Oracle® Database

2 Day + Performance Tuning Guide

10*g* Release 2 (10.2)

B28051-03

April 2010

Easy, Automatic, and Step-By-Step Performance Tuning Using Oracle Diagnostics Pack, Oracle Database Tuning Pack, and Oracle Enterprise Manager

ORACLE®

Oracle Database 2 Day + Performance Tuning Guide, 10*g* Release 2 (10.2)

B28051-03

Primary Author: Immanuel Chan

Contributors: Karl Dias, Cecilia Grant, Connie Green, Andrew Holdsworth, Sushil Kumar, Herve Lejeune, Colin McGregor, Mughees Minhas, Valarie Moore, Deborah Owens, Mark Townsend, Graham Wood

Contents

8 Resolving Performance Degradation Over Time

Preface

This preface contains the following topics:

Audience

This document is intended for Oracle database administrators (DBAs) who want to tune and optimize the performance of their Oracle Database. Before using this document, you should complete *Oracle Database 2 Day DBA*.

In particular, this document is targeted toward the following groups of users:

- Oracle DBAs who want to acquire database performance tuning skills
- DBAs who are new to Oracle Database

Documentation Accessibility

Our goal is to make Oracle products, services, and supporting documentation accessible to all users, including users that are disabled. To that end, our documentation includes features that make information available to users of assistive technology. This documentation is available in HTML format, and contains markup to facilitate access by the disabled community. Accessibility standards will continue to evolve over time, and Oracle is actively engaged with other market-leading technology vendors to address technical obstacles so that our documentation can be accessible to all of our customers. For more information, visit the Oracle Accessibility Program Web site at `http://www.oracle.com/accessibility/`.

Accessibility of Code Examples in Documentation

Screen readers may not always correctly read the code examples in this document. The conventions for writing code require that closing braces should appear on an otherwise empty line; however, some screen readers may not always read a line of text that consists solely of a bracket or brace.

Accessibility of Links to External Web Sites in Documentation

This documentation may contain links to Web sites of other companies or organizations that Oracle does not own or control. Oracle neither evaluates nor makes any representations regarding the accessibility of these Web sites.

Access to Oracle Support

Oracle customers have access to electronic support through My Oracle Support. For information, visit `http://www.oracle.com/support/contact.html` or visit `http://www.oracle.com/accessibility/support.html` if you are hearing impaired.

Related Documents

For more information about the topics covered in this document, see the following documents:

- *Oracle Database 2 Day DBA*
- *Oracle Database Administrator's Guide*
- *Oracle Database Concepts*
- *Oracle Database Performance Tuning Guide*

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

Part I

Getting Started

Part I provides an introduction to this document and explains the Oracle Database performance method. This part contains the following chapters:

1

Introduction

As an Oracle database administrator (DBA), you are responsible for the performance of your Oracle database. Tuning a database to reach a desirable performance level may be a daunting task, especially for DBAs who are new to Oracle Database. *Oracle Database 2 Day + Performance Tuning Guide* is a quick start guide that teaches you how to perform day-to-day database performance tuning tasks using features provided by the Oracle Diagnostics Pack, Oracle Tuning Pack, and Oracle Enterprise Manager.

This chapter contains the following sections:

About This Guide

Before using this guide, you need to:

- Complete the *Oracle Database 2 Day DBA*
- Obtain the necessary products and tools described in "Tools for Tuning the Database" on page 1-2

Oracle Database 2 Day + Performance Tuning Guide is task oriented. The objective is to describe why and when tuning tasks need to be performed.

This guide is not an exhaustive discussion of all Oracle Database concepts. For this type of information, see *Oracle Database Concepts*.

This guide does not describe basic Oracle database administrative tasks. For this type of information, see *Oracle Database 2 Day DBA*. For a complete discussion of administrative tasks, see *Oracle Database Administrator's Guide.*

The primary interface used in this guide is Oracle Enterprise Manager Database Control console. This guide is not an exhaustive discussion of all Oracle database performance tuning features and does not cover available APIs that provide comparable tuning options to those presented in this guide. For this type of information, see *Oracle Database Performance Tuning Guide*.

Common Oracle DBA Tasks

As an Oracle DBA, you can expect to be involved in the following tasks:

- Installing Oracle software
- Creating an Oracle database

- Upgrading the database and software to new releases
- Starting up and shutting down the database
- Managing the storage structures of the database
- Managing users and security
- Managing schema objects, such as tables, indexes, and views
- Making database backups and performing database recovery, when necessary
- Proactively monitoring the condition of the database and taking preventive or corrective actions, as required
- Monitoring and tuning database performance

In a small-to-midsize database environment, you might be the sole person performing these tasks. In large, enterprise environments, the job is often divided among several DBAs—each with their own specialty—such as database security or database tuning. *Oracle Database 2 Day + Performance Tuning Guide* describes how to accomplish the last two tasks in this list.

Tools for Tuning the Database

The intent of this guide is to allow you to quickly and efficiently tune and optimize the performance of Oracle Database.

To achieve the goals of this guide, you will need to acquire the following products, tools, and utilities:

- **Oracle Database 10*g* Enterprise Edition**

  Oracle Database 10*g* Enterprise Edition offers enterprise-class performance, scalability and reliability on clustered and single-server configurations. It includes many performance features that are used in this guide.

- **Oracle Enterprise Manager**

  The primary tool to manage your database is Oracle Enterprise Manager, a Web-based interface. After you install the Oracle software, create or upgrade a database, and configure the network, you can use Oracle Enterprise Manager to manage your database. In addition, Oracle Enterprise Manager provides an interface for performance advisors and for Oracle utilities, such as SQL*Loader and Recovery Manager.

- **Oracle Diagnostics Pack**

  Oracle Diagnostics Pack offers a complete, cost-effective, and easy-to-use solution to manage the performance of Oracle Database environments by providing unique features, such as automatic identification of performance bottlenecks, guided problem resolution, and comprehensive system monitoring. Key features of the Oracle Diagnostics Pack that are used in this guide include the Automatic Database Diagnostics Monitor (ADDM) and the Automatic Workload Repository (AWR).

- **Oracle Database Tuning Pack**

  Oracle Database Tuning Pack automates the entire database application tuning process, thereby significantly lowering database management costs while enhancing performance and reliability. Key features of the Oracle Database Tuning Pack that are used in this guide include the SQL Tuning Advisor and the SQL Access Advisor.

Note: Oracle Diagnostics Pack and Oracle Database Tuning Pack require separate licenses. For more information, see *Oracle Database Licensing Information*.

2

Oracle Database Performance Method

Performance improvement is an iterative process. Removing the first bottleneck might not lead to performance improvement immediately, because another bottleneck might be revealed that has an even greater performance impact on the system. For this reason, the Oracle performance method is iterative. Accurately diagnosing the performance problem is the first step towards ensuring that the changes you make to the system will result in improved performance.

Performance problems generally result from a lack of throughput, unacceptable user or job response time, or both. The problem might be localized to specific application modules, or it might span the entire system. Before looking at any database or operating system statistics, it is crucial to get feedback from the most important components of the system: the users of the system and the people ultimately paying for the application. Getting feedback from users makes determining the performance goal easier, and improved performance can be measured in terms of real business goals, rather than system statistics.

The Oracle performance method can be applied until performance goals are met or deemed impractical. This process is iterative, and it is likely that some investigations will be made that have little impact on the performance of the system. It takes time and experience to accurately pinpoint critical bottlenecks in a timely manner. The Automatic Database Diagnostic Monitor (ADDM) implements the Oracle performance method and analyzes statistics to provide automatic diagnosis of major performance problems. Using ADDM can significantly shorten the time required to improve the performance of a system, and it is the method used in this guide.

This chapter discusses the Oracle Database performance method and contains the following sections:

Gathering Database Statistics Using the Automatic Workload Repository

Database statistics provide information about the type of load on the database and the internal and external resources used by the database. To accurately diagnose performance problems with the database using ADDM, statistics must be available.

Oracle Database generates many types of cumulative statistics for the system, sessions, and individual SQL statements. Oracle Database also tracks cumulative statistics about segments and services. The Automatic Workload Repository (AWR) automates database statistics gathering by collecting, processing, and maintaining performance statistics for database problem detection and self-tuning purposes. By default, this

process is repeated every hour, and the result is called an AWR snapshot. The delta values captured by the snapshot represent the changes for each statistic over the time period. Statistics gathered by the AWR are queried from memory. The gathered data can be displayed in both reports and views.

Gathering database statistics using the AWR is enabled by default and is controlled by the `STATISTICS_LEVEL` initialization parameter. The `STATISTICS_LEVEL` parameter should be set to `TYPICAL` or `ALL` to enable statistics gathering by the AWR. The default setting is `TYPICAL`. Setting the `STATISTICS_LEVEL` parameter to `BASIC` disables many Oracle Database features, including the AWR, and is not recommended. For information about the `STATISTICS_LEVEL` initialization parameter, see *Oracle Database Reference*.

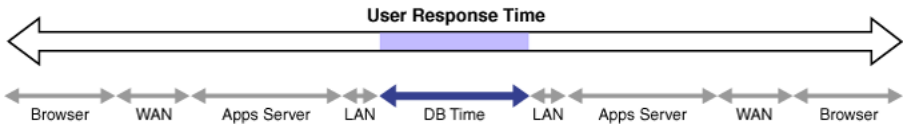
The database statistics collected and processed by AWR include:

- Time Model Statistics
- Wait Event Statistics
- Session and System Statistics
- Active Session History Statistics
- High-Load SQL Statistics

Time Model Statistics

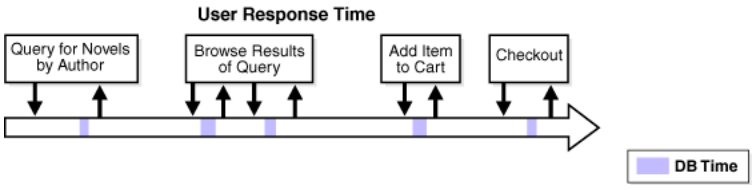
Time model statistics are statistics that measure the time spent in the database by operation type. The most important time model statistic is database time, or DB time. Database time represents the total time spent in database calls, and is an indicator of the total instance workload. As shown in Figure 2–1, database time makes up a portion of an application's overall user response time.

Figure 2–1 DB Time in Overall User Response Time



Database time is calculated by aggregating the CPU time and wait time of all user sessions not waiting for idle wait events (nonidle user sessions). For example, a user session may involve an online transaction made at a bookseller's Web site consisting of the following actions, as shown in Figure 2–2:

Figure 2–2 DB Time in User Transaction



1. Query for novels by author

   The user performs a search for novels by a particular author. This causes the application to perform a database query for novels by the author.

2. Browse results of query

   The user browses the list of novels by the author that are returned and accesses additional details, such as user reviews and inventory status, about the novels. This causes the application to perform additional database queries.

3. Add item to cart

   After browsing details on the novels, the user decides to add one of the novels to the shopping cart. This causes the application to make a database call to update the shopping cart.

4. Checkout

   The user completes the transaction by checking out using the address and payment information previously saved at the bookseller's Web site from a previous purchase. This causes the application to perform various database operations to retrieve the user's information, add a new order, update the inventory, and generate an E-mail confirmation.

For each of these actions, the user makes a request to the database, as represented by the down arrow in Figure 2–2 on page 2-2. The CPU time spent by the database processing the request and the wait time spent waiting for the database are considered DB time, as represented by the shaded areas. Once the request is completed, the results are returned to the user, as represented by the up arrow. The space between the up and down arrows represents the total user response time for processing the request, which contains other components besides DB time, as illustrated in Figure 2–1 on page 2-2.

The objective of database tuning is to reduce the time that users spend performing actions on the database, or reducing database time. By doing so, the overall response time of user transactions on the application can be improved.

Wait Event Statistics

Wait events are statistics that are incremented by a session to indicate that it had to wait for an event to complete before being able to continue processing. When a session has to wait while processing a user request, the AWR records the wait using one of a set of predefined wait events that are then grouped into wait classes. Wait event data reveals various symptoms of problems that might be impacting performance, such as latch contention, buffer contention, and I/O contention.

See Also:

- *Oracle Database Performance Tuning Guide*
- *Oracle Database Reference*.

Session and System Statistics

A large number of cumulative database statistics are available on a system and session level. Some of these statistics are collected by the AWR.

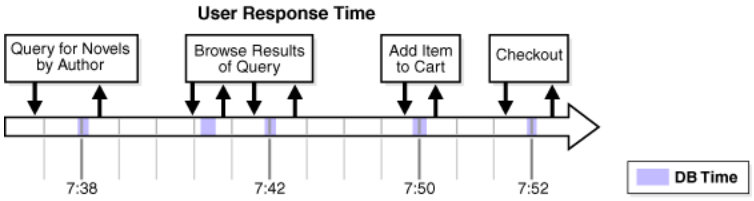
Active Session History Statistics

The Active Session History (ASH) statistics are samples of session activity in the database. Active sessions are sampled every second, and are stored in a circular buffer in the system global area (SGA). Any session that is connected to the database and using CPU, or is waiting for an event that does not belong to the idle wait class, is considered an active session. By capturing only active sessions, a manageable set of

data is represented with the size being directly related to the work being performed, rather than the number of sessions allowed on the system.

Using the DB time example described in "Time Model Statistics" on page 2-2, samples of session activity are collected from the online transaction made at the bookseller's Web site, represented as vertical lines below the horizontal arrow in Figure 2–3.

Figure 2–3 Active Session History



The light vertical lines represent samples of inactive session activity that are not captured in the ASH statistics. The bold vertical lines represent samples of active sessions that are captured at:

- 7:38, while novels by the author are being queried
- 7:42, while the user is browsing the query results
- 7:50, when one of the novels is added to the shopping cart
- 7:52, during the checkout process

Table 2–1 lists the ASH statistics that are collected for the active sessions, along with examples of the session ID, module, SQL ID, session state, and wait events that are sampled.

Table 2–1 Active Session History

Time	SID	Module	SQL ID	State	Event
7:38	213	Book by author	qa324jffritcf	Waiting	db file sequential read
7:42	213	Get review ID	aferv5desfzs5	CPU	
7:50	213	Add item to cart	hk32pekfcbdfr	Waiting	buffer busy wait
7:52	213	Checkout	abngldf95f4de	Waiting	log file sync

High-Load SQL Statistics

SQL statements that are consuming the most resources produce the highest load on the system, based on criteria such as elapsed time and CPU time.

Using the Oracle Performance Method

Performance tuning using the Oracle performance method is driven by identifying and eliminating bottlenecks in the database, and by developing efficient SQL statements. Database tuning is performed in two phases: proactively and reactively.

In the proactive tuning phase, you need to perform tuning tasks as part of your daily database maintenance routine, such as reviewing ADDM analysis and findings, monitoring the real-time performance of the database, and responding to alerts.

In the reactive tuning phase, you need to respond to issues reported by the users, such as performance problems that may occur for only a short duration of time, or performance degradation to the database over time.

SQL tuning is an iterative process to identify, tune, and improve the efficiency of high-load SQL statements.

Applying the Oracle performance method involves:

- Performing pre-tuning preparations, as described in "Preparing the Database for Tuning" on page 2-5
- Tuning the database proactively on a regular basis, as described in "Tuning the Database Proactively" on page 2-5
- Tuning the database reactively when performance problems are reported by the users, as described in "Tuning the Database Reactively" on page 2-6
- Identifying, tuning, and optimizing high-load SQL statements, as described in "Tuning SQL Statements" on page 2-7

To improve the performance of your database, you will need to apply these principles iteratively.

Preparing the Database for Tuning

This section lists and describes the steps that need to be performed before the database can be properly tuned.

To prepare the database for tuning:

1. Get feedback from users.

   Determine the scope of the performance project and subsequent performance goals, and determine performance goals for the future. This process is key for future capacity planning.

2. Sanity-check the operating systems of all systems involved with user performance.

   Check for hardware or operating system resources that are fully utilized. List any overused resources as possible symptoms for later analysis. In addition, ensure that all hardware is functioning properly.

3. Ensure that the `STATISTICS_LEVEL` initialization parameter is set to `TYPICAL` or `ALL` to enable the automatic performance tuning features of Oracle Database, including the AWR and ADDM.

   The default setting for this parameter is `TYPICAL`.

   See Also:

   - "Gathering Database Statistics Using the Automatic Workload Repository" on page 2-1 for information about configuring the AWR
   - "Configuring the Automatic Database Diagnostics Monitor" on page 3-2 for information about configuring ADDM

Tuning the Database Proactively

This section lists and describes the steps required to keep the database properly tuned on a regular basis. These tuning procedures are considered proactive and should be performed as part of your daily maintenance of Oracle Database.

To tune the database proactively:

1. Review the ADDM findings, as described in Chapter 3, "Automatic Database Performance Monitoring".

   ADDM automatically detects and reports on performance problems with the database, including most of the "Common Performance Problems Found in Oracle Databases" on page 2-7. The results are displayed as ADDM findings on the Database Home page in Enterprise Manager. Reviewing these findings enables you to quickly identify the performance problems that require your attention.

2. Implement the ADDM recommendations, as described in Chapter 3, "Automatic Database Performance Monitoring".

   ADDM automatically provides a list of recommendations for reducing the impact of the performance problem with each ADDM finding. Implementing a recommendation applies the suggested changes to improve the database performance.

3. Monitor performance problems with the database in real time, as described in Chapter 4, "Monitoring Real-Time Database Performance".

   The Database Performance page in Oracle Enterprise Manager enables you to identify and respond to real-time performance problems. By drilling down to the appropriate pages, you can identify and resolve performance problems with the database in real time, without having to wait until the next ADDM analysis.

4. Respond to performance-related alerts, as described in Chapter 5, "Monitoring Performance Alerts".

   The Database Home page in Oracle Enterprise Manager enables you to view performance-related alerts generated by the system. Typically, these alerts reveal performance problems that, once resolved, will improve the performance of your database.

5. Validate that the changes made have produced the desired effect, and verify if the perception of performance to the users has improved.

6. Repeat these steps until your performance goals are met or become impossible to achieve due to other constraints.

Tuning the Database Reactively

This section lists and describes the steps required to tune the database based on user feedback. These tuning procedure are considered reactive and should be performed periodically when performance problems are reported by the users.

To tune the database reactively:

1. Run ADDM manually to diagnose current and historical database performance when performance problems are reported by the users, as described in Chapter 6, "Manual Database Performance Monitoring".

   This is useful if you want to run ADDM before the next ADDM analysis to analyze current database performance, or to analyze historical database performance when you were not proactively monitoring the system.

2. Resolve transient performance problems, as described in Chapter 7, "Resolving Transient Performance Problems".

   The Active Session History (ASH) reports enable you to analyze transient performance problems with the database that are short-lived and do not appear in the ADDM analysis.

3. Resolve performance degradation over time, as described in Chapter 8, "Resolving Performance Degradation Over Time".

   The Automatic Workload Repository (AWR) Compare Periods reports enable you to compare database performance between two periods of time, and resolve performance degradation that may happen from one time period to another.

4. Validate that the changes made have produced the desired effect, and verify if the perception of performance to the users has improved.

5. Repeat these steps until your performance goals are met or become impossible to achieve due to other constraints.

Tuning SQL Statements

This section lists and describes the steps required to identify, tune, and optimize high-load SQL statements.

To tune SQL statements:

1. Identify high-load SQL statements, as described in Chapter 9, "Identifying High-Load SQL Statements".

   Use the ADDM findings and the Top SQL to identify high-load SQL statements that are causing the greatest contention.

2. Tune high-load SQL statements, as described in Chapter 10, "Tuning SQL Statements".

   You can improve the efficiency of high-load SQL statements by tuning them using the SQL Tuning Advisor.

3. Optimize data access paths, as described in Chapter 11, "Optimizing Data Access Paths".

   You can optimize the performance of data access paths by creating the proper set of materialized views, materialized view logs, and indexes for a given workload by using the SQL Access Advisor.

4. Repeat these steps until all high-load SQL statements are tuned for greatest efficiency.

Common Performance Problems Found in Oracle Databases

This section lists and describes common performance problems found in Oracle databases. By following the Oracle performance method outlined in this chapter, you should be able to avoid these problems. If you have these problems, then repeat the steps in the Oracle performance method, as described in "Using the Oracle Performance Method" on page 2-4, or consult the chapter or section that addresses these problems, as described in this section.

- CPU bottlenecks

  Is the application performing poorly because the system is CPU bound? Performance problems caused by CPU bottlenecks are diagnosed by ADDM automatically, as described in Chapter 3, "Automatic Database Performance Monitoring". You can also identify CPU bottlenecks by using the Database Performance page in Oracle Enterprise Manager, as described in "Monitoring CPU Utilization" on page 4-9.

- Undersized memory structures

Are the Oracle memory structures—such as the SGA, PGA, and buffer cache—adequately sized? Performance problems caused by undersized memory structures are diagnosed by ADDM automatically, as described in Chapter 3, "Automatic Database Performance Monitoring". You can also identify memory usage issues by using the Database Performance page in Oracle Enterprise Manager, as described in "Monitoring Memory Utilization" on page 4-11.

- I/O capacity issues

  Is the I/O subsystem performing as expected? Performance problems caused by I/O capacity issues are diagnosed by ADDM automatically, as described in Chapter 3, "Automatic Database Performance Monitoring". You can also identify disk I/O issues by using the Database Performance page in Oracle Enterprise Manager, as described in "Monitoring Disk I/O Utilization" on page 4-13.

- Suboptimal use of Oracle Database by the application

  Is the application making suboptimal use of Oracle Database? Problems such as establishing new database connection repeatedly, excessive SQL parsing, and high level of contention for a small amount of data (also known as application-level block contention) can degrade the application performance significantly. Performance problems caused by suboptimal use of Oracle Database by the application are diagnosed by ADDM automatically, as described in Chapter 3, "Automatic Database Performance Monitoring". You can also monitor top activity in various dimensions—including SQL, session, services, modules, and actions—by using the Database Performance page in Oracle Enterprise Manager, as described in "Monitoring User Activity" on page 4-2.

- Concurrency issues

  Is the database performing suboptimally due to a high degree of concurrent activities in the database? A high degree of concurrent activities might result in contention for shared resources that can manifest in the forms of locks or waits for buffer cache. Performance problems caused by concurrency issues are diagnosed by ADDM automatically, as described in Chapter 3, "Automatic Database Performance Monitoring". You can also identify concurrency issues by using Top Sessions in Oracle Enterprise Manager, as described in "Monitoring Top Sessions" on page 4-4.

- Database configuration issues

  Is the database configured optimally to provide desired performance levels. For example, is there evidence of incorrect sizing of log files, archiving issues, excessive checkpoints, or suboptimal parameter settings? Performance problems caused by database configuration issues are diagnosed by ADDM automatically, as described in Chapter 3, "Automatic Database Performance Monitoring".

- Short-lived performance problems

  Are your users complaining about short-lived or intermittent performance problems? Depending on the interval between snapshots taken by the AWR, performance problems that have a short duration may not be captured by ADDM. You can identify short-lived performance problems by using the Active Session History report, as described in Chapter 7, "Resolving Transient Performance Problems".

- Degradation of database performance over time

  Is there evidence that the database performance has degraded over time? For example, are you or your users noticing that the database is not performing as well as it used to 6 months ago? You can generate an Automatic Workload Repository

Compare Periods report to compare the period when the performance was poor to a period when the performance is stable to identify configuration settings, workload profile, and statistics that are different between these two time periods. This will help you identify the cause of the performance degradation, as described in Chapter 8, "Resolving Performance Degradation Over Time".

- Inefficient or high-load SQL statements

  Are there any SQL statements that are using excessive system resources that impact the system? Performance problems caused by high-load SQL statements are diagnosed by ADDM automatically, as described in Chapter 3, "Automatic Database Performance Monitoring" and "Identifying High-Load SQL Statements Using ADDM Findings" on page 9-1. You can also identify high-load SQL statements by using Top SQL in Oracle Enterprise Manager, as described in "Identifying High-Load SQL Statements Using Top SQL" on page 9-2. After they have been identified, you can tune the high-load SQL statements using the SQL Tuning Advisor, as described in Chapter 10, "Tuning SQL Statements".

- Data access paths to hot objects

  Are there database objects that are the source of bottlenecks because they are continuously accessed? Performance problems caused by hot objects are diagnosed by ADDM automatically, as described in Chapter 3, "Automatic Database Performance Monitoring". You can also optimize the data access path to these objects using the SQL Access Advisor, as described in Chapter 11, "Optimizing Data Access Paths" on page 4-13.

Part II

Proactive Database Tuning

Part II describes how to tune Oracle Database proactively on a regular basis and contains the following chapters:

- Chapter 3, "Automatic Database Performance Monitoring"
- Chapter 4, "Monitoring Real-Time Database Performance"
- Chapter 5, "Monitoring Performance Alerts"

3

Automatic Database Performance Monitoring

This chapter describes how to use the automatic diagnostic feature of the Automatic Database Diagnostic Monitor (ADDM) to monitor database performance. ADDM automatically detects and reports on performance problems with the database. The results are displayed as ADDM findings on the Database Home page in Enterprise Manager. Reviewing the ADDM findings enables you to quickly identify the performance problems that require your attention. Each ADDM finding also provides a list of recommendations for reducing the impact of the performance problem. Reviewing ADDM findings and implementing the recommendations are tasks that you should perform daily as part of the regular database maintenance. Even when the database is operating at an optimal performance level, you should continue to use the ADDM to monitor database performance on an ongoing basis.

This chapter contains the following sections:

Overview of the Automatic Database Diagnostic Monitor

The Automatic Database Diagnostic Monitor (ADDM) is a self-diagnostic engine built into Oracle Database. ADDM examines and analyzes data captured in the Automatic Workload Repository (AWR) to determine possible performance problems in Oracle Database. ADDM then locates the root causes of the performance problems, provides recommendations for correcting them, and quantifies the expected benefits. ADDM also identifies areas of the database for informational purposes where no action is necessary.

An ADDM analysis is performed after each AWR snapshot (every hour by default), and the results are saved in the database, which can then be viewed using Oracle Enterprise Manager. Before using another performance tuning method presented in this guide, you should first review the results of the ADDM analysis.

The ADDM analysis is performed from the top down, first identifying symptoms and then refining the analysis to reach the root causes of performance problems. ADDM uses the DB time statistic to identify performance problems. DB time is the cumulative time spent by the database in processing user requests, including both wait time and CPU time of all user sessions that are not idle. The goal of tuning the performance of a

database is to reduce the DB time of the system for a given workload. By reducing DB time, the database is able to support more user requests using the same resources. System resources that are using a significant portion of DB time are reported as problem areas by ADDM, and they are sorted in descending order by the amount of related DB time spent. For more information about the DB time statistic, see "Time Model Statistics" on page 2-2.

In addition to diagnosing performance problems, ADDM recommends possible solutions. When appropriate, ADDM recommends multiple solutions from which you can choose. ADDM recommendations include:

- Hardware changes

  Adding CPUs or changing the I/O subsystem configuration

- Database configuration

  Changing initialization parameter settings

- Schema changes

  Hash partitioning a table or index, or using automatic segment-space management (ASSM)

- Application changes

  Using the cache option for sequences or using bind variables

- Using other advisors

  Running the SQL Tuning Advisor on high-load SQL statements or running the Segment Advisor on hot objects

ADDM benefits apply beyond production systems; even on development and test systems, ADDM can provide an early warning of potential performance problems.

It is important to realize that performance tuning is an iterative process, and fixing one problem can cause a bottleneck to shift to another part of the system. Even with the benefit of the ADDM analysis, it can take multiple tuning cycles to reach a desirable level of performance.

Configuring the Automatic Database Diagnostics Monitor

This section describes how to configure ADDM and contains the following topics:

- Setting the STATISTICS_LEVEL parameter
- Setting the DBIO_EXPECTED parameter
- Managing Snapshots

Setting the STATISTICS_LEVEL parameter

ADDM is enabled by default and is controlled by the `STATISTICS_LEVEL` initialization parameter. The `STATISTICS_LEVEL` parameter should be set to `TYPICAL` or `ALL` to enable the automatic database diagnostic feature of ADDM. The default setting is `TYPICAL`. Setting the `STATISTICS_LEVEL` parameter to `BASIC` disables many Oracle Database features, including ADDM, and is not recommended.

See Also:

- *Oracle Database Reference* for information about the `STATISTICS_LEVEL` initialization parameter

Setting the DBIO_EXPECTED parameter

The ADDM analysis of I/O performance partially depends on a single argument, `DBIO_EXPECTED`, that describes the expected performance of the I/O subsystem. The value of the `DBIO_EXPECTED` argument is the average time it takes to read a single database block, in microseconds. Oracle Database uses the default value of 10 milliseconds, which is an appropriate value for most hard drives. If your hardware is significantly different, consider using a different value.

To determine the correct setting for the `DBIO_EXPECTED` parameter:

1. Measure the average read time of a single database block for your hardware.

   This measurement needs to be taken for random I/O, which includes seek time if you use standard hard drives. Typical values for hard drives are between 5000 and 20000 microseconds.

2. Set the value one time for all subsequent ADDM executions.

   For example, if the measured value if 8000 microseconds, run the following command as SYS user:

```
EXECUTE DBMS_ADVISOR.SET_DEFAULT_TASK_PARAMETER(
                     'ADDM', 'DBIO_EXPECTED', 8000);
```

Managing Snapshots

By default, the Automatic Workload Repository (AWR) generates snapshots of performance data once every hour, and retains the statistics in the workload repository for 7 days. It is possible to change the default values for both the snapshot interval and the retention period. The data in the snapshot interval is then analyzed by ADDM. AWR compares the difference between snapshots to determine which SQL statements to capture, based on the effect on the system load. This reduces the number of SQL statements that need to be captured over time.

This section contains the following topics:

- Creating Snapshots
- Modifying Snapshot Settings

Creating Snapshots

You can manually create snapshots, but this is usually not necessary because the AWR generates snapshots of the performance data once every hour by default. In some cases, however, it may be necessary to manually create snapshots to capture different durations of activity, such as when you want to compare performance data over a shorter period of time than the snapshot interval.

To create snapshots:

1. On the Database Performance page, under Additional Monitoring Links, click **Snapshots**.

   The Snapshots page appears with a list of the most recent snapshots.

2. Click **Create**.

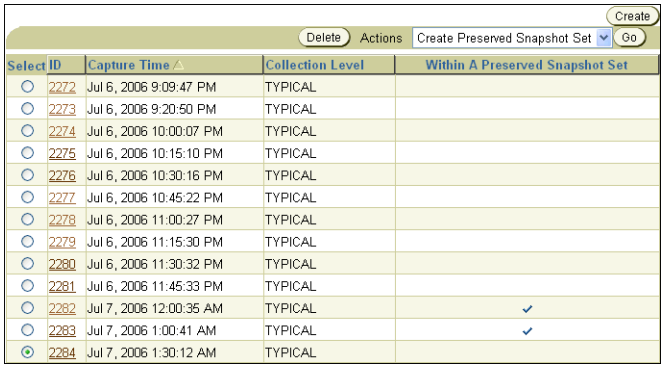
   The Confirmation page appears.

3. Click **Yes**.

   The Processing: Create Snapshot page is displayed while the snapshot is being taken.

4. Once the snapshot is taken, the Snapshots page reappears with a Confirmation message.

   In this example, the ID of the snapshot that was created is 2284.



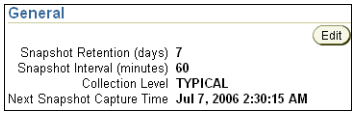
Modifying Snapshot Settings

By default, the AWR generates snapshots of the performance data once every hour. Alternatively, you can modify the default values of both the interval between snapshots and their retention period.

To modify the snapshot settings:

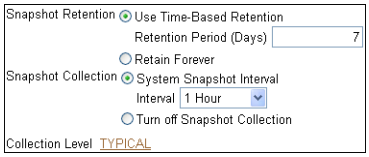
1. On the Database Administration page, under Statistics Management, click **Automatic Workload Repository**.

   The Automatic Workload Repository page appears.



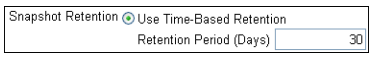
2. Click **Edit**.

   The Edit Settings page appears.



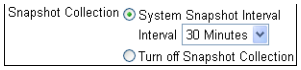
3. To change the retention period, in the Retention Period (Days) field, enter the number of days to retain the snapshots.

   You can also choose to retain snapshots indefinitely by selecting **Retain Forever**. It is recommended that you increase the snapshot retention period whenever possible based on the available disk space. In this example, the retention period is changed to 30 days.



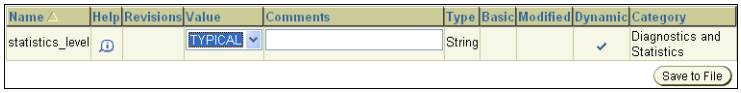
4. To change the interval between snapshots, select the desired interval from the Interval list.

   You can also choose to disable snapshot collection by selecting **Turn off Snapshot Collection**. In this example, the snapshot collection interval is changed to 30 minutes.



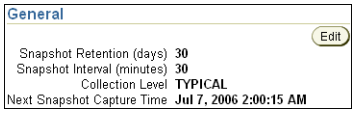
5. To change the level of statistics that are captured in snapshots, click the **Collection Level** link.

   The Initialization Parameter page appears. To change the statistics level, select the desired value in the Value list for the `statistics_level` parameter and click **Save to File**. In this example, the default value of Typical is used.



6. After the snapshot settings are modified, click **OK** to apply the changes.

   The Automatic Workload Repository page appears and the new settings are displayed.



Reviewing the Automatic Database Diagnostics Monitor Analysis

By default, ADDM runs every hour to analyze snapshots taken by the AWR during that period. If performance problems are found, the results of the analysis are displayed under Diagnostic Summary on the Database Home page, as shown in Figure 3–1.

Figure 3–1 Diagnostic Summary

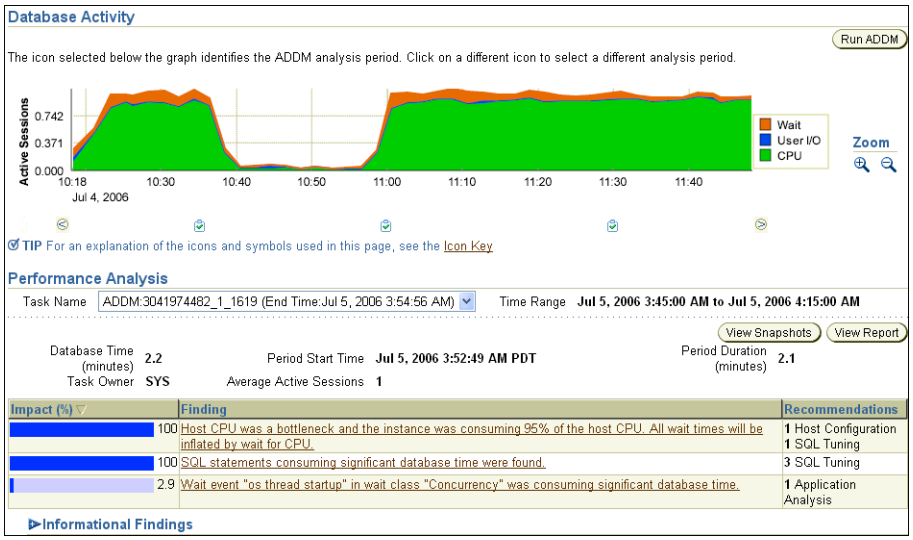
The link next to ADDM Findings shows how many ADDM findings were found in the most recent ADDM analysis.

To view ADDM findings:

- On the Database Home page, under Diagnostic Summary, click the link next to ADDM Findings.

  The Automatic Database Diagnostic Monitor (ADDM) page appears. The results of the ADDM run are displayed, as shown in Figure 3–2.

Figure 3–2 Automatic Database Diagnostic Monitor (ADDM) Page



On the Automatic Database Diagnostic Monitor (ADDM) page, the Database Activity graph shows the database activity during the ADDM analysis period. Database activity types are defined in the legend based on its corresponding color in the graph. In the example shown in Figure 3–2, the largest block of activity appears in green and corresponds to CPU, as described in the legend. This suggests that the host CPU may be a performance bottleneck during the ADDM analysis period. To select a different analysis period, click the left arrow icon to move to the previous analysis period, or the right arrow icon to move to the next analysis period. You can also click the Zoom icons to shorten or lengthen the analysis period displayed on the graph.

The ADDM findings for the analysis period are listed under Performance Analysis and contains the following columns:

- Impact (%)

  Displays an estimate of the portion of database time that is used by the performance problem that was found.

- Finding

  Displays a summary of the ADDM finding. To view details about a finding, click the link in this column.

- Recommendations

  Displays the type of operation ADDM recommends to resolve the performance problem identified by the finding.

The Informational Findings section lists the areas that do not have a performance impact and are for informational purpose only.

The results of the ADDM finding can also be viewed in a report that can be saved for later access. To view the ADDM report, click **View Report**.

Interpreting the Automatic Database Diagnostics Monitor Findings

The ADDM analysis results are represented as a set of findings. Each ADDM finding belongs to one of three types:

- Problem

Findings that describe the root cause of a database performance issue.

- Symptom

  Findings that contain information that often lead to one or more problem findings.

- Information

  Findings that are used to report areas of the system that do not have a performance impact.

Each problem finding is quantified with an estimate of the portion of DB time that resulted from the performance problem that was found.

When a specific problem has multiple causes, ADDM may report multiple findings. In this case, the impacts of these multiple findings can contain the same portion of DB time. Because the performance problems can overlap, summing all the impacts of the reported findings can yield a number higher than 100 percent of DB time. For example, if a system performs many read I/Os, ADDM may report a SQL statement responsible for 50 percent of DB time due to I/O activity as one finding, and an undersized buffer cache responsible for 75 percent of DB time as another finding.

A problem finding can be associated with a list of recommendations for reducing the impact of a performance problem. Each recommendation has a benefit that is an estimate of the portion of DB time that can be saved if the recommendation is implemented. When multiple recommendations are associated with an ADDM finding, the recommendations may contain alternatives for solving the same problem. In this case, the sum of the benefits may be higher than the impact of the finding. You do not need to apply all the recommendations to solve the same problem.

Recommendations are composed of actions and rationales. You need to apply all the actions of a recommendation to gain the estimated benefit of that recommendation. The rationales explain why the set of actions were recommended, and provide additional information for implementing the suggested recommendation. An ADDM action may present multiple solutions to you. If this is the case, choose the easiest solution to implement.

Implementing ADDM Recommendations

On the Automatic Database Diagnostic Monitor (ADDM) page shown in Figure 3–2 on page 3-6, three ADDM finding are displayed in the Performance Analysis section, as shown in Figure 3–3.

Figure 3–3 Performance Analysis

Impact (%) ▽	Finding	Recommendations
100	Host CPU was a bottleneck and the instance was consuming 97% of the host CPU. All wait times will be inflated by wait for CPU.	**1** Host Configuration **1** SQL Tuning
100	SQL statements consuming significant database time were found.	**4** SQL Tuning
2.2	Wait event "os thread startup" in wait class "Concurrency" was consuming significant database time.	**1** Application Analysis

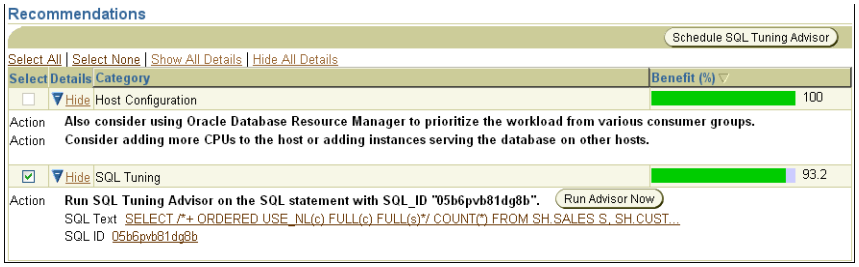
To implement ADDM recommendations:

1. On the Automatic Database Diagnostic Monitor (ADDM) page, under Performance Analysis, click the ADDM finding that has the greatest impact.

   In this example, there are two ADDM findings with 100% impact. The first ADDM finding dealing with host CPU will be first examined.

   The Performance Finding Details page appears.

2. Under Recommendations, identify the recommendations and required actions for each recommendation.



   In this example, two recommendations are displayed for this finding. The first recommendation contains two actions and is estimated to have a maximum benefit of up to 100% of DB time in the analysis period. The second recommendation contains one action and is estimated to have a maximum benefit of up to 93% of DB time in the analysis period.

3. Perform the required action of a chosen recommendation.

   In this example, the most effective solution is to use Oracle Database Resource Manager to prioritize the workload from various consumer groups and add more CPUs to the host system. However, adding CPUs to the host system may be costly. Running the SQL Tuning Advisor on the high-load SQL statement that ADDM has identified is easier to implement and can still provide a significant improvement.

   To run the SQL Tuning Advisor on the SQL statement, click **Run Advisor Now**. This will immediately run a SQL Tuning Advisor task on the SQL statement.

   See Also:

   - Chapter 10, "Tuning SQL Statements" for information about tuning SQL statements

Viewing Snapshot Statistics

You can view the data contained in snapshots taken by the AWR using Oracle Enterprise Manager. Typically, it is not necessary to review snapshot data because they consist primarily of raw statistics. Instead, you should rely on ADDM, which analyzes these statistics automatically to identify performance problems. Snapshot statistics should be used primarily by advanced users, or by DBAs who are accustomed to using Statspack for performance analysis.

To view snapshot statistics:

1. On the Database Administration page, under Statistics Management, click **Automatic Workload Repository**.

   The Automatic Workload Repository page appears.

2. Under Manage Snapshots and Preserved Snapshot Sets, click the Snapshots link.

   The Snapshots page appears.

3. To view the statistics gathered in a snapshot, click the **ID** link of the snapshot you want to view.

   The Snapshot Details page appears with the statistics gathered from the previous snapshot (snapshot 2283) to the selected snapshot (snapshot 2284) displayed.

Details | Report

Beginning Snapshot ID	2283	Ending Snapshot ID	2284
Beginning Snapshot Capture Time	Jul 7, 2006 1:00:41 AM	Ending Snapshot Capture Time	Jul 7, 2006 1:30:12 AM

Previous 1-27 of 27 Next

Name	Value	Per Second	Per Transaction
DB cpu (seconds)	0.00	0.00	0.00
DB time (seconds)	5228.43	2.95	17.31
db block changes	8173.00	4.62	27.06
execute count	16982.00	9.59	56.23
global cache cr block receive time (seconds)	0.00	0.00	0.00
global cache cr blocks received	0.00	0.00	0.00
global cache current block receive time (seconds)	0.00	0.00	0.00
global cache current blocks received	0.00	0.00	0.00
global cache get time (seconds)	0.00	0.00	0.00
global cache gets	0.00	0.00	0.00
opened cursors cumulative	10220.00	5.77	33.84
parse count (total)	5349.00	3.02	17.71
parse time cpu (seconds)	2.55	0.00	0.01
parse time elapsed (seconds)	2.78	0.00	0.01
physical reads	142.00	0.08	0.47
physical writes	1089.00	0.62	3.61
redo size (KB)	2079.07	1.17	6.88
session cursor cache hits	7485.00	4.23	24.78
session logical reads	80001.00	45.20	264.90
sql execute cpu time (seconds)	0.00	0.00	0.00
sql execute elapsed time (seconds)	0.00	0.00	0.00
user calls	4738.00	2.68	15.69
user commits	286.00	0.16	0.95
user rollbacks	16.00	0.01	0.05
workarea executions - multipass	0.00	0.00	0.00
workarea executions - onepass	0.00	0.00	0.00
workarea executions - optimal	2560.00	1.45	8.48

4. To view a Workload Repository report of the statistics, click the **Report** tab.

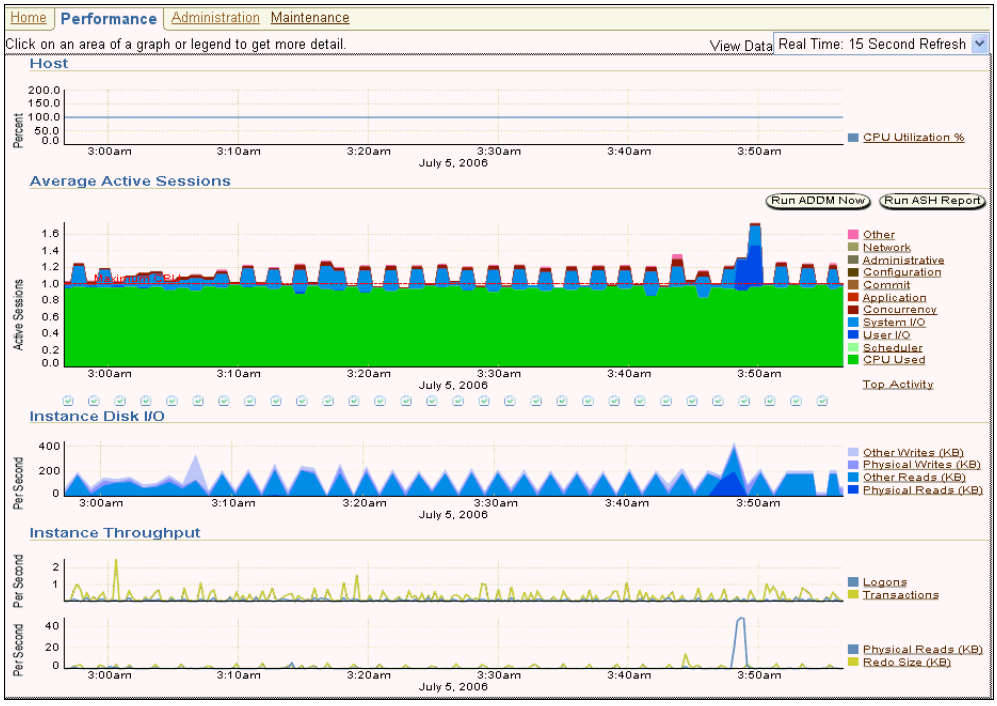
   The Workload Repository report is displayed.

4

Monitoring Real-Time Database Performance

This chapter describes how to identify and respond to real-time performance problems using the Database Performance page in Enterprise Manager. The Database Performance page in Oracle Enterprise Manager, shown in Figure 4–1, displays information in four sections that can be used to assess the overall performance of the database in real time.

Figure 4–1 Database Performance Page



Typically, you should use the automatic diagnostic feature of ADDM to identify performance problems with the database, as described in Chapter 3, "Automatic Database Performance Monitoring". In some cases, however, you may want to monitor the database performance in real time to identify performance problems as they happen. For example, ADDM performs its analysis after each AWR snapshot, which by default is once every hour. However, if you notice a sudden spike in database activity on the Database Performance page, you may want to investigate the incident before the next ADDM analysis. By drilling down to appropriate pages from the Database Performance page, you can identify performance problems with the database in real time. If a performance problem is identified, you can choose to run ADDM

manually to analyze it immediately, without having to wait until the next ADDM analysis. For more information about running ADDM manually to analyze performance in real time, see "Manually Running ADDM to Analyze Current Database Performance" on page 6-1.

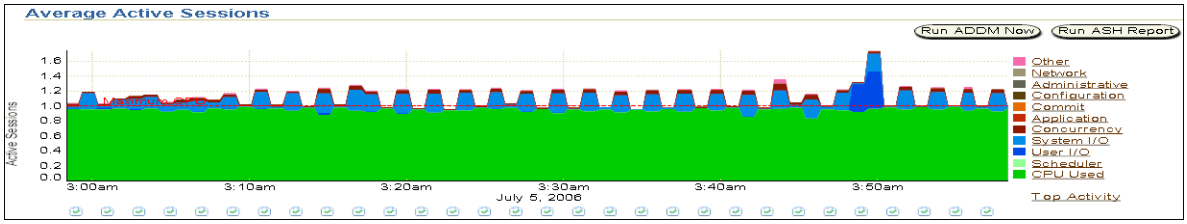
This chapter contains the following sections:

- Monitoring User Activity
- Monitoring Instance Activity
- Monitoring Host Activity

Monitoring User Activity

The Average Active Sessions section on the Database Performance page, shown in Figure 4–2, shows how much CPU time each user is using, and whether or not there are users waiting for resources.

Figure 4–2 Monitoring User Activity



You can monitor database health and user activity using the Average Active Sessions graph. When the CPU Used value (shown in green) reaches the Maximum CPU line (shown as a dotted red line), the database instance is running at 100 percent of CPU time on the host system. All other values in the graph represent users waiting and contention for resources, which are categorized by wait classes in the legend. Values that use a larger block of active sessions represent bottlenecks caused a particular wait class, as indicated by the corresponding color in the legend. In the graph shown in Figure 4–2, the largest block of activity appears in green and corresponds to the CPU Used wait class as described in the legend. To identify each wait class, mouse over the block in the graph, and its corresponding wait class will be highlighted in the legend.

After a performance problem is discovered, it can be resolved in real time, directly from the Average Active Sessions section by:

- Clicking a snapshot below the graph that corresponds to the time when the performance problem occurred to run ADDM for that time period.

  For information about ADDM analysis, see "Reviewing the Automatic Database Diagnostics Monitor Analysis" on page 3-5.

- Creating a snapshot manually and clicking **Run ADDM Now**.

  For information about creating snapshots manually, see "Creating Snapshots" on page 3-3. For information about running ADDM manually, see "Manually Running ADDM to Analyze Current Database Performance" on page 6-1.

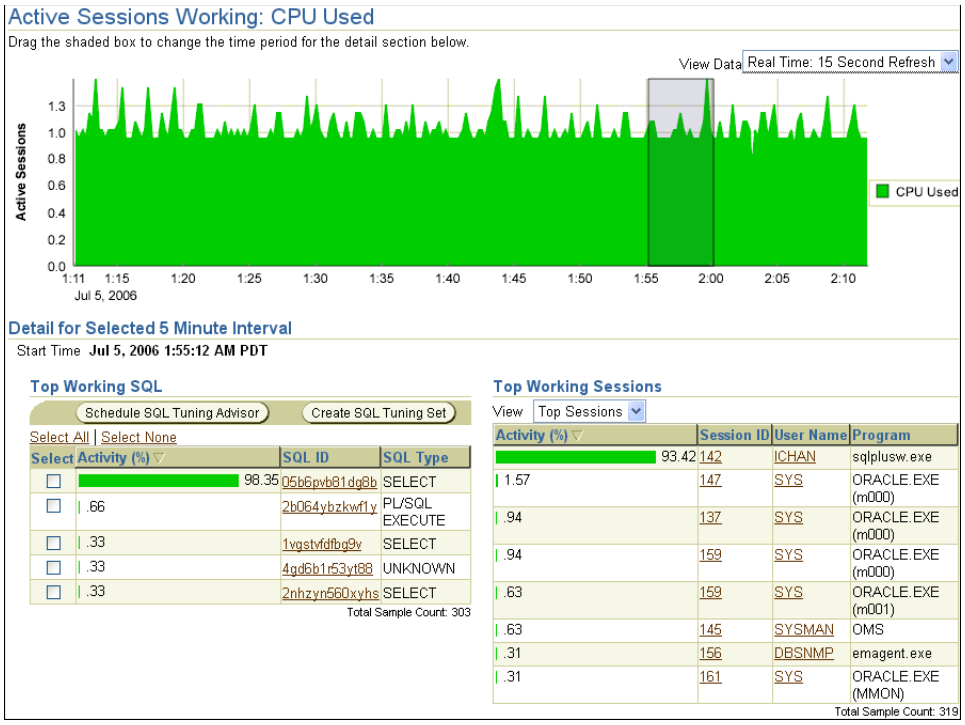
- Clicking **Run ASH Report** to create an ASH report to analyze transient performance problems that last for only a short period of time.

  For information about ASH reports, see "Using Active Session History Reports" on page 7-3

- Performing further investigation by drilling down to the wait class with the most active sessions, as described in the remainder of this section

In some cases, you may want to investigate the wait class that is using the greatest amount of wait time. To do so, identify and drill down to the wait class with the most active sessions by clicking the largest block of color on the graph or its corresponding wait class in the legend. This displays the Active Sessions Working page for that wait class, as shown in Figure 4–3.

Figure 4–3 Active Sessions Working page



The Active Sessions Working page shows a 1-hour time line. Details for each wait class are displayed in 5-minute intervals under Detail for Selected 5 Minute Interval. To move the 5-minute interval, drag and drop the shaded box to the time of interest. The information contained in the Detail for Selected 5 Minute Interval section is automatically updated to display the selected time period. Use this page to dynamically identify SQL statements or sessions that may be causing performance problems by analyzing the details of a wait class during the time period when the wait event occurred.

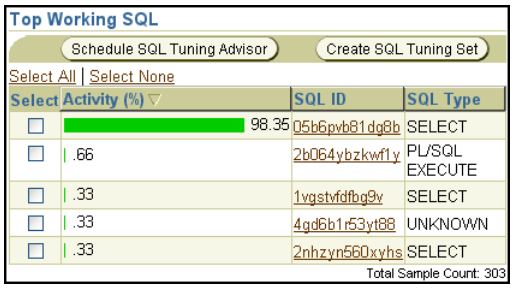
You can view the details of a wait class in different dimensions by:

- Monitoring Top SQL
- Monitoring Top Sessions
- Monitoring Top Services
- Monitoring Top Modules
- Monitoring Top Actions

Monitoring Top SQL

The Top Working SQL section appears on the left side of the Active Sessions Working page under Detail for Selected 5 Minute Interval, as shown in Figure 4–4.

Figure 4–4 Monitoring Top SQL



The Top Working SQL section displays the top SQL statements waiting for the corresponding wait class during the selected time period. If one or several SQL statements are using the majority of the wait time, as is the case shown in Figure 4–3, then they should be investigated.

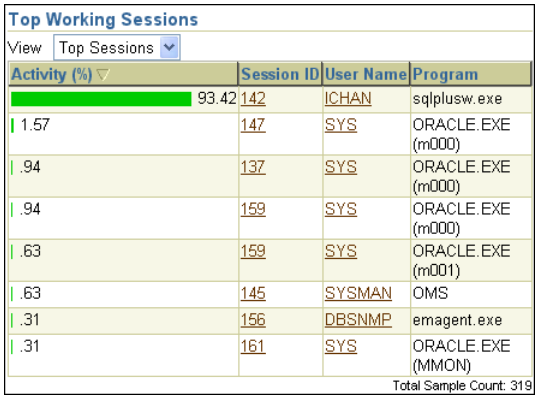
To view details about a SQL statement, click the **SQL ID** link of the SQL statement. This displays the SQL Details page for the selected SQL statement. For information about viewing details on a SQL statement, see "Viewing Details of SQL Statements" on page 9-3.

For SQL statements that are using the majority of the wait time, as is the case shown in Figure 4–3, use the SQL Tuning Advisor or create a SQL tuning set to tune the problematic SQL statements. For information about tuning SQL statements, see "Tuning SQL Statements Using the SQL Tuning Advisor" on page 10-2.

Monitoring Top Sessions

By default, the Top Working Sessions section appears on the right side of the Active Sessions Working page under Detail for Selected 5 Minute Interval, as shown in Figure 4–5.

Figure 4–5 Monitoring Top Sessions

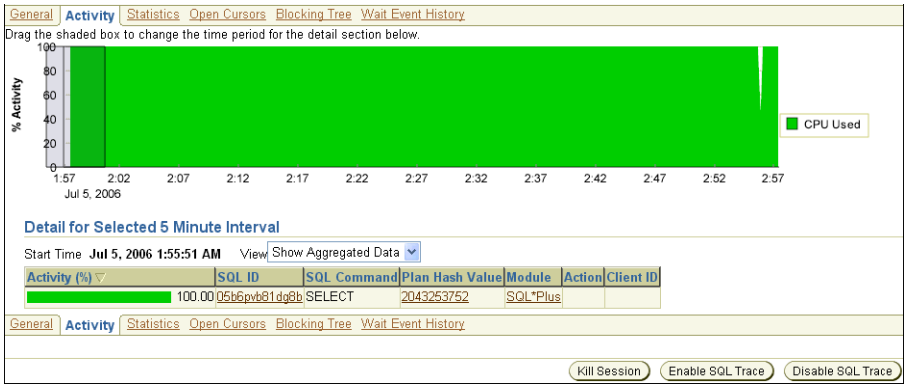


The Top Working Sessions section displays the top sessions waiting for the corresponding wait class during the selected time period. Sessions represent specific

user connections to the database through a user process. A session lasts from the time the user connects until the time the user disconnects or exits the database application. For example, when a user starts SQL*Plus, the user must provide a valid user name and password, and then a session is established for that user. If a single session is using the majority of the wait time, as is the case shown in Figure 4–5, then it should be investigated.

To view details about a session, click the **Session ID** link of the session. This displays the Session Details page, as shown in Figure 4–6, which contains information such as session activity, session statistics, open cursors, blocking sessions, and wait events for the selected session.

Figure 4–6 Viewing Session Details



After the information is analyzed, you can choose to end the session by clicking **Kill Session**. In this example, because the session is consuming 100 percent of database activity, you should consider ending the session and proceeding to tune the SQL statement that this session is running.

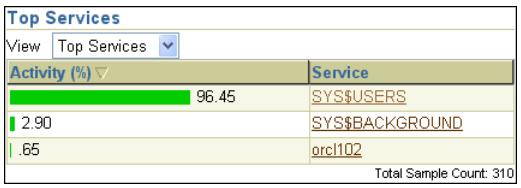
See Also:

- Chapter 10, "Tuning SQL Statements" for information about tuning SQL statements

Monitoring Top Services

The Top Services section can be accessed by selecting **Top Services** from the View list on the right side of the Active Sessions Working page under Detail for Selected 5 Minute Interval. The Top Services section displays the top services waiting for the corresponding wait event during the selected time period, as shown in Figure 4–7.

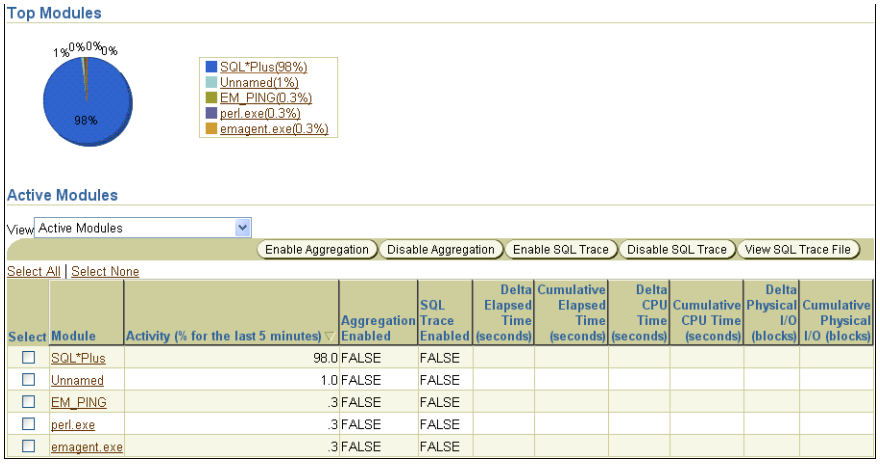
Figure 4–7 Monitoring Top Services



Services represent groups of applications with common attributes, service-level thresholds, and priorities. For example, the SYS$USERS service is the default service name used when a user session is established without explicitly identifying its service

name, and the SYS$BACKGROUND service consists of all Oracle Database background processes. If a single service is using the majority of the wait time, as is the case shown in Figure 4–7, then it should be investigated. To view details about a service, click the **Service** link of the service. This displays the Service page, as shown in Figure 4–8, which lists the top modules and statistics for the selected service.

Figure 4–8 Viewing Service Details

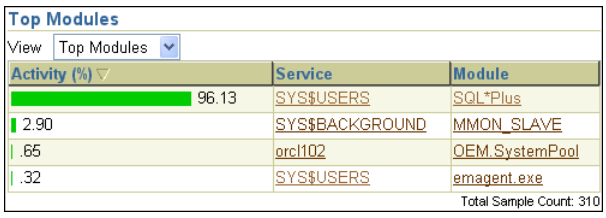


In this example, the SQL*Plus module is consuming 98 percent of the database activity for this service. The SQL statements that this service is running should be investigated.

Monitoring Top Modules

The Top Modules section can be accessed by selecting **Top Modules** from the View list on the right side of the Active Sessions Working page under Detail for Selected 5 Minute Interval. The Top Modules section displays the top modules waiting for the corresponding wait event during the selected time period, as shown in Figure 4–9.

Figure 4–9 Monitoring Top Modules



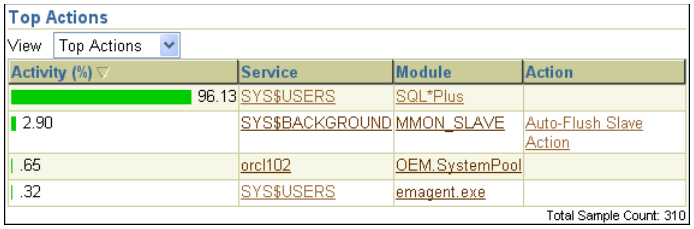
Modules represent the applications that set the service name as part of the workload definition. For example, the DBMS_SCHEDULER module may assigns jobs that run within the SYS$BACKGROUND service. If a single module is using the majority of the wait time, as is the case shown in Figure 4–9, then it should be investigated. To view details about a module, click the **Module** link of the module. This displays the Module page, which lists the top actions and contains statistics for the selected module.

Monitoring Top Actions

The Top Actions section can be accessed by selecting **Top Actions** from the View list on the right side of the Active Sessions Working page under Detail for Selected 5 Minute

Interval. The Top Actions section displays the top actions waiting for the corresponding wait event during the selected time period, as shown in Figure 4–10.

Figure 4–10 Monitoring Top Actions



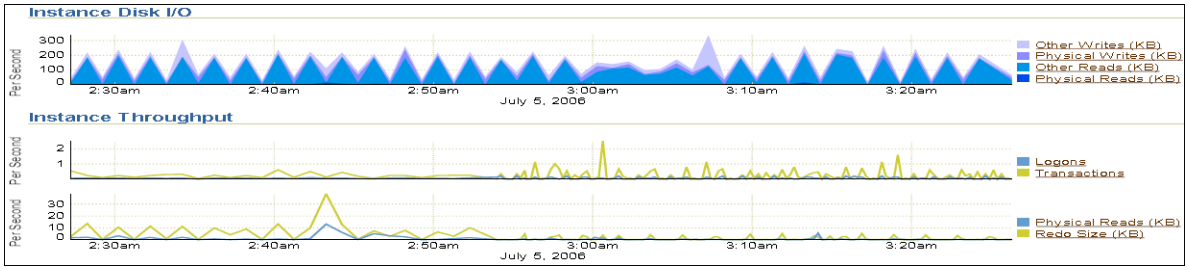
Actions represent the jobs that are performed by a module. For example, the `DBMS_SCHEDULER` module can run the `GATHER_STATS_JOB` action to gather statistics on all database objects. If a single action is using the majority of the wait time, then it should be investigated. To view details about an action, click the **Action** link of the action. This displays the Action page, which contains statistics for the selected action.

Monitoring Instance Activity

The Instance Disk I/O and Instance Throughput sections on the Database Performance page shown in Figure 4–11 displays:

- Number of physical reads, physical writes, logons, and transactions per second
- Number of physical reads, physical writes, and logons per transaction

Figure 4–11 Monitoring Instance Activity

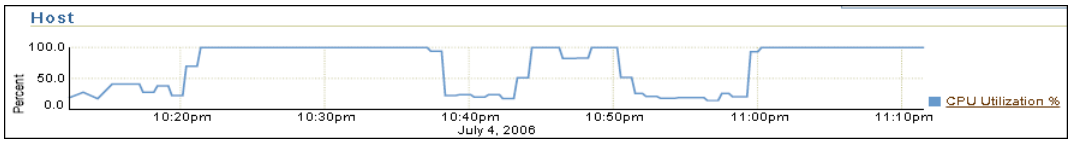


To view the top sessions for each type of activity, click the corresponding link in the legend. This displays the Top Sessions page for that activity. On the Top Sessions page, you can view information about each session by selecting a session and clicking **View**. After the information is analyzed, you can choose to end the session by clicking **Kill Session**.

Monitoring Host Activity

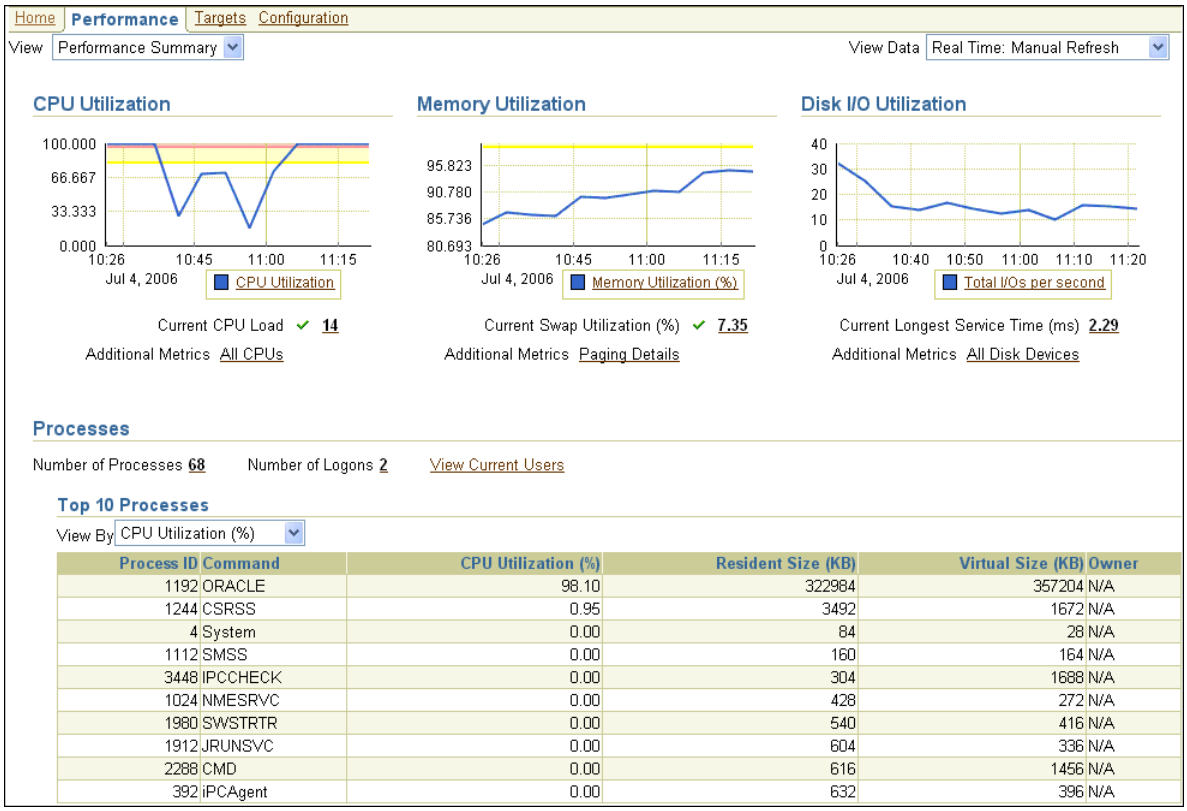
The Host section on the Database Performance page, shown in Figure 4–12, displays utilization information about the system hosting the database.

Figure 4–12 Monitoring Host Activity



Before analyzing the database, you should first verify if the host system has enough CPU, memory, and disk resources available to run the database. To view details about CPU, memory, and disk utilization, click the **CPU Utilization %** link in the legend. The Performance Summary page appears, as shown in Figure 4–13.

Figure 4–13 Performance Summary



The Performance Summary page displays metric values for CPU utilization, memory utilization, disk I/O utilization, and the top 10 processes ordered by both CPU and memory utilization.

To determine if the host system has enough resources available to run the database, first establish appropriate expectations for the amount of resources your system should be using. Then, determine whether sufficient resources are available and recognize when your system is using too many resources. Begin by determining the amount of CPU, memory, and disk resources the database uses in the following scenarios:

- When your system is idle, or when little database and non-database activity exists
- At average workloads
- At peak workloads

Workload is an important factor when evaluating the level of resource utilization for your system. During peak workload hours, 90 percent utilization of a resource, such as a CPU with 10 percent idle and waiting time, can be acceptable. However, if your system shows high utilization at normal workload, then there is no room for additional workload. After these values are determined, you can set the appropriate threshold values for the related metrics so the system can automatically generate alerts when these thresholds are exceeded. For information about setting metric thresholds, see "Setting Metric Thresholds for Performance Alerts" on page 5-1.

This section contains the following topics:

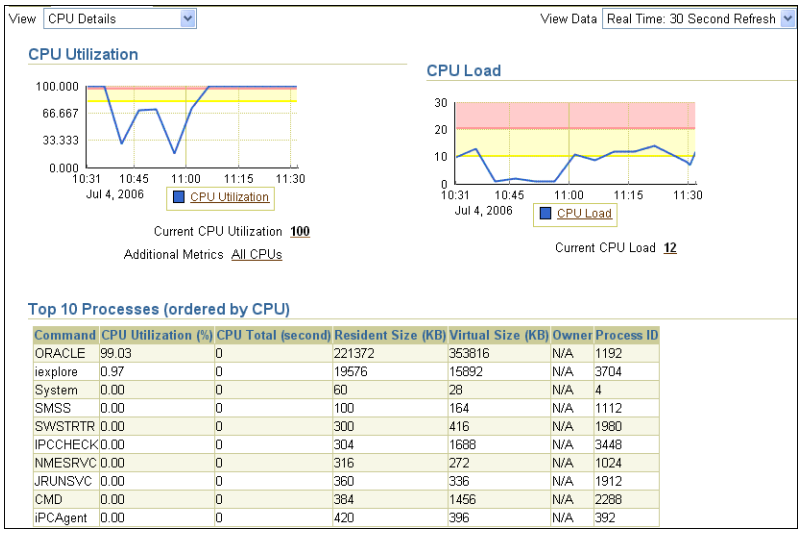
Monitoring CPU Utilization

This section describes how to monitor CPU utilization.

To monitor CPU utilization:

1. On the Performance Summary page, from the View list, select **CPU Details**.

   The CPU Details view appears. This view contains CPU utilization statistics gathered over the last hour, the CPU load, and the top 10 processes ordered by CPU utilization.



Command	CPU Utilization (%)	CPU Total (second)	Resident Size (KB)	Virtual Size (KB)	Owner	Process ID
ORACLE	99.03	0	221372	353816	N/A	1192
iexplore	0.97	0	19576	15892	N/A	3704
System	0.00	0	60	28	N/A	4
SMSS	0.00	0	100	164	N/A	1112
SWSTRTR	0.00	0	300	416	N/A	1980
IPCCHECK	0.00	0	304	1688	N/A	3448
NMESRVC	0.00	0	316	272	N/A	1024
JRUNSVC	0.00	0	360	336	N/A	1912
CMD	0.00	0	384	1456	N/A	2288
iPCAgent	0.00	0	420	396	N/A	392

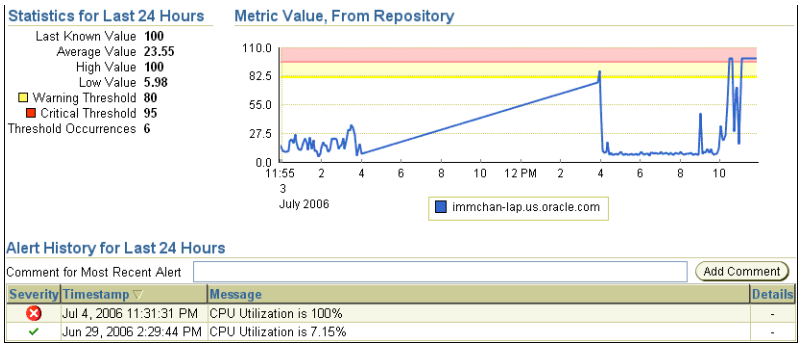
2. Verify the current CPU utilization using the CPU Utilization graph.

   The CPU Utilization graph shows how much CPU is currently used. During normal workload hours, the value should not exceed the critical threshold (shown in red).

3. Verify the CPU statistics over the last 24 hours.

   Click **CPU Utilization**.

   The CPU Utilization page appears. This page contains CPU utilization statistics and related alerts generated over the last 24 hours.



In this example, the CPU utilization crossed the critical threshold value at 11:31 p.m., so an alert for CPU utilization is generated to indicate that a CPU performance problem may exist. If you notice an unexpected spike in this value that is sustained through normal workload hours, then the CPU performance problem should be investigated.

4. Verify the current CPU load using the CPU Load graph.

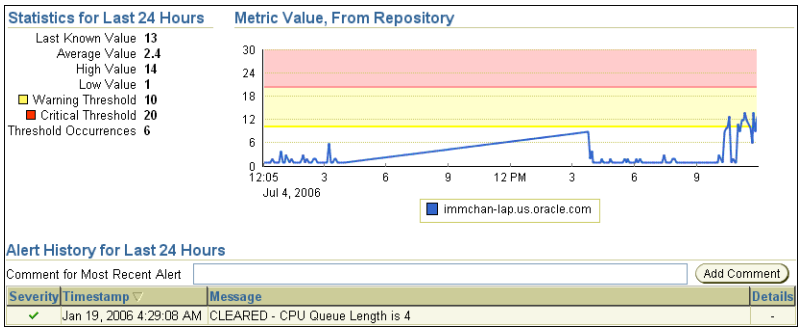
   The CPU Load graph shows the current CPU load. During normal workload hours, the value should not exceed the warning threshold (shown in yellow).

   CPU load represents the average number of processes waiting to be scheduled for CPU resources in the previous minute, or the level of CPU contention time over time.

5. Verify the CPU load over the last 24 hours.

   Click **CPU Load**.

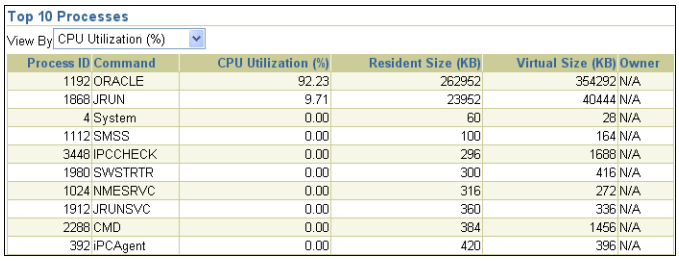
   The CPU Queue Length page appears. This page contains CPU load statistics and related alerts generated over the last 24 hours.



In this example, the CPU load crossed the warning threshold after 10:00 p.m., but it is still below the critical threshold, so an alert was not generated. If you notice an unexpected spike in this value that is sustained through normal workload hours, then a CPU performance problem might exist.

6. On the Performance Summary page, verify the top processes in the Top 10 Processes section.

   If a process is taking up too much of the CPU utilization percentage, then this process should be investigated.



Process ID	Command	CPU Utilization (%)	Resident Size (KB)	Virtual Size (KB)	Owner
1192	ORACLE	92.23	262952	354292	N/A
1868	JRUN	9.71	23952	40444	N/A
4	System	0.00	60	28	N/A
1112	SMSS	0.00	100	164	N/A
3448	IPCCHECK	0.00	296	1688	N/A
1980	SWSTRTR	0.00	300	416	N/A
1024	NMESRVC	0.00	316	272	N/A
1912	JRUNSVC	0.00	360	336	N/A
2288	CMD	0.00	384	1456	N/A
392	iPCAgent	0.00	420	396	N/A

In this example, the database is consuming 92 percent of CPU utilization. Therefore, the database is the likely source of the CPU performance problem and should be investigated.

7. If a CPU performance problem is identified, you can try to resolve the issue by:
   - Using Oracle Database Resource Manager to reduce the impact of peak-load-use patterns by prioritizing CPU resource allocation
   - Avoiding running too many processes that use a lot of CPU
   - Increasing hardware capacity, including changing the system architecture

   See Also:

   - *Oracle Database Performance Tuning Guide* for information about resolving CPU issues
   - *Oracle Database Administrator's Guide* for information about Oracle Database Resource Manager

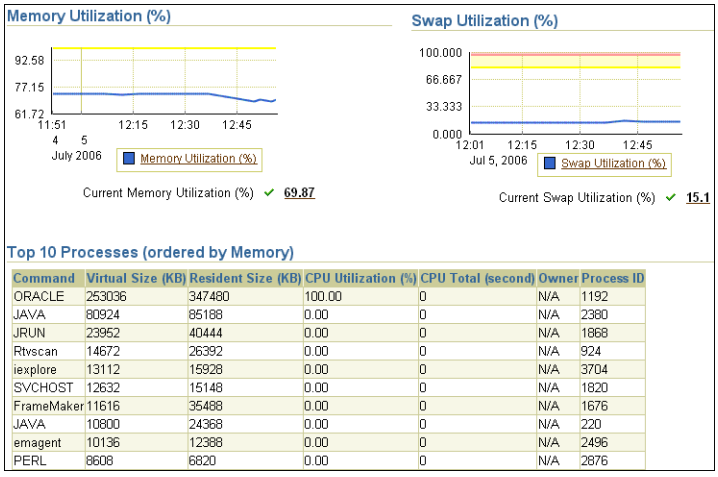
Monitoring Memory Utilization

This section describes how to monitor memory utilization.

To monitor memory utilization:

1. On the Performance Summary page, from the View list, select **Memory Details**.

   The Memory Details view appears. This view contains memory and swap utilization statistics gathered over the last hour, and the top 10 processes ordered by memory utilization.



Command	Virtual Size (KB)	Resident Size (KB)	CPU Utilization (%)	CPU Total (second)	Owner	Process ID
ORACLE	253036	347480	100.00	0	N/A	1192
JAVA	80924	85188	0.00	0	N/A	2380
JRUN	23952	40444	0.00	0	N/A	1868
Rtvscan	14672	26392	0.00	0	N/A	924
iexplore	13112	15928	0.00	0	N/A	3704
SVCHOST	12632	15148	0.00	0	N/A	1820
FrameMaker	11616	35488	0.00	0	N/A	1676
JAVA	10800	24368	0.00	0	N/A	220
emagent	10136	12388	0.00	0	N/A	2496
PERL	8608	6820	0.00	0	N/A	2876

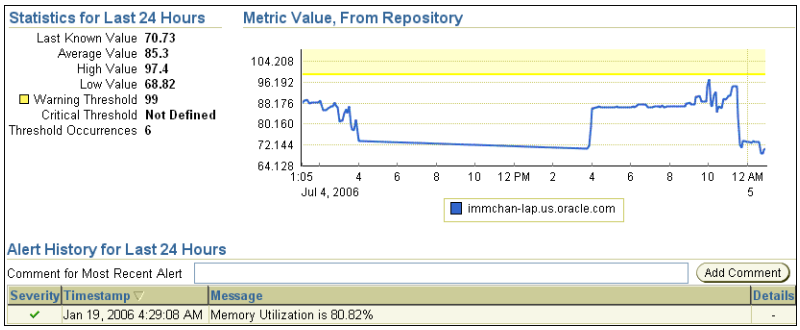
2. Verify the current memory utilization using the Memory Utilization graph.

The Memory Utilization graph shows how much memory is currently being used. During normal workload hours, the value should not exceed the warning threshold (shown in yellow).

3. Verify the memory statistics over the last 24 hours.

   Click **Memory Utilization**.

   The Memory Utilization page appears. This page contains memory utilization statistics and related alerts generated over the last 24 hours.



   In this example, memory utilization is near, but does not exceed, the warning threshold value (99 percent) from 4:00 p.m. to 11:00 p.m., so an alert is not generated. If you notice an unexpected spike in this value that is sustained through normal workload hours, then a memory performance problem might exist.

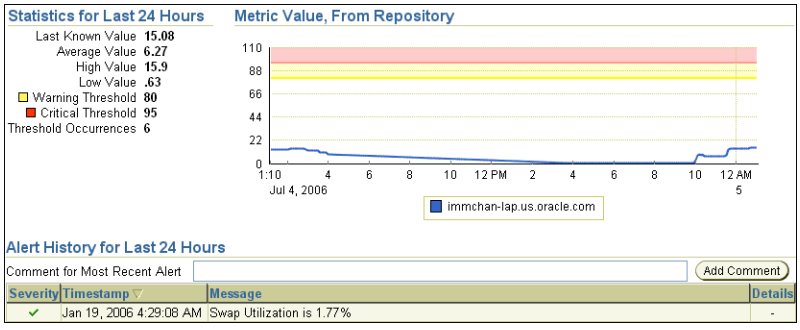
4. Verify current swap utilization using the Swap Utilization graph.

   The Swap Utilization graph shows how much swapping space is currently being used. During normal workload hours, the value should not exceed the warning threshold (shown in yellow).

5. Verify swap utilization over the last 24 hours.

   Click **Swap Utilization**.

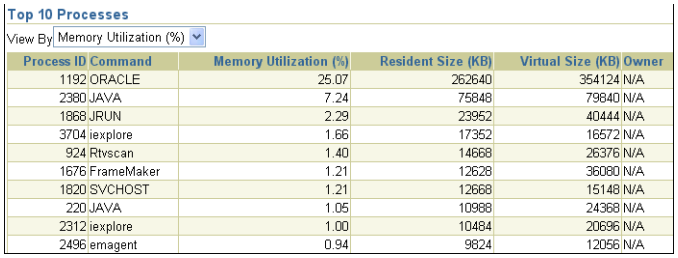
   The Swap Utilization page appears. This page contains swap utilization statistics and related alerts generated over the last 24 hours.



   In this example, swap utilization is below the warning threshold, so an alert is not generated. If you notice an unexpected spike in this value that is sustained through normal workload hours, then a memory performance problem might exist.

6. On the Performance Summary page, verify the top processes in the Top 10 Processes section.

If a process is taking up too much memory, then this process should be investigated.



Top 10 Processes

View By: Memory Utilization (%)

Process ID	Command	Memory Utilization (%)	Resident Size (KB)	Virtual Size (KB)	Owner
1192	ORACLE	25.07	262640	354124	N/A
2380	JAVA	7.24	75848	79840	N/A
1868	JRUN	2.29	23952	40444	N/A
3704	iexplore	1.66	17352	16572	N/A
924	Rtvscan	1.40	14668	26376	N/A
1676	FrameMaker	1.21	12628	36080	N/A
1820	SVCHOST	1.21	12668	15148	N/A
220	JAVA	1.05	10988	24368	N/A
2312	iexplore	1.00	10484	20696	N/A
2496	emagent	0.94	9824	12056	N/A

In this example, the database is consuming 25 percent of memory utilization, and there does not appear to be a memory performance problem.

7. If a memory performance problem is identified, you can attempt to resolve the issue by:
   - Using Automatic Shared Memory Management to manage the SGA memory
   - Using Automatic PGA Management to manage SQL memory execution
   - Avoiding running too many processes that use a lot of memory
   - Reducing paging or swapping
   - Reducing the number of open cursors and hard parsing with cursor sharing

   See Also:

   - *Oracle Database Performance Tuning Guide* for information about resolving memory issues

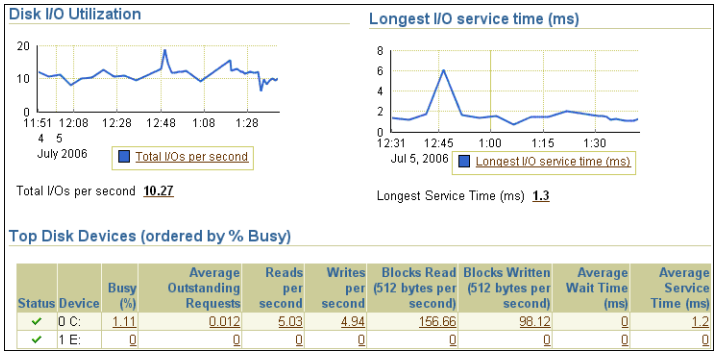
Monitoring Disk I/O Utilization

This section describes how to monitor disk I/O utilization.

To monitor disk I/O utilization:

1. On the Performance Summary page, from the View list, select **Disk Details**.

   The Disk Details view appears. This view contains disk I/O utilization and service time statistics gathered over the last hour, and the top disk devices ordered by busy percentage.



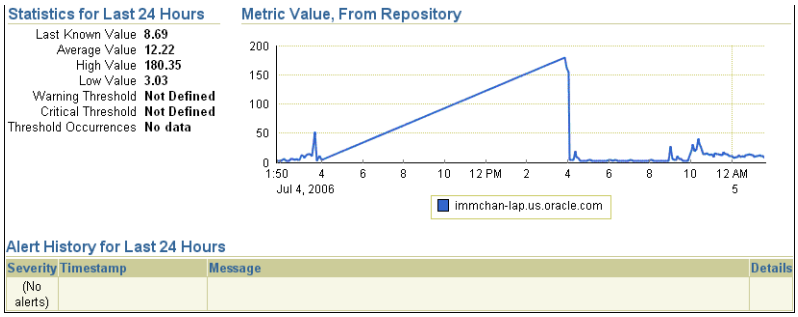
2. Verify the current disk I/O utilization using the Disk I/O Utilization graph.

The Disk I/O Utilization graph shows how many disk I/Os are being performed per second.

3. Verify the disk I/O statistics over the last 24 hours.

   Click **Total I/Os per Second**.

   The Total I/O page appears. This page contains disk utilization statistics and related alerts generated over the last 24 hours.



   In this example, an alert is not generated because a threshold is not defined. If you notice an unexpected spike in this value that is sustained through normal workload hours, as is the case in this example from 2:00 p.m. to 4:00 p.m., then a disk I/O performance problem might exist and should be investigated.

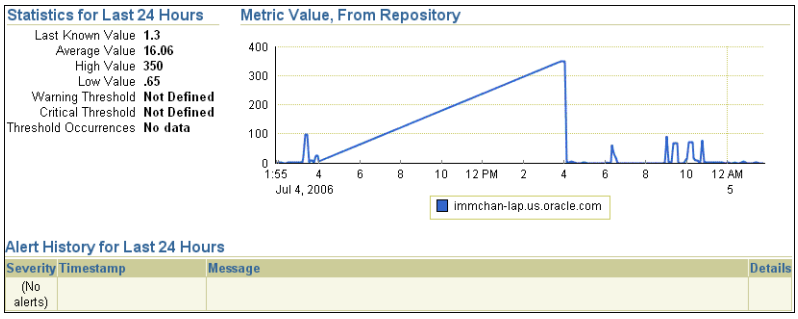
4. Verify the current I/O service time using the Longest I/O Service Time graph.

   The Longest I/O Service Time graph shows the longest service time for disk I/Os.

5. Verify I/O service time over the last 24 hours.

   Click **Longest I/O Service Time**.

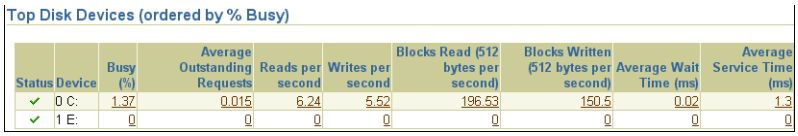
   The Longest Service Time page appears. This page contains I/O service time statistics and related alerts generated over the last 24 hours.



   In this example, an alert is not generated because a threshold is not defined. If you notice an unexpected spike in this value that is sustained through normal workload hours, as is the case in this example from 2:00 p.m. to 4:00 p.m., then a disk I/O performance problem might exist and should be investigated.

6. On the Disk Details page, verify the disk devices in the Top Disk Devices section.

   If a particular disk is busy a high percentage of the time, then this disk should be investigated.



Top Disk Devices (ordered by % Busy)

Status	Device	Busy (%)	Average Outstanding Requests	Reads per second	Writes per second	Blocks Read (512 bytes per second)	Blocks Written (512 bytes per second)	Average Wait Time (ms)	Average Service Time (ms)
✓	0 C:	1.37	0.015	6.24	5.52	196.53	150.5	0.02	1.3
✓	1 E:	0	0	0	0	0	0	0	0

In this example, the drive that hosts Oracle Database (drive C) is only busy about 1.3 percent of the time, and there does not appear to be a disk performance problem.

7. If a disk I/O performance problem is identified, you can attempt to resolve the problem by:
   - Using Automatic Storage Management (ASM) to manage database storage
   - Striping everything across every disk to distribute I/O
   - Moving files, such as archive logs and redo logs, to separate disks
   - Storing required data in memory to reduce the number of physical I/Os

   See Also:

   - *Oracle Database Performance Tuning Guide* for information about resolving disk I/O issues

5

Monitoring Performance Alerts

Oracle Database includes a built-in alerts infrastructure to notify you of impending problems with the database. By default, Oracle Database enables the following alerts:

- Table Space Usage
- Snapshot Too Old
- Recovery Area Low on Free Space
- Resumable Session Suspended

For information about alerts and how to manage them, see *Oracle Database 2 Day DBA*.

In addition to these default alerts, you can use performance alerts to detect any unusual changes in database performance.

This chapter contains the following sections:

Setting Metric Thresholds for Performance Alerts

Performance alerts are based on metrics that are performance related. These alerts are either environment-dependent or application-dependent.

Environment-dependent performance alerts may not be relevant on all systems. For example, the `AVERAGE_FILE_READ_TIME` metric generates an alert when the average time to read a file exceeds the metric threshold. This may be useful on a system with only one disk. On a system with multiple disks, however, the alert may not be relevant because I/O is spread across the entire subsystem.

Application-dependent performance alerts are typically relevant on all systems. For example, the `BLOCKED_USERS` metric generates a performance alert when the number of users blocked by a particular session exceeds the metric threshold. This alert is relevant regardless of how the environment is configured.

To get the most relevant information from performance alerts, you should set the threshold values of performance metrics to values that represent desirable boundaries for your system. You can then fine-tune these values over time until an equilibrium is reached.

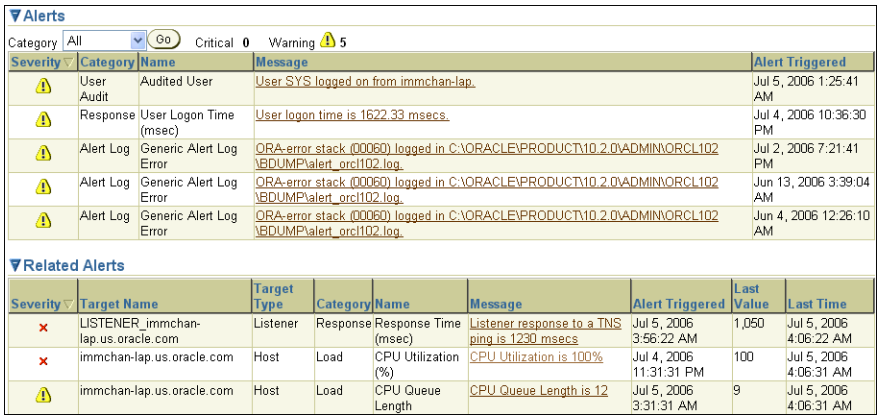
See Also:

- *Oracle Database 2 Day DBA* for information about setting metric thresholds

Responding to Alerts

When an alert is generated by Oracle Database, it appears in the Alerts section on the Database Home page, as shown in Figure 5–1.

Figure 5–1 Responding to Alerts



When you receive an alert, click the **Message** link. The corresponding page that contains further information for the alert will be displayed. Follow the recommendations it provides, or consider running ADDM or another advisor to get more detailed diagnostics of the system or object behavior.

In the example shown in Figure 5–1, clicking the **Message** link for the CPU Utilization alert displays the CPU Utilization page. You can use the statistics on this page to identify the performance problem that triggered the alert.

Oracle Enterprise Manager also enables you to configure alerts to be sent using E-mail, pager, or cellular phone text messaging. For information about how to configure the alert notification method, see *Oracle Database 2 Day DBA*.

Clearing Alerts

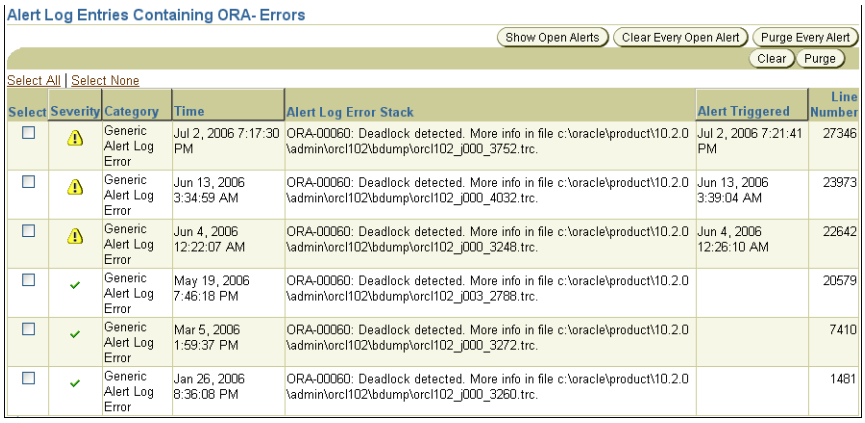
Most alerts, such as the CPU Utilization alert, are cleared automatically when the cause of the problem disappears. However, other alerts, such as the Generic Alert Log Error alert, need to be acknowledged.

After taking the necessary corrective measures, you can acknowledge an alert by clearing or purging it. Clearing an alert sends the alert to the Alert History, which is viewable from the Database Home page under Related Links. Purging an alert removes it from the Alert History.

To clear alerts:

1. On the Database Home page, under the Diagnostic Summary heading, click **Alert Log**.

   The Alert Log page appears.



Alert Log Entries Containing ORA- Errors

Show Open Alerts | Clear Every Open Alert | Purge Every Alert | Clear | Purge

Select All | Select None

Select	Severity	Category	Time	Alert Log Error Stack	Alert Triggered	Line Number
☐	⚠	Generic Alert Log Error	Jul 2, 2006 7:17:30 PM	ORA-00060: Deadlock detected. More info in file c:\oracle\product\10.2.0\admin\orcl102\bdump\orcl102_j000_3752.trc.	Jul 2, 2006 7:21:41 PM	27346
☐	⚠	Generic Alert Log Error	Jun 13, 2006 3:34:59 AM	ORA-00060: Deadlock detected. More info in file c:\oracle\product\10.2.0\admin\orcl102\bdump\orcl102_j000_4032.trc.	Jun 13, 2006 3:39:04 AM	23973
☐	⚠	Generic Alert Log Error	Jun 4, 2006 12:22:07 AM	ORA-00060: Deadlock detected. More info in file c:\oracle\product\10.2.0\admin\orcl102\bdump\orcl102_j000_3248.trc.	Jun 4, 2006 12:26:10 AM	22642
☐	✓	Generic Alert Log Error	May 19, 2006 7:46:18 PM	ORA-00060: Deadlock detected. More info in file c:\oracle\product\10.2.0\admin\orcl102\bdump\orcl102_j003_2788.trc.		20579
☐	✓	Generic Alert Log Error	Mar 5, 2006 1:59:37 PM	ORA-00060: Deadlock detected. More info in file c:\oracle\product\10.2.0\admin\orcl102\bdump\orcl102_j000_3272.trc.		7410
☐	✓	Generic Alert Log Error	Jan 26, 2006 8:36:08 PM	ORA-00060: Deadlock detected. More info in file c:\oracle\product\10.2.0\admin\orcl102\bdump\orcl102_j000_3260.trc.		1481

2. Select the alerts you want to clear and click **Clear**.

   To clear all open alerts, click **Clear Every Open Alert**.

3. Select the alerts you want to purge and click **Purge**.

   To purge all alerts, click **Purge Every Alert**.

Part III

Reactive Database Tuning

Part III describes how to tune Oracle Database in response to a reported problem, such as when the user reports a performance problem with the database that needs to be tuned immediately.

This part contains the following chapters:

6

Manual Database Performance Monitoring

This chapter describes how to run the Automatic Database Diagnostics Monitor (ADDM) manually to monitor current and historical database performance. Typically, you should use the automatic diagnostic feature of ADDM to identify performance problems with the database. As described in Chapter 3, "Automatic Database Performance Monitoring", ADDM runs once every hour by default, and it is possible to configure ADDM to run more or less frequently. However, there may be some cases where you may want to run ADDM manually.

For example, you may notice significant performance degradation that did not exist in the previous ADDM analysis period, and the next ADDM analysis is not scheduled to run for another 30 minutes. In this case, you may want to run ADDM manually before the next scheduled ADDM analysis to immediately identify and resolve the performance problem.

Another case of when you may want to run ADDM manually is if you want to analyze a longer time period than one ADDM analysis period. For example, you may want to analyze the database performance in a full workday by analyzing 8 consecutive hours of database activity. One way to do this is to analyze each of the individual ADDM analysis within this 8-hour period. However, this may become complicated if there are performance problems that exist for only part of the 8-hour period, because they may appear in only some of the individual ADDM analysis. Alternatively, you can run ADDM manually using a pair of AWR snapshots that encompass the 8-hour period, and ADDM will identify the most critical performance problems in the entire time period.

This chapter contains the following sections:

- Manually Running ADDM to Analyze Current Database Performance
- Manually Running ADDM to Analyze Historical Database Performance
- Accessing Previous ADDM Results

Manually Running ADDM to Analyze Current Database Performance

Although ADDM runs every hour by default to analyze snapshots taken by the AWR during that period, it is possible to run ADDM manually to analyze current database performance. This can be very useful if you notice a drop in current performance level, or a sudden spike in database activity on the Database Performance page, as described in Chapter 4, "Monitoring Real-Time Database Performance".

When you run ADDM manually, a manual AWR snapshot is created automatically. This may impact the ADDM run cycle. For example, if you scheduled ADDM to run hourly at the start of each hour and the last ADDM run was at 8:00 p.m., running ADDM manually at 8:30 p.m. will cause the next scheduled ADDM run to start at 9:30

p.m., not 9:00 p.m. All subsequent ADDM run will continue on the new run cycle, occurring hourly at the half-hour instead of the start of each hour.

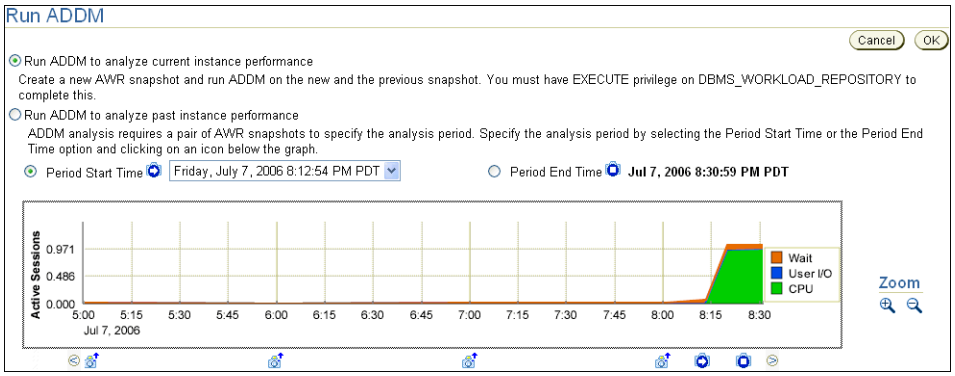
To analyze current database performance by manually running ADDM:

1. On the Database Home page, under Related Links, click **Advisor Central**.

   The Advisor Central page appears.

2. Under Advisors, click **ADDM**.

   The Run ADDM page appears.



   In this example, CPU usage spiked in the last 10 minutes.

3. Select **Run ADDM to analyze current instance performance** and click **OK**.

   The Confirmation page appears.

4. On the Confirmation page, click **Yes**.

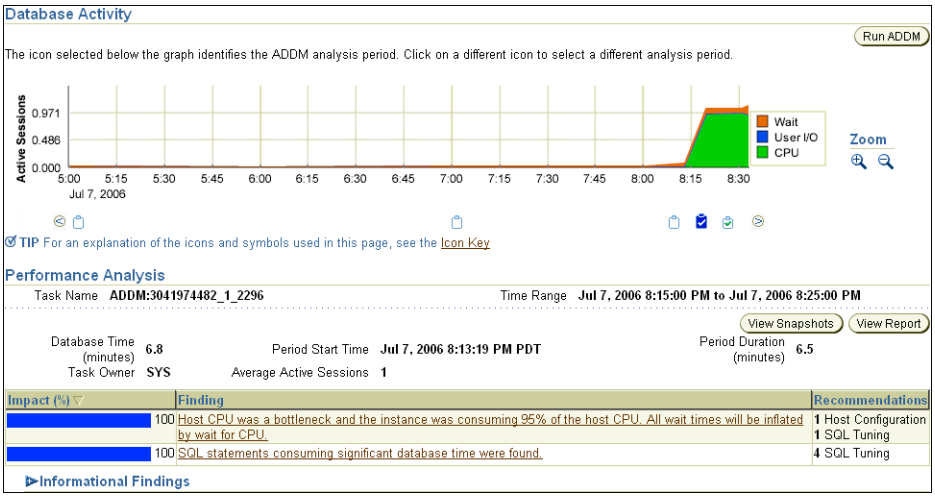
   The Processing: Run ADDM Now page appears while a new AWR snapshot is taken, and an ADDM run is performed on the time period between the new and the previous snapshot.

5. After ADDM completes the analysis, the Automatic Database Diagnostic Monitor (ADDM) page appears with the results of the ADDM run, as shown on Figure 6–1.

   The results of the ADDM task can also be viewed in a report that can be saved for later access. To view the ADDM report, click **View Report**.

   For information about reviewing ADDM results, see "Reviewing the Automatic Database Diagnostics Monitor Analysis" on page 3-5.

Figure 6–1 Analyzing Current Database Performance



Manually Running ADDM to Analyze Historical Database Performance

You can run ADDM manually to analyze historical database performance by selecting a pair or range of AWR snapshots as the analysis period. This is useful when you have identified a specific time period in the past when the database performance was poor.

You can monitor historical database performance using the Database Performance page by selecting **Historical** from the View Data list. In Historical view, you can monitor database performance in the past, up to the duration defined by the AWR retention period. If you notice any performance degradation, you can drill down to appropriate pages from the Database Performance page to identify historical performance problems with the database, as described in Chapter 4, "Monitoring Real-Time Database Performance". If a historical performance problem is identified, you can choose to run ADDM manually to analyze that particular time period.

To analyze historical database performance by manually running ADDM:

1. On the Database Home page, under Related Links, click **Advisor Central**.

   The Advisor Central page appears.

2. Under Advisors, click **ADDM**.

   The Run ADDM page appears.

3. Select **Run ADDM to analyze past instance performance**.

4. Specify a time period for analysis by selecting a pair of AWR snapshots.

   To select snapshots:

   a. Select **Period Start Time**.

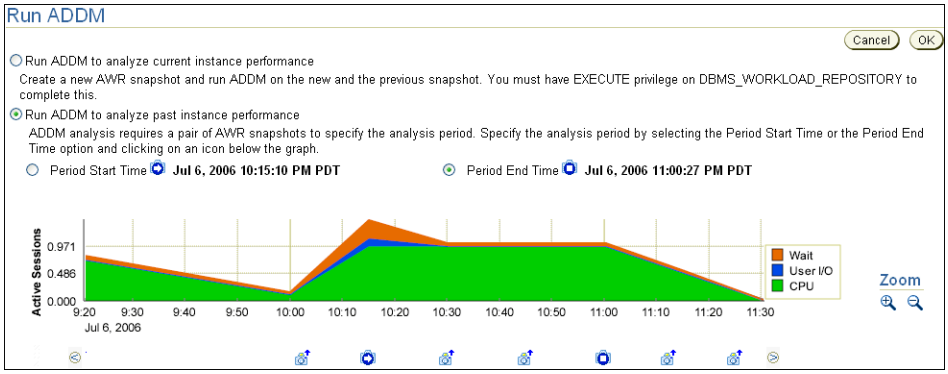
   b. Below the graph for the starting snapshot, click the snapshot you want to use for the start time.

      A play icon (displayed as an arrow) appears over the snapshot icon. In this example, database activity peaked from 10:15 p.m. to 11:00 p.m., so the snapshot taken at 10:15 p.m. is selected for the start time.

   c. Select **Period End Time**.

d. Below the graph for the ending snapshot, click the snapshot you want to use for the end time.

A stop icon (displayed as a square) appears over the snapshot icon. In this example, the snapshot taken at 11:00 p.m. is selected.

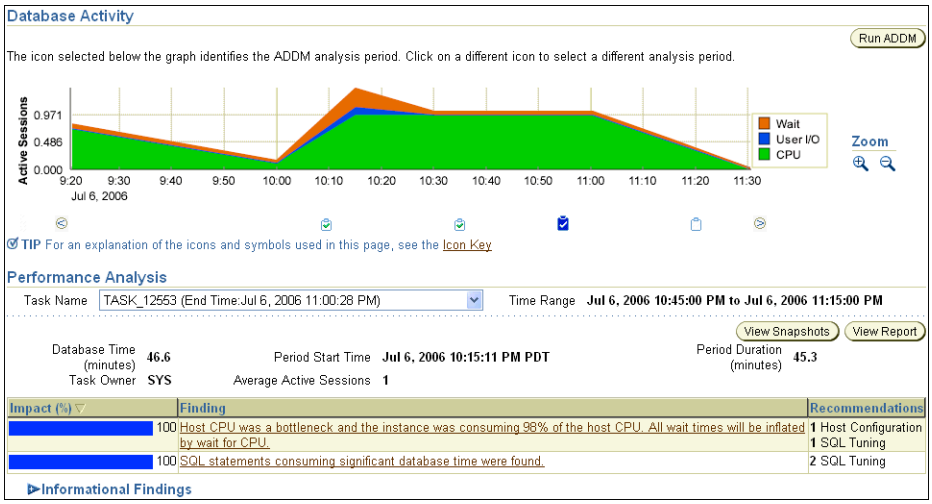


5. Click **OK**.
6. After ADDM completes the analysis, the Automatic Database Diagnostic Monitor (ADDM) page appears with the results of the ADDM run, as shown on Figure 6–2.

   The results of the ADDM task can also be viewed in a report that can be saved for later access. To view the ADDM report, click **View Report**.

   For information about reviewing ADDM results, see "Reviewing the Automatic Database Diagnostics Monitor Analysis" on page 3-5.

Figure 6–2 Analyzing Historical Database Performance



Accessing Previous ADDM Results

If you ran ADDM manually to analyze current or historical database performance, the results are displayed on the Automatic Database Diagnostic Monitor (ADDM) page after the ADDM run has completed.

You can access the ADDM results at a later time, or access the ADDM results from previous executions.

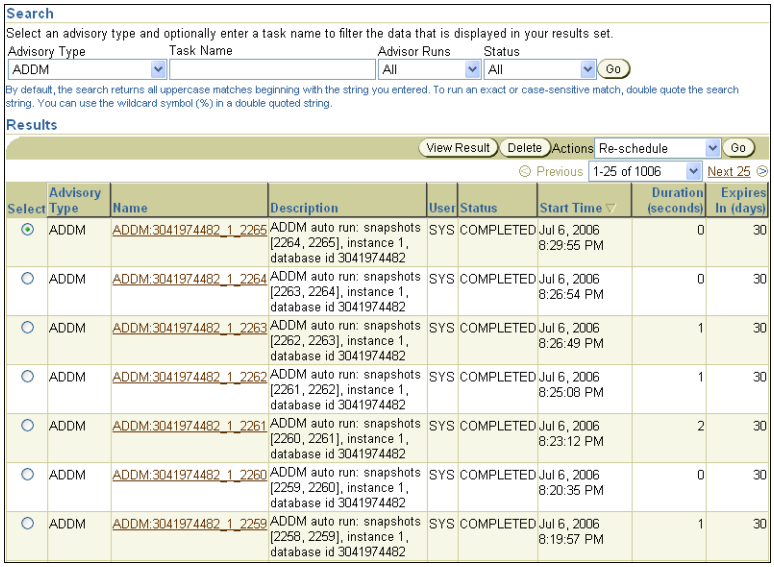
To access the ADDM results:

1. On the Database Home page, under Related Links, click **Advisor Central**.

   The Advisor Central page appears.

2. Under Advisor Tasks, select **ADDM** for Advisory Type and the appropriate search criteria and click **Go**.

   For example, you may want to select **All** for Advisor Runs to view all ADDM tasks.



Select	Advisory Type	Name	Description	User	Status	Start Time	Duration (seconds)	Expires In (days)
◉	ADDM	ADDM:3041974482_1_2265	ADDM auto run: snapshots [2264, 2265], instance 1, database id 3041974482	SYS	COMPLETED	Jul 6, 2006 8:29:55 PM	0	30
○	ADDM	ADDM:3041974482_1_2264	ADDM auto run: snapshots [2263, 2264], instance 1, database id 3041974482	SYS	COMPLETED	Jul 6, 2006 8:26:54 PM	0	30
○	ADDM	ADDM:3041974482_1_2263	ADDM auto run: snapshots [2262, 2263], instance 1, database id 3041974482	SYS	COMPLETED	Jul 6, 2006 8:26:49 PM	1	30
○	ADDM	ADDM:3041974482_1_2262	ADDM auto run: snapshots [2261, 2262], instance 1, database id 3041974482	SYS	COMPLETED	Jul 6, 2006 8:25:08 PM	1	30
○	ADDM	ADDM:3041974482_1_2261	ADDM auto run: snapshots [2260, 2261], instance 1, database id 3041974482	SYS	COMPLETED	Jul 6, 2006 8:23:12 PM	2	30
○	ADDM	ADDM:3041974482_1_2260	ADDM auto run: snapshots [2259, 2260], instance 1, database id 3041974482	SYS	COMPLETED	Jul 6, 2006 8:20:35 PM	0	30
○	ADDM	ADDM:3041974482_1_2259	ADDM auto run: snapshots [2258, 2259], instance 1, database id 3041974482	SYS	COMPLETED	Jul 6, 2006 8:19:57 PM	1	30

3. The ADDM tasks are displayed under Results. To view an ADDM result, select the desired ADDM task and click **View Result**.

   The results from the selected ADDM task are displayed in the Automatic Database Diagnostic Monitor (ADDM) page.

See Also:

- "Reviewing the Automatic Database Diagnostics Monitor Analysis" on page 3-5

7

Resolving Transient Performance Problems

Transient performance problems are short-lived and do not appear in the Automatic Database Diagnostics Monitor (ADDM) analysis. ADDM tries to report the most significant performance problems during an analysis period in terms of their impact on DB time. If a particular problem lasts for a very short duration, its severity might be averaged out or minimized by other performance problems in the entire analysis period; therefore, the problem may not appear in the ADDM findings. Whether or not a performance problem is captured by ADDM depends on its duration compared to the interval between the Automatic Workload Repository (AWR) snapshots.

If a performance problem lasts for a significant portion of the time between snapshots, it will be captured by ADDM. For example, if the snapshot interval is set to one hour, a performance problem that lasts for 30 minutes should not be considered as a transient performance problem because its duration represents a significant portion of the snapshot interval and will likely be captured by ADDM.

On the other hand, a performance problem that lasts for only 2 minutes could be a transient performance problem because its duration represents a small portion of the snapshot interval and will likely not show up in the ADDM findings. For example, if the user notifies you that the system was slow between 10:00 p.m. and 10:10 p.m., but the ADDM analysis for the time period between 10:00 p.m. and 11:00 p.m. does not show a performance problem, it is likely that a transient performance problem occurred that lasted for only a few minutes of the 10-minute interval reported by the user. This chapter describes how to resolve these types of transient performance problems with Oracle Database.

This chapter contains the following sections:

Overview of Active Session History

In order to capture a detailed history of database activity, Oracle Database samples active sessions each second using the Active Session History (ASH) sampler. The sampled data is collected into memory and written to persistent storage by the Automatic Workload Repository (AWR) snapshot processing. ASH is an integral part of the Oracle Database self-management framework and is extremely useful for diagnosing performance problems.

Unlike the instance-level statistics gathered by the AWR, sampled data is gathered at the session level by ASH. By capturing statistics for only active sessions, a manageable

set of data is represented, with the size being directly related to the work being performed rather than the entire instance.

Sampled data captured by ASH can be rolled up based on the various dimensions that it captures, including:

- SQL identifier of a SQL statement
- Object number, file number, and block number
- Wait event identifier and parameters
- Session identifier and session serial number
- Module and action name
- Client identifier of the session
- Service hash identifier

You can run Active Session History (ASH) reports to analyze transient performance problems with the database that only occur during specific times. This is especially useful when trying to:

- Resolve transient performance problems that may last for only a short period of time, such as why a particular job or session is not responding when the rest of the instance is performing normally
- Perform scoped or targeted performance analysis by various dimensions or their combinations, such as time, session, module, action, or SQL identifier

See Also:

- "Active Session History Statistics" on page 2-3

Running Active Session History Reports

This section describes how to run ASH reports.

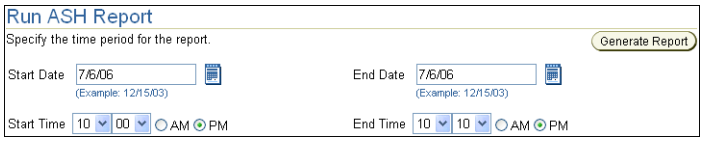
To run ASH reports:

1. On the Database Performance page, under Average Active Sessions, click **Run ASH Report**.

   The Run ASH Report page appears.

2. Enter the start date and time, and the end date and time, of the time period when the transient performance problem occurred.

   In this example, database activity spiked between 10:00 p.m. and 10:10 p.m., so an ASH report needs to be created for that time period.

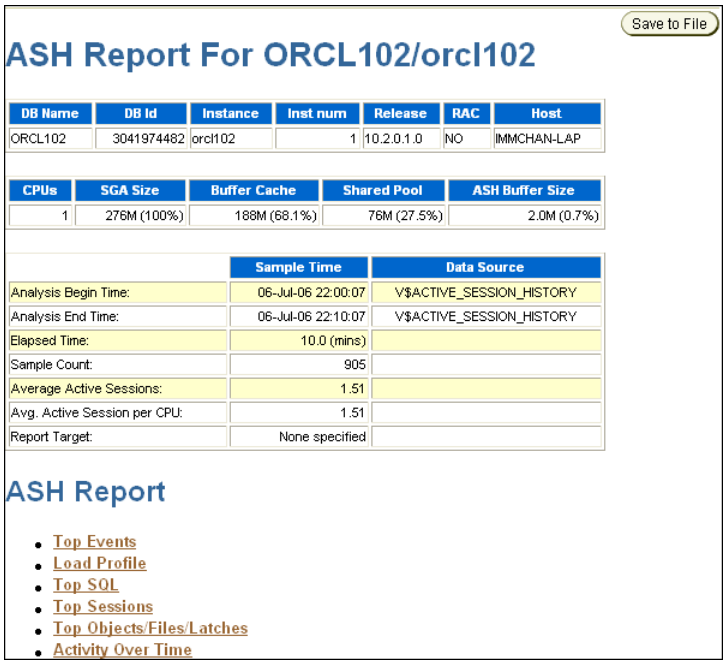


3. Click **Generate Report**.

   The Processing: View Report page appears while the report is being generated.

4. After the report is generated, the ASH report appears under Report Results on the Run ASH Report page, as shown in Figure 7–1. To save the report in HTML for future analysis, click **Save to File**.

Figure 7–1 ASH Report Results


Save to File

ASH Report For ORCL102/orcl102

DB Name	DB Id	Instance	Inst num	Release	RAC	Host
ORCL102	3041974482	orcl102	1	10.2.0.1.0	NO	IMMCHAN-LAP

CPUs	SGA Size	Buffer Cache	Shared Pool	ASH Buffer Size
1	276M (100%)	188M (68.1%)	76M (27.5%)	2.0M (0.7%)

	Sample Time	Data Source
Analysis Begin Time:	06-Jul-06 22:00:07	V$ACTIVE_SESSION_HISTORY
Analysis End Time:	06-Jul-06 22:10:07	V$ACTIVE_SESSION_HISTORY
Elapsed Time:	10.0 (mins)	
Sample Count:	905	
Average Active Sessions:	1.51	
Avg. Active Session per CPU:	1.51	
Report Target:	None specified	

ASH Report

- Top Events
- Load Profile
- Top SQL
- Top Sessions
- Top Objects/Files/Latches
- Activity Over Time

Using Active Session History Reports

To use an ASH report to analyze transient performance problems, you need to review the contents of the ASH report to identify the problem.

The contents of the ASH report are divided into the following sections:

- Top Events
- Load Profiles
- Top SQL
- Top Sessions
- Top Objects/Files/Latches
- Activity Over Time

Top Events

The Top Events section describes the top wait events of the sampled session activity categorized by user, background, and priority. Use the information in this section to identify the wait events that may be the cause of the transient performance problem.

The Top Events section contains the following subsections:

- Top User Events
- Top Background Events
- Top Event P1/P2/P3 Values

Top User Events

The Top User Events subsection lists the top wait events from user processes that accounted for the highest percentages of sampled session activity.

The example in Figure 7–2 shows that a high percentage of database activity (73 percent) is consumed by the `CPU + Wait for CPU` event. In this example, the Load Profiles section should be examined next to determine the type of activity that is causing this wait event.

Figure 7–2 Top User Events

Event	Event Class	% Activity	Avg Active Sessions
CPU + Wait for CPU	CPU	73.37	1.11
db file sequential read	User I/O	12.82	0.19
db file scattered read	User I/O	2.21	0.03

Top Background Events

The Top Background Events subsection lists the top wait events from backgrounds that accounted for the highest percentages of sampled session activity.

Top Event P1/P2/P3 Values

The Top Event P1/P2/P3 subsection lists the wait event parameter values of the top wait events that accounted for the highest percentages of sampled session activity, ordered by the percentage of total wait time (% Event). For each wait event, values in the P1 Value, P2 Value, P3 Value column correspond to wait event parameters displayed in the Parameter 1, Parameter 2, and Parameter 3 columns.

For example, in Figure 7–3, `db file sequential read` is the top wait event, consuming 13.26 percent of total wait time while the session waits for a sequential read from the database to be performed.

Figure 7–3 Top Event P1/P2/P3 Values

Event	% Event	P1 Value, P2 Value, P3 Value	% Activity	Parameter 1	Parameter 2	Parameter 3
db file sequential read	13.26	"1","1801","1"	0.11	file#	block#	blocks
control file sequential read	3.20	"0","1","1"	0.99	file#	block#	blocks
db file scattered read	2.21	"1","1896","7"	0.11	file#	block#	blocks
db file parallel write	1.44	"1","0","2147483647"	1.44	requests	interrupt	timeout

The wait event has the following parameters: `file#` (P1), `block#` (P2), and `blocks` (P3). The corresponding values for these wait event parameters are displayed in the P1 Value, P2 Value, P3 Value column:

- `file#` = "1"
- `block#` = "1801"
- `block` = "1"

Using the parameter values in this example, it can be determined that 1801 is the block number of the single block for which the session is waiting.

See Also:

- *Oracle Database Reference* for information about common wait event parameters

Load Profiles

The Load Profile section describes the load analyzed in the sampled session activity. Use the information in this section to identify the service, client, or SQL command type that may be the cause of the transient performance problem.

The Load Profile section contains the following subsections:

- Top Service/Module
- Top Client IDs
- Top SQL Command Types

Top Service/Module

The Top Service/Module subsection lists the services and modules that accounted for the highest percentages of sampled session activity.

The example in Figure 7–4 shows that the majority of database activity (65 percent) is consumed by the SYS$USERS service running the SQL*Plus module. In this example, it appears that the user is running a high-load SQL statement that is causing the performance problem. The Top SQL Command Types subsection should be analyzed next to determine if a particular type of SQL statement makes up the load.

Figure 7–4 Top Service/Module

Service	Module	% Activity	Action	% Action
SYS$USERS	SQL*Plus	65.75	UNNAMED	65.75
	DBMS_SCHEDULER	21.77	GATHER_STATS_JOB	21.22
SYS$BACKGROUND	UNNAMED	5.41	UNNAMED	5.41
	MMON_SLAVE	3.98	Auto-Flush Slave Action	3.43
SYS$USERS	Admin Connection	1.33	UNNAMED	1.33

See Also:

- "Monitoring Top Services" on page 4-5 for information about services
- "Monitoring Top Modules" on page 4-6 for information about modules
- "Monitoring Top Actions" on page 4-6 for information about actions

Top Client IDs

The Top Client IDs subsection lists the clients that accounted for the highest percentages of sampled session activity based on their client ID, which is the application-specific identifier of the database session.

Top SQL Command Types

The Top SQL Command Types subsection lists the SQL command types (such as `SELECT`, `UPDATE`, `INSERT`, and `DELETE`) that accounted for the highest percentages of sampled session activity.

The example in Figure 7–5 shows that the majority of database activity (68 percent) is used by the SQL command type `SELECT`. From this information, it appears that SQL statements causing the performance problem are `SELECT` statements. The Top SQL section should be analyzed next to identify the high-load SQL statement.

Figure 7–5 Top SQL Command Types

SQL Command Type	Distinct SQLIDs	% Activity	Avg Active Sessions
SELECT	9	67.96	1.03

Top SQL

The Top SQL section describes the top SQL statements of the sampled session activity. Use this information to identify high-load SQL statements that may be the cause of the transient performance problem.

The Top SQL section contains the following subsections:

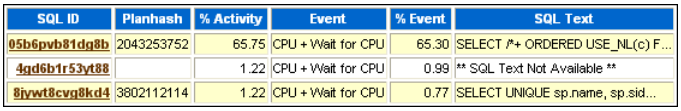
- Top SQL Statements
- Top SQL Using Literals
- Complete List of SQL Text

Top SQL Statements

The Top SQL Statements subsection lists the SQL statements that accounted for the highest percentages of sampled session activity. To view the text of the Top SQL statements displayed, click the link in the SQL ID column of the SQL statement that you want to view.

The example in Figure 7–6 shows that the majority of database activity (65 percent) is used by a particular SQL `SELECT` statement. Based on this information, it appears that this is the high-load SQL statement causing the performance problem. The Top Sessions section should be analyzed next to identify the session running this SQL statement.

Figure 7–6 Top SQL Statements



SQL ID	Planhash	% Activity	Event	% Event	SQL Text
05b6pvb81dg8b	2043253752	65.75	CPU + Wait for CPU	65.30	SELECT /*+ ORDERED USE_NL(c) F...
4qd6b1r53yt88		1.22	CPU + Wait for CPU	0.99	** SQL Text Not Available **
8jywt8cvq8kd4	3802112114	1.22	CPU + Wait for CPU	0.77	SELECT UNIQUE sp.name, sp.sid...

See Also:

- "Monitoring Top SQL" on page 4-4

Top SQL Using Literals

The Top SQL Statements subsection lists the SQL statements using literals that accounted for the highest percentages of sampled session activity.

Complete List of SQL Text

The Complete List of SQL Text subsection displays the entire text of the Top SQL statements shown in this section.

Top Sessions

The Top Sessions section describes the sessions that were waiting for a particular wait event. Use this information to identify the sessions that accounted for the highest percentages of sampled session activity, which may be the cause of the transient performance problem.

The Top Sessions section contains the following subsections:

- Top Sessions
- Top Blocking Sessions
- Top Sessions Running PQs

Top Sessions

The Top Session subsection lists the sessions that were waiting for a particular wait event that accounted for the highest percentages of sampled session activity.

The example in Figure 7–7 shows that the majority of database activity (65 percent) is used by the user `ICHAN` with the session ID of 135. From this information, it appears that this is the user running the high-load SQL statement identified earlier. You should investigate this session to determine whether it is performing a legitimate operation and tune the SQL statement if possible. If tuning the SQL statement is not possible and the session is causing an unacceptable performance impact on the system, you may want to consider terminating the session.

Figure 7–7 Top Sessions

Sid, Serial#	% Activity	Event	% Event	User	Program	# Samples Active	XIDs
135, 3909	65.75	CPU + Wait for CPU	65.30	ICHAN	sqlplusw.exe	591/600 [99%]	0
146, 4168	21.22	db file sequential read	12.60	SYS	ORACLE.EXE (J001)	114/600 [19%]	17
		CPU + Wait for CPU	5.75			52/600 [9%]	15
		db file scattered read	2.10			19/600 [3%]	3
136, 195	3.43	control file sequential read	2.10	SYS	ORACLE.EXE (m000)	19/600 [3%]	0
167, 1	2.65	db file parallel write	1.44	SYS	ORACLE.EXE (DBW0)	13/600 [2%]	0
		CPU + Wait for CPU	1.22			11/600 [2%]	0
143, 560	1.33	CPU + Wait for CPU	0.77	SYS	OMS	7/600 [1%]	0

See Also:

- "Monitoring Top Sessions" on page 4-4 for information about sessions
- Chapter 10, "Tuning SQL Statements" for information about tuning SQL statements

Top Blocking Sessions

The Top Blocking Sessions subsection lists the blocking sessions that accounted for the highest percentages of sampled session activity.

Top Sessions Running PQs

The Top Sessions Running PQs subsection lists the sessions running parallel queries (PQs) that were waiting for a particular wait event which accounted for the highest percentages of sampled session activity.

Top Objects/Files/Latches

The Top Objects/Files/Latches section provides additional information about the most commonly-used database resources and contains the following subsections:

- Top DB Objects
- Top DB Files
- Top Latches

Top DB Objects

The Top DB Objects subsection lists the database objects (such as tables and indexes) that accounted for the highest percentages of sampled session activity.

Top DB Files

The Top DB Files subsection lists the database files that accounted for the highest percentages of sampled session activity.

Top Latches

The Top Latches subsection lists the latches that accounted for the highest percentages of sampled session activity.

Latches are simple, low-level serialization mechanisms to protect shared data structures in the system global area (SGA). For example, latches protect the list of users currently accessing the database and the data structures describing the blocks in the buffer cache. A server or background process acquires a latch for a very short time while manipulating or looking at one of these structures. The implementation of latches is operating system dependent, particularly in regard to whether and how long a process will wait for a latch.

See Also:

- *Oracle Database Performance Tuning Guide* for information about latches

Activity Over Time

The Activity Over Time section is one of the most informative sections of the ASH report. This section is particularly useful for longer time periods because it provides in-depth details about activities and workload profiles during the analysis period. The Activity Over Time section is divided into multiple time slots, as shown in Figure 7–8.

Figure 7–8 Activity Over Time

Slot Time (Duration)	Slot Count	Event	Event Count	% Event
22:00:07 (53 secs)	157	CPU + Wait for CPU	91	10.06
		db file sequential read	26	2.87
		control file sequential read	22	2.43
22:01:00 (1.0 min)	126	CPU + Wait for CPU	84	9.28
		db file sequential read	22	2.43
		db file scattered read	9	0.99
22:02:00 (1.0 min)	122	CPU + Wait for CPU	60	6.63
		db file sequential read	59	6.52
		db file parallel write	1	0.11
22:03:00 (1.0 min)	89	CPU + Wait for CPU	72	7.96
		db file sequential read	12	1.33
		db file scattered read	3	0.33
22:04:00 (1.0 min)	62	CPU + Wait for CPU	59	6.52
		db file parallel write	1	0.11
		db file sequential read	1	0.11
22:05:00 (1.0 min)	94	CPU + Wait for CPU	80	8.84
		db file parallel write	8	0.88
		direct path read temp	3	0.33
22:06:00 (1.0 min)	63	CPU + Wait for CPU	60	6.63
		db file parallel write	2	0.22
		control file sequential read	1	0.11
22:07:00 (1.0 min)	63	CPU + Wait for CPU	63	6.96
22:08:00 (1.0 min)	60	CPU + Wait for CPU	59	6.52
		os thread startup	1	0.11
22:09:00 (1.0 min)	61	CPU + Wait for CPU	60	6.63
		os thread startup	1	0.11
22:10:00 (7 secs)	8	CPU + Wait for CPU	7	0.77
		control file sequential read	1	0.11

Each of the time slots contains information regarding that particular time slot, as described in Table 7–1.

Table 7–1 Activity Over Time

Column	Description
Slot Time (Duration)	Duration of the slot
Slot Count	Number of sampled sessions in the slot
Event	Top three wait events in the slot
Event Count	Number of ASH samples waiting for the wait event
% Event	Percentage of ASH samples waiting for wait events in the entire analysis period

The first and last slots are usually odd-sized. All inner slots and are 1 minute each, and can be compared to each other.

When comparing the inner slots, perform a skew analysis by identifying spikes in the Event Count and Slot Count columns. A spike in the Event Count column indicates an increase in the number of sampled sessions waiting for a particular event. A spike in the Slot Count column indicates an increase in active sessions, because ASH data is sampled from active sessions only and a relative increase in database workload. Typically, when the number of active session samples and the number of sessions associated with a wait event increases, the slot may be the cause of the transient performance problem.

8

Resolving Performance Degradation Over Time

This chapter describes how to resolve performance degradation over time with Oracle Database. Performance degradation of the database over time happens when your database was performing optimally in the past, such as 6 months ago, but has gradually degraded to a point where it becomes noticeable to the users.

The Automatic Workload Repository (AWR) Compare Periods report enables you to compare database performance between two periods of time. While an AWR report shows AWR data between two snapshots (or two points in time), the AWR Compare Periods report shows the difference between two periods (or two AWR reports, which equates to four snapshots).

Using the AWR Compare Periods report helps you to identify detailed performance attributes and configuration settings that differ between two time periods. For example, if the application workload is known to be stable between 10:00 p.m. and midnight every night, but the performance on a particular Thursday was poor between 10:00 p.m. and 11:00 p.m., generating an AWR Compare Periods report for Thursday from 10:00 p.m. to 11:00 p.m. and Wednesday from 10:00 p.m. to 11:00 p.m. should identify configuration settings, workload profile, and statistics that were different in these two time periods. Based on the differences identified, the cause of the performance degradation can be more easily diagnosed. The two time periods selected for the AWR Compare Periods Report can be of different durations, because the report normalizes the statistics by the amount of time spent on the database for each time period, and presents statistical data ordered by the largest difference between the periods.

This chapter contains the following sections:

See Also:

- "Gathering Database Statistics Using the Automatic Workload Repository" on page 2-1

Creating Baselines

Baselines are an effective way to diagnose performance problems. AWR supports the capture of baseline data by enabling you to specify and preserve a pair or a range of

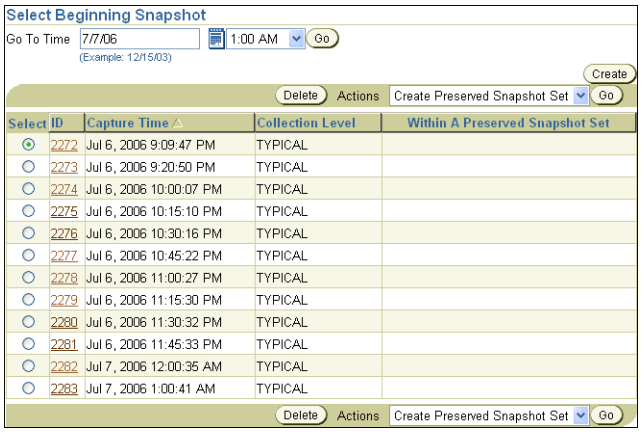
snapshots as a baseline. The snapshots contained in a baseline are excluded from the automatic AWR purging process and are retained indefinitely.

Carefully consider the time period you choose as a baseline, because the baseline should represent the system operating at an optimal level. In the future, you can compare these baselines with the snapshots captured during periods of poor performance to analyze performance degradation over time.

To create baselines:

1. On the Database Performance page, under Additional Monitoring Links, click **Snapshots**.

   The Snapshots page appears with a list of the most recent snapshots.

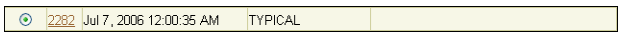


2. To filter the snapshots and only display the snapshots taken at the start of the desired baseline, select the time for the starting snapshot in the Go To Time field and click **Go**.

   In this example, 12:00AM on July 7, 2006 is selected.

3. Select the starting snapshot for the baseline.

   In this example, snapshot 2282 is selected.



4. From the Actions list, select **Create Preserved Snapshot Set** and click **Go**.

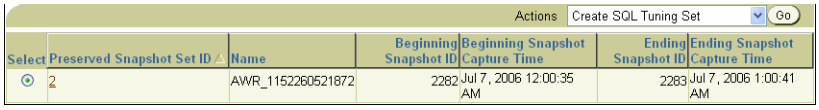
   The Create Preserved Snapshot Set page appears with a list of subsequent snapshots displayed. Note that a system generated value is assigned in the Preserved Snapshot Name field.

5. Under Select Ending Snapshot, select the ending snapshot for the baseline and click **OK**.

   In this example, snapshot 2283 is selected.

6. After the preserved snapshot set is taken, the Preserved Snapshot Sets page appears with a Confirmation message.

   In this example, the preserved snapshot set that was created has a Preserved Snapshot Set ID of 2 and contains snapshots 2282 through 2283.

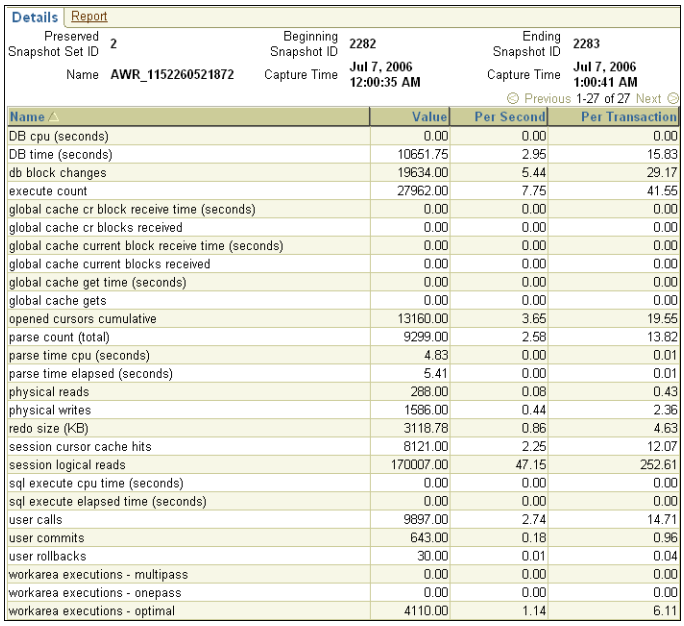


Actions Create SQL Tuning Set Go

Select	Preserved Snapshot Set ID	Name	Beginning Snapshot ID	Beginning Snapshot Capture Time	Ending Snapshot ID	Ending Snapshot Capture Time
⊙	2	AWR_1152260521872	2282	Jul 7, 2006 12:00:35 AM	2283	Jul 7, 2006 1:00:41 AM

7. The preserved snapshot set that was created can now be used as a baseline for comparison to other snapshots when performance problems occur. To view the statistics gathered for this baseline, click the **Preserved Snapshot Set ID** link.

   The Preserved Snapshot Set Details page appears with the statistics gathered for the baseline displayed.



Details | Report

Preserved Snapshot Set ID: 2 | Beginning Snapshot ID: 2282 | Ending Snapshot ID: 2283

Name: AWR_1152260521872 | Capture Time: Jul 7, 2006 12:00:35 AM | Capture Time: Jul 7, 2006 1:00:41 AM

Previous 1-27 of 27 Next

Name	Value	Per Second	Per Transaction
DB cpu (seconds)	0.00	0.00	0.00
DB time (seconds)	10651.75	2.95	15.83
db block changes	19634.00	5.44	29.17
execute count	27962.00	7.75	41.55
global cache cr block receive time (seconds)	0.00	0.00	0.00
global cache cr blocks received	0.00	0.00	0.00
global cache current block receive time (seconds)	0.00	0.00	0.00
global cache current blocks received	0.00	0.00	0.00
global cache get time (seconds)	0.00	0.00	0.00
global cache gets	0.00	0.00	0.00
opened cursors cumulative	13160.00	3.65	19.55
parse count (total)	9299.00	2.58	13.82
parse time cpu (seconds)	4.83	0.00	0.01
parse time elapsed (seconds)	5.41	0.00	0.01
physical reads	288.00	0.08	0.43
physical writes	1586.00	0.44	2.36
redo size (KB)	3118.78	0.86	4.63
session cursor cache hits	8121.00	2.25	12.07
session logical reads	170007.00	47.15	252.61
sql execute cpu time (seconds)	0.00	0.00	0.00
sql execute elapsed time (seconds)	0.00	0.00	0.00
user calls	9897.00	2.74	14.71
user commits	643.00	0.18	0.96
user rollbacks	30.00	0.01	0.04
workarea executions - multipass	0.00	0.00	0.00
workarea executions - onepass	0.00	0.00	0.00
workarea executions - optimal	4110.00	1.14	6.11

Running the Automatic Workload Repository Compare Periods Reports

This section describes how to run the AWR Compare Periods reports using Oracle Enterprise Manager.

You can use AWR Compare Periods reports to compare the database performance between two time periods by:

- Comparing a Baseline to Another Baseline or Pair of Snapshots
- Comparing Two Pairs of Snapshots

Comparing a Baseline to Another Baseline or Pair of Snapshots

When performance degradation happens to a database over time, you should run the AWR Compare Periods report to compare the degraded performance, captured as a new baseline or a pair of snapshots, to an existing baseline that represents a time when the system was operating at an optimal level. To do so, you will need to have preserved a baseline that represents the system operating at an optimal level. If an

existing baseline is not available, you can compare database performance between two periods of time by using two arbitrary pairs of snapshots, as described in "Comparing Two Pairs of Snapshots" on page 8-6.

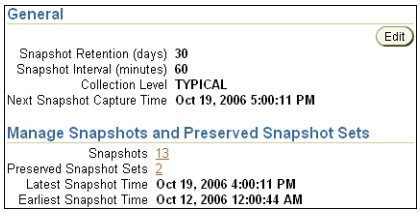
See Also:

- "Creating Baselines" on page 8-1

To compare a baseline to another baseline:

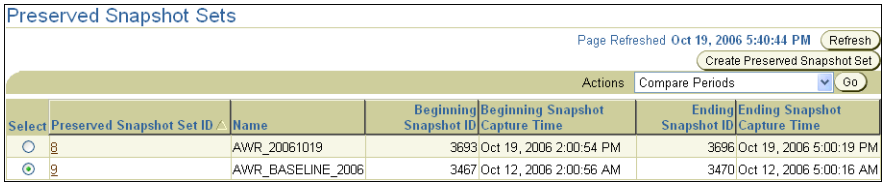
1. On the Database Administration page, under Statistics Management, click **Automatic Workload Repository**.

   The Automatic Workload Repository page appears.



2. Under Manage Snapshots and Preserved Snapshot Sets, click the link next to Preserved Snapshot Sets.

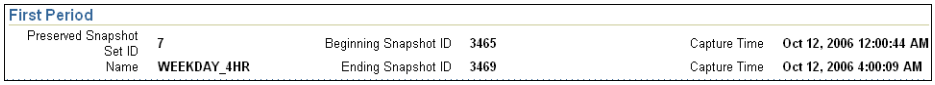
   The Preserved Snapshot Sets page appears.



3. Select the baseline you want to use for the report. At least one existing preserved snapshot set must be available to be used as a baseline.

   From the Actions list, select **Compare Periods** and click **Go**.

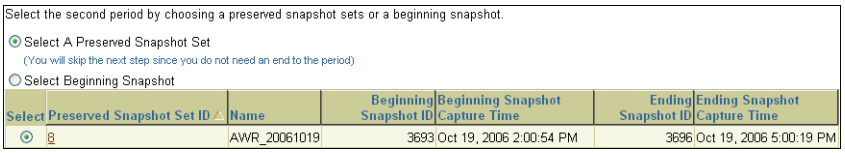
   The Compare Periods: Second Period Start page appears. Under First Period, the selected baseline is displayed. In this example, the baseline named `AWR_BASELINE_2006` is selected.



4. You can compare the baseline selected in the first period to another baseline or a pair of snapshots.

   - To compare to another baseline, select **Select a Preserved Snapshot Set** and then the baseline you want to use in the second period.

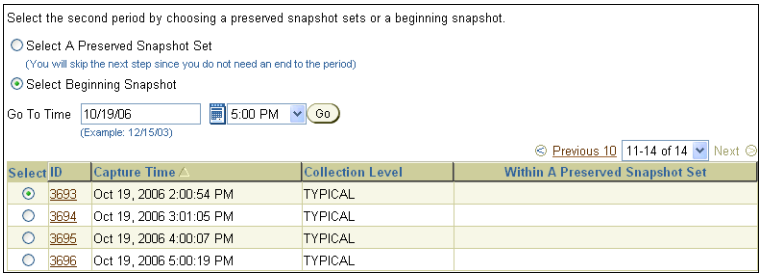
     In this example, the preserved snapshot set named `AWR_20061019` is selected.



Click **Next**. The Compare Periods: Review page appears. Proceed to Step 6.

- To compare to a pair of snapshots, select **Select Beginning Snapshot** and then the beginning snapshot you to use in the second period.

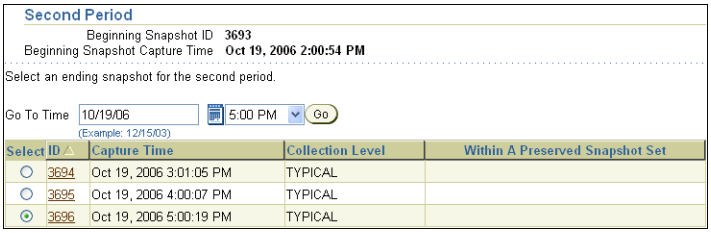
  In this example, snapshot 3693, taken on October 19, 2006 at 2:00 p.m., is selected.



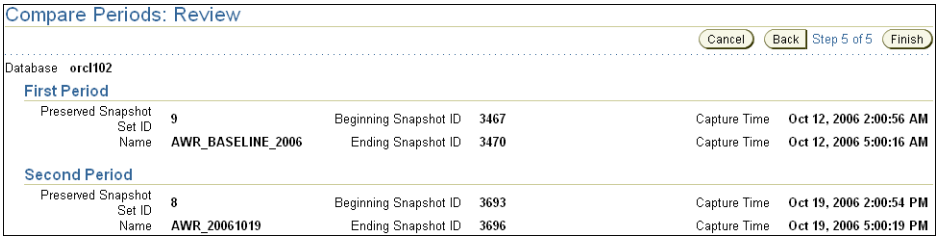
Click **Next**. The Compare Periods: Second Period End page appears. Continue with the next step.

5. Select the ending snapshot for the snapshot period that will be included in the report and click **Next**.

   In this example, snapshot 3696, taken on October 19, 2006 at 5:00 p.m., is selected.



The Compare Periods: Review page appears.



6. Review the selected periods that will be included in the report and click **Finish**.

   The Compare Periods: Results page appears. Data from the selected periods appears under the General subpage. Data can be viewed per second or per transaction by selecting the desired option from the View Data list.

Name	First Period Metric Ratio	Second Period Metric Ratio	First Period Value	Second Period Value	First Period Rate Per Second	Second Period Rate Per Second
DB cpu (seconds)			0.00	0.00	0.00	0.00
DB time (seconds)			31,541.46	31,565.35	2.93	2.93
db block changes			46,621.00	66,371.00	4.33	6.17
execute count			74,086.00	75,844.00	6.89	7.05
global cache cr block receive time (seconds)			0.00	0.00	0.00	0.00
global cache cr blocks received			0.00	0.00	0.00	0.00
global cache current block receive time (seconds)			0.00	0.00	0.00	0.00
global cache current blocks received			0.00	0.00	0.00	0.00
global cache get time (seconds)			0.00	0.00	0.00	0.00
global cache gets			0.00	0.00	0.00	0.00
opened cursors cumulative			36,462.00	38,036.00	3.39	3.53
parse count (total)			28,821.00	24,525.00	2.68	2.28
parse time cpu (seconds)			22.45	7.63	0.00	0.00
parse time elapsed (seconds)			22.85	8.04	0.00	0.00
physical reads			89.00	93.00	0.01	0.01
physical writes			4,788.00	5,423.00	0.45	0.50
redo size (KB)			7,718.69	10,489.71	0.72	0.97
session cursor cache hits			22,485.00	24,683.00	2.09	2.29
session logical reads			375,588.00	396,673.00	34.91	36.85
sql execute cpu time (seconds)			0.00	0.00	0.00	0.00
sql execute elapsed time (seconds)			0.00	0.00	0.00	0.00
user calls			22,704.00	24,215.00	2.11	2.25
user commits			1,750.00	1,754.00	0.16	0.16
user rollbacks			96.00	94.00	0.01	0.01
workarea executions - multipass			0.00	0.00	0.00	0.00
workarea executions - onepass			0.00	0.00	0.00	0.00
workarea executions - optimal			10,446.00	9,214.00	0.97	0.86

In this example, parse time in the first period is much higher than the second.

7. To view the report, click the **Report** tab.

   The Processing: View Report page appears while the report is being generated. After it completes, the report will appear, as shown in Figure 8–1 on page 8-9. To change periods, click **Change Periods**. To save the report as an HTML file, click **Save to File**.

See Also:

- "Using the Automatic Workload Repository Compare Periods Reports" on page 8-9

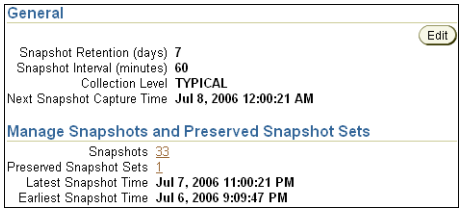
Comparing Two Pairs of Snapshots

If an existing baseline is not available, you can compare the database performance using two arbitrary pairs of snapshots, one pair taken when the database is performing optimally, and another pair when the database is performing poorly.

To compare performance using two pairs of snapshots:

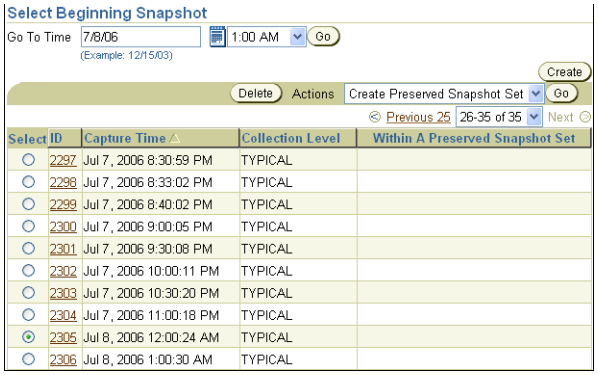
1. On the Database Administration page, under Statistics Management, click **Automatic Workload Repository**.

   The Automatic Workload Repository page appears.



2. Under Manage Snapshots and Preserved Snapshot Sets, click the link next to Snapshots.

The Snapshots page appears. At least four existing snapshots must be available.

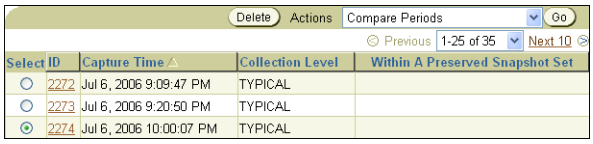


3. To filter the snapshots and display only the snapshot taken at the start of the comparison period, in the Go To Time field, select the time for the starting snapshot and click **Go**.

   In this example, 10:00 p.m. on Thursday, July 6, 2006, is selected.

4. Under Select Beginning Snapshot, select the starting point for the first snapshot period that will be included in the report.

   In this example, snapshot 2274, taken on Thursday, July 6, 2006 at 10:00 p.m., is selected.

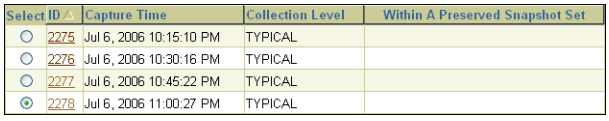


5. From the Actions list, select **Compare Periods** and click **Go**.

   The Compare Periods: First Period End page appears.

6. Select the ending point for the first snapshot period that will be included in the report and click **Next**.

   In this example, snapshot 2278, taken on Thursday, July 6, 2006 at 11:00 p.m., is selected.



   The Compare Periods: Second Period Start page appears.

7. Select the starting point for the second snapshot period that will be included in the report and click **Next**.

   In this example, snapshot 2302, taken on Friday, July 7, 2006 at 10:00 p.m., is selected.

Select	ID	Capture Time	Collection Level	Within A Preserved Snapshot Set
◉	2302	Jul 7, 2006 10:00:11 PM	TYPICAL	

The Compare Periods: Second Period End page appears.

8. Select the end point for the second period that will be included in the report and click **Next**.

   In this example, snapshot 2304, taken on Friday, July 7, 2006 at 11:00 p.m., is selected.

Select	ID	Capture Time	Collection Level	Within A Preserved Snapshot Set
○	2303	Jul 7, 2006 10:30:20 PM	TYPICAL	
◉	2304	Jul 7, 2006 11:00:18 PM	TYPICAL	

The Compare Periods: Review page appears.

First Period

Beginning Snapshot ID	**2274**	Beginning Snapshot Capture Time	**Jul 6, 2006 10:00:07 PM**
Ending Snapshot ID	**2278**	Ending Snapshot Capture Time	**Jul 6, 2006 11:00:27 PM**

Second Period

Beginning Snapshot ID	**2302**	Beginning Snapshot Capture Time	**Jul 7, 2006 10:00:11 PM**
Ending Snapshot ID	**2304**	Ending Snapshot Capture Time	**Jul 7, 2006 11:00:18 PM**

9. Review the selected periods that will be included in the report and click **Finish**.

   The Compare Periods: Results page appears. Data from the selected periods appears under the General subpage. Data can be viewed per second or per transaction by selecting the desired option from the View Data list.

Name	First Period Metric Ratio	Second Period Metric Ratio	First Period Value	Second Period Value	First Period Rate Per Second	Second Period Rate Per Second
DB cpu (seconds)			0.00	0.00	0.00	0.00
DB time (seconds)			10,852.78	10,666.03	3.00	2.96
db block changes			123,109.00	328,917.00	34.01	91.21
execute count			74,819.00	69,826.00	20.67	19.36
global cache cr block receive time (seconds)			0.00	0.00	0.00	0.00
global cache cr blocks received			0.00	0.00	0.00	0.00
global cache current block receive time (seconds)			0.00	0.00	0.00	0.00
global cache current blocks received			0.00	0.00	0.00	0.00
global cache get time (seconds)			0.00	0.00	0.00	0.00
global cache gets			0.00	0.00	0.00	0.00
opened cursors cumulative			46,670.00	41,510.00	12.89	11.51
parse count (total)			25,692.00	21,096.00	7.10	5.85
parse time cpu (seconds)			12.44	14.04	0.00	0.00
parse time elapsed (seconds)			29.01	16.97	0.01	0.00
physical reads			12,302.00	6,036.00	3.40	1.67
physical writes			8,026.00	7,905.00	2.22	2.19
redo size (KB)			20,216.50	44,704.53	5.58	12.40
session cursor cache hits			48,009.00	47,763.00	13.26	13.25
session logical reads			336,853,037.00	957,123.00	93,053.33	265.43
sql execute cpu time (seconds)			0.00	0.00	0.00	0.00
sql execute elapsed time (seconds)			0.00	0.00	0.00	0.00
user calls			9,928.00	9,395.00	2.74	2.61
user commits			915.00	844.00	0.25	0.23
user rollbacks			28.00	31.00	0.01	0.01
workarea executions - multipass			0.00	0.00	0.00	0.00
workarea executions - onepass			1.00	0.00	0.00	0.00
workarea executions - optimal			10,717.00	10,157.00	2.96	2.82

In this example, logical reads in the first period are much higher than the second.

10. To view the report, click the **Report** tab.

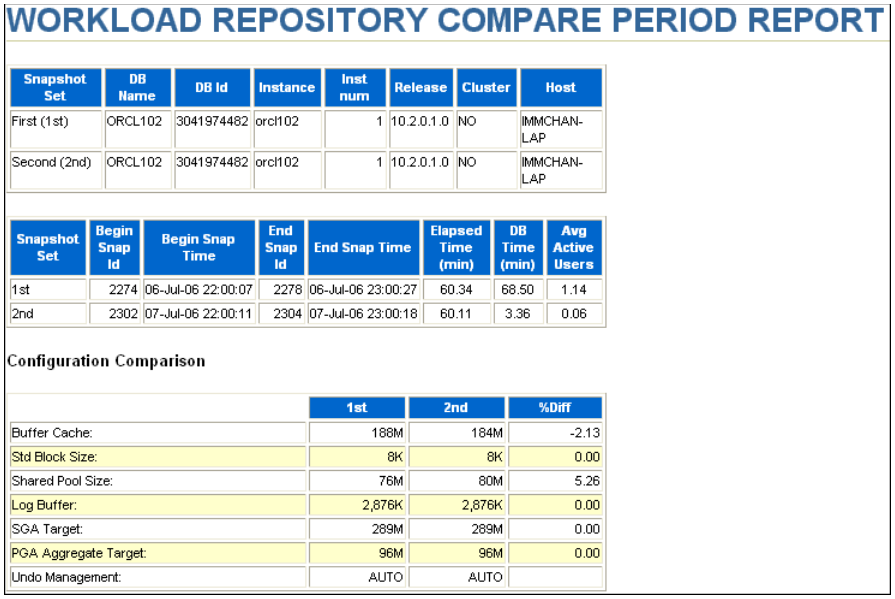
    The Processing: View Report page appears while the report is being generated. After it completes, the report will appear, as shown in Figure 8–1 on page 8-9. To change periods, click **Change Periods**. To save the report as an HTML file, click **Save to File**.

Using the Automatic Workload Repository Compare Periods Reports

After an AWR Compare Periods report is generated for the time periods you want to compare, you can use it to perform an analysis of performance degradation with Oracle Database that may have happened over time. For information about generating AWR Compare Periods reports, see "Running the Automatic Workload Repository Compare Periods Reports" on page 8-3.

Figure 8–1 shows an example of an AWR Compare Periods report.

Figure 8–1 AWR Compare Periods Report



WORKLOAD REPOSITORY COMPARE PERIOD REPORT

Snapshot Set	DB Name	DB Id	Instance	Inst num	Release	Cluster	Host
First (1st)	ORCL102	3041974482	orcl102	1	10.2.0.1.0	NO	IMMCHAN-LAP
Second (2nd)	ORCL102	3041974482	orcl102	1	10.2.0.1.0	NO	IMMCHAN-LAP

Snapshot Set	Begin Snap Id	Begin Snap Time	End Snap Id	End Snap Time	Elapsed Time (min)	DB Time (min)	Avg Active Users
1st	2274	06-Jul-06 22:00:07	2278	06-Jul-06 23:00:27	60.34	68.50	1.14
2nd	2302	07-Jul-06 22:00:11	2304	07-Jul-06 23:00:18	60.11	3.36	0.06

Configuration Comparison

	1st	2nd	%Diff
Buffer Cache:	188M	184M	-2.13
Std Block Size:	8K	8K	0.00
Shared Pool Size:	76M	80M	5.26
Log Buffer:	2,876K	2,876K	0.00
SGA Target:	289M	289M	0.00
PGA Aggregate Target:	96M	96M	0.00
Undo Management:	AUTO	AUTO	

The contents of the AWR Compare Periods report are divided into the following sections:

- Report Summary
- Wait Events
- Time Model Statistics
- Operating System Statistics
- Service Statistics
- SQL Statistics
- Instance Activity Statistics
- I/O Statistics
- Advisory Statistics
- Wait Statistics
- Latch Statistics
- Segment Statistics
- Dictionary Cache Statistics
- Library Cache Statistics

- SGA Statistics
- init.ora Parameters

Report Summary

The report summary is at the beginning of the AWR Compare Periods report, and summarizes information about the snapshot sets and loads used in the report. The report summary contains the following sections:

- Snapshot Sets
- Configuration Comparison
- Load Profile
- Top 5 Timed Events

Snapshot Sets

The Snapshot Sets section, shown in Figure 8–2, displays information about the snapshot sets used for this report, such as instance, host, and snapshot information.

Figure 8–2 Snapshot Sets

Snapshot Set	DB Name	DB Id	Instance	Inst num	Release	Cluster	Host
First (1st)	ORCL102	3041974482	orcl102	1	10.2.0.1.0	NO	IMMCHAN-LAP
Second (2nd)	ORCL102	3041974482	orcl102	1	10.2.0.1.0	NO	IMMCHAN-LAP

Snapshot Set	Begin Snap Id	Begin Snap Time	End Snap Id	End Snap Time	Elapsed Time (min)	DB Time (min)	Avg Active Users
1st	2274	06-Jul-06 22:00:07	2278	06-Jul-06 23:00:27	60.34	68.50	1.14
2nd	2302	07-Jul-06 22:00:11	2304	07-Jul-06 23:00:18	60.11	3.36	0.06

In this example, the first snapshot period corresponds to the time when performance degradation was experienced on July 6, 2006 from 10:00 p.m. to 11:00 p.m. The second snapshot period represents a time when performance was stable on July 7, 2006 from 10:00 p.m. to 11:00 p.m.

Configuration Comparison

The Configuration Comparison section compares the configurations used in the two snapshot sets. Any differences in the configurations are quantified as percentages differed in the %Diff column.

Load Profile

The Load Profile section compares the loads used in the two snapshot sets. Any differences in the loads are quantified as percentages differed in the %Diff column.

Top 5 Timed Events

The Top 5 Timed Events section, shown in Figure 8–3, displays the five events or operations that consumed the most CPU time (represented as a percentage of total DB time) in each of the snapshot sets.

Figure 8–3 Top 5 Timed Events

1st					2nd				
Event	Waits	Time(s)	Percent Total DB Time	Wait Class	Event	Waits	Time(s)	Percent Total DB Time	Wait Class
CPU time		3,546.1	86.28		CPU time		142.5	70.62	
db file sequential read	5,188	113.6	2.76	User I/O	db file sequential read	1,557	14.0	6.95	User I/O
control file sequential read	3,893	80.0	1.95	System I/O	*db file parallel write	2,857	7.3	3.64	System I/O
*os thread startup	58	52.2	1.27	Concurrency	db file scattered read	819	7.2	3.55	User I/O
db file scattered read	639	17.9	.44	User I/O	control file sequential read	2,380	4.9	2.45	System I/O
-db file parallel write	2,181	4.3	.10	System I/O	-os thread startup	53	1.1	.56	Concurrency

In this example, CPU time is much higher in the first period than in the second. Waits for the `db file sequential read` event is also significantly higher in the first period than in the second.

Wait Events

The Wait Events section, shown in Figure 8–4, compares the wait events in the two snapshot sets. The wait events are ordered based on the difference in total DB time spent on the wait event between the two snapshot sets, and are listed in descending order. Wait events at the top of this section have the greatest differential between the two snapshot sets, and may be possible causes for performance degradation over time.

Figure 8–4 Wait Events

	% DB Time			#Waits/sec (Elapsed Time)			Avg Wait Time (ms)		
Event	1st	2nd	Diff	1st	2nd	%Diff	1st	2nd	%Diff
db file sequential read	2.76	6.95	4.19	1.43	0.43	-69.93	21.89	9.01	-58.84
db file parallel write	0.10	3.64	3.53	0.60	0.79	31.67	1.97	2.57	30.46
db file scattered read	0.44	3.55	3.11	0.18	0.23	27.78	28.05	8.74	-68.84
log file parallel write	0.10	1.60	1.50	0.48	0.57	18.75	2.31	1.58	-31.60
control file parallel write	0.09	1.36	1.27	0.35	0.36	2.86	2.95	2.11	-28.47
os thread startup	1.27	0.56	-0.71	0.02	0.01	-50.00	900.24	21.34	-97.63
control file sequential read	1.95	2.45	0.50	1.08	0.66	-38.89	20.56	2.08	-89.88
log file sync	0.05	0.29	0.24	0.10	0.10	0.00	6.15	1.58	-74.31
latch: cache buffers chains	0.14	0.00	-0.14	0.07	0.00	-100.00	23.40	0.00	-100.00
direct path read temp	0.09	0.00	-0.09	0.05	0.00	-100.00	20.17	0.00	-100.00
SQL*Net break/reset to client	0.00	0.04	0.04	0.10	0.08	-20.00	0.18	0.27	50.00
LGWR wait for redo copy	0.00	0.04	0.04	0.01	0.01	0.00	3.65	3.07	-15.89
latch: shared pool	0.00	0.03	0.03	0.00	0.00	0.00	2.71	3.84	41.70
direct path write temp	0.02	0.00	-0.02	0.07	0.00	-100.00	4.20	0.00	-100.00
latch free	0.05	0.03	-0.02	0.03	0.02	-33.33	16.55	0.72	-95.65
latch: library cache	0.01	0.03	0.02	0.01	0.00	-100.00	17.75	5.05	-71.55
SQL*Net message to client	0.00	0.02	0.02	2.48	2.42	-2.42	0.00	0.00	0.00
log file sequential read	0.00	0.01	0.01	0.00	0.00	0.00	0.00	8.54	100.00
rdbms ipc reply	0.02	0.01	-0.00	0.00	0.00	0.00	89.45	5.61	-93.73
latch: session allocation	0.00	0.00	-0.00	0.00	0.00	0.00	18.47	0.04	-99.78
buffer busy waits	0.00	0.00	-0.00	0.00	0.00	0.00	28.44	0.00	-100.00
direct path write	0.00	0.00	0.00	0.03	0.00	-100.00	0.00	0.06	100.00
latch: row cache objects	0.00	0.00	-0.00	0.00	0.00	0.00	10.35	0.00	-100.00
SQL*Net more data from client	0.00	0.00	0.00	0.01	0.00	-100.00	0.04	0.08	100.00
log file single write	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.32	100.00
latch: library cache pin	0.00	0.00	-0.00	0.00	0.00	0.00	2.55	0.00	-100.00
latch: redo allocation	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.01	0.00
db file single write	0.00	0.00	-0.00	0.00	0.00	0.00	0.64	0.00	-100.00
direct path read	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	100.00
SQL*Net more data to client	0.00	0.00	0.00	0.01	0.00	-100.00	0.06	0.06	0.00
SQL*Net message from client	547.48	7,310.77	6,763.29	2.48	2.42	-2.42	2,501.73	1,690.55	-32.42
Streams AQ: qmn slave idle wait	88.17	1,779.46	1,691.29	0.04	0.04	0.00	28,089.62	28,061.26	-0.10
Streams AQ: qmn coordinator idle wait	88.17	1,779.46	1,691.29	0.05	0.05	0.00	18,209.10	18,232.64	0.13
virtual circuit status	87.63	1,777.23	1,689.61	0.03	0.03	0.00	30,010.55	29,894.50	-0.39
Streams AQ: waiting for messages in the queue	87.85	1,776.60	1,688.76	0.20	0.20	0.00	5,007.58	5,001.50	-0.12
wait for unread message on broadcast channel	87.28	1,767.04	1,679.77	0.99	0.99	0.00	1,005.58	1,001.62	-0.39
jobq slave wait	81.55	1,685.39	1,603.84	0.31	0.31	0.00	2,998.02	2,997.32	-0.02
class slave wait	0.12	2.48	2.36	0.00	0.00	0.00	5,016.12	5,004.81	-0.23
SGA: MMAN sleep for component shrink	0.00	0.00	-0.00	0.00	0.00	0.00	20.49	0.00	-100.00

In this example, the wait time for several events is much higher in the first period than in the second, such as `db file sequential read`, `db file scattered read`, `latch free`, and `buffer busy waits`.

See Also:

- "Wait Event Statistics" on page 2-3

Time Model Statistics

The Time Model Statistics section, shown in Figure 8–5, compares time model statistics in the two snapshot sets. The time model statistics are ordered based on the difference in total DB time spent on a particular type of operation between the two snapshot sets, and are listed in descending order. Time model statistics at the top of this section have the greatest differential between the two snapshot sets, and the related operations may be possible causes for performance degradation over time.

Figure 8–5 Time Model Statistics

	% DB Time			Time (seconds)		Time per Trans (seconds)		
Statistic Name	1st	2nd	Diff	1st	2nd	1st	2nd	%Diff
DB time	100.00	100.00	0.00	4,109.86	201.85	4.36	0.23	-94.72
PL/SQL execution elapsed time	1.31	22.59	21.28	53.77	45.60	0.06	0.05	-16.67
DB CPU	86.28	70.62	-15.67	3,546.10	142.54	3.76	0.16	-95.74
parse time elapsed	1.24	13.56	12.32	50.98	27.36	0.05	0.03	-40.00
hard parse elapsed time	1.12	12.00	10.88	46.00	24.21	0.05	0.03	-40.00
sql execute elapsed time	99.94	95.82	-4.12	4,107.45	193.42	4.36	0.22	-94.95
PL/SQL compilation elapsed time	0.06	0.74	0.68	2.53	1.49	0.00	0.00	0.00
hard parse (sharing criteria) elapsed time	0.02	0.40	0.38	0.94	0.80	0.00	0.00	0.00
repeated bind elapsed time	0.01	0.19	0.18	0.26	0.38	0.00	0.00	0.00
connection management call elapsed time	0.03	0.10	0.06	1.42	0.19	0.00	0.00	0.00
sequence load elapsed time	0.01	0.02	0.01	0.21	0.03	0.00	0.00	0.00
hard parse (bind mismatch) elapsed time	0.00	0.01	0.01	0.01	0.02	0.00	0.00	0.00
failed parse elapsed time	0.00	0.00	-0.00	0.05	0.00	0.00	0.00	0.00
Java execution elapsed time	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
RMAN cpu time (backup/restore)	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
failed parse (out of shared memory) elapsed t	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
inbound PL/SQL rpc elapsed time	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
background elapsed time	6.06	23.90	17.84	248.95	48.25	0.26	0.06	-76.92
background cpu time	0.47	10.59	10.12	19.12	21.37	0.02	0.02	0.00

In this example, several statistics are much higher in the first period than in the second, such as `DB time`, `DB CPU`, and `sql execute elapsed time`. This information suggests that the majority of the database activity and CPU utilization are being used by the execution of SQL statements. The SQL Statistics section should be analyzed next to determine if a particular SQL statement is causing the performance degradation.

See Also:

- "Time Model Statistics" on page 2-2

Operating System Statistics

The Operating System Statistics section compares operating system statistics in the two snapshot sets. This section provides an overall state of the operating system during each of the two periods being compared.

Service Statistics

The Service Statistics section compares services in the two snapshot sets. The services are ordered based on the difference in total DB time spent on a particular service between the two snapshot sets, and are listed in descending order.

SQL Statistics

The SQL Statistics section compares the top SQL statements in the two snapshot sets. The SQL statements are ordered based on different comparison methods, but in all cases, the top ten SQL statements with the greatest differential between the two snapshot sets are shown. These SQL statements may be possible causes for performance degradation over time, and are ordered based on the following categories:

- Top 10 SQL Comparison by Execution Time
- Top 10 SQL Comparison by CPU Time
- Top 10 SQL Comparison by Buffer Gets
- Top 10 SQL Comparison by Physical Reads
- Top 10 SQL Comparison by Executions
- Top 10 SQL Comparisons by Parse Calls
- Complete List of SQL Text

Top 10 SQL Comparison by Execution Time

SQL statements in this section are ordered based on the difference in total DB time spent processing the SQL statement between the two snapshot sets and are listed in descending order, as shown in Figure 8–6.

Figure 8–6 Top 10 SQL Comparison by Execution Time

	Exec Time % of DB Time			Exec Time (ms) / Exec		#Exec/sec (DB Time)		CPU Time (ms) / Exec		Physical Reads / Exec		#Rows Processed / Exec			
SQL Id	1st	2nd	Diff	1st	2nd	1st	2nd	1st	2nd	1st	2nd	1st	2nd	Multiple Plans	SQL Text
05b6pvb81dq8b	87.93		-87.93	3,613,737		0.00		3,467,674		0.00		0.00		Yes	SELECT /*+ ORDERED USE_NL(c) F...
68ccwtzvyn7qt	87.93		-87.93	3,613,737		0.00		3,467,674		0.00		0.00		Yes	DECLARE n number; BEGIN for i...
2b064ybzkwf1y	0.67	14.57	13.91	235	251	0.03	0.58	118	235	0.00	0.00	0.00	0.00	Yes	BEGIN EMD_NOTIFICATION.QUEUE_R...
257rmrxgvaj4z	0.03	8.25	8.21	14	35	0.02	2.38	11	31	0.00	0.00	13.00	13.00	Yes	select begin_time, wait_class...
fsz8wz5pmvamh	0.02	3.86	3.85	13	32	0.01	1.19	10	31	0.00	0.00	8.00	8.00	Yes	select metric_id, value from ...

In this example, the time it took to execute the top two SQL statements consumed 87.93 percent of DB time in the first period, but not in the second. These two SQL statements are likely the high-load SQL statements that caused the performance degradation in the first period and should be investigated. Review the SQL statements in the Complete List of SQL Text subsection of the report and tune them, if necessary.

See Also:

- Chapter 10, "Tuning SQL Statements" for information about tuning SQL statements

Top 10 SQL Comparison by CPU Time

SQL statements in this section are ordered based on the difference in CPU time spent processing the SQL statement between the two snapshot sets, and are listed in descending order.

Top 10 SQL Comparison by Buffer Gets

SQL statements in this section are ordered based on the difference in the number of total buffer cache reads or buffer gets made when processing the SQL statement between the two snapshot sets, and are listed in descending order.

Top 10 SQL Comparison by Physical Reads

SQL statements in this section are ordered based on the difference in the number of physical reads made when processing the SQL statement between the two snapshot sets, and are listed in descending order.

Top 10 SQL Comparison by Executions

SQL statements in this section are ordered based on the difference in the number of executions per second (based on DB time) when processing the SQL statement between the two snapshot sets, and are listed in descending order.

Top 10 SQL Comparisons by Parse Calls

SQL statements in this section are ordered based on the difference in the number of total parses made when processing the SQL statement between the two snapshot sets, and are listed in descending order. Parsing is one stage in the processing of a SQL statement. When an application issues a SQL statement, the application makes a parse call to Oracle Database. Making parse calls can greatly affect the performance of a database and should be minimized as much as possible.

See Also:

- *Oracle Database Concepts* for information about parsing

Complete List of SQL Text

This section displays the SQL text of all SQL statements listed in the SQL Statistics section.

Instance Activity Statistics

The Instance Activity Statistics section compares the statistic values of instance activity between the two snapshot sets. For each statistic, the value of the statistic is shown along with the differentials measured by DB time, elapsed time, and per transaction.

The instance activity statistics are categorized into the following sections:

- Key Instance Activity Statistics
- Other Instance Activity Statistics

Key Instance Activity Statistics

As the name suggests, the Key Instance Activity Statistics section displays the difference in key instance activity statistic values between the two snapshot sets.

Other Instance Activity Statistics

The Other Instance Activity Statistics sections displays the difference in instance activity for all other statistics between the two snapshot sets.

I/O Statistics

The I/O Statistics section compares the I/O operations performed on tablespaces and database files between the two snapshot sets. A drastic increase in I/O operations between the two snapshots may be the cause of performance degradation over time.

For each tablespace or database file, the difference in the number of reads, writes, and buffer cache waits (or buffer gets) are quantified as a percentage. The database files are ordered based on different comparison methods, but in all cases, the top 10 database files with the greatest differential between the two snapshot sets are shown.

I/O statistics comparison are divided into the following categories:

- Tablespace I/O Statistics
- Top 10 File Comparison by I/O
- Top 10 File Comparison by Read Time
- Top 10 File Comparison by Buffer Waits

Tablespace I/O Statistics

Tablespaces in this section are ordered by the difference in the number of normalized I/Os performed on the tablespace between the two snapshot sets, and are listed in descending order. Normalized I/Os are the sum of average reads and writes per second.

Top 10 File Comparison by I/O

Database files in this section are ordered by the difference in the number of normalized I/Os performed on the database file between the two snapshot sets, and are listed in descending order. Normalized I/Os are the sum of average reads and writes per second.

Top 10 File Comparison by Read Time

Database files in this section are ordered by the difference in the percentage of DB time spent reading data from the database file between the two snapshot sets, and are listed in descending order.

Top 10 File Comparison by Buffer Waits

Database files in this section are ordered by the difference in the number of buffer waits (waits caused during a free buffer lookup in the buffer cache) performed on the database file between the two snapshot sets, and are listed in descending order.

Advisory Statistics

The Advisory Statistics section compares program global area (PGA) memory statistics between the two snapshot sets, and is divided into the following categories:

- PGA Aggregate Summary
- PGA Aggregate Target Statistics

PGA Aggregate Summary

The PGA Aggregate Summary section compares the PGA cache hit ratio between the two snapshot sets.

PGA Aggregate Target Statistics

The PGA Aggregate Target Statistics section compares the key statistics related to the automatic PGA memory management between the two snapshot sets.

Wait Statistics

The Wait Statistics section compares statistics for buffer waits and enqueues between the two snapshot sets.

Wait statistics are divided into the following categories:

- Buffer Wait Statistics
- Enqueue Activity

Buffer Wait Statistics

The Buffer Wait Statistics section compares buffer waits between the two snapshot sets. Buffer waits happen during a free buffer lookup in the buffer cache.

Enqueue Activity

The Enqueue Activity section compares enqueue activities between the two snapshot sets. Enqueues are shared memory structures (or locks) that serialize access to database resources and can be associated with a session or transaction.

See Also:

- *Oracle Database Performance Tuning Guide*
- *Oracle Database Reference* for information about enqueues

Latch Statistics

The Latch Statistics section compares the number of total sleeps for latches between the two snapshot sets in descending order.

Latches are simple, low-level serialization mechanisms to protect shared data structures in the system global area (SGA). For example, latches protect the list of users currently accessing the database and the data structures describing the blocks in the buffer cache. A server or background process acquires a latch for a very short time while manipulating or looking up one of these structures. The implementation of latches is operating system dependent, particularly in regard to whether and how long a process will wait for a latch.

See Also:

- *Oracle Database Performance Tuning Guide* for information about latches

Segment Statistics

The Segment Statistics section compares segments, or database objects (such as tables and indexes), between the two snapshot sets. The segments are ordered based on different comparison methods, but in all cases the top five segments with the greatest differential between the two snapshot sets are shown. These segments may be the causes of performance degradation over time, and are ordered based on the following categories:

- Top 5 Segments Comparison by Logical Reads

- Top 5 Segments Comparison by Physical Reads
- Top 5 Segments by Row Lock Waits
- Top 5 Segments by ITL Waits
- Top 5 Segments by Buffer Busy Waits

Top 5 Segments Comparison by Logical Reads

Segments in this section, shown in Figure 8–7, are ordered based on the difference in the number of logical reads (total number of reads from disk or memory) performed on the segment between the two snapshot sets, and are listed in descending order.

Figure 8–7 Top 5 Segments Comparison by Logical Reads

					% of Total Logical Reads			Logical Reads	
Owner	Tablespace	Object Name	Subobject Name	Type	1st	2nd	Diff	1st	2nd
SH	EXAMPLE	CUSTOMERS		TABLE	99.79		-99.79	336,134,208	
SYS	SYSTEM	OBJ$		TABLE	0.04	12.90	12.86	142,944	123,472
SYS	SYSTEM	I_OBJ1		INDEX	0.03	8.82	8.79	93,040	84,384
SYS	SYSTEM	HISTGRM$		TABLE	0.01	7.69	7.68	45,040	73,616
SYS	SYSAUX	I_WRI$_OPTSTAT_H_OBJ#_ICOL#_ST		INDEX	0.00	7.50	7.50	4,944	71,808

In the example, an extremely high percentage of logical reads are made on the `CUSTOMERS` table in the first period. Depending on the investigation of the high-load SQL statements, data access to this table may need to be tuned using an index or a materialized view.

See Also:

- Chapter 11, "Optimizing Data Access Paths" for information about creating indexes and materialized views

Top 5 Segments Comparison by Physical Reads

Segments in this section are ordered based on the difference in the number of physical reads (such as disk reads) performed on the segment between the two snapshot sets, and are listed in descending order.

Top 5 Segments by Row Lock Waits

Segments in this section are ordered based on the difference in the number of waits on row locks for the segment between the two snapshot sets, and are listed in descending order. Row-level locks are primarily used to prevent two transactions from modifying the same row. When a transaction needs to modify a row, a row lock is acquired.

See Also:

- *Oracle Database Concepts* for information about row locks

Top 5 Segments by ITL Waits

Segments in this section are ordered based on the difference in the number of interested transaction list (ITL) waits for the segment between the two snapshot sets, and are listed in descending order.

See Also:

- *Oracle Database Performance Tuning Guide* for information about ITL waits

Top 5 Segments by Buffer Busy Waits

Segments in this section are ordered based on the difference in the number of buffer busy waits for the segment between the two snapshot sets, and are listed in descending order.

See Also:

- *Oracle Database Performance Tuning Guide* for information about buffer busy waits

Dictionary Cache Statistics

The Dictionary Cache Statistics section compares the number of get requests performed on the dictionary cache between the two snapshot sets in descending order. The difference is measured by the number of get requests per second of both total DB time and elapsed time. The dictionary cache is a part of the SGA that stores information about the database, its structures, and its users. The dictionary cache also stores descriptive information (or metadata) about schema objects, which is accessed by Oracle Database during the parsing of SQL statements.

See Also:

- *Oracle Database Performance Tuning Guide* for information about the dictionary cache

Library Cache Statistics

The Library Cache Statistics section compares the number of get requests performed on the library cache between the two snapshot sets in descending order. The difference is measured by the number of get requests per second of both total DB time and elapsed time. The library cache is a part of the SGA that stores table information, object definitions, SQL statements, and PL/SQL programs.

See Also:

- *Oracle Database Performance Tuning Guide* for information about the library cache

SGA Statistics

The SGA Statistics section compares SGA memory statistics between the two snapshot sets, and is divided into the following categories:

- SGA Memory Summary
- SGA Breakdown Difference

SGA Memory Summary

The SGA Memory Summary section summarizes the SGA memory configurations for the two snapshot sets.

SGA Breakdown Difference

The SGA Breakdown Difference section compares SGA memory usage for each of its subcomponents between the two snapshot sets. The difference is measured based on the percentage changed in the beginning and ending values of memory usage between the two snapshot sets.

init.ora Parameters

The init.ora Parameters section lists all the initialization parameter values for the first snapshot set. Any changes in the values of the initialization parameters between the two snapshot sets are listed for the second snapshot set with the changed value shown.

Part IV

SQL Tuning

Part IV describes how to effectively tune SQL statements and contains the following chapters:

9

Identifying High-Load SQL Statements

High-load SQL statements are SQL statements that are very resource intensive and may consume a disproportionate amount of system resources. These SQL statements oftentimes cause a large impact on database performance, and need to be tuned to optimize their performance and resource consumption. Even when a database itself is properly tuned, inefficient SQL statements can significantly degrade the performance of a database.

Identifying high-load SQL statements is an important SQL tuning activity that must be performed regularly. The Automatic Database Diagnostic Monitor (ADDM) automates this task by proactively identifying potential high-load SQL statements. Additionally, Enterprise Manager can be used to identify high-load SQL statements that require further investigation. Once the high-load SQL statements have been identified, they can be tuned using the SQL Tuning Advisor and SQL Access Advisor.

This chapter describes how to identify high-load SQL statements and contains the following sections:

Identifying High-Load SQL Statements Using ADDM Findings

By default, ADDM runs proactively once every hour, and it analyzes key statistics gathered by the Automatic Workload Repository (AWR) over the last hour to identify any performance problems, including high-load SQL statements. When performance problems are identified by ADDM, they are displayed as ADDM findings in the Automatic Database Diagnostic Monitor (ADDM) page. ADDM provides recommendations with each ADDM finding. When a high-load SQL statement is identified, ADDM will give appropriate recommendations, such as running the SQL Tuning Advisor on the SQL statement, and tuning can begin as described in Chapter 10, "Tuning SQL Statements".

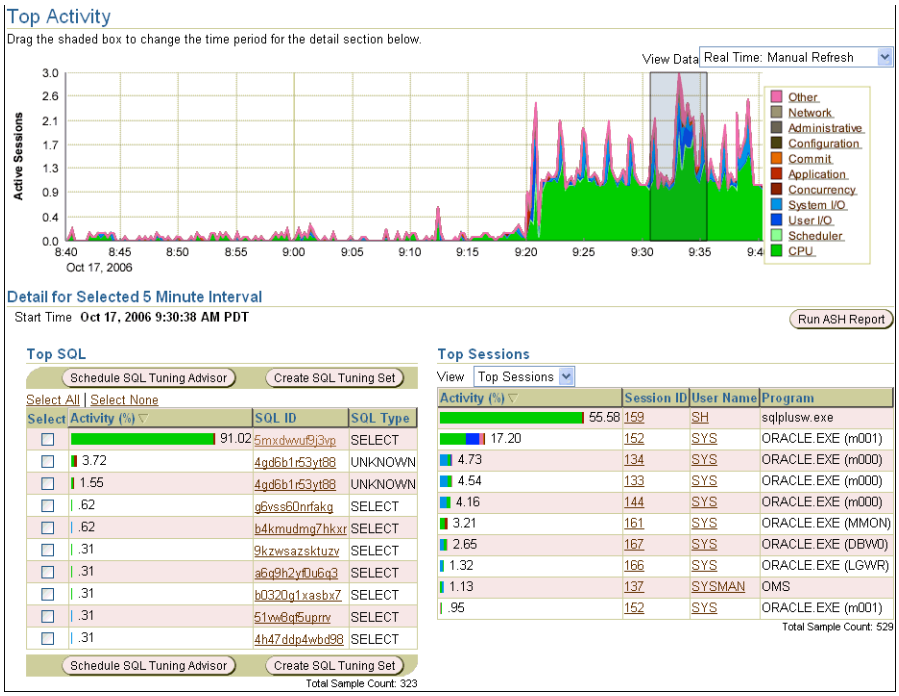
See Also:

- "Overview of the Automatic Database Diagnostic Monitor" on page 3-1
- "Interpreting the Automatic Database Diagnostics Monitor Findings" on page 3-6
- "Implementing ADDM Recommendations" on page 3-7

Identifying High-Load SQL Statements Using Top SQL

ADDM automatically identifies high-load SQL statements that may be causing system-wide performance degradation. Under normal circumstances, manual identification of high-load SQL statements is not necessary. In some cases, however, you may want to monitor SQL statements at a more granular level. The Top SQL section of the Top Activity page in Enterprise Manager, shown in Figure 9–1, enables you to identify high-load SQL statements for any 5-minute interval.

Figure 9–1 Top Activity Page



To access the Top Activity page, on the Database Performance page, click **Top Activity**.

The Top Activity page shows a 1-hour timeline of the top activity running on the database. SQL statements that are using the highest percentage of database activity are listed under the Top SQL section, and are displayed in 5-minute intervals. To move the 5-minute interval, drag and drop the shaded box to the time of interest. The information contained in the Top SQL section will be automatically updated to reflect the selected time period. Use this page to identify high-load SQL statements that may be causing performance problems.

To monitor SQL statements for a longer duration than one hour, switch to Historical view by selecting **Historical** from the View Data list. In Historical view, you can view the top SQL statements for the duration defined by the AWR retention period.

This section contains the following topics:

- Viewing SQL Statements by Wait Class
- Viewing Details of SQL Statements

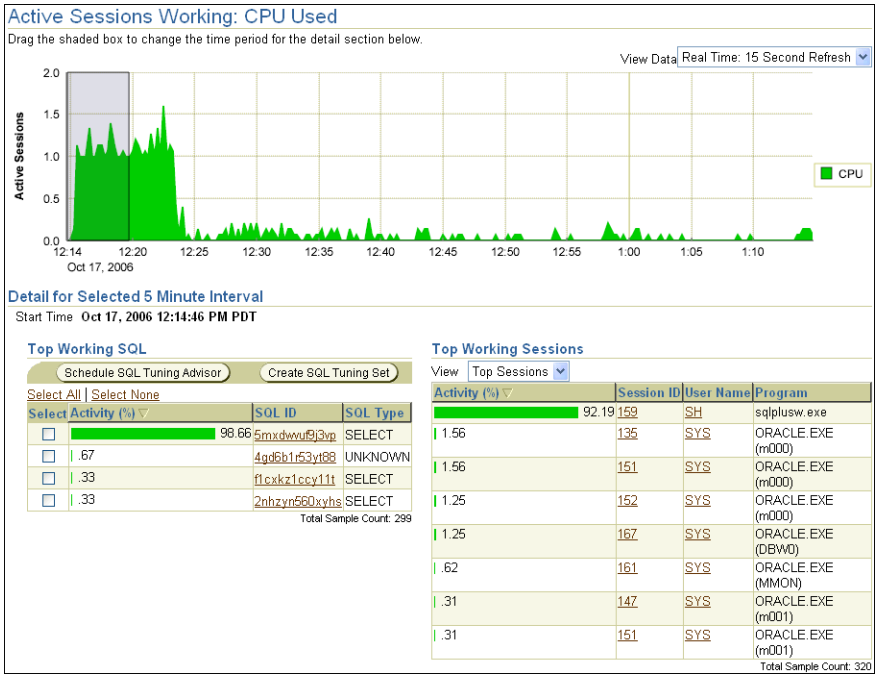
Viewing SQL Statements by Wait Class

The SQL statements that appear in the Top SQL section are categorized into various wait classes, based on their corresponding color as described in the legend on the Top

Activity graph. On the Top Activity page, shown in Figure 9–1, SQL statements are displayed for CPU usage (shown in green), concurrency (shown in dark red), and system I/O (shown in light blue) wait classes. To view only SQL statements for a particular wait class, click the block of color on the graph for the wait class, or its corresponding wait class in the legend. The Active Sessions Working page for the selected wait class appears, and the Top SQL section will be automatically updated to show only the SQL statements for that wait class.

The example in Figure 9–2 shows the Active Sessions Working page for the CPU Used wait class. Only SQL statements that are consuming the most CPU time is displayed in the Top Working SQL section

Figure 9–2 Viewing SQL Statement By Wait Class



See Also:

- "Monitoring User Activity" on page 4-2 for information about using the Active Sessions Working page

Viewing Details of SQL Statements

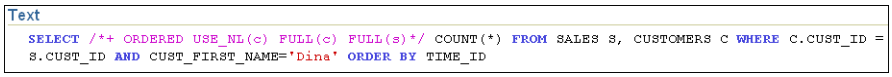
The Top SQL section displays the SQL statements executed within the selected 5-minute interval in descending order based on their resource consumption. The SQL statement at the top of this table represents the most resource intensive SQL statement during that time period, followed by the second most resource intensive SQL statement, and so forth. In the example shown in Figure 9–1 on page 9-2, the SQL statement with `SQL_ID` 5mxdwvuf9j3vp is using 91 percent of database activity and should be investigated.

To view details about a SQL statement, in the Top SQL section, click the **SQL ID** link of the SQL statement. This displays the SQL Details page for the selected SQL statement, as shown in Figure 9–3.

Figure 9–3 SQL Details Page

The Text section contains the SQL text for the selected SQL statement, as shown in Figure 9–4.

Figure 9–4 Viewing SQL Text



If only part of the SQL statement is displayed, an + icon will appear next to the Text heading. To view the SQL text for the entire SQL statement, click the + icon.

If the SQL statement has multiple plans, you can display SQL details for all plans by selecting **All** in the Plan Hash Value list. Alternatively, you can select a particular plan to display SQL details for that plan only.

The Real Time view shows SQL details for the past hour. To view SQL details for a longer time period, switch to Historical view by selecting **Historical** from the View Data list. In Historical view, you can view SQL details in the past, up to the duration defined by the AWR retention period.

The SQL Details page also contains four subpages that you can use to perform the following tasks:

- Viewing SQL Statistics
- Viewing Session Activity
- Viewing SQL Execution Plan
- Viewing SQL Tuning Information

If the SQL statement is identified to be a high-load SQL statement after reviewing the SQL details, you can proceed to tune the SQL statement, as described in Chapter 10, "Tuning SQL Statements".

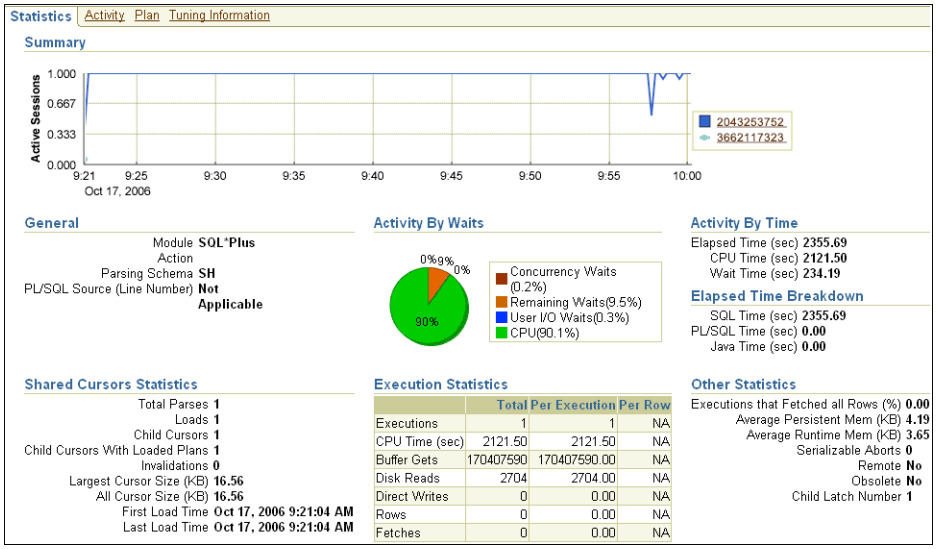
Viewing SQL Statistics

To view statistics for the SQL statement, under Details, click **Statistics**.

The Statistics subpage, as shown in Figure 9–5, displays statistical information about the SQL statement in the following sections:

- Viewing SQL Statistics Summary
- Viewing General SQL Statistics
- Viewing Activity by Wait Statistics and Activity by Time Statistics
- Viewing Elapsed Time Breakdown Statistics
- Viewing Shared Cursors Statistics and Execution Statistics
- Viewing Other SQL Statistics

Figure 9–5 Viewing SQL Statistics



Viewing SQL Statistics Summary The Summary section displays SQL statistics and activity on a chart.

In Real Time view, the Active Sessions chart shows the average number of active sessions executing the SQL statement in the last hour. If the SQL statement has multiple plans and **All** is selected in the Plan Hash Value list, the chart will display each plan in different colors, enabling you to easily spot if the plan changed and whether this may be the cause of the performance degradation. Alternatively, you can select a particular plan to display that plan only.

In Historical view, the chart shows execution statistics in different dimensions. To view execution statistics, select the desired dimension from the View list:

- Elapsed time per execution
- Executions per hour
- Disk reads per execution

- Buffer gets per execution

This enables you to track the response time of the SQL statement using different dimensions and determine if the performance of the SQL statement has degraded based on the dimension selected.

Viewing General SQL Statistics The General section enables you to identify the origin of the SQL statement by listing the following information:

- Module, if specified using the `DBMS_APPLICATION_INFO` package
- Action, if specified using the `DBMS_APPLICATION_INFO` package
- Parsing schema, or the database users account that is used to execute the SQL statement
- PL/SQL source, or the line if the SQL statement is part of PL/SQL program unit

Viewing Activity by Wait Statistics and Activity by Time Statistics The Activity by Wait and Activity by Time sections enable you to identify where the SQL statement spent most of its time. The Activity by Wait section contains a graphical representation of how much elapsed time is consumed by CPU and by remaining waits. The Activity by Time section breaks out the total elapsed time into CPU time and wait time by seconds.

Viewing Elapsed Time Breakdown Statistics The Elapsed Time Breakdown section enables you to identify if the SQL statement itself is consuming a lot of time, or whether the total elapsed time is inflated due to the amount of time the originating program or application is spending with the PL/SQL or Java engine. If the PL/SQL time or Java time makes up a significant portion of the elapsed time, there may be minimal benefit gained by tuning the SQL statement. Instead, you should examine the application to determine how the PL/SQL time or Java time can be reduced.

Viewing Shared Cursors Statistics and Execution Statistics The Shared Cursors Statistics and Execution Statistics sections provide information about the efficiency of various stages of the SQL execution process.

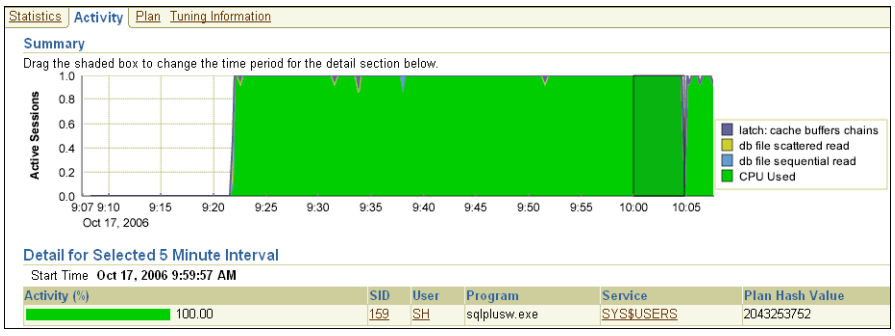
Viewing Other SQL Statistics The Other Statistics section provides additional information about the SQL statement, such as average persistent and runtime memory.

Viewing Session Activity

To view session activity for the SQL statement, in the Details section, click **Activity**.

The Activity subpage contains a graphical representation of the session activity, as shown in Figure 9–6.

Figure 9–6 Viewing Session Activity



The Activity subpage displays details of various sessions executing the SQL statement. The Active Sessions chart profiles the average number of active sessions over time. You can drag the shaded box to select a 5-minute interval. The Detail for Selected 5 Minute Interval section lists the sessions that executed the SQL statement during the selected 5-minute interval. The multi-colored bar in the Activity % column depicts how the database time is divided for each session while executing the SQL statement. To view more details for a particular session, click the link in the SID column of the session you want to view.

See Also:

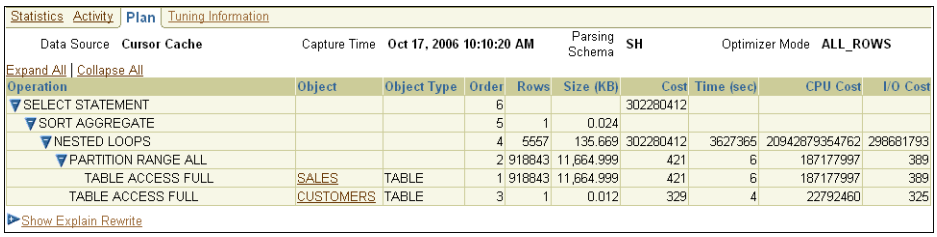
- "Monitoring Top Sessions" on page 4-4 for information about monitoring session activity and details

Viewing SQL Execution Plan

To view the execution plan for the SQL statement, in the Details section, click **Plan**.

The Plan subpage contains the execution plan for the SQL statement, as shown in Figure 9–7. Oracle Database compares the cost for the query, with and without query rewrite, and selects the least costly alternative. If a rewrite is necessary, the query rewrite and its cost benefit are displayed in the Explain Rewrite section.

Figure 9–7 Viewing SQL Execution Plan


Statistics | Activity | Plan | Tuning Information

Data Source Cursor Cache Capture Time Oct 17, 2006 10:10:20 AM Parsing Schema SH Optimizer Mode ALL_ROWS

Expand All | Collapse All

Operation	Object	Object Type	Order	Rows	Size (KB)	Cost	Time (sec)	CPU Cost	I/O Cost
SELECT STATEMENT			6			302280412			
SORT AGGREGATE			5	1	0.024				
NESTED LOOPS			4	5557	135.669	302280412	3627365	20942879354762	298681793
PARTITION RANGE ALL			2	918843	11,664.999	421	6	187177997	389
TABLE ACCESS FULL	SALES	TABLE	1	918843	11,664.999	421	6	187177997	389
TABLE ACCESS FULL	CUSTOMERS	TABLE	3	1	0.012	329	4	22792460	325

Show Explain Rewrite

See Also:

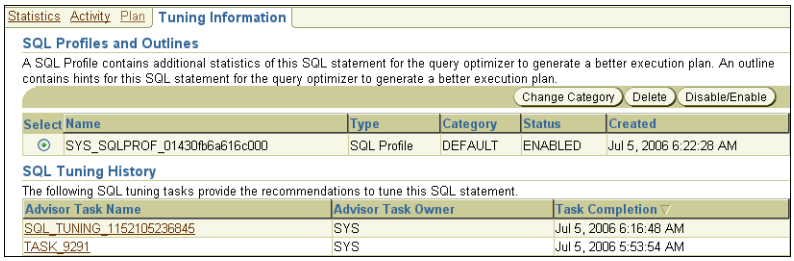
- Chapter 10, "Tuning SQL Statements" for information about execution plan and the query optimizer

Viewing SQL Tuning Information

To view the tuning information for the SQL statement, in the Details section, click **Tuning Information**.

As shown in Figure 9–8, the Tuning Information subpage contains information about the SQL tuning tasks and the SQL profiles recommended by the SQL Tuning Advisor for the SQL statement. The SQL Tuning History section displays a history of tuning activities using the SQL Tuning Advisor or SQL Access Advisor.

Figure 9–8 Viewing SQL Tuning Information



See Also:

- Chapter 10, "Tuning SQL Statements" for information about the SQL Tuning Advisor and SQL profiles
- Chapter 11, "Optimizing Data Access Paths" for information about the SQL Access Advisor

10

Tuning SQL Statements

A SQL statement expresses the data you want Oracle Database to retrieve. For example, you can use a SQL statement to retrieve all employees in a department. When Oracle Database executes the SQL statement, it first determines the best and most efficient way to retrieve the results. The part of Oracle Database that makes this determination is called the query optimizer, or simply the optimizer. The optimizer determines if it is more efficient to read all data in the table, called a full table scan, or to use an index. It compares the cost of all possible approaches and chooses the approach with the least cost. The method that a SQL statement is physically executed is called an execution plan, which the optimizer is responsible for generating. The determination of an execution plan is an important step in the processing of any SQL statement, and can greatly affect execution time.

Starting with Oracle Database 10*g*, the query optimizer can also help you tune SQL statements. Using the SQL Tuning Advisor and SQL Access Advisor, you can invoke the query optimizer in advisory mode to examine a given SQL statement, or a set of SQL statements, and provide recommendations to improve their efficiency. The SQL Tuning Advisor and SQL Access Advisor can make various types of recommendations, such as creating SQL profiles, restructuring SQL statements, creating additional indexes or materialized views, and refreshing optimizer statistics. Additionally, Enterprise Manager enables you to accept and implement many of these recommendations with just a few mouse clicks.

The SQL Access Advisor is primarily responsible for making schema modification recommendations, such as adding or dropping indexes and materialized views. The SQL Tuning Advisor makes other types of recommendations, such as creating SQL profiles and restructuring SQL statements. In some cases where significant performance improvements can be gained by creating a new index, the SQL Tuning Advisor may recommend doing so. However, such recommendations should be verified by running the SQL Access Advisor using a SQL workload that contains a set of representative SQL statements.

This chapter describes how to tune SQL statements using the SQL Tuning Advisor and contains the following sections:

- Tuning SQL Statements Using the SQL Tuning Advisor
- Managing SQL Tuning Sets
- Managing SQL Profiles

See Also:

- Chapter 9, "Identifying High-Load SQL Statements"
- Chapter 11, "Optimizing Data Access Paths" for information about the SQL Access Advisor

Tuning SQL Statements Using the SQL Tuning Advisor

You can use the SQL Tuning Advisor to tune a single or multiple SQL statements. When tuning multiple SQL statements, keep in the mind that the SQL Tuning Advisor does not recognize interdependencies between the SQL statements. Instead, it is meant to be a convenient way for you to run the SQL Tuning Advisor for a large number of SQL statements.

As described in Chapter 9, "Identifying High-Load SQL Statements", ADDM automatically identifies high-load SQL statements. In such cases, simply click **Schedule/Run SQL Tuning Advisor** in the Recommendation Detail page to invoke the SQL Tuning Advisor. This section describes how to run the SQL Tuning Advisor manually to tune SQL statements.

To tune SQL statements using the SQL Tuning Advisor:

1. On the Database Home page, under Related Links, click **Advisor Central**.

   The Advisor Central page appears.

2. Under Advisors, click **SQL Tuning Advisor**.

   The SQL Tuning Advisor Links page appears.

3. To run a SQL tuning task for:

   - One or more high-load SQL statements, click **Top Activity**.

     The Top Activity page appears.

     In the Top SQL section, select the SQL statement you want to tune and click **Schedule SQL Tuning Advisor**. For information about identifying high-load SQL statements using the Top Activity page, see "Identifying High-Load SQL Statements Using Top SQL" on page 9-2.

   - Historical SQL statements from the Automatic Workload Repository (AWR), click **Period SQL**.

     The Period SQL page appears. Under Historical Interval Selection, click the band below the chart, and select the 24-hour interval for which you want to view SQL statements that ran on the database. Under Detail for Selected 24 Hour Interval, select the SQL statement you want to tune, and click **Schedule SQL Tuning Advisor**.

   - A SQL tuning set, click **SQL Tuning Sets**.

     The SQL Tuning Sets page appears. Select the SQL tuning set that contains the SQL statements you want to tune and click **Schedule SQL Tuning Advisor**. For information about creating SQL tuning sets, see "Creating a SQL Tuning Set" on page 10-5.

   The Schedule Advisor page appears.

4. Under Scope, select the scope of tuning to perform.

   A limited scope takes approximately 1 second to tune each SQL statement but does not recommend a SQL profile. A comprehensive scope performs a complete analysis and recommends a SQL profile, when appropriate, but may take much longer. When running a comprehensive tuning task, you can set a time limit (in minutes) in the Total Time Limit field. Note that setting the time limit too small may affect the quality of the recommendations. Running a SQL Tuning Advisor task in comprehensive mode may take several minutes to tune a single SQL statement, and is both time and resource intensive to do so every time a query has to be hard-parsed. This method should only be used for high-load SQL statements that have a significant impact on the entire system.

   For information about SQL profiles, see "Managing SQL Profiles" on page 10-15.

5. Under Schedule, select **Immediately** to run the SQL tuning task immediately, or **Later** to schedule a specific time in the future, and click **OK**.

   Depending on your selection, the SQL tuning task will either run immediately or at its scheduled time.

6. If the SQL tuning task runs immediately, the SQL Tuning Results page appears once the task is complete. Proceed to Step 8.

7. If the SQL tuning task is scheduled to run at a later time, various actions can be performed on the Advisor Central page:

   - To view results for the SQL tuning task after it completes, select the SQL Tuning Advisor task and click **View Result**.

     The SQL Tuning Results page appears. Proceed to the next step.

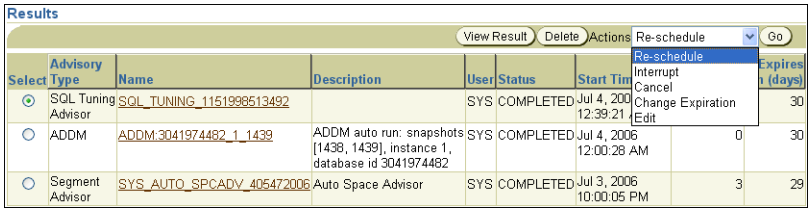
   - To delete a SQL tuning task, select the SQL Tuning Advisor task and click **Delete**.

   - To reschedule a SQL tuning task, select the SQL Tuning Advisor task. From the Actions list, select **Re-schedule** and click **Go**.

   - To interrupt a SQL tuning task that is running, select the SQL Tuning Advisor task. From the Actions list, select **Interrupt** and click **Go**.

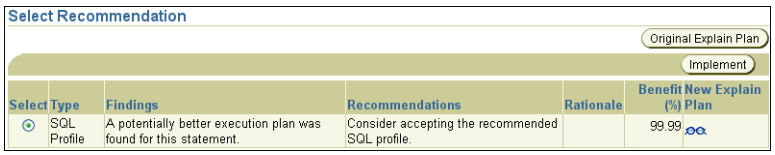
- To cancel a scheduled SQL tuning task, select the SQL Tuning Advisor task. From the Actions list, select **Cancel** and click **Go**.
- To change the expiration of a SQL tuning task, select the SQL Tuning Advisor task. From the Actions list, select **Change Expiration** and click **Go**.

  Results of each advisor run are stored in the database so that they can be referenced later. This data is stored until it expires, at which point it will be deleted by the AWR purging process.
- To edit a scheduled SQL tuning task, select the SQL Tuning Advisor task. From the Actions list, select **Edit** and click **Go**.

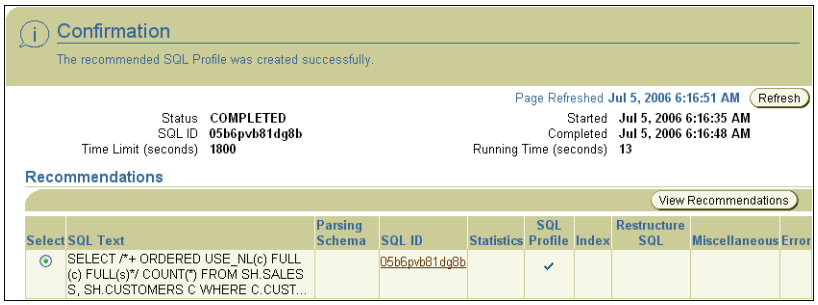


8. The Recommendations for SQL ID page appears. To implement the recommendation, click **Implement**.

   If you used a SQL tuning set, multiple recommendations may be displayed. To help you decide whether or not to implement a recommendation, an estimated benefit of implementing the recommendation is displayed in the Benefit (%) column. The Rationale column displays an explanation of why the recommendation is made. To view the original execution plan for the SQL statement, click **Original Explain Plan**. To view the new execution plan for the SQL statement, click the icon in the New Explain Plan column.



9. The SQL Tuning Results page appears with a confirmation that the recommended action was completed.



Managing SQL Tuning Sets

A SQL tuning set is a database object that includes one or more SQL statements, their execution statistics and execution context, and can be used as an input source for the SQL Tuning Advisor and SQL Access Advisor. You can load SQL statements into a

SQL tuning set from different SQL sources, such as the Automatic Workload Repository (AWR), the cursor cache, or high-load SQL statements that you identified.

A SQL tuning set includes:

- A set of SQL statements
- Associated execution context, such as user schema, application module name and action, list of bind values, and the cursor compilation environment
- Associated basic execution statistics, such as elapsed time, CPU time, buffer gets, disk reads, rows processed, cursor fetches, the number of executions, the number of complete executions, optimizer cost, and the command type
- Associated execution plans and row source statistics for each SQL statement (optional)

SQL statements can be filtered using the application module name and action, or any of the execution statistics. In addition, SQL statements can be ranked based on any combination of execution statistics.

SQL tuning sets are transportable across databases and can be exported from one system to another, allowing SQL workloads to be transferred between databases for remote performance diagnostics and tuning. When high-load SQL statements are identified on a production system, it may not be desirable to perform investigation and tuning activities on the production system directly. This feature enables you to transport the high-load SQL statements to a test system, where they can be safely analyzed and tuned.

Using Oracle Enterprise Manager, you can manage SQL tuning sets by:

- Creating a SQL Tuning Set
- Deleting a SQL Tuning Set
- Transporting a SQL Tuning Set

Creating a SQL Tuning Set

This section describes how to create a SQL tuning set using Oracle Enterprise Manager.

To create a SQL tuning set:

1. Specify the initial options for the SQL tuning set, as described in "Creating a SQL Tuning Set: Options" on page 10-5.
2. Select the load method to use for collecting and loading SQL statements into the SQL tuning set, as described in "Creating a SQL Tuning Set: Load Method" on page 10-7.
3. Specify the filter options for the SQL tuning set, as described in "Creating a SQL Tuning Set: Filter Options" on page 10-9.
4. Schedule and submit a job to collect the SQL statements and load them into the SQL tuning set, as described in "Creating a SQL Tuning Set: Schedule" on page 10-11.

Creating a SQL Tuning Set: Options

The first step in creating a SQL tuning set is to specify the initial options, such as name, owner, and description.

To specify options for creating a SQL tuning set:

1. On the Database Home page, under Related Links, click **Advisor Central**.

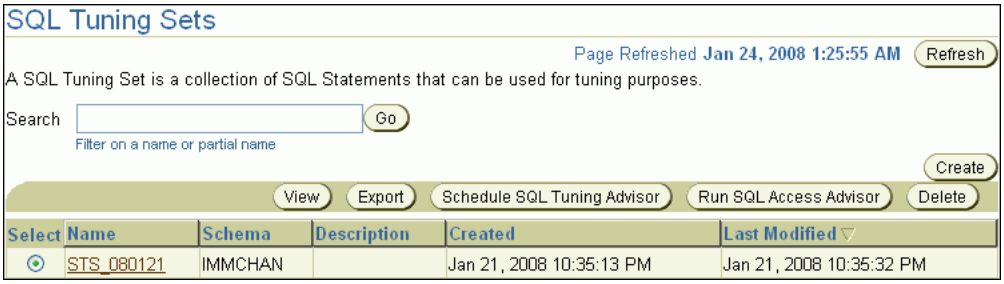
   The Advisor Central page appears.

2. Under Advisors, click **SQL Tuning Advisor**.

   The SQL Tuning Advisor Links page appears.

3. Click **SQL Tuning Sets**.

   The SQL Tuning Sets page appears. Existing SQL tuning sets are displayed on this page.



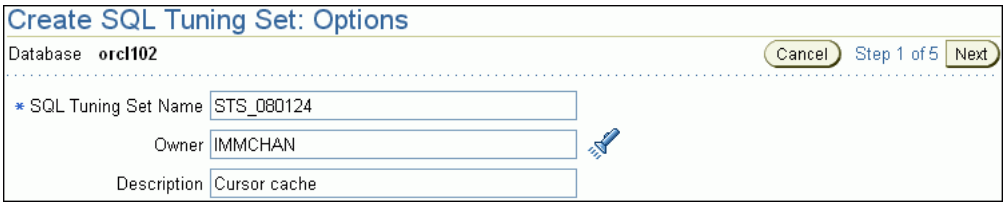
4. Click **Create**.

   The Create SQL Tuning Set: Options page appears.

5. In the **SQL Tuning Set Name** field, enter a name for the SQL tuning set.

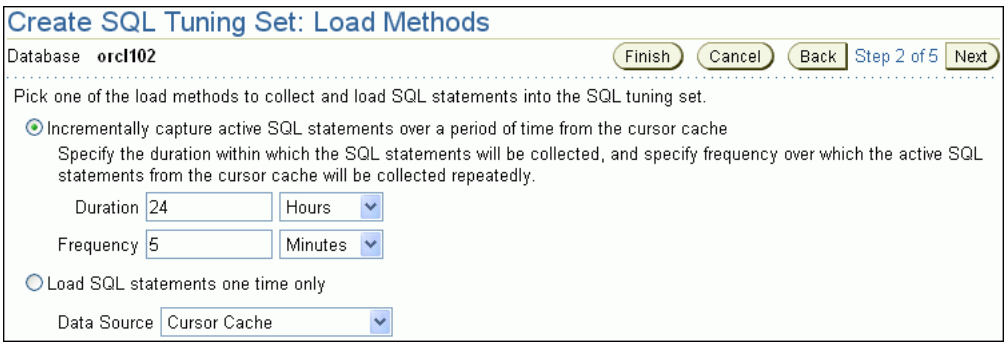
6. In the **Owner** field, enter the owner of the SQL tuning set.

7. In the **Description** field, enter a description of the SQL tuning set.



8. Click **Next**.

   The Create SQL Tuning Set: Load Methods page appears.



9. Proceed to the next step, as described in "Creating a SQL Tuning Set: Load Method" on page 10-7.

Creating a SQL Tuning Set: Load Method

After specifying options for the SQL tuning set, select the load method to use for collecting and loading SQL statements into the SQL tuning set, as described in the following sections:

- Loading Active SQL Statements Incrementally from the Cursor Cache
- Loading SQL Statements from the Cursor Cache
- Loading SQL Statements from AWR Snapshots
- Loading SQL Statements from Preserved Snapshot Sets
- Loading SQL Statements from a User-Defined Workload

> **Tip:** Before selecting the load method for the SQL tuning set, you need to create a SQL tuning set and specify the initial options, as described in "Creating a SQL Tuning Set: Options" on page 10-5

Loading Active SQL Statements Incrementally from the Cursor Cache You can load active SQL statements from the cursor cache into the SQL tuning set incrementally over a specified period of time. This allows you to not only collect current and recent SQL statements stored in the SQL cache, but also SQL statements that will run during the specified time period in the future.

To load active SQL statements incrementally from the cursor cache:

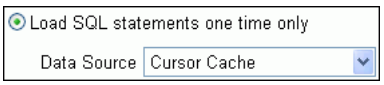
1. On the Create SQL Tuning Set: Load Methods page, select **Incrementally capture active SQL statements over a period of time from the cursor cache**.
2. In the **Duration** field, specify how long active SQL statements will be captured.
3. In the **Frequency** field, specify how often active SQL statements will be captured during the specified duration.
4. Click **Next**.

   The Create SQL Tuning Set: Filter Options page appears.
5. Proceed to the next step, as described in "Creating a SQL Tuning Set: Filter Options" on page 10-9.

Loading SQL Statements from the Cursor Cache You can load SQL statements from the cursor cache into the SQL tuning set. However, because only current and recent SQL statements are stored in the SQL cache, collecting these SQL statements only once may result in a SQL tuning set this is not representative of the entire workload on your database.

To load SQL statements from the cursor cache:

1. On the Create SQL Tuning Set: Load Methods page, select **Load statements one time only**.
2. In the Data Source field, select **Cursor Cache**.



3. Click **Next**.

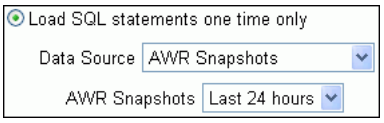
   The Create SQL Tuning Set: Filter Options page appears.

4. Proceed to the next step, as described in "Creating a SQL Tuning Set: Filter Options" on page 10-9.

Loading SQL Statements from AWR Snapshots You can load SQL statements captured in AWR snapshots. This is useful when you want to collect SQL statements for specific snapshot periods of interest that can be used for later comparison or tuning purposes.

To load SQL statements from AWR snapshots:

1. On the Create SQL Tuning Set: Load Methods page, select **Load statements one time only**.
2. In the Data Source field, select **AWR Snapshots**.



3. In the **AWR Snapshots** field, select the snapshots to include. Do one of the following:
   - Select **Last 24 hours**.

     Only snapshots that are captured and stored in the AWR in the last 24 hours will be included.
   - Select **Last 7 days**.

     Only snapshots that are captured and stored in the AWR in the last 7 days will be included.
   - Select **Last 31 days**.

     Only snapshots that are captured and stored in the AWR in the last 31 days will be included.
   - Select **ALL**.

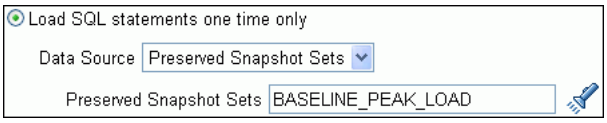
     All snapshots that are captured and stored in the AWR will be included.
4. Click **Next**.

   The Create SQL Tuning Set: Filter Options page is shown.
5. Proceed to the next step, as described in "Creating a SQL Tuning Set: Filter Options" on page 10-9.

Loading SQL Statements from Preserved Snapshot Sets You can load SQL statements captured in preserved snapshot sets. This is useful when you want to collect SQL statements that are representative of a time period during known performance levels that can be used for later comparison or tuning purposes.

To load SQL statements from preserved snapshot sets:

1. On the Create SQL Tuning Set: Load Methods page, select **Load statements one time only**.
2. In the Data Source field, select **Preserved Snapshot Sets**.
3. In the Preserved Snapshot Sets field, select the preserved snapshot set to include.



4. Click **Next**.

   The Create SQL Tuning Set: Filter Options page is shown.

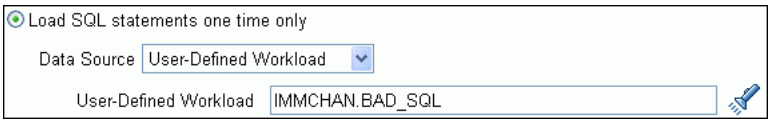
5. Proceed to the next step, as described in "Creating a SQL Tuning Set: Filter Options" on page 10-9.

Loading SQL Statements from a User-Defined Workload You can load SQL statements by importing from a table or view. This is useful if the workload you want to analyze is not currently running on the database, or captured in an existing AWR snapshot or a preserved snapshot set.

There are no restrictions on which schema the workload resides in, the name of the table, or the number of tables that you can define. The only requirement is that the format of the table must match the `USER_WORKLOAD` table.

To load SQL statements from a user-defined workload:

1. On the Create SQL Tuning Set: Load Methods page, select **Load statements one time only**.
2. In the Data Source field, select **User-Defined Workload**.
3. In the User-Defined Workload field, select the table or view to include.



4. Click **Next**.

   The Create SQL Tuning Set: Filter Options page is shown.

5. Proceed to the next step, as described in "Creating a SQL Tuning Set: Filter Options" on page 10-9.

Creating a SQL Tuning Set: Filter Options

After the load method is selected, you can apply filters to reduce the scope of the SQL statements found in the SQL tuning set. While using filters is optional, it can be very beneficial due to the following benefits:

- Using filters directs the various advisors that uses the SQL tuning set as a workload source—such as the SQL Tuning Advisor, the SQL Access Advisor, and the SQL Performance Analyzer—to make recommendations based on a specific subset of SQL statements, which may lead to better recommendations.
- Using filters removes extraneous SQL statements from the SQL tuning set, which may greatly reduce processing time when it is used as a workload source for the various advisors.

Tip: Before you can specify the filter options for the SQL tuning set, you need to:

- Create a SQL tuning set and specify the initial options, as described in "Creating a SQL Tuning Set: Options" on page 10-5
- Select the load method, as described in "Creating a SQL Tuning Set: Load Method" on page 10-7

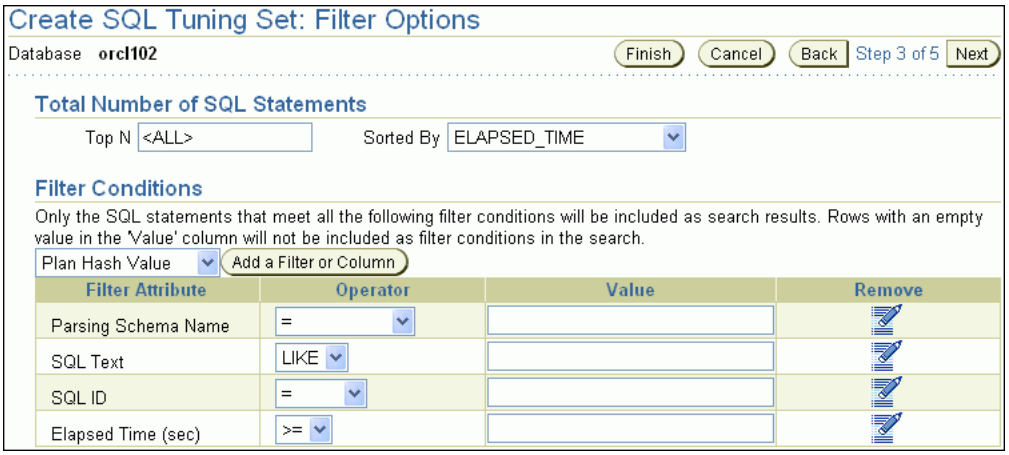
To specify filter options for a SQL tuning set:

1. On the Create SQL Tuning Set: Filter Options page, specify the values of filter conditions that you want use in the search in the Value column, and an operator or a condition in the Operator column.

   Only the SQL statements that meet all of the specified filter conditions will be added into the SQL tuning set. Unspecified filter values will not be included as filter conditions in the search.

   By default, the following filter conditions are displayed:

   - Parsing schema name
   - SQL text
   - SQL ID
   - Elapsed time (sec)



2. To add filter conditions, under Filter Conditions, select the filter condition you want to add and click **Add a Filter or Column**.

   The available filter conditions include:

   - Plan hash value
   - Module
   - Action
   - CPU time (sec)
   - Buffer gets
   - Disk reads
   - Disk writes
   - Rows processed

- Fetches
- Executions
- End of fetch count
- Command type

After the desired filter conditions have been added, specify their values in the Value column, and an operator or a condition in the Operator column.

3. To remove any unused filter conditions, click the icon in the **Remove** column for the corresponding filter condition you want to remove.
4. Click **Next**.

   The Create SQL Tuning Set: Schedule page appears.
5. Proceed to the next step, as described in "Creating a SQL Tuning Set: Schedule" on page 10-11.

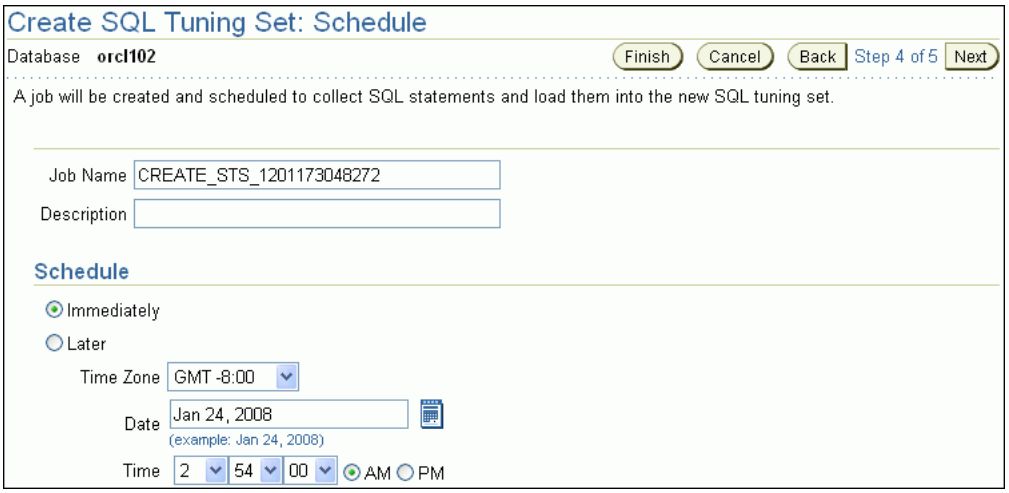
Creating a SQL Tuning Set: Schedule

After the filter options are specified for the SQL tuning set, you can schedule and submit a job to collect the SQL statements and load them into the SQL tuning set.

> **Tip:** Before you can schedule a job to create the SQL tuning set, you need to:
>
> - Create a SQL tuning set and specify the initial options, as described in "Creating a SQL Tuning Set: Options" on page 10-5
> - Select the load method, as described in "Creating a SQL Tuning Set: Load Method" on page 10-7
> - Specify the filter options, as described in "Creating a SQL Tuning Set: Filter Options" on page 10-9

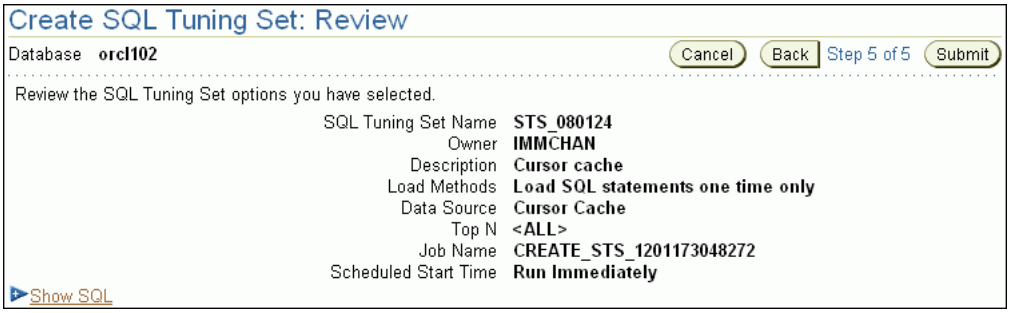
To schedule and submit a job to create a SQL tuning set:

1. On the Create SQL Tuning Set: Schedule page, enter a name for the job in the **Job Name** field if you do not want to use the system-generated job name.
2. In the **Description** field, enter a description of the job.
3. Under Schedule, select:
   - **Immediately** to run the job immediately after it has been submitted
   - **Later** to run the job at a later time as specified using the Time Zone, Date, and Time fields



4. Click **Next**.

   The Create SQL Tuning Set: Review page appears.



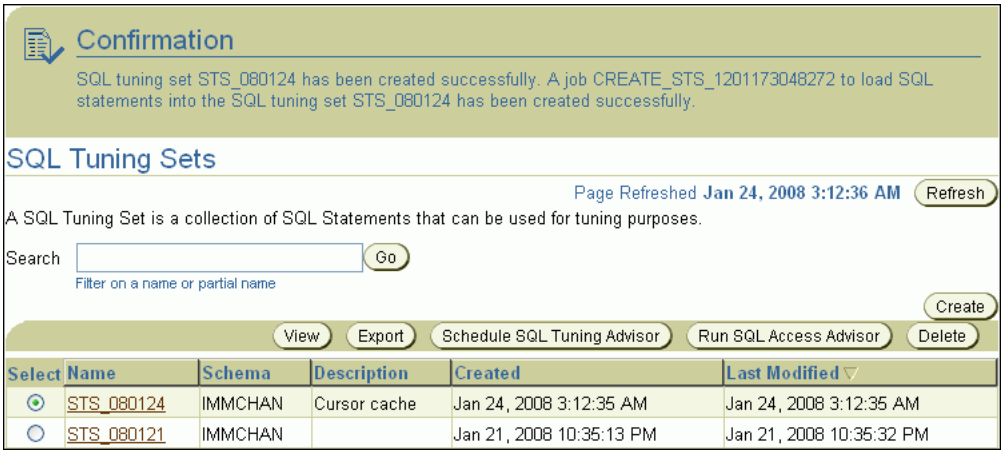
5. Review the SQL tuning set options that you have selected.

   To view the SQL statements used by the job, expand **Show SQL**.

6. Click **Submit**.

   The SQL Tuning Sets page appears.

   If the job was scheduled to run immediately, a message is displayed to inform you that the job and the SQL tuning set was created successfully. If the job was scheduled to run at a later time, a message is displayed to inform you that the job was created successfully.



7. To view details about the SQL tuning set, select the SQL tuning set and click **View**.

   The SQL Tuning Set page appears to display the SQL statements captured in the selected SQL tuning set.

Deleting a SQL Tuning Set

This section describes how to delete a SQL tuning set. To conserve storage space, you may want to periodically remove unused SQL tuning sets stored in the database.

To delete a SQL tuning set:

1. On the Database Home page, under Related Links, click **Advisor Central**.

   The Advisor Central page appears.

2. Under Advisors, click **SQL Tuning Advisor**.

   The SQL Tuning Advisor Links page appears.

3. Click **SQL Tuning Sets**.

   The SQL Tuning Sets page appears. Existing SQL tuning sets are displayed on this page.

4. Select the SQL tuning set you want to delete and click **Delete**.

   The Confirmation page appears to verify if you want to delete the selected SQL tuning set.

5. Click **Yes**.

   The SQL Tuning Sets page appears.

   A confirmation message is displayed to indicate that the SQL tuning set was successfully deleted.

Transporting a SQL Tuning Set

You can transport SQL tuning sets from one system to another by first exporting a SQL tuning set from one system, then importing it into another system.

To export a SQL tuning set:

1. On the Database Home page, under Related Links, click **Advisor Central**.

   The Advisor Central page appears.

2. Under Advisors, click **SQL Tuning Advisor**.

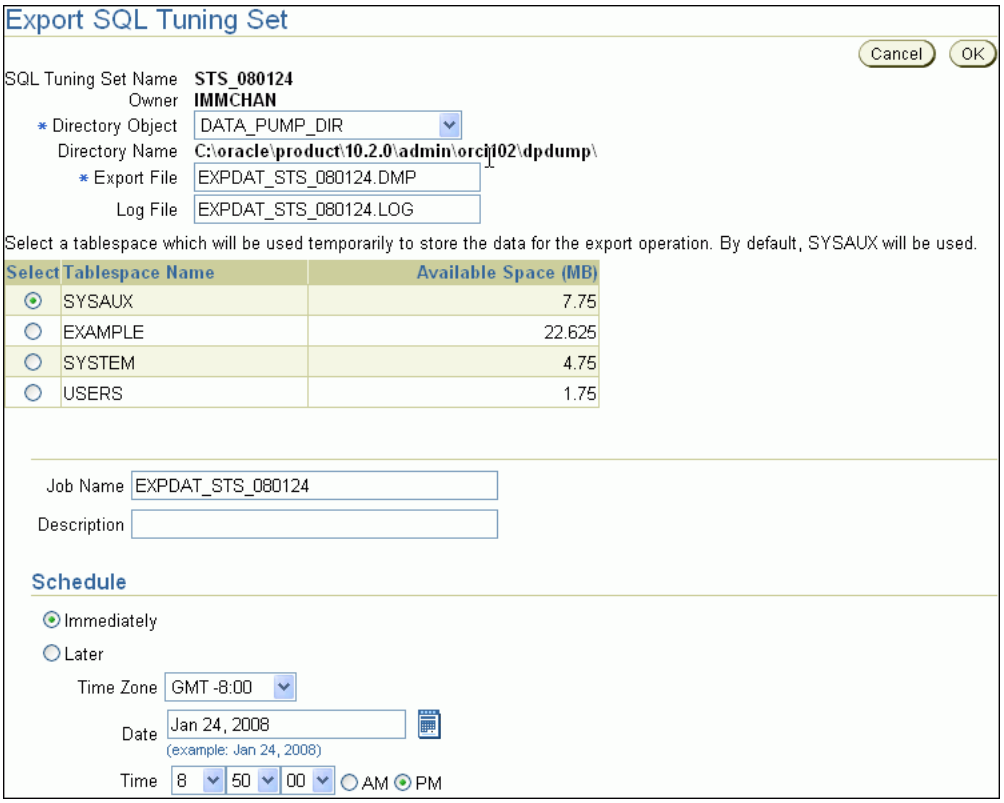
   The SQL Tuning Advisor Links page appears.

3. Click **SQL Tuning Sets**.

   The SQL Tuning Sets page appears. Existing SQL tuning sets are displayed on this page.

4. Select the SQL tuning set you want to export and click **Export**.

   The Export SQL Tuning Set page appears.



5. In the **Directory Object** field, select a directory where the export file will be created.

   For example, to use the Data Pump directory, select `DATA_PUMP_DIR`. The Directory Name field refreshes automatically to indicate the selected directory.

6. In the **Export File** field, enter a name for the dump file that will be exported.

   Alternatively, you can accept the name generated by the system.

7. In the **Log File** field, enter a name for the log file for the export operation.

   Alternatively, you can accept the name generated by the system.

8. Select a tablespace to temporarily store the data for the export operation.

   By default, `SYSAUX` is used.

9. In the **Job Name** field, enter a name for the job.

   Alternatively, you can accept the name generated by the system.

10. Under Schedule, select:

    - **Immediately** to run the job immediately after it has been submitted.
    - **Later** to run the job at a later time as specified by the values in the Time Zone, Date, and Time fields.

11. Click **OK**.

    The SQL Tuning Sets page appears.

    A confirmation message is displayed to indicate that the job was successfully created.

12. Transport the export file to another system using the mechanism of choice (such as Data Pump or database link).

Managing SQL Profiles

When running a SQL Tuning Advisor task with a limited scope, the query optimizer makes estimates about cardinality, selectivity, and cost. These estimates can sometimes be off by a significant amount, resulting in poor execution plans.

To address this problem, consider running a SQL Tuning Advisor task with a comprehensive scope to collect additional information using sampling and partial execution techniques to verify and, if necessary, adjust these estimates. These auxiliary statistics about the SQL statement are collected into a SQL profile.

During SQL profiling, the query optimizer uses the execution history information about the SQL statement to set appropriate settings for optimizer parameters. After the SQL profiling completes, the query optimizer uses the information stored in the SQL profile, in conjunction with regular database statistics, to generate execution plans. The availability of the additional information makes it possible to produce well-tuned plans for corresponding SQL statements.

After running a SQL Tuning Advisor task with a comprehensive scope, a SQL profile may be recommended. If you accept the recommendation, the SQL profile will be created and enabled for the SQL statement.

In some cases, you may want to disable a SQL profile. For example, you may want to test the performance of a SQL statement without using a SQL profile to determine if the SQL profile is actually beneficial. If the SQL statement is performing poorly after the SQL profile is disabled, you should enable it again to avoid performance deterioration. If the SQL statement is performing optimally after you have disabled the SQL profile, you may want to remove the SQL profile from your database.

To enable, disable, or delete a SQL profile:

1. On the Database Performance page, click **Top Activity**.

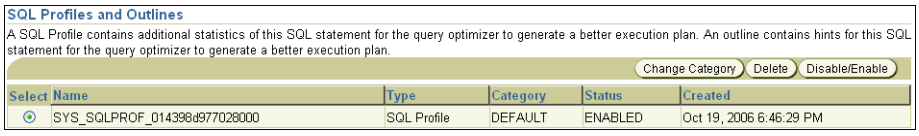
   The Top Activity page appears.

2. Under Top SQL, click the **SQL ID** link of the SQL statement this is using a SQL profile.

   The SQL Details page appears.

3. Click the **Tuning Information** tab.

   A list of SQL profiles are displayed under SQL Profiles and Outlines.



4. Select the SQL profile you want to manage. To:

   - Enable a SQL profile that is disabled, click **Disable/Enable**.
   - Disable a SQL profile that is enabled, **Disable/Enable**.
   - Remove a SQL profile, click **Delete**.

   A confirmation page appears.

5. Click **Yes** to continue, or **No** to cancel the action.

11

Optimizing Data Access Paths

To achieve optimum performance for data-intensive queries, materialized views and indexes are essential when tuning SQL statements. Implementing these objects, however, does not come without a cost. Creation and maintenance of these objects can be time consuming, and space requirements can be significant. The SQL Access Advisor enables you to optimize data access paths of SQL queries by recommending the proper set of materialized views, materialized view logs, and indexes for a given workload.

A materialized view provides access to table data by storing the results of a query in a separate schema object. Unlike an ordinary view, which does not take up any storage space or contain any data, a materialized view contains the rows resulting from a query against one or more base tables or views. A materialized view log is a schema object that records changes to a master table's data, so that a materialized view defined on the master table can be refreshed incrementally. The SQL Access Advisor recommends how to optimize materialized views so that they can be fast refreshable and take advantage of general query rewrite. For more information about materialized views and materialized view logs, see *Oracle Database Concepts*.

The SQL Access Advisor also recommends bitmap, function-based, and B-tree indexes. A bitmap index provides a reduced response time for many types of ad hoc queries and reduced storage requirements compared to other indexing techniques. A function-based index derives the indexed value from the table data. For example, to find character data in mixed cases, a function-based index can be used to look for the values as if they were all in uppercase characters. B-tree indexes are most commonly used to index unique or near-unique keys.

Using the SQL Access Advisor to recommend the proper set of materialized views, materialize view logs, and indexes for a SQL workload involves:

- Running the SQL Access Advisor
- Reviewing the SQL Access Advisor Recommendations
- Implementing the SQL Access Advisor Recommendations

See Also:

- Chapter 9, "Identifying High-Load SQL Statements"
- Chapter 10, "Tuning SQL Statements" for information about the SQL Tuning Advisor

Running the SQL Access Advisor

This section describes how to run the SQL Access Advisor to make recommendations on a SQL workload.

To run the SQL Access Advisor:

1. Select the initial options, as described in "Running the SQL Access Advisor: Initial Options" on page 11-2.
2. Select the workload source you want to use for the analysis, as described in "Running the SQL Access Advisor: Workload Source" on page 11-3.
3. Define the filters options, as described in "Running the SQL Access Advisor: Filter Options" on page 11-5.
4. Choose the types of recommendations, as described in "Running the SQL Access Advisor: Recommendation Options" on page 11-7.
5. Schedule the SQL Access Advisor task, as described in "Running the SQL Access Advisor: Schedule" on page 11-10.

Running the SQL Access Advisor: Initial Options

The first step in running the SQL Access Advisor is to select the initial options on the SQL Access Advisor: Initial Options page.

To select initial options:

1. On the Database Home page, under Related Links, click **Advisor Central**.

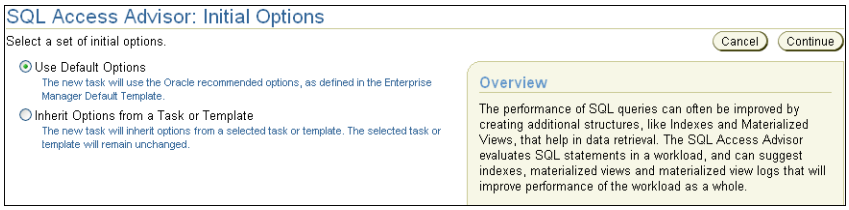
   The Advisor Central page appears.
2. Under Advisors, click **SQL Access Advisor**.

   The SQL Access Advisor: Initial Options page appears.
3. Select the initial options. To use the:
   - Recommended options defined in the Oracle Enterprise Manager default template, select **Use Default Options**.
   - Options defined in an existing SQL Access Advisor task or another template, select **Inherit Options from a Task or Template**.

     The Tasks and Templates section appears. In the View list, select to display **Templates Only**, **Tasks Only**, or **Tasks and Templates**.

     Select the task or template you want to use.

   In this example, **Use Default Options** is selected.



   Click **Continue**.

   The SQL Access Advisor: Workload Source page appears.
4. Proceed to the next step, as described in "Running the SQL Access Advisor: Workload Source" on page 11-3.

Running the SQL Access Advisor: Workload Source

After initial options are specified for the SQL Access Advisor, you need to select the workload source you want to use for the analysis, as described in the following sections:

- Using SQL Statements from the Cache
- Using an Existing SQL Tuning Set
- Using a User-Defined Workload
- Using a Hypothetical Workload

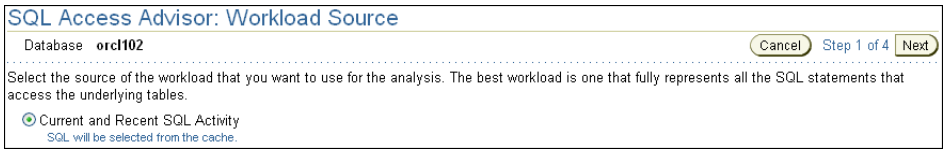
> **Tip:** Before you can select the workload source for the SQL Access Advisor, you need to select the initial options, as described in "Running the SQL Access Advisor: Initial Options" on page 11-2.

Using SQL Statements from the Cache

You can use SQL statements from the cache as the workload source. However, because only current and recent SQL statements are stored in the SQL cache, this workload source may not be representative of the entire workload on your database.

To use SQL statements from the cache as the workload source:

1. On the SQL Access Advisor: Workload Source page, select **Current and Recent SQL Activity**.



2. Proceed to the next step, as described in "Running the SQL Access Advisor: Filter Options" on page 11-5.

Using an Existing SQL Tuning Set

You can use an existing SQL tuning set as the workload source. This is useful because SQL tuning sets can be used repeatedly as the workload source for not only the SQL Access Advisor, but also the SQL Tuning Advisor.

To use a SQL tuning set as the workload source:

1. On the SQL Access Advisor: Workload Source page, select **Import Workload from SQL Repository**.
2. To display available SQL tuning sets, click the SQL Tuning Set search icon.

   The Search and Select: SQL Tuning Set dialog box appears.
3. In the Schema field, enter the name of the schema containing the SQL tuning set you want to use and click **Go**.

   A list of SQL tuning sets contained in the selected schema is displayed.
4. Select the SQL tuning set you want to use for the workload source and click **Select**.

   The Search and Select: SQL Tuning Set dialog box closes and the selected SQL tuning set now appears in the SQL Tuning Set field.

5. Proceed to the next step, as described in "Running the SQL Access Advisor: Filter Options" on page 11-5.

Using a User-Defined Workload

You can create a workload source by importing SQL statements from a table or view. This is useful if the workload you want to analyze is not currently running on the database or captured in an existing SQL tuning set. There are no restrictions on which schema the workload resides in, the name of the table, or the number of tables that you can define. The only requirement is that the format of the table must match the `USER_WORKLOAD` table.

See Also:

- *Oracle Database Performance Tuning Guide* for information about creating a user-defined table or view that can be used as the workload source

To use a user-defined workload as the workload source:

1. On the SQL Access Advisor: Workload Source page, select **User-Defined Workload; Import SQL from a Table or View**.
2. To display available tables and views, click the Table search icon.

   The Search and Select: User Defined Workload dialog box appears.
3. In the Schema field, enter the name of the schema containing the table or view you want to use and click **Go**.

   A list of tables and views in the selected schema is displayed.
4. Select the table or view you want to use for the workload source and click **Select**.

   The Search and Select: User Defined Workload dialog box closes and the selected table or view now appears in the Table field.
5. Proceed to the next step, as described in "Running the SQL Access Advisor: Filter Options" on page 11-5.

Using a Hypothetical Workload

You can create a hypothetical workload from dimension tables containing primary or foreign key constraints. This is useful if the workload you want to analyze does not exist. In this case, the SQL Access Advisor will examine the current logical schema design, and provide recommendations based on the defined relationships between tables.

To use a hypothetical workload as the workload source:

1. On the SQL Access Advisor: Workload Source page, select **Create a Hypothetical Workload from the Following Schemas and Tables**.
2. To search for tables, click the Table search icon.

   The Search and Select: Schema and Table dialog box appears.
3. In the Schema field, enter the name of the schema you want to search and click **Go**.

   A list of tables in the selected schema is displayed.
4. Select the tables you want to use in creating the hypothetical workload and click **Select**.

The Search and Select: Schema and Table dialog box closes and the selected tables now appear in the Table field.

5. Proceed to the next step, as described in "Running the SQL Access Advisor: Filter Options" on page 11-5.

Running the SQL Access Advisor: Filter Options

After the workload source is selected, you can apply filters to reduce the scope of the SQL statements found in the workload. While using filters is optional, it can be very beneficial due to the following benefits:

- Using filters directs the SQL Access Advisor to make recommendations based on a specific subset of SQL statements from the workload, which may lead to better recommendations.
- Using filters removes extraneous SQL statements from the workload, which may greatly reduce processing time.

> **Tip:** Before you can select the filter options for the workload, you need to:
>
> - Select initial options, as described in "Running the SQL Access Advisor: Initial Options" on page 11-2
> - Select the workload source, as described in "Running the SQL Access Advisor: Workload Source" on page 11-3

To apply filters to the workload source:

1. On the SQL Access Advisor: Workload Source page, click **Filter Options**.

   The Filter Options section expands.

2. Select **Filter Workload Based on these Options**.

   The Filter Options section is enabled.

3. Define the filters you want to apply, as described in the following sections:
   - Defining Filters for Resource Consumption
   - Defining Filters for Users
   - Defining Filters for Tables
   - Defining Filters for SQL Text
   - Defining Filters for Module ID
   - Defining Filters for Actions
4. Click **Next**.

   The Recommendation Options page appears.

5. Proceed to the next step, as described in "Running the SQL Access Advisor: Recommendation Options" on page 11-7.

Defining Filters for Resource Consumption

The resource consumption filter restricts the workload to only include a number of high-load SQL statements that you specify.

To define a filter for resource consumption:

1. User Resource Consumption, enter the number of high-load SQL statements in the Number of Statements field.
2. From the Order by list, select one of the method by which the SQL statements are to be ordered:
   - Optimizer cost
   - Buffer gets
   - CPU Time
   - Disk Reads
   - Elapsed Time
   - Executions

Defining Filters for Users

The users filter restricts the workload to include or exclude SQL statements executed by users that you specify.

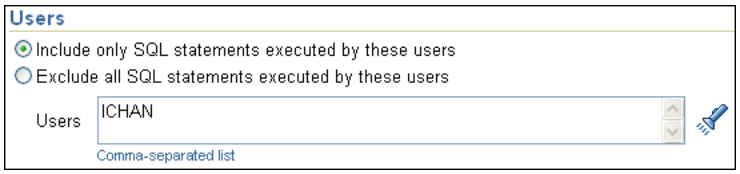
To define a filter for users:

1. Under Users, select **Include only SQL statements executed by these users** or **Exclude all SQL statements executed by these users**.
2. To search for available users, click the Users search icon.

   The Search and Select: Users dialog box appears.
3. Select the users for which you want to include or exclude SQL statements and click **Select**.

   The Search and Select: Users dialog box closes and the selected tables now appear in the Users field.

In this example, a filter is defined to include only SQL statements executed by the user `ICHAN`.



Defining Filters for Tables

The tables filter restricts the workload to include or exclude SQL statements that access a list of tables that you specify.

To define a filter for tables:

1. To include only SQL statements that access a specific list of tables, enter the table names in the **Include only SQL statements that access any of these tables** field.
2. To exclude all SQL statements that access a specific list of tables, enter the table names in the **Exclude all SQL statements that access any of these tables** field.
3. To search for available users, click the Tables search icon.

   The Search and Select: Schema and Table dialog box appears.

4. Select the tables for which you want to include or exclude SQL statements and click **Select**.

   The Search and Select: Schema and Table dialog box closes and the selected tables now appear in the corresponding Tables field.

Defining Filters for SQL Text

The SQL text filter restricts the workload to include or exclude SQL statements that contains SQL text substrings that you specify.

To define a filter for SQL text:

1. To include only SQL statements that contains specific SQL text, enter the SQL text to be included in the **Include only SQL statements containing these SQL text substrings** field.
2. To exclude all SQL statements that contains specific SQL text, enter the SQL text to be excluded in the **Exclude all SQL statements containing these SQL text substrings** field.

Defining Filters for Module ID

The module ID filter restricts the workload to include or exclude SQL statements that are associated with modules IDs that you specify.

To define a filter for module ID:

1. To include only SQL statements associated to a specific module ID in the workload, select **Include only SQL statements associated with these module IDs**.
2. To exclude all SQL statements associated to a specific module ID from the workload, select **Exclude all SQL statements associated with these module IDs**.
3. In the Module IDs field, enter the module IDs for which associated SQL statements will be included or excluded.

Defining Filters for Actions

The actions filter restricts the workload to include or exclude SQL statements that are associated with actions that you specify.

To define a filter for actions:

1. To include only SQL statements associated to a specific action in the workload, select **Include only SQL statements associated with these actions**.
2. To exclude all SQL statements associated to a specific action from the workload, select **Exclude all SQL statements associated with these actions**.
3. In the Actions field, enter the actions for which associated SQL statements will be included or excluded.

Running the SQL Access Advisor: Recommendation Options

To improve the underlying data access methods chosen by the optimizer for the workload, the SQL Access Advisor provides recommendation for indexes, materialized views, or both indexes and materialized views. Using these access structures can significantly improve the performance of the workload by reducing the time required to read data from the database. However, you must balance the benefits of using these access structures against the cost to maintain them.

Tip: Before you can select the recommendation options for the SQL Access Advisor, you need to:

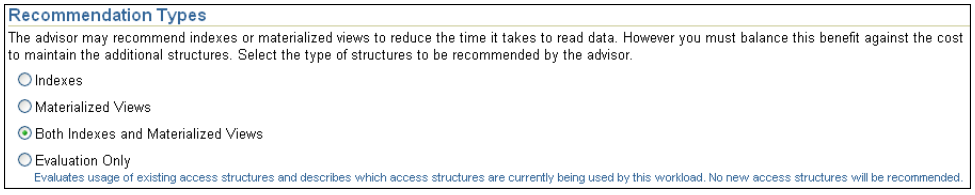
- Select initial options, as described in "Running the SQL Access Advisor: Initial Options" on page 11-2
- Select the workload source, as described in "Running the SQL Access Advisor: Workload Source" on page 11-3
- Define the filter options, as described in "Running the SQL Access Advisor: Filter Options" on page 11-5

To specify recommendation options:

1. On the SQL Access Advisor: Recommended Options page, under Recommendation Types, select the type of access structures to be recommended by the SQL Access Advisor:
   - Indexes
   - Materialized Views
   - Indexes and Materialized Views
   - Evaluation Only

     Choose this option to evaluate only existing access structures. New access structures will not be recommended for the workload.

   In this example, **Indexes and Materialized Views** is selected.



2. Under Advisor Mode, select the mode in which the SQL Access Advisor will run:
   - Limited Mode

     In limited mode, the SQL Access Advisor focuses on SQL statements with the highest cost in the workload. The analysis is quicker, but the recommendations may be limited.
   - Comprehensive Mode

     In comprehensive mode, the SQL Access Advisor analyzes all SQL statements in the workload. The analysis can take much longer, but the recommendations will be exhaustive.
   - Advanced Options

     To use advanced options, click **Advanced Options**. The Advanced Options section expands. You can specify advanced options for:

     – Workload Categorization

       Under Workload Volatility, you can choose whether or not to allow the SQL Access Advisor to consider the volatility of referenced objects when forming recommendations. Using volatility data is useful for online transaction processing (OLTP) systems, where the performance of `INSERT`, `UPDATE`, and `DELETE` operations is critical. On the other hand, if the

workload contains primarily read-only operations, such is the case in database warehouses, volatility data should not be used.

Under Workload Scope, select **Partial Workload** to exclude volatility data, or **Complete Workload** to include volatility data.

– Space Restrictions

Indexes and materialized views increase performance at the cost of space. When the SQL Access Advisor is invoked with no space limits, it will make the best possible performance recommendations. To specify no space limit, under Space Restrictions, select **No, show me all recommendations (unlimited space)**.

When the SQL Access Advisor is invoked with a space limit, it will produce only recommendations with space requirements that do not exceed the specified limit. To specify a space limit, under Space Restrictions, select **Yes, space is limited**. In the Space Adjustment field, enter the space limit in Megabytes, Gigabytes, or Terabytes.

– Tuning Options

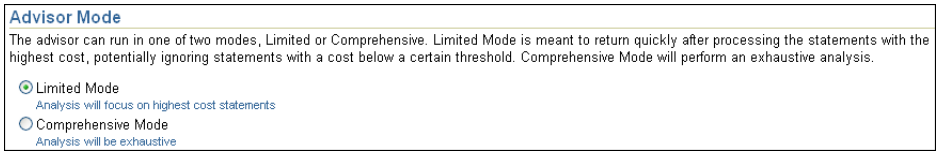
The Tuning Options section enables you to specify how SQL statements will be tuned. From the Prioritize Tuning of SQL Statements by list, select the method by which SQL statements are to be tuned: Optimizer Cost, Buffer Gets, CPU Time, Disk Reads, Elapsed Time, or Executions.

To weigh the cost of creating access structures against the frequency and potential improvement of SQL statement execution time, check the **Allow Advisor to consider creation costs when forming recommendations** box. Otherwise, creation cost will be ignored. You should check this box if you want specific recommendations generated for SQL statements that are executed frequently.

– Default Storage Locations

Use the Default Storage Locations section to override the defaults defined for schema and tablespace locations. By default, indexes are placed in the schema and tablespace of the table they reference. Materialized views are placed in the schema and tablespace of the first table referenced in the query. Materialized view logs are placed in the default tablespace of the schema of the table they reference.

In this example, **Limited Mode** is selected.



3. Click **Next**.

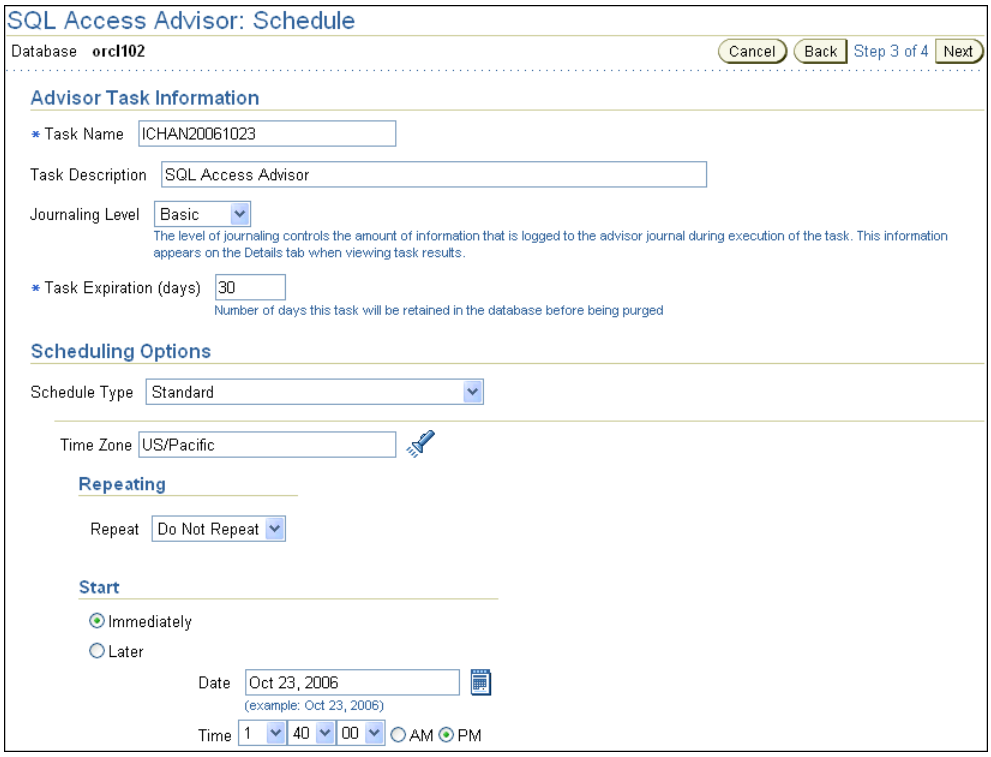
   The Schedule page appears.

4. Proceed to the next step, as described in "Running the SQL Access Advisor: Schedule" on page 11-10.

Running the SQL Access Advisor: Schedule

Use the SQL Access Advisor Schedule page, shown in Figure 11–1, to set or modify the schedule parameters for the SQL Access Advisor task.

Figure 11–1 Scheduling a SQL Access Advisor Task



Tip: Before you can schedule a SQL Access Advisor task, you need to:

- Select initial options, as described in "Running the SQL Access Advisor: Initial Options" on page 11-2
- Select the workload source, as described in "Running the SQL Access Advisor: Workload Source" on page 11-3
- Define the filter options, as described in "Running the SQL Access Advisor: Filter Options" on page 11-5
- Specify the recommendation options, as described in "Running the SQL Access Advisor: Recommendation Options" on page 11-7

To schedule a SQL Access Advisor task:

1. On the SQL Access Advisor: Schedule page, under Advisor Task Information, enter a name in the Task Name field if you do not want to use the system-generated task name.

   In the example shown in Figure 11–1, `ICHAN20061023` is entered.

2. In the Task Description field, enter a description of the task.

   In the example shown in Figure 11–1, `SQL Access Advisor` is entered.

3. From the Journaling Level list, select the level of journaling for the task.

Journaling level controls the amount of information that is logged to the SQL Access Advisor journal during task execution. This information appears on the Details subpage when viewing task results. Available journaling levels include:

- None

  No information is logged.

- Basic

  Information such as errors, warnings, and roll-ups of activities are logged.

- Moderate

  Information such as major progress steps, SQL statements skipped, SQL statements processed, and analysis details are logged.

- Full

  All available information is logged.

In the example shown in Figure 11–1 on page 11-10, **Basic** is selected.

4. In the Task Expiration (Days) field, enter the number of days the task will be retained in the database before it is purged.

   In the example shown in Figure 11–1 on page 11-10, `30` is entered.

5. Under Scheduling Options, in the Schedule Type list, select a schedule type for the task and a maintenance window in which you want the task to run.

   The available schedule types include:

   - Standard

     This schedule type enables you to select a repeating interval and start time for the task.

     Under Repeating, select the repeating interval using the Repeat and Interval fields, or **Do Not Repeat** if you do not want the task to repeat automatically.

     Under Start, select **Immediately** to start the task immediately, or **Later** to schedule the task to start at a time specified using the Date and Time fields.

   - User pre-defined schedule

     This schedule type enables you to select an existing schedule to use.

     In the Schedule field, enter the name of the schedule you want to use for the task. To search for a schedule, click the Schedule search icon. The Search and Select: Schedule dialog box appears. Select the schedule you want to use and click **Select**. The selected schedule now appears in the Schedule field. To display details about the schedule, click **View Details**.

   - Standard using PL/SQL for repeated interval

     This schedule types enables you to select a repeating interval and an execution window for the task.

     Under Available to Start, select **Immediately** to make the task available to start immediately, or **Later** to schedule the task to become available at a time specified using the Date and Time fields.

     Under Repeated Interval, enter a PL/SQL schedule expression, such as `SYSDATE+1`.

Under Not Available After, select **No End Date** if you do not want to use an end date for the execution window, or **Specified End Date** to specify an end date using the Date and Time fields.

- User pre-defined window

  In the Window field, select a window to run the task during specific time periods. You can run the task in multiple windows by grouping the windows in a window group.

  To search for a window, click the Window search icon. The Search and Select: Window and Window Groups dialog box appears. Select the window or window group you want to use and click **Select**. The selected window or window group now appears in the Window field.

  To stop the task if the available window or window group closes, check the **Stop on Window Close** box.

In the example shown in Figure 11–1 on page 11-10, **Standard** is selected for schedule type. The task will not repeat and is scheduled to start immediately.

6. Click **Next**.

   The Review page appears. Under Options, a list of all options for the SQL Access Advisor task is displayed. Options that are modified can be identified by the check marks in the Modified column. The value for each option is displayed in the Value column. To view the SQL text for the task, click **Show SQL**.

Options

Modified	Option	Value	Description
	Advisor Mode	Limited Mode	Specifies the mode in which SQL Access Advisor will operate during an analysis, either limited (quicker results) or comprehensive (higher quality recommendations)
	Creation Cost	Consider creation cost	When specified, the SQL Access Advisor will weigh the cost of creation of access structures against the frequency of the queries and potential improvement in query execution time
	Default Index Schema	None	Specifies the default owner for new index recommendations
	Default Index Tablespace	None	Specifies the default tablespace for new index recommendations
	Default MVLog Tablespace	None	Specifies the default tablespace for new materialized view log recommendations
	Default MView Schema	None	Specifies the default owner for new materialized view recommendations
	Default MView Tablespace	None	Specifies the default tablespace for new materialized view recommendations
	Excluded Actions	None	Contains a list of application actions that are NOT eligible for tuning
	Excluded Module IDs	None	Contains a list of application modules that are NOT eligible for tuning
	Excluded SQL Text	None	Contains a list of text items that can appear in SQL statements. If a statement does contain one the specified items, then it is NOT eligible for tuning
	Excluded Tables	None	Contains a fully qualified list of tables that are NOT eligible for tuning
	Excluded Users	None	Contains a list of users who execute SQL statements that are NOT eligible for tuning
	Included Actions	None	Contains a list of application actions that are eligible for tuning
	Included Module IDs	None	Contains a list of application modules that are eligible for tuning
	Included SQL Text	None	Contains a list of text items that can appear in SQL statements. If a statement does not contain one the specified items, then it is not eligible for tuning
	Included Tables	None	Contains a fully qualified list of tables that are eligible for tuning
✓	Included Users	ICHAN	Contains a list of users who execute SQL statements that are eligible for tuning
	Journaling Level	Basic	Controls the logging of messages to the advisor journal
✓	Recommendation Type	All Methods	The type of recommendations that are desired
	SQL Priority Statistic	Optimizer Cost	Contains the primary natural order in which SQL Access Advisor processes workload elements during the analysis operation
	Space Adjustment	Unlimited	Added or subtracted from current space usage to determine the amount of space that can be consumed by SQL Access Advisor recommendations
	Task Expiration (days)	30	Specifies the expiration time in days for the current SQL Access Advisor task
	Workload SQL Filter	Optimizer Cost	Contains the primary natural order in which SQL Access Advisor processes workload elements during the workload import operation
✓	Workload SQL Limit	25	Specifies the number of SQL statements to be analyzed
	Workload Scope	Partial Workload	Describes the level of application coverage the workload represents
	Workload Source	SQL Cache	The source of SQL statements to be used to create the workload
	Workload Volatility	Allow Advisor to consider the volatility of referenced objects when forming recommendations	When specified, the SQL Access Advisor will consider the impact of index maintenance and materialized view refresh in determining the recommendations

7. After reviewing the options on the Review page, click **Submit** to submit the SQL Access Advisor task.

   The Advisor Central page appears and a message is displayed to inform you that the task was created successfully.

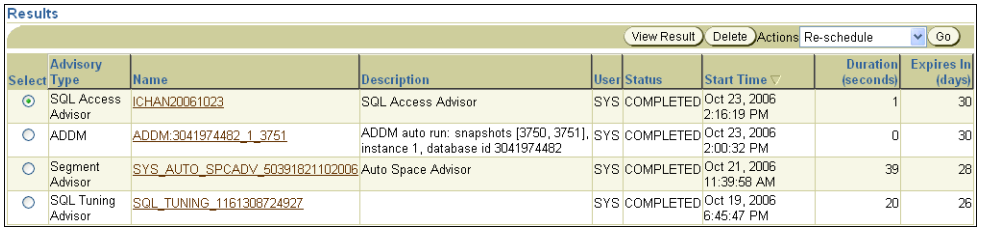
Reviewing the SQL Access Advisor Recommendations

This section describes how to review the SQL Access Advisor recommendations.

> **Tip:** Before reviewing the SQL Access Advisor recommendations, you need to run the SQL Access Advisor to make the recommendations, as described in "Running the SQL Access Advisor" on page 11-1.

To review the SQL Access Advisor recommendations:

1. On the Advisor Central page, select the SQL Access Advisor task you want to review and click **View Result**.



Select	Advisory Type	Name	Description	User	Status	Start Time	Duration (seconds)	Expires In (days)
◉	SQL Access Advisor	ICHAN20061023	SQL Access Advisor	SYS	COMPLETED	Oct 23, 2006 2:16:19 PM	1	30
○	ADDM	ADDM:3041974482_1_3751	ADDM auto run: snapshots [3750, 3751], instance 1, database id 3041974482	SYS	COMPLETED	Oct 23, 2006 2:00:32 PM	0	30
○	Segment Advisor	SYS_AUTO_SPCADV_50391821102006	Auto Space Advisor	SYS	COMPLETED	Oct 21, 2006 11:39:58 AM	39	28
○	SQL Tuning Advisor	SQL_TUNING_1161308724927		SYS	COMPLETED	Oct 19, 2006 6:45:47 PM	20	26

   If the task is not displayed, you may need to refresh the screen. The Results for Task page appears.

2. Review the Summary subpage, which provides an overview of the SQL Access Advisor analysis, as described in "Reviewing the SQL Access Advisor Recommendations: Summary" on page 11-13.

3. Review the Recommendations subpage, which enables you to view the recommendations ranked by cost improvement, as described in "Reviewing the SQL Access Advisor Recommendations: Recommendations" on page 11-15.

4. Review the SQL statements analyzed in the workload, as described in "Reviewing the SQL Access Advisor Recommendations: SQL Statements" on page 11-16.

5. Review the details of the workload, task options, and the SQL Access Advisor task, as described in "Reviewing the SQL Access Advisor Recommendations: Details" on page 11-17.

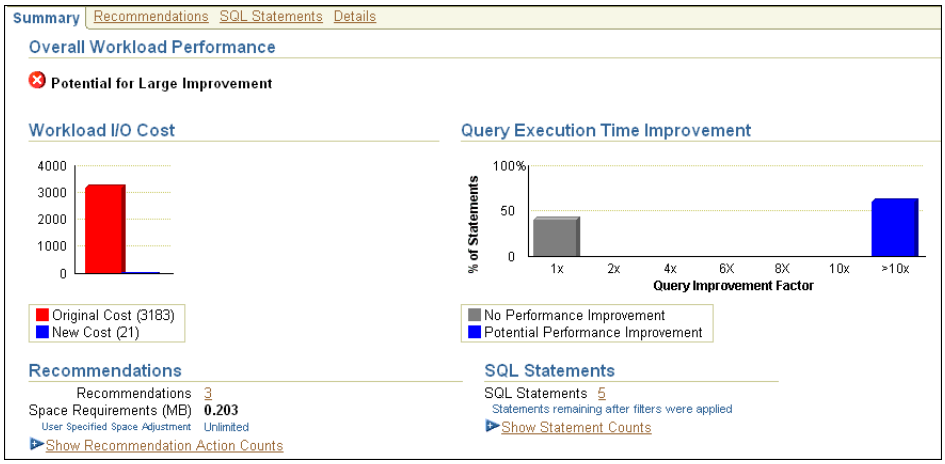
Reviewing the SQL Access Advisor Recommendations: Summary

The Summary subpage displays an overview of the SQL Access Advisor analysis.

To review recommendations summary:

1. On the Results for Tasks page, click **Summary**.

   The Summary subpage is displayed.



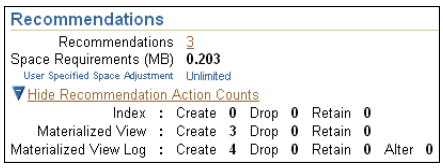
2. Under Overall Workload Performance, assess the potential for improvement in implementing the recommendations.
3. Use the Workload I/O Cost chart to compare the original workload I/O cost (in red) with the new cost (in blue).

   In this example, the workload I/O cost will improve from 3183 to 21 by implementing the recommendations.
4. Use the Query Execution Time Improvement chart to compare the improvement in query execution time.

   The Query Execution Time Improvement chart shows the percentage of SQL statements in the workload whose execution time will improve by accepting the recommendations. The SQL statements are grouped by the projected improvement factor along the horizontal axis on the chart (1x to >10x). The percentage of SQL statements that will improve by the projected improvement factor are calculated along the vertical axis (0% to 100%).

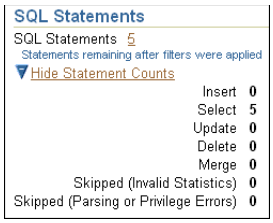
   In this example, approximately 45 percent of SQL statements in the workload will gain a minimal performance improvement in execution time, but 55 percent will gain a significant improvement of over 10x.
5. Under Recommendations, the number of recommendations made by the SQL Access Advisor and their space requirements are displayed.

   To display the type of recommendations, click **Show Recommendation Action Counts**.



   In this example, creating 3 materialized views and 4 materialized view logs are recommended.
6. Under SQL Statements, the number of SQL statements analyzed in the workload is displayed.

   To display the types of SQL statement, click **Show SQL Statement Counts**.



In this example, 5 SELECT statements are analyzed.

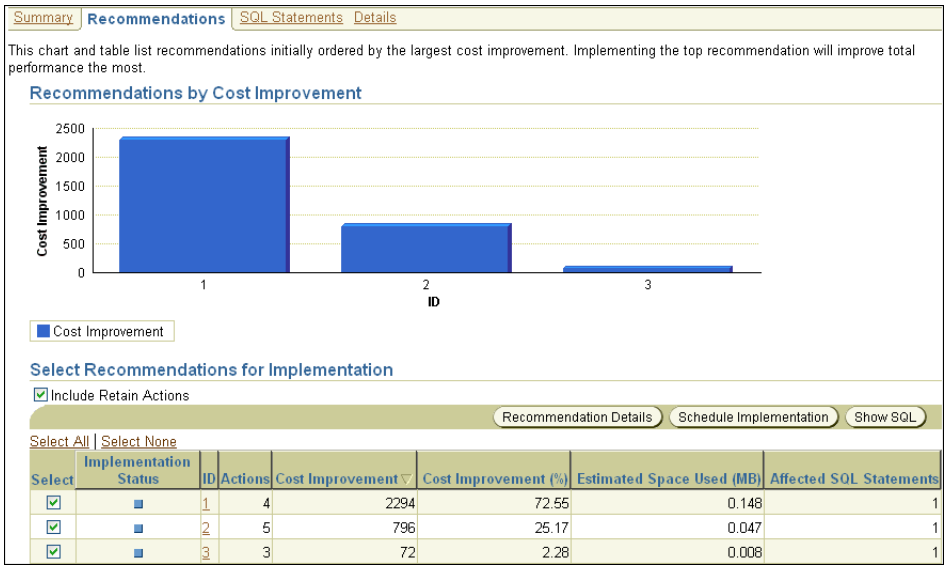
Reviewing the SQL Access Advisor Recommendations: Recommendations

The Recommendations subpage ranks the SQL Access Advisor recommendations by cost improvement. You can also view details about each recommendation using this page.

To review recommendation details:

1. On the Results for Tasks page, click **Recommendations**.

   The Recommendations subpage is displayed.



2. Use the Recommendations by Cost Improvement chart to view recommendations ordered by the cost improvement.

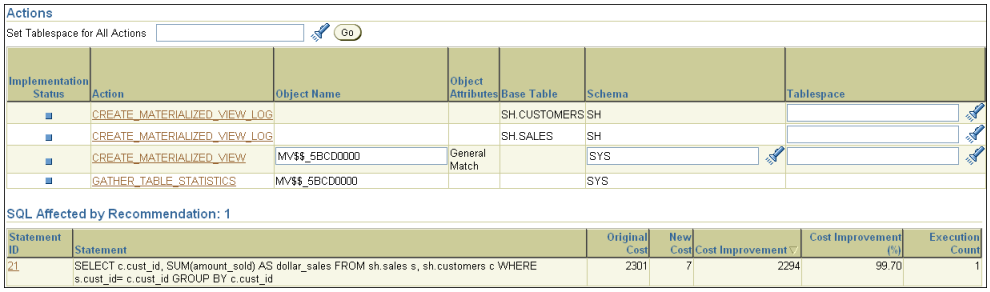
   Implementing the top recommendation will have the biggest benefit to the total performance of the workload.

3. Under Select Recommendations for Implementation, each recommendation is listed with its implementation status, recommendation ID, cost improvement, space consumption, and the number of affected SQL statement for each recommendation.

   In this example, implementing the recommendation with ID 1 will produce the biggest benefit, with a cost improvement of 72.55 percent, on the workload.

4. To view details for a particular recommendation, select the recommendation and click **Recommendation Details**.

The Recommendation Details page appears.



5. The Recommendation Details page displays all actions for the specified recommendation.

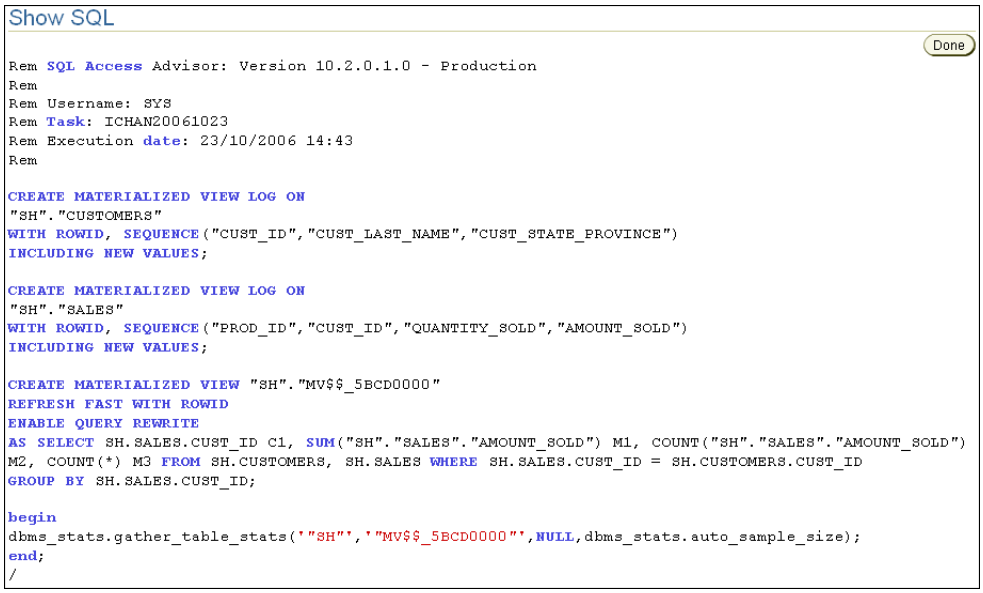
   Under Actions, you can choose to modify the schema name, tablespace name, and storage clause for each action. To view the SQL text of an action, click the link in the Action column for the specified action.

   Under SQL Affected by Recommendation, the SQL text of the SQL statement and cost improvement information are displayed.

   Click **OK**. The Recommendations subpage is displayed.

6. To view the SQL text of a recommendation, select the recommendation and click **Show SQL**.

   The Show SQL page for the selected recommendation appears.



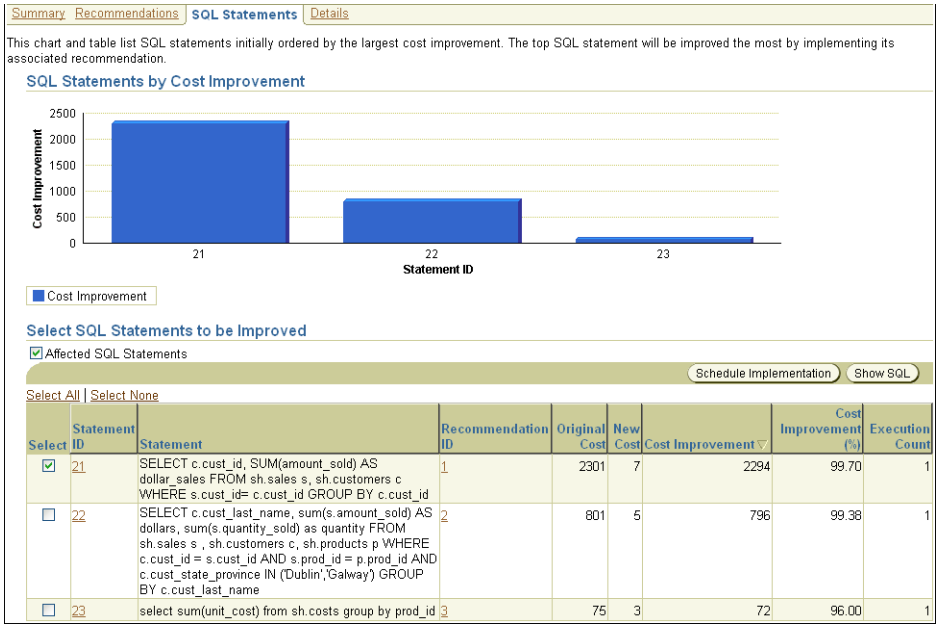
Reviewing the SQL Access Advisor Recommendations: SQL Statements

The SQL Statements subpage ranks SQL statements in the workload by cost improvement. You can use this page to view details about the SQL statements analyzed in the workload.

To review SQL statements:

1. On the Results for Tasks page, click **SQL Statements**.

The SQL Statements subpage is displayed.



2. Use the SQL Statements by Cost Improvement chart to view SQL statements in the workload ordered by the cost improvement.

   Implementing the recommendation associated with the top SQL statement will have the biggest benefit to the total performance of the workload.

3. Under Select SQL Statements to be Improved, each SQL statement is listed with its statement ID, SQL text, associated recommendation, cost improvement, and execution count.

   In this example, implementing the recommendation with ID 1 will produce the biggest benefit, a cost improvement of 99.7 percent, for the SQL statement with ID 21.

4. To view the SQL text of a recommendation, select the recommendation and click **Show SQL**.

   The Show SQL page for the selected recommendation appears.

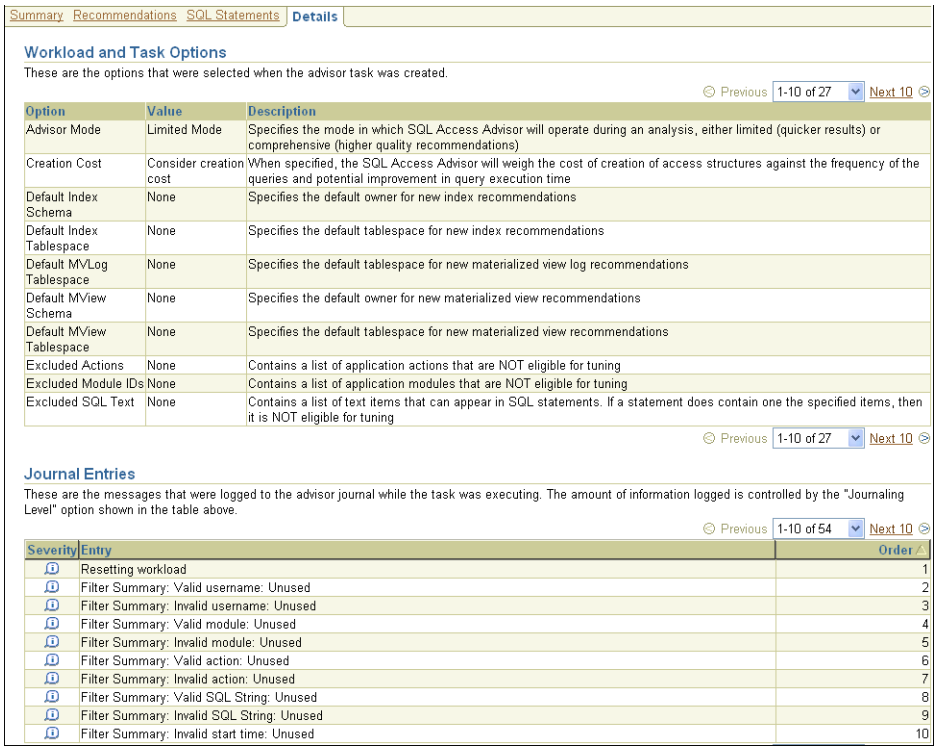
Reviewing the SQL Access Advisor Recommendations: Details

The Details subpage displays a list of all the workload and task options used in the analysis. You can also use this subpage to view a list of journal entries for the task, based on the journaling level used when the task was created.

To review workload and task details:

1. On the Results for Tasks page, click **Details**.

   The Details subpage is displayed.



2. Under Workload and Task Options, a list of options that were selected when the advisor task was created is displayed.
3. Under Journal Entries, a list of messages that were logged to the SQL Access Advisor journal while the task was executing is displayed.

Implementing the SQL Access Advisor Recommendations

This section describes how to implement the SQL Access Advisor recommendations.

> **Tip:** Before implementing the SQL Access Advisor recommendations, you need to review them for cost benefits to determine which ones, if any, should be implemented. For more information, see "Reviewing the SQL Access Advisor Recommendations" on page 11-13.

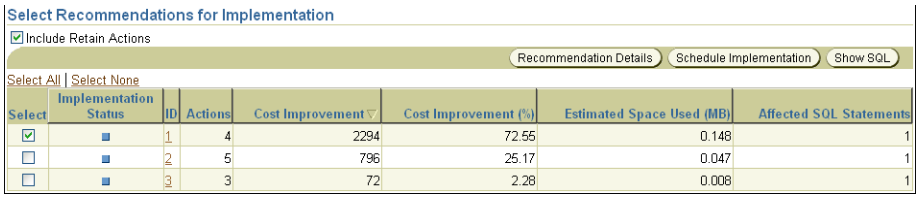
To implement the SQL Access Advisor recommendations:

1. On the Results for Tasks page, click **Recommendations**.

   The Recommendations subpage is displayed.

2. Under Select Recommendations for Implementation, select the recommendation you want to implement and click **Schedule Implementation**.

   In this example, the recommendation with ID value 1 is selected.



The Schedule Implementation page appears.

3. In the Job Name field, enter a name for the job if you do not want to use the system-generated job name.
4. Determine whether or not the implementation job should stop if an error is encountered:
   - To stop processing if an error occurs, check the **Stop on Error** box.
   - To continue processing even if an error occurs, uncheck the **Stop on Error** box.
5. Under Scheduling Options, in the Schedule Type list, select a schedule type for the implementation job and a maintenance window in which you want the job to run.

   The available schedule types include:

   - Standard

     This schedule type enables you to select a repeating interval and start time for the job.

     Under Repeating, select the repeating interval using the Repeat and Interval fields, or **Do Not Repeat** if you do not want the job to repeat automatically.

     Under Start, select **Immediately** to start the job immediately, or **Later** to schedule the job to start at a time specified using the Date and Time fields.

   - User pre-defined schedule

     This schedule type enables you to select an existing schedule to use.

     In the Schedule field, enter the name of the schedule you want to use for the job. To search for a schedule, click the Schedule search icon. The Search and Select: Schedule dialog box appears. Select the schedule you want to use and click **Select**. The selected schedule now appears in the Schedule field. To display details about the schedule, click **View Details**.

   - Standard using PL/SQL for repeated interval

     This schedule types enables you to select a repeating interval and an execution window for the job.

     Under Available to Start, select **Immediately** to make the job available to start immediately, or **Later** to schedule the job to become available at a time specified using the Date and Time fields.

     Under Repeated Interval, enter a PL/SQL schedule expression, such as `SYSDATE+1`.

     Under Not Available After, select **No End Date** if you do not want to use an end date for the execution window, or **Specified End Date** to specify an end date using the Date and Time fields.

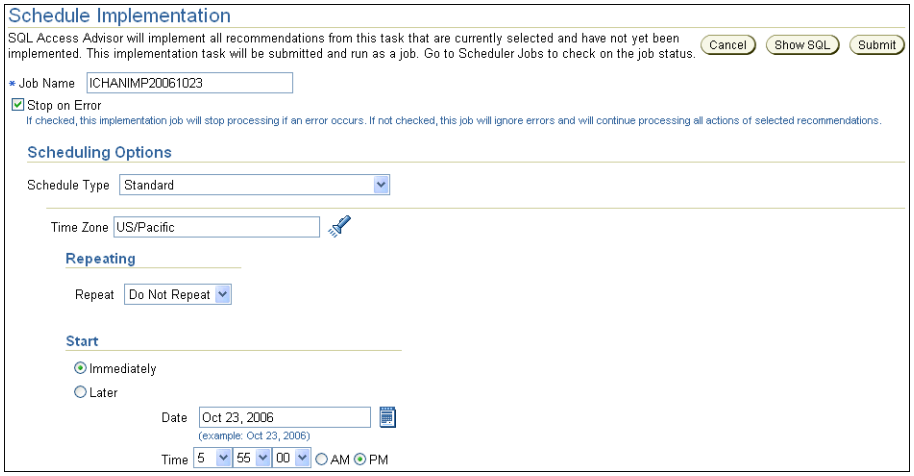
   - User pre-defined window

In the Window field, select a window to run the job during specific time periods. You can run the job in multiple windows by grouping the windows in a window group.

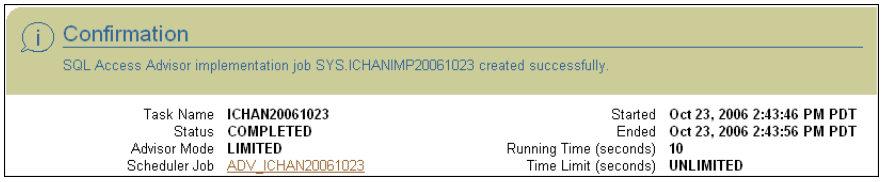
To search for a window, click the Window search icon. The Search and Select: Window and Window Groups dialog box appears. Select the window or window group you want to use and click **Select**. The selected window or window group now appears in the Window field.

To stop the job if the available window or window group closes, check the **Stop on Window Close** box.

In this example, **Standard** is selected for schedule type. The job will not repeat and is scheduled to start immediately.



6. To view the SQL text for the job, click **Show SQL**.
7. To submit the job, click **Submit**.
8. If the job is scheduled to start immediately, the Results for Tasks page for the SQL Access Advisor task appears with a confirmation that the job was successfully created.



In the Scheduler Job field, click the link to display the View Job page for the implementation job. Skip the next step and proceed to step 10.

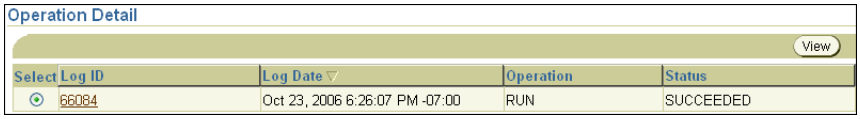
9. If the job is scheduled to start at a later time, you will need to access the View Job page after the job is completed.
   a. On the Database Administration page, under Database Scheduler, click **Jobs**.

      The Scheduler Jobs page appears.

   b. Select the implementation job and click **View Job Definition**.

      The View Job page for the selected job appears.

10. On the View Job page, under Operation Detail, verify that the status of the operation was successful.

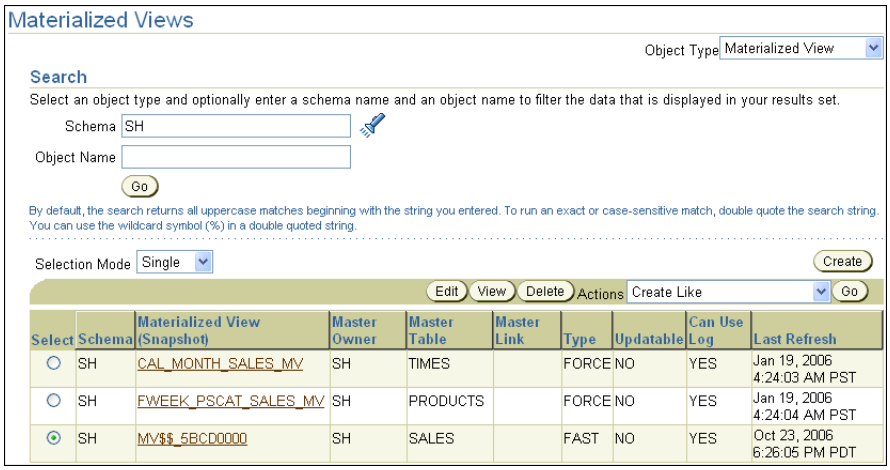


If the operation is not successful, you can view details about the operation by selecting the operation and clicking **View**. This displays the Operation Detail page, which contains information (such as start date and time, run duration, CPU time used, and session ID) that you can use to troubleshoot the failure.

11. Verify that the access structure recommended by the SQL Access Advisor is created.

Depending on the type of access structure that is created, you can display the access structure using the Indexes page, Materialized Views page, or the Materialized View Logs page.

In this example, a materialized view named `MV$$_5BCD0000` is created in the `SH` schema.



Index

A

B

C

D

E

H

I

L

M

O

P

S

T

U

W