shared KSXR large reply queue                   166,104          166,104    0.00
shared KSXR pending messages que                841,036          841,036    0.00
shared KSXR receive buffers                   1,033,000        1,033,000    0.00
shared MTTR advisory                              8,352            8,352    0.00
shared PL/SQL DIANA                           1,977,180          824,776  -58.29
shared PL/SQL MPCODE                            183,432          844,676  360.48
shared PLS non-lib hp                             2,068            2,068    0.00
shared character set object                     330,844          330,844    0.00
shared dictionary cache                       1,610,880        1,610,880    0.00
shared enqueue                                  171,860          171,860    0.00
shared errors                                    23,468           23,468    0.00
shared event statistics per sess              1,718,360        1,718,360    0.00
shared fixed allocation callback                    180              180    0.00
shared free memory                            1,275,960        1,085,252  -14.95
shared joxs heap init                             4,220            4,220    0.00
shared kgl simulator                            578,108          603,524    4.40
shared ksm_file2sga region                      148,652          148,652    0.00
shared library cache                          2,828,032        3,009,736    6.43
shared message pool freequeue                   834,752          834,752    0.00
shared miscellaneous                          3,540,008        4,439,036   25.40
shared parameters                                 7,308            3,880  -46.91
shared sessions                                 410,720          410,720    0.00
shared sim memory hea                            38,108           38,108    0.00
shared sql area                               1,691,548        1,834,540    8.45
shared table definiti                               840            2,800  233.33
shared trigger defini                             2,788            1,384  -50.36
shared trigger inform                             1,076            1,492   38.66
shared trigger source                             1,228              264  -78.50
       buffer_cache                           4,194,304        4,194,304    0.00
       fixed_sga                                452,992          452,992    0.00
       log_buffer                               133,120          133,120    0.00
          -------------------------------------------------------------
```

```
init.ora Parameters for DB: ORCL  Instance: orcl  Snaps: 1 -2

                                                                  End value
Parameter Name                Begin value                       (if different)
----------------------------- --------------------------------- --------------
background_dump_dest          E:\orant\ora92\admin\ORCL\bdump
compatible                    9.2.0.0.0
control_files                 E:\orant\ora92\oradata\ORCL\contr
core_dump_dest                E:\orant\ora92\admin\ORCL\cdump
db_block_size                 4096
db_cache_size                 4194304
db_domain
db_file_multiblock_read_count 16
db_keep_cache_size            0
db_name                       ORCL
db_recycle_cache_size         0
fast_start_mttr_target        10
hash_join_enabled             TRUE
instance_name                 ORCL
java_pool_size                0
large_pool_size               0
log_buffer                    64000
log_checkpoint_timeout        10
open_cursors                  300
pga_aggregate_target          25165824
processes                     150
query_rewrite_enabled         FALSE
remote_login_passwordfile     EXCLUSIVE
shared_pool_reserved_size     1024000
shared_pool_size              8388608
sort_area_size                512
star_transformation_enabled   FALSE
timed_statistics              TRUE
undo_management               AUTO
undo_retention                10800
undo_tablespace               UNDOTBS1
user_dump_dest                E:\orant\ora92\admin\ORCL\udump
workarea_size_policy          MANUAL
          -------------------------------------------------------------

End of Report
```

E

Redundant Arrays of Inexpensive Disks Technology (RAID)

System Hardware Configuration

Storage Subsystem Detail

Storage subsystem performance is one of the most important aspects of tuning an Oracle database server for optimal performance. The architecture and design of the storage subsystem must therefore be considered early in the system design process. In performing the storage subsystem design, system requirements such as the required volume of online transaction data, peak transactions per second load, and system availability are transformed into specific storage subsystem design requirements for storage capacity, peak sustainable I/Os per second, and fault tolerance. Values for design parameters in the selected technology are then chosen to meet these specific requirements.

Modern storage systems offer great flexibility in meeting a wide range of design criteria; technologies such as striping, mirroring, and other fault-tolerant RAID configurations provide the ability to meet these design requirements. Matching the right technology with application I/O characteristics is key to achieving the promised performance and fault tolerance levels; conversely, using the wrong technology for a specific I/O characteristic can lead to I/O bottlenecks and degraded response times. In this section, key parameters in the design of the storage subsystem are described, as well as how they relate to the performance of an Oracle database.

Storage Subsystem Design Parameters and Oracle

When designing a storage subsystem, the available design parameters are weighed against each other until a design solution is achieved that meets or exceeds all design requirements. In the context of an Oracle database server, certain measures are available to specify these requirements. These measures can be categorized under performance, availability, and cost.

Performance

- Random read performance: Important for Oracle indexed or hash-based queries and rollback segment reads
- Random write performance: Important for Oracle DBWn writes; heavy in an OLTP environment, light in a data warehouse
- Sequential read performance: Backups, Oracle full table scans, index creations, parallel queries, temporary segment reads, and recovery from archived redo log files
- Sequential write performance: Oracle LGWR writes, temporary segment writes, direct-path loader writes, tablespace creations
- Impact of concurrency

Availability

- Outage frequency: Expected number of occurrences of a possible outage per unit of time specified in mean time to failure (MTTF)
- Outage duration: The mean time to repair (MTTR) for a given outage event
- Performance degradation during outage: Whether a disk configuration provides service during a fault, and if so, at what level

Storage Subsystem Design Parameters and Oracle (continued)

Cost

- Acquisition cost: The cost of purchasing, installing, and configuring the storage subsystem
- Operational cost: The cost of running and maintaining the system to meet the system availability and service level requirements

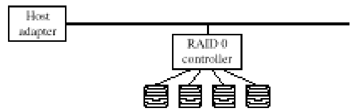
The Redundant Arrays of Inexpensive Disks (RAID) technologies have been developed for nonmainframe, open system solutions. RAID provides low-cost fault tolerance and improved performance. Several levels of RAID are available and can be mixed within one storage subsystem design. Each level of RAID can be categorized against the measures listed above and its impact on Oracle database performance. Some levels of RAID configurations have been available for many years under different names. Key parameters when configuring a RAID system are:

- Array size: The number of drives in the array
- Disk size: The size of each disk
- Stripe size: The size of an I/O chunk, written to or read from a contiguous location on a disk (Striping allows data files to be interleaved and spread across the disks in an array in an attempt to parallelize file I/O.)

The throughput of a storage subsystem, expressed in I/Os per second, determines how many transactions can be processed by the subsystem before queuing delays begin to occur. If the I/Os-per-second requirement of the application is known, broken down into reads per transaction, writes per transaction, and transactions per second, you can determine whether a particular RAID configuration will support the transaction rate applied by an application. To calculate throughput, or total sustainable I/Os-per-second load that a RAID array can support, simply multiply the number of drives in the array times the sustainable I/Os per second of one drive, currently in the approximate range of 35 to 50 I/Os per second. Different RAID configurations, however, add to the I/O load on a disk array applied by the application in order to provide the fault tolerance function. Once the total I/O load on the array is known, the number of drives required to support the load can be found. Simply divide the total I/Os per second load by the random I/O throughput rating for the selected drive.

The sections below briefly describe each RAID level and some of its characteristics. For each level, an equation is provided to calculate the total I/Os per second load on a RAID array, to illustrate the impact of the RAID configuration on the array's capacity. Also provided is an equation for calculating the size of the disk drives required for an array.

RAID Level 0, Nonredundant Striping



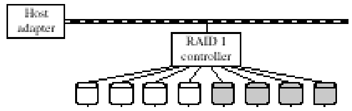
RAID 0 refers to simple data striping of multiple disks into a single logical volume, and has no fault tolerance. When properly configured, it provides excellent response times for high concurrency random I/O and excellent throughput for low concurrency sequential I/O. Selection of the array and stripe sizes requires careful consideration in order to achieve the promised throughput. For RAID 0, the total I/Os-per-second load generated against the array is calculated directly from the application load, because there is no fault tolerance in this configuration:

- Total I/O per second load on array = (reads/transaction + writes/ transaction) * transactions/second

The size of each drive in the array can be calculated from the online volume requirements as follows:

- Drive size = [total space required by application / number of drives in array] Rounded up to next drive size.
- Below is a summary of RAID 0 characteristics:
  - Random read performance: Excellent under all concurrency levels if each I/O request fits within a single striping segment
  - Random write performance: Same as random read performance
  - Sequential read performance: Excellent with fine-grained striping at low concurrency levels
  - Sequential write performance: Same as sequential read performance
  - Outage frequency: Poor; any single disk failure will cause application outage
  - Outage duration: Poor; the duration of a RAID 0 outage is the time required to detect the failure, replace the disk drive, and perform Oracle media recovery
  - Performance degradation during outage: Poor; any disk failure causes all applications requiring use of the array to crash
  - Acquisition cost: Excellent, because there is no redundancy; you buy only enough for storage and I/Os per second requirements
  - Operational cost: Fair to poor; frequent media recoveries increase operational costs and may outweigh the acquisition cost advantage

RAID Level 1, Mirroring



RAID Level 1, or disk mirroring, provides the best fault tolerance of any of the RAID configurations. Each disk drive is backed up by an exact copy of itself on an identical drive. A storage subsystem of mirrored drives can continue at full performance with a multiple disk failure as long as no two drives in a mirrored pair have failed. The total I/Os per load applied to a mirrored pair is calculated as follows:

> Total I/O per second load on array = (reads/transaction + 2*writes/ transaction) * transactions/second

Note the two multiplier of the writes/transaction factor. This is due to the fact that each write request by an application to a mirrored pair actually results in two writes, one to the primary disk and one to the backup disk. The size of the drive required is:

> Drive size = [total space required by application / number of drives in array/2] Rounded up to next drive size.

In the simplest RAID 1 configuration, the number of drives in the array is two: the primary drive and its backup. The definition of RAID 1, however, includes the ability to expand the array in units of two drives to achieve a striped and mirrored configuration. Striping occurs in an array of four or more disks. Some industry literature (for example, Millsap,1996) refers to striped and mirrored configurations as RAID 0 + 1. The Compaq hardware used as an example configuration in this document supports both configurations. Compaq uses only the RAID 1 term to describe all 100% mirrored configurations in arrays of even-numbered disks. Because the performance of a simple two-drive RAID 1 pair is somewhat different from a striped and mirrored array, the figures for striped and mirrored are presented separately under the RAID 0 + 1 section.

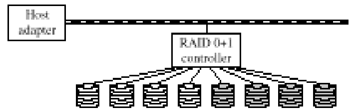
Below is a summary of characteristics of the two-disk array RAID configuration:

- Random read performance: Good; if the implementation uses read-optimized RAID 1 controllers, which read from the drive with the smallest I/O setup cost, then slightly better than an independent disk
- Random write performance: Good (Application write requests are multiplied by two, because the data must be written to two disks. Thus, some of the I/Os-per-second capacity of the two drives is used up by the mirroring function.)
- Sequential read performance: Fair; throughput is limited to the speed of one disk
- Sequential write performance: Fair; same factors as are influencing the random write performance
- Outage frequency: Excellent
- Outage duration: Excellent; for “hot swapable” drives, no application outage is encountered by a single failure

RAID Level 1, Mirroring (continued)

- Performance degradation during outage: Excellent; there is no degradation during a disk outage (After replacing of the failed drive, the resilvering operation that takes place when the failed disk is replaced will consume some of the available I/Os per second capacity.)
- Acquisition cost: Poor; each RAID 1 pair requires two drives to achieve the storage capacity of one.
- Operational cost: Fair; increased complexity of the configuration leads to higher training costs and costs to develop custom software to integrate the mirroring procedures into scheduled maintenance operations.

RAID Level 0+1, Striping and Mirroring

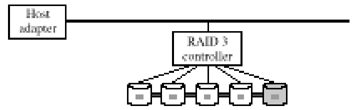


As noted in the previous section, the striped and mirrored configuration is an expansion of the RAID 1 configuration from a simple mirrored pair to an array of even-numbered drives. This configuration offers the performance benefits of RAID 0 striping and the fault tolerance of simple RAID 1 mirroring. The striped and mirrored configuration is especially valuable for Oracle data files holding files with high write rates, such as table data files and online and archived redo log files. Unfortunately, it also presents the high costs of simple RAID 1. The equations for the total I/Os per second and disk drive size calculations for RAID 0 + 1 are identical to those presented for RAID 1 above. The Compaq SMART array controller used in the example configuration supports RAID 0 + 1 (RAID 1 in Compaq terminology) in arrays up to 14 drives, providing the effective storage of 7 drives.

Below is a summary of characteristics of RAID 0 + 1 storage arrays:

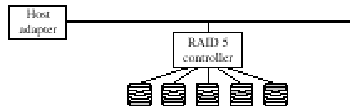
- Random read performance: Excellent under all concurrency levels if each I/O requests fits within a single striping segment (Using a stripe size that is too small can cause dramatic performance breakdown at high concurrency levels.)
- Random write performance: Good (Application write requests are multiplied by two because the data must be written to two disks. Thus, some of the I/Os-per-second capacity of the two drives is used up by the mirroring function.)
- Sequential read performance: Excellent under all concurrency levels if each I/O request fits within a single striping segment
- Sequential write performance: Good
- Outage frequency: Excellent; same as RAID 1
- Outage duration: Excellent; same as RAID 1
- Performance degradation during outage: Excellent; there is no degradation during a disk outage (The resilvering operation that takes place when the failed disk is replaced will consume a significant amount of the available I/Os-per-second capacity.)
- Acquisition cost: Poor; same as RAID 1
- Operational cost: Fair; same as RAID 1

RAID Level 3, Bit Interleaved Parity



In the RAID 3 configuration, disks are organized into arrays in which one disk is dedicated to storage of parity data for the other drives in the array. The stripe size in RAID 3 is 1 bit. This enables recovery time to be minimized, because data can be reconstructed with a simple exclusive-OR operation. However, using a stripe size of 1 bit reduces I/O performance. RAID 3 is not recommended for storing any Oracle database files. Also, RAID 3 is not supported by the Compaq SMART array controllers.

RAID Level 5, Block-Interleaved with Distributed Parity



RAID 5 is similar to RAID 3, except that RAID 5 striping segment sizes are configurable, and RAID 5 distributes parity across all the disks in an array. A RAID 5 striping segment contains either data or parity.

Battery-backed cache greatly reduces the impact of this overhead for write calls, but its effectiveness is implementation-dependent. Large write-intensive batch jobs generally fill the cache quickly, reducing its ability to offset the write-performance penalty inherent in the RAID 5 definition.

The total I/Os per second load applied to a RAID 5 array is calculated as follows:

Total I/O per second load on array = (reads/transaction + 4*writes/ transaction) * transactions/second

The writes/transaction figure is multiplied by four because the parity data must be written in a six-step process:

1. Read the data drive containing the old value of the data to be overwritten. This requires one I/O.
2. Read the parity drive. This requires one I/O.
3. Subtract the contribution of the old data from the parity value.
4. Add the contribution of the new data to the parity value.
5. Write the new value of the parity requiring one I/O.
6. Write the new data value to the data drive. This requires one I/O.

Summing up all I/Os in this process yields four I/Os required for each write requested by the application. This is the main reason that RAID 5 is not recommended for storing files with a high I/O performance requirement; the 4 multiplier reduces the effective I/Os-per-second capacity of the array.

The size of the drive required is:

Drive size = [total space required by application /(total number drives - number of arrays)] Rounded up to next drive size.

RAID Level 5, Block-Interleaved with Distributed Parity (continued)

Note that the figure "number of arrays" is used to account for the space of one drive per array consumed by the parity data. If it is necessary to exceed the maximum recommended array size to meet the I/Os-per-second performance requirement, then multiple arrays are required.

Below is a summary of the characteristics of RAID 5 storage arrays:

- Random read performance: Excellent under all concurrency levels if each I/O requests fits within a single striping segment (Using a stripe size that is too small can cause dramatic performance breakdown at high concurrency levels.)
- Random write performance: Poor; worst at high concurrency levels (The read-modify-write cycle requirement of RAID 5 parity implementation reduces the effective throughput or I/Os-per-second capacity of the array, especially for heavy write I/O files. It should be noted, however, that under light load, response time is not degraded from that provided by a faster array configuration such as RAID 0. This is due to the asynchronous write capabilities provided by most array controllers. With asynchronous write, the application does not have to wait until the storage subsystem has completed the read-modify-write cycle before continuing. Instead, the controller buffers the data in battery-backed RAM, signals the application that the write has completed, then completes the write to disk. (This buffering does nothing to increase throughput, however.)
- Sequential read performance: Excellent under high concurrency levels if each I/O request fits within a single striping segment; also excellent with fine grain striping under low concurrency levels
- Sequential write performance: Fair for low concurrency levels, poor for high concurrency levels (See random write performance.)
- Outage frequency: Good; can withstand the loss of any single disk in a given array without incurring an application outage (Multiple simultaneous disk failures causes application outage. The possibility of multiple simultaneous failures increase as the size of the array increases.)
- Outage duration: Good; a single disk failure causes no application outage
- Performance degradation during outage: Fair; there is no degradation for reads and writes to or from surviving drives in the array (Reads and writes to a failed drive incur a high performance penalty, requiring data from all surviving drives in the array to be read. Reconstruction of the failed drive's data also degrades performance.)
- Acquisition cost: Fair (If storage capacity were the only factor, the cost would be g/(g-1) times the cost of the equivalent RAID 0 capacity, where g is the number of disks in the array. However, when factoring I/Os-per-second performance requirement, the cost can meet or exceed the cost of a RAID 0 + 1 implementation.)
- Operational cost: Fair (Training is required to configure striped disk arrays for optimal performance.)

Ranking of RAID Levels Against Oracle File Types

The following table provides relative rankings for RAID configurations for specific Oracle file types. The rankings range from 1 (best) to 5 (worst). Adapted from Millsap,1996, page 13.

Query	None	0	1	0+1	3	5
Control file performance	2	1	2	1	5	3
Redo log file performance	4	1	5	1	2	3
System tablespace performance	2	1	2	1	5	3
Sort segment performance	4	1	5	1	2	3
Rollback segment performance	2	1	2	1	5	5
Indexed read-only data files	2	1	2	1	5	1
Sequential read-only data files	4	1	5	1	2	3
DBWn intensive data files	1	1	2	1	5	5
Direct load-intensive data files	4	5	1	1	2	2
Data protection	4	5	1	1	2	2
Acquisition and operating costs	1	1	5	5	3	3

F

Tuning Undo Segments

Objectives

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

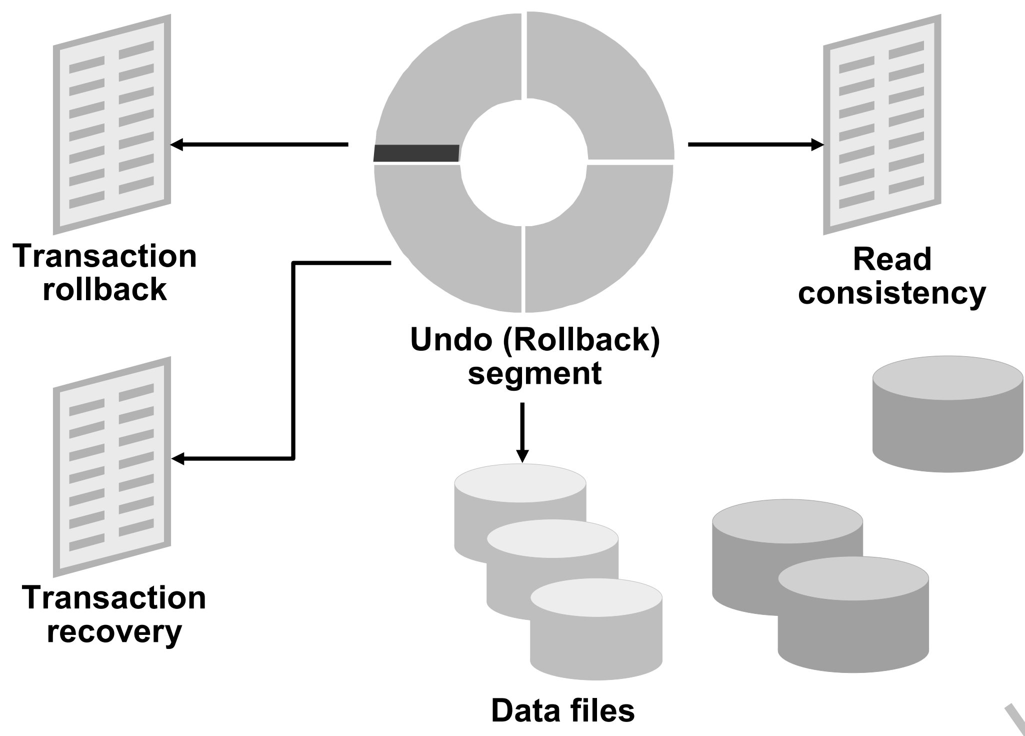
- **Describe the concept of automatic undo management**
- **Create and maintain the automatic managed undo tablespace**
- **Set the retention period**
- **Use dynamic performance views to check rollback segment performance**
- **Reconfigure and monitor rollback segments**
- **Define the number and sizes of rollback segments**
- **Allocate rollback segments to transactions**

Objectives

This lesson helps you understand the automatic undo management feature. You will also learn to configure automatic undo management in an Oracle database.

Good rollback segment configuration is crucial to a well-tuned Oracle database. This lesson helps you to recognize and solve problems arising from inappropriate numbers or sizes of rollback segments.

Undo Segments: Usage



Undo Segments: Usage

Transaction Rollback

When a transaction makes changes to a row in a table, the old image is saved in an undo segment, also called a rollback segment. If the transaction is rolled back, the value in the undo segment is written back to the row, restoring the original value.

Transaction Recovery

If the instance fails when transactions are in progress, then the Oracle server rolls back the uncommitted changes when the instance is restarted.

Read Consistency

Read consistency consists of the following conditions:

- When transactions are in progress, other sessions in the database should not see any uncommitted changes.
- A query statement should not see any changes that are made, committed or uncommitted, after the statement commenced execution.
- DML statements do not see any uncommitted changes. However, the changed rows are locked by the transaction making the changes and the lock is released only when the transaction ends.

The old values in the undo segments, also referred to as undo information, are used to provide the read-consistent image.

Using Less Undo Space Per Transaction

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

Transactions

You may be able to reduce undo space wastage by training users and developers to do the following:

- Users should commit work regularly so that their transactions do not lock resources for longer than required.
- Developers should not code unnecessarily long transactions.

Using Less Undo Space

- **Import**
  - **Set `COMMIT = Y`**
  - **Size the set of rows with the `BUFFER` keyword**
- **Export: Set `CONSISTENT = N`**
- **SQL*Loader operations: Set the commit intervals with `ROWS`**
- **Developers should make sure that the transactions are not unduly long**

Using Less Undo Space

Import

- Set `COMMIT = Y` to make sure that each set of inserted rows is committed as the import goes on.
- Size the set of rows with the `BUFFER` keyword.

Export

The `CONSISTENT` option specifies whether or not Export uses the `SET TRANSACTION READ ONLY` statement to ensure that the data seen by Export is consistent to a single point in time and does not change during the execution of the export command.

You should specify `CONSISTENT = Y` when you anticipate that other applications will be updating the target data after an export has started. However, this will mean that any modified data must be kept in the undo space until required by the export process. If this data is not available an error will be reported and the export process will abort.

Setting `CONSISTENT = N` prevents the transaction from being set as read-only.

SQL*Loader

For conventional path loading, set the commit intervals with the `ROWS` keyword.

Automatic Undo Management

- The automatic undo management feature simplifies the management of undo segments.
- Set the `UNDO_MANAGEMENT` parameter to:
  - `AUTO` for automatic undo management
  - `MANUAL` for managing rollback segments manually
- The `UNDO_RETENTION` parameter specifies the time (in seconds) to retain undo information.

Automatic Managed Undo

The automatic undo management (AUM) feature, manages undo space in Oracle databases.

You have the choice of using manually managed rollback segments or automatic undo management. Set the `UNDO_MANAGEMENT` initialization parameter to:

- Auto to enable the instance to manage rollback segments automatically (AUM)
- Manual to create and manage rollback segments manually (RBU)

When the database is set to use auto-managed undo, you do not need to perform many explicit actions for management of the undo space. The Oracle server, with no management required from the DBA, maintains the number and size of the undo segments. When the first DML operation is executed within a transaction, the transaction is assigned to an undo segment in the current undo tablespace.

The DBA must create and size the undo tablespace. You can specify the amount of undo information retained in the auto-managed undo segments by using the `UNDO_RETENTION` parameter. The size of the tablespace should be large enough to accommodate the amount of undo generated during the time specified by `UNDO_RETENTION`.

Automatic Undo Management Tablespaces

- **Create a tablespace for automatic undo management in one of the following ways:**
  - **Using the `UNDO TABLESPACE` clause in the `CREATE DATABASE` command**
  - **Using the `CREATE UNDO TABLESPACE` command**
- **`MINIMUM EXTENT` and `DEFAULT STORAGE` are system generated for undo tablespaces.**
- **Restrictions:**
  - **Database objects cannot be created in this tablespace.**
  - **You can specify the data file and the `extent_management` clause only.**

Tablespace for Automatic Undo Management

The `UNDO_MANAGEMENT` initialization parameter must be set to Auto.

You can create an auto-managed undo tablespace when creating the database. To do this you use the `UNDO TABLESPACE` clause to specify the name, data file, size, and block size of the undo tablespace.

If you do not specify the `UNDO TABLESPACE` clause when creating the database but have set `UNDO_MANAGEMENT` to Auto, then:

- An undo tablespace with the name `SYS_UNDOTBS` is created.
- On a UNIX system the data file with name `DBU1<ORACLE.SID>.dbf` is placed in the `$ORACLE_HOME/dbs` folder.
- `AUTOEXTEND` is set to On.

You can also create an undo tablespace with the `CREATE UNDO TABLESPACE` command.

Set the `UNDO_TABLESPACE` initialization parameter to the name of the undo tablespace that you want the database to use on startup.

Altering an Undo Tablespace

- The `ALTER TABLESPACE` command can be used to make changes to undo tablespaces.
- The following example adds another data file to the undo tablespace:

```
ALTER TABLESPACE undotbs1
ADD DATAFILE '/u02/oradata/testdb/undotbs1_02.dbf'
AUTOEXTEND ON;
```

- You cannot take an undo tablespace offline that has an active undo segment.

Altering an Undo Tablespace

The following clauses are supported when altering an undo tablespace:

- ADD DATAFILE
- RENAME
- DATAFILE [ONLINE|OFFLINE]
- BEGIN BACKUP
- END BACKUP

This example depicts the addition of a data file to an existing undo tablespace:

```
SQL> ALTER TABLESPACE undotbs1
  2  ADD DATAFILE '/u02/oradata/testdb/undotbs1_02.dbf'
  3  AUTOEXTEND ON;
```

Switching Undo Tablespaces

- **A DBA can switch from using one undo tablespace to another.**
- **Only one undo tablespace per instance can be assigned as active.**
- **Switching is performed by using the `ALTER SYSTEM` command:**

```
ALTER SYSTEM SET UNDO_TABLESPACE=undotbs2;
```


Number of Active Undo Tablespaces

At any given moment of time, there can be only one active undo tablespace. However, an instance may have more than one undo tablespace in use per instance. If an instance has two undo tablespaces (`undotbs1` and `undotbs2`), only one can be active (in this example: `undotbs1`). This means that all new transactions must use this tablespace to store any undo data.

If the DBA switches the undo tablespace using the command:

```
SQL> ALTER SYSTEM SET UNDO_TABLESPACE=undotbs2;
```

then all new transactions are directed to the undo tablespace, `undotbs2`; however, all current transactions (that is, those already assigned to `undotbs1`) will continue to use the undo tablespace `undotbs1` until the transaction completes.

Dropping an Undo Tablespace

The `DROP TABLESPACE` command can be used to drop an undo tablespace:

```
DROP TABLESPACE undotbs_2;
```

- **An undo tablespace can be dropped only if:**
  - **It is not the active undo tablespace**
  - **It is not utilized by an active transaction**
- **Queries that require a read-consistent image of undo data that is stored in an dropped undo tablespace will return an error.**

Dropping an Undo Tablespace

An undo tablespace can be dropped only if:

- It is not currently used by any instance.
- Its transaction tables do not contain any uncommitted transactions.

The `DROP TABLESPACE` undo tablespace name command behaves the same as `DROP TABLESPACE` tablespace name `INCLUDING CONTENTS`.

Setting UNDO_RETENTION

UNDO_RETENTION parameter:

- Is specified in time (seconds)
- A target value. If space is required, then committed data will be overwritten.
- Controls the amount of undo data that is to be retained after committing

Setting UNDO_RETENTION

UNDO_RETENTION controls the amount of committed undo information to retain. You can use UNDO_RETENTION to satisfy queries that require old undo information to rollback changes to produce older images of data blocks. You can set the value at instance startup. The value should be large enough to cover any long running queries that require a read- consistent image of data that was changed since starting the query.

The UNDO_RETENTION parameter works best if the current undo tablespace has enough space for all of the transactions during an UNDO_RETENTION period. If an active transaction needs undo space and the undo tablespace does not have any free space, then the database starts reusing undo space that would have been retained due to UNDO_RETENTION. This may cause long queries to fail due to SNAPSHOT TOO OLD errors. Be sure to allocate enough space in the undo tablespace to satisfy the space requirement for the current setting of the UNDO_RETENTION parameter.

The UNDO_RETENTION parameter value can also be changed dynamically using the ALTER SYSTEM command. The effect of the UNDO_RETENTION parameter is immediate, but it can be honored only if the current undo tablespace has enough space for the active transactions.

Setting `UNDO_RETENTION` (continued)

`UNDO_RETENTION` is specified in units of seconds, with the default value of 900 seconds. Because undo segments are on disk, they can survive system crashes.

Space Requirement for Undo Tablespace

You can use the following query to set the `UNDO_RETENTION` parameter and size the undo tablespace:

```
SQL> SELECT (rd * (ups * overhead) + overhead) AS "Bytes"
     FROM
         (SELECT value AS RD
          FROM v$parameter
          WHERE  name = 'undo_retention'),
         (SELECT (SUM(undoblks) /
                  SUM( ((end_time - begin_time) * 86400)))
                  AS UPS
          FROM   v$undostat),
         (SELECT value AS overhead
          FROM v$parameter
          WHERE  name = 'db_block_size');
```

Other Parameters for Automatic Undo Management

- `UNDO_MANAGEMENT`: **Specifies whether the database uses Auto or Manual mode**
- `UNDO_TABLESPACE`: **Specifies a particular undo tablespace to be used**
- `UNDO_SUPPRESS_ERRORS`: **Set to True, this parameter suppresses errors while attempting to execute manual operations, such as `ALTER ROLLBACK SEGMENT ONLINE`, while in Auto mode**

Other Parameters for System Managed Undo

The following parameters are used with the system managed undo feature:

- **`UNDO_MANAGEMENT`:** Specifies what mode of undo management to use. The parameter can be reassigned when the database is open.
  - If set to Auto, the system managed undo feature is used. Make sure that you have already created an undo tablespace.
  - A value of Manual means that the rollback segments are managed by the DBA.
- **`UNDO_TABLESPACE`:** Specifies the name of the undo tablespace
  - If the database is in SMU mode and the `UNDO_TABLESPACE` parameter is omitted at startup, the first available undo tablespace in the database is chosen.
  - If no undo tablespace is available, the instance starts without an undo tablespace using the `system` rollback segment. Make sure that an undo tablespace is available immediately thereafter.
  - To replace one active undo tablespace with another, you can use the `ALTER SYSTEM SET UNDO_TABLESPACE` … command.

Other Parameters for System Managed Undo (continued)

Parameters for Automatic Managed Undo

- **`UNDO_SUPPRESS_ERRORS`:** Primarily meant for applications and tools that use statements such as `SET TRANSACTION USE ROLLBACK SEGMENT`. Setting this parameter enables users to use the SMU feature before all application programs and scripts are converted to SMU mode. So at the beginning of such sessions, you can suppress the (ORA 30019) error by adding the command:

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

Space Requirement for Undo Tablespace

Given a specific `UNDO_RETENTION` parameter setting and some system statistics, the amount of undo space required to satisfy the undo retention requirement can be estimated using the formula:

```
Undo Space = (UNDO_RETENTION * (Undo Blocks Per Second *
                DB_BLOCK_SIZE) ) + DB_BLOCK_SIZE
```

Monitoring Automatic Undo Management

- **Use `v$undostat` view to monitor undo segments.**
- **This view is available for both Manual and Auto mode.**
- **The `undoblks` column displays the number of undo blocks allocated.**

Monitoring Automatic Managed Undo

Use the `v$undostat` view to monitor space allocation and usage for automatically managed undo. Each row in the view keeps statistics collected in the instance for a 10-minute interval.

You can use this view to estimate the amount of undo space required for the current workload. This view is available in both SMU and RBU modes.

Using v$undostat

```
SQL> SELECT begin_time, end_time, undoblks,
  2  txncount, maxquerylen
  3  FROM v$undostat;

BEGIN_TIME      END_TIME        UNDOBLKS TXNCOUNT
--------------- --------------- -------- --------
25-oct-01:06:04 25-oct-01:06:14      234       12
25-oct-01:05:44 25-oct-01:05:54      587       21
25-oct-01:05:34 25-oct-01:05:44    1,187       45
25-oct-01:05:24 25-oct-01:05:34      346       15
25-oct-01:05:14 25-oct-01:05:24      642       23
……
```

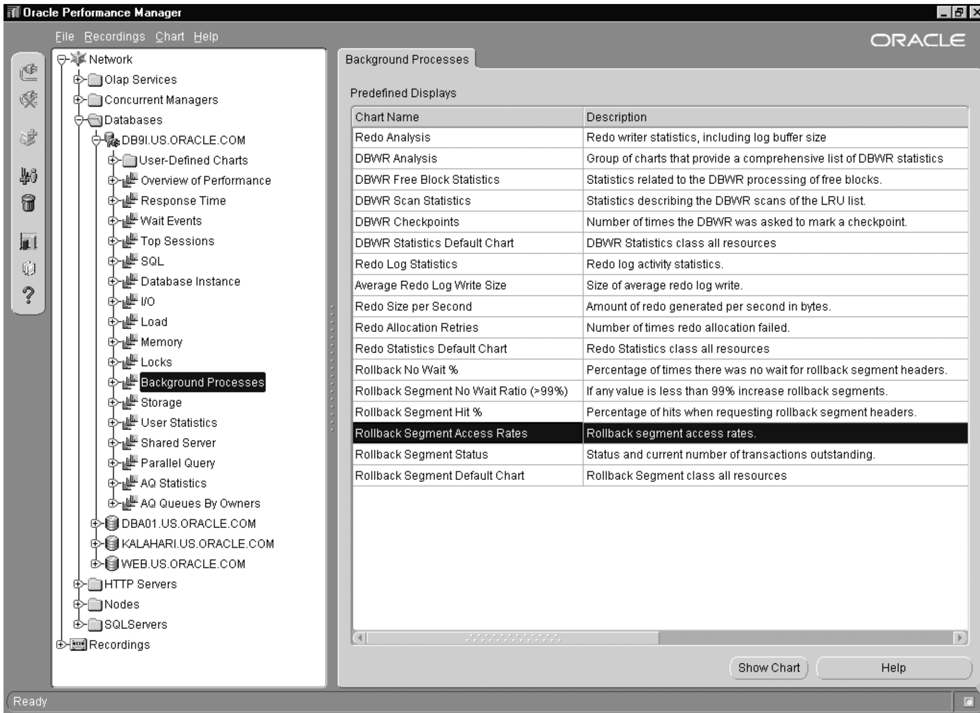

Using v$undostat

The example on the slide shows that the peak undo consumption occurred between 05:34 and 15:44; 1,187 undo blocks were consumed in 10 minutes (or about 2 blocks per second). Also, the highest transaction concurrency occurred during that same period, with 45 transactions executing at the same time.

Two aspects can be tuned under automatic undo management:

- The size of the undo tablespace
- The amount of time that undo blocks are retained before being overwritten

Performance Manager: Rollback/Undo



Performance Manager: Rollback/Undo

The Background Processes set of charts provide the DBA with information on rollback or undo segments.

If using undo segments, you may need to resize the tablespace using the Oracle Enterprise Manager console. The Tablespace Manager will not show information regarding undo tablespaces.

Overview

Setting manual rollback segments is:

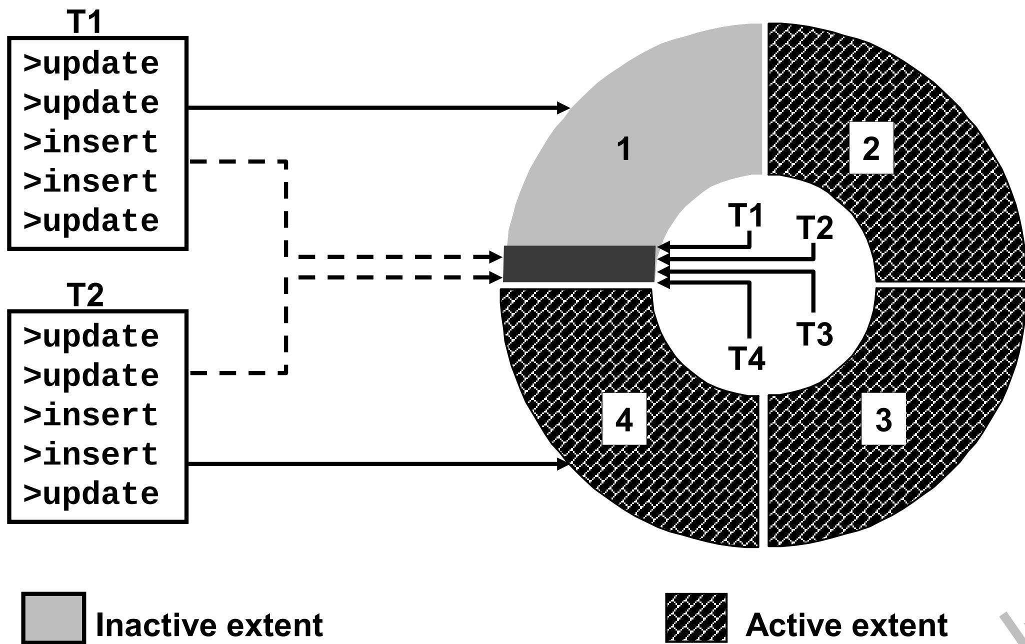
- **Optional in the Oracle database**
- **Time consuming for the DBA**

Overview

When you use System Managed Undo the Oracle database automatically manages the number and size of the undo segments in an undo tablespace. You cannot create, drop, or resize these segments manually.

If you want to have full control, at the cost of more administrative tasks, then you can revert to the pre-Oracle9*i* database behavior by setting the `UNDO_MANAGEMENT` parameter to Manual.

Rollback Segment Activity



Rollback Segment Activity

Active and Inactive Extents

Transactions use extents of a rollback segment in an ordered, circular fashion, moving from one extent to the next after the current extent is full. A transaction writes a record to the current location in the rollback segment and advances the current pointer by the size of the record.

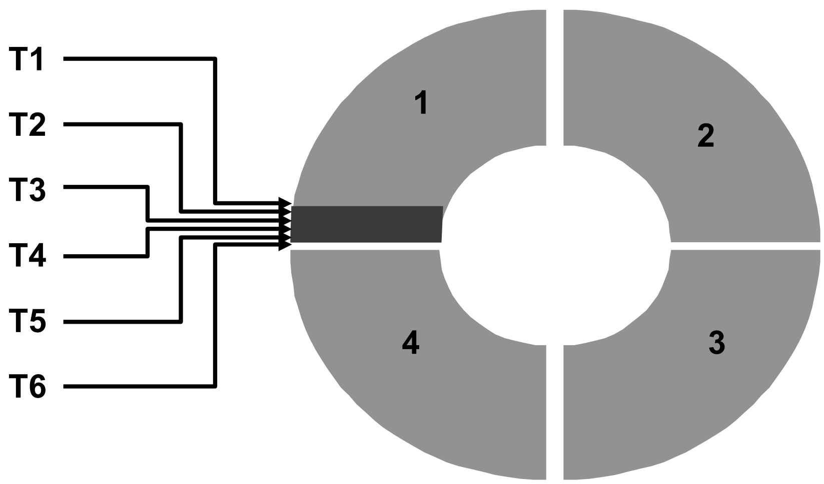
Note: More than one transaction can write to the same extent of a rollback segment. Each rollback segment block contains information from only one active transaction.

Rollback Segment Activity

Writing to rollback segments requires that the corresponding undo data is available in the database buffer cache. To maintain large amounts of undo information, the buffer cache should be quite large or there is a higher number of physical I/Os.

Rollback Segment Header Activity

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



Rollback Segment Header Activity

The Oracle server keeps a transaction table in the header of every rollback segment.

The rollback segment header activity controls the writing of changed data blocks to the rollback segments. Because the rollback segment header is a data block and it is frequently modified, the rollback segment header block remains in the data block buffer cache for long periods of time. Therefore, accesses to the rollback segment header block increase the hit ratio for the application, even though it is not related to the data blocks.

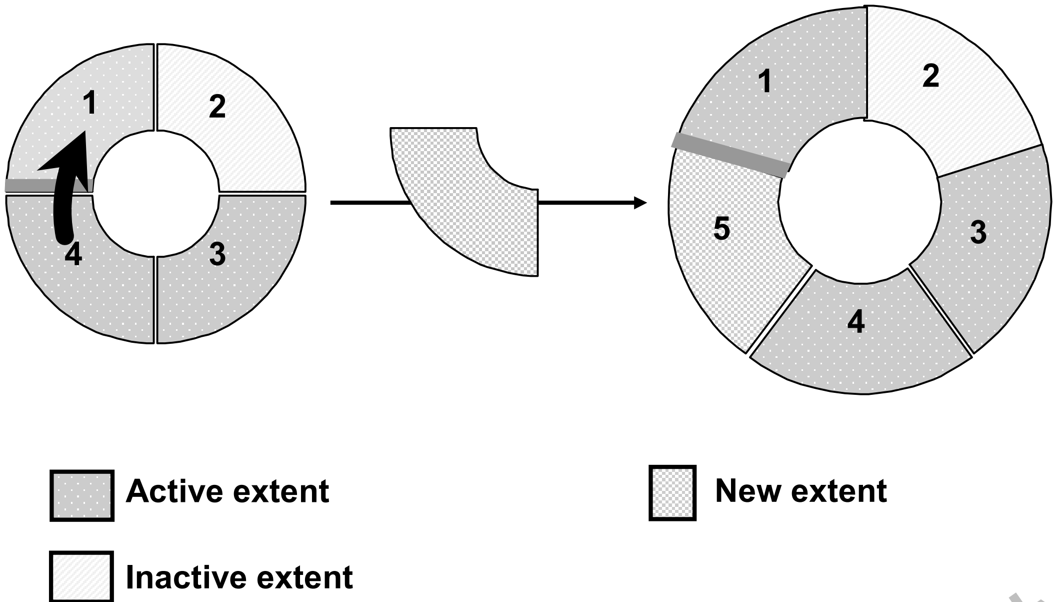
The Impact of Rollback Segment Header Activity

The impact of the rollback segment header activity on the cache hit ratio is important for OLTP systems that feature many small transactions.

Every transaction must have update access to the transaction table for its rollback segment. You need enough rollback segments to prevent transactions from contending for the transaction table.

If you underestimate the number of rollback segments needed, performance is degraded and transactions may generate errors. If you overestimate, you use unnecessary space.

Growth of Rollback Segments



Growth of Rollback Segments

The pointer or the head of the rollback segment moves to the next extent when the current extent is full. When the last extent that is currently available is full, the pointer can move back to the beginning of the first extent only if that extent is free. The pointer cannot skip over an extent and move to the second or any other extent.

If the first extent is being used, then the transaction allocates an additional extent for the rollback segment. This is called an extend. Similarly, if the head tries to move into an active extent, the rollback segment allocates an additional extent.

The Impact of Rollback Segment Extending

Rollback segments should not be extended during normal running. To prevent this, rollback segments must have enough extents to hold the rollback entries for the transactions.

As with other objects, you should avoid dynamic space management.

If you underestimate the size of rollback segments, performance is degraded and transactions may generate errors. If you overestimate you use unnecessary space and some performance issues may arise from having rollback segments that are too large.

Tuning Manually Managed Rollback Segments

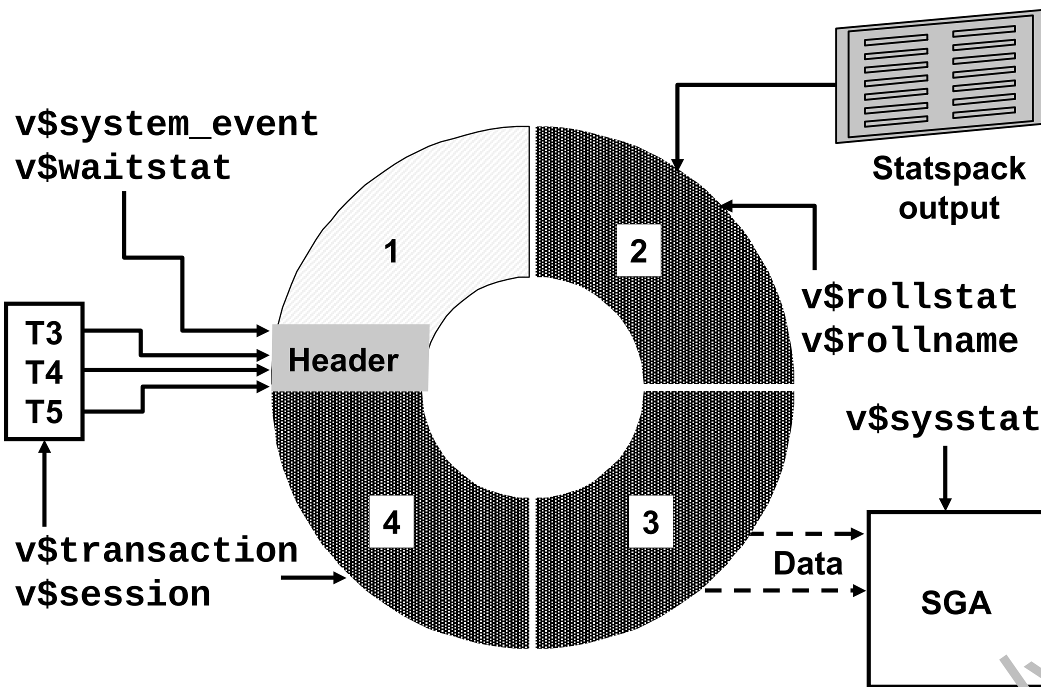
Goals in tuning rollback segments:

- **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 per transaction.**
- **No transaction should ever run out of rollback space.**
- **Readers should always see the read-consistent images they need.**

Tuning Manually Managed Rollback Segments

- Transactions should never wait for access to rollback segments. This requires you to have enough rollback segments.
- Rollback segments should not extend during normal running. This requires:
  - An appropriate number of extents per segment
  - The correct sizing of the extents
  - The appropriate number of rollback segments
  - Using less rollback per transaction, by committing more frequently
- No transaction, however large or exceptional, should ever run out of rollback space. This means that rollback segments should be sized correctly.
- For large transactions investigate whether these could be split into smaller transactions, by committing more frequently.
- Readers should always be able to see the read-consistent images they need. This requires the appropriate:
  - Number of rollback segments
  - Sizing of rollback segments

Diagnostic Tools



Dynamic Views to Monitor Rollback Activity

- `v$rollname`: Displays the name and number of the online rollback segments
- `v$rollstat`: Displays statistics of the activity for each online rollback segment:
  - Number of waits on the header transaction table
  - Volume of data written by the transactions
- `v$system_event`: The Undo Segment Tx Slot event shows waits for transaction slots and therefore contention on rollback segment headers.
- `v$waitstat`: Displays the cumulative statistics of waits on header blocks and data blocks of all rollback segments
- `v$sysstat`: Displays the number of consistent and data block gets. You can compare the number of waits with the total number of requests for data.
- `v$transaction`: Displays the current transactions using rollback segments and therefore the number of rollback segments required

Except for `v$rollname`, all of these views use the undo segment number (USN) as the identifier for rollback. So when you need to get the name of the rollback segment, join the `v$rollname` on `usn` column.

Diagnosing Contention for Manual Rollback Segment Header

If the number of waits for any rollback header is greater than 1% of the total number of requests, then create more rollback segments.

```
SQL> SELECT class, count FROM v$waitstat
  2  WHERE class LIKE '%undo%';
or
SQL> SELECT event, total_waits, total_timeouts
  2  FROM v$system_event
  3  WHERE event LIKE 'undo segment tx slot';
or
SQL> SELECT sum(waits)* 100 /sum(gets) "Ratio",
  2    sum(waits) "Waits", sum(gets) "Gets"
  3  FROM   v$rollstat;
```


Diagnosing Contention for Rollback Segment Header

A nonzero value in the following indicates contention for rollback segments:

- Waits column of the `v$rollstat` view
- Undo header row of the `v$waitstat` view
- Undo Segment Tx Slot event of the `v$system_event` view

The following statement queries the `v$waitstat` view to look for contention on the rollback segment:

```
SQL> SELECT class, count
  2  FROM v$waitstat
  3  WHERE class LIKE ‘%undo%‘;
```

Diagnosing Contention for Rollback Segment Header (continued)

The rollback and undo related information from the Statspack report is located mainly in the Rollback Segment Stats section. Following is an example of the Rollback Segment Stats section:

```
Rollback Segment Stats for DB: ED31 Instance: ed31 Snaps: 1 –
2
->A high value for "Pct Waits" suggests more rollback
segments may be required

Trans Table      Pct   Undo Bytes
RBS No   Gets Waits  Written Wraps Shrinks Extends
------   ---- -----  ------- ----- ------- --------
     0    5.0  0.00        0     0       0        0
     1   66.0  0.00    5,636     0       0        0
     2  439.0  0.00  358,772     5       0        0
     3   50.0  0.00    6,314     0       0        0
     4   53.0  0.00    7,004     0       0        0
---------------------------------------------------
```

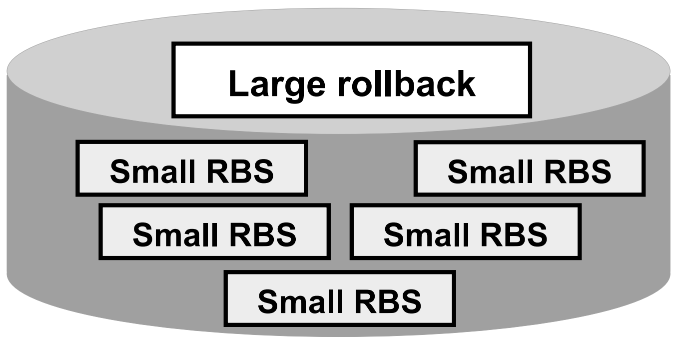
Guideline

When you observe contention for rollback segments, you should investigate further the cause of contention to determine whether the mere addition of rollback segments would alleviate the problem or whether it is necessary to configure the rollback tablespaces to manage the I/O.

Guidelines: Number of Manual Rollback Segments (RBSs)

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

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



Guidelines: Number of Manual Rollback Segments (RBSs)

OLTP Transactions

- OLTP applications are characterized by frequent concurrent transactions, each of which modifies a small amount of data. Assign small rollback segments to OLTP transactions.
- The reasonable rule of thumb is one rollback segment for every four concurrent transactions.

Long Batch Transactions

Assign large rollback segments to transactions that modify large amounts of data. Such transactions generate large rollback entries. If a rollback entry does not fit into a rollback segment, the Oracle server extends the segment. Dynamic extension reduces performance and should be avoided whenever possible.

Allow for the growth of the rollback segments by creating them in large or auto-extending tablespaces, with an unlimited value for `MAXEXTENTS`.

Guidelines: Number of Manual Rollback Segments (RBSs) (continued)

Long Batch Transactions (continued)

For exceptionally long transactions, you may want to assign a large rollback segment using the following syntax:

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

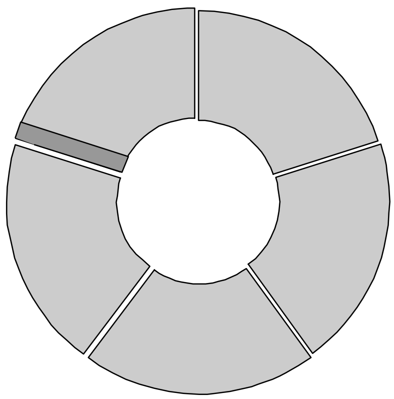
You can also use a supplied procedure:

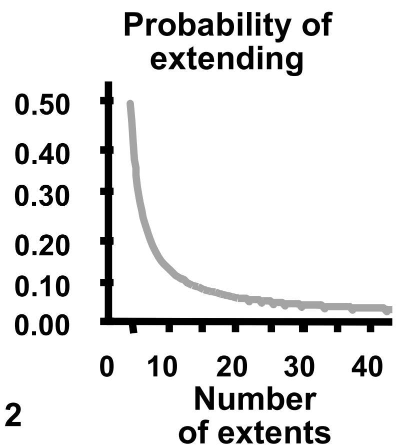
```
SQL> EXECUTE dbms_transaction.use_rollback_segment
  2          ('large_rbs');
```

Remember that any commit operation, explicit or implicit, ends the transaction. This means that the command may have to be included repeatedly.

Note: The `SET TRANSACTION USE` rollback segment command must be the first one in the transaction.

Guidelines: Sizing Manual Rollback Segments





Rollback segment 1 = Rollback segment 2

INITIAL = NEXT = 2^n Mb

MINEXTENTS = 20

OPTIMAL = 20 * INITIAL

Guidelines: Sizing Manual Rollback Segments

Storage Parameters

Setting the right size for the rollback segments is significant for performance. The aim is to reduce dynamic extension and increase the chances that undo blocks are in the buffer cache when needed.

- Choose a value for the `INITIAL` storage parameter from the list 8 KB, 16 KB, 32 KB, and 64 KB for small transactions and 128 KB, 256 KB, 512 KB, 1 MB, 2 MB, 4 MB, and so on for larger transactions.
- Use the same value for `NEXT` as for `INITIAL`. Because `PCTINCREASE` is 0, all the other extents will have the same size as the `NEXT`.
- Make all your rollback segments the same size. Take the large rollback segments offline if they are not needed.
- Set `MINEXTENTS` to 20. This makes it unlikely that the rollback segment would need to grab another extent, because the extent that it should move into is still being used by an active transaction.

Tablespace Size

Leave enough free space in the rollback segments tablespace for a larger-than-usual transaction to be able to extend the rollback segment it is using. The `OPTIMAL` setting will later cause the extended rollback segment to shrink.

Sizing Transaction Rollback Data

- Deletes are expensive for rollback activity.
- Inserts use minimal rollback space.
- Updates use rollback space, depending on the amount of data changed in the transaction.
- Index maintenance adds rollback.

```
SQL> SELECT s.username, t.used_ublk,
  2    t.start_time
  3  FROM v$transaction t, v$session s
  4  WHERE t.addr = s.taddr;
```


Transaction Statements

The number of bytes required to store information that is needed in case of rollback depends on the type of transaction being performed:

- Deletes are expensive for rollback segments; they need to store the actual row itself. If you can use TRUNCATE instead, performance is improved.
- Inserts use little rollback space; only the rowid is kept.
- The amount used for updates depends on how many columns are being updated.
- Indexed values generate more rollback, because the server process must change values in the index as well as in the table. For updates on indexed columns, the Oracle server records in the rollback segment the old data value, the old index value, and the new index value. Updating rows that change partitions will also generate more rollback.
- Direct path inserts / appends / loads are not likely to use much rollback.

Note: Columns of the LOB data type do not use rollback segment space for changes. They use their own segment space defined by the PCTVERSION clause setting.

Estimate the size of the rollback segment by running the longest expected transaction and checking the size of the rollback segment. For the current transaction, get the number of blocks used in a rollback segment using the above statement.

Sizing Transaction Rollback Data

- **The number of bytes in rollback segments before execution of statements:**

```
SQL> SELECT usn,writes
  2  FROM v$rollstat;
```

- **After execution of statements:**

```
SQL> SELECT usn,writes
  2  FROM v$rollstat;
```


Sizing Transaction Rollback Data Volume

Another way to estimate the volume of rollback data for a test transaction is to perform the following steps:

1. Before you execute the test transaction, display the current number of writes in the rollback segments:

```
SQL> SELECT usn, writes
  2  FROM v$rollstat;

 USN    WRITES
----    -------
   0       1962
   1    1102686
   2      32538
   3    1226096
```

2. Execute the test transaction:

```
SQL> UPDATE employees SET salary=1000;
6560 rows updated.
```

Sizing Transaction Rollback Data Volume (continued)

3. Display the new number of writes in the rollback segments:

```
SQL> SELECT usn, writes
  2  FROM v$rollstat;

 USN    WRITES
----   -------
   0      1962
   1   2232270
   2     32538
   3   1226096
```

Calculate the difference between the new and the old number of writes in the USN1 rollback segment to determine the amount of rollback data used for this test transaction. For this to be accurate you must ensure that:

- The transaction used the rollback segment USN1.
- No other transaction used the rollback segment USN1.

Possible Problems Caused by Small Rollback Segments

- The transaction fails for lack of rollback space.
- A "snapshot too old" error occurs if the statement requires data that has been modified, committed, and the rollback data is no longer available.

Possible Problems Caused by Small Rollback Segments

Large Transactions

If a transaction is exceptionally large, it may fail because the rollback segment cannot expand:

- The rollback segment has reached its maximum number of extents.
- There is no room left in the tablespace for the rollback segment to expand.

You need bigger rollback segments or more space in the tablespace.

Snapshot Too Old

If a query fails with the following error message, the rollback image needed for read consistency has probably been overwritten by an active transaction:

```
ORA-01555: snapshot too old (rollback segment too small)
```

Occasionally, you may get this error even if there are no other transactions active.

To resolve this error, you need bigger rollback segments. You should also avoid running batch type queries during the daytime. If not avoidable, then the long running (queries/load operations) should use a `set transaction use rollback segment` statement at their beginning.

Summary

In this lesson, you should have learned how to:

- Describe the concept of automatic undo management
- Create and maintain the automatic managed undo tablespace
- Set the retention period
- Use dynamic performance views to check rollback segment performance
- Reconfigure and monitor rollback segments
- Define the number and sizes of rollback segments
- Allocate rollback segments to transactions

ORACLE