Oracle® Database

2 Day + Performance Tuning Guide

11*g* Release 2 (11.2)

E10822-04

November 2011

ORACLE®

Oracle Database 2 Day + Performance Tuning Guide, 11*g* Release 2 (11.2)

E10822-04

Primary Authors: David McDermid, Lance Ashdown, Immanuel Chan

Contributing Author: Sushil Kumar

Contributors: Pete Belknap, Supiti Buranawatanachoke, Nancy Chen, Kakali Das, Karl Dias, Mike Feng, Yong Feng, Cecilia Grant, Connie Green, William Hodak, Andrew Holdsworth, Kevin Jernigan, Caroline Johnston, Sue K. Lee, Herve Lejeune, Colin McGregor, Mughees Minhas, Valarie Moore, Deborah Owens, Mark Ramacher, Uri Shaft, Susan Shepard, Janet Stern, Hsiao-Te Su, Minde Sun, Mark Townsend, Stephen Wexler, Graham Wood, Khaled Yagoub, Michael Zampiceni

Contents

10 Tuning SQL Statements

11 Optimizing Data Access Paths

Index

Preface

This preface contains the following topics:

Audience

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

In particular, this guide 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

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc`.

Access to Oracle Support

Oracle customers have access to electronic support through My Oracle Support. For information, visit `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info` or visit `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs` if you are hearing impaired.

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 Websites in Documentation

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

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 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 guide and explains the Oracle Database performance method. This part contains the following chapters:

- Chapter 1, "Introduction"
- Chapter 2, "Oracle Database Performance Method"

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 Oracle Diagnostics Pack, Oracle Tuning Pack, and Oracle Enterprise Manager (Enterprise Manager).

This chapter contains the following sections:

About This Guide

Before using this guide, you must do the following:

- Read *Oracle Database 2 Day DBA* in its entirety.
- 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 that type of information, see *Oracle Database Concepts*.

This guide does not describe basic Oracle Database administrative tasks. For that 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 the Enterprise Manager Database Control console. This guide is not an exhaustive discussion of all Oracle Database performance tuning features. It does not cover available application programming interfaces (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 user accounts 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

Oracle Database 2 Day + Performance Tuning Guide describes how to accomplish the last two tasks in the preceding 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 must acquire the following products, tools, features, and utilities:

- **Oracle Database 11*g* Enterprise Edition**

  Oracle Database 11*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 the database is Enterprise Manager, a Web-based interface. After you install the Oracle software, create or upgrade a database, and configure the network, you can use Enterprise Manager to manage the database. In addition, Enterprise Manager provides an interface for performance advisors and for database utilities, such as SQL*Loader and Recovery Manager (RMAN).

- **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 Oracle Diagnostics Pack used in this guide include Automatic Workload Repository (AWR), Automatic Database Diagnostic Monitor (ADDM), and Active Session History (ASH).

- **Oracle Database Tuning Pack**

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

  – SQL Tuning Advisor

    This feature enables you to submit one or more SQL statements as input and receive output in the form of specific advice or recommendations for how to

tune statements, along with a rationale for each recommendation and its expected benefit. A recommendation relates to collection of statistics on objects, creation of new indexes, restructuring of the SQL statements, or creation of SQL profiles.

– SQL Access Advisor

  This feature enables you to optimize data access paths of SQL queries by recommending the proper set of materialized views and view logs, indexes, and partitions for a given SQL workload.

- **Oracle Real Application Testing**

  Oracle Real Application Testing consists of the following key features:

  – Database Replay

    This feature enables you to capture the database workload on a production system, and replay it on a test system with the exact same timing and concurrency as the production system on the same or newer release of Oracle Database.

  – SQL Performance Analyzer

    This feature enables you to assess the effect of system changes on SQL performance by identifying SQL statements that have regressed, improved, or remained unchanged.

  See *Oracle Database Real Application Testing User's Guide* to learn how to use these features.

Note: Some of the products and tools in the preceding list, including 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** (a point where resource contention is highest) may 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 toward ensuring that your changes improve performance.

Typically, performance problems result from a lack of **throughput** (the amount of work that can be completed in a specified time), unacceptable user or job **response time** (the time to complete a specified workload), or both. The problem might be localized to specific application modules or span the system.

Before looking at database or operating system statistics, it is crucial to get feedback from the system users and the people paying for the application. This feedback makes it easier to set performance goals. Improved performance can be measured in terms of business goals rather than system statistics.

The Oracle performance method can be applied until performance goals are met or deemed impractical. Because this process is iterative, some investigations may have little impact on system performance. It takes time and experience to accurately pinpoint critical bottlenecks quickly. Automatic Database Diagnostic Monitor (ADDM) implements the Oracle performance method and analyzes statistics to provide automatic diagnosis of major performance problems. Because ADDM can significantly shorten the time required to improve the performance of a system, 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.

A **cumulative statistic** is a count such as the number of block reads. 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. 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, the database gathers statistics every hour and creates an **AWR snapshot**, which is a set of data for a specific time that is used for performance comparisons. The delta values captured by the snapshot represent the changes for each statistic over the time period. Statistics gathered by AWR are queried from memory. The gathered data can be displayed in both reports and views.

The following initialization parameters are relevant for AWR:

- `STATISTICS_LEVEL`

  Set this parameter to `TYPICAL` (default) or `ALL` to enable statistics gathering by AWR. Setting `STATISTICS_LEVEL` to `BASIC` disables many database features, including AWR, and is not recommended. To learn more about this initialization parameter, see *Oracle Database Reference*.

- `CONTROL_MANAGEMENT_PACK_ACCESS`

  Set to `DIAGNOSTIC+TUNING` (default) or `DIAGNOSTIC` to enable automatic database diagnostic monitoring. Setting `CONTROL_MANAGEMENT_PACK_ACCESS` to `NONE` disables many database features, including ADDM, and is strongly discouraged. To learn more about this initialization parameter, see *Oracle Database Reference*.

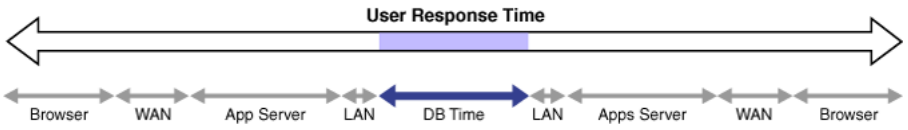
The database statistics collected and processed by AWR include:

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

Time Model Statistics

Time model statistics measure the time spent in the database by operation type. The most important time model statistic is **database time (DB time)**. Database time represents the total time spent in database calls by foreground sessions, 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.

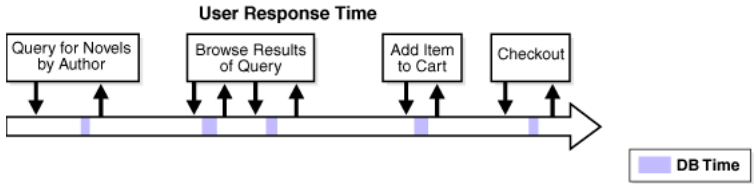
Figure 2–1 DB Time in Overall User Response Time



A **session** is a logical entity in the database instance memory that represents the state of a current user login to a database. Database time is calculated by aggregating the CPU time and wait time of all **active sessions** (sessions that are not idle). For any database request, the CPU time is the sum of the time spent working on the request, while the wait time is the sum of all the waits for various database instance resources. DB time includes only time spent on client processes and does not include time spent on background processes such as PMON.

For example, a user session may involve an online transaction made at an online bookseller consisting of the actions 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 action causes the application to perform a database query for novels by the author.

2. Browse results of query

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

3. Add item to cart

   After browsing details about the novels, the user decides to add one novel to the shopping cart. This action 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 website from a previous purchase. This action 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 the preceding actions, the user makes a request to the database, as represented by the down arrow in Figure 2–2 on page 2-3. 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. After 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.

> **Note:** DB time is measured cumulatively from when the instance started. Because DB time combines times from all non-idle user sessions, DB time can exceed the time elapsed since the instance started. For example, an instance that has run 5 minutes could have four active sessions whose cumulative DB time is 20 minutes.

The objective of database tuning is to reduce database time. In this way, you can improve the overall response time of user transactions in the application.

Wait Event Statistics

Wait events are incremented by a session to indicate that the session 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 database records the wait by using one of a set of predefined wait events. The events are then grouped into wait classes, such as User I/O and Network. Wait event data reveals symptoms of problems that might be affecting performance, such as latch, buffer, or I/O contention.

See Also:

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

Session and System Statistics

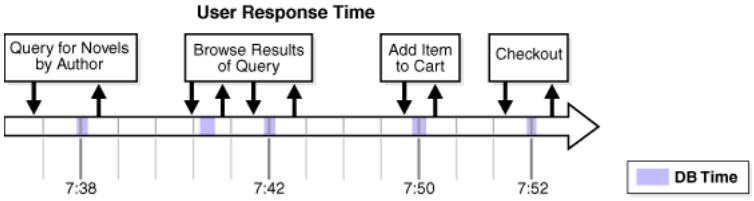
A large number of cumulative database statistics are available on a system and session level. Some of these statistics are collected by AWR.

Active Session History Statistics

The Active Session History (ASH) statistics are samples of session activity in the database. The database samples active sessions every second and stores them 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. The size of the data is directly related to the work being performed, rather than the number of sessions allowed on the database.

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 website, 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 novel is added to the shopping cart
- 7:52, during the checkout process

Table 2–1 lists ASH statistics collected for the active sessions, along with examples of the session ID (SID), 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	n/a
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 must 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 must respond to issues reported by users, such as performance problems that may occur for only a short duration of time, or performance degradation to the database over a period of 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 the following:

- 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 2-6
- Tuning the database reactively when performance problems are reported by the users, as described in "Tuning the Database Reactively" on page 2-7
- Identifying, tuning, and optimizing high-load SQL statements, as described in "Tuning SQL Statements" on page 2-7

To improve database performance, you must apply these principles iteratively.

Preparing the Database for Tuning

This section lists and describes the steps that must 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. 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 for possible later analysis. In addition, ensure that all hardware is functioning properly.

3. Ensure that the `STATISTICS_LEVEL` initialization parameter is set to `TYPICAL` (default) or `ALL` to enable the automatic performance tuning features of Oracle Database, including AWR and ADDM.
4. Ensure that the `CONTROL_MANAGEMENT_PACK_ACCESS` initialization parameter is set to `DIAGNOSTIC+TUNING` (default) or `DIAGNOSTIC` to enable ADDM.

See Also:

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

Tuning the Database Proactively

This section lists and describes the proactive steps required to keep the database properly tuned on a regular basis. Perform these steps as part of your daily maintenance of Oracle Database. Repeat the tuning process until your performance goals are met or become impossible to achieve because of other constraints.

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 2-8. The results are displayed as ADDM findings on the Database Home page in Oracle Enterprise Manager (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".

   With each ADDM finding, ADDM automatically provides a list of recommendations for reducing the impact of the performance problem. 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 Performance page in 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 Enterprise Manager displays performance-related alerts generated by the database. Typically, resolving the problems indicated by these alerts improves database performance.

5. Validate that any changes made have produced the desired effect, and verify that the users experience performance improvements.

Tuning the Database Reactively

This section lists and describes the steps required to tune the database based on user feedback. This tuning procedure is considered reactive. Perform this procedure 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".

   In this way you can analyze current database performance before the next ADDM analysis, or 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 report enables 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 that the users experience performance improvements.

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 section 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 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 SQL Access Advisor.

4. Analyze the SQL performance impact of SQL tuning and other system changes by using SQL Performance Analyzer.

To learn how to use SQL Performance Analyzer, see *Oracle Database Real Application Testing User's Guide*.

5. 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, you should be able to avoid these problems. If you experience these problems, then repeat the steps in the Oracle performance method, as described in "Using the Oracle Performance Method" on page 2-5, or consult the appropriate section that addresses these problems:

- CPU bottlenecks

  Is the application performing poorly because the system is CPU-bound? Performance problems caused by CPU bottlenecks are diagnosed by ADDM, as described in Chapter 3, "Automatic Database Performance Monitoring". You can also identify CPU bottlenecks by using the Performance page in Enterprise Manager, as described in "Monitoring CPU Utilization" on page 4-20.

- Undersized memory structures

  Are the Oracle memory structures such as the System Global Area (SGA), Program Global Area (PGA), and buffer cache adequately sized? Performance problems caused by undersized memory structures are diagnosed by ADDM, as described in Chapter 3, "Automatic Database Performance Monitoring". You can also identify memory usage issues by using the Performance page in Enterprise Manager, as described in "Monitoring Memory Utilization" on page 4-22.

- I/O capacity issues

  Is the I/O subsystem performing as expected? Performance problems caused by I/O capacity issues are diagnosed by ADDM, as described in Chapter 3, "Automatic Database Performance Monitoring". You can also identify disk I/O issues by using the Performance page in Oracle Enterprise Manager, as described in "Monitoring Disk I/O Utilization" on page 4-24.

- Suboptimal use of Oracle Database by the application

  Is the application making suboptimal use of Oracle Database? Problems such as establishing new database connections repeatedly, excessive SQL parsing, and high levels 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, 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 Performance page in 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, as described in Chapter 3, "Automatic Database Performance

Monitoring". You can also identify concurrency issues by using Top Sessions in Enterprise Manager, as described in "Monitoring Top Sessions" on page 4-6.

- 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, too many checkpoints, or suboptimal parameter settings? Performance problems caused by database configuration issues are diagnosed by ADDM, as described in Chapter 3, "Automatic Database Performance Monitoring".

- Short-lived performance problems

  Are users complaining about short-lived or intermittent performance problems? Depending on the interval between snapshots taken by 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 was 6 months ago? You can generate an AWR 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 technique helps 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 any SQL statements using excessive system resources that impact the system? Performance problems caused by high-load SQL statements are diagnosed by ADDM, as described in Chapter 3, "Automatic Database Performance Monitoring" and "Identification of High-Load SQL Statements Using ADDM Findings" on page 9-1. You can also identify high-load SQL statements by using Top SQL in 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 SQL Tuning Advisor, as described in Chapter 10, "Tuning SQL Statements".

- Object contention

  Are any database objects the source of bottlenecks because they are continuously accessed? Performance problems caused by object contention are diagnosed by ADDM, as described in Chapter 3, "Automatic Database Performance Monitoring". You can also optimize the data access path to these objects using SQL Access Advisor, as described in Chapter 11, "Optimizing Data Access Paths" on page 4-24.

- Unexpected performance regression after tuning SQL statements

  Is the performance of SQL statements degrading after they have been tuned? Tuning SQL statements may cause changes to their execution plans, resulting in a significant impact on SQL performance. In some cases, the changes may result in the improvement of SQL performance. In other cases, the changes may cause SQL statements to regress, resulting in a degradation of SQL performance.

  Before making changes on a production system, you can analyze the impact of SQL tuning on a test system by using SQL Performance Analyzer. This feature enables you to forecast the impact of system changes on a SQL workload by:

- Measuring the performance before and after the change
- Generating a report that describes the change in performance
- Identifying the SQL statements that regressed or improved
- Providing tuning recommendations for each SQL statement that regressed
- Enabling you to implement the tuning recommendations when appropriate

To learn how to use SQL Performance Analyzer, see *Oracle Database Real Application Testing User's Guide*.

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

Automatic Database Diagnostic Monitor (ADDM) automatically detects and reports performance problems with the database. The results are displayed as ADDM findings on the Database Home page in Oracle Enterprise Manager (Enterprise Manager). Reviewing the ADDM findings enables you to quickly identify the performance problems that require your attention.

Each ADDM finding provides a list of recommendations for reducing the impact of the performance problem. You should review ADDM findings and implement the recommendations every day as part of regular database maintenance. Even when the database is operating at an optimal performance level, you should continue to use ADDM to monitor database performance on an ongoing basis.

This chapter contains the following sections:

See Also:

- *Oracle Database Performance Tuning Guide* for information about using the `DBMS_ADVISOR` package to diagnose and tune the database with the Automatic Database Diagnostic Monitor

Overview of Automatic Database Diagnostic Monitor

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

This section contains the following topics:

- ADDM for Oracle Real Application Clusters

ADDM Analysis

An ADDM analysis is performed after each AWR snapshot (every hour by default), and the results are saved in the database. You can then view the results using Enterprise Manager. Before using another performance tuning method described in this guide, review the results of the ADDM analysis first.

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 the wait time and CPU time of all user sessions that are not idle.

The goal of database performance tuning is to reduce the DB time of the system for a given workload. By reducing DB time, the database can support more user requests by using the same or fewer resources. ADDM reports system resources that are using a significant portion of DB time as problem areas and sorts them 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.

ADDM Recommendations

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

- 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 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.

Performance tuning is an iterative process. 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.

See Also:

- *Oracle Database 2 Day DBA* for information the Segment Advisor

ADDM for Oracle Real Application Clusters

In an Oracle Real Application Clusters (Oracle RAC) environment, you can use ADDM to analyze the throughput performance of a database cluster. ADDM for Oracle RAC considers DB time as the sum of database times for all database instances and reports findings that are significant at the cluster level. For example, the DB time of each cluster node may be insignificant when considered individually, but the aggregate DB time may be a significant problem for the cluster as a whole.

See Also:

- *Oracle Database 2 Day + Real Application Clusters Guide* for information about using ADDM for Oracle RAC

Configuring Automatic Database Diagnostic Monitor

This section contains the following topics:

Setting Initialization Parameters to Enable ADDM

Automatic database diagnostic monitoring is enabled by default and is controlled by the `CONTROL_MANAGEMENT_PACK_ACCESS` and the `STATISTICS_LEVEL` initialization parameters.

Set `CONTROL_MANAGEMENT_PACK_ACCESS` to `DIAGNOSTIC+TUNING` (default) or `DIAGNOSTIC` to enable automatic database diagnostic monitoring. Setting `CONTROL_MANAGEMENT_PACK_ACCESS` to `NONE` disables many Oracle Database features, including ADDM, and is strongly discouraged.

Set `STATISTICS_LEVEL` to `TYPICAL` (default) or `ALL` to enable automatic database diagnostic monitoring. Setting `STATISTICS_LEVEL` to `BASIC` disables many Oracle Database features, including ADDM, and is strongly discouraged.

To determine whether ADDM is enabled:

1. From the Database Home page, click **Server**.

   The Server subpage appears.

2. In the Database Configuration section, click **Initialization Parameters**.

   The Initialization Parameters page appears.

3. In the **Name** field, enter `statistics_level` and then click **Go**.

   The table shows the setting of this initialization parameter.

Name	Help	Revisions	Value	Comments	Type	Basic	Modified	Dynamic	Category
statistics_level			TYPICAL		String			✓	Diagnostics and Statistics

4. Do one of the following:
   - If the Value list shows **ALL** or **TYPICAL**, then do nothing.
   - If the Value list shows **BASIC**, then select **ALL** or **TYPICAL**, and then click **Apply**.

5. In the **Name** field, enter `control_management_pack_access`, and then click **Go**.

   The table shows the setting of this initialization parameter.

6. Do one of the following:

   - If the Value column shows **DIAGNOSTIC** or **DIAGNOSTIC+TUNING**, then do nothing.
   - If the Value column shows **NONE**, then select **DIAGNOSTIC** or **DIAGNOSTIC+TUNING** and click **Apply**.

     See Also:

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

Setting the DBIO_EXPECTED Parameter

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 `DBIO_EXPECTED` 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. You can choose a different value based on the characteristics of your hardware.

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

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

   This measurement must 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. See *Oracle Database Performance Tuning Guide* to learn how to assess the I/O capability of the storage subsystem.

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

   For example, if the measured value is 8000 microseconds, then execute the following PL/SQL code as the `SYS` user:

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

Managing AWR 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 8 days. You can change the default values for both the snapshot interval and the retention period.

Oracle recommends that you adjust the AWR retention period to at least one month. You can also extend the period to one business cycle so you can compare data across time frames such as the close of the fiscal quarter. You can also create AWR baselines to retain snapshots indefinitely for important time periods.

The data in the snapshot interval is analyzed by ADDM. ADDM compares the differences between snapshots to determine which SQL statements to capture, based on the effect on the system load. The ADDM analysis shows 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

Manually creating snapshots is usually not necessary because 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 than the snapshot interval.

To create snapshots:

1. From the Database Home page, click **Performance**.

   The Performance page appears.

2. Under Additional Monitoring Links, click **Snapshots**.

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

3. Click **Create**.

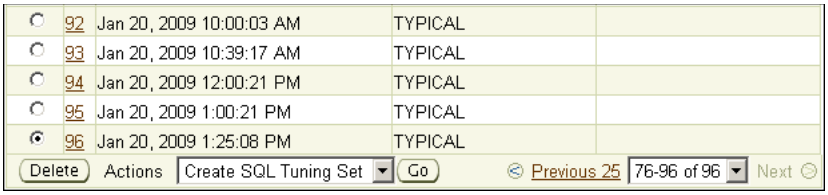
   The Confirmation page appears.

4. Click **Yes**.

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

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

   In the following example, the ID of the snapshot that was created is 96.



Modifying Snapshot Settings

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

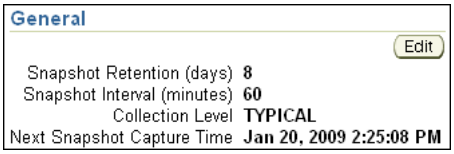
To modify the snapshot settings:

1. From the Database Home page, click **Server**.

   The Server subpage appears.

2. In the Statistics Management section, click **Automatic Workload Repository**.

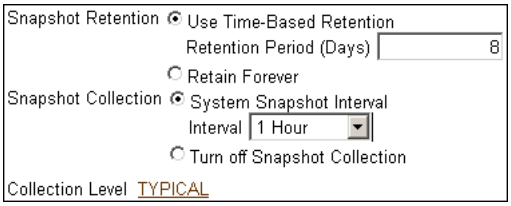
   The Automatic Workload Repository page appears.



In this example, snapshot retention is set to 8 days and snapshot interval is set to 60 minutes.

3. Click **Edit**.

   The Edit Settings page appears.

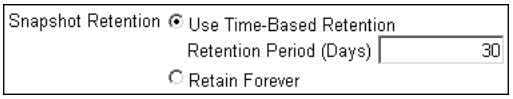


4. For **Snapshot Retention**, do one of the following:

   - Select **Use Time-Based Retention Period (Days)**, and in the associated field enter the number of days to retain the snapshots.
   - Select **Retain Forever** to retain snapshots indefinitely.

   It is recommended that you increase the snapshot retention period to the maximum allowed by the available disk space.

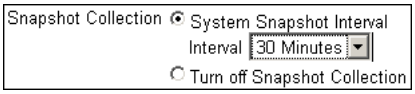
   In this example, the snapshot retention period is changed to 30 days.



5. For **Snapshot Collection**, do one of the following:

   - Select **System Snapshot Interval**, and in the **Interval** list, select the desired interval to change the interval between snapshots.
   - Select **Turn off Snapshot Collection** to disable snapshot collection.

   In this example, the snapshot collection interval is changed to 30 minutes.

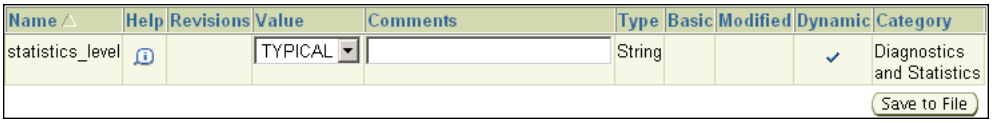


6. Click the link next to **Collection Level**.

   The Initialization Parameter page appears.

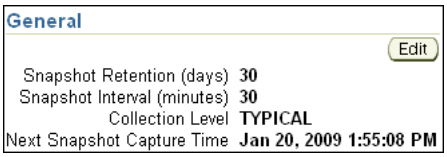
   To change the statistics level, select the desired value in the Value list for the `statistics_level` parameter. Click **Save to File** to set the value in the server parameter file.

   In this example, the default value of Typical is used.



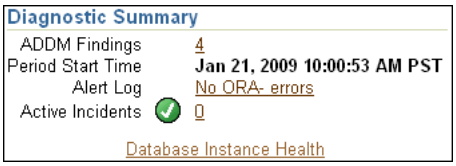
7. Click **OK** to apply the changes.

   The Automatic Workload Repository page appears and displays the new settings.



Reviewing the Automatic Database Diagnostic Monitor Analysis

By default, ADDM runs every hour to analyze snapshots taken by AWR during that period. If the database finds performance problems, then it displays the results of the analysis under Diagnostic Summary on the Database Home page.

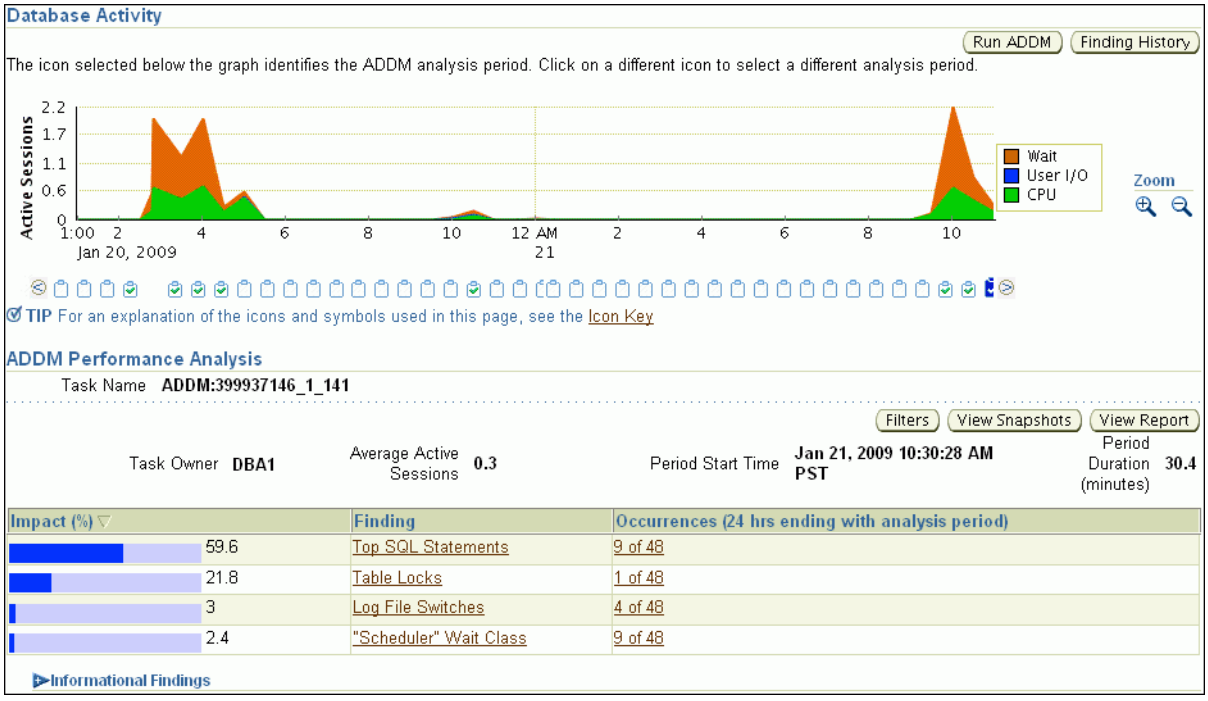


The ADDM Findings link shows how many ADDM findings were found in the most recent ADDM analysis.

To view ADDM findings:

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

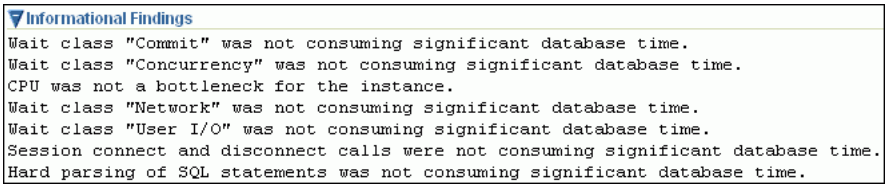
   The Automatic Database Diagnostic Monitor (ADDM) page appears. The results of the ADDM run are displayed.



On the Automatic Database Diagnostic Monitor (ADDM) page, the Database Activity chart shows the database activity during the ADDM analysis period. Database activity types are defined in the legend based on their corresponding colors in the chart. Each icon below the chart represents a different ADDM task, which in turn corresponds to a pair of snapshots saved in AWR.

In this example, the two largest blocks of activity were 2:30 p.m. to 5:30 p.m. on January 20 and 9 a.m. to 11 a.m. the next day. The thick CPU and thin Wait bars in the hour after 4:30 p.m. indicate that CPU may have been a bottleneck during this period. In other areas of the chart, the Wait bar is thicker than the CPU bar, indicating that wait events had a greater performance impact than CPU.

In the ADDM Performance Analysis section, ADDM findings are listed in descending order, from highest to least impact. The Informational Findings section lists areas that have no performance impact and are for information only.



2. Optionally, click the Zoom icons to shorten or lengthen the analysis period displayed on the chart.
3. To view the ADDM findings in a report, click **View Report**.

   The View Report page appears.

   You can click **Save to File** to save the report for later access.

Interpretation of Automatic Database Diagnostic 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 leads 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.

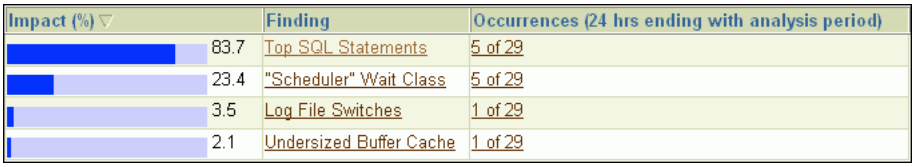
When a specific problem has multiple causes, ADDM may report multiple findings. In this case, the impacts of these multiple findings can contain the same portion of DB time. Because performance problems can overlap, summing the impacts of the reported findings can yield a number higher than 100% of DB time. For example, if a system performs many read I/O operations, ADDM may report a SQL statement responsible for 50% of DB time due to I/O activity as one finding, and an undersized buffer cache responsible for 75% 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 must apply all the actions of a recommendation to gain its estimated benefit. The rationales explain why the set of actions was recommended, and provide additional information for implementing them. An ADDM action may present multiple solutions. If this is the case, then choose the easiest solution to implement.

Implementing Automatic Database Diagnostic Monitor Recommendations

This section describes how to implement ADDM recommendations. ADDM findings are displayed in the Automatic Database Diagnostic Monitor (ADDM) page under ADDM Performance Analysis.



Impact (%)	Finding	Occurrences (24 hrs ending with analysis period)
83.7	Top SQL Statements	5 of 29
23.4	"Scheduler" Wait Class	5 of 29
3.5	Log File Switches	1 of 29
2.1	Undersized Buffer Cache	1 of 29

To implement ADDM recommendations:

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

   The Automatic Database Diagnostic Monitor (ADDM) page appears.

2. In the Database Activity section, click the icon for the ADDM to investigate.

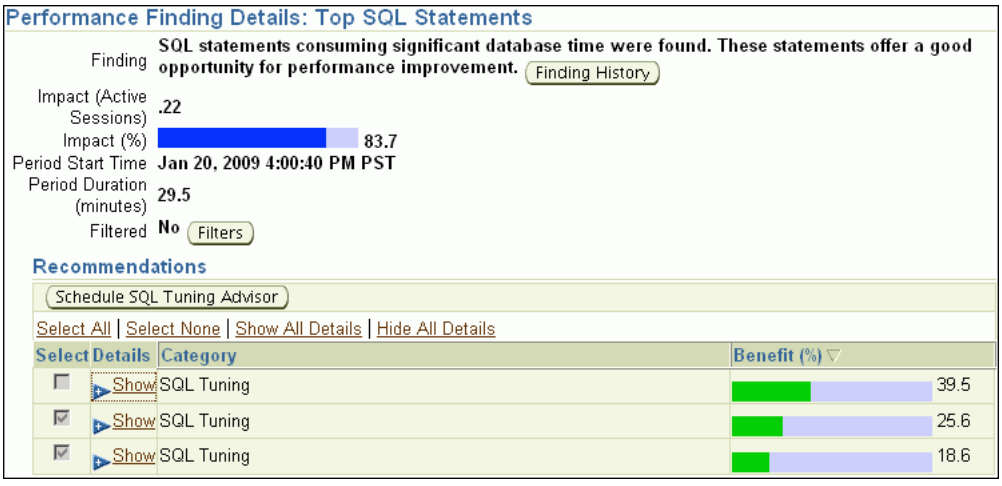
   The data in the ADDM Performance Analysis section changes based on the ADDM run that you selected.

3. In the ADDM Performance Analysis section, click the ADDM finding that has the greatest impact.

   In this example, the finding with the greatest impact is Top SQL Statements.

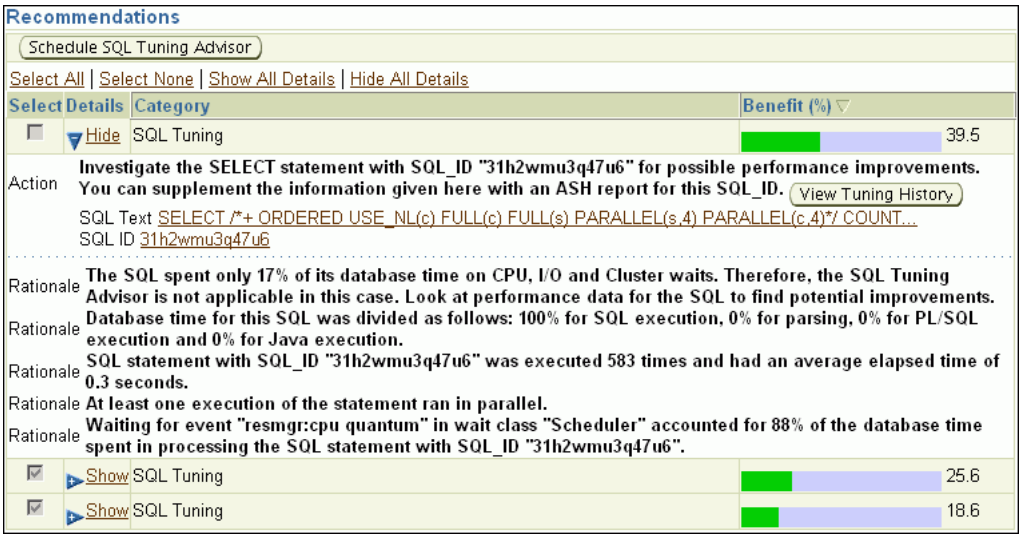
   The Performance Finding Details page appears.

   In the following example, three recommendations are shown. The first is estimated to have a maximum benefit of up to 39.5% of DB time in the analysis period. The second recommendation is estimated to have a maximum benefit of up to 25.6% of DB time, while the third has a maximum of 18.6%.



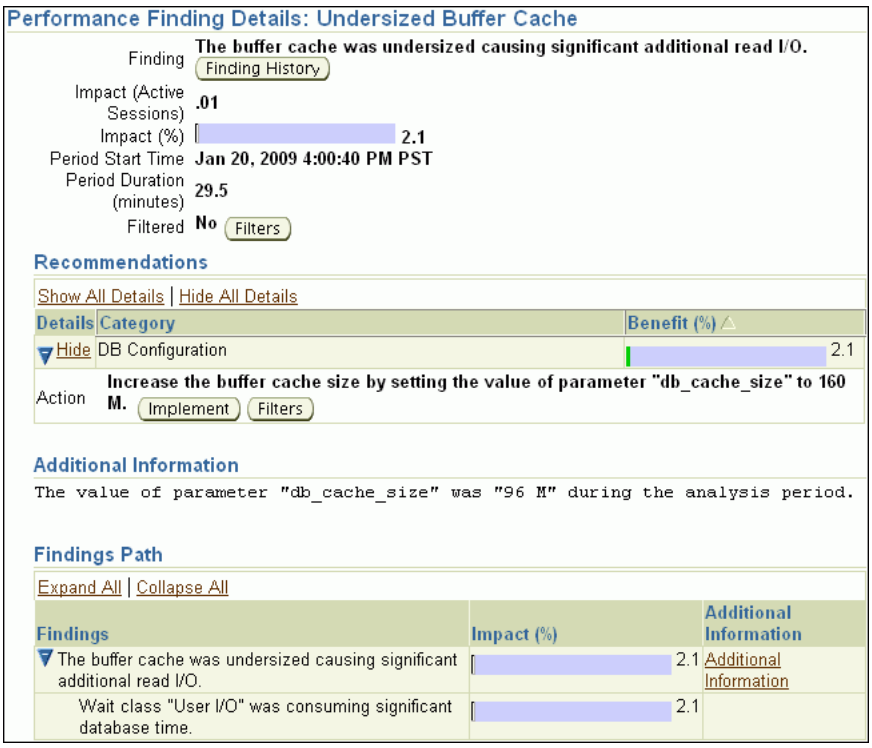
4. Under Recommendations, click **Show** to review the recommendations and required actions for each recommendation.

   The Category column displays the category of the recommendation. The Benefit (%) column displays the estimated benefit of implementing the recommendation.



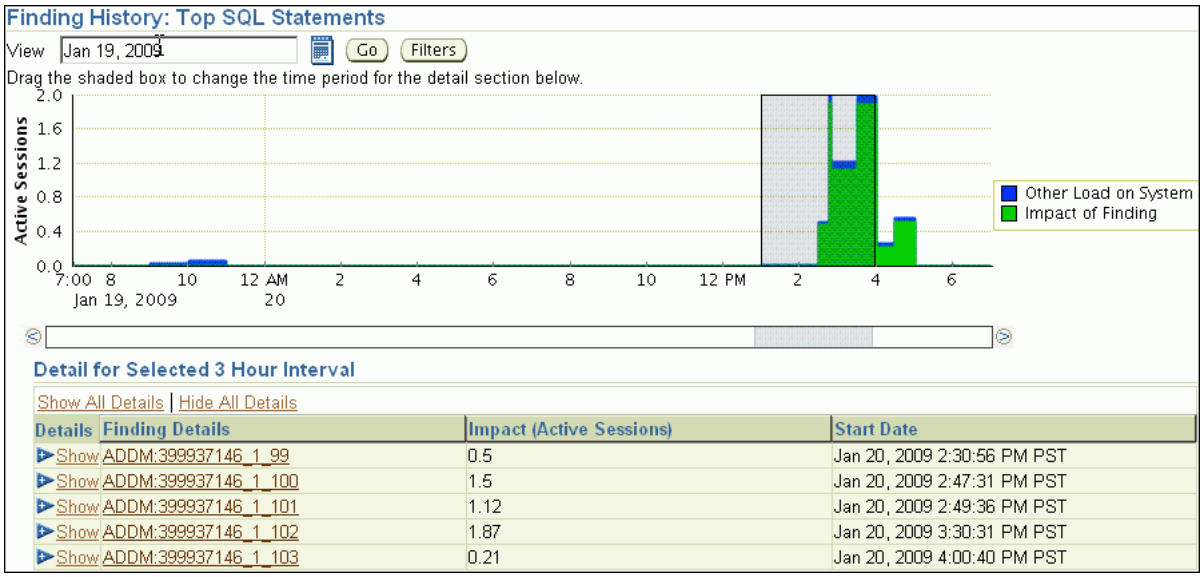
5. If additional information is available about why the set of actions was recommended, then click **Additional Information**, or review the content displayed under Additional Information.

   For example, the Undersized Buffer Cache finding contains additional information to indicate the recommended value of the `DB_CACHE_SIZE` initialization parameter.



6. To view the history of a finding, click **Finding History**.

The Finding History page appears.



The Finding History page shows how often a particular finding has occurred in a selected 3-hour interval. You can use this information to determine whether the finding was a transient or a persistent problem in the system. Based on this information, you can determine whether the actions associated with the finding should be implemented.

The Active Sessions chart shows the impact of the finding and of the other loads on the system. You can change the display as follows:

a. To move the 3-hour interval, click and drag the shaded box in the Active Sessions chart.

b. To change dates, enter the desired date in the **View** field, and then click **Go**.

c. To view details about a finding, under Detail for Selected 3 Hour Interval, click the link in the **Finding Details** column to display the Performance Finding Details page for the corresponding ADDM finding.

7. Optionally, create a filter to suppress known findings that have been tuned or cannot be tuned further. To create filters for a selected ADDM finding:

a. Click **Filters**.

The Filters for Finding page appears.

b. Click **Create**.

The Create Filter for Finding page appears.

c. In the **Name** field, enter a name for the ADDM filter.

d. In the **Active Sessions** field, specify the filter criteria in terms of the number of active sessions.

The database filters the ADDM finding for future ADDM runs if the number of active sessions for the finding is less than the specified filter criteria.

e. In the **% Active Sessions** field, specify the filter criteria in terms of percentage of active sessions.

The database filters the ADDM finding for future ADDM runs if the number of active sessions for the finding is less than the specified filter criteria.

f. Click **OK**.

8. Perform the required action of a chosen recommendation.

   Depending on the type of action you choose to perform, various options may be available, such as **Implement** or **Run Advisor Now**. These options enable you to implement the recommendation immediately with a single mouse click.

   In the example shown in Step 4, the simplest solution is to click **Run Advisor Now** to immediately run a SQL Tuning Advisor task on the SQL statement.

   See Also:

   - Chapter 10, "Tuning SQL Statements"

Viewing Snapshot Statistics

You can view the data contained in snapshots taken by AWR using Enterprise Manager. Typically, it is not necessary to review snapshot data because it primarily contains raw statistics. Instead, rely on ADDM, which analyzes statistics to identify performance problems. Snapshot statistics are intended primarily for advanced users, DBAs accustomed to using Statspack for performance analysis.

To view snapshot statistics:

1. From the Database Home page, click **Performance**.

   The Performance page appears.

2. Under Additional Monitoring Links, click **Snapshots**.

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

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

   The Snapshot Details appears, showing the Details subpage.

Details | Report

Beginning Snapshot ID: 100
Beginning Snapshot Capture Time: Jan 20, 2009 2:49:35 PM
Ending Snapshot ID: 101
Ending Snapshot Capture Time: Jan 20, 2009 3:30:30 PM

Previous 1-27 of 27 Next

Name	Value	Per Second	Per Transaction
DB cpu (seconds)	0.00	0.00	0.00
DB time (seconds)	12,477.31	5.08	20.42
db block changes	23,112.00	9.41	37.83
execute count	127,769.00	52.04	209.11
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	139,612.00	56.87	228.50
parse count (total)	105,593.00	43.01	172.82
parse time cpu (seconds)	1.54	0.00	0.00
parse time elapsed (seconds)	2.18	0.00	0.00
physical reads	740.00	0.30	1.21
physical writes	1,759.00	0.72	2.88
redo size (KB)	4,216.85	1.72	6.90
session cursor cache hits	82,729.00	33.70	135.40
session logical reads	397,020,644.00	161,719.20	649,788.29
sql execute cpu time (seconds)	0.00	0.00	0.00
sql execute elapsed time (seconds)	0.00	0.00	0.00
user calls	338,015.00	137.68	553.22
user commits	449.00	0.18	0.73
user rollbacks	162.00	0.07	0.27
workarea executions - multipass	0.00	0.00	0.00
workarea executions - onepass	2.00	0.00	0.00
workarea executions - optimal	3,101.00	1.26	5.08

In this example, statistics gathered from the previous snapshot (snapshot 100) to the selected snapshot (snapshot 101) are displayed.

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

   The Workload Repository report appears.

5. Optionally, click **Save to File** to save the report for later access.

See Also:

- Chapter 8, "Resolving Performance Degradation Over Time"

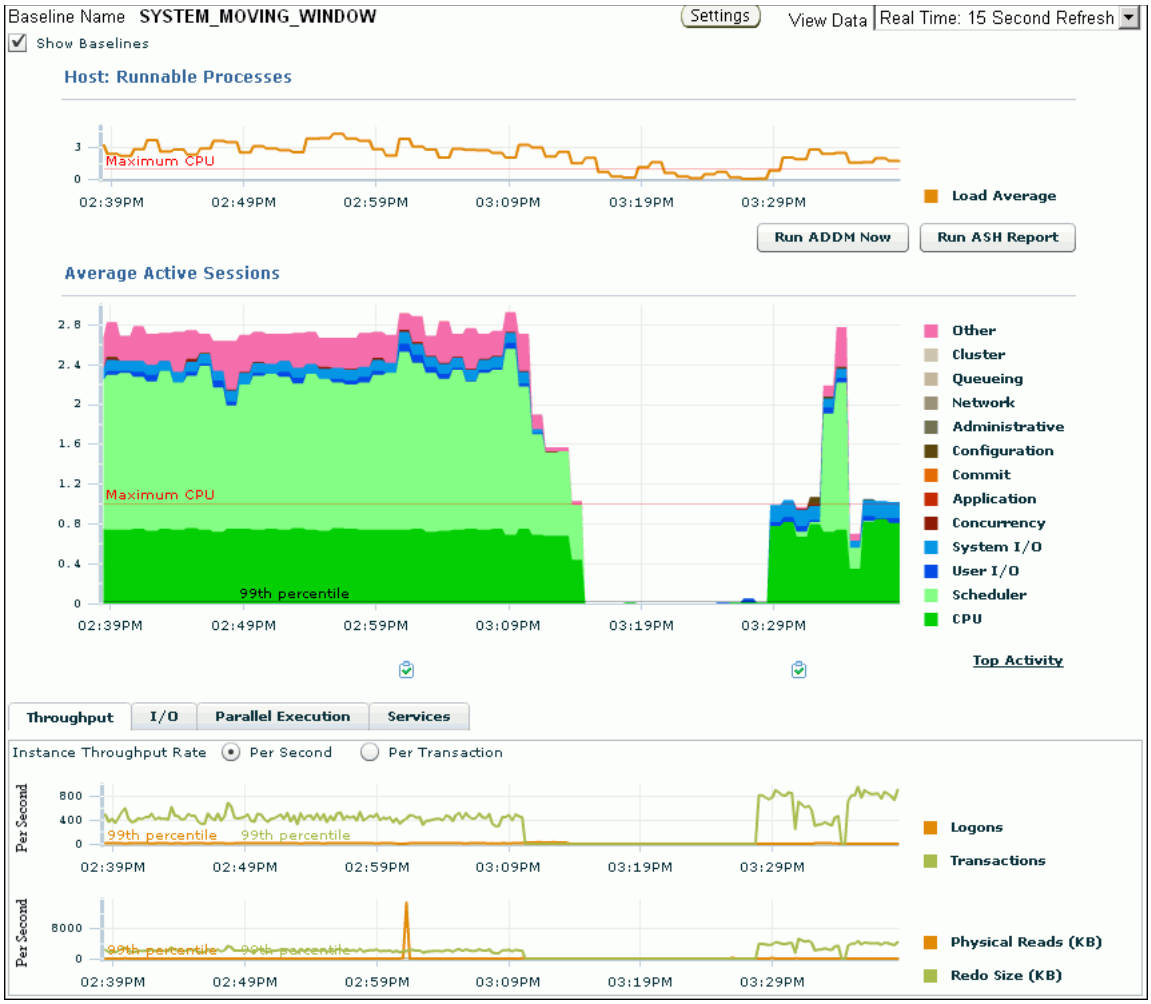
4

Monitoring Real-Time Database Performance

The Performance page in Oracle Enterprise Manager (Enterprise Manager) displays information in three sections that you can use to assess the overall performance of the database in real time.

Figure 4–1 shows the Performance page.

Figure 4–1 Performance Page



Typically, you should use the automatic diagnostic feature of Automatic Database Diagnostic Monitor (ADDM) to identify performance problems with the database, as described in Chapter 3, "Automatic Database Performance Monitoring". In some cases, you may want to monitor the database performance in real time to identify performance problems as they occur. For example, ADDM performs its analysis after each Automatic Workload Repository (AWR) snapshot, which by default is once every hour. However, if you notice a sudden spike in database activity on the Performance page, then you may want to investigate the incident before the next ADDM analysis.

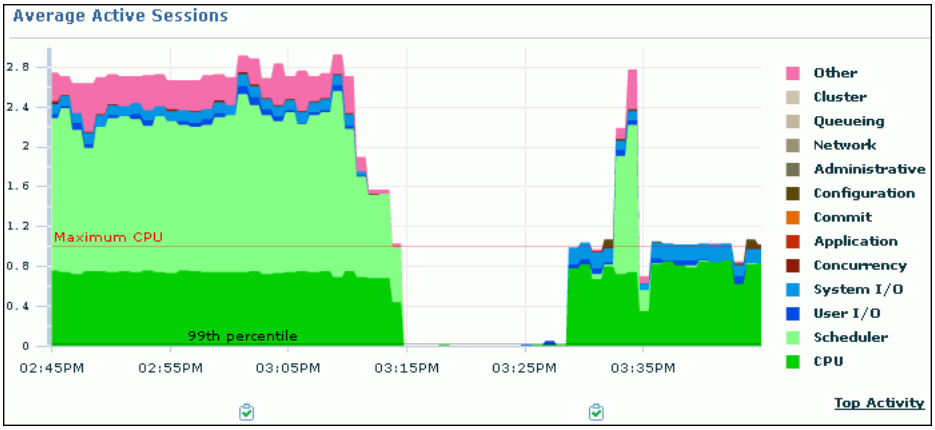
By drilling down to other pages from the Performance page, you can identify database performance problems in real time. If you find a problem, then you can run ADDM manually to analyze it immediately without having to wait until the next ADDM analysis. To learn how to run ADDM manually, see "Manually Running ADDM to Analyze Current Database Performance" on page 6-1.

This chapter contains the following sections:

Monitoring User Activity

The Average Active Sessions chart of the Performance page shows the average load on the database. The **average active sessions** for a time period equals the total DB time of all sessions during this period divided by the elapsed time (wall clock time) for this period. The chart shows which active sessions are running on the CPU or waiting on an event.

Figure 4–2 Monitoring User Activity



By following the performance method explained in Chapter 2, "Oracle Database Performance Method", you can drill down from the charts to identify the causes of instance-related performance issues and resolve them.

To monitor user activity:

1. From the Database Home page, click **Performance**.

   The Performance page appears.

2. Locate any sudden increases in the Average Active Sessions chart.

   Each component shows the average number of active sessions in the specified state for the specified time. For example, if only one session were active, then the value .8 for CPU would mean that the session consumed CPU in 4 of 5 sampled seconds around the target time. The Maximum CPU equals the number of CPUs on the system. When the CPU Used value reaches the Maximum CPU line, the database instance is consuming 100 percent of CPU time on the host system.

   The wait classes show how much database activity is consumed by waiting for a resource such as disk I/O. Values that use a larger block of active sessions represent bottlenecks caused by a particular wait class, as indicated by the corresponding color in the legend.

   In the chart shown in Figure 4–2 on page 4-2, the largest amount of activity after 3:35 p.m. appears in dark green and corresponds to the CPU Used wait class.

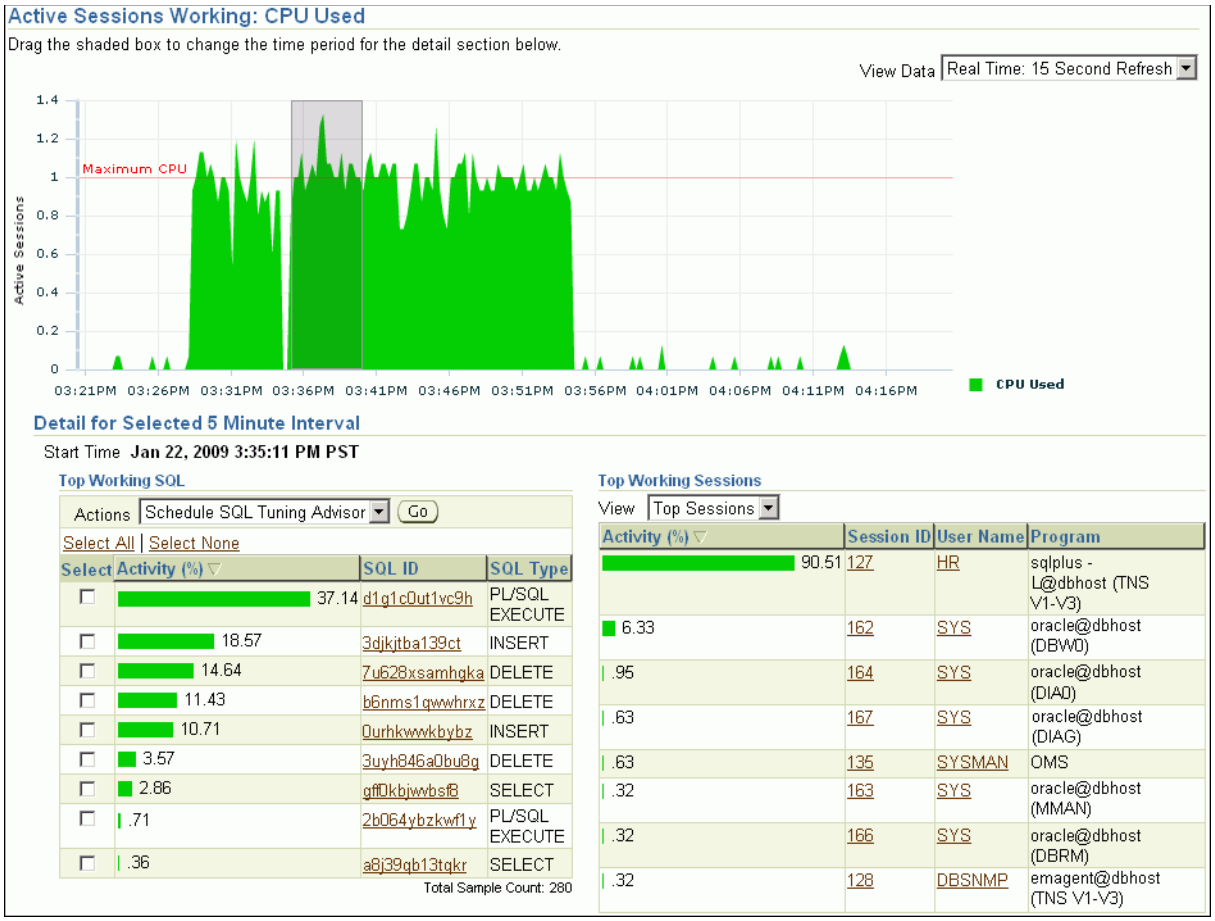
3. To identify each wait class, move your cursor over the block in the Average Active Sessions chart corresponding to the class.

   The corresponding wait class is highlighted in the chart legend.

4. Click the largest block of color on the chart or its corresponding wait class in the legend to drill down to the wait class with the most active sessions.

   If you click **CPU Used**, then the Active Sessions Working page for the wait class appears. If you click a different wait class, such as **User I/O**, then the Active Sessions Waiting page appears.

Figure 4–3 Active Sessions Working page



The Active Sessions Working page shows a 1-hour timeline. Details for each wait class are shown in 5-minute intervals under Detail for Selected 5 Minute Interval.

You can view the details of wait classes in different dimensions by proceeding to one of the following sections:

- "Monitoring Top SQL" on page 4-5
- "Monitoring Top Sessions" on page 4-6
- "Monitoring Top Services" on page 4-7
- "Monitoring Top Modules" on page 4-7
- "Monitoring Top Actions" on page 4-8
- "Monitoring Top Clients" on page 4-9
- "Monitoring Top PL/SQL" on page 4-10
- "Monitoring Top Files" on page 4-10
- "Monitoring Top Objects" on page 4-11

5. To change the selected time interval, move the slider below the chart to a different interval.

   The information contained in the Detail for Selected 5 Minute Interval section is automatically updated to display the selected time period.

In the example shown in Figure 4–3, the 5 -minute interval from 5:03 to 5:08 is selected for the CPU Used wait class.

6. If you discover a performance problem, then you can attempt to resolve it in real time. On the Performance page, do one of the following:

   - Below the chart, click the snapshot corresponding to the time when the performance problem occurred to run ADDM for this time period.

     For information about ADDM analysis, see "Reviewing the Automatic Database Diagnostic Monitor Analysis" on page 3-7.

   - Click **Run ADDM Now** to create a snapshot manually.

     For information about creating snapshots manually, see "Creating Snapshots" on page 3-5. For information about running ADDM manually, see "Manually Running ADDM to Analyze Current Database Performance" on page 6-1.

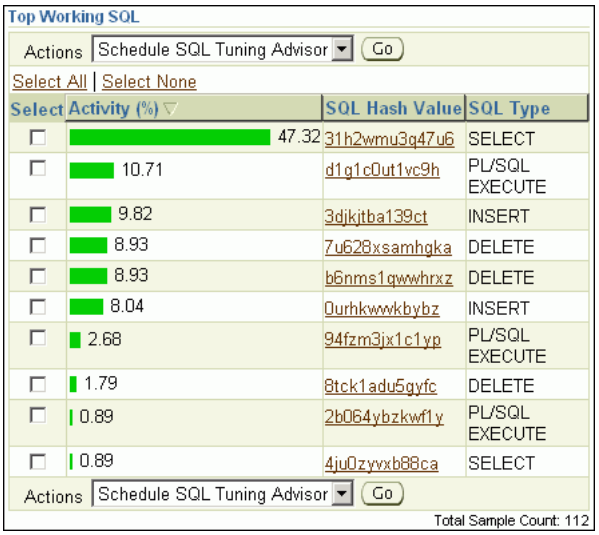
   - Click **Run ASH Report** to create an Active Session History (ASH) report to analyze transient, short-lived performance problems.

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

Monitoring Top SQL

On the Active Sessions Working page, the Top Working SQL table shows the database activity for actively running SQL statements that are consuming CPU resources. The Activity (%) column shows the percentage of this activity consumed by each SQL statement. If one or several SQL statements are consuming most of the activity, then you should investigate them.

Figure 4–4 Monitoring Top SQL



In the example shown in Figure 4–4, a single `SELECT` statement is consuming over 47% of database activity, while four modification DML statements are consuming about 35%. These statements should be investigated.

To monitor the top working SQL statements:

1. On the Performance page, in the Average Active Sessions chart, click the CPU block on the chart or its corresponding wait class in the legend.

The Active Sessions Working page appears.

2. In the Top Working SQL table, click the **SQL ID** link of the most active SQL statement.

   The SQL Details page appears.

   For SQL statements that are using the majority of the wait time, use SQL Tuning Advisor or create a SQL tuning set to tune the problematic SQL statements.

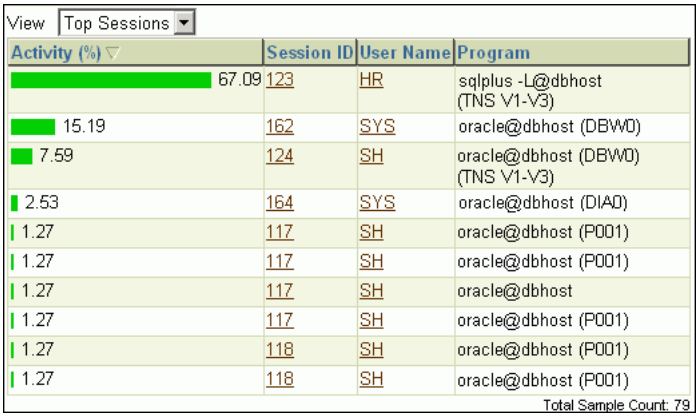
   See Also:

   - "Viewing Details of SQL Statements" on page 9-4
   - "Tuning SQL Statements Using SQL Tuning Advisor" on page 10-2

Monitoring Top Sessions

On the Active Sessions Working page, the Top Working Sessions table displays the top sessions waiting for the corresponding wait class during the selected time period. A **session** is a logical entity in the database instance memory that represents the state of a current user login to the database.

Figure 4–5 Monitoring Top Sessions



A session lasts from the time a user logs in to the database until the user disconnects. For example, when a user starts SQL*Plus, the user must provide a valid database user name and password to establish a session. If a single session is consuming the majority of database activity, then you should investigate it.

To monitor the top working sessions:

1. On the Performance page, in the Average Active Sessions chart, click the CPU Used block on the chart or its corresponding wait class in the legend.

   The Active Sessions Working page appears.

2. Under Detail for Selected 5 Minute Interval, select **Top Sessions** from the View list.

   The Top Working Sessions table appears.

3. In the Top Working Sessions table, click the **Session ID** link of the session consuming the most database activity.

   The Session Details page appears.

   This page contains information such as session activity, session statistics, open cursors, blocking sessions, wait events, and parallel SQL for the selected session.

If a session is consuming too much database activity, then consider clicking **Kill Session**, and then tuning the session's SQL statement.

See Also:

- Chapter 10, "Tuning SQL Statements"

Monitoring Top Services

The Top Services table displays the top services waiting for the corresponding wait event during the selected time period.

A **service** is a group of of applications with common attributes, service-level thresholds, and priorities. For example, the `SYS$USERS` service is the default service name used when a user session is established without explicitly identifying its service name. The `SYS$BACKGROUND` service consists of all database background processes. If a service is using the majority of the wait time, then you should investigate it.

调查、研究

To monitor a service:

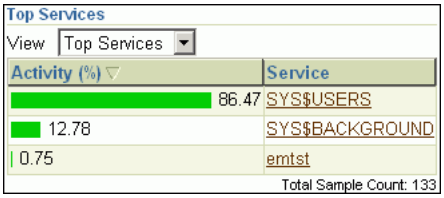
1. On the Performance page, in the Average Active Sessions chart, click a block on the chart or its corresponding wait class in the legend.

   The Active Sessions Working page appears.

2. Under Detail for Selected 5 Minute Interval, select **Top Services** from the View list.

   The Top Services table appears.

Figure 4–6 Monitoring Top Services



   In the example shown in Figure 4–6, the `SYS$USERS` service is consuming 86.47% of database activity. This service corresponds to the database sessions for users `hr` and `sh` shown in Figure 4–5.

3. Click the **Service** link of the most active service.

   The Service page appears.

   This page contains information about the modules, activity, and statistics for the selected service.

Monitoring Top Modules

The Top Modules table displays the top modules waiting for the corresponding wait event during the selected time period.

Modules represent the applications that set the service name as part of the workload definition. For example, the `DBMS_SCHEDULER` module may assign jobs that run within the `SYS$BACKGROUND` service. If a single module is using the majority of the wait time, then it should be investigated.

To monitor a module:

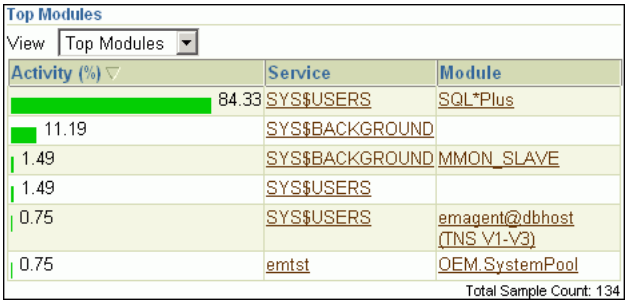
1. On the Performance page, in the Average Active Sessions chart, click a block on the chart or its corresponding wait class in the legend.

   The Active Sessions Working page appears.

2. Under Detail for Selected 5 Minute Interval, select **Top Modules** from the View list.

   The Top Modules table appears.

Figure 4–7 Monitoring Top Modules



3. Click the **Module** link of the module that is showing the highest percentage of activity.

   The Module page appears.

   This page contains information about the actions, activity, and statistics for the selected module.

   In the example shown in Figure 4–7, the SQL*Plus module is consuming over 84% of database activity and should be investigated. As shown in Figure 4–5, the SQL*Plus session for users `sh` and `hr` are consuming a huge percentage of database activity.

Monitoring Top Actions

The Top Actions table displays the top actions waiting for the corresponding wait event during the selected time period.

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 you should investigate it.

To monitor an action:

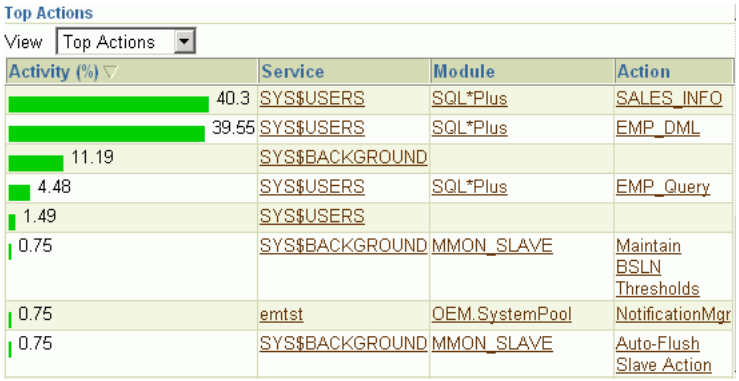
1. On the Performance page, in the Average Active Sessions chart, click a block on the chart or its corresponding wait class in the legend.

   The Active Sessions Working page appears.

2. Under Detail for Selected 5 Minute Interval, select **Top Actions** from the View list.

   The Top Actions table appears.

Figure 4–8 Monitoring Top Actions



3. Click the **Action** link of the most active action.

   The Action page appears.

   This page contains statistics for the selected action.

   In the example shown in Figure 4–8, the `SALES_INFO` action associated with the SQL*Plus module is consuming 40.3% of the database activity, while `EMP_DML` is consuming 39.55% and `EMP_QUERY` is consuming 4.48%. This information is consistent with Figure 4–5, which shows that the two database sessions for users `HR` and `SH` are consuming over 84% of database activity.

Monitoring Top Clients

The Top Clients table displays the top clients waiting for the corresponding wait event during the selected time period. A client can be a Web browser or any client process that initiates requests for an operation to be performed by the database. If a single client is using the majority of the wait time, then you should investigate it.

To monitor a client:

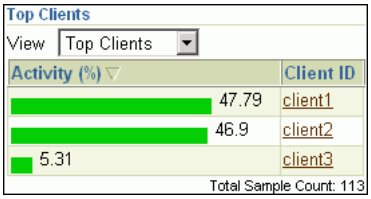
1. On the Performance page, in the Average Active Sessions chart, click a block on the chart or its corresponding wait class in the legend.

   The Active Sessions Working page appears.

2. Under Detail for Selected 5 Minute Interval, select **Top Clients** from the View list.

   The Top Clients table appears.

Figure 4–9 Monitoring Top Clients



3. Click the **Client ID** link of the most active client.

   The Clients page appears.

   This page contains statistics for the selected client process.

Monitoring Top PL/SQL

The Top PL/SQL table displays the top PL/SQL subprograms waiting for the corresponding wait event during the selected time period. If a single PL/SQL subprogram is using the majority of the wait time, then you should investigate it.

To monitor a PL/SQL subprogram:

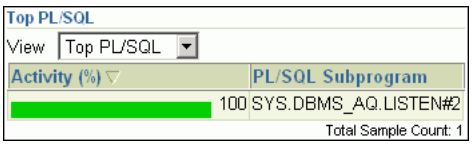
1. On the Performance page, in the Average Active Sessions chart, click a block on the chart or its corresponding wait class in the legend.

   The Active Sessions Working page appears.

2. Under Detail for Selected 5 Minute Interval, select **Top PL/SQL** from the View list.

   The Top PL/SQL table appears.

Figure 4–10 Monitoring Top PL/SQL



3. Click the **PL/SQL Subprogram** link of the most active subprogram.

   The PL/SQL Subprogram page appears.

   This page contains statistics for the selected subprogram.

   In Figure 4–10, the `SYS.DBMS_AQ.LISTEN#2` subprogram is consuming 100% of database activity.

Monitoring Top Files

The Top Files table displays the average wait time for specific files during the selected time period. This data is available from the Active Sessions Waiting: User I/O page.

To monitor a file:

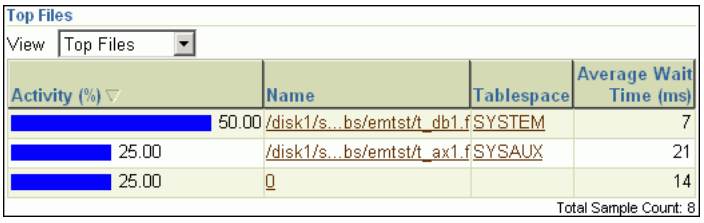
1. On the Performance page, in the Average Active Sessions chart, click the User I/O block on the chart or its corresponding wait class in the legend.

   The Active Sessions Waiting: User I/O page appears.

2. Under Detail for Selected 5 Minute Interval, select **Top Files** from the View list.

   The Top Files table appears.

Figure 4–11 Monitoring Top Files



3. Click the **Tablespace** link of the file with the highest average wait time.

   The View Tablespace page appears.

In the example shown in Figure 4–11, 75% of the wait times are associated with I/O to the files in the SYSTEM and SYSAUX tablespaces.

Monitoring Top Objects

The Top Objects table displays the top database objects waiting for the corresponding wait event during the selected time period. This data is available from the Active Sessions Waiting: User I/O page.

To monitor an object:

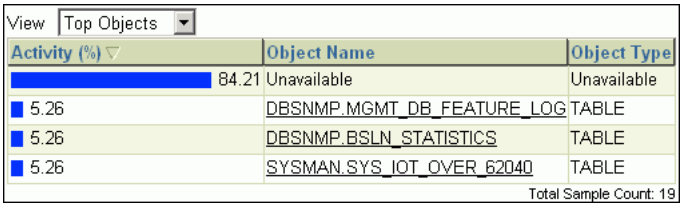
1. On the Performance page, in the Average Active Sessions chart, click the User I/O block on the chart or its corresponding wait class in the legend.

   The Active Sessions Waiting: User I/O page appears.

2. Under Detail for Selected 5 Minute Interval, select **Top Objects** from the View list.

   The Top Objects table appears.

Figure 4–12 Monitoring Top Objects



3. Click the **Object Name** link of the object with the highest average wait time.

   The View page for the object appears.

   This example in Figure 4–12 shows that over 84% of the waits are for an object whose name is unavailable. Based on the information in Figure 4–4 and Figure 4–5, you can conclude that the performance problem is caused by the query and modification DML statements.

Monitoring Instance Activity

In the Average Active Sessions section of the Performance page, you can use the instance charts to monitor database instance activity. As explained in "Customizing the Database Performance Page" on page 4-27, you can also customize the Performance page so that the most useful charts are displayed by default.

You can use the instance activity charts to perform the following tasks:

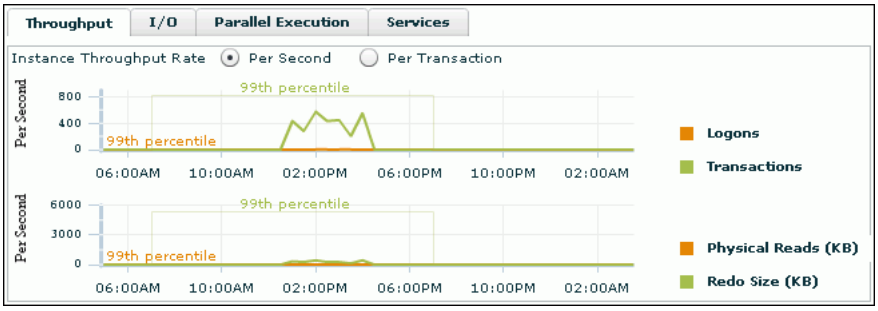
- Monitoring Throughput
- Monitoring I/O
- Monitoring Parallel Execution
- Monitoring Services

Monitoring Throughput

Database **throughput** measures the amount of work the database performs in a unit of time. The Throughput charts show any contention that appears in the Average Active Sessions chart. The Throughput charts on the Performance page display:

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

Figure 4–13 Monitoring Throughput



Compare the peaks on the Throughput charts with the peaks on the Average Active Sessions chart. If the Average Active Sessions chart displays a large number of sessions waiting, indicating internal contention, but throughput is high, then the situation may be acceptable. The database is probably also performing efficiently if internal contention is low but throughput is high. However, if internal contention is high but throughput is low, then consider tuning the database.

To monitor throughput:

1. From the Database Home page, click **Performance**.

   The Performance page appears.

2. In the instance activity chart, click **Throughput**.

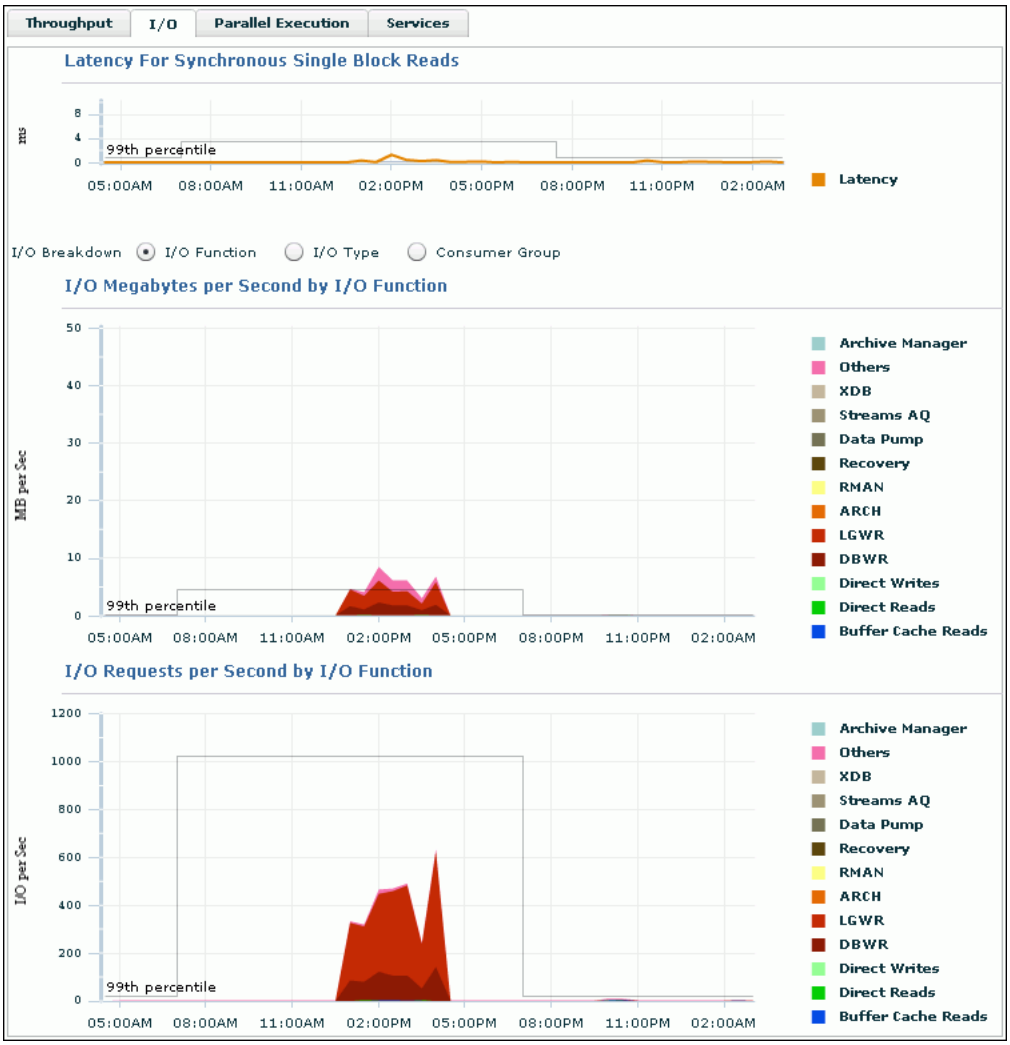
   The Throughput charts are shown with **Instance Throughput Rate** set to the default value of **Per Second**. You can select **Per Transaction** to show the throughput rate per transaction.

   In the example in shown in Figure 4–13, the number of transactions and physical reads per second went up around 12:30 p.m. and remained up until around 5 p.m.

Monitoring I/O

The I/O charts show I/O statistics collected from all database clients. The I/O wait time for a database process represents the amount of time that the process could have been doing useful work if a pending I/O had completed. Oracle Database captures the I/O wait times for all important I/O components in a uniform fashion so that every I/O wait by any Oracle process can be derived from the I/O statistics.

Figure 4–14 Monitoring I/O



The Latency for Synchronous Single Block Reads chart shows the total perceived **I/O latency** for a block read, which is the time difference between when an I/O request is submitted and when the first byte of the transfer arrives. Most systems are performing satisfactorily if latency is fewer than 10 milliseconds. This type of I/O request is the best indicator of I/O performance for the following reasons:

- Write operations may exhibit good performance because of write caches in storage.
- Because multiblock I/O requests have varying sizes, they can take different amounts of time.
- The latency of asynchronous I/O requests does not represent the full I/O wait time.

The other charts shown depend on your selection for **I/O Breakdown**, as described in the following sections:

- Monitoring I/O by Function
- Monitoring I/O by Type
- Monitoring I/O by Consumer Group

Monitoring I/O by Function

The I/O Function charts determine I/O usage level by application or job. The component-level statistics give a detailed view of the I/O bandwidth usage, which you can then use in scheduling jobs and I/O provisioning. The component-level statistics fall in the following categories:

- Background type

  This category includes ARCH, LGWR, and DBWR.

- Activity

  This category includes XML DB, Streams AQ, Data Pump, Recovery, and RMAN.

- I/O type

  The category includes the following:

  – Direct Write

    This write is made by a foreground process and is not from the buffer cache.

  – Direct Read

    This read is physical I/O from a data file that bypasses the buffer cache and reads the data block directly into process-private memory.

  – Buffer Cache Read

- Others

  This category includes I/Os such as control file I/Os.

To monitor I/O by function:

1. From the Database Home page, click **Performance**.

   The Performance page appears.

2. In the instance activity chart, click **I/O**.

   The I/O Megabytes per Second and I/O Requests per Second charts appear.

3. For **I/O Breakdown**, select **I/O Function**.

   The I/O Megabytes per Second by I/O Function and I/O Requests per Second by I/O Function charts appear.

   The example in Figure 4–14 shows that a significant amount of I/O is being performed by the log writer. The log writer activity peaked at approximately 500 I/O requests per second.

4. Click the largest block on the chart or its corresponding function in the legend to drill down to the function with the highest I/O rate.

   The I/O Details page appears.

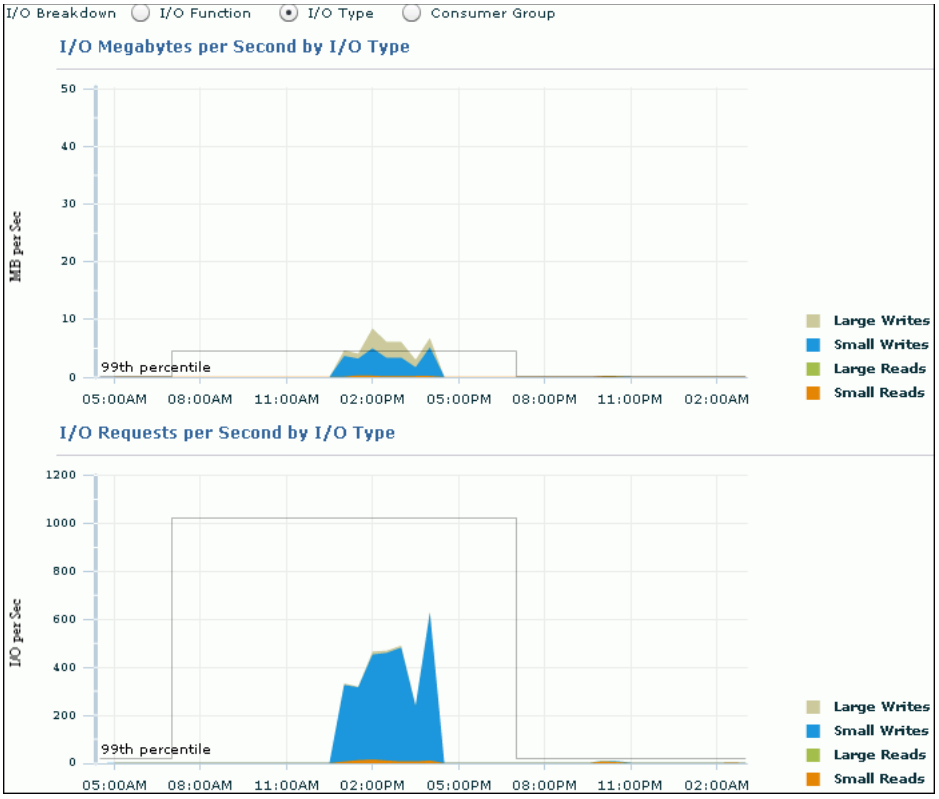
   You can view real-time or historical data for details on I/O megabytes or I/O requests.

   See Also:

   - *Oracle Database Concepts* to learn about database background processes such as ARCH, LGWR, and DBWR

Monitoring I/O by Type

The I/O Type charts enable you to monitor I/O by the types of read and write operations. Small I/Os are requests smaller than 128 KB and are typically single database block I/O operations. Large I/Os are requests greater than or equal to 128 KB. Large I/Os are generated by database operations such as table/index scans, direct data loads, backups, restores, and archiving.



When optimizing for short transaction times, such as in an OLTP environment, monitor latency for small I/Os. High latencies typically indicate that the storage system is a bottleneck.

When optimizing for large queries, such as in a data warehouse, performance depends on the maximum throughput the storage system can achieve rather than the latency of the I/O requests. In this case, monitor the I/O megabytes per second rather than the synchronous single-block I/O latencies.

To monitor I/O by type:

1. From the Database Home page, click **Performance**.

   The Performance page appears.

2. In the instance activity chart, click **I/O**.

   The I/O Megabytes per Second and I/O Requests per Second charts appear.

3. For **I/O Breakdown**, select **I/O Type**.

   The I/O Megabytes per Second by I/O Type and I/O Requests per Second by I/O Type charts appear.

   In this example, the number of small writes per second increased to more than 600. These writes correspond to the log writer I/O requests shown in Figure 4–14.

4. Click the largest block on the chart or its corresponding function in the legend to drill down to the function with the highest I/O rate.

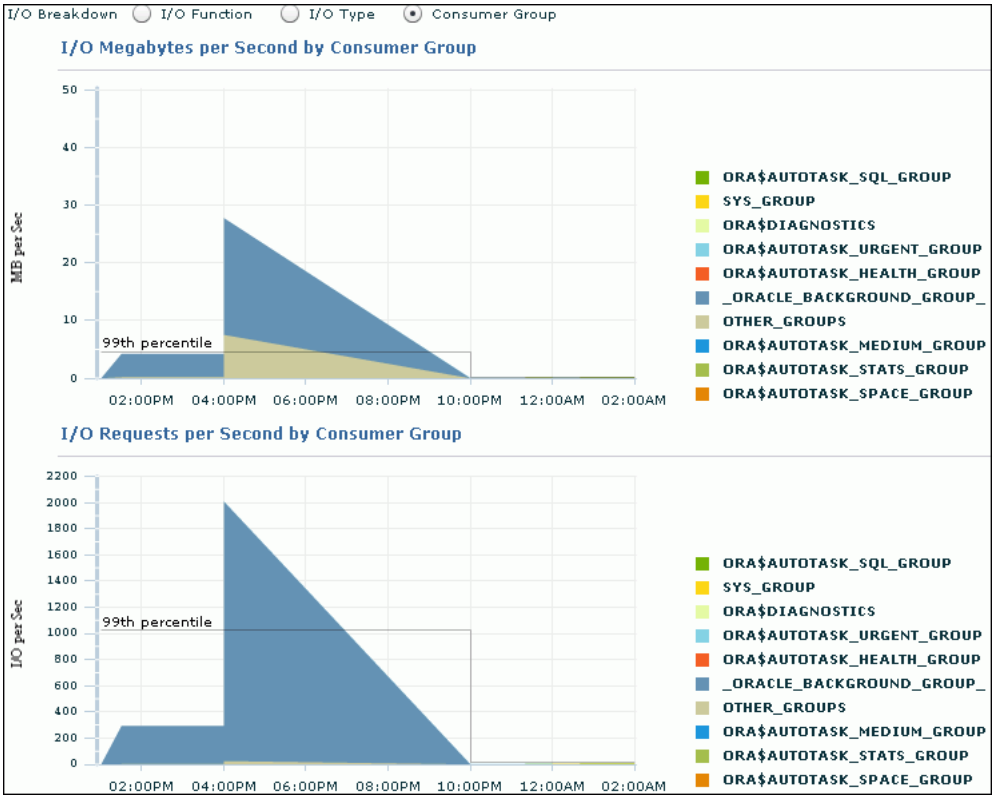
   The I/O Details page appears.

   You can view real-time or historical data for details on I/O megabytes or I/O requests.

Monitoring I/O by Consumer Group

When Oracle Database Resource Manager is enabled, the database collects I/O statistics for all consumer groups that are part of the currently enabled resource plan. The Consumer Group charts enable you to monitor I/O by consumer group.

A resource plan specifies how the resources are to be distributed among various users (resource consumer groups). Resource consumer groups enable you to organize user sessions by resource requirements. Note that the `_ORACLE_BACKGROUND_GROUP_` consumer group contains I/O requests issued by background processes.



To monitor I/O requests by consumer group:

1. From the Database Home page, click **Performance**.

   The Performance page appears.

2. In the instance activity chart, click **I/O**.

   The I/O Megabytes per Second and I/O Requests per Second charts appear.

3. For **I/O Breakdown**, select **Consumer Group**.

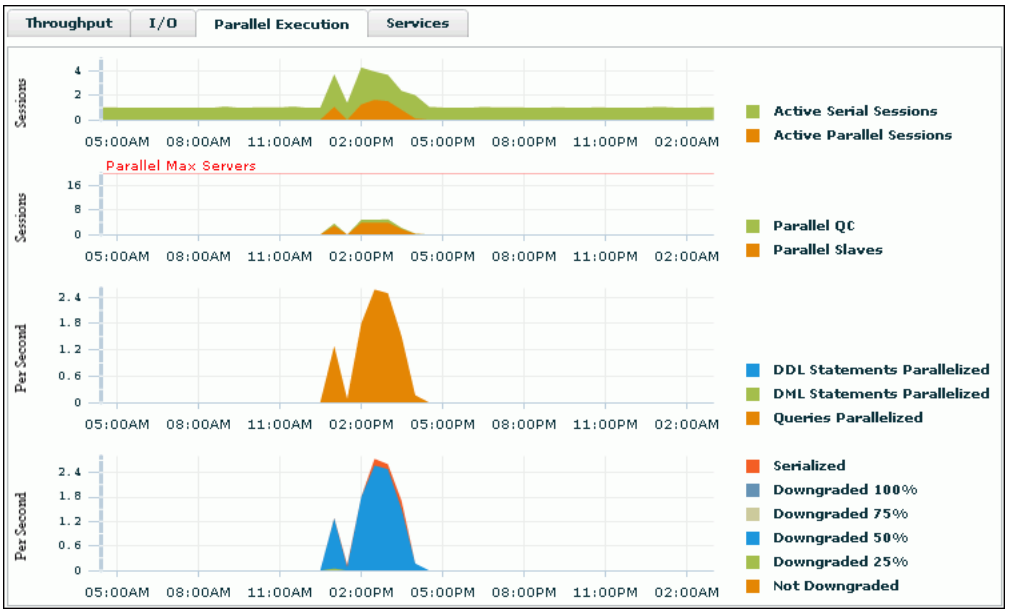
   The I/O Megabytes per Second by Consumer Group and I/O Requests per Second by Consumer Group charts appear.

Monitoring Parallel Execution

The Parallel Execution charts show system metrics related to parallel queries. **Metrics** are statistical counts per unit. The unit could be a time measure, such as seconds, or per transaction, or session.

A parallel query divides the work of executing a SQL statement across multiple processes. The charts show parallel queries that were waiting for a particular wait event that accounted for the highest percentages of sampled session activity.

Figure 4–15 Monitoring Parallel Execution



To monitor parallel execution:

1. From the Database Home page, click **Performance**.

   The Performance page appears.

2. In the instance activity chart, click **Parallel Execution**.

   The Parallel Execution charts appear.

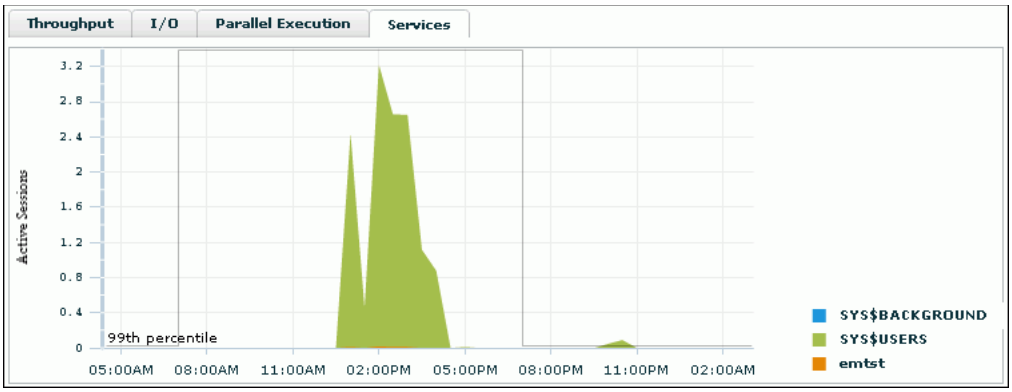
   Two pairs of charts are shown. The first pair shows the number of sessions on the y-axis, whereas the second pair shows the per second rate on the y-axis.

   In the example shown in Figure 4–15, query parallelization was active between 12:30 p.m. to 4 p.m.

Monitoring Services

The Services charts show services waiting for the corresponding wait event during the time period shown. 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. Only active services are shown.

Figure 4–16 Monitoring Services



To monitor services:

1. From the Database Home page, click **Performance**.

   The Performance page appears.

2. In the instance activity chart, click **Services**.

   The Services chart appears.

   In Figure 4–16, the `SYS$USERS` service has the greatest number of active sessions.

3. Click the largest block of color on the chart or its corresponding service in the legend to drill down to the service with the highest number of active sessions.

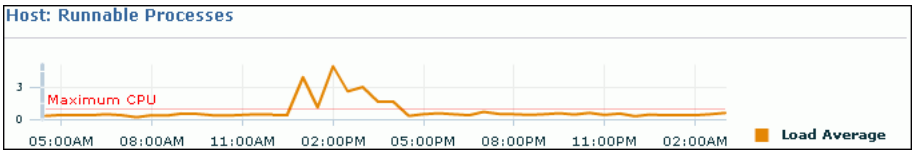
   The Service page appears, showing the Activity subpage.

   You can view real-time data showing the session load for all wait classes associated with the service.

Monitoring Host Activity

The Host chart on the Performance page displays utilization information about the system hosting the database.

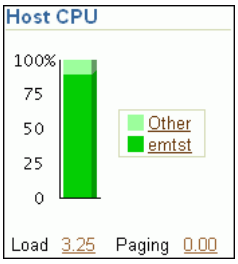
Figure 4–17 Monitoring Host Activity



To determine if the host system has enough resources available to run the database, establish appropriate expectations for the amount of CPU, memory, and disk resources that your system should be using. You can then verify that the database is not consuming too many of these resources.

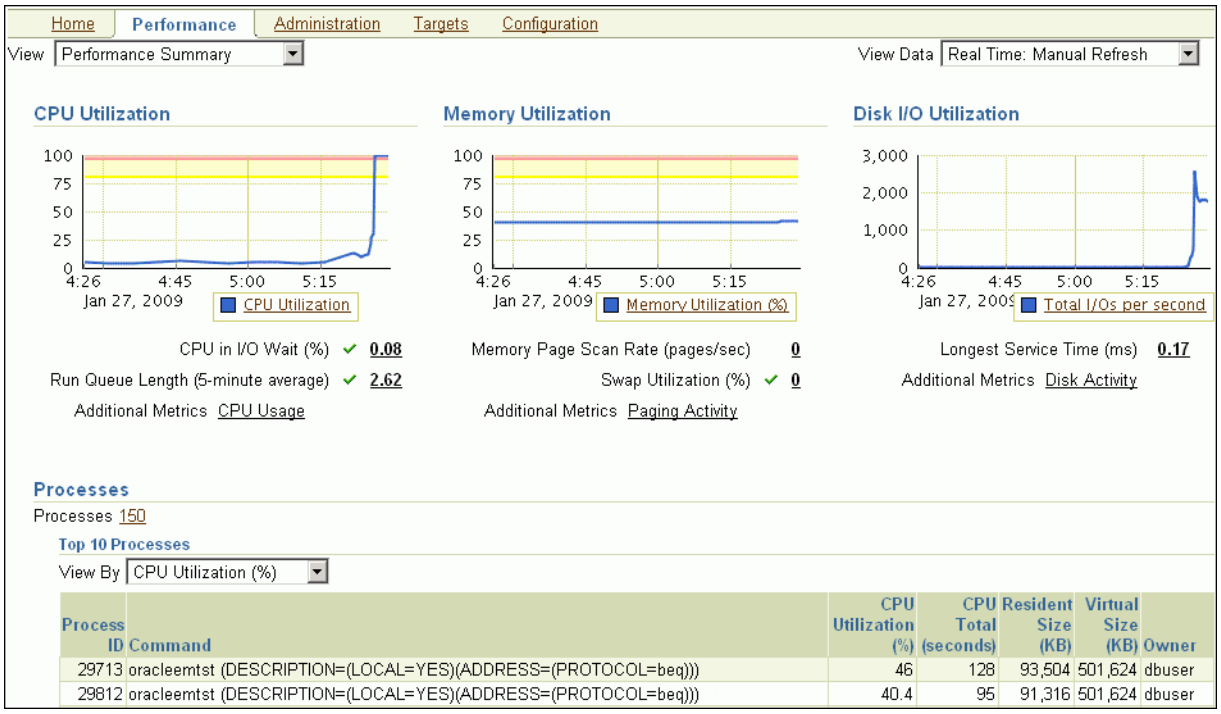
To view details about CPU, memory, and disk utilization:

1. From the Database Home page, under Host CPU, click the **Load** link.



The Host page appears, showing the Performance subpage.

Figure 4–18 Performance Summary



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

2. Determine whether sufficient resources are available and whether your system is using too many resources.

   For example, determine 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 nondatabase 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.

Perform the following tasks to monitor the host activity for your database:

- Monitoring CPU Utilization
- Monitoring Memory Utilization
- Monitoring Disk I/O Utilization

3. Set the appropriate threshold values for the performance 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.

Monitoring CPU Utilization

利用

To address CPU problems, first establish appropriate expectations for the amount of CPU resources your system should be using. You can then determine whether sufficient CPU resources are available and recognize when your system is consuming too many resources. This section describes how to monitor CPU utilization.

To monitor CPU utilization:

1. From the Database Home page, under Host CPU, click the **Load** link.

   The Host page appears, showing the Performance subpage.

2. Select **CPU Details** from the View list.

   The CPU Details view appears.

   This view contains statistics about CPU utilization, I/O wait times, and load gathered over the last hour. The top 10 processes are listed based on CPU utilization.

3. Verify the current CPU utilization using the CPU Utilization chart.

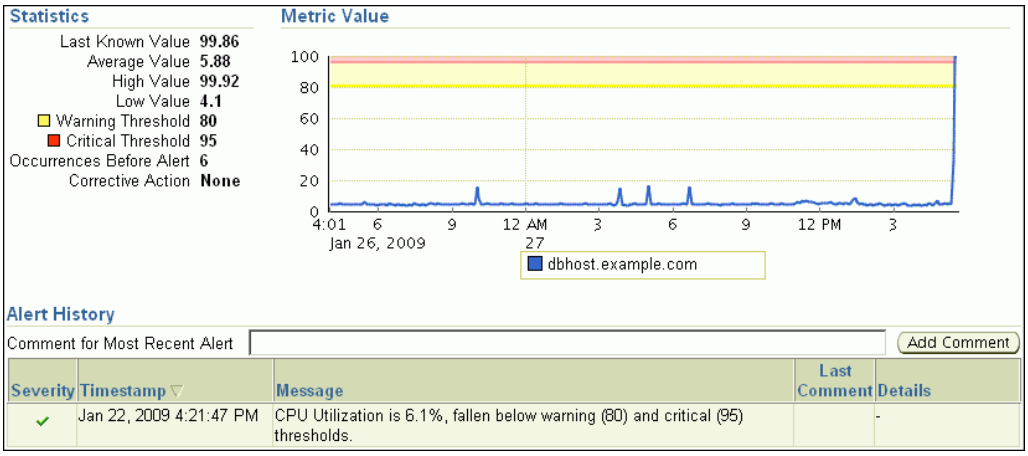
   The CPU Utilization chart shows CPU utilization over the last hour. The current value is displayed below the chart. During standard workload hours, the value should not exceed the critical threshold.

4. Click **CPU Utilization**.

   The CPU Utilization page appears.

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

   In the following example, the CPU utilization has suddenly spiked from approximately 6% to 99.86%, which is above the warning threshold of 80%.



If you notice an unexpected spike in this value that is sustained through normal workload hours, then the CPU performance problem should be investigated.

5. Verify the current CPU I/O wait time using the CPU I/O Wait chart.

   The CPU I/O Wait chart shows CPU I/O wait time over the last hour. The current value is displayed below the chart. During normal workload hours, the value of CPU I/O wait should not exceed the warning threshold.

   CPU I/O wait represents the average number of jobs waiting for I/O during an interval.

6. Click **CPU I/O Wait**.

   The CPU in I/O Wait page appears.

   This page contains CPU I/O wait statistics and related alerts generated over the last 24 hours.

   If you notice an unexpected increase in this value that is sustained through standard workload hours, then a CPU performance problem may exist.

7. Verify the current CPU load using the CPU Load chart.

   The CPU Load chart shows the CPU load over the last hour. The current value is displayed below the chart. During standard workload hours, the value of CPU load should not exceed the warning threshold.

   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.

8. Click **CPU Load**.

   The Run Queue Length page appears.

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

   If you notice an unexpected spike in this value that is sustained through normal workload hours, then a CPU performance problem might exist.

9. Return to the CPU Details view of the Host Performance subpage and review the Top 10 Processes table.

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

In the following example, two database processes are consuming 87.6% of CPU utilization. Therefore, the database is the likely source of a potential CPU performance problem and should be investigated.

Top 10 Processes (ordered by CPU)

Command	CPU Utilization (%)	CPU Total (seconds)	Resident Size (KB)	Virtual Size (KB)	Owner	Process ID
oracleemtst (DESCRIPTION=(LOCAL=YES)(ADDRESS=(PROTOCOL=beq)))	44.5	457	93,484	501,624	dbuser	29713
oracleemtst (DESCRIPTION=(LOCAL=YES)(ADDRESS=(PROTOCOL=beq)))	43.1	425	90,948	501,624	dbuser	29812

10. If a CPU performance problem is identified, then you can try to resolve the issue by doing the following:
    - Use Oracle Database Resource Manager to reduce the impact of peak-load-use patterns by prioritizing CPU resource allocation
    - Avoid running too many processes that use a large amount of CPU
    - Increase 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

Operating system performance issues commonly involve process management, memory management, and scheduling. This section describes how to monitor memory utilization and identify problems such as paging and swapping.

To monitor memory utilization:

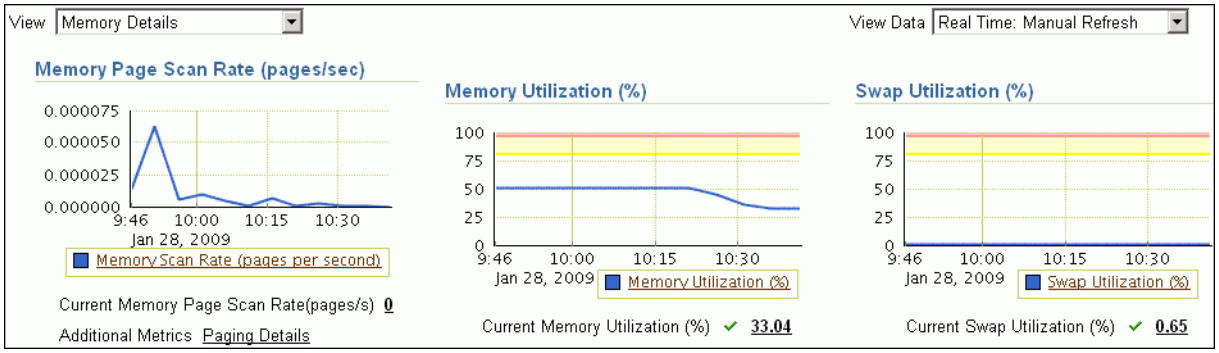
1. From the Database Home page, under Host CPU, click the **Load** link.

   The Host page appears, showing the Performance subpage.

2. Select **Memory Details** from the View list.

   The Memory Details view of the Performance subpage appears.

   This view contains statistics about memory utilization, page scan rates, and swap utilization gathered over the last hour. The top 10 processes are also listed ordered by memory utilization.



3. Verify the current memory page scan rate using the Memory Page Scan Rate chart.

The current value of the memory page scan rate is displayed below the chart. On UNIX and Linux, this value represents the number of pages scanned per second. On Microsoft Windows, this value represents the rate at which pages are read from or written to disk to resolve hard page faults. This value is a primary indicator of the kinds of faults that may be causing systemwide delays.

4. Click **Memory Scan Rate**.

   The Memory Page Scan Rate page appears.

   This page contains memory page scan rate statistics and related alerts over the last 24 hours.

   If you notice an unexpected increase in this value that is sustained through standard workload hours, then a memory performance problem might exist.

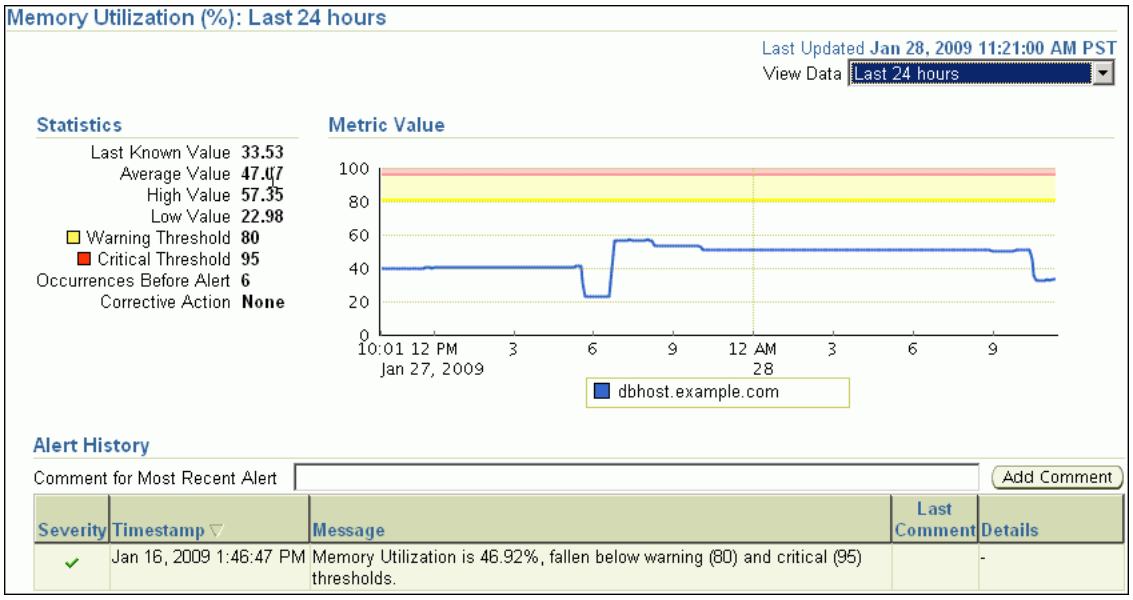
5. Verify the current memory utilization using the Memory Utilization chart.

   The Memory Utilization chart shows how much memory is being used. The current value of memory utilization is displayed below the chart. During standard workload hours, the value should not exceed the warning threshold (shown in yellow).

6. 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 never exceeded 60%, so a warning was 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.

7. Verify current swap utilization using the Swap Utilization chart.

   The Swap Utilization chart shows how much swap space is being used. The current value of swap utilization is displayed below the chart. During normal workload hours, the value should not exceed the warning threshold.

8. Click **Swap Utilization**.

   The Swap Utilization page appears.

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

   If you notice an unexpected spike in this value that is sustained through normal workload hours, then a memory performance problem might exist.

9. Return to the Memory Details view of the Host Performance subpage and review the top processes in the Top 10 Processes (ordered by Memory) table.

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

10. If a memory performance problem is identified, you can attempt to resolve the issue by doing the following:

    - Use Automatic Memory Management to automatically manage and distribute memory between the System Global Area (SGA) and the aggregate program global area (PGA aggregate).
    - Use the Memory Advisor to set SGA and PGA memory target values.
    - Use Automatic PGA Management to manage SQL memory execution.
    - Avoid running too many processes that consume large amounts of memory.
    - Reduce paging or swapping.
    - Reduce the number of open cursors and hard parsing with cursor sharing.

    See Also:

    - *Oracle Database Administrator's Guide* for information about using Automatic Memory Management
    - *Oracle Database 2 Day DBA* for information about using the Memory Advisor
    - *Oracle Database Performance Tuning Guide* for information about resolving memory issues

Monitoring Disk I/O Utilization

Because the database resides on a set of disks, the performance of the I/O subsystem is very important to database performance. Important disk statistics include the disk I/Os per second and the length of the service times. These statistics show if the disk is performing optimally or if the storage system is being overworked. This section describes how to monitor disk I/O utilization.

To monitor disk I/O utilization:

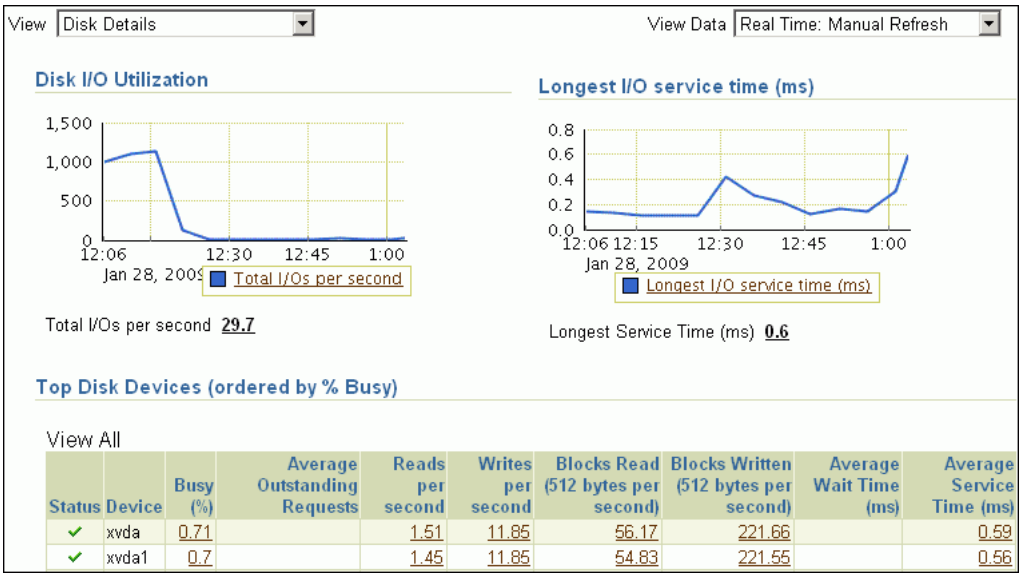
1. From the Database Home page, under Host CPU, click the **Load** link.

   The Host page appears, showing the Performance subpage.

2. Select **Disk Details** from the View list.

   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 the percentage of time that they were in use.



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)
✓	xvda	0.71		1.51	11.85	56.17	221.66		0.59
✓	xvda1	0.7		1.45	11.85	54.83	221.55		0.56

3. Verify the current disk I/O utilization using the Disk I/O Utilization chart.

   The Disk I/O Utilization chart shows how many disk I/Os are being performed per second. The current value for total I/Os per second is displayed below the chart.

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

   The Total Disk I/O Per Second page appears.

   This page contains disk utilization statistics and related alerts generated over the last 24 hours.

   If you notice an unexpected spike in this value that is sustained through standard workload hours, then a disk I/O performance problem might exist and should be investigated.

5. Verify the current I/O service time using the Longest I/O Service Time chart.

   The Longest I/O Service Time chart shows the longest service time for disk I/Os in milliseconds. The current value for longest I/O service time is displayed below the chart.

6. 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.

   If you notice an unexpected spike in this value that is sustained through normal workload hours, then a disk I/O performance problem might exist and should be investigated.

7. On the Disk Details page, verify the disk devices in the Top Disk Devices table.

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

   In this example, the drives that host Oracle Database (xvda and xvda1) are only busy about 1.41 percent of the time, so no disk performance problem appears to exist.

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)
✔	xvda	0.71		1.51	11.85	56.17	221.66		0.59
✔	xvda1	0.7		1.45	11.85	54.83	221.55		0.56

8. If a disk I/O performance problem is identified, you can attempt to resolve the problem by doing the following:
   - Use Oracle Automatic Storage Management (Oracle ASM) to manage database storage.
   - Stripe everything across every disk to distribute I/O.
   - Move files such as archived redo logs and online redo logs to separate disks.
   - Store 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

Determining the Cause of Spikes in Database Activity

If you see a spike in database activity in the Performance page, you can access the ASH Analytics page to find out which sessions are consuming the most database time. This page provides a stacked chart to help you visualize the active session activity for several dimensions, such as Event, Activity Class, Module/Action, Session, Instance ID, and PL/SQL function. You can drill down into specific members of a dimension (vertical zooming), and zoom in and out of any time period (horizontal zooming).

1. From the Performance menu in the database instance, select **ASH Analytics**.
2. View a macro perspective of top activity during a selected time period by dragging the shaded area in the top chart to the desired time period.

   Tip: When you move your mouse over the shaded block area, two user controls on both ends of the slider appear that enable you to select the desired time period.

   You can select a time period within the default setting of one hour, or you can use the selector buttons to display time periods of one day, one week, or one month. You can also use the Calendar and Custom buttons to display a time period other than one of the preset choices.

3. View a micro perspective of your selected time period by viewing the Activity chart on the page. By default, the chart shows workload activity breakdown values for wait classes.
4. Investigate the impact by viewing detailed statistics for the top activity sessions that are adversely affecting the system.
   a. Click the largest spike in the chart or the corresponding wait class in the legend beside the chart. The viewer now filters out everything in the chart except for the wait class of interest.

      For instance, if the chart shows that the Concurrency wait class has the biggest spike, click the chart area of the legend for Concurrency. The viewer refreshes the chart and now only shows the Concurrency value and displays a Wait Class: Concurrency icon in the Filters bar.

Tip: You can create an unlimited number of filters.

b. Expand the **Activity** icon, then select **Top Dimensions** from the menu, which provides a list of common processes of interest. The chart refreshes in response to your selection, displaying values for the particular category you selected.

   For instance, if you create a filter for Concurrency as described above, select Top Dimensions from the menu, then User Session, the chart shows user sessions only for Concurrency.

5. *Optional:* Use the Load Map for a graphical view of system activity.

   The Load Map is useful for viewing activity in a single- or multi-dimensional layout when you are not interested in seeing how activity has changed over time within the selected period.

6. *Optional:* Click **Save** to save the current page view as an HTML file for offline reference. When you click Save, a pop-up window appears, and you can specify where you want to save the report. This action creates an Enterprise Manager Active Report covering all data currently gathered as part of the analysis. You can use this later to conduct a more thorough postmortem analysis, for instance. You can view the report without Enterprise Manager or database connectivity.

   You can also click **Mail** and specify an email address to send the page as an attachment.

Customizing the Database Performance Page

You can customize the Performance page so that it specifically addresses your requirements. As explained in "Monitoring Instance Activity" on page 4-11, you can specify which charts you want to appear by default in the Performance page, and how you want them to appear. You can also decide whether to include baseline values in the Throughput and Services charts.

Enterprise Manager stores persistent customization information for each user in the repository. Enterprise Manager retrieves the customization data when you access the Performance page and caches it for the remainder of the browser session until you change the settings.

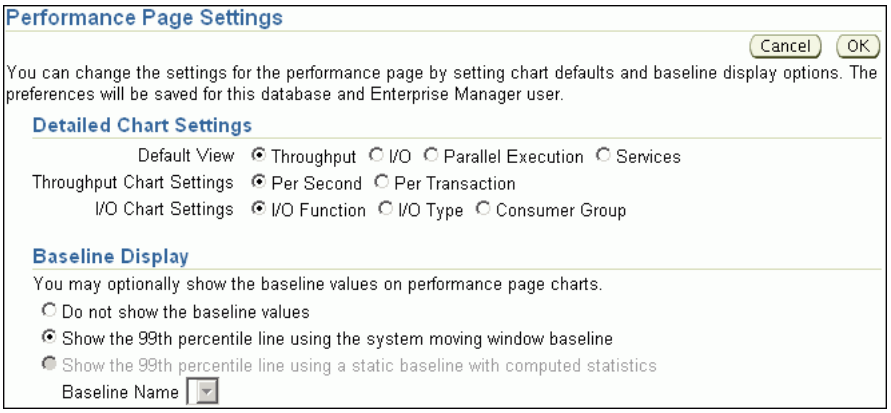
To customize the Performance page:

1. From the Database Home page, click **Performance**.

   The Performance page appears.

2. On the Performance page, click **Settings**.

   The Performance Page Settings page appears.



3. In the Detailed Chart Settings section, choose the defaults for display of the instance activity charts. Complete the following steps:

   a. In **Default View**, select the instance activity chart to appear by default in the Average Active Session section.

      See "Monitoring Instance Activity" on page 4-11 for a description of the Throughput, I/O, Parallel Execution, and Services charts.

   b. In **Throughput Chart Settings**, select **Per Second** or **Per Transaction** as the default instance throughput rate to be displayed in the Throughput chart.

      See "Monitoring Throughput" on page 4-11 to learn how to use the Throughput charts.

   c. In **I/O Chart Settings**, select the default I/O breakdown to be displayed in the I/O chart.

      See "Monitoring I/O" on page 4-12 to learn how to use the I/O charts.

4. In the Baseline Display section, choose how AWR baselines will be displayed in the performance charts. Do one of the following:

   - Select **Do not show the baseline values** to prevent baselines from appearing.
   - Select **Show the 99th percentile line using the system moving window baseline** to specify a percentile to display for the Throughput and Services charts.
   - Select **Show the 99th percentile line using a static baseline with computed statistics** and then select a baseline name from the Baseline Name list.

     You can select only baselines that have undergone schedule statistics computation, as described in "Computing Threshold Statistics for Baselines" on page 8-6.

5. Click **OK**.

   The Performance page appears.

   The charts are now displayed according to your customized settings.

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:

- Tablespace 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

A **metric** is the rate of change in a cumulative statistic. This rate can be measured against a variety of units, including time, transactions, or database calls. For example, the number of database calls per second is a metric. You can set thresholds on a metric so that an alert is generated when the threshold is passed.

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 alert 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 processing 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 obtain the most relevant information from performance alerts, 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 your system meets or exceeds your performance goals.

To set thresholds for performance metrics:

1. On the Database Home page, under Related Links, click **Metric and Policy Settings**.

   The Metric and Policy Settings page appears, showing the Metric Thresholds subpage.

2. For each performance metric relevant for your system, click the **Edit** icon.

   The Edit Advanced Settings page appears.

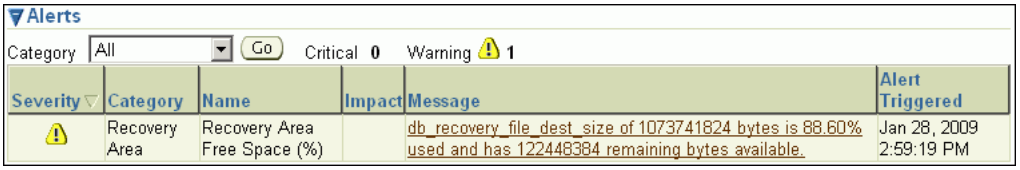
3. Follow the steps of the wizard to set the threshold value.

See Also:

- "Setting Metric Thresholds for Baselines" on page 8-7
- *Oracle Database 2 Day DBA* to learn how to set metric thresholds

Responding to Alerts

When an alert is generated by Oracle Database, it appears under Alerts on the Database Home page.



Oracle Enterprise Manager (Enterprise Manager) enables you to configure alerts to be sent by e-mail, pager, or cellular phone text messaging.

To respond to an alert:

1. On the Database Home page, under Alerts, locate the alert that you want to investigate and click the **Message** link.

   A page that contains further information about the alert appears.

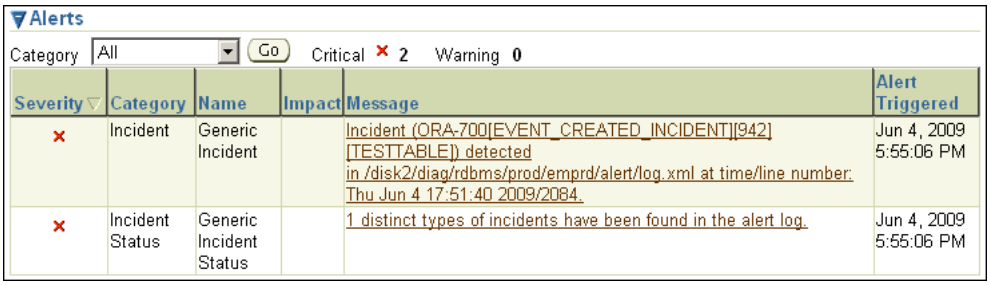
2. Do one of the following:
   - Follow the recommendations.
   - Run Automatic Database Diagnostic Monitor (ADDM) or another advisor to get more detailed diagnostics of the system or object behavior.

See Also:

- *Oracle Database 2 Day DBA* for information about how to configure the alert notification method

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 or Generic Incident alert, must be acknowledged.

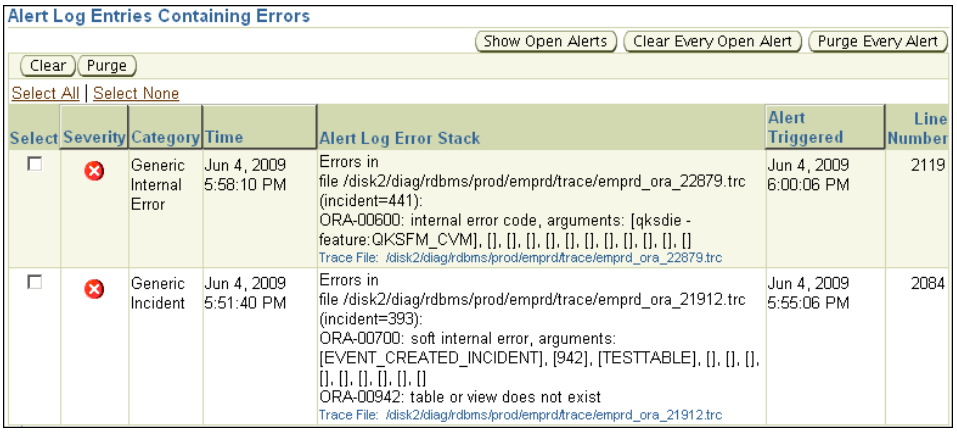


After taking the necessary corrective measures, you can acknowledge an alert by clearing or purging it. Clearing an alert sends the alert to the Alert History, which can be viewed from the Database Home page under Related Links. Purging an alert removes it from the Alert History.

To clear alerts:

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

   The Alert Log Errors page appears.



2. Do one of the following:
   - Select the alerts that you want to clear and click **Clear**.
   - To clear all open alerts, click **Clear Every Open Alert**.
3. Do one of the following:
   - Select the alerts that you want to purge and click **Purge**.
   - To purge all alerts, click **Purge Every Alert**.

See Also:

- *Oracle Database 2 Day DBA* to learn how to manage alerts

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 must be tuned immediately.

This part contains the following chapters:

6

Manual Database Performance Monitoring

You can run the Automatic Database Diagnostic Monitor (ADDM) manually to monitor current and historical database performance. Typically, you 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. It is possible to configure ADDM to run more or less frequently. However, in some cases you may want to run ADDM manually.

You can run ADDM manually to analyze a time period that is longer than one ADDM analysis period. For example, you may want to analyze database performance in a workday by analyzing 8 consecutive hours. You could analyze each of the individual ADDM periods within the workday, but this approach may become complicated if performance problems appear in only some ADDM periods. Alternatively, you can run ADDM manually with a pair of Automatic Workload Repository (AWR) snapshots that encompass the 8-hour period. In this case, ADDM identifies the most critical performance problems in the entire time period.

This chapter contains the following sections:

Manually Running ADDM to Analyze Current Database Performance

By default, ADDM runs every hour to analyze snapshots taken by AWR during this period. In some cases you may notice performance degradation that did not exist in the previous ADDM analysis period, or a sudden spike in database activity on the Performance page, as described in Chapter 4, "Monitoring Real-Time Database Performance". If the next ADDM analysis is not scheduled to run for 30 minutes, then you can run ADDM manually to identify and resolve the performance problem.

When you run ADDM manually, a manual AWR snapshot is created automatically. This manual run may affect 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. causes the next scheduled run to start at 9:30 p.m., not 9:00 p.m. Subsequent ADDM runs will continue on the new run cycle, occurring hourly at the half-hour instead of the start of each hour.

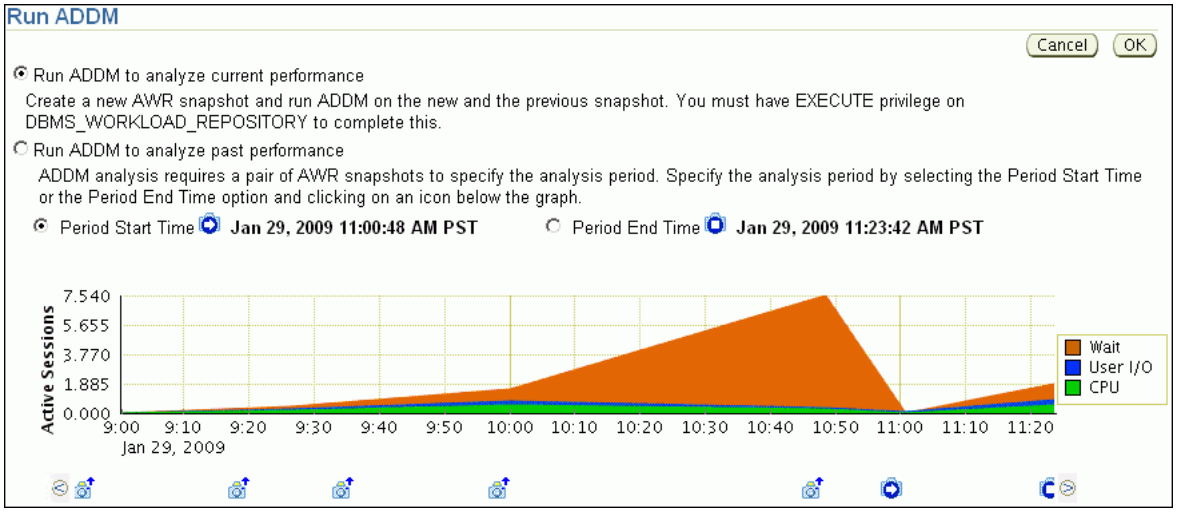
To analyze current database performance by manually running ADDM:

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, the average active sessions with wait events rose at 10:00 a.m., peaking at 10:50 a.m. The number dipped at 11:00 a.m. and then started to rise again at 11:10 a.m.

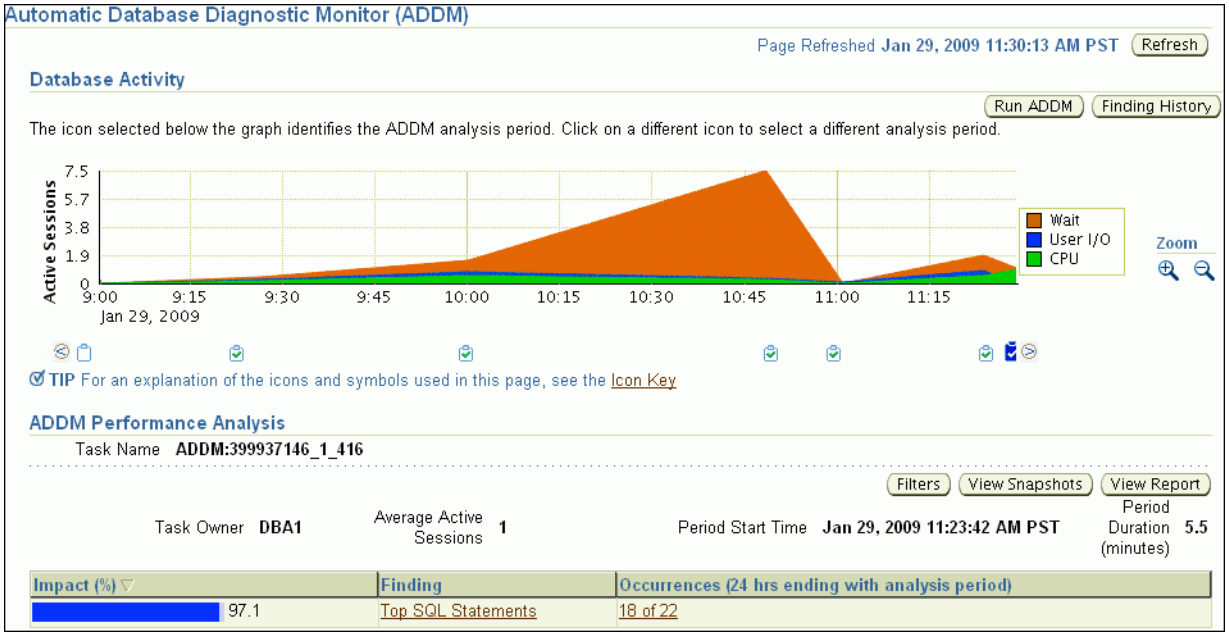
3. Select **Run ADDM to analyze current instance performance** and click **OK**.

   The Confirmation page appears.

4. Click **Yes**.

   The Processing: Run ADDM Now page appears while the database takes a new AWR snapshot.

   An ADDM run occurs for the time period between the new and the previous snapshot. After ADDM completes the analysis, the Automatic Database Diagnostic Monitor (ADDM) page appears with the results.



5. Click **View Report**.

   The View Report page appears.

6. Optionally, click **Save to File** to save the results of the ADDM task in a report for later access.

See Also:

- "Reviewing the Automatic Database Diagnostic Monitor Analysis" on page 3-7

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 technique is useful when you have identified a previous time period when database performance was poor.

In the Performance page, you can monitor historical performance by selecting **Historical** from the View Data list. In the Historical view, you can monitor database performance in the past, up to the duration defined by the AWR retention period. If you notice performance degradation, then you can drill down from the Performance page to identify historical performance problems with the database, as described in Chapter 4, "Monitoring Real-Time Database Performance". If you identify a problem, then you can run ADDM manually to analyze a 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. Complete the following steps:

   a. Select **Period Start Time**.

   b. Below the chart 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.

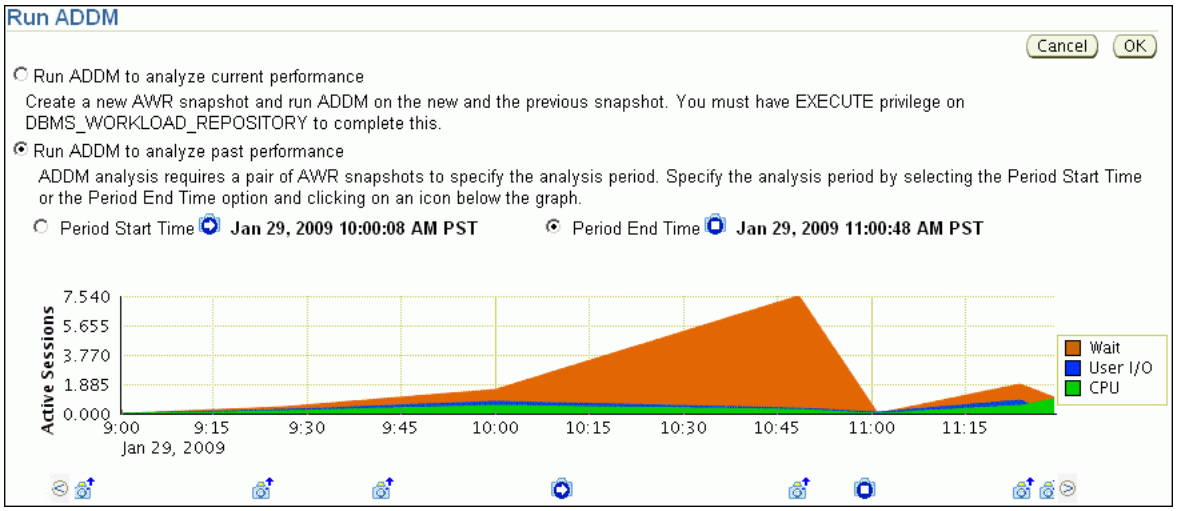
      In this example, database activity peaked from 10 a.m. to 11 a.m., so the snapshot taken at 10 a.m. is selected for the start time.

   c. Select **Period End Time**.

   d. Below the chart 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.

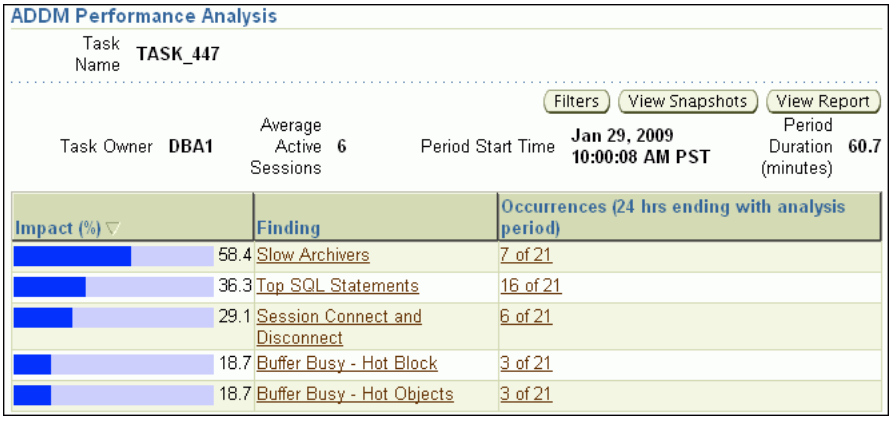
      In this example, the ending snapshot is at 11:00 a.m.



5. Click **OK**.

   After ADDM completes the analysis, the Automatic Database Diagnostic Monitor (ADDM) page appears with the results of the ADDM run.

Figure 6–1 Analyzing Historical Database Performance



6. Click **View Report**.

   The View Report page appears.

7. Optionally, click **Save to File**.

See Also:

- "Reviewing the Automatic Database Diagnostic Monitor Analysis" on page 3-7

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 run cycles.

To access the ADDM results:

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

   The Advisor Central page appears.

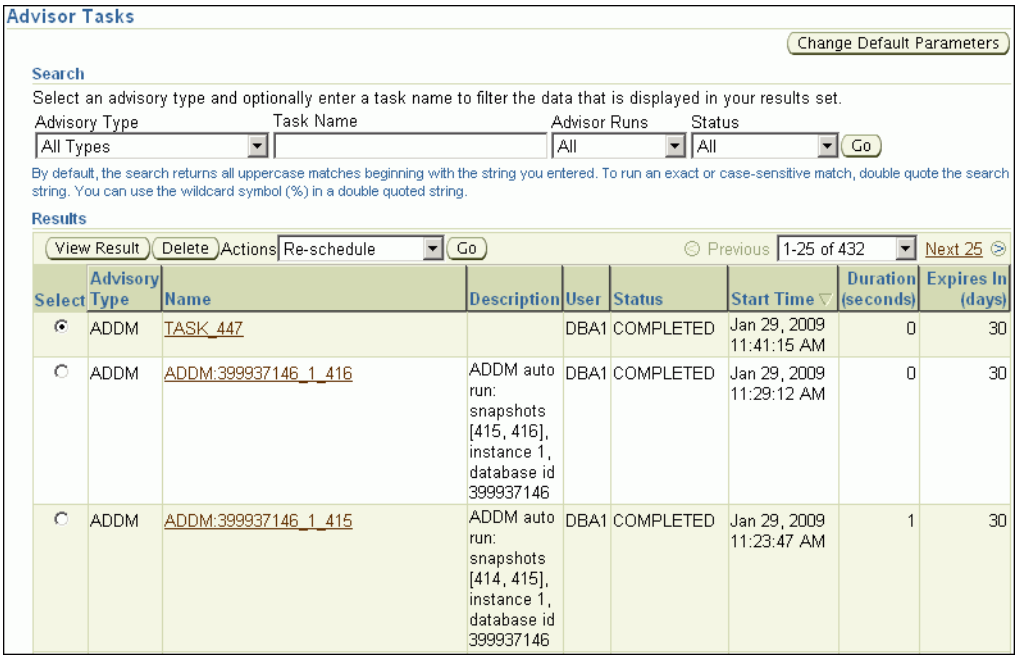
2. Complete the following steps:

   a. Under Advisor Tasks, select **ADDM** from the Advisory Type list.

   b. Select the appropriate search criteria.

      For example, you can select **All** in the Advisor Runs list to view all ADDM tasks.

   c. Click **Go**.



   The ADDM tasks are displayed under Results.

3. To view an ADDM result, select the desired ADDM task and click **View Result**.

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

See Also:

- "Reviewing the Automatic Database Diagnostic Monitor Analysis" on page 3-7

7

Resolving Transient Performance Problems

Transient performance problems are short-lived and typically do not appear in the Automatic Database Diagnostic Monitor (ADDM) analysis. ADDM tries to report the most significant performance problems during an analysis period in terms of their effect on DB time. If a problem lasts for a brief time, then 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, then it will be captured by ADDM. For example, if the snapshot interval is one hour, then a performance problem that lasts 30 minutes should not be considered 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 2 minutes could be transient because its duration is a small portion of the snapshot interval and will probably not appear in the ADDM findings. For example, if the system was slow between 10:00 p.m. and 10:10 p.m., and if the ADDM analysis for the time period between 10:00 p.m. and 11:00 p.m. does not show a problem, then a transient problem may have occurred for only a few minutes of the 10-minute interval.

This chapter contains the following sections:

Overview of Active Session History

To capture a detailed history of database activity, Oracle Database samples active sessions each second with the Active Session History (ASH) sampler. AWR snapshot processing collects the sampled data into memory and writes it to persistent storage. ASH is an integral part of the Oracle Database self-management framework and is extremely useful for diagnosing performance problems.

ASH gathers sampled data at the session level rather than at the instance level. By capturing statistics for only active sessions, ASH collects a manageable set of data. The size of this data is directly related to the work being performed, rather than to the size of the entire database instance.

Sampled data captured by ASH can be aggregated based on the dimensions in the data, including the following:

- 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 ASH reports to analyze transient performance problems with the database that only occur during specific times. This technique is especially useful when you are trying to do either of the following:

- 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 as usual
- 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 2-4

Running Active Session History Reports

This section describes how to generate ASH reports using Oracle Enterprise Manager (Enterprise Manager).

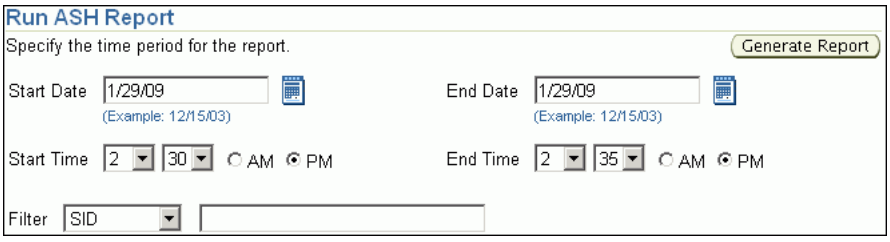
To run ASH reports:

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

   The Run ASH Report page appears.

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

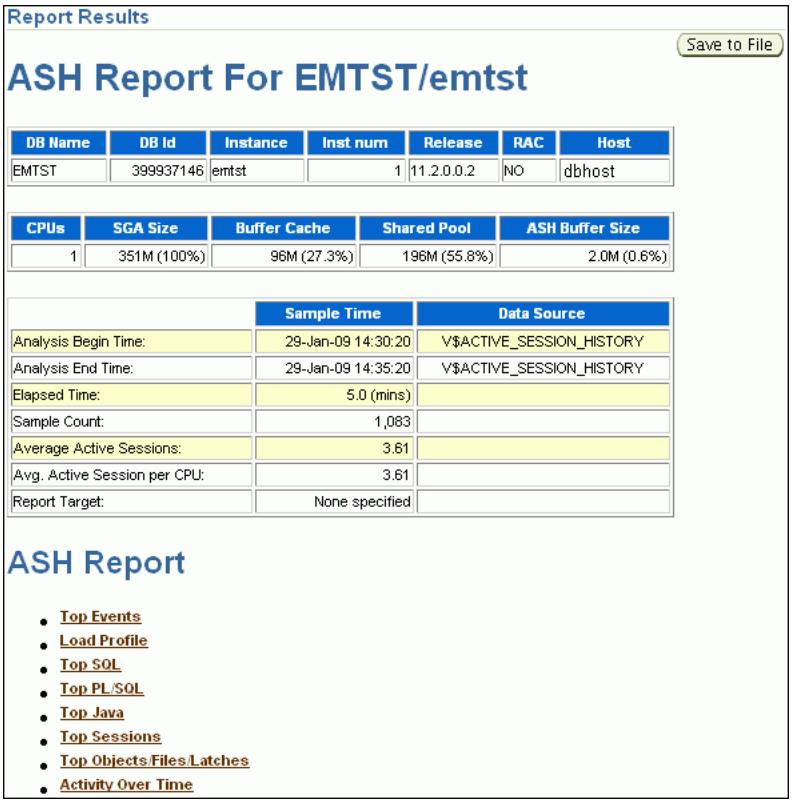
   In this example, database activity increased between 2:30 p.m. and 2:35 p.m., so an ASH report should be created for that time period.



3. Click **Generate Report**.

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

   After the report is generated, the ASH report appears under Report Results on the Run ASH Report page.



Report Results

Save to File

ASH Report For EMTST/emtst

DB Name	DB Id	Instance	Inst num	Release	RAC	Host
EMTST	399937146	emtst	1	11.2.0.0.2	NO	dbhost

CPUs	SGA Size	Buffer Cache	Shared Pool	ASH Buffer Size
1	351M (100%)	96M (27.3%)	196M (55.8%)	2.0M (0.6%)

	Sample Time	Data Source
Analysis Begin Time:	29-Jan-09 14:30:20	V$ACTIVE_SESSION_HISTORY
Analysis End Time:	29-Jan-09 14:35:20	V$ACTIVE_SESSION_HISTORY
Elapsed Time:	5.0 (mins)	
Sample Count:	1,083	
Average Active Sessions:	3.61	
Avg. Active Session per CPU:	3.61	
Report Target:	None specified	

ASH Report

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

4. Optionally, click **Save to File** to save the report in HTML format for future analysis.

Active Session History Reports

You can use an ASH report to identify the source of transient performance problems. The report is divided into titled sections. The following sections of the ASH report are useful places to begin the investigation:

调查

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

See Also:

- *Oracle Database Performance Tuning Guide* for more detailed information about the ASH report

Top Events

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

The Top Events section of the report contains the following subsections:

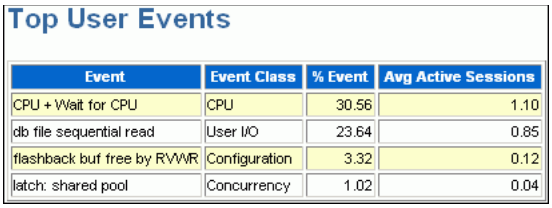
- Top User Events
- Top Background Events

Top User Events

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

Figure 7–1 shows that most database activity is consumed by the `CPU + Wait for CPU` event. The Wait for CPU is the time the process spent in the operating system run queue. The `%Event` column shows the percentage of DB time consumed by this event. In this example, over 30 percent of DB time was spent either on the CPU or waiting to get on it. The Load Profile section should be examined next to determine the type of activity that is causing this CPU consumption.

Figure 7–1 Top User Events



Top User Events

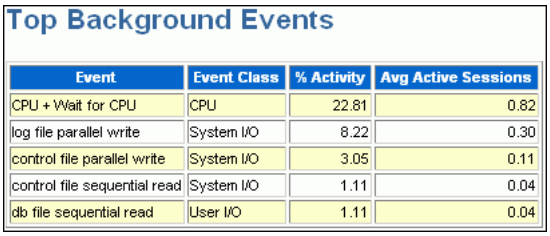
Event	Event Class	% Event	Avg Active Sessions
CPU + Wait for CPU	CPU	30.56	1.10
db file sequential read	User I/O	23.64	0.85
flashback buf free by RVWR	Configuration	3.32	0.12
latch: shared pool	Concurrency	1.02	0.04

Top Background Events

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

The example in Figure 7–2 shows that 22.81 percent of sampled session activity is consumed by the `CPU + Wait for CPU` event.

Figure 7–2 Top Background Events



Top Background Events

Event	Event Class	% Activity	Avg Active Sessions
CPU + Wait for CPU	CPU	22.81	0.82
log file parallel write	System I/O	8.22	0.30
control file parallel write	System I/O	3.05	0.11
control file sequential read	System I/O	1.11	0.04
db file sequential read	User I/O	1.11	0.04

Load Profile

The Load Profile section of the report 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 Top Service/Module subsection lists the services and modules that accounted for the highest percentages of sampled session activity. A **service** is a group of related database tasks that share common functionality, quality expectations, and priority. Services are a convenient way to monitor multiple applications. The `SYS$USERS` and `SYS$BACKGROUND` services are always defined.

Figure 7–3 shows that over half of the database activity is consumed by the SYS$USERS service running the SQL*Plus module. In this example, it appears that the user is running high-load SQL that is causing the performance problem indicated in Figure 7–1. The Top SQL section of the report should be analyzed next to determine whether a particular type of SQL statement makes up the load.

Figure 7–3 Top Service/Module

Top Service/Module

Service	Module	% Activity	Action	% Action
SYS$USERS	SQL*Plus	54.66	EMP_DML	27.33
			SALES_INFO	27.33
SYS$BACKGROUND	UNNAMED	35.83	UNNAMED	35.83
	MMON_SLAVE	4.16	Auto-Flush Slave Action	2.86
			Auto ADDM Slave Action	1.29
SYS$USERS	UNNAMED	2.31	UNNAMED	2.31
	emagent@dbhost (TNS V1-V3)	1.57	UNNAMED	1.57

See Also:

- "Monitoring Top Services" on page 4-7
- "Monitoring Top Modules" on page 4-7

Top SQL

The Top SQL section of the report 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 with Top Events subsection lists the SQL statements that accounted for the highest percentages of sampled session activity. The Sampled # of Executions column shows how many distinct executions of a particular SQL statement were sampled. To view the text of the SQL statements, click the **SQL ID** link.

Figure 7–4 shows that over half of DB time is consumed by three DML statements. These statements were run in the SQL*Plus module shown in Figure 7–3. The Top Sessions section should be analyzed to identify the sessions running these statements.

Figure 7–4 Top SQL with Top Events

Top SQL with Top Events

SQL ID	Planhash	Sampled # of Executions	% Activity	Event	% Event	Top Row Source	% RwSrc	SQL Text
31h2wmu3q47u6	4090065844	277	26.22	CPU + Wait for CPU	26.13	TABLE ACCESS - FULL	19.94	SELECT /*+ ORDERED USE_NL(c) F...
3djkjtba139ct	1426549735	148	13.76	db file sequential read	12.00	** Row Source Not Available **	12.00	INSERT INTO EMPLOYEES VALUES (...
				flashback buf free by RVWR	1.39	** Row Source Not Available **	1.39	
7u628xsamhqka	781635435	127	11.82	db file sequential read	9.70	DELETE	9.70	DELETE FROM EMPLOYEES WHERE EM...
				flashback buf free by RVWR	1.75	DELETE	1.75	

See Also:

- "Monitoring Top SQL" on page 4-5

Top Sessions

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

The `# Samples Active` column shows the number of ASH samples in which the session was found waiting for that particular event. The percentage is calculated based on wall-clock time.

In Figure 7–5, the `# Samples Active` column shows that of the 300 times that ASH sampled database activity, the `HR` session (SID 123) performed a sequential read 243 times and a flashback operation 36 times. So, `HR` was active at least 93% of the time. The session consumed 27% of the total activity (much less than 93%) because other sessions, including the `SH` session, were also active.

It appears that the `HR` and `SH` sessions were running the high-load SQL statement in Figure 7–4. 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 is not possible, and if a session is causing an unacceptable performance impact on the system, then consider terminating the session.

Figure 7–5 Top Sessions

dbh

Sid, Serial#	% Activity	Event	% Event	User	Program	# Samples Active	XIDs
123, 275	27.33	db file sequential read	22.44	HR	sqlplus -L@dbh...1 (TNS V1-V3)	243/300 [81%]	16
		flashback buf free by RVWR	3.32			36/300 [12%]	4
126,28313	27.33	CPU + Wait for CPU	26.78	SH	sqlplus -L@dbh...1 (TNS V1-V3)	290/300 [97%]	0
154, 5	15.24	CPU + Wait for CPU	11.54	SYS	oracle@dbhost (RVWR)	125/300 [42%]	0
		control file parallel write	2.12			23/300 [8%]	0
161, 1	8.31	log file parallel write	8.22	SYS	oracle@dbhost (LGWR)	89/300 [30%]	0
110,36790	2.86	CPU + Wait for CPU	1.11	SYS	oracle@dbhost (M000)	12/300 [4%]	0

See Also:

- "Monitoring Top Sessions" on page 4-6
- Chapter 10, "Tuning SQL Statements"

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.

The example in Figure 7–6 shows that the `hr.departments` and `hr.employees` tables account for a high percentage of activity. Enqueue waits are waits for locks. In this example, the wait is for the TM (table) lock. Sometimes these waits indicate unindexed foreign key constraints. The `buffer busy waits` event records waits for a buffer to become available. These waits indicate that multiple processes are attempting to concurrently access the same buffers in the buffer cache.

Figure 7–6 Top DB Objects

Top DB Objects

- With respect to Application, Cluster, User I/O and buffer busy waits only.

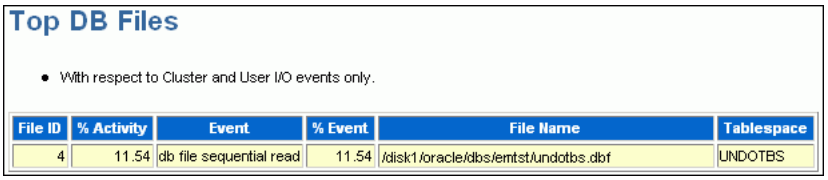
Object ID	% Activity	Event	% Event	Object Name (Type)	Tablespace
66590	32.10	enq: TM - contention	32.10	HR.DEPARTMENTS (TABLE)	EXAMPLE
66595	1.99	buffer busy waits	1.99	HR.EMPLOYEES (TABLE)	EXAMPLE

Top DB Files

The Top DB Files subsection lists the database files that accounted for the highest percentages of sampled session activity. Only cluster and I/O events are considered. The `% Event` column breaks down the activity by event, so if multiple rows exist in this table, then the sampled activity is divided among multiple events.

Figure 7–7 shows that about 11 percent of DB time involves waits for the `UNDOTBS` tablespace. This information is consistent with Figure 7–4, which shows significant DML activity from multiple sessions.

Figure 7–7 Top DB Files


Top DB Files

- With respect to Cluster and User I/O events only.

File ID	% Activity	Event	% Event	File Name	Tablespace
4	11.54	db file sequential read	11.54	/disk1/oracle/dbs/emtst/undotbs.dbf	UNDOTBS

Activity Over Time

The Activity Over Time section of the ASH report 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 **time slots**. The ASH report time span is divided into 10 time slots unless the time period is short or the data is sparse.

Figure 7–8 shows an activity report for the period between 2:10 p.m. and 2:40 p.m. The report indicates that the number of sampled sessions rose sharply in the fifth inner slot (2:24 p.m.) and stayed up. During this period CPU activity and lock enqueue waits increased dramatically.

Figure 7–8 Activity Over Time

Slot Time (Duration)	Slot Count	Event	Event Count	% Event
14:10:50 (1.2 min)	5	control file sequential read	4	0.11
		CPU + Wait for CPU	1	0.03
14:12:00 (3.0 min)	9	CPU + Wait for CPU	5	0.14
		control file parallel write	2	0.05
		null event	1	0.03
14:15:00 (3.0 min)	8	control file parallel write	4	0.11
		control file sequential read	4	0.11
14:18:00 (3.0 min)	10	control file sequential read	6	0.16
		control file parallel write	3	0.08
		SQL*Net break/reset to client	1	0.03
14:21:00 (3.0 min)	14	CPU + Wait for CPU	5	0.14
		control file parallel write	5	0.14
		control file sequential read	3	0.08
14:24:00 (3.0 min)	275	CPU + Wait for CPU	95	2.60
		enq: TM - contention	60	1.64
		control file sequential read	36	0.99
14:27:00 (3.0 min)	703	enq: TM - contention	187	5.12
		CPU + Wait for CPU	175	4.79
		log file switch (checkpoint incomplete)	81	2.22
14:30:00 (3.0 min)	737	enq: TM - contention	210	5.75
		CPU + Wait for CPU	199	5.45
		enq: CF - contention	95	2.60
14:33:00 (3.0 min)	713	enq: TM - contention	181	4.96
		CPU + Wait for CPU	176	4.82
		enq: CF - contention	84	2.30
14:36:00 (3.0 min)	740	enq: TM - contention	222	6.08
		CPU + Wait for CPU	212	5.81
		enq: CF - contention	80	2.19
14:39:00 (1.8 min)	437	enq: TM - contention	126	3.45
		CPU + Wait for CPU	114	3.12
		enq: CF - contention	52	1.42

Each time slot contains session and wait event activity, 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

All inner slots are the same number of minutes each for easy comparison. The first and last slots, called **outer slots**, are odd-sized because they do not have a fixed slot time.

In Figure 7–8, the first outer slot has a duration of 1.2 minutes, whereas the last outer slot has a duration of 1.8 minutes. The duration of each inner slot is 3.0 minutes.

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

Diagnosing Malfunctioning Systems in an Emergency

The following procedure is useful for diagnosing serious performance problems at the moment they are occurring. This may assist you in finding a remedy to resolve the problem other than rebooting the system.

1. From the Performance Home page, select the Performance menu, then select **Emergency Monitoring**.

   The Emergency Performance page appears and displays collected ASH data. The page also shows top blocking sessions in the Hang Analysis table.

   If the information displayed on this page does not help you resolve the problem, continue to the next step.

2. Select the Performance menu, then select **Real-Time ADDM**.

3. Click **Start** from the Real-Time ADDM page.

   The system responds by collecting performance data from all target database instances, analyzing data for problem diagnosis and resolutions, and showing you results of the analysis.

4. Click the **Findings** tab for a clear and interactive summary of all of the findings the analysis has detected, and to view actionable recommendations.

5. *Optional*: Click **Save** to save the current page view as an HTML file for offline reference. When you click Save, a pop-up window appears, and you can specify where you want to save the report. This action creates an Enterprise Manager Active Report covering all data currently gathered as part of the analysis. You can use this later to conduct a more thorough postmortem analysis, for instance. You can view the report without Enterprise Manager or database connectivity.

   You can also click **Mail** and specify an email address to send the report as an attachment.

Example

1. Your database is currently having severe performance problems, so you click the **Emergency Performance** page link.

2. Neither ASH nor Hang Analysis indicate the root cause or offer quick solutions, so you then click **Emergency ADDM**.

3. You view the active report generated by Emergency ADDM, which shows that the system is paging due to excessive PGA consumption by session S1, possibly due to a leak. The report recommends that you terminate S1 immediately.

4. You terminate session S1, then return to the Emergency Performance page to see if normal system behavior is restored.

5. You note that the system is making progress according to the activity charts on the page.

6. You return to the regular Performance Home page, and observe that database performance has improved.

8

Resolving Performance Degradation Over Time

退化

Performance degradation of the database occurs 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 totals four snapshots). Using the AWR Compare Periods report helps you to identify detailed performance attributes and configuration settings that differ between two time periods. The two time periods selected for the AWR Compare Periods report can be of different durations. 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.

For example, a batch workload that historically completed in the maintenance window between 10:00 p.m. and midnight is currently showing poor performance and completing at 2 a.m. You can generate an AWR Compare Periods report from 10:00 p.m. to midnight on a day when performance was good and from 10:00 a.m. to 2 a.m. on a day when performance was poor. The comparison of these reports should identify configuration settings, workload profile, and statistics that were different in these two time periods. Based on the differences identified, you can more easily diagnose the cause of the performance degradation.

This chapter contains the following sections:

See Also:

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

Managing 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.

清除

A moving window baseline corresponds to all AWR data that exists within the AWR retention period. Oracle Database automatically maintains a system-defined moving window baseline. The default size of the window is the current AWR retention period, which by default is 8 days.

This section contains the following topics:

- Creating a Baseline
- Deleting a Baseline
- Computing Threshold Statistics for Baselines
- Setting Metric Thresholds for Baselines

Creating a Baseline

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

You can create the following types of baseline:

- Creating a Single Baseline
- Creating a Repeating Baseline

Creating a Single Baseline

A single baseline is captured at a single, fixed time interval. For example, a single baseline may be captured on February 5, 2009 from 5:00 p.m. to 8:00 p.m.

You can choose future start and end times to create a baseline that captures future database activity. If both the start time and the end time are in the future, then a baseline template with the same name as the baseline will also be created. A baseline template is a specification that enables Oracle Database to automatically generate a baseline for a future time period.

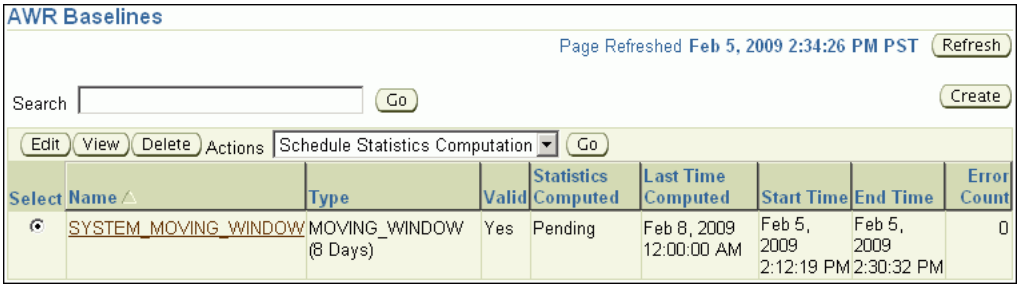
To create a single baseline:

1. From the Database Home page, click **Server**.

   The Server page appears.

2. Under Statistics Management, click **AWR Baselines**.

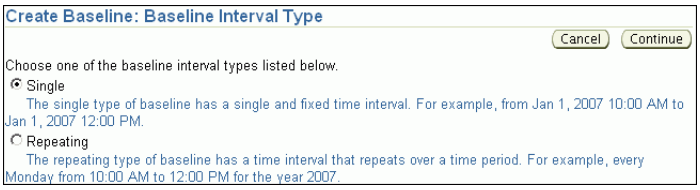
   The AWR Baselines page appears with a list of existing baselines displayed.



3. Click **Create**.

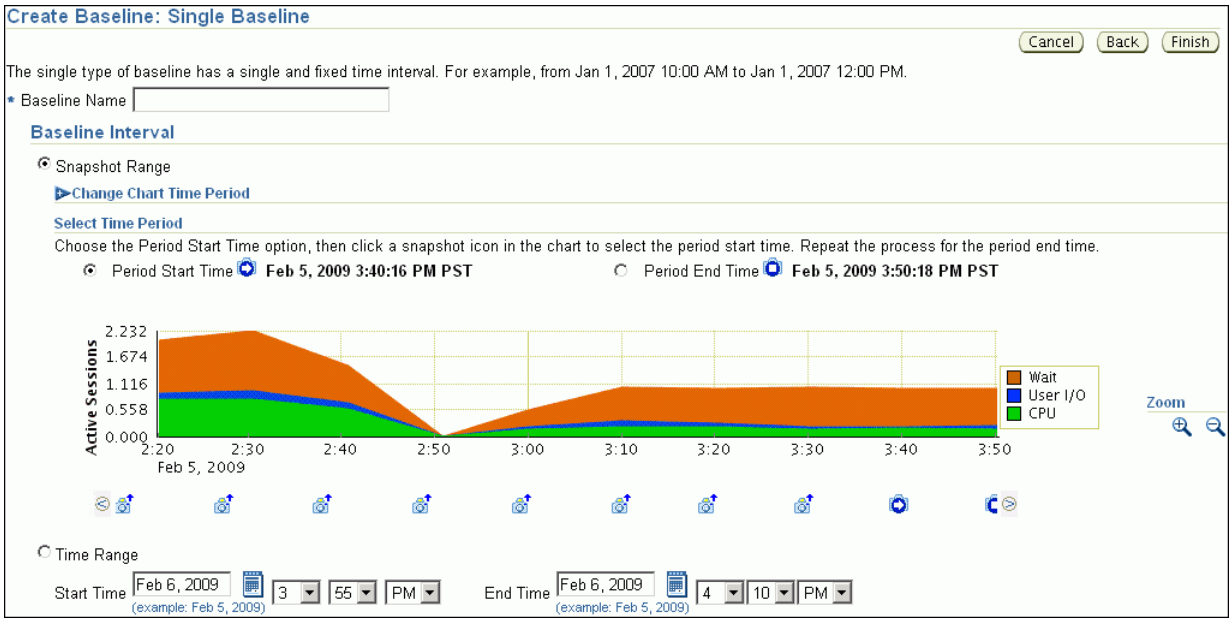
   The Create Baseline: Baseline Interval Type page appears.

4. Select **Single**.



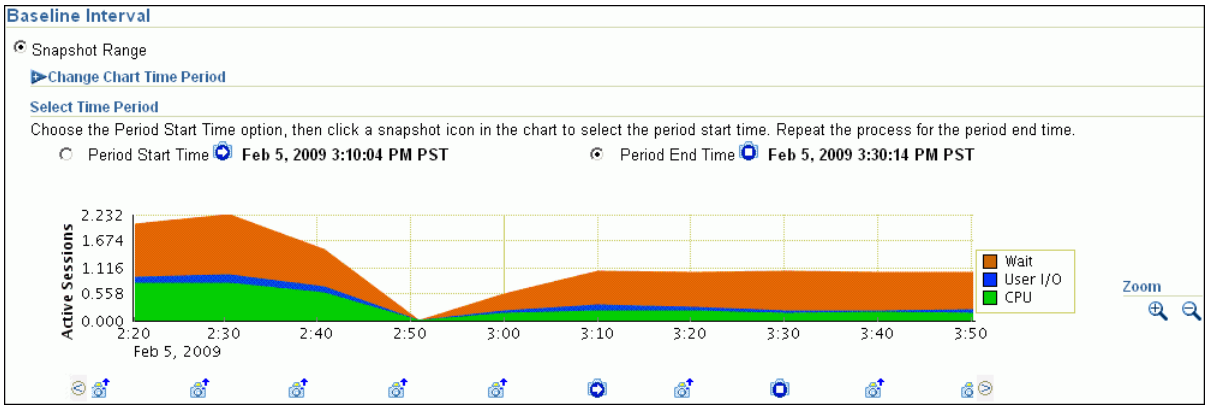
5. Click **Continue**.

   The Create Baseline: Single Baseline page appears.



6. In the **Baseline Name** field, enter a name for the baseline.
7. Under Baseline Interval, select whether to use a snapshot range or a time range for the baseline. Do one of the following:
   - To use a range, select **Snapshot Range**. Complete the following steps:
     - Under Select Time Period, select a start time for the baseline by selecting **Period Start Time** and the snapshot icon below the Active Sessions chart that corresponds to the desired start time.
     - Select an end time for the baseline by selecting **Period End Time** and the snapshot icon below the Active Sessions chart that corresponds to the desired end time.
     - Optionally, to view older snapshots that are not displayed below the Active Sessions chart, expand **Change Chart Time Period**. Enter the desired start date in the **Chart Start Date** field and the desired end date in the **Chart End Date** field, and click **Go**.

     In this example, a snapshot range on February 5, 2009 from 3:10 p.m. to 3:30 p.m. is selected.



- To use a time range, select **Time Range**. Complete the following steps:
  - In the **Start Time** fields, select a start time for the baseline.
  - In the **End Time** fields, select an end time for the baseline.

  In the following example, a time range from 3:10 p.m. to 3:30 p.m. on February 6, 2009 is selected.

8. Click **Finish**.

   The AWR Baselines page reappears with the newly created baseline displayed.

Creating a Repeating Baseline

A repeating baseline is a baseline that repeats during a time interval over a specific period. For example, a repeating baseline may repeat every Monday from 8:00 a.m. to 10:00 a.m. from February 6, 2009 to February 6, 2010.

To create a repeating baseline:

1. From the Database Home page, click **Server**.

   The Server page appears.

2. Under Statistics Management, click **AWR Baselines**.

   The AWR Baselines page appears with a list of existing baselines displayed.

3. Click **Create**.

   The Create Baseline: Baseline Interval Type page appears.

4. Select **Repeating** and then click **Continue**.

   The Create Baseline: Repeating Baseline Template page appears.

5. In the **Baseline Name Prefix** field, enter a name prefix for the baseline.

6. Under Baseline Time Period, specify the time of the day that you want the baseline to begin collecting AWR data and the duration of the baseline collection.

7. Under Frequency, do one of the following:

   - Select **Daily** if you want the baseline to repeat on a daily basis.

- Select **Weekly** if you want the baseline to repeat on a weekly basis, and then select the day of the week on which the baseline will repeat.

8. Under Interval of Baseline Creation, complete the following steps:

   a. In the **Start Time** fields, select a date and time in the future when the data collection should begin.

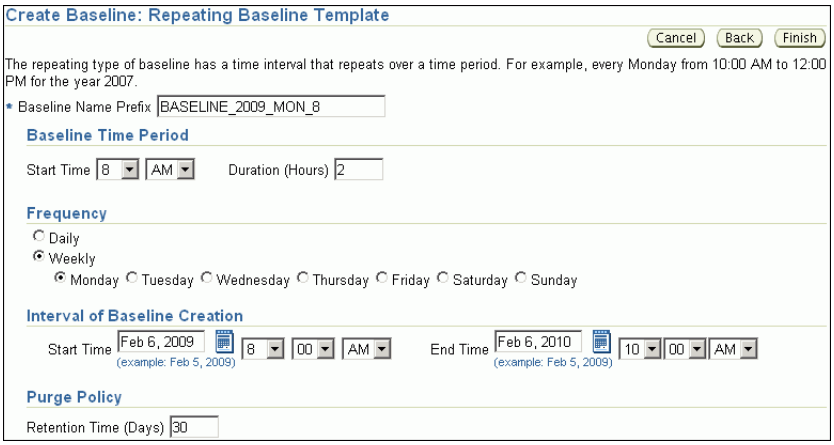
   b. In the **End Time** fields, select a date and time in the future when the data collection should end.

9. Under Purge Policy, enter the number of days to retain captured baselines.

10. Click **Finish**.

    A baseline template with the same name as the baseline name prefix will be created. A baseline template is a specification that enables Oracle Database to automatically generate a baseline for a future time period.

    This example creates a baseline that repeats weekly on Mondays from 8:00 a.m. to 10:00 a.m. from February 6, 2009 to February 6, 2010. Every captured baseline expires after 30 days.



Deleting a Baseline

To conserve storage space, you may want to periodically delete unused baselines stored in the database.

To delete a baseline:

1. From the Database Home page, click **Server**.

   The Server page appears.

2. Under Statistics Management, click **AWR Baselines**.

   The AWR Baselines page appears with a list of existing baselines displayed.

3. Select a baseline and click **Delete**.

   The Confirmation page appears.

4. Select whether to purge the underlying data associated with the baseline.

   The underlying data includes the individual snapshots preserved in the baseline and any statistics that are computed for the baseline. Do one of the following:

   - To delete the underlying data, select **Purge the underlying data associated with the baseline**.

- To preserve the underlying data, select **Do not purge the underlying data associated with the baseline**.

5. Click **Yes**.

   The AWR Baselines page reappears. A message informs you that the baseline was deleted successfully.

Computing Threshold Statistics for Baselines

Computing threshold statistics for baselines enables you to graphically display the computed statistics in the charts on the Performance page.

To compute threshold statistics for baselines:

1. From the Database Home page, click **Server**.

   The Server page appears.

2. Under Statistics Management, click **AWR Baselines**.

   The AWR Baselines page appears with a list of existing baselines displayed.

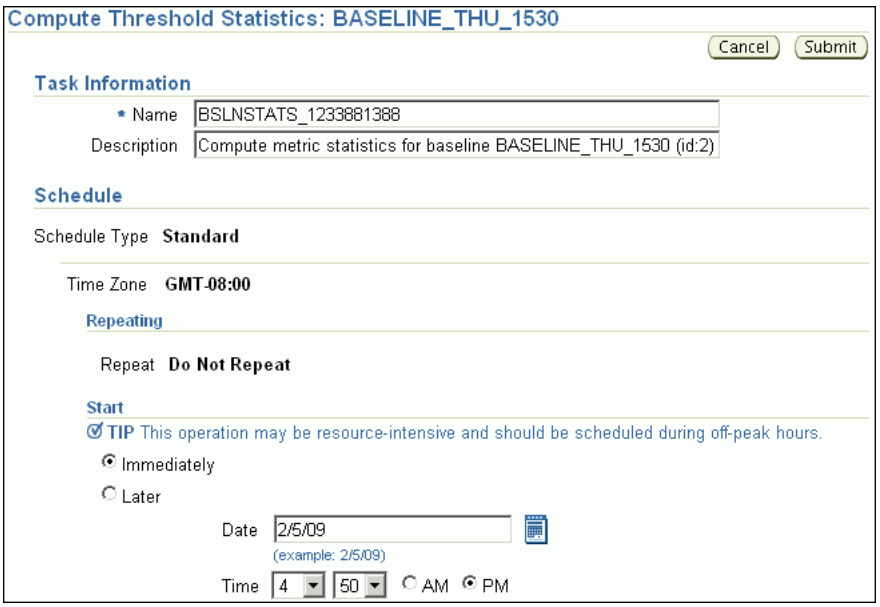
3. Select the baseline for which you want to compute statistics.

   Select a baseline that does not already have computed statistics. These baselines are identified by No in the Statistics Computed column.

4. From the Actions list, select **Schedule Statistics Computation**, and then click **Go**.

   The Compute Threshold Statistics page appears.

   This example computes statistics for the baseline `BASELINE_THU_1530`.



5. In the **Name** field, enter a name for the task.

   Alternatively, you can choose to use the system-generated name.

6. In the **Description** field, enter a description for the task.

   Alternatively, you can choose to use the system-generated description.

7. Under Start, do one of the following:

- Select **Immediately** to run the task immediately after it has been submitted.
- Select **Later** to run the task at a later time as specified using the Date and Time fields.

This computation is resource-intensive, so you may want to schedule it to run during off-peak hours.

8. Click **Submit**.

   The AWR Baselines page appears. A message informs you that statistics computation has been scheduled for the selected baseline.

   See Also:

   - "Customizing the Database Performance Page" on page 4-27 for information about displaying computed statistics on the Performance page
   - *Oracle Database 2 Day DBA* for information about thresholds and how to manage them

Setting Metric Thresholds for Baselines

As explained in "Setting Metric Thresholds for Performance Alerts" on page 5-1, a metric is the rate of change in a cumulative statistic. Alerts notify you when particular metric thresholds are crossed. When the metric thresholds are crossed, the system is in an undesirable state. You can edit the threshold settings for baseline metrics.

You can create the following types of baseline:

- Setting Metric Thresholds for the Default Moving Baseline
- Setting Metric Thresholds for Selected Baselines

Setting Metric Thresholds for the Default Moving Baseline

This section describes the easiest technique for setting the metric thresholds for the default moving baseline. You can choose a group of basic metric threshold settings based on common database workload profiles such as OLTP, data warehousing, and OLTP with nighttime batch jobs. After choosing a workload profile, you can expand or change the threshold values as needed.

To set metric thresholds for the default moving baseline:

1. On the Database Home page, under Related Links, click **Baseline Metric Thresholds**.

   The Threshold Configuration tab of the Baseline Metric Thresholds page appears.

2. Click **Quick Configuration**.

   The Quick Configuration: Baseline Metric Thresholds page appears.

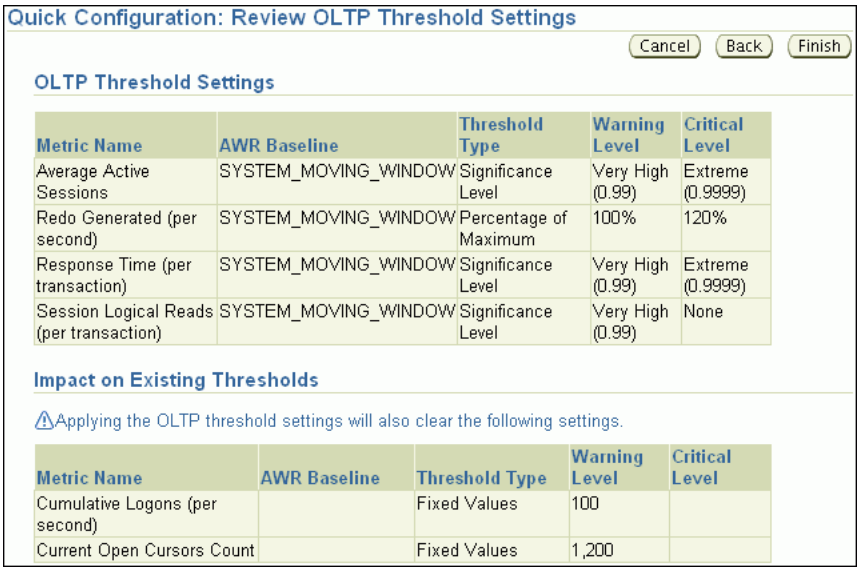
3. In **Workload Profile**, select one of the following options, depending on how you are using the database:

   - **Primarily OLTP (pure transaction processing 24 hours a day)**
   - **Primarily Data Warehousing (query and load intensive)**
   - **Alternating (OLTP during the daytime and batch during the nighttime)**

   In this example, select **Primarily OLTP**.

4. Click **Continue**.

The Quick Configuration: Review OLTP Threshold Settings page appears.



5. Review the metric threshold settings and then click **Finish**.

   You are returned to the Baseline Metric Thresholds page, with the Threshold Configuration tab selected. The metric threshold settings are displayed.

Setting Metric Thresholds for Selected Baselines

This section explains how to select a baseline and edit its thresholds. You can configure the type of threshold, for example, whether it is based on significance levels, percentage of maximum values, or fixed values. You can also configure the threshold levels that determine when the database generates critical alerts and warnings.

You can edit thresholds for the default moving baseline or a baseline that you created in the AWR Baselines page. You can select a baseline in the Edit Thresholds page after you have scheduled statistics computation from the AWR Baselines page and the statistics have finished computing on the static baseline.

To set a metric threshold for a selected moving baseline:

1. On the Database Home page, under Related Links, click **Baseline Metric Thresholds**.

   The Threshold Configuration tab of the Baseline Metric Thresholds page appears.

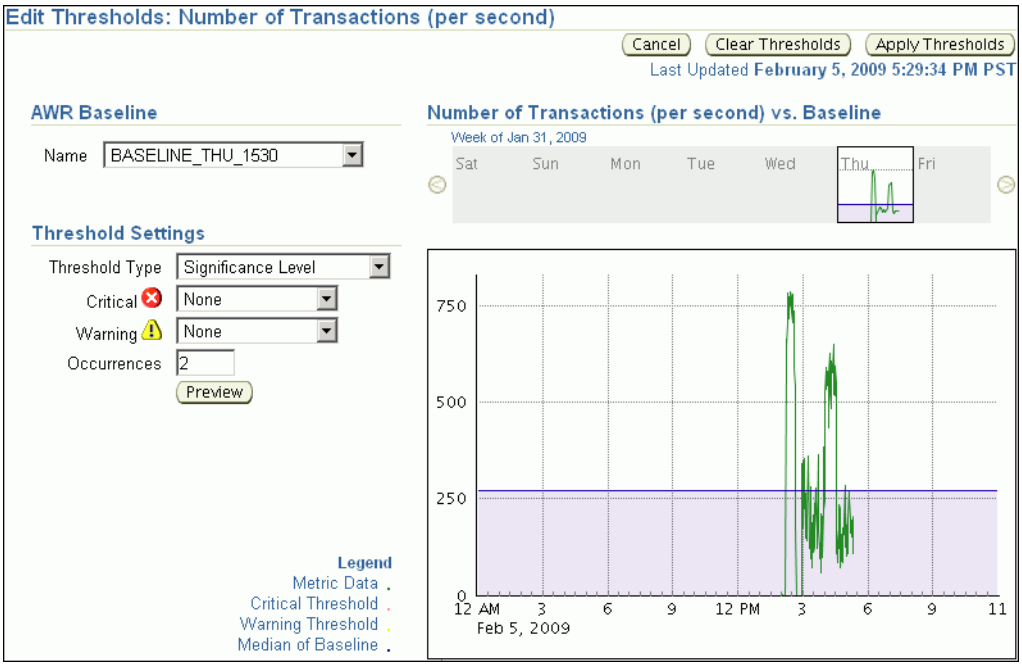
2. In the View list, select **Basic Metrics**.

   The Baseline Metric Thresholds page appears.

3. In the **Category/Name** column, click the link for the metric whose threshold you want to set or change.

   For example, click **Number of Transactions (per second)**.

   The Edit Thresholds: Number of Transactions (per second) appears.



The charts on this page provide simple and detailed views of metric activity for a 24-hour period. In the top simple chart, click a day to view the value of the metric plotted against a 24-hour period.

4. Under **AWR Baseline**, in the Name list, select either the default **SYSTEM_MOVING_WINDOW** or the name of a baseline created in the AWR Baselines page.

   A baseline appears in the AWR Baseline list after you have scheduled statistics computation from the AWR Baselines page and the statistics have finished computing on the static baseline.

   In this example, **BASELINE_THU_1530** is selected.

   The page refreshes to show the charts for the baseline that you selected.

5. In the Threshold Settings section, complete the following steps to change the settings:

   a. In **Threshold Type**, leave **Significance Level** selected.

   b. In **Critical**, select **Extreme**.

   c. In **Warning**, select **Very High**.

   d. In **Occurrences**, leave the current value.

6. Click **Apply Thresholds**.

   You are returned to the Baseline Metric Thresholds page. This page shows the altered metric threshold settings.

Running the AWR Compare Periods Reports

This section describes how to run the AWR Compare Periods reports using 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 occurs over time, you can 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. You must have a baseline that represents the system operating at an optimal level. If an existing baseline is unavailable, then compare database performance between two periods of time using two arbitrary pairs of snapshots, as described in "Comparing Two Pairs of Snapshots" on page 8-14.

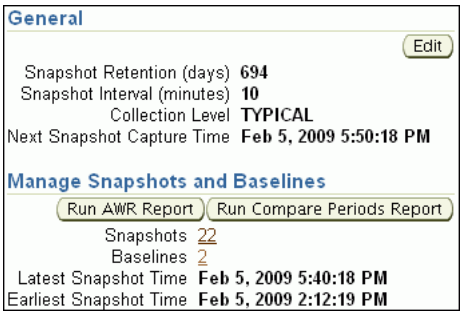
To compare a baseline to another baseline:

1. From the Database Home page, click **Server**.

   The Server subpage appears.

2. Under Statistics Management, click **Automatic Workload Repository**.

   The Automatic Workload Repository page appears.



3. Under Manage Snapshots and Baselines, click the link next to **Baselines**.

   The AWR Baselines page appears.

4. Complete the following steps:

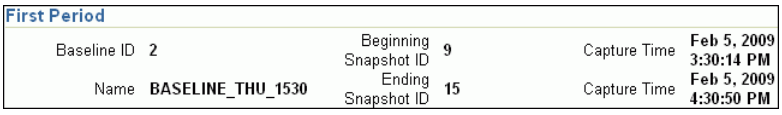
   a. Select the baseline to use for the report.

      At least one existing baseline must be available.

   b. 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 `BASELINE_THU_1530` is selected.



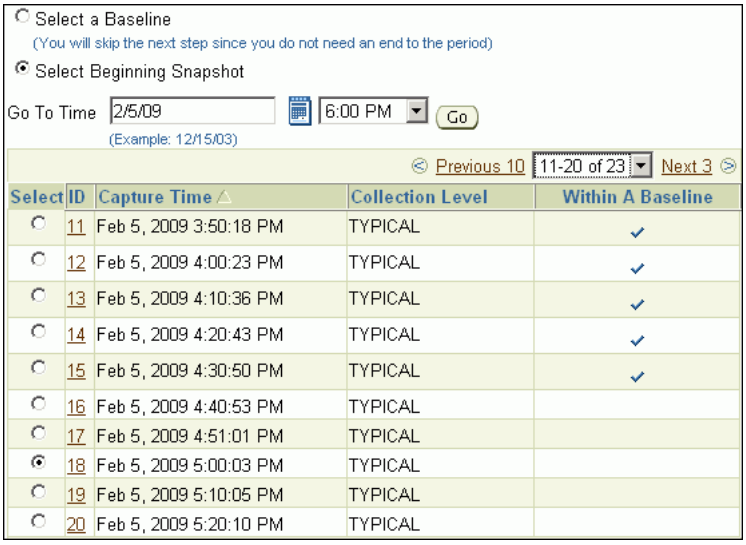
5. Compare the baseline selected in the first period to another baseline or a pair of snapshots. Do one of the following:

   - To compare to another baseline, select **Select a Baseline** and the baseline you want to use in the second period, and then click **Next**.

     The Compare Periods: Review page appears. Go to Step 7.

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

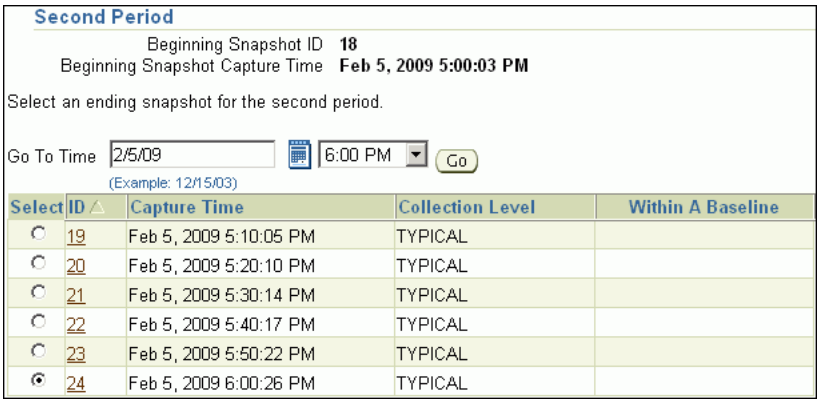
  This example selects snapshot 18, taken on February 5, 2009 at 5:00 p.m.



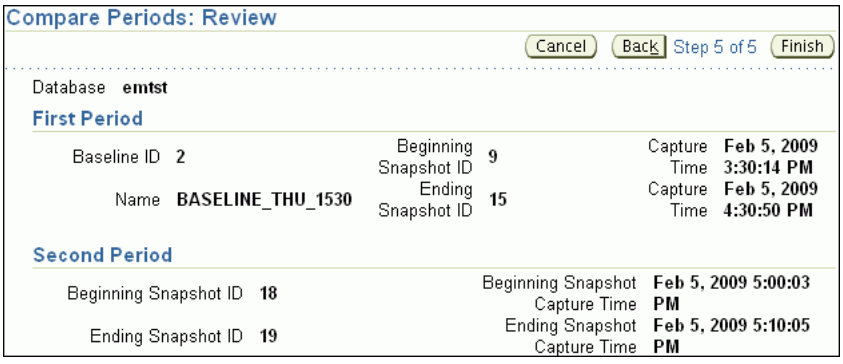
The Compare Periods: Second Period End appears. Proceed to the next step.

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

   In this example, snapshot 24, taken on Feb 5, 2009 at 6:00 p.m., is selected.



The Compare Periods: Review page appears.



7. Review the periods to be included in the report and then click **Finish**.

   The Compare Periods: Results page appears.

   Data from the selected periods appears under the General subpage. You can view data per second or per transaction by selecting an option from the View Data list.

   Note: If the time periods have different lengths, then the data is normalized over database time before calculating the difference so that periods of different lengths can be compared.

   In this example, almost every metric shows that more resources were consumed in the first period. The bar graphs indicate the proportions of the values in the two periods. The absence of bars indicates equivalent values. The report for this example shows significantly more database block changes per second and parse time in the first period than in the second.

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)		10,345.23	1,715.34	2.85	2.85
db block changes		75,874,412.00	5,251,519.00	20,867.55	8,723.45
execute count		6,125,168.00	488,636.00	1,684.59	811.69
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		7,820,328.00	630,508.00	2,150.81	1,047.36
parse count (total)		1,751,622.00	144,402.00	481.74	239.87
parse time cpu (seconds)		3.10	0.24	0.00	0.00
parse time elapsed (seconds)		22.50	5.54	0.01	0.01
physical reads		3,744.00	429.00	1.03	0.71
physical writes		725,739.00	68,520.00	199.60	113.82
redo size (KB)		7,894,446.08	549,390.19	2,171.19	912.61

8. Click **Report** to view the report.

   The Processing: View Report page appears while the report is being generated. After it completes, the report will appear. To change periods, click **Change Periods**. To save the report as an HTML file, click **Save to File**.

9. Optionally, do the following:

   - To change periods, click **Change Periods**.
   - To save the report as an HTML file, click **Save to File**.

   See Also:

   - "Creating a Baseline" on page 8-2
   - "Using the AWR Compare Periods Reports" on page 8-16

Comparing Current System Performance to a Baseline Period

You may have noticed a performance change on a production system and would like to know why, or you may have implemented a change to a production system and want to know the effect of the change, such as increased concurrency waits.

The Compare Period ADDM compares the performance of the database server in two time periods, and returns a report describing the performance changes and the root origin of the changes. The Advisor can analyze any Oracle RDBMS version 10.2.0.4 or greater monitored by Cloud Control. The following procedure explains how to initiate a report from the Compare Period ADDM.

1. From the Performance menu, select **AWR**, then **Compare Period ADDM**.
2. From the Compare Period Report page, specify the desired comparison and base periods:
   - **Comparison Period** — Generally represents an improperly functioning time period. However, you could also use the advisor to understand why performance has improved now when compared with an earlier time period.
   - **Base Period** — Represents a known (baseline or reference) period in which the database is functioning properly. You should select a base period in which the performance was acceptable, and the workload was as similar or identical as possible.
3. Click **Run** to display the Database Compare Period Report.
4. Examine the sections of the report to understand the performance change between the two periods and the cause of the change:
   - **Overview**

     This portion of the report shows SQL commonality, which is the comparability between the base and comparison periods based on the average resource consumption of the SQL statements common to both periods.

     A commonality value of 100% means that the workload "signature" in both time periods is identical. A commonality of 100% is expected for this use case, because the workload being replayed is the same (assuming that you are not using replay filters). A value of 0% means that the two time periods have no items in common for the specific workload dimension.

     Commonality is based on the type of input (that is, which SQL is executing) as well as the load of the executing SQL statements. Consequently, SQL statements running in only one time period, but not consuming significant time, do not affect commonality. Therefore, two workloads could have a commonality of 100% even if some SQL statements are running only in one of the two periods, provided that these SQL statements do not consume significant resources.
   - **Configuration**

     The information displayed shows base period and comparison period values for various parameters categorized by instance, host, and database.
   - **Findings**

     The findings can show performance improvements and identify the major performance differences caused by system changes. For negative outcomes, if you understand and remove the cause, the negative outcome can be eliminated.

The values shown for the Base Period and Comparison Period represent performance with regard to database time.

The Change Impact value represents a measurement of the scale of a change in performance from one time period to another. It is applicable to issues or items measured by the total database time they consumed in each time period. The absolute values are sorted in descending order.

If the value is positive, an improvement has occurred, and if the value is negative, a regression has occurred. For instance, a change impact of -200% means that period 2 is three times as slow as period 1.

You can run performance tuning tools, such as ADDM and the SQL Tuning Advisor, to fix issues in the comparison period to improve general system performance.

- **Resources**

  The information shown here provides a summary of the division of database time for both time periods, and shows the resource usage for CPU, memory, I/O, and interconnect (Oracle RAC only).

5. Based on your observations, decide how to proceed to resolve performance regressions, then implement your action plan.

Comparing Two Pairs of Snapshots

If an existing baseline is unavailable, then you can compare database performance by using two arbitrary pairs of snapshots. Use one pair taken when the database is performing optimally, and another pair when the database is performing poorly. At least four existing snapshots must be available.

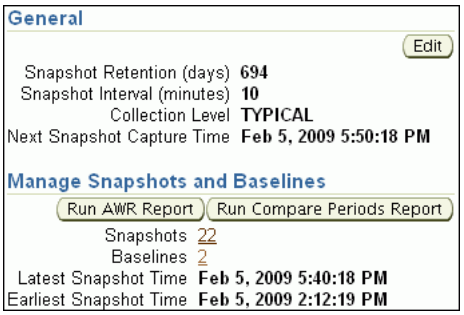
To compare performance using two pairs of snapshots:

1. From the Database Home page, click **Server**.

   The Server page appears.

2. Under Statistics Management, click **Automatic Workload Repository**.

   The Automatic Workload Repository page appears.



3. Under Manage Snapshots and Baselines, click the link next to **Snapshots**.

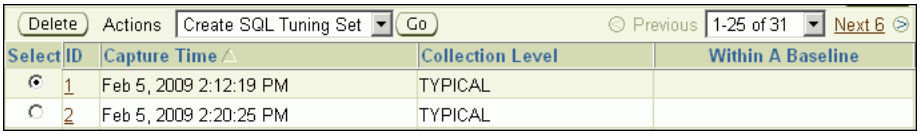
   The Snapshots page appears.

4. From the Go To Time list, select the time for the starting snapshot and then click **Go**.

   This action filters the snapshots and displays only the snapshot taken at the start of the comparison period. The time in this example is at 2 p.m. on February 5.

5. Under Select Beginning Snapshot, select the starting point for the first snapshot period to be included in the report.

   In this example, snapshot 1 is selected.

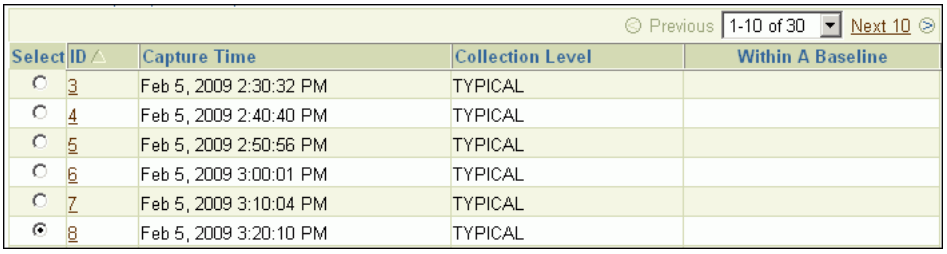


6. From the Actions list, select **Compare Periods** and click **Go**.

   The Compare Periods: First Period End page appears.

7. Select the ending point for the first snapshot period to be included in the report and click **Next**.

   In this example, snapshot 8, taken on February 5 at 3:20:10 p.m., is selected.



   The Compare Periods: Second Period Start page appears.

8. Select the starting point for the second snapshot period to be included in the report and click **Next**. This snapshot is the third of four.

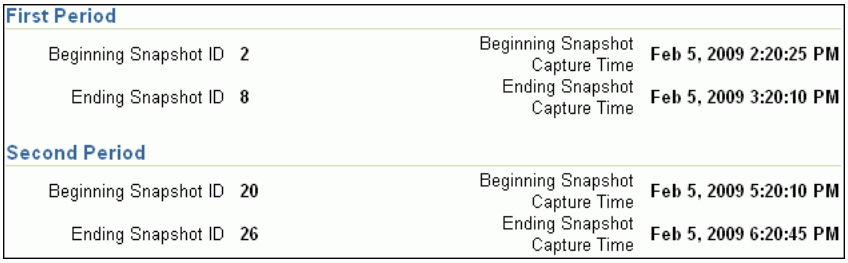
   In this example, snapshot 20, taken on February 5 at 5:20 p.m., is selected.

   The Compare Periods: Second Period End page appears.

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

   In this example, snapshot 26, taken on February 5 at 6:20 p.m., is selected.

   The Compare Periods: Review page appears.

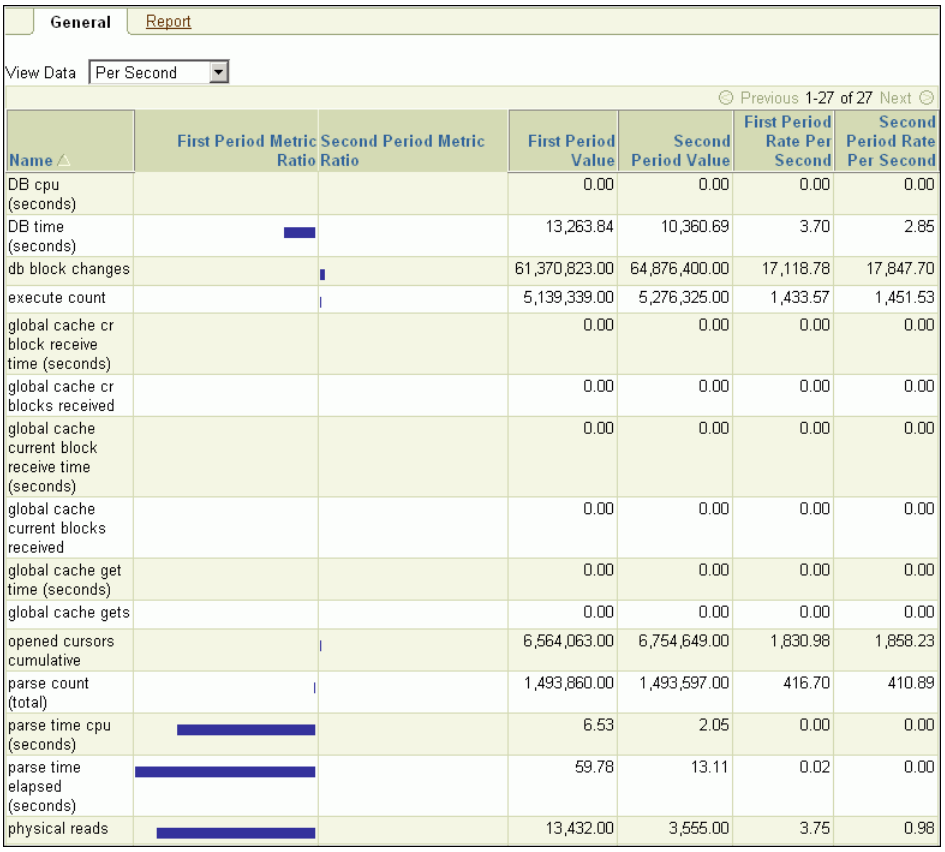


10. Review the selected periods to be included in the report and then click **Finish**.

    The Compare Periods: Results page appears.

Data from the selected periods appears under the General subpage. You can view data per second or per transaction by selecting an option from the View Data list.

In the following example, the first period shows more database activity, especially in parse time and physical reads, than the second period.



General | Report

View Data: Per Second

Previous 1-27 of 27 Next

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)			13,263.84	10,360.69	3.70	2.85
db block changes			61,370,823.00	64,876,400.00	17,118.78	17,847.70
execute count			5,139,339.00	5,276,325.00	1,433.57	1,451.53
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			6,564,063.00	6,754,649.00	1,830.98	1,858.23
parse count (total)			1,493,860.00	1,493,597.00	416.70	410.89
parse time cpu (seconds)			6.53	2.05	0.00	0.00
parse time elapsed (seconds)			59.78	13.11	0.02	0.00
physical reads			13,432.00	3,555.00	3.75	0.98

11. 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.

12. Optionally, do the following:

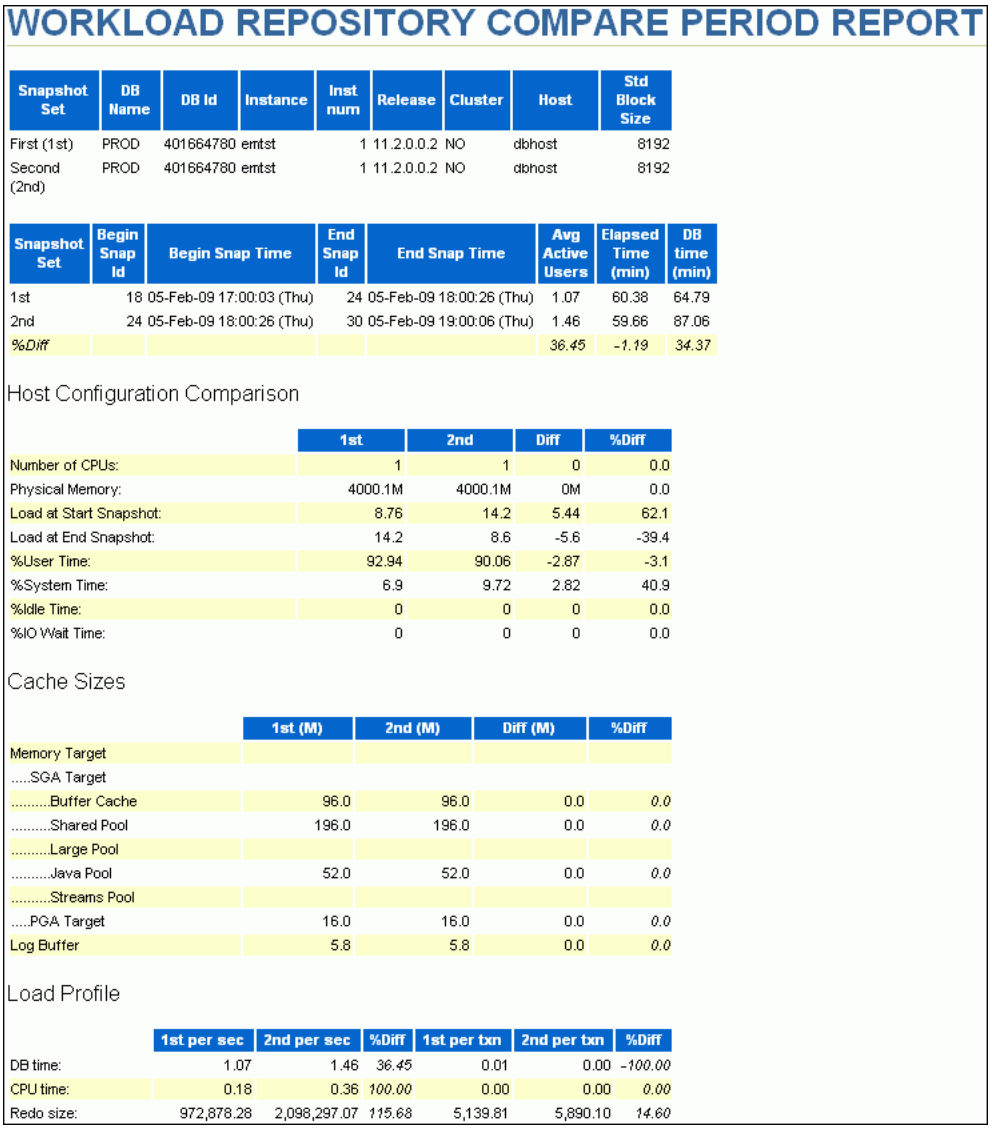
    - To change periods, click **Change Periods**.
    - To save the report as an HTML file, click **Save to File**.

Using the AWR Compare Periods Reports

After an AWR Compare Periods report is generated for the time periods you want to compare, you can use it to analyze performance degradation. To learn how to create the report, see "Running the AWR Compare Periods Reports" on page 8-9.

Figure 8–1 shows a portion 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	Std Block Size
First (1st)	PROD	401664780	emtst	1	11.2.0.0.2	NO	dbhost	8192
Second (2nd)	PROD	401664780	emtst	1	11.2.0.0.2	NO	dbhost	8192

Snapshot Set	Begin Snap Id	Begin Snap Time	End Snap Id	End Snap Time	Avg Active Users	Elapsed Time (min)	DB time (min)
1st	18	05-Feb-09 17:00:03 (Thu)	24	05-Feb-09 18:00:26 (Thu)	1.07	60.38	64.79
2nd	24	05-Feb-09 18:00:26 (Thu)	30	05-Feb-09 19:00:06 (Thu)	1.46	59.66	87.06
%Diff					*36.45*	*-1.19*	*34.37*

Host Configuration Comparison

	1st	2nd	Diff	%Diff
Number of CPUs:	1	1	0	0.0
Physical Memory:	4000.1M	4000.1M	0M	0.0
Load at Start Snapshot:	8.76	14.2	5.44	62.1
Load at End Snapshot:	14.2	8.6	-5.6	-39.4
%User Time:	92.94	90.06	-2.87	-3.1
%System Time:	6.9	9.72	2.82	40.9
%Idle Time:	0	0	0	0.0
%IO Wait Time:	0	0	0	0.0

Cache Sizes

	1st (M)	2nd (M)	Diff (M)	%Diff
Memory Target				
.....SGA Target				
.....Buffer Cache	96.0	96.0	0.0	*0.0*
.....Shared Pool	196.0	196.0	0.0	*0.0*
.....Large Pool				
.....Java Pool	52.0	52.0	0.0	*0.0*
.....Streams Pool				
.....PGA Target	16.0	16.0	0.0	*0.0*
Log Buffer	5.8	5.8	0.0	*0.0*

Load Profile

	1st per sec	2nd per sec	%Diff	1st per txn	2nd per txn	%Diff
DB time:	1.07	1.46	*36.45*	0.01	0.00	*-100.00*
CPU time:	0.18	0.36	*100.00*	0.00	0.00	*0.00*
Redo size:	972,878.28	2,098,297.07	*115.68*	5,139.81	5,890.10	*14.60*

The AWR Compare Periods report is divided into the following sections:

- Summary of the AWR Compare Periods Report
- Details of the AWR Compare Periods Report
- Supplemental Information in the AWR Compare Periods Report

Summary of the AWR Compare Periods Report

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
- Load Profile
- Top Timed Events
- Host Configuration Comparison

- System Configuration Comparison

Snapshot Sets

The Snapshot Sets section displays information about the snapshot sets used for this report, such as instance, host, and snapshot information.

In the example shown in Figure 8–1 on page 8-17, the first snapshot period corresponds to the time when performance was stable on February 5 from 5:00 p.m. to 6:00 p.m. The second snapshot period corresponds to the time when performance degradation occurred on the same day from 6:00 p.m. to 7:00 p.m.

Load Profile

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

Load Profile

	1st per sec	2nd per sec	%Diff	1st per txn	2nd per txn	%Diff
DB time:	1.07	1.46	36.45	0.01	0.00	-100.00
CPU time:	0.18	0.36	100.00	0.00	0.00	0.00
Redo size:	972,878.28	2,098,297.07	115.68	5,139.81	5,890.10	14.60
Logical reads:	11,441.37	23,063.30	101.58	60.45	64.74	7.10
Block changes:	9,080.75	19,386.33	113.49	47.97	54.42	13.45
Physical reads:	0.96	0.97	1.04	0.01	0.00	-100.00
Physical writes:	110.86	189.22	70.68	0.59	0.53	-10.17
User calls:	3.38	2.65	-21.60	0.02	0.01	-50.00
Parses:	239.04	442.48	85.11	1.26	1.24	-1.59
Hard parses:	0.03	0.10	233.33	0.00	0.00	0.00
WA MB processed:	8,263,422.18	69,867,920.23	745.51	43,656.43	196,125.18	745.51
Logons:	0.05	0.05	0.00	0.00	0.00	0.00
Executes:	816.88	1,565.44	91.64	4.32	4.39	1.62
Transactions:	189.28	356.24	88.21			

	1st	2nd	Diff
% Blocks changed per Read:	79.37	84.06	4.69
Recursive Call %:	99.80	99.92	0.12
Rollback per transaction %:	50.73	53.48	2.75
Rows per Sort:	0.18	0.08	-0.10
Avg DB time per Call (sec):	0.32	0.55	0.23

In this example, the DB time per second was 36% higher in the second period. CPU time per second was 100% higher.

Top Timed Events

The Top 5 Timed Events section is one of the most useful sections in the report. This section displays the five timed events or operations that consumed the highest percentage of total DB time in each of the snapshot sets.

1st						2nd					
Event	Wait Class	Waits	Time(s)	Avg Time(ms)	%DB time	Event	Wait Class	Waits	Time(s)	Avg Time(ms)	%DB time
CPU time			635.13		16.34	CPU time			1,277.56		24.46
log file parallel write	System I/O	591,729	197.66	0.33	5.08	enq: TM - contention	Application	444,234	1,046.23	2.36	20.03
LGWR wait for redo copy	Other	4,414	156.13	35.37	4.02	log file parallel write	System I/O	748,172	685.71	0.92	13.13
enq: TM - contention	Application	54,824	133.25	2.43	3.43	LGWR wait for redo copy	Other	25,719	300.55	11.69	5.75
os thread startup	Concurrency	86	29.50	343.08	0.76	buffer busy waits	Concurrency	10,715	144.62	13.50	2.77
-buffer busy waits	Concurrency	1,297	14.21	10.95	0.37	-os thread startup	Concurrency	84	58.20	692.80	1.11

In this example, CPU time is about twice as much in the second period than in the first. The number of waits for TM locks in the second period is about eight times the number in the first.

Host Configuration Comparison

The Host Configuration Comparison section compares the host configurations used in the two snapshot sets. For example, the report compares physical memory and number of CPUs. Differences in the configurations are quantified as percentages in the %Diff column.

System Configuration Comparison

The System Configuration Comparison section compares the database configurations used in the two snapshot sets. For example, the report compares the SGA and log buffer size. Differences in the configurations are quantified as percentages in the %Diff column.

Details of the AWR Compare Periods Report

The details section follows the summary of the AWR Compare Periods report, and provides statistics about the snapshot sets and loads used in the report. For example, the section includes statistics for database time, wait events, SQL execution time, and instance activity.

Supplemental Information in the AWR Compare Periods Report

The supplemental information is at the end of the AWR Compare Periods report, and provides additional information about initialization parameters and SQL statements. The init.ora Parameters section lists all the initialization parameter values for the first snapshot set. The Complete List of SQL Text section lists each statement by SQL ID and shows the text of the SQL statement.

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 may consume a disproportionate amount of system resources. These SQL statements often greatly affect database performance and must be tuned to optimize their performance and resource consumption. Even when a database is properly tuned, inefficient SQL can significantly degrade performance.

Identifying high-load SQL statements is an important SQL tuning activity that you must perform regularly. Automatic Database Diagnostic Monitor (ADDM) automates this task by proactively identifying potential high-load SQL statements. Additionally, you can use Oracle Enterprise Manager (Enterprise Manager) to identify high-load SQL statements that require further investigation. After you have identified the high-load SQL statements, you can tune them with SQL Tuning Advisor and SQL Access Advisor.

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

- Identification of High-Load SQL Statements Using ADDM Findings
- Identifying High-Load SQL Statements Using Top SQL

Identification of High-Load SQL Statements Using ADDM Findings

By default, ADDM runs proactively once every hour. 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 the system finds performance problems, it displays them 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 gives recommendations, such as running SQL Tuning Advisor on the SQL statement. You can begin tuning SQL statements as described in Chapter 10, "Tuning SQL Statements".

See Also:

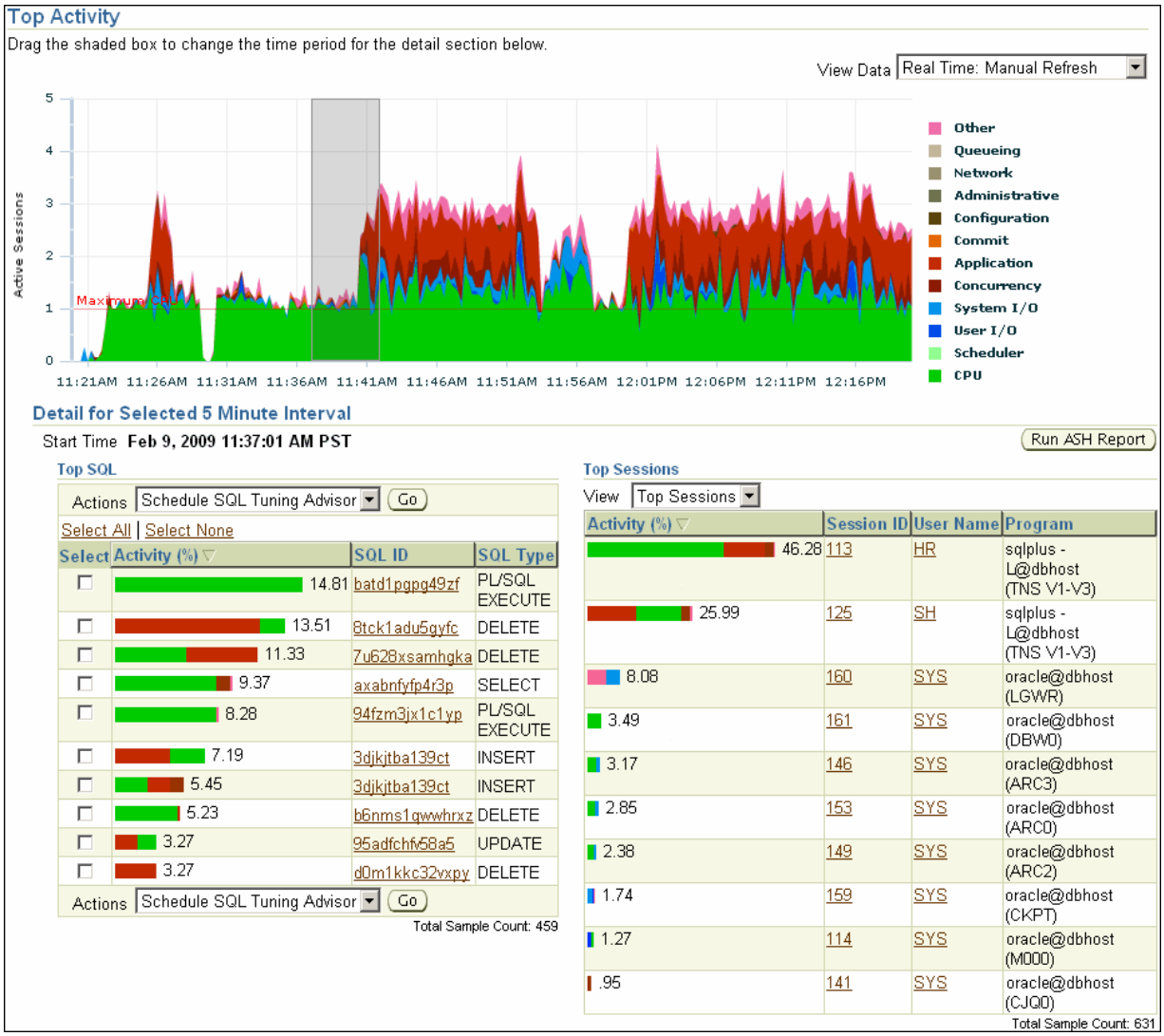
- "Overview of Automatic Database Diagnostic Monitor" on page 3-1
- "Interpretation of Automatic Database Diagnostic Monitor Findings" on page 3-8
- "Implementing Automatic Database Diagnostic Monitor Recommendations" on page 3-9

Identifying High-Load SQL Statements Using Top SQL

ADDM automatically identifies high-load SQL statements that may be causing systemwide 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 enables you to identify high-load SQL statements for any 5-minute interval.

Figure 9–1 shows an example of the Top Activity page.

Figure 9–1 Top Activity Page



To access the Top Activity page:

1. From the Database Home page, click **Performance**.

   The Performance page appears.

2. Under Additional Monitoring Links, click **Top Activity**.

   The Top Activity page appears.

This page shows a 1-hour time line 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.

3. To move the 5-minute interval, drag the shaded box to the desired time.

   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.

4. To monitor SQL statements for a longer duration than one hour, select **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:

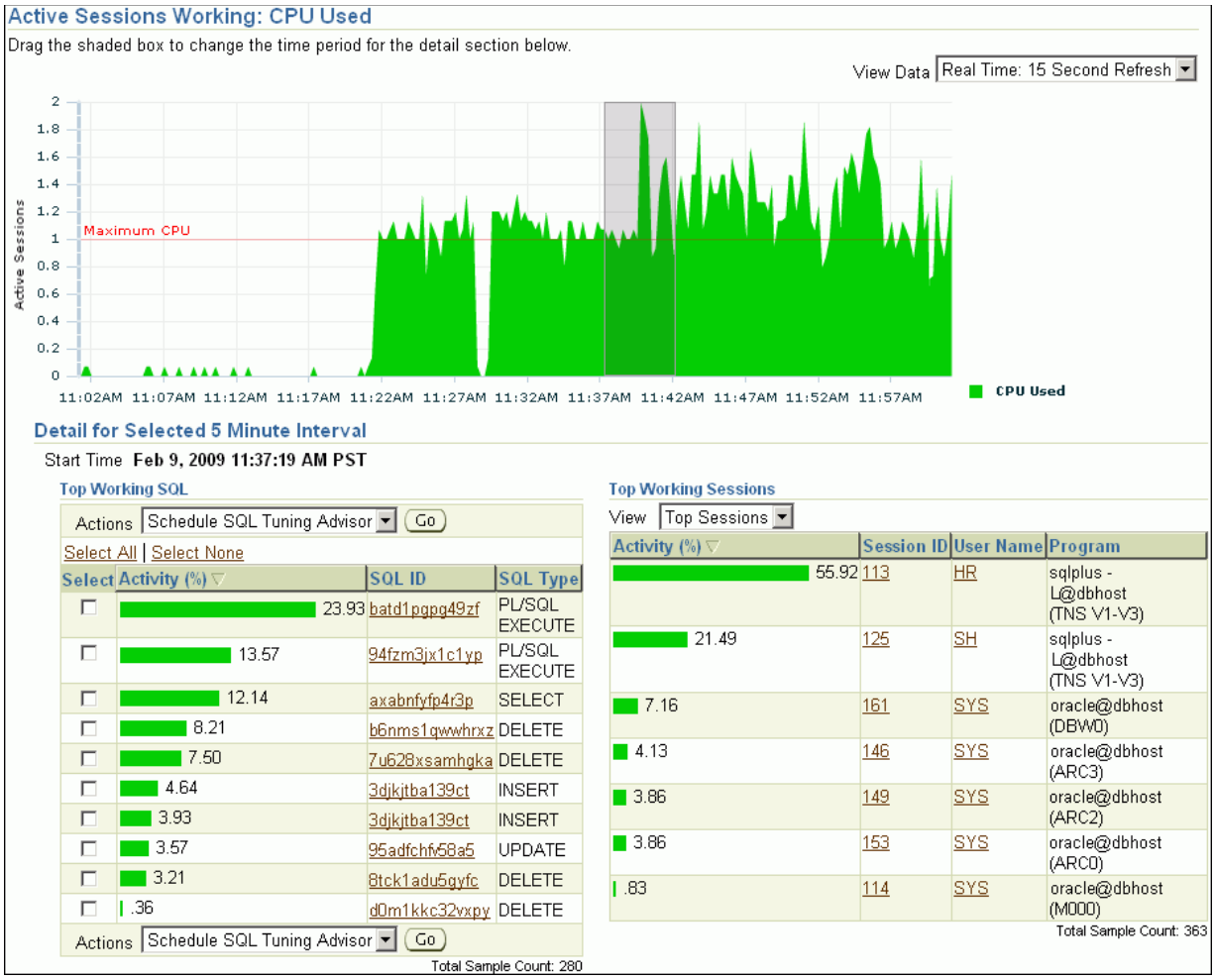
Viewing SQL Statements by Wait Class

The SQL statements that appear in the Top SQL section of the Top Activity page are categorized into various wait classes, based on their corresponding class as described in the legend on the Top Activity chart.

To view the SQL statements for a particular wait class, click the block of color on the chart 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 are displayed in the Top Working SQL section.

Figure 9–2 Viewing SQL Statements 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 of the Top Activity page 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 on.

In the example shown in Figure 9–2, "Viewing SQL Statements by Wait Class", the `SELECT` statement with the SQL ID `axabnfyfp4r3p` is consuming 12.14% of database activity and should be investigated.

To view details of SQL statements:

1. From the Database Home page, click **Performance**.

   The Performance page appears.

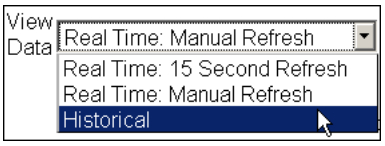
2. Under Additional Monitoring Links, click **Top Activity**.

The Top Activity page appears.

3. In the Top SQL section, click the **SQL ID** link of the SQL statement.

   The SQL Details page for the selected SQL statement appears.

4. To view SQL details for a longer period, select **Historical** from the View Data list.



   You can now view SQL details in the past, up to the duration defined by the AWR retention period.

5. In the Text section, review the SQL text for the SQL statement.

   The Text section contains the SQL text for the selected SQL statement. Note that if only part of the SQL statement is displayed, then a plus sign (+) icon appears next to the Text heading. To view the SQL text for the entire SQL statement, click the plus sign (+) icon.

   In this example, the text of SQL statement `batd1pgpg49zf` is as follows:

```
SELECT E.LAST_NAME, J.JOB_TITLE, D.DEPARTMENT_NAME
FROM   HR.EMPLOYEES E, HR.DEPARTMENTS D, HR.JOBS J
WHERE  E.DEPARTMENT_ID = D.DEPARTMENT_ID
AND    E.JOB_ID = J.JOB_ID
AND    E.LAST_NAME LIKE 'A%'
```

6. In the Plan Hash Values list in the Details section, do one of the following:
   - If the SQL statement has multiple plans, then select **All** to show SQL details for all plans.
   - Select a particular plan to display SQL details for this plan only.

7. View the subpages available on the SQL Details page to display additional information about the SQL statement, as described in the following sections:
   - Viewing SQL Statistics
   - Viewing Session Activity
   - Viewing the SQL Execution Plan
   - Viewing the Plan Control
   - Viewing the Tuning History

8. If the SQL statement is a high-load SQL statement, then tune it as described in Chapter 10, "Tuning SQL Statements".

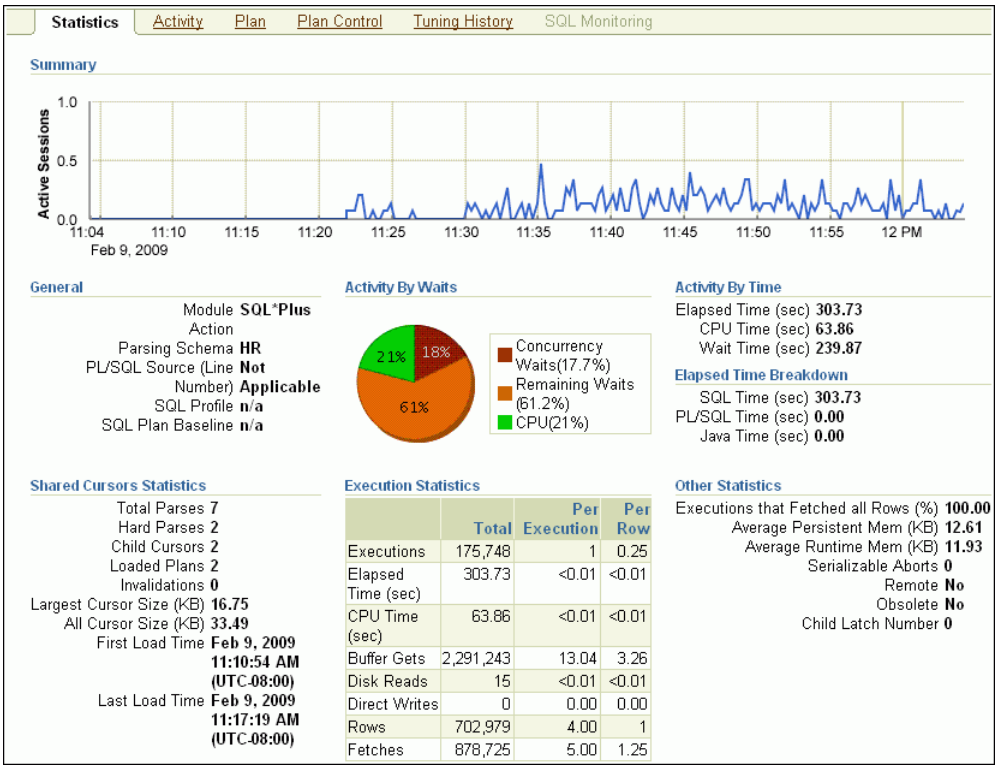
Viewing SQL Statistics

The Statistics subpage of the SQL Details page displays statistical information about the SQL statement.

To view statistics for the SQL statement:

1. On the SQL Details page, under Details, click **Statistics**.

   The SQL Details page appears, showing the Statistics subpage.



2. View the statistics for the SQL statement, as described in the following sections:
   - SQL Statistics Summary
   - General SQL Statistics
   - Activity by Wait Statistics and Activity by Time Statistics
   - Elapsed Time Breakdown Statistics
   - Shared Cursors Statistics and Execution Statistics
   - Other SQL Statistics

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

In the 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, then 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 the 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 technique enables you to track the response time of the SQL statement using different dimensions. You can determine whether the performance of the SQL statement has degraded based on the dimension selected.

To view statistics of the SQL statement for a particular time interval, click the snapshot icon below the chart. You can also use the arrows to scroll the chart to locate a desired snapshot.

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 account used to execute the SQL statement
- PL/SQL source, or the code line if the SQL statement is part of a PL/SQL program

Activity by Wait Statistics and Activity by Time Statistics The Activity by Wait and Activity by Time sections enable you to identify how 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.

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 if the total elapsed time is inflated due to the amount of time the originating program or application spent with the PL/SQL or Java engine. If the PL/SQL time or Java time makes up a significant portion of the elapsed time, then 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.

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.

Other SQL Statistics The Other Statistics section provides additional information about the SQL statement, such as average persistent and run-time memory.

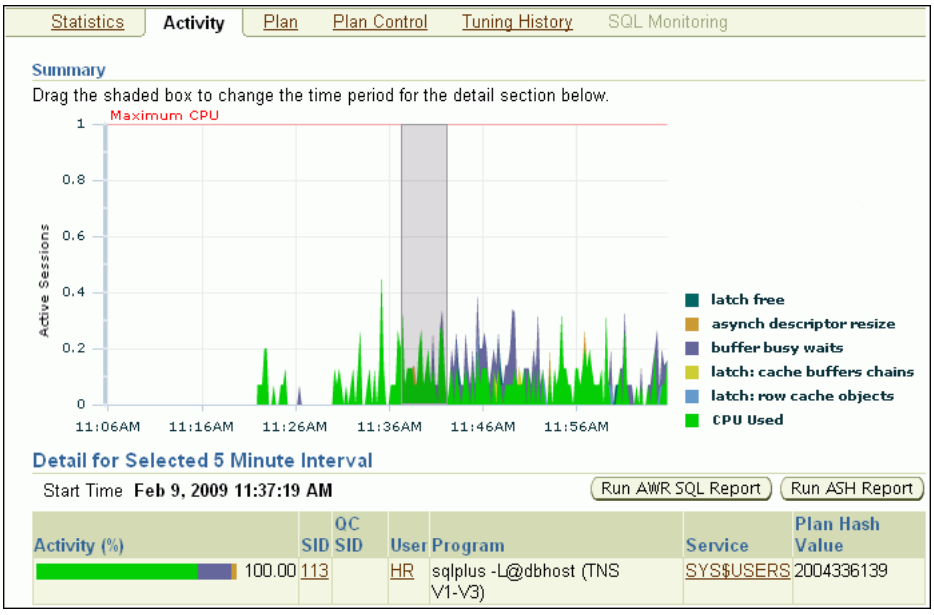
Viewing Session Activity

The Activity subpage contains a graphical representation of the session activity.

To view session activity for the SQL statement:

1. On the SQL Details page, under Details, click **Activity**.

   The SQL Details page appears, showing the Activity subpage.



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.

2. Optionally, 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 multicolored bar in the Activity % column depicts how the database time is divided for each session while executing the SQL statement.

3. Optionally, click the link in the **SID** column of the session you want to view to display the Session Details page.

See Also:

- "Monitoring Top Sessions" on page 4-6 for information about monitoring session activity and details

Viewing the SQL Execution Plan

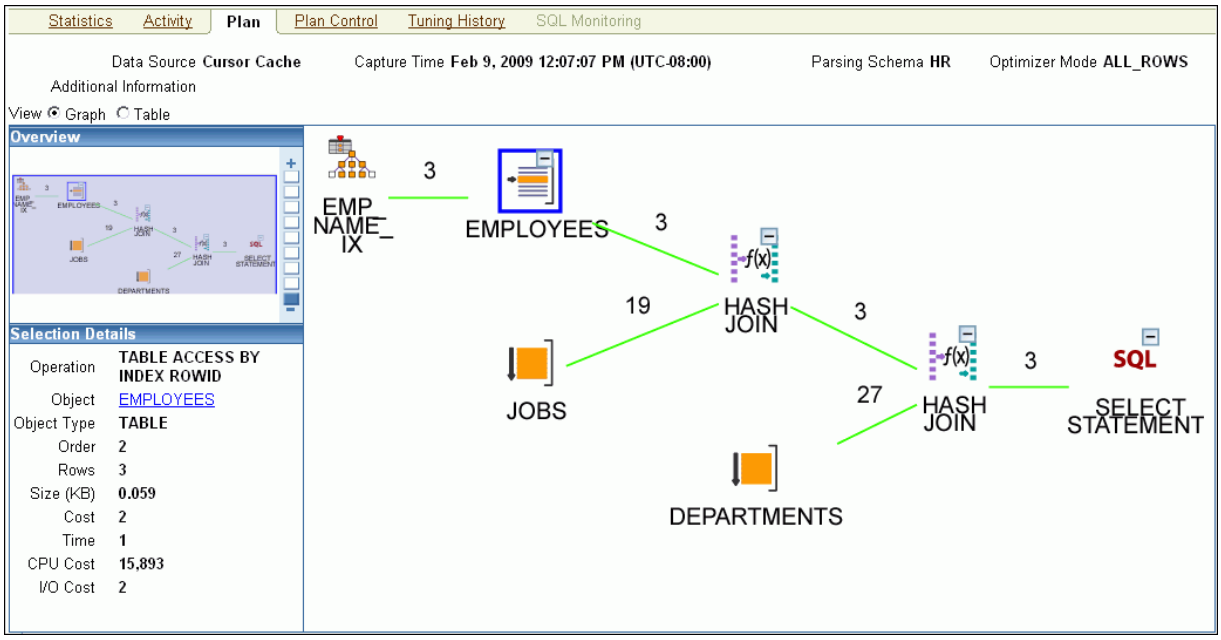
The **execution plan** for a SQL statement is the sequence of operations Oracle Database performs to run the statement. The Plan subpage displays the execution plan for the SQL statement in a graph view and a table view.

To view the execution plan for the SQL statement:

1. On the SQL Details page, under Details, click **Plan**.

   The SQL Details page appears, showing the Plan subpage.

2. Click **Graph** to view the SQL execution in a graph view.



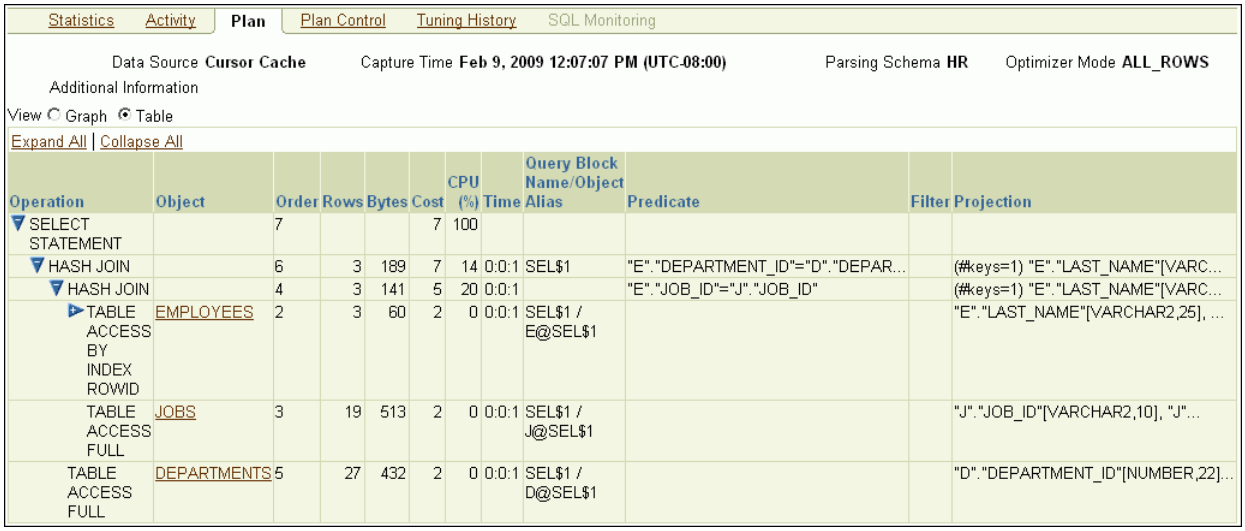
3. Optionally, select an operation in the graph to display details about the operations shown in the execution plan.

   The Selection Details section refreshes to show details about the selected operations.

4. If the selected operation is on a particular database object (such as a table), then click the **Object** link to view further details about the database object.

5. To view the SQL execution in a table view, click **Table**.

   The Plan subpage refreshes to show the explain plan in a table.



Operation	Object	Order	Rows	Bytes	Cost	CPU (%)	Time	Query Block Name/Object Alias	Predicate	Filter	Projection
SELECT STATEMENT		7			7	100					
HASH JOIN		6	3	189	7	14	0:0:1	SEL$1	"E"."DEPARTMENT_ID"="D"."DEPAR...		(#keys=1) "E"."LAST_NAME"[VARC...
HASH JOIN		4	3	141	5	20	0:0:1		"E"."JOB_ID"="J"."JOB_ID"		(#keys=1) "E"."LAST_NAME"[VARC...
TABLE ACCESS BY INDEX ROWID	EMPLOYEES	2	3	60	2	0	0:0:1	SEL$1 / E@SEL$1			"E"."LAST_NAME"[VARCHAR2,25], ...
TABLE ACCESS FULL	JOBS	3	19	513	2	0	0:0:1	SEL$1 / J@SEL$1			"J"."JOB_ID"[VARCHAR2,10], "J"...
TABLE ACCESS FULL	DEPARTMENTS	5	27	432	2	0	0:0:1	SEL$1 / D@SEL$1			"D"."DEPARTMENT_ID"[NUMBER,22]...

Query rewrite is an optimization technique that transforms a user request written in terms of master tables into a semantically equivalent request that includes materialized views. The database compares the cost for the query, with and without query rewrite, and selects the least costly option. If a rewrite is necessary, then query rewrite and its cost benefit are shown in the Explain Rewrite section.

See Also:

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

Viewing the Plan Control

The Plan Control subpage contains information about the following items:

- SQL profiles

  A SQL profile contains additional statistics for the SQL statement. The optimizer uses these statistics to generate a better execution plan for the statement.

- SQL patches

  A SQL patch is automatically generated to work around an error or performance problem for a single SQL statement.

- SQL plan baselines

  A SQL plan baseline is an execution plan proven to have acceptable performance for a given SQL statement.

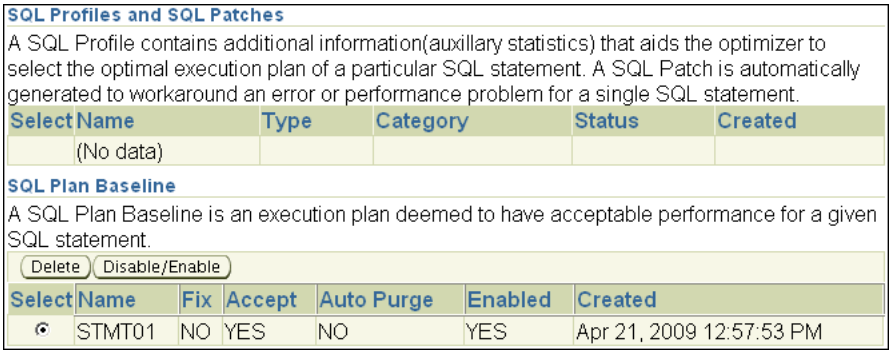
To view plan control information:

1. On the SQL Details page, under Details, click **Plan Control**.

   The SQL Details page appears, showing the Plan Control subpage.

2. Review the plan-related information.

   In the following example, the optimizer used a SQL plan baseline named `STMT01` for the SQL statement.



SQL Profiles and SQL Patches

A SQL Profile contains additional information(auxillary statistics) that aids the optimizer to select the optimal execution plan of a particular SQL statement. A SQL Patch is automatically generated to workaround an error or performance problem for a single SQL statement.

Select	Name	Type	Category	Status	Created
	(No data)				

SQL Plan Baseline

A SQL Plan Baseline is an execution plan deemed to have acceptable performance for a given SQL statement.

Delete | Disable/Enable

Select	Name	Fix	Accept	Auto Purge	Enabled	Created
◉	STMT01	NO	YES	NO	YES	Apr 21, 2009 12:57:53 PM

See Also:

- Chapter 10, "Tuning SQL Statements" for information about SQL Tuning Advisor and SQL profiles
- "Managing SQL Profiles" on page 10-16
- Chapter 11, "Optimizing Data Access Paths" for information about SQL Access Advisor

Viewing the Tuning History

The SQL Tuning History section displays a history of SQL Tuning Advisor and SQL Access Advisor tasks.

To view the SQL tuning history:

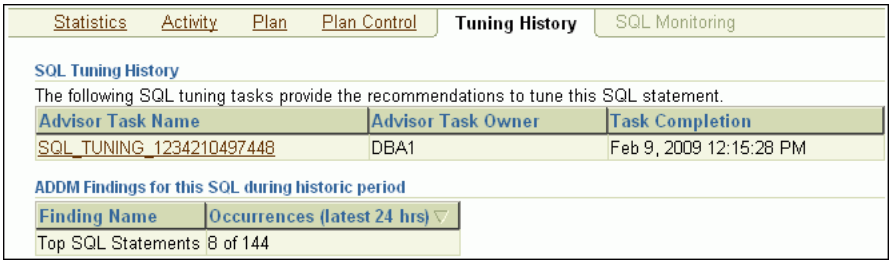
1. On the SQL Details page, under Details, click **Tuning History**.

The SQL Details page appears, showing the Tuning History subpage.

2. Review the information about the tuning history.

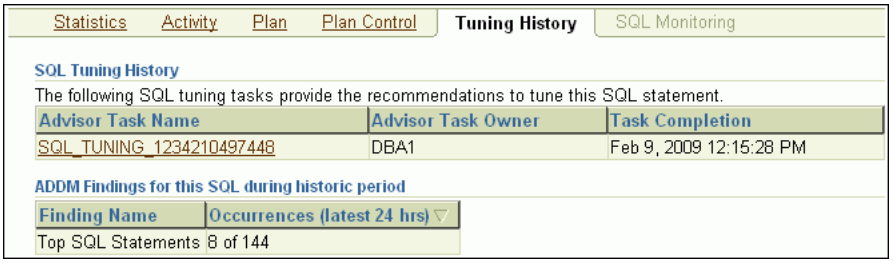
   The ADDM Findings for this SQL During Historic Period section displays the number of occurrences of ADDM findings that are associated with the SQL statement.

   The following example shows that SQL tuning task was performed by user `DBA1` on February 9, 2009.



The SQL Tuning History section displays a history of SQL Tuning Advisor or SQL Access Advisor tasks.

The ADDM Findings for this SQL During Historic Period section displays the number of occurrences of ADDM findings that are associated with the SQL statement.



See Also:

- Chapter 10, "Tuning SQL Statements" for information about SQL Tuning Advisor and SQL profiles
- "Managing SQL Profiles" on page 10-16
- Chapter 11, "Optimizing Data Access Paths" for information about SQL Access Advisor

10

Tuning SQL Statements

A SQL statement expresses the data you want Oracle Database to retrieve. For example, a SQL statement can retrieve the names of employees in a department. When Oracle Database executes the SQL statement, the **query optimizer** (also called the **optimizer**) first determines the best and most efficient way to retrieve the results.

The optimizer determines whether it is more efficient to read all data in the table, called a full table scan, or use an index. It compares the cost of all possible approaches and chooses the approach with the least cost. The access method for physically executing a SQL statement 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.

The query optimizer can also help you tune SQL statements. By using SQL Tuning Advisor and SQL Access Advisor, you can run the query optimizer in advisory mode to examine a SQL statement or set of statements and determine how to improve their efficiency. SQL Tuning Advisor and SQL Access Advisor can make various recommendations, such as creating SQL profiles, restructuring SQL statements, creating additional indexes or materialized views, and refreshing optimizer statistics. Additionally, Oracle Enterprise Manager (Enterprise Manager) enables you to accept and implement many of these recommendations easily.

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

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

- Tuning SQL Statements Using SQL Tuning Advisor
- Managing SQL Tuning Sets
- Managing SQL Profiles
- Managing SQL Execution Plans

See Also:

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

Tuning SQL Statements Using SQL Tuning Advisor

You can use SQL Tuning Advisor to tune one or more SQL statements. When tuning multiple statements, SQL Tuning Advisor does not recognize interdependencies between the SQL statements. Instead, SQL Tuning Advisor provides a convenient way to obtain tuning advice for a large number of SQL statements.

Oracle Database can generate SQL tuning reports automatically. Automatic SQL Tuning runs during system **maintenance windows** as an automated maintenance task, searching for ways to improve the execution plans of high-load SQL statements. A maintenance window is a contiguous time interval during which automated maintenance tasks are run.

Tuning SQL Manually Using SQL Tuning Advisor

As described in Chapter 9, "Identifying High-Load SQL Statements", Automatic Database Diagnostic Monitor (ADDM) automatically identifies high-load SQL statements. If ADDM identifies such statements, then click **Schedule/Run SQL Tuning Advisor** on the Recommendation Detail page to run SQL Tuning Advisor.

To tune SQL statements manually using SQL Tuning Advisor:

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

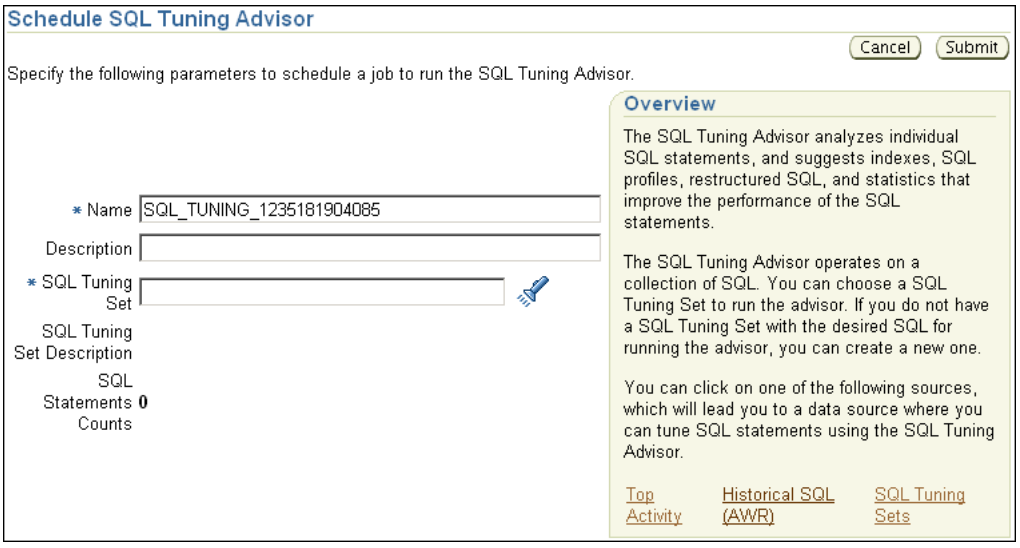
   The Advisor Central page appears.

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

   The SQL Advisors page appears.

3. Under SQL Tuning Advisor, click **SQL Tuning Advisor**.

   The Schedule SQL Tuning Advisor page appears.



4. In the **Name** field, enter a name for the SQL tuning task.

   If unspecified, then SQL Tuning Advisor uses a system-generated name.

5. Do one of the following:

   - To run a SQL tuning task for one or more high-load SQL statements, under SQL Tuning Advisor Data Source Links, click **Top Activity**.

     The Top Activity page appears.

Under Top SQL, select the SQL statement you want to tune and click **Schedule SQL Tuning Advisor**. See "Identifying High-Load SQL Statements Using Top SQL" on page 9-2 to learn how to identify high-load SQL statements using the Top Activity page.

- To run a SQL tuning task for historical SQL statements from the Automatic Workload Repository (AWR), under SQL Tuning Advisor Data Source Links, click **Historical SQL (AWR)**.

  The Historical SQL (AWR) page appears.

  Under Historical SQL (AWR), 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**.

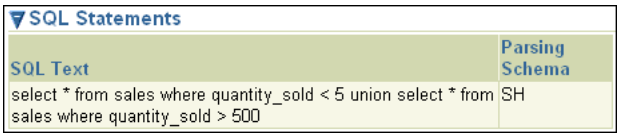
- To run a SQL tuning task for a SQL tuning set, click **SQL Tuning Sets**.

  The SQL Tuning Sets page appears.

  Select the SQL tuning set that contains the SQL statements you want to tune and then click **Schedule SQL Tuning Advisor**. See "Creating a SQL Tuning Set" on page 10-7 to learn how to create SQL tuning sets.

The Schedule SQL Tuning Advisor page reappears.

6. To display the SQL text of the selected statement, expand **SQL Statements**.



7. Under Scope, select the scope of tuning to perform. Do one of the following:

   - Select **Limited**.

     A limited scope takes approximately 1 second to tune each SQL statement but does not recommend a SQL profile.

   - Select **Comprehensive**, and then set a time limit (in minutes) for each SQL statement in the **Time Limit per Statement** field, and a total 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.

     Comprehensive mode may take several minutes to tune a single SQL statement. This mode is both time and resource intensive because each query must be hard-parsed. Thus, you should only use comprehensive scope for high-load SQL statements that have a significant impact on the entire system.

   See "Managing SQL Profiles" on page 10-16 to learn more about SQL profiles.

8. Under Schedule, do one of the following:

   - Select **Immediately** and then click **Submit** to run the SQL tuning task immediately.

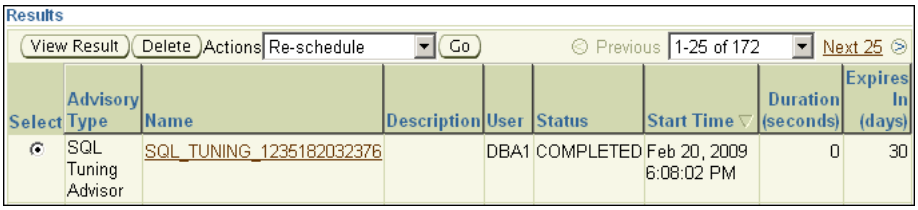
     The Processing: SQL Tuning Advisor Task page appears.

   - Select **Later** to schedule a specific time in the future, and then click **OK**.

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

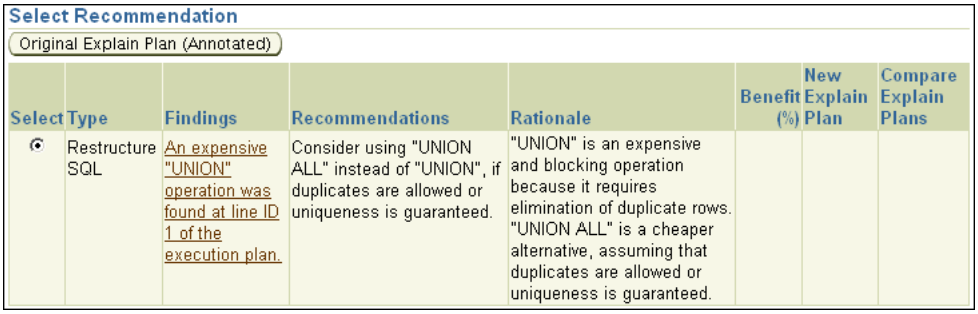
   The Advisor Central page appears.

Under Advisor Tasks, the Results sections lists the result of advisors.



10. Select a result from the table and then click **View Result**.

    The Recommendations for SQL ID page appears.



    If you used a SQL tuning set, then multiple recommendations may be shown. To help you decide whether 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.

11. To implement the recommendation, do one of the following:

    - If an automated solution is recommended, then click **Implement**.

      A confirmation page appears. Click **Yes** to confirm the change.

    - If a manual solution is recommended, then consider implementing the recommendation.

Viewing Automatic SQL Tuning Results

By analyzing data in the Automatic Workload Repository (AWR), the database can identify routine maintenance tasks. The automated maintenance tasks infrastructure (known as **AutoTask**) schedules these tasks to run in maintenance windows.

Maintenance windows are Oracle Scheduler time intervals that belong to the window group named `MAINTENANCE_WINDOW_GROUP`. By default, one window is scheduled for each day of the week. You can customize attributes of these maintenance windows, including start and end times, frequency, and days of the week.

By default, AutoTask runs the following automated maintenance tasks in all maintenance windows:

- Optimizer Statistics Collection
- Segment Advisor
- SQL Tuning Advisor

You can view the results of automated execution of SQL Tuning Advisor on observed high-load SQL statements.

To view automatic SQL tuning results:

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

   The Advisor Central page appears.

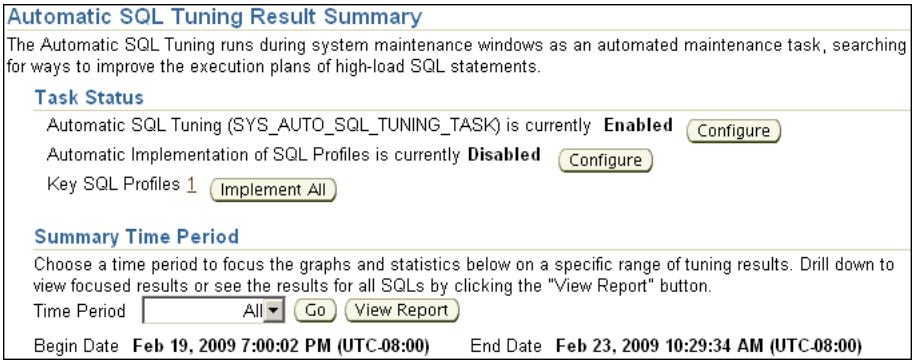
2. Under Advisors, click **SQL Advisors**.

   The SQL Advisors page appears.

3. Under SQL Tuning Advisor, click **Automatic SQL Tuning Results**.

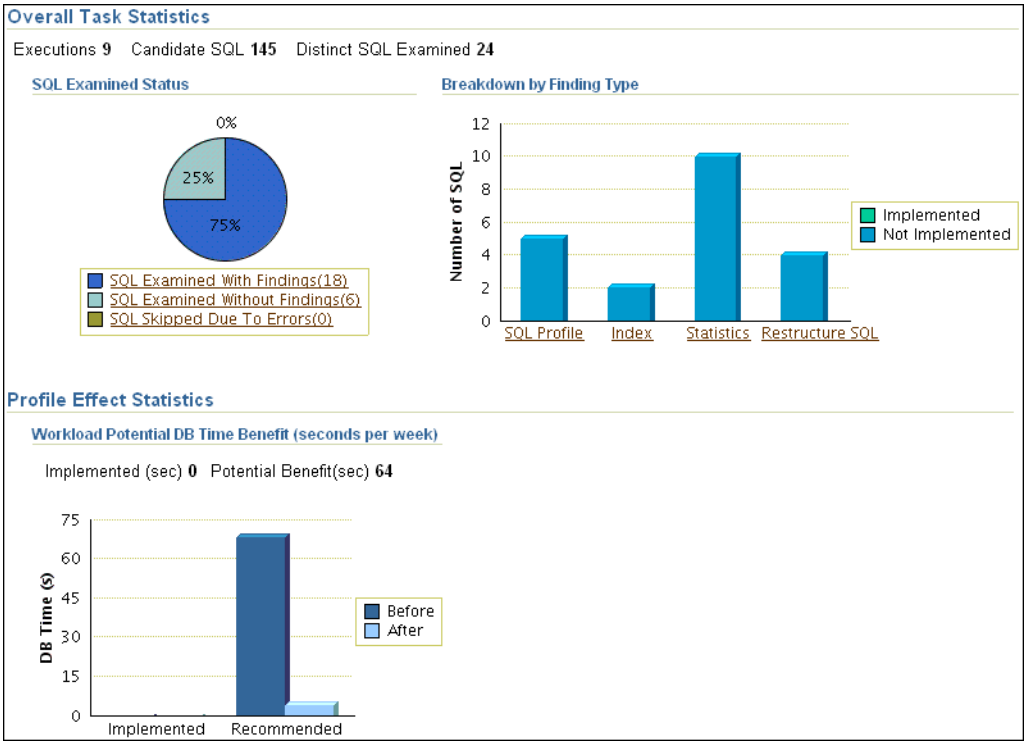
   The Automatic SQL Tuning Result Summary page appears.

   The top half of the page includes sections for the status and activity summary of the SQL Tuning task.



4. In the Time Period list, select **All** and then click **Go**.

   The Overall Task Statistics and Profile Effect Statistics sections are refreshed.



5. Optionally, in the Task Status section, click **Configure** to change the attributes of the Automatic SQL Tuning task.

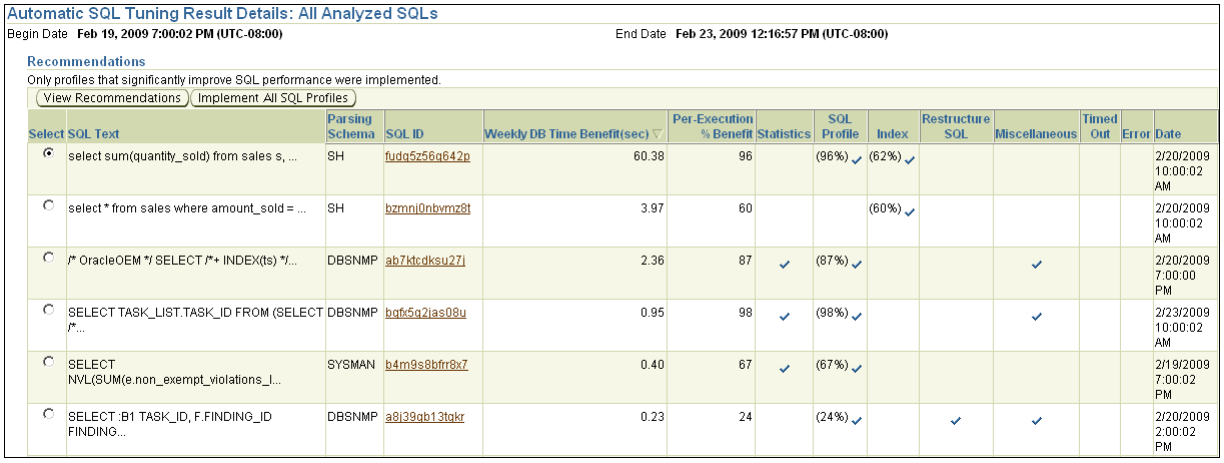
   The Automated Maintenance Tasks Configuration page appears.

In this page, you can enable or disable the Automatic SQL Tuning task and specify which days it should run. Click **Apply** or **Revert** to return to the previous page.

6. In the Task Activity Summary section, leave **All** selected for the **Time Period** and then click **View Report**.

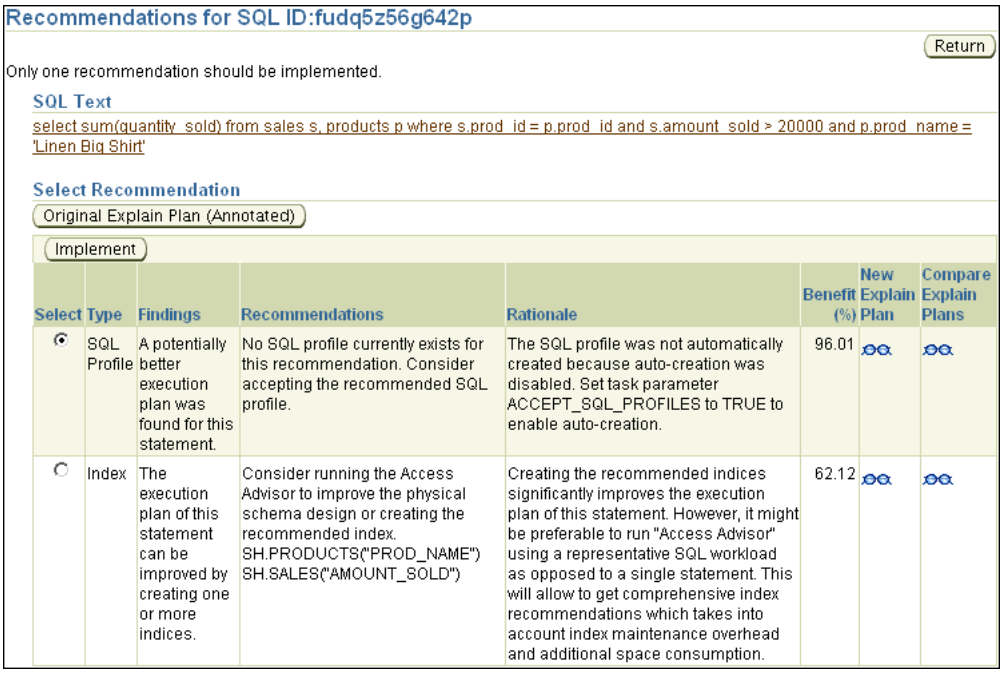
   The Automatic SQL Tuning Result Details page appears.

   The page lists SQL statements that have been automatically selected by the database as candidates for SQL tuning.



7. Under Recommendations, select a SQL statement and then click **View Recommendations**.

   The Recommendations for SQL ID page appears.



This page can include recommendations for SQL profiles and indexes. See "Tuning SQL Manually Using SQL Tuning Advisor" on page 10-2 to learn how to implement recommendations made by SQL Tuning Advisor.

Managing SQL Tuning Sets

A **SQL tuning set** is a database object that includes one or more SQL statements and their execution statistics and context. You can use the set as an input for advisors such as SQL Tuning Advisor, SQL Access Advisor, and SQL Performance Analyzer. You can load SQL statements into a SQL tuning set from different SQL sources, such as AWR, the cursor cache, or high-load SQL statements that you identify.

A SQL tuning set includes the following:

- A set of SQL statements
- Associated execution context such as:
  - User schema
  - Application module name and action
  - List of bind values
  - Cursor compilation environment
- Associated basic execution statistics such as:
  - Elapsed time and CPU time
  - Buffer gets
  - Disk reads
  - Rows processed
  - Cursor fetches
  - Number of executions and number of complete executions
  - Optimizer cost
  - 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, enabling 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 directly on this system. This feature enables you to transport the high-load SQL statements to a test system, where they can be safely analyzed and tuned. For information about transporting SQL tuning sets, see *Oracle Database Performance Tuning Guide*.

Using Enterprise Manager, you can manage SQL tuning sets by doing the following:

- Creating a SQL Tuning Set
- Dropping a SQL Tuning Set
- Transporting SQL Tuning Sets

Creating a SQL Tuning Set

This section describes how to create a SQL tuning set with 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 10-8.
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 10-9.
3. Specify the filter options for the SQL tuning set, as described in "Creating a SQL Tuning Set: Filter Options" on page 10-12.
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 10-12.

Creating a SQL Tuning Set: Options

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

To specify options for creating a SQL tuning set:

1. On the Database Performance page, under Additional Monitoring Links, click **SQL Tuning Sets**.

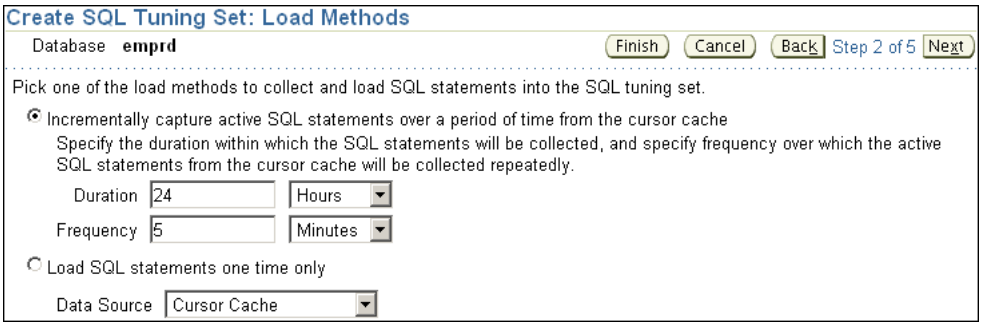
   The SQL Tuning Sets page appears.
2. Click **Create**.

   The Create SQL Tuning Set: Options page appears.
3. Enter the following details:
   - In **SQL Tuning Set Name**, enter a name for the SQL tuning set.
   - In **Owner**, enter the owner of the SQL tuning set.
   - In **Description**, enter a description of the SQL tuning set.
4. Optionally, to create an empty SQL tuning set and add SQL statements to it at a later time, complete the following steps:
   a. Select **Create an empty SQL tuning set**.
   b. Click **Next**.

      The Create SQL Tuning Set: Review page appears.
   c. Review your SQL tuning set options and click **Submit**.

      The empty SQL tuning set is created. You can add SQL statements to it later.
5. Click **Next**.

   The Create SQL Tuning Set: Load Methods page appears.



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

Creating a SQL Tuning Set: Load Method

After options are specified 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 AWR Baselines
- Loading SQL Statements from a User-Defined Workload

> **Tip:** Before selecting the load method for the SQL tuning set, create a SQL tuning set and specify the initial options, as described in "Creating a SQL Tuning Set: Options" on page 10-8

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 technique enables you to not only collect current and recent SQL statements stored in the SQL cache, but also SQL statements that will run during a 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 10-12.

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 in the SQL cache, collecting these statements only once may result in a SQL tuning set this is not representative of the entire database workload.

To load SQL statements from the cursor cache:

1. On the Create SQL Tuning Set: Load Methods page, select **Load SQL statements one time only**.
2. From the Data Source list, select **Cursor Cache**.
3. Click **Next**.

   The Create SQL Tuning Set: Filter Options page is shown.
4. Proceed to the next step, as described in "Creating a SQL Tuning Set: Filter Options" on page 10-12.

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:

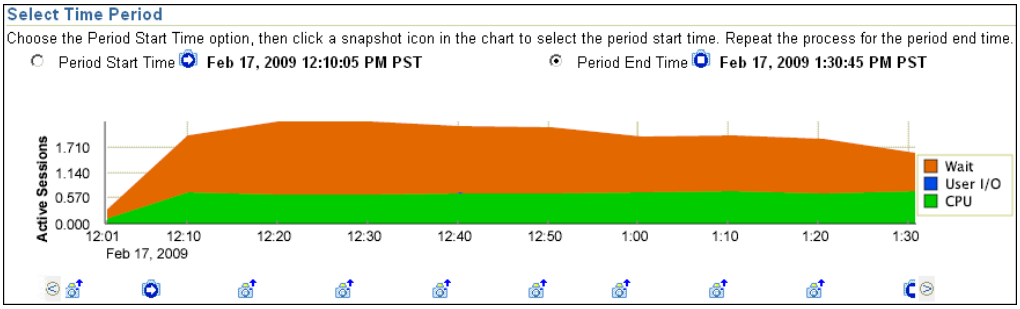
1. On the Create SQL Tuning Set: Load Methods page, select **Load statements one time only**.
2. In the **Data Source** list, select **AWR Snapshots**.
3. In the **AWR Snapshots** field, select the snapshots to include. Do one of the following:
   - Select either **ALL** or a time period such as **Last 24 hours** and then go to Step 5.

     Only snapshots that are captured and stored in AWR in the specified time will be included.
   - Select **Customize** and then go to Step 4.

     Only snapshots that are captured and stored in AWR during a customized time period that you specify will be included.
4. To select a customized time period of snapshots, complete the following steps:
   a. Select **Customize** and then click **Go**.

      The Select Time Period window opens.
   b. For the starting snapshot, select **Period Start Time** and then click the snapshot icon below the Active Session graph that corresponds to the desired start time.
   c. For the ending snapshot, select **Period End Time** and then click the snapshot icon below the Active Session graph that corresponds to the desired end time.
   d. Click **Select**.

   In this example, the snapshot taken on February 17, 2009 at 12:10 p.m. is selected as the start time, and the snapshot taken on February 17, 2009 at 1:30 p.m. is selected as the end time.



5. Click **Next**.

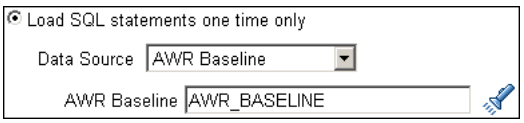
   The Create SQL Tuning Set: Filter Options page is shown.

6. Proceed to the next step, as described in "Creating a SQL Tuning Set: Filter Options" on page 10-12.

Loading SQL Statements from AWR Baselines You can load SQL statements captured in AWR baselines. This technique 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 AWR baselines:

1. On the Create SQL Tuning Set: Load Methods page, select **Load SQL statements one time only**.
2. In the **Data Source** field, select **AWR Baseline**.
3. In the **AWR Baseline** field, select the baseline 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 10-12.

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

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 format of 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 10-12.

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:

- Using filters directs the various advisors that use the SQL tuning set as a workload source, such as SQL Tuning Advisor, SQL Access Advisor, and 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, do the following:
>
> - Create a SQL tuning set and specify the initial options, as described in "Creating a SQL Tuning Set: Options" on page 10-8
> - Select the load method, as described in "Creating a SQL Tuning Set: Load Method" on page 10-9

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 to 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**.

   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 10-12.

Creating a SQL Tuning Set: Schedule

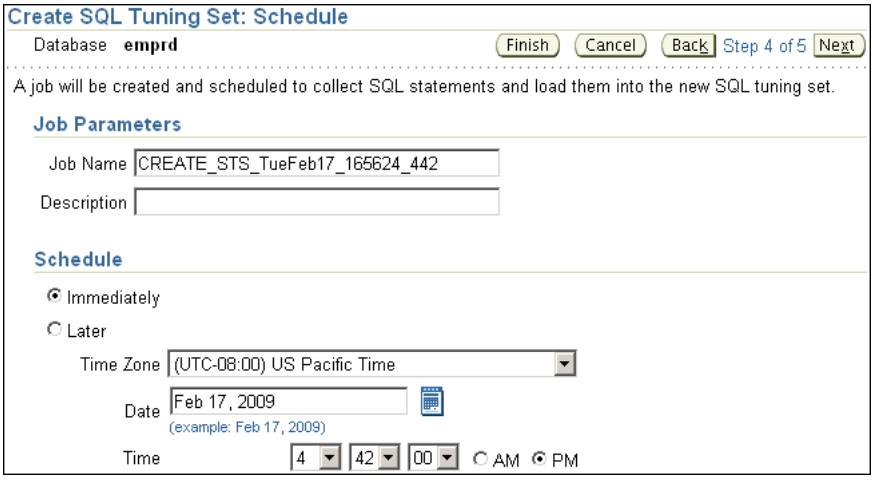
After the filter options are specified for the SQL tuning set, you can schedule and submit a job to collect the SQL statements and load them into the SQL tuning set.

Tip: Before you can schedule a job to create the SQL tuning set, do the following:

- Create a SQL Tuning Set and specify the initial options, as described in "Creating a SQL Tuning Set: Options" on page 10-8.
- Select the load method, as described in "Creating a SQL Tuning Set: Load Method" on page 10-9.
- Specify the filter options, as described in "Creating a SQL Tuning Set: Filter Options" on page 10-12.

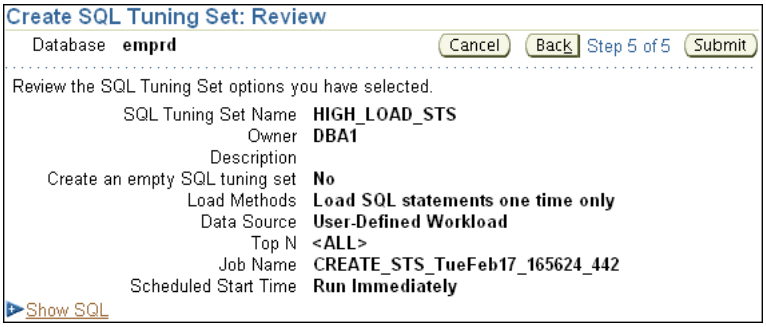
To schedule and submit a job to create a SQL tuning set:

1. On the Create SQL Tuning Set: Schedule page, under Job Parameters, enter a name 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, do one of the following:
   - **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, then a message is displayed to inform you that the job and the SQL tuning set were 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 job, such as operation status, click **View Job Details**.

   The View Job page appears to display details about the job.

Dropping a SQL Tuning Set

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

To drop a SQL tuning set:

1. On the Database Performance page, under Additional Monitoring Links, click **SQL Tuning Sets**.

   The SQL Tuning Sets page appears.

   Existing SQL tuning sets are displayed on this page.

2. Select the SQL tuning set you want to drop and then click **Drop**.

   The Confirmation page appears to verify that you want to drop the selected SQL tuning set.

3. Click **Yes**.

   The SQL Tuning Sets page appears.

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

Transporting SQL Tuning Sets

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

This section contains the following topics:

- Exporting a SQL Tuning Set
- Importing a SQL Tuning Set

Exporting a SQL Tuning Set

This section describes how to export a SQL tuning set, enabling it to be transported to another system.

To export a SQL tuning set:

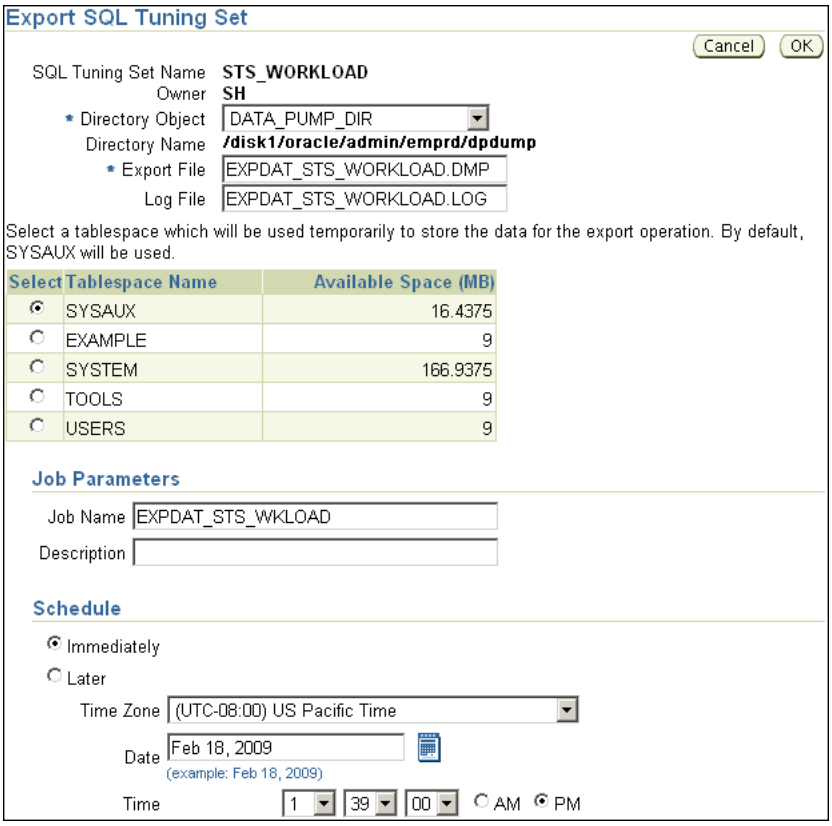
1. On the Database Performance page, under Additional Monitoring Links, click **SQL Tuning Sets**.

   The SQL Tuning Sets page appears.

   Existing SQL Tuning Sets are displayed on this page.

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

   The Export SQL Tuning Set page appears.



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

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

4. In the **Export File** field, enter a name for the file to be database.

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

5. 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 database.

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

   By default, `SYSAUX` is used.

7. Under Job Parameters, in the **Job Name** field, enter a name for the job.

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

8. Under Schedule, do one of the following:

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

9. Click **OK**.

   The SQL Tuning Sets page appears.

   A confirmation message indicates that the job was created successfully.

10. Transport the export file to another system using the mechanism of choice, such as Oracle Data Pump or a database link.

Importing a SQL Tuning Set

Before a SQL tuning set can be imported, you must first export a SQL tuning set from another system and transport it to your current system. For more information, see "Exporting a SQL Tuning Set" on page 10-14.

To import a SQL tuning set:

1. On the Database Performance page, under Additional Monitoring Links, click **SQL Tuning Sets**.

   The SQL Tuning Sets page appears.

2. Click **Import**.

   The Import SQL Tuning Set page appears.

3. In **Directory Object**, select the directory containing the file to be imported.

   The directory should contain the export file that was transported to your current system. For example, if the file resides in the Data Pump directory, then select `DATA_PUMP_DIR`. The Directory Name field refreshes automatically to indicate the selected directory.

4. In the **Import File** field, enter the name of the dump file that will be imported.

5. In the **Log File** field, enter a name for the log file for the import operation.

6. To replace an existing SQL tuning set with the one that you are importing, select **Replace the existing SQL tuning set if one exists**.

7. Select a tablespace to temporarily store the data for the import operation.

   By default, `SYSAUX` is used.

8. Under Job Parameters, in the **Job Name** field, enter a name for the job.

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

9. Under Schedule, do one of the following:

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

10. Click **OK**.

    The SQL Tuning Sets page appears.

    A confirmation message is displayed to indicate that the job was successfully created. If the job is scheduled to run immediately, then the imported SQL tuning set will be shown on this page. You may need to refresh to see the SQL tuning set.

Managing SQL Profiles

A **SQL profile** is a set of auxiliary information that is built during automatic tuning of a SQL statement. A SQL profile is to a SQL statement what statistics are to a table.

When running a SQL Tuning Advisor task with a limited scope, the optimizer makes estimates about cardinality, selectivity, and cost that are sometimes significantly off, 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 into a SQL profile. The database can use the profile to verify and, if necessary, adjust optimizer estimates.

During SQL profiling, the optimizer uses the execution history of the SQL statement to create appropriate settings for optimizer parameters. After SQL profiling completes, the optimizer uses the information in the SQL profile and regular database statistics to generate execution plans. The additional information enables the database 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, then the database creates the SQL profile and enables it 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, then you should enable it again to avoid performance degradation. If the SQL statement is performing optimally after you have disabled the SQL profile, then you may want to remove the SQL profile from your database.

To enable, disable, or delete a SQL profile:

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

   The Top Activity page appears.

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

   The SQL Details page appears.

3. Click the **Plan Control** tab.

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

4. Select the SQL profile you want to manage. Do one of the following:

   - To enable a SQL profile that is disabled, click **Disable/Enable**.
   - To disable a SQL profile that is enabled, click **Disable/Enable**.
   - To remove a SQL profile, click **Delete**.

   A confirmation page appears.

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

> **See Also:** *Oracle Database Performance Tuning Guide* to learn how to manage SQL profiles using an API

Managing SQL Execution Plans

SQL plan management is a preventative mechanism that records and evaluates execution plans of SQL statements over time. The database builds **SQL plan baselines** consisting of a set of efficient plans. If the same SQL statement runs repeatedly, and if the optimizer generates a new plan differing from the baseline, then the database compares the plan with the baseline and chooses the best one.

SQL plan management avoids SQL performance regression. Events such as new optimizer statistics, changes to initialization parameters, database upgrades, and so on can cause changes to execution plans. These changes can cause SQL performance regressions that are difficult and time-consuming to fix manually. SQL plan baselines preserve performance of SQL statements, regardless of changes in the database.

You can have the database capture SQL plan baselines automatically or you can load SQL execution plans manually.

To capture SQL plan baselines automatically:

1. Access the Database Home page.

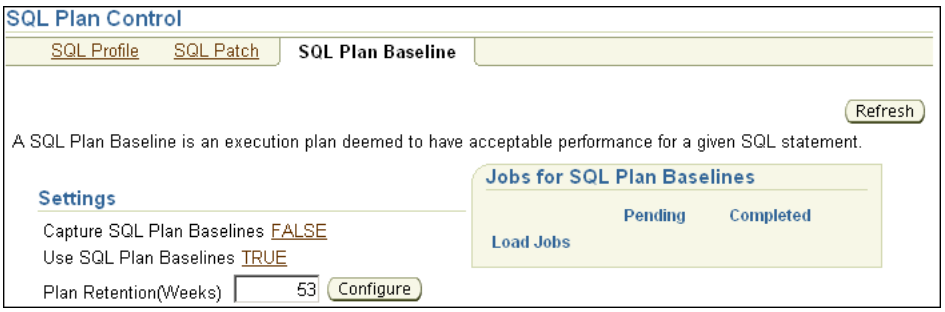
   See *Oracle Database 2 Day DBA* for more information.

2. Select **Performance**, then **SQL**, and then **SQL Plan Control**.

   If the Database Login page appears, then log in as a user with administrator privileges. The SQL Profile subpage of the SQL Plan Control page appears.

3. Click **SQL Plan Baseline**.

   The SQL Plan Baseline subpage appears.



4. Under Settings, click the link next to **Capture SQL Plan Baselines**.

   The Initialization Parameters page appears.

5. In the **Value** column of the table, select **TRUE** and then click **OK**.

   You are returned to the SQL Plan Baseline subpage, which now shows **Capture SQL Baselines** set to **TRUE**.

   Because you configured baselines to be captured, the database automatically keeps a history of execution plans for all SQL statements executed more than once.

To load SQL execution plans manually, the Capture SQL Baselines setting must be FALSE.

To manually load SQL execution plans:

1. Access the Database Home page.

   See *Oracle Database 2 Day DBA* for more information.

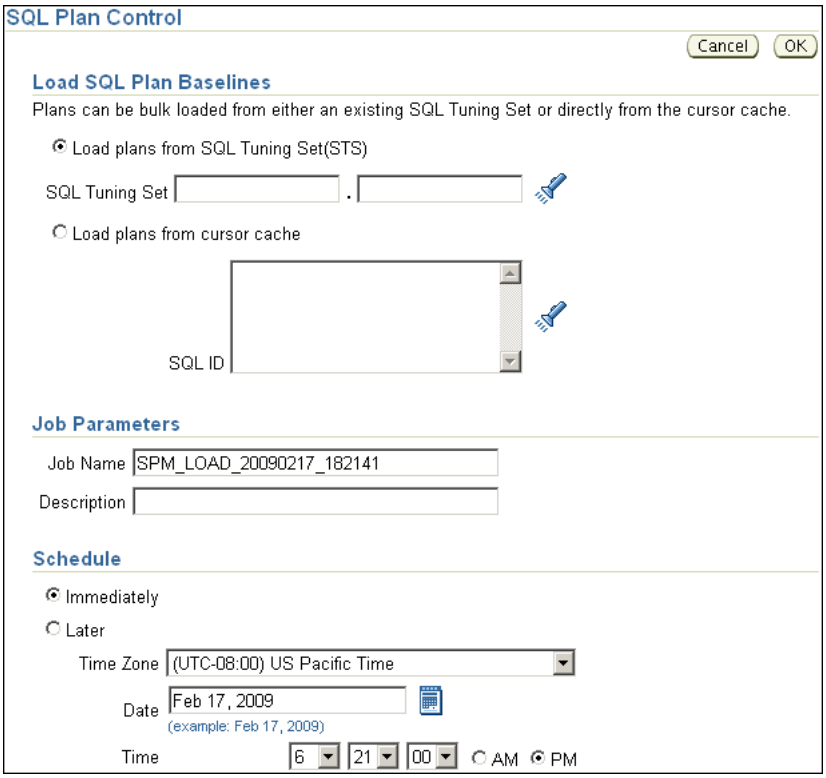
2. Select **Performance**, then **SQL**, and then **SQL Plan Control**.

3. Click **SQL Plan Baseline**.

   The SQL Plan Baseline subpage appears.

4. Click **Load**.

   The SQL Plan Control page appears.



5. Select the SQL plan baselines to be loaded by completing the following steps:

   a. Under Load SQL Plan Baselines, select **Load plans from SQL Tuning Set (STS)**.

      In this example, load plans from the SQL tuning set that you created in "Creating a SQL Tuning Set" on page 10-7.

   b. In **Job Name**, enter a name for the job. For example, enter `SPM_LOAD_TEST`.

   c. Under Schedule, select **Immediately**.

   d. Click **OK**.

   The SQL Profile subpage of the SQL Plan Control page appears.

   The table displays a list of SQL plans that are stored as SQL plan baselines.

Select	Name	SQL Text	Enabled	Accepted	Fixed	Auto Purge	Created	Last Modified
☐	SQL_PLAN_dk7t8ctf0wx8bc0c79166	select value from v$parameter where name = lower('...	YES	YES	NO	YES	Feb 17, 2009 6:24:35 PM	Feb 17, 2009 6:24:35 PM
☐	SQL_PLAN_brbbc88xqkvm34f0cc5e3	/* OracleOEM */ SELECT TO_CHAR(CAST (md.en...	YES	YES	NO	YES	Feb 17, 2009 6:21:53 PM	Feb 17, 2009 6:21:53 PM
☐	SQL_PLAN_7t8gmw5336hsy570ba0a9	SELECT count(*) from (SELECT owner, name, descript...	YES	YES	NO	YES	Feb 17, 2009 6:23:25 PM	Feb 17, 2009 6:23:25 PM
☐	SQL_PLAN_7pgx5q0bj6wjcf6142ce6	select job_name, state , 'LOAD' from all_scheduler...	YES	YES	NO	YES	Feb 17, 2009 6:24:35 PM	Feb 17, 2009 6:24:35 PM
☐	SQL_PLAN_18w49gtxr5kc2c0c79166	select value from v$parameter where name = lower('...	YES	YES	NO	YES	Feb 17, 2009 6:24:35 PM	Feb 17, 2009 6:24:35 PM
☐	SQL_PLAN_07pmpy8bksmgz6d032274	/* OracleOEM */ SELECT SEVERITY_INDEX, CR...	YES	YES	NO	YES	Feb 17, 2009 6:21:54 PM	Feb 17, 2009 6:21:54 PM

6. Optionally, fix the execution plan of a baseline to prevent the database from using an alternative SQL plan baseline. Complete the following steps:

   a. Select a SQL plan baseline that is not fixed.

   b. Select **Fixed - Yes** from the list preceding the baseline table.

   c. Click **Go**.

   The table is refreshed to show the SQL execution plan with the value `YES` in the Fixed column of the table.

See Also:

- *Oracle Database Performance Tuning Guide* to learn how to use SQL plan management

11

Optimizing Data Access Paths

To achieve optimal performance for data-intensive queries, materialized views and indexes are essential for SQL statements. However, implementing these objects does not come without cost. Creation and maintenance of these objects can be time-consuming. Space requirements can be significant. SQL Access Advisor enables you to optimize query access paths by recommending materialized views and view logs, indexes, SQL profiles, and partitions for a specific workload.

A **materialized view** provides access to table data by storing query results in a separate schema object. Unlike an ordinary view, which does not take up storage space or contain data, a materialized view contains the rows from a query of 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. SQL Access Advisor recommends how to optimize materialized views so that they can be rapidly refreshed and make use of the query rewrite feature. To learn more about materialized views, see *Oracle Database Concepts*.

SQL Access Advisor also recommends bitmap, function-based, and B-tree indexes. A **bitmap index** reduces response time for many types of ad hoc queries and can also reduce storage space compared to other indexes. 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 search for values as if they were all in uppercase. **B-tree indexes** are commonly used to index unique or near-unique keys.

Using SQL Access Advisor involves the following tasks:

- Running 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 SQL Tuning Advisor
- *Oracle Database Concepts* to learn about indexes

Running SQL Access Advisor

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

To run SQL Access Advisor:

1. Select the initial options, as described in "Running 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 SQL Access Advisor: Workload Source" on page 11-3.
3. Define the filters options, as described in "Running SQL Access Advisor: Filter Options" on page 11-5.
4. Choose the types of recommendations, as described in "Running SQL Access Advisor: Recommendation Options" on page 11-7.
5. Schedule the SQL Access Advisor task, as described in "Running SQL Access Advisor: Schedule" on page 11-9.

Running SQL Access Advisor: Initial Options

The first step in running 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**.

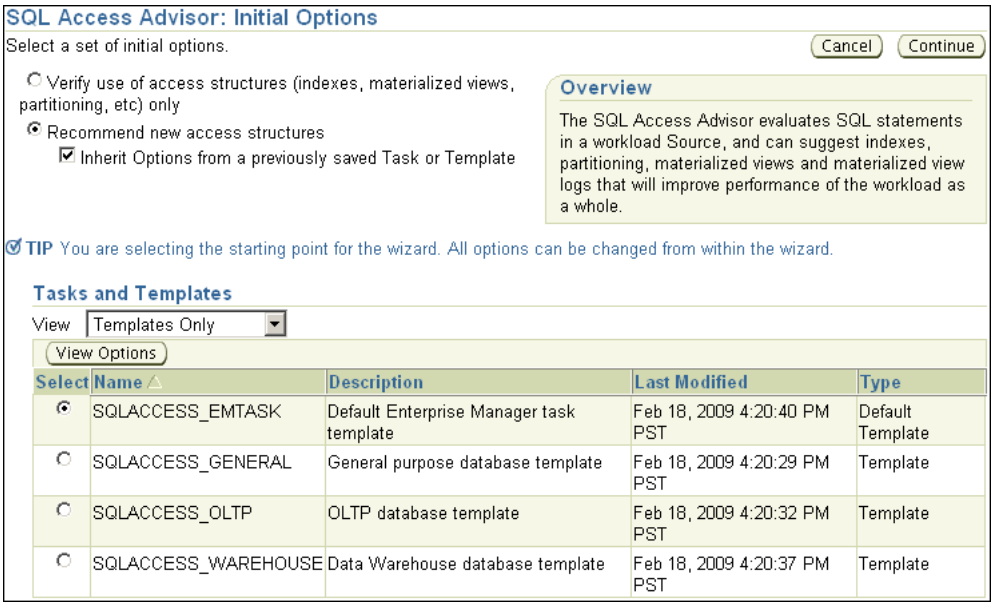
   The Advisor Central page appears.
2. Under Advisors, click **SQL Advisors**.

   The SQL Advisors page appears.
3. Click **SQL Access Advisor**.

   The SQL Access Advisor: Initial Options page appears.
4. Do one of the following:
   - Select **Verify use of access structures (indexes, materialized views, partitioning, and so on) only** to verify existing structures.
   - Select **Recommend new access structures** to use the recommended options defined in the Oracle Enterprise Manager default template.

     If you select this option, then you can optionally complete the following steps:
     - Select **Inherit Options from a previously saved Task or Template** to use the options defined in an existing SQL Access Advisor task or another template.
     - In Tasks and Templates, select the task or template that you want to use.

   In this example, **Recommend new access structures** is selected.



5. Click **Continue**.

   The SQL Access Advisor: Workload Source page appears.

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

Running SQL Access Advisor: Workload Source

After initial options are specified for SQL Access Advisor, select the workload source that 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 Hypothetical Workload

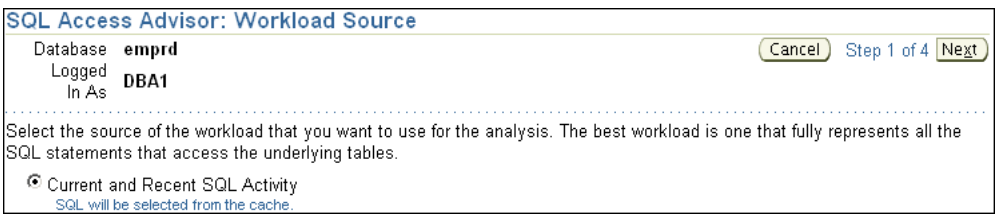
> **Tip:** Before you can select the workload source for SQL Access Advisor, select the initial options, as described in "Running SQL Access Advisor: Initial Options" on page 11-2.

Using SQL Statements from the Cache

You can use SQL statements from the cache as the workload source. However, only current and recent SQL statements are stored in the SQL cache, so 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 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 option is useful because SQL tuning sets can be used repeatedly as the workload source for SQL Access Advisor and SQL Tuning Advisor.

To use a SQL tuning set as the workload source:

1. On the SQL Access Advisor: Workload Source page, select **Use an existing SQL Tuning Set**.
2. Click the **SQL Tuning Set** search icon to use an existing SQL tuning set.

   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 then click **Go**.

   A list of SQL tuning sets contained in the selected schema appears.
4. Select the SQL tuning set to be used 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 SQL Access Advisor: Filter Options" on page 11-5.

See Also:

- "Managing SQL Tuning Sets" on page 10-7

Using a Hypothetical Workload

A **dimension table** stores all or part of the values for a logical dimension in a star or snowflake schema. You can create a hypothetical workload from dimension tables containing primary or foreign key constraints. This option is useful if the workload to be analyzed does not exist. In this case, SQL Access Advisor examines the current logical schema design, and provides 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. Leave **Schemas and Tables** empty and then click **Add** to search for tables.

   The Workload Source: Search and Select Schemas and Tables page appears.
3. In the Tables section, enter a schema name in the **Schema** field and then click **Search**.

   A list of tables in the selected schema is displayed.
4. Select the tables to be used in creating the hypothetical workload and then click **Add Tables**.

   The selected tables now appear in the **Schemas and Tables** field.
5. Click **OK**.

The SQL Access Advisor: Workload Source page appears with the selected tables now added.

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

See Also:

- *Oracle Database Concepts* for an overview of materialized views

Running SQL Access Advisor: Filter Options

After the workload source is selected, you can optionally apply filters to reduce the scope of the SQL statements found in the workload. Filters are beneficial for the following reasons:

- Using filters directs 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, do the following:

- Select initial options, as described in "Running SQL Access Advisor: Initial Options" on page 11-2.
- Select the workload source, as described in "Running 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 Modules
   - Defining Filters for Actions
4. Click **Next**.

   The Recommendation Options page appears.

5. Proceed to the next step, as described in "Running SQL Access Advisor: Recommendation Options" on page 11-7.

Defining Filters for Resource Consumption

The resource consumption filter restricts the workload to include only the number of high-load SQL statements that you specify.

To define a filter for resource consumption:

1. On the SQL Access Advisor: Workload Source page, under 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 methods by which the SQL statements are to be ordered.

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. On the SQL Access Advisor: Workload Source page, 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 whose SQL statements you want to include or exclude and then 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 `SH`.

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. Table filters are not permitted if you selected the **Create a Hypothetical Workload from the Following Schemas and Tables** option, as described in "Using a Hypothetical Workload" on page 11-4.

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 tables, 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 contain 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 Modules

The module filter restricts the workload to include or exclude SQL statements that are associated with modules that you specify.

To define a filter for module ID:

1. Do one of the following:
   - To include only SQL statements associated with a specific module ID in the workload, select **Include only SQL statements associated with these modules**.
   - To exclude all SQL statements associated to a specific module ID from the workload, select **Exclude all SQL statements associated with these modules**.
2. In the Modules field, enter the names of the modules 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. Do one of the following:
   - To include only SQL statements associated with a specific action in the workload, select **Include only SQL statements associated with these actions**.
   - To exclude all SQL statements associated with a specific action from the workload, select **Exclude all SQL statements associated with these actions**.
2. In the Actions field, enter the actions for which associated SQL statements will be included or excluded.

Running SQL Access Advisor: Recommendation Options

To improve the underlying data access methods chosen by the optimizer for the workload, SQL Access Advisor provides recommendations for indexes, materialized views, and partitioning. 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.

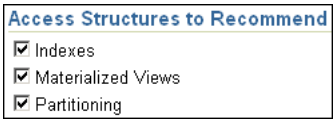
Tip: Before you can select the recommendation options for SQL Access Advisor, do the following:

- Select initial options, as described in "Running SQL Access Advisor: Initial Options" on page 11-2.
- Select the workload source, as described in "Running SQL Access Advisor: Workload Source" on page 11-3.
- Define the filter options, as described in "Running SQL Access Advisor: Filter Options" on page 11-5.

To specify recommendation options:

1. On the SQL Access Advisor: Recommendation Options page, under Access Structures to Recommend, select the type of access structures to be recommended by SQL Access Advisor:
   - Indexes
   - Materialized Views
   - Partitioning

   In this example, all of the preceding access types are selected.

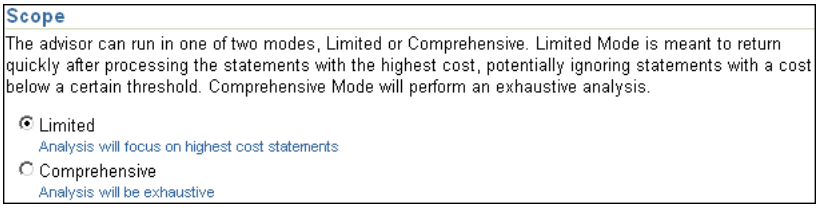


2. Under Scope, select the mode in which SQL Access Advisor will run. Do one of the following:
   - Select **Limited**.

     In limited mode, SQL Access Advisor focuses on SQL statements with the highest cost in the workload. The analysis is quicker, but the recommendations may be limited.
   - Select **Comprehensive**.

     In comprehensive mode, SQL Access Advisor analyzes all SQL statements in the workload. The analysis can take much longer, but the recommendations will be exhaustive.

   In this example, **Limited Mode** is selected.



3. Optionally, click **Advanced Options**.

   The Advanced Options section expands. This section contains the following subsections:
   - Workload Categorization

In this section, you can specify the type of workload for which you want a recommendation. The following categories are available:

– Workload Volatility

Select **Consider only queries** if the workload primarily contains read-only operations, as in data warehouses. Volatility data is useful for online transaction processing (OLTP) systems, where the performance of `INSERT`, `UPDATE`, and `DELETE` operations is critical.

– Workload Scope

Select **Recommend dropping unused access structures** if the workload represents all access structure use cases.

- Space Restrictions

Indexes and materialized views increase performance at the cost of space. Do one of the following:

– Select **No, show me all recommendations (unlimited space)** to specify no space limits. When SQL Access Advisor is invoked with no space limits, it makes the best possible performance recommendations.

– Select **Yes, limit additional space to** and then enter the space limit in megabytes, gigabytes, or terabytes. When SQL Access Advisor is invoked with a space limit, it produces only recommendations with space requirements that do not exceed the specified limit.

- Tuning Prioritization

This section enables you to specify how SQL statements will be tuned. Complete the following steps:

– From the Prioritize tuning of SQL statements by list, select a method by which SQL statements are to be tuned and then click **Add**.

– Optionally, select **Allow Advisor to consider creation costs when forming recommendations** to weigh the cost of creating access structures against the frequency and potential improvement of SQL statement execution time. Otherwise, creation cost will be ignored. You should select this option if you want specific recommendations generated for SQL statements that are executed frequently.

- Default Storage Locations

Use this section to override the defaults defined for schema and tablespace locations. By default, indexes are in the schema and tablespace of the table they reference. Materialized views are in the schema and tablespace of the first table referenced in the query. Materialized view logs are in the default tablespace of the schema of the table that they reference.

4. Click **Next**.

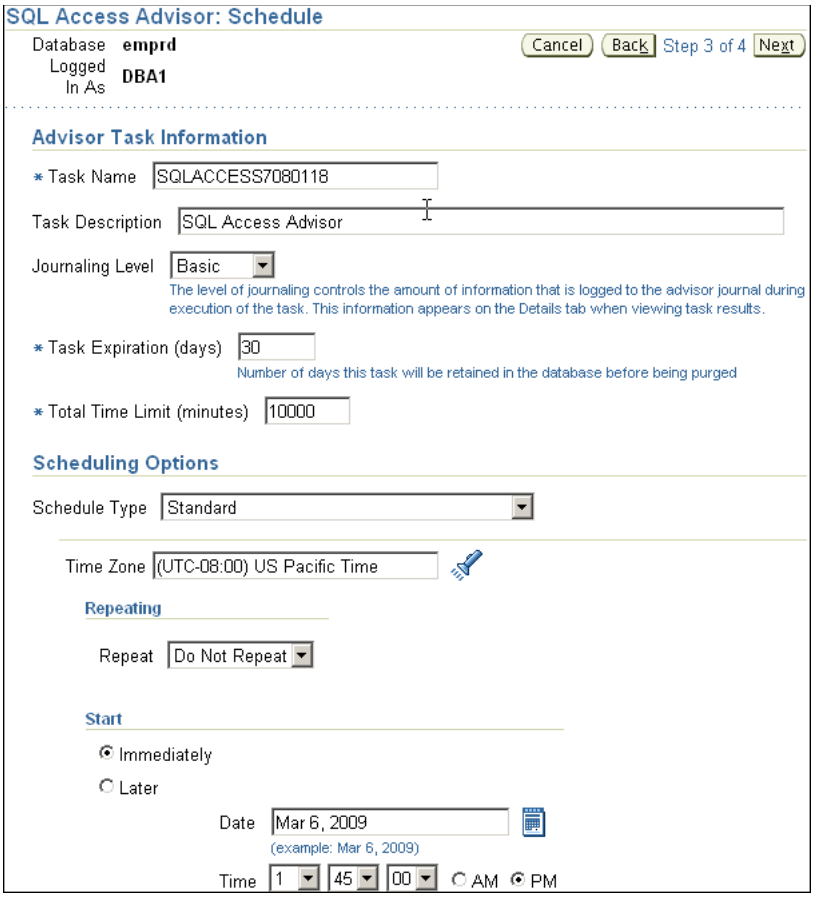
The SQL Access Advisor: Schedule page appears.

5. Proceed to the next step, as described in "Running SQL Access Advisor: Schedule" on page 11-9.

Running SQL Access Advisor: Schedule

Use the SQL Access Advisor Schedule page 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, do the following:

- Select initial options, as described in "Running SQL Access Advisor: Initial Options" on page 11-2.
- Select the workload source, as described in "Running SQL Access Advisor: Workload Source" on page 11-3.
- Define the filter options, as described in "Running SQL Access Advisor: Filter Options" on page 11-5.
- Specify the recommendation options, as described in "Running 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, `SQLACCESS9084523` 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.

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. In the **Total Time Limit (minutes)** field, enter the maximum number of minutes that the job is permitted to run.

   You must enter a time in this field rather than use the default.

6. Under Scheduling Options, in the Schedule Type list, select a schedule type for the task and a maintenance window in which the task should run. Do one of the following:

   - Click **Standard**.

     This schedule type enables you to select a repeating interval and start time for the task. Complete the following steps:

     – Enter your time zone code in the **Time Zone** field or click the search icon to locate the code for your area.

     – In the Repeat list, select **Do Not Repeat** to perform the task only once, or select a unit of time and enter the number of units in the **Interval** field.

     – Under Start, select **Immediately** to start the task now, or **Later** to schedule the task to start at a time specified using the **Date** and **Time** fields.

   - Click **Use predefined schedule**.

     This schedule type enables you to select an existing schedule. Do one of the following:

     – In the **Schedule** field, enter the name of the schedule to be used for the task.

     – To search for a schedule, click the search icon.

       The Search and Select: Schedule dialog box appears.

       Select the desired schedule and click **Select**. The selected schedule now appears in the **Schedule** field.

   - Click **Standard using PL/SQL for repeated interval**.

     This schedule type enables you to select a repeating interval and an execution time period (window) for the task. Complete the following steps:

     – Enter your time zone code in the **Time Zone** field or click the search icon to locate the code for your area.

     – Under Available to Start, select **Immediately** to start the task now, or **Later** to schedule the task to start at a time specified using the **Date** and **Time** fields.

     – In the Repeat list, select **Do Not Repeat** to perform the task only once, or select a unit of time and enter the number of units in the **Interval** field.

     – In the **Repeated Interval** field, enter a PL/SQL schedule expression, such as `SYSDATE+1`.

  - Under Not Available After, select **No End Date** to indicate that there is no end date for the execution window, or **Specified End Date** to specify an end date using the **Date** and **Time** fields.

- Click **Use predefined window**.

  This schedule type enables you to select an existing window. Select **Stop on Window Close** to stop the job when the window closes. Do one of the following:

  - In the **Window** field, enter the name of the window to be used for the task.
  - To search for a window, click the search icon.

    The Search and Select: Window and Window Groups dialog box appears.

    Select the desired window and click **Select**. The selected window now appears in the **Schedule** field.

- Click **Event**.

  Complete the following steps:

  - Enter your time zone code in the **Time Zone** field or click the search icon to locate the code for your area.
  - Under Event Parameters, enter values in the **Queue Name** and **Condition** fields.
  - Under Start, select **Immediately** to start the task now, or **Later** to schedule the task to start at a time specified using the **Date** and **Time** fields.
  - Under Not Available After, select **No End Date** to indicate that there is no end date for the execution window, or **Specified End Date** to specify an end date using the **Date** and **Time** fields.

- Click **Calendar**.

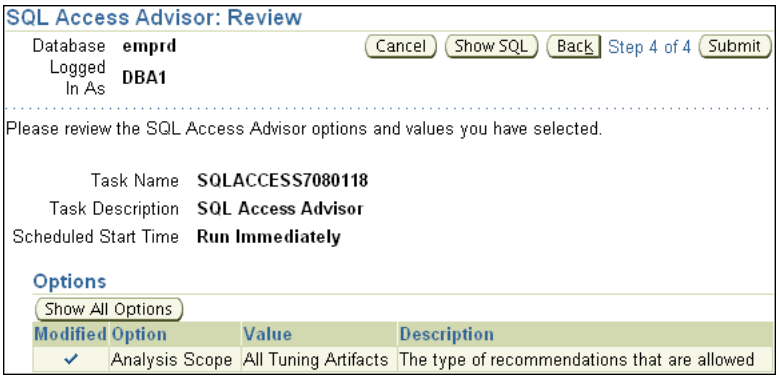
  Complete the following steps:

  - Enter your time zone code in the **Time Zone** field or click the search icon to locate the code for your area.
  - Under Calendar Expression, enter a calendar expression.
  - Under Start, select **Immediately** to start the task now, or **Later** to schedule the task to start at a time specified using the **Date** and **Time** fields.
  - Under Not Available After, select **No End Date** to indicate that there is no end date for the execution window, or **Specified End Date** to specify an end date using the **Date** and **Time** fields.

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.

7. Click **Next**.

   The SQL Access Advisor: Review page appears.



Under Options, a list of modified options for the SQL Access Advisor task is shown. To display both modified and unmodified options, click **Show All Options**. To view the SQL text for the task, click **Show SQL**.

8. Click **Submit**.

   The Advisor Central page appears. A message informs you that the task was created successfully.

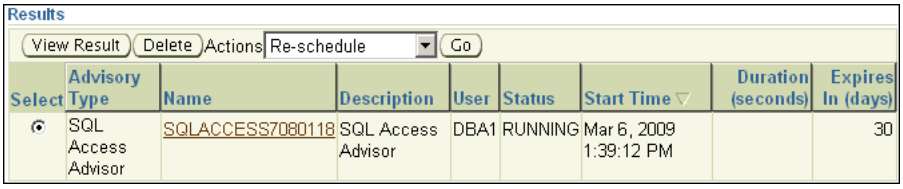
Reviewing the SQL Access Advisor Recommendations

SQL Access Advisor graphically displays the recommendations and provides hyperlinks so that you can quickly see which SQL statements benefit from a recommendation. Each recommendation produced by the SQL Access Advisor is linked to the SQL statement it benefits.

> **Tip:** Before reviewing the SQL Access Advisor recommendations, run SQL Access Advisor to make the recommendations, as described in "Running 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 for review and click **View Result**.



   If the task is not displayed, then 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 11-14.

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 11-18.

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 11-19.

Reviewing the SQL Access Advisor Recommendations: Summary

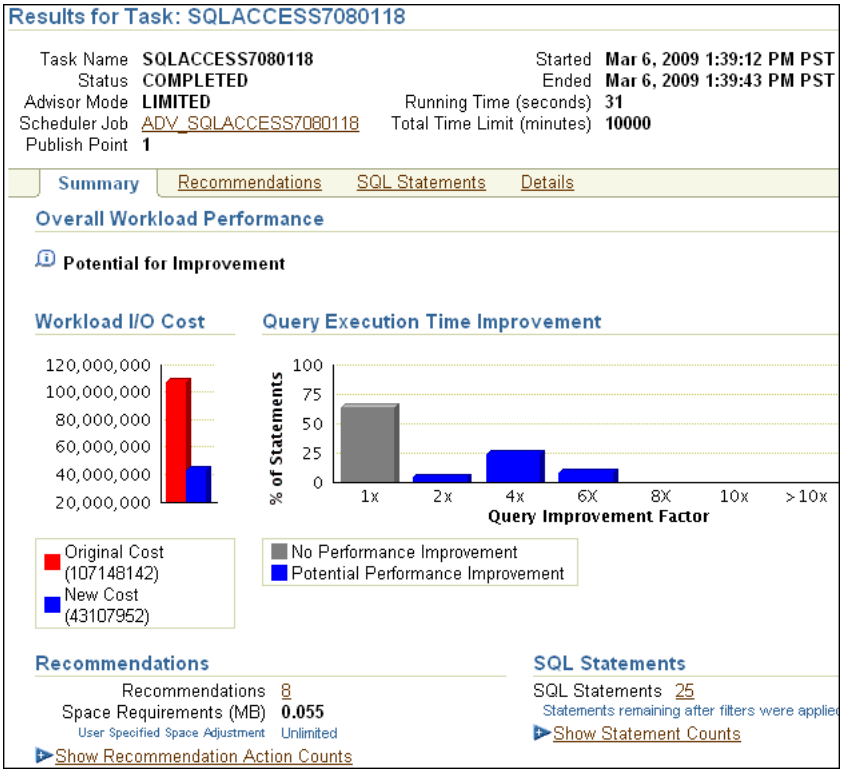
The Summary subpage displays an overview of the SQL Access Advisor analysis.

To review the recommendations summary:

1. On the Results for Tasks page, click **Summary**.

   The Summary subpage of the Results for Tasks page appears.

   In this example, **Limited Mode** is selected so that SQL Access Advisor analyzes the highest cost statements rather than all statements.



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 with the new cost.

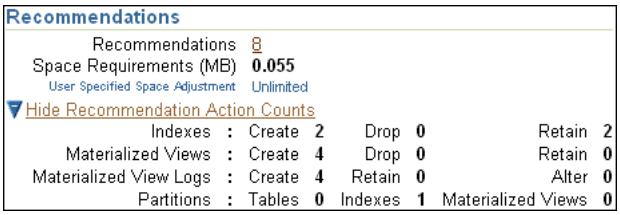
   In this example, the workload I/O cost will decrease from 107.1 million to 43.1 million by implementing the recommendations.
4. Use the Query Execution Time Improvement chart to compare the improvement in query execution time.

   This 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 along the vertical axis (0% to 100%).

In this example, approximately 62 percent of SQL statements in the workload will not improve execution time, but about 25 percent will have the potential for improvement of over 4x or more.

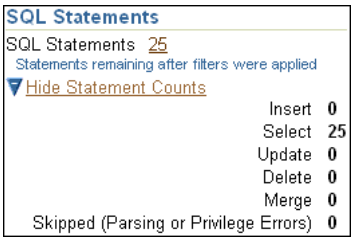
5. Under Recommendations, click **Show Recommendation Action Counts**.

   In the following example, creating 2 indexes, 4 materialized views, and 4 materialized view logs is recommended.



6. Under SQL Statements, click **Show Statement Counts** to display the type of SQL statement.

   In the following example, 25 `SELECT` statements are analyzed.



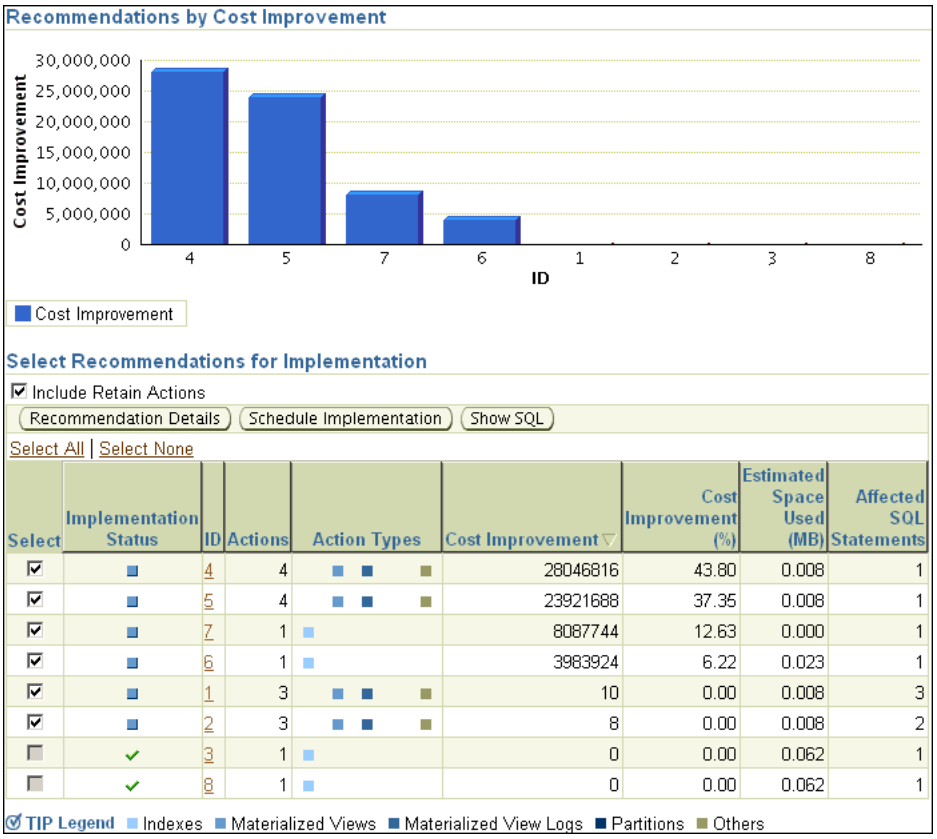
Reviewing the SQL Access Advisor Recommendations: Recommendations

The Recommendations subpage ranks the SQL Access Advisor recommendations by cost improvement. You can also view details about each recommendation.

To review recommendation details:

1. On the Results for Tasks page, click **Recommendations**.

   The Recommendations subpage appears.

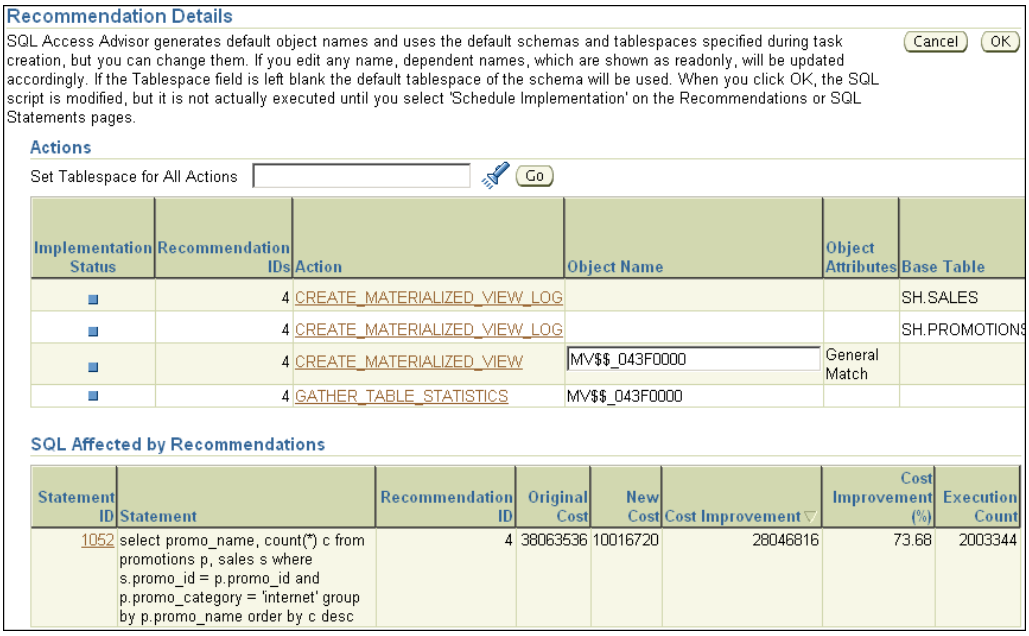


2. Use the Recommendations by Cost Improvement chart to view recommendations ordered by the cost improvement.

   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 statements for each recommendation. Implementing the top recommendation will have the biggest benefit to the total performance of the workload.

3. To view details for a particular recommendation, select the recommendation and click **Recommendation Details**.

   The Recommendation Details page appears.



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.

4. Click **OK**.

   The Recommendations subpage appears.

5. To view the SQL text of a recommendation, select the recommendation and click **Show SQL**.

   The Show SQL page for the selected recommendation appears.

Show SQL

Done

```
Rem SQL Access Advisor: Version 11.2.0.0.2 - Production
Rem
Rem Username: DBA1
Rem Task: SQLACCESS7080118
Rem Execution date:
Rem

CREATE MATERIALIZED VIEW LOG ON
"SH"."SALES"
WITH ROWID, SEQUENCE
("PROD_ID","PROMO_ID","QUANTITY_SOLD","AMOUNT_SOLD")
INCLUDING NEW VALUES;

CREATE MATERIALIZED VIEW LOG ON
"SH"."PROMOTIONS"
WITH ROWID, SEQUENCE("PROMO_ID","PROMO_NAME","PROMO_CATEGORY")
INCLUDING NEW VALUES;

CREATE MATERIALIZED VIEW "DBA1"."MV$$_043F0000"
REFRESH FAST WITH ROWID
ENABLE QUERY REWRITE
AS SELECT SH.PROMOTIONS.PROMO_CATEGORY C1,
SH.PROMOTIONS.PROMO_NAME C2, COUNT(*)
M1 FROM SH.SALES, SH.PROMOTIONS WHERE SH.PROMOTIONS.PROMO_ID =
SH.SALES.PROMO_ID
AND (SH.PROMOTIONS.PROMO_CATEGORY = 'internet') GROUP BY
SH.PROMOTIONS.PROMO_CATEGORY,
SH.PROMOTIONS.PROMO_NAME;

begin
dbms_stats.gather_table_stats
('"DBA1"','"MV$$_043F0000"',NULL,dbms_stats.auto_sample_size);
end;
/
```

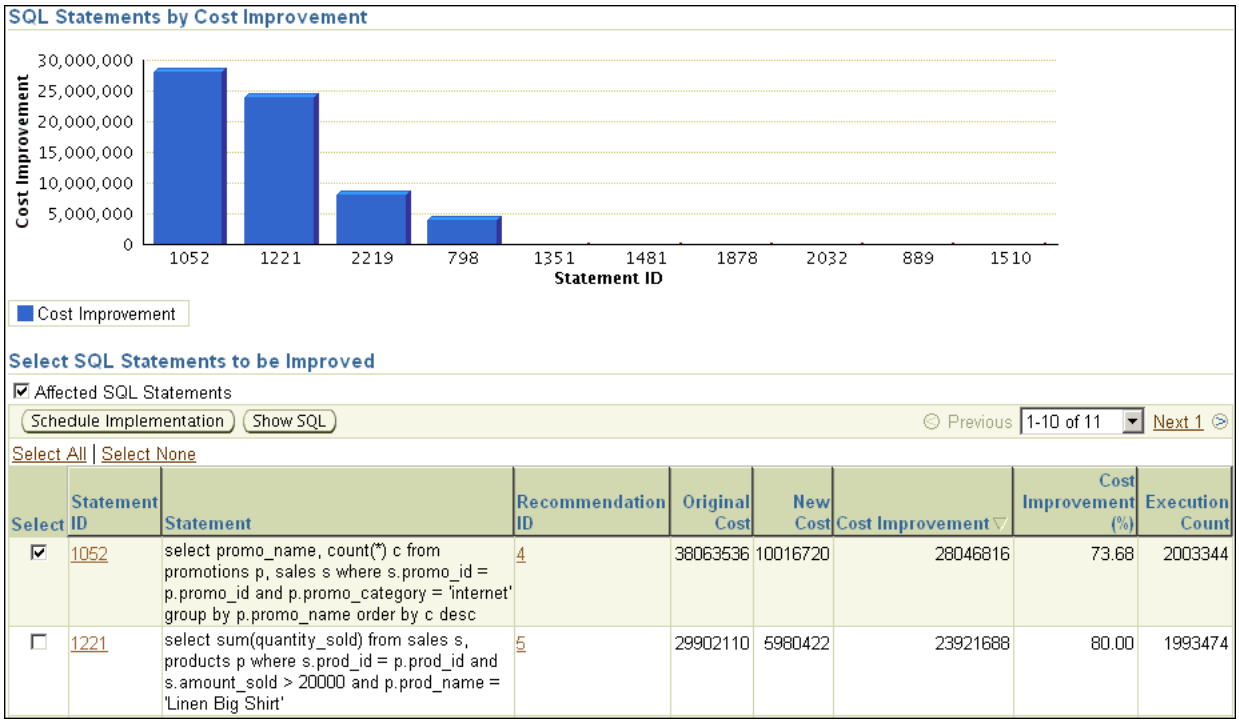
Reviewing the SQL Access Advisor Recommendations: SQL Statements

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 appears.



2. Use the SQL Statements by Cost Improvement chart to view SQL statements in the workload ordered by the cost improvement.

   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.

   Implementing the recommendation associated with the top SQL statement will have the biggest benefit to the total performance of the workload. In this example, implementing the recommendation with ID 4 will produce the biggest benefit, a cost improvement of 73.68%, for the SQL statement with ID 1052.

3. To view the SQL text of a recommendation, select the recommendation and click **Show SQL**.

   The Show SQL page for the selected recommendation appears.

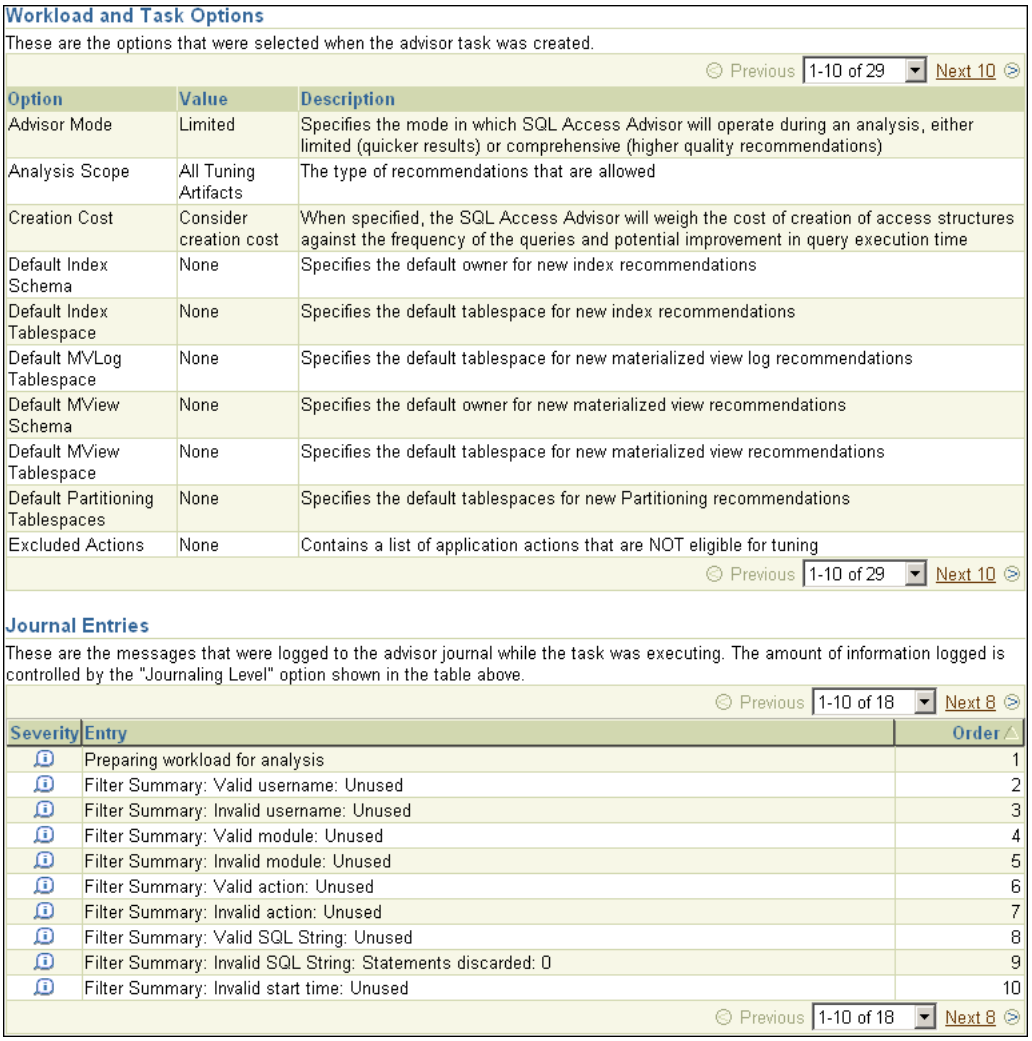
Reviewing the SQL Access Advisor Recommendations: Details

The Details subpage displays a list of all the workload and task options used in the analysis. You can also use this page 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:

- On the Results for Tasks page, click **Details**.

  The Details subpage appears.



Workload and Task Options

These are the options that were selected when the advisor task was created.

Previous 1-10 of 29 Next 10

Option	Value	Description
Advisor Mode	Limited	Specifies the mode in which SQL Access Advisor will operate during an analysis, either limited (quicker results) or comprehensive (higher quality recommendations)
Analysis Scope	All Tuning Artifacts	The type of recommendations that are allowed
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
Default Partitioning Tablespaces	None	Specifies the default tablespaces for new Partitioning recommendations
Excluded Actions	None	Contains a list of application actions that are NOT eligible for tuning

Previous 1-10 of 29 Next 10

Journal Entries

These are the messages that were logged to the advisor journal while the task was executing. The amount of information logged is controlled by the "Journaling Level" option shown in the table above.

Previous 1-10 of 18 Next 8

Severity	Entry	Order
	Preparing workload for analysis	1
	Filter Summary: Valid username: Unused	2
	Filter Summary: Invalid username: Unused	3
	Filter Summary: Valid module: Unused	4
	Filter Summary: Invalid module: Unused	5
	Filter Summary: Valid action: Unused	6
	Filter Summary: Invalid action: Unused	7
	Filter Summary: Valid SQL String: Unused	8
	Filter Summary: Invalid SQL String: Statements discarded: 0	9
	Filter Summary: Invalid start time: Unused	10

Previous 1-10 of 18 Next 8

Under Workload and Task Options, a list of options that were selected when the advisor task was created is displayed.

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

A SQL Access Advisor recommendation can range from a simple suggestion to a complex solution that requires partitioning a set of existing base tables and implementing a set of database objects such as indexes, materialized views, and materialized view logs. You can select the recommendations for implementation and schedule when the job should be executed.

> **Tip:** Before implementing the SQL Access Advisor recommendations, 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.

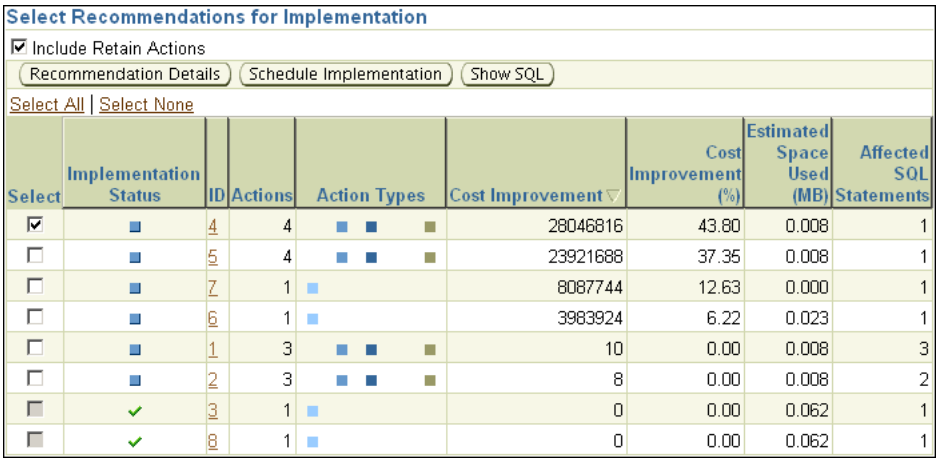
To implement the SQL Access Advisor recommendations:

1. On the Results for Tasks page, click **Recommendations**.

   The Recommendations subpage appears.

2. Under Select Recommendations for Implementation, select the recommendation you want to implement and then click **Schedule Implementation**.

   In the following example, the recommendation with ID value 4 is selected.



Select	Implementation Status	ID	Actions	Action Types	Cost Improvement	Cost Improvement (%)	Estimated Space Used (MB)	Affected SQL Statements
☑	■	4	4	■ ■ ■	28046816	43.80	0.008	1
☐	■	5	4	■ ■ ■	23921688	37.35	0.008	1
☐	■	7	1	■	8087744	12.63	0.000	1
☐	■	6	1	■	3983924	6.22	0.023	1
☐	■	1	3	■ ■ ■	10	0.00	0.008	3
☐	■	2	3	■ ■ ■	8	0.00	0.008	2
☐	✓	3	1	■	0	0.00	0.062	1
☐	✓	8	1	■	0	0.00	0.062	1

   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 the implementation job should stop if an error is encountered. Do one of the following:

   - To stop processing if an error occurs, select **Stop on Error**.
   - To continue processing even if an error occurs, deselect **Stop on Error**.

5. Under Scheduling Options, in the Schedule Type list, select a schedule type for the task and a maintenance window in which the task should run. Do one of the following:

   - Click **Standard**.

     This schedule type enables you to select a repeating interval and start time for the task. Complete the following steps:

     – Enter your time zone code in the **Time Zone** field or click the search icon to locate the code for your area.

     – In the Repeat list, select **Do Not Repeat** to perform the task only once, or select a unit of time and enter the number of units in the **Interval** field.

     – Under Start, select **Immediately** to start the task now, or **Later** to schedule the task to start at a time specified using the **Date** and **Time** fields.

   - Click **Use predefined schedule**.

     This schedule type enables you to select an existing schedule. Do one of the following:

     – In the **Schedule** field, enter the name of the schedule to be used for the task.

     – To search for a schedule, click the search icon.

The Search and Select: Schedule dialog box appears.

Select the desired schedule and click **Select**. The selected schedule now appears in the **Schedule** field.

- Click **Standard using PL/SQL for repeated interval**.

  This schedule type enables you to select a repeating interval and an execution window for the task. Complete the following steps:

  - Enter your time zone code in the **Time Zone** field or click the search icon to locate the code for your area.
  - Under Available to Start, select **Immediately** to start the task now, or **Later** to schedule the task to start at a time specified using the **Date** and **Time** fields.
  - In the Repeat list, select **Do Not Repeat** to perform the task only once, or select a unit of time and enter the number of units in the **Interval** field.
  - In the **Repeated Interval** field, enter a PL/SQL schedule expression, such as `SYSDATE+1`.
  - Under Not Available After, select **No End Date** to indicate that there is no end date for the execution window, or **Specified End Date** to specify an end date using the **Date** and **Time** fields.

- Click **Use predefined window**.

  This schedule type enables you to select an existing window. Select **Stop on Window Close** to stop the job when the window closes. Do one of the following:

  - In the **Window** field, enter the name of the window to be used for the task.
  - To search for a window, click the search icon.

    The Search and Select: Window and Window Groups dialog box appears.

    Select the desired window and click **Select**. The selected window now appears in the **Schedule** field.

- Click **Event**.

  Complete the following steps:

  - Enter your time zone code in the **Time Zone** field or click the search icon to locate the code for your area.
  - Under Event Parameters, enter values in the **Queue Name** and **Condition** fields.
  - Under Start, select **Immediately** to start the task now, or **Later** to schedule the task to start at a time specified using the **Date** and **Time** fields.
  - Under Not Available After, select **No End Date** to indicate that there is no end date for the execution window, or **Specified End Date** to specify an end date using the **Date** and **Time** fields.

- Click **Calendar**.

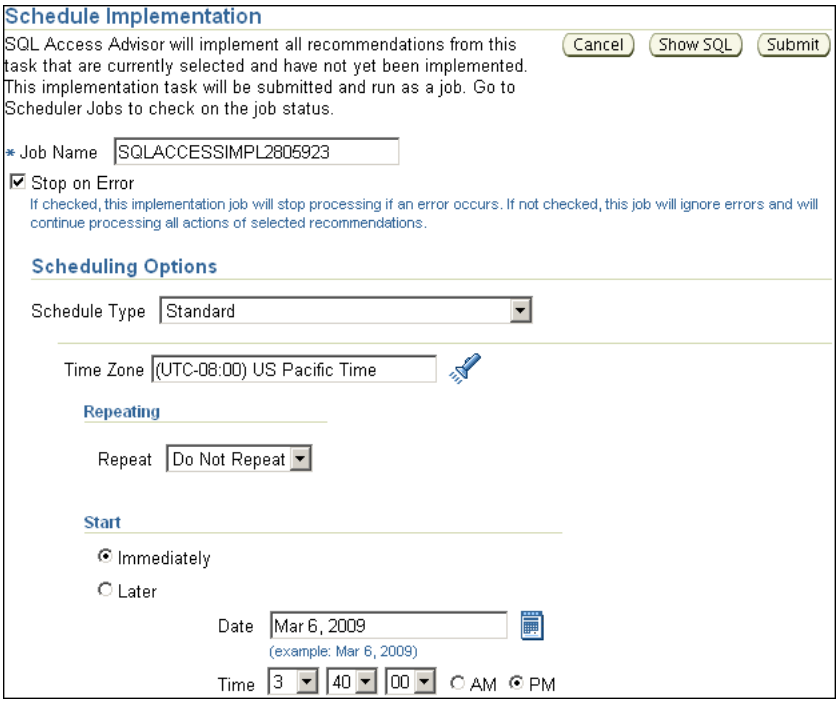
  Complete the following steps:

  - Enter your time zone code in the **Time Zone** field or click the search icon to locate the code for your area.
  - Under Calendar Expression, enter a calendar expression.

– Under Start, select **Immediately** to start the task now, or **Later** to schedule the task to start at a time specified using the **Date** and **Time** fields.

– Under Not Available After, select **No End Date** to indicate that there is no end date for the execution window, or **Specified End Date** to specify an end date using the **Date** and **Time** fields.

In this example, **Standard** is selected for schedule type. The job will not repeat and is scheduled to start immediately.

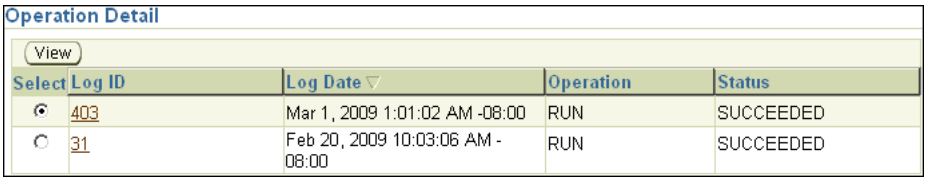


6. Optionally, click **Show SQL** to view the SQL text for the job.
7. To submit the job, click **Submit**.
8. Do one of the following, depending on whether the job is scheduled to start immediately or later:
   - If you submitted the job immediately, and if the Results for Task page is shown, then click the link in the **Scheduler Job** field to display the View Job page. Go to Step 10.
   - If the job is scheduled to start at a later time, then proceed to Step 9.
9. Complete the following steps:
   a. On the Server page, under Oracle 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, check the status of the operation.



11. Optionally, select the operation and click **View**.

    The Operation Detail page appears.

    This page contains information (such as start date and time, run duration, CPU time used, and session ID) that you can use when troubleshooting.

12. Optionally, from the Database Home page, click **Schema**.

    The Schema subpage appears.

    On this page you can verify that the access structure recommended by 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.

Index

A

B

C

S

T

U

W