

Oracle Database 12c: Performance Management and Tuning

Activity Guide

D79236GC10
Edition 1.0
May 2014
D86569

ORACLE®

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Disclaimer

This document contains proprietary information and is protected by copyright and other intellectual property laws. You may copy and print this document solely for your own use in an Oracle training course. The document may not be modified or altered in any way. Except where your use constitutes "fair use" under copyright law, you may not use, share, download, upload, copy, print, display, perform, reproduce, publish, license, post, transmit, or distribute this document in whole or in part without the express authorization of Oracle.

The information contained in this document is subject to change without notice. If you find any problems in the document, please report them in writing to: Oracle University, 500 Oracle Parkway, Redwood Shores, California 94065 USA. This document is not warranted to be error-free.

Restricted Rights Notice

If this documentation is delivered to the United States Government or anyone using the documentation on behalf of the United States Government, the following notice is applicable:

U.S. GOVERNMENT RIGHTS

The U.S. Government's rights to use, modify, reproduce, release, perform, display, or disclose these training materials are restricted by the terms of the applicable Oracle license agreement and/or the applicable U.S. Government contract.

Trademark Notice

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Authors

Donna Keesling, James Spiller

Technical Contributors and Reviewers

Harald van Breederode, Yio Liong Liew, Sailaja Pasupuleti, Naoki Kato, Joel Goodman, Joe Fong, Ira Singer, Herbert Bradbury, Gerlinde Frenzen, Christopher D. Andrews, Branislav Valny, Anthony Woodell, Andy Fortunak

This book was published using: **Oracle Tutor**

Table of Contents

Practices for Lesson 1: Introduction	1-1
Practices for Lesson 2: Basic Tuning Diagnostics	2-1
Practices for Lesson 2: Overview.....	2-2
Practice 2-1: Viewing Diagnostic Information.....	2-3
Practice 2-2: Using the Alert Log.....	2-13
Practice 2-3: Viewing System Statistics and Wait Events	2-17
Practices for Lesson 3: Using Automatic Workload Repository	3-1
Practices for Lesson 3: Overview.....	3-2
Practice 3-1: Using Automatic Workload Repository	3-3
Practices for Lesson 4: Defining Problems	4-1
Practices for Lesson 4: Overview.....	4-2
Practice 4-1: Using Enterprise Manager to Identify OS Issues	4-3
Practices for Lesson 5: Using Metrics and Alerts	5-1
Practices for Lesson 5: Overview.....	5-2
Practice 5-1: Using Metrics	5-3
Practice 5-2: Disable Database Vault Metric Collection	5-14
Practices for Lesson 6: Using Baselines	6-1
Practices for Lesson 6: Overview.....	6-2
Practice 6-1: Using Baselines	6-3
Practices for Lesson 7: Using AWR-Based Tools	7-1
Practices for Lesson 7: Overview.....	7-2
Practice 7-1: Using AWR-Based Tools	7-3
Practices for Lesson 8: Real-Time Database Operation Monitoring	8-1
Practices for Lesson 8: Overview.....	8-2
Practice 8-1: Monitoring a Composite Database Operation	8-3
Practice 8-2: Monitoring a PL/SQL Operation	8-8
Practices for Lesson 9: Monitoring Applications	9-1
Practices for Lesson 9: Overview.....	9-2
Practice 9-1: Using Services in a Single-Instance Oracle Database	9-3
Practice 9-2: Tracing Services in a Single-Instance Oracle Environment	9-17
Practices for Lesson 10: Identifying Problem SQL Statements	10-1
Practices for Lesson 10: Overview.....	10-2
Practice 10-1: Using AUTOTRACE and DBMS_XPLAN.....	10-3
Practice 10-2: Using the SQL TRACE Facility	10-7
Practices for Lesson 11: Influencing the Optimizer	11-1
Practices for Lesson 11: Overview.....	11-2
Practice 11-1: Capturing Extended Statistics	11-3
Practice 11-2: Influencing Optimizer Performance by Statistics Changes.....	11-8
Practices for Lesson 12: Reducing the Cost of SQL Operations.....	12-1
Practices for Lesson 12: Overview.....	12-2
Practice 12-1: Excess Blocks	12-3
Practice 12-2: Coalescing an Index.....	12-9
Practices for Lesson 13: Using SQL Performance Analyzer.....	13-1
Practices for Lesson 13: Overview.....	13-2

Practice 13-1: Using SQL Performance Analyzer	13-3
Practices for Lesson 14: SQL Performance Management.....	14-1
Practices for Lesson 14: Overview.....	14-2
Practice 14-1: Seeding SQL Plan Baselines from SQL Performance Analyzer	14-3
Practice 14-2: SQL Plan Management (SPM).....	14-8
Practices for Lesson 15: Using Database Replay	15-1
Practices for Lesson 15: Overview.....	15-2
Practice 15-1: Using Database Replay	15-3
Practices for Lesson 16: Tuning the Shared Pool.....	16-1
Practices for Lesson 16: Overview.....	16-2
Practice 16-1: Sizing the Shared Pool.....	16-3
Practice 16-2: Tuning a Hard-Parse Workload.....	16-17
Practice 16-3: Keeping Objects in the Shared Pool	16-27
Practices for Lesson 17: Tuning the Buffer Cache	17-1
Practices for Lesson 17: Overview.....	17-2
Practice 17-1: Using the DB Cache Advisor.....	17-3
Practice 17-2: Using the Keep Pool	17-11
Practices for Lesson 18: Tuning PGA and Temporary Space.....	18-1
Practices for Lesson 18: Overview.....	18-2
Practice 18-1: Tuning PGA_AGGREGATE_TARGET	18-3
Practices for Lesson 19: Automatic Memory Management.....	19-1
Practices for Lesson 19: Overview.....	19-2
Practice 19-1: Enabling Automatic Shared Memory Management.....	19-3
Practice 19-2: Adjusting Memory as Workloads Change	19-14
Practices for Lesson 20: Performance Tuning Summary.....	20-1
Practices for Appendix A: Using Statspack.....	21-1
Practice for Appendix A-1: Installing Statspack.....	21-2
Practice for Appendix A-2: Creating Snapshots	21-5
Practice for Appendix A-3: Generating Statspack Reports.....	21-8

Practices for Lesson 1: Introduction

Chapter 1

Practices for Lesson 1

Practices Overview

There are no practices for this lesson.

Practices for Lesson 2: Basic Tuning Diagnostics

Chapter 2

Practices for Lesson 2: Overview

Practices Overview

In this practice, you look at various diagnostics to view the types of information available in the database.

Practice 2-1: Viewing Diagnostic Information

In this practice, you use the alert log to gather information about the performance of the instance while running a workload. You review the alert log for information that is relevant to performance tuning.

- Set up the `orcl` database for a workload. Execute the `prepare` script:

```
./prepare 2 1
```

```
$ . oraenv
ORACLE_SID = [oracle] ? orcl
The Oracle base for
ORACLE_HOME=/u01/app/oracle/product/12.1.0/dbhome_1 is
/u01/app/oracle
$ cd /home/oracle/workshops
$ ./prepare 2 1
The Oracle base for
ORACLE_HOME=/u01/app/oracle/product/12.1.0/dbhome_1 is
/u01/app/oracle

Database closed.
Database dismounted.
ORACLE instance shut down.
ORACLE instance started.

Total System Global Area  501059584 bytes
Fixed Size                  2290024 bytes
Variable Size                457182872 bytes
Database Buffers            33554432 bytes
Redo Buffers                 8032256 bytes
Database mounted.
Database opened.
$
```

- Run a workload by using the workload generator: `./workgen 2 1`.

Wait a few minutes for the operating system prompt to appear, and then check whether the workload has started by using the `ps` command in the same terminal window. You see a set of processes as illustrated in the following output. The order and number of each vary. In most cases, you see several `.sql` scripts, and `sqlplus` and `sleep` processes. All of these processes may not be active at the time that the `ps` command is issued. Continue to the next step in the practice.

```
$ ./workgen 2 1

The Oracle base for
ORACLE_HOME=/u01/app/oracle/product/12.1.0/dbhome_1 is
/u01/app/oracle

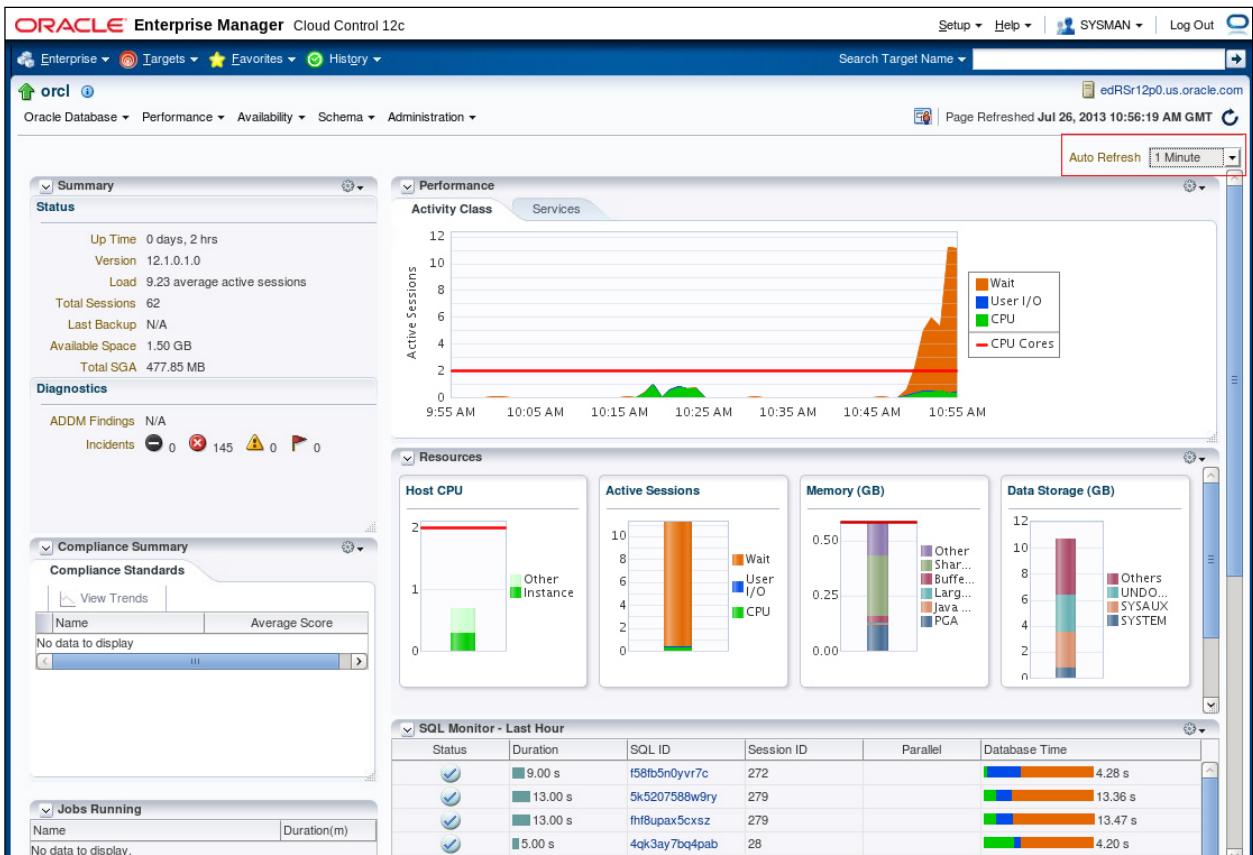
PL/SQL procedure successfully completed.

PL/SQL procedure successfully completed.
```

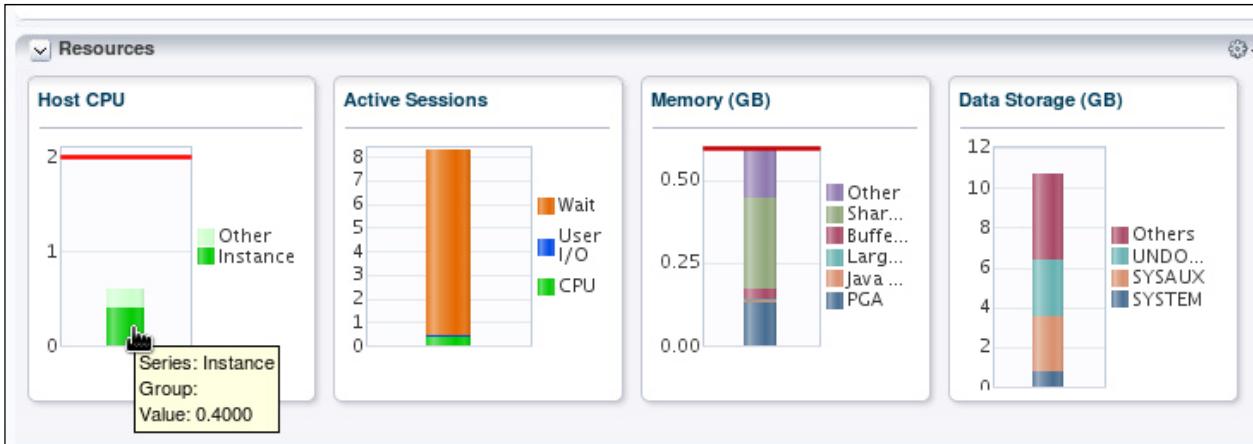
```
$ ps
  PID TTY          TIME CMD
 2932 pts/0        00:00:00 insert_orders.s
 2950 pts/0        00:00:00 update_orders.s
 2976 pts/0        00:00:00 delete_orders.s
 2983 pts/0        00:00:00 bash
 3002 pts/0        00:00:00 insert_orders.s
 3027 pts/0        00:00:00 update_orders.s
 3079 pts/0        00:00:00 delete_orders.s
 3113 pts/0        00:00:00 insert_orders.s
...
5389 pts/0        00:00:00 sleep
 5403 pts/0        00:00:00 sleep
 5404 pts/0        00:00:00 sleep
 5405 pts/0        00:00:00 sleep
 5406 pts/0        00:00:00 sleep
 5407 pts/0        00:00:00 ps
$
```

3. You will use Enterprise Manager Cloud Control to view diagnostic information about the `orcl` database throughout this course. First, you need to configure the `orcl` database as a target in Enterprise Manager Cloud Control.
 - a. Launch a browser and enter the following URL for Enterprise Manager Cloud Control:
`https://<hostname>:7802/em`
Note: `<hostname>` in the URL is the name of your PC. You can find that name with the `hostname` operating system command.
 - b. On the first connection to this URL, you will see a page “This Connection is Untrusted.”
 - c. Expand “I Understand the Risks”. Click **Add Exception**.
 - d. In the Add Security Exception window, verify that **Permanently store this exception** is selected. Click **Confirm Security Exception**.
 - e. Enter `sysman` in the User Name field and `oracle_4U` in the Password field. Click **Login**.
 - f. Click **I Accept** to accept the license agreement.
 - g. Click **Select as My Home** to set the **Summary** page as the home page.
 - h. Expand the **Targets** menu and select **Databases**.
 - i. Select **Search List**.
 - j. If no targets appear, click **Add** to add the `orcl` database target.
 - k. Use the Search icon to select the host. After selecting the host, click **Continue**.
 - l. Click the configure icon for the `orcl` database.
 - m. Enter `oracle_4U` in the Monitor Password field. Click **Next**.
 - n. Click **OK**.
 - o. On the Discovered Targets page, deselect the `em12rep` database. Click **Finish**.
 - p. On the Summary page, click **Save**.
 - q. On the Target Configuration Results page, click **OK**.
4. View `orcl` database diagnostic information on the database home page in Enterprise Manager Cloud Control.
 - a. Expand the **Targets** menu and select **Databases**.

- Click the link for the `orcl` database.
- On the `orcl` database home page, set Auto Refresh to 1 minute. On the database home page, what diagnostic information is available?

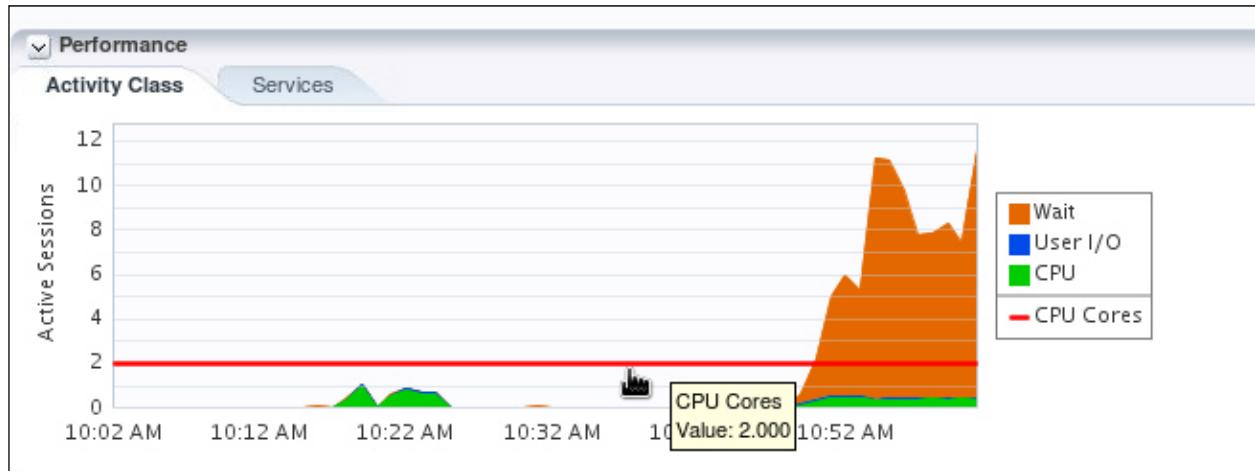


- Use Enterprise Manager Cloud Control to observe the characteristics of the workload on the server. This is shown in the Host CPU graph.
- What is the load shown in the Host CPU graph? *This number varies at each refresh.*
- What percentage of the CPU are the `orcl` processes using? *Place the cursor over the Instance part of the bar.*



- The characteristics of the workload on the database instance are shown in the Active Sessions graph. When there are a large number of waiting sessions relative to the number of sessions that use the CPU, you must check other diagnostics.

- a. What is the average number of sessions that use the CPU?
- b. What is the average number of sessions that are waiting? Number waiting on I/O?
7. How many CPU cores are in your system? *Place the cursor over the CPU Cores line in the Activity Class graph to see the number of CPU cores.*



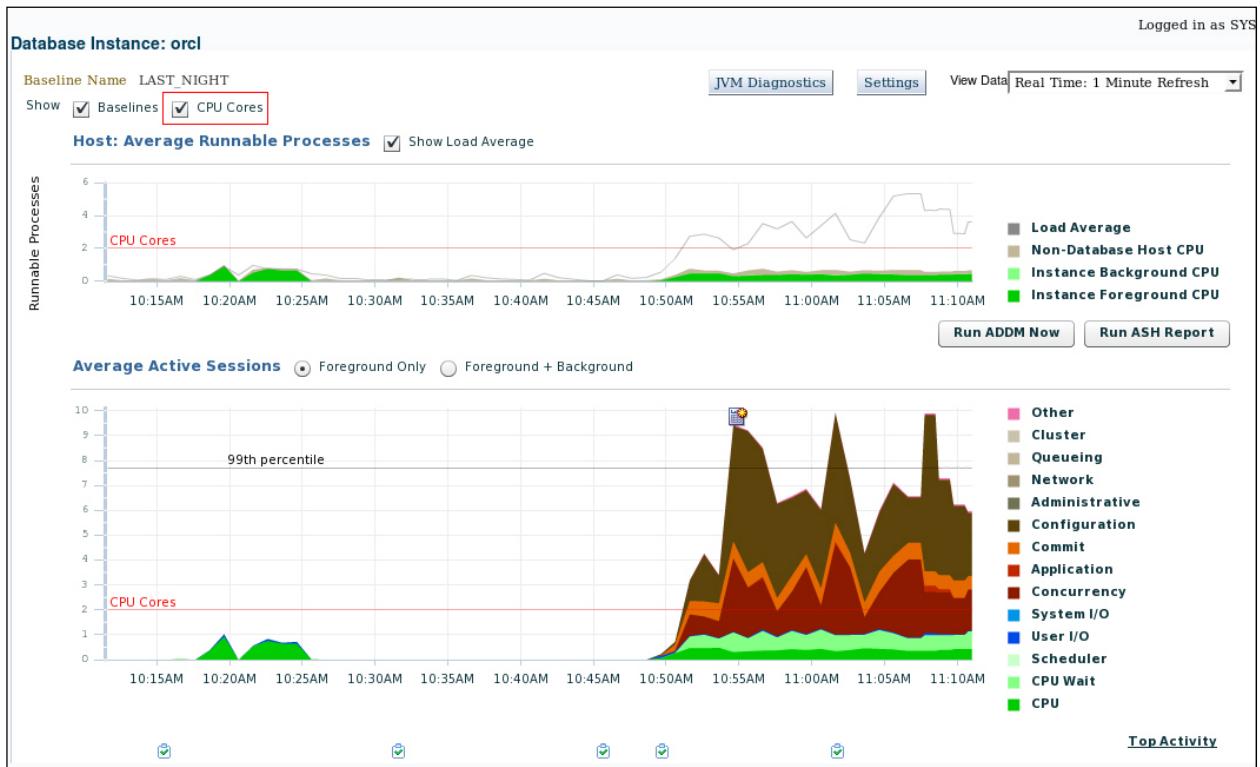
8. Use the Enterprise Manager Cloud Control Performance Home page to view additional information.
 - a. Navigate to the Performance Home page by expanding the Performance menu and selecting **Performance Home**.
 - b. Log in to the `orcl` database. Enter `sys` in the Username field and `oracle_4U` in the Password field. Select `SYSDBA` from the Role drop-down list. Select **Save As** and enter `SYS_SYSDBA` in the field. Select **Set As Preferred Credentials** and select `SYSDBA Database Credentials` from the drop-down list. Click **Login**.

The figure is a screenshot of the 'Database Login' dialog box. It contains the following fields:

- * Username: sys
- * Password: *****
- Role: SYSDBA
- Save As: SYS_SYSDBA
- Set As Preferred Credentials: **SYSDBA Database Credentials**
- Buttons: Login, Cancel

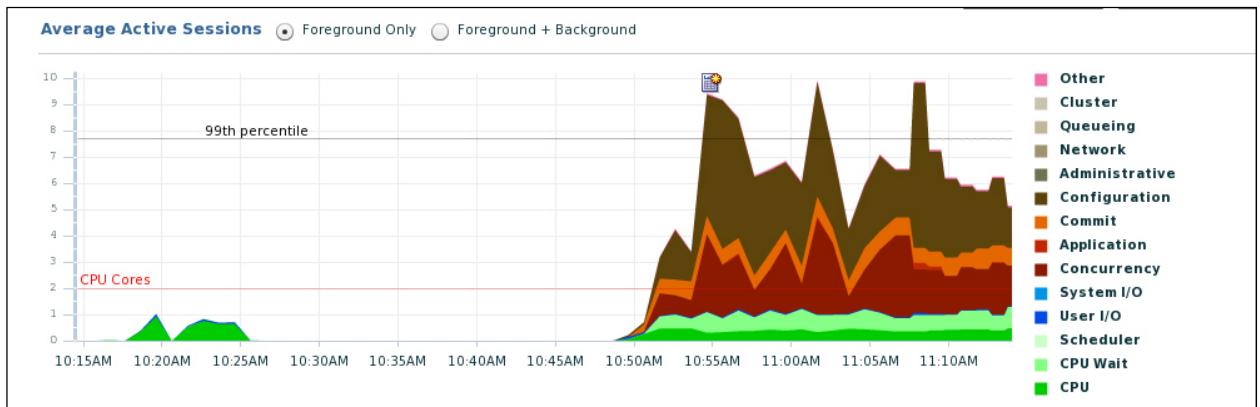
- c. On the Performance Home page, select **Real Time: 1 Minute Refresh** in the **View Data** menu.

- d. On the Performance Home page, load average (Average Runnable Processes) and average active sessions are graphed. The number of CPU cores can be shown in the graph by selecting the CPU Cores check box.



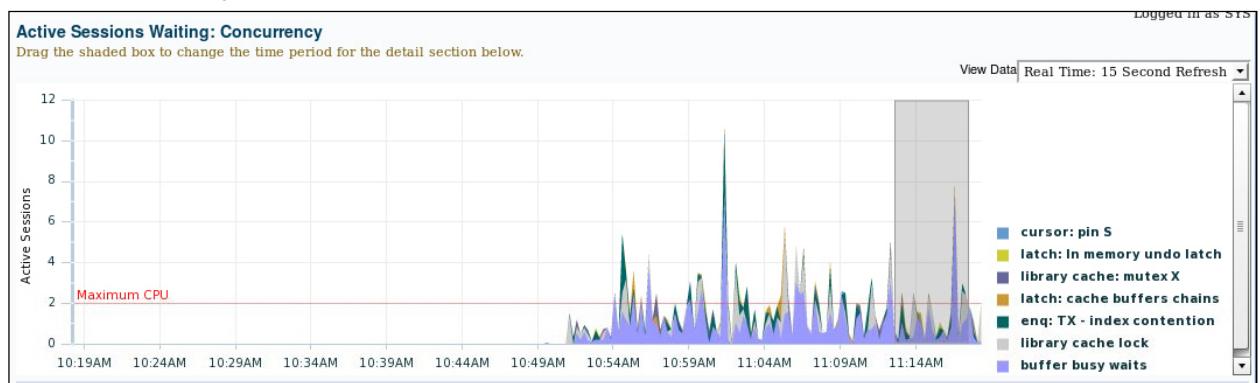
- e. What is the value of the load average in relation to the maximum CPU? *The load average is greater than the maximum CPU. When the load average is greater than the maximum CPU, processes need to wait on the run queue.*
9. In the **Average Active Sessions** graph, the colors of the graph show various components of the waits and service times that the sessions are experiencing. Each component shows the number of active sessions in that state. For example, the component at the bottom of the graph shows the number of active sessions that use the CPU.

Which component of the graph, other than the CPU, consistently has the highest number of active sessions? *This can vary. CPU Wait, Concurrency, Configuration, User I/O, and Commit are valid answers.*

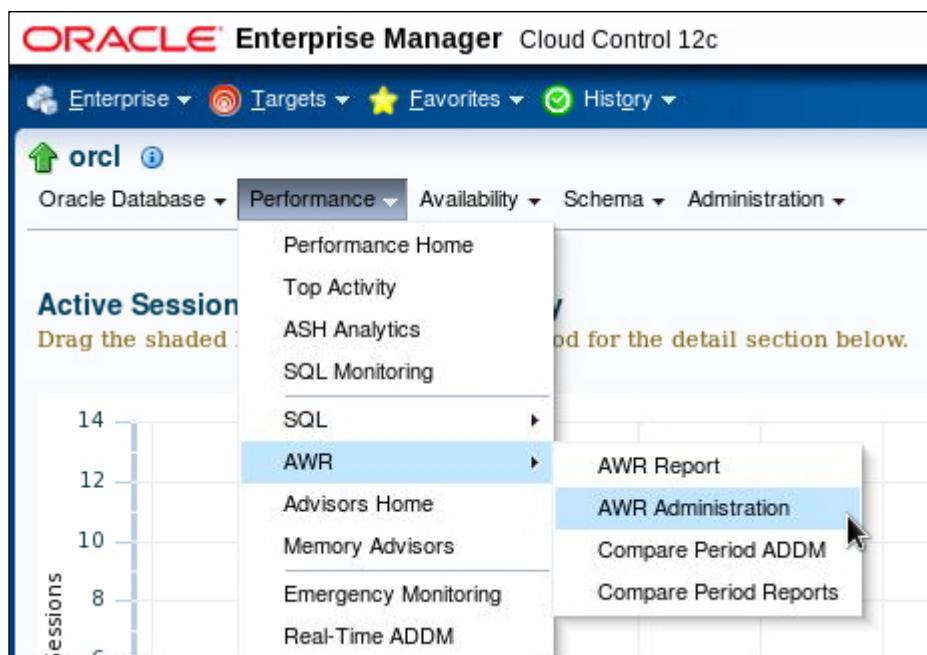


10. Click **Concurrency** in the **Average Active Sessions graph** legend to view the waits that are included in this category. From a visual scan of the Active Sessions Waiting:

Concurrency page, what are the highest waits? *buffer busy waits, enq: TX - index contention, library cache: mutex X, and latch: cache buffer chains* are all valid answers.



11. Create an AWR snapshot.
 - a. Expand the **Performance** menu. Expand the **AWR** menu and select **AWR Administration**.



- b. On the Automatic Workload Repository page, click the number beside Snapshots in the Manage Snapshots and Baselines section.

Automatic Workload Repository

The Automatic Workload Repository is used for storing database statistics that are used for performance tuning.

General

Snapshot Retention (days) 12
Snapshot Interval (minutes) 15
Collection Level TYPICAL
Next Snapshot Capture Time Jul 26, 2013 11:31:15 AM

Edit

Manage Snapshots and Baselines

Snapshots 1225
Baselines 4

Run AWR Report Run Compare Periods Report

Latest Snapshot Time Jul 26, 2013 11:16:15 AM
Earliest Snapshot Time Jul 3, 2013 12:00:12 AM

- c. On the Snapshots page, note the number of the last snapshot number shown. *This number may be over 100.*
- d. Click **Create** to create a new snapshot.

Automatic Workload Repository > Snapshots

Snapshots

A snapshot is a collection of database statistics at a single point in time. You can use the information in snapshots to diagnose database problems.

Page Refreshed Jul 26, 2013

Select Beginning Snapshot

Go To Time 7/26/13 11:00 AM Go
(Example: 12/15/03)

Create

Actions	Create SQL Tuning Set	Go	Previous 25	1201-1225 of 1225	Next
Delete					
Select	ID	Capture Time ▲	Collection Level	Within A Baseline	
<input type="radio"/> 1903	1903	Jul 26, 2013 5:15:46 AM	TYPICAL		
<input type="radio"/> 1904	1904	Jul 26, 2013 5:30:48 AM	TYPICAL		
<input type="radio"/> 1905	1905	Jul 26, 2013 5:45:49 AM	TYPICAL		
<input type="radio"/> 1906	1906	Jul 26, 2013 6:00:51 AM	TYPICAL		

- e. On the Confirmation page, click **Yes** to create a manual snapshot.
12. Create an AWR report.
- a. On the Snapshots page, select the snapshot that you recorded in step 11c.

- b. Select **View Report** from the Actions drop-down list, and then click **Go** next to it.

Automatic Workload Repository > Snapshots

Snapshots

A snapshot is a collection of database statistics at a single point in time. You can use the information in snapshots to diagnose database problems.

Page Refreshed Jul 26, 2013

Select Beginning Snapshot

Go To Time: 7/26/13 11:00 AM Go
(Example: 12/15/03)

Select	ID	Capture Time	Collection Level	Within A Baseline
<input type="radio"/>	1928	Jul 26, 2013 11:23:00 AM	TYPICAL	
<input checked="" type="radio"/>	1927	Jul 26, 2013 11:15:56 AM	TYPICAL	
<input type="radio"/>	1926	Jul 26, 2013 11:00:48 AM	TYPICAL	

Previous 1-25 of 1226 Next 25 Create

- c. Select the next snapshot after the one chosen in step 11c. Click **OK**.

13. View the Top Timed Foreground Wait Events.

- a. When the report appears, scroll down to “**Top 10 Foreground Events by Total Wait Time**.”

Top 10 Foreground Events by Total Wait Time

Event	Waits	Total Wait Time (sec)	Wait Avg(ms)	% DB time	Wait Class
buffer busy waits	45,078	1907.9	42	41.2	Concurrency
write complete waits	766	995	1299	21.5	Configuration
free buffer waits	4,947	597.9	121	12.9	Configuration
DB CPU		281.3		6.1	
log file sync	2,058	208	101	4.5	Commit
enq: TX - index contention	3,206	145.5	45	3.1	Concurrency
library cache lock	466	137.9	296	3.0	Concurrency
log file switch (checkpoint incomplete)	36	36.4	1012	.8	Configuration
latch: cache buffers chains	12,965	30.2	2	.7	Concurrency
db file sequential read	464,369	23.2	0	.5	User I/O

- b. What is the top timed event?
c. What is the second timed event?
d. Are any of the other timed events found in step 9 listed here? If so, list the timed events that are found.

14. View the Time Model Statistics

- a. Scroll down to the **Wait Events Statistics** section.

- b. Click the **Time Model Statistics** link.

Wait Events Statistics

- [Time Model Statistics](#)
- [Operating System Statistics](#)
- [Operating System Statistics - Detail](#)
- [Foreground Wait Class](#)
- [Foreground Wait Events](#)
- [Background Wait Events](#)
- [Wait Event Histogram](#)
- [Wait Event Histogram Detail \(64 msec to 2 sec\)](#)
- [Wait Event Histogram Detail \(4 sec to 2 min\)](#)
- [Wait Event Histogram Detail \(4 min to 1 hr\)](#)
- [Service Statistics](#)
- [Service Wait Class Stats](#)

[Back to Top](#)

- c. Which category collects the most %DB Time? *sql execute elapsed time is expected, but this item might be second in the list.*

Time Model Statistics

- Total time in database user-calls (DB Time): 4632s
- Statistics including the word "background" measure background process time,
- Ordered by % or DB time desc, Statistic name

Statistic Name	Time (s)	% of DB Time
sql execute elapsed time	4,263.12	92.04
DB CPU	281.31	6.07
connection management call elapsed time	261.06	5.64
PL/SQL execution elapsed time	41.98	0.91
parse time elapsed	8.59	0.19
hard parse elapsed time	5.31	0.11
sequence load elapsed time	0.55	0.01
PL/SQL compilation elapsed time	0.08	0.00
hard parse (sharing criteria) elapsed time	0.01	0.00
repeated bind elapsed time	0.00	0.00
DB time	4,631.95	
background elapsed time	596.62	
background cpu time	6.83	

[Back to Wait Events Statistics](#)

[Back to Top](#)

- d. Can you guess what performance issue is slowing this instance? *Poor SQL execution plans that produce index contention may be contributing to the sql elapsed time. In addition, log file waits may be holding up commit processing that is also contributing to SQL elapsed time.*

- e. Stop the workload. In the terminal window, issue the following command:

```
$ rm $HOME/workshops/runload
```

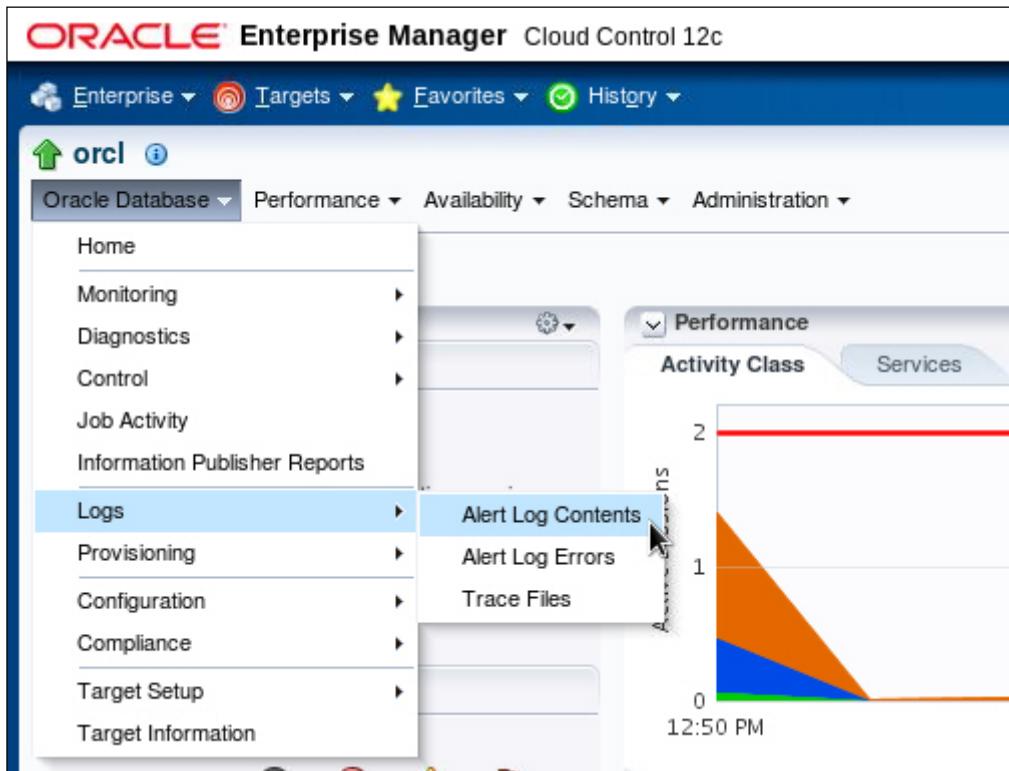
```
$ rm $HOME/workshops/runload  
$
```

Note: Most workload scripts in this course run until stopped. They repeat as long as the \$HOME/workshops/runload file exists.

Practice 2-2: Using the Alert Log

In this practice, you use the alert log to gather information about instance configuration and performance.

1. View the alert log through Enterprise Manager Cloud Control. Find the last 500 entries.
 - a. On the **Databases** page, click the link for the **orcl** database.
 - b. On the **orcl** database home page, expand the **Oracle Database** menu. Expand the **Logs** menu and select **Alert Log Contents**.



- c. From View Entries, select **Last 500** and click **Go**.

View Alert Log Contents					Page Refreshed
TIP You are currently viewing XML Alert Log Contents. Switch to Text Alert Log Contents					
Log File Contents					
View Entries	Last 500	Go	Search		
Timestamp	Type	Level	Incident ID	Group	
(No data retrieved)					

- d. Scroll through the entries. Note that the entries are in reverse chronological order. The most recent entries are at the top of the page.
3. Find the current Redo Log number and how many redo logs have been generated since yesterday. Each time the log file is full, a log switch occurs and a message is sent to the alert log giving the current log number.

- a. Click **Search**.

View Alert Log Contents

Log Location /u01/app/oracle/diag/rdbms/orcl/orcl/alert
Modified Jul 26, 2013 11:32:48 AM GMT+00:00
Size (MB) 2.63

TIP You are currently viewing XML Alert Log Contents. [Switch to Text Alert Log Contents](#)

Log File Contents

View Entries Last 500 Go **Search**

Timestamp ▾	Type	Level	Incident ID	Group	Message ID
Jul 26, 2013 11:32:48 AM UTC	UNKNOWN	16			
Jul 26, 2013 11:32:48 AM UTC	UNKNOWN	16			
Jul 26, 2013 11:32:44 AM UTC	UNKNOWN	16			
Jul 26, 2013 11:32:44 AM UTC	UNKNOWN	16			

- b. On the Search Alert Log Contents page, select **Most Recent**, enter **25** in the field, and select **Hours**. In the Message Text field, enter **.*Current log.*** and select the **Regular Expression** check box. Then click **Go**.

View Alert Log Contents > Search Alert Log Contents

Page Refreshed **Jul 26, 2013**

Search Criteria

Date Range

Most Recent 25 Hours

Time Interval

Start Date (Example: Jul 26, 2013)

End Date (Example: Jul 28, 2013)

Start Time 01 00 AM PM

End Time 01 00 AM PM

Message Types

Incident Error Warning Trace
 Error Notification Unknown

Message Text Regular Expression

Maximum Entries Retrieved
Entries Per Page

What is the current sequence number (seq#)?

How many log switches have taken place?

- c. It is recommended that log switches take place no more than every 20 minutes. Is the number of log switches excessive? Yes. *To reduce the number of log switches, increase the size of the redo log files.*

4. Find the name of the current SPFILE and the names of any other SPFILEs that have been used in the last 10 days. Use the Search feature of the View Alert Log Contents page.
- Use the Search feature again. Set the date range to **Most Recent 10 Days**, and then change the message text to **.*spfile.***. Ensure that the **Regular Expression** check box is selected. Click **Go**.

Search Criteria

Date Range

Most Recent 10 Days

Time Interval

Start Date (Example: Jul 26, 2013) End Date (Example: Jul 28, 2013)

Start Time 01 00 AM PM End Time 01 00 AM PM

Go

Message Types

Incident Error Error Warning Notification Trace Unknown

Message Text **.*spfile.*** Regular Expression

Maximum Entries Retrieved 500 Entries Per Page 25

- What is the SPFILE name?
/u01/app/oracle/product/12.1.0/dbhome_1/dbs/spfileorcl.ora
 - Why is the SPFILE that is used to start the database important? *Changes in the initialization parameters can have a large effect on database performance.*
 - How else can initialization parameters change? *By the use of the ALTER SYSTEM command*
 - Have any initialization parameters been changed by using the ALTER SYSTEM command? **Hint:** Use **.*ALTER SYSTEM.*** in the Message Text field. *The prepare script issues ALTER SYSTEM commands. The course setup scripts also modify the SPFILE with ALTER SYSTEM commands.*
5. Using SQL*Plus, find and view the alert log file for the `orcl` database. Use the operating system login, and locate and view the log for the `orcl` database. The text version of the alert log in Oracle Database 12c is located in the directory specified by the Diag Trace entry in the `V$DIAG_INFO` view.
- Return to the terminal window. Invoke SQL*Plus and connect as `sysdba`. Execute the query shown in the code box to find the Diag Trace directory. Exit SQL*Plus.

```
$ sqlplus / as sysdba

/* version and option banner omitted */

SQL> SELECT value FROM v$diag_info
  2 WHERE NAME = 'Diag Trace';

VALUE
-----
/u01/app/oracle/diag/rdbms/orcl/orcl/trace

SQL> exit
```

- b. In the terminal window, change directories to the directory found in the previous step. Then view the `alert_orcl.log` file. (gedit, a WYSIWYG editor, and text file viewers—less and more—are available.)
 - c. Find the **time of the last startup**. Start at the end of the file. (Search from the most recent entries to the “Starting ORACLE instance” string.) The time stamp for this event is the time stamp immediately above the entry.
6. Determine how often the log files have switched in the last 10 minutes, while the workload is running.
- The log file may have switched several times. The tail command lists the last 10 lines of a file. The -n 100 option causes the tail command to display 100 lines; you may need to use tail with more lines to see a full 10 minutes.*
7. While viewing the alert log, find the settings for the `db_cache_size`, `log_buffer`, `sga_max_size`, and `compatible` parameters.
- Hint:** All nondefault parameter values are listed in the alert log on every instance startup. Scroll through the alert log entries. Start from the most recent entries in the file to find the current parameter setting. Note that with this method, you see only those parameters that have been set to a nondefault value. (`log_buffer` is not visible because it has a default value.)
8. Use SQL*Plus to find the values for the `db_cache_size`, `log_buffer`, `sga_max_size`, and `compatible` parameters. Use the `SHOW PARAMETER` command.

```
$ sqlplus / as sysdba

/* version and option banner omitted */

SQL> show parameter db_cache_size

NAME                           TYPE        VALUE
-----
db_cache_size                  big integer 32M

SQL> show parameter log_buffer

NAME                           TYPE        VALUE
-----
log_buffer                     integer    7684096

SQL> show parameter sga_max_size

NAME                           TYPE        VALUE
-----
sga_max_size                   big integer 480M

SQL> show parameter compatible

NAME                           TYPE        VALUE
-----
Compatible                     string     12.1.0.0.0
noncldb_compatible             boolean   FALSE

SQL> exit
```

Practice 2-3: Viewing System Statistics and Wait Events

In this practice, you use dynamic performance views to view tuning information.

- Start the workload generator and continue. The workload continues until it is stopped by removing the `runload` file.

```
$ cd /home/oracle/workshops
$ ./workgen 2 3

The Oracle base for
ORACLE_HOME=/u01/app/oracle/product/12.1.0/dbhome_1 is
/u01/app/oracle

PL/SQL procedure successfully completed.

PL/SQL procedure successfully completed.

$
```

- Use the `V$SYSSTAT` view to view system statistics.

- Log in to SQL*Plus as the `sys` user.
- Query the `NAME`, `CLASS`, and `VALUE` columns in `V$SYSSTAT`. Order the output by `CLASS`. The statistics are cumulative from the time that the instance was started.

```
$ sqlplus / as sysdba
SQL> COLUMN name FORMAT A40 WORD_WWRAPPED
SQL> SELECT name, class, value
      2  FROM v$sysstat
      3  ORDER BY class;
```

- What can you deduce about the instance workload and performance from a listing of the system statistics?

There is very little that you can determine from a listing of the system statistics by itself. Two listings that are used to create a difference in values over a known period of time can yield rate and load information.

- View system wait events. Use the `V$SYSTEM_EVENT` view to see wait event statistics. Find the total number of waits, the time waited, and the average time waited for the `log file sync` and `eng: TX - index contention` events, and the time waited relative to the other wait events. The number of rows returned will vary; events that have zero counts will not be listed.

The log file sync and eng: TX - index contention events have collected some waits and the average wait time is displayed. There are several wait events that are not important, such as `rdbms ipc message` and `smon timer`. These are internal waits for a command to be issued, or for a timer to expire to initiate a task. They do not affect performance.

```

SQL> COLUMN event FORMAT A30 WORD_WWRAPPED
SQL> COLUMN average_wait HEADING AVERAGE_WAIT
SQL> COLUMN TIME_WAITED HEADING TIME_WAITED
SQL> COLUMN TOTAL_WAITS HEADING TOTAL_WAITS
SQL> SET PAGESIZE 60
SQL> SELECT event, total_waits, time_waited, average_wait
  2  FROM v$system_event
  3  ORDER BY time_waited DESC;

```

- Check the session wait events. Find the sessions that are currently waiting on the log file sync event. Use the V\$SESSION_WAIT view. Why are very few or no sessions listed?

The V\$SESSION_WAIT view shows waits only for currently connected sessions. The waits for all sessions, past and current, are summed in the V\$SYSTEM_EVENTS view. Your output from this query will vary from the example shown.

```

SQL> COLUMN time_waited HEADING TIME_WAITED
SQL> COLUMN AVERAGE_WAIT HEADING AVERAGE_WAIT
SQL> COLUMN seconds_in_wait HEADING SECONDS_IN_WAIT
SQL> COLUMN EVENT FORMAT A20
SQL> COLUMN total_waits HEADING TOTAL_WAITS
SQL>
SQL> SELECT sid, wait_time, seconds_in_wait, state
  2  FROM v$session_wait
  3  WHERE event = 'log file sync';

no rows selected

SQL>
SQL>
SQL> SELECT sid, event, total_waits,
  2  time_waited, average_wait
  3  FROM v$session_event
  4  WHERE event = 'log file sync';

          TOTAL           TIME           AVERAGE
          SID EVENT        WAITS        WAITED
----- -----
          24 log file sync      1          17       16.56
          262 log file sync     2          29       14.3
SQL> exit
$
```

- Stop the workload by executing the rm unload command.

```

$ rm /home/oracle/workshops/unload

$
```

- Log out of Enterprise Manager Cloud Control.

Practices for Lesson 3: Using Automatic Workload Repository

Chapter 3

Practices for Lesson 3: Overview

Practices Overview

In this practice, you use AWR snapshots and reports to identify and fix the issues that you discover in a running workload.

Practice 3-1: Using Automatic Workload Repository

In this practice, you run a workload on the `orcl` database, capture AWR snapshots, and diagnose and fix a performance issue. Note that during this practice, you explicitly capture AWR snapshots. However, by default, the system automatically captures them every hour.

1. In a terminal session with the oracle environment, set for the `orcl` database, execute the `prepare` script as shown to set up this practice. You can find this script in your `$HOME/workshops` directory.

```
$ cd $HOME/workshops  
$ ./prepare 3 1
```

```
The Oracle base for  
ORACLE_HOME=/u01/app/oracle/product/12.1.0/dbhome_1 is  
/u01/app/oracle

drop tablespace tbsspc including contents and datafiles
*
ERROR at line 1:
ORA-00959: tablespace 'TBSSPC' does not exist
```

Tablespace created.

PL/SQL procedure successfully completed.

PL/SQL procedure successfully completed.

```
drop user spc cascade
*
ERROR at line 1:
ORA-01918: user 'SPC' does not exist
```

User created.

Grant succeeded.

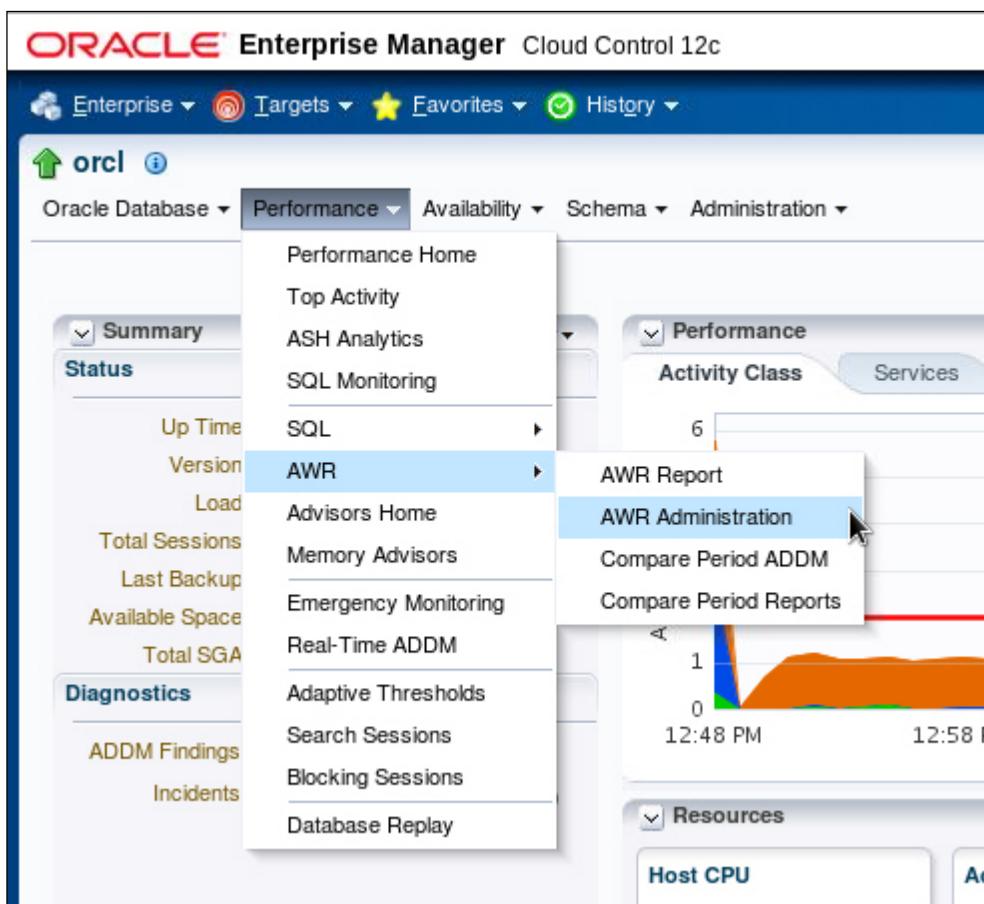
```
drop table spct purge
*
ERROR at line 1:
ORA-00942: table or view does not exist
```

Table created.

```
PL/SQL procedure successfully completed.  
  
Database closed.  
Database dismounted.  
ORACLE instance shut down.  
ORACLE instance started.  
  
Total System Global Area  501059584 bytes  
Fixed Size                  2290024  bytes  
Variable Size                465571480  bytes  
Database Buffers             25165824  bytes  
Redo Buffers                 8032256  bytes  
  
Database mounted.  
Database opened.  
$
```

2. Take your first AWR snapshot. You can use either Enterprise Manager Cloud Control or the DBMS_WORKLOAD_REPOSITORY package. The PL/SQL solution is available in the `sol_03_01_02.sh` script located in your `$HOME/solutions` directory. The script creates an AWR snapshot by using the DBMS_WORKLOAD_REPOSITORY package. The solution in this document uses Enterprise Manager Cloud Control.
 - a. Open your browser and enter the following URL: `https://<hostname>:7802/em`.
 - b. Enter `SYSMAN` in the User Name field and `oracle_4U` in the Password field. Click **Login**.
 - c. Select **Databases** in the **Targets** menu.
 - d. Click the `orcl` database link.

- e. Expand the **Performance** menu, expand the **AWR** menu, and select **AWR Administration**.



- f. Log in to the `orcl` database by selecting **SYSDBA Database Credentials** in the Preferred Credential Name menu and clicking **Login**.

- g. On the Automatic Workload Repository page, click the **number in the Snapshots field**. The number of snapshots and the snapshot ID that you see will vary from what is shown in the screenshots.

Automatic Workload Repository

Page Refreshed

The Automatic Workload Repository is used for storing database statistics that are used for performance tuning.

General

[Edit](#)

Snapshot Retention (days)	12
Snapshot Interval (minutes)	Not collecting
Collection Level	TYPICAL
Next Snapshot Capture Time	Not collecting

Manage Snapshots and Baselines

Run AWR Report	Run Compare Periods Report
Snapshots 9	
Baselines 1	
Latest Snapshot Time Jan 10, 2014 4:46:31 AM	
Earliest Snapshot Time Jan 9, 2014 12:00:22 PM	

- h. On the Snapshots page, click **Create**.

Snapshots

A snapshot is a collection of database statistics at a single point in time. You can use the information in snapshots to diagnose database problems.

Page Refreshed **Jan 10, 2014**

Select Beginning Snapshot

Go To Time 8:00 AM

(Example: 12/15/03)

[Create](#)

Delete	Actions	Create SQL Tuning Set	<input type="button" value="Go"/>	
Select	ID	Capture Time ▲	Collection Level	Within A Baseline
<input type="radio"/>	726	Jan 9, 2014 12:00:22 PM	TYPICAL	
<input type="radio"/>	727	Jan 9, 2014 1:00:26 PM	TYPICAL	
<input type="radio"/>	728	Jan 9, 2014 2:00:32 PM	TYPICAL	
<input type="radio"/>	729	Jan 9, 2014 2:27:39 PM	TYPICAL	
<input type="radio"/>	730	Jan 9, 2014 2:32:15 PM	TYPICAL	
<input type="radio"/>	731	Jan 9, 2014 4:00:38 PM	TYPICAL	
<input checked="" type="radio"/>	732	Jan 9, 2014 5:00:43 PM	TYPICAL	

- i. On the Confirmation page, click **Yes**.
- j. The processing page appears. After the snapshot is created, a confirmation message appears on the Snapshots page indicating that a new snapshot has been created.
Record the snapshot number. _____

3. Execute the `workgen` script with parameters as shown to generate the workload for this practice. The script is located in your `$HOME/workshops` directory. Wait until the “Load is finished” message appears, and then continue. This script takes about two minutes. Proceed to step 4 immediately after the script finishes.

```
$ ./workgen 3 1

The Oracle base for
ORACLE_HOME=/u01/app/oracle/product/12.1.0/dbhome_1 is
/u01/app/oracle

start 12 load_31 processes

PL/SQL procedure successfully completed.
...
PL/SQL procedure successfully completed.

Load is finished
$
```

4. Take a second AWR snapshot right after the workload is completed.
 - a. Navigate to the Snapshots page, and then click **Create**.
 - b. On the Confirmation page, click **Yes**.
 - c. The processing page appears. Wait until the snapshot is created.
 - d. The Confirmation message appears on the Snapshots page, indicating that a new snapshot has been created. **Record the snapshot number**.

5. Generate an AWR report. You can use the awrrpt.sql script located in the \$ORACLE_HOME/rdbms/admin directory to create an AWR report or Enterprise Manager Cloud Control. The solution in this document uses Enterprise Manager Cloud Control.
- Still on the Snapshots page, select **View Report** from the Actions drop-down list. Make sure that your first AWR snapshot is selected. This is the snapshot number that you recorded in step 2j. Click **Go**.

Snapshots

A snapshot is a collection of database statistics at a single point in time. You can use the information in snapshots to diagnose database problems.

Select Beginning Snapshot

Go To Time 1/10/14 8:00 AM Go
(Example: 12/15/03)

Select	ID	Capture Time	Collection Level
<input type="radio"/>	726	Jan 9, 2014 12:00:22 PM	TYPICAL
<input type="radio"/>	727	Jan 9, 2014 1:00:26 PM	TYPICAL
<input type="radio"/>	728	Jan 9, 2014 2:00:32 PM	TYPICAL
<input type="radio"/>	729	Jan 9, 2014 2:27:39 PM	TYPICAL
<input type="radio"/>	730	Jan 9, 2014 2:32:15 PM	TYPICAL
<input type="radio"/>	731	Jan 9, 2014 4:00:38 PM	TYPICAL
<input type="radio"/>	732	Jan 9, 2014 5:00:43 PM	TYPICAL
<input checked="" type="radio"/>	739	Jan 10, 2014 8:01:27 AM	TYPICAL
<input type="radio"/>	740	Jan 10, 2014 8:05:05 AM	TYPICAL

- On the View Report page, make sure that your second AWR snapshot is selected. This is the snapshot that you recorded in step 4d. Click **OK**.

View Report

Beginning Snapshot ID 739
Beginning Snapshot Capture Time Jan 10, 2014 8:01:27 AM

Select Ending Snapshot

Go To Time 1/10/14 8:00 AM Go
(Example: 12/15/03)

Select	ID	Capture Time	Collection Level
<input checked="" type="radio"/>	740	Jan 10, 2014 8:05:05 AM	TYPICAL

- The processing page appears. Wait until the report is created.

6. Review the report. What can you determine from this report about the performance of this workload?
 - a. After the report is created, the Snapshot Details page appears where you can see the report.
 - b. Scroll down to the “**Top 10 Foreground Events by Total Wait Time**” section. What wait event accounts for most of the %DB Time? *Buffer busy waits*

Top 10 Foreground Events by Total Wait Time						
Event	Waits	Total Wait Time (sec)	Wait Avg(ms)	% DB time	Wait Class	
buffer busy waits	39,393	211.9	5	62.6	Concurrency	
enq: HW - contention	630	58.9	93	17.4	Configuration	
DB CPU		21.7		6.4		
free buffer waits	16	11.8	738	3.5	Configuration	
db file sequential read	1,883	7.9	4	2.3	User I/O	
log file sync	25	3.7	147	1.1	Commit	
log file switch (checkpoint incomplete)	4	2.1	527	.6	Configuration	
log file switch (private strand flush incomplete)	2	.8	411	.2	Configuration	
cursor: pin S	378	.8	2	.2	Concurrency	
log buffer space	12	.5	45	.2	Configuration	

- c. Scroll down to **Time Model Statistics**. Which category collects the most %DB Time?
sql execute elapsed time **Note:** Elapsed time includes wait time.
From these first-level diagnostics, you know that the buffer busy waits occurring during the SQL execution are the cause of the performance issue.

Time Model Statistics

- Total time in database user-calls (DB Time): 338.3s
- Statistics including the word “background” measure background process time,
- Ordered by % or DB time desc, Statistic name

Statistic Name	Time (s)	% of DB Time
sql execute elapsed time	333.95	98.72
DB CPU	21.70	6.42
PL/SQL execution elapsed time	3.27	0.97
parse time elapsed	1.68	0.50
hard parse elapsed time	1.60	0.47
connection management call elapsed time	0.82	0.24
PL/SQL compilation elapsed time	0.04	0.01
repeated bind elapsed time	0.00	0.00
hard parse (sharing criteria) elapsed time	0.00	0.00
DB time	338.28	
background elapsed time	47.77	
background cpu time	1.87	

[Back to Wait Events Statistics](#)

[Back to Top](#)

- d. Scroll down to the **Buffer Wait Statistics** report in the Wait Statistics section, or click the following links in succession: Back to Top, Wait Statistics (in Main Report section), and Buffer Wait Statistics. What types of blocks have buffer busy waits? *Data blocks. These could be table or index blocks.*

Buffer Wait Statistics

- ordered by wait time desc, waits desc

Class	Waits	Total Wait Time (s)	Avg Time (ms)
data block	56,279	210	4
segment header	1,438	0	0
undo header	74	0	2

[Back to Wait Statistics](#)

[Back to Top](#)

- e. Scroll down to “**Segments by Buffer Busy Waits**” or click the following links in succession: Back to Top, Segment Statistics in the Main Report section, and Segments by Buffer Busy Waits. Are the buffer busy waits against one or a few tables and indexes? If so, which segments are affected? In what tablespace do these segments exist? Yes, *one table is experiencing most of the waits. The SPCT table is experiencing the waits. The SPCT table is in the TBSSPC tablespace.*

Segments by Buffer Busy Waits

- % of Capture shows % of Buffer Busy Waits for each top segment compared
- with total Buffer Busy Waits for all segments captured by the Snapshot

Owner	Tablespace Name	Object Name	Subobject Name	Obj. Type	Buffer Busy Waits	% of Capture
SPC	TBSSPC	SPCT		TABLE	57,721	100.00

[Back to Segment Statistics](#)

[Back to Top](#)

Diagnostics: This workload is causing buffer busy waits and block contention by performing inserts from multiple processes into the same table.

Additional Information: If the tablespace has segment space management set to manual, all the processes attempt to insert rows into the same small set of blocks. If the segment space management is set to automatic, the processes will insert rows into different blocks, choosing blocks based on a hash of the `process_id`.

- f. Check the properties of the TBSSPC tablespace. Expand the **Administration** menu. Expand the **Storage** menu and select **Tablespaces**.

The screenshot shows the Oracle Enterprise Manager interface for Cloud Control 12c. The top navigation bar includes links for Enterprise, Targets, Favorites, and History, along with a search bar. Below the bar, the database name 'orcl' is shown with a green arrow icon. The main menu has tabs for Oracle Database, Performance, Availability, Schema, and Administration. The Administration tab is currently selected and has a dropdown menu. The 'Storage' option in this dropdown is also highlighted. A sub-menu for 'Tablespaces' is open, showing options like Control Files, Datafiles, Tablespaces (which is also highlighted), Make Tablespace Locally Managed..., Temporary Tablespace Groups, and Rollback Segments. On the left, a section titled 'Segments by Buffer Busy Wait' displays a table with columns: Owner, Tablespace Name, Object Name, and Subobject Name. The data shows one entry: SPC, TBSSPC, SPCT. Below the table are links to 'Back to Segment Statistics' and 'Back to Top'.

- g. Select the TBSSPC tablespace, and then click **View**. What type of Segment Space Management is being used for the TBSSPC tablespace? *The TBSSPC tablespace is set to Manual Segment Space Management. Moving the SPCT table to a tablespace with Automatic Segment Space Management could make a significant difference in performance.*

The screenshot shows the 'View Tablespace: TBSSPC' page. At the top, it says 'Tablespaces > View Tablespace: TBSSPC'. The main title is 'View Tablespace: TBSSPC'. Below it, detailed information about the tablespace is listed:

- Name**: TBSSPC
- Bigfile tablespace**: No
- Status**: ReadWrite
- Type**: Permanent
- Extent Management**: local
- Encryption**: No

A section titled 'Storage' is expanded, showing the following properties:

- Allocation Type**: Automatic
- Segment Space Management**: **Manual** (This is highlighted with a red box)
- Enable logging**: Yes
- Compression**: No Compression
- Block Size (B)**: 8192

7. To implement the changes to the SPCT table, log in to SQL*Plus as the SYSDBA user and execute the `wksh_03_01_07.sql` script. This script drops the SPCT table, drops the TBSSPC tablespace, re-creates the tablespace with the ASSM property, and creates the SPCT table. You can locate the `wksh_03_01_07.sql` script in your \$HOME/workshops directory.

```
$ sqlplus / as sysdba

SQL> @wksh_03_01_07.sql
SQL>
SQL> drop tablespace tbsspc
  2 including contents and datafiles;

Tablespace dropped.

SQL>
SQL> CREATE SMALLFILE TABLESPACE "TBSSPC"
  2 DATAFILE 'tbsspc1.dbf' SIZE 50M
  3 AUTOEXTEND ON NEXT 10M MAXSIZE 200M
  4 LOGGING
  5 EXTENT MANAGEMENT LOCAL
  6 SEGMENT SPACE MANAGEMENT AUTO;

Tablespace created.

SQL>
SQL> connect spc/spc
Connected.
SQL>
SQL> drop table spct purge;
drop table spct purge
*
ERROR at line 1:
ORA-00942: table or view does not exist

SQL> create table spct(id number, name varchar2(2000))
  2 tablespace tbsspc;

Table created.

SQL>
SQL> exec DBMS_STATS.GATHER_TABLE_STATS(
  > ownname=>'SPC', tabname=>'SPCT',
  > estimate_percent=>DBMS_STATS.AUTO_SAMPLE_SIZE);

PL/SQL procedure successfully completed.

SQL>
SQL> exit;
$
```

8. Take a new snapshot.
 - a. Navigate to the Snapshots page, and then click **Create**.
 - b. On the Confirmation page, click **Yes**.
 - c. The processing page appears. Wait until the snapshot is created.
 - d. The Confirmation message appears on the Snapshots page, indicating that a new snapshot has been created. **Record the snapshot number**.
9. Execute the `workgen` script again to start generating the workload for this practice. Wait until the “Load is finished” message appears, and then proceed to the next step.

```
$ ./workgen 3 1

The Oracle base for
ORACLE_HOME=/u01/app/oracle/product/12.1.0/dbhome_1 is
/u01/app/oracle

start 12 load_31 processes
...
Load is finshed
$
```

10. Take a fourth AWR snapshot.
 - a. Navigate to the **Snapshots** page. (Performance > AWR > AWR Administration, and then click the Snapshots number)
 - b. On the Snapshots page, click **Create**.
 - c. On the Confirmation page, click **Yes**.
 - d. The processing page appears. Wait until the snapshot is created.
 - e. The Confirmation message appears on the Snapshots page.
11. Generate and review a Compare Periods report comparing the periods between the four snapshots. You can use the `awrddrpt.sql` script located in your `$ORACLE_HOME/rdbms/admin` directory to generate the Compare Periods report or use Enterprise Manager Cloud Control. The solution in this document uses Enterprise Manager Cloud Control.

Step	Window/Page Description	Choices or Values
a.	Snapshots	Select the snapshot from step 2j. Select Compare Periods from Actions drop-down list.
b.	Compare Periods: First Period End	Select the snapshot from step 4. Click Next .
c.	Compare Periods:Second Period Start	Select Snapshot from step 8. Click Next .
d.	Compare Periods:Second Period End	Select Snapshot from step 10. Click Next .
e.	Compare Periods: Review	Click Finish .

Compare Periods: Review

Cancel Back Step 5 of 5 **Finish**

Database orcl

First Period

Beginning Snapshot ID 739	Beginning Snapshot Capture Time Jan 10, 2014 8:01:27 AM
Ending Snapshot ID 740	Ending Snapshot Capture Time Jan 10, 2014 8:05:05 AM

Second Period

Beginning Snapshot ID 741	Beginning Snapshot Capture Time Jan 10, 2014 8:11:45 AM
Ending Snapshot ID 742	Ending Snapshot Capture Time Jan 10, 2014 8:14:19 AM

Cancel Back Step 5 of 5 **Finish**

Step	Window/Page Description	Choices or Values
f.	Compare Periods: Results	Select Per Transaction from the View Data drop-down list on the General tabbed page. You can see performance differences between the two analyzed periods.

Compare Periods: Results

Change Periods

First Period

Beginning Snapshot ID 739	Beginning Snapshot Capture Time Jan 10, 2014 8:01:27 AM
Ending Snapshot ID 740	Ending Snapshot Capture Time Jan 10, 2014 8:05:05 AM

Second Period

Beginning Snapshot ID 741	Beginning Snapshot Capture Time Jan 10, 2014 8:11:45 AM
Ending Snapshot ID 742	Ending Snapshot Capture Time Jan 10, 2014 8:14:19 AM

General **Report**

View Data **Per Transaction**

Name ▲	First Period Metric Ratio	Second Period Metric Ratio	First Period Value	Second Period Value	First Period Rate Per Transaction	Sec
DB cpu (seconds)			0.00	0.00	0.00	
DB time (seconds)	■		359.35	365.72	17.97	
db block changes	■		1,461,342.00	1,458,313.00	73,067.10	
execute count	■		726,353.00	759,949.00	36,317.65	

Step	Window/Page Description	Choices or Values
g.	Compare Periods: Results	Click the Report tab to view the Workload Repository Compare Period Report.

- h. Scroll down to the **Top Timed Events** section. You can see the difference between the two periods for the buffer busy waits event. Be sure to note the difference in %DB Time. Because this value is normalized, it is the best value for comparison. (Your results will vary from the following example.) The order of the waits will likely change from the first to second run.

Top Timed Events											
• Events with a "-" did not make the Top list in this set of snapshots, but are displayed for comparison purposes											
Event	Wait Class	1st			2nd						
		Waits	Time(s)	Avg Time(ms)	%DB time	Waits	Time(s)	Avg Time(ms)	%DB time		
buffer busy waits	Concurrency	38,215	273.48	7.16	70.19	free buffer waits	Configuration	190	107.63	566.48	31.25
library cache lock	Concurrency	16	34.91	2,181.89	8.96	buffer busy waits	Concurrency	3,782	67.08	17.74	19.48
db file async I/O submit	System I/O	172	31.90	185.46	8.19	log file switch (checkpoint incomplete)	Configuration	28	35.06	1,252.13	10.18
CPU time			25.26		6.48	db file async I/O submit	System I/O	41	32.11	783.10	9.32
free buffer waits	Configuration	380	16.45	43.30	4.22	CPU time			24.47		7.11
log file parallel write	System I/O	384	14.25	37.10	3.66	log file switch (private strand flush incomplete)	Configuration	11	17.16	1,560.03	4.98
enq: KO - fast object checkpoint	Application	2	8.81	4,404.47	2.26	log file parallel write	System I/O	196	14.23	72.58	4.13
enq: HW - contention	Configuration	567	7.05	12.43	1.81	control file parallel write	System I/O	218	6.13	28.11	1.78
control file parallel write	System I/O	135	3.41	25.26	0.88	cursor: pin S	Concurrency	1,104	4.02	3.64	1.17
local write wait	User I/O	128	3.14	24.50	0.81	log file sync	Commit	49	1.59	32.46	0.46
-log file switch (checkpoint incomplete)	Configuration	4	2.47	618.04	0.63	enq: HW - contention	Configuration	51	0.22	4.25	0.06
-log file sync	Commit	29	1.19	40.87	0.30	library cache lock	Concurrency	27	0.12	4.57	0.04
-cursor: pin S	Concurrency	392	0.83	2.12	0.21						
-log file switch (private strand flush incomplete)	Configuration	2	0.46	230.03	0.12						

- i. Scroll down to the **Buffer Wait Statistics** in the Wait Stats section. Here also, you can clearly see the difference. **Note:** The Wait Time % of DB Time is the best comparison especially when the period of the two runs is different.

Buffer Wait Statistics										
Wait Time % of DB time				Total Wait Time (s)			# Waits		Avg Wait Time (ms)	
Class	1st	2nd	Diff	1st	2nd	1st	2nd	1st	2nd	%Diff
data block	0.70	0.19	-0.51	272.71	66.81	54,500	6,850	0.05	0.10	100.00
undo header	0.00	0.00	-0.00	0.57	0.01	66	4	0.09	0.03	-66.67
segment header	0.00	0.00	-0.00	0.18	0.00	1,416	7	0.00	0.00	0.00
1st level bmb	0.00	0.00	0.00	0.00	0.08	0	23	0.00	0.03	100.00

- j. Scroll down to **Top Segments by Buffer Busy Waits** in the Segment Statistics section. The number of buffer busy waits has changed. Possibly, the classes of buffer busy waits have changed as well. (This can be seen in some reports where one period or the other is reporting 0 waits.) The bitmap block (BMB) class of wait cannot occur in the first period because the BMB block exists only in ASSM tablespaces.

Top Segments by Buffer Busy Waits

- Ordered by absolute value of 'Diff' column of % of Total Buffer Busy Waits'
- Top 5 Segments from each period are compared
- Total Buffer Busy Waits First: 38,215, Second: 3,782
- Buffer Busy Wait Time First: 273.48 seconds, Second: 67.08 seconds
- Buffer Busy Wait Time as % of DB time First: 70.19%, Second: 19.48%
- Captured Buffer Busy Waits First: 55,917, Second: 6,849

Owner	Tablespace	Object Name	Subobject Name	Type	% of Total Buffer Busy Waits				Buffer Busy Waits			% of Captured Buffer Busy Waits			
					1st	1st Total	2nd	2nd Total	Diff	1st	2nd	%Diff	1st	2nd	Diff
SPC	TBSSPC	SPCT		TABLE		181.09	181.09	181.09		6,849		100.00	100.00		
** MISSING	TBSSPC	** MISSING:	** MISSING **	UNDEFINED	146.32	146.32	0.00	181.09 -146.32	55,917	0 -100.00	100.00	0.00	-100.00		
**		93325/93325													

- k. Log out of Enterprise Manager Cloud Control.

12. In preparation for the next practice, execute the `cleanup` script with the parameters as shown.

```
$ ./cleanup 3 1
```

```
The Oracle base for
ORACLE_HOME=/u01/app/oracle/product/12.1.0/dbhome_1 is
/u01/app/oracle
```

```
PL/SQL procedure successfully completed.
```

```
$
```

Practices for Lesson 4: Defining Problems

Chapter 4

Practices for Lesson 4: Overview

Practices Overview

The goal of this practice is to identify the sources of performance issues.

Practice 4-1: Using Enterprise Manager to Identify OS Issues

In this practice, the workload generator starts an application load on the database and an OS load.

Assumptions: You will start using a terminal session as the `oracle` user with the environment variables set for the `orcl` database instance.

1. Prepare the database for this practice. Change directory to `home/oracle/workshops`. Execute the `prepare` script with the `OS` parameter.

```
$ ./prepare OS
```

2. In a terminal window, use the `date` command to determine the start time of the workload.

```
$ date
Tue Jul 30 13:14:18 UTC 2013
```

3. In a terminal window, change directory to `/home/oracle/workshops`. Start the workload generator with the command `./workgen 4 1`. Verify that the workload generator has started the workload processes by executing the `ps` command.

```
$ cd /home/oracle/workshops
$ ./workgen 4 1
The Oracle base for
ORACLE_HOME=/u01/app/oracle/product/12.1.0/dbhome_1 is
/u01/app/oracle

PL/SQL procedure successfully completed.

PL/SQL procedure successfully completed.

cpus: 2
OLTP sessions=4
$ ps
12786 pts/0    00:00:00 sleep
17930 pts/0    00:00:00 cpupload2.sh
17931 pts/0    00:00:00 insert_orders.s
17932 pts/0    00:00:00 update_orders.s
17933 pts/0    00:00:00 delete_orders.s
17934 pts/0    00:00:00 insert_orders.s
17935 pts/0    00:00:00 update_orders.s
17936 pts/0    00:00:00 delete_orders.s
...
17943 pts/0    00:00:00 sqlplus
17945 pts/0    00:00:00 sqlplus
17951 pts/0    00:00:00 sqlplus
17953 pts/0    00:00:00 sqlplus
17956 pts/0    00:00:00 inf_loop.sh
17957 pts/0    00:00:00 inf_loop.sh
17958 pts/0    00:00:00 sleep
...
18081 pts/0    00:00:00 sleep
18229 pts/0    00:00:00 sleep
19729 pts/0    00:00:00 ps
```

```
20683 pts/0    00:00:00 bash $
```

4. Use Enterprise Manager Cloud Control to determine the source of the loads.
 - a. Open the browser and enter the following URL: <https://<hostname>:7802/em>
 - b. Enter **SYSMAN** in the User Name field and **oracle_4U** in the Password field. Click **Login**.
 - c. Expand the Targets menu and select **Databases**.
 - d. Click the **orcl** database link.
 - e. Set Auto Refresh for 30 Seconds
5. Wait for a while until the Activity Class (Active Sessions) graph shows some waits. This may take two to three minutes. Find the OS process that is consuming the most CPU.
 - a. Observe the **Host CPU** measurement. Notice that a significant percentage of the Host CPU is dedicated to **Other** processes. This indicates that the OS processes, NOT the database processes are using the CPU.



- b. Expand the **Targets** menu and select **Hosts**.
- c. Click the link for your host.

- d. Expand **Host**. Expand **Monitoring** and click **CPU Details**.

- e. View the **Top 10 Processes (ordered by CPU)** section. If necessary, refresh the page a few times until an OS process that does not have the SID name in it or is running from the `ORACLE_HOME` directory appears. The `inf_loop.sh` process should appear although it may not be the top process.

Command	CPU Utilization (%)	CPU Total (seconds)	Resident Size (KB)	Virtual Size (KB)	Owner	Process ID
/bin/bash /home/oracle/workshops/inf_loop.sh	4	10	1,280	106,144	oracle	25824
/bin/bash /home/oracle/workshops/inf_loop.sh	4	10	1,280	106,144	oracle	25823
oraclecl (LOCAL=NO)	0.6	1	155,480	829,928	oracle	31054

- f. Return to the terminal window and stop the `inf_loop.sh` process by using the `killall` command and the process name you determined in the previous step.

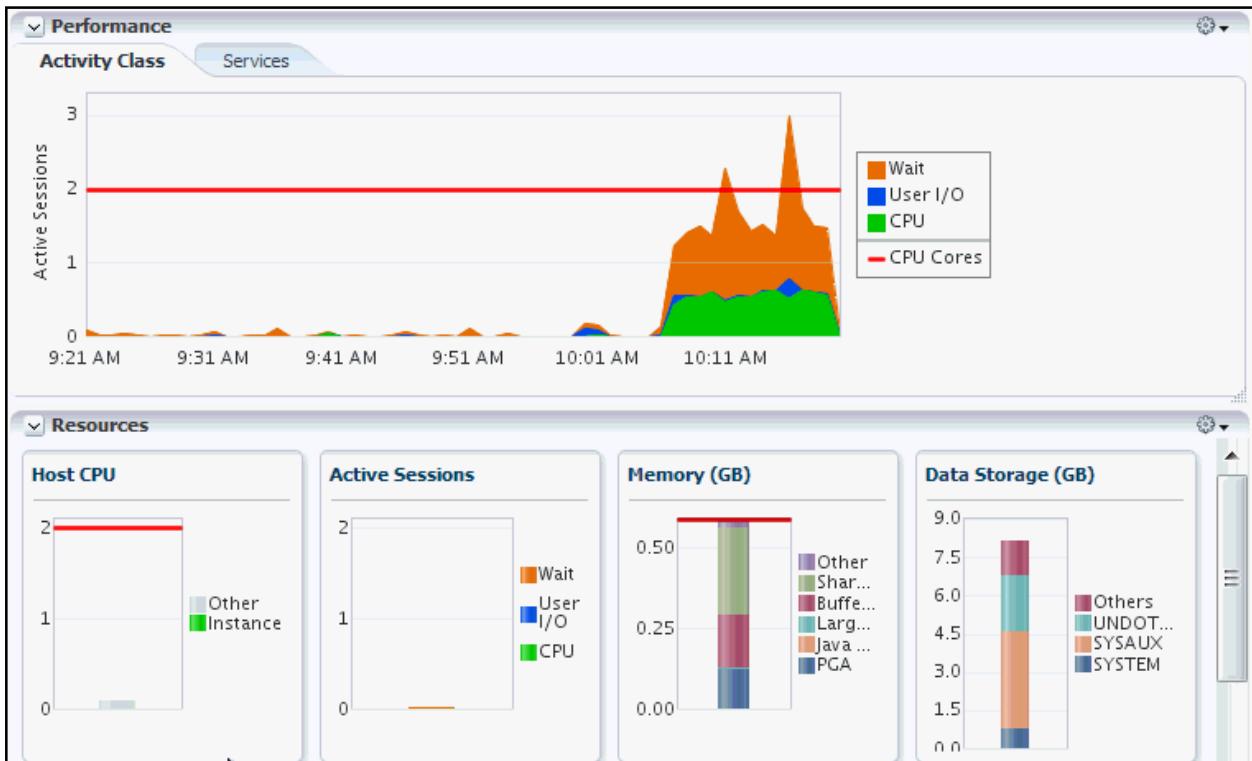
```
$ killall inf_loop.sh
$
```

6. Refresh the CPU details page. Notice that the `inf_loop.sh` processes are gone.
7. Return to the `orcl` home page. After a minute or two, you will notice that the Other processes are taking much less of the Host CPU and the Active Session waits are reduced.

- Stop the workload by executing the `rm runload` command.

```
$ rm runload  
$
```

- Return to the `orcl` database home page. After a few minutes, you should observe that there are no longer any waits shown in the Activity Class graph. In this practice, you found and stopped the OS processes that were consuming CPU.



- Log out of Enterprise Manager Cloud Control.

Practices for Lesson 5: Using Metrics and Alerts

Chapter 5

Practices for Lesson 5: Overview

Practices Overview

The goal of this practice is to create and monitor a server-generated alert.

Assumptions: You are using a terminal session logged in as the `oracle` OS user. The `orcl` instance environment variables are set.

Practice 5-1: Using Metrics

This practice shows you how to set up server-generated alerts and how to retrieve alert information from Enterprise Manager. In this practice, you see how alerts are triggered. However, this practice does not involve analysis of the cause of the generated alerts. This analysis is performed during subsequent practices by using AWR snapshots and ADDM.

1. Change to the /home/oracle/workshops directory. Execute the prepare script to set up the necessary objects before starting the workload.

```
$ cd /home/oracle/workshops
$ ./prepare 5 1
The Oracle base for
ORACLE_HOME=/u01/app/oracle/product/12.1.0/dbhome_1 is
/u01/app/oracle

Database closed.
Database dismounted.
ORACLE instance shut down.
ORACLE instance started.

Total System Global Area  501059584 bytes
Fixed Size                  2290024 bytes
Variable Size                465571480 bytes
Database Buffers              25165824 bytes
Redo Buffers                  8032256 bytes
Database mounted.
Database opened.
drop tablespace tbsjfv including contents and datafiles
*
ERROR at line 1:
ORA-00959: tablespace 'TBSJFV' does not exist

Tablespace created.

PL/SQL procedure successfully completed.

Drop user jfv cascade
*
ERROR at line 1:
ORA-01918: user 'JFV' does not exist

User created.

Grant succeeded.

drop table jfvt purge
*
ERROR at line 1:
```

```
ORA-00942: table or view does not exist
```

Table created.

PL/SQL procedure successfully completed.

PL/SQL procedure successfully completed.

\$

2. Using SQL*Plus, determine the current threshold values for the **DB_TIME_WAITING** metric, specifically the **Concurrency** wait class.
 - a. Use the `$HOME/labs/lab_05_01_02.sql` script to determine the values.

```
$ cd $HOME/labs
$ sqlplus / as sysdba

SQL> @lab_05_01_02.sql
SQL>
SQL> variable wo number
SQL> variable wv varchar2(10)
SQL> variable co number
SQL> variable cv varchar2(10)
SQL> variable op number
SQL> variable ct number
SQL>
SQL> BEGIN
 2  DBMS_SERVER_ALERT.get_threshold(
 3    metrics_id => dbms_server_alert.DB_TIME_WAITING,
 4    warning_operator      => :wo,
 5    warning_value        => :wv,
 6    critical_operator    => :co,
 7    critical_value       => :cv,
 8    observation_period   => :op,
 9    consecutive_occurrences => :ct,
10    instance_name       => 'orcl',
11    object_type          =>
12      dbms_server_alert.OBJECT_TYPE_EVENT_CLASS,
13    object_name           => 'Concurrency') ;
14 END;
15 /
```

```

PL/SQL procedure successfully completed.

SQL>
SQL> print wv

WV
-----
30

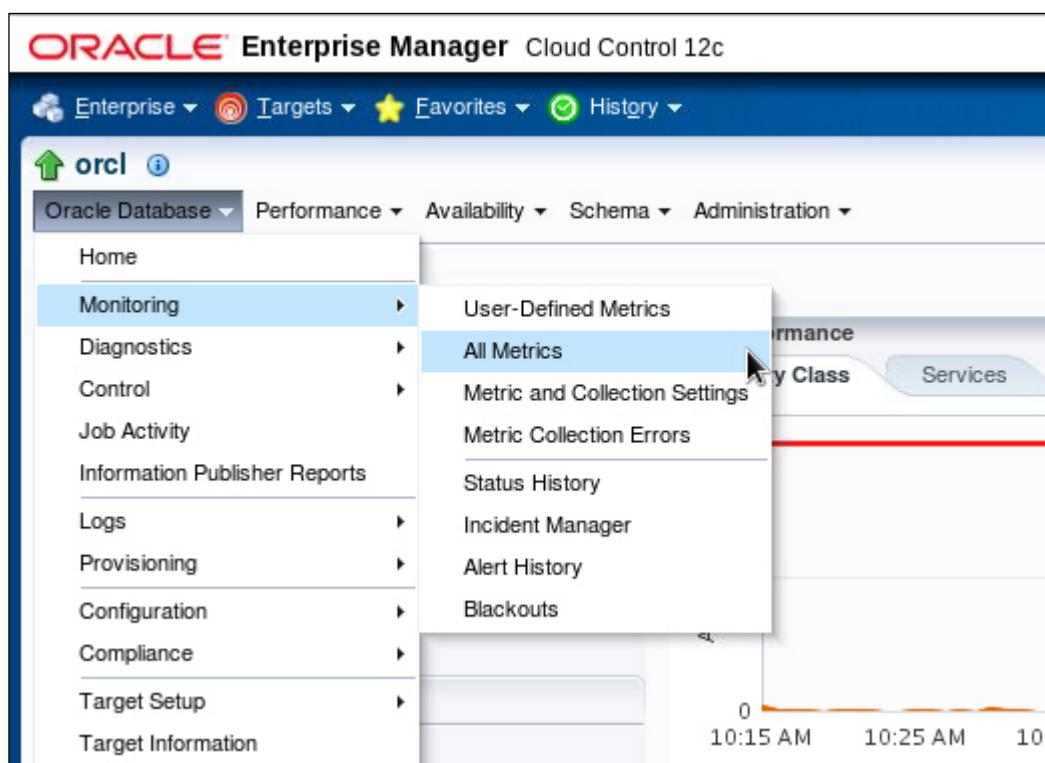
SQL>
SQL> col object_name format a20
SQL> col metrics_name format a25
SQL> col warning_value format a10
SQL> col critical_value format a10
SQL>
SQL> select object_name,warning_value,critical_value
  2  from dba_thresholds
  3  where metrics_name='Database Time Spent Waiting (%)';

OBJECT_NAME          WARNING_VALUE CRITICAL_V
-----
Concurrency           30

SQL>
SQL> exit;
Disconnected from Oracle Database 12c Enterprise Edition Release
12.1.0.1.0 - 64bit Production
With the Partitioning, OLAP, Advanced Analytics and Real
Application Testing options
$
```

3. Using Enterprise Manager Cloud Control, change the thresholds for the Concurrency class to 30% for the warning level and 50% for the critical level.
 - a. Open the browser and enter the following URL: <https://localhost:7802/em>.
 - b. Enter **sysman** in the User Name field and **oracle_4U** in the Password field. Click **Login**.
 - c. Expand the **Targets** menu and select **Databases**.
 - d. Click the link for the **orcl** database.

- e. Expand the **Oracle Database** menu. Expand the **Monitoring** menu and select **All Metrics**.



- f. On the All Metrics page, scroll to **Waits by Wait Class** and expand it. Click **Database Time Spent Waiting (%)**.

The screenshot shows the Oracle Enterprise Manager interface under the 'All Metrics' tab. In the left sidebar, under 'View By', 'Metrics' is selected. A tree view shows various metrics, with 'Waits by Wait Class' expanded. Under 'Database Time Spent Waiting (%)', 'Concurrency' is selected. The main panel displays a table titled 'Database Time Spent Waiting (%)' with the following data:

Wait Class	Average Value
Administrative	0.05
Application	0.07
Cluster	0
Commit	23.17
Concurrency	0.48
Configuration	2.5

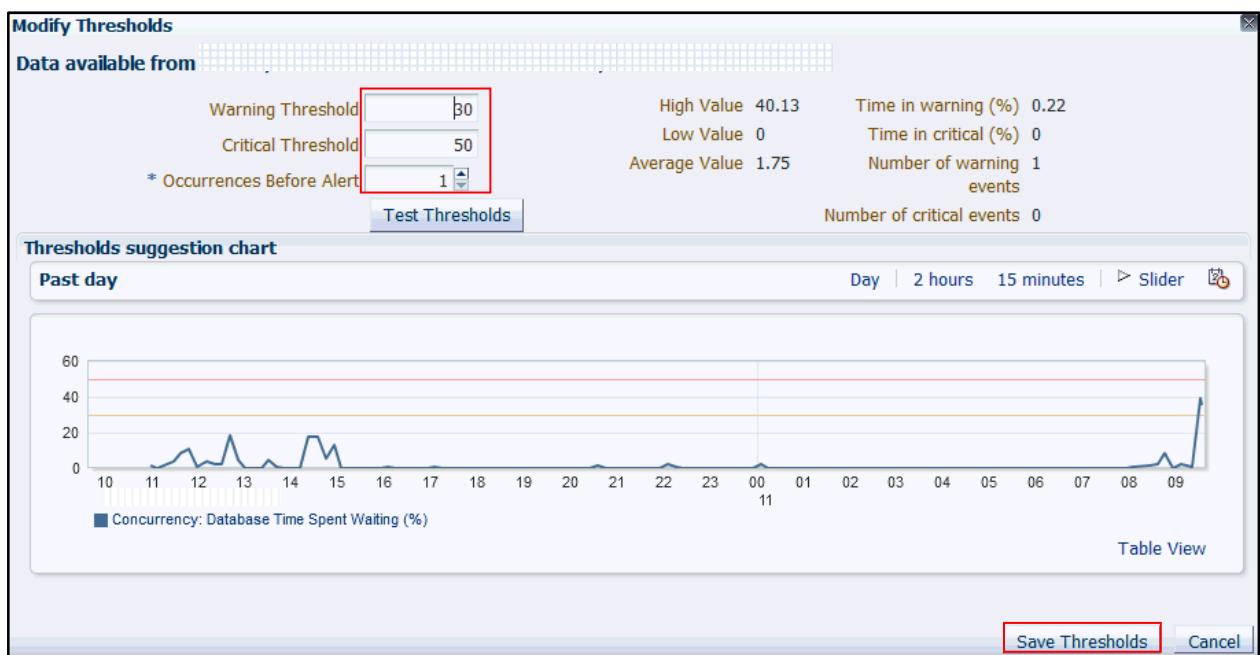
Below the table, a section titled 'Wait Class : Concurrency' contains 'Statistics' and a 'Metric Value History' chart. The chart shows historical data points for the Concurrency wait class.

- g. On the Database Time Spent Waiting (%) page, you can see all thresholds set for each class. Click **Concurrency**.
 h. When the page updates, the Wait Class: Concurrency section appears.
 i. In the Wait Class: Concurrency section, click **Modify Thresholds**.

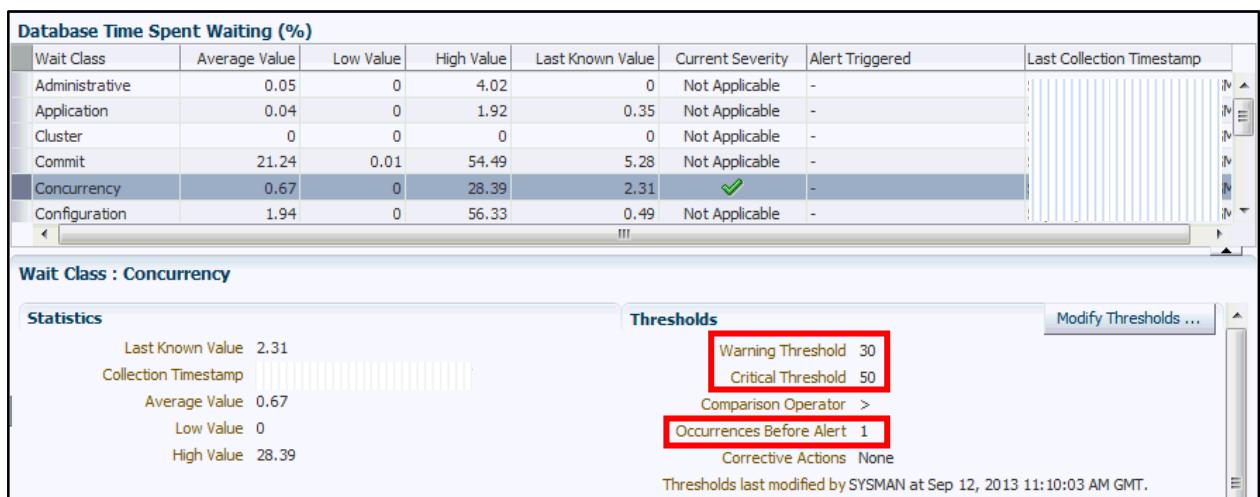
The screenshot shows the 'Wait Class : Concurrency' page. The 'Statistics' section on the left shows current values for the Concurrency wait class. The 'Thresholds' section on the right lists threshold settings. A red box highlights the 'Modify Thresholds ...' button in the top right corner of the thresholds section.

Statistics	Thresholds
Last Known Value: 2.29	Warning Threshold: Not Defined
Collection Timestamp: [Timeline]	Critical Threshold: Not Defined
Average Value: 0.41	Comparison Operator: >
Low Value: 0	Occurrences Before Alert: 3
High Value: 35.46	Corrective Actions: None

- j. Enter **30** in the Warning Threshold field, **50** in the Critical Threshold field, and **1** in the Occurrences Before Alert field. Click **Test Thresholds**. Click **Save Thresholds**.



- k. The new threshold values are shown in the Thresholds section.



4. Execute the **lab_05_01_04.sql** script to verify your change.

```
$ sqlplus / as sysdba

SQL> @lab_05_01_04.sql

SQL>
SQL> col object_name format a20
SQL> col metrics_name format a25
SQL> col warning_value format a10
SQL> col critical_value format a10
SQL>
SQL> select object_name,warning_value,critical_value
  2  from dba_thresholds
  3  where metrics_name=
```

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

```

4          'Database Time Spent Waiting (%)';

OBJECT_NAME      WARNING_VA CRITICAL_V
-----
Concurrency      30.0       50.0

SQL>
SQL> exit;

$
```

5. Change to the \$HOME/workshops directory. Execute the workgen script to start the workload for this practice. While the script is executing, observe the variation of the curve on the Wait Class Concurrency page. After the workload is finished, find the alert history from a SQL*Plus session. Check both the outstanding alerts and the alert history. The workload script takes up to six minutes to complete on some machines.
 - a. Change to the \$HOME/workshops directory and execute the workgen script. Proceed to the next step after starting the execution of the workgen script.

```

$ cd $HOME/workshops
$ ./workgen 5 1

The Oracle base for
ORACLE_HOME=/u01/app/oracle/product/12.1.0/dbhome_1 is
/u01/app/oracle
users= 75
maxrec=51440
... The following messages will appear for 1 to 6 minutes ...
PL/SQL procedure successfully completed.

PL/SQL procedure successfully completed.
```

PL/SQL procedure successfully completed.

PL/SQL procedure successfully completed.

PL/SQL procedure successfully completed.

\$

- b. While the workgen script executes, go to the orcl database home page. Set the Auto Refresh to **30 Seconds**. After a while, you should see a warning or critical alert in the Incidents and Problems section.

Incidents and Problems						
View	Target	Local target and related targets	Category	All		
Summary	Target	Severity	Status	Escalation level	Type	Time since last update
Metrics "Database Time S			New	-	Incident	0 days 0 hours

- 6. After the script finishes, you can retrieve the history of your alerts by using SQL*Plus. Depending on your timing, the alert may appear in the DBA_OUTSTANDING_ALERTS or DBA_ALERT_HISTORY view. If the alert is not yet cleared, it shows in the DBA_OUTSTANDING_ALERTS view. After it is cleared, it is visible in the DBA_ALERT_HISTORY view. The following example shows the output when the alert has been cleared. You can use lab_05_01_06.sql in the \$HOME/labs directory.

Note: DBTIME_IN_WAIT is the percentage value that is being compared to the thresholds.

```
$ sqlplus / as sysdba

SQL> @../labs/lab_05_01_06.sql
SQL>
SQL> col reason format a75
SQL>
SQL> SELECT reason FROM DBA_OUTSTANDING_ALERTS;

no rows selected

SQL>
SQL> SELECT TO_CHAR(begin_time,'hh24:mi:ss'),
2      dbtime_in_wait,average_waiter_count
3  FROM V$WAITCLASSMETRIC_HISTORY
4 WHERE wait_class#=4
5   AND wait_class_id=3875070507
6   AND begin_time > SYSDATE-(10/1440);
```

```
TO_CHAR( DBTIME_IN_WAIT AVERAGE_WAITER_COUNT
-----
12:39:26      0          0
12:38:25 .001043592 1.1653E-07
12:37:25 66.5233478 2.07057105
12:36:26 85.0239299 36.9401072
12:35:26 82.9625412 47.696141
12:34:26 89.3825446 26.4765881
12:33:26 66.6765431 .268499334
12:32:26      0          0
12:31:26      0          0

9 rows selected.

SQL>
SQL>
SQL> -- Cleared!
SQL>
SQL> SELECT reason,resolution
  2  FROM DBA_ALERT_HISTORY
  3 WHERE reason like
  4       '%Database Time Spent Waiting (%)%Concurrency%'
  5 AND TO_DATE(SUBSTR(TO_CHAR(creation_time),1,18) ||
  6              SUBSTR(TO_CHAR(creation_time),26,3) ,
  7              'DD-MON-YY HH:MI:SS AM') > SYSDATE-(10/1440)
  8 ORDER BY creation_time DESC;

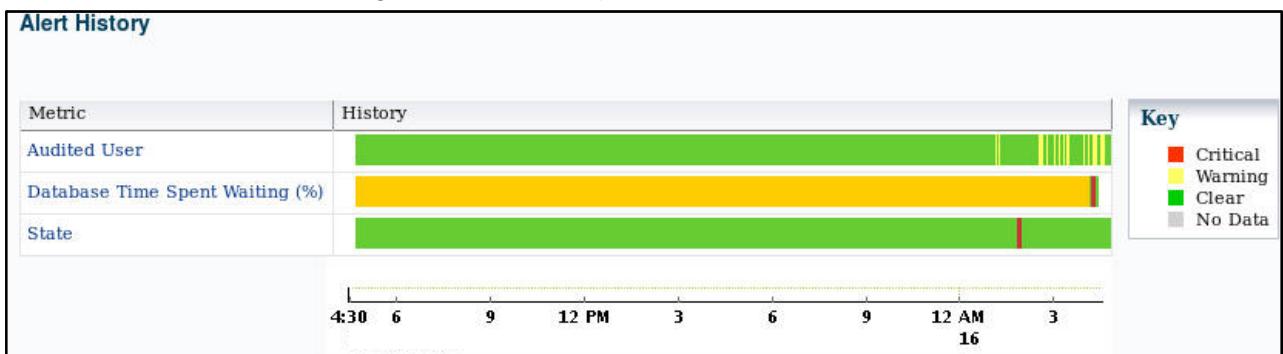
no rows selected

SQL>
SQL> EXIT;
Disconnected from Oracle Database 12c Enterprise Edition Release
12.1.0.1.0 - 64bit Production
With the Partitioning, OLAP, Advanced Analytics and Real Application
Testing options
$
```

7. Use Enterprise Manager Cloud Control to check the alert history.
- Expand **Oracle Database**. Expand **Monitoring** and click **Alert History**.

The screenshot shows the Oracle Enterprise Manager Cloud Control interface. The top navigation bar includes links for Enterprise, Targets, Favorites, and History. Below the bar, the database name 'orcl' is displayed with a green arrow icon. The main menu has 'Oracle Database' selected, followed by sub-options: Performance, Availability, Schema, and Administration. A context menu is open over the 'Alert History' link in the Monitoring submenu. The menu items are: User-Defined Metrics, All Metrics, Metric and Collection Settings, Metric Collection Errors, Status History, Incident Manager, Alert History (which is highlighted with a blue selection bar), and Blackouts.

- Are the alerts that were generated in this practice visible?



- Click **Database Time Spent Waiting (%)** to once again display the Database Time Spent Waiting (%) page.
- Click the **Concurrency** link to display the Database Time Spent Waiting (%) and Wait Class Concurrency page with the graph showing the waits and thresholds.
- Click Modify Thresholds.
- Set Warning Threshold to **Blank**, Critical Threshold to **Blank**, and Occurrences Before Alert to **3**. Click **Save Thresholds**.
- Log out of Enterprise Manager Cloud Control.
- On the Confirmation page, select **Logout of Enterprise Manager and all targets**. Click Logout.

8. Execute the `cleanup` script to prepare your environment for additional practices.

```
$ ./cleanup 5 1
The Oracle base for
ORACLE_HOME=/u01/app/oracle/product/12.1.0/dbhome_1 is
/u01/app/oracle

User dropped.

Tablespace dropped.

PL/SQL procedure successfully completed.

$
```

Practice 5-2: Disable Database Vault Metric Collection

Overview

In this practice, you will disable the database vault metric collections.

Tasks

Using Enterprise Manager Cloud Control, disable collections for Database Vault Metrics.

- Open the browser and enter the following URL: <https://localhost:7802/em>.

Step	Window/Page Description	Choices or Values
b.	Enterprise Manager login	Enter: Username: sysman Password: oracle_4U Click Login .
c.	Enterprise Summary	Expand the Targets menu and select Databases .
d.	Databases	Click the link for the orcl database.
e.	Orcl	Expand the Oracle Database menu. Expand the Monitoring menu and select Metric and Collection Settings .
f.	Metric and Collection Settings	Change View to All Metrics . Click Collapse All . Expand orcl . Find “Database Vault Attempted Violations - Command Rules.” Click Every 1 Hour link.
g.	Edit Collection Settings: Database Vault Attempted Violations - Command Rules	Under Collection Schedule: Click Data Collection Disable Click Continue
h.	Metric and Collection Settings	Click Collapse All . Expand orcl . Find “Database Vault Attempted Violations – Realms.” Click the Every 1 Hour link.
i.	Edit Collection Settings: Database Vault Attempted Violations - Realms	Under Collection Schedule: Click Data Collection Disable Click Continue
j.	Metric and Collection Settings	Click Collapse All . Expand orcl . Find “Database Vault Configuration

Step	Window/Page Description	Choices or Values
		Issues - Command Rules.” Click the Every 1 Hour link.
k.	Edit Collection Settings: Database Vault Configuration Issues - Command Rules	Under Collection Schedule: Click Data Collection Disable Click Continue
l.	Metric and Collection Settings	Click Collapse All . Expand orcl . Find “Database Vault Configuration Issues – Realms.” Click Every 1 Hour link
m.	Edit Collection Settings: Database Vault Configuration Issues - Realms	Under Collection Schedule: Click Data Collection Disable Click Continue
n.	Metric and Collection Settings	Click Collapse All . Expand orcl . Find “Database Vault Policy Changes.” Click the Every 1 Hour link.
o.	Edit Collection Settings: Database Vault Policy Changes	Under Collection Schedule: Click Data Collection Disable . Click Continue .
p.	Metric and Collection Settings	Click OK .
q.	Confirmation	Click OK .
r.	orcl	Click Log Out .

Practices for Lesson 6: Using Baselines

Chapter 6

Practices for Lesson 6: Overview

Practices Overview

The goal of this practice is to create a static baseline and a repeating baseline. The static baseline is used to create an adaptive threshold on a metric.

Practice 6-1: Using Baselines

In this practice, you create a static baseline over the snapshots that are present in the database, from 9:00 PM the day before this class started to 5:00 AM the following morning. This set of snapshots should be in the range 1–100. (Your instructor may have different instructions.) Name the baseline `LAST_NIGHT`.

1. Use Enterprise Manager Cloud Control to create the `LAST_NIGHT` baseline.

Step	Window/Page Description	Choices or Values
a.		Log in to the EMCC with: Username: sysman Password: oracle_4U
b.	Enterprise Summary	Navigate to the <code>orcl</code> database home page.
c.	<code>orcl</code> database home	From the database menu, select Performance > AWR > AWR Administration .
d.	Database Login	Set Preferred Credential: SYSDBA Database Credential Click Login .
e.	Automatic Workload Repository	Click the Snapshots link.
f.	Snapshots	Record the time and date of snapshots 2 and 100 (or the most recent, whichever is the least). 2 Date _____ Time _____ 100 Date _____ Time _____ Click Automatic Workload Repository in the breadcrumb at the top of the page to return to the Automatic Workload Repository page.
g.	Automatic Workload Repository	Click the Baselines link.
h.	AWR Baselines	Click Create .
i.	Create Baseline: Baseline Interval Type	Select Single . Click Continue .
j.	Create Baseline: Single Baseline	Enter a baseline name in the Baseline Name field. Baseline Name: LAST_NIGHT Select Snapshot Range. Move the chart view range until you can see the set of snapshots 2 through 100. To change the days shown, expand the Change Chart Time Period. Set the Chart Start Date to Snapshot 2 date. Set Chart End Date to Snapshot 100 date.

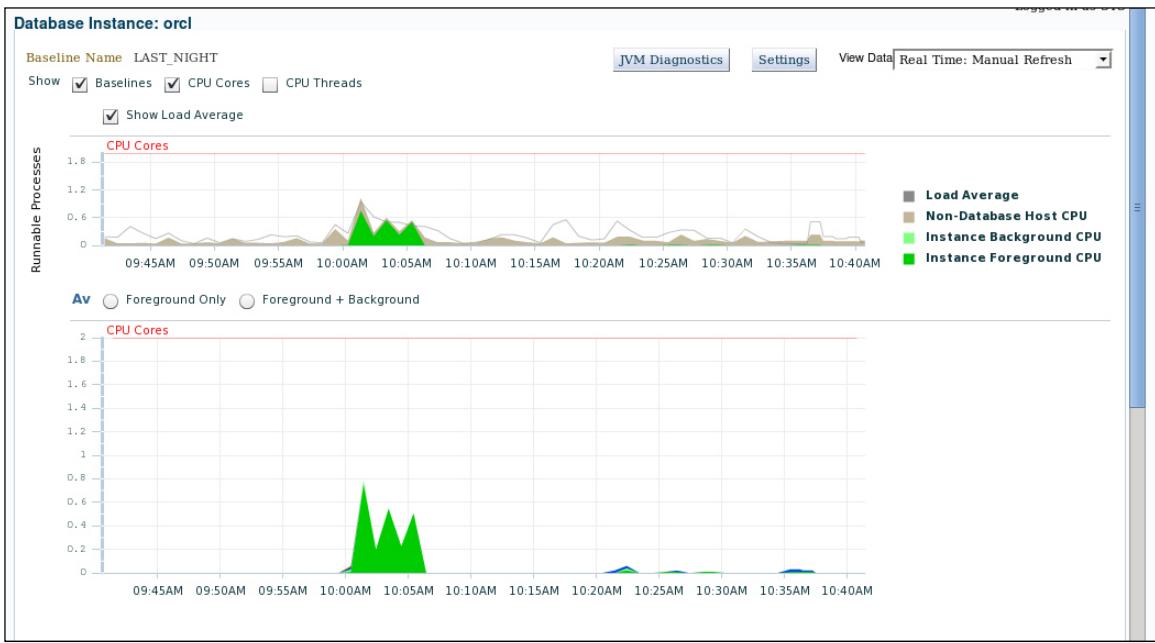
Step	Window/Page Description	Choices or Values
		<p>Click Go.</p> <p>Select Period Start Time.</p> <p><i>Using pulldown, select the Snapshot 2 time</i></p> <p>Select Period End Time. Place the cursor above the snapshot icons, and click the icon corresponding to snapshot 100.</p> <p>Click Finish.</p>

2. On the AWR Baselines page, calculate the statistics for the LAST_NIGHT baseline.

Step	Window/Page Description	Choices or Values
a.	AWR Baselines	<p>Select LAST_NIGHT.</p> <p>From the Actions drop-down list, select Schedule Statistics Computation.</p> <p>Click Go.</p>
b.	Compute Threshold Statistics: LAST_NIGHT	<p>Select Immediately.</p> <p>Click Submit.</p>
c.	AWR Baselines	Click Refresh until Statistics Computed becomes "Yes."

3. Set the Performance page to show the LAST_NIGHT baseline statistics.

Step	Window/Page Description	Choices or Values
a.	AWR Baselines	Expand the Performance menu and select Performance Home .
b.	Performance home	Click Settings .
c.	Performance Page Settings	<p>Select "Show the 99th percentile line using a static baseline with computed statistics."</p> <p>From the Baseline Name drop-down list, select LAST_NIGHT.</p> <p>Click OK.</p>



4. Create a template that creates a baseline for the Monday maintenance window for the next month. Keep the baseline for 60 days. The weekday maintenance window start and end times can be found on the Oracle Scheduler Windows page.

Step	Window/Page Description	Choices or Values
a.	Performance	Select Administration > Oracle Scheduler > Windows from the menu.
b.	Scheduler Windows	Record the start time of MONDAY_WINDOW . This is shown in the Next Open Date column and should be 10:00 PM. Record the duration of MONDAY_WINDOW .
c.	Scheduler Windows	Select Performance > AWR > AWR Administration .
d.	Automatic Workload Repository	Click the Baselines link.
e.	AWR Baselines	Click Create .
f.	Create Baseline: Baseline Interval Type	Click Repeating . Click Continue .
g.	Create Baseline: Repeating Baseline Template (See the example in the following screenshot.)	Set values on the page: Baseline Name Prefix: MONDAY Baseline Time Period: Start Time: 10 PM Baseline Time Period: Duration: 4 Frequency: Weekly and Monday Interval of Baseline Creation: Start Time: Today 10:45 AM Interval of Baseline Creation: End Time: One Month from today 11:00 AM

Step	Window/Page Description	Choices or Values
		Retention Time (Days): 60 Click Finish .
h.	AWR Baselines	Click AWR Baseline Templates in the Related Links section.
i.	AWR Baseline Templates	Verify that the MONDAY template was created as expected.

The screenshot shows two pages from Oracle Enterprise Manager Cloud Control 12c:

- Create Baseline: Repeating Baseline Template**: This page allows defining a baseline template with a repeating time period. It includes fields for Baseline Name Prefix (MONDAY), Baseline Time Period (Start Time: 10 PM, Duration: 4 hours), Frequency (Weekly, Monday selected), Interval of Baseline Creation (Start Time: Jul 10, 2013, End Time: Aug 10, 2013), and Purge Policy (Retention Time: 60 days). A tip note states: "TIP A baseline template with the same name as the baseline name prefix will be created."
- AWR Baseline Templates**: This page lists existing baseline templates. It shows one entry: MONDAY, Repeating Start Time: Jul 10, 2013 10:45:00 AM, Repeating End Time: Aug 10, 2013 11:00:00 AM, Day of the Week: MONDAY, Start Time: 10:00 PM, Retention Days: 60, and Expired: No.

5. Define an alert threshold for transactions per second based on the LAST_NIGHT baseline. Set the critical threshold to 120% of maximum and the warning to 110% of maximum.

Step	Window/Page Description	Choices or Values
a.	AWR Baseline Templates	Click AWR Baselines in the Related Links section.
b.	AWR Baselines	Click Baseline Metric Thresholds in the Related Links section.
c.	Baseline Metric	Change the view to Basic Metrics .

Step	Window/Page Description	Choices or Values
	Thresholds	In the Workload Volume Section, edit the Number of Transactions (per second) metric. (Click the pencil icon on the right.)
d.	Edit Thresholds: Number of Transactions (per second)	<p>Set values:</p> <p>Name: LAST_NIGHT</p> <p>Threshold Type: Percentage of Maximum</p> <p>Critical: 120</p> <p>Warning: 110</p> <p>Occurrences: 1</p> <p>Click Preview. To move the active window slider (in the upper left) over the period of time that covers the LAST_NIGHT baseline, click the day that has the majority of the baseline statistics in it.</p> <p>Write down the estimated number of transactions per second required to produce a critical alert:_____.</p> <p>(This should be in the range of 8 to 10.)</p> <p>Click Apply Thresholds.</p>

Database Instance: orcl > Baseline Metric Thresholds > Edit Thresholds: Number of Transactions (per second)

Edit Thresholds: Number of Transactions (per second)

AWR Baseline

Name: LAST_NIGHT

Threshold Settings

Threshold Type:	Percentage of Maximum
Critical:	120 %
Warning:	110 %
Occurrences:	1

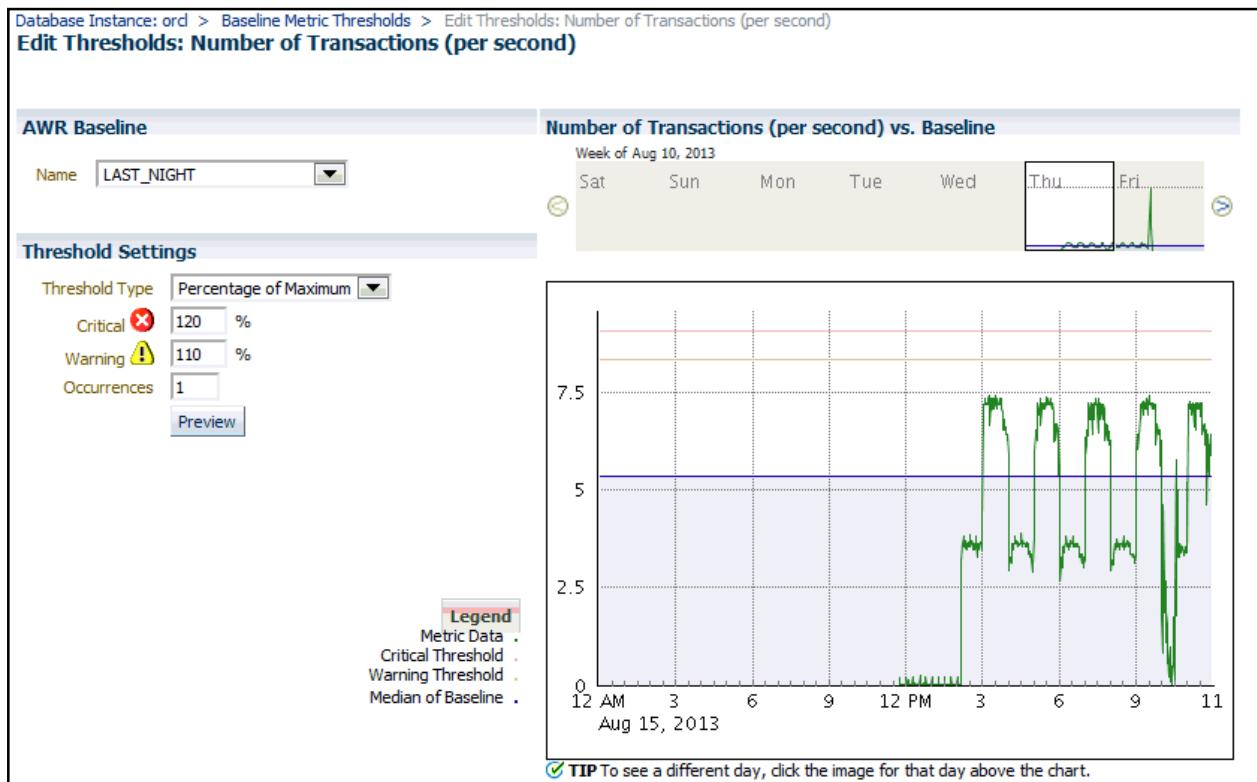
Number of Transactions (per second) vs. Baseline

Week of Jul 6, 2013

Number of Transactions (per second) vs. Baseline

Week of Jul 6, 2013

Sat Sun Mon Tue Wed Thu Fri



6. Open a terminal window, set the environment variables for the `orcl` instance, and run a workload by executing the `workgen` script. The script will ask for the number of transactions per second that is required to produce a critical alert. Enter the value you recorded in Step 5d. If the script is running correctly, you can run the `ps` command and see the `baseline_wkld.sh` process.

```
$ . oraenv
ORACLE_SID = [oracle] ? orcl
The Oracle base for
ORACLE_HOME=/u01/app/oracle/product/12.1.0/dbhome_1 is
/u01/app/oracle
$ cd /home/oracle/workshops
$ ./workgen 6 1
The Oracle base for
ORACLE_HOME=/u01/app/oracle/product/12.1.0/dbhome_1 is
/u01/app/oracle

Enter the number of transactions needed to produce a critical
alert:8
$ ps
 PID TTY          TIME CMD
29422 pts/2    00:00:00 bash
31484 pts/2    00:00:00 baseline_wkld.sh
31511 pts/2    00:00:00 sleep
31512 pts/2    00:00:00 ps
```

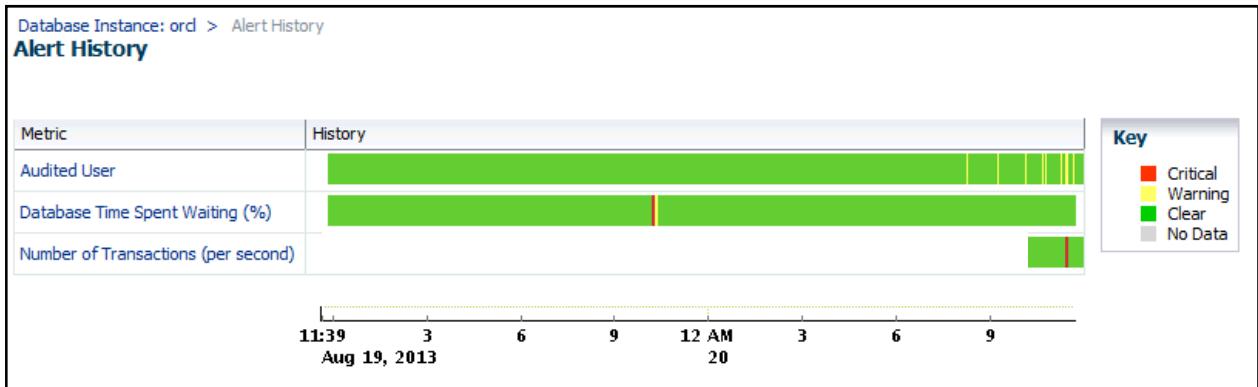
7. Return to Enterprise Manager Cloud Control. On the Database Home page, set Auto Refresh to 30 seconds. After some time (three to five minutes), an alert appears in the Incidents and Problems section of the Database home page of EM. This alert is named “Number of Transactions (per second).”

Incidents and Problems						
View	Target	Local target and related targets	Category	All		
Summary	Target	Severity	Status	Escalation level	Type	Time since last update
Metrics "User Transaction Per Sec"			New	-	Incident	0 days 0 hours
Count of targets not uploading exc			New	-	Incident	10 days 19 hours

8. Return to the terminal window and stop the workload by deleting the `runload` file from the `/home/oracle/workshops` directory.

```
$ cd /home/oracle/workshops
$ rm runload
$
```

9. A while later (three to five minutes), the alert is cleared and no longer appears on the home page. The alert history shows that this metric had one or more critical alerts. View the alert history for this metric. Expand Menu **Oracle Database > Monitoring > Alert History**.



10. Clean up from this practice by removing the adaptive threshold.

Step	Window/Page Description	Choices or Values
a.	Database Home page	From the menu, select Performance > AWR > AWR Administration .
b.	Automatic Workload Repository	Click the Baselines link.
c.	AWR Baselines	Click the Baselines Metric Thresholds link.
d.	Baseline Metric Thresholds	Click “ Number of Transactions (per second) .”
e.	Edit Thresholds: Number of Transactions (per second)	Click Clear Thresholds .

11. Log out of Enterprise Manager Cloud Control.

Practices for Lesson 7: Using AWR-Based Tools

Chapter 7

Practices for Lesson 7: Overview

Practices Overview

The goal of this practice is to become familiar with the AWR tools used for diagnosis.

Practice 7-1: Using AWR-Based Tools

In this practice, you use AWR-based tools to find and tune a database problem. Note that during this practice, you explicitly capture AWR snapshots. However, by default, the system automatically captures AWR snapshots every hour.

1. Execute the prepare script to set up this practice. You can find this script in your \$HOME/workshops directory. The script is displayed here to show the type of segment management that is specified for the tablespace.

```
$ cd $HOME/workshops
$ ./prepare 7 1

The Oracle base for
ORACLE_HOME=/u01/app/oracle/product/12.1.0/dbhome_1 is
/u01/app/oracle

SQL> Connected.
SQL> Database closed.
Database dismounted.
ORACLE instance shut down.
ORACLE instance started.

Total System Global Area  501059584 bytes
Fixed Size                  2290024  bytes
Variable Size                465571480  bytes
Database Buffers              25165824  bytes
Redo Buffers                  8032256  bytes
Database mounted.
Database opened.
SQL> drop tablespace tbsspc including contents and datafiles;

Tablespace dropped.

SQL>
SQL> CREATE SMALLFILE TABLESPACE "TBSSPC"
  2  DATAFILE 'tbsspc1.dbf' SIZE 50M
  3  AUTOEXTEND ON NEXT 10M MAXSIZE 200M
  4  LOGGING
  5  EXTENT MANAGEMENT LOCAL
  6  SEGMENT SPACE MANAGEMENT MANUAL;

Tablespace created.

SQL>
SQL> execute
dbms_workload_repository.modify_snapshot_settings(interval => 1440);

PL/SQL procedure successfully completed.

SQL>
SQL> exec
dbms_advisor.set_default_task_parameter('ADDM','DB_ACTIVITY_MIN',30)
;
```

```
PL/SQL procedure successfully completed.

SQL>
SQL> drop user spc cascade;

User dropped.

SQL>
SQL> create user spc identified by spc
  2 default tablespace tbsspc
  3 temporary tablespace temp;

User created.

SQL>
SQL> grant connect, resource, dba to spc;

Grant succeeded.

SQL>
SQL> connect spc/spc
Connected.
SQL>
SQL> drop table spct purge;
drop table spct purge
*
ERROR at line 1:
ORA-00942: table or view does not exist

SQL> create table spct(id number, name varchar2(2000));

Table created.

SQL>
SQL> exec DBMS_STATS.GATHER_TABLE_STATS(
  > ownname=>'SPC', tabname=>'SPCT', -
  > estimate_percent=>DBMS_STATS.AUTO_SAMPLE_SIZE);

PL/SQL procedure successfully completed.

SQL>
SQL> exit;
$
```

2. Take the first AWR snapshot. You can use either Enterprise Manager Cloud Control or the DBMS_WORKLOAD_REPOSITORY package. This document contains the steps for Enterprise Manager Cloud Control.

Step	Window/Page Description	Choices or Values
a.	orcl database home	From the menu, select Performance > AWR > AWR Administration .
b.	Database Login	Select SYSDBA Database Credentials from the Preferred Credential Name menu. Click Login .
c.	Automatic Workload Repository	Click the Snapshots link.
d.	Snapshots	Click Create .
e.	Confirmation	Click Yes .
f.	Processing	Wait until the snapshot is created.
g.	Snapshots	Record the new snapshot number

3. Execute the workgen script with parameters as shown to generate the workload for this practice. The script is located in your \$HOME/workshops directory. Wait until the “Load is finished” message appears, and then continue.

```
$ ./workgen 7 1
The Oracle base for
ORACLE_HOME=/u01/app/oracle/product/12.1.0/dbhome_1 is
/u01/app/oracle

PL/SQL procedure successfully completed
...
Load is finished
$
```

4. Take a second AWR snapshot. You can take an AWR snapshot by using the DBMS_WORKLOAD_REPOSITORY package, or you can use Enterprise Manager Cloud Control to take a snapshot by repeating the steps in step 2.

Record the snapshot number._____

5. Generate the AWR report. You can use the `awrrpt.sql` script located in the `$ORACLE_HOME/rdbms/admin` directory to create an AWR report or Enterprise Manager Cloud Control. The solution in this document uses Enterprise Manager Cloud Control.

Step	Page	Action
a	Snapshots	Select the first snapshot number from step 2. Select View Report from Actions. Click Go .
b	View Report	Select the second snapshot number from step 4. Click OK .
c	Processing:View Report	Wait until the report is created.
d	Snapshot Details	Go to Step 6.

6. Review the AWR Report. The Snap Id numbers in the example below will not match the snapshot number you used to produce the report.
- What is the DB time? What is the elapsed time? Why is there a difference?
DB Time includes the sum of all sessions in database calls, so DB Time could be as high as the product of the active sessions and the elapsed time.

WORKLOAD REPOSITORY report for						
DB Name	DB Id	Instance	Inst num	Startup Time	Release	RAC
ORCL	1346382964	orcl		1 01-Aug-13 07:08	12.1.0.1.0	NO
Host Name	Platform	CPU	Cores	Sockets	Memory (GB)	
EDRSR12P0	Linux x86 64-bit	2	2	1	7.65	
Snap Id		Snap Time	Sessions	Cursors/Session		
Begin Snap:	2240	01-Aug-13 07:58:14	40	2.1		
End Snap:	2241	01-Aug-13 08:02:57	40	1.8		
Elapsed:		4.71 (mins)				
DB Time:		30.33 (mins)				

- b. Scroll down to **Top 10 Foreground Events by Total Wait Time**. What is the top event? “*buffer busy waits*” is expected.

Top 10 Foreground Events by Total Wait Time

Event	Waits	Total Wait Time (sec)	Wait Avg(ms)	% DB time	Wait Class
buffer busy waits	102,191	1445.1	14	79.4	Concurrency
enq: HW - contention	878	227.2	259	12.5	Configuration
DB CPU		36.2		2.0	
free buffer waits	683	20.2	30	1.1	Configuration
db file sequential read	4,015	11.6	3	.6	User I/O
log file switch (checkpoint incomplete)	10	11.5	1147	.6	Configuration
log file sync	89	9	101	.5	Commit
library cache lock	139	6.9	50	.4	Concurrency
library cache: mutex X	102	1.9	19	.1	Concurrency
cursor: pin S	740	1.7	2	.1	Concurrency

- c. Scroll down to **Time Model Statistics (Main Report > Wait Event Statistics > Time Model Statistics)**. In which category is most of the DB time being collected? “*sql execute elapsed time*” is expected.

Time Model Statistics

- Total time in database user-calls (DB Time): 1820s
- Statistics including the word “background” measure background process time,
- Ordered by % or DB time desc, Statistic name

Statistic Name	Time (s)	% of DB Time
sql execute elapsed time	1,798.96	98.84
DB CPU	36.22	1.99
parse time elapsed	15.80	0.87
connection management call elapsed time	8.61	0.47
hard parse elapsed time	7.35	0.40
PL/SQL execution elapsed time	5.67	0.31
PL/SQL compilation elapsed time	0.17	0.01
hard parse (sharing criteria) elapsed time	0.04	0.00
repeated bind elapsed time	0.00	0.00
DB time	1,820.04	
background elapsed time	65.74	
background cpu time	2.42	

[Back to Wait Events Statistics](#)

[Back to Top](#)

- d. Scroll down to the **Buffer Wait Statistics** section ([Back to Top > Main Report > Wait Statistics > Buffer Wait Statistics](#)). What is the class of waits? “Waits on data blocks” is expected.

Buffer Wait Statistics

- ordered by wait time desc, waits desc

Class	Waits	Total Wait Time (s)	Avg Time (ms)
data block	134,966	1,444	11
segment header	7,873	2	0
undo header	95	0	0
file header block	2	0	0

[Back to Wait Statistics](#)

[Back to Top](#)

- e. Scroll down to the **Segments by Buffer Busy Waits** section ([Back to Top > Main Report > Segment Statistics > Segments by Buffer Busy Waits](#)). Which segment has the most buffer busy waits? *SPCT*

Segments by Buffer Busy Waits

- % of Capture shows % of Buffer Busy Waits for each top segment compared
- with total Buffer Busy Waits for all segments captured by the Snapshot

Owner	Tablespace Name	Object Name	Subobject Name	Obj. Type	Buffer Busy Waits	% of Capture
SPC	TBSSPC	SPCT		TABLE	142,842	100.00
SYS	SYSTEM	SCHEDULER\$_JOB		TABLE	1	0.00

[Back to Segment Statistics](#)

[Back to Top](#)

- f. At this point, you have determined that there are a large number of buffer busy waits on a single segment.

7. View the ADDM report that was generated when you created the snapshot in step 4.

Step	Window/Page Description	Choices or Values
a.	Snapshot Details	Expand the Performance menu and select Advisors Home .
b.	Advisors Home	Enter search values: Advisory Type: ADDM Advisor Runs: Last 24 Hours Click Go .

8. Find and select the ADDM run that covers the snapshots you recorded in steps 2 and 4 by checking the description field of each ADDM report. The description will be in the “ADDM auto run: snapshots [177, 178], instance 1, database id 1159023676” format. Substitute your start and end snapshot numbers. (The latest ADDM report should be the one that meets the criteria.) Click **View Result**.

Advisor Tasks

Search
Select an advisory type and optionally enter a task name to filter the data that is displayed in your results set.

Advisory Type	Task Name	Advisor Runs	Status
ADDM		Last 24 Hours	All

By default, the search returns all uppercase matches beginning with the string you entered. To run an exact or case-sensitive match, double quote the search string. You can use the wildcard symbol (%) in a double quoted string.

Results

Select	Name	Advisory Type	Description	User	Status	Start Time	Duration (seconds)	Expires In (days)	
<input checked="" type="radio"/>	ADDM:1346382964_1_2241	ADDM	ADDM auto run: snapshots [2240, 2241], instance 1, database id 1346382964	SYS	COMPLETED	Aug 1, 2013 8:03:00 AM		2	30
<input type="radio"/>	ADDM:1346382964_1_2239	ADDM	ADDM auto run: snapshots [2238, 2239], instance 1, database id 1346382964	SYS	COMPLETED	Jul 31, 2013 11:00:36 AM		1	29
<input type="radio"/>	ADDM:1346382964_1_2238	ADDM	ADDM auto run: snapshots [2237, 2238], instance 1, database id 1346382964	SYS	COMPLETED	Jul 31, 2013 10:00:31 AM		0	29
<input type="radio"/>	ADDM:1346382964_1_2237	ADDM	ADDM auto run: snapshots [2236, 2237], instance 1, database id 1346382964	SYS	COMPLETED	Jul 31, 2013 9:00:26 AM		1	29

- a. On the Automatic Database Diagnostic Monitor (ADDM) page, click the “**Buffer Busy**” finding. If more than one finding with this name appears on the page, check all of them.

ADDM Performance Analysis

Task Name ADDM:1346382964_1_2241

Filters

Task Owner	SYS	Average Active Sessions	6.4	Period Start Time	Aug 1, 2013 7:58:14 AM
Impact (%)	97	Finding	Top SQL Statements	Occurrences (24 hrs ending with a	
	79.4		Buffer Busy - Hot Objects	4 of 4	
	33.5		Buffer Busy - Hot Block	1 of 4	
	22.6		Buffer Busy - Hot Block	2 of 4	
	12.5		High Watermark Waits	2 of 4	

- b. On the Performance Finding Details page, you should see that ADDM recommends reorganizing the SPCT table to use Automatic Segment Space Management. Although you are able to infer this by looking at either a Statspack report or an AWR report, ADDM automatically tells you what to do.

Note: This recommendation may not appear. ADDM considers many factors before it gives a recommendation. A delay in creating the snapshot after the workload finishes is one reason that the recommendation may not appear.

The screenshot shows the Oracle Advisor Central interface. The top navigation bar includes 'Advisor Central > Automatic Database Diagnostic Monitor (ADDM):SYS.ADDM:1346382964_1_2241 > Performance Finding Details'. The right side of the header says 'Logged in as SYS'. Below the header, the title is 'Performance Finding Details: Buffer Busy - Hot Objects'. A summary section shows a finding: 'Read and write contention on database blocks was consuming significant database time.' with an 'Impact (Active Sessions)' of 5.11 and a 'Percentage of Finding's Impact (%)' of 79.4. The period is from Aug 1, 2013 7:58:14 AM to Aug 1, 2013 8:02:57 AM. There are 'Filtered' and 'No Filters' buttons. A 'Recommendations' section follows, with a table comparing 'Show All Details' and 'Hide All Details' options. It lists a schema recommendation with a benefit of 79.4% and an action: 'Consider using ORACLE's recommended solution of automatic segment space management in a locally managed tablespace for the tablespace "TBSSPC" containing the TABLE "SPC.SPCT" with object ID 95240. Alternatively, you can move this object to a different tablespace that is locally managed with automatic segment space management. Database Object SPC.SPCT'. A rationale section states there was significant contention on the table SPC.SPCT. Finally, two 'Show Schema' buttons are shown, each with a benefit of 79.4%.

9. Implement the recommendation.

- a. To implement the suggested recommendation, execute the lab_07_01_09.sql script. This script moves the table used in the workload to a tablespace that uses Automatic Segment Space Management. You can locate the lab_07_01_09.sql script in your \$HOME/labs directory.

```
$ cd $HOME/labs
$ sqlplus / as sysdba
SQL> @lab_07_01_09.sql
SQL>
SQL> drop tablespace tbsspc
  2 including contents and datafiles;

Tablespace dropped.

SQL>
SQL> CREATE SMALLFILE TABLESPACE "TBSSPC"
  2 DATAFILE 'tbsspc1.dbf' SIZE 50M
  3 AUTOEXTEND ON NEXT 10M MAXSIZE 200M
  4 LOGGING
  5 EXTENT MANAGEMENT LOCAL
  6 SEGMENT SPACE MANAGEMENT AUTO;

Tablespace created.

SQL>
SQL> connect spc/spc
Connected.
SQL>
SQL> drop table spct purge;
drop table spct purge
```

```

*
ERROR at line 1:
ORA-00942: table or view does not exist

SQL> create table spct(id number, name varchar2(2000))
  2 tablespace tbsspc;

Table created.

SQL>
SQL> exec DBMS_STATS.GATHER_TABLE_STATS(
-> ownname=>'SPC', tabname=>'SPCT',-
-> estimate_percent=>DBMS_STATS.AUTO_SAMPLE_SIZE);

PL/SQL procedure successfully completed.

SQL>
SQL> exit;
Disconnected from Oracle Database 12c Enterprise Edition Release
12.1.0.1.0 - 64bit Production
With the Partitioning, OLAP, Advanced Analytics and Real Application
Testing options
$
```

10. When the script finishes, take a third AWR snapshot. **Record the snapshot number.**
11. Generate the workload again to observe the differences that the recommendation made. Execute the **workgen** script to start generating the workload for this practice again.

```

$ cd $HOME/workshops
$ ./workgen 7 1
The Oracle base for
ORACLE_HOME=/u01/app/oracle/product/12.1.0/dbhome_1 is
/u01/app/oracle

PL/SQL procedure successfully completed

PL/SQL procedure successfully completed

PL/SQL procedure successfully completed
...
Load is finished
$
```

12. When the script finishes, take a fourth AWR snapshot. **Record the snapshot number.**

13. Generate a Compare Periods Report comparing the periods between the two periods (four snapshots). You can use the `awrddrpt.sql` script located in your `$ORACLE_HOME/rdbms/admin` directory to generate the diff-diff report. The solution in this document uses Enterprise Manager Cloud Control.

Step	Window/Page Description	Choices or Values
a.	Snapshots	Select the first snapshot in this practice recorded in Step 2g. Select Compare Periods from the Actions drop-down list. Click Go .
b.	Compare Periods: First Period End	Select the second snapshot in this practice recorded in Step 4. Click Next .
c.	Compare Periods: Second Period Start	Select the third AWR snapshot (the snapshot number recorded in step 10). Click Next .
d.	Compare Periods: Second Period End	Select the last AWR snapshot (the snapshot number recorded in step 12). Click Next .
e.	Compare Periods: Review	Click Finish .

Compare Periods: Review

[Cancel](#) [Back](#) Step 5 of 5 [Finish](#)

Database orcl

First Period		Second Period	
Beginning Snapshot ID	143	Beginning Snapshot Capture Time	Dec 13, 2013 10:42:12 AM
Ending Snapshot ID	144	Ending Snapshot Capture Time	Dec 13, 2013 10:45:44 AM
Second Period		First Period	
Beginning Snapshot ID	145	Beginning Snapshot Capture Time	Dec 13, 2013 11:32:45 AM
Ending Snapshot ID	146	Ending Snapshot Capture Time	Dec 13, 2013 11:37:27 AM

14. Review the Compare Periods Report.
- On the Compare Periods: Results page, select **Per Transaction** from the View Data drop-down list on the General tabbed page.
 - You can see the performance differences between the two analyzed periods. Click the **Report** tab.
 - The Processing page appears. Wait until the report is generated.
 - On the Report tabbed page, you can now see the Workload Repository Compare Period Report.

- e. Scroll down to the **Top Timed Events** section. What is the %DB time for buffer busy waits in the first period? What is the %DB time for buffer busy waits in the second period? *It is expected that the %DB Time will be much smaller in the second period.*

Top Timed Events

- Events with a "-" did not make the Top list in this set of snapshots, but are displayed for comparison purposes

1st						2nd					
Event	Wait Class	Waits	Time(s)	Avg Time(ms)	%DB time	Event	Wait Class	Waits	Time(s)	Avg Time(ms)	%DB time
buffer busy waits	Concurrency	102,191	1,445.05	14.14	79.40	buffer busy waits	Concurrency	7,587	945.46	124.62	33.42
eng: HW - contention	Configuration	878	227.16	258.72	12.48	free buffer waits	Configuration	5,830	872.16	149.60	30.83
db file async I/O submit	System I/O	44	44.25	1,005.65	2.43	CPU time			277.63		9.81
CPU time			36.22		1.99	log file switch (checkpoint incomplete)	Configuration	90	159.27	1,769.63	5.63
log file parallel write	System I/O	368	22.65	61.55	1.24	write complete waits	Configuration	64	119.12	1,861.26	4.21
free buffer waits	Configuration	686	20.28	29.56	1.11	latch: redo copy	Configuration	3,820	70.41	18.43	2.49
log file switch (checkpoint incomplete)	Configuration	12	13.17	1,097.10	0.72	db file async I/O submit	System I/O	359	68.88	191.86	2.43
db file sequential read	User I/O	5,348	13.07	2.44	0.72	cursor: pin S	Concurrency	4,001	48.65	12.16	1.72
log file sync	Commit	90	9.14	101.57	0.50	log file parallel write	System I/O	493	22.87	46.38	0.81
library cache lock	Concurrency	139	6.95	49.99	0.38	db file scattered read	User I/O	1,212,320	22.54	0.02	0.80
-cursor: pin S	Concurrency	740	1.74	2.36	0.10	-log file sync	Commit	129	16.75	129.85	0.59
-db file scattered read	User I/O	291	0.11	0.38	0.01	-db file sequential read	User I/O	172,097	10.39	0.06	0.37
-						-library cache lock	Concurrency	67	1.13	16.92	0.04
-						-eng: HW - contention	Configuration	55	0.15	2.70	0.01

- f. Review the **Buffer Wait Statistics** report (**Report Details > Wait Stats > Buffer Wait Statistics**).

What is the wait time % of DB time for data blocks for the first period? _____

What is the wait time % of DB time for data blocks for the second period?

Buffer Wait Statistics

- Ordered by absolute value of 'Diff' column of 'Wait Time % of DB time' descending

Class	Wait Time % of DB time			Total Wait Time (s)		# Waits		Avg Wait Time (ms)		
	1st	2nd	Diff	1st	2nd	1st	2nd	1st	2nd	%Diff
data block	0.79	0.33	-0.47	1,444.02	920.06	134,966	13,753	0.11	0.67	509.09
1st level bmb	0.00	0.01	0.01	0.00	16.56	0	208	0.00	0.80	100.00
file header block	0.00	0.00	0.00	0.00	4.54	2	8	0.00	5.68	100.00
undo header	0.00	0.00	0.00	0.02	2.67	95	27	0.00	0.99	100.00
segment header	0.00	0.00	-0.00	1.76	1.21	7,873	146	0.00	0.08	100.00

[Back to Wait Stats](#)

[Back to Top](#)

- g. Review the **Top Segments by Buffer Busy Waits** section (**Back to Top > Segment Statistics > Top Segments by Buffer Busy Waits**). What is the number of buffer busy waits in the first period? What is the number of buffer busy waits in the second period?

Top Segments by Buffer Busy Waits

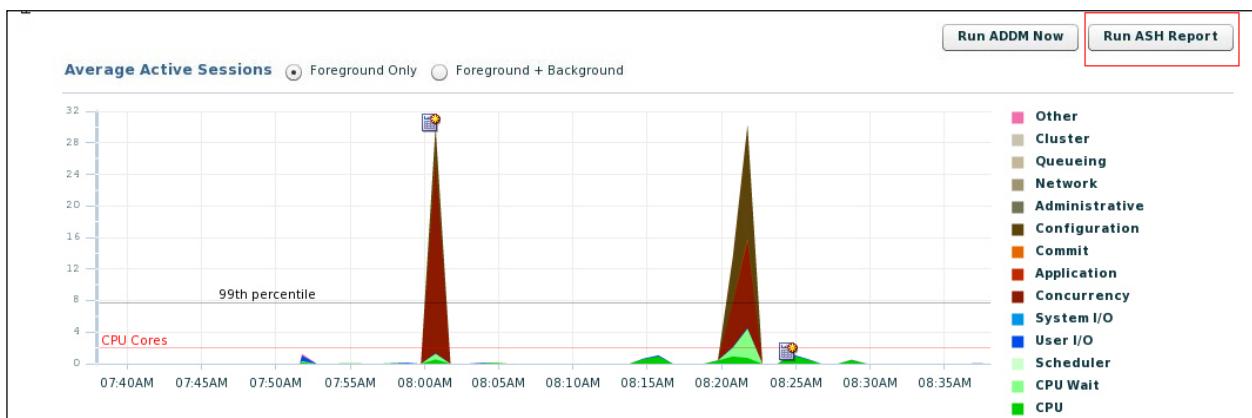
- Ordered by absolute value of 'Diff' column or % of Total Buffer Busy Waits'
- Top 5 Segments from each period are compared
- Total Buffer Busy Waits First: 102,191, Second: 7,587
- Buffer Busy Wait Time First: 1,445.05 seconds, Second: 945.46 seconds
- Buffer Busy Wait Time as % of DB time First: 79.4%, Second: 33.42%
- Captured Buffer Busy Waits First: 142,843, Second: 14,100

Owner	Tablespace	Object Name	Subobject Name	Type	% of Total Buffer Busy Waits				Buffer Busy Waits			% of Captured Buffer Busy Waits		
					1st	1st Total	2nd	2nd Total	Diff	1st	2nd	%Diff	1st	2nd
SPC	TBSSPC	SPCT		TABLE		185.84	185.84	185.84		14,100			100.00	100.00
** MISSING **	TBSSPC	** MISSING: 95240/95240 **	** MISSING **	UNDEFINED	139.78	139.78	0.00	185.84 - 139.78	142,842	0 - 100.00	100.00	0.00	-100.00	
SYS	SYSTEM	SCHEDULER\$_JOB		TABLE	0.00	139.78			-0.00	1		0.00	-0.00	

[Back to Segment Statistics](#)
[Back to Top](#)

15. Generate the ASH Report from the Top Activity page for the period with the greatest load. You can use the `ashrpt.sql` script located in the `$ORACLE_HOME/rdbms/admin` directory to generate the corresponding ASH Report. The solution in this document uses Enterprise Manager Cloud Control.

Step	Page	Action
a	Snapshot Details	Expand the Performance menu and select Performance Home .
b	Performance	Note the time of the first peak in the Active Sessions Graph. Pick a 10-minute interval that brackets the peak.
		Click Run ASH Report .
c	Run ASH Report	Enter the Start time and End time from the times that you noted in the previous step. Click Generate Report .
d	Processing	Wait until the report is generated.



16. View the ASH report.

- a. In the first section, you can see general information.

ASH Report For ORCL/orcl								
DB Name	DB Id	Instance	Inst num	Release	RAC	Host		
ORCL	1346382964	orcl	1	12.1.0.1.0	NO	EDRSR12P0		
CPUs	SGA Size	Buffer Cache		Shared Pool	ASH Buffer Size			
2	478M (100%)	24M (5.0%)		280M (58.6%)	4.0M (0.8%)			
		Sample Time		Data Source				
Analysis Begin Time:		01-Aug-13 08:15:58		V\$ACTIVE_SESSION_HISTORY				
Analysis End Time:		01-Aug-13 08:25:58		V\$ACTIVE_SESSION_HISTORY				
Elapsed Time:		10.0 (mins)						
Sample Count:		2,899						
Average Active Sessions:		4.83						
Avg. Active Session per CPU:		2.42						
Report Target:		None specified						

- b. Scroll down to the **Top User Events** section. What is the top event? *buffer busy waits is in the top events.*

Top User Events				
Event	Event Class	% Event	Avg Active Sessions	
buffer busy waits	Concurrency	34.29	1.66	
free buffer waits	Configuration	30.77	1.49	
CPU + Wait for CPU	CPU	16.97	0.82	
log file switch (checkpoint incomplete)	Configuration	5.73	0.28	
write complete waits	Configuration	4.24	0.21	

[Back to Top Events](#)
[Back to Top](#)

- c. Scroll down to the **Top SQL with Top Events** section. What is the type of the top SQL statement? *INSERT*

Top SQL with Top Events								
SQL ID	Planhash	Sampled # of Executions	% Activity	Event	% Event	Top Row Source	% Rwsr	SQL Text
3csh3g3mjhzh		1355	89.46	buffer busy waits	30.63	** Row Source Not Available **	30.63	INSERT INTO SPCT VALUES (NULL,...)
		1355	89.46	free buffer waits	29.74	** Row Source Not Available **	29.74	INSERT INTO SPCT VALUES (NULL,...)
		1355	89.46	CPU + Wait for CPU	9.60	** Row Source Not Available **	9.60	INSERT INTO SPCT VALUES (NULL,...)
fcu83t10rr413		36	2.95	CPU + Wait for CPU	2.95	** Row Source Not Available **	2.95	declare t number; begin for t ...

- d. What is the top SQL statement and what was the primary wait event for that statement? _____
 What is the SQL ID? _____
- e. Scroll down to the **Activity Over Time** section. How many periods are reported? What is the elapsed time of each period? Which periods show buffer busy waits? *Possibly none if buffer busy waits are not the top event.*
 These answers may vary greatly.

Activity Over Time

- Analysis period is divided into smaller time slots
- Top 3 events are reported in each of those slots
- 'Slot Count' shows the number of ASH samples in that slot
- 'Event Count' shows the number of ASH samples waiting for that event in that slot
- '% Event' is 'Event Count' over all ASH samples in the analysis period

Slot Time (Duration)	Slot Count	Event	Event Count	% Event
08:16:00 (1.0 min)	2	control file parallel write	1	0.03
		db file sequential read	1	0.03
08:18:00 (1.0 min)	2	CPU + Wait for CPU	1	0.03
		control file parallel write	1	0.03
08:19:00 (1.0 min)	52	CPU + Wait for CPU	40	1.38
		db file async I/O submit	3	0.10
		log file parallel write	3	0.10
08:20:00 (1.0 min)	1,554	buffer busy waits	557	19.21
		free buffer waits	552	19.04
		CPU + Wait for CPU	255	8.80
08:21:00 (1.0 min)	1,162	buffer busy waits	440	15.18
		free buffer waits	352	12.14
		log file switch (checkpoint incomplete)	123	4.24
08:22:00 (1.0 min)	1	CPU + Wait for CPU	1	0.03
08:23:00 (1.0 min)	21	CPU + Wait for CPU	20	0.69
		control file parallel write	1	0.03
08:24:00 (1.0 min)	87	CPU + Wait for CPU	60	2.07
		db file async I/O submit	8	0.28
		db file sequential read	5	0.17
08:25:00 (58 secs)	18	CPU + Wait for CPU	16	0.55
		db file scattered read	2	0.07

- f. Log out of Enterprise Manager Cloud Control.

17. Clean up from this practice by running the `cleanup` script.

```
$ ./cleanup 7 1
The Oracle base for
ORACLE_HOME=/u01/app/oracle/product/12.1.0/dbhome_1 is
/u01/app/oracle

Tablespace dropped.

User dropped.

PL/SQL procedure successfully completed.

$
```


Practices for Lesson 8: Real-Time Database Operation Monitoring

Chapter 8

Practices for Lesson 8: Overview

Practices Overview

In these practices, you will monitor a composite database operation and a PL/SQL operation.

Practice 8-1: Monitoring a Composite Database Operation

Overview

In this practice, you will create a composite database operation that will always be monitored.

Assumptions

The terminal session is opened as the oracle OS user, and the environment variables for the orcl database instance are set.

Tasks

1. Navigate to the SQL Monitoring Page in Enterprise Manager Cloud Control.

Step	Window/Page Description	Choices or Values
a.	EMCC Login	Log in as the sysman user.
b.		Navigate to the orcl database home page.
c.	orcl home page	Select Performance > SQL Monitoring .
d.	Database Login	Select SYSDBA Database Credentials for the Preferred Credential Name. Click Login .
e.	Monitored SQL Executions	Select 30 Seconds in the Auto Refresh menu.

2. In a terminal window, navigate to the \$HOME/labs/RT_DBOP directory.
3. Review the dbOP_cust.sql script. Identify the name of the database operation. What commands are used to assign an operation name to the set of SQL commands?

```
$ less dbOP_cust.sql
variable eid Number;

exec :eid := dbms_sql_monitor.begin_operation(dbop_name =>
'DBOP_CUST',FORCED_TRACKING => 'Y');

declare

v1 varchar2(100);
v2 varchar2(100);
v3 number;

cursor c1 is
select cust_city from (select count(*) cnt, cust_city from
sh.customers
group by cust_city
order by 1 desc) where rownum < 21;
```

```
begin

for i in c1
loop

v3 := 0;

Select max(amount_sold)
into v3
from sh.sales
where cust_id IN (select cust_id from sh.customers where
cust_city=i.cust_city);

dbms_output.put_line('Amount: '||v3);

end loop;

end;
/

Select max(asld) from (Select max(amount_sold) asld,cust_id
from sh.sales
where cust_id IN (select cust_id from sh.customers
where cust_city IN ( select cust_city from
(select /*+ NO_MERGE */ count(*)
cnt, cust_city from sh.customers
group by cust_city having count(*) > 1)
))
group by cust_id)
/

exec dbms_sql_monitor.end_operation('DBOP_CUST', :eid);
$
```

4. Log in to SQL*Plus as the **sh** user with the password **sh**.

```
$ sqlplus sh

SQL*Plus: Release 12.1.0.1.0 Production on Wed Aug 21 12:45:37
2013

Copyright (c) 1982, 2013, Oracle. All rights reserved.

Enter password: sh /* password will not be displayed */
Last Successful login time: Wed Aug 21 2013 12:45:30 +00:00

Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.1.0 -
64bit Production
With the Partitioning, OLAP, Advanced Analytics and Real
Application Testing options

SQL>
```

5. Execute the **dbOP_cust.sql** script.

```
SQL> set serveroutput on
SQL> @dbOP_cust

PL/SQL procedure successfully completed.

Amount: 1782.72
Amount: 1765.14
Amount: 1782.72
Amount: 1764.26
Amount: 1282.7
Amount: 1764.26
Amount: 1782.72
Amount: 1697.83
Amount: 1472.05
Amount: 1764.26
Amount: 1758.11
Amount: 1782.72
Amount: 1764.26
Amount: 1674.39
Amount: 1764.26
Amount: 1764.26
Amount: 1764.26
Amount: 1782.72
```

```

Amount: 1782.72
Amount: 1782.72

PL/SQL procedure successfully completed.

MAX (ASLD)
-----
1782.72

PL/SQL procedure successfully completed.

SQL>

```

6. Return to Enterprise Manager Cloud Control to observe the monitored database operation.

Monitored SQL Executions									
Top 100 By Last Active Time		Type All	Execution Detail		SQL Detail		Session Detail		
Status	Duration	Type	ID	SQL Plan Hash	User	Parallel	Database Time	IO Requests	Start
<input checked="" type="checkbox"/>	5.0s		DBOP_CUST		SH			8,884	12:49:24 PM
<input checked="" type="checkbox"/>	4.0s		gs9bkf7j1mjv		SH			8,398	12:49:24 PM
<input checked="" type="checkbox"/>	2.1m		6fw4jyxz4y45b	343110142	DBSNMP			706K	12:35:42 PM
<input checked="" type="checkbox"/>	2.1m		c70zuQfg2w2j	169637089	DBSNMP			705K	12:30:27 PM
<input checked="" type="checkbox"/>	2.0m		74k6v71dkh1cj	343110142	DBSNMP			705K	12:26:15 PM
<input checked="" type="checkbox"/>	2.1m		6fw4jyxz4y45b	343110142	DBSNMP			706K	11:35:42 AM

7. What type of operation is DBOP_CUST?

Monitored SQL Executions						
Top 100 By Last Active Time		Type All	Execution Detail		SQL Detail	
Status	Duration	Type	ID	SQL Plan Hash	User	
<input checked="" type="checkbox"/>	5.0s		DBOP_CUST		SH	
<input checked="" type="checkbox"/>	4.0s		gs9bkf7j1mjv		SH	
<input checked="" type="checkbox"/>	2.1m		6fw4jyxz4y45b	343110142	DBSNMP	

8. Click DBOP_CUST id to view Execution Detail.

9. Review the execution details in the Overview section.

Monitored SQL Executions >
Monitored SQL Execution Details: DBOP_CUST

Logged in as SYS

Save Page Refreshed 1:47:40 PM GMT+00

Overview

General

- Execution Started: Wed Aug 21, 2013 12:49:24 PM
- Last Refresh Time: Wed Aug 21, 2013 12:49:29 PM
- Execution ID: 1
- User: SH

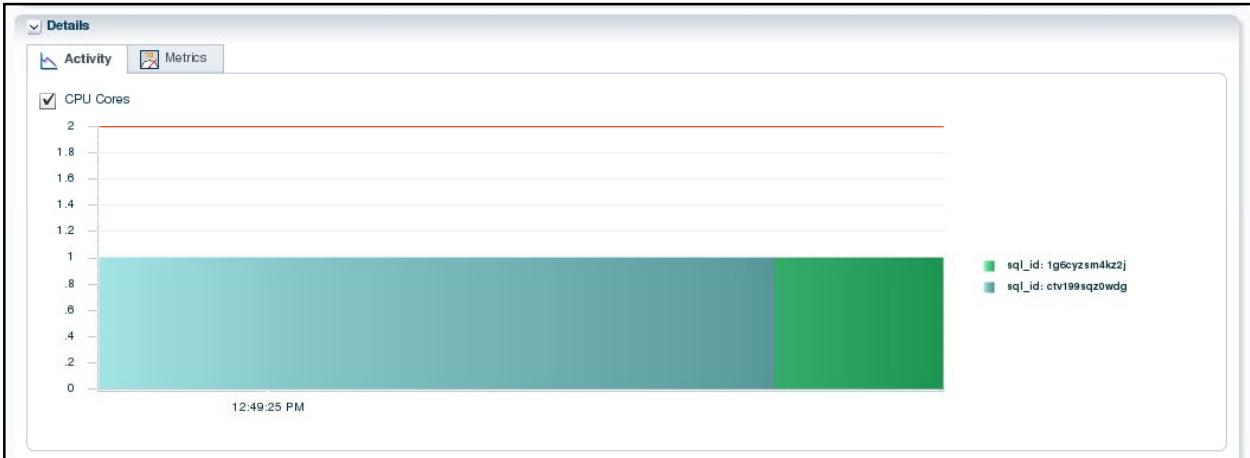
Time & Wait Statistics

Duration	5.0s
Database Time	4.8s
PL/SQL & Java	0us
Wait Activity %	100

IO Statistics

Buffer Gets	1,868
IO Requests	8,884
IO Bytes	524MB

10. Scroll down and review the Activity details.



11. Click the Metrics tab to view the metrics collected on this operation.



12. Return to the SQL Monitoring page.
13. Return to the terminal window and log out of SQL*Plus.

```
SQL> exit
Disconnected from Oracle Database 12c Enterprise Edition Release
12.1.0.1.0 - 64bit Production
With the Partitioning, OLAP, Advanced Analytics and Real
Application Testing options
$
```

Practice 8-2: Monitoring a PL/SQL Operation

Overview

In this practice, you will create a PL/SQL operation that is always monitored.

Assumptions

The terminal session is opened as the oracle OS user, and the environment variables for the orcl database instance are set.

Tasks

- Review the dbOP_dbaobj.sql script. Identify the name of the database operation. What commands are used to provide the name?

```
$ cd $HOME/labs/RT_DBOP
$ less dbOP_dbaobj.sql

DECLARE

DBOP_ID NUMBER;

cursor dba_obj is
  select * from dba_objects;
dummy NUMBER(10);
junk NUMBER(10);

my_cur  DBA_OBJECTS%ROWTYPE;

begin
  dummy := 0;
  junk := 0;

  DBOP_ID := DBMS_SQL_MONITOR.begin_operation(DBOP_NAME =>
'DBA_OBJECTS', FORCED_TRACKING => 'Y');

  for my_cur in dba_obj loop

    if my_cur.owner = 'SOE' THEN
      dummy := dummy + 1;
    ELSE
      junk := junk + 1;
    END IF;

  END LOOP;
```

```

DBMS_OUTPUT.PUT_LINE('SOE OBJECTS: '||to_char(dummy));

DBMS_OUTPUT.PUT_LINE('OTHER OBJECTS: '||to_char(junk));

DBMS_SQL_MONITOR.END_OPERATION('DBA_OBJECTS',DBOP_ID);

END;
/
$
```

2. In the terminal window, log in to SQL*Plus as the **SYSDBA** user.

```

$ sqlplus / as sysdba

SQL*Plus: Release 12.1.0.1.0 Production on Wed Aug 21 13:25:55
2013

Copyright (c) 1982, 2013, Oracle. All rights reserved.

Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.1.0 -
64bit Production
With the Partitioning, OLAP, Advanced Analytics and Real
Application Testing options

SQL>
```

3. Execute the **dbOP_dbaobj.sql** script.

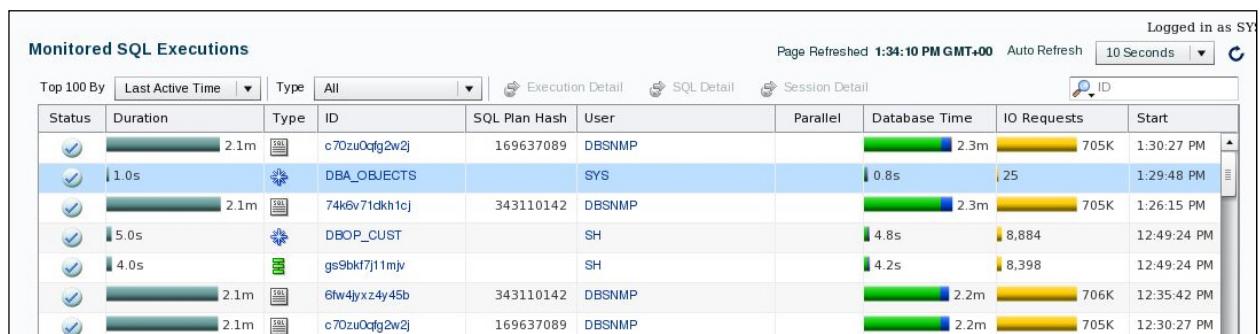
```

SQL> @dbOP_dbaobj.sql

PL/SQL procedure successfully completed.

SQL>
```

4. Return to Enterprise Manager Cloud Control to observe the monitored operation. Navigate to the Monitored SQL Executions page.



5. What type of operation is **DBA_OBJECTS**?

Status	Duration	Type	ID	SQL Plan Hash	User
	2.1m		c70zu0cfg2w2j	169637089	DBSNMP
	1.0s		DBA_OBJECTS		SYS
	2.1m		Composite Database Operation	343110142	DBSNMP
	5.0s		DBOP_CUST		SH
	4.0s		gs9bkf7j11mjv		SH

6. Click the **DBA_OBJECTS** ID to view **Execution Detail**.

7. Review the detailed information in the Overview section.

Monitored SQL Execution Details: DBA_OBJECTS
Save
Page Refreshed 1:40:14 PM GMT+00

General

Execution Started: Wed Aug 21, 2013 1:29:48 PM
Last Refresh Time: Wed Aug 21, 2013 1:29:49 PM
Execution ID: 2
User: SYS

Time & Wait Statistics

Duration	1.0s
Database Time	0.8s
PL/SQL & Java	0us
Wait Activity %	100

IO Statistics

Buffer Gets	39
IO Requests	25
IO Bytes	312KB

8. Scroll down to review the Activity Details.



9. Log out of Enterprise Manager Cloud Control.

10. **Exit SQL*Plus**.

Practices for Lesson 9: Monitoring Applications

Chapter 9

Practices for Lesson 9: Overview

Practices Overview

In this practice, you monitor applications with services.

Practice 9-1: Using Services in a Single-Instance Oracle Database

The first step in creating a service configuration is to plan services for each application or application function that uses your database.

In this practice, you create the following configuration in the `orcl` database:

Service Name: SERV1

Usage: Client service

Response Time (sec)–Warning/Critical: 0.4, 1.0

Assumptions: You are using a terminal session that belongs to the `oracle` OS user, and the environment variables are set for the `orcl` database instance.

1. Create a new service named `SERV1`. Add the `SERV1` service name to your `tnsnames.ora` file. Use the `DBMS_SERVICE` package to create the service.
 - a. The recommended method of adding a service name to the `tnsnames.ora` file is to use Net Manager. For this exercise, execute the `lab_09_01_01.sh` script. This script finds the name of your host, and creates a `tnsnames` entry.

```
$ cd /home/oracle/labs
$ ./lab_09_01_01.sh
EDRSRnnnn (output is the name of your host)
$
```

- b. Review the `tnsnames.ora` file in the `$ORACLE_HOME/network/admin` directory to verify that the entry for the `SERV1` service name has been added to the `tnsnames.ora` file.

```
$ cd $ORACLE_HOME/network/admin
$ cat tnsnames.ora
# tnsnames.ora Network Configuration File:
/u01/app/oracle/product/12.1.0/dbhome
_1/network/admin/tnsnames.ora
# Generated by Oracle configuration tools.

EMREP =
(DESCRIPTION =
  (ADDRESS = (PROTOCOL = TCP)(HOST = <hostname>)(PORT = 1521))
  (CONNECT_DATA =
    (SERVER = DEDICATED)
    (SERVICE_NAME = emrep)
  )
)

ORCL =
(DESCRIPTION =
  (ADDRESS = (PROTOCOL = TCP)(HOST = <hostname>)(PORT = 1521))
  (CONNECT_DATA =
    (SERVER = DEDICATED)
```

```

        (SERVICE_NAME = orcl)
    )
)

SERV1 =
(DESCRIPTION =
  (ADDRESS = (PROTOCOL = TCP)(HOST = <hostname>)(PORT = 1521))
  (CONNECT_DATA =
    (SERVER = DEDICATED)
    (SERVICE_NAME = SERV1.us.oracle.com)
  )
)
$
```

- c. Log in to SQL*Plus and use the DBMS_SERVICE.CREATE_SERVICE procedure to create a service.

```

$ sqlplus / as sysdba

SQL> EXEC
DBMS_SERVICE.CREATE_SERVICE('SERV1','SERV1.us.oracle.com')

PL/SQL procedure successfully completed.

SQL>
```

2. After you have created your services, try connecting to your database by using your service name.

- a. What happens and why? You cannot connect by using your service because, although it is defined, it is not started on your instance.

```

SQL> connect system@SERV1
Enter password: oracle_4U /* password is not displayed */
ERROR:
ORA-12514: TNS:listener does not currently know of service
requested in connect descriptor

Warning: You are no longer connected to ORACLE.
```

- b. You can verify this by looking at the SERVICE_NAMES initialization parameter, and by looking at the services known to the listener.

```

SQL> connect / as sysdba
Connected.
SQL> show parameter service

NAME                                     TYPE        VALUE
-----                                     -----
service_names                           string      orcl.oracle.com
```

```

SQL> exit

$ lsnrctl services
LSNRCTL for Linux: Version 12.1.0.1.0 - Production on 10-JUL-2013
07:27:26

Copyright (c) 1991, 2013, Oracle. All rights reserved.

Connecting to
(DESCRIPTION=(ADDRESS=(PROTOCOL=IPC) (KEY=EXTPROC1521)))
Services Summary...
Service "emrep" has 1 instance(s).
  Instance "emrep", status READY, has 1 handler(s) for this
service...
    Handler(s):
      "DEDICATED" established:25033 refused:0 state:ready
        LOCAL SERVER
Service "emrepXDB" has 1 instance(s).
  Instance "emrep", status READY, has 1 handler(s) for this
service...
    Handler(s):
      "D000" established:0 refused:0 current:0 max:1022 state:ready
        DISPATCHER <machine: EDRSR12P0, pid: 2098>

(ADDRESS=(PROTOCOL=tcp) (HOST=edRSr12p0.us.oracle.com) (PORT=43054))
Service "orcl" has 1 instance(s).
  Instance "orcl", status READY, has 1 handler(s) for this
service...
    Handler(s):
      "DEDICATED" established:11623 refused:0 state:ready
        LOCAL SERVER
Service "orcl.oracle.com" has 1 instance(s).
  Instance "orcl", status READY, has 1 handler(s) for this
service...
    Handler(s):
      "DEDICATED" established:11623 refused:0 state:ready
        LOCAL SERVER
Service "orclXDB" has 1 instance(s).
  Instance "orcl", status READY, has 1 handler(s) for this
service...
    Handler(s):
      "D000" established:0 refused:0 current:0 max:1022 state:ready
        DISPATCHER <machine: EDRSR12P0, pid: 20747>

(ADDRESS=(PROTOCOL=tcp) (HOST=edRSr12p0.us.oracle.com) (PORT=10279))
The command completed successfully
$
```

3. How would you make sure that you can connect using your service? Attempt your solution, and connect to your instance using your service.

- a. *You must start the SERV1 service on your instance.*

Note: If the service does not appear in the lsnrctl services listing immediately,

run the ALTER SYSTEM REGISTER command, and then retry the connection with the connect system/oracle@SERV1 command.

Note: The service name is not added to the service_names initialization parameter.

```
$ sqlplus / as sysdba

SQL> EXEC DBMS_SERVICE.START_SERVICE('SERV1');

PL/SQL procedure successfully completed.

SQL> show parameter service

NAME                                TYPE        VALUE
-----
service_names                        string      orcl.oracle.com
SQL> exit
```

- b. Verify that the service has started by executing the lsnrctl services command.

```
$ lsnrctl services
LSNRCTL for Linux: Version 12.1.0.1.0 - Production on 10-JUL-2013
07:41:04

Copyright (c) 1991, 2013, Oracle. All rights reserved.

Connecting to
(DESCRIPTION=(ADDRESS=(PROTOCOL=IPC)(KEY=EXTPROC1521)))
Services Summary...
Service "SERV1.us.oracle.com" has 1 instance(s).
  Instance "orcl", status READY, has 1 handler(s) for this
service...
    Handler(s):
      "DEDICATED" established:1 refused:0 state:ready
      LOCAL SERVER
Service "emrep" has 1 instance(s).
  Instance "emrep", status READY, has 1 handler(s) for this
service...
    Handler(s):
      "DEDICATED" established:25042 refused:0 state:ready
      LOCAL SERVER
Service "emrepXDB" has 1 instance(s).
  Instance "emrep", status READY, has 1 handler(s) for this
service...
    Handler(s):
      "D000" established:0 refused:0 current:0 max:1022 state:ready
      DISPATCHER <machine: EDRSR12P0, pid: 2098>

(ADDRESS=(PROTOCOL=tcp)(HOST=edRSR12p0.us.oracle.com)(PORT=43054))
Service "orcl" has 1 instance(s).
```

```
Instance "orcl", status READY, has 1 handler(s) for this
service...
Handler(s):
  "DEDICATED" established:1 refused:0 state:ready
    LOCAL SERVER
Service "orcl.oracle.com" has 1 instance(s).
  Instance "orcl", status READY, has 1 handler(s) for this
service...
  Handler(s):
    "DEDICATED" established:1 refused:0 state:ready
      LOCAL SERVER
Service "orclXDB" has 1 instance(s).
  Instance "orcl", status READY, has 1 handler(s) for this
service...
  Handler(s):
    "D000" established:0 refused:0 current:0 max:1022 state:ready
      DISPATCHER <machine: EDRSR12P0, pid: 20747>

(ADDRESS=(PROTOCOL=tcp) (HOST=edRSr12p0.us.oracle.com) (PORT=10279))
The command completed successfully
$
```

- c. Connect to the database instance by using the SERV1 service.

```
$ sqlplus /nolog
SQL> connect SYSTEM@SERV1
Enter password: oracle_4U /* Password is not displayed */
Connected.
SQL> exit
$
```

4. Change to the /home/oracle/workshops directory and execute the workgen script with parameters as shown. This workload will run until it is stopped by the removal of the runload file in a later step; therefore, continue to the next step. This script starts three workloads, each connecting with a different service name: SERV1, ORCL, and SYS\$.

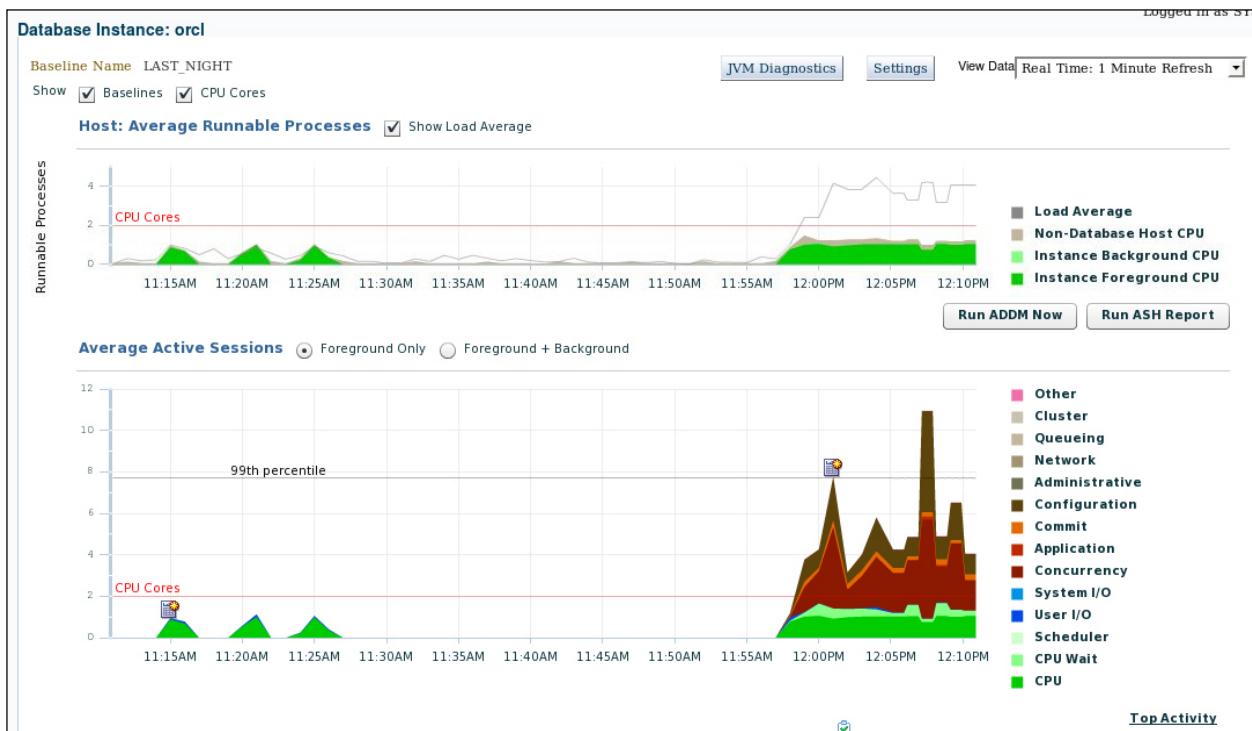
```
$ cd /home/oracle/workshops
$ ./workgen SERV

The Oracle base for
ORACLE_HOME=/u01/app/oracle/product/12.1.0/dbhome_1 is
/u01/app/oracle

$ ps
  PID TTY      TIME CMD
 5804 pts/0    00:00:00 bash
13512 pts/0    00:00:00 workgen
13513 pts/0    00:00:00 workgen
13514 pts/0    00:00:00 wksh_wSERV_orcl
13518 pts/0    00:00:00 sqlplus
13526 pts/0    00:00:00 sleep
13539 pts/0    00:00:00 insert_orders.s
13541 pts/0    00:00:00 sqlplus
13583 pts/0    00:00:00 update_orders.s
13584 pts/0    00:00:00 sleep
13589 pts/0    00:00:00 sleep
13596 pts/0    00:00:00 ps
$
```

5. After the execution starts, access the Enterprise Manager Cloud Control Top Consumers page and observe that SERV1 is using some resources. Also, check the statistics on your service with V\$SERVICE_STATS from a SQL*Plus session connected as SYSDBA.

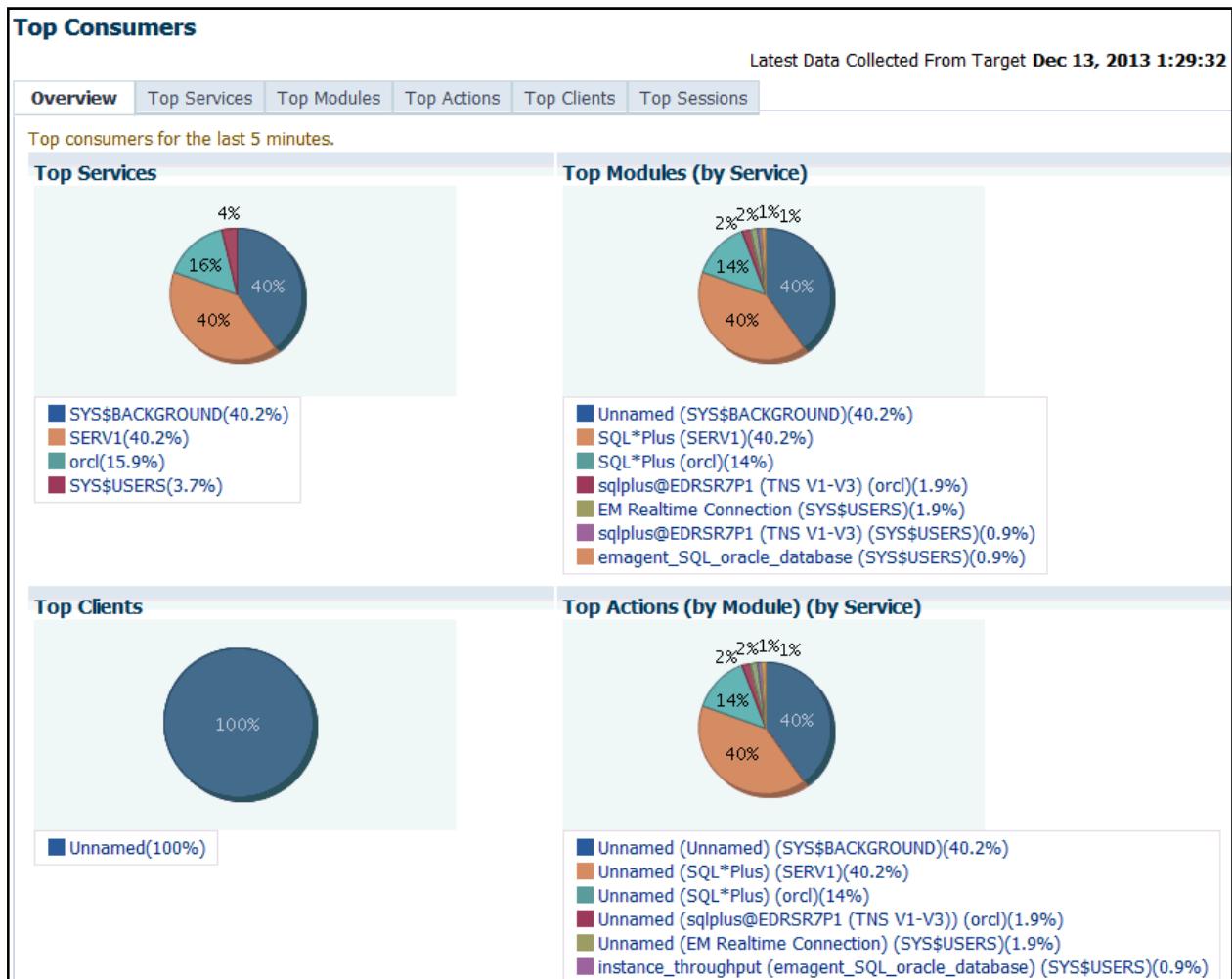
Step	Window/Page Description	Choices or Values
a.		Navigate to the orcl database home page.
b.	Database Instance: orcl	Select Performance > Performance Home .
c.	Database Login	Log in to the orcl database with the SYSDBA Database Credentials .
d.	Database Instance: orcl	Select Real Time: 1 Minute Refresh . Wait until the Average Active Sessions graph shows a significant rise.



Step	Window/Page Description	Choices or Values
e.	Database Instance: orcl	Click Top Consumers in the Additional Monitoring Links section at the bottom of the page.



- f. The names and number of services listed in the Top Services graph depend on the number and type of connections to the database. The network service name of each connection is recorded as a separate service. As a result, all the connections made without a service name will be aggregated, as will all the connections made as SERV1. Click the **Top Services** tab.



- g. On the Top Services page, click the **SERV1** link.

Top Consumers

Latest Data Collected From Target Aug 2, 2013

Overview Top Services Top Modules Top Actions Top Clients Top Sessions

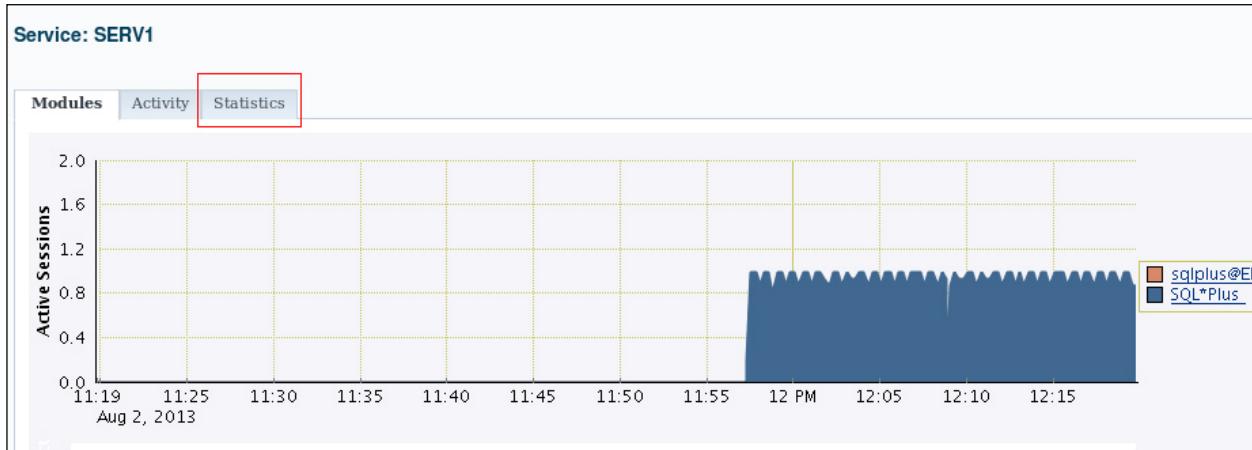
View Active Services

Enable SQL Trace | Disable SQL Trace | View SQL Trace File

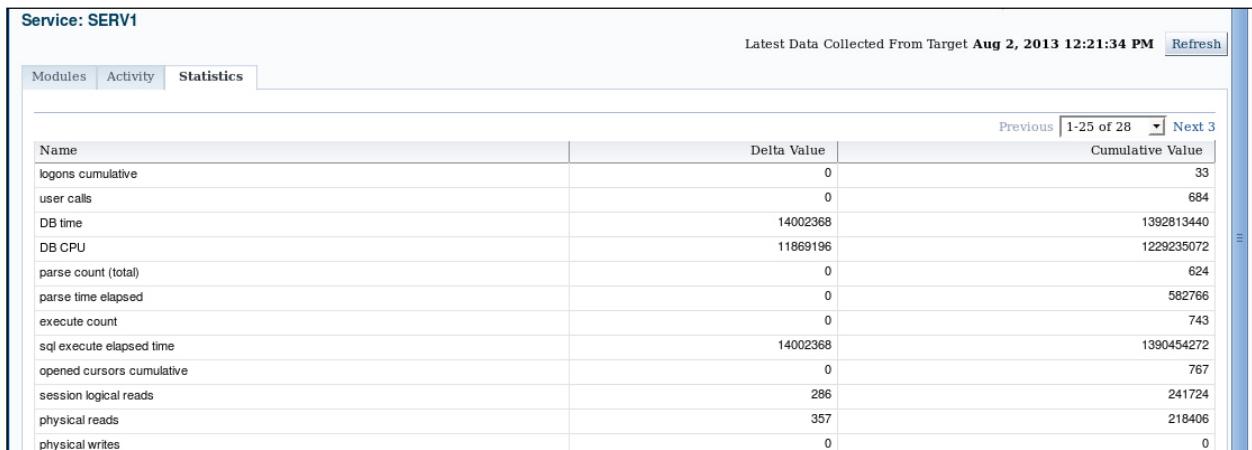
Select All | Select None

Select	Service	Activity (%) for the last 5 minutes	SQL Trace Enabled	Delta Elapsed Time (seconds)	Cumulative Elapsed Time (seconds)	Delta
<input type="checkbox"/>	orcl	71.4	FALSE	0	5713	
<input type="checkbox"/>	SYS\$BACKGROUND	13.6	FALSE	0	0	
<input type="checkbox"/>	SERV1	9.9	FALSE	0	1180	
<input type="checkbox"/>	SYS\$USERS	5.0	FALSE	0	14325	

- h. Click the **Statistics** tab.



- i. Notice that the statistics are similar to what is shown in an AWR report, but are aggregated by service on this page.



6. Return to the terminal window and stop the running workload by removing the \$HOME/workshops/runload file.

```
$ rm $HOME/workshops/runload
```

7. Set alert thresholds for your SERV1 service by using Enterprise Manager Cloud Control. Specify a warning threshold of 40,000,000 microseconds and a critical threshold of 100,000,000 microseconds.

Step	Window/Page Description	Choices or Values
a.	Service: SERV1	Select Oracle Database > Monitoring > Metric and Collection Settings .
b.	Metric and Collection Settings	Select All Metrics in the View menu. (The page will refresh.)
c.	Metric and Collection Settings	Click the multiedit icon (the group of pencils) for the Service Response Time (per user call) (microseconds) metric (under Database Services).

Step	Window/Page Description	Choices or Values
d.	Edit Advanced Settings: Service Response Time (per user call) (microseconds)	<p>Click Add.</p> <p>Specify the threshold values:</p> <ul style="list-style-type: none"> Service Name: SERV1 Warning Threshold: 40000000 Critical Threshold: 100000000 <p>Click Continue.</p>
e.	Metric and Collection Settings	Click OK .

Database Instance: orcl > Metric and Collection Settings
Metric and Collection Settings

Metrics Other Collected Items

View All metrics

Expand All | Collapse All

Metric	Comparison Operator	Warning Threshold	Critical Threshold	Corrective Actions	Collection Schedule	Edit
orcl						
> Alert Log				Disabled		
> Alert Log Error Status				Disabled		
> Archive Area				Every 15 Minutes		
> Database Files				Disabled		
> Database Job Status				Every 30 Minutes		
> Database Limits				Every 10 Minutes		
> Database Services				Every 10 Minutes		
Service CPU Time (per user call) (microseconds)	>			None		
Service Response Time (per user call) (microseconds)	>			None		
> Database Vault Attempted Violations - Command Rules					Every 1 Hour	

Database Instance: orcl > Metric and Collection Settings > Edit Advanced Settings: Service Response Time (per user call) (microseconds)
Edit Advanced Settings: Service Response Time (per user call) (microseconds)

Monitored Objects

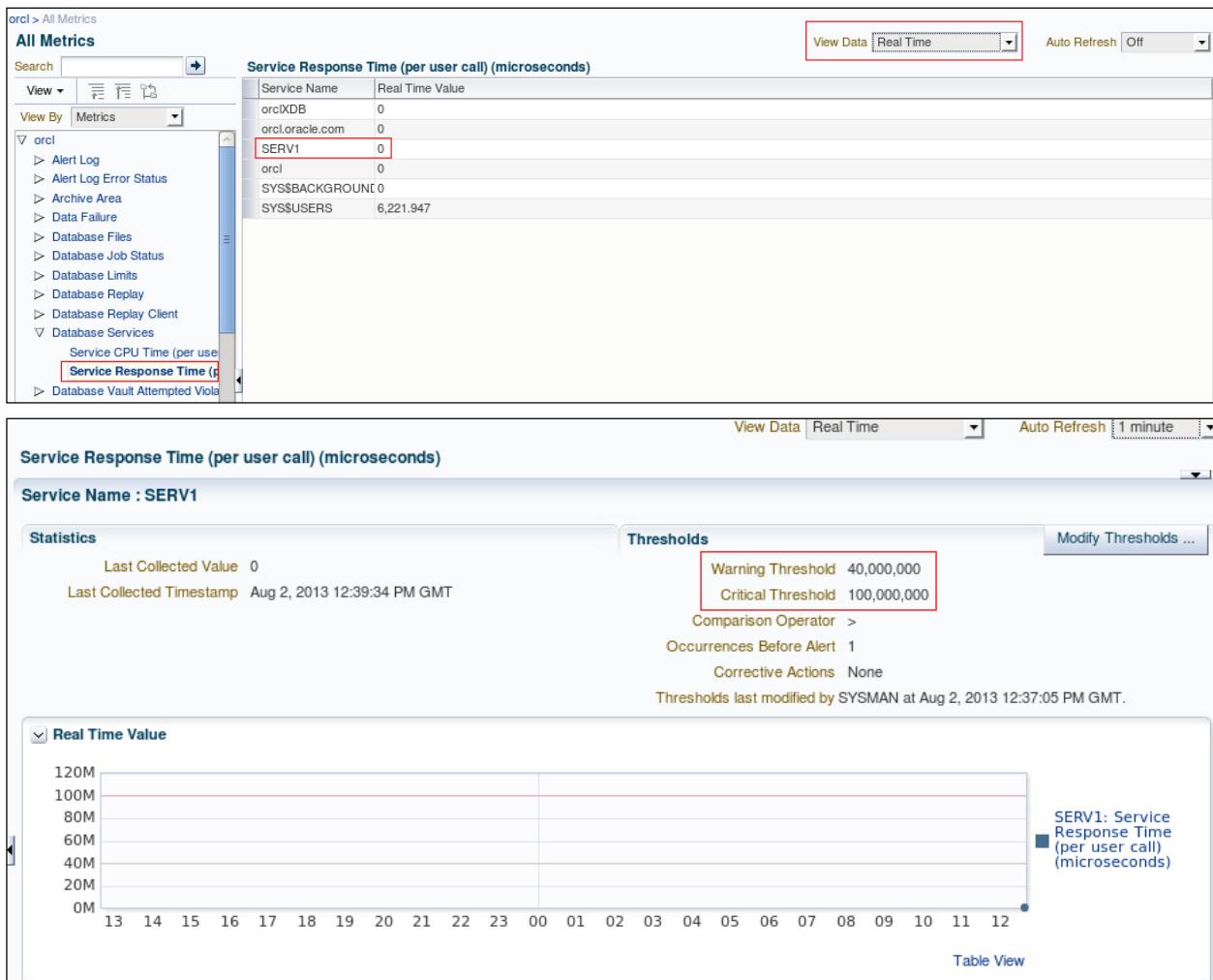
The table lists all Service Name objects monitored for this metric. You can specify different threshold settings for each Service Name object.

Edit **Remove**

Select	Service Name	Comparison Operator	Warning Threshold	Critical Threshold	Corrective Action
<input checked="" type="radio"/>	SERV1	>	40000000	100000000	None
<input type="radio"/>	All others	>			None

8. Verify the metric settings.

Step	Window/Page Description	Choices or Values
a.	orcl database home	Select Oracle Database > Monitoring > All Metrics .
b.	All Metrics	Expand Database Services . Click Service Response Time (per user call) (microseconds) . Select Real Time in the View Data menu.
c.	Service Response Time (per user call) (microseconds) section	Select the SERV1 row.
d.	Service Response Time (per user call) (microseconds): Service Name SERV1	Select 1 Minute in the Auto Refresh menu.



9. Return to the terminal window and execute the /home/oracle/labs/lab_09_01_10.sh script. This script executes a workload under the SERV1 service. The script takes a while to complete, so **proceed to the next step**.

```
$ cd $HOME/labs
$ ./lab_09_01_10.sh
SQL*Plus: Release 12.1.0.1.0 Production on Fri Aug 2 12:43:27 2013

Copyright (c) 1982, 2013, Oracle. All rights reserved.

Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.1.0 - 64bit
Production With the Partitioning, OLAP, Advanced Analytics and Real
Application Testing options

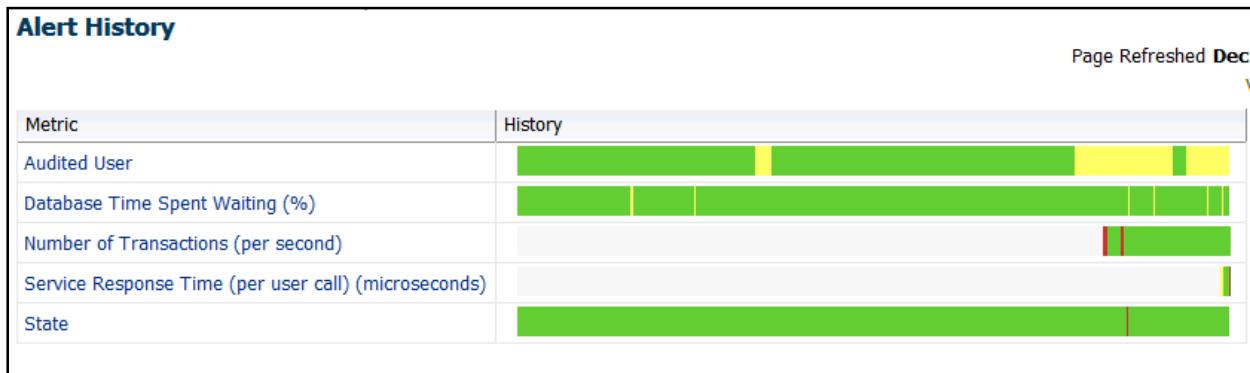
SQL> SQL> Connected.
SQL> SQL> SQL> 2      3      4      5      6      7      8
PL/SQL procedure successfully completed.

SQL> SQL> Disconnected from Oracle Database 12c Enterprise Edition
Release 12.1.0.1.0 - 64bit Production With the Partitioning, OLAP,
Advanced Analytics and Real Application Testing options
$
```

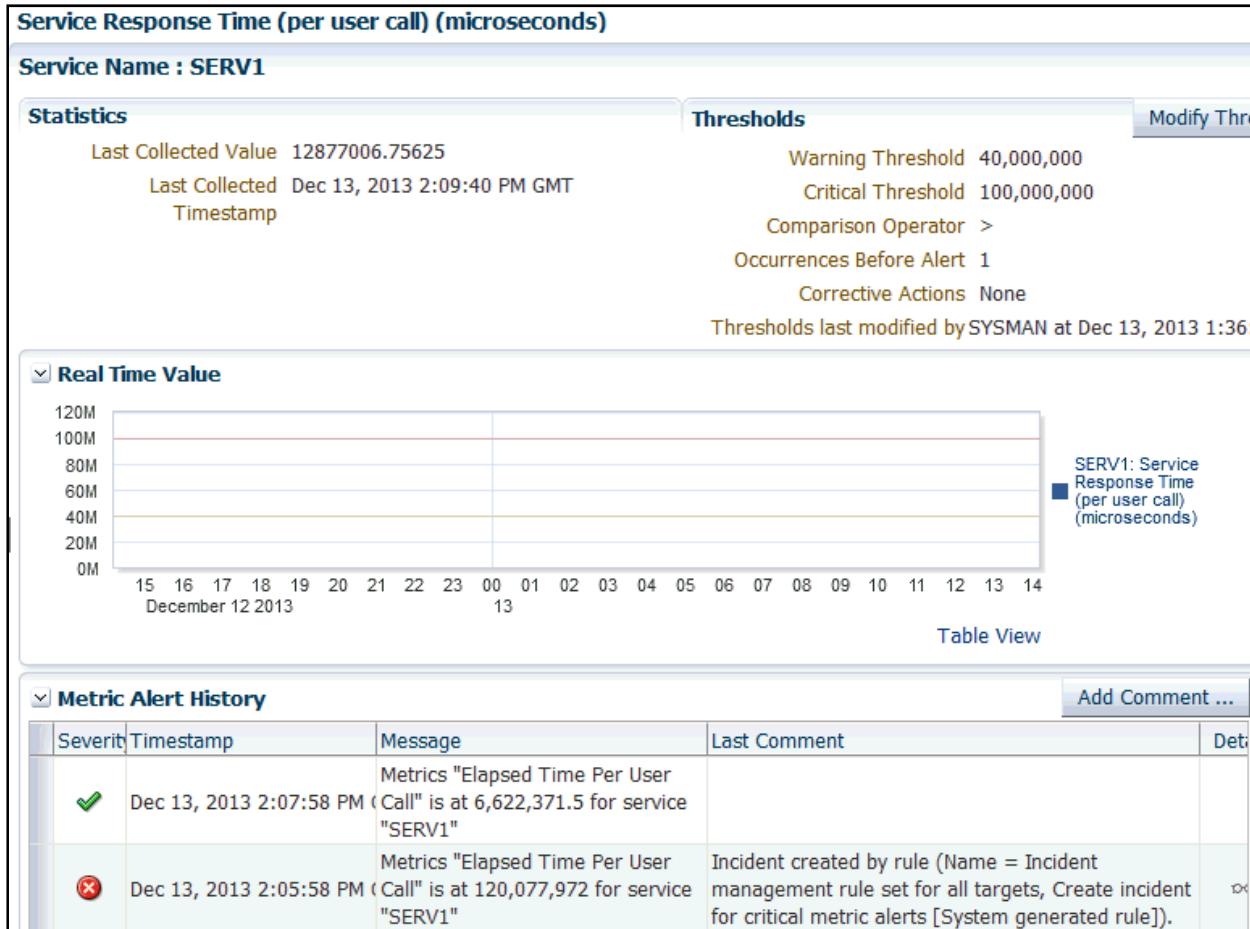
10. While the workload is executing, observe the Service Response Time graph for the SERV1 service. You may have to wait a couple of minutes for the graph to be updated.

Step	Window/Page Description	Choices or Values
a.	Service Response Time (per user call) (microseconds): Service Name SERV1	Monitor the Real Time Value graph. In this version, the graph does not update. Navigate to the Alert history page Oracle Database > Monitoring > Alert History .
b.	Alert History	Notice the Service Response Time (per user call) (microsecond) appears and is cleared shortly afterwards. Navigate to the Service Response Time (per user call) (microseconds): Service Name SERV1 page.
c.	Service Response Time (per user call) (microseconds): Service Name SERV1	The Metric Alert history should show that an alert was generated and then cleared.

10b)



10c)



11. Use Enterprise Manager Cloud Control to remove the thresholds that you specified during this practice.

Step	Window/Page Description	Choices or Values
a.	orcl database home	Expand the Oracle Database menu. Expand the Monitoring menu and select Metric and Collection Settings .
b.	Metric and Collection Settings	Click the multiedit icon on the right for Service Response Time (per user call) (microseconds) .
c.	Edit Advanced Settings: Service Response Time (per user call) (microseconds)	Select SERV1 . Click Remove . Click Continue .
d.	Metrics and Collection Settings	Click OK .
e.	Confirmation	Click OK .

Database Instance: orcl > Metric and Collection Settings

Metric and Collection Settings

Metrics Other Collected Items

View Metrics with thresholds

Expand All | Collapse All

Metric	Comparison Operator	Warning Threshold	Critical Threshold	Corrective Actions	Collection Schedule	Edit
orcl						
> Alert Log				Disabled		
> Alert Log Error Status				Disabled		
> Archive Area				Every 15 Minutes		
Database Job Status				Every 30 Minutes		
Database Services				Every 10 Minutes		
> Service Response Time (per user call) (microseconds)	>	40000000	100000000	None		
SERV1	>					
All others	>			None		
Database Vault Configuration Issues - Command Rules				Every 1 Hour		

Database Instance: orcl > Metric and Collection Settings > Edit Advanced Settings: Service Response Time (per user call) (microseconds)

Edit Advanced Settings: Service Response Time (per user call) (microseconds)

Monitored Objects

The table lists all Service Name objects monitored for this metric. You can specify different threshold settings for each Service Name object.

Select	Service Name	Comparison Operator	Warning Threshold	Critical Threshold	Corrective Action
<input checked="" type="radio"/>	SERV1	>	40000000	100000000	None
<input type="radio"/>	All others	>			None

TIP Empty Thresholds will disable alerts for that metric.
TIP Wildcards for object names are not supported as this is a Server Managed Metric. (Example: Key 'A%' is not allowed, but you can use 'A\%' exact phrase 'A%').

Practice 9-2: Tracing Services in a Single-Instance Oracle Environment

1. Execute the `lab_09_02_01.sh` script to remove all trace files from the `$ORACLE_BASE/diag/rdbms/orcl/orcl/trace` directory.

```
$ cd $HOME/labs
$ ./lab_09_02_01.sh

trace_dir=$ORACLE_BASE/diag/rdbms/orcl/orcl/trace

mv $trace_dir $trace_dir.old
mkdir $trace_dir
mv $trace_dir.old/alert_orcl.log $trace_dir/
rm -rf $trace_dir.old
$
```

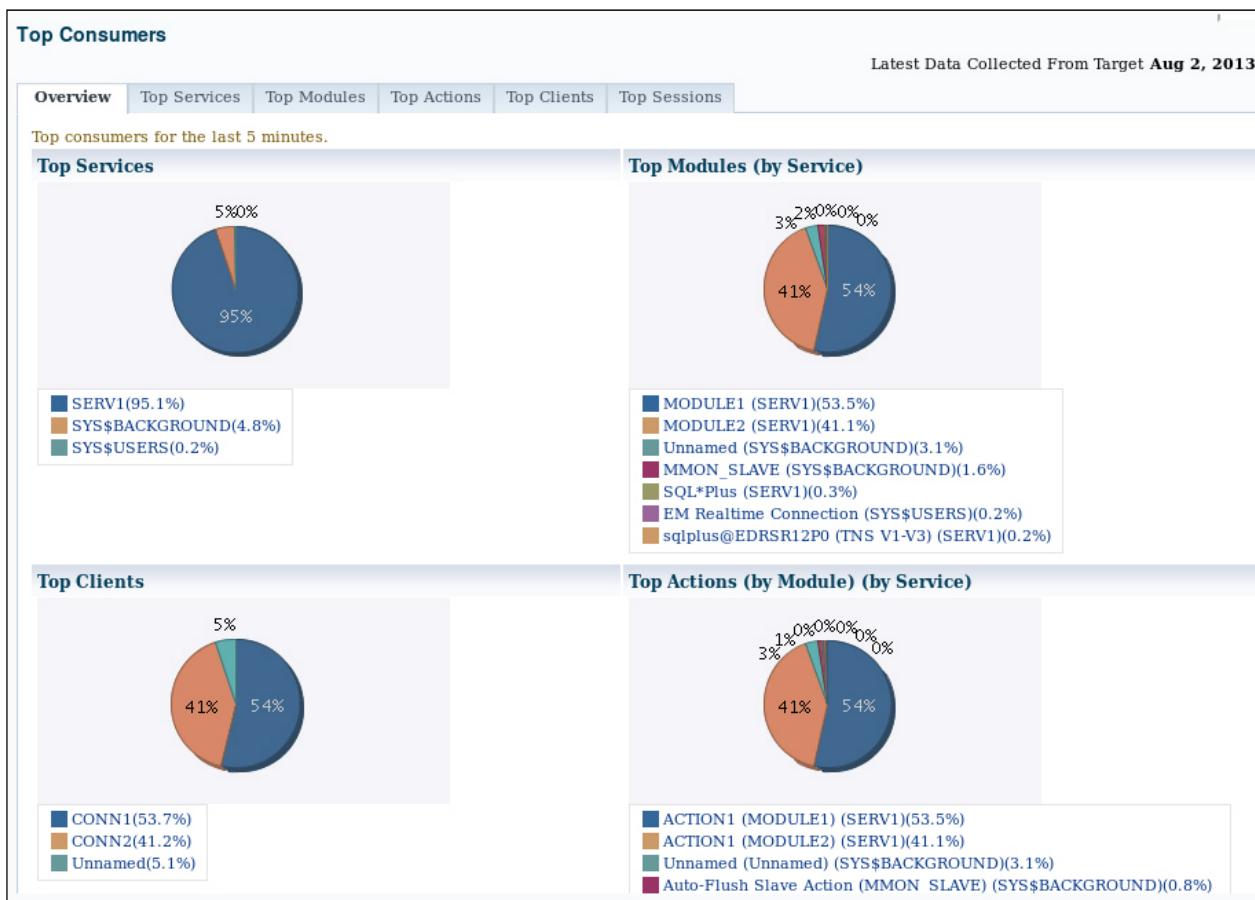
2. Execute the `start_servwork.sh` script to start four sessions that use the SERV1 service.

```
$ ./start_servwork.sh
Started stream with pid=31092
Started stream with pid=31093
Started stream with pid=31094
Started stream with pid=31095
$
PL/SQL procedure successfully completed.

$
```

3. Using Enterprise Manager Cloud Control, determine the list of services, modules, and actions that the workload is using.

Step	Window/Page Description	Choices or Values
a.	orcl database home	Expand the Performance menu and select Performance Home .
b.	Performance	Scroll to the bottom of the page and click the Top Consumers link in the Additional Monitoring Links section.
c.	Top Consumers (See the example screenshot, which follows the table.)	You should see that the current workload uses only the SERV1 service, and that some of the sessions are using the SERV1/MODULE1/ACTION1 action and some others are using the SERV1/MODULE2/ACTION1 action.



4. Using Enterprise Manager Cloud Control, enable statistics aggregation for both SERV1/MODULE1/ACTION1 and SERV1/MODULE2/ACTION1.

Step	Window/Page Description	Choices or Values
a.	Top Consumers	Click the Top Actions tab.
b.	Top Actions	Select the SERV1/MODULE1/ACTION1 action. Click Enable Aggregation .
c.	Top Actions	Select the SERV1/MODULE2/ACTION1 action. Click Enable Aggregation .
d.	Top Actions	You should see that additional statistics at the right of the window are now tracked for both actions (may require a refresh to see the additional statistics).

The screenshot shows the 'Top Consumers' page with the 'Top Actions' tab selected. The toolbar includes buttons for 'View Active Actions' (dropdown), 'Enable Aggregation' (highlighted with a red box), 'Disable Aggregation', 'Enable SQL Trace', 'Disable SQL Trace', and 'View SQL Trace File'. Below the toolbar is a checkbox group for 'Select All' or 'Select None'. The main area displays a table with columns: Select, Service, Module, Action, Activity (% for the last 5 minutes), Aggregation Enabled, SQL Trace Enabled, and Delta. The table data is as follows:

Select	Service	Module	Action	Activity (% for the last 5 minutes)	Aggregation Enabled	SQL Trace Enabled	Delta
<input type="checkbox"/>	SERV1	MODULE1	ACTION1	64.0	FALSE	FALSE	
<input checked="" type="checkbox"/>	SERV1	MODULE2	ACTION1	33.6	TRUE	FALSE	
<input type="checkbox"/>	SYS\$BACKGROUND	Unnamed	Unnamed	1.1	TRUE	FALSE	
<input type="checkbox"/>	SYS\$BACKGROUND	MMON_SLAVE	Automatic Report Flush	.5	FALSE	FALSE	
<input type="checkbox"/>	SYS\$USERS	EM Realtime Connection	Unnamed	.4	FALSE	FALSE	
<input type="checkbox"/>	SERV1	SQL*Plus	Unnamed	.3	FALSE	FALSE	
<input type="checkbox"/>	SERV1	sqlplus@EDRSR12P0 (TNS V1-V3)	Unnamed	.1	FALSE	FALSE	

5. Using Enterprise Manager Cloud Control, enable tracing for both SERV1/MODULE1/ACTION1 and SERV1/MODULE2/ACTION1. Let the system generate the trace files for one minute, and then disable tracing for both SERV1/MODULE1/ACTION1 and SERV1/MODULE2/ACTION1. After this is done, determine the list of generated trace files.

Step	Window/Page Description	Choices or Values
a.	Top Actions	Select Actions with Aggregation Enabled from the View menu. Select the SERV1/MODULE1/ACTION1 action. Click Enable SQL Trace .
b.	Enable SQL Trace	Click OK .
c.	Top Actions	Select the SERV1/MODULE2/ACTION1 action. Click Enable SQL Trace .
d.	Enable SQL Trace	Click OK .

Select	Service	Module	Action	Activity (% for the last 5 minutes)	SQL Trace Enabled	Delta Elapsed Time (seconds)	Cumulative Elapsed Tim
<input checked="" type="checkbox"/>	SERV1	MODULE1	ACTION1	92.4	FALSE	0	
<input type="checkbox"/>	SYS\$BACKGROUND	Unnamed	Unnamed	5.3	FALSE	0	
<input type="checkbox"/>	SERV1	MODULE2	ACTION1	.0	FALSE	0	

6. Find the trace files. In another terminal session, set the environment for the `orcl` database instance. Use the following commands:

```
$ . oraenv
$ ls -lt $ORACLE_BASE/diag/rdbms/orcl/orcl/trace/*.trc
```

```
$ . oraenv
ORACLE_SID = [orcl] ? orcl
The Oracle base for
ORACLE_HOME=/u01/app/oracle/product/12.1.0/dbhome_1 is
/u01/app/oracle
$ ls -lt $ORACLE_BASE/diag/rdbms/orcl/orcl/trace/*.trc
-rw-r----- 1 oracle oinstall 1315 Aug  2 13:15
/u01/app/oracle/diag/rdbms/orcl/orcl/trace/orcl_ora_32585.trc
-rw-r----- 1 oracle oinstall 1120 Aug  2 13:13
/u01/app/oracle/diag/rdbms/orcl/orcl/trace/orcl_mmon_27303.trc
$
```

7. Stop your workload by executing the `stop_servwork.sh` script, and then start it again by executing the `start_servwork.sh` script. Then check whether the statistics for your enabled aggregation actions are increasing.

```
$ ./stop_servwork.sh
Killing stream with pid=31095
Killing stream with pid=31094
Killing stream with pid=31093
Killing stream with pid=31092
$
$ ./start_servwork.sh
Started stream with pid=700
Started stream with pid=701
Started stream with pid=702
Started stream with pid=703
$
PL/SQL procedure successfully completed.

$
```

8. Look again at your **Top Actions** page. You should see statistics refreshing for your two actions.

Select	Service	Module	Action	Activity (% for the last 5 minutes) ▾	SQL Trace Enabled	Delta Elapsed Time (seconds)	Cumulative Elapsed Time (seconds)
<input type="checkbox"/>	SERV1	MODULE1	ACTION1	36.4	TRUE	26	121
<input checked="" type="checkbox"/>	SERV1	MODULE2	ACTION1	36.4	TRUE	28	123
<input type="checkbox"/>	SYS\$BACKGROUND	Unnamed	Unnamed	1.5	FALSE	0	0

9. Using Enterprise Manager Cloud Control, disable statistic aggregations for both the actions.
- On the Top Actions page, select both the actions and click **Disable Aggregation**.

Select	Service	Module	Action	Activity (% for the last 5 minutes) ▾	SQL Trace Enabled	Delta Elapsed Time (seconds)	Cumulative Elapsed Time (seconds)
<input checked="" type="checkbox"/>	SERV1	MODULE1	ACTION1	42.5	TRUE	28	28
<input checked="" type="checkbox"/>	SERV1	MODULE2	ACTION1	42.5	TRUE	30	30
<input type="checkbox"/>	SYS\$BACKGROUND	Unnamed	Unnamed	1.0	FALSE	0	0

- Log out of Enterprise Manager Cloud Control.
10. Stop the workload by using the `stop_servwork.sh` script.

```
$ ./stop_servwork.sh
Killing stream with pid=703
Killing stream with pid=702
Killing stream with pid=701
Killing stream with pid=700
$
```

11. Use the trace session utility and tkprof to interpret the generated trace files for the SERV1 service and the MODULE1 module.

```
$ cd $ORACLE_BASE/diag/rdbms/orcl/orcl/trace
$ trcsess output=serv1_module1.trc service=SERV1 *ora*.trc
$ tkprof serv1_module1.trc results.trc

TKPROF: Release 12.1.0.1.0 - Development on Fri Aug 2 13:24:51 2013

Copyright (c) 1982, 2013, Oracle and/or its affiliates. All rights reserved.

$
```

12. View the output of tkprof with your favorite viewer or editor. The viewer less is shown in the code box.

```
$ less results.trc
TKPROF: Release 12.1.0.1.0 - Development on Fri Aug 2 13:24:51 2013

Copyright (c) 1982, 2013, Oracle and/or its affiliates. All rights reserved.

Trace file: serv1_module1.trc
Sort options: default

*****
count      = number of times OCI procedure was executed
cpu        = cpu time in seconds executing
elapsed    = elapsed time in seconds executing
disk       = number of physical reads of buffers from disk
query      = number of buffers gotten for consistent read
current    = number of buffers gotten in current mode (usually for update)
rows       = number of rows processed by the fetch or execute call
*****


SQL ID: 1hb6a71hd3d53 Plan Hash: 0

BEGIN dbms_application_info.set_module('MODULE1','ACTION1'); END;

call      count      cpu    elapsed      disk      query      current      rows
-----  -----  -----  -----  -----  -----  -----  -----
Parse      0      0.00      0.00      0          0          0          0          0
Execute     2      0.00      0.00      0          0          0          0          2
Fetch       0      0.00      0.00      0          0          0          0          0
-----  -----  -----  -----  -----  -----  -----  -----
total      2      0.00      0.00      0          0          0          0          2

Misses in library cache during parse: 0
Optimizer mode: ALL_ROWS
Parsing user id: 150

Elapsed times include waiting on following events:
Event waited on          Times   Max. Wait  Total Waited
-----  -----  -----  -----
SQL*Net message to client      2      0.00      0.00
SQL*Net message from client      2      0.00      0.00
*****


...
(type q to exit)
```

13. Clean up from this practice by running the lab_09_02_13.sh script.

```
$ cd $HOME/labs
$ ./lab_09_02_13.sh

PL/SQL procedure successfully completed.

PL/SQL procedure successfully completed.
$
```

Practices for Lesson 10: Identifying Problem SQL Statements

Chapter 10

Practices for Lesson 10: Overview

Practices Overview

In this practice, you examine and compare the information provided by some built-in diagnostic tools. You start with a poorly performing SQL statement and examine the output of the various tools to try to determine the reasons for poor performance.

Practice 10-1: Using AUTOTRACE and DBMS_XPLAN

In this practice, you examine a bad SQL statement and gather information about it by using the SQL*Plus AUTOTRACE command.

- Create the PLUSTRACE role. To use the SQL*Plus AUTOTRACE command, the PLUSTRACE role must be granted to the user that wants to use AUTOTRACE.

```
$ sqlplus / as sysdba

SQL> @?/sqlplus/admin/plustrce.sql
SQL>
SQL> drop role plustrace;
drop role plustrace
*
ERROR at line 1:
ORA-01919: role 'PLUSTRACE' does not exist

SQL> create role plustrace;

Role created.

SQL>
SQL> grant select on v_$sesstat to plustrace;

Grant succeeded.

SQL> grant select on v_$statname to plustrace;

Grant succeeded.

SQL> grant select on v_$mystat to plustrace;

Grant succeeded.

SQL> grant plustrace to dba with admin option;

Grant succeeded.

SQL>
SQL> set echo off
SQL>
```

- Grant the PLUSTRACE role to the SH user. Exit from SQL*Plus.

```
SQL> grant plustrace to sh;

Grant succeeded.

SQL> exit
```

3. The SQL statement has been isolated, and you can view it in the `bad_sql.sql` file. The script is in the `$HOME/labs/sqltune` directory. What can be determined by inspecting this statement? Each observation produces more specific questions.

```
$ cd $HOME/labs/sqltune
$ cat bad_sql.sql
select time_id, QUANTITY SOLD, AMOUNT SOLD
  from sales s, customers c
 where c.cust_id = s.cust_id and CUST_FIRST_NAME='Dina'
   order by time_id;
```

- a. This statement has a join between two tables: SALES and CUSTOMERS.

Information: These are large tables in the Sales History (SH) schema.

Question: Are the statistics up-to-date on these tables?

- b. The join column is `CUST_ID` in both the tables.

Questions:

Are there indexes on these columns?

Do the indexes have up-to-date statistics?

Are the indexes built properly?

- c. The selection predicate is `CUST_FIRST_NAME='Dina'`.

Questions: Does `CUST_FIRST_NAME` have an index?

- d. The `ORDER BY` clause may require a sort.

Question: Is the sort area in memory sized properly?

4. Review the information provided by AUTOTRACE about the execution of this statement.

Note: The Explain Plan output has been reformatted for readability.

- a. Execute the `bad_sql.sql` script with AUTOTRACE ON.

```
$ sqlplus /nolog

SQL> connect sh/sh
Connected.
SQL> set autotrace on
SQL> @bad_sql.sql

...
TIME_ID      QUANTITY SOLD  AMOUNT SOLD
-----      -----
24-DEC-01          1        32.27
24-DEC-01          1        33.25
28-DEC-01          1        56.96
28-DEC-01          1        56.96

895 rows selected.
```

```

Execution Plan
-----
Plan hash value: 1897253553

-----
| Id  | Operation          | Name   | Rows  | Bytes | Cost (%CPU) | Time    | Pstart| Pstop |
|-----|
| 0   | SELECT STATEMENT   |        | 5557  | 179K | 941    (1) | 00:00:01 |        |        |
| 1   | SORT ORDER BY     |        | 5557  | 179K | 941    (1) | 00:00:01 |        |        |
|* 2  | HASH JOIN          |        | 5557  | 179K | 941    (1) | 00:00:01 |        |        |
|* 3  | TABLE ACCESS FULL | CUSTOMERS | 43  | 516  | 423    (1) | 00:00:01 |        |        |
| 4  | PARTITION RANGE ALL|        | 918K | 18M  | 516    (2) | 00:00:01 | 1      |        |
| 5  | TABLE ACCESS FULL | SALES  | 918K | 18M  | 516    (2) | 00:00:01 | 1      | 28    |
|-----|
Predicate Information (identified by operation id):
-----
2 - access("C"."CUST_ID"="S"."CUST_ID")
3 - filter("CUST_FIRST_NAME"='Dina')

Note
-----
- this is an adaptive plan

Statistics
-----
 1799 recursive calls
    0 db block gets
 6345 consistent gets
 3255 physical reads
    0 redo size
21271 bytes sent via SQL*Net to client
 1193 bytes received via SQL*Net from client
    61 SQL*Net roundtrips to/from client
  143 sorts (memory)
    0 sorts (disk)
  895 rows processed
SQL> set autotrace off

```

- What objects are accessed? *The SALES and CUSTOMERS tables*
- What object is accessed first? *The CUSTOMERS table is read into a hash table.*
- Which object is partitioned? *The SALES table is partitioned.*
- How many partitions are accessed? *All 28 partitions*
- How many rows are returned (Cardinality)? *895 rows*
- How many rows does the execution plan expect? *5557 rows*
- What is the cost? *941, may vary due to specific hardware*

- i. Are any indexes used? *No*
 - j. How much sort space is expected? *179 KB*
 - k. How many consistent reads (logical reads) were done? *6345; will vary*
 - l. How many physical reads were done? *3255; may vary due to the preceding activity*
 - m. Did the sort spill to disk? *No. "sorts, disk" was 0.*
5. Use DBMS_XPLAN.DISPLAY to get only the explain plan for the SQL statement. The command requires that the statement be explained with the EXPLAIN PLAN command; then, the plan table is displayed. The terminal window must be resized and set for at least a 108-character width to display properly.
- Generate an explain plan for the SQL statement.

```
SQL> EXPLAIN PLAN FOR
 2> select time_id, QUANTITY_SOLD, AMOUNT_SOLD
 3>   from sales s, customers c
 4>   where c.cust_id = s.cust_id and CUST_FIRST_NAME='Dina'
 5>   order by time_id;
```

Explained

- b. Display the plan table. Note that this is the same explain plan that you received from the AUTOTRACE command. Exit from SQL*Plus.

```
SQL> SET LINESIZE 130
SQL> SET PAGESIZE 0
SQL> SELECT * FROM table(DBMS_XPLAN.DISPLAY);
```

Plan hash value: 1897253553

Id	Operation	Name	Rows	Bytes	Cost	(%CPU)	Time	Pstart	Pstop
0	SELECT STATEMENT		5557	179K	941	(1)	00:00:01		
1	SORT ORDER BY		5557	179K	941	(1)	00:00:01		
* 2	HASH JOIN		5557	179K	941	(1)	00:00:01		
* 3	TABLE ACCESS FULL	CUSTOMERS	43	516	423	(1)	00:00:01		
4	PARTITION RANGE ALL		918K	18M	516	(2)	00:00:01	1	28
5	TABLE ACCESS FULL	SALES	918K	18M	516	(2)	00:00:01	1	28

Predicate Information (identified by operation id):

```
2 - access("C"."CUST_ID"="S"."CUST_ID")
3 - filter("CUST_FIRST_NAME">'Dina')
```

Note

- this is an adaptive plan

22 rows selected.

```
SQL> exit
```

Practice 10-2: Using the SQL TRACE Facility

In this practice, you trace a bad SQL statement and format the trace file with the tkprof utility.

- Start a session as the SH user to trace the SQL statement in the `bad_sql.sql` script. Set the `TRACEFILE_IDENTIFIER` parameter for your session. Add the "badsql" string to the trace file name.

```
ALTER SESSION SET TRACEFILE_IDENTIFIER="badsql";
```

```
$ sqlplus /nolog

SQL> connect sh/sh
Connected.
SQL>

SQL> ALTER SESSION SET TRACEFILE_IDENTIFIER="badsql";
```

- Enable tracing in your session and then run the `bad_sql.sql` script from the `$HOME/labs/sqltune` directory.

```
SQL> EXECUTE DBMS_SESSION.SET_SQL_TRACE(TRUE);

PL/SQL procedure successfully completed.

SQL> @$HOME/labs/sqltune/bad_sql.sql

...
TIME_ID      QUANTITY SOLD AMOUNT SOLD
-----      -----
24-DEC-01          1      32.27
24-DEC-01          1      33.25
28-DEC-01          1      56.96
28-DEC-01          1      56.96

895 rows selected.

SQL> exit
```

- Find and format the trace file. The trace file will be in the `$ORACLE_BASE/diag/rdbms/orcl/orcl/trace` directory with the `badsql` string embedded in the name. The `ls *badsql*.trc` command returns the name of the trace file. Use that name of the trace file in the `tkprof` command.

```
$ cd $ORACLE_BASE/diag/rdbms/orcl/orcl/trace
$ ls *badsql*.trc
orcl_ora_7029_badsql.trc
$ tkprof orcl_ora_7029_badsql.trc badsql.txt EXPLAIN=sh/sh SYS=NO
TKPROF: Release 12.1.0.1.0 - Development on Mon Aug 5 09:05:29 2013

Copyright (c) 1982, 2013, Oracle and/or its affiliates. All rights reserved.
$
```

4. View the `badsql.txt` file. Locate the following SQL statement:

```
select time_id, QUANTITY SOLD, AMOUNT SOLD
  from sales s, customers c
 where c.cust_id = s.cust_id and CUST_FIRST_NAME='Dina'
   order by time_id
```

Find the trace summary below the statement as shown in the following example.

The values shown are examples. The values you see may differ.

How much CPU time was used to fetch the records? 0.28

How much elapsed time was used for the entire query? 0.31

What was the number of physical reads (disk)? 3122

What was the number of logical reads (query + current)? 3140

```
$ less $ORACLE_BASE/diag/rdbms/orcl/orcl/trace/badsql.txt

TKPROF: Release 12.1.0.1.0 - Development on Mon Aug 5 09:05:29 2013

Copyright (c) 1982, 2013, Oracle and/or its affiliates. All rights reserved.

Trace file: orcl_ora_7029_badsql.trc
Sort options: default

*****
count      = number of times OCI procedure was executed
cpu        = cpu time in seconds executing
elapsed    = elapsed time in seconds executing
disk       = number of physical reads of buffers from disk
query      = number of buffers gotten for consistent read
current    = number of buffers gotten in current mode (usually for update)
rows       = number of rows processed by the fetch or execute call
*****
...
select time_id, QUANTITY SOLD, AMOUNT SOLD
  from sales s, customers c
 where c.cust_id = s.cust_id and CUST_FIRST_NAME='Dina'
   order by time_id

call      count      cpu      elapsed      disk      query      current      rows
-----  -----  -----  -----  -----  -----  -----  -----
Parse      1      0.02      0.02          0          0          0          0
Execute    1      0.00      0.00          0          0          0          0
Fetch     61      0.28      0.28      3122      3140          0      895
-----
total     63      0.31      0.31      3122      3140          0      895

Misses in library cache during parse: 1
Optimizer mode: ALL_ROWS
Parsing user id: 105 (SH)
Number of plan statistics captured: 1

Rows (1st) Rows (avg) Rows (max)  Row Source Operation
-----  -----  -----  -----
  895      895      895  SORT ORDER BY (cr=3140 pr=3122 pw=0 time=288094 us cost=941
size=183381 card=5557)
  895      895      895  HASH JOIN  (cr=3140 pr=3122 pw=0 time=51833 us cost=941
size=183381 card=5557)
    65      65      65  NESTED LOOPS (cr=1520 pr=1518 pw=0 time=2017 us)
    65      65      65  NESTED LOOPS (cr=1520 pr=1518 pw=0 time=2015 us cost=941
size=183381 card=5557)
    65      65      65  STATISTICS COLLECTOR (cr=1520 pr=1518 pw=0 time=2015 us)
    65      65      65  TABLE ACCESS FULL CUSTOMERS (cr=1520 pr=1518 pw=0
time=1937 us cost=423 size=516 card=43)
      0      0      0  PARTITION RANGE ALL PARTITION: 1 28 (cr=0 pr=0 pw=0 time=0
us)
      0      0      0  BITMAP CONVERSION TO ROWIDS (cr=0 pr=0 pw=0 time=0 us)
      0      0      0  BITMAP INDEX SINGLE VALUE SALES_CUST_BIX PARTITION: 1 28
(cr=0 pr=0 pw=0 time=0 us) (object id 92706)
```

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

```

          0          0          0      TABLE ACCESS BY LOCAL INDEX ROWID SALES PARTITION: 1 1 (cr=0
pr=0 pw=0 time=0 us cost=516 size=2730 card=130)
918843    918843    918843      PARTITION RANGE ALL PARTITION: 1 28 (cr=1620 pr=1604 pw=0
time=290094 us cost=516 size=19295703 card=918843)
918843    918843    918843      TABLE ACCESS FULL SALES PARTITION: 1 28 (cr=1620 pr=1604
pw=0 time=172463 us cost=516 size=19295703 card=918843)

Rows     Execution Plan
-----
 0  SELECT STATEMENT  MODE: ALL_ROWS
 895   SORT (ORDER BY)
 895   HASH JOIN
 65    NESTED LOOPS
 65    NESTED LOOPS
 65    STATISTICS COLLECTOR
 65      TABLE ACCESS  MODE: ANALYZED (FULL) OF 'CUSTOMERS'
                  (TABLE)
 0      PARTITION RANGE (ALL) PARTITION: START=1 STOP=28
 0      BITMAP CONVERSION (TO ROWIDS)
 0      BITMAP INDEX (SINGLE VALUE) OF 'SALES_CUST_BIX'
                  (INDEX (BITMAP)) PARTITION: START=1 STOP=28
 0      TABLE ACCESS  MODE: ANALYZED (BY LOCAL INDEX ROWID) OF
                  'SALES' (TABLE) PARTITION: START=1 STOP=1
918843  PARTITION RANGE (ALL) PARTITION: START=1 STOP=28
918843  TABLE ACCESS  MODE: ANALYZED (FULL) OF 'SALES' (TABLE)
                  PARTITION: START=1 STOP=28

*****
OVERALL TOTALS FOR ALL NON-RECURSIVE STATEMENTS

call      count        cpu      elapsed       disk      query      current      rows
-----
Parse        1        0.02        0.02         0          0          0          0
Execute      2        0.00        0.00        11         72          0          1
Fetch       61        0.28        0.28      3122       3140          0        895
-----
total      64        0.31        0.31      3133       3212          0        896

Misses in library cache during parse: 1
Misses in library cache during execute: 1

OVERALL TOTALS FOR ALL RECURSIVE STATEMENTS

call      count        cpu      elapsed       disk      query      current      rows
-----
Parse       91        0.01        0.01         0          0          0          0
Execute     860        0.05        0.06         0          0          0          0
Fetch      1515        0.02        0.02       120       3273          0      1441
-----
total     2466        0.09        0.10       120       3273          0      1441

Misses in library cache during parse: 33
Misses in library cache during execute: 34

 3  user  SQL statements in session.
 81 internal SQL statements in session.
 84 SQL statements in session.
 1 statement EXPLAINed in this session.
*****
Trace file: orcl_ora_7029_badsql.trc
Trace file compatibility: 11.1.0.7
Sort options: default

 1  session in tracefile.
 3  user  SQL statements in trace file.
 81 internal SQL statements in trace file.

```

```
84 SQL statements in trace file.  
55 unique SQL statements in trace file.  
1 SQL statements EXPLAINed using schema:  
  SH.prof$plan_table  
    Default table was used.  
    Table was created.  
    Table was dropped.  
3921 lines in trace file.  
20 elapsed seconds in trace file.  
  
/* use 'q' to exit the less file viewer */  
  
[oracle@EDRSR12P0 trace]$
```

Practices for Lesson 11: Influencing the Optimizer

Chapter 11

Practices for Lesson 11: Overview

Practices Overview

In these practices, you demonstrate various ways to influence the optimizer by changing the SQL statements.

Practice 11-1: Capturing Extended Statistics

In this practice, you create extended statistics over a multicolumn set and an expression.

1. In the SH.CUSTOMERS table, the CUST_CITY, CUST_STATE_PROVINCE, and COUNTRY_ID columns are frequently used together as predicates. Because these columns are related, the combined selectivity is very different from the selectivity that the optimizer would calculate. The optimizer assumes that the columns are independent. To capture the statistics over this set of columns, there are two methods: create a concatenated index over these columns and collect the statistics over the index, or create a column group to collect statistics over this column group without creating an index. Examine the lab_11_01_01.sql script in the \$HOME/labs/sqltune directory, and then execute it.

```
$ cd $HOME/labs/sqltune
$ sqlplus /nolog

SQL> connect sh/sh
Connected.
SQL> set echo on
SQL> @lab_11_01_01.sql
SQL> SET SERVEROUTPUT ON
SQL>
SQL> exec
dbms_stats.drop_extended_stats('sh','customers','(CUST_CITY,CUST_STA
TE_PROVINCE,COUNTRY_ID)');
BEGIN
dbms_stats.drop_extended_stats('sh','customers','(CUST_CITY,CUST_STA
TE_PROVINCE,COUNTRY_ID)'); END;

*
ERROR at line 1:
ORA-20000: extension "(CUST_CITY,CUST_STATE_PROVINCE,COUNTRY_ID)"
does not
exist
ORA-06512: at "SYS.DBMS_STATS", line 13055
ORA-06512: at "SYS.DBMS_STATS", line 45105
ORA-06512: at line 1

SQL>
SQL> DECLARE
 2      cg_name  VARCHAR2(30);
 3  BEGIN
 4    cg_name := dbms_stats.create_extended_stats('SH','CUSTOMERS',
 5                      '(CUST_CITY,CUST_STATE_PROVINCE,COUNTRY_ID)');
 6
 7    dbms_output.put_line('column group name is:'||cg_name);
 8
 9    dbms_stats.gather_table_stats('SH','CUSTOMERS',
10      method_opt=>'for all columns size 1, for columns
(CUST_CITY,CUST_STATE_PROVINCE,COUNTRY_ID) size 3');
11  END;
12 /
column group name is:SYS_STUMZ$C3AIHLPBROI#SKA58H_N
```

PL/SQL procedure successfully completed.

SQL>
SQL>

2. The SELECT statement in lab_11_01_02.sql gives distinct values of each column in the group.

```
SQL> @lab_11_01_02.sql
SQL> select count(distinct(cust_city)) cities,
  2      count(distinct(CUST_STATE_PROVINCE)) states,
  3      count(distinct(country_id)) countries
  4  from customers;

          CITIES      STATES    COUNTRIES
-----  -----  -----
        620        145          19
```

3. The SELECT statement in lab_11_01_03.sql shows the selectivity that the default algorithm in the optimizer would find.

```
SQL> @lab_11_01_03.sql
SQL> select 1/( count(distinct(cust_city)))*
  2      1/( count(distinct(CUST_STATE_PROVINCE)))*
  3      1/( count(distinct(country_id)))  default_selectivity
  4  from customers
  5  /

DEFAULT_SELECTIVITY
-----
5.8545E-07
```

4. The PL/SQL block in lab_11_01_04.sql retrieves the column group name.

```
SQL> @lab_11_01_04.sql
SQL> declare
  2      cg_name VARCHAR2(30);
  3  BEGIN
  4      cg_name := dbms_stats.show_extended_stats_name
  5                  ('SH','CUSTOMERS',
  6                   '(CUST_CITY,CUST_STATE_PROVINCE,COUNTRY_ID)');
  7
  8      dbms_output.put_line('column group name is:'||cg_name);
  9
 10 END;
 11 /
column group name is:SYS_STUMZ$C3AIHLPBROI#SKA58H_N

PL/SQL procedure successfully completed.
```

5. The `lab_11_01_05.sql` script retrieves the statistics on the column group. `NUM_DISTINCT` shows the number of distinct values for the three columns. The selectivity for the column group is 1/620 or 1.61E-3, rather than the 5.8E-7 that the default calculation would have given. The extended statistics help the optimizer make better choices, with more accurate selectivity when these columns are used together in a predicate.

```
SQL> @lab_11_01_05.sql
SQL> select e.extension col_group, t.num_distinct, t.histogram
  2      from user_stat_extensions e, user_tab_col_statistics t
  3      where e.extension_name=t.column_name
  4      and t.table_name='CUSTOMERS';

COL_GROUP
-----
-----
-----  

NUM_DISTINCT HISTOGRAM
-----  

("CUST_CITY", "CUST_STATE_PROVINCE", "COUNTRY_ID")
   620 HYBRID
```

6. Tables often have columns that are not populated that are called “virtual columns.” They are calculated as part of the `SELECT` statement. When these virtual columns are used in a predicate, `ORDER BY`, or join conditions, the optimizer needs statistics to make fully informed decisions. The `lab_11_01_06.sql` script creates a table with four columns and populates it. The `id` column contains 1,000 distinct numbers: 1 to 999. The second column contains only an X. The `Cid` column contains `id` as a character string. The `Nid` column contains `(id*100)+6`, so that an `id` of 27 becomes the `Nid` of 2706.

```
SQL> @lab_11_01_06.sql
SQL>
SQL> DROP TABLE EXP_TEST PURGE;
DROP TABLE EXP_TEST PURGE
*
ERROR at line 1:
Ora-00942: table or view does not exist

SQL>
SQL> Create table exp_test ( id Number(6,0),
  2                           pad      varchar2 (20),
  3                           Cid      VARCHAR(20),
  4                           Nid      NUMBER(6,0));

Table created.

SQL>
SQL> Begin
 2   FOR i in 1..1000 LOOP
 3
 4     INSERT INTO exp_test values(
 5       i, 'X', TO_CHAR(i), i*100+6);
 6   END LOOP;
 7 END;
```

```
8 /
PL/SQL procedure successfully completed.
SQL>
```

7. When a column named in the predicate has a function applied to it, the index on the column cannot be used. A function-based index on the expression can be created to allow an index access to be used. The lab_11_01_07.sql script creates a function-based index over the `(mod(id,10))` expression.

```
SQL> @lab_11_01_07.sql
SQL> set echo on
SQL>
SQL> CREATE index exp_mod on exp_test(mod(id,10));
Index created.

SQL>
```

8. Execute the lab_11_01_08.sql script to create a virtual column over the `(mod(Nid,10))` expression.

```
SQL> @lab_11_01_08.sql
SQL> set echo on
SQL>
SQL> select dbms_stats.create_extended_stats(
  2       'sh','exp_test','(mod(Nid,10))') from dual;

DBMS_STATS.CREATE_EXTENDED_STATS('SH','EXP_TEST','(MOD(NID,10))')
-----
SYS_STUFYQ8TH8YVE2JG8NLWCN9XTR
```

9. Gather statistics on the EXP_TEST table by using the lab_11_01_09.sql script. The virtual column created in step 8 is explicitly named in the `method_opt` parameter.

```
SQL> @lab_11_01_09.sql
SQL> set echo on
SQL>
SQL> exec dbms_stats.gather_table_stats('sh','exp_test',-
> method_opt=>'for all columns size 1 for columns (mod(Nid,10))
size 3');

PL/SQL procedure successfully completed.

SQL>
```

10. The extended statistics have been calculated for the EXP_TEST table. The extended statistics have been defined for `(mod(Mid,10))`. Use the lab_11_01_10.sql script to see the values of the extended statistics,

```
SQL> @lab_11_01_10.sql
SQL> select e.extension col_group, t.num_distinct, t.histogram
  2      from user_stat_extensions e, user_tab_col_statistics t
  3      where e.extension_name=t.column_name
  4      and t.table_name='EXP_TEST';
```

```
COL_GROUP          NUM_DISTINCT HISTOGRAM
-----
(MOD ("NID",10))           1 FREQUENCY
(MOD ("ID",10))            10 NONE
SQL>
```

The extended statistics were calculated over the EXP_TEST functional index that was created in step 8, as well as over the virtual column defined in step 9.

11. Exit from SQL*Plus.

```
SQL> exit
$
```

Practice 11-2: Influencing Optimizer Performance by Statistics Changes

In this practice, you see that statistics can influence the optimizer performance.

1. Create a table, and gather statistics on it and the associated indexes as the SH user. Use the `cr_badstats.sql` script in the `$HOME/labs/sqltune` directory to create the CTEST table and the associated indexes, and to set up the scenario. Statistics are gathered, and then enough rows are inserted into the CTEST table to render the statistics stale.

```
$ cd $HOME/labs/sqltune
$ sqlplus /nolog

SQL> connect sh/sh
Connected
SQL> @cr_badstats.sql
DROP SEQUENCE ctest_pk_seq
*
ERROR at line 1:
ORA-02289: sequence does not exist

DROP TABLE CTEST PURGE
*
ERROR at line 1:
ORA-00942: table or view does not exist

Table created.

Table altered.

Index created.

Index created.

PL/SQL procedure successfully completed.

Sequence created.

18520 rows created.

37040 rows created.

SQL>
```

2. Enable AUTOTRACE on the SQL statement by using the `bad_stats.sql` script, and observe the execution plan and the differences between the estimated cost and the actual cost. The script is in the `$HOME/labs/sqltune` directory.

```
SQL> @bad_stats.sql
SQL> SET AUTOTRACE TRACEONL
SQL>
SQL> SELECT CUST_ID, CUST_LAST_NAME, CUST_TOTAL
  2  FROM CTEST
```

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

```
3 WHERE COUNTRY_ID = 52779;
3833 rows selected.
```

Execution Plan

```
Plan hash value: 3591594933
```

Id	Operation	Name	Rows	Bytes	Cost	(%CPU)	Time
0	SELECT STATEMENT		2921	96393	423	(1)	00:00:06
*	1 TABLE ACCESS FULL	CTEST	2921	96393	423	(1)	00:00:06

```
Predicate Information (identified by operation id):
```

```
1 - filter("COUNTRY_ID"=52779)
```

Statistics

```
2 recursive calls
1 db block gets
3292 consistent gets
0 physical reads
184 redo size
103199 bytes sent via SQL*Net to client
3349 bytes received via SQL*Net from client
257 SQL*Net roundtrips to/from client
0 sorts (memory)
0 sorts (disk)
3833 rows processed
```

```
SQL>
SQL> SET AUTOTRACE OFF
SQL> SET ECHO OFF
SQL>
```

What was the estimated number of rows? 2921

What was the actual number of rows fetched? 3833

What was the number of logical reads? 3293 (*DB block gets + consistent gets*)

- Gather statistics against the table and indexes. Use the GATHER_TABLE_STATS procedure.

```
SQL> EXEC DBMS_STATS.GATHER_TABLE_STATS('SH','CTEST');
PL/SQL procedure successfully completed.

SQL>
```

4. Enable **AUTOTRACE** for the SQL statement by using the `bad_stats.sql` script, and observe the differences in the execution plan.

```

SQL> @bad_stats
SQL> SET AUTOTRACE TRACEONL
SQL>
SQL> SELECT CUST_ID, CUST_LAST_NAME, CUST_TOTAL
  2  FROM CTEST
  3  WHERE COUNTRY_ID = 52779;

3833 rows selected.

Execution Plan
-----
Plan hash value: 3334397676

-----
| Id  | Operation          | Name      | Rows  | Bytes | Cost (%CPU)
| Time |                      |           |       |       |       |
-----
|   0 | SELECT STATEMENT    |           | 3833 | 123K|  474  (0
| 00:00:01 |
|   1 |  TABLE ACCESS BY INDEX ROWID BATCHED| CTEST| 3833 | 123K| 474  (0
| 00:00:01 |
| * 2 |  INDEX RANGE SCAN     | CTEST_COUNTRY | 3833 |       | 10   (0
| 00:00:01 |

-----
Predicate Information (identified by operation id):
-----
2 - access ("COUNTRY_ID"=52779)

Statistics
-----
  0 recursive calls
  0 db block gets
 3268 consistent gets
 3016 physical reads
  0 redo size
103199 bytes sent via SQL*Net to client
 3349 bytes received via SQL*Net from client
  257 SQL*Net roundtrips to/from client
    0 sorts (memory)
    0 sorts (disk)
 3833 rows processed

```

```
SQL>
SQL> SET AUTOTRACE OFF
SQL> SET ECHO OFF
SQL>
```

What was the number of rows estimated? 3833

What was the number of rows fetched? 3833

What was the number of logical reads? 3268

The number of logical reads is a rough measure of the cost of the statement. There should be a small reduction in the number of logical reads between this execution and the previous execution of this statement.

5. Clean up from this practice by executing the `cleanup_bad_stats.sql` script.

```
SQL> @cleanup_bad_stats.sql

SQL> DROP SEQUENCE ctest_pk_seq;

Sequence dropped.

SQL>
SQL> DROP TABLE CTEST PURGE;

Table dropped.

SQL>
SQL> EXIT
$
```

6. In this exercise, stale statistics cause the optimizer to choose a full table scan (FTS) rather than an index access. This example shows a small cost reduction. The differences in cost can be magnified by repeated execution of the same statements and by larger differences in the statistics.

Practices for Lesson 12: Reducing the Cost of SQL Operations

Chapter 12

Practices for Lesson 12: Overview

Practices Overview

In this practice, you reduce the cost of queries by reducing the number of blocks that must be retrieved.

Practice 12-1: Excess Blocks

In this practice, you see the differences in the explain plan and the costs of using a table that has more blocks than necessary. This condition can be caused in a number of ways: direct loads followed by deletes, inserts and deletes with a high PCTFREE setting, and batch deletes. The immediate solution is to shrink or reorganize the table; the long-term solution is to modify the processing or PCTUSED so that blocks are reused for inserts.

1. Create a table that uses more blocks than necessary. Use the `cr_ctest.sql` script as the SH user. This script creates a table with the same data as the SH.CUSTOMERS table, but with PCTFREE=80, so that it uses many more blocks than needed.

```
$ cd $HOME/labs/sqltune
$ sqlplus /nolog

SQL> connect sh/sh
Connected.

SQL> @cr_ctest.sql
SQL>
SQL> DROP TABLE CTEST PURGE;
DROP TABLE CTEST PURGE
*
ERROR at line 1:
ORA-00942: table or view does not exist

SQL>
SQL> CREATE TABLE CTEST
  2  PCTFREE 80 PCTUSED 10
  3  as SELECT * FROM customers;

Table created.

SQL>
SQL> EXEC DBMS_STATS.GATHER_TABLE_STATS('SH', 'CTEST');

PL/SQL procedure successfully completed.

SQL>
```

2. Execute the `ctest.sql` script as the SH user.

```
SQL> @ctest.sql
SQL>
SQL> -- set autotrace on --
SQL> set echo on
SQL>
SQL> SET AUTOTRACE TRACEONLY
SQL>
SQL> select time_id, QUANTITY SOLD, AMOUNT SOLD
  2  from sales s, ctest c
  3  where c.cust_id = s.cust_id and CUST_FIRST_NAME='Dina'
  4  order by time_id;
```

```
895 rows selected.
```

Execution Plan

```
Plan hash value: 2054825740
```

Pstart	Pstop	Id	Operation	Name	Rows	Bytes	Cost	(%CPU)	Time
00:00:01		0	SELECT STATEMENT		5557	179K	2480	(1)	
00:00:01		1	SORT ORDER BY		5557	179K	2480	(1)	
00:00:01		2	HASH JOIN		5557	179K	2480	(1)	
00:00:01		3	TABLE ACCESS FULL	CTEST	43	516	1962	(1)	
00:00:01		4	PARTITION RANGE ALL		918K	18M	516	(2)	
00:00:01		1		28					
00:00:01		5	TABLE ACCESS FULL	SALES	918K	18M	516	(2)	
00:00:01		1		28					

Predicate Information (identified by operation id):

```
2 - access("C"."CUST_ID"="S"."CUST_ID")
3 - filter("CUST_FIRST_NAME">'Dina')
```

Statistics

```
1807 recursive calls
```

```

      0 db block gets
  11783 consistent gets
    8826 physical reads
      0 redo size
 26967 bytes sent via SQL*Net to client
  1193 bytes received via SQL*Net from client
    61 SQL*Net roundtrips to/from client
   123 sorts (memory)
     0 sorts (disk)
   895 rows processed

SQL>
SQL> SET AUTOTRACE OFF
SQL>
SQL>
```

Note the cost, both in terms of the number of physical reads and logical reads (consistent gets + DB block gets), and the estimated cost.

Physical reads: 8826, *may vary*

Logical reads: 11783, *may vary*

Estimated cost: 2480

- Reorganize the table to use fewer blocks. Execute the `reorg_ctest.sql` script. This script sets PCTFREE to 10, and then moves the table. This script also checks the number of blocks used before and after the reorganization.

```

SQL> @reorg_ctest.sql
SQL>
SQL> -- Rebuild the table with higher density
SQL> SET ECHO ON
SQL>
SQL> SELECT BLOCKS, EMPTY_BLOCKS FROM USER_TABLES
  2 WHERE TABLE_NAME = 'CTEST';

  BLOCKS  EMPTY_BLOCKS
-----  -----
    7229          0

SQL>
SQL> ALTER TABLE CTEST
  2 PCTFREE 10;

Table altered.

SQL>
SQL> ALTER TABLE CTEST MOVE;

Table altered.

SQL>
SQL> EXEC DBMS_STATS.GATHER_TABLE_STATS('SH','CTEST');

PL/SQL procedure successfully completed.
```

```

SQL>
SQL> SELECT BLOCKS, EMPTY_BLOCKS FROM USER_TABLES
  2 WHERE TABLE_NAME = 'CTEST';

      BLOCKS  EMPTY_BLOCKS
-----  -----
        1550          0

SQL>

```

4. Execute the `ctest.sql` script as the `SH` user again.

```

SQL> @ctest.sql
SQL>
SQL> -- set autotrace on --
SQL> set echo on
SQL>
SQL> SET AUTOTRACE TRACEONLY
SQL>
SQL> select time_id, QUANTITY_SOLD, AMOUNT_SOLD
  2   from sales s, ctest c
  3   where c.cust_id = s.cust_id and CUST_FIRST_NAME='Dina'
  4   order by time_id;

895 rows selected.

```

Execution Plan

```
Plan hash value: 2054825740
```

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
Pstart	Pstop					
<hr/>						
0	SELECT STATEMENT		5557	179K	941 (1)	00:00:01
1	SORT ORDER BY		5557	179K	941 (1)	00:00:01
* 2	HASH JOIN		5557	179K	941 (1)	00:00:01
* 3	TABLE ACCESS FULL	CTEST	43	516	423 (1)	00:00:01

4	PARTITION RANGE ALL	918K	18M	516	(2)
00:00:01	1	28			

5	TABLE ACCESS FULL	SALES	918K	18M	516	(2)
00:00:01	1	28				

Predicate Information (identified by operation id):

```

2 - access("C"."CUST_ID"="S"."CUST_ID")
3 - filter("CUST_FIRST_NAME"='Dina')

```

Note

- this is an adaptive plan

Statistics

```

1 recursive calls
0 db block gets
3139 consistent gets
3120 physical reads
0 redo size
21271 bytes sent via SQL*Net to client
1193 bytes received via SQL*Net from client
61 SQL*Net roundtrips to/from client
1 sorts (memory)
0 sorts (disk)
895 rows processed

```

```

SQL>
SQL> SET AUTOTRACE OFF
SQL>
SQL>

```

Note the various costs. Determine the number of physical reads, logical reads, and the estimated cost.

Physical reads: 3120, *may vary*

Logical reads (consistent gets + db block gets): 3139, *may vary*

Estimated cost: 941

The actual numbers that you see may vary from those in the example, but you should see a reduction in the number of logical and physical reads, and the estimated cost. These reductions are due to reducing the number of physical blocks used to hold the table rows. This effect is most pronounced with full table scans. But even with index access methods, if the table uses more blocks, it will require more memory buffers and more block visits to access the same number of rows.

5. Drop the CTEST table that was created for this practice. Log out of SQL*Plus.

```
SQL> DROP TABLE CTEST PURGE;  
SQL> EXIT
```

Practice 12-2: Coalescing an Index

In this practice, you observe the effect of coalescing an index on a SQL statement and an execution plan.

1. The `coalesce_setup.sql` script creates the TESTLM tablespace, the CTEST table, and the CTEST_PK index, with excess space in the index. CTEST is created as an empty version of the SH.CUSTOMERS table, and then populated with four times the rows from CUSTOMERS. To make the index sparse, every second row is deleted based on CUST_ID, which is unique and is generated by a sequence.

```
$ cd $HOME/labs/sqltune
$ sqlplus /nolog

SQL> @coalesce_setup.sql

SQL> CONNECT / as sysdba
Connected.

SQL>
SQL> DROP TABLESPACE TESTLM INCLUDING CONTENTS AND DATAFILES;
DROP TABLESPACE TESTLM INCLUDING CONTENTS AND DATAFILES
*
ERROR at line 1:
ORA-00959: tablespace 'TESTLM' does not exist


SQL>
SQL> CREATE TABLESPACE TESTLM
  2  DATAFILE '/u01/app/oracle/oradata/orcl/testlm01.dbf'
  3  SIZE 100M
  4  AUTOEXTEND ON NEXT 10M MAXSIZE 300M
  5  SEGMENT SPACE MANAGEMENT MANUAL;

Tablespace created.

SQL>
SQL> CONNECT sh/sh
Connected.

SQL>
SQL> -- Create a table with indexes that have large number of
SQL> -- partially empty blocks and will benefit from an index
SQL> -- coalesce
SQL>
SQL> -- Create a table, indexes and gather statistics
SQL> DROP SEQUENCE ctest_pk_seq;
DROP SEQUENCE ctest_pk_seq
*
ERROR at line 1:
ORA-02289: sequence does not exist


SQL> DROP TABLE CTEST PURGE;
DROP TABLE CTEST PURGE
```

```

*
ERROR at line 1:
ORA-00942: table or view does not exist

SQL>
SQL> CREATE TABLE CTEST TABLESPACE TESTLM
  2 AS SELECT *
  3   FROM CUSTOMERS where 1=2;

Table created.

SQL>
SQL> ALTER TABLE CTEST ADD Constraint CTEST_PK
  2           Primary KEY (cust_id);

Table altered.

SQL> ALTER TABLE CTEST ADD (DUMMY VARCHAR2(300));

Table altered.

SQL>
SQL>
SQL> CREATE SEQUENCE ctest_pk_seq START WITH 200000 NOMAXVALUE;

Sequence created.

SQL>
SQL> INSERT INTO CTEST
  2   SELECT ctest_pk_seq.nextval, CUST_FIRST_NAME,
  3         CUST_LAST_NAME,CUST_GENDER, CUST_YEAR_OF_BIRTH,
  4         CUST_MARITAL_STATUS,CUST_STREET_ADDRESS,CUST_POSTAL_CODE,
  5         CUST_CITY,CUST_CITY_ID,CUST_STATE_PROVINCE,
  6         CUST_STATE_PROVINCE_ID,COUNTRY_ID,CUST_MAIN_PHONE_NUMBER,
  7         CUST_INCOME_LEVEL,CUST_CREDIT_LIMIT,CUST_EMAIL,
  8         CUST_TOTAL,CUST_TOTAL_ID,CUST_SRC_ID,CUST_EFF_FROM,
  9         CUST_EFF_TO,CUST_VALID,
 10        CUST_FIRST_NAME||' '||CUST_LAST_NAME
 11   FROM CUSTOMERS;

55500 rows created.

SQL>
SQL>
SQL> INSERT INTO CTEST
  2   SELECT ctest_pk_seq.nextval, CUST_FIRST_NAME,
  3         CUST_LAST_NAME,CUST_GENDER, CUST_YEAR_OF_BIRTH,
  4         CUST_MARITAL_STATUS,CUST_STREET_ADDRESS,CUST_POSTAL_CODE,
  5         CUST_CITY,CUST_CITY_ID,CUST_STATE_PROVINCE,
  6         CUST_STATE_PROVINCE_ID,COUNTRY_ID,CUST_MAIN_PHONE_NUMBER,
  7         CUST_INCOME_LEVEL,CUST_CREDIT_LIMIT,CUST_EMAIL,
```

```

8      CUST_TOTAL,CUST_TOTAL_ID,CUST_SRC_ID,CUST_EFF_FROM,
9      CUST_EFF_TO,CUST_VALID,
10     CUST_FIRST_NAME||' '||CUST_LAST_NAME
11    FROM CUSTOMERS;

55500 rows created.

SQL>
SQL>
SQL> INSERT INTO CTEST
2  SELECT ctest_pk_seq.nextval, CUST_FIRST_NAME,
3        CUST_LAST_NAME,CUST_GENDER, CUST_YEAR_OF_BIRTH,
4        CUST_MARITAL_STATUS,CUST_STREET_ADDRESS,CUST_POSTAL_CODE,
5        CUST_CITY,CUST_CITY_ID,CUST_STATE_PROVINCE,
6        CUST_STATE_PROVINCE_ID,COUNTRY_ID,CUST_MAIN_PHONE_NUMBER,
7        CUST_INCOME_LEVEL,CUST_CREDIT_LIMIT,CUST_EMAIL,
8        CUST_TOTAL,CUST_TOTAL_ID,CUST_SRC_ID,CUST_EFF_FROM,
9        CUST_EFF_TO,CUST_VALID,
10       CUST_FIRST_NAME||' '||CUST_LAST_NAME
11      FROM CUSTOMERS;

55500 rows created.

SQL>
SQL>
SQL> INSERT INTO CTEST
2  SELECT ctest_pk_seq.nextval, CUST_FIRST_NAME,
3        CUST_LAST_NAME,CUST_GENDER, CUST_YEAR_OF_BIRTH,
4        CUST_MARITAL_STATUS,CUST_STREET_ADDRESS,CUST_POSTAL_CODE,
5        CUST_CITY,CUST_CITY_ID,CUST_STATE_PROVINCE,
6        CUST_STATE_PROVINCE_ID,COUNTRY_ID,CUST_MAIN_PHONE_NUMBER,
7        CUST_INCOME_LEVEL,CUST_CREDIT_LIMIT,CUST_EMAIL,
8        CUST_TOTAL,CUST_TOTAL_ID,CUST_SRC_ID,CUST_EFF_FROM,
9        CUST_EFF_TO,CUST_VALID,
10       CUST_FIRST_NAME||' '||CUST_LAST_NAME
11      FROM CUSTOMERS;

55500 rows created.

SQL>
SQL> DELETE FROM CTEST
2 WHERE mod(cust_id,2) = 1;

111000 rows deleted.

SQL>
SQL> commit;

Commit complete.

SQL>
SQL> EXEC DBMS_STATS.GATHER_TABLE_STATS ('SH','CTEST', -

```

```
>     method_opt=>'FOR ALL INDEXED COLUMNS SIZE AUTO') ;
PL/SQL procedure successfully completed.

SQL>
```

2. Execute the `coalesce_idx_1.sql` script. The script sets AUTOTRACE TRACEONLY.
 What is the access method? *Full table scan*
 What is the estimated cost? *1760*
 What is the number of consistent reads? *8343, may vary*

```
SQL> @coalesce_idx_1.sql
SQL> -- Will yield different plans before and after index coalesce
SQL>
SQL> SET AUTOTRACE TRACEONLY
SQL>
SQL> select /* BEFORE */ cust_id, cust_last_name, cust_first_name
  2   from ctest
  3  where cust_id < 257500
  4 /

28750 rows selected.
```

Execution Plan

```
-----  
Plan hash value: 3591594933
```

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		28750	1094K	1760	(1)
* 1	TABLE ACCESS FULL	CTEST	28750	1094K	1760	(1)

Predicate Information (identified by operation id):

```
-----  
1 - filter("CUST_ID"<257500)
```

Statistics

```
-----  
1 recursive calls  
0 db block gets  
8343 consistent gets  
6484 physical reads  
0 redo size  
736287 bytes sent via SQL*Net to client  
21620 bytes received via SQL*Net from client
```

```

1918  SQL*Net roundtrips to/from client
      0  sorts (memory)
      0  sorts (disk)
  28750  rows processed

```

```

SQL>
SQL> SET AUTOTRACE OFF
SQL>

```

3. Execute the coalesce_idx_size.sql script to view index statistics.

```

SQL> @coalesce_idx_size
SQL> -- Find the important index stats before and
SQL> -- after coalesce
SQL>
SQL> SELECT Blevel, LEAF_BLOCKS, DISTINCT_KEYS, CLUSTERING_FACTOR
  2  FROM USER_INDEXES
  3 WHERE index_name = 'CTEST_PK';

    BLEVEL LEAF_BLOCKS DISTINCT_KEYS CLUSTERING_FACTOR
----- -----
        1          417       111000         6480

SQL>

```

4. Coalesce the CTEST_PK index by using the coalesce_fix.sql script.

```

SQL> @coalesce_fix
SQL> -- Fix the index (Coalesce)
SQL>
SQL> ALTER INDEX CTEST_PK COALESCE;

Index altered.

SQL>
SQL> -- GATHER STATS
SQL>
SQL> EXEC DBMS_STATS.GATHER_TABLE_STATS('SH','CTEST',-
  2  METHOD_OPT=>'FOR ALL INDEXED COLUMNS SIZE AUTO');

PL/SQL procedure successfully completed.

SQL>

```

5. View index statistics again by using the coalesce_idx_size.sql script. Notice the difference in the number of leaf blocks.

```

SQL> @coalesce_idx_size
SQL> -- Find the important index stats before and
SQL> -- after coalesce
SQL>
SQL> SELECT Blevel, LEAF_BLOCKS, DISTINCT_KEYS, CLUSTERING_FACTOR
  2  FROM USER_INDEXES
  3 WHERE index_name = 'CTEST_PK';

    BLEVEL LEAF_BLOCKS DISTINCT_KEYS CLUSTERING_FACTOR
----- -----

```

```

-----  

1          232        111000        6480  

SQL>

```

6. Execute the `coalesce_idx.sql` script.

```

SQL> @coalesce_idx_2.sql
SQL> -- Will yield different plans before and after index coalesce
SQL>
SQL> Set AUTOTRACE TRACEONLY
SQL> select /*AFTER*/ cust_id, cust_last_name, cust_first_name
  2   from ctest
  3  where cust_id < 257500
  4 /

```

28750 rows selected.

Execution Plan

```
-----  
Plan hash value: 1754492574
```

```

-----  

| Id  | Operation           | Name    | Rows  | Bytes | Cost  

(%  

CPU) | Time      |  

-----  

|   0 | SELECT STATEMENT    |         | 28750 | 1094K | 1742  

| (1)| 00:00:01  |  

|   1 | TABLE ACCESS BY INDEX ROWID BATCHED | CTEST    | 28750 |  

| 1094K| 1742  

| (1)| 00:00:01  |  

|*  2 | INDEX RANGE SCAN     | CTEST_PK | 28750 |  

| 62|  

| (0)| 00:00:01  |

```

Predicate Information (identified by operation id):

```
-----  
2 - access ("CUST_ID"<257500)
```

Statistics

```
1 recursive calls
0 db block gets
5454 consistent gets
1695 physical reads
0 redo size
748666 bytes sent via SQL*Net to client
21620 bytes received via SQL*Net from client
1918 SQL*Net roundtrips to/from client
0 sorts (memory)
0 sorts (disk)
28750 rows processed
```

```
SQL>
SQL> SET AUTOTRACE OFF
SQL>
```

Observe the differences in the estimated cost and the number of logical reads (consistent gets).

What is the access method? *Index range scan*

What is the estimated cost? *1742*

How many consistent reads were performed? *5454, may vary*

7. Log out of SQL*Plus and clean up from this practice by executing the `cleanup` script in the `$HOME/workshops` directory with parameters as shown.

```
SQL> exit
$ cd $HOME/workshops
$ ./cleanup OPT

Tablespace dropped
$
```


Practices for Lesson 13: Using SQL Performance Analyzer

Chapter 13

Practices for Lesson 13: Overview

Practices Overview

The goal of this practice is to use SQL Performance Analyzer to identify the SQL statements that regress when the environment is upgraded.

Practice 13-1: Using SQL Performance Analyzer

In this practice, you simulate exporting a SQL Tuning Set (STS) from an Oracle Database 11g and importing it back into an Oracle Database 12c test environment. There, you assess the performance of the SQL statements that you imported before upgrading the 11g database.

- From a terminal window, referred to as the “first session,” execute the `setup_SPA11g.sh` script to set up your simulated 11g environment. In this simulated environment, the `OPTIMIZER_FEATURES_ENABLED` parameter is set to 11.2.0.2.

```
$ cd $HOME/labs/SPA
$ cat setup_SPA11g.sh

#!/bin/bash

rm /u01/app/oracle/admin/orcl/dpdump/apps.dmp
rm /u01/app/oracle/admin/orcl/dpdump/appsandstage.dmp

cp /home/oracle/labs/SPA/apps.dmp
/u01/app/oracle/admin/orcl/dpdump/apps.dmp

sqlplus /nolog <<FIN!
connect / as sysdba

@$HOME/workshops/spwkshSPA.sql

shutdown immediate

startup

set echo on

exec dbms_sqltune.drop_sqlset('STS_JFV', 'SYS');

drop user apps cascade;

host impdp system/oracle_4U directory=DATA_PUMP_DIR
dumpfile=apps.dmp

select distinct last_analyzed from dba_tab_statistics where
owner='APPS';

EXECUTE DBMS_STATS.LOCK_SCHEMA_STATS('APPS');

ALTER DATABASE TEMPFILE '/u01/app/oracle/oradata/orcl/temp01.dbf'
AUTOEXTEND ON NEXT 6400K MAXSIZE UNLIMITED;

FIN!

$ ./setup_SPA11g.sh
```

2. **(Perform steps 2 and 3 at the same time.)** Generate a SQL Tuning Set (STS) called `STS_JFV` that captures SQL statements from the cursor cache for approximately five minutes every five seconds. Make sure that you try to capture only those statements that come out of the `SQL_JFV` module in the `APPS` schema. Also, this STS should belong to the `SYS` user. Use the `capsts11g.sh` script to perform this step.

```
$ cd /home/oracle/labs/SPA
$ cat capsts11g.sh
#!/bin/bash

sqlplus /nolog <<FIN!
connect / as sysdba

SET ECHO ON;
SET TIMING ON;

begin
DBMS_SQLTUNE.CREATE_SQLSET (sqlset_name  => 'STS_JFV');

dbms_sqltune.capture_cursor_cache_sqlset(
sqlset_name => 'STS_JFV' ,
basic_filter=> q'# module
like 'DWH_TEST%' and sql_text not like '%applicat%' and
parsing_schema_name in ('APPS') #' ,
time_limit  => 5*60,
repeat_interval => 5);
end ;
/
show errors

FIN!

$ ./capsts11g.sh
```

3. **(Perform steps 2 and 3 at the same time.)** From a second terminal window, connected as the `oracle` user, execute your workload by using the `wkld11g_jfv.sh` script. This script runs a workload of 45 statements that are going to be captured in `STS_JFV` automatically.

```
-- Second session
$ cd /home/oracle/labs/SPA
$ ./wkld11g_jfv.sh

...
SQL> @6 Results in 1296 Ticks
SQL> SQL> SQL> SQL> @Statement 45
SQL> SQL> SQL>   2     3     4     5     6     7     8     9     10    11    12
13    14    15    16    17    18    19    20    21    22    23    24    25    26
27    28    29    30    31    32    33    34    35    36    37    38    39    40
41    42    43    44    45    46    47    48    49    50    51    52    53    54
55    56    57    58    59    60    61    62    63    64    65    66    67    68
69    70    71    72    73    74    75    76    77    78    79    80    81    82
```

```

83    84    85    86    87    88    89    90    91    92    93    94    95    96
97    98    99   100   101   102   103   104
10 rows selected.

Elapsed: 00:00:30.24

Statistics
-----
      92 recursive calls
       0 db block gets
 6871559 consistent gets
 10770 physical reads
   116 redo size
 2003 bytes sent via SQL*Net to client
 3147 bytes received via SQL*Net from client
     2 SQL*Net roundtrips to/from client
     2 sorts (memory)
     0 sorts (disk)
    10 rows processed

SQL> @10 Results in 1188 Ticks
SQL> SQL> SQL> SQL>
PL/SQL procedure successfully completed.

SQL> SQL> we are done
SQL> SQL> SQL> @End 2010:FEB-11:18:50:40

SQL> SQL> Disconnected ...

```

4. After approximately five minutes, both sessions should have finished. Check the content of `STS_JFV`, and stage it in a table called `APPS.STS_JFV_TAB`. Use the `stage_sts.sh` script to perform this step.

```

$ cd /home/oracle/labs/SPA
$ cat stage_sts.sh

#!/bin/bash
sqlplus /nolog <<FIN!

connect / as sysdba

set echo on
select name,statement_count from dba_sqlset;

drop table apps.sts_jfv_tab purge;

exec DBMS_SQLTUNE.CREATE_STGTAB_SQLSET('STS_JFV_TAB', 'APPS');

exec
DBMS_SQLTUNE.PACK_STGTAB_SQLSET('STS_JFV', 'SYS', 'STS_JFV_TAB', 'APPS'
);

FIN!

```

```
$ ./stage_sts.sh
```

- Using Data Pump Export, export the APPS schema to the default Data Pump directory. Use the `exportapps.sh` script to perform this step.

```
$ cd /home/oracle/labs/SPA
$ cat exportapps.sh

#!/bin/bash

sqlplus /nolog <<FIN!
connect apps/apps

drop table plan_table purge;

FIN!

rm /u01/app/oracle/admin/orcl/dpdump/appsandstage.dmp

expdp system/oracle_4U directory=DATA_PUMP_DIR dumpfile=appsandstage
schemas=apps

$ ./exportapps.sh
```

- In your site environments, you would copy the generated dump file from the default Data Pump directory in the source database to the default Data Pump directory in the target. But in this simulated environment, they are the same directory. So nothing needs to be done.
- Now, restart your 11g environment to restore the database to a 12c environment. Use the `setup_SPA12c.sh` script to perform this step.

```
$ ./setup_SPA12c.sh

SQL> SQL> Connected.
SQL> SQL> Database closed.
Database dismounted.
ORACLE instance shut down.
SQL>
SQL> ORACLE instance started.

Total System Global Area  835104768 bytes
Fixed Size                  2293880 bytes
Variable Size                792727432 bytes
Database Buffers              33554432 bytes
Redo Buffers                  6529024 bytes
Database mounted.
Database opened.
SQL> SQL> Disconnected ...
```

8. Using Data Pump Import, import the APPS schema into your 12c system. Use the importapps.sh script to perform this step.

```
$ cd /home/oracle/labs/SPA
$ cat importapps.sh

#!/bin/bash

sqlplus /nolog <<FIN!

Connect / as sysdba
drop user apps cascade;

host impdp system/oracle_4U DIRECTORY=DATA_PUMP_DIR
DUMPFILE=appsandstage

FIN!

$ ./importapps.sh
```

9. Unpack the previously imported staging table in the SYS schema. Use the unpack_sts.sh script to perform this step.

```
$ cd /home/oracle/labs/SPA
$ cat unpack_sts.sh

#!/bin/bash

sqlplus /nolog <<FIN!

Connect / as sysdba
set echo on

exec
DBMS_SQLTUNE.UNPACK_STGTAB_SQLSET('STS_JFV', 'SYS', TRUE, 'STS_JFV_TAB'
, 'APPS');

alter system flush buffer_cache;
alter system flush shared_pool;

FIN!

$ ./unpack_sts.sh
```

10. Use Enterprise Manager Cloud Control, as the SYS user, to test the behavior of STS_JFV in the simulated 11g environment and compare it to the 12c environment. You will do this by changing the OPTIMIZER_FEATURES_ENABLE parameter. What are your conclusions?

Note: Screenshots of some of the steps described in the table are provided, as noted in the Page column, below the table. Refer to these screenshots as necessary.

Step	Window/Page Description	Choices or Values																				
a.	EMCC login	User Name: sysman Password: oracle_4U																				
b.	Enterprise Summary	Targets > databases																				
c.	Databases	Click orcl .																				
d.	orcl Database Home page	Click Performance > SQL > SQL Performance Analyzer .																				
e.	Database Login	Select Preferred and SYSDBA Database Credentials . Click Login .																				
f.	SQL Performance Analyzer	Click Parameter Change .																				
g.	Parameter Change (See the example screenshot, which follows.)	<p>Set values.</p> <table border="1"> <thead> <tr> <th>Field</th><th>Value</th></tr> </thead> <tbody> <tr> <td>Task Name</td><td>SPA_JFV1</td></tr> <tr> <td>SQL Tuning Set</td><td>SYS.STS_JFV</td></tr> <tr> <td>Creation Method</td><td>Execute SQLs</td></tr> <tr> <td>Per-SQL Time Limit</td><td>Unlimited</td></tr> <tr> <td>Parameter Name</td><td>optimizer_features_enable (click the flashlight icon to select)</td></tr> <tr> <td>Base Value</td><td>11.2.0.2</td></tr> <tr> <td>Changed Value</td><td>12.1.0.1</td></tr> <tr> <td>Comparison Metric</td><td>Elapsed Time</td></tr> <tr> <td>Schedule</td><td>Immediately</td></tr> </tbody> </table> <p>Click Submit.</p>	Field	Value	Task Name	SPA_JFV1	SQL Tuning Set	SYS.STS_JFV	Creation Method	Execute SQLs	Per-SQL Time Limit	Unlimited	Parameter Name	optimizer_features_enable (click the flashlight icon to select)	Base Value	11.2.0.2	Changed Value	12.1.0.1	Comparison Metric	Elapsed Time	Schedule	Immediately
Field	Value																					
Task Name	SPA_JFV1																					
SQL Tuning Set	SYS.STS_JFV																					
Creation Method	Execute SQLs																					
Per-SQL Time Limit	Unlimited																					
Parameter Name	optimizer_features_enable (click the flashlight icon to select)																					
Base Value	11.2.0.2																					
Changed Value	12.1.0.1																					
Comparison Metric	Elapsed Time																					
Schedule	Immediately																					

Parameter Change

Task Information

* Task Name SPA_JFV1

* SQL Tuning Set SYS.STS_JFV

Description

Creation Method Execute SQLs

Per-SQL Time Limit Unlimited

TIP Time limit is on elapsed time of test execution of SQL.

Parameter Change

* Parameter Name optimizer_features_enable

* Base Value 11.2.0.2

* Changed Value 12.1.0.1

Trial Comparison

Comparison Metric Elapsed Time

Schedule

Time Zone UTC

Immediately

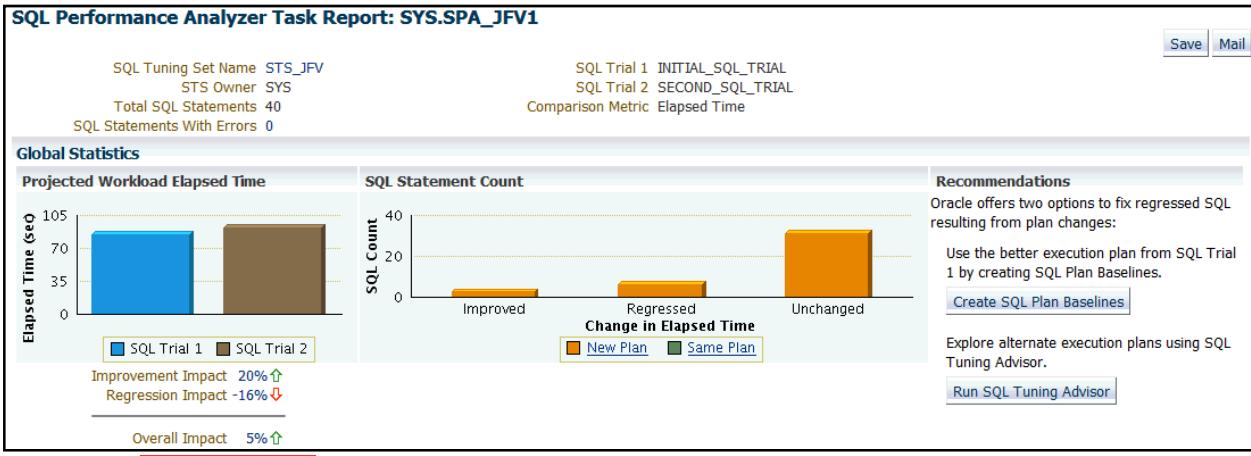
Later

Date Sep 16, 2013
(example: Sep 16, 2013)

Time 2 18 00 AM

Step	Window/Page Description	Choices or Values
h.	SQL Performance Analyzer	Let the screen refresh until the Last Run Status field shows Completed for your submitted task (SPA_JFV1). It may take up to 10 minutes to execute. Click the SPA_JFV1 link.
i.	SQL Performance Analyzer Task: SYS.SPA_JFV1	This page shows the SQL Trials and the SQL Trial Comparisons. Click the eyeglass icon in the SQL Trial Comparisons table in the Comparison Report column. There should be only one entry in this table.
j.	SQL Performance Analyzer Task Report: SYS.SPA_JFV1.	The graph shows which trial was faster. The second trial corresponds to the run that used the 12c optimizer features. The improvement or regression depends on the statements that were captured and executed. You will see the number of the statements that use a new plan or same plan with 12c optimization. However, some statements regressed with 12c optimization.

Step	Window/Page Description	Choices or Values
		Note: If the graph is not clear, check the Overall Impact value. It is 20% in the example below.

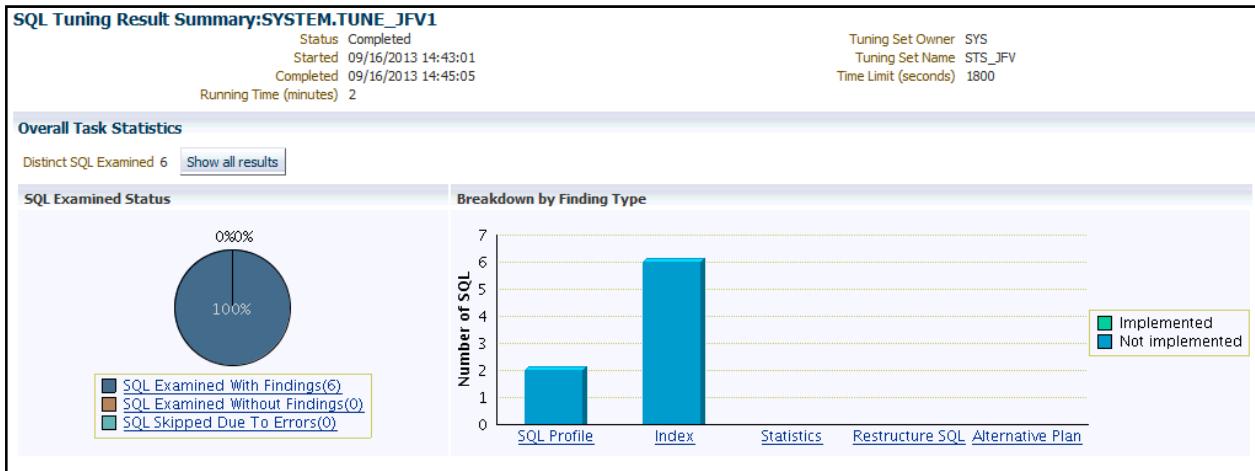


Step	Window/Page Description	Choices or Values
k.	SQL Performance Analyzer Task Report: SYS.SPA_JFV1	<p>From “Top 10 SQL Statements Based on Impact on Workload,” make a note of the SQL ID corresponding to the statements that have the highest negative value in the column labeled Net Impact on (SQL%). You use this information in a later step.</p> <p>Click the SQL ID link corresponding to the most regressed one. In the example below, it is 9wgj8btxt2122.</p>

Top 10 SQL Statements Based on Impact on Workload					
	SQL ID	Net Impact on Workload (%)	Elapsed Time (sec)		Net Impact on SQL (%)
			SQL Trial 1	SQL Trial 2	
↑	gkcgw4zhaysbk	16.780	78.067	63.478	18.690
↓	9wgj8btxt2122	-2.750	0.003	2.393	-78,478.650
↓	0mcv88ft7fwdj	-2.720	0.077	2.440	-3,069.170
↓	17mnbug99320v	-2.710	0.003	2.363	-80,015.330
↓	amny7vwjss6pp	-2.450	0.445	2.574	-477.870

Step	Window/Page Description	Choices or Values
l.	SQL Details	<p>You can see the text of your statement Compare the different execution plans that were used during both trials.</p> <p>Click SQL Performance Analyzer Task Report: SYS.SPA_JFV1 in the breadcrumb path at the top of the page.</p>
m.	SQL Performance Analyzer Task Report: SYS.SPA_JFV1	Click Run SQL Tuning Advisor in the Recommendations section.

Step	Window/Page Description	Choices or Values						
n.	Schedule SQL Tuning Task	<p>Set values.</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th>Field</th><th>Value</th></tr> </thead> <tbody> <tr> <td>Tuning Task Name</td><td>TUNE_JFV1</td></tr> <tr> <td>Schedule</td><td>Immediately</td></tr> </tbody> </table> <p>Click OK.</p>	Field	Value	Tuning Task Name	TUNE_JFV1	Schedule	Immediately
Field	Value							
Tuning Task Name	TUNE_JFV1							
Schedule	Immediately							
o.	SQL Performance Analyzer Task Report: SYS.SPA_JFV1	<p>Click the SQL Tune Report SYS.TUNE_JFV1 link in the Recommendations section.</p> <p>Use the browser refresh button until you see a report that includes graphics. This operation can take about two minutes. An error message may appear until the report generation is complete.</p>						
p.	SQL Tuning Result Summary:SYS.TUNE_JFV1 (see screenshot)	<p>This page will show you that there are SQL Statements that can be improved, and the methods you can use: profiles, indexes, or alternative SQL plans (using SQL Baselines). Using one of these recommendations, you can reduce or eliminate the regressions.</p> <p>Do not implement at this time; the following lessons explore these options in more detail.</p>						



Practices for Lesson 14: SQL Performance Management

Chapter 14

Practices for Lesson 14: Overview

Practices Overview

SQL Plan Management (SPM) is an Oracle Database feature that provides controlled execution plan evolution.

With SPM, the optimizer automatically manages execution plans and ensures that only known or verified plans are used.

When a new plan is found for a SQL statement, it will not be used until it has been verified to have comparable or better performance than the current plan.

SPM has three main components:

- Plan Capture
- Plan Selection
- Plan Verification

In these practices, you will see each of these components in action.

Practice 14-1: Seeding SQL Plan Baselines from SQL Performance Analyzer

In this practice, you use the result of your SQL Performance Analyzer comparison trial to seed SQL plan baselines for regressing SQL statements to avoid future plan regressions.

The regressions could have come from a variety of sources. Regressions could be due to changes in statistics or changes in database parameters.

1. Use Enterprise Manager Cloud Control to check whether there are baselines for your most regressing statements. Then use Enterprise Manager Cloud Control to seed SQL Plan Baselines for your most regressing statements.

Step	Window/Page Description	Choices or Values
a.	Login	Log in as the SYSMAN user with the password oracle_4U .
b.	Summary	Expand the Targets menu and select Databases .
c.		Click the link for the orcl database.
d.	orcl database home	Expand the Performance menu. Expand the SQL menu and select SQL Plan Control .
e.	Database Login	Select Preferred and choose SYSDBA Database Credentials. Click Login .
f.	SQL Plan Control	Click the SQL Plan Baseline tab. You should not see any baseline for your regressing statements.
g.	SQL Plan Control: SQL Plan Baseline tab	In the Settings section, click the FALSE value for Capture SQL Plan Baselines.
h.	Initialization Parameters	Check Apply changes in current running instance(s) mode to SPFile . Set the value of the parameter to TRUE . Click OK .
i.	SQL Plan Control: SQL Plan Baseline tab	In the Settings section, click the FALSE value for Use SQL Plan Baselines.
j.	Initialization Parameters	Check Apply changes in current running instance(s) mode to SPFile . Set the value of the parameter to TRUE . Click OK .
k.		Expand the Performance menu. Expand the SQL menu and select SQL Performance Analyzer .
l.	SQL Performance Analyzer	Click the SPA_JFV1 link in the SQL Performance Analyzer Tasks table.
m.	SQL Performance Analyzer Task: SYS.SPA_JFV1	Click the eyeglass icon in the Comparison Report column for your first replay trial comparison. Or, you can click the View Latest Report link at the top of the page to view the same report, when there is only

Step	Window/Page Description	Choices or Values
		one Comparison Report.
n.	SQL Performance Analyzer Task Report: SYS.SPA_JFV1	Click Create SQL Plan Baselines in the Recommendations section.
o.	SQL Performance Analyzer Task Report: SYS.SPA_JFV1 (See the example screenshot, which follows the table.)	You should now see all regressing statements that meet the criteria. If the impact is less than a certain threshold, they will not be listed. Set values: Job Name: baseline_jfv1 Schedule: Immediately Click OK .
p.	SQL Performance Analyzer Task Report: SYS.SPA_JFV1	You should see an Information message, at the top of the page, saying that your job has been successfully submitted. Click Performance > SQL > SQL Plan Control .
q.	SQL Plan Control	Click the SQL Plan Baseline tab. You should now see the SQL plan baselines that were created. You should have one baseline for each regressing statement and possibly additional baselines for system-generated SQL statements.

- 1(o) The following screenshot displays the SQL Performance Analyzer Task Result: SYS.SPA_JFV1: Create Plan Baseline page:

Create SQL Plan Baselines

SQL Plan Baselines enable the optimizer to avoid performance regressions by requiring new plans to be at least as good as the better plans found in SQL trial 1.

Regressed New Plan SQL Statements

SQL ID	Net Impact on Workload (%)	Elapsed Time		Net Impact on SQL (%)	% of Workload	
		INITIAL_SQL_TRIAL	SECOND_SQL_TRIAL		INITIAL_SQL_TRIAL	SECOND_SQL_TRIAL
9wgj8btxt2122	-2.580	0.003	2.364	-78,255.490		
17mnbug99320v	-2.580	0.003	2.358	-79,053.610		
0mcv88ft7fwdj	-2.530	0.079	2.391	-2,915.600		
bxajng3zk2vn1	-2.280	0.398	2.477	-521.930		
amny7vwjss6pp	-2.230	0.447	2.483	-455.370		
3gshh31g2s0x8	-2.170	0.397	2.380	-498.890		

Job Parameters

Job Name: baseline_jfv1

Description:

Schedule

Immediately

Later

Time Zone: (UTC+00:00) Universal Time

Date: Sep 17, 2013 (example: Sep 17, 2013)

Time: 10:35:00 AM

2. Use Enterprise Manager to create a new SQL Performance Analyzer session to look at the difference between the previous 12c trial and a new one that takes into account the newly created baselines.

Note: Screenshots of some of the steps described in the table are provided, as noted in the Page column, below the table. Refer to these screenshots as necessary.

Step	Window/Page Description	Choices or Values
a.	SQL Plan Control: SQL Plan Baselines tab	Click Performance > SQL > SQL Performance Analyzer .
b.	SQL Performance Analyzer	Click the SPA_JFV1 link.
c.	SQL Performance Analyzer Task: SYS.SPA_JFV1	Click Create SQL Trial .
d.	Create SQL Trial	Set values: SQLTrial Name: after_baselines Select Trial environment established check box (on the lower right of the page) Per-SQL Time Limit: Unlimited Schedule: Immediately

Step	Window/Page Description	Choices or Values
		Click Submit .
e.	SQL Performance Analyzer Task: SYS.SPA_JFV1	Click Refresh until you see COMPLETED in the Status column for the AFTER_BASELINES trial. Click Run SQL Trial Comparison .
f.	Run SQL Trial Comparison (See the example screenshot, which follows the table.)	Set values: Trial 1 Name: INITIAL_SQL_TRIAL Trial 2 Name: AFTER_BASELINES Comparison Metric: Elapsed Time Schedule: Immediately Click Submit .
g.	SQL Performance Analyzer Task: SYS.SPA_JFV1	You should see a Confirmation message, and a new line in the SQL Trial Comparisons section of the page Click the corresponding eyeglass icon in the Comparison Report column.
h.	SQL Performance Analyzer Task Result: SYS.SPA_JFV1	You should no longer see any significant regressing statements, and all previously regressing statements should now be executing faster.

2f) Run SQL Trial Comparison.

Run SQL Trial Comparison

Task Name SYS.SPA_JFV1
 SQL Tuning Set SYS.STS_JFV

Trial 1 Name **INITIAL_SQL_TRIAL**
 Description parameter optimizer_features_enable set to '12.1.0.1'
 SQL Executed Yes

Trial 2 Name **AFTER_BASELINES**
 Description
 SQL Executed Yes

Comparison Metric Elapsed Time

Schedule

Time Zone UTC
 Immediately
 Later
 Date Sep 17, 2013 (example: Sep 17, 2013)
 Time 11 25 00 AM

- In a terminal window, run the `./cleanup 14 1` script. The cleanup script drops all baselines and resets the initialization parameters `optimizer_capture_sql_plan_baselines` and `optimizer_use_sql_plan_baselines` to FALSE.

```
$ cd $HOME/workshops
$ ./cleanup 14 1
```

Practice 14-2: SQL Plan Management (SPM)

In this practice, you see all phases of using SQL Plan Management.

- Before you can start this practice, you must set up a new user. Execute the `spm_setup.sh` script in the `$HOME/labs/SPM` directory to set up the environment for this practice. This script creates the `SPM` user that you use throughout the practice.

```
$ cd /home/oracle/labs/SPM
$ cat ./spm_setup.sh

#!/bin/bash

sqlplus /nolog <<FIN!
connect / as sysdba

set echo on

drop user spm cascade;

create user spm identified by spm;

grant dba to spm;

alter system flush shared_pool;

FIN!

$ ./spm_setup.sh
```

Automatic Plan Capture

- The first component of SPM is Plan Capture. There are two main ways to capture plans: automatically (at run time) and bulk load. In this practice, you turn on automatic plan capture so that the SPM repository is automatically populated for any repeatable SQL statement. Turn on automatic plan capture by setting the `OPTIMIZER_CAPTURE_SQL_PLAN_BASELINES` initialization parameter to `TRUE` in your SQL*Plus session, connected as the SPM user. After you have connected in your session, do not disconnect. Verify that the `*SQL_PLAN_BASELINES` parameter are set to `TRUE` (the default is `FALSE`). Set both to `TRUE` for your session. The `ALTER SESSION` commands are shown below.

```
$ sqlplus spm/spm

SQL> show parameter baseline

NAME                           TYPE        VALUE
-----
optimizer_capture_sql_plan_baselines boolean    TRUE
optimizer_use_sql_plan_baselines   boolean    FALSE

SQL> alter session set optimizer_capture_sql_plan_baselines = TRUE;
```

```
Session altered.
```

```
SQL> alter session set optimizer_use_sql_plan_baselines = TRUE;
```

```
Session altered.
```

```
SQL>
```

3. Use the `query1.sql` script to execute the following query:

```
select /*LOAD_AUTO*/ * from sh.sales
where quantity_sold > 40 order by prod_id;
```

```
SQL> @query1.sql
```

```
SQL> select /*LOAD_AUTO*/ * from sh.sales
2> where quantity_sold > 40 order by prod_id;
```

```
no rows selected
```

```
SQL>
```

4. Because this is the first time that you have seen this SQL statement, it is not yet repeatable so there is no plan baseline for it. To confirm this, use the `check_baselines.sql` script to check whether there are any plan baselines that exist for your statement.

```
-- Should not return any rows
```

```
SQL> @check_baselines.sql
SQL> set echo on
SQL> set pagesize 100
SQL>
SQL> column SQL_HANDLE format A22
SQL> column PLAN_NAME format A32
SQL>
SQL> select signature, sql_handle, plan_name,
2          origin, enabled, accepted, fixed, autopurge,sql_text
3  from dba_sql_plan_baselines
4  where sql_text like 'select /*LOAD_AUTO*/%';
no rows selected
```

5. Re-execute `query1.sql` script to execute the query again.

```
SQL> @query1.sql
SQL> set echo on
SQL>
SQL> select /*LOAD_AUTO*/ * from sh.sales
2  where quantity_sold > 40 order by prod_id;
no rows selected
SQL>
```

6. The SQL statement is now known to be repeatable and a plan baseline is automatically captured. Check whether the plan baseline was loaded for the previous statement by executing the `check_baselines.sql` script again. What do you observe?

You can see from the output that a baseline has been created and enabled for this SQL statement. You can also tell that this plan was captured automatically by checking the values of the ORIGIN column.

```
-- You should see one accepted entry

SQL> @check_baselines.sql
SQL> set echo on
SQL> set pagesize 100
SQL>
SQL> column SQL_HANDLE format A22
SQL> column PLAN_NAME format A32
SQL>
SQL> select signature, sql_handle, plan_name,
2       origin, enabled, accepted, fixed, autopurge,sql_text
3   from dba_sql_plan_baselines
4  where sql_text like 'select /*LOAD_AUTO*/%';

SIGNATURE SQL_HANDLE          PLAN_NAME
-----
ORIGIN      ENA ACC FIX AUT
-----
SQL_TEXT
-----
-----
8.0622E+18 SQL_6fe28d438dfc352f    SQL_PLAN_6zsnd8f6zsd9g54bc8843
AUTO-CAPTURE YES YES NO YES
select /*LOAD_AUTO*/ * from sh.sales
where quantity_sold > 40 order by prod_id
```

7. Change or alter the optimizer mode to use `FIRST_ROWS` optimization and execute your statement. Describe what happens.

- a. Change the optimizer mode to `FIRST_ROWS` and execute the `query1.sql` script. *This triggers the SQL statement to execute with a different plan.*

```
SQL> alter session set optimizer_mode = first_rows;

Session altered.

SQL> @query1.sql
SQL> set echo on
SQL>
SQL> select /*LOAD_AUTO*/ * from sh.sales
2   where quantity_sold > 40 order by prod_id;

no rows selected

SQL>
```

- b. Because the SQL statement will have a new plan, another plan baseline is automatically captured. You can confirm this by using the `check_baseline.sql` script.

```
SQL> @check_baseline.sql
SQL> set echo on
SQL> set pagesize 100
SQL>
SQL> column SQL_HANDLE format A22
SQL> column PLAN_NAME format A32
SQL>
SQL> select signature, sql_handle, plan_name,
2       origin, enabled, accepted, fixed, autopurge,sql_text
3   from dba_sql_plan_baselines
4  where sql_text like 'select /*LOAD_AUTO*/%';

SIGNATURE SQL_HANDLE          PLAN_NAME
-----
ORIGIN      ENA ACC FIX AUT
-----
SQL_TEXT
-----
8.0622E+18 SQL_6fe28d438dfc352f    SQL_PLAN_6zsnd8f6zsd9g54bc8843
AUTO-CAPTURE YES YES NO YES
select /*LOAD_AUTO*/ * from sh.sales
where quantity_sold > 40 order by prod_id

8.0622E+18 SQL_6fe28d438dfc352f    SQL_PLAN_6zsnd8f6zsd9gce885bc0
AUTO-CAPTURE YES NO NO YES
select /*LOAD_AUTO*/ * from sh.sales
where quantity_sold > 40 order by prod_id
```

Now you see two plan baselines for your query, but notice that the new plan has not been accepted. This new plan will have to be validated before it is acceptable as a good plan to use.

8. Reset the optimizer mode to the default values and disable autocapture of plan baselines.

```
SQL> alter session set optimizer_mode = all_rows;
Session altered.

SQL> alter session set optimizer_capture_sql_plan_baselines = FALSE;
Session altered.
```

9. Purge the plan baselines and confirm that the SQL plan baseline is empty by using the `purge_auto_baseline.sql` script.

```
SQL> @purge_auto_baseline.sql
SQL> set echo on
SQL>
SQL> variable cnt number;
SQL>
```

```

SQL> exec :cnt :=
dbms_spm.drop_sql_plan_baseline('SQL_6fe28d438dfc352f');

PL/SQL procedure successfully completed.

SQL>
SQL> select signature, sql_handle, sql_text, plan_name,
2          origin, enabled, accepted, fixed, autopurge
3    from dba_sql_plan_baselines
4   where sql_text like 'select /*LOAD_AUTO*/%';

no rows selected

SQL>
```

Loading Plans from SQL Tuning Sets

- Still connected in your SQL*Plus session, use `explain_query2.sql` to check the execution plan for the following SQL statement and then execute it by using `query2.sql`.

```

select /*LOAD_STS*/ * from sh.sales
where quantity_sold > 40 order by prod_id;
```

```

SQL> @explain_query2.sql

-- You should see a Full Table Scan

SQL> set echo on
SQL>
SQL> explain plan for
2  select /*LOAD_STS*/ * from sh.sales
3  where quantity_sold > 40 order by prod_id;

Explained.

SQL>
SQL> select * from table(dbms_xplan.display(null,null,'basic'));

PLAN_TABLE_OUTPUT
-----
Plan hash value: 3803407550

-----
| Id  | Operation           | Name   |
|---|---|
| 0  | SELECT STATEMENT   |        |
| 1  |  SORT ORDER BY     |        |
| 2  |   PARTITION RANGE ALL |
| 3  |     TABLE ACCESS FULL | SALES |
|---|---|
-----
```

10 rows selected.

```

SQL> @query2.sql
SQL> set echo on
SQL>
SQL> select /*LOAD_STS*/ * from sh.sales
  2 where quantity_sold > 40 order by prod_id;

no rows selected

SQL>
```

11. Alter the optimizer mode to use FIRST_ROWS optimization. Check the execution plan by using `explain_query2.sql` and execute the query by using `query2.sql`.

```

SQL> alter session set optimizer_mode = first_rows;

Session altered.

-- You should see a different plan
-- (Table Access By Local Index)

SQL> @explain_query2.sql
SQL> set echo on
SQL>
SQL> explain plan for
  2 select /*LOAD_STS*/ * from sh.sales
  3 where quantity_sold > 40 order by prod_id;

Explained.

SQL>
SQL> select * from table(dbms_xplan.display(null,null,'basic'));
PLAN_TABLE_OUTPUT
-----
Plan hash value: 1095194413

-----
| Id  | Operation          | Name   |
|---|---|
| 0  | SELECT STATEMENT   |        |
| 1  |  SORT ORDER BY     |        |
| 2  |   PARTITION RANGE ALL |
| 3  |     TABLE ACCESS BY LOCAL INDEX ROWID BATCHED | SALES |
| 4  |       BITMAP CONVERSION TO ROWIDS   |        |
| 5  |       BITMAP INDEX FULL SCAN    |        |
SALES_PROMO_BIX |

-----
12 rows selected.
SQL>
SQL> @query2.sql
SQL> set echo on
SQL>
```

```

SQL> select /*LOAD_STS*/ * from sh.sales
  2 where quantity_sold > 40 order by prod_id;
no rows selected

SQL>
```

12. Reset the optimizer mode to ALL_ROWS optimization.

```

SQL> alter session set optimizer_mode = all_rows;

Session altered.
```

13. Verify that there are no baseline plans for your statement by using the `check_baselines2.sql` script.

```

SQL> @check_baselines2.sql
SQL> set echo on
SQL>
SQL> select signature, sql_handle, sql_text, plan_name,
  2       origin, enabled, accepted, fixed, autopurge
  3   from dba_sql_plan_baselines
  4  where sql_text like 'select /*LOAD_STS*/%';
no rows selected

SQL>
```

14. Create a SQL tuning set that captures the SELECT statements that contain the LOAD_STS hint. These statements are in the cursor cache. The STS should be called SPM_STS and owned by the SPM user. Use the `catchup_sts.sql` script to capture these statements.

```

SQL> @catchup_sts.sql
SQL> set echo on
SQL>
SQL> exec sys.dbms_sqltune.create_sqlset(
  >   sqlset_name  => 'SPM_STS', sqlset_owner => 'SPM');

PL/SQL procedure successfully completed.

SQL>
SQL> DECLARE
  2   stscur    dbms_sqltune.sqlset_cursor;
  3   BEGIN
  4     OPEN stscur FOR
  5       SELECT VALUE(P)
  6         FROM TABLE(dbms_sqltune.select_cursor_cache(
  7           'sql_text like ''select /*LOAD_STS*/%''' ,
  8           null, null, null, null, null, null, 'ALL')) P;
  9
 10    -- populate the sqlset
 11    dbms_sqltune.load_sqlset(sqlset_name      => 'SPM_STS',
 12                           populate_cursor  => stscur,
 13                           sqlset_owner     => 'SPM');
 14  END;
 15 /
```

```
PL/SQL procedure successfully completed.

SQL>
```

15. Verify which SQL statements are in SPM_STS by using the `check_sts.sql` script. SPM_STS has your SQL statement with two different plans.

```
SQL> @check_sts.sql
SQL> set echo on
SQL>
SQL> select sql_text from dba_sqlset_statements
  2 where sqlset_name='SPM_STS';

SQL_TEXT
-----
select /*LOAD_STS*/ * from sh.sales
where quantity_sold > 40 order by prod_id

select /*LOAD_STS*/ * from sh.sales
where quantity_sold > 40 order by prod_id

SQL>
```

16. Use the `populate_baseline.sql` script to populate the plan baseline repository with the plans found in SPM_STS.

```
SQL> @populate_baseline.sql
SQL> set echo on
SQL>
SQL> variable cnt number;
SQL>
SQL> exec :cnt := dbms_spm.load_plans_from_sqlset(
  >           sqlset_name  => 'SPM_STS', -
  >           basic_filter => 'sql_text like ''select
/*LOAD_STS*/%'''';

PL/SQL procedure successfully completed.

SQL>
```

17. Confirm that the plan baselines are loaded by using the `check_baselines2.sql` script and note the value in the origin column. What do you observe?

You should see MANUAL-LOAD because you manually loaded these plans. Also note that this time, both the plans are accepted.

```
SQL> @check_baselines2.sql
SQL> set echo on
SQL>
SQL> select signature, sql_handle, sql_text, plan_name,
2       origin, enabled, accepted, fixed, autopurge
3   from dba_sql_plan_baselines
4  where sql_text like 'select /*LOAD_STS*/%';

SIGNATURE SQL_HANDLE
-----
SQL_TEXT
-----

PLAN_NAME          ORIGIN      ENA ACC FIX AUT
-----
1.2134E+19 SQL_a8632bd857a4a25e
select /*LOAD_STS*/ * from sh.sales
where quantity_sold > 40 order by prod_id
SQL_PLAN_ahstbv1bu98ky54bc8843  MANUAL-LOAD  YES YES NO  YES

1.2134E+19 SQL_a8632bd857a4a25e
select /*LOAD_STS*/ * from sh.sales
where quantity_sold > 40 order by prod_id
SQL_PLAN_ahstbv1bu98kyce885bc0  MANUAL-LOAD  YES YES NO  YES

SQL>
```

18. Purge the plan baselines and drop SPM_STS by using the `purge_sts_baseline.sql` script.

```
SQL> @purge_sts_baseline.sql
SQL> set echo on
SQL>
SQL> variable cnt number;
SQL>
SQL> exec :cnt := dbms_spm.drop_sql_plan_baseline(
->           'SQL_a8632bd857a4a25e');

PL/SQL procedure successfully completed.

SQL>
SQL> print cnt;

      CNT
-----
      2
```

```

SQL>
SQL> select signature, sql_handle, sql_text, plan_name,
2          origin, enabled, accepted, fixed, autopurge
3  from dba_sql_plan_baselines
4 where sql_text like 'select /*LOAD_STS*/%';

no rows selected

SQL>
SQL> exec sys.dbms_sqltune.drop_sqlset(
->           sqlset_name  => 'SPM_STS', -
->           sqlset_owner => 'SPM');

PL/SQL procedure successfully completed.

SQL>

```

Loading Plans from the Cursor Cache

19. Now, you see how to directly load plan baselines from the cursor cache. Before you begin, you need some SQL statements. Still connected in your SQL*Plus session, use the `explain_query3.sql` script to check the execution plan for the following SQL statement and then execute it by using the `query3.sql` script.

```

select /*LOAD_CC*/ * from sh.sales
      where quantity_sold > 40 order by prod_id;

SQL> @explain_query3.sql
SQL> set echo on
SQL>
SQL> explain plan for
2  select /*LOAD_CC*/ * from sh.sales
3  where quantity_sold > 40 order by prod_id;

Explained.

SQL>
SQL> select * from table(dbms_xplan.display(null,null,'basic'));

PLAN_TABLE_OUTPUT
-----
Plan hash value: 3803407550

-----
| Id  | Operation          | Name   |
|---|---|
| 0  | SELECT STATEMENT   |        |
| 1  |  SORT ORDER BY     |        |
| 2  | PARTITION RANGE ALL|        |
| 3  |   TABLE ACCESS FULL| SALES  |
-----

10 rows selected.

```

```

SQL>
SQL> @query3.sql
SQL> set echo on
SQL>
SQL> select /*LOAD_CC*/ * from sh.sales
   2 where quantity_sold > 40 order by prod_id;

no rows selected

SQL>

```

20. Now change the optimizer mode to use FIRST_ROWS optimization and execute the same script as in the previous step. What do you observe? You should see a *different execution plan*.

```

SQL> alter session set optimizer_mode = first_rows;

Session altered.

SQL> @explain_query3.sql
SQL> set echo on
SQL>
SQL> explain plan for
   2 select /*LOAD_CC*/ * from sh.sales
   3 where quantity_sold > 40 order by prod_id;

Explained.

SQL>
SQL> select * from table(dbms_xplan.display(null,null,'basic'));

PLAN_TABLE_OUTPUT
-----
Plan hash value: 1095194413

| Id  | Operation          | Name      |
|---|---|
| 0  | SELECT STATEMENT   |
| 1  |  SORT ORDER BY     |
| 2  |   PARTITION RANGE ALL |
| 3  |     TABLE ACCESS BY LOCAL INDEX ROWID BATCHED | SALES |
| 4  |       BITMAP CONVERSION TO ROWIDS               |
| 5  |       BITMAP INDEX FULL SCAN                   | SALES_PROMO_BIX |
|---|---|
12 rows selected.

SQL>
SQL> @query3.sql
SQL> set echo on
SQL>
SQL> select /*LOAD_CC*/ * from sh.sales
   2 where quantity_sold > 40 order by prod_id;

```

```
no rows selected
SQL>
```

21. Reset the optimizer mode to ALL_ROWS.

```
SQL> alter session set optimizer_mode = all_rows;
Session altered.
```

22. Now that the cursor cache is populated, you must get the SQL ID for your SQL statement. Use the SQL ID to filter the content of the cursor cache and load the baselines with these two plans. Use the `load_cc_baseline.sql` script for these steps.

```
SQL> @load_cc_baseline.sql
SQL> set echo on
SQL>
SQL> variable cnt number;
SQL>
SQL> variable sqlid varchar2(20);
SQL>
SQL> begin
 2   select distinct sql_id into :sqlid from v$sql
 3   where sql_text like 'select /*LOAD_CC*/%';
 4 end;
 5 /
PL/SQL procedure successfully completed.

SQL>
SQL> print sqlid;

SQLID
-----
6qc9wxhgzzz8g

SQL>
SQL> exec :cnt := dbms_spm.load_plans_from_cursor_cache(
->           sql_id => :sqlid);

PL/SQL procedure successfully completed.

SQL>
```

23. Confirm that the baselines were loaded by using `check_baseline3.sql`.

```
SQL> @check_baseline3.sql
SQL> set echo on
SQL>
SQL> select signature, sql_handle, sql_text, plan_name,
 2          origin, enabled, accepted, fixed, autopurge
 3  from dba_sql_plan_baselines
 4  where sql_text like 'select /*LOAD_CC*/%';

SIGNATURE SQL_HANDLE
-----
```

```

SQL_TEXT
-----
----- PLAN_NAME ORIGIN ENA ACC FIX AUT -----
1.7783E+19 SQL_f6cb7f742ef93547
select /*LOAD_CC*/ * from sh.sales
where quantity_sold > 40 order by prod_id
SQL_PLAN_gdkvzfhrkgda754bc8843      MANUAL-LOAD     YES YES NO YES

1.7783E+19 SQL_f6cb7f742ef93547
select /*LOAD_CC*/ * from sh.sales
where quantity_sold > 40 order by prod_id
SQL_PLAN_gdkvzfhrkgda7ce885bc0      MANUAL-LOAD     YES YES NO YES

-- You should see two accepted baselines

```

24. Purge the plan baselines by using `purge_cc_baseline.sql`.

```

SQL> @purge_cc_baseline.sql
SQL> set echo on
SQL>
SQL> variable cnt number;
SQL>
SQL> exec :cnt := dbms_spm.drop_sql_plan_baseline(
->           'SQL_f6cb7f742ef93547');

PL/SQL procedure successfully completed.

SQL>
SQL> print cnt;

      CNT
-----
      2

SQL>
SQL> REM Check that plan baselines were purged:
SQL>
SQL> select signature, sql_handle, sql_text, plan_name,
2          origin, enabled, accepted, fixed, autopurge
3  from dba_sql_plan_baselines
4  where sql_text like 'select /*LOAD_CC*/%';

no rows selected

SQL>

```

Optimizer Plan Selection

25. Now that you know how to capture plans, look at how the optimizer selects which plan baselines to use. First, you create two baselines for a statement and show them being used. Then you disable one of the baselines and see the second baseline being used. Finally, you disable both the baselines and show that the optimizer falls back to the default behavior of a cost-based plan. Start by executing the same query twice, with different plans. Determine the execution of the following statement by using `explain_query4.sql` and then execute it by using the `query4.sql` script.

```

select /*SPM_USE*/ * from sh.sales
  where quantity_sold > 40 order by prod_id

SQL> @explain_query4.sql
SQL> set echo on
SQL>
SQL> explain plan for
  2 select /*SPM_USE*/ * from sh.sales
  3 where quantity_sold > 40 order by prod_id;

Explained.

SQL>
SQL> select * from table(dbms_xplan.display(null,null,'basic'));

PLAN_TABLE_OUTPUT
-----
Plan hash value: 3803407550

-----
| Id  | Operation          | Name   |
|---|---|---|
| 0  | SELECT STATEMENT   |        |
| 1  |   SORT ORDER BY    |        |
| 2  |     PARTITION RANGE ALL |
| 3  |       TABLE ACCESS FULL | SALES |
|---|---|---|
```

10 rows selected.

```

SQL>
SQL> @query4.sql
SQL> set echo on
SQL>
SQL> select /*SPM_USE*/ * from sh.sales
  2 where quantity_sold > 40 order by prod_id;

no rows selected

SQL>
```

26. Change the optimizer mode to use FIRST_ROWS optimization and execute the same script as in the previous step. What do you observe? You should see a different execution plan.

```
SQL> alter session set optimizer_mode = first_rows;

Session altered.

SQL> @explain_query4.sql
SQL> set echo on
SQL>
SQL> explain plan for
  2  select /*SPM_USE*/ * from sh.sales
  3  where quantity_sold > 40 order by prod_id;

Explained.

SQL>
SQL> select * from table(dbms_xplan.display(null,null,'basic'));

PLAN_TABLE_OUTPUT
-----
Plan hash value: 1095194413

-----
| Id  | Operation          | Name      |
| --- | :----- | :----- |
| 0   | SELECT STATEMENT   |           |
| 1   |   SORT ORDER BY    |           |
| 2   |   PARTITION RANGE ALL |
| 3   |     TABLE ACCESS BY LOCAL INDEX ROWID BATCHED | SALES |
| 4   |       BITMAP CONVERSION TO ROWIDS   |           |
| 5   |       BITMAP INDEX FULL SCAN    | SALES_PROMO_BIX |
|-----|
```

12 rows selected.

```
SQL>
SQL> @query4.sql
SQL> set echo on
SQL>
SQL> select /*SPM_USE*/ * from sh.sales
  2  where quantity_sold > 40 order by prod_id;

no rows selected

SQL>
```

27. Reset the optimizer mode to ALL_ROWS.

```
SQL> alter session set optimizer_mode = all_rows;

Session altered.
```

28. Use the `load_use_baseline.sql` script to populate the baseline with the two plans for your statement directly from the cursor cache. Then verify that baselines were loaded. What do you observe?

You should see both plan baselines loaded. Note that both plans have been marked acceptable. This is because both plans were present in the cursor cache at the time of the load, and because these plans have been manually loaded, it is assumed that both plans have acceptable performance.

```
SQL> @load_use_baseline.sql
SQL> set echo on
SQL>
SQL> variable cnt number;
SQL>
SQL> variable sqlid varchar2(20);
SQL>
SQL> begin
 2   select distinct sql_id into :sqlid from v$sql
 3   where sql_text like 'select /*SPM_USE*/%';
 4 end;
 5 /
PL/SQL procedure successfully completed.

SQL>
SQL> print sqlid;
SQLID
-----
2pma6tcaczdc8

SQL>
SQL> exec :cnt := dbms_spm.load_plans_from_cursor_cache(
->           sql_id => :sqlid);

PL/SQL procedure successfully completed.

SQL>
SQL> print cnt;

      CNT
-----
      2

SQL>
SQL> select signature, sql_handle, sql_text, plan_name,
 2          origin, enabled, accepted, fixed, autopurge
 3    from dba_sql_plan_baselines
 4   where sql_text like 'select /*SPM_USE*/%';

SIGNATURE SQL_HANDLE
-----
SQL_TEXT
-----
PLAN_NAME          ORIGIN      ENA ACC FIX AUT
-----
```

```

7.6492E+17 SQL_0a9d872600ece455
select /*SPM_USE*/ * from sh.sales
where quantity_sold > 40 order by prod_id
SQL_PLAN_0p7c74s0ftt2p54bc8843    MANUAL-LOAD      YES YES NO YES

7.6492E+17 SQL_0a9d872600ece455
select /*SPM_USE*/ * from sh.sales
where quantity_sold > 40 order by prod_id
SQL_PLAN_0p7c74s0ftt2pce885bc0    MANUAL-LOAD      YES YES NO YES

SQL>

```

29. Use the `explain_query4_note.sql` script to determine the baseline and the execution plan used to execute the following query:

```

select /*SPM_USE*/ * from sh.sales
where quantity_sold > 40 order by prod_id.

```

What do you observe?

The note at the end of the explain output tells you that the system is using a baseline. From the execution plan, you can see that you are using the first baseline, a full table scan.

```

SQL> @explain_query4_note.sql
SQL> set echo on
SQL>
SQL> explain plan for
2  select /*SPM_USE*/ * from sh.sales
3  where quantity_sold > 40 order by prod_id;

Explained.

SQL>
SQL> select * from table(dbms_xplan.display(null,null,'basic
+note'));

PLAN_TABLE_OUTPUT
-----
Plan hash value: 3803407550
-----
```

Id	Operation	Name
0	SELECT STATEMENT	
1	SORT ORDER BY	
2	PARTITION RANGE ALL	
3	TABLE ACCESS FULL	SALES

```
PLAN_TABLE_OUTPUT
```

```
Note
```

```
-----
- SQL plan baseline "SQL_PLAN_0p7c74s0ftt2p54bc8843" used for
this statement

14 rows selected.

SQL>
```

30. Use the `check_baseline_used.sql` script to disable that plan baseline and check whether the system uses the other plan baseline when executing the statement again. What do you observe?

From the execution plan, you see that you are using an index scan instead of a full table scan. So this is the second baseline.

```
SQL> @check_baseline_used.sql
SQL> set echo on
SQL>
SQL> variable cnt number;
SQL>
SQL> select sql_handle,plan_name
  2  from dba_sql_plan_baselines
  3  where sql_text like 'select /*SPM_USE*/%';

SQL_HANDLE          PLAN_NAME
-----
SQL_0a9d872600ece455  SQL_PLAN_0p7c74s0ftt2p54bc8843
SQL_0a9d872600ece455  SQL_PLAN_0p7c74s0ftt2pce885bc0

SQL>
SQL>
SQL> exec :cnt := dbms_spm.alter_sql_plan_baseline( -
  >   sql_handle    => 'SQL_0a9d872600ece455', -
  >   plan_name     => 'SQL_PLAN_0p7c74s0ftt2p54bc8843', -
  >   attribute_name => 'ENABLED', -
  >   attribute_value => 'NO');

PL/SQL procedure successfully completed.

SQL>
SQL> select signature, sql_handle, sql_text, plan_name,
  2        origin, enabled, accepted, fixed, autopurge
  3  from dba_sql_plan_baselines
  4  where sql_text like 'select /*SPM_USE*/%';

SIGNATURE SQL_HANDLE
-----
SQL_TEXT
-----
PLAN_NAME          ORIGIN      ENA ACC FIX AUT
-----
7.6492E+17 SQL_0a9d872600ece455
select /*SPM_USE*/ * from sh.sales
```

```

where quantity_sold > 40 order by prod_id
SQL_PLAN_0p7c74s0ftt2p54bc8843           MANUAL-LOAD      NO  YES  NO  YES

7.6492E+17 SQL_0a9d872600ece455
select /*SPM_USE*/ * from sh.sales
where quantity_sold > 40 order by prod_id
SQL_PLAN_0p7c74s0ftt2pce885bc0           MANUAL-LOAD      YES  YES  NO  YES

SQL>
SQL>
SQL> explain plan for select /*SPM_USE*/ * from sh.sales
   2       where quantity_sold > 40 order by prod_id;

Explained.

SQL>
SQL> select * from table(dbms_xplan.display(null, null, 'BASIC
NOTE'));

PLAN_TABLE_OUTPUT
-----
Plan hash value: 1095194413

-----
| Id  | Operation          | Name |
|---|---|
| 0  | SELECT STATEMENT   |
| 1  |  SORT ORDER BY     |
| 2  |   PARTITION RANGE ALL |
| 3  |     TABLE ACCESS BY LOCAL INDEX ROWID BATCHED | SALES |
| 4  |       BITMAP CONVERSION TO ROWIDS |
| 5  |       BITMAP INDEX FULL SCAN    | SALES_PROMO_BIX |
|---|---|
```

Note

- SQL plan baseline "SQL_PLAN_0p7c74s0ftt2pce885bc0" used for this statement

16 rows selected.

SQL>

31. Use the `check_baseline_used2.sql` script to disable the other plan baseline and check whether the system falls back to the cost-based approach when executing the explain plan for the statement.

You know that the optimizer has gone back to the default cost-based approach because there is no note at the end of the plan stating that a baseline was used.

```
SQL> @check_baseline_used2.sql
SQL> set echo on
SQL>
SQL> variable cnt number;
SQL>
SQL> exec :cnt := dbms_spm.alter_sql_plan_baseline( -
>      sql_handle      => 'SQL_0a9d872600ece455', -
>      plan_name       => 'SQL_PLAN_0p7c74s0ftt2pce885bc0', -
>      attribute_name  => 'ENABLED', -
>      attribute_value => 'NO');

PL/SQL procedure successfully completed.

SQL>
SQL> select signature, sql_handle, sql_text, plan_name,
2          origin, enabled, accepted, fixed, autopurge
3  from dba_sql_plan_baselines
4 where sql_text like 'select /*SPM_USE*/%';

SIGNATURE SQL_HANDLE
-----
SQL_TEXT
-----
PLAN_NAME          ORIGIN        ENA ACC FIX AUT
-----
7.6492E+17 SQL_0a9d872600ece455
select /*SPM_USE*/ * from sh.sales
where quantity_sold > 40 order by prod_id
SQL_PLAN_0p7c74s0ftt2p54bc8843  MANUAL-LOAD      NO  YES NO  YES

7.6492E+17 SQL_0a9d872600ece455
select /*SPM_USE*/ * from sh.sales
where quantity_sold > 40 order by prod_id
SQL_PLAN_0p7c74s0ftt2pce885bc0  MANUAL-LOAD      NO  YES NO  YES

SQL>
SQL>
SQL> explain plan for select /*SPM_USE*/ * from sh.sales
2  where quantity_sold > 40 order by prod_id;

Explained.

SQL>
SQL> select * from table(dbms_xplan.display(null, null, 'basic
+note'));
```

```

PLAN_TABLE_OUTPUT
-----
-----
Plan hash value: 3803407550

-----
| Id  | Operation          | Name   |
-----
| 0   | SELECT STATEMENT   |         |
| 1   |   SORT ORDER BY    |         |
| 2   |     PARTITION RANGE ALL |
| 3   |       TABLE ACCESS FULL | SALES |
-----
10 rows selected.

SQL>

```

32. Drop the plan baselines and check whether they are purged by using the `purge_use_baseline.sql` script.

```

SQL> @purge_use_baseline
SQL> set echo on
SQL>
SQL> variable cnt number;
SQL>
SQL> exec :cnt := dbms_spm.drop_sql_plan_baseline(
->           'SQL_0a9d872600ece455');

PL/SQL procedure successfully completed.

SQL>
SQL> print cnt;

      CNT
-----
      2

SQL>
SQL> select signature, sql_handle, sql_text, plan_name,
2        origin, enabled, accepted, fixed, autopurge
3  from dba_sql_plan_baselines
4 where sql_text like 'select /*SPM_USE*/%';

no rows selected

SQL>

```

33. One of the methods used to enable plan evolution (or plan verification) is Automatic SQL Tuning that is run as an automated task in a maintenance window. Automatic SQL Tuning targets only the high-load SQL statements; for those statements, it automatically implements actions, such as making a successfully verified plan an accepted plan. Here, you manually trigger SQL tuning to find a better plan for a given SQL statement. First, determine the execution plan of the following statement by using the `check_evolve_plan.sql` script:

```
select /*+ USE_NL(s c) FULL(s) FULL(c) */
       c.cust_id, sum(s.quantity_sold)
  from sh.sales s, sh.customers c
 where s.cust_id = c.cust_id and c.cust_id < 2
   group by c.cust_id
```

What do you observe? *Some optimizer hints have been added to the statements to ensure that you get a less than optimal plan at first.*

As you can see, the execution plan is being forced by the hints to perform two full table scans, followed by a nest loop join.

```
SQL> @check_evolve_plan.sql
SQL> set echo on
SQL>
SQL> explain plan for
  2  select /*+ USE_NL(s c) FULL(s) FULL(c) */
  3        c.cust_id, sum(s.quantity_sold)
  4    from sh.sales s, sh.customers c
  5   where s.cust_id = c.cust_id and c.cust_id < 2
  6   group by c.cust_id;

Explained.

SQL>
SQL> select * from table(dbms_xplan.display(null, null));
```

PLAN_TABLE_OUTPUT										

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time	Pstart	Pstop		
0	SELECT STATEMENT		1	13	938 (1)	00:00:01				
1	HASH GROUP BY		1	13	938 (1)	00:00:01				
2	NESTED LOOPS		1	13	938 (1)	00:00:01				
*3	TABLE ACCESS FULL	CUSTOMERS	1	5	423 (1)	00:00:01				
4	PARTITION RANGE ALL		1	8	515 (1)	00:00:01	1	28		
*5	TABLE ACCESS FULL	SALES	1	8	515 (1)	00:00:01	1	28		

```
PLAN_TABLE_OUTPUT
-----
Predicate Information (identified by operation id):
-----
3 - filter("C"."CUST_ID"><2)
5 - filter("S"."CUST_ID"><2 AND "S"."CUST_ID"="C"."CUST_ID")
```

```
18 rows selected.
```

```
SQL>
```

34. Now use the `load_evolve_baseline.sql` script to execute the statement so that you can get the plan in the cursor cache and load the corresponding plan baseline. What do you observe?

You see that the current plan is both enabled and accepted, but not fixed.

```
SQL> @load_evolve_baseline
SQL> set echo on
SQL>
SQL> variable cnt number;
SQL>
SQL> variable sqlid varchar2(20);
SQL>
SQL> select /*+ USE_NL(s c) FULL(s) FULL(c) */
  2      c.cust_id, sum(s.quantity_sold)
  3  from sh.sales s, sh.customers c
  4 where s.cust_id = c.cust_id and c.cust_id < 2
  5 group by c.cust_id;

no rows selected

SQL>
SQL> begin
  2   select sql_id into :sqlid from v$sql
  3   where sql_text like 'select /*+ USE_NL(s c) FULL(s) FULL(c)
*/%';
  4 end;
  5 /

PL/SQL procedure successfully completed.

SQL>
SQL> exec :cnt := dbms_spm.load_plans_from_cursor_cache(
->           sql_id => :sqlid);

PL/SQL procedure successfully completed.

SQL>
SQL> select signature, sql_handle, sql_text, plan_name,
  2          origin, enabled, accepted, fixed, autopurge
  3  from dba_sql_plan_baselines
  4 where sql_text like 'select /*+ USE_NL(s c) FULL(s) FULL(c)
*/%';

SIGNATURE SQL_HANDLE
-----
SQL_TEXT
-----
PLAN_NAME          ORIGIN        ENA ACC FIX AUT
-----
```

```

1.7750E+18 SQL_18a1ef14c17f5b75
select /*+ USE_NL(s c) FULL(s) FULL(c) */
c.cust_id, sum(s.quantity_sold)
SQL_PLAN_1j8gg2m0ryqvpdc5f94e5 MANUAL-LOAD YES YES NO YES

```

35. Manually create and execute a SQL tuning task to tune your statement by using the `tune_evolve_sql.sql` script.

```

SQL> @tune_evolve_sql.sql
SQL> set echo on
SQL>
SQL> variable sqltext varchar2(4000);
SQL>
SQL> BEGIN
 2   :sqltext := q'# select /*+ USE_NL(s c) FULL(s) FULL(c) */
 3                           c.cust_id, sum(s.quantity_sold)
 4                           from sh.sales s, sh.customers c
 5                           where s.cust_id = c.cust_id
 6                               and c.cust_id < 2
 7                           group by c.cust_id
 8 #
 9 END;
10 /
PL/SQL procedure successfully completed.

SQL>
SQL> variable spmtune    varchar2(30);
SQL>
SQL> exec :spmtune := dbms_sqltune.create_tuning_task(
->                               sql_text => :sqltext);

PL/SQL procedure successfully completed.

SQL>
SQL> exec dbms_sqltune.execute_tuning_task(:spmtune);

PL/SQL procedure successfully completed.

SQL>

```

36. Now that the tuning task has been completed, run the report by using the `report_evolve_tuning.sql` script and see what recommendations have been made for your statement. What do you observe?

What recommendations have been made? A SQL profile, and possibly a new index

```

SQL> @report_evolve_tuning.sql
SQL> set echo on
SQL>
SQL> set long 10000
SQL>
SQL> select dbms_sqltune.report_tuning_task(:spmtune, 'TEXT')
 2   from dual;

```

```
-- Report Follows -
...
FINDINGS SECTION (1 finding)
-----
1- SQL Profile Finding (see explain plans section below)
-----
A potentially better execution plan was found for this statement.

Recommendation (estimated benefit: 98.98%)
-----
- Consider accepting the recommended SQL profile. A SQL plan
  baseline corresponding to the plan with the SQL profile will
  also be created.
  execute dbms_sqltune.accept_sql_profile(task_name =>
    'TASK_1188', task_owner => 'SPM', replace => TRUE);

Validation results
-----
The SQL profile was tested by executing both its plan and the
original plan and measuring their respective execution statistics.
A plan may have been only partially executed if the other could be
run to completion in less time.

      Original Plan   With SQL Profile % Improved
-----      COMPLETE      COMPLETE
Completion Status:      COMPLETE
Elapsed Time (s):     .131832     .000141    99.89 %
CPU Time (s):          .130855     .000199    99.84 %
User I/O Time (s):     .007693     .000018    99.76 %
Buffer Gets:            3142        32       98.98 %
Physical Read Requests: 399         3       99.24 %
Physical Write Requests: 0           0
Physical Read Bytes:    25436160    26214    99.89 %
Physical Write Bytes:   0           0
Rows Processed:          0           0
Fetches:                  0           0
Executions:                 1           1

Notes
-----
1. Statistics for the original plan were averaged over 8
executions.
2. Statistics for the SQL profile plan were averaged over 10
executions.
...
```

37. Accept the SQL profile proposed by the SQL Tuning Advisor. Use the `accept_evolve_baseline.sql` script to accept the recommended SQL profile. What happens?

Accepting the profile causes a new SQL profile and plan baseline to be created for your statement. Now you see two baselines for your statement. Both of them are enabled and accepted.

Note: One is MANUAL-LOAD and the other is MANUAL-SQLTUNE.

```
SQL> @accept_evolve_baseline.sql
SQL> set echo on
SQL>
SQL> select signature, sql_handle, sql_text, plan_name,
2      origin, enabled, accepted, fixed, autopurge
3  from dba_sql_plan_baselines
4  where sql_text like
5      'select /*+ USE_NL(s c) FULL(s) FULL(c) */%';

```

SIGNATURE	SQL_HANDLE				
-----	-----				
SQL_TEXT	-----				
PLAN_NAME	ORIGIN	ENA	ACC	FIX	AUT
-----	-----	-----	-----	-----	-----
1.7750E+18 SYS_SQL_18alef14c17f5b75					
select /*+ USE_NL(s c) FULL(s) FULL(c) */ c.cust_id,					
sum(s.quantity_sold)					
from sh.sales s, sh.customers c					
where s.cust_id = c.cust_id and c.cust_id < 2					
group by c.cust_id					
SYS_SQL_PLAN_c17f5b75dc5f94e5	MANUAL-LOAD	YES	YES	NO	YES


```
SQL>
SQL> exec dbms_sqltune.accept_sql_profile(
->       task_name => :spmtune,-
->       name => 'SPM_SQL_PROF');

PL/SQL procedure successfully completed.

SQL>
SQL> select signature, category, name, sql_text
2  from dba_sql_profiles where name like 'SPM%';


```

SIGNATURE	CATEGORY	NAME
-----	-----	-----
SQL_TEXT	-----	-----
-----	-----	-----
1.7750E+18 DEFAULT		SPM_SQL_PROF
select /*+ USE_NL(s c) FULL(s) FULL(c) */ c.cust_id,		
sum(s.quantity_sold)		
from sh.sales s, sh.customers c		
where s.cust_id = c.cust_id and c.cust_id < 2		

```

group by c.cust_id

SQL> select signature, sql_handle, sql_text, plan_name,
  2 origin, enabled, accepted, fixed, autopurge
  3 from dba_sql_plan_baselines
  4 where sql_text like
  5   'select /*+ USE_NL(s c) FULL(s) FULL(c) */';

SIGNATURE SQL_HANDLE
-----
SQL_TEXT
-----
PLAN_NAME          ORIGIN      ENA ACC FIX AUT
-----
1.7750E+18 SQL_18a1ef14c17f5b75
select /*+ USE_NL(s c) FULL(s) FULL(c) */
       c.cust_id, sum(s.quantity_sold)
from sh.sales s, sh.customers c
where s.cust_id = c.cust_id and c.cust_id < 2
group by c.cust_id
SQL_PLAN_1j8gg2m0ryqv08763816    MANUAL-SQLTUNE YES YES NO YES

1.7750E+18 SQL_18a1ef14c17f5b75
select /*+ USE_NL(s c) FULL(s) FULL(c) */
       c.cust_id, sum(s.quantity_sold)
from sh.sales s, sh.customers c
where s.cust_id = c.cust_id and c.cust_id < 2
group by c.cust_id
SQL_PLAN_1j8gg2m0ryqv0dc5f94e5    MANUAL-LOAD     YES YES NO YES

SQL>

```

38. Determine the plan used for your statement when executing it by executing the `explain_query5.sql` script. What do you observe?

The next time that you execute the query, it uses the plan baseline and the SQL profile.

```

SQL> @explain_query5.sql
SQL> set echo on
SQL>
SQL> explain plan for
  2 select /*+ USE_NL(s c) FULL(s) FULL(c) */
  3       c.cust_id, sum(s.quantity_sold)
  4 from sh.sales s, sh.customers c
  5 where s.cust_id = c.cust_id and c.cust_id < 2
  6 group by c.cust_id;

Explained.

SQL>
SQL> select * from table(dbms_xplan.display(null,
  2                               null, 'basic +note'));

```

```

PLAN_TABLE_OUTPUT
-----
Plan hash value: 34974602

-----| Id | Operation | Name |
-----| 0 | SELECT STATEMENT |          |
| 1 |   HASH GROUP BY |          |
| 2 |     NESTED LOOPS |          |
| 3 |       PARTITION RANGE ALL |          |
| 4 |         TABLE ACCESS BY LOCAL INDEX ROWID BATCHED | SALES |
| 5 |           BITMAP CONVERSION TO ROWIDS |          |
| 6 |             BITMAP INDEX RANGE SCAN | SALES_CUST_BIX |
| 7 |           INDEX UNIQUE SCAN | CUSTOMERS_PK |
-----|          |          |          |

Note
-----
- SQL profile "SPM_SQL_PROF" used for this statement
- SQL plan baseline "SQL_PLAN_1j8gg2m0ryqvp08763816" used for
this statement

19 rows selected.

SQL>

```

39. Execute the `cleanup_spm.sql` script to remove profiles and baselines created during this practice.

```

SQL> @cleanup_spm.sql
SQL> set echo on
SQL>
SQL> exec dbms_sqltune.drop_sql_profile(
->       name => 'SPM_SQL_PROF');

PL/SQL procedure successfully completed.

SQL>
SQL> exec :cnt := dbms_spm.drop_sql_plan_baseline(
->       'SQL_18alef14c17f5b75');

PL/SQL procedure successfully completed.

SQL>
SQL> select signature, sql_handle, sql_text, plan_name,
2          origin, enabled, accepted, fixed, autopurge
3  from dba_sql_plan_baselines
4  where sql_text like
5          'select /*+ USE_NL(s c) FULL(s) FULL(c) */%';

no rows selected

SQL>
SQL> select signature, category, name, sql_text

```

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

```
2  from dba_sql_profiles where name like 'SPM%';

no rows selected

SQL> alter system set optimizer_use_sql_plan_baselines=FALSE;

System altered.

SQL> exit
$
```

40. Prepare for the next practice by executing the `prepare` command with the `CAPTURE` option. This script is in the `$HOME/workshops` directory. Log out of Enterprise Manager Cloud Control before running the script. This script takes about six minutes.

```
$ cd $HOME/workshops
$ ./prepare CAPTURE
```

Practices for Lesson 15: Using Database Replay

Chapter 15

Practices for Lesson 15: Overview

Practices Overview

In this practice, you use Database Replay to test a simulation of changing the initialization parameter `OPTIMIZER_FEATURES_ENABLE` from 11.2.0.1 to 12.1.0.1. In the current instance, the memory and SQL are well tuned. Because, you do not want to see any regressions on SQL statements that are already running acceptably on Oracle Database 11gR2, and you want to see what improvements the new optimizer setting will give, the `OPTIMIZER_CAPTURE_SQL_BASELINE`, and `OPTIMIZER_USE_SQL_BASELINE` parameters have been set to TRUE in the prepare scripts. You will use the Performance Tuning Analyzer to make further adjustments.

Prerequisites: Practice 5-2 has been executed.

Practice 15-1: Using Database Replay

In this practice, you capture a workload, process the workload, and replay the workload with changes.

Note: The practice is retained here for reference. Use the videos provided in the /home/oracle/demos directory.

- Practice15_Capture (4:12) Part 1 Capture a workload
- Practice15_Replay1(11:59) Part 2 Replay a captured workload and view the reports
- Practice15_Replay2 (4:22) Part 3 Replay a workload for comparison. SQL Profiles have been applied and the workload replayed.

The videos follow the text of the practice, and include a few views that are suggested but not required in the practice.

1. If you have not already done so, execute the `prepare` command with the CAPTURE option. This script performs a flashback database, restoring the database to a known state, and starts the database with the parameters set to simulate an 11g database with Automatic Memory tuning. This script is in the \$HOME/workshops directory. Log out of Enterprise Manager before running the script. This script takes five to nine minutes.

```
$ cd $HOME/workshops
$ ./prepare CAPTURE
```

2. Create a directory to capture the workload files named \$ORACLE_BASE/capture.

```
$ mkdir /u01/app/oracle/capture
```

3. Use Enterprise Manager Cloud Control to start a workload capture. Start EM and log in as sysman with the password oracle_4U.

Step	Window/Page Description	Choices or Values
a.		Click Enterprise > Quality Management > Database Replay .
b.	Database Replay Captures Workloads tab	Click Create .
c.	Create Capture: Plan Environment	Acknowledge the prerequisites. (Select the check boxes.) Click Next .

Database Replay

Plan Environment Database Options Storage Schedule Review

Create Capture : Plan Environment

Back Step 1 of 6 Next Cancel

Capture Prerequisites

The following prerequisites should be met to avoid potential problems before proceeding to capture the workload.

- Make sure there is enough disk space to hold the captured workload. Consider doing a short duration workload capture and using it for estimating the disk space requirement of a full workload capture.
- Make sure you can restore the replay database to match the capture database at the start of the workload capture. A successful workload replay depends on application transactions accessing application data identical to that on a capture system. Common ways to restore application data state include point-in-time recovery, flashback, and import/export.

Step	Window/Page Description	Choices or Values
d.	Create Capture: Database	Click Add
e.	Add dialog	Set values: Capture Name: CAPTURE_11gSim Target Database: (Click the search icon )
f.	Search and Select	Select target orcl . (Highlight the orcl row.) Click Select .
g.	Add dialog	Database Credential Credential: select Preferred Preferred Credential Name: SYSDBA Database Credentials Database Host Credential Credential: select New UserName: oracle Password: oracle Confirm Password: oracle Save As: oracle_host Select Set As Preferred Credentials Set: Database Host Credentials Database Capture Intermediate Storage Location: /u01/app/oracle/capture Click OK .
h.	Create Capture: Database	Click Next .

Create Capture : Database

Back Step 2 of 6 Next Cancel

Capture Databases

Add the production database for which you want to perform a capture. To perform concurrent captures, add multiple database targets. Concurrent captures are associated by a common concurrent capture name. Concurrent captures can be useful for consolidated replays. Each individual capture can be used for replays independently.

	Add	Edit	Delete
Database Name	Capture Name	Intermediate Storage Location	Description
orcl	CAPTURE_11gSim	/u01/app/oracle/capture	

Step	Window/Page Description	Choices or Values
i.	Create Capture: Options	Verify that Capture SQL statements into a SQL Tuning Set during workload capture is selected. Filter Mode: Exclusion Excluded Sessions include: Oracle Management Service (DEFAULT) Oracle Management Agent (DEFAULT) Click Next .
j.	Create Capture : Storage	Storage Host: (Click the search icon
k.	Search and Select	Select target (the name of your machine). Click Select .
l.	Create Capture : Storage	Host Credential Credential: select New UserName: oracle Password: oracle Confirm Password: oracle Save As: oracle_NC_host Select Set As Preferred Credentials Set: Normal Host Credentials Storage Location: /u01/app/oracle/capture Click Next .
m.	Create Capture: Schedule	Set values: Start: Immediately Duration: Indefinitely Automatic Workload Repository Start: Immediately Click Next .
n.	Create Capture : Review	Click Submit .

Create Capture : Review

Back | Step 6 of 6 | Submit | Cancel

Database

Concurrent Capture No

Database Name	Capture Name	Intermediate Storage Location	Description
orcl	Capture11gSim	/u01/app/oracle/capture	

Options

SQL Performance Analyzer

Capture SQL statements into a SQL Tuning Set during workload capture.

Workload Filters

Excluded Sessions

Filter Name	Type	Session Attribute	Value
Oracle Management Service (DEFAULT)	Excluded	Program	OMS
Oracle Management Agent (DEFAULT)	Excluded	Module	emagent%

Storage

Storage Host edRSr7p1.us.oracle.com
Storage Location /u01/app/oracle/capture

Schedule

Capture Schedule

Start Immediately
Duration Indefinitely

Automatic Workload Repository

Start Immediately

Step	Window/Page Description	Choices or Values
o.	Database Replay CAPTURE_11gSim	CAPTURE_11gSim appears in the Captured Workloads List. Click the name CAPTURE_11gSim . Click the Summary tab. Go to the next step.

4. In a terminal window, start the capture workload. After about four minutes of capture, stop the workload by running `rm unload`. When the `ps` command shows only the `ps` and `bash` processes (this may take a couple of minutes), continue to the next step.
- While this is running, you can see the progress and the Duration in the summary section on the Capture:CAPTURE_11gSim page. Click the Refresh button  regularly to view the current state of the capture.

```
$ ./workgen capture
The Oracle base for
ORACLE_HOME=/u01/app/oracle/product/12.1.0/dbhome_1 is
/u01/app/oracle
Starting buffer cache load
Starting Soft Parse load
Starting PGA load
Starting Hard Parse load

System altered.

PL/SQL procedure successfully completed.

PL/SQL procedure successfully completed.

PL/SQL procedure successfully completed.

/* wait about 4 minutes */

$ rm unload

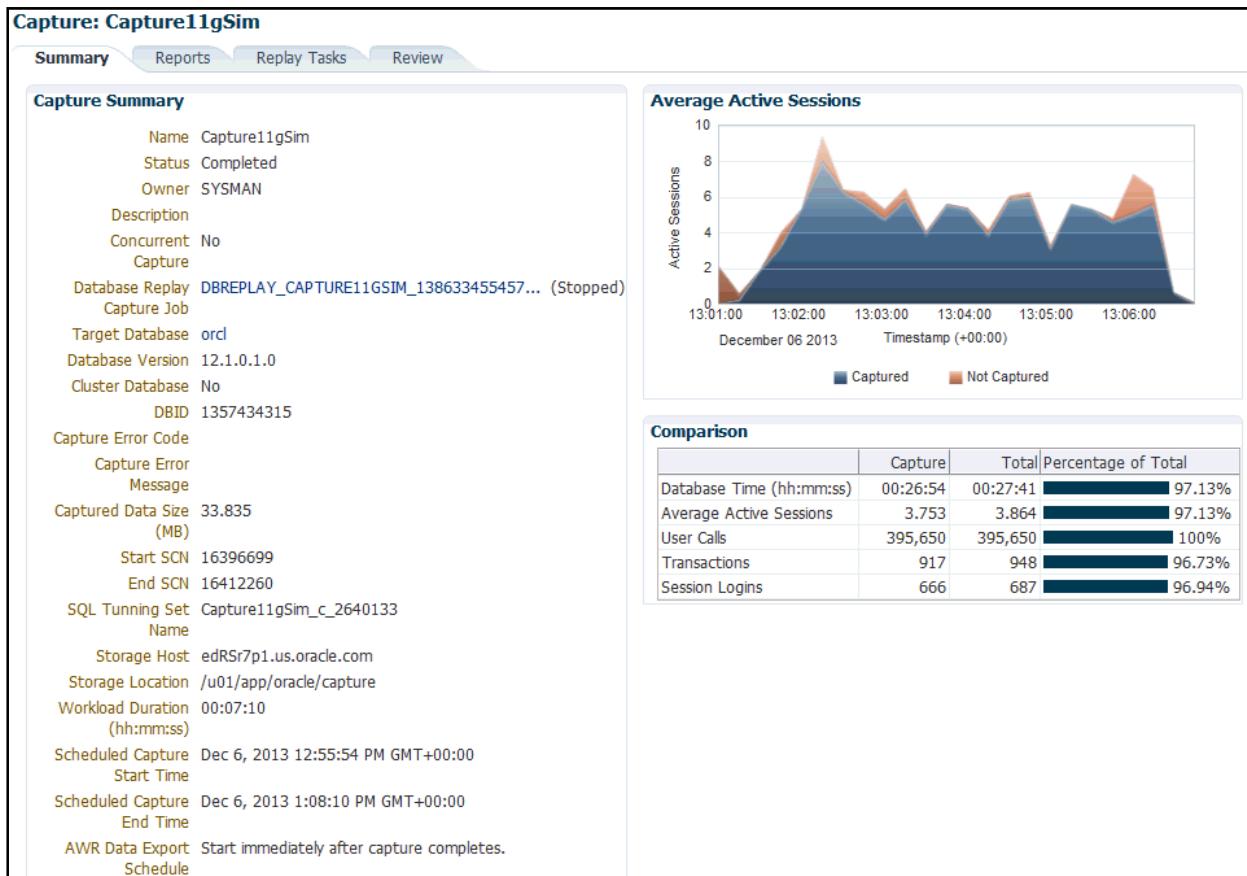
$ ps
  PID TTY      TIME CMD
 8432 pts/6    00:00:00 delete_orders.s
 8433 pts/6    00:00:00 insert_orders.s
 8450 pts/6    00:00:00 update_orders.s
 8459 pts/6    00:00:00 delete_orders.s
 8460 pts/6    00:00:00 insert_orders.s
 8478 pts/6    00:00:00 update_orders.s
 8576 pts/6    00:00:00 delete_orders.s
10540 pts/6    00:00:00 sleep
10552 pts/6    00:00:00 sleep
10560 pts/6    00:00:00 sleep
10580 pts/6    00:00:00 sleep
10592 pts/6    00:00:00 sleep
10599 pts/6    00:00:00 sleep
10618 pts/6    00:00:00 sleep
11117 pts/6    00:00:00 ps
20352 pts/6    00:00:00 bash

/* 1-2 minutes */
```

```
$ ps
  PID TTY          TIME CMD
11126 pts/6    00:00:00 ps
20352 pts/6    00:00:00 bash
```

5. Stop the capture and export the AWR data.

Step	Window/Page Description	Choices or Values
a.	Database Replay Capture:CAPTURE_11gSim	Click Stop Capture . Status changes to:Replay_1 In Progress Stopping
b.	Capture: CAPTURE-11gSIM	Refresh every 20- 30 seconds until Status shows completed. This may take three to seven minutes.



6. Create a Replay task, and a replay.

Step	Window/Page Description	Choices or Values
a.		Navigate to the Database Replay page. Click Database Replay , in the breadcrumb.
b.	Database Replay:	Click the Replay Tasks tab.
c.	Database Replay: Replay Tasks	Click Create .
d.	Database Replay: Create Task	Set values: Name: Replay_orcl_12c Select the workload: CAPTURE_11gSim Click Submit .
e.	Database Replay: Replay Tasks	Confirmation Message appears. Replay_orcl_12c appears in the Replay Tasks list. Click the Replay_orcl_12c link.
f.	Database Replay: Replay Task: Replay_orcl_12c	In the Replays Section, click Create .
g.	Create Replay dialog	Name: Replay_1 Target database: Use the Search icon to select the orcl database. Click Ok .
h.	Replay: Replay_1	On this page, the Replay tasks can be selected. Instead of using EMCC to set up the test database, go to a terminal window as described in the next step.

7. Adjust the database setup. Execute the `./prepare REPLAY` script. This script restores the database to the same point (SCN, where the capture started) using flashback database, and changes the parameters to use 12c optimizer features and Automatic Memory Management. Log out of EMCC before you run `./prepare REPLAY`.

```
$ cd $HOME/workshops
$ ./prepare REPLAY
```

8. Determine the directory where the capture was stored. In a terminal window, issue the command:

```
$ ls /u01/app/oracle/capture
1 DBReplayWorkload_CAPTURE_11gSim_1
/* record the value shown on your machine */
/* the last digits will vary */
```

Record the name of the capture directory here _____

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

9. Return to the EMCC Replay: Replay_1 page, and preprocess the captured workload.

Step	Window/Page Description	Choices or Values
a.	EMCC Login	User Name: sysman Password: oracle_4U
b.		Select Enterprise > Quality Management > Database Replay .
c.	Database Replay: Replay Tasks tab	Click the Replay_orcl_12c link.
d.	Replay Task: Replay_orcl_12c	Click the Replay_1 link.
e.	Replay:Replay_1	Verify that Target Database is orcl (If needed, use the search Icon ) Click the Preprocess Workload link.
f.	Database Login	Select Preferred . Select SYSDBA Database Credentials . Click Login .
g.	Preprocess Captured Workload: Locate Workload	Select “Use an existing workload directory on this host.” Click Next .
h.	Preprocess Captured Workload: Select Directory	Click Create Directory Object .
i.	Create Directory Object	Set values: Name: WORKLOAD Path: <i>Enter the full path and name of the capture directory from Step 8</i> Click OK .
j.	Preprocess Captured Workload: Select Directory	Verify Directory Object Verify that there is information in the Capture Summary Section. Click Next .
k.	Preprocess Captured Workload: Schedule	Click Next .
l.	Preprocess Captured Workload: Review	Click Submit . A Confirmation message of the job submission appears. Click the Job Name link.
m.	Execution: orcl	Set Auto Refresh to 30 seconds . As the page refreshes the steps of the job will be displayed. When Status in the Summary section, shows Succeeded, continue.

Database Replay Replay: Repaly_1 > Execution: orcl

Page Refreshed Dec 6, 2013 1:43:10 PM UTC C

Execution: orcl

[Delete Run](#) [Edit](#) [View Definition](#)

Summary

Status Succeeded

Scheduled Dec 6, 2013 1:30:00 PM GMT+00:00
 Started Dec 6, 2013 1:40:24 PM GMT+00:00
 Ended Dec 6, 2013 1:42:59 PM GMT+00:00
 Elapsed Time 2 minutes, 35 seconds

Type Workload Preprocess
 Owner SYSMAN
 Description
 Execution ID ECDEC964954865E1E0436B23B98BD5EE
 Username oracle

Targets
 Status All
[Expand All](#) [Collapse All](#)

Name	Targets	Status	Started	Ended	Elapsed Time
Execution: orcl	orcl	Succeeded	Dec 6, 2013 1:40:24 PM GMT+00:00	Dec 6, 2013 1:42:59 PM GMT+00:00	2.6 minutes
Step: Analyze	orcl	Succeeded	Dec 6, 2013 1:40:24 PM GMT+00:00	Dec 6, 2013 1:42:35 PM GMT+00:00	2.2 minutes
Step: Save Workload Analyzer Report	orcl	Succeeded	Dec 6, 2013 1:42:36 PM GMT+00:00	Dec 6, 2013 1:42:37 PM GMT+00:00	0 seconds
Step: Preprocess	orcl	Succeeded	Dec 6, 2013 1:42:37 PM GMT+00:00	Dec 6, 2013 1:42:57 PM GMT+00:00	19 seconds
Step: Save Preprocess Results		Succeeded	Dec 6, 2013 1:42:57 PM GMT+00:00	Dec 6, 2013 1:42:58 PM GMT+00:00	0 seconds
Step: Estimate Replay Clients Needed	orcl	Succeeded	Dec 6, 2013 1:42:58 PM GMT+00:00	Dec 6, 2013 1:42:58 PM GMT+00:00	0 seconds
Step: Save Estimate Results		Succeeded	Dec 6, 2013 1:42:59 PM GMT+00:00	Dec 6, 2013 1:42:59 PM GMT+00:00	0 seconds

[Log Report](#)

10. Review the Workload Capture Analyzer report.

Step	Window/Page Description	Choices or Values
a.		Navigate to the Database Replay page. Use the menu Enterprise > Quality Management > Database Replay .
b.	Database Replay	On the Captured Workloads tab, click CAPTURE_11gSim .
c.	Capture: CAPTURE_11gSim	Click the Reports tab. Click View beside Workload Analyzer Report.
d.	New Browser window Workload Capture Analyzer Report	Note the Capture Directory name. Examine the Summary of Findings. Findings and recommendation provide a guide to adjust the replay or perhaps collect a new capture. <i>In this practice, if the INCOMPLETE FILES percentage is significant, it means the workload was not completely stopped before the capture was ended. In this case, several divergences are expected.</i> Close the browser window for the Workload Capture Analyzer Report.

Workload Capture Analyzer Report

Capture Details

Capture Directory

/u01/app/oracle/capture/DBReplayWorkload_Capture11gSim_162

Summary of Findings

Finding	Maximum Workload Impact
PL/SQL	52 %
SYSDATE and other time-dependent functions	18 %

Findings and Recommendations

PL/SQL

Maximum Workload Impact: 52 % of DB Time

Rationale

If the replay is much slower than expected, try to run in unsynchronized mode.

Action

A significant part of your workload comes from PL/SQL.

If the PL/SQL blocks or functions have 'complicated' logic or multiple commits in them, they are slower during replay.

You might see a different workload profile during replay if this is the case.

SYSDATE and other time-dependent functions

Maximum Workload Impact: 18 % of DB Time

Rationale

A significant part of your SQL workload is referencing SYSDATE.

Action

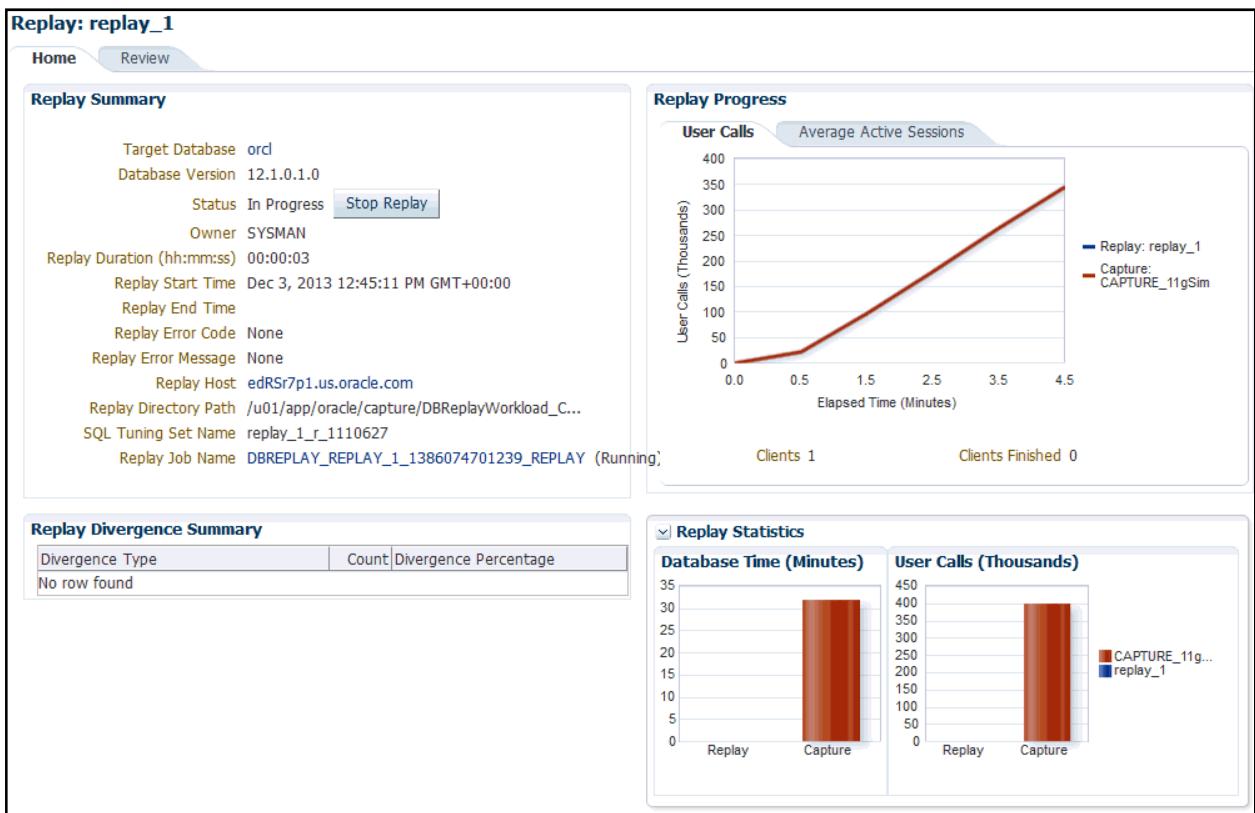
The system clock needs to be restored to its capture value before replay.

End of Report.

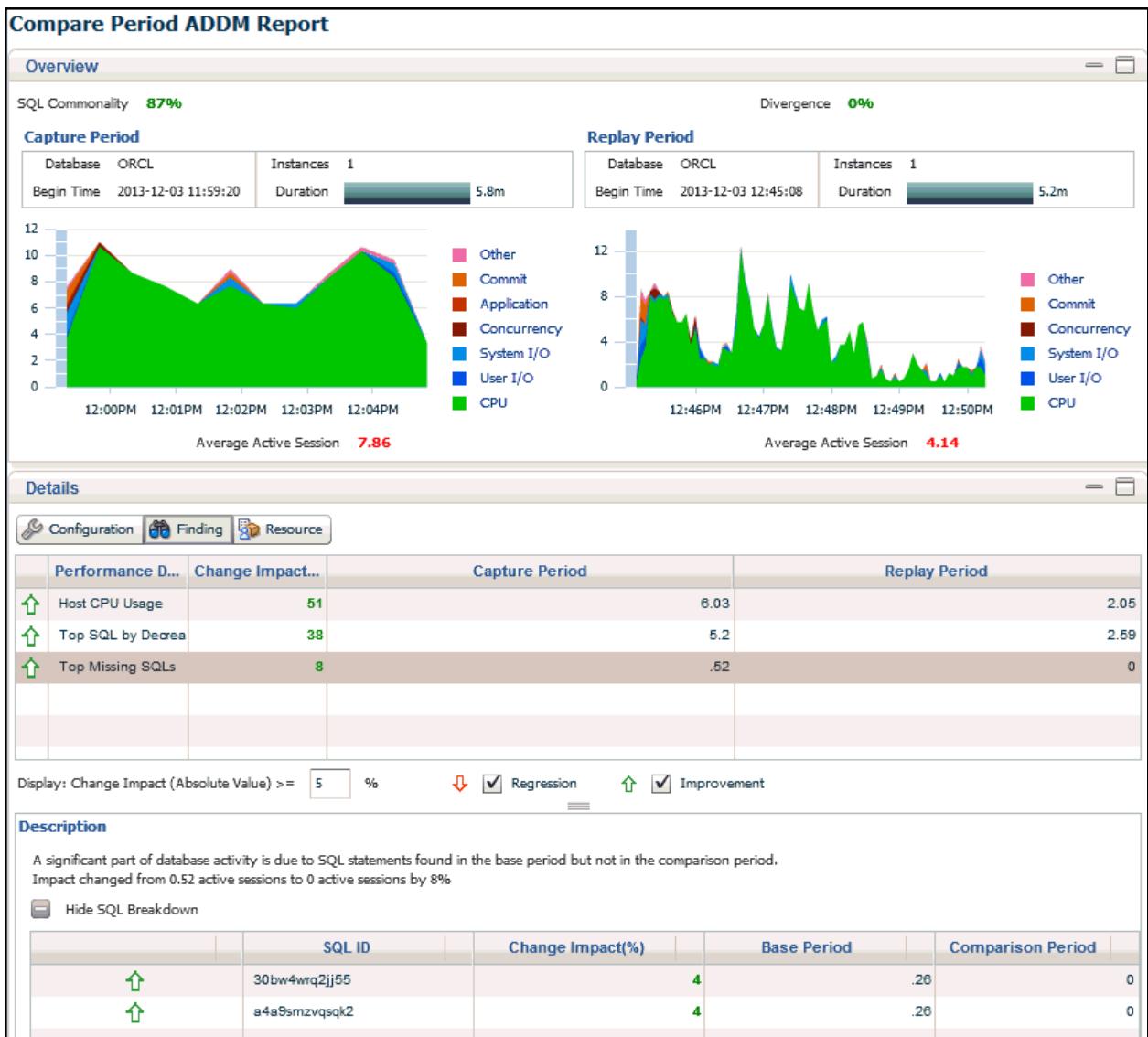
11. Create a replay task and start the replay. Navigate to the Replay Task: Replay_orcl_12c page. There are multiple navigation paths.

Step	Window/Page Description	Choices or Values
a.		Click Enterprise > Quality Management > Database Replay .
b.	Database Replay	Click the Replay Tasks tab. Click the Replay_orcl_12c link.
c.	Replay Task: Replay_orcl_12c	In the Replays section, click the Replay_1 link.
d.	Replay: Replay_1	Click Replay Workload .
e.	Replay Workload: Locate Workload	Click Use an existing workload directory on this host . Click Next .
f.	Replay Workload: Select Directory	Use pull down to select Directory Object: WORKLOAD . Verify the Capture Summary section is populated. Click Next .
g.	Replay Workload: Initialize Options	Click Next .
h.	Replay Workload: Customize Options	Click Next .
i.	Replay Workload: Prepare Replay Clients	In Number of Replay Clients and CPU Cores, click Estimate . The number of clients needed remains 1. Click Next .
j.	Replay Workload: Wait for Client Connections	Go to the terminal window.
k.	Terminal window	Switch to the terminal window logged in as the <code>oracle</code> user, with the <code>orcl</code> instance environment set. Start the workload client with <code>wrc <u>system/oracle 4U@orcl</u> mode=replay replaydir=/u01/app/oracle/capture/DBReplayW orkload_CAPTURE_11gSim_xx</code> Be sure to substitute your capture directory name 'xx'
l.	Replay Workload: Wait for Client Connections	When the client connection appears on the connection list, click Next .
m.	Replay Workload: Review	Click Submit .
n.	View Workload Replay: Replay_1	As the replay progresses, the chart changes when the page is manually refreshed. The time to run the replay depends heavily on the length of time the capture was running.

Note: The first updates of the replay graph may take a few minutes to appear on refresh.



12. Continue to refresh the Replay:Replay_1 page until the Status field shows **Completed**. This may take about 15 minutes (more if the capture is longer than 4 minutes). When the replay completes, observe Replay Divergence Summary. There may be some divergences reported. Ideally, there will be zero divergences. If there were INCOMPLETE FILES reported in the Workload Analyzer report, there will be divergences reported.
13. Examine the Replay Statistics section. It shows the differences in Database Time and User Calls. The replay may be faster or slower. This section does NOT show whether the workload is actually performing better or worse. Although, decrease in Database Time when the replay is finished will generally indicate that the replay was more efficient.
14. Create a Compare Period ADDM Report.
 - a. Click the **Reports** tab after the workload is completed. (The Reports tab is not available while the replay is running.)
 - b. Click the **View** button beside the Compare Period ADDM reports.



- c. Examine the SQL Commonality percentage at the top left of the report. This number shows the difference in the workload during the two periods. This number should be in the 80 through 100% range to be a valid comparison. Through a smaller commonality is still useful. The Top Missing SQL in the findings of this report will show some of the differences.
- d. Examine the Average Active Sessions below the graph for each of the Capture Period and the Replay Period. The smaller of the two values indicates the most efficient of the two periods, if the workload has a high commonality.
Record the values for Capture _____ and Replay_1 _____
- e. In the Details section, click the **Configuration** tab. This section shows clearly the differences between the configurations of the two periods.
- f. Click the **Finding** tab. Each Finding may be highlighted to show more detail about the finding in the Description section below the Details section.
- g. Close the browser window for the Compare Period ADDM Report

15. Create a Replay Compare Period Report.

Step	Window/Page Description	Choices or Values
a.	Replay: Replay_1	
b.	Replay: Replay_1 Reports tab	Click the View button beside Replay Compare Period Report.

16. When the report appears, review the report.

- a. Notice the difference between Duration (Elapsed time) and DB Time. Duration is in the Information about AWR and Time Periods section. DB Time is in the Main Performance Statistics section.

Note: Because of the short duration of the capture and the artificial nature of the workload, the Replay_1 report may show improvement, or regression.

	Duration (Elapsed Time)	DB Time
Capture		
Replay		

- b. Scroll down to the Top SQL/Call section. These are the SQL statements that have changed the most. The regressed statements should be the first ones tuned.
- c. Scroll down to the Common SQL Section, note that several SQL statements have regressed. These will be candidates for SQL Tuning.
- d. Close the Browser window for the Replay Compare Period Report.
- e. Return to the Replay:Replay_1 page.
- f. View the **SQL Performance Analyzer Task Report**

17. Run SQL Performance Analyzer on the SQL Tuning Set.

Step	Window/Page Description	Choices or Values
a.	SQL Performance Analyzer Task Report: *	Note: The report shows which SQL statements have regressed, and lists the Top 10 SQL statements based on impact on the workload.
b.	SQL Performance Analyzer Task Report: *	Click the SQL Tuning Set Name link for the SQL Trial 2 tuning set in the header of the SQL Performance Analyzer Task Report.
c.	SQL Tuning Set: "Replay_1_*	Click Select All . Click Schedule SQL Advisor .
d.	Schedule SQL Tuning Advisor	Click Submit .
e.	Processing: SQL Tuning Advisor Task SQL_TUNING_*	Observe the progress of the Advisor task. This will take from 4 to 10 minutes The page will advance to the Tuning Result as soon as the Task is finished.
f.	SQL Tuning Result Summary:SYS.SQL_TUNING_*	Review the findings. Notice The Profile Effects Statistics. This

Step	Window/Page Description	Choices or Values
		should show a benefit in DB Time. Scroll Down: in the Index Finding Summary, there are recommended indexes. These should be considered later using the SQL Access Advisor. Click Show all results .
g.	SQL Tuning Result Details: All Analyzed SQLs	Click Implement All SQL Profiles .
h.	Confirmation	Select Implement the new profile(s) with forced matching . Click Yes .

18. Perform a second replay having applied the SQL Profiles. Navigate to the Database Replay page.

Note: The second replay is expected to run in approximately the same time or less than the time required for the capture.

Step	Window/Page Description	Choices or Values
a.		Click Enterprise > Quality Management > Database Replay .
b.	Database Replay	Click the Replay_orcl_12c link.
c.	Replay Task: Replay_orcl_12c	In the Replays section, click Create .
d.	Create Replay	Name: Replay_2 Target Database: orcl Click Ok .
e.	Replay:Replay_2	Click Replay Workload .
f.	Database Login	Select Preferred . Select SYSDBA Database Credentials .
g.	Replay Workload: Locate Workload	Click Use an existing workload directory on this host .
h.	Replay Workload: Select Directory	Select WORKLOAD . Click Next .
i.	Replay Workload: Initialize Options	Click Next .
j.	Replay Workload: Customize Options	Click Next .
k.	Replay Workload: Prepare Replay Clients	Click Next .
l.	Replay Workload: Wait for Client Connections	Switch to the terminal window logged in as the oracle user, with the orcl instance environment set.

Step	Window/Page Description	Choices or Values
		<p>Start the workload client with <code>wrc <u>system/oracle 4U@orcl</u> mode=replay</code> <code>replaydir=/u01/app/oracle/capture/</code> <code>DBReplayWorkload_CAPTURE_11gSim_xx</code></p> <p>When the client connection appears on the connection list, click Next.</p>
m.	Replay Workload: Review	Click Submit .
n.	View Workload Replay: Replay_2	As the replay progresses, the chart changes when the page is refreshed. The time for the second replay should take less time than the capture.

19. View the Replay Compare Period Report.

Step	Window/Page Description	Choices or Values
a.	View Workload Replay: Replay_2	Click the Reports tab.
b.	View Workload Replay: Replay_2 Reports tab	Click the View button beside Replay Compare Period Report.
c.	Compare Period Report: Capture vs. Replay	Examine the report.

d. What is the overall performance difference?

It is expected that the DB Time for the replay is reduced. This indicates better performance.

	Duration (Elapsed Time)	DB Time
Capture		
Replay		

e. What can you determine about individual SQL statement performance?

The Common SQL section now shows none or few regressed SQL. The combination of reduced DB Time and improved top SQL would indicate that this configuration with SQL profiles and SQL Baselines will run more efficiently with the change to OPTIMIZER FEATURES ENABLED set to 12.1.0.1.

(Statistics on all SQL statements are not reported; only the top statements are reported. You could get some comparisons, but DB Replay is intended for overall performance comparison. SQL Performance Analyzer is intended for comparison of an entire set of SQL statements.)

f. Close the Compare Period report browser window.

20. View the Compare Periods ADDM Report. Notice the difference in the Average Active sessions number in the capture and Replay_2. Compare that to the difference in the Replay_1 and capture values you recorded in Step 14 d.

21. From the \$HOME/workshops directory, use the ./cleanup REPLAY command to reset the database environment for the next practice.

```
$ cd $HOME/workshops  
$ ./cleanup REPLAY
```


Practices for Lesson 16: Tuning the Shared Pool

Chapter 16

Practices for Lesson 16: Overview

Practices Overview

In this practice, you tune the shared pool.

Practice 16-1: Sizing the Shared Pool

In this practice, you use the ADDM reports to diagnose and fix the shared pool size.

1. Prepare the database instance for this scenario.

- a. Log out of Enterprise Manager.
- b. In a terminal window, run the `./prepare SP` command.

```
$ cd /home/oracle/workshops
$ ./prepare SP
The Oracle base for
ORACLE_HOME=/u01/app/oracle/product/12.1.0/dbhome_1 is
/u01/app/oracle

Database closed.
Database dismounted.
ORACLE instance shut down.
ORACLE instance started.

Total System Global Area  501059584 bytes
Fixed Size                  2290024 bytes
Variable Size                457182872 bytes
Database Buffers            33554432 bytes
Redo Buffers                 8032256 bytes
Database mounted.
Database opened.
Finished Shared Pool setup
$
```

2. In a terminal window, determine the machine time and run the workload generator. The `workgen` script creates an AWR snapshot and a Statspack snapshot, and then runs the workload. Record the time returned by the `date` command.

```
$ date
Tue Jul 16 07:59:37 UTC 2013
$ ./workgen SP

The Oracle base for
ORACLE_HOME=/u01/app/oracle/product/12.1.0/dbhome_1 is
/u01/app/oracle

PL/SQL procedure successfully completed.

PL/SQL procedure successfully completed.

Starting buffer cache load
Starting Soft Parse load
Starting Hard Parse load

System altered.

PL/SQL procedure successfully completed.
```

```
PL/SQL procedure successfully completed.
```

```
PL/SQL procedure successfully completed.
```

```
$
```

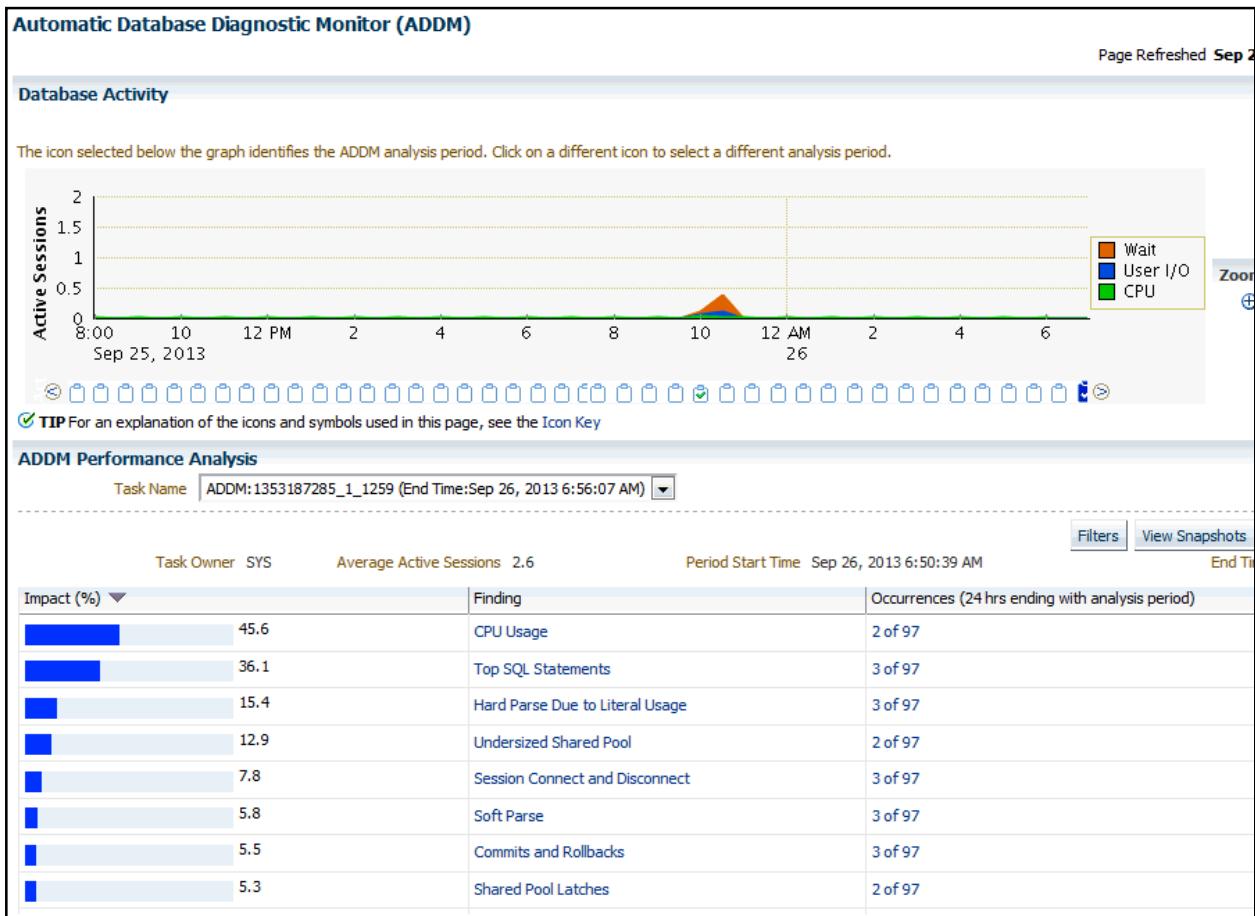
3. Check the date-time with the `date` command. Wait until five minutes have elapsed, and then continue. Stop the workload with the `rm runload` command.

Note: (Optional) While you are waiting, view the workload characteristics in EMCC on the `orcl` Performance Home page. Instructions for logging in are in the next step. Be sure to connect to the `orcl` database as `SYS`.

```
$ rm runload  
$
```

4. Use Enterprise Manager Cloud Control to:
 - a. Log in to Enterprise Manager Cloud Control.
 - b. Expand the Targets menu and select Databases.
 - c. Click the link for the `orcl` database.
 - d. Expand the Performance menu and select Performance Home.
 - e. On the Database Login page, enter `SYS` in the Username field and `oracle_4U` in the Password field, and then select `SYSDBA` from the role menu. Click Login.
 - f. Wait until the Average Active Sessions graph has dropped to almost 0. Create an ADDM report by clicking **Run ADDM Now**, and then click **Yes** on the Confirmation page. Notice any findings referring to the shared pool. These findings may include Hard Parse, Soft Parse, and Undersized Shared Pool.

The particular findings and the order of the findings may vary from the example shown.



5. Create an AWR report between the last two snapshots.
 - a. On the ADDM Report page, click View Snapshots to navigate to the snapshots page.
 - b. Click the Report tab to generate the AWR report.

6. View the AWR report and note the values for the Top 10 Foreground Events by Total Wait Time, Time Model Statistics, Instance Efficiencies, and the Library Cache Activity Statistics for the SQL AREA namespace.
- a. The Top 10 Timed Foreground Events may show time waiting on the library cache mutex, library cache load lock, shared pool latch, cursor pin waits, or row cache objects. Any of these indicate a possible problem in the shared pool.

Top 10 Foreground Events by Total Wait Time

Event	Waits	Total Wait Time (sec)	Wait Avg(ms)	% DB time	Wait Class
DB CPU		339.1		46.7	
log file sync	1,580	42.9	27	5.9	Commit
write complete waits	35	20.4	584	2.8	Configuration
direct path read	568,850	20	0	2.8	User I/O
cursor: pin S wait on X	625	10.7	17	1.5	Concurrency
latch: shared pool	7,813	10.5	1	1.4	Concurrency
enq: KO - fast object checkpoint	2	3.2	1595	.4	Application
db file sequential read	73,607	3	0	.4	User I/O
latch: row cache objects	1,438	2.6	2	.4	Concurrency
library cache: mutex X	422	1.3	3	.2	Concurrency

- b. Time Model Statistics, in this example, shows a significant % of DB Time being used for hard and soft parses (parse time elapsed). Parse time elapsed is hard parse elapsed time plus soft parse elapsed time.

Time Model Statistics

- Total time in database user-calls (DB Time): 725.5s
- Statistics including the word "background" measure background process time,
- Ordered by % of DB time desc, Statistic name

Statistic Name	Time (s)	% of DB Time
sql execute elapsed time	590.99	81.45
DB CPU	339.13	46.74
parse time elapsed	234.27	32.29
hard parse elapsed time	199.82	27.54
connection management call elapsed time	57.83	7.97
PL/SQL execution elapsed time	21.97	3.03
PL/SQL compilation elapsed time	7.92	1.09
hard parse (sharing criteria) elapsed time	3.32	0.46
failed parse elapsed time	1.69	0.23
sequence load elapsed time	0.61	0.08
hard parse (bind mismatch) elapsed time	0.28	0.04
repeated bind elapsed time	0.09	0.01
DB time	725.55	
background elapsed time	106.40	
background cpu time	5.92	

- c. The latch and mutex waits taken with the hard parses shown in Time Model leads to a check of the Load Profile section in the Report Summary section. In this example, hard parses are a significant percentage of all parses.

Load Profile				
	Per Second	Per Transaction	Per Exec	Per Call
DB Time(s):	1.7	0.0	0.00	0.00
DB CPU(s):	0.8	0.0	0.00	0.00
Redo size (bytes):	98,115.7	2,531.4		
Logical read (blocks):	66,851.5	1,724.8		
Block changes:	679.8	17.5		
Physical read (blocks):	10,799.4	278.6		
Physical write (blocks):	18.7	0.5		
Read IO requests:	1,650.9	42.6		
Write IO requests:	13.1	0.3		
Read IO (MB):	84.4	2.2		
Write IO (MB):	0.2	0.0		
User calls:	2,481.4	64.0		
Parses (SQL):	699.9	18.1		
Hard parses (SQL):	194.8	5.0		
SQL Work Area (MB):	17.4	0.5		
Logons:	3.6	0.1		
Executes (SQL):	4,912.5	126.7		
Rollbacks:	0.0	0.0		
Transactions:	38.8			

- d. Navigate to Library Cache Statistics from the Main Report menu. Check the Reloads statistic of the SQL AREA namespace in the Library Cache Activity Statistics section. The hard parses can be caused by SQL that cannot be shared, or by cursors aging out of the shared pool before they can be reused. This example shows a high value for reloads and a high Pct Miss for Gets and Pins. This indicates that the cursor could have been shared, but the shared pool is too small. Reloads should be less than 1% of Pin Requests.

Library Cache Activity

- "Pct Misses" should be very low

Namespace	Get Requests	Pct Miss	Pin Requests	Pct Miss	Reloads	Invali- dations
ACCOUNT_STATUS	7,551	2.83	0		0	0
AUDIT POLICY	1,512	4.50	1,512	4.83	0	0
BODY	5,218	2.03	23,023	0.90	85	0
CLUSTER	1,708	1.11	1,712	1.11	0	0
DBLINK	4,535	1.48	0		0	0
EDITION	1,523	4.53	4,556	4.74	0	0
INDEX	35,885	0.16	35,885	0.86	249	0
OBJECT ID	17	100.00	0		0	0
QUEUE	1	0.00	169	9.47	8	0
SCHEMA	6,005	1.08	0		0	0
SQL AREA	266,787	25.56	2,302,656	9.85	12,767	8,788
SQL AREA BUILD	79,528	91.49	0		0	0
SQL AREA STATS	77,824	89.42	77,825	89.42	0	0
TABLE/PROCEDURE	247,910	1.18	542,004	3.15	8,453	0
TRIGGER	51	11.76	11,040	0.06	1	0

- e. Further investigation of the dictionary cache shows that the only significant misses in the dictionary cache are with histograms. Click Back to Top, then click Dictionary Cache Statistics in the Main Report menu.

Note: A high miss percentage associated with a small number of requests, such as with dc_sequences is NOT significant.

Dictionary Cache Stats

- "Pct Misses" should be very low (< 2% in most cases)
- "Final Usage" is the number of cache entries being used

Cache	Get Requests	Pct Miss	Scan Reqs	Pct Miss	Mod Reqs	Final Usage
dc_awr_control	39	89.74	0		2	5
dc_files	272	75.00	0		0	17
dc_global_oids	27,722	1.18	0		0	14
dc_histogram_data	117,526	19.35	0		0	992
dc_histogram_defs	1,229,731	2.48	0		352	1,261
dc_object_grants	13,094	1.26	0		0	13
dc_objects	2,216,465	0.30	0		69	406
dc_profiles	3,023	0.10	0		0	0
dc_props	3,317	0.33	0		0	1
dc_rollback_segments	482	0.00	0		0	73
dc_segments	450,049	2.39	0		4	186
dc_sequences	130	70.77	0		130	0
dc_tablespace_quotas	8	50.00	0		0	0
dc tablespaces	339,386	0.02	0		0	5
dc_users	957,092	0.03	6,048	0.60	0	24
kqlsubheap_object	7,186	0.28	0		0	0
qtmrciq_cache_entries	57,456	2.19	0		0	93
sch_li_oids	2	50.00	0		0	0

7. Find an optimum shared pool size for this workload. The solution can be obtained by using Enterprise Manager Cloud Control or the Shared Pool Advisory in the Statspack report.
- a. Select **Performance > Memory Advisors**.

Memory Advisors

When Automatic Memory Management is enabled, the database will automatically set the optimal distribution of memory. The distribution of memory is based on the current workload.

Automatic Memory Management Disabled [Enable](#)

[SGA](#) [PGA](#)

The System Global Area (SGA) is a group of shared memory structures that contains data and control information for one Oracle database. The SGA is divided into several components:

Automatic Shared Memory Management Disabled [Enable](#)

Shared Pool	200	MB	Advice
Buffer Cache	28	MB	Advice
Large Pool	12	MB	
Java Pool	4	MB	
Other (MB)	13		
Total SGA (MB)	257		Calculate

Maximum SGA Size

The Maximum SGA Size specifies the maximum memory that the database may allocate. If you specify the Maximum SGA Size, you can later change it without restarting the database (as long as it does not exceed the Maximum SGA Size).

Maximum SGA Size (MB)

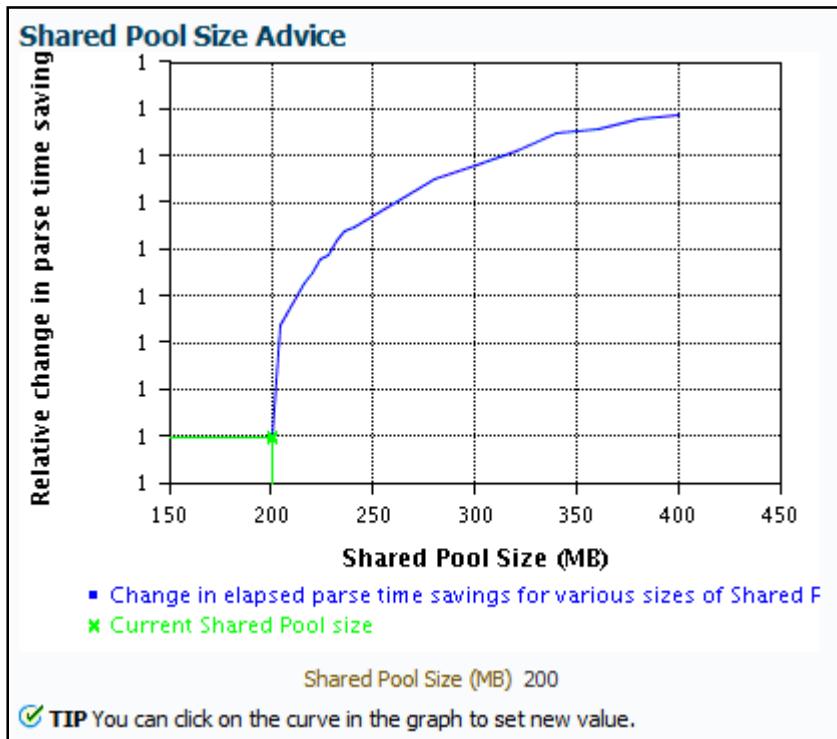
The database must be restarted before any changes to this value take effect.

Apply changes to SPFILE only

The changes are made to both the SPFILE and the running instance which requires that you restart the database to invoke static parameters.

- b. Click Shared Pool Advice. The graph shows performance versus shared pool size. The point where the curve flattens is where there is no more improvement in performance as the shared pool size is increased. (The graph generated for your instance may vary.) Make a note of this point for your instance. Click Cancel.

Note: This graph displays estimates based on statistics collected during the previous workload. If the workload changes, so do the estimates. The advice curve you see may vary from the example shown.



- c. On the Memory Advisors Page, change the Shared Pool value to 280 MB. Click Calculate.
 What is the calculated value? _____
 If the calculated value (Total SGA) **is less than** the Maximum SGA size, click Apply.
 If the calculated value (Total SGA) **is greater than** the Maximum SGA size, an "Insufficient memory to grow pool" error message appears.
 Change the Maximum SGA size to the calculated value + 2 MB. Click **Apply**.
- Note:** The SHARED_POOL_SIZE parameter can be increased but not decreased dynamically. This is new behavior in Oracle Database 12c.
8. Use the prepare script to change the SHARED_POOL_SIZE parameter..
- Log out of EMCC.
 - Run the ./prepare SP2 script from the \$HOME/workshops directory. This script changes the SHARED_POOL_SIZE parameter to 280 MB in the SPFILE. Because the change is to the SPFILE, the change requires the database to be restarted.

```
$ ./prepare SP2
The Oracle base for
ORACLE_HOME=/u01/app/oracle/product/12.1.0/dbhome_1 is
/u01/app/oracle

Database closed.
```

```

Database dismounted.
ORACLE instance shut down.
ORACLE instance started.

Total System Global Area  354951168 bytes
Fixed Size                  2288248 bytes
Variable Size                314574216 bytes
Database Buffers            33554432 bytes
Redo Buffers                 4534272 bytes
Database mounted.
Database opened.
Finished Shared Pool second setup
$
```

9. Run the workload generator again (`./workgen SP`) for five minutes, and then stop the workload with the `rm unload` command.
 - a. Find the system time so that you can determine when five minutes have passed, and then start the workload.

```

$ date
Tue Jul 16 11:17:28 UTC 2013
$ ./workgen SP

The Oracle base for
ORACLE_HOME=/u01/app/oracle/product/12.1.0/dbhome_1 is
/u01/app/oracle

PL/SQL procedure successfully completed.

PL/SQL procedure successfully completed.

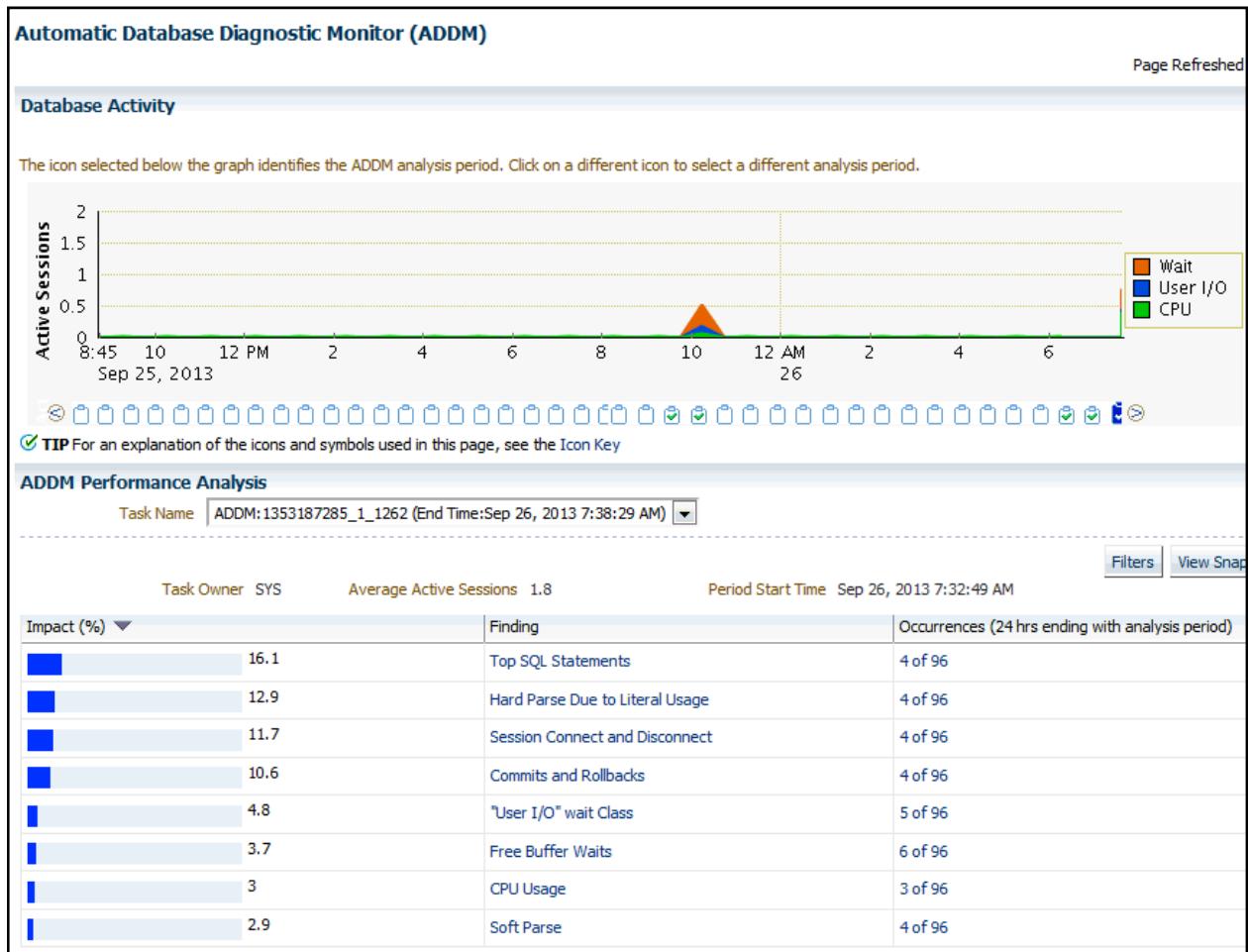
$
```

- b. After five minutes, in another terminal window, stop the workload by removing the `unload` file.

```

$ cd /home/oracle/workshops
$ rm unload
```

10. Create another ADDM task and review the findings.
 - a. Log out of the `orcl` database in Enterprise Manager Cloud Control, and log in to the `orcl` database as `SYS`. This avoids possible errors due to restarting the `orcl` database while EMCC is connected.
 - b. Expand the Targets menu and select Databases.
 - c. Click the link for the `orcl` database.
 - d. Expand the Performance menu and select Performance Home.
 - e. Create another ADDM task by clicking **Run ADDM Now**. Click **Yes** to confirm.
 - f. What is the difference compared with your earlier ADDM task? What are the primary issues now? *Notice the findings. The hard-parse findings have been reduced or eliminated from the findings. There may be no findings.*



11. Create an AWR report.
 - a. On the ADDM page, click View Snapshots to navigate to the Snapshot Details page.
 - b. Click the Report tab to create an AWR report.

12. View the AWR report and compare the values from the previous report.

Note: The actual values that you see may vary from those shown in the solutions.

- a. In the Top 10 Foreground Events by Total Wait Time, the latch and mutex waits for the shared pool and cursors are reduced or eliminated.

Top 10 Foreground Events by Total Wait Time						
Event	Waits	Total Wait Time (sec)	Wait Avg(ms)	% DB time	Wait Class	
DB CPU		317.1		52.5		
log file sync	1,673	64.2	38	10.6	Commit	
write complete waits	26	22.3	858	3.7	Configuration	
direct path read	572,956	17.3	0	2.9	User I/O	
enq: KO - fast object checkpoint	6	6.4	1065	1.1	Application	
db file sequential read	101,122	5.3	0	.9	User I/O	
buffer busy waits	408	2.9	7	.5	Concurrency	
db file scattered read	47,339	2.4	0	.4	User I/O	
latch: shared pool	1,778	2.4	1	.4	Concurrency	
Disk file operations I/O	2,263	2	1	.3	User I/O	

- b. Time Model (in the example) shows that hard parses are still contributing to parse elapsed time, but take a very small % of DB Time. Your values may vary.

Time Model Statistics

Statistic Name	Time (s)	% of DB Time
sql execute elapsed time	489.10	81.03
DB CPU	317.06	52.53
parse time elapsed	101.68	16.85
hard parse elapsed time	83.90	13.90
connection management call elapsed time	70.43	11.67
PL/SQL execution elapsed time	18.46	3.06
failed parse elapsed time	1.47	0.24
hard parse (sharing criteria) elapsed time	1.31	0.22
PL/SQL compilation elapsed time	0.82	0.14
sequence load elapsed time	0.02	0.00
repeated bind elapsed time	0.02	0.00
hard parse (bind mismatch) elapsed time	0.00	0.00
DB time	603.58	
background elapsed time	129.88	
background cpu time	4.65	

- c. The Load Profile section shows that hard parses are reduced (as are all parses).

Load Profile		Per Second	Per Transaction	Per Exec	Per Call
DB Time(s):		1.8	0.0	0.00	0.00
DB CPU(s):		0.9	0.0	0.00	0.00
Redo size (bytes):		125,970.0	2,679.3		
Logical read (blocks):		89,371.5	1,900.8		
Block changes:		867.5	18.5		
Physical read (blocks):		14,781.5	314.4		
Physical write (blocks):		25.0	0.5		
Read IO requests:		2,455.8	52.2		
Write IO requests:		18.2	0.4		
Read IO (MB):		115.5	2.5		
Write IO (MB):		0.2	0.0		
User calls:		3,075.7	65.4		
Parses (SQL):		743.7	15.8		
Hard parses (SQL):		220.5	4.7		
SQL Work Area (MB):		21.7	0.5		
Logons:		4.6	0.1		
Executes (SQL):		5,714.9	121.6		
Rollbacks:		0.0	0.0		
Transactions:		47.0			

- d. In the Library Cache Activity section, reloads for the SQL AREA namespace are now close to an acceptable range. The % of Reloads versus Pin Requests have been reduced. The value is still high enough to warrant additional tuning.

Library Cache Activity

Namespace	Get Requests	Pct Miss	Pin Requests	Pct Miss	Reloads	Inval- dations
ACCOUNT_STATUS	7,721	0.39	0		0	0
AUDIT POLICY	1,546	0.13	1,546	0.19	0	0
BODY	5,465	0.81	23,003	0.37	38	0
CLUSTER	372	1.88	372	1.88	0	0
DBLINK	4,637	0.04	0		0	0
EDITION	1,564	0.13	4,672	0.15	0	0
INDEX	35,498	0.08	35,498	0.26	64	0
OBJECT ID	22	100.00	0		0	0
QUEUE	7	0.00	143	2.80	3	0
SCHEMA	3,519	0.68	0		0	0
SQL AREA	204,488	32.27	2,130,051	10.07	8,885	9,483
SQL AREA BUILD	74,431	89.03	0		0	0
SQL AREA STATS	73,997	89.17	73,998	89.17	0	0
TABLE/PROCEDURE	227,898	0.52	472,770	1.02	2,048	0
TRIGGER	50	14.00	10,578	0.07	0	0

- e. The Dictionary Cache Stats section shows a much reduced percentage of misses.
- Dictionary Cache Stats

Cache	Get Requests	Pct Miss	Scan Reqs	Pct Miss	Mod Reqs	Final Usage
dc_awr_control	45	20.00	0		2	5
dc_files	204	25.00	0		0	17
dc_global_oids	26,289	0.31	0		0	58
dc_histogram_data	58,345	5.23	0		0	1,768
dc_histogram_defs	371,644	1.76	0		798	3,210
dc_object_grants	10,399	0.70	0		0	32
dc_objects	2,112,536	0.09	0		91	1,156
dc_profiles	3,095	0.03	0		0	1
dc_props	3,337	0.06	0		0	2
dc_rollback_segments	496	0.00	0		0	73
dc_segments	415,721	0.93	0		1	1,168
dc_sequences	72	18.06	0		72	3
dc_tablespace_quotas	12	33.33	0		0	2
dc tablespaces	68,728	0.03	0		0	19
dc_users	854,931	0.01	6,184	0.36	0	47
kqlsubheap_object	7,097	0.06	0		0	2
outstanding_alerts	18	100.00	0		0	18
qmtmrciq_cache_entries	57,288	0.59	0		0	168
sch_li_oids	11	81.82	0		0	9

Practice 16-2: Tuning a Hard-Parse Workload

In this section, you tune a hard-parse workload, view the characteristic report events, and fix the problem.

1. Prepare the database for this scenario.
 - a. Log out of Enterprise Manager.
 - b. Set up the scenario by running the `prepare` script with the `HARDPARSE` parameter. Change to the `/home/oracle/workshops` directory, and then execute the `prepare` script.

```
$ cd /home/oracle/workshops
$ ./prepare HARDPARSE
The Oracle base for
ORACLE_HOME=/u01/app/oracle/product/12.1.0/dbhome_1 is
/u01/app/oracle

Database closed.
Database dismounted.
ORACLE instance shut down.

ORACLE instance started.

Total System Global Area  354951168 bytes
Fixed Size                  2288248 bytes
Variable Size                314574216 bytes
Database Buffers              33554432 bytes
Redo Buffers                  4534272 bytes
Database mounted.
Database opened.
Finished setup Hard Parse
$
```

2. Run the workload generator: `./workgen HARDPARSE`.

The workload generator creates an AWR snapshot and a Statspack snapshot at the beginning. Let this workload run for about five minutes, then stop it with `rm unload`.

```
$ ./workgen HARDPARSE
The Oracle base for
ORACLE_HOME=/u01/app/oracle/product/12.1.0/dbhome_1 is
/u01/app/oracle

PL/SQL procedure successfully completed.

PL/SQL procedure successfully completed.

System altered.

PL/SQL procedure successfully completed.

PL/SQL procedure successfully completed.

PL/SQL procedure successfully completed.

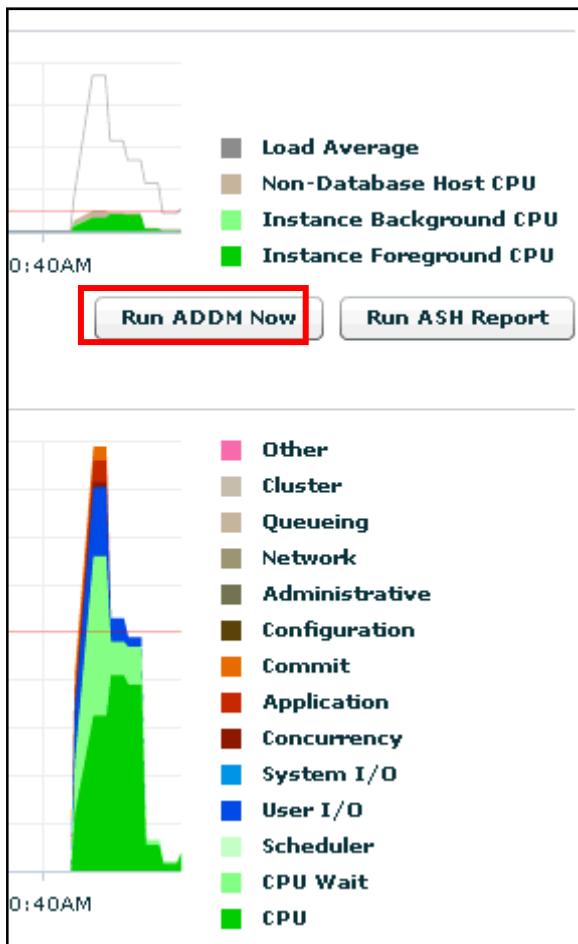
$ rm unload

$
```

3. Generate an ADDM task and report.

- a. Log in to Enterprise Manager Cloud Control.
- b. Expand Targets and select Databases.
- c. Click the `orcl` database link.
- d. Expand the Performance menu and select Performance Home.
- e. Log in to the `orcl` database.

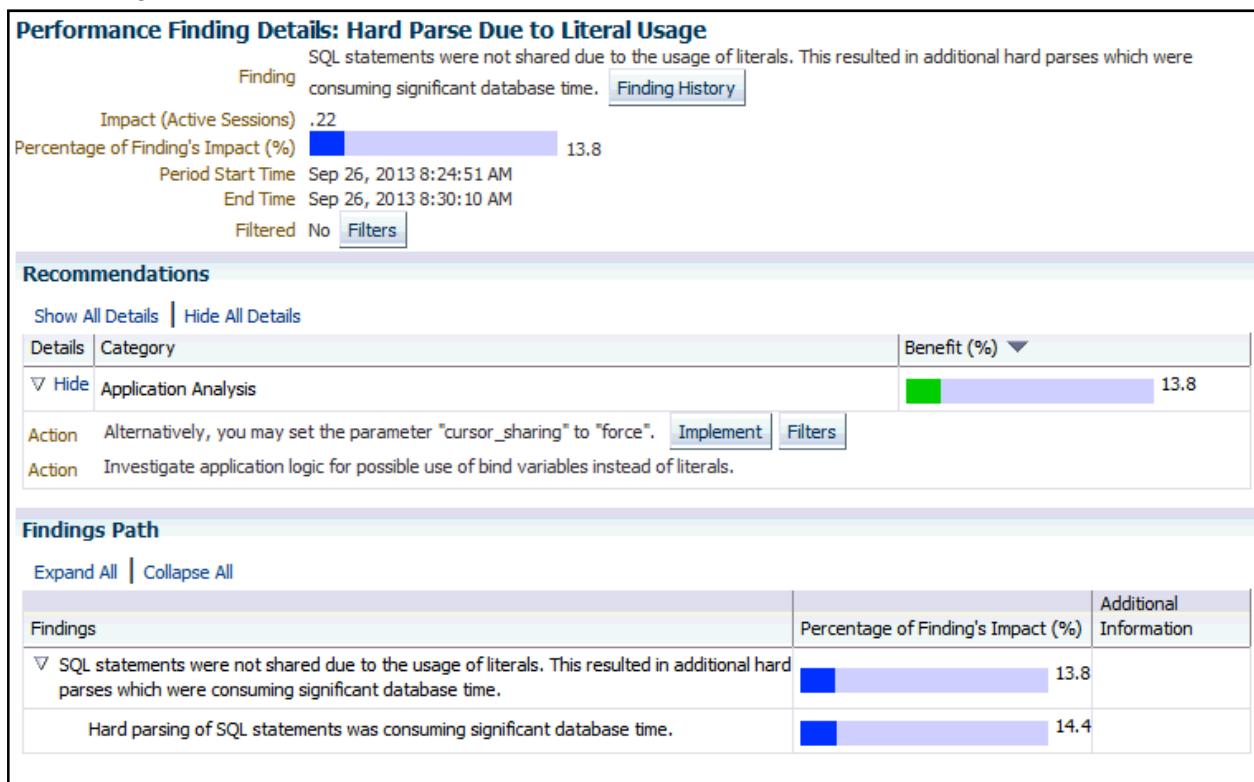
- f. Wait until the workload begins to drop (about five minutes), and then click **Run ADDM Now**. Click **Yes** on the Confirmation page.



- g. On the Automatic Database Diagnostics Monitor page, what is the value of Average Active Sessions? _____

ADDM Performance Analysis		
Task Name ADDM:1353187285_1_1268 (End Time:Sep 26, 2013 8:30:10 AM)		
Task Owner	SYS	Average Active Sessions 1.6
Impact (%) ▾		Period Start Time Sep 26, 2013 8:24:51 AM
23.9	Top SQL Statements	Occurrences (24) 6 of 97
13.8	Hard Parse Due to Literal Usage	6 of 97
12.1	Session Connect and Disconnect	7 of 97
11.2	Commits and Rollbacks	7 of 97
3.9	Free Buffer Waits	9 of 97
3.2	Top Segments by "User I/O" and "Cluster"	1 of 97
3.1	Soft Parse	6 of 97

- h. Drill down to the recommendations. Click the “Hard Parse Due to Literal Usage” finding.



- i. Based on the ADDM report, what is the recommended action?

The recommendation is to “set the CURSOR_SHARING parameter to FORCE.”

4. Create an AWR or Statspack report from the last two snapshots. The following table shows the solution using AWR and Enterprise Manager Cloud Control.

Step	Window/Page Description	Choices or Values
a.	Current page	Navigate to the Snapshots page. Performance > AWR > AWR Administration Click Snapshots link
b.	Snapshots	Select the next to last snapshot. Select View Report from Actions. Click Go .
c.	View Report	Select the last snapshot. Click OK .

5. Review the AWR report for the characteristics of a hard-parse problem. The goal is to reduce hard parses because the cost of a hard parse is at least an order of magnitude greater than the cost of a soft parse.
- a. The Top 10 Foreground Event by Total Wait Time shows that there are no significant wait events. CPU Time by itself does not indicate a problem. The larger %DB Time

values are associated with I/O Waits that indicate the database is performing under a load.

Top 10 Foreground Events by Total Wait Time					
Event	Waits	Total Wait Time (sec)	Wait Avg(ms)	% DB time	Wait Class
DB CPU		290.6		56.9	
log file sync	1,595	57	36	11.2	Commit
write complete waits	31	19.9	643	3.9	Configuration
direct path read	533,912	15.2	0	3.0	User I/O
enq: KO - fast object checkpoint	3	2.4	788	.5	Application
db file sequential read	54,361	2.3	0	.5	User I/O
Disk file operations I/O	1,318	1.5	1	.3	User I/O
latch: shared pool	1,427	1.4	1	.3	Concurrency
buffer busy waits	141	1.4	10	.3	Concurrency
SQL*Net break/reset to client	15,600	.8	0	.2	Application

- b. Time Model gives the first indication of a problem. Parse time elapsed is a significant portion of DB Time, and hard parse elapsed time is the majority of the parse time elapsed.

Time Model Statistics		
Statistic Name	Time (s)	% of DB Time
sql execute elapsed time	410.86	80.43
DB CPU	290.59	56.89
parse time elapsed	89.57	17.54
hard parse elapsed time	73.66	14.42
connection management call elapsed time	61.98	12.13
PL/SQL execution elapsed time	16.13	3.16
failed parse elapsed time	1.36	0.27
hard parse (sharing criteria) elapsed time	0.85	0.17
PL/SQL compilation elapsed time	0.70	0.14
sequence load elapsed time	0.05	0.01
repeated bind elapsed time	0.02	0.00
DB time	510.83	
background elapsed time	103.73	
background cpu time	3.52	

- c. In the Load Profile section, compare Parses per second to Hard parses per second. A large portion of the parses are hard parses.

Load Profile				
	Per Second	Per Transaction	Per Exec	Per Call
DB Time(s):	1.6	0.0	0.00	0.00
DB CPU(s):	0.9	0.0	0.00	0.00
Redo size (bytes):	117,747.7	2,572.4		
Logical read (blocks):	88,857.9	1,941.3		
Block changes:	820.9	17.9		
Physical read (blocks):	13,325.8	291.1		
Physical write (blocks):	23.2	0.5		
Read IO requests:	1,947.0	42.5		
Write IO requests:	16.6	0.4		
Read IO (MB):	104.1	2.3		
Write IO (MB):	0.2	0.0		
User calls:	3,125.3	68.3		
Parses (SQL):	716.6	15.7		
Hard parses (SQL):	232.5	5.1		
SQL Work Area (MB):	21.3	0.5		
Logons:	4.7	0.1		
Executes (SQL):	5,829.8	127.4		
Rollbacks:	0.0	0.0		
Transactions:	45.8			

- d. Further investigation into the Instance Efficiencies Percentages section is not conclusive. It shows that % Non-Parse CPU is lower than it could be, indicating that a significant portion of the CPU time is spent parsing. The Soft Parse %: shows that few statements are found in the cache.

Instance Efficiency Percentages (Target 100%)		
Buffer Nowait %:	100.00	Redo NoWait %:
Buffer Hit %:	99.62	In-memory Sort %:
Library Hit %:	89.38	Soft Parse %:
Execute to Parse %:	87.71	Latch Hit %:
Parse CPU to Parse Elapsd %:	58.10	% Non-Parse CPU:

6. Implement the ADDM recommendation by clicking the Implement button on the ADDM Finding Details page or by executing the `ALTER SYSTEM SET CURSOR_SHARING = FORCE SCOPE=BOTH` command.
 - a. Using the SQL*Plus interface, execute the `ALTER SYSTEM SET CURSOR_SHARING = FORCE SCOPE=BOTH` command.

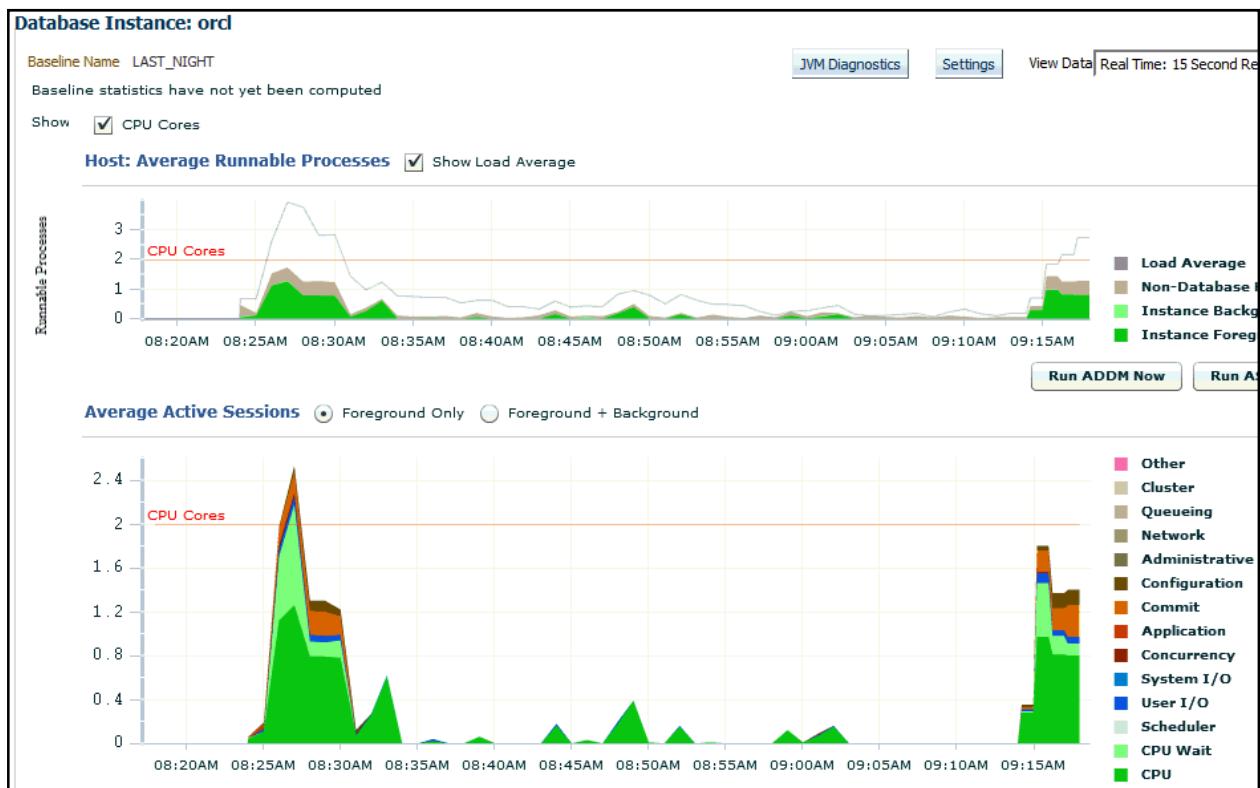
```
$ sqlplus / as sysdba

...
SQL> ALTER SYSTEM SET CURSOR_SHARING = FORCE SCOPE=BOTH;

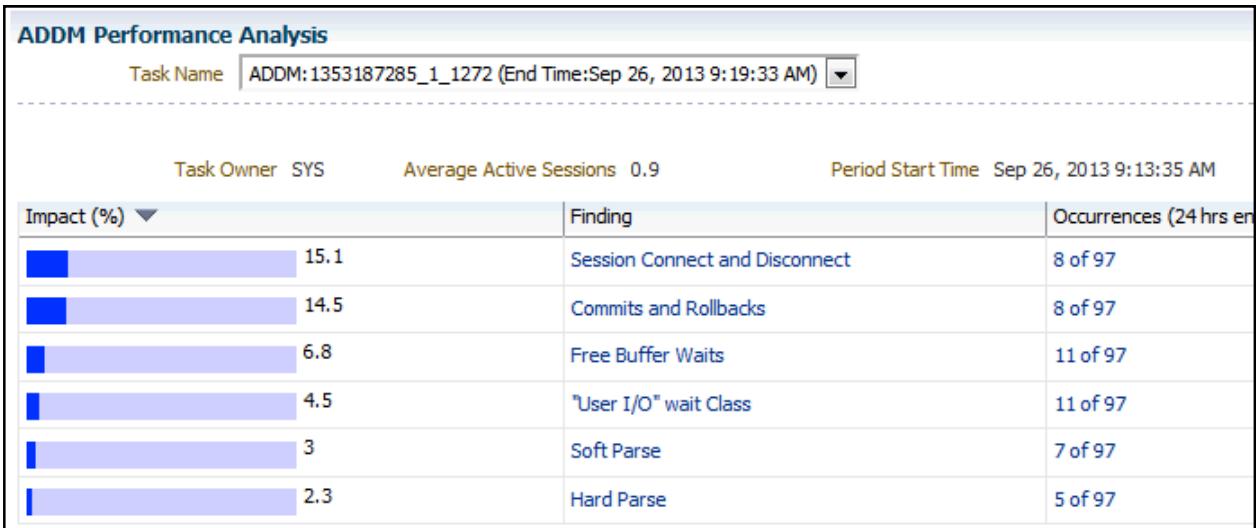
System altered.

SQL> exit
$
```

7. Run the workload generator again and observe the differences.
 - a. Execute the `workgen` script with the `HARDPARSE` parameter.
 - b. After about three minutes, run the `rm runload` command,
 - c. Run ADDM from the Performance page as in step 2. If you are watching the Average Active Sessions graph on the Performance Home page, you should see that the load graph is reduced.



- d. Using the ADDM tool, you see the impact of changing the CURSOR_SHARING parameter. The performance penalty of hard-parsing has dropped some. The hard-parse finding may not appear in the ADDM report. It is possible that there will be no findings. Notice that the value for Average Active Sessions has dropped significantly. This indicates that fewer sessions are waiting.



- e. Create an AWR report. On the ADDM page, click View Snapshots. On the Snapshot detail page, click the Report tab.

- f. Using the AWR report, review the significant statistics. Notice that all significant statistics show improvement over the report from the previous workload run. Notice that hard parses are a much smaller percentage of the parses, Soft Parse % is much improved, and % Non-Parse CPU is much higher. You may have also noticed that the workload script ran more quickly as well.

Load Profile				
	Per Second	Per Transaction	Per Exec	Per Call
DB Time(s):	0.9	0.0	0.00	0.00
DB CPU(s):	0.5	0.0	0.00	0.00
Redo size (bytes):	78,397.7	2,720.4		
Logical read (blocks):	58,057.3	2,014.6		
Block changes:	526.9	18.3		
Physical read (blocks):	8,275.0	287.2		
Physical write (blocks):	16.6	0.6		
Read IO requests:	1,631.1	56.6		
Write IO requests:	12.5	0.4		
Read IO (MB):	64.7	2.2		
Write IO (MB):	0.1	0.0		
User calls:	1,904.0	66.1		
Parses (SQL):	403.0	16.8		
Hard parses (SQL):	20.3	0.7		
SQL Work Area (MB):	13.2	0.5		
Logons:	2.9	0.1		
Executes (SQL):	3,763.9	130.6		
Rollbacks:	0.0	0.0		
Transactions:	28.8			

Note: The % DB Time for parse time elapsed and for hard parse elapsed time are both reduced.

Time Model Statistics

- Total time in database user-calls (DB Time): 309.8s
- Statistics including the word "background" measure background processes
- Ordered by % or DB time desc, Statistic name

Statistic Name	Time (s)	% of DB Time
sql execute elapsed time	238.10	76.84
DB CPU	180.77	58.34
connection management call elapsed time	46.84	15.12
parse time elapsed	16.52	5.33
PL/SQL execution elapsed time	11.21	3.62
hard parse elapsed time	7.18	2.32
failed parse elapsed time	0.86	0.28
PL/SQL compilation elapsed time	0.53	0.17
hard parse (sharing criteria) elapsed time	0.29	0.09
sequence load elapsed time	0.04	0.01
repeated bind elapsed time	0.01	0.00
DB time	309.85	
background elapsed time	85.88	
background cpu time	2.59	

Note: The Soft Parse % and % Non-Parse CPU are improved.

Instance Efficiency Percentages (Target 100%)

Buffer Nowait %:	100.00	Redo NoWait %:	99.99
Buffer Hit %:	98.67	In-memory Sort %:	100.00
Library Hit %:	98.74	Soft Parse %:	95.80
Execute to Parse %:	87.17	Latch Hit %:	99.99
Parse CPU to Parse Elapsd %:	43.92	% Non-Parse CPU:	96.66

Practice 16-3: Keeping Objects in the Shared Pool

The performance of some applications is hindered by having to reload frequently used objects into the shared pool. If an object (such as a procedure, package, or sequence) is used often, but not frequently enough to prevent aging out of the shared pool, keeping it will reduce the number of reloads. Keeping objects that are used very frequently is not necessary, but it does not hurt performance because frequently accessed objects will tend to stay in the shared pool without being kept.

1. Start the workload generator: `./workgen KEEP`.

```
$ ./workgen KEEP
```

2. After the workload has run for a few minutes, choose a candidate object to `KEEP` in the shared pool. Write and save a query to find a candidate object. Write the query to test whether the object has been `KEPT`. A suggested query is:

```
SELECT name, type, sharable_mem, kept
  FROM v$db_object_cache
 WHERE sharable_mem > 4000
   AND EXECUTIONS > 5
   AND (type='FUNCTION' OR type='PROCEDURE')
  /

```

This query is available in the `$HOME/workshops` directory as `sp_objects.sql`. The results of the query will vary. The `DEPLET_INV` procedure should appear in all queries.

```
$ sqlplus / as sysdba

SQL> @sp_objects.sql
SQL> set echo on
SQL>
SQL> set pagesize 100
SQL> column Name format A20
SQL> column type format A12
SQL>
SQL> SELECT name, type, sharable_mem, kept
  2      FROM v$db_object_cache
  3      WHERE sharable_mem > 4000
  4      AND EXECUTIONS > 5
  5      AND (type='FUNCTION' OR type='PROCEDURE')
  6  /

```

NAME	TYPE	SHARABLE_MEM	KEP
IS_VPD_ENABLED	FUNCTION	24552	NO
SQL_TXT	FUNCTION	8168	NO
DICTIONARY_OBJ_NAME	FUNCTION	24552	NO
DVLANG	FUNCTION	32744	NO
DICTIONARY_OBJ_OWNER	FUNCTION	24552	NO
DEPLET_INV	PROCEDURE	24552	NO
AW_TRUNC_PROC	PROCEDURE	28648	NO
DICTIONARY_OBJ_TYPE	FUNCTION	24552	NO
...			

3. Use the KEEP procedure to keep the DEPLET_INV PROCEDURE object in the shared pool.

```
SQL> EXECUTE dbms_shared_pool.keep('OE.DEPLET_INV') ;
```

```
PL/SQL procedure successfully completed.
```

```
SQL> @sp_objects.sql
```

```
SQL> set echo on
SQL> set pagesize 100
SQL> column Name format A20
SQL> column type format A12
SQL>
SQL> SELECT name, type, sharable_mem, kept
  2      FROM v$db_object_cache
  3      WHERE sharable_mem > 4000
  4      AND EXECUTIONS > 5
  5      AND (type='FUNCTION' OR type='PROCEDURE')
  6  /
```

NAME	TYPE	SHARABLE_MEM	KEP
IS_VPD_ENABLED	FUNCTION	8168	NO
SQL_TXT	FUNCTION	8168	NO
DICTIONARY_OBJ_NAME	FUNCTION	8168	NO
DVLANG	FUNCTION	32744	NO
DICTIONARY_OBJ_OWNER	FUNCTION	8168	NO
DEPLET_INV	PROCEDURE	24552	YES
AW_TRUNC_PROC	PROCEDURE	8168	NO
DICTIONARY_OBJ_TYPE	FUNCTION	8168	NO
...			

4. Flush the shared pool and observe the state of KEPT objects.

```
SQL> ALTER SYSTEM FLUSH SHARED_POOL;
```

```
System altered.
```

```
SQL> @sp_objects
SQL> set echo on
SQL>
SQL> set pagesize 100
SQL> column Name format A20
SQL> column type format A12
SQL>
SQL> SELECT name, type, sharable_mem, kept
  2      FROM v$db_object_cache
  3      WHERE sharable_mem > 4000
  4      AND EXECUTIONS > 5
  5      AND (type='FUNCTION' OR type='PROCEDURE')
  6  /
```

NAME	TYPE	SHARABLE_MEM	KEP
DEPLETETE_INV	PROCEDURE	24552	YES
SQL> exit			

5. Stop the workload generator by executing the `rm runload` command.

```
$ rm runload
```

6. Clean up the database by running the `./cleanup SP` script.

```
$ ./cleanup SP
```

```
The Oracle base for  
ORACLE_HOME=/u01/app/oracle/product/12.1.0/dbhome_1 is  
/u01/app/oracle
```

```
Cleanup Finished  
$
```


Practices for Lesson 17: Tuning the Buffer Cache

Chapter 17

Practices for Lesson 17: Overview

Practices Overview

The goal of this practice is to observe the symptoms of, diagnose, and apply solutions for buffer cache problems.

Practice 17-1: Using the DB Cache Advisor

In this practice, assume that you have an OLTP workload that needs some additional tuning and that the SQL statements are not available to tune, or have already been tuned using the SQL Tuning Advisor and SQL Access Advisor. You use the diagnostics in the AWR report and the DB Cache Advisor to choose an optimal size for the database buffer cache. The workload generator for this scenario creates an AWR snapshot, and then runs the workload.

1. Complete the setup for this scenario.
 - a. Log out of the `orcl` database in Enterprise Manager Cloud Control.
 - b. Run `./prepare BC`.

```
$ cd $HOME/workshops
$ ./prepare BC
The Oracle base for
ORACLE_HOME=/u01/app/oracle/product/12.1.0/dbhome_1 is
/u01/app/oracle

Database closed.
Database dismounted.
ORACLE instance shut down.

ORACLE instance started.

Total System Global Area 384200704 bytes
Fixed Size                  2288536 bytes
Variable Size                364905576 bytes
Database Buffers             8388608 bytes
Redo Buffers                 8617984 bytes
Database mounted.
Database opened.
Finished setup Buffer Cache
$
```

2. Verify that `DB_CACHE_SIZE` is set to `8M`.

```
$ sqlplus / as sysdba

SQL> show parameter DB_CACHE_SIZE

NAME                           TYPE        VALUE
-----                         -----      -----
db_cache_size                 big integer    8M

SQL> exit
$
```

3. Watch the Average Active Sessions graph on the Performance home page findings as the workload progresses.
 - a. Log in to Enterprise Manager Cloud Control with `sysman` as the username and `oracle_4U` as the password.
 - b. Expand the **Targets** menu and select **Databases**.
 - c. Click the link for the `orcl` database.

- d. Expand the **Performance** menu and select **Performance Home**.
- e. Log in to the `orcl` database with `sys` as the username, `oracle_4U` as the password, and `SYSDBA` as the role.
- 4. In a terminal window with the environment set for the `orcl` database, and in the `/home/oracle/workshops` directory, start the workload generator with `./workgen BC`. The `workgen` script creates an AWR snapshot and a Statspack snapshot before starting the workload in the background.

```
$ ./workgen BC

The Oracle base for
ORACLE_HOME=/u01/app/oracle/product/12.1.0/dbhome_1 is
/u01/app/oracle

PL/SQL procedure successfully completed.

PL/SQL procedure successfully completed.

$
```

- 5. Wait for about five minutes, and then in a terminal window stop the workload with the `rm unload` command.

```
$ rm unload
```

- 6. Review the ADDM Performance Analysis findings.
 - a. On the Performance page, click **Run ADDM Now** to generate an ADDM report. Click **Yes** to confirm.



- b. Review the ADDM Performance Analysis section and check the performance findings. Examine the top finding that you, as a DBA, can address. When you select the findings

with the greatest impact, you may find several that you may do nothing about, such as “Top SQL Statements” shown in the following example. Assume that you have already tuned these statements as covered in the previous lessons and move down the list of findings. Other findings that you may see are: “CPU Usage,” which usually recommends SQL tuning and “Free Buffer Waits,” which requires more analysis, sometimes leading to a change in the Number of DB writer processes. The finding of interest in the example list is “Undersized Buffer Cache.”



- c. Record the Average Active Sessions Value. This number is the average number of sessions working or waiting. _____

- d. Drill down on the “Undersized Buffer Cache” finding. Notice the recommendation to increase the buffer cache size to 12 MB. This value is limited by the advisor, which takes estimates only from 50% to 200% of the current setting.

Performance Finding Details: Undersized Buffer Cache

Finding	The buffer cache was undersized causing significant additional read I/O.
Impact (Active Sessions)	.44
Percentage of Finding's Impact (%)	 14.3
Period Start Time	Oct 3, 2013 12:07:31 PM
End Time	Oct 3, 2013 12:14:48 PM
Filtered	No Filters

Recommendations

[Show All Details](#) | [Hide All Details](#)

Details	Category	Benefit (%) ▾
▽ Hide	DB Configuration	 
Action	Increase the buffer cache size by setting the value of parameter "db_cache_size" to 12 M.	Implement

Additional Information

The value of parameter "db_cache_size" was "8 M" during the analysis period.

Findings Path

[Expand All](#) | [Collapse All](#)

Findings	Percentage of Finding's Impact (%)
▽ The buffer cache was undersized causing significant additional read I/O.	  14.3
▽ Waits for free buffers were consuming significant database time.	  9.3
Wait class "Configuration" was consuming significant database time.	  9.3
Wait class "User I/O" was consuming significant database time.	  16

7. Create an AWR report. Save the report as `awr_BC_1.html`.
- Return to the ADDM page by clicking Automatic Diagnostic Monitor (ADDM) in the breadcrumb at the top of the page.
 - On the ADDM report page, click **View Snapshots**.
 - On the Snapshot Details page, click the **Report** tab.
 - Record the Period Start and End times (hr:min:sec):
Start _____ End _____
 - Set the download preferences in the browser. (Note that these instructions are for Mozilla Firefox.) In the Browser menu, click **Edit > Preferences**. In the Downloads section of the Preferences dialog box, select “Always ask me where to save files.” Click **Close**.
 - At the top right of the Workload Repository report, click “**Save to File**.” Select **Save File** in the dialog box and click **OK**. When prompted, specify `awr_BC_1.html` as the name of the report. Notice which directory the report is being created in so that you can find it later; default is `/home/oracle/Downloads`. Click **Save**. Close the Downloads dialog box.

8. Review the AWR report and identify the primary performance issue. The **Top 10 Foreground Events by Total Wait Time** section shows one or more of the events that point to the buffer cache size: db file sequential read, db file scattered read, buffer busy waits, free buffer waits, latch: cache buffer Iru chain, or latch: cache buffers chains.

Top 10 Foreground Events by Total Wait Time					
Event	Waits	Total Wait Time (sec)	Wait Avg(ms)	% DB time	Wait Class
DB CPU		695.5		51.3	
write complete waits	226	110.7	490	8.2	Configuration
db file parallel read	907,484	104.9	0	7.7	User I/O
db file sequential read	8,126,555	75.1	0	5.5	User I/O
latch: cache buffers chains	21,609	17.9	1	1.3	Concurrency
read by other session	51,298	15.1	0	1.1	User I/O
free buffer waits	1,047	14.9	14	1.1	Configuration
log file sync	602	14.4	24	1.1	Commit
db file scattered read	758,101	10.9	0	.8	User I/O
direct path read	786,970	9.8	0	.7	User I/O

9. From the report, determine the possible impact of the buffer cache problem. If all waits associated with the physical reads to the buffer cache are eliminated, what percent increase in performance could you expect? _____ (5 to 25 % is expected. The example shows ~14% = db file sequential reads + db file parallel reads + latch: cache buffers chains.) Of course, removing all the waits for physical reads is not possible. Reducing the physical reads will often reduce the DB CPU.

Note: Your results will vary.

- What is **Time Model** showing as the greatest loads? (sql execute elapsed time)
 - What is the buffer cache size in the report in the **Cache Sizes section of Report Summary?** (8M)
 - What does the **Load Profile** section show for **physical reads per second** and **transaction**? These numbers will have meaning when you compare them to the next test.
Physical Reads(blocks) _____ per second, _____ per transaction
 - What is the **buffer hit %** (buffer cache hit ratio) in the **Instance Efficiency** section? (For OLTP type activity, less than 90% is considered a low buffer cache hit ratio.)
 - To confirm the symptoms, check the **top misses** in the **Latch Sleep Breakdown** section. **Main Menu > Latch Statistics > Latch Sleep Breakdown.** Look for misses and sleeps for latches related to the buffer cache. The number of misses and sleeps may be small enough in comparison to the get requests that they would not be considered at all, except that because they are the top in the latch sleep breakdown, they are additional evidence pointing to an undersized buffer cache. What latches are listed at or near the top related to the buffer cache? (You may see “cache buffers Iru chain” or “cache buffers chains.”)
- Use the **Buffer Pool Advisory** in the AWR report.
 - The navigation path is **Back to Top > Main Report > Advisory Statistics > Buffer Pool Advisory.**
 - What is an appropriate **size for the database buffer cache?** (16 MB or more)
 - In the advisory, what is the value shown for the **estimated number of physical reads** for the recommendation of 16 MB? (The change in Est Physical Read Factor should be in the range 0.5 through 0.03 depending on your machine.)

Note: Because of the limitations of the advisor, the maximum estimated cache is only 200% of the current size. A larger buffer pool may provide greater benefit.

- d. Log out of the `orcl` database in Enterprise Manager Cloud Control. **Logout > Logout of orcl (select “Login page after logout”)**, and click **Logout**.
11. Change the buffer cache size. To reduce the number of trials, increase `DB_CACHE_SIZE` to 28 MB.

In a terminal window, connect to the `orcl` instance, and issue the following commands in SQL*Plus:

```
$ sqlplus / as sysdba
SQL> ALTER SYSTEM SET DB_CACHE_SIZE=28M SCOPE=SPFILE;
SQL> SHUTDOWN IMMEDIATE
SQL> STARTUP
SQL> EXIT
```

From the AWR report, you can see that the optimal size is at least 16 MB. By changing `DB_CACHE_SIZE` to 28 MB, the next trial should allow the advisor to show a better value, and reduce the number of iterations of this test.

12. Start the workload generator again: `./workgen BC`.

```
$ ./workgen BC
```

13. After about five minutes, stop the workload with the `rm runload` command.

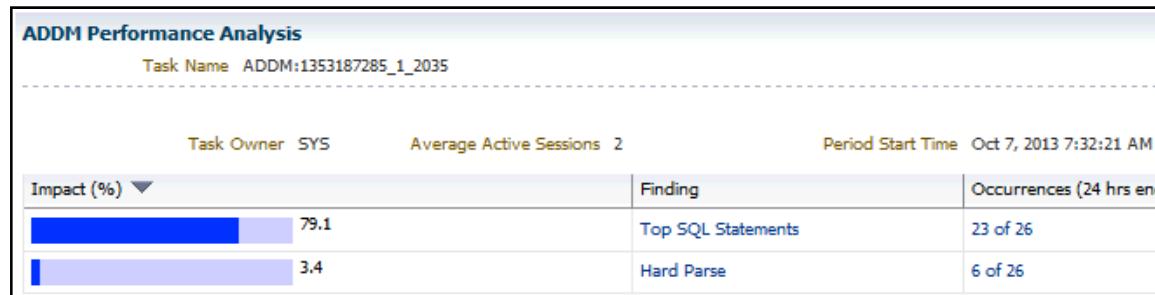
```
$ rm runload
```

14. Create an ADDM report and review the Performance Analysis. (This is the same procedure as shown in step 6.) Check the performance findings. Did the primary issues change?

Note: If creating an ADDM report produces an error, log out of EMCC. Log in and navigate to Performance > Advisors Home. Then click the name of the most recent ADDM report.

Yes, the issues have changed. The major change shown in the example report is that Undersized Buffer Cache finding is gone or much reduced.

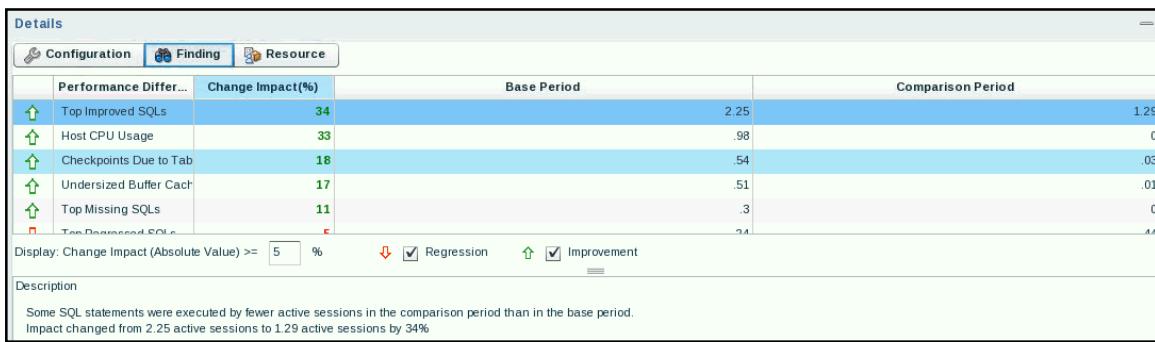
Note: The report you get could be different from the example report depending on the classroom machine. The Average Active Sessions number will be reduced when performance is improved. The example report shows a reduction in the impact of the Top SQL Statements finding and the elimination of the Undersized Buffer Cache finding. The result of increasing the buffer cache is that SQL statements that were delayed by I/O and buffer waits are now waiting on the CPU. This shift of finding from one area to another is common.



15. Create an AWR report and save it to a file.
 - a. On the ADDM page, click **View Snapshots**.
 - b. On the Snapshots page, click the **Report** tab to create an AWR report.

- c. Record the Period Start and End times (hr:min:sec):
Start _____ End _____
 - d. Save this report as **awr_BC_2.html**.
16. In another window or browser tab, display the **awr_BC_1.html** file, so that you can compare the two reports.
17. Review the AWR report and identify the primary performance changes.
- a. In **Top 10 Foreground Events by Total Wait Time**, the order of events and the amount of time waiting changed. For “db file sequential reads,” the % DB time spent waiting dropped, now that more blocks are being held in the cache. Waits in terms of % DB time for the other buffer cache-related items have dropped. Overall, more time is going to useful work (SQL execution) and less time to physical reads. In the following example, almost all the wait time is part of the DB CPU.
- Note:** When the elapsed time of the test period is not the same, compare only the metrics that are normalized. Per second, per transaction, and % DB time metrics are normalized metrics.
- | Event | Waits | Total Wait Time (sec) | Wait Avg(ms) | % DB time | Wait Class |
|----------------------------------|---------|-----------------------|--------------|-----------|---------------|
| DB CPU | | 600.1 | | 65.5 | |
| log file sync | 574 | 11.2 | 20 | 1.2 | Commit |
| direct path read | 617,476 | 9.8 | 0 | 1.1 | User I/O |
| write complete waits | 19 | 6.5 | 341 | .7 | Configuration |
| enq: KO - fast object checkpoint | 12 | 4.4 | 363 | .5 | Application |
| db file sequential read | 19,212 | 1.4 | 0 | .2 | User I/O |
| cursor: pin S | 253 | .4 | 2 | .0 | Concurrency |
| log file switch completion | 2 | .1 | 71 | .0 | Configuration |
| Disk file operations I/O | 719 | .1 | 0 | .0 | User I/O |
| latch: cache buffers chains | 88 | .1 | 1 | .0 | Concurrency |
- b. What is the number of **physical reads per second** shown in the **Load Profile** section? Compare this number with the number of physical reads per second in the AWR_BC_1 report. (The number of physical reads per second should have dropped significantly because the buffer cache size is increased, and more blocks are being held in the buffer cache.)
 - c. What is the **buffer hit ratio** in the **Instance Efficiencies** section? (OLTP applications expect a buffer hit ratio of 90% or higher.) You should see an improvement from the earlier report.
 - d. The **Buffer Cache Advisory** indicates an optimal buffer cache size. What size is indicated for your instance and workload? (A value of approximately 28–44 MB is expected.) An optimal size is the point where additional memory reduces the Estimated Physical Reads by a small amount. The optimal value for your instance may vary from the expected value.
18. Create an ADDM Compare Report between the last two ADDM runs.
- a. Click **Performance > AWR > Compare Period ADDM**.
 - b. Set the comparison period to the start and end times recorded for the second AWR report in step 16c.
 - c. Set the base period in the Customized report section to the start and end times recorded for the first AWR report in step 6d.
19. Review the Compare Period ADDM report

- At the bottom of the Overview section, note the difference in the Average Active Sessions. A reduction indicates that the same workload had fewer sessions, the workload was the same so each session ran more quickly with less waiting (an improvement).
- Scroll down to the Details section and click the Finding tab. Most findings will show improvement. There may be some regressed findings such as Top Regressed SQL. Overall there was improvement. The example shows a 5% improvement.



- Explore the other tabs in the Details section. Notice the reduction in I/O and CPU usage.

Note: Every block read into the buffer cache requires accessing the buffer chains latch to find and load the data block into the buffer cache, so reducing the number of reads also reduces the CPU usage.

Practice 17-2: Using the Keep Pool

In this practice, you are constrained on memory resources. Adjust the buffer pool memory areas to use as little memory as possible with the most benefit. The `prepare KC` script sets `DB_CACHE_SIZE` to 28 MB. On the second trial, reduce `DB_CACHE_SIZE` to 20 MB and set `KEEP_CACHE_SIZE` to 8 MB. A script that assigns a set of frequently used small tables to the keep pool is provided. Will this perform as well or better than having only `DB_CACHE_SIZE` set to 28 MB?

1. Prepare the database for this scenario.

- a. Log out of the `orcl` database in Enterprise Manager Cloud Control (EMCC).
- b. Run the `./prepare KC` script.

```
$ cd $HOME/workshops
$ ./prepare KC

The Oracle base for
ORACLE_HOME=/u01/app/oracle/product/12.1.0/dbhome_1 is
/u01/app/oracle

Database closed.
Database dismounted.
ORACLE instance shut down.
ORACLE instance started.

Total System Global Area  405078016 bytes
Fixed Size                  2288776 bytes
Variable Size                364905336 bytes
Database Buffers              29360128 bytes
Redo Buffers                  8523776 bytes
Database mounted.
Database opened.

$
```

2. While the workload is running, navigate to the Performance Home page in EMCC.
3. Test the base configuration with a workload.

- a. Execute the `./workgen KC` script.

```
$ ./workgen KC
The Oracle base for
ORACLE_HOME=/u01/app/oracle/product/12.1.0/dbhome_1 is
/u01/app/oracle

PL/SQL procedure successfully completed.

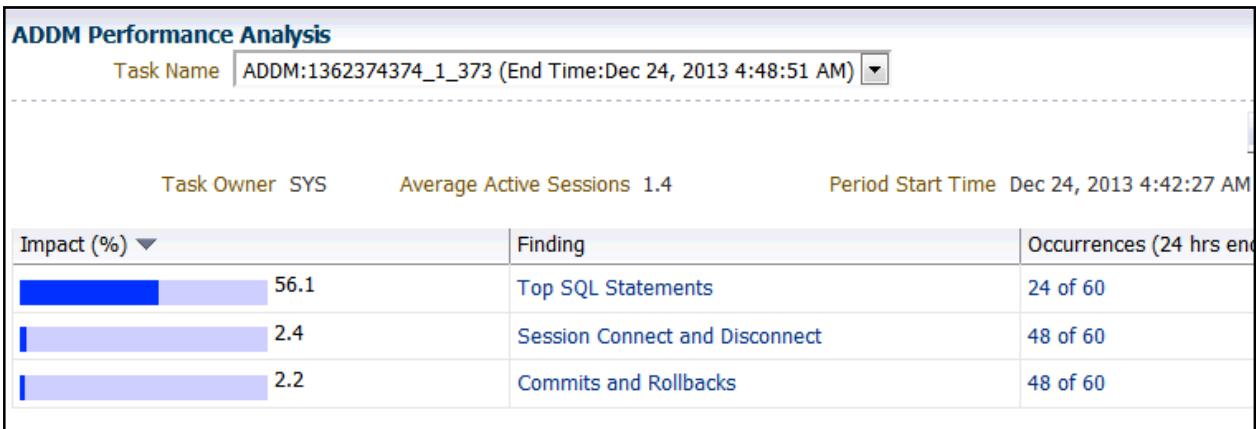
PL/SQL procedure successfully completed.

$
```

- Wait for about five minutes, and then end the workload by deleting the `runload` file. While you are waiting, examine the graphs on the Performance Home page. Notice the peak values and the I/O per second values.

```
$ rm unload
$
```

- Create an ADDM report to review the findings.
 - Click **Run ADDM Now**. Click **Yes** to confirm.
 - What are the findings?



Note: The workload for KC is not identical to the BC workload. The KC workload has an increased OLTP component

- Create an AWR report from the snapshots used for this ADDM report.
 - Click View Snapshots, and then the Report tab.
 - Record the Start and End times (hr:min:sec):
Start _____ End _____
 - Save this report as **AWR_KC_1.html**.

7. The AWR report will show the top SQL by Reads. The database tables referenced in the SQL statements listed in this section can yield a list of tables that could be assigned to the keep pool. Are there tables that could be cached in a keep pool?
- The **Top 10 Foreground Events by Total Wait Time** shows one or more of the events that point to the buffer cache size: db file sequential read, db file scattered read, free buffer waits, latch: cache buffer chains, or latch: cache buffer lru chain.

Top 10 Foreground Events by Total Wait Time					
Event	Waits	Total Wait Time (sec)	Wait Avg(ms)	% DB time	Wait Class
DB CPU		467.2		86.4	
log file sync	684	12	17	2.2	Commit
write complete waits	8	2.6	319	.5	Configuration
direct path read	132,826	2	0	.4	User I/O
free buffer waits	88	1.7	19	.3	Configuration
db file sequential read	18,449	.7	0	.1	User I/O
ADR block file read	20	.2	9	.0	Other
direct path sync	1	.1	84	.0	User I/O
Disk file operations I/O	656	.1	0	.0	User I/O
direct path write	1	.1	52	.0	User I/O

- In the load profile, check the number of physical reads.

Load Profile					
	Per Second	Per Transaction	Per Exec	Per Call	
DB Time(s):	1.4	0.1	0.00	0.04	
DB CPU(s):	1.2	0.1	0.00	0.04	
Redo size (bytes):	40,739.7	2,790.2			
Logical read (blocks):	208,616.0	14,287.6			
Block changes:	252.1	17.3			
Physical read (blocks):	1,452.6	99.5			
Physical write (blocks):	7.4	0.5			
Read IO requests:	412.4	28.2			
Write IO requests:	5.3	0.4			
Read IO (MB):	11.4	0.8			
Write IO (MB):	0.1	0.0			
User calls:	32.0	2.2			
Parses (SQL):	51.7	3.5			
Hard parses (SQL):	2.3	0.2			
SQL Work Area (MB):	1.6	0.1			
Logons:	1.5	0.1			
Executes (SQL):	13,632.0	933.6			
Rollbacks:	0.0	0.0			
Transactions:	14.6				

- How many **full-table scans** are being performed? Check the **Other instance activity statistics** in the **Main Report** section. Scroll to the Main Report section. Click **Instance Activity Statistics > Other Instance Activity Stats**.

Note: Table scans (direct read) do not normally go into the buffer cache; these reads go directly into the session PGA, but your workload has forced the full table scans to

be read into the cache. You should find that the number of table scans (long) and table scans (direct read) are almost the same.

Statistic	Per second	Per transaction
table scans (direct read)		
table scans (long tables)		
table scans (short tables)		

- d. Find the SQL statements that have the highest number of **physical reads**.

Hint: Back to **Top > SQL Statistics > SQL Ordered by Reads**.

SQL_ID _____

SQL_ID _____

SQL Ordered by Reads

Physical Reads	Executions	Reads per Exec	%Total	Elapsed Time (s)	%CPU	%IO	SQL Id
469,684	330	1,423.28	84.13	15.47	67.55	11.33	a48pc5hd541nq
469,507	330	1,422.75	84.10	9.48	68.80	18.47	djfiaszxrt89wk
49,602	6	8,267.00	8.88	1.53	60.18	14.21	3mm4k3svx4nn3
15,672	1	15,672.00	2.81	0.63	95.86	7.18	6phrcjjvtbhjj
6,708	0		1.20	1.48	42.64	3.06	029dn07qqaw3q
4,607	75	61.43	0.83	222.57	87.76	0.02	3gmqqdvdn93cc
3,314	749,925	0.00	0.59	151.40	87.40	0.02	431mwkyt65jbb
1,448	26	55.69	0.26	5.46	50.63	15.33	fhf8upax5cxsz
1,410	1	1,410.00	0.25	0.05	97.16	9.04	32qq8k1n8ynn9
880	5,400	0.16	0.16	5.25	84.00	0.25	6g215w0qfstc2

- e. Check some of the SQL IDs for the full text of the statements with the highest number of physical reads to determine the segments involved. Click the **SQL_ID** to go to the full listing. Find the table names that are being accessed. What tables are most accessed? (SALES, PRODUCTS, COSTS, EMPLOYEES, DEPARTMENTS, LOCATIONS, COUNTRIES, and USER_TABLES)

The SQL IDs shown in this example correspond to the SQL IDs in the previous step. There may be differences in which SQL IDs show the highest number of reads from the actual practice and the examples shown. Additional SQL_IDs are included below, to allow for system-generated SQL_IDs that may appear for some tests.

The order of the SQL_IDs will vary.

SQL ID	SQL Text
1fg23b7ddarws	<pre> DECLARE /* is is expected that the cursors will use a direct read */ /* and not use the cache*/ CURSOR C2 IS SELECT /*+ CACHE (CUSTOMERS) */ CUST_ID, CUST_EMAIL, COUNTRY_ID FROM CUSTOMERS; CURSOR C1 IS SELECT /*+ CACHE (PRODUCTS) */ PROD_ID, PROD_NAME FROM PRODUCTS; CURSOR C3 IS SELECT /*+ CACHE (COUNTRIES) */ COUNTRY_ID FROM COUNTRIES; TYPE emp_list IS TABLE OF VARCHAR2(30); parameters emp_list; j NUMBER(10); tot_amount NUMBER(20); BEGIN FOR C1_ROW in C1 LOOP SELECT SUM(unit_cost) total INTO TOT_AMOUNT FROM </pre>

	<pre> sh.costs WHERE prod_id = C1_ROW.prod_id; END LOOP; FOR C2_ROW in C2 LOOP SELECT SUM(AMOUNT_SOLD) total INTO TOT_AMOUNT FROM sh.sales WHERE cust_id = C2_ROW.cust_id; END LOOP; FOR C1_ROW in C1 LOOP SELECT SUM(AMOUNT_SOLD) total INTO TOT_AMOUNT FROM sh.sales WHERE prod_id = C1_ROW.prod_id; END LOOP; FOR C3_ROW in C3 LOOP SELECT count(*) into j FROM CUSTOMERS C WHERE C.COUNTRY_ID = C3_ROW.country_id; END LOOP; END; </pre>
32qq8k1n8ynn9	Select BYTES, extents from dba_segments where OWNER = :1 and PARTITION_NAME = :2
3gmqqdvdn93cc	<pre> DECLARE CURSOR C2 IS SELECT CUST_ID FROM CUSTOMERS WHERE rownum < 10000; CURSOR C1 IS SELECT PROD_ID FROM PRODUCTS; TYPE emp_list IS TABLE OF VARCHAR2(30); parameters emp_list; j NUMBER(10); tot_amount NUMBER(20); BEGIN FOR C1_ROW in C1 LOOP SELECT SUM(unit_cost) total INTO TOT_AMOUNT FROM sh.costs WHERE prod_id = C1_ROW.prod_id; END LOOP; FOR C2_ROW in C2 LOOP SELECT SUM(AMOUNT_SOLD) total INTO TOT_AMOUNT FROM sh.sales WHERE cust_id = C2_ROW.cust_id; END LOOP; FOR C1_ROW in C1 LOOP SELECT SUM(AMOUNT_SOLD) total INTO TOT_AMOUNT FROM sh.sales WHERE prod_id = C1_ROW.prod_id; END LOOP; END; </pre>
3mm4k3svx4nn3	<pre> select :"SYS_B_00" stat_type, round((sysdate - startup_time)*:"SYS_B_01") v1, null v2, version v3, null v4 from v\$instance union all select :"SYS_B_02", sum(value)/:"SYS_B_03"/:"SYS_B_04" v1, null v2, null v3, null v4 from v\$sga union all select :"SYS_B_05", sum(nvl(f.total_gb, :"SYS_B_06") - nvl(s.used_gb, :"SYS_B_07")) v1, null v2, null v3, null v4 from dba_tablespaces t, (select tablespace_name, sum(nvl(bytes, :"SYS_B_08"))/(:"SYS_B_09"*:"SYS_B_10"*:"SYS_B_11") total_gb from dba_data_files group by tablespace_name) f, (select tablespace_name, sum(nvl(bytes, :"SYS_B_12"))/(:"SYS_B_13"*:"SYS_B_14"*:"SYS_B_15") used_gb from dba_segments group by tablespace_name) s where t.tablespace_name = f.tablespace_name (+) and t.tablespace_name = s.tablespace_name (+) and t.contents != : "SYS_B_16" and not (t.extent_management = :"SYS_B_17" and t.contents = :"SYS_B_18") union all select :"SYS_B_19", sum(case when metric_name = :"SYS_B_20" then value else :"SYS_B_21" end) v1, sum(case when metric_name </pre>

	= :"SYS_B_22" then value else :"SYS_B_23" end) v2, null v3, null v4 from v\$sysmetric where group_id = :"SYS_B_24" and metric_name in (:"SYS_B_25", :"SYS_B_26") union all SELECT :"SYS_B_27", count(*) v1, f.task_id v2, null v3, null v4 FROM dba_advisor_findings f WHERE f.task_id = (WITH snaps as (SELECT /*+ NO_MERGE */ max(s.snap_id) as end_snap, max(v.dbid) as dbid FROM DBA_HIST_SNAPSHOT s, V\$DATABASE v WHERE s.dbid = v.dbid) SELECT max(t.task_id) as task_id FROM dba_adm_tasks t, snaps s, dba_adm_instances i WHERE t.dbid = s.dbid AND t.begin_snap_id = s.end_snap - :"SYS_B_28" AND t.end_snap_id = s.end_snap AND t.how_created = :"SYS_B_29" AND t.requested_analysis = :"SYS_B_30" AND t.task_id = i.task_id AND i.instance_number = SYS_CONTEXT(:"SYS_B_31", :"SYS_B_32")) AND f.type not in (:"SYS_B_33", :"SYS_B_34") AND f.parent = :"SYS_B_35" AND (f.filtered IS NULL OR f.filtered <> :"SYS_B_36") group by f .task_id
4275pc3yjd1tw	SELECT COUNT(*) FROM CUSTOMERS C WHERE C.COUNTRY_ID = :B1
431mwkyt65jbb	SELECT SUM(AMOUNT_SOLD) TOTAL FROM SH.SALES WHERE CUST_ID = :B1
4qk3ay7bq4pab	BEGIN statspack.snap; END;
a48pc5hd541nq	DECLARE CURSOR C1 IS SELECT /*+ CACHE (hr.employees) */ * FROM hr.employees; CURSOR C2 IS SELECT /*+ CACHE (hr.departments) */ * FROM hr.Departments; CURSOR C3 IS SELECT /*+ CACHE (hr.locations) */ * FROM hr.Locations; CURSOR C4 IS SELECT /*+ CACHE (hr.regions) */ * FROM hr.regions; CURSOR C5 IS SELECT /*+ CACHE (user_tables) */ * FROM user_tables; total NUMBER(10); v_dept Varchar2(30); v_region VARCHAR2(25); v_city VARCHAR2(30); v_country VARCHAR2(40); BEGIN FOR c5row in C5 LOOP SELECT count(index_name) into total from user_indexes WHERE table_name = c5row.table_name; END LOOP; FOR clrow in C1 LOOP BEGIN SELECT D.Department_name, R.region_name, L.City, C.country_name INTO v_dept, v_region, v_city, v_country FROM departments D, locations L, regions R, countries C WHERE D.location_id = L.location_id AND C.region_id = R.region_id AND L.country_id = C.country_id AND D.department_id = clrow.department_id; EXCEPTION WHEN NO_DATA_FOUND THEN NULL; END; END LOOP; FOR clrow in C1 LOOP BEGIN SELECT

	count (*) into total FROM JOB_HISTORY WHERE employee_id = c1row.employee_id; EXCEPTION WHEN NO_DATA_FOUND THEN NULL; END; END LOOP; FOR c3row IN C3 LOOP BEGIN Select count(*) into total FROM employees E, Departments D WHERE c3row.location_id = D.location_id AND D.department_id = E.department_id; EXCEPTION WHEN NO_DATA_FOUND THEN NULL; END; END LOOP; END;
djfazsxrt89wk	SELECT /*+ CACHE (user_tables) */ * FROM USER_TABLES

- f. Check the Segments by Physical Reads in the Segment Statistics section. What segments have the highest physical reads?

Partitions of the SH.SALES table are expected to appear. This is a very large table. The small tables that should be placed in the keep pool do not appear. Knowledge of the application is needed to properly choose the tables and indexes to keep.

8. The developers have provided a script named `keep_pool.sh` that will create a small keep pool of 8 MB, and reduces the `DB_CACHE_SIZE` by the same amount. It assigns some small but very active tables and indexes to the keep pool, and loads these tables into the pool. The `keep_pool.sh` script also assigns the most active partitions of the `SALES` table to the keep pool. Review this script, and then execute it. The full table scan caused by the `SELECT * FROM <table_name>` statement will cause the table data to be loaded into the keep pool before the test run. The `ORDER BY` clause is used to force the use of an index so that the index is loaded into the keep cache. These accesses mean that the physical reads due to the first access is not included in the test statistics.
- Log out of the `orcl` database in EMCC.
 - View the `keep_pool.sh` script, and then execute it.

```
$ cat keep_pool.sh
#!/bin/bash
--- create a KEEP pool
--- and set the table properties to make use of it
--- Select statements populate the keep cache

sqlplus -S /nolog >/dev/null <<-EOF
set termout off
set feedback off
connect / as sysdba
ALTER SYSTEM SET db_cache_size = 20M SCOPE=SPFILE;
ALTER SYSTEM SET db_keep_cache_size = 8M SCOPE=SPFILE;
SHUTDOWN IMMEDIATE
STARTUP
exit
EOF

sqlplus -S /nolog >/dev/null <<-EOF
set termout off
set feedback off
```

```

connect hr/hr

ALTER TABLE EMPLOYEES STORAGE(BUFFER_POOL KEEP);
ALTER TABLE COUNTRIES STORAGE(BUFFER_POOL KEEP);
ALTER TABLE DEPARTMENTS STORAGE(BUFFER_POOL KEEP);
ALTER TABLE LOCATIONS STORAGE(BUFFER_POOL KEEP);
ALTER TABLE REGIONS STORAGE(BUFFER_POOL KEEP);
ALTER TABLE JOB_HISTORY STORAGE(BUFFER_POOL KEEP);

select * from employees;
select * from departments;
select * from countries;
select * from locations;
select * from regions;
select * from job_history;
exit
EOF

sqlplus -S /nolog > /dev/null <<-EOF
set termout off
set feedback off
connect oe/oe

ALTER TABLE orders STORAGE(BUFFER_POOL KEEP);
ALTER TABLE order_items STORAGE(BUFFER_POOL KEEP);
ALTER TABLE inventories STORAGE(BUFFER_POOL KEEP);
ALTER TABLE customers STORAGE(BUFFER_POOL KEEP);
ALTER TABLE warehouses STORAGE(BUFFER_POOL KEEP);
ALTER TABLE product_information STORAGE(BUFFER_POOL KEEP);
ALTER INDEX INV_PRODUCT_IX STORAGE(BUFFER_POOL KEEP);
ALTER INDEX PRODUCT_INFORMATION_PK STORAGE(BUFFER_POOL KEEP);

select * from orders;
select * from order_items;
select * from inventories;
select * from customers;
select * from warehouses;
select * from product_information;
exit
EOF

sqlplus -S /nolog > /dev/null <<-EOF
set termout off
set feedback off
connect sh/sh
ALTER TABLE SALES MODIFY PARTITION SALES_Q3_1999 STORAGE(BUFFER_POOL
KEEP);
ALTER TABLE SALES MODIFY PARTITION SALES_Q1_1999 STORAGE(BUFFER_POOL
KEEP);
select * from sales;
EOF

```

```
$ ./keep_pool.sh  
$
```

9. Navigate to the Performance Home page of the `orcl` database.
10. Start the same workload as in Step 2.
 - a. Execute the `./workgen KC` script.
 - b. After about five minutes, stop the workload with the `rm runload` command.
 - c. In EMCC, run an ADDM report.
 - d. View the findings.



11. Create an AWR report.
 - a. Click **View Snapshots**.
 - b. When the Snapshot Details page appears, click the **Reports** tab.
 - c. Record the Start and End times (hr:min:sec):
Start _____ End _____
 - d. Save this report as `awr_KC_2.html`.

12. Examine the AWR report and compare it with the previous report, awr_KC_1.html.

Note: If the elapsed time for awr_KC_1.html is very different from the elapsed time for awr_KC_2.html, the values of % DB Time and Physical Reads may not show the expected values because of the short durations of the workloads.

Answer the following questions:

- a. What is the difference in **Top 10 Foreground Events by Total Wait Time?** (Make the comparison based on % DB Time.)

The changes in % DB Time may be very small, but it may be enough to change the order of the top waits.

Note: If other waits are reduced, more DB Time will accumulate in DB CPU.

Top 10 Foreground Events by Total Wait Time					
Event	Waits	Total Wait Time (sec)	Wait Avg(ms)	% DB time	Wait Class
DB CPU		392.5		88.0	
log file sync	556	9.8	18	2.2	Commit
free buffer waits	257	2.6	10	.6	Configuration
direct path read	106,169	1.7	0	.4	User I/O
db file sequential read	50,749	1.1	0	.2	User I/O
write complete waits	4	.3	83	.1	Configuration
enq: KO - fast object checkpoint	1	.3	253	.1	Application
db file parallel read	6,496	.2	0	.0	User I/O
Disk file operations I/O	761	.1	0	.0	User I/O
direct path sync	1	.1	67	.0	User I/O

- b. In the **Load Profile** section, is there a difference in **physical reads per second** and **physical reads per transactions** for each of the two periods?

Load Profile				
	Per Second	Per Transaction	Per Exec	Per Call
DB Time(s):	1.4	0.1	0.00	0.04
DB CPU(s):	1.2	0.1	0.00	0.04
Redo size (bytes):	37,015.3	3,369.2		
Logical read (blocks):	202,388.6	18,422.0		
Block changes:	217.6	19.8		
Physical read (blocks):	1,579.4	143.8		
Physical write (blocks):	11.6	1.1		
Read IO requests:	586.9	53.4		
Write IO requests:	8.0	0.7		
Read IO (MB):	12.3	1.1		
Write IO (MB):	0.1	0.0		
User calls:	30.9	2.8		
Parses (SQL):	46.3	4.2		
Hard parses (SQL):	1.7	0.2		
SQL Work Area (MB):	1.6	0.1		
Logons:	1.5	0.1		
Executes (SQL):	13,670.1	1,244.3		
Rollbacks:	0.0	0.0		
Transactions:	11.0			

	Statistic	Per second	Per transaction
AWR Step 7b	Physical reads		
AWR Step 12b	Physical reads		

The number of physical reads in the second report may be higher than the physical reads reported in the first. This illustrates the efficiency of the buffer cache algorithms. If you recall, the physical reads for the first test in practice 17-1 were very high; this also illustrates that the keep pool can be an effective way of reducing physical reads when the application access patterns are known.

- c. In the **Load Profile**, compare **DB Time per second** for each of the two reports.

	Statistic	Per second
AWR Step 7b	DB Time	
AWR Step 12b	DB Time	

Note: The DB Time difference is the clearest indicator, in this practice, of a difference in performance. Lower DB Time per second shows better performance.

13. Is there a difference in **Buffer Hit %**? The overall Buffer Hit % may change very little. The buffer cache hit ratio is a combination of the pools.
14. Check the **Buffer Pool Statistics**. Is the keep pool being used?

Yes, the Buffer Pool Statistics section shows 100% hit ratio on the keep pool. The frequently used blocks are moved to the keep pool and a large number of hits are removed from the default pool calculation. You would expect the hit ratio of the default pool to decline. The statistics over such a small data set and short duration may not show any difference. A small number of physical reads may appear in the Keep Pool statistics; this indicates that the Pool has more blocks assigned to it than there is space available, and the physical reads required to load the “kept” objects into the KEEP POOL.

How many **full table scans** are being performed? Check the instance activity statistics for both the reports: Step 7 and current AWR. (Table scans are usually unrelated to the physical reads when tables are in the keep buffer cache, but placing indexes in the keep pool can influence the optimizer to use an index rather than do a full-table scan.)

Note: Nearly all of the full table scans are being forced to use the DB buffer cache. This means that these reads are not going to PGA (server process memory) as is usual for full table scans.

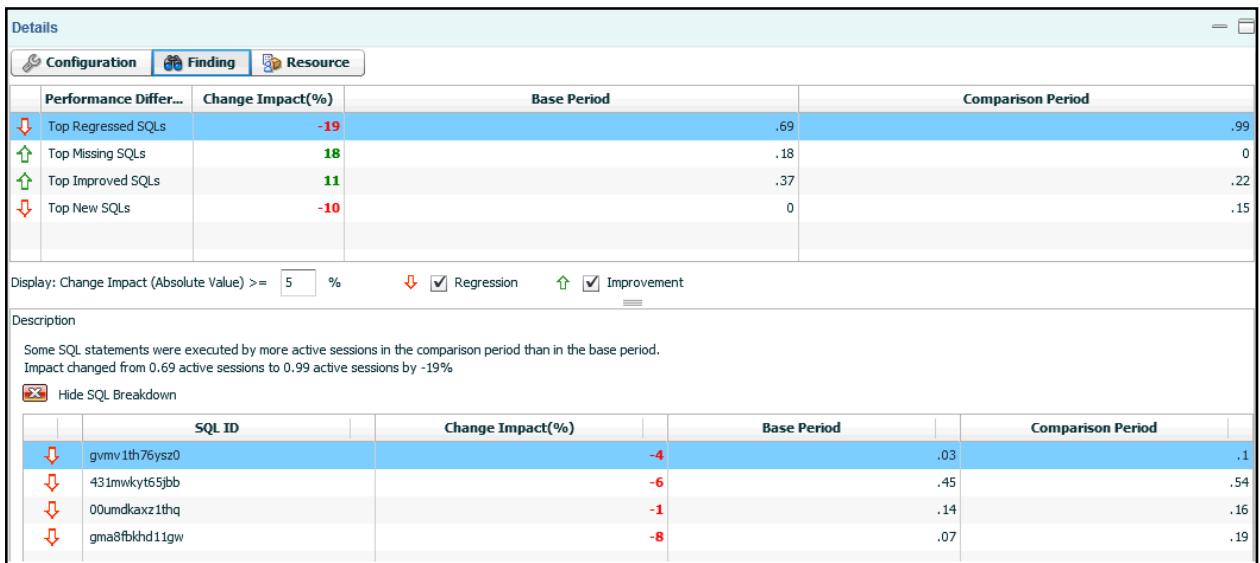
Statistic	Per second (step 7c)	Per second (current)
table scans (long)		
table scans (short)		

15. Considering that this instance is memory constrained, what is the smallest DB_CACHE_SIZE that would significantly affect performance based on the Buffer Pool Advisor?

The advisor shows that there could be a slight added benefit by increasing the buffer cache to between 28 and 36 MB. Using the keep pool in this example may or may not show a small performance improvement. *This practice did show the process to identify and cache tables in the keep cache.*

16. Run a Compare Period ADDM report using the start and end times of the previous two AWR reports recorded in steps 6 and 11.

17. Examine the Compare Period ADDM report.
- Did the Average Active Session change? The expected answer is yes. You should see a small reduction in Average Active Sessions.
 - Did the findings Change? Navigate to the Details section and click the Finding tab.
 - If there were regressed SQL, select the Top Regressed SQLs line and expand SQL Breakdown in the Description section. This view provide a list of SQL statements that are performing worse than before and the relative performance difference.



18. To prepare for later practices and clean up from this practice, execute the `cleanup` script in the `/home/oracle/workshops` directory.

```
$ ./cleanup KC
```

Practices for Lesson 18: Tuning PGA and Temporary Space

Chapter 18

Practices for Lesson 18: Overview

Practices Overview

In this practice, you tune PGA Memory.

Practice 18-1: Tuning PGA_AGGREGATE_TARGET

You have a decision support system (DSS) that has a variety of queries being performed. Users are complaining that the queries are slow. PGA_AGGREGATE_TARGET is set at 20 MB. Determine the smallest setting that eliminates all multipass work areas.

1. Prepare the database for this scenario by executing the \$HOME/workshops/prepare script with PGA as the input parameter.

```
$ cd $HOME/workshops
$ ./prepare PGA
The Oracle base for
ORACLE_HOME=/u01/app/oracle/product/12.1.0/dbhome_1 is
/u01/app/oracle

Database closed.
Database dismounted.
ORACLE instance shut down.

ORACLE instance started.

Total System Global Area  363302912 bytes
Fixed Size                  2288344 bytes
Variable Size                322962728 bytes
Database Buffers            33554432 bytes
Redo Buffers                 4497408 bytes
Database mounted.
Database opened.

Tablespace created.

Database altered.

System altered.

Finished setup PGA
$
```

2. Start the workload generator with ./workgen PGA. This workload consists of several sessions performing a variety of queries that require different sizes of work areas.

```
$ date
Mon Dec 16 09:04:07 UTC 2013
$ ./workgen PGA
The Oracle base for
ORACLE_HOME=/u01/app/oracle/product/12.1.0/dbhome_1 is
/u01/app/oracle

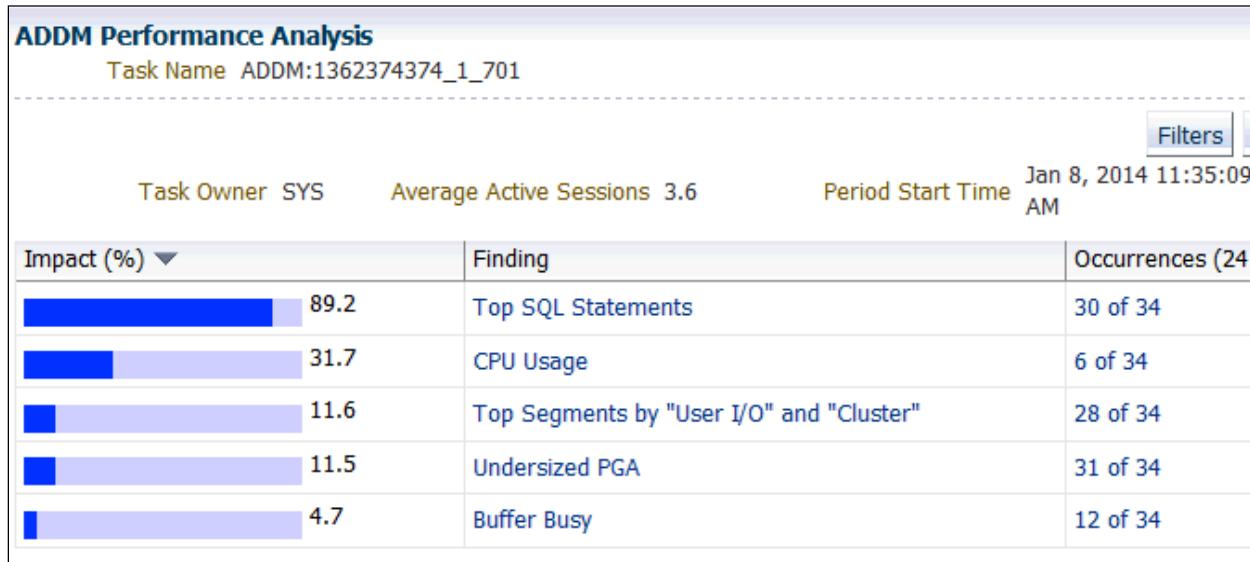
PL/SQL procedure successfully completed.

PL/SQL procedure successfully completed.
```

3. Log in to Enterprise Manager Cloud Control and navigate to the `orcl` database Performance page. Let the workload run for about five minutes. Then create an AWR report, and view the PGA memory advice.

Step	Window/Page Description	Choices or Values
a.	orcl database home	Expand the Performance menu and select Performance Home .
b.	Database Login	Select SYSDBA Database Credentials . Click Login .
c.	Database Instance: orcl	Observe the activity graphs. After about five minutes of activity, click Run ADDM Now .
d.	Confirmation	Click Yes .
e.	Automatic Database Diagnostics Report (ADDM) See example below	Examine the Findings, Undersized PGA should appear near the top of the list. Drill down on the Undersized PGA finding.
f.	Performance Finding Details: Undersized PGA See example below	PGA_AGGREGATE_TARGET is set to 20 M, and a recommended size of 160M is expected. The recommendation does not always appear. Return to the ADDM report
g.	Automatic Database Diagnostics Report (ADDM)	Click View Snapshots .
h.	Snapshot Details	Click the Report tab.

3e)



3f)

Performance Finding Details: Undersized PGA

The PGA was inadequately sized, causing additional I/O to temporary tablespaces to consume significant database time. [Finding History](#)

Finding Impact (Active Sessions)	.41
Percentage of Finding's Impact (%)	 11.5
Period Start Time	Jan 8, 2014 11:35:09 AM
End Time	Jan 8, 2014 11:39:36 AM
Filtered	No Filters

Recommendations

Show All Details | Hide All Details

Details	Category	Benefit (%) ▾
▼ Hide	DB Configuration	 9.2
Action	Increase the size of the PGA by setting the value of parameter "pga_aggregate_target" to 160 M.	Implement Filters

Additional Information

The value of parameter "pga_aggregate_target" was "20 M" during the analysis period.

Findings Path

Expand All | Collapse All

Findings	Percentage of Finding's Impact (%)	Additional Information
▼ The PGA was inadequately sized, causing additional I/O to temporary tablespaces to consume significant database time.	 11.5	Additional Information
Wait class "User I/O" was consuming significant database time.	 18.3	

- Stop the workload with `rm runload`.

```
$ rm runload
```

- Review the AWR report for symptoms and recommendations concerning PGA area sizing.
 - Save the report as `awr_PGA_1.html`.

- b. Check the Top ADDM Findings. Notice the Undersized PGA finding.

Top ADDM Findings by Average Active Sessions

Finding Name	Avg active sessions of the task	Percent active sessions of finding	Task Name	Begin Snap Time	End Snap Time
Top SQL Statements	3.56	89.25	ADDM:1362374374_1_701	08-Jan-14 11:35	08-Jan-14 11:39
CPU Usage	3.56	31.68	ADDM:1362374374_1_701	08-Jan-14 11:35	08-Jan-14 11:39
Undersized PGA	3.56	11.50	ADDM:1362374374_1_701	08-Jan-14 11:35	08-Jan-14 11:39
Top Segments by "User I/O" and "Cluster"	3.56	11.56	ADDM:1362374374_1_701	08-Jan-14 11:35	08-Jan-14 11:39
Buffer Busy	3.56	4.73	ADDM:1362374374_1_701	08-Jan-14 11:35	08-Jan-14 11:39

- c. Check **Top 10 Foreground Events by Total Wait Time**. Is there something that is taking a large amount of % DB Time? *direct path write temp*, and possibly *direct path read temp*

Top 10 Foreground Events by Total Wait Time

Event	Waits	Total Wait Time (sec)	Wait Avg(ms)	% DB time	Wait Class
DB CPU		414.6		43.6	
direct path write temp	5,972	106.7	18	11.2	User I/O
local write wait	1,950	63.4	32	6.7	User I/O
buffer busy waits	534	45	84	4.7	Concurrency
log file sync	206	4.9	24	.5	Commit
direct path read temp	174,310	2.7	0	.3	User I/O
direct path read	21,712	.8	0	.1	User I/O
db file sequential read	6,158	.4	0	.0	User I/O
enq: KO - fast object checkpoint	2	.4	176	.0	Application
direct path sync	2	.1	38	.0	User I/O

- d. Check **Time Model Statistics**. What areas are taking the greatest portion of DB Time?
sql execute elapsed time, DB CPU

Without the ADDM findings, all that you would know is that the SQL is slow, and the main diagnostics agree.

Time Model Statistics

- Total time in database user-calls (DB Time): 951.2s
- Statistics including the word "background" measure background processes
- Ordered by % of DB time desc, Statistic name

Statistic Name	Time (s)	% of DB Time
sql execute elapsed time	938.44	98.66
DB CPU	414.65	43.59
parse time elapsed	10.54	1.11
hard parse elapsed time	10.32	1.08
connection management call elapsed time	6.40	0.67
PL/SQL execution elapsed time	0.30	0.03
PL/SQL compilation elapsed time	0.17	0.02
failed parse elapsed time	0.04	0.00
hard parse (sharing criteria) elapsed time	0.02	0.00
repeated bind elapsed time	0.01	0.00
sequence load elapsed time	0.00	0.00
DB time	951.20	
background elapsed time	76.68	
background cpu time	1.79	

- e. Determine the SQL with the greatest elapsed time (**Back to Top > SQL Statistics > SQL ordered by Elapsed Time**). There are several statements that are taking a large amount of elapsed time and CPU time, but very few executions. This combination of elapsed time and CPU time could be caused by excessive reads or sorting. In the example below, three of the top six SQL statements is also in the top six of **SQL ordered by Reads**, and one out of the top six are also in the **SQL ordered by Gets**. The order of the SQL statements in your output may vary. This indicates that many blocks are being accessed, but most of them are already in the buffer cache. This would indicate that the buffer cache size is not a problem. Confirm this by checking the buffer cache hit ratio (in the next step).

SQL ordered by Elapsed Time

- Resources reported for PL/SQL code includes the resources used by all SQL statements called by the c
- % Total DB Time is the Elapsed Time of the SQL statement divided into the Total Database Time multiplied b
- %Total - Elapsed Time as a percentage of Total DB time
- %CPU - CPU Time as a percentage of Elapsed Time
- %IO - User I/O Time as a percentage of Elapsed Time
- Captured SQL account for 99.9% of Total DB Time (s): 951
- Captured PL/SQL account for 0.7% of Total DB Time (s): 951

Elapsed Time (s)	Executions	Elapsed Time per Exec (s)	%Total	%CPU	%IO	SQL Id
262.96	1	262.96	27.65	52.96	11.84	814rmvhdzkyk6s
200.08	1	200.08	21.03	56.98	0.91	70dwppwx3wnhd
169.88	9	18.88	17.86	54.32	0.00	cj4bx12rruv4p
107.12	9	11.90	11.26	4.83	74.30	7t4wtc73qur3b
55.08	1	55.08	5.79	0.47	72.44	b57rvwy9jbvv6
52.04	20	2.60	5.47	46.71	0.01	8vjh4nbv1phu3

SQL ordered by Gets

- Resources reported for PL/SQL code includes the resources used by all SQL statements called by the c
- %Total - Buffer Gets as a percentage of Total Buffer Gets
- %CPU - CPU Time as a percentage of Elapsed Time
- %IO - User I/O Time as a percentage of Elapsed Time
- Total Buffer Gets: 413,916
- Captured SQL account for 95.9% of Total

Buffer Gets	Executions	Gets per Exec	%Total	Elapsed Time (s)	%CPU	%IO	SQL Id
94,348	4	23,587.00	22.79	4.16	19.7	38.7	f58fb5n0yvr7c
47,031	9	5,225.67	11.36	107.12	4.8	74.3	7t4wtc73qur3b
34,020	21	1,620.00	8.22	8.02	48.3	1.9	3q36d1uum3662
32,871	1	32,871.00	7.94	1.13	88.1	12.5	f2fn1pip86yh3
32,400	20	1,620.00	7.83	2.15	45.9	8.9	8rtsdaxm7apq7
31,980	21	1,522.86	7.73	0.51	52.5	23.9	brijahu0up7qrh
27,767	2	13,883.50	6.71	4.64	24.2	18.5	fhf8upax5cxsz

SQL ordered by Reads

- %Total - Physical Reads as a percentage of Total Disk Reads
- %CPU - CPU Time as a percentage of Elapsed Time
- %IO - User I/O Time as a percentage of Elapsed Time
- Total Disk Reads: 881,053
- Captured SQL account for 99.5% of Total

Physical Reads	Executions	Reads per Exec	%Total	Elapsed Time (s)	%CPU	%IO	SQL Id
679,058	1	679,058.00	77.07	262.96	52.96	11.84	814rmvhzyk6s
33,348	21	1,588.00	3.79	8.02	48.33	1.89	3q36d1uum3662
32,762	4	8,190.50	3.72	4.16	19.73	38.72	f58fb5n0yvr7c
31,860	21	1,517.14	3.62	0.51	52.51	23.89	brjahu0up7qrh
31,776	20	1,588.80	3.61	2.15	45.86	8.91	8rtsdaxm7apq7
26,931	9	2,992.33	3.06	107.12	4.83	74.30	7t4wtc73qur3b

- f. Navigate to the **Instance Efficiencies Percentage** report in the Report Summary section ([Back to Top > Report Summary](#)). What is the buffer cache hit ratio? (above 95% is expected)

Instance Efficiency Percentages (Target 100%)

Buffer Nowait %:	99.58	Redo NoWait %:	100.00
Buffer Hit %:	96.34	In-memory Sort %:	99.82
Library Hit %:	88.16	Soft Parse %:	90.82
Execute to Parse %:	58.26	Latch Hit %:	99.99
Parse CPU to Parse Elapsd %:	63.12	% Non-Parse CPU:	99.20

- g. Check **Load Profile** for logical reads and physical reads to confirm the buffer cache hit ratio. Note the physical reads per second and logical reads per second.

With a buffer cache hit ratio of nearly 99%, the physical reads should be about 1% of the logical reads. Is this true? *No, the physical reads are much higher than 1%.*

There are physical reads that are not counted in the buffer cache hit ratio because the blocks do not go through the buffer cache. These are direct reads.

Load Profile					
	Per Second	Per Transaction	Per Exec	Per Call	
DB Time(s):	3.6	67.9	0.05	0.02	
DB CPU(s):	1.6	29.6	0.02	0.01	
Redo size (bytes):	12,433.2	236,796.0			
Logical read (blocks):	1,552.4	29,565.4			
Block changes:	59.9	1,140.7			
Physical read (blocks):	3,304.3	62,932.4			
Physical write (blocks):	120.4	2,292.9			
Read IO requests:	761.3	14,499.4			
Write IO requests:	32.3	615.4			
Read IO (MB):	25.8	491.7			
Write IO (MB):	0.9	17.9			
User calls:	218.1	4,153.9			
Parses (SQL):	31.4	598.9			
Hard parses (SQL):	2.9	55.0			
SQL Work Area (MB):	3.6	67.7			
Logons:	0.7	14.1			
Executes (SQL):	75.3	1,434.9			
Rollbacks:	0.0	0.0			
Transactions:	0.1				

- h. Confirm that direct reads are a large part of the physical reads by checking the **Instance Activity Statistics** section for physical reads (**Main Report > Instance Activity Statistics > Key Instance Activity Stats and Other Instance Activity Stats**). Of the physical reads in the following example, a large percentage is direct reads. Most of the direct reads, shown as “physical reads direct,” are to temporary tablespaces, shown as “physical reads direct temporary tablespace.” Physical reads on a temporary tablespace usually indicate a sorting operation and will have nearly matching physical writes. It appears that there may be a problem with temporary space usage, which usually means sort and join operations. Because physical reads and writes are 10-1000 times more expensive in time than in-memory operations such as gets, even reducing the physical operations a small amount can significantly increase performance.

Note: The workarea executions in the Key Activity Statistics show that there are significant sub-optimal sorts being performed.

Key Instance Activity Stats

- Ordered by statistic name

Statistic	Total	per Second	per Trans
db block changes	15,970	59.89	1,140.71
execute count	20,088	75.34	1,434.86
logons cumulative	198	0.74	14.14
opened cursors cumulative	19,977	74.92	1,426.93
parse count (total)	8,384	31.44	598.86
parse time elapsed	526	1.97	37.57
physical reads	881,053	3,304.32	62,932.36
physical writes	32,101	120.39	2,292.93
redo size	3,315,144	12,433.17	236,796.00
session cursor cache hits	16,052	60.20	1,146.57
session logical reads	413,916	1,552.36	29,565.43
user calls	58,155	218.11	4,153.93
user commits	14	0.05	1.00
user rollbacks	0	0.00	0.00
workarea executions - multipass	18	0.07	1.29
workarea executions - onepass	6	0.02	0.43
workarea executions - optimal	4,024	15.09	287.43
physical reads cache	8,981	33.68	641.50
physical reads cache prefetch	2,023	7.59	144.50
physical reads direct	872,072	3,270.63	62,290.86
physical reads direct (lob)	2	0.01	0.14
physical reads direct temporary tablespace	703,750	2,639.36	50,267.86
physical reads prefetch warmup	0	0.00	0.00
physical write IO requests	8,615	32.31	615.36
physical write bytes	262,971,392	986,252.44	18,783,670.86
physical write total IO requests	9,128	34.23	652.00
physical write total bytes	269,305,856	1,010,009.32	19,236,132.57
physical write total multi block requests	17	0.06	1.21
physical writes direct	29,162	109.37	2,083.00
physical writes direct (lob)	1	0.00	0.07
physical writes direct temporary tablespace	29,161	109.37	2,082.93
physical writes from cache	2,939	11.02	209.93

- i. Look at the SQL text of the top 5 **SQL ordered by Elapsed Time** ([Back to Top > SQL Statistics > SQL ordered by Elapsed Time](#)). All these statements have a group by or join, or both. These operations force sorts and additional reads if the PGA is not properly sized. In the example below, a table has been added to show the SQL text. In the AWR report, you can click the SQL_ID to jump to the full SQL text.

SQL ordered by Elapsed Time

- Resources reported for PL/SQL code includes the resources used by all SQL statements called
- % Total DB Time is the Elapsed Time of the SQL statement divided into the Total Database Time
- %Total - Elapsed Time as a percentage of Total DB time
- %CPU - CPU Time as a percentage of Elapsed Time
- %IO - User I/O Time as a percentage of Elapsed Time
- Captured SQL account for 99.9% of Total DB Time (s): 951
- Captured PL/SQL account for 0.7% of Total DB Time (s): 951

Elapsed Time (s)	Executions	Elapsed Time per Exec (s)	%Total	%CPU	%IO	SQL Id
262.96	1	262.96	27.65	52.96	11.84	814rmvhdyk6s
200.08	1	200.08	21.03	56.98	0.91	70dwppwx3wnhd
169.88	9	18.88	17.86	54.32	0.00	cj4bx12rruv4p
107.12	9	11.90	11.26	4.83	74.30	7t4wtc73qur3b
55.08	1	55.08	5.79	0.47	72.44	b57rvwy9jbvv6
52.04	20	2.60	5.47	46.71	0.01	8vjh4nbv1phu3
42.94	21	2.04	4.51	45.26	0.00	6rz119f0r1wqy

SQL ID	Text
6rz119f0r1wqy	select s.prod_id, sum(s.amount_sold) - sum(c.unit_cost) from costs c, sales s where c.prod_id = s.prod_id and s.prod_id = 142 group by s.prod_id
70dwppwx3wnhd	select s.prod_id, sum(s.amount_sold) - sum(c.unit_cost) from costs c, sales s where c.prod_id = s.prod_id and s.prod_id > 114 group by s.prod_id
7t4wtc73qur3b	select * from (select * from dba_objects order by timestamp) where rownum < 80000
814rmvhdyk6s	select s.prod_id, sum(s.amount_sold) - sum(c.unit_cost) from costs c, sales s where c.prod_id = s.prod_id and s.prod_id < 40 group by s.prod_id
8vjh4nbv1phu3	select sum(s.AMOUNT_sold) - sum(c.unit_cost) from costs c, sales s where c.prod_id = s.prod_id and s.prod_id = 139
b57rvwy9jbvv6	select * from customers order by cust_gender, cust_marital_status
cj4bx12rruv4p	select s.prod_id, SUM(s.amount_sold) - SUM(c.unit_cost) from costs c, sales s

	where c.prod_id = s.prod_id and s.prod_id = 117 group by s.prod_id
fkmrxvv6fxqar	select * from (select * from dba_objects order by timestamp) where rownum < 30000

- j. Determine the Mbytes of extra writes that are caused by the current PGA sizing. Check **PGA Aggr Summary** in the Advisory Statistics section ([Back to Top > Advisory Statistics > PGA Aggr Summary](#)).
- k. What is the PGA Cache Hit ratio? What is the Extra W/A (work area) MB Read/Written? The example indicates that 1,048 MB were read and written because the work area could not be completed in memory.

PGA Aggr Summary

- PGA cache hit % - percentage of W/A (WorkArea) data processed

PGA Cache Hit %	W/A MB Processed	Extra W/A MB Read/Written
47.48	948	1,048

- I. Determine the recommended PGA sizing from the AWR report. In the Advisories section, find the **PGA Memory Advisory** ([Back to Advisory Statistics > PGA Memory Advisory](#)). To find the minimum size with no multipass work areas, choose the smallest value in which **Estd PGA Overalloc Count is 0**. Using the example, the chart includes a row where Estd PGA Overalloc Count is 0, that is **160 MB**. Your results may vary.

PGA Memory Advisory

- When using Auto Memory Mgmt, minimally choose a pga_aggregate_target value where Estd PGA Overalloc Count is 0

PGA Target Est (MB)	Size Factor	W/A MB Processed	Estd Extra W/A MB Read/Written to Disk	Estd PGA Cache Hit %	Estd PGA Overalloc Count	Estd Time
10	0.50	987.74	965.20	51.00	33	886,857
15	0.75	987.74	965.20	51.00	33	886,857
20	1.00	987.74	576.40	63.00	33	710,299
24	1.20	987.74	576.40	63.00	33	710,299
28	1.40	987.74	576.40	63.00	33	710,299
32	1.60	987.74	576.40	63.00	33	710,299
36	1.80	987.74	576.40	63.00	33	710,299
40	2.00	987.74	576.40	63.00	33	710,299
60	3.00	987.74	576.40	63.00	33	710,299
80	4.00	987.74	576.40	63.00	30	710,299
120	6.00	987.74	323.57	75.00	4	595,485
160	8.00	987.74	111.68	90.00	0	499,262

- m. Observe the number of multipass and one-pass work areas that are being used. Find **PGA Aggr Target Histogram** in the Advisories section of the AWR report ([Back to Advisory Statistics > PGA Aggr Target Histogram](#)). The work areas that cannot be contained in memory write to disk (temporary tablespace) and are recorded as one-pass or multipass work areas. In PGA Aggr Target Histogram, the column on the right shows the minimum requested size for a one-pass execution. The example shows a number of multipass executions and one-pass executions for sorts larger than 1 MB.

PGA Aggr Target Histogram

- Optimal Executions are purely in-memory operations

Low Optimal	High Optimal	Total Execs	Optimal Execs	1-Pass Execs	M-Pass Execs
2K	4K	3,506	3,506	0	0
64K	128K	34	34	0	0
128K	256K	7	7	0	0
512K	1024K	142	142	0	0
1M	2M	298	296	1	1
2M	4M	65	65	0	0
8M	16M	20	0	4	16
16M	32M	2	0	1	1

6. Set `PGA_AGGREGATE_TARGET` using the value that you determined from the PGA Memory Advisory in Step 5!. The goal is to eliminate multipass executions. In a terminal window, connect as `sysdba` and change the value of `PGA_AGGREGATE_TARGET`.

```
$ sqlplus / as sysdba
SQL> ALTER SYSTEM SET PGA_AGGREGATE_TARGET = 160M;
SQL> exit
```

7. Execute the workload with the `./workgen PGA` script.

```
$ date
Mon Dec 16 07:52:36 UTC 2013
$ ./workgen PGA
The Oracle base for
ORACLE_HOME=/u01/app/oracle/product/12.1.0/dbhome_1 is
/u01/app/oracle

PL/SQL procedure successfully completed.

PL/SQL procedure successfully completed.

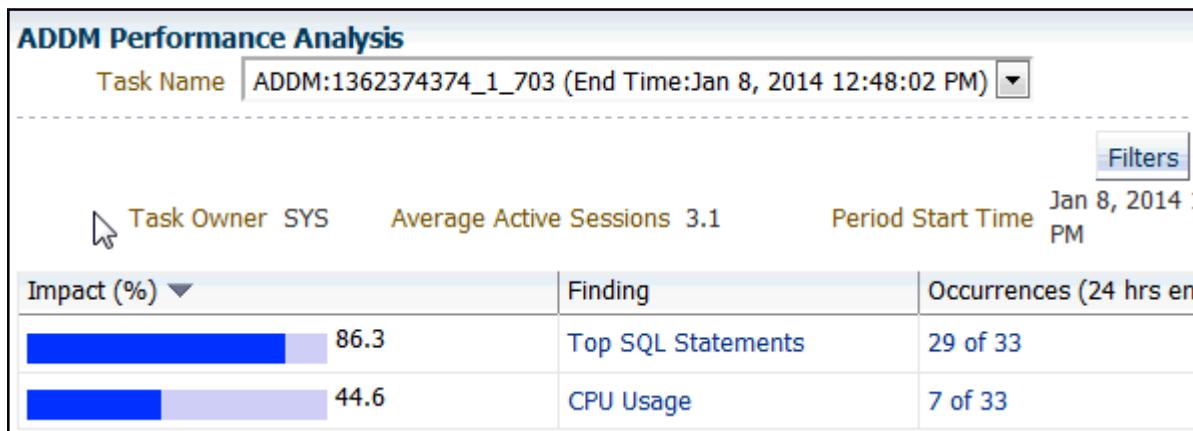
$
```

8. After the workload has been running for about the same duration as the first run, create an ADDM report and an AWR report for the last two snapshots.

```
$ date
Mon Dec 16 07:57:41 UTC 2013
```

Step	Window/Page Description	Choices or Values
a.	Current page	Expand the Performance menu, Select Performance Home .
b.	Database Instance	Click Run ADDM Now .
c.	Confirmation	Click Yes .
d.	Automatic Database Diagnostics Report (ADDM)	Examine the Findings. Undersized PGA may still appear in the list. Drill down on the Undersized PGA finding, if it exists, else skip to step f.
e.	Performance Finding Details: Undersized PGA	The PGA_AGGREGATE_TARGET is set to 160 M, and a recommended size of 256 MB suggested. Return to the ADDM report.
f.	Automatic Database Diagnostics Report (ADDM)	Click View Snapshots .
g.	Snapshot Details	Click the Report tab.

8d)



9. Stop the workload with `rm unload`.

```
$ rm unload
```

10. Review the AWR report for symptoms and recommendations concerning PGA area sizing.
- Save the report as `awr_PGA_2.html`.
 - Check the Top ADDM Findings. Notice the Undersized PGA finding. The impact of the finding should have decreased, and may have been eliminated.

- c. What is the PGA Cache Hit ratio? What is the Extra W/A (work area) MB Read/Written? Has it improved since the last run?

PGA Aggr Summary

- PGA cache hit % - percentage of W/A (WorkArea) data processed or

PGA Cache Hit %	W/A MB Processed	Extra W/A MB Read/Written
100.00	1,962	0

- d. What does the PGA Memory Advisory recommend? *160 MB in the example*

PGA Memory Advisory

- When using Auto Memory Mgmt, minimally choose a pga_aggregate_target value where Estd PGA Overalloc Count is 0

PGA Target Est (MB)	Size Factor	W/A MB Processed	Estd Extra W/A MB Read/Written to Disk	Estd PGA Cache Hit %	Estd PGA Overalloc Count	Estd Time
20	0.13	1,953.05	1,410.13	58.00	47	5,264,529
40	0.25	1,953.05	1,410.13	58.00	47	5,264,529
80	0.50	1,953.05	1,368.45	59.00	46	5,199,284
120	0.75	1,953.05	322.64	86.00	0	3,562,227
160	1.00	1,953.05	0.00	100.00	0	3,057,187
192	1.20	1,953.05	0.00	100.00	0	3,057,187
224	1.40	1,953.05	0.00	100.00	0	3,057,187
256	1.60	1,953.05	0.00	100.00	0	3,057,187
288	1.80	1,953.05	0.00	100.00	0	3,057,187
320	2.00	1,953.05	0.00	100.00	0	3,057,187
480	3.00	1,953.05	0.00	100.00	0	3,057,187
640	4.00	1,953.05	0.00	100.00	0	3,057,187
960	6.00	1,953.05	0.00	100.00	0	3,057,187
1,280	8.00	1,953.05	0.00	100.00	0	3,057,187

- e. Are there any suboptimal sorts? Notice that the all the suboptimal sorts have been eliminated in this example.

PGA Aggr Target Histogram

- Optimal Executions are purely in-memory operations

Low Optimal	High Optimal	Total Execs	Optimal Execs	1-Pass Execs	M-Pass Execs
2K	4K	4,772	4,772	0	0
64K	128K	46	46	0	0
128K	256K	15	15	0	0
256K	512K	4	4	0	0
512K	1024K	368	368	0	0
1M	2M	732	732	0	0
2M	4M	74	74	0	0
8M	16M	44	44	0	0

Note: The goal of eliminating multipass work areas was achieved in the example. The number of one-pass work areas may not be completely eliminated in your test. The one-pass sorts cannot be directly compared due to the differences in time periods of the reports and the multipass sorts that become one-pass sorts. There is the additional complication

that because the system is more efficient, more statements—and thus more sorts—can be performed in the same period.

11. Clean up this practice by executing the `./cleanup PGA` command

```
$ ./cleanup PGA
```


Practices for Lesson 19: Automatic Memory Management

Chapter 19

Practices for Lesson 19: Overview

Practices Overview

The goal of this practice is to use the Automatic Memory Tuning capability of Oracle Database 12c.

Assumptions: The scripts in the practice are being run in a terminal window as the `oracle` user and the environment variables have been set for the `orcl` instance.

Practice 19-1: Enabling Automatic Shared Memory Management

In this practice, you enable automatic shared memory management. The initial configuration is simulating a memory-constrained system. The buffer cache is 20 MB, and the shared pool is 200 MB. The shared pool is loaded with kept objects as in a previous practice. For this practice, SGA_MAX_SIZE is set to 600 MB and MEMORY_MAX_TARGET is set to 806 MB. These values are larger than necessary so that you can easily determine the SGA optimal size, without having to adjust these values and restart the database instance multiple times.

1. Prepare the instance for this practice.

- a. Log out of Enterprise Manager to avoid errors due to lost connection.
- b. Run the prepare script with the ASMM parameter.

```
$ cd /home/oracle/workshops
$ ./prepare ASMM
The Oracle base for
ORACLE_HOME=/u01/app/oracle/product/12.1.0/dbhome_1 is
/u01/app/oracle

Database closed.
Database dismounted.
ORACLE instance shut down.

ORACLE instance started.

Total System Global Area  626327552 bytes
Fixed Size                  2291472 bytes
Variable Size                595593456 bytes
Database Buffers              20971520 bytes
Redo Buffers                  7471104 bytes
Database mounted.
Database opened.
Finished setup ASMM
$
```

2. Enable the Automatic Shared Memory Management feature by using Enterprise Manager Cloud Control. Assume that SGA memory is constrained to 600 MB. TARGET SGA should be set to a value that is as low as possible with a minimum of performance issues.

Step	Window/Page Description	Choices or Values
a.		Log in to Enterprise Manager Cloud Control with the username sysman and password oracle_4U .
b.	Enterprise Summary	Expand the Targets menu and select Databases .
c.	Databases	Click the <code>orcl</code> database link.
d.	Orcl database home	Expand the Performance menu and select Memory Advisors .
e.	Database Login	Select Preferred and SYSDBA Database Credentials . Click Login .
f.	Memory Advisors (See the example screenshot, which follows the table.)	Verify that Maximum SGA Size (MB) is set to 600 MB . Click Enable for Automatic Shared Memory Management.
g.	Enable Automatic Shared Memory Management	The memory size displayed is the sum of the current pool settings. Change this value to 280 MB . (The value must be evenly divisible by 4.) Note that smaller values of Total SGA Size (MB) will cause the Total SGA Size to default to a much larger SGA Size. Click OK .
h.	Memory Advisors	A confirmation message is shown. Click Refresh . Note: A refresh is needed to view the changed memory values.

- 2f) The following screenshot shows the Memory Advisors page, with Automatic Memory Management (AMM) and Automatic Shared Memory Management (ASMM) disabled. The sizes you see for the Shared Pool and Buffer Cache allocations may vary from the example below.

Memory Advisors

When Automatic Memory Management is enabled, the database will automatically set the optimal distribution of memory to accommodate changes in the workload.

Automatic Memory Management Disabled

SGA **PGA**

The System Global Area (SGA) is a group of shared memory structures that contains data and control information for the Oracle database instance is started.

Automatic Shared Memory Management Disabled

Shared Pool	212	MB	<input type="button" value="Advice"/>
Buffer Cache	8	MB	<input type="button" value="Advice"/>
Large Pool	4	MB	<input type="button" value="Advice"/>
Java Pool	12	MB	<input type="button" value="Advice"/>
Other (MB)	13		
Total SGA (MB)		249	<input type="button" value="Calculate"/>

Shared Pool (85%)
Buffer Cache (3.2%)
Large Pool (1.6%)
Java Pool (4.8%)
Other (5.3%)

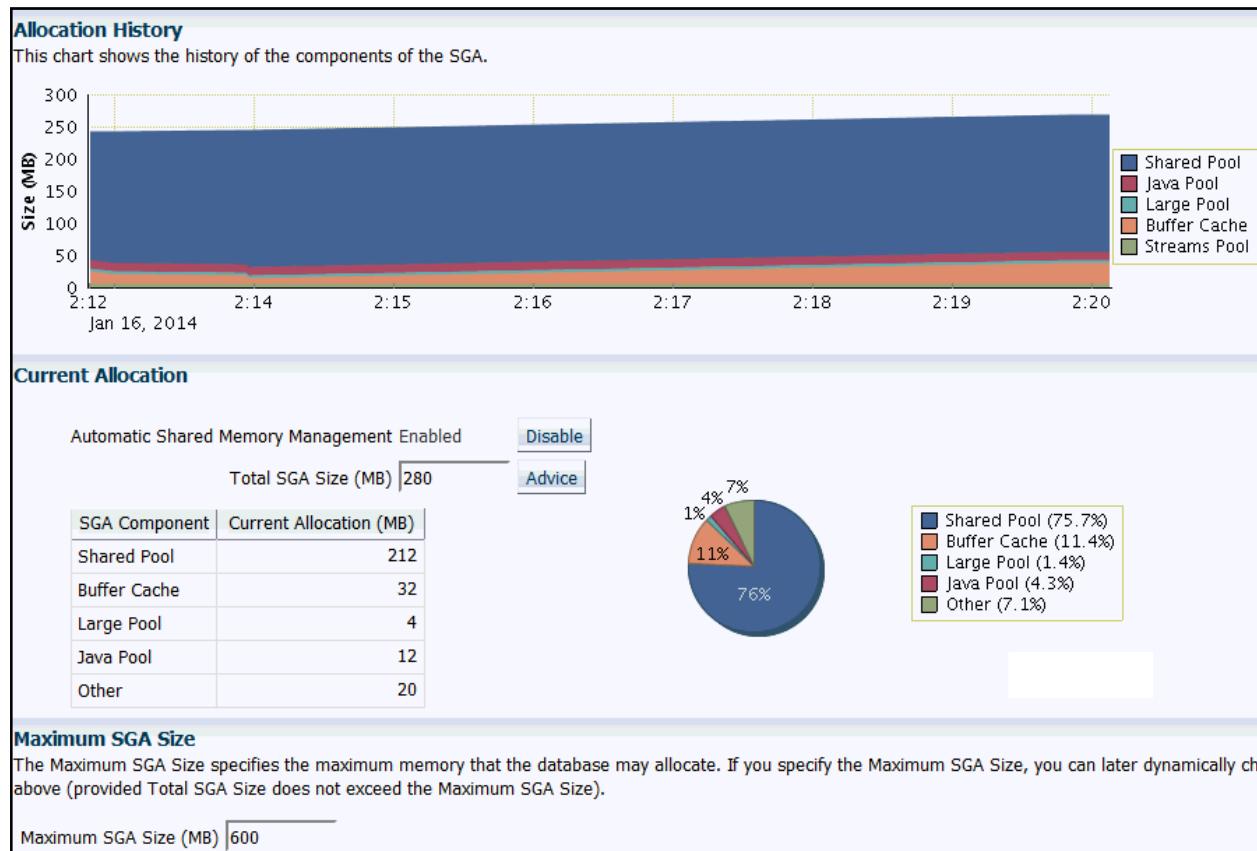
Maximum SGA Size

The Maximum SGA Size specifies the maximum memory that the database may allocate. If you specify the Maximum SGA Size, the database automatically sets the other SGA components to the remaining memory (provided the total SGA size does not exceed the Maximum SGA Size).

Maximum SGA Size (MB)

The database must be restarted before any changes to this value take effect.

- 2h) The following screenshot shows the Memory Advisors page. Because the allocations depend on the history and the current workloads, background processes may cause the allocations to vary from what is shown.



3. Run the workload generator `./workgen ASMM`.

```
$ ./workgen ASMM
The Oracle base for
ORACLE_HOME=/u01/app/oracle/product/12.1.0/dbhome_1 is
/u01/app/oracle

PL/SQL procedure successfully completed.

PL/SQL procedure successfully completed.

Starting buffer cache load
Starting Soft Parse load
Starting Hard Parse load

System altered.

PL/SQL procedure successfully completed.

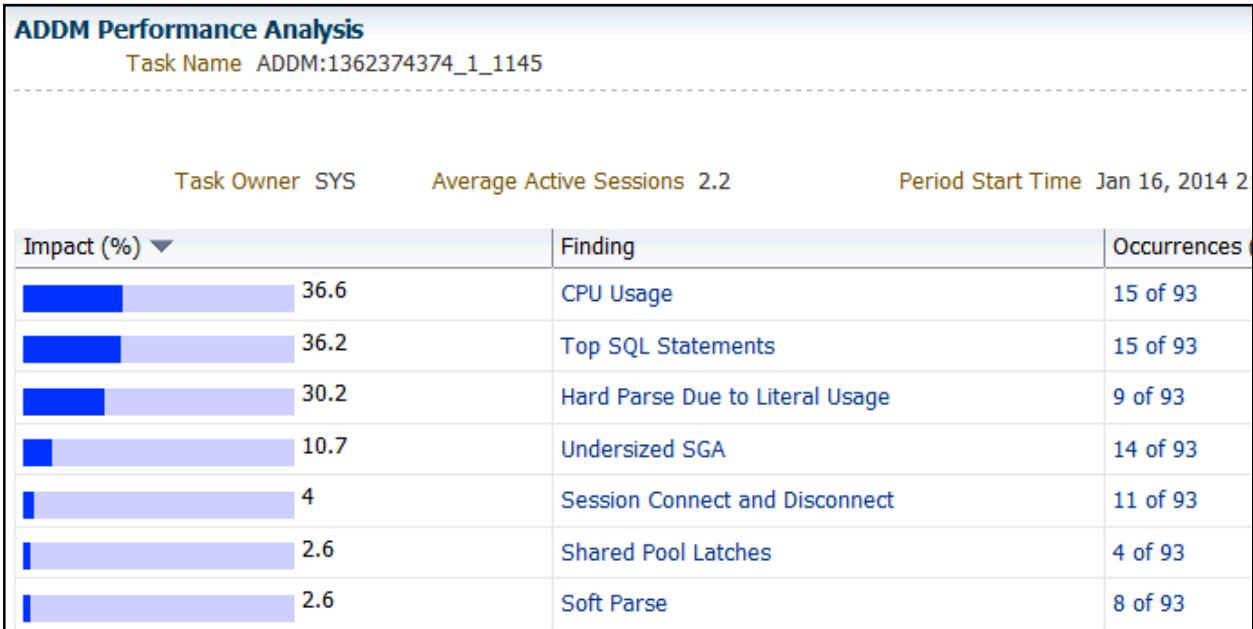
PL/SQL procedure successfully completed.
```

PL/SQL procedure successfully completed.
\$

4. Return to Enterprise Manager Cloud Control and view ADDM performance analysis findings.
 - a. Navigate to the Performance Home page by expanding the **Performance** menu and selecting **Performance Home**.
 - b. Set the View Data refresh rate to Real Time: 1 Minute Refresh.
 - c. After about five minutes, click “**Run ADDM Now**” to generate an ADDM report. Click **Yes** on the Confirmation page.
 - d. View the ADDM report. For any findings related to memory sizing, drill down to the details. There may be findings that are related to SQL statements and CPU usage, such as “CPU Usage” or “Top SQL by DB Time.” There should be an “Undersized SGA” finding. Note the ADDM period start and end times:

Start Time; _____ End Time _____

What is the Average Active Sessions value ? _____



- e. View the finding details for “Undersized SGA” and view the recommendation. What is the recommended size for the SGA? (The recommendation will vary.)

Performance Finding Details: Undersized SGA

Finding Impact (Active Sessions)	.23	The SGA was inadequately sized, causing additional I/O or hard parses.	Finding
Percentage of Finding's Impact (%)	10.7		
Period Start Time	Jan 16, 2014 2:23:29 PM		
End Time	Jan 16, 2014 2:28:20 PM		
Filtered	No	Filters	

Recommendations

Show All Details | Hide All Details

Details	Category	Benefit (%)
Hide DB Configuration		
Action Increase the size of the SGA by setting the parameter "sga_target" to 350 M.		Implement Filters

Additional Information

The value of parameter "sga_target" was "280 M" during the analysis period.

Findings Path

Expand All | Collapse All

Findings	Percentage of Finding's Impact (%)
▼ The SGA was inadequately sized, causing additional I/O or hard parses.	10.7
Hard parsing of SQL statements was consuming significant database time.	30.6

- f. Return to the ADDM page by clicking the link at the top of the page.
 g. On the ADDM page, click **View Snapshots**, and then click the **Report** tab.
 h. In the Report Summary section, examine the cache sizes. What are the start and end sizes for the buffer cache and shared pool? *The sizes in your report may vary from the example shown below.*

Cache	Begin	End
Buffer Cache		
Shared Pool Size		

Cache Sizes		Begin	End	
Buffer Cache:		32M	28M	Std Block Size: 8K
Shared Pool Size:		212M	216M	Log Buffer: 7,296K

- i. What does the SGA Target Advisory in the AWR report recommend? In the Main Report menu, Click **Advisory Statistics > SGA Target Advisory**. *The number of estimated physical reads to the size of the SGA is not a smooth curve. We expect that for each increase in SGA size, there should be some decrease in physical reads. But this is not always true. In the example, an increase in physical reads can be observed with an increase of SGA size. In any case, there should be a point where increasing the SGA size further does not significantly reduce the number of physical reads. The optimal point, the smallest memory size that still yields close to the smallest Estimated DB Time. Smaller DB time does not always correlate to reduced Physical Reads.*

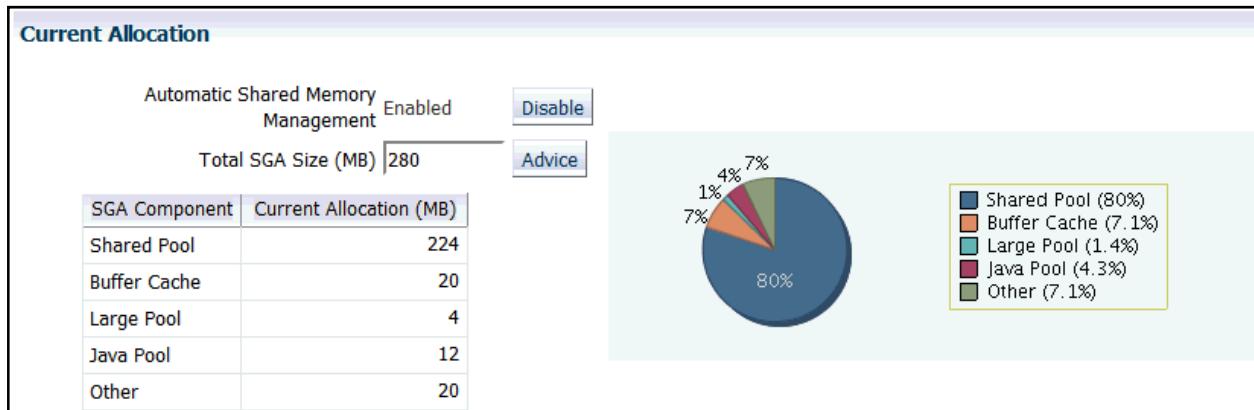
SGA Target Size (M)	SGA Size Factor	Est DB Time (s)	Est Physical Reads
280	1.00	679	125,206
350	1.25	640	15,187
420	1.50	622	21,160
490	1.75	613	21,160
560	2.00	612	4,720

5. Return to the Memory Advisors page by expanding the **Performance** menu and selecting **Memory Advisors**. What are the current memory parameter settings?

The memory parameter settings may vary depending on the machine that you are using.

- Buffer Cache (20 MB)
- Shared Pool (224 MB)
- Large Pool (4 MB)
- Java Pool (12 MB)
- Other (20 MB)

Note: The Memory Advisors page does not show the value of the keep pool separately. It is included in Other.



6. Stop the workload with the `rm runload` command.

```
$ rm runload
$
```

7. In the AWR SGA Target Advisory, the greatest benefit for the memory added will be seen at approximately 420 MB, the first inflection point in the SGA Target Advisory. However, additional benefit will be gained until approximately 480 MB, when there is little or no more benefit for adding memory.

- On the Memory Advisors page, change **Total SGA Size** to **420 MB**. Click **Apply**.
- Refresh the page to get accurate values.
- Observe the initial settings for the SGA on the Memory Advisors page.

The values for the shared pool and buffer cache may vary based on the machine you are using.

8. Start the workload again: `./workgen ASMM`.

```
$ ./workgen ASMM
```

9. Return to the **Performance Home** page. Observe the performance graphs for about five minutes. What are the differences between this and the previous run?

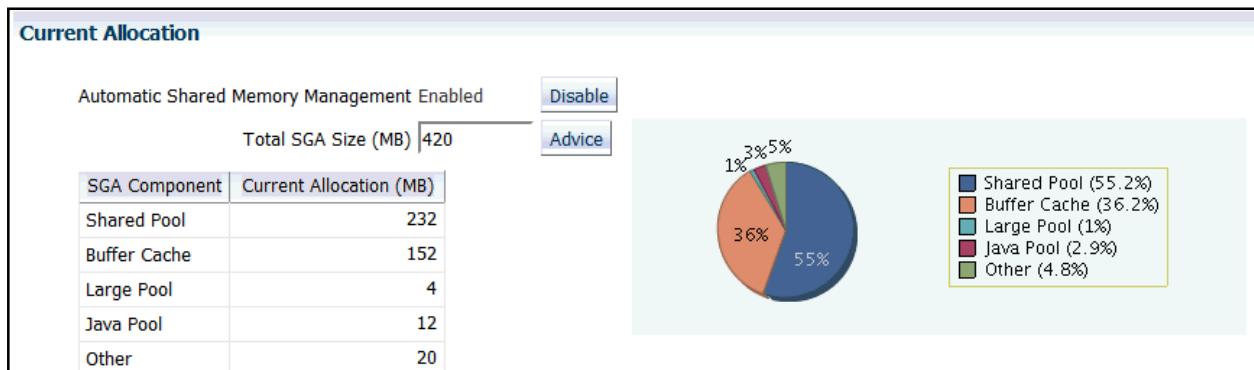
The Average Active Sessions graph and the Load average graphs may show small differences. The I/O Megabytes per Second by I/O Function graph shows a large difference in the number of buffer cache reads.

10. Observe the current settings for the SGA on the **Memory Advisors** page. (Click Refresh to view any changes in the distribution of memory.)

What is the value of the buffer cache?

What is the value of the shared pool size?

The values for the shared pool and buffer cache may vary based on the machine you are using.



11. After about five minutes, stop the workload with `rm runload`.

```
$ rm runload
$
```

12. Generate a new ADDM report and review the findings.

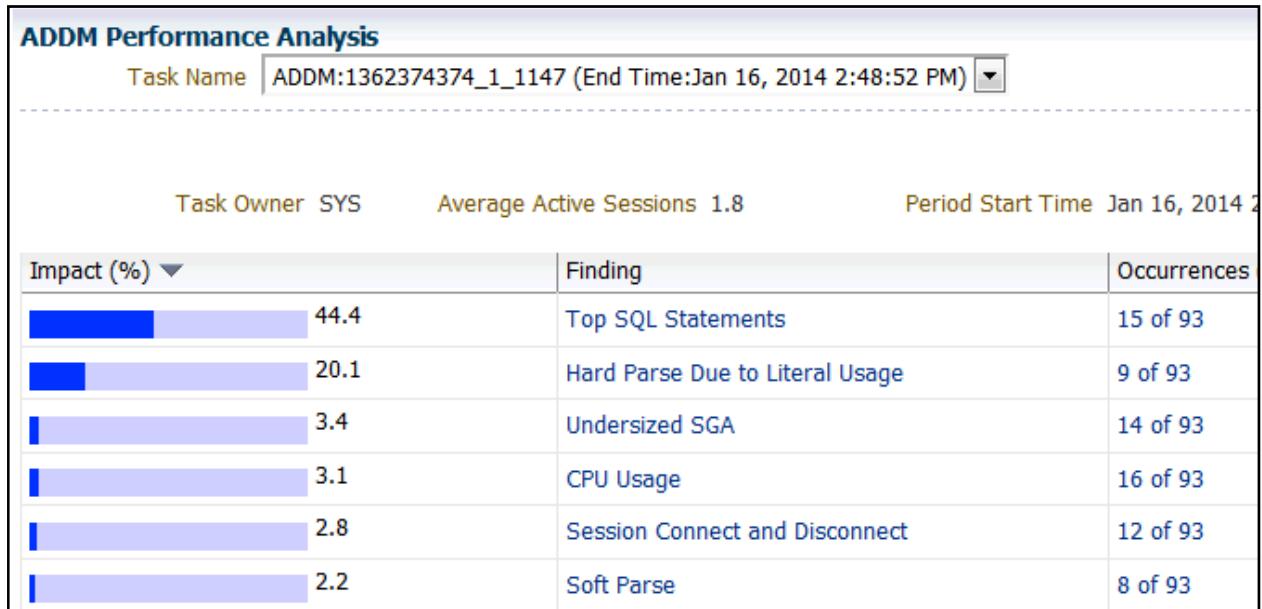
- Click **Run ADDM Now** on the Performance Home page to run an ADDM report.
- Record the start and end times of this ADDM report.

Start Time: _____

End Time: _____

- View the results and drill down to the detail pages.
- The number of findings and the impact of the issues may be misleading. It is common that correcting memory issues has a much greater impact than is estimated. It is possible that there are more findings and Impact % may be larger. Notice that the

number of Average Active Sessions is reduced even though the load was identical. This indicates that the instance is running more efficiently. Because there are fewer total waits, each wait category has increased Impact%.



- e. The finding "Undersized SGA" may still be reported. In the example, the recommendation is to increase the SGA to 525 MB. The impact of this issue as shown in the screenshot is 3.4%. The recommended size may range to a much higher value.

Performance Finding Details: Undersized SGA

Finding The SGA was inadequately sized, causing additional I/O or hard parses. [Finding History](#)

Impact (Active Sessions) .06
 Percentage of Finding's Impact (%) 3.4
 Period Start Time Jan 16, 2014 2:42:40 PM
 End Time Jan 16, 2014 2:48:52 PM
 Filtered No [Filters](#)

Recommendations

Show All Details | Hide All Details

Details	Category	Benefit (%)
Hide	DB Configuration	2.9

Action Increase the size of the SGA by setting the parameter "sga_target" to 525 M. [Implement](#) [Filters](#)

Additional Information

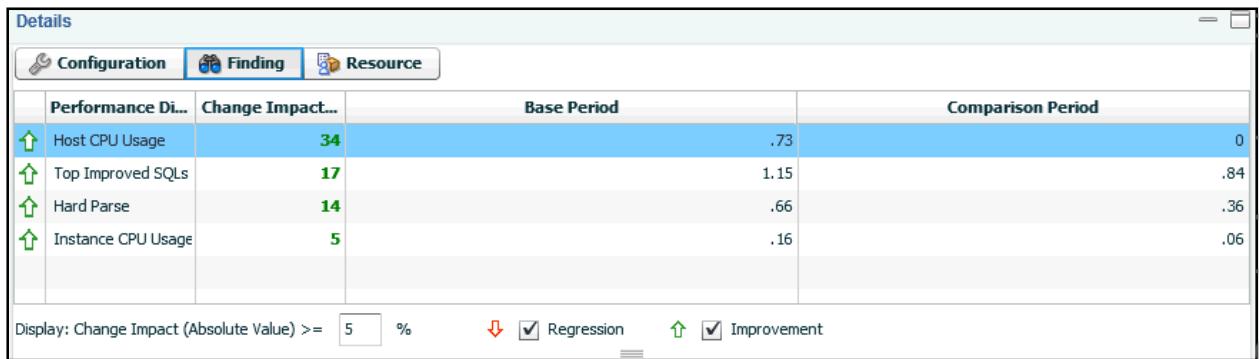
The value of parameter "sga_target" was "420 M" during the analysis period.

Findings Path

Expand All | Collapse All

Findings	Percentage of Finding's Impact (%)	Additional Information
The SGA was inadequately sized, causing additional I/O or hard parses.	3.4	Additional Information
Hard parsing of SQL statements was consuming significant database time.	20.4	

- f. A careful comparison of the AWR reports from this and the previous runs will show an improvement. In the load profile, you may see a small improvement in DB Time per second, and a significant reduction in physical reads.
13. Use the Compare Period ADDM report to see the differences in the last two periods.
- Navigate to **Performance > AWR > Compare Period ADDM**.
 - In section Step 1, set begin and end times to those of your most recent ADDM report. These are the times you recorded in the previous step.
 - In section Step 2, select **Customize** and set the begin and end times to the values you recorded in Step 4d.
 - Click **Run**.
 - The Overview section shows that much more time was spent in the CPU, than in waits. Notice that the number of Active sessions (the scale along the Y axis) is somewhat reduced in the second period. Scroll down if needed to see the Average active Sessions value for each period. Because you know that the workload is the same, this alone shows that the instance is functioning more efficiently.
 - On the Details Section Configuration tab, notice that the SGA Target parameter changed. Click the Finding tab.



- Notice in the example that all the findings show improvement, your results may vary, and there could be minor regressions.
 - Check the Resource tab. Is there an improvement in either the CPU resource or I/O resources? *If there is significant improvement in overall performance, you could expect improvement in one or both of these resources.*
14. What would be a reasonable next tuning step based on the ADDM report?
- Note:** The number and ranking of the issues shown in the ADDM reports of various machines will vary.
- Hard Parse: (If seen) This is likely to be literals in the SQL statements. The impact is larger because some of the other findings have been eliminated.
 - CPU Usage: The hard-parse activity uses CPU. Implementing a solution for the hard-parse finding will impact CPU usage.
 - Soft Parse: This is primarily an application issue. Check the SESSION_CURSOR_CACHE value and set it to a higher value if it is already set.
 - Free Buffer waits: This indicate that the DBWR process is not freeing buffers quickly enough. The recommendation is to increase the value of the DB_WRITER_PROCESSES parameter.

- e. Buffer Busy Waits: This could be due to locally managed tablespace for the objects in question, or an application issue where multiple sessions are writing to the same block at the same time.
- f. The sizing of the SGA has an impact on a variety of issues, which is usually larger than the impact rating assigned to it. Therefore, it is reasonable to assume that if the SGA is increased again, there will be further improvements in performance.

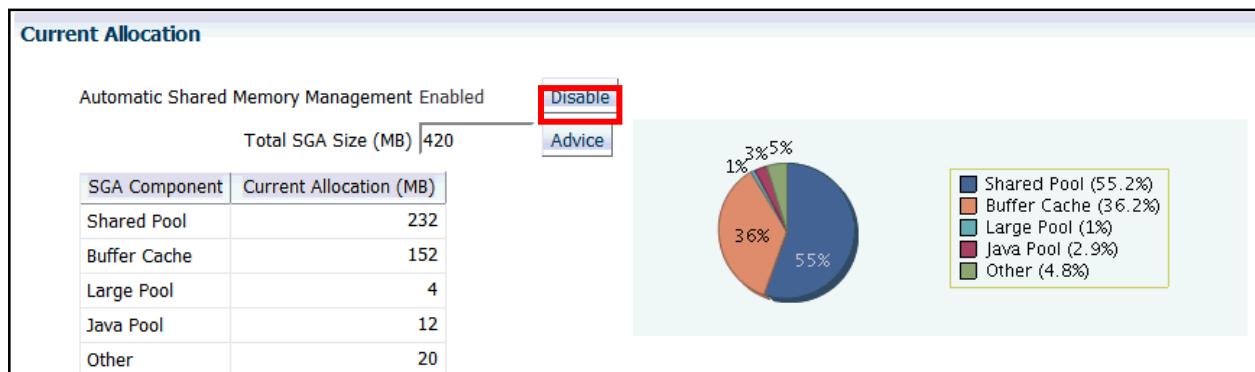
Practice 19-2: Adjusting Memory as Workloads Change

On the basis of the recommendation of the ADDM report in the previous scenario, the SGA memory constraint was raised to 420 MB. You have been using an allocation of 280 MB of memory for the PGA work areas. The current application has seen an increase in activity, and now another application is being added to the database. With it comes an additional allocation of instance memory. Make use of Automatic Memory Management to allocate this additional memory for best performance.

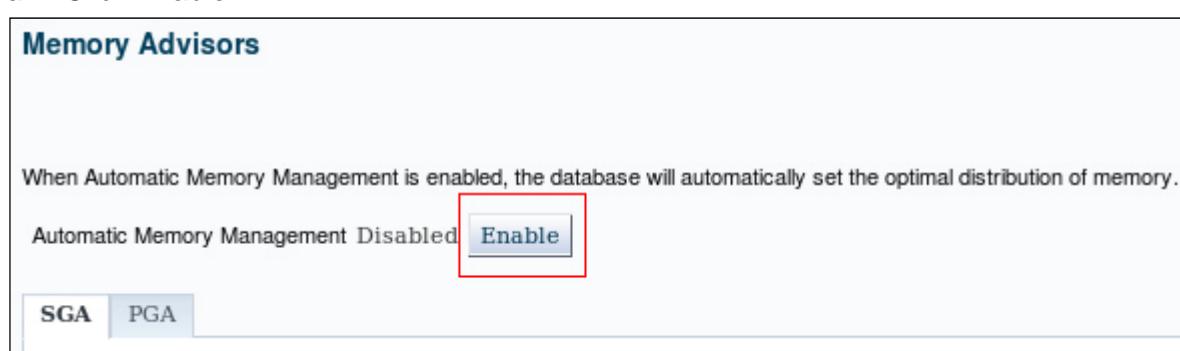
1. You are given an absolute maximum of 808 MB of instance memory and are instructed to use as little as required to get reasonable performance. Set the maximum memory size to 808 MB. Enable Automatic Memory Management (AMM). Set the Memory Target to 550 MB.

Note: By setting the maximum memory size value larger than needed during testing, you can find the optimal value, and then lower the setting. To set the parameters to enable AMM, you must disable ASMM and remove the `SGA_TARGET` value.

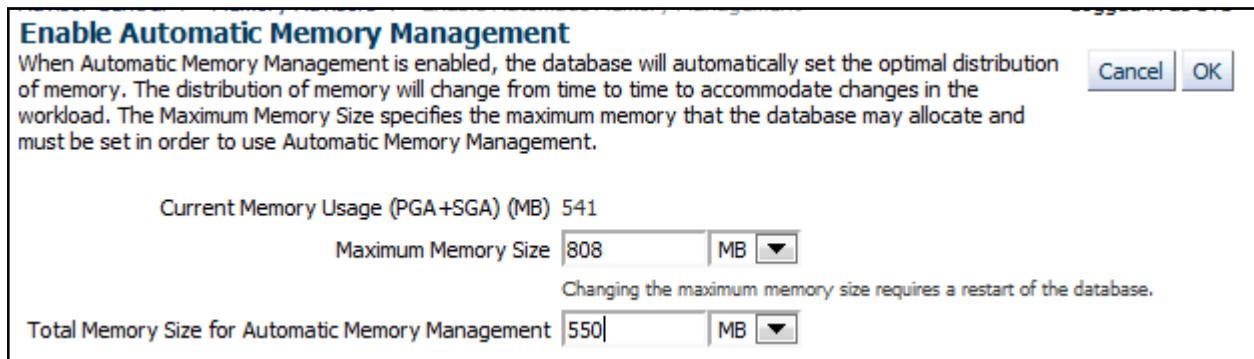
- a. On the **Memory Advisors** page in the Current Allocation section, disable Automatic Shared Memory Management by clicking **Disable**.



- b. On the Disable Automatic Shared Memory Management page, change the **Buffer Cache to 16 MB**, so that the value will not impose a minimum.
 - c. Click **Calculate**; the Total SGA value should change.
 - d. Click **OK**.
 - e. On the Memory Advisors page, a Confirmation message appears: "Automatic Shared Memory Management has been successfully disabled. The SGA component sizes will remain fixed as specified below."
2. On the Memory Advisors page, enable Automatic Memory Management.
 - a. Click **Enable**.



- b. On the Enable Automatic Memory Management page, set the **Maximum Memory Size** to **808 MB** and the **Total Memory Size for Automatic Memory Management** to **550 MB**. Click **OK**. Click **Refresh** to see the new allocation values.



- c. On the Memory Advisors page, review the new allocation for the SGA components.

Total SGA Size:

Buffer Cache:

Shared Pool:

- d. Click the PGA tab. What is the value of the current PGA allocated memory?

3. Start the workload generator by entering: `./workgen`. This workload adds a new application that uses a significant amount of PGA. And the existing workload has increased.

```
$ ./workgen AMM
```

4. Observe the **Memory Advisor graphs** for about three minutes. Be sure to click Refresh.

- a. Did the memory settings change?

Note: These values change during the course of the workload.

- b. What is the value of the buffer cache?

- c. What is the value of the shared pool?

- d. What is the value of the PGA? (Current PGA Allocated (KB) changes.)

5. On the Memory Advisors page, change the **Total Memory Size** to **660 MB** and click **Apply**.

6. Observe the **Memory Advisor graphs** for about two or three minutes with regular refresh.

- a. Did the memory settings change?

Note: These values can change during the course of the workload.

- b. What is the value of the buffer cache?

- c. What is the value of the shared pool?

- d. What is the value of the PGA?

- e. Click Advice. View the Memory Size Advice graph. What would be a better value of Total Memory?

At this point, the graph may show that there is little or no advantage to increasing the total memory.

7. Stop the workload with `rm runload`.

```
$ rm runload
```

8. In this practice, the SGA changed very little. It may not have changed at all on some machines. The PGA, however, was being used by the new workload and the distribution of memory between the PGA and SGA was affected.

9. How much tuning was required to find an optimum memory size for this workload?
The tuning did not require a shutdown after the maximum allowed memory value was set.
The Advice graph gave a clear indication of the optimal size, so the number of iterations was reduced.
10. Clean up from this practice.
 - a. Execute the cleanup script.

```
$ ./cleanup AMM
```

- b. Log out of Enterprise Manager Cloud Control.

Practices for Lesson 20: Performance Tuning Summary

Chapter 20

Practices for Lesson 20

Practices Overview

There are no practices for this lesson.

Practices for Appendix A: Using Statspack

Chapter 21

Practice for Appendix A-1: Installing Statspack

The goal of this practice is to install the Statspack repository and packages in the `orcl` database. For this practice, install the Statspack repository in the `SYSAUX` tablespace. The Statspack repository can be installed in any permanent tablespace. Make sure that you use the `perfstat` password for the `PERFSTAT` user. The `sp*.sql` scripts are located in the `$ORACLE_HOME/rdbms/admin` directory.

1. To prepare the database for this practice, connect with / as `SYSDBA` to drop the Statspack repository. Use the `spddrop.sql` script located in the `$ORACLE_HOME/rdbms/admin` directory.

```
$ sqlplus / as sysdba

SQL> @?/rdbms/admin/spdrop.sql
...
View dropped.

Synonym dropped.

User dropped.

NOTE:
SPDUSR complete. Please check spdusr.lis for any errors.

SQL>
```

2. Still connected as `SYSDBA`, using SQL*Plus, execute the `spcreate.sql` script to create the Statspack repository and packages.

The script prompts you for the password of the owner of the Statspack object. The username is `perfstat`. Set the password to `perfstat`. The default tablespace for the Statspack objects is `SYSAUX`. Set the default temporary tablespace for the `perfstat` user to `TEMP`.

```
SQL> @?/rdbms/admin/spcreate.sql
...
choose the PERFSTAT user's password
-----
Not specifying a password will result in the installation FAILING

Enter value for perfstat_password: perfstat
perfstat

Choose the Default tablespace for the PERFSTAT user
-----
Below is the list of online tablespaces in this database which can
store user data. Specifying the SYSTEM tablespace for the user's
default tablespace will result in the installation FAILING, as
using SYSTEM for performance data is not supported.
```

Choose the PERFSTAT user's default tablespace. This is the tablespace in which the STATSPACK tables and indexes will be created.

TABLESPACE_NAME	CONTENTS	STATSPACK DEFAULT TABLESPACE
<hr/>		
-		
EXAMPLE	PERMANENT	
LOADTEST1	PERMANENT	
LOADTEST10	PERMANENT	
LOADTEST2	PERMANENT	
LOADTEST3	PERMANENT	
LOADTEST4	PERMANENT	
LOADTEST5	PERMANENT	
LOADTEST6	PERMANENT	
LOADTEST7	PERMANENT	
LOADTEST8	PERMANENT	
LOADTEST9	PERMANENT	
<hr/>		
TABLESPACE_NAME	CONTENTS	STATSPACK DEFAULT TABLESPACE
<hr/>		
-		
SOE	PERMANENT	
SOEINDEX	PERMANENT	
SYSAUX	PERMANENT *	
USERS	PERMANENT	

Pressing <return> will result in STATSPACK's recommended default tablespace (identified by *) being used.

Enter value for default_tablespace:

Using tablespace SYSAUX as PERFSTAT default tablespace.

Choose the Temporary tablespace for the PERFSTAT user

Below is the list of online tablespaces in this database which can store temporary data (e.g. for sort workareas). Specifying the SYSTEM tablespace for the user's temporary tablespace will result in the installation FAILING, as using SYSTEM for workareas is not supported.

Choose the PERFSTAT user's Temporary tablespace.

TABLESPACE_NAME	CONTENTS	DB DEFAULT TEMP TABLESPACE
<hr/>		
TEMP	TEMPORARY	
TEMP_L	TEMPORARY *	

Pressing <return> will result in the database's default Temporary tablespace (identified by *) being used.

```
Enter value for temporary_tablespace: temp

...
Creating Package STATSPACK...

Package created.

No errors.
Creating Package Body STATSPACK...

Package body created.

No errors.

NOTE:
SPCPKG complete. Please check spcpkg.lis for any errors.

SQL>
```

3. Make sure that you did not have errors while installing Statspack. Check the log files that were generated during installation. The following is a command-line solution:

```
SQL> host ls *.lis
spcpkg.lis  spctab.lis  spcusr.lis  spdtab.lis  spdusr.lis

SQL> host grep error *.lis
spcpkg.lis:No errors.
spcpkg.lis:No errors.
spcpkg.lis:SPCPKG complete. Please check spcpkg.lis for any errors.
spctab.lis:SPCTAB complete. Please check spctab.lis for any errors.
spcusr.lis:SPCUSR complete. Please check spcusr.lis for any errors.
spdtab.lis:SPDTAB complete. Please check spdtab.lis for any errors.
spdusr.lis:SPDUSR complete. Please check spdusr.lis for any errors.

SQL>
SQL> exit
```

Practice for Appendix A-2: Creating Snapshots

The goal of this practice is to run a workload on the `orcl` database and to capture Statspack snapshots for that workload.

1. Before you capture your first Statspack snapshot, execute the `lab_A_02_01.sh` script located in your `$HOME/labs` directory. This script creates a tablespace and a user as the setup for the rest of the practice.

```
$ cd $HOME/labs

$ ./lab_A_02_01.sh
...
SQL> Connected.
SQL> SQL> SQL> SQL> drop tablespace tbsspc including contents and
datafiles
*
ERROR at line 1:
ORA-00959: tablespace 'TBSSPC' does not exist

SQL> SQL> 2      3      4      5
Tablespace created.

SQL> SQL>
PL/SQL procedure successfully completed.

SQL> SQL> drop user spc cascade
*
ERROR at line 1:
ORA-01918: user 'SPC' does not exist

SQL> SQL> 2      3
User created.

SQL> SQL>
Grant succeeded.

SQL> SQL> Connected.
SQL> SQL> drop table spct purge
*
ERROR at line 1:
ORA-00942: table or view does not exist

SQL>
Table created.

SQL> SQL> > >
PL/SQL procedure successfully completed.

SQL> SQL> Disconnected ...
$
```

2. Create a level 7 Statspack snapshot. Make sure that you retrieve the corresponding snapshot ID. Set the `i_snap_level` parameter of the `statspack.snap` procedure to 7.

```
$ sqlplus /nolog
SQL> CONNECT perfstat/perfstat

SQL> variable snap number;
SQL> begin
 2 :snap := statspack.snap(i_snap_level=>7);
 3 end;
 4 /

PL/SQL procedure successfully completed.

SQL> print snap

      SNAP
-----
      1

SQL> exit
```

3. Execute the `start_A_02_03.sh` script from the `labs` directory. This script starts the workload on your `orcl` database. The script takes about one minute to execute.

```
$ cd $HOME/labs
$ ./start_A_02_03.sh
PL/SQL procedure successfully completed.

PL/SQL procedure successfully completed.

PL/SQL procedure successfully completed.

...

PL/SQL procedure successfully completed.

PL/SQL procedure successfully completed.

PL/SQL procedure successfully completed.
$
```

4. After the workload finishes running, capture a new level 7 Statspack snapshot. Make sure that you retrieve the corresponding snapshot ID. The number is system assigned and is not controllable. The snap numbers are not necessarily consecutive.

```
$ cd $HOME/labs
$ sqlplus perfstat/perfstat
...
SQL> variable snap number;
SQL> begin
 2      :snap := statspack.snap(i_snap_level=>7);
 3  end;
 4 /
PL/SQL procedure successfully completed.

SQL> print snap
      SNAP
-----
      2

SQL> exit
...
```

Practice for Appendix A-3: Generating Statspack Reports

The goal of this practice is to generate a Statspack report, examine it, and fix any issues that you discover while interpreting the Statspack report.

1. Generate a Statspack report between the two previously captured snapshots. Use the `spreport.sql` script. Be sure to use the snap numbers that you found in the previous practice. When prompted for a file name, use `sp_1_2.lst`.

```
$ cd $HOME/labs
$ sqlplus perfstat/perfstat
...
SQL> set echo on
SQL> @?/rdbms/admin/spreport
...
Listing all Completed Snapshots

Instance      DB Name      Snap Id      Snap Started      Snap Level Comment
-----        -----        -----        -----
orcl          ORCL          1 16 Mar 2010 09:54      7
                           2 16 Mar 2010 09:57      7

Specify the Begin and End Snapshot Ids
~~~~~
Enter value for begin_snap: 1
Begin Snapshot Id specified: 1

Enter value for end_snap: 2
End   Snapshot Id specified: 2

Specify the Report Name
~~~~~
The default report file name is sp_1_2. To use this name, press <return> to continue, otherwise enter an alternative.

Enter value for report_name: sp_1_2.lst
...
sga_max_size           419430400
sga_target              0
shared_pool_size         0
spfile
/u01/app/oracle/product/11.2.0/dbhome_1/dbs/spfileorcl.ora
streams_pool_size       4194304
undo_tablespace         UNDOTBS1
-----
End of Report ( sp_1_2.lst )
```

```
SQL>exit;
```

2. Examine the report that you created. You can use the less utility, which uses vi navigation commands, or the gedit graphic text viewer. What are your conclusions?
 - a. Invoke a viewer. (gedit is shown.) Navigation in gedit is similar to other graphic-based editors. To find a section of the report, use Search > Find and the report section name.

```
$ cd $HOME/labs
$ gedit sp_1_2.lst
```

- b. Scroll to the Top 5 Timed Events section. "Buffer busy waits" is among Top 5 Timed Events.
 - c. View Time Model System Stats. The top entry is "sql execute elapsed time."
 - d. Looking at the Buffer Wait Statistics section, you can see that most of the waits are due to the data block category.
 - e. Looking at the "Segments by Buffer Busy Waits" section, SPCT is shown as the segment in the TBSSPC tablespace that is collecting the waits.
 - f. All this leads you to think that there is an issue with the SPCT table. Looking at the tablespace definition, you discover that the table was defined without Automatic Segment Space Management.

```
$ sqlplus spc/spc
...
SQL> SELECT segment_space_management
  2  FROM dba tablespaces
  3  WHERE tablespace_name=(select tablespace_name
                           from user_tables);

SEGMENT
-----
MANUAL

SQL> exit;
```

3. Fix the problem by implementing the recommendations from the previous step. You decide to re-create the table by using Automatic Segment Space Management. This script is available as lab_A_03_03.sql in the labs directory.

```
$ sqlplus / as sysdba
...
SQL> @$LABS/lab_A_03_03
SQL>
SQL> drop tablespace tbsspc including contents and datafiles;
Tablespace dropped.

SQL>
SQL> CREATE SMALLFILE TABLESPACE "TBSSPC"
  2  DATAFILE 'tbsspc1.dbf' SIZE 50M
```

```

3 LOGGING
4 EXTENT MANAGEMENT LOCAL
5 SEGMENT SPACE MANAGEMENT AUTO;

Tablespace created.

SQL>
SQL> connect spc/spc
Connected.
SQL>
SQL> create table spct(id number, name varchar2(2000)) tablespace
tbsspc;

Table created.

SQL>
SQL> exec DBMS_STATS.GATHER_TABLE_STATS(
-> ownname=>'SPC', tabname=>'SPCT', -
-> estimate_percent=>DBMS_STATS.AUTO_SAMPLE_SIZE);

PL/SQL procedure successfully completed.

SQL>
SQL> exit;
...
$
```

4. Regenerate the same workload, and verify that the problem disappears.
 - a. Take a new snapshot like you did in the previous practice. Run the same workload again and take another snapshot. Using the \$HOME/labs/lab_A_03_04a.sh script, you can create a snapshot, run the workload, and take a second snapshot script. Note the snap ID values. The script takes two minutes to complete.

```
$ $HOME/labs/lab_A_03_04a.sh
```

- b. Generate the corresponding Statspack report by using the snap ID values generated in the previous step.

```
$ sqlplus perfstat/perfstat

SQL> @?/rdbms/admin/spreport

...
Listing all Completed Snapshots
```

Instance	DB Name	Snap Id	Snap Started	Snap Level	Comment
orcl	ORCL	1	16 Mar 2010 09:54	7	
		2	16 Mar 2010 09:57	7	
		3	16 Mar 2010 10:10	7	
		4	16 Mar 2010 10:12	7	

```
Specify the Begin and End Snapshot Ids
```

```
~~~~~  
Enter value for begin_snap: 3  
Begin Snapshot Id specified: 3  
  
Enter value for end_snap: 4  
End   Snapshot Id specified: 4
```

```
Specify the Report Name  
~~~~~
```

```
The default report file name is sp_3_4. To use this name,  
press <return> to continue, otherwise enter an alternative.
```

```
Enter value for report_name: sp_3_4.lst
```

- c. Look at the generated report by using your preferred text browser. You can see that the number of buffer busy waits has been reduced.

