Oracle® Database

Testing Guide

23ai
F47494-04
October 2024

Oracle Database Testing Guide, 23ai

F47494-04

Primary Authors: Sunil Surabhi, Roopesh Ashok Kumar

Contributing Authors: Prakash Jashnani

Contributors: Daniel Suherman, Ashish Agrawal, Waleed Ahmed, Helen Altmar, Lance Ashdown, Pete Belknap, Supiti Buranawatanachoke, Romain Colle, Karl Dias, Kurt Engeleiter, Leonidas Galanis, Veeranjaneyulu Goli, Prabhaker Gongloor, Prakash Gupta, Shantanu Joshi, Prathiba Kalirengan, Karen McKeen, Mughees Minhas, Konstantinos Morfonios, Valarie Moore, Ravi Pattabhi, Bert Rich, Yujun Wang, Keith Wong, Qinyi Wu, Khaled Yagoub, Hailing Yu, Yury Berezin

Contents

13 Analyzing Captured and Replayed Workloads

14 Using Workload Intelligence

15 Using Consolidated Database Replay

Preface

This preface contains the following topics:

Audience

This document provides information about how to assure the integrity of database changes and manage test data using Oracle Real Application Testing. This document is intended for database administrators, application designers, and programmers who are responsible for performing real-world testing of 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 that have purchased support 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.

Diversity and Inclusion

Oracle is fully committed to diversity and inclusion. Oracle respects and values having a diverse workforce that increases thought leadership and innovation. As part of our initiative to build a more inclusive culture that positively impacts our employees, customers, and partners, we are working to remove insensitive terms from our products and documentation. We are also mindful of the necessity to maintain compatibility with our customers' existing technologies and the need to ensure continuity of service as Oracle's offerings and industry standards evolve. Because of these technical constraints, our effort to remove insensitive terms is ongoing and will take time and external cooperation.

Conventions

The following text conventions are used in this document:

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

1

Introduction to Oracle Database Testing

The Oracle Real Application Testing option of Oracle Database help you to securely assure the integrity of database changes and to manage test data.

Oracle Real Application Testing option enables you to perform real-world testing of Oracle Database. By capturing production workloads and assessing the impact of system changes on these workloads before production deployment, Oracle Real Application Testing minimizes the risk of instabilities associated with system changes. SQL Performance Analyzer and Database Replay are key components of Oracle Real Application Testing. Depending on the nature and impact of the system change being tested, and on the type of system the test will be performed, you can use either or both components to perform your testing.

This chapter contains the following sections:

> **Note:**
>
> The use of SQL Performance Analyzer and Database Replay requires the Oracle Real Application Testing licensing option. For more information, see *Oracle Database Licensing Information User Manual*.

SQL Performance Analyzer

System changes—such as a upgrading a database or adding an index—may cause changes to execution plans of SQL statements, resulting in a significant impact on SQL performance. In some cases, the system changes may cause SQL statements to regress, resulting in performance degradation. In other cases, the system changes may improve SQL performance. Being able to accurately forecast the potential impact of system changes on SQL performance enables you to tune the system beforehand, in cases where the SQL statements regress, or to validate and measure the performance gain in cases where the performance of the SQL statements improves.
SQL Performance Analyzer automates the process of assessing the overall effect of a change on the full SQL workload by identifying performance divergence for each SQL statement. A report that shows the net impact on the workload performance due to the change is provided. For regressed SQL statements, SQL Performance Analyzer also provides appropriate execution plan details along with tuning recommendations. As a result, you can remedy any negative outcome before the end users are affected. Furthermore, you can validate—with significant time and cost savings—that the system change to the production environment will result in net improvement.

You can use the SQL Performance Analyzer to analyze the impact on SQL performance of any type of system changes, including:

- Database upgrade

- Database consolidation testing for pluggable databases (PDBs) and manual schema consolidation
- Configuration changes to the operating system or hardware
- Schema changes
- Changes to database initialization parameters
- Refreshing optimizer statistics
- Validating SQL tuning actions

> **See Also:**
>
> - Introduction to SQL Performance Analyzer for information about using SQL Performance Analyzer

Database Replay

Before system changes are made, such as hardware and software upgrades, extensive testing is usually performed in a test environment to validate the changes. However, despite the testing, the new system often experiences unexpected behavior when it enters production because the testing was not performed using a realistic workload. The inability to simulate a realistic workload during testing is one of the biggest challenges when validating system changes.
Database Replay enables realistic testing of system changes by essentially re-creating the production workload environment on a test system. Using Database Replay, you can capture a workload on the production system and replay it on a test system with the exact timing, concurrency, and transaction characteristics of the original workload. This enables you to fully assess the impact of the change, including undesired results, new contention points, or plan regressions. Extensive analysis and reporting is provided to help identify any potential problems, such as new errors encountered and performance divergence.

Database Replay captures the workload of external database clients at the database level and has negligible performance overhead. Capturing the production workload eliminates the need to develop simulation workloads or scripts, resulting in significant cost reduction and time savings. By using Database Replay, realistic testing of complex applications that previously took months using load simulation tools can now be completed in days. This enables you to rapidly test changes and adopt new technologies with a higher degree of confidence and at lower risk.

You can use Database Replay to test any significant system changes, including:

- Database and operating system upgrades
- Database consolidation testing for PDBs and manual schema consolidation
- Authoring and experimenting with various scenarios using workload scale-up
- Configuration changes, such as conversion of a database from a single instance to an Oracle Real Application Clusters (Oracle RAC) environment
- Storage, network, and interconnect changes
- Operating system and hardware migrations

See Also:

- Introduction to Database Replay for information about using Database Replay

Part I
SQL Performance Analyzer

SQL Performance Analyzer enables you to assess the impact of system changes on the response time of SQL statements.

Part I covers SQL Performance Analyzer and contains the following chapters:

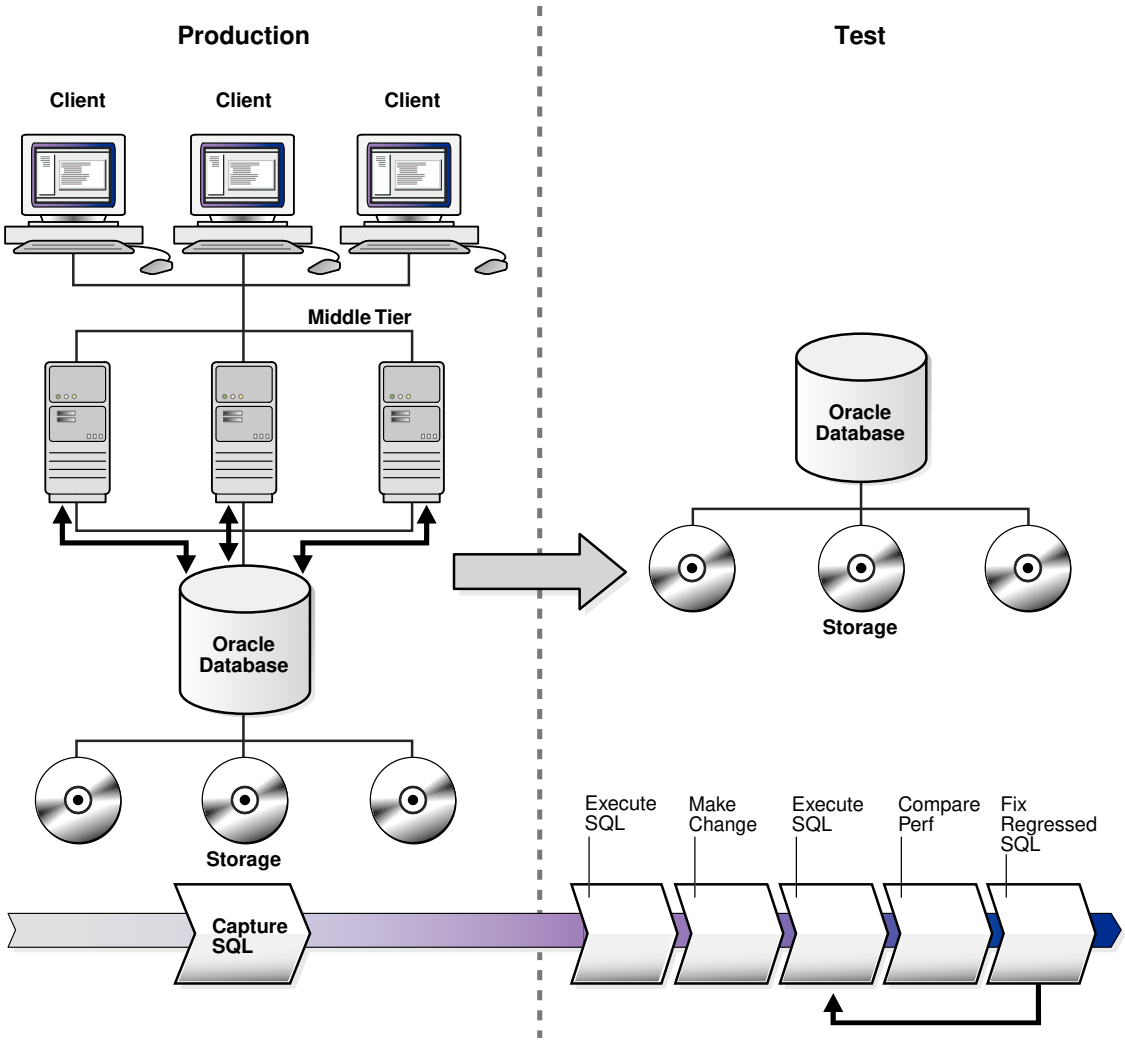
2

Introduction to SQL Performance Analyzer

You can run SQL Performance Analyzer on a production system or a test system that closely resembles the production system. Testing a system change on a production system will impact the system's throughput because SQL Performance Analyzer must execute the SQL statements that you are testing. Any global changes made on the system to test the performance effect may also affect other users of the system. If the system change does not impact many sessions or SQL statements, then running SQL Performance Analyzer on the production system may be acceptable. However, for systemwide changes—such as a database upgrade—using a production system is not recommended. Instead, consider running SQL Performance Analyzer on a separate test system so that you can test the effects of the system change without affecting the production system. Using a test system also ensures that other workloads running on the production system will not affect the analysis performed by SQL Performance Analyzer. Running SQL Performance Analyzer on a test system is the recommended approach and the methodology described here. If you choose to run the SQL Performance Analyzer on the production system, then substitute the production system for the test system where applicable.

Analyzing the SQL performance effect of system changes using SQL Performance Analyzer involves the following steps, as illustrated in Figure 2-1:

Figure 2-1 SQL Performance Analyzer Workflow



1. Capture the SQL workload that you intend to analyze and store it in a SQL tuning set, as described in "Capturing the SQL Workload".
2. If you plan to use a test system separate from your production system, then perform the following steps:
   a. Set up the test system to match the production environment as closely as possible.
   b. Transport the SQL tuning set to the test system.
3. On the test system, create a SQL Performance Analyzer task, as described in "Creating a SQL Performance Analyzer Task".
4. Build the pre-change SQL trial by test executing or generating execution plans for the SQL statements stored in the SQL tuning set, as described in "Measuring the Pre-Change SQL Performance"
5. Perform the system change, as described in "Making a System Change"
6. Build the post-change SQL trial by re-executing the SQL statements in the SQL tuning set on the post-change test system, as described in "Measuring the Post-Change SQL Performance"

7. Compare and analyze the pre-change and post-change versions of performance data, and generate a report to identify the SQL statements that have improved, remained unchanged, or regressed after the system change, as described in "Comparing Performance Measurements"
8. Tune any regressed SQL statements that are identified, as described in "Fixing Regressed SQL Statements".
9. Ensure that the performance of the tuned SQL statements is acceptable by repeating steps 6 through 8 until your performance goals are met.

   For each comparison, you can use any previous SQL trial as the pre-change SQL trial and the current SQL trial as the post-change SQL trial. For example, you may want to compare the first SQL trial to the current SQL trial to assess the total change, or you can compare the most recent SQL trial to the current SQL trial to assess just the most recent change.

> **Note:**
>
> Oracle Enterprise Manager provides automated workflows for steps 3 through 9 to simplify this process.

> **Note:**
>
> Data visibility and privilege requirements may differ when using SQL Performance Analyzer with pluggable databases (PDBs).

> **See Also:**
>
> - "Setting Up the Test System"
> - For information about how manageability features—including SQL Performance Analyzer—work in a multitenant container database (CDB), see *Oracle Database Administrator's Guide*

Capturing the SQL Workload

Before running SQL Performance Analyzer, capture a set of SQL statements on the production system that represents the SQL workload which you intend to analyze.

The captured SQL statements should include the following information:

- SQL text
- Execution environment
  - SQL binds, which are bind values needed to execute a SQL statement and generate accurate execution statistics
  - Parsing schema under which a SQL statement can be compiled
  - Compilation environment, including initialization parameters under which a SQL statement is executed

- Number of times a SQL statement was executed

Capturing a SQL workload has a negligible performance impact on your production system and should not affect throughput. A SQL workload that contains more SQL statements will better represent the state of the application or database. This will enable SQL Performance Analyzer to more accurately forecast the potential impact of system changes on the SQL workload. Therefore, you should capture as many SQL statements as possible. Ideally, you should capture all SQL statements that are either called by the application or are running on the database.

You can store captured SQL statements in a SQL tuning set and use it as an input source for SQL Performance Analyzer. A SQL tuning set is a database object that includes one or more SQL statements, along with their execution statistics and execution context. SQL statements can be loaded into a SQL tuning set from different sources, including the cursor cache, Automatic Workload Repository (AWR), SQL trace files, and existing SQL tuning sets. Capturing a SQL workload using a SQL tuning set enables you to:

- Store the SQL text and any necessary auxiliary information in a single, persistent database object
- Populate, update, delete, and select captured SQL statements in the SQL tuning set
- Load and merge content from various data sources, such as the Automatic Workload Repository (AWR) or the cursor cache
- Export the SQL tuning set from the system where the SQL workload is captured and import it into another system
- Reuse the SQL workload as an input source for other advisors, such as the SQL Tuning Advisor and the SQL Access Advisor

> **See Also:**
>
> - *Oracle Database 2 Day + Performance Tuning Guide* for information about creating SQL tuning sets using Oracle Enterprise Manager
> - *Oracle Database SQL Tuning Guide* for information about creating SQL tuning sets using APIs

Setting Up the Test System

After you have captured the SQL workload into a SQL tuning set on the production system, you can conduct SQL Performance Analyzer analysis on the same database where the workload was captured or on a different database. Because the analysis is resource-intensive, it is recommended that you capture the workload on a production database and transport it to a separate test database where the analysis can be performed. To do so, export the SQL tuning set from the production system and import it into a separate system where the system change will be tested.

There are many ways to create a test database. For example, you can use the `DUPLICATE` command of Recovery Manager (RMAN), Oracle Data Pump, or transportable tablespaces. Oracle recommends using RMAN because it can create the test database from pre-existing backups or from the active production datafiles. The production and test databases can reside on the same host or on different hosts.

You should configure the test database environment to match the database environment of the production system as closely as possible. In this way, SQL Performance Analyzer can more accurately forecast the effect of the system change on SQL performance.

After the test system is properly configured, export the SQL tuning set from the production system to a staging table, then import it from the staging table into the test system.

See Also:

- *Oracle Database Backup and Recovery User's Guide* for information about duplicating databases using RMAN
- *Oracle Database 2 Day + Performance Tuning Guide* for information about transporting SQL tuning sets using Oracle Enterprise Manager
- *Oracle Database SQL Tuning Guide* for information about transporting SQL tuning sets using APIs

Creating a SQL Performance Analyzer Task

After the SQL workload is captured and transported to the test system, and the initial database environment is properly configured, you can run SQL Performance Analyzer to analyze the effects of a system change on SQL performance.

To run SQL Performance Analyzer, you must first create a SQL Performance Analyzer task. A task is a container that encapsulates all of the data about a complete SQL Performance Analyzer analysis. A SQL Performance Analyzer analysis comprises of at least two SQL trials and a comparison. A SQL trial encapsulates the execution performance of a SQL tuning set under specific environmental conditions. When creating a SQL Performance Analyzer task, you will need to select a SQL tuning set as its input source. When building SQL trials using the test execute or explain plan methods, the SQL tuning set will be used as the source for SQL statements. The SQL Performance Analyzer analysis will show the impact of the environmental differences between the two trials.

See Also:

- Creating an Analysis Task for information about how to create a SQL Performance Analyzer task

Measuring the Pre-Change SQL Performance

Create a pre-change SQL trial before making the system change. You can use the following methods to generate the performance data needed for a SQL trial with SQL Performance Analyzer:

- Test execute

  This method test executes SQL statements through SQL Performance Analyzer. This can be done on the database running SPA Performance Analyzer or on a remote database.

- Explain plan

This method generates execution plans only for SQL statements through SQL Performance Analyzer. This can be done on the database running SPA Performance Analyzer or on a remote database. Unlike the `EXPLAIN PLAN` statement, SQL trials using the explain plan method take bind values into account and generate the actual execution plan.

- Convert SQL tuning set

  This method converts the execution statistics and plans stored in a SQL tuning set. This is only supported for APIs.

The test execute method runs each of the SQL statements contained in the workload to completion. During execution, SQL Performance Analyzer generates execution plans and computes execution statistics for each SQL statement in the workload. Each SQL statement in the SQL tuning set is executed separately from other SQL statements, without preserving their initial order of execution or concurrency. This is done at least twice for each SQL statement, for as many times as possible until the execution times out (up to a maximum of 10 times). The first execution is used to warm the buffer cache. All subsequent executions are then used to calculate the run-time execution statistics for the SQL statement based on their averages. The actual number of times that the SQL statement is executed depends on how long it takes to execute the SQL statement. Long-running SQL statement will only be executed a second time, and the execution statistics from this execution will be used. Other (faster-running) SQL statements are executed multiple times, and their execution statistics are averaged over these executions (statistics from the first execution are not used in the calculation). By averaging statistics over multiple executions, SQL Performance Analyzer can calculate more accurate execution statistics for each SQL statement. To avoid a potential impact to the database, DDLs are not supported. By default, only the query portion of DMLs is executed. Using APIs, you can execute the full DML by using the `EXECUTE_FULLDML` task parameter. Parallel DMLs are not supported and the query portion is not executed unless the parallel hints are removed.

Depending on its size, executing a SQL workload can be time and resource intensive. With the explain plan method, you can choose to generate execution plans only, without collecting execution statistics. This technique shortens the time to run the trial and lessens the effect on system resources, but a comprehensive performance analysis is not possible because only the execution plans will be available during the analysis. However, unlike generating a plan with the `EXPLAIN PLAN` command, SQL Performance Analyzer provides bind values to the optimizer when generating execution plans, which provides a more reliable prediction of what the plan will be when the SQL statement is executed.

In both cases, you can execute the SQL workload remotely on a separate database using a database link. SQL Performance Analyzer will establish a connection to the remote database using the database link, execute the SQL statements on that database, collect the execution plans and run-time statistics for each SQL statement, and store the results in a SQL trial on the local database that can be used for later analysis. This method is useful in cases where you want to:

- Test a database upgrade
- Execute the SQL workload on a system running another version of Oracle Database
- Store the results from the SQL Performance Analyzer analysis on a separate test system
- Perform testing on multiple systems with different hardware configurations
- Use the newest features in SQL Performance Analyzer even if you are using an older version of Oracle Database on your production system

Once the SQL workload is executed, the resulting execution plans and run-time statistics are stored in a SQL trial.

You can also build a SQL trial using the execution statistics and plan stored in a SQL tuning set. While this method is only supported for APIs, it may be useful in cases where you have another method to run your workload (such as Database Replay or another application testing tool), and you do not need SQL Performance Analyzer to drive the workload on the test system. In such cases, if you capture a SQL tuning set during your test runs, you can build SQL trials from these SQL tuning sets using SQL Performance Analyzer to view a more comprehensive analysis report. Unlike a standard SQL Performance Analyzer report—which has only one execution plan in each trial and one set of execution statistics generated by executing the SQL statement with one set of binds—you can generate a report that compares SQL trials built from SQL tuning sets that show all execution plans from both trials with potentially many different sets of binds across multiple executions.

See Also:

- Creating a Pre-Change SQL Trial for information about how to measure the pre-change performance
- Testing a Database Upgrade for information about executing a SQL workload on a remote system to test a database upgrade

Making a System Change

Make the change whose effect on SQL performance you intend to measure. SQL Performance Analyzer can analyze the effect of many types of system changes. For example, you can test a database upgrade, new index creation, initialization parameter changes, or optimizer statistics refresh. If you are running SQL Performance Analyzer on the production system, then consider making a change using a private session to avoid affecting the rest of the system.

Measuring the Post-Change SQL Performance

After performing the system change, create a post-change SQL trial. It is highly recommended that you create the post-change SQL trial using the same method as the pre-change SQL trial. Once built, the post-change SQL trial represents a new set of performance data that can be used to compare to the pre-change version. The results are stored in a new, or post-change, SQL trial.

See Also:

- Creating a Post-Change SQL Trial for information about how to measure the post-change performance

Comparing Performance Measurements

SQL Performance Analyzer compares the performance of SQL statements before and after the change and produces a report identifying any changes in execution plans or performance of the SQL statements.

SQL Performance Analyzer measures the impact of system changes both on the overall execution time of the SQL workload and on the response time of every individual SQL

statement in the workload. By default, SQL Performance Analyzer uses elapsed time as a metric for comparison. Alternatively, you can choose the metric for comparison from a variety of available SQL run-time statistics, including:

- CPU time
- User I/O time
- Buffer gets
- Physical I/O
- Optimizer cost
- I/O interconnect bytes
- Any combination of these metrics in the form of an expression

If you chose to generate explain plans only in the SQL trials, then SQL Performance Analyzer will use the optimizer cost stored in the SQL execution plans.

Once the comparison is complete, the resulting data is generated into a SQL Performance Analyzer report that compares the pre-change and post-change SQL performance. The SQL Performance Analyzer report can be viewed as an HTML, text, or active report. Active reports provides in-depth reporting using an interactive user interface that enables you to perform detailed analysis even when disconnected from the database or Oracle Enterprise Manager.

See Also:

- Comparing SQL Trials for information about comparing performance measurements and reporting

Fixing Regressed SQL Statements

If the performance analysis performed by SQL Performance Analyzer reveals regressed SQL statements, then you can make changes to remedy the problem. For example, you can fix regressed SQL by running SQL Tuning Advisor or using SQL plan baselines. You can then repeat the process of executing the SQL statements and comparing its performance to the first execution. Repeat these steps until you are satisfied with the outcome of the analysis.

See Also:

- Comparing SQL Trials for information about fixing regressed SQL statements

3
Creating an Analysis Task

Once you have captured a SQL workload that you want to analyze into a SQL tuning set (STS), you can run SQL Performance Analyzer to analyze the effects of a system change on SQL performance. To run SQL Performance Analyzer, you must first create a SQL Performance Analyzer task. A task is a container that encapsulates all of the data about a complete SQL Performance Analyzer analysis. A SQL Performance Analyzer analysis comprises of at least two SQL trials and a comparison. A SQL trial captures the execution performance of a SQL tuning set under specific environmental conditions and can be generated automatically using SQL Performance Analyzer by one of the following methods:

- Test executing SQL statements
- Generating execution plans for SQL statements
- Referring to execution statistics and plans captured in a SQL tuning set

When creating a SQL Performance Analyzer task, you will need to select a SQL tuning set as its input source. The SQL tuning set will be used as the source for test executing or generating execution plans for SQL trials. Thus, performance differences between trials are caused by environmental differences.

This chapter describes how to create a SQL Performance Analyzer task and contains the following topics:

- Creating an Analysis Task Using Enterprise Manager
- Creating an Analysis Task Using APIs
- Configuring an Analysis Task Using APIs

> **Note:**
>
> The primary interface for running SQL Performance Analyzer is Oracle Enterprise Manager. If for some reason Oracle Enterprise Manager is unavailable, you can run SQL Performance Analyzer using the `DBMS_SQLPA` PL/SQL package.

> **See Also:**
>
> "Creating a SQL Performance Analyzer Task"

Creating an Analysis Task Using Enterprise Manager

There are several workflows available in Oracle Enterprise Manager for creating a SQL Performance Analyzer task.

Before running SQL Performance Analyzer, capture the SQL workload to be used in the performance analysis into a SQL tuning set on the production system, then transport it to the

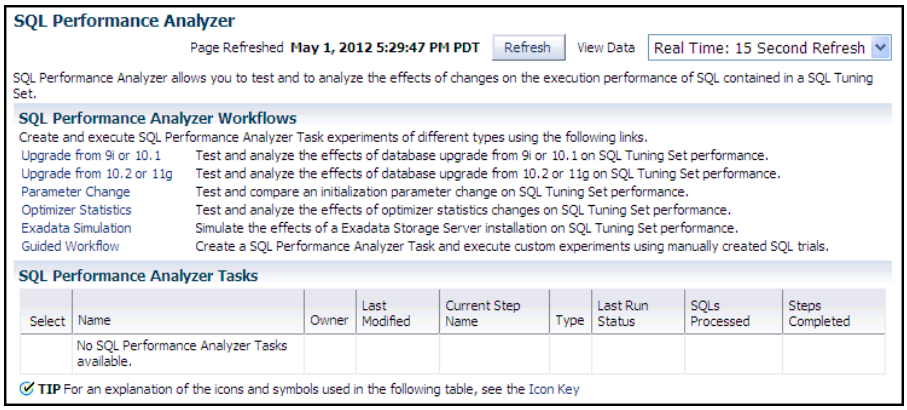
test system where the performance analysis will be performed, as described in "Capturing the SQL Workload".

To create an analysis task using Enterprise Manager:

1. From the **Performance** menu, select **SQL**, then **SQL Performance Analyzer**.

   If the Database Login page appears, then log in as a user with administrator privileges.

   The SQL Performance Analyzer Home page appears.



2. Under SQL Performance Analyzer Workflows, select the workflow for creating the desired type of analysis task:

   - Upgrade from 9*i* or 10.1

     Use the upgrade from 9*i* or 10.1 workflow to test a database upgrade from Oracle9*i* Database or Oracle Database 10*g* Release 1 to Oracle Database 10*g* Release 2 and newer releases, as described in "Upgrading from Oracle9*i* Database and Oracle Database 10*g* Release 1".

   - Upgrade from 10.2 or 11*g*

     Use the upgrade from 10.2 or 11*g* workflow to test a database upgrade from Oracle Database 10*g* Release 2 or Oracle Database 11*g* to a later release, as described in "Upgrading from Oracle Database 10*g* Release 2 and Newer Releases".

   - Parameter Change

     Use the parameter change workflow to determine how a database initialization parameter change will affect SQL performance, as described in "Using the Parameter Change Workflow".

   - Optimizer Statistics

     Use the optimizer statistics workflow to analyze how changes to optimizer statistics will affect SQL performance, as described in "Using the Optimizer Statistics Workflow".

   - Exadata Simulation

     Use the Exadata simulation workflow to simulate how using Oracle Exadata will affect SQL performance, as described in "Using the Exadata Simulation Workflow".

   - Guided workflow

     Use the guided workflow to compare SQL performance for all other types of system changes, as described in "Using the Guided Workflow".

Using the Parameter Change Workflow

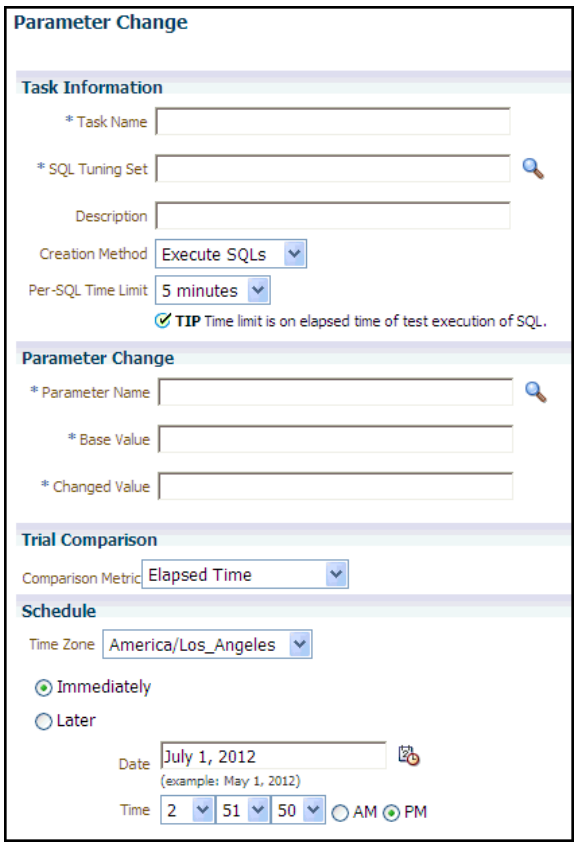
The parameter change workflow enables you to test the performance effect on a SQL workload when you change the value of a single environment initialization parameter. For example, you can compare SQL performance by setting the `OPTIMIZER_FEATURES_ENABLE` initialization parameter to 10.2.0.4 and 12.1.0.1.
After you select a SQL tuning set and a comparison metric, SQL Performance Analyzer creates a task and performs a trial with the initialization parameter set to the original value. SQL Performance Analyzer then performs a second trial with the parameter set to the changed value by issuing an `ALTER SESSION` statement. The impact of the change is thus contained locally to the testing session. Any regression or change in performance is reported in a system-generated SQL Performance Analyzer report.

> **Note:**
>
> To create an analysis task for other types of system changes, use the guided workflow instead, as described in "Using the Guided Workflow".

To use the SQL Performance Analyzer parameter change workflow:

1. On the SQL Performance Analyzer Home page, under SQL Performance Analyzer Workflows, click **Parameter Change**.

   The Parameter Change page appears.



2. In the Task Name field, enter the name of the task.
3. In the SQL Tuning Set field, enter the name of the SQL tuning set that contains the SQL workload to be analyzed.

   Alternatively, click the search icon to search for a SQL tuning set using the Search and Select: SQL Tuning Set window.

   The selected SQL tuning set now appears in the SQL Tuning Set field.
4. In the Description field, optionally enter a description of the task.
5. In the Creation Method list, determine how the SQL trial is created and what contents are generated by performing one of the following actions:
   - Select **Execute SQLs**.

     The SQL trial generates both execution plans and statistics for each SQL statement in the SQL tuning set by actually running the SQL statements.
   - Select **Generate Plans**.

     The SQL trial invokes the optimizer to create execution plans only without actually running the SQL statements.
6. In the Per-SQL Time Limit list, determine the time limit for SQL execution during the trial by performing one of the following actions:
   - Select **5 minutes**.

     The execution will run each SQL statement in the SQL tuning set up to 5 minutes and gather performance data.
   - Select **Unlimited**.

     The execution will run each SQL statement in the SQL tuning set to completion and gather performance data. Collecting execution statistics provides greater accuracy in the performance analysis but takes a longer time. Using this setting is not recommended because the task may be stalled by one SQL statement for a prolonged time period.
   - Select **Customize** and enter the specified number of seconds, minutes, or hours.
7. In the Parameter Change section, complete the following steps:
   a. In the Parameter Name field, enter the name of the initialization parameter whose value you want to modify, or click the Search icon to select an initialization parameter using the Search and Select: Initialization Parameters window.
   b. In the Base Value field, enter the current value of the initialization parameter.
   c. In the Changed Value field, enter the new value of the initialization parameter.
8. In the Comparison Metric list, select the comparison metric to use for the analysis:
   - If you selected **Generate Plans** in Step 5, then select **Optimizer Cost**.
   - If you selected **Execute SQLs** in Step 5, then select one of the following options:
     - **Elapsed Time**
     - **CPU Time**
     - **User I/O Time**
     - **Buffer Gets**
     - **Physical I/O**
     - **Optimizer Cost**

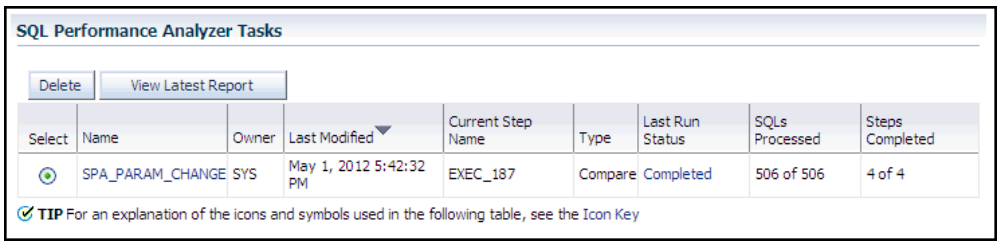
- **I/O Interconnect Bytes**

To perform the comparison analysis by using more than one comparison metric, perform separate comparison analyses by repeating this procedure using different metrics.

9. In the Schedule section:
   a. In the Time Zone list, select your time zone code.
   b. 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.
10. Click **Submit**.

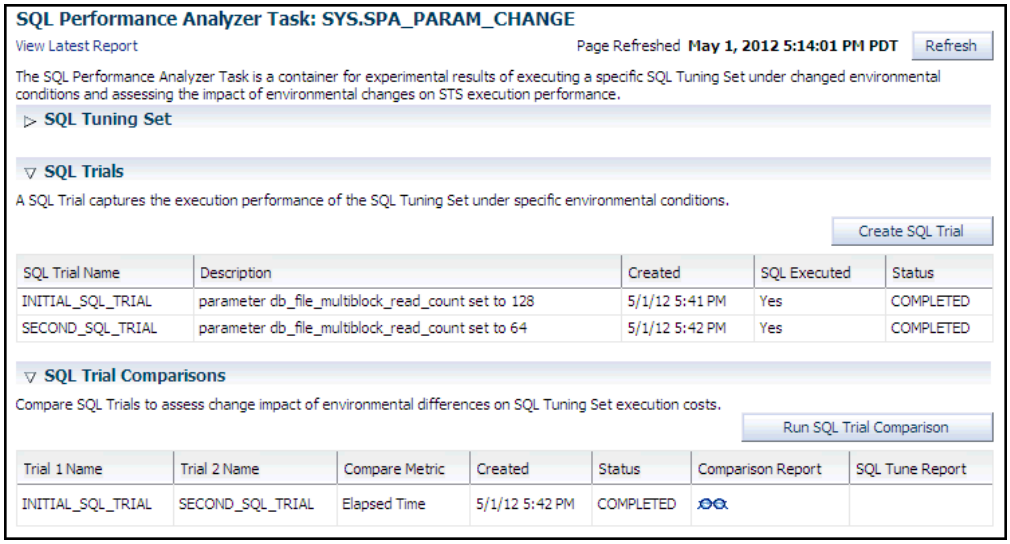
    The SQL Performance Analyzer Home page appears.

    In the SQL Performance Analyzer Tasks section, the status of this task is displayed. To refresh the status icon, click **Refresh**. After the task completes, the Status field changes to Completed.



11. In the SQL Performance Analyzer Tasks section, select the task and click the link in the Name column.

    The SQL Performance Analyzer Task page appears.



This page contains the following sections:

- SQL Tuning Set

  This section summarizes information about the SQL tuning set, including its name, owner, description, and the number of SQL statements it contains.

- SQL Trials

  This section includes a table that lists the SQL trials used in the SQL Performance Analyzer task.

- SQL Trial Comparisons

  This section contains a table that lists the results of the SQL trial comparisons

12. Click the icon in the Comparison Report column.

    The SQL Performance Analyzer Task Result page appears.

13. Review the results of the performance analysis, as described in "Reviewing the SQL Performance Analyzer Report Using Oracle Enterprise Manager".

14. In cases when regression are identified, click the icon in the SQL Tune Report column to view a SQL tuning report.

Using the Optimizer Statistics Workflow

The optimizer statistics workflow enables you to analyze the effects of optimizer statistics changes on the performance of a SQL workload.

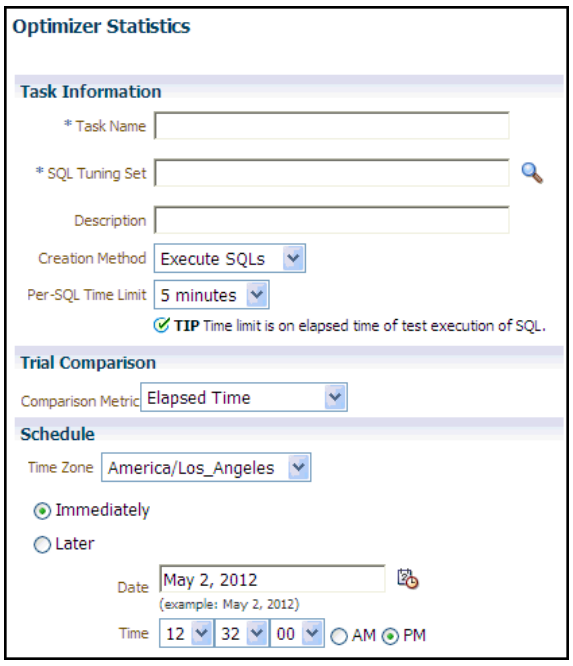
The

SQL Performance Analyzer tests the effect of new optimizer statistics by enabling pending optimizer statistics in the testing session. The first SQL trial measures the baseline SQL tuning set performance; the second SQL trial uses the pending optimizer statistics. You can then run a comparison report for the two SQL trials.

To use the optimizer statistics workflow:

1. On the SQL Performance Analyzer Home page, under SQL Performance Analyzer Workflows, click **Optimizer Statistics**.

   The Optimizer Statistics page appears.



2. In the Task Name field, enter the name of the task.
3. In the SQL Tuning Set field, enter the name of the SQL tuning set that contains the SQL workload to be analyzed.

   Alternatively, click the search icon to search for a SQL tuning set using the Search and Select: SQL Tuning Set window.

   The selected SQL tuning set now appears in the SQL Tuning Set field.
4. In the Description field, optionally enter a description of the task.
5. In the Creation Method list, determine how the SQL trial is created and what contents are generated by performing one of the following actions:
   - Select **Execute SQLs**.

     The SQL trial generates both execution plans and statistics for each SQL statement in the SQL tuning set by actually running the SQL statements.
   - Select **Generate Plans**.

     The SQL trial invokes the optimizer to create execution plans only without actually running the SQL statements.
6. In the Per-SQL Time Limit list, determine the time limit for SQL execution during the trial by performing one of the following actions:
   - Select **5 minutes**.

     The execution will run each SQL statement in the SQL tuning set up to 5 minutes and gather performance data.
   - Select **Unlimited**.

     The execution will run each SQL statement in the SQL tuning set to completion and gather performance data. Collecting execution statistics provides greater accuracy in the performance analysis but takes a longer time. Using this setting is not recommended because the task may be stalled by one SQL statement for a prolonged time period.
   - Select **Customize** and enter the specified number of seconds, minutes, or hours.
7. In the Comparison Metric list, select the comparison metric to use for the comparison analysis:
   - **Elapsed Time**
   - **CPU Time**
   - **User I/O Time**
   - **Buffer Gets**
   - **Physical I/O**
   - **Optimizer Cost**
   - **I/O Interconnect Bytes**

   Optimizer Cost is the only comparison metric available if you chose to generate execution plans only in the SQL trials.

   To perform the comparison analysis by using more than one comparison metric, perform separate comparison analyses by repeating this procedure with different metrics.
8. Ensure that pending optimizer statistics are collected, and select **Pending optimizer statistics collected**.
9. In the Schedule section:

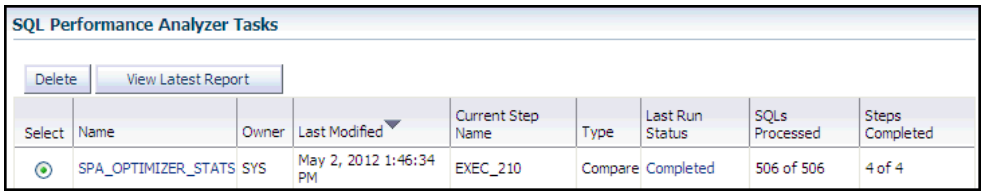
a. In the Time Zone list, select your time zone code.

b. 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.

10. Click **Submit**.

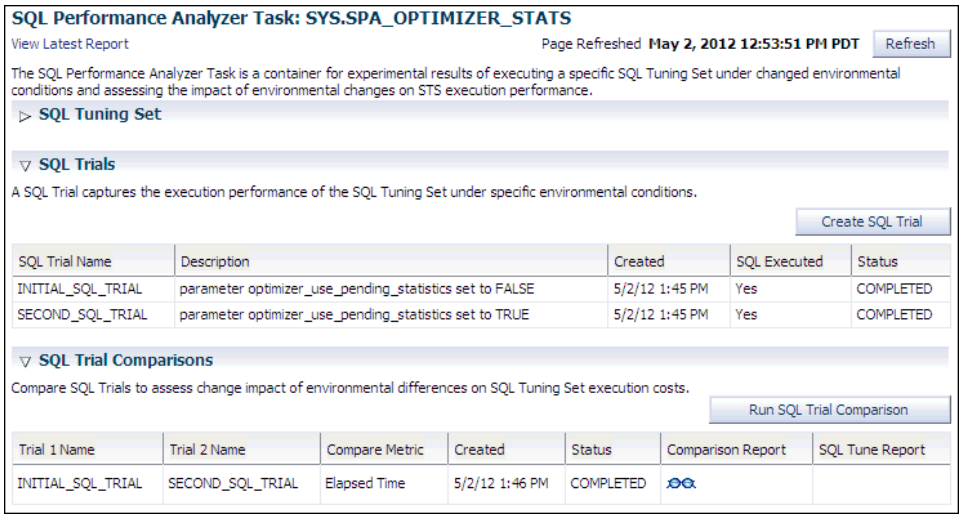
    The SQL Performance Analyzer Home page appears.

    In the SQL Performance Analyzer Tasks section, the status of this task is displayed. To refresh the status icon, click **Refresh**. After the task completes, the Status field changes to Completed.



11. In the SQL Performance Analyzer Tasks section, select the task and click the link in the Name column.

    The SQL Performance Analyzer Task page appears.



    This page contains the following sections:

    - SQL Tuning Set

      This section summarizes information about the SQL tuning set, including its name, owner, description, and the number of SQL statements it contains.

    - SQL Trials

      This section includes a table that lists the SQL trials used in the SQL Performance Analyzer task.

    - SQL Trial Comparisons

      This section contains a table that lists the results of the SQL trial comparisons

12. Click the icon in the Comparison Report column.

The SQL Performance Analyzer Task Result page appears.

13. Review the results of the performance analysis, as described in "Reviewing the SQL Performance Analyzer Report Using Oracle Enterprise Manager".

    Any regressions found in performance can be fixed using SQL plan baselines and the SQL Tuning Advisor. If the pending optimizer statistics produce satisfactory performance, you can publish for use.

Using the Exadata Simulation Workflow

The Exadata simulation workflow enables you to simulate the effects of an Exadata Storage Server installation on the performance of a SQL workload.

Oracle Exadata provides extremely large I/O bandwidth coupled with a capability to offload SQL processing from the database to storage. This allows Oracle Database to significantly reduce the volume of data sent through the I/O interconnect, while at the same time offloading CPU resources to the Exadata storage cells.

SQL Performance Analyzer can analyze the effectiveness of Exadata SQL offload processing by simulating an Exadata Storage Server installation and measuring the reduction in I/O interconnect usage for the SQL workload.

Running the Exadata simulation does not require any hardware or configuration changes to your system. After you select a SQL tuning set, SQL Performance Analyzer creates a task and performs an initial trial with the Exadata Storage Server simulation disabled. SQL Performance Analyzer then performs a second trial with the Exadata Storage Server simulation enabled. SQL Performance Analyzer then compares the two trials using the I/O Interconnect Bytes comparison metric and generates a SQL Performance Analyzer report, which estimates the amount of data that would not need to be sent from the Exadata storage cells to the database if Oracle Exadata is being used. In both SQL trials, the SQL statements are executed to completion and I/O interconnect bytes measurements are taken as the actual and simulated Exadata values for the first and second trials, respectively. The measured change in I/O interconnect bytes provides a good estimate of how much filtering can be performed in the Exadata storage cells and, in turn, the amount of CPU that normally would be used to process this data, but now can be offloaded from the database.

> **Note:**
>
> Using the Exadata simulation will not result in any plan changes. Execution plans do not change in an Exadata Storage Server installation because the simulation focuses on measuring the improvement in I/O interconnect usage. Moreover, I/O interconnect bytes will not increase, except when data compression is used (see next note), because Oracle Exadata will only decrease the amount of data sent to the database.

> **Note:**
>
> Because I/O interconnect bytes is the only metric used to measure the performance change impact of using an Exadata Storage Server installation, it will not work properly if Oracle Exadata is used with data compression. Since Exadata storage cells also decompress data, the I/O interconnect bytes with Oracle Exadata (or the second SQL trial) of a SQL statement may be greater than the I/O interconnect bytes without Oracle Exadata (or the first SQL trial) where the data is compressed. This comparison will be misleading because the SQL statement will be reported as a regression; when in fact, it is not.

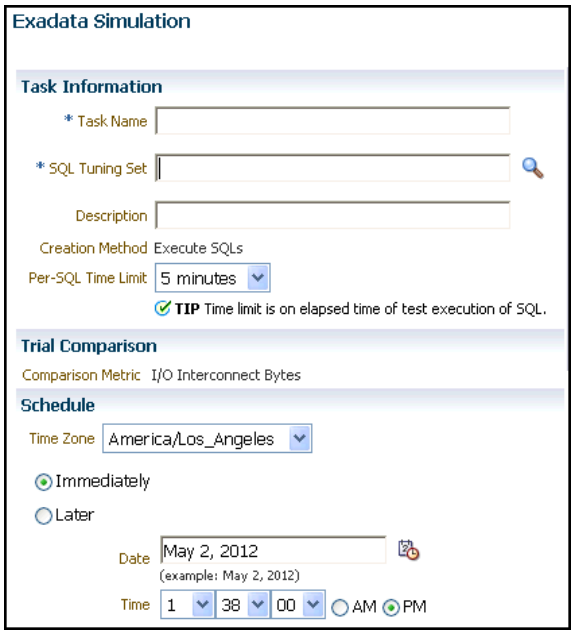
> **Note:**
>
> The Exadata simulation workflow is used to simulate an Exadata Storage Server installation on non-Exadata hardware. To test changes on Exadata hardware, use the standard SQL Performance Analyzer workflows.

> **Note:**
>
> The Exadata simulation is supported for DSS and data warehouse workloads only.

To use the SQL Performance Analyzer Exadata simulation workflow:

1. On the SQL Performance Analyzer Home page, under SQL Performance Analyzer Workflows, click **Exadata Simulation**.

   The Exadata Simulation page appears.



2. In the Task Name field, enter the name of the task.
3. In the SQL Tuning Set field, enter the name of the SQL tuning set that contains the SQL workload to be analyzed.

   Alternatively, click the search icon to search for a SQL tuning set using the Search and Select: SQL Tuning Set window.

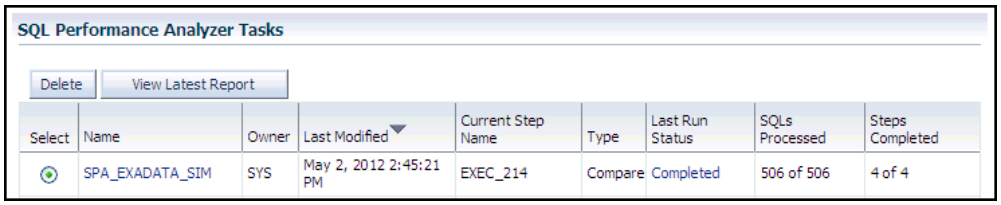
   The selected SQL tuning set now appears in the SQL Tuning Set field.
4. In the Description field, optionally enter a description of the task.
5. In the Per-SQL Time Limit list, determine the time limit for SQL execution during the trial by performing one of the following actions:
   - Select **5 minutes**.

     The execution will run each SQL statement in the SQL tuning set up to 5 minutes and gather performance data.
   - Select **Unlimited**.

     The execution will run each SQL statement in the SQL tuning set to completion and gather performance data. Collecting execution statistics provides greater accuracy in the performance analysis but takes a longer time. Using this setting is not recommended because the task may be stalled by one SQL statement for a prolonged time period.
   - Select **Customize** and enter the specified number of seconds, minutes, or hours.
6. In the Schedule section:
   a. In the Time Zone list, select your time zone code.
   b. 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.
7. Click **Submit**.

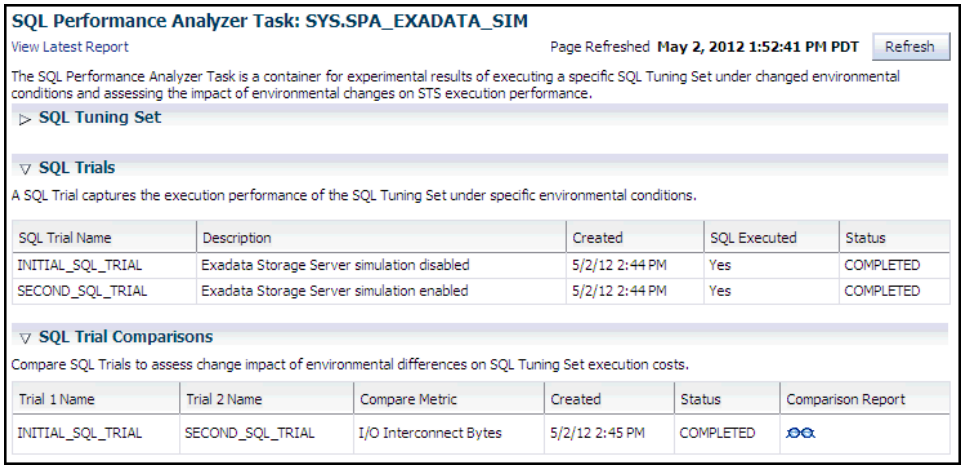
   The SQL Performance Analyzer Home page appears.

   In the SQL Performance Analyzer Tasks section, the status of this task is displayed. To refresh the status icon, click **Refresh**. After the task completes, the Status field changes to Completed.



8. In the SQL Performance Analyzer Tasks section, select the task and click the link in the Name column.

   The SQL Performance Analyzer Task page appears.



This page contains the following sections:

- SQL Tuning Set

  This section summarizes information about the SQL tuning set, including its name, owner, description, and the number of SQL statements it contains.

- SQL Trials

  This section includes a table that lists the SQL trials used in the SQL Performance Analyzer task.

- SQL Trial Comparisons

  This section contains a table that lists the results of the SQL trial comparisons

9. Click the icon in the Comparison Report column.

   The SQL Performance Analyzer Task Result page appears.

10. Review the results of the performance analysis, as described in "Reviewing the SQL Performance Analyzer Report Using Oracle Enterprise Manager".

    Any SQL performance improvement with the Exadata simulation between the first and second trials is captured in the report. In general, you can expect a greater impact if the SQL workload contains queries that scan a large number of rows or a small subset of table columns. Conversely, a SQL workload that queries indexed tables or tables with fewer rows will result in a lesser impact from the Exadata simulation.

Using the Guided Workflow

The guided workflow enables you to test the performance effect of any types of system changes on a SQL workload. See "SQL Performance Analyzer" for a list of system changes that can impact SQL performance.

> **Note:**
>
> To create an analysis task to test database initialization parameter changes, use the simplified parameter change workflow instead, as described in "Using the Parameter Change Workflow".

To use the SQL Performance Analyzer task guided workflow:

1. On the SQL Performance Analyzer Home page, under SQL Performance Analyzer Workflows, click **Guided Workflow**.

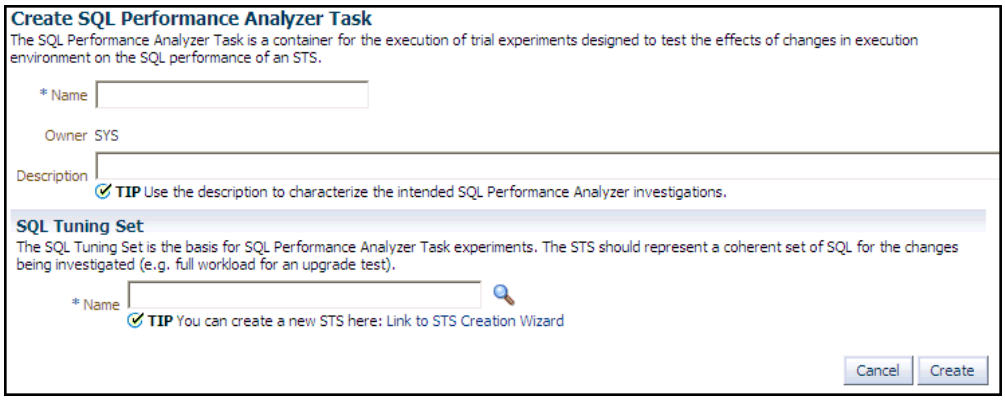
   The Guided Workflow page appears.

   The guided workflow enables you to test the performance effect on a SQL workload when you perform any type of system changes, as described in "SQL Performance Analyzer".

   This page lists the required steps in the SQL Performance Analyzer task in sequential order. Each step must be completed in the order displayed before the next step can begin.

2. On the Guided Workflow page, click the **Execute** icon for the Step 1: Create SQL Performance Analyzer Task based on SQL Tuning Set.

   The Create SQL Performance Analyzer Task page appears.



3. In the Name field, enter the name of the task.
4. In the Description field, optionally enter a description of the task.
5. Under SQL Tuning Set, in the Name field, enter the name the SQL tuning set that contains the SQL workload to be analyzed.

   Alternatively, click the search icon to select a SQL tuning set from the Search and Select: SQL Tuning Set window.

6. Click **Create**.

   The Guided Workflow page appears.

The Status icon of this step has changed to a check mark and the **Execute** icon for the next step is now enabled.

7. Once the analysis task is created, you can build the pre-change performance data by executing the SQL statements stored in the SQL tuning set, as described in Creating a Pre-Change SQL Trial .

Creating an Analysis Task Using APIs

This section describes how to create a SQL Performance Analyzer task by using the `DBMS_SQLPA.CREATE_ANALYSIS_TASK` function. A task is a database container for SQL Performance Analyzer execution inputs and results.

Before proceeding, capture the SQL workload to be used in the performance analysis into a SQL tuning set on the production system, then transport it to the test system where the performance analysis will be performed, as described in "Capturing the SQL Workload".

To create an analysis task:

- Call the `CREATE_ANALYSIS_TASK` function using the following parameters:
  - Set `task_name` to specify an optional name for the SQL Performance Analyzer task.
  - Set `sqlset_name` to the name of the SQL tuning set.
  - Set `sqlset_owner` to the owner of the SQL tuning set. The default is the current schema owner.
  - Set `basic_filter` to the SQL predicate used to filter the SQL from the SQL tuning set.
  - Set `order_by` to specify the order in which the SQL statements will be executed.

    You can use this parameter to ensure that the more important SQL statements will be processed and not skipped if the time limit is reached.
  - Set `top_sql` to consider only the top number of SQL statements after filtering and ranking.

  The following example illustrates a function call:

```
VARIABLE t_name VARCHAR2(100);
EXEC :t_name := DBMS_SQLPA.CREATE_ANALYSIS_TASK(sqlset_name => 'my_sts', -
       task_name => 'my_spa_task');
```

Once the analysis task is created, you can build the pre-change performance data by executing the SQL statements stored in the SQL tuning set, as described in Creating a Pre-Change SQL Trial .

> **See Also:**
>
> - *Oracle Database PL/SQL Packages and Types Reference* to learn more about the `DBMS_SQLPA.CREATE_ANALYSIS_TASK` function

Configuring an Analysis Task Using APIs

This section describes how to configure a SQL Performance Analyzer task once it has been created. You can configure an analysis task by setting its parameters using the `DBMS_SQLPA.SET_ANALYSIS_TASK_PARAMETER` procedure.
This section contains the following topics:

Configuring the Execution Plan Comparison Method of an Analysis Task Using APIs

You can configure the comparison method that determines when a SQL Performance Analyzer task performs line-by-line comparison of execution plans. By default, a SQL Performance Analyzer task performs line-by-line comparison of execution plans only if the plan hash value is unknown.

To configure the execution plan comparison method of an analysis task:

- Use the `SET_ANALYSIS_TASK_PARAMETER` procedure to set the value of the `PLAN_LINES_COMPARISON` parameter.

  Table 3-1 lists the valid values for the `PLAN_LINES_COMPARISON` parameter.

Table 3-1 SQL Performance Analyzer Task Execution Plan Methods

Method	Description
`ALWAYS`	The analysis task always performs a line-by-line comparison of execution plans.
`AUTO`	The analysis task performs a line-by-line comparison of execution plans only if the computation of the plan hash value for the first SQL trial has changed or the second SQL trial is unavailable.
`NONE`	The analysis task performs a line-by-line comparison of execution plans only if the plan hash value is unknown. This is the default value.

The following example shows how to set the execution plan method for an analysis task to `AUTO`:

```
EXEC DBMS_SQLPA.SET_ANALYSIS_TASK_PARAMETER(task_name => 'my_spa_task', -
       parameter => 'PLAN_LINES_COMPARISON', -
       value => 'AUTO');
```

> **See Also:**
>
> - *Oracle Database PL/SQL Packages and Types Reference* for information about the `DBMS_SQLPA.SET_ANALYSIS_TASK_PARAMETER` procedure

Configuring an Analysis Task for Exadata Simulation Using APIs

You can configure a SQL Performance Analyzer to run the Oracle Exadata simulation. For information about how SQL Performance Analyzer simulates the effects of an Exadata Storage Server installation on the performance of a SQL workload, see "Using the Exadata Simulation Workflow".

To enable Exadata simulation for an analysis task:

- Call the `SET_ANALYSIS_TASK_PARAMETER` procedure before creating the post-change SQL trial, as shown in the following example:

```
EXEC DBMS_SQLPA.SET_ANALYSIS_TASK_PARAMETER(task_name => 'my_spa_task', -
       parameter => 'CELL_SIMULATION_ENABLED', -
       value => 'TRUE');
```

  This will enable Exadata simulation when you create the post-change SQL trial, which can then be compared to the pre-change SQL trial that was created with Exadata simulation disabled.

Alternatively, you can run the Exadata simulation using the `tcellsim.sql` script.

To run the Exadata simulation using tcellsim.sql:

1. At the SQL prompt, enter:

```
@$ORACLE_HOME/rdbms/admin/tcellsim.sql
```

2. Enter the name and owner of the SQL tuning set to use:

```
Enter value for sts_name: MY_STS
Enter value for sts_owner: IMMCHAN
```

   The script then runs the following four steps automatically:

   - Creates a SQL Performance Analyzer task
   - Test executes SQL statements with Exadata simulation disabled
   - Test executes SQL statements with Exadata simulation enabled
   - Compares performance and generates analysis report

> **See Also:**
>
> - *Oracle Database PL/SQL Packages and Types Reference* for information about the `DBMS_SQLPA.SET_ANALYSIS_TASK_PARAMETER` procedure

Remapping Multitenant Container Database Identifiers in an Analysis Task Using APIs

You can store captured SQL statements in a SQL tuning set, and use it as an input source when creating a SQL Performance Analyzer task. SQL Performance Analyzer then uses the SQL tuning set as the source for test executing or generating execution plans for SQL trials.

If you use a SQL tuning set that was transported from a non-CDB to a multitenant container database (CDB) as the input source, the CDB identifiers of the SQL statements in the SQL tuning set must be remapped to make the STS usable in the CDB. Remapping CDB identifiers associates each SQL statement in the SQL tuning set with a CDB identifier that can be remapped to the corresponding pluggable databases (PDBs) within the CDB.

Typically, CDB identifiers should be remapped when the SQL tuning set is transported from a non-CDB to a CDB. In this case, you can simply use the SQL tuning set as an input source for SQL Performance Analyzer. However, if you are using a SQL tuning set whose CDB identifiers have not been remapped, you can specify the remapping as a SQL Performance Analyzer task property.

To remap CDB identifiers for an analysis task:

- Use the `SET_ANALYSIS_TASK_PARAMETER` procedure, as shown in the following example:

```
EXEC DBMS_SQLPA.SET_ANALYSIS_TASK_PARAMETER(task_name => 'non_cdb_spa1', -
       parameter => 'CON_DBID_MAPPING', -
       value => '1234:5678,1357:2468');
```

  In this example, the CDB identifiers `1234` and `1357` are remapped to `5678` and `2468`, respectively.

After the CDB identifiers are remapped, SQL Performance Analyzer uses the new CDB identifier when it finds a match for the old CDB identifier, and executes the SQL statements in the appropriate PDB within the CDB.

> **See Also:**
>
> - *Oracle Database SQL Tuning Guide* for information about transporting SQL tuning sets
> - *Oracle Database PL/SQL Packages and Types Reference* for information about the `DBMS_SQLPA.SET_ANALYSIS_TASK_PARAMETER` procedure

Configuring Trigger Execution in an Analysis Task

You can configure whether or not triggers are executed in an analysis task. By default, triggers are executed by SQL Performance Analyzer.

To configure trigger execution in an analysis task:

- Use the `SET_ANALYSIS_TASK_PARAMETER` procedure to set the value of the `EXECUTE_TRIGGERS` parameter.

  Table 3-2 lists the valid values for the `EXECUTE_TRIGGERS` parameter.

Table 3-2 Valid Values for the EXECUTE_TRIGGERS Parameter

Value	Description
FALSE	Triggers are not executed by SQL Performance Analyzer, even in the EXECUTE_FULLDML mode of TEST EXECUTE. This is the default value.
TRUE	All triggers are executed by SQL Performance Analyzer.

The following example shows how to set the value of the EXECUTE_TRIGGERS parameter to FALSE, ensuring that triggers are not executed by SQL Performance Analyzer:

```
EXEC DBMS_SQLPA.SET_ANALYSIS_TASK_PARAMETER(task_name => 'my_spa_task', -
       parameter => 'EXECUTE_TRIGGERS', -
       value => 'FALSE');
```

See Also:

- *Oracle Database PL/SQL Packages and Types Reference* for information about the DBMS_SQLPA.SET_ANALYSIS_TASK_PARAMETER procedure

Configuring a Date to be Returned by Calls in an Analysis Task

You can configure how SQL statements that refer to SYSDATE in an analysis task are handled.

When you set the REPLACE_SYSDATE_WITH parameter, all calls to SYSDATE within the task execution return a date specified by the parameter. This can be used when the input to a SPA task is a SQL tuning set (STS).

To configure the date to be returned by calls to SYSDATE in an analysis task:

- Use the SET_ANALYSIS_TASK_PARAMETER procedure to set the value of the REPLACE_SYSDATE_WITH parameter.

Table 3-3 lists the valid values for the REPLACE_SYSDATE_WITH parameter.

Table 3-3 Valid Values for the REPLACE_SYSDATE_WITH Parameter

Value	Description
CURRENT_SYSDATE	All calls to SYSDATE within the task execution return the current SYSDATE. This is the default.
SQLSET_SYSDATE	For every SQL statement that has a SYSDATE call, SQL Performance Analyzer will replace its value with the value in the LAST_EXEC_START_TIME column of the DBA_SQLSET_STATEMENTS view for that SQL statement.

> **Note:**
>
> The setting for this parameter does not affect calls to SYSDATE outside of the SQL Performance Analyzer task execution.

The following example shows how to set the value of the `REPLACE_SYSDATE_WITH` parameter to `SQLSET_SYSDATE`, ensuring that calls to SYDATE within the task execution return the SYSDATE in the SQL tuning set.

```
EXEC DBMS_SQLPA.SET_ANALYSIS_TASK_PARAMETER(task_name => 'my_spa_task', -
       parameter => 'REPLACE_SYSDATE_WITH', -
       value => 'SQLSET_SYSDATE');
```

> **See Also:**
>
> - *Oracle Database PL/SQL Packages and Types Reference* for information about the `DBMS_SQLPA.SET_ANALYSIS_TASK_PARAMETER` procedure
> - *Oracle Database Reference* for more information about the `DBA_SQLSET_STATEMENTS` view.

Configuring the Number of Rows to Fetch for an Analysis Task

You can configure how many rows are fetched for the SQL statements in an analysis task.

To configure the number of rows to fetch in an analysis task:

- Use the `SET_ANALYSIS_TASK_PARAMETER` procedure to set the value of the `NUM_ROWS_TO_FETCH` parameter.

  Table 3-4 lists the valid values for the `NUM_ROWS_TO_FETCH` parameter.

Table 3-4 Valid Values for the NUM_ROWS_TO_FETCH Parameter

Value	Description
ALL_ROWS	Fetches all the rows for the SQL. This is the default value.
AUTO	The number of result rows is determined using the value of the `OPTIMIZER_MODE` parameter in the optimizer environment captured in the SQL tuning set. If the value of `OPTIMIZER_MODE` was `ALL_ROWS`, then all result rows will be fetched. If its value was `FIRST_ROWS_`*n*, then *n* result rows will be fetched by SQL Performance Analyzer.
AVERAGE	The number of result rows is calculated as the ratio of total rows processed and total executions for each SQL in the SQL tuning set.
A valid number	The number of result rows will be equal to the specified value, or fewer, if there were fewer rows to fetch.

The following example shows how to set the value of the NUM_ROWS_TO_FETCH parameter to ALL_ROWS, so that all the rows for the SQL are fetched.

```
EXEC DBMS_SQLPA.SET_ANALYSIS_TASK_PARAMETER(task_name => 'my_spa_task', -
       parameter => 'NUM_ROWS_TO_FETCH', -
       value => 'ALL_ROWS');
```

See Also:

- *Oracle Database PL/SQL Packages and Types Reference* for information about the DBMS_SQLPA.SET_ANALYSIS_TASK_PARAMETER procedure
- *Oracle Database Reference* for more information about the OPTIMIZER_MODE database initialization parameter

Configuring the Degree of Parallelism for an Analysis Task

You can set the degree of parallelism and enable concurrent SQL execution.

To configure degree of parallelism for an analysis task:

- Use the SET_ANALYSIS_TASK_PARAMETER procedure to set the value of the TEST_EXECUTE_DOP parameter.

The following table lists the valid values for the TEST_EXECUTE_DOP parameter.

Table 3-5 Valid Values for the TEST_EXECUTE_DOP parameter

Value	Description
0	This is the default value. The task is executed serially.
Greater than or equal to 2	Concurrent execution is enabled.

The following example shows how to set the value of the TEST_EXECUTE_DOP parameter to 4 and enable concurrent execution:

```
EXEC DBMS_SQLPA.SET_ANALYSIS_TASK_PARAMETER(task_name => 'my_spa_task', -
       parameter => 'TEST_EXECUTE_DOP', -
       value => 4);
```

Note:

A concurrent execution is supported only by the EXPLAIN PLAN and TEST EXECUTE execution types.

> **See Also:**
>
> *Oracle Database PL/SQL Packages and Types Reference* for information about the `DBMS_SQLPA.SET_ANALYSIS_TASK_PARAMETER` procedure

Validating SQL Result Sets Using SQL Performance Analyzer

Comparison of SQL result sets is now supported when running two “test-execute” SQL Performance Analyzer (SPA) trials.

The SQL result sets are validated and if the number of rows or values do not match, it is recorded in the SQL Performance Analyzer report. The SQL result set validation is controlled by the `COMPARE_RESULTSET` parameter.

The `SET_ANALYSIS_TASK_PARAMETER` procedure is used to set the value of the `COMPARE_RESULTSET` parameter.

The following table lists the valid values for the `COMPARE_RESULTSET` parameter.

Table 3-6 Valid Values for the COMPARE_RESULTSET Parameter

Value	Description
`TRUE`	SQL result set comparison is performed.
`FALSE`	SQL result set comparison is not performed.

The following example shows how to set the value of `COMPARE_RESULTSET` parameter to `TRUE`.

```
EXEC DBMS_SQLPA.SET_ANALYSIS_TASK_PARAMETER(task_name => 'my_spa_task', -
       parameter => 'COMPARE_RESULTSET', -
       value => 'TRUE');
```

> **See Also:**
>
> *Oracle Database PL/SQL Packages and Types Reference* for information about the `DBMS_SQLPA.SET_ANALYSIS_TASK_PARAMETER` procedure

4

Creating a Pre-Change SQL Trial

After creating a SQL Performance Analyzer task and selecting a SQL tuning set as the input source, you need to establish the initial environment on the test system. Establishing the database environment on the test system involves manually making any necessary environmental changes that affect SQL optimization and performance. These changes may include changing initialization parameters, gathering or setting optimizer statistics, and creating indexes. It is recommended that you build a test system that is as similar to the production system as possible. The dedicated workflows in Enterprise Manager simplifies this process by creating both SQL trials automatically and performing the change restricted to the testing session.

> **Note:**
>
> You can optionally run SQL trials on a remote system by providing access to a public database link. When conducting remote SQL trials, the database version of the remote database where the SQL statements are executed must be less than or equal to the database version of the database to which it connects. Starting with Oracle Database release 11.2.0.2, the remote database can be a read-only database, such as an Oracle Active Data Guard instance.

Once the environment on the test system is properly configured, you can build the pre-change version of performance data before performing the system change. You can build SQL trials using SQL Performance Analyzer by using one of the following methods:

- Executing the SQL statements in the workload
- Generating execution plans for the SQL statements in the workload
- Loading performance data and execution plans from a SQL tuning set (APIs only)

This chapter describes how to create the pre-change SQL trial and contains the following topics:

> **Note:**
>
> The primary interface for creating a pre-change SQL trial is Oracle Enterprise Manager. If for some reason Oracle Enterprise Manager is unavailable, you can create a pre-change SQL trial using the `DBMS_SQLPA` PL/SQL package.

> **See Also:**
>
> - "Setting Up the Test System"
> - "Measuring the Pre-Change SQL Performance"

Creating a Pre-Change SQL Trial Using Enterprise Manager

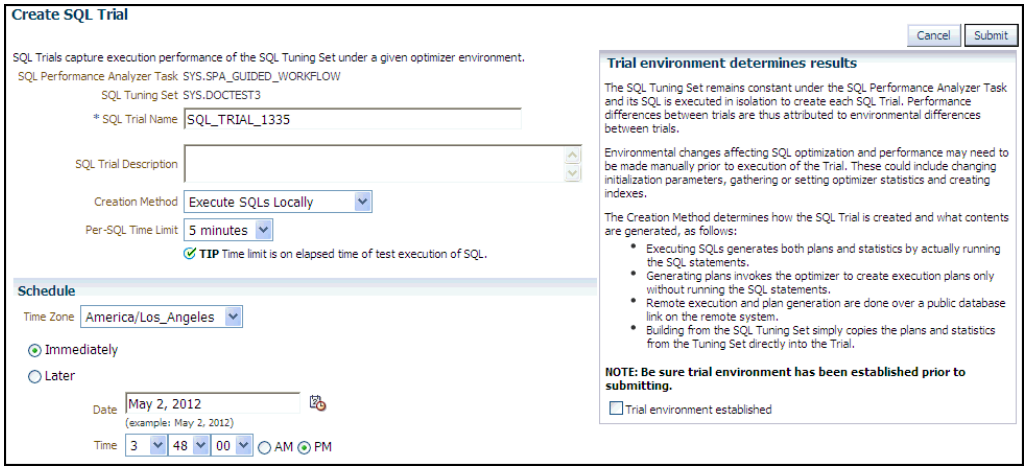
This section describes how to collect the pre-change SQL performance data using Oracle Enterprise Manager.

Before creating a pre-change SQL trial, you need to create a SQL Performance Analyzer task, as described in Creating an Analysis Task .

To create a pre-change SQL trial using Enterprise Manager:

1. On the Guided Workflow page, click the **Execute** icon for the Create SQL Trial in Initial Environment step.

   The Create SQL Trial page appears. A summary of the selected SQL tuning set containing the SQL workload is displayed.



2. In the SQL Trial Name field, enter the name of the SQL trial.
3. In the SQL Trial Description field, enter a description of the SQL trial.
4. In the Creation Method list, determine how the SQL trial is created and what contents are generated by performing one of the following actions:

   - Select **Execute SQLs Locally**.

     The SQL trial generates both execution plans and statistics for each SQL statement in the SQL tuning set by actually running the SQL statements locally on the test system.

   - Select **Execute SQLs Remotely**.

     The SQL trial generates both execution plans and statistics for each SQL statement in the SQL tuning set by actually running the SQL statements remotely on another test system over a public database link.

   - Select **Generate Plans Locally**.

The SQL trial invokes the optimizer to create execution plans locally on the test system, after taking bind values and optimizer configuration into account, without actually running the SQL statements.

- Select **Generate Plans Remotely**.

  The SQL trial invokes the optimizer to create execution plans remotely on another test system, after taking bind values and optimizer configuration into account, over a public database link without actually running the SQL statements.

- Select **Build From SQL Tuning Set**.

  The SQL trial copies the execution plans and statistics from the SQL tuning set directly into the trial.

For more information about the different methods, see "Measuring the Pre-Change SQL Performance".

5. In the Per-SQL Time Limit list, determine the time limit for SQL execution during the trial by performing one of the following actions:

   - Select **5 minutes**.

     The execution will run each SQL statement in the SQL tuning set up to 5 minutes and gather performance data.

   - Select **Unlimited**.

     The execution will run each SQL statement in the SQL tuning set to completion and gather performance data. Collecting execution statistics provides greater accuracy in the performance analysis but takes a longer time. Using this setting is not recommended because the task may be stalled by one SQL statement for a prolonged period.

   - Select **Customize** and enter the specified number of seconds, minutes, or hours.

6. Ensure that the database environment on the test system matches the production environment as closely as possible, and select **Trial environment established**.

7. In the Schedule section:

   a. In the Time Zone list, select your time zone code.

   b. 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.

8. Click **Submit**.

   The Guided Workflow page appears when the execution begins.

   The status icon of this step changes to a clock while the execution is in progress. To refresh the status icon, click **Refresh**. Depending on the options selected and the size of the SQL workload, the execution may take a long time to complete. After the execution is completed, the Status icon will change to a check mark and the Execute icon for the next step is enabled.

9. Once the pre-change performance data is built, you can make the system change and build the post-change performance data by re-executing the SQL statements in the SQL tuning set on the post-change test system, as described in Creating a Post-Change SQL Trial .

Creating a Pre-Change SQL Trial Using APIs

This section describes how to build the pre-change performance data by using the `DBMS_SQLPA` package.

Before creating a pre-change SQL trial, you need to create a SQL Performance Analyzer task, as described in Creating an Analysis Task .

To create a pre-change SQL trial:

- Call the `EXECUTE_ANALYSIS_TASK` procedure using the following parameters:
  - Set the `task_name` parameter to the name of the SQL Performance Analyzer task that you want to execute.
  - Set the `execution_type` parameter in one of the following ways:
    - Set to `EXPLAIN PLAN` to generate execution plans for all SQL statements in the SQL tuning set without executing them.
    - Set to `TEST EXECUTE` (recommended) to execute all statements in the SQL tuning set and generate their execution plans and statistics. When `TEST EXECUTE` is specified, the procedure generates execution plans and execution statistics. The execution statistics enable SQL Performance Analyzer to identify SQL statements that have improved or regressed. Collecting execution statistics in addition to generating execution plans provides greater accuracy in the performance analysis, but takes longer.
    - Set to `CONVERT SQLSET` to refer to a SQL tuning set for the execution statistics and plans for the SQL trial. Values for the execution parameters `SQLSET_NAME` and `SQLSET_OWNER` should also be specified.
  - Specify a name to identify the execution using the `execution_name` parameter. If not specified, then SQL Performance Analyzer automatically generates a name for the task execution.
  - Specify execution parameters using the `execution_params` parameters. The `execution_params` parameters are specified as (*name*, *value*) pairs for the specified execution. For example, you can set the following execution parameters:
    - The `time_limit` parameter specifies the global time limit to process all SQL statements in a SQL tuning set before timing out.
    - The `local_time_limit` parameter specifies the time limit to process each SQL statement in a SQL tuning set before timing out.
    - To perform a remote test execute, set the `DATABASE_LINK` task parameter to the global name of a public database link connecting to a user with the `EXECUTE` privilege for the `DBMS_SQLPA` package and the `ADVISOR` privilege on the test system.
    - To fully execute DML statements—including acquiring row locks and modifying row —set the `EXECUTE_FULLDML` parameter to `TRUE`. SQL Performance Analyzer will issue a rollback after executing the DML statements to prevent persistent changes from being made. The default value for this parameter is `FALSE`, which executes only the query portion of the DML statement without modifying the data.
    - To restore the relevant captured `init.ora` settings during a test execute, set the `APPLY_CAPTURED_COMPILENV` parameter to `TRUE`. This is not the default behavior because typically you are running SQL trials to test changes when changing the environment. However, this method may be used in cases when the `init.ora`

settings are not being changed (such as creating an index). This method is not supported for remote SQL trials.

The following example illustrates a function call made *before* a system change:

```
EXEC DBMS_SQLPA.EXECUTE_ANALYSIS_TASK(task_name => 'my_spa_task', -
       execution_type => 'TEST EXECUTE', -
       execution_name => 'my_exec_BEFORE_change');
```

Once the pre-change performance data is built, you can make the system change and build the post-change performance data by re-executing the SQL statements in the SQL tuning set on the post-change test system, as described in Creating a Post-Change SQL Trial .

See Also:

- *Oracle Database PL/SQL Packages and Types Reference* to learn about the `DBMS_SQLPA.EXECUTE_ANALYSIS_TASK` function

5

Creating a Post-Change SQL Trial

After computing the pre-change SQL performance data, you can perform the system change on the test system. Before making the system change, ensure that you have executed the SQL workload in the initial environment to generate the pre-change performance data. For example, if you are testing how changing a database initialization parameter will affect SQL performance, execute the SQL workload once before changing the database initialization parameter to a new value. Depending on the type of change you are making, it may be necessary to reconfigure the environment on the test system to match the new environment for which you want to perform SQL performance analysis.

> **Note:**
>
> You can optionally run SQL trials on a remote system by providing access to a public database link. When conducting remote SQL trials, the database version of the remote database where the SQL statements are executed must be less than or equal to the database version of the database to which it connects. Starting with Oracle Database release 11.2.0.2, the remote database can be a read-only database, such as an Oracle Active Data Guard instance.

"SQL Performance Analyzer" lists examples of possible system changes that can be analyzed using SQL Performance Analyzer. For example, you may want to determine how a database initialization parameter change or database upgrade will affect SQL performance. You may also decide to change the system based on recommendations from an advisor such as Automatic Database Diagnostic Monitor (ADDM), SQL Tuning Advisor, or SQL Access Advisor.

After you have made the system change, you can build the post-change version of performance data by executing the SQL workload again. SQL Performance Analyzer will store the results from executing the SQL statements in a post-change SQL trial.

This section describes how to create the post-change SQL trial and contains the following topics:

- Creating a Post-Change SQL Trial Using Oracle Enterprise Manager
- Creating a Post-Change SQL Trial Using APIs

> **Note:**
>
> The primary interface for creating a post-change SQL trial is Oracle Enterprise Manager. If for some reason Oracle Enterprise Manager is unavailable, you can create a post-change SQL trial using the `DBMS_SQLPA` PL/SQL package.

> **See Also:**
>
> - "Making a System Change"
> - "Measuring the Post-Change SQL Performance"

Creating a Post-Change SQL Trial Using Oracle Enterprise Manager

This section describes how to collect the post-change SQL performance data using Oracle Enterprise Manager.

Before making the system change creating a post-change SQL trial, you need to create a pre-change SQL trial, as described in Creating a Pre-Change SQL Trial .

To create a post-change SQL trial using Enterprise Manager:

1. On the Guided Workflow page, click the **Execute** icon for the Create SQL Trial in Changed Environment step.

   The Create SQL Trial page appears.
2. In the SQL Trial Name field, enter the name of the SQL trial.
3. In the SQL Trial Description field, enter a description of the SQL trial.
4. In the Creation Method list, determine how the SQL trial is created and what contents are generated by performing one of the following actions:
   - Select **Execute SQLs Locally**.

     The SQL trial generates both execution plans and statistics for each SQL statement in the SQL tuning set by actually running the SQL statements locally on the test system.
   - Select **Execute SQLs Remotely**.

     The SQL trial generates both execution plans and statistics for each SQL statement in the SQL tuning set by actually running the SQL statements remotely on another test system over a public database link.
   - Select **Generate Plans Locally**.

     The SQL trial invokes the optimizer to create execution plans locally on the test system without actually running the SQL statements.
   - Select **Generate Plans Remotely**.

     The SQL trial invokes the optimizer to create execution plans remotely on another test system over a public database link without actually running the SQL statements.

   For each of these creation methods, the application schema and data should already exist on the local or remote test system.
5. In the Per-SQL Time Limit list, determine the time limit for SQL execution during the trial by performing one of the following actions:
   - Select **5 minutes**.

     The execution will run each SQL statement in the SQL tuning set up to 5 minutes and gather performance data.

- Select **Unlimited**.

  The execution will run each SQL statement in the SQL tuning set to completion and gather performance data. Collecting execution statistics provides greater accuracy in the performance analysis but takes a longer time. Using this setting is not recommended because the task may be stalled by one SQL statement for a prolonged time period.

- Select **Customize** and enter the specified number of seconds, minutes, or hours.

6. Ensure that the system change you are testing has been performed on the test system, and select **Trial environment established**.
7. In the Schedule section:
   a. In the Time Zone list, select your time zone code.
   b. 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.
8. Click **Submit**.

   The Guided Workflow page appears when the execution begins.

   The status icon of this step changes to a clock while the execution is in progress. To refresh the status icon, click **Refresh**. Depending on the options selected and the size of the SQL workload, the execution may take a long time to complete. After the execution is completed, the Status icon will change to a check mark and the Execute icon for the next step is enabled.
9. Once the post-change performance data is built, you can compare the pre-change SQL trial to the post-change SQL trial by running a comparison analysis, as described in Comparing SQL Trials .

Creating a Post-Change SQL Trial Using APIs

This section describes how to collect the post-change SQL performance data using the `DBMS_SQLPA` package.

Before making the system change creating a post-change SQL trial, you need to create a pre-change SQL trial, as described in Creating a Pre-Change SQL Trial .

> **Note:**
>
> If you are running the SQL statements remotely on another test system over a database link, the remote user calling this procedure needs to have the `EXECUTE` privilege for the `DBMS_SQLPA` package.

To create a post-change SQL trial:

- Call the `EXECUTE_ANALYSIS_TASK` procedure using the parameters described in "Creating a Pre-Change SQL Trial Using APIs".

  Be sure to specify a different value for the `execution_name` parameter. It is also highly recommended that you create the post-change SQL trial using the same method as the pre-change SQL trial by using the same value for the `execution_type` parameter.

> **Note:**
>
> If you want to run an Oracle Exadata simulation, you should first set the `CELL_SIMULATION_ENABLED` task parameter to `TRUE`.

The following example illustrates a function call made after a system change:

```
EXEC DBMS_SQLPA.EXECUTE_ANALYSIS_TASK(task_name => 'my_spa_task', -
       execution_type => 'TEST EXECUTE', -
       execution_name => 'my_exec_AFTER_change');
```

Once the post-change performance data is built, you can compare the pre-change SQL trial to the post-change SQL trial by running a comparison analysis, as described in Comparing SQL Trials .

> **See Also:**
>
> - "Configuring an Analysis Task for Exadata Simulation Using APIs"
> - *Oracle Database PL/SQL Packages and Types Reference* to learn about the `DBMS_SQLPA.EXECUTE_ANALYSIS_TASK` function

6

Comparing SQL Trials

After the post-change SQL performance data is built, you can compare the performance data collected in the pre-change SQL trial to the post-change SQL trial by running a comparison analysis using SQL Performance Analyzer. After the comparison analysis is completed, you can generate a report to identify the SQL statements that have improved, remained unchanged, or regressed due to the system change. The SQL Performance Analyzer report calculates two chief impact measurements for the change in performance of each SQL statement:

- Impact on workload

  This represents the percentage of impact that this change to the SQL statement has on the cumulative execution time of the workload, after accounting for execution frequency. For example, a change that causes a SQL statement's cumulative execution time to improve from 101 seconds to 1 second—where the rest of the workload had a total execution time of 99 seconds before the change—would have a 50% (2x) value for this measurement.

- Impact on SQL

  This represents the percentage of impact that this change to the SQL statement has on the SQL statement's response time. For example, a change that causes a SQL statement's response time to improve from 10 seconds to 1 second will have a 90% (10x) value for this measurement.

This chapter describes how to compare and analyze the performance data from the pre-change and post-change SQL trials and contains the following topics:

- Comparing SQL Trials Using Oracle Enterprise Manager
- Comparing SQL Trials Using APIs

> **Note:**
>
> The primary interface for comparing SQL trials is Oracle Enterprise Manager. If for some reason Oracle Enterprise Manager is unavailable, you can compare SQL trials using the `DBMS_SQLPA` PL/SQL package.

> **See Also:**
>
> "Comparing Performance Measurements"

Comparing SQL Trials Using Oracle Enterprise Manager

Comparing SQL trials using Oracle Enterprise Manager involves the following steps:

- Analyzing SQL Performance Using Oracle Enterprise Manager
- Reviewing the SQL Performance Analyzer Report Using Oracle Enterprise Manager

- Tuning Regressed SQL Statements Using Oracle Enterprise Manager

Analyzing SQL Performance Using Oracle Enterprise Manager

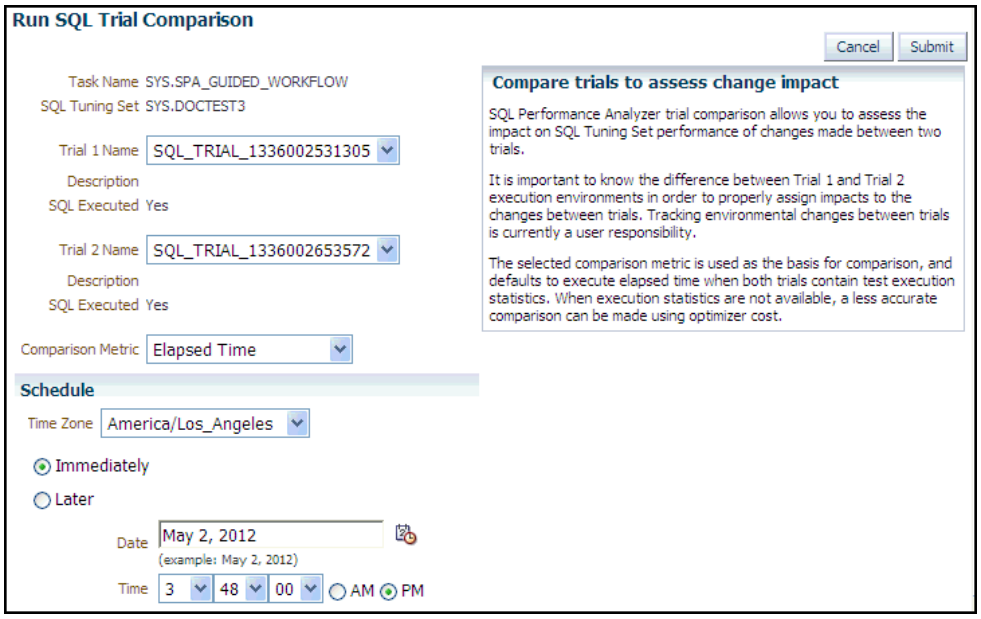
This section describes how to analyze SQL performance before and after the system change using Oracle Enterprise Manager.

Before comparing SQL trials, you need to create a post-change SQL trial, as described in Creating a Post-Change SQL Trial .

To analyze SQL performance using Enterprise Manager:

1. On the Guided Workflow page, click the **Execute** icon for Compare Step 2 and Step 3.

   The Run SQL Trial Comparison page appears.



   In this example, the `SQL_TRIAL_1241213421833` and `SQL_TRIAL_1241213881923` trials are selected for comparison.

2. To compare trials other than those listed by default, select the desired trials in the **Trial 1 Name** and **Trial 2 Name** lists.

   Note that you cannot compare a statistical trial with a trial that tests the explain plan only.

3. In the Comparison Metric list, select the comparison metric to use for the comparison analysis:
   - **Elapsed Time**
   - **CPU Time**
   - **User I/O Time**
   - **Buffer Gets**
   - **Physical I/O**
   - **Optimizer Cost**

- **I/O Interconnect Bytes**

Optimizer Cost is the only comparison metric available if you generated execution plans only in the SQL trials.

To perform the comparison analysis by using more than one comparison metric, perform separate comparison analyses by repeating this procedure with different metrics.

4. In the Schedule section:
   a. In the Time Zone list, select your time zone code.
   b. 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.
5. Click **Submit**.

   The Guided Workflow page appears when the comparison analysis begins.

   The status icon of this step changes to an arrow icon while the comparison analysis is in progress. To refresh the status icon, click **Refresh**. Depending on the amount of performance data collected from the pre-change and post-change executions, the comparison analysis may take a long time to complete. After the comparison analysis is completed, the Status icon changes to a check mark and the Execute icon for the next step is enabled.
6. Once SQL Performance Analyzer has analyzed the pre-change and post-change performance data, generate a SQL Performance Analyzer report that you can use for further analysis.

   On the Guided Workflow page, click the **Execute** icon for View Trial Comparison Report.

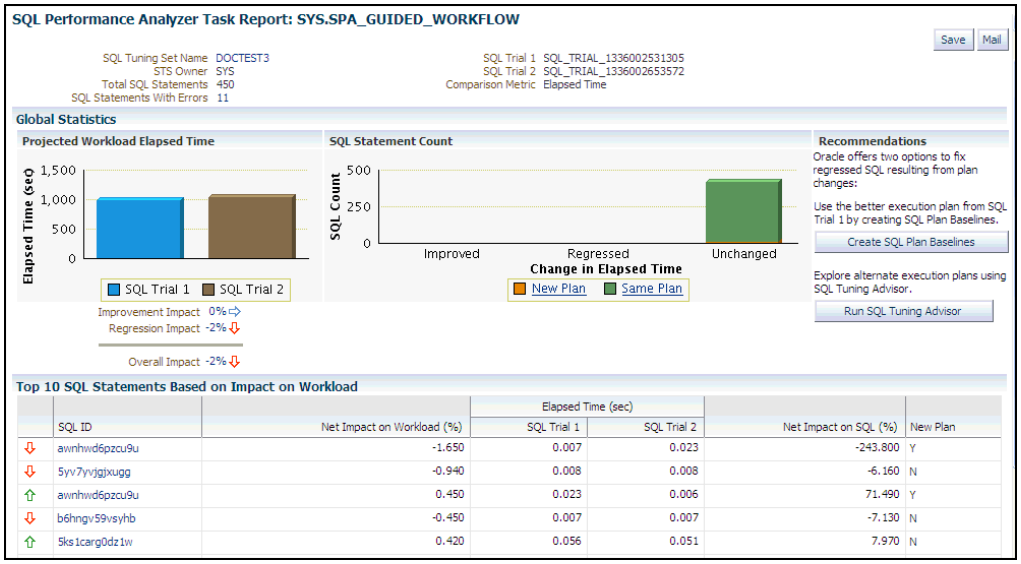
   The SQL Performance Analyzer Task Report page appears. Review the report, as described in "Reviewing the SQL Performance Analyzer Report Using Oracle Enterprise Manager".

Reviewing the SQL Performance Analyzer Report Using Oracle Enterprise Manager

When a SQL Performance Analyzer task is completed, the resulting data is generated into a SQL Performance Analyzer report that compares the pre-change and post-change SQL performance.

Figure 6-1 shows a sample SQL Performance Analyzer report. This sample report uses the elapsed time comparison metric to compare the pre-change and post-change executions of a SQL workload.

Figure 6-1 SQL Performance Analyzer Report



Before you can view the SQL Performance Analyzer report, compare the pre-change version of performance data with the post-change version, as described in "Comparing SQL Trials Using Oracle Enterprise Manager"

To generate and review the SQL Performance Analyzer report:

1. From the **Performance** menu, select **SQL**, then **SQL Performance Analyzer**.

   If the Database Login page appears, then log in as a user with administrator privileges.

   The SQL Performance Analyzer Home page appears. A list of existing SQL Performance Analyzer tasks are displayed.

2. Under SQL Performance Analyzer Tasks, select the task for which you want to view a SQL Performance Analyzer report and click **View Latest Report**.

   The SQL Performance Analyzer Task Report page appears.

3. Review the general information about the performance analysis, as described in "Reviewing the SQL Performance Analyzer Report: General Information".

4. Review general statistics, as described in "Reviewing the SQL Performance Analyzer Report: Global Statistics".

5. Optionally, review the detailed statistics, as described in "Reviewing the SQL Performance Analyzer Report: Global Statistics Details".

6. To generate an active report, click **Save** to generate and save the report, or **Mail** to generate and mail the report as an HTML attachment.

   Active reports include information about the top SQL statements from each category (such as improved, regressed, and changed plans) with pre-change and post-change statistics, explain plans, and task summary.

   For more information, see "About SQL Performance Analyzer Active Reports".

Reviewing the SQL Performance Analyzer Report: General Information

The General Information section contains basic information and metadata about the workload comparison performed by SQL Performance Analyzer.

To review general information:

1. On the SQL Performance Analyzer Task Report page, review the summary at the top of the page.

   This summary includes the following information:

   - The name and owner of the SQL tuning set
   - The total number of SQL statements in the tuning set and the number of SQL statements that had errors, are unsupported, or timed out
   - The names of the SQL trials and the comparison metric used

2. Optionally, click the link next to SQL Tuning Set Name.

   The SQL Tuning Set page appears.

   This page contains information—such as SQL ID and SQL text—about every SQL statement in the SQL tuning set.

3. Click the link next to SQL Statements With Errors if errors were found.

   The Errors table reports all errors that occurred while executing a given SQL workload. An error may be reported at the SQL tuning set level if it is common to all SQL executions in the SQL tuning set, or at the execution level if it is specific to a SQL statement or execution plan.

4. Review the global statistics, as described in "Reviewing the SQL Performance Analyzer Report: Global Statistics".

Reviewing the SQL Performance Analyzer Report: Global Statistics

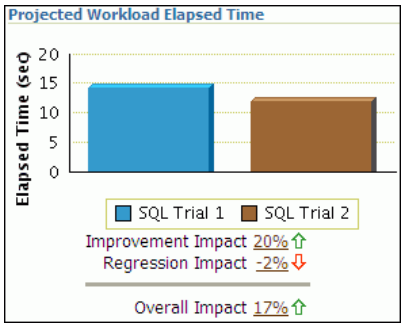
The Global Statistics section reports statistics that describe the overall performance of the entire SQL workload. This section is a very important part of the SQL Performance Analyzer analysis, because it reports on the impact of the system change on the overall performance of the SQL workload. Use the information in this section to understand the tendency of the workload performance, and determine how it will be affected by the system change.

To review global statistics:

1. Review the chart in the Projected Workload Elapsed Time subsection.

> **Note:**
>
> The name of the subsection may vary based on the comparison metric that is selected.

The chart shows the two trials on the x-axis and the elapsed time (in seconds) on the y-axis.



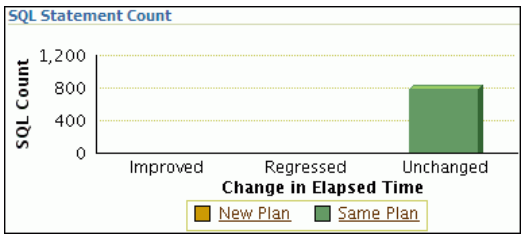
The most important statistic is the overall impact, which is given as a percentage. The overall impact is the difference between the improvement impact and the regression impact. You can click the link for any impact statistic to obtain more details, as described in "Reviewing the SQL Performance Analyzer Report: Global Statistics Details".

In this example, the improvement impact is 20%, while the regression impact is -2%, so the overall impact of the system change is an improvement of approximately 18%. This means that if all regressions are fixed in this example, the overall impact of the change will be an improvement of 20%.

> **Note:**
>
> The overall impact percentage may sometimes be off by 1% compared to the sum of the improvement impact and the regression impact. This discrepancy may be caused by rounding or if the SQL and workload time limits are set at 1%, which is the recommended value. This enables the analysis to focus on SQL statements with higher impact by filtering out those that have a minimal impact.

2. Review the chart in the SQL Statement Count subsection.

   The x-axis of the chart shows the number of SQL statements whose performance improved, regressed, or remain unchanged after the system change. The y-axis shows the number of SQL statements. The chart also indicates whether the explain plans changed for the SQL statements.



This chart enables you to quickly weigh the relative performance of the SQL statements. You can click any bar in the chart to obtain more details about the SQL statements, as described in "Reviewing the SQL Performance Analyzer Report: Global Statistics Details".

Only up to the top 100 SQL statements will be displayed, even if the actual number of SQL statements exceeds 100.

In this example, all SQL statements were unchanged after the system change.

Reviewing the SQL Performance Analyzer Report: Global Statistics Details

You can use the SQL Performance Analyzer Report to obtain detailed statistics for the SQL workload comparison. The details chart enables you to drill down into the performance of SQL statements that appears in the report. Use the information in this section to investigate why the performance of a particular SQL statement regressed.

> **Note:**
>
> The report displays only up to the top 100 SQL statements, even if the actual number of SQL statements exceeds 100.

To review global statistics details:

1. In the Projected Workload Elapsed Time subsection, click the impact percentage of the SQL statements for which you want to view details. To view SQL statements whose performance:
   - Improved, click the percentage for Improvement Impact
   - Regressed, click the percentage for Regression Impact
   - Improved or regressed, click the percentage for Overall Impact

   A table including the detailed statistics appears. Depending on the type of SQL statements chosen, the following columns are included:
   - SQL ID

     This column indicates the ID of the SQL statement.
   - Net Impact on Workload (%)

     This column indicates the impact of the system change relative to the performance of the SQL workload.
   - Elapsed Time

     This column indicates the total time (in seconds) of the SQL statement execution.
   - Net Impact on SQL (%)

     This column indicates the local impact of the change on the performance of a particular SQL statement.
   - New Plan

     This column indicates whether the SQL execution plan changed.
2. To view details about a particular SQL statement, click the SQL ID link for the SQL statement that you are interested in.

   The SQL Details page appears.

   You can use this page to access the SQL text and obtain low-level details about the SQL statement, such as its execution statistics and execution plan.

About SQL Performance Analyzer Active Reports

SQL Performance Analyzer active reports are HTML files that display all reporting data using a Web-hosted interactive user interface. Similar to the SQL Performance Analyzer reports available in Oracle Enterprise Manager, active reports include information about the top SQL statements from each category (such as improved, regressed, and changed plans) with pre-change and post-change statistics, explain plans, and task summary.
SQL Performance Analyzer active reports are more useful than traditional HTML or text reports because they offer a similar user interface as Oracle Enterprise Manager, yet they can be viewed even when the database is unavailable, or even after a database is dropped. Hence active reports offer the advantages of traditional reporting and dynamic Oracle Enterprise Manager analysis, but eliminates the disadvantages of both. Moreover, active reports contain more information about the comparison analysis and provide more user interactive options. It is strongly recommended that you use active reports instead of HTML or text reports.

The active report user interface components are very similar to those displayed in Oracle Enterprise Manager. For descriptions of the user interface components, see the related sections described in "Reviewing the SQL Performance Analyzer Report Using Oracle Enterprise Manager".

Tuning Regressed SQL Statements Using Oracle Enterprise Manager

After reviewing the SQL Performance Analyzer report, you should tune any regressed SQL statements that are identified after comparing the SQL performance. If there are large numbers of SQL statements that appear to have regressed, you should try to identify the root cause and make system-level changes to rectify the problem. In cases when only a few SQL statements have regressed, consider using one of the following tuning methods to implement a point solution for them:

- Creating SQL Plan Baselines
- Running SQL Tuning Advisor

After tuning the regressed SQL statements, you should test these changes using SQL Performance Analyzer. Run a new SQL trial on the test system, followed by a second comparison (between this new SQL trial and the first SQL trial) to validate your results. Once SQL Performance Analyzer shows that performance has stabilized, the testing is complete. Implement the fixes from this step to your production system.

Starting with Oracle Database 11*g* Release 1, SQL Tuning Advisor performs an alternative plan analysis when tuning a SQL statement. SQL Tuning Advisor searches the current system for previous execution plans, including the plans from the first SQL trial. If the execution plans from the first SQL trial differ from those of the second SQL trial, SQL Tuning Advisor will recommend the plans from the first SQL trial. If these execution plans produce better performance, you can create plan baselines using the plans from the first SQL trial.

> **Note:**
>
> SQL Performance Analyzer does not provide the option to create SQL plan baselines or run SQL Tuning Advisor directly after after completing a remote SQL trial. In such cases, you need to use APIs to manually transport the SQL tuning set and complete the appropriate procedure on the remote database.

> **See Also:**
>
> - *Oracle Database SQL Tuning Guide* for information about alternative plan analysis

Creating SQL Plan Baselines

Creating SQL plan baselines enables the optimizer to avoid performance regressions by using execution plans with known performance characteristics. If a performance regression occurs due to plan changes, a SQL plan baseline can be created and used to prevent the optimizer from picking a new, regressed execution plan.

To create SQL plan baselines:

1. On the SQL Performance Analyzer Task Result page, under Recommendations, click **Create SQL Plan Baselines**.

   The Create SQL Plan Baselines page appears. The Regressed SQL Statements section lists the regressed SQL statements that will be associated with the new SQL plan baselines.
2. Under Job Parameters, specify the parameters for the job:
   a. In the Job Name field, enter a name for the job.
   b. In the Description field, optionally enter a description for the job.
3. Under Schedule, select:
   - **Immediately** to start the job now.
   - **Later** to schedule the job to start at a time specified using the Time Zone, Date, and Time fields.
4. Click **OK**.

   The SQL Performance Analyzer Task Result page appears. A message is displayed to inform you that the job has been submitted successfully.

> **See Also:**
>
> - *Oracle Database 2 Day + Performance Tuning Guide* for information about creating and managing SQL plan baselines

Running SQL Tuning Advisor

The SQL Tuning Advisor performs an in-depth analysis of regressed SQL statements and attempts to fix the root cause of the problem.

To run SQL Tuning Advisor:

1. On the SQL Performance Analyzer Task Result page, under Recommendations, click **Run SQL Tuning Advisor**.

   The Schedule SQL Tuning Task page appears.

2. In the Tuning Task Name field, enter a name for the SQL tuning task.
3. In the Tuning Task Description field, optionally enter a name for the SQL tuning task.
4. Under Schedule, select:
   - **Immediately** to start the job now.
   - **Later** to schedule the job to start at a time specified using the Time Zone, Date, and Time fields.
5. Click **OK**.

   The SQL Performance Analyzer Task Result page appears. A link to the SQL tuning report appears under Recommendations.
6. To view the SQL tuning report, click the SQL Tune Report link.

   The SQL Tuning Results page appears.

> **See Also:**
>
> - *Oracle Database 2 Day + Performance Tuning Guide* for information about running the SQL Tuning Advisor

Comparing SQL Trials Using APIs

Comparing SQL trials using APIs involves the following steps:

- Analyzing SQL Performance Using APIs
- Reviewing the SQL Performance Analyzer Report in Command-Line
- Comparing SQL Tuning Sets Using APIs
- Tuning Regressed SQL Statements Using APIs
- Tuning Regressed SQL Statements From a Remote SQL Trial Using APIs
- Creating SQL Plan Baselines Using APIs
- Using SQL Performance Analyzer Views

Before comparing SQL trials, you need to create a post-change SQL trial, as described in Creating a Post-Change SQL Trial .

Analyzing SQL Performance Using APIs

After the post-change SQL performance data is built, you can compare the pre-change version of performance data to the post-change version. Run a comparison analysis using the `DBMS_SQLPA.EXECUTE_ANALYSIS_TASK` procedure or function.

To compare the pre-change and post-change SQL performance data:

1. Call the `EXECUTE_ANALYSIS_TASK` procedure or function using the following parameters:
   - Set the `task_name` parameter to the name of the SQL Performance Analyzer task.
   - Set the `execution_type` parameter to `COMPARE PERFORMANCE`. This setting will analyze and compare two versions of SQL performance data.

- Specify a name to identify the execution using the `execution_name` parameter. If not specified, it will be generated by SQL Performance Analyzer and returned by the function.
- Specify two versions of SQL performance data using the `execution_params` parameters. The `execution_params` parameters are specified as (*name*, *value*) pairs for the specified execution. Set the execution parameters that are related to comparing and analyzing SQL performance data as follows:
  - Set the `execution_name1` parameter to the name of the first execution (before the system change was made). This value should correspond to the value of the `execution_name` parameter specified in "Creating a Pre-Change SQL Trial Using APIs".
  - Set the `execution_name2` parameter to the name of the second execution (after the system change was made). This value should correspond to the value of the `execution_name` parameter specified in "Creating a Post-Change SQL Trial Using APIs" when you executed the SQL workload after the system change. If the caller does not specify the executions, then by default SQL Performance Analyzer will always compare the last two task executions.
  - Set the `comparison_metric` parameter to specify an expression of execution statistics to use in the performance impact analysis. Possible values include the following metrics or any combination of them: `elapsed_time` (default), `cpu_time`, `buffer_gets`, `disk_reads`, `direct_writes`, `optimizer_cost`, and `io_interconnect_bytes`.

  For other possible parameters that you can set for comparison, see the description of the `DBMS_SQLPA` package in *Oracle Database PL/SQL Packages and Types Reference*.

The following example illustrates a function call:

```
EXEC DBMS_SQLPA.EXECUTE_ANALYSIS_TASK(task_name => 'my_spa_task', -
       execution_type => 'COMPARE PERFORMANCE', -
       execution_name => 'my_exec_compare', -
       execution_params => dbms_advisor.arglist(-
                           'comparison_metric', 'buffer_gets'));
```

2. Call the `REPORT_ANALYSIS_TASK` function using the following parameters:
   - Set the `task_name` parameter to the name of the SQL Performance Analyzer task.
   - Set the `execution_name` parameter to the name of the execution to use. This value should match the `execution_name` parameter of the execution for which you want to generate a report.

     To generate a report to display the results of:
     - Execution plans generated for the SQL workload, set this value to match the `execution_name` parameter of the desired `EXPLAIN PLAN` execution.
     - Execution plans and execution statistics generated for the SQL workload, set this parameter to match the value of the `execution_name` parameter used in the desired `TEST EXECUTE` execution.
     - A comparison analysis, set this value to match the `execution_name` parameter of the desired `ANALYZE PERFORMANCE` execution.

     If unspecified, SQL Performance Analyzer generates a report for the last execution.
   - Set the `type` parameter to specify the type of report to generate. Possible values include `TEXT` (default), `HTML`, `XML`, and `ACTIVE`.

Active reports provides in-depth reporting using an interactive user interface that enables you to perform detailed analysis even when disconnected from the database or Oracle Enterprise Manager. It is recommended that you use active reports instead of HTML or text reports when possible.

For information about active reports, see "About SQL Performance Analyzer Active Reports".

- Set the `level` parameter to specify the format of the recommendations. Possible values include `TYPICAL` (default), `ALL`, `BASIC`, `CHANGED`, `CHANGED_PLANS`, `ERRORS`, `IMPROVED`, `REGRESSED`, `TIMEOUT`, `UNCHANGED`, `UNCHANGED_PLANS`, and `UNSUPPORTED`.
- Set the `section` parameter to specify a particular section to generate in the report. Possible values include `SUMMARY` (default) and `ALL`.
- Set the `top_sql` parameter to specify the number of SQL statements in a SQL tuning set to generate in the report. By default, the report shows the top 100 SQL statements impacted by the system change.

To generate an active report, run the following script:

```
set trimspool on
set trim on
set pages 0
set linesize 1000
set long 1000000
set longchunksize 1000000
spool spa_active.html
SELECT DBMS_SQLPA.REPORT_ANALYSIS_TASK(task_name => 'my_spa_task',
          type => 'active', section => 'all') FROM dual;
spool off
```

The following example illustrates a portion of a SQL script that you could use to create and display a comparison summary report in text format:

```
VAR rep   CLOB;
EXEC :rep := DBMS_SQLPA.REPORT_ANALYSIS_TASK('my_spa_task', -
                'text', 'typical', 'summary');
SET LONG 100000 LONGCHUNKSIZE 100000 LINESIZE 130
PRINT :rep
```

3. Review the SQL Performance Analyzer report, as described in "Reviewing the SQL Performance Analyzer Report in Command-Line".

> **See Also:**
>
> - *Oracle Database PL/SQL Packages and Types Reference* for information about the `DBMS_SQLPA.EXECUTE_ANALYSIS_TASK` and `DBMS_SQLPA.REPORT_ANALYSIS_TASK` functions

Reviewing the SQL Performance Analyzer Report in Command-Line

The SQL Performance Analyzer report is divided into the following sections:

- General Information
- Result Summary
- Result Details

This section uses a sample report to illustrate how to review the SQL Performance Analyzer report. The sample report uses `buffer_gets` as the comparison metric to compare the pre-change and post-change executions of a SQL workload.

General Information

The General Information section contains basic information and metadata about the SQL Performance Analyzer task, the SQL tuning set used, and the pre-change and post-change executions. Example 6-1 shows the General Information section of a sample report.
In Example 6-1, the Task Information section indicates that the task name is `my_spa_task`. The Workload Information section indicates that the task compares executions of the `my_sts` SQL tuning set, which contains 101 SQL statements. As shown in the Execution Information section, the comparison execution is named `my_exec_compare`.

The Analysis Information sections shows that SQL Performance Analyzer compares two executions of the `my_sts` SQL tuning set, `my_exec_BEFORE_change` and `my_exec_AFTER_change`, using `buffer_gets` as a comparison metric.

Example 6-1 General Information

```
-------------------------------------------------------------------------------------------------
General Information
-------------------------------------------------------------------------------------------------

 Task Information:                                Workload Information:
 -----------------------------------------        -----------------------------------------
  Task Name    : my_spa_task                       SQL Tuning Set Name        : my_sts
  Task Owner   : APPS                              SQL Tuning Set Owner       : APPS
  Description  :                                   Total SQL Statement Count  : 101

Execution Information:
-------------------------------------------------------------------------------------------------
  Execution Name  : my_exec_compare        Started             : 05/21/2007 11:30:09
  Execution Type  : ANALYZE PERFORMANCE    Last Updated        : 05/21/2007 11:30:10
  Description     :                        Global Time Limit   : UNLIMITED
  Scope           : COMPREHENSIVE          Per-SQL Time Limit  : UNUSED
  Status          : COMPLETED              Number of Errors    : 0

Analysis Information:
-------------------------------------------------------------------------------------------------
 Comparison Metric: BUFFER_GETS
 ------------------
 Workload Impact Threshold: 1%
 --------------------------
 SQL Impact Threshold: 1%
 ----------------------
 Before Change Execution:                         After Change Execution:
 -----------------------------------------        -----------------------------------------
  Execution Name      : my_exec_BEFORE_change      Execution Name      : my_exec_AFTER_change
  Execution Type      : TEST EXECUTE               Execution Type      : TEST EXECUTE
  Description         :                            Description         :
  Degree of Parallelism: 4                         Degree of Parallelism: 4
  Scope               : COMPREHENSIVE              Scope               : COMPREHENSIVE
  Status              : COMPLETED                  Status              : COMPLETED
  Started             : 05/21/2007 11:22:06        Started             : 05/21/2007 11:25:56
  Last Updated        : 05/21/2007 11:24:01        Last Updated        : 05/21/2007 11:28:30
  Global Time Limit   : 1800                       Global Time Limit   : 1800
  Per-SQL Time Limit  : UNUSED                     Per-SQL Time Limit  : UNUSED
  Number of Errors    : 0                          Number of Errors    : 0

-------------------------------------------------------------------------------------------------
```

Result Summary

The Result Summary section summarizes the results of the SQL Performance Analyzer task. The Result Summary section is divided into the following subsections:

- Overall Performance Statistics
- Performance Statistics of SQL Statements
- Errors

Overall Performance Statistics

The Overall Performance Statistics subsection displays statistics about the overall performance of the entire SQL workload. This section is a very important part of the SQL Performance Analyzer analysis because it shows the impact of the system change on the overall performance of the SQL workload. Use the information in this section to understand the change of the workload performance, and determine whether the workload performance will improve or degrade after making the system change.
Example 6-2 shows the Overall Performance Statistics subsection of a sample report.

This example indicates that the overall performance of the SQL workload improved by 47.94%, even though regressions had a negative impact of -10.08%. This means that if all of the regressions are fixed in this example, the overall change impact will be 58.02%. After the system change, 2 of the 101 SQL statements ran faster, while 1 ran slower. Performance of 98 statements remained unchanged.

Example 6-2 Overall Performance Statistics

```
Report Summary
------------------------------------------------------------------------------------------------

Projected Workload Change Impact:
-----------------------------------------
 Overall Impact      :   47.94%
 Improvement Impact  :   58.02%
 Regression Impact   :  -10.08%

SQL Statement Count
-----------------------------------------
 SQL Category  SQL Count  Plan Change Count
 Overall       101        6
 Improved      2          2
 Regressed     1          1
 Unchanged     98         3
.
.
.
------------------------------------------------------------------------------------------------
```

Performance Statistics of SQL Statements

The Performance Statistics subsection highlights the SQL statements that are the most impacted by the system change. The pre-change and post-change performance data for each SQL statement in the workload are compared based on the following criteria:

- Execution frequency, or importance, of each SQL statement
- Impact of the system change on each SQL statement relative to the entire SQL workload
- Impact of the system change on each SQL statement

- Whether the structure of the execution plan for each SQL statement has changed

Example 6-3 shows the Performance Statistics of SQL Statements subsection of a sample report. The report has been altered slightly to fit on the page.

The SQL statements are sorted in descending order by the absolute value of the net impact on the SQL workload, that is, the sort order does not depend on whether the impact was positive or negative.

Example 6-3 Performance Statistics of SQL Statements

```
SQL Statements Sorted by their Absolute Value of Change Impact on the Workload
---------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------
|           |               | Impact on | Execution |  Metric  |  Metric  | Impact | Plan   |
| object_id | sql_id        | Workload  | Frequency |  Before  |  After   | on SQL | Change |
---------------------------------------------------------------------------------------------
|    205    | 73s2sgy2svfrw |   29.01%  |   100000  | 1681683 |  220590  | 86.88% |   y    |
|    206    | gq2a407mv2hsy |   29.01%  |   949141  | 1681683 |  220590  | 86.88% |   y    |
|    204    | 2wtgxbjz6u2by |  -10.08%  |   478254  | 1653012 |  2160529 | -30.7% |   y    |
---------------------------------------------------------------------------------------------
```

Errors

The Errors subsection reports all errors that occurred during an execution. An error may be reported at the SQL tuning set level if it is common to all executions in the SQL tuning set, or at the execution level if it is specific to a SQL statement or execution plan.
Example 6-4 shows an example of the Errors subsection of a SQL Performance Analyzer report.

Example 6-4 Errors

```
--------------------------------------------------------------------------------
                            SQL STATEMENTS WITH ERRORS
--------------------------------------------------------------------------------
SQL ID        Error
------------- ----------------------------------------------------------------
47bjmcdtw6htn ORA-00942: table or view does not exist
br61bjp4tnf7y ORA-00920: invalid relational operator
--------------------------------------------------------------------------------
```

Result Details

The Result Details section represents a drill-down into the performance of SQL statements that appears in the Result Summary section of the report Use the information in this section to investigate why the performance of a particular SQL statement regressed.
This section will contain an entry of every SQL statement processed in the SQL performance impact analysis. Each entry is organized into the following subsections:

- SQL Details
- Execution Statistics
- Execution Plans

SQL Details

This section of the report summarizes the SQL statement, listing its information and execution details.

Example 6-5 shows the SQL Details subsection of a sample report.

In Example 6-5, the report summarizes the regressed SQL statement whose ID is `2wtgxbjz6u2by` and corresponding object ID is `204`.

Example 6-5 SQL Details

```
SQL Details:
-----------------------------
 Object ID            : 204
 Schema Name          : APPS
 SQL ID               : 2wtgxbjz6u2by
 Execution Frequency  : 1
 SQL Text             : SELECT /* my_query_14_scott */ /*+ ORDERED INDEX(t1)
                        USE_HASH(t1) */ 'B' || t2.pg_featurevalue_05_id
                        pg_featurevalue_05_id, 'r' || t4.elementrange_id
                        pg_featurevalue_15_id, 'G' || t5.elementgroup_id
                        pg_featurevalue_01_id, 'r' || t6.elementrange_id . . .
.
.
.
--------------------------------------------------------------------------------------------
```

Execution Statistics

The Execution Statistics subsection compares execution statistics of the SQL statement from the pre-change and post-change executions and then summarizes the findings.

Example 6-6 shows the Execution Statistics subsection of a sample report.

Example 6-6 Execution Statistics

```
Execution Statistics:
-----------------------------
-----------------------------------------------------------------------------------------
|              | Impact on | Value    | Value    | Impact      | % Workload | % Workload |
| Stat Name    | Workload  | Before   | After    | on SQL      | Before     | After      |
-----------------------------------------------------------------------------------------
| elapsed_time |    -95.54%|  36.484  | 143.161  |   -292.39%  |     32.68% |     94.73% |
| parse_time   |    -12.37%|    .004  |    .062  |     -1450%  |       .85% |     11.79% |
| exec_elapsed |    -95.89%|   36.48  | 143.099  |   -292.27%  |     32.81% |     95.02% |
| exec_cpu     |    -19.73%|  36.467  |  58.345  |    -59.99%  |     32.89% |     88.58% |
| buffer_gets  |    -10.08%| 1653012  | 2160529  |     -30.7%  |     32.82% |     82.48% |
| cost         |     12.17%|   11224  |    2771  |     75.31%  |     16.16% |      4.66% |
| reads        |  -1825.72%|    4091  |  455280  | -11028.82%  |     16.55% |     96.66% |
| writes       |     -1500%|       0  |      15  |     -1500%  |         0% |       100% |
| rows         |           |     135  |     135  |             |            |            |
-----------------------------------------------------------------------------------------

Notes:
-----------------------------
Before Change:
1. The statement was first executed to warm the buffer cache.
2. Statistics shown were averaged over next 9 executions.

After Change:
1. The statement was first executed to warm the buffer cache.
2. Statistics shown were averaged over next 9 executions.

Findings (2):
-----------------------------
1. The performance of this SQL has regressed.
2. The structure of the SQL execution plan has changed.
--------------------------------------------------------------------------------------------
```

Execution Plans

The Execution Plans subsection displays the pre-change and post-change execution plans for the SQL statement. In cases when the performance regressed, this section also contains findings on root causes and symptoms.

Example 6-7 shows the Execution Plans subsection of a sample report.

Example 6-7 Execution Plans

```
Execution Plan Before Change:
-----------------------------
 Plan Id          : 1
 Plan Hash Value  : 3412943215

-----------------------------------------------------------------------------------------------------
| Id   | Operation                          | Name                | Rows   | Bytes   | Cost  | Time     |
-----------------------------------------------------------------------------------------------------
|    0 | SELECT STATEMENT                   |                     |      1 |     126 | 11224 | 00:02:15 |
|    1 |   HASH GROUP BY                    |                     |      1 |     126 | 11224 | 00:02:15 |
|    2 |    NESTED LOOPS                    |                     |      1 |     126 | 11223 | 00:02:15 |
|  * 3 |     HASH JOIN                      |                     |      1 |     111 | 11175 | 00:02:15 |
|  * 4 |      TABLE ACCESS FULL             | LU_ELEMENTGROUP_REL |      1 |      11 |   162 | 00:00:02 |
|  * 5 |      HASH JOIN                     |                     |    487 |   48700 | 11012 | 00:02:13 |
|    6 |       MERGE JOIN                   |                     |     14 |     924 |  1068 | 00:00:13 |
|    7 |        SORT JOIN                   |                     |   5391 |  274941 |  1033 | 00:00:13 |
|  * 8 |         HASH JOIN                  |                     |   5391 |  274941 |   904 | 00:00:11 |
|  * 9 |          TABLE ACCESS FULL         | LU_ELEMENTGROUP_REL |    123 |    1353 |   175 | 00:00:03 |
| * 10 |          HASH JOIN                 |                     |   5352 |  214080 |   729 | 00:00:09 |
| * 11 |           TABLE ACCESS FULL        | LU_ITEM_293         |   5355 |  128520 |    56 | 00:00:01 |
| * 12 |           TABLE ACCESS FULL        | ADM_PG_FEATUREVALUE |   1629 |   26064 |   649 | 00:00:08 |
| * 13 |        FILTER                      |                     |        |         |       |          |
| * 14 |         SORT JOIN                  |                     |      1 |      15 |    36 | 00:00:01 |
| * 15 |          TABLE ACCESS FULL         | LU_ELEMENTRANGE_REL |      1 |      15 |    35 | 00:00:01 |
|   16 |       INLIST ITERATOR              |                     |        |         |       |          |
| * 17 |        TABLE ACCESS BY INDEX ROWID | FACT_PD_OUT_ITM_293 | 191837 | 6522458 |  9927 | 00:02:00 |
|   18 |         BITMAP CONVERSION TO ROWIDS |                    |        |         |       |          |
| * 19 |          BITMAP INDEX SINGLE VALUE | FACT_274_PER_IDX    |        |         |       |          |
| * 20 |     TABLE ACCESS FULL              | LU_ELEMENTRANGE_REL |      1 |      15 |    49 | 00:00:01 |
-----------------------------------------------------------------------------------------------------
.
.
.

Execution Plan After Change:
----------------------------
 Plan Id          : 102
 Plan Hash Value  : 1923145679

-----------------------------------------------------------------------------------------------
| Id   | Operation                          | Name                | Rows | Bytes   | Cost | Time     |
-----------------------------------------------------------------------------------------------
|    0 | SELECT STATEMENT                   |                     |    1 |     126 | 2771 | 00:00:34 |
|    1 |   HASH GROUP BY                    |                     |    1 |     126 | 2771 | 00:00:34 |
|    2 |    NESTED LOOPS                    |                     |    1 |     126 | 2770 | 00:00:34 |
|  * 3 |     HASH JOIN                      |                     |    1 |     111 | 2722 | 00:00:33 |
|  * 4 |      HASH JOIN                     |                     |    1 |     100 | 2547 | 00:00:31 |
|  * 5 |       TABLE ACCESS FULL            | LU_ELEMENTGROUP_REL |    1 |      11 |  162 | 00:00:02 |
|    6 |       NESTED LOOPS                 |                     |      |         |      |          |
|    7 |        NESTED LOOPS                |                     |  484 |   43076 | 2384 | 00:00:29 |
|  * 8 |         HASH JOIN                  |                     |   14 |     770 |  741 | 00:00:09 |
|    9 |          NESTED LOOPS              |                     |    4 |     124 |  683 | 00:00:09 |
| * 10 |           TABLE ACCESS FULL        | LU_ELEMENTRANGE_REL |    1 |      15 |   35 | 00:00:01 |
| * 11 |           TABLE ACCESS FULL        | ADM_PG_FEATUREVALUE |    4 |      64 |  649 | 00:00:08 |
| * 12 |          TABLE ACCESS FULL         | LU_ITEM_293         | 5355 |  128520 |   56 | 00:00:01 |
|   13 |         BITMAP CONVERSION TO ROWIDS |                    |      |         |      |          |
| * 14 |          BITMAP INDEX SINGLE VALUE | FACT_274_ITEM_IDX   |      |         |      |          |
| * 15 |        TABLE ACCESS BY INDEX ROWID | FACT_PD_OUT_ITM_293 |   36 |    1224 | 2384 | 00:00:29 |
| * 16 |      TABLE ACCESS FULL             | LU_ELEMENTGROUP_REL |  123 |    1353 |  175 | 00:00:03 |
| * 17 |     TABLE ACCESS FULL              | LU_ELEMENTRANGE_REL |    1 |      15 |   49 | 00:00:01 |
-----------------------------------------------------------------------------------------------
```

Comparing SQL Tuning Sets Using APIs

You can compare two SQL tuning sets using the `DBMS_SQLPA` package. For example, while using Database Replay, you may have captured a SQL tuning set on the production system during workload capture, and another SQL tuning set on a test system during workload replay. You can then use SQL Performance Analyzer to compare these SQL tuning sets, without having to re-execute the SQL statements. This is useful in cases where you already have

another utility to run your workload before and after making the system change, such as a custom script.

When comparing SQL tuning sets, SQL Performance Analyzer uses the runtime statistics captured in the SQL tuning sets to perform its comparison analysis, and reports on any new or missing SQL statements that are found in one SQL tuning set, but not in the other. Any changes in execution plans between the two SQL tuning sets are also reported. For each SQL statement in both SQL tuning sets, improvement and regression findings are reported for each SQL statement—calculated based on the average statistic value per execution—and for the entire workload—calculated based on the cumulative statistic value.

To compare SQL tuning sets using APIs:

1. Create a SQL Performance Analyzer task:

```
VAR aname varchar2(30);
EXEC :aname := 'compare_s2s';
EXEC :aname := DBMS_SQLPA.CREATE_ANALYSIS_TASK(task_name => :aname);
```

   It is not necessary to associate a SQL tuning set to the task during creation.

2. Create the first SQL trial and convert the first SQL tuning set:

```
EXEC DBMS_SQLPA.EXECUTE_ANALYSIS_TASK(task_name => :aname, -
          execution_type => 'convert sqlset', -
          execution_name => 'first trial', -
          execution_params => DBMS_ADVISOR.ARGLIST(
                              'sqlset_name', 'my_first_sts', -
                              'sqlset_owner', 'APPS'));
```

   Specify the name and owner of the SQL tuning set using the `SQLSET_NAME` and `SQLSET_OWNER` task parameters. The content of the SQL tuning set will not be duplicated by the SQL Performance Analyzer task. Instead, a reference to the SQL tuning set is recorded in association to the new SQL trial, which in this example is "first trial".

3. Create a second SQL trial and associate it to the second SQL tuning second to which you want to compare:

```
EXEC DBMS_SQLPA.EXECUTE_ANALYSIS_TASK(task_name => :aname, -
          execution_type => 'convert sqlset', -
          execution_name => 'second trial', -
          execution_params => DBMS_ADVISOR.ARGLIST(
                              'sqlset_name', 'my_second_sts', -
                              'sqlset_owner', 'APPS'));
```

4. Compare the performance data from the two SQL trials (or SQL tuning sets) by running a comparison analysis:

```
EXEC DBMS_SQLPA.EXECUTE_ANALYSIS_TASK(task_name => :aname, -
          execution_type => 'compare', -
          execution_name => 'comparison', -
          execution_params => DBMS_ADVISOR.ARGLIST(
                              'workload_impact_threshold', 0, -
                              'sql_impact_threshold', 0));
```

   In this example, the workload and per-SQL impact threshold are set to 0% for comparison (the default value is 1%).

5. After the comparison analysis is complete, generate a SQL Performance Analyzer report using the `DBMS_SQLPA.REPORT_ANALYSIS_TASK` function.

   For information about generating a SQL Performance Analyzer report using APIs, see "Analyzing SQL Performance Using APIs".

Once the report is generated, review it to identify any differences between the contents of the two SQL tuning sets. Example 6-8 shows the Analysis Information and Report Summary sections of a sample report generated by comparing two SQL tuning sets:

Example 6-8 Analysis Information and Report Summary

```
Analysis Information:
-----------------------------------------------------------------------------------------------
 Before Change Execution:                        After Change Execution:
 ----------------------------------------        ----------------------------------------
  Execution Name      : first trial               Execution Name      : second trial
  Execution Type      : CONVERT SQLSET            Execution Type      : CONVERT SQLSET
  Status              : COMPLETED                 Status              : COMPLETED
  Started             : …
  Last Updated        : …

 Before Change Workload:                         After Change Workload:
 ----------------------------------------        ----------------------------------------
  SQL Tuning Set Name        : my_first_sts       SQL Tuning Set Name        : my_second_sts
  SQL Tuning Set Owner       : APPS               SQL Tuning Set Owner       : APPS
  Total SQL Statement Count  : 5                  Total SQL Statement Count  : 6

-----------------------------------------------------------------------------------------------
Report Summary
-----------------------------------------------------------------------------------------------

Projected Workload Change Impact:
-----------------------------------------
 Overall Impact      :  72.32%
 Improvement Impact  :  47.72%
 Regression Impact   :   -.02%
 Missing-SQL Impact  :   33.1%
 New-SQL Impact      :  -8.48%

SQL Statement Count
-----------------------------------------
 SQL Category   SQL Count  Plan Change Count
 Overall                7                  1
 Common                 4                  1
  Improved              3                  1
  Regressed             1                  0
 Different              3                  0
  Missing SQL           1                  0
  New SQL               2                  0
```

As shown in Example 6-8, this report contains two additional categories that are not found in standard SQL Performance Analyzer reports; both categories are grouped under the heading Different:

- Missing SQL

  This category represents all SQL statements that are present in the first SQL tuning set, but are not found in the second SQL tuning set. In this example, only one SQL statement is missing. As shown in Example 6-9, this SQL statement has:

  – A `sql_id` value of gv7xb8tyd1v91

  – A performance impact on the workload of 33.1% based on the change

  – No performance impact on the SQL statement based on the change because its "Total Metric After" change value is missing

- New SQL

This category represents all SQL statements that are present in the second SQL tuning set, but are not found in the first SQL tuning set. In this example, only two SQL statements are new in the second SQL tuning set. As shown in Example 6-9, these SQL statements have:

- `sql_id` values of 4c8nrqxhtb2sf and 9utadgu5udmh4
- A total performance impact on the workload of -8.48%
- Missing "Total Metric Before" change values

Example 6-9 shows a table in the sample report that lists the missing and new SQL statements, as well as other top SQL statements as determined by their impact on the workload:

Example 6-9 Top 7 SQL Sorted by Absolute Value of Change Impact on the Workload

```
Top 7 SQL Sorted by Absolute Value of Change Impact on the Workload
-------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------
|           |               | Impact on | Total Metric | Total Metric | Impact   | Plan   |
| object_id | sql_id        | Workload  |    Before    |    After     | on SQL   | Change |
-------------------------------------------------------------------------------------------------
|         4 | 7gj3w9ya4d9sj |    41.04% |       812791 |        36974 |      95% | y      |
|         7 | gv7xb8tyd1v91 |     33.1% |       625582 |              |          | n      |
|         2 | 4c8nrqxhtb2sf |    -8.35% |              |       157782 |          | n      |
|         1 | 22u3tvrt0yr6g |     4.58% |       302190 |       215681 |  28.63%  | n      |
|         6 | fgdd0fd56qmt0 |      2.1% |       146128 |       106369 |  27.21%  | n      |
|         5 | 9utadgu5udmh4 |     -.13% |              |         2452 |          | n      |
|         3 | 4dtv43awxnmv3 |     -.02% |         3520 |         3890 | -47.35%  | n      |
-------------------------------------------------------------------------------------------------
```

Once you have identified a SQL statement of interest, you can generate a report for the SQL statement to perform more detailed investigation. For example, you may want to investigate the SQL statement with the `sql_id` value of 7gj3w9ya4d9sj and `object_id` value of 4 because it has the highest impact on the workload:

```
SELECT DBMS_SQLPA.REPORT_ANALYSIS_TASK(task_name => :aname, object_id => 4) rep
FROM dual;
```

Example 6-10 shows a sample report generated for this SQL statement:

Example 6-10 Sample Report for SQL Statement

```
SQL Details:
-----------------------------
 Object ID  : 4
 SQL ID     : 7gj3w9ya4d9sj
 SQL Text   : /* my_csts_query1 */ select * FROM emp where empno=2

SQL Execution Statistics (average):
---------------------------------------
-----------------------------------------------------------
|              | Impact on | Value    | Value    | Impact |
| Stat Name    | Workload  | Before   | After    | on SQL |
-----------------------------------------------------------
| elapsed_time |    41.04% | .036945 | .001849 |    95% |
| cpu_time     |    13.74% | .004772 |  .00185 | 61.24% |
| buffer_gets  |     9.59% |       8 |       2 | 69.01% |
| cost         |    11.76% |       1 |       1 |    10% |
| reads        |     4.08% |       0 |       0 | 63.33% |
| writes       |        0% |       0 |       0 |     0% |
| rows         |           |       0 |       0 |        |
```

```
| executions   |           |      22 |      20 |         |
| plan_count   |           |       3 |       2 |         |
-------------------------------------------------------------

Findings (2):
-----------------------------
 1. The performance of this SQL has improved.
 2. The structure of the SQL execution plan has changed.

Plan Execution Statistics (average):
---------------------------------------
----------------------------------------------------------------------------------------------
| Statistic Name  | Plans Before Change                | Plans After Change                  |
----------------------------------------------------------------------------------------------
| plan hash value | 440231712  571903972  3634526668 | 571903972  3634526668               |
| --------------- | ---------  ---------  ---------- | ---------  ----------               |
|  schema name    | APPS1      APPS2      APPS2      | APPS2      APPS2                    |
|  executions     | 7          5          10         | 10         10                       |
|  cost           | 2          1          2          | 1          2                        |
|  elapsed_time   | .108429    .000937    .00491     | .000503    .003195                  |
|  cpu_time       | .00957     .0012      .0032      | .0005      .0032                    |
|  buffer_gets    | 18         0          5          | 0          5                        |
|  reads          | 0          0          0          | 0          0                        |
|  writes         | 0          0          0          | 0          0                        |
|  rows           | 0          0          0          | 0          0                        |
----------------------------------------------------------------------------------------------
Execution Plans Before Change:
-----------------------------
Plan Hash Value  : 440231712
-------------------------------------------------------------------------------
| Id | Operation              | Name     | Rows | Bytes | Cost | Time     |
-------------------------------------------------------------------------------
|  0 | SELECT STATEMENT       |          |      |       |    2 |          |
|  1 |   PX COORDINATOR       |          |      |       |      |          |
|  2 |    PX SEND QC (RANDOM) | :TQ10000 |    1 |    87 |    2 | 00:00:01 |
|  3 |     PX BLOCK ITERATOR  |          |    1 |    87 |    2 | 00:00:01 |
|  4 |      TABLE ACCESS FULL | EMP      |    1 |    87 |    2 | 00:00:01 |
-------------------------------------------------------------------------------
Note
-----
- dynamic sampling used for this statement
Plan Hash Value  : 571903972
----------------------------------------------------------------------------------------
| Id | Operation                     | Name       | Rows | Bytes | Cost | Time    |
----------------------------------------------------------------------------------------
|  0 | SELECT STATEMENT              |            |      |       |    1 |         |
|  1 |   TABLE ACCESS BY INDEX ROWID | EMP        |    1 |    87 |    1 |00:00:01|
|  2 |    INDEX UNIQUE SCAN          | MY_EMP_IDX |    1 |       |    0 |         |
----------------------------------------------------------------------------------------
Plan Hash Value  : 3634526668
-----------------------------------------------------------------------
| Id | Operation           | Name | Rows | Bytes | Cost | Time     |
-----------------------------------------------------------------------
|  0 | SELECT STATEMENT    |      |      |       |    2 |          |
|  1 |   TABLE ACCESS FULL | EMP  |    1 |    87 |    2 | 00:00:01 |
-----------------------------------------------------------------------
Note
-----
- dynamic sampling used for this statement

Executions Plan After Change:
-----------------------------
Plan Hash Value  : 571903972
```

```
--------------------------------------------------------------------------------
| Id | Operation                     | Name       | Rows | Bytes | Cost | Time    |
--------------------------------------------------------------------------------
|  0 | SELECT STATEMENT              |            |      |       |    1 |         |
|  1 |   TABLE ACCESS BY INDEX ROWID | EMP        |    1 |    87 |    1 |00:00:01|
|  2 |    INDEX UNIQUE SCAN          | MY_EMP_IDX |    1 |       |    0 |         |
--------------------------------------------------------------------------------
Plan Hash Value  : 3634526668
-------------------------------------------------------------------
| Id | Operation           | Name | Rows | Bytes | Cost | Time     |
-------------------------------------------------------------------
|  0 | SELECT STATEMENT    |      |      |       |    2 |          |
|  1 |   TABLE ACCESS FULL | EMP  |    1 |    87 |    2 | 00:00:01 |
-------------------------------------------------------------------
Note
-----
- dynamic sampling used for this statement
--------------------------------------------------------------------------------
```

The SQL Execution Statistics section shows the average runtime statistics (per execution) of the SQL statement. The data in this table reveals that this SQL statement is present in both SQL tuning sets, but that it has only three execution plans in the first SQL tuning set and two execution plans in the second SQL tuning set. Furthermore, the SQL statement was executed 22 times in the first SQL tuning set, but only 20 times in the second SQL tuning set.

The Plan Execution Statistics section shows runtime statistics per execution plan (or plan hash value). The Plans Before Change column lists plans and their associated execution statistics for the first SQL tuning set; the Plans After Change columns lists these values for the second SQL tuning set. Execution plans structures for both SQL tuning sets are shown at the end of the report.

You can use these sections in the report to identify changes in execution plans between two SQL tuning sets. This is important because changes in execution plans may be a result of test changes that can have a direct impact to performance. When comparing two SQL tuning sets, SQL Performance Analyzer reports execution plan changes when a SQL statement has:

- One plan in both SQL tuning sets, but the plan structure is different
- More than one plan, and the number of plans in both SQL tuning sets are:
  - The same, but at least one plan in the second SQL tuning set is different from all plans in the first SQL tuning set
  - Different

After evaluating the SQL statement and plan changes, determine if further action is required. If the SQL statement has regressed, perform one of the following actions:

- Tune the regressed SQL statement, as described in "Tuning Regressed SQL Statements Using APIs"
- Create SQL plan baselines, as described in "Creating SQL Plan Baselines Using APIs"

Tuning Regressed SQL Statements Using APIs

After reviewing the SQL Performance Analyzer report, you should tune any regressed SQL statements that are identified after comparing the SQL performance. If there are large numbers of SQL statements that appear to have regressed, you should try to identify the root cause and make system-level changes to rectify the problem. In cases when only a few SQL statements have regressed, consider using the SQL Tuning Advisor to implement a point solution for them,

or creating SQL plan baselines to instruct the optimizer to select the original execution plan in the future.

To tune regressed SQL statements using APIs:

- Create a SQL tuning task for the SQL Performance Analyzer execution by using the `CREATE_TUNING_TASK` function in the `DBMS_SQLTUNE` package:

```
BEGIN
  DBMS_SQLTUNE.CREATE_TUNING_TASK(
    spa_task_name => 'my_spa_task',
    spa_task_owner => 'immchan',
    spa_compare_exec => 'my_exec_compare');
  DBMS_SQLTUNE.EXECUTE_TUNING_TASK(spa_task_name => 'my_spa_task');
END;
/
```

  This example creates and executes a SQL tuning task to tune the SQL statements that regressed in the compare performance execution named `my_exec_compare` of the SQL Performance Analyzer task named `my_spa_task`. In this case, it is important to use this version of the `CREATE_TUNING_TASK` function call. Otherwise, SQL statements may be tuned in the environment from the production system where they were captured, which will not reflect the system change.

> **Note:**
>
> If you chose to execute the SQL workload remotely on a separate database, you should not use this version of the `CREATE_TUNING_TASK` function call to tune regressed SQL statements. Instead, you should tune any regressions identified by the SQL trials on the remote database, because the application schema is not on the database running SQL Performance Analyzer. Therefore, you need to run SQL Tuning Advisor on the database where the schema resides and where the change was made.

Table 6-1 lists the SQL Performance Analyzer parameters that can be used with the `DBMS_SQLTUNE.CREATE_TUNING_TASK` function.

Table 6-1 CREATE_TUNING_TASK Function SQL Performance Analyzer Parameters

Parameter	Description
`SPA_TASK_NAME`	Name of the SQL Performance Analyzer task.
`SPA_TASK_OWNER`	Owner of the specified SQL Performance Analyzer task. If unspecified, this parameter will default to the current user.
`SPA_COMPARE_EXEC`	Execution name of the compare performance trial for the specified SQL Performance Analyzer task. If unspecified, this parameter defaults to the most recent execution of the `COMPARE PERFORMANCE` type for the given SQL Performance Analyzer task.

After tuning the regressed SQL statements, you should test these changes using SQL Performance Analyzer. Run a new SQL trial on the test system, followed by a second comparison (between this new SQL trial and the first SQL trial) to validate your results. Once SQL Performance Analyzer shows that performance has stabilized, implement the fixes from this step to your production system.

Starting with Oracle Database 11*g* Release 2, SQL Tuning Advisor performs an alternative plan analysis when tuning a SQL statement. SQL Tuning Advisor reviews the execution history of the SQL statement, including any historical plans stored in the Automatic Workload Repository. If SQL Tuning Advisor finds alternate plans, it allows you to choose a specific plan and create a plan baseline to ensure that the desired execution plan is used for that SQL statement.

See Also:

- "Tuning Regressed SQL Statements From a Remote SQL Trial Using APIs"
- *Oracle Database SQL Tuning Guide* for information about using the SQL Tuning Advisor
- *Oracle Database SQL Tuning Guide* for information about alternative plan analysis
- *Oracle Database PL/SQL Packages and Types Reference* for information about the `DBMS_SQLTUNE` package

Tuning Regressed SQL Statements From a Remote SQL Trial Using APIs

If you chose to execute the SQL workload remotely on a separate database, then you should tune any regressions identified by the SQL trials on the remote database, instead of the system where the SQL Performance Analyzer task resides.

To tune regressed SQL statements from a remote SQL trial using APIs:

1. On the system running SQL Performance Analyzer, create a subset of the regressed SQL statements as a SQL tuning set:

```
DECLARE
  sqlset_cur  DBMS_SQLTUNE.SQLSET_CURSOR;
BEGIN
  DBMS_SQLTUNE.CREATE_SQLSET('SUB_STS1', 'test purpose');

  OPEN sqlset_cur FOR
    SELECT value(p)
    FROM table(
      DBMS_SQLTUNE.SELECT_SQLPA_TASK(
        task_name  => 'SPA_TASK1',
        execution_name => 'COMP',
        level_filter => 'REGRESSED')) p;

  DBMS_SQLTUNE.LOAD_SQLSET('SUB_STS1', sqlset_cur);

  CLOSE sqlset_cur;
END;
/
```

   Other than `'REGRESSED'`, you can use other filters to select SQL statements for the SQL tuning set, such as `'CHANGED'`, `'ERRORS'`, or `'CHANGED_PLANS'`. For more information, see *Oracle Database PL/SQL Packages and Types Reference*.

2. Create a staging table to where the SQL tuning set will be exported:

```
BEGIN
  DBMS_SQLTUNE.CREATE_STGTAB_SQLSET(
```

```
    table_name  => 'STG_TAB1',
    schema_name => 'JOHNDOE',
    tablespace_name => 'TBS_1',
    db_version => DBMS_SQLTUNE.STS_STGTAB_11_1_VERSION);
END;
/
```

Use the `db_version` parameter to specify the appropriate database version to where the SQL tuning set will be exported and tuned. In this example, the staging table will be created with a format so that it can be exported to a system running Oracle Database 11*g* Release 1, where it will later be tuned using SQL Tuning Advisor. For other database versions, see *Oracle Database PL/SQL Packages and Types Reference* for that release.

3. Export the SQL tuning set into the staging table:

```
BEGIN
  DBMS_SQLTUNE.PACK_STGTAB_SQLSET(
    sqlset_name => 'SUB_STS1',
    sqlset_owner => 'JOHNDOE',
    staging_table_name => 'STG_TAB1',
    staging_schema_owner => 'JOHNDOE',
    db_version => DBMS_SQLTUNE.STS_STGTAB_11_1_VERSION);
END;
/
```

4. Move the staging table to the remote database (where the SQL workload was executed) using the mechanism of choice (such as Oracle Data Pump or database link).

5. On the remote database, import the SQL tuning set from the staging table:

```
BEGIN
  DBMS_SQLTUNE.UNPACK_STGTAB_SQLSET(
    sqlset_name => 'SUB_STS1',
    staging_table_name => 'STG_TAB1',
    replace => TRUE);
END;
/
```

6. Tune the regressed SQL statements in the SQL tuning set by running SQL Tuning Advisor:

```
BEGIN
  sts_name  := 'SUB_STS1';
  sts_owner := 'JOHNDOE';
  tune_task_name := 'TUNE_TASK1';
  tname := DBMS_SQLTUNE.CREATE_TUNING_TASK(sqlset_name  => sts_name,
                                           sqlset_owner => sts_owner,
                                           task_name    => tune_task_name);
  EXEC DBMS_SQLTUNE.SET_TUNING_TASK_PARAMETER(:tname,
                                              'APPLY_CAPTURED_COMPILENV',
                                              'FALSE');
  exec_name := DBMS_SQLTUNE.EXECUTE_TUNING_TASK(tname);
END;
/
```

> **Note:**
>
> The `APPLY_CAPTURED_COMPILENV` parameter used in this example is only supported by Oracle Database 11*g* Release 1 and newer releases. If you are testing a database upgrade from an earlier version of Oracle Database, SQL Tuning Advisor will use the environment variables stored in the SQL tuning set instead.

After tuning the regressed SQL statements, you should test these changes using SQL Performance Analyzer. Run a new SQL trial on the test system, followed by a second comparison (between this new SQL trial and the first SQL trial) to validate your results. Once SQL Performance Analyzer shows that performance has stabilized, implement the fixes from this step to your production system.

> **See Also:**
>
> - *Oracle Database SQL Tuning Guide* for information about using the SQL Tuning Advisor and transporting SQL tuning sets
> - *Oracle Database PL/SQL Packages and Types Reference* for information about the `DBMS_SQLTUNE` package

Creating SQL Plan Baselines Using APIs

Creating SQL plan baselines for regressed SQL statements with plan changes is another option to running the SQL Tuning Advisor. Doing so instructs the optimizer to use the original execution plans for these SQL statements in the future.

To create SQL plan baselines for the original plans:

1. Create a subset of a SQL tuning set of only the regressed SQL statements.
2. Create SQL plan baselines for this subset of SQL statements by loading their plans using the `LOAD_PLANS_FROM_SQLSET` function of the `DBMS_SPM` package, as shown in the following example:

```
DECLARE
  my_plans PLS_INTEGER;
BEGIN
  my_plans := DBMS_SPM.LOAD_PLANS_FROM_SQLSET(
    sqlset_name => 'regressed_sql');
END;
/
```

> **See Also:**
>
> - *Oracle Database SQL Tuning Guide* for information about using SQL plan baselines
> - *Oracle Database PL/SQL Packages and Types Reference* for information about the `DBMS_SPM` package

Using SQL Performance Analyzer Views

You can query the following views to monitor SQL Performance Analyzer and view its analysis results:

> **Note:**
>
> The information available in these views are also contained in the SQL Performance Analyzer report. It is recommended that you use the SQL Performance Analyzer report to view analysis results instead. Consider using these views only for performing more advanced analysis of the results.

- The `DBA_ADVISOR_TASKS` and `USER_ADVISOR_TASKS` views display descriptive information about the SQL Performance Analyzer task that was created.
- The `DBA_ADVISOR_EXECUTIONS` and `USER_ADVISOR_EXECUTIONS` views display information about task executions. SQL Performance Analyzer creates at least three executions to analyze the SQL performance impact caused by a database change on a SQL workload. The first execution collects a pre-change version of the performance data. The second execution collects a post-change version of the performance data. The third execution performs the comparison analysis.
- The `DBA_ADVISOR_FINDINGS` and `USER_ADVISOR_FINDINGS` views display the SQL Performance Analyzer findings. SQL Performance Analyzer generates the following types of findings:
  - Problems, such as performance regression
  - Symptoms, such as when the structure of an execution plan has changed
  - Errors, such as nonexistence of an object or view
  - Informative messages, such as when the structure of an execution plan in the pre-change version is different than the one stored in the SQL tuning set
- The `DBA_ADVISOR_SQLPLANS` and `USER_ADVISOR_SQLPLANS` views display a list of all execution plans.
- The `DBA_ADVISOR_SQLSTATS` and `USER_ADVISOR_SQLSTATS` views display a list of all SQL compilations and execution statistics.
- The `V$ADVISOR_PROGRESS` view displays the operation progress of SQL Performance Analyzer. Use this view to monitor how many SQL statements have completed or are awaiting execution in a SQL trial. The `SOFAR` column indicates the number of SQL statements processed so far, and the `TOTAL WORK` column shows the total number of SQL statements to be processed by the task execution.

You must have the `SELECT_CATALOG_ROLE` role to access the DBA views.

> **See Also:**
>
> - *Oracle Database Reference* for information about the `DBA_ADVISOR_TASKS`, `DBA_ADVISOR_EXECUTIONS`, and `DBA_ADVISOR_SQLPLANS` views

7

Using SPA Quick Check

Oracle Enterprise Manager Cloud Control (Cloud Control) includes the SQL Performance Analyzer Quick Check (SPA Quick Check) feature. On some Cloud Control database management pages, SPA Quick Check can validate the impact of a system change to the database workload before you make the change.

You can use SPA Quick Check to validate what the impact to your database workload will be for the following changes:

- Changing the value of an initialization parameter
- Gathering pending optimizer statistics
- Implementing key SQL profiles

> **Note:**
>
> SPA Quick Check is available starting with Cloud Control Release 12.1.0.4 Bundle Patch 8 and later.
>
> SPA Quick Check is supported for any database running Oracle Database 10*g* Release 2 (10.2) and later. However, not all the SPA Quick Check features are supported in Oracle Database 10*g* Release 2 (10.2).
>
> For example, optimizer pending statistics and Automatic SQL Tuning Advisor features are available starting with Oracle Database 11*g* Release 1 (11.1), so SPA Quick Check workflows for these features are only supported for databases running Oracle Database 11*g* Release 1 (11.1) and later.

This chapter describes how to use SPA Quick Check and contains the following topics:

About Configuring SPA Quick Check

Before you can use SPA Quick Check to validate the impact of an initialization parameter change or of gathering pending optimizer statistics, you must specify default settings for SPA Quick Check.

You will specify a default SQL tuning set for SPA Quick Check to use as one of the settings, and this SQL tuning set should include SQL statements used in the database application you are trying to tune.

> **Note:**
>
> It is not necessary to set default values for SPA Quick Check before using SPA Quick Check to validate the impact of implementing one or more key SQL profiles.

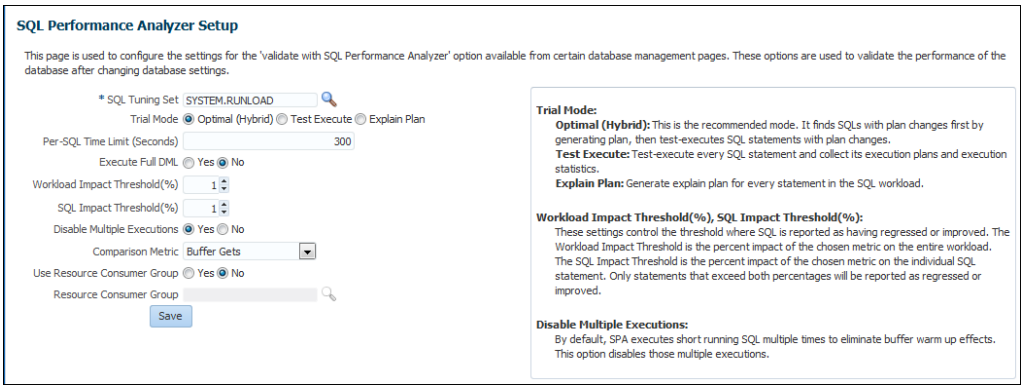
Specifying Default Values for SPA Quick Check

You specify default settings for SPA Quick Check on the SQL Performance Analyzer Setup page in Cloud Control.

To specify default settings for SPA Quick Check:

1. On the Database Home page in Cloud Control, from the **Performance** menu, select **SQL**, then **SQL Performance Analyzer Setup**. If the Database Login page appears, enter administrator privileges for the database, then click **Login**.

   The SQL Performance Analyzer Setup page appears.



2. Configure the settings for the SPA Quick Check feature, which is available on some Cloud Control database management pages. The SQL tuning set that you specify should be representative of the workload for the application that you want to tune.
3. Click **Save** to save the default SPA Quick Check settings you specified.

Validating the Impact of an Initialization Parameter Change

Before you change the value of a session-modifiable initialization parameter, you can validate the impact of that change on your database workload by using SPA Quick Check. Session-modifiable parameters are initialization parameters whose values can be changed using the `ALTER SESSION` statement.

> **Note:**
>
> You can use SPA Quick Check to validate the impact of an initialization parameter change in databases running Oracle Database 10*g* Release 2 (10.2) and later.

To validate the impact of an initialization parameter change:

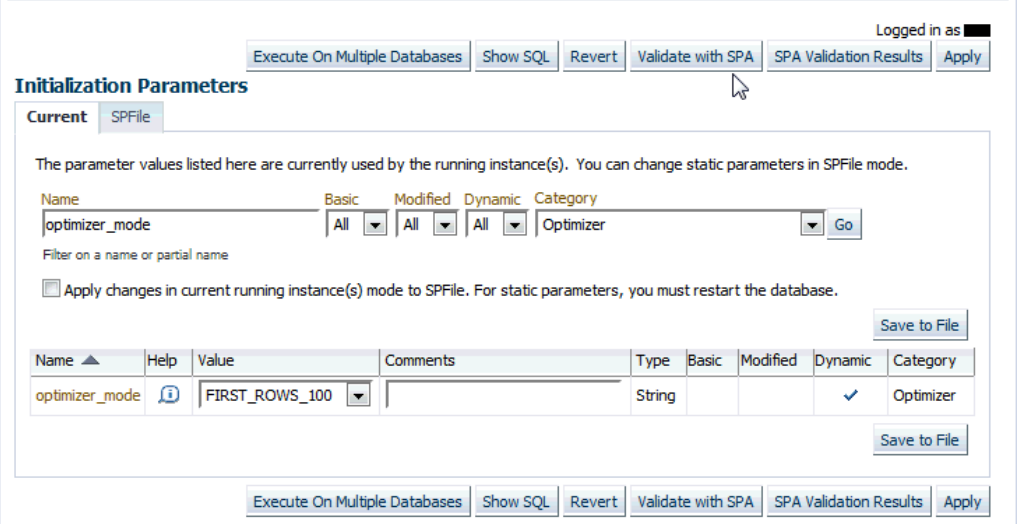
1. On the Database Home page in Cloud Control, from the **Administration** menu, select **Initialization Parameters**.

   The Initialization Parameters page appears.

2. Use the filter on the Initialization Parameters page to identify the session-modifiable initialization parameter whose value you want to change, and click **Go** to display that parameter in the table at the bottom of the page. Most of the parameters in the Optimizer category are session-modifiable.

3. In the table, change the current value of the parameter to the new value whose impact you would like to validate using SPA Quick Check.

4. Click **Validate with SPA**.



   An Information message appears at the top of the page, and says that a SPA task for validating the impact of the initialization parameter change has been submitted.

5. Click the link for the SPA task in the Information message.

   The SQL Performance Analyzer Home page appears.

6. In the SQL Performance Analyzer Tasks section at the bottom of the page, select the task for the initialization parameter job, and click **View Latest Report**.

   The SQL Performance Analyzer Task Report page appears.

7. View the table at the bottom of the page to see what the result of changing the initialization parameter's value would be on the most impactful SQL statements in the workload.

Validating the Impact of Pending Optimizer Statistics

Before you gather pending optimizer statistics, you can validate the impact of gathering those statistics on your database workload by using SPA Quick Check.

> **Note:**
>
> You can use SPA Quick Check to validate the impact of gathering pending optimizer statistics in databases running Oracle Database 11*g* Release 1 (11.1) and later.

To validate the impact of gathering pending optimizer statistics:

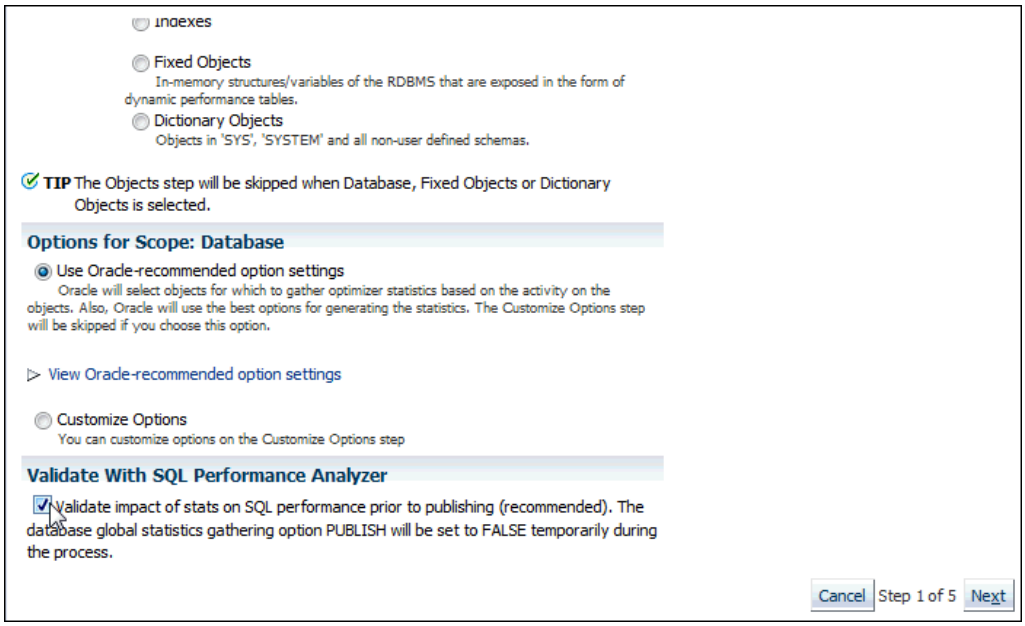
1. On the Database Home page in Cloud Control, from the **Performance** menu, select **SQL**, and then **Optimizer Statistics**.

   The Optimizer Statistics Console page appears.

2. In the Operations section, click **Gather**.

   The Gather Optimize Statistics wizard appears.

3. In the Validate with SQL Performance Analyzer section at the bottom of the Gather Optimizer Statistics: Scope page, enable the **Validate impact of stats on SQL performance prior to publishing (recommended)** option. The database global statistics gathering option PUBLISH will be set to FALSE temporarily during the process. Then click **Next**.



4. Continue through the wizard, and on the Gather Optimizer Statistics: Scope page, click **Submit**.

   Along with gathering pending statistics, this starts a job that creates a SQL Performance Analyzer task that validates the impact of gathering optimizer statistics for the database.

5. When the job starts, a Confirmation message appears on the Manage Optimizer Statistics page that says that the Gather Optimizer Statistics job has been successfully submitted. Click the link in that message.

   The SQL Performance Analyzer Home page appears.

6. In the SQL Performance Analyzer Tasks table at the bottom of the page, make sure that the statistics gathering job has completed. It may take several minutes for the job to complete. Then select the row for the Gather Optimizer Statistics job and click **View Latest Report**.

   The SQL Performance Analyzer Task Report page appears.

7. View the table at the bottom of the page to see what the result of publishing the pending optimizer statistics would be on the most impactful SQL statements in the workload.

Validating the Impact of Implementing Key SQL Profiles

Before you implement key SQL profiles for SQL statements, you can validate the impact of using those profiles by using SPA Quick Check. You can validate the impact of key SQL profiles on the Automatic SQL Tuning Result Summary page. Key SQL profiles are profiles verified to yield at least a 3 times performance improvement, and which would have been implemented automatically if auto-implementation had been enabled for Automatic SQL Tuning Advisor.

> **Note:**
>
> You can use SPA Quick Check to validate the impact of implementing key SQL profiles in databases running Oracle Database 11*g* Release 1 (11.1) and later.

To validate the impact of key SQL profiles:

1. On the Database Home page in Cloud Control, from the **Performance** menu, select **Advisors Home**.

   The Advisor Central page appears.

2. In the Advisors section, click **SQL Advisors**.

   The SQL Advisors page appears.

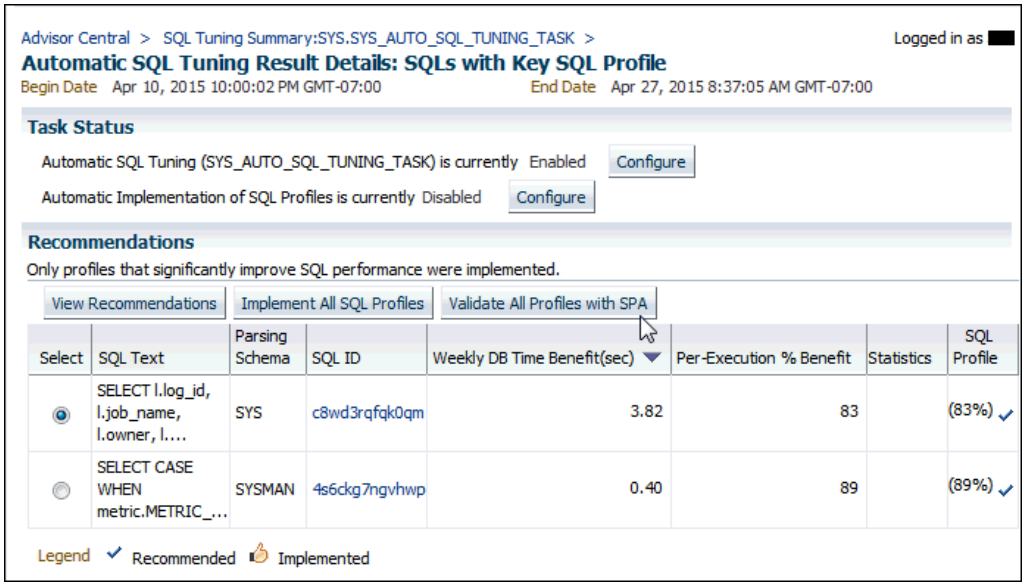
3. In the SQL Tuning Advisor section, click **Automatic SQL Tuning Results**.

   The Automatic SQL Tuning Result Summary page appears.

4. The **Key SQL Profiles** field in the Task Status section lists the number of key SQL profiles for the current automatic SQL tuning task. If a value of `0` appears in the field, there are no key SQL profiles to use (or validate). If a value greater than `0` appears in the **Key SQL Profiles** field, click the value to validate the impact of using the key SQL profile or profiles.

   The Automatic SQL Tuning Result Details: SQLs with Key SQL Profile page appears.

5. The key SQL profiles appear in the Recommendations section. Click **Validate All Profiles with SPA**.



A Confirmation statement appears at the top of the page that indicates that a SPA task for validating the SQL profiles has been submitted.

6. Click the link for the SPA task in the Confirmation statement.

   The SQL Performance Analyzer Home page appears, and the SPA task for validating the key SQL profiles appears in the SQL Performance Analyzer Tasks table at the bottom of the page.

7. Select the task and click **View Latest Report**.

   The SQL Performance Analyzer Task Report page appears.

8. View the table at the bottom of the page to see what the result would be of implementing the key SQL profiles recommended on the Automatic SQL Tuning Result Summary page on the most impactful SQL statements in the workload.

Validating Statistics Findings from Automatic SQL Tuning Advisor

You can validate the impact of statistics findings from Automatic SQL Tuning Advisor using SPA Quick Check.

> **Note:**
>
> You can use SPA Quick Check to validate the impact of validating statistics findings from Automatic SQL Tuning Advisor in databases running Oracle Database 11*g* Release 1 (11.1) and later.

To validate the impact of statistics findings from Automatic SQL Tuning Advisor:

1. On the Database Home page in Cloud Control, from the **Performance** menu, select **Advisors Home**.

   The Advisor Central page appears.

2. In the Advisors section, click **SQL Advisors**.

The SQL Advisors page appears.

3. In the SQL Tuning Advisor section, click **Automatic SQL Tuning Results**.

   The Automatic SQL Tuning Result Summary page appears.

4. If any statistics findings are available, they appear in the Statistics Finding Summary section near the bottom of the page. To validate the impact of statistics findings in user schemas, click **Validate with SPA**.

   A Confirmation statement appears at the top of the page that indicates a SPA task for validating the statistics findings has been submitted.

5. Click the link for the SPA task in the Confirmation statement.

   The SQL Performance Analyzer Home page appears, and the SPA task for validating the statistics findings appears in the SQL Performance Analyzer Tasks table at the bottom of the page.

6. After all the steps in the task have completed successfully, and **Completed** appears in the **Last Run Status** column of the table, select the task and click **View Latest Report**. It may take several minutes for all of the steps in the task to complete.

   The SQL Performance Analyzer Task Report page appears.

7. View the table at the bottom of the page to see what the result would be of implementing the statistics on the Automatic SQL Tuning Result Summary page on the most impactful SQL statements in the workload.

8

Testing a Database Upgrade

SQL Performance Analyzer supports testing database upgrades from Oracle9*i* and later releases to Oracle Database 10*g* Release 2 or newer releases. The methodology used to test a database upgrade from Oracle9*i* Database and Oracle Database 10*g* Release 1 is slightly different from the one used to test a database upgrade from Oracle Database 10*g* Release 2 and later releases, so both methodologies are described here.
This chapter describes how to use SQL Performance Analyzer in a database upgrade and contains the following sections:

- Upgrading from Oracle9*i* Database and Oracle Database 10*g* Release 1
- Upgrading from Oracle Database 10*g* Release 2 and Newer Releases
- Tuning Regressed SQL Statements After Testing a Database Upgrade

> **See Also:**
>
> - "SQL Performance Analyzer" for information about using SQL Performance Analyzer in other cases
> - *Oracle Database Upgrade Guide* for information on upgrade paths for Oracle Database 12*c*

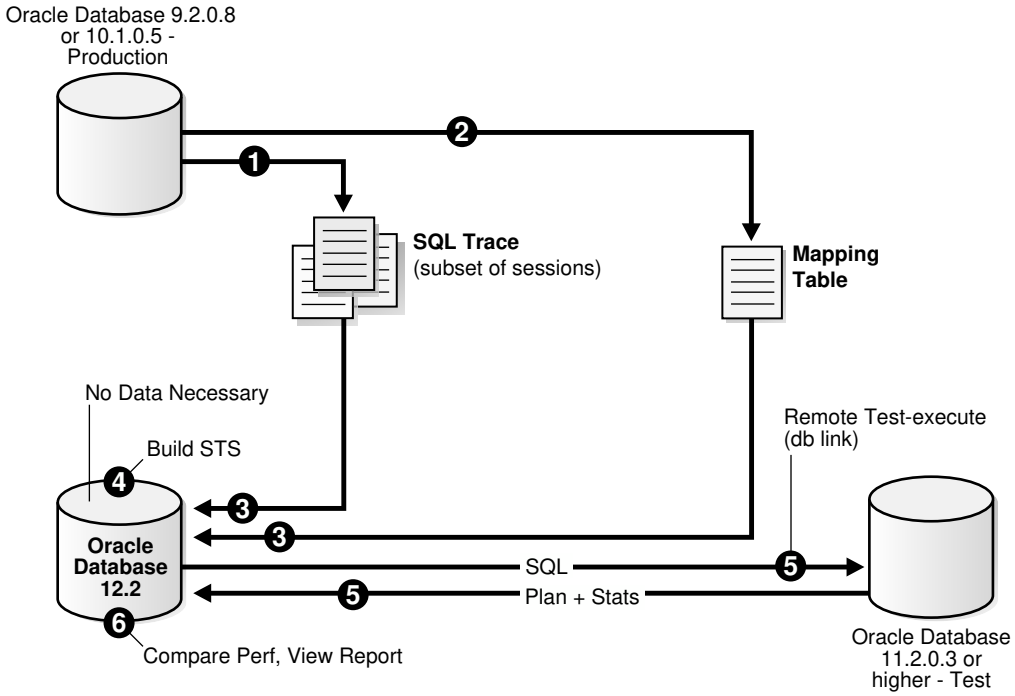
Upgrading from Oracle9*i* Database and Oracle Database 10*g* Release 1

SQL Performance Analyzer supports testing database upgrades of Oracle9*i* Database and Oracle Database 10*g* Release 1 to Oracle Database 11g Release 2 or higher releases. Use the following steps and see Figure 8-1:

- Building a SQL tuning set from SQL trace files captured on the production system
- Executing the SQL tuning set on the upgraded database remotely over a database link
- Comparing the results to those captured on the production system

Because SQL Performance Analyzer only accepts a set of SQL statements stored in a SQL tuning set as its input source, and SQL tuning sets are not supported for Oracle9*i* Database, a SQL tuning set must be constructed so that it can be used as an input source for SQL Performance Analyzer if you are upgrading from Oracle9*i* Database.

Figure 8-1 SQL Performance Analyzer Workflow for Database Upgrade from Oracle9i or 10g Release 1 to Oracle Database 11g Release 2 or Higher



Before the database upgrade can be tested, ensure that the following conditions are met:

- The production system which you are upgrading from is running Oracle9*i* (9.2.0.8) or Oracle Database 10*g* Release 1 (10.1.0.5).
- The test system which you are upgrading to is running Oracle Database 11*g* Release 2 or higher.

  The database version can be release 11.2.0.3 or higher.
- The test system must resemble the production system as closely as possible because the performance on both systems will be compared to each other.
- The hardware configurations on both systems must also be as similar as possible.

You will also need to set up a separate SQL Performance Analyzer system running Oracle Database 12*c* Release 2. You will be using this system to build a SQL tuning set and to run SQL Performance Analyzer. Neither your production data or schema need to be available on this system, since the SQL tuning set will be built using statistics stored in the SQL trace files from the production system. SQL Performance Analyzer tasks will be executed remotely on the test system to generate the execution plan and statistics for the SQL trial over a database link that you specify. The database link must be a public database link that connects to a user with the `EXECUTE` privilege for the `DBMS_SQLPA` package and the `ADVISOR` privilege on the test system. You should also drop any existing `PLAN_TABLE` from the user's schema on the test system.

Once the upgrade environment is configured as described, perform the steps as described in the following procedure to use SQL Performance Analyzer in a database upgrade from Oracle9*i* or Oracle Database 10*g* Release 1 to a newer release.

1. Enable the SQL Trace facility on the production system, as described in "Enabling SQL Trace on the Production System".

To minimize the performance impact on the production system and still be able to fully capture a representative set of SQL statements, consider enabling SQL Trace for only a subset of the sessions, for as long as required, to capture all important SQL statements at least once.

2. On the production system, create a mapping table, as described in "Creating a Mapping Table".

   This mapping table will be used to convert the user and object identifier numbers in the SQL trace files to their string equivalents.

3. Move the SQL trace files and the mapping table from the production system to the SQL Performance Analyzer system, as described in "Creating a Mapping Table".

4. On the SQL Performance Analyzer system, construct a SQL tuning set using the SQL trace files, as described in "Building a SQL Tuning Set".

   The SQL tuning set will contain the SQL statements captured in the SQL trace files, along with their relevant execution context and statistics.

5. On the SQL Performance Analyzer system, use SQL Performance Analyzer to create a SQL Performance Analyzer task and convert the contents in the SQL tuning set into a pre-upgrade SQL trial that will be used as a baseline for comparison, then remotely test execute the SQL statements on the test system over a database link to build a post-upgrade SQL trial, as described in "Testing Database Upgrades from Oracle9*i* Database and Oracle Database 10*g* Release 1".

6. Compare SQL performance and fix regressed SQL.

   SQL Performance Analyzer compares the performance of SQL statements read from the SQL tuning set during the pre-upgrade SQL trial to those captured from the remote test execution during the post-upgrade SQL trial. A report is produced to identify any changes in execution plans or performance of the SQL statements.

   If the report reveals any regressed SQL statements, you can make further changes to fix the regressed SQL, as described in "Tuning Regressed SQL Statements After Testing a Database Upgrade".

   Repeat the process of executing the SQL tuning set and comparing its performance to a previous execution to test any changes made until you are satisfied with the outcome of the analysis.

Enabling SQL Trace on the Production System

Oracle9*i* uses the SQL Trace facility to collect performance data on individual SQL statements. The information generated by SQL Trace is stored in SQL trace files. SQL Performance Analyzer consumes the following information from these files:

- SQL text and username under which parse occurred
- Bind values for each execution
- CPU and elapsed times
- Physical reads and logical reads
- Number of rows processed
- Execution plan for each SQL statement (only captured if the cursor for the SQL statement is closed)

Although it is possible to enable SQL Trace for an instance, it is recommended that you enable SQL Trace for a subset of sessions instead. When the SQL Trace facility is enabled for an instance, performance statistics for all SQL statements executed in the instance are stored into

SQL trace files. Using SQL Trace in this way can have a severe performance impact and may result in increased system overhead, excessive CPU usage, and inadequate disk space. It is required that trace level be set to 4 to capture bind values, along with the execution plans.

For production systems running Oracle Database 10*g* Release 1, use the `DBMS_MONITOR.SESSION_TRACE_ENABLE` procedure to enable SQL Trace transparently in another session. You should also enable binds explicitly by setting the `binds` procedure parameter to `TRUE` (its default value is `FALSE`).

After enabling SQL Trace, identify the SQL trace files containing statistics for a representative set of SQL statements that you want to use with SQL Performance Analyzer. You can then copy the SQL trace files to the SQL Performance Analyzer system. Once the SQL workload is captured in the SQL trace files, disable SQL Trace on the production system.

See Also:

- *Oracle Database SQL Tuning Guide* for additional considerations when using SQL Trace, such as setting initialization parameters to manage SQL trace files
- *Oracle Database PL/SQL Packages and Types Reference* for information about the `DBMS_MONITOR` package

Creating a Mapping Table

To convert the user and object identifier numbers stored in the SQL trace files to their respective names, you need to provide a table that specifies each mapping. The SQL Performance Analyzer system will read this mapping table when converting the trace files into a SQL tuning set.

To create a mapping table:

- Run the following SQL statements on the production database:

```
CREATE TABLE mapping AS
    SELECT object_id id, owner, SUBSTR(object_name, 1, 30) name FROM dba_objects
    WHERE object_type NOT IN ('CONSUMER GROUP', 'EVALUATION CONTEXT', 'FUNCTION',
                              'INDEXTYPE', 'JAVA CLASS', 'JAVA DATA',
                              'JAVA RESOURCE', 'LIBRARY', 'LOB', 'OPERATOR',
                              'PACKAGE', 'PACKAGE BODY', 'PROCEDURE', 'QUEUE',
                              'RESOURCE PLAN', 'SYNONYM', 'TRIGGER', 'TYPE',
                              'TYPE BODY')
    UNION ALL
    SELECT user_id id, username owner, null name FROM dba_users;
```

Once the mapping table is created, you can use Data Pump to transport it to the SQL Performance Analyzer system.

See Also:

- *Oracle Database Utilities* for information about using Data Pump

Building a SQL Tuning Set

Once the SQL trace files and mapping table are moved to the SQL Performance Analyzer system, you can build a SQL tuning set using the `DBMS_SQLTUNE` package.

To build a SQL tuning set:

1. Copy the SQL trace files to a directory on the SQL Performance Analyzer system.
2. Create a directory object for this directory.
3. Use the `DBMS_SQLTUNE.SELECT_SQL_TRACE` function to read the SQL statements from the SQL trace files.

   For each SQL statement, only information for a single execution is collected. The execution frequency of each SQL statement is not captured. Therefore, when performing a comparison analysis for a production system running Oracle Database 10*g* Release 1 and older releases, you should ignore the workload-level statistics in the SQL Performance Analyzer report and only evaluate performance changes on an execution level.

   The following example reads the contents of SQL trace files stored in the `sql_trace_prod` directory object and loads them into a SQL tuning set.

```
DECLARE
  cur sys_refcursor;
BEGIN
  DBMS_SQLTUNE.CREATE_SQLSET('my_sts_9i');
  OPEN cur FOR
    SELECT VALUE (P)
    FROM table(DBMS_SQLTUNE.SELECT_SQL_TRACE('sql_trace_prod', '%ora%')) P;
  DBMS_SQLTUNE.LOAD_SQLSET('my_sts_9i', cur);
  CLOSE cur;
END;
/
```

The syntax for the `SELECT_SQL_TRACE` function is as follows:

```
DBMS_SQLTUNE.SELECT_SQL_TRACE (
  directory              IN VARCHAR2,
  file_name              IN VARCHAR2 := NULL,
  mapping_table_name     IN VARCHAR2 := NULL,
  mapping_table_owner    IN VARCHAR2 := NULL,
  select_mode            IN POSITIVE := SINGLE_EXECUTION,
  options                IN BINARY_INTEGER := LIMITED_COMMAND_TYPE,
  pattern_start          IN VARCHAR2 := NULL,
  parttern_end           IN VARCHAR2 := NULL,
  result_limit           IN POSITIVE := NULL)
RETURN sys.sqlset PIPELINED;
```

Table 8-1 describes the available parameters for the `SELECT_SQL_TRACE` function.

Table 8-1 DBMS_SQLTUNE.SELECT_SQL_TRACE Function Parameters

Parameter	Description
`directory`	Specifies the directory object pointing to the directory where the SQL trace files are stored.

Table 8-1 (Cont.) DBMS_SQLTUNE.SELECT_SQL_TRACE Function Parameters

Parameter	Description
`file_name`	Specifies all or part of the name of the SQL trace files to process. If unspecified, the current or most recent trace file in the specified directory will be used. % wildcards are supported for matching trace file names.
`mapping_table_name`	Specifies the name of the mapping table. If set to the default value of `NULL`, mappings from the current database will be used. Note that the mapping table name is not case-sensitive.
`mapping_table_owner`	Specifies the schema where the mapping table resides. If set to `NULL`, the current schema will be used.
`select_mode`	Specifies the mode for selecting SQL statements from the trace files. The default value is `SINGLE_EXECUTION`. In this mode, only statistics for a single execution per SQL statement will be loaded into the SQL tuning set. The statistics are not cumulative, as is the case with other SQL tuning set data source table functions.
`options`	Specifies the options for the operation. The default value is `LIMITED_COMMAND_TYPE`, only SQL types that are meaningful to SQL Performance Analyzer (such as `SELECT`, `INSERT`, `UPDATE`, and `DELETE`) are returned from the SQL trace files.
`pattern_start`	Specifies the opening delimiting pattern of the trace file sections to consider. This parameter is currently not used.
`pattern_end`	Specifies the closing delimiting pattern of the trace file sections to process. This parameter is currently not used.
`result_limit`	Specifies the top SQL from the (filtered) source. The default value is 2^{31}, which represents unlimited.

See Also:

- *Oracle Database PL/SQL Packages and Types Reference* for information about the `DBMS_SQLTUNE` package

Testing Database Upgrades from Oracle9*i* Database and Oracle Database 10*g* Release 1

Once the SQL tuning set is built, you can use SQL Performance Analyzer to build a pre-upgrade SQL trial from the execution plans and run-time statistics in the SQL tuning set. After the pre-upgrade SQL trial is built, perform a test execute or generate plans of SQL statements in the SQL tuning set on the test system to build a post-upgrade SQL trial. SQL Performance Analyzer test executes the SQL statements using a public database link that you specify by connecting to the test system remotely and generating the execution plans and statistics for the SQL trial. The database link should exist on the SQL Performance Analyzer system and connect to a remote user with privileges to execute the SQL tuning set on the test system. You can run SQL Performance Analyzer to test a database upgrade from Oracle9*i* Database or Oracle Database 10*g* Release 1 using Oracle Enterprise Manager or APIs, as described in the following sections:

- Testing Database Upgrades from Releases 9.x and 10.1 Using Cloud Control

- Testing Database Upgrades from Releases 9.x and 10.1 Using APIs

Testing Database Upgrades from Releases 9.x and 10.1 Using Cloud Control

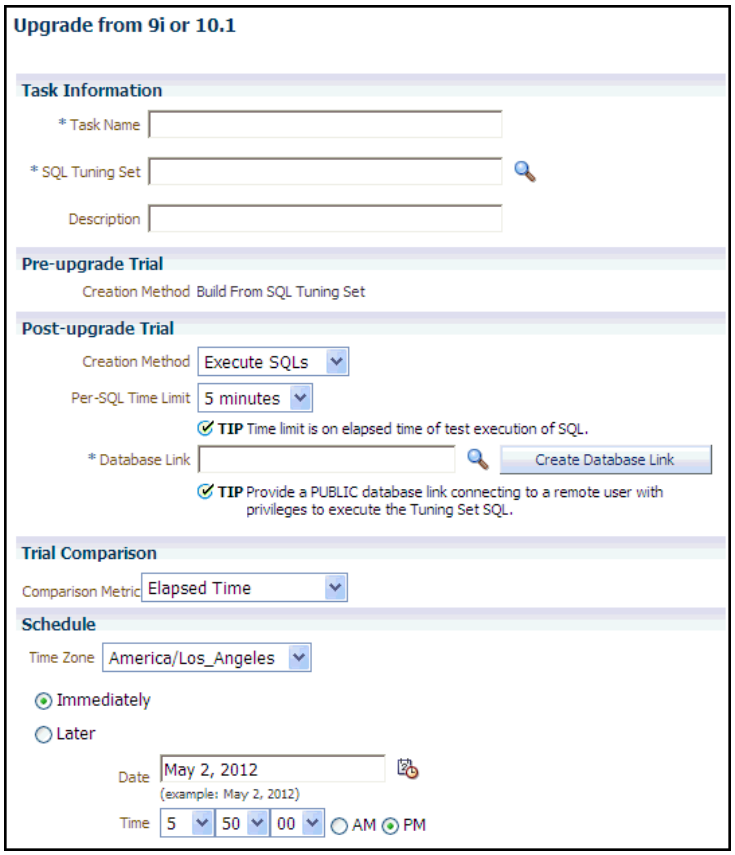
To test a database upgrade from Oracle9*i* Database or Oracle Database 10*g* Release 1 using SQL Performance Analyzer:

1. From the **Performance** menu, select **SQL**, then **SQL Performance Analyzer**.

   If the Database Login page appears, then log in as a user with administrator privileges.

   The SQL Performance Analyzer Home page appears.
2. Under SQL Performance Analyzer Workflows, click **Upgrade from 9i or 10.1**.

   The Upgrade from 9i or higher releases page appears.



3. Under Task Information:
   a. In the Task Name field, enter the name of the task.
   b. In the SQL Tuning Set field, enter the name of the SQL tuning set that was built.

      Alternatively, click the search icon to search for the SQL tuning set using the Search and Select: SQL Tuning Set window.

      The selected SQL tuning set now appears in the SQL Tuning Set field.
   c. In the Description field, optionally enter a description of the task.
4. In the Creation Method field, select:

- **Execute SQLs** to generate both execution plans and statistics for each SQL statement in the SQL tuning set by actually running the SQL statements remotely on the test system over a public database link.
- **Generate Plans** to create execution plans remotely on the test system over a public database link without actually running the SQL statements.

5. In the Per-SQL Time Limit list, determine the time limit for SQL execution during the trial by performing one of the following actions:
   - Select **5 minutes**.

     The execution will run each SQL statement in the SQL tuning set up to 5 minutes and gather performance data.
   - Select **Unlimited**.

     The execution will run each SQL statement in the SQL tuning set to completion and gather performance data. Collecting execution statistics provides greater accuracy in the performance analysis but takes a longer time. Using this setting is not recommended because the task may be stalled by one SQL statement for a prolonged time period.
   - Select **Customize** and enter the specified number of seconds, minutes, or hours.
6. In the Database Link field, enter the global name of a public database link connecting to a user with the `EXECUTE` privilege for the `DBMS_SQLPA` package and the `ADVISOR` privilege on the test system.

   Alternatively, click the search icon to search for and select a database link, or click **Create Database Link** to create a database link using the Create Database Link page.
7. In the Comparison Metric list, select the comparison metric to use for the comparison analysis:
   - **Elapsed Time**
   - **CPU Time**
   - **User I/O Time**
   - **Buffer Gets**
   - **Physical I/O**
   - **Optimizer Cost**
   - **I/O Interconnect Bytes**

   Optimizer Cost is the only comparison metric available if you generated execution plans only in the SQL trials.

   To perform the comparison analysis by using more than one comparison metric, perform separate comparison analyses by repeating this procedure with different metrics.
8. Under Schedule:
   a. In the Time Zone list, select your time zone code.
   b. 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.
9. Click **Submit**.

   The SQL Performance Analyzer Home page appears.

In the SQL Performance Analyzer Tasks section, the status of this task is displayed. To refresh the status icon, click **Refresh**. After the task completes, the Status field changes to Completed.

10. Under SQL Performance Analyzer Tasks, select the task and click the link in the Name column.

    The SQL Performance Analyzer Task page appears.

    This page contains the following sections:

    - SQL Tuning Set

      This section summarizes information about the SQL tuning set, including its name, owner, description, and the number of SQL statements it contains.

    - SQL Trials

      This section includes a table that lists the SQL trials used in the SQL Performance Analyzer task.

    - SQL Trial Comparisons

      This section contains a table that lists the results of the SQL trial comparisons

11. Click the icon in the Comparison Report column.

    The SQL Performance Analyzer Task Result page appears.

12. Review the results of the performance analysis, as described in "Reviewing the SQL Performance Analyzer Report Using Oracle Enterprise Manager".

    If regressed SQL statements are found following the database upgrade, tune them as described in "Tuning Regressed SQL Statements After Testing a Database Upgrade".

Testing Database Upgrades from Releases 9.x and 10.1 Using APIs

This section describes how to test database upgrades from Oracle Database releases 9.x and 10.1 using APIs.

To test a database upgrade from releases 9.x and 10.1:

1. On the system running SQL Performance Analyzer, create an analysis task.
2. Build the pre-upgrade SQL trial from the execution plans and run-time statistics in the SQL tuning set by calling the `EXECUTE_ANALYSIS_TASK` procedure using the following parameters:

   - Set the `task_name` parameter to the name of the SQL Performance Analyzer task that you want to execute.
   - Set the `execution_type` parameter to `CONVERT SQLSET` to direct SQL Performance Analyzer to treat the statistics in the SQL tuning set as a trial execution.
   - Specify a name to identify the execution using the `execution_name` parameter. If not specified, then SQL Performance Analyzer automatically generates a name for the task execution.

   The following example executes the SQL Performance Analyzer task named `my_spa_task` as a trial execution:

```
EXEC DBMS_SQLPA.EXECUTE_ANALYSIS_TASK(task_name => 'my_spa_task', -
       execution_type => 'CONVERT SQLSET', -
       execution_name => 'my_trial_9i');
```

3. Build the post-upgrade SQL trial by performing an explain plan or test execute using the `EXECUTE_ANALYSIS_TASK` procedure:
   - Set the `execution_type` parameter to `EXPLAIN PLAN` or `TEST EXECUTE`:
     - If you choose to use `EXPLAIN PLAN`, only execution plans will be generated. Subsequent comparisons will only be able to yield a list of changed plans without making any conclusions about performance changes.
     - If you choose to use `TEST EXECUTE`, the SQL workload will be executed to completion. This effectively builds the post-upgrade SQL trial using the statistics and execution plans generated from the test system. Using `TEST EXECUTE` is recommended to capture the SQL execution plans and performance data at the source, thereby resulting in a more accurate analysis.
   - Set the `DATABASE_LINK` task parameter to the global name of a public database link connecting to a user with the `EXECUTE` privilege for the `DBMS_SQLPA` package and the `ADVISOR` privilege on the test system.

   The following example performs a test execute of the SQL statements remotely over a database link:

```
EXEC DBMS_SQLPA.EXECUTE_ANALYSIS_TASK(task_name => 'my_spa_task', -
       execution_type => 'TEST EXECUTE', -
       execution_name => 'my_remote_trial_10g', -
       execution_params => dbms_advisor.arglist('database_link',
                                                'LINK.A.B.C.BIZ.COM'));
```

See Also:

- "Creating an Analysis Task Using APIs"
- "Creating a Pre-Change SQL Trial Using APIs"
- "Creating a Post-Change SQL Trial Using APIs"
- *Oracle Database PL/SQL Packages and Types Reference* to learn about the `DBMS_SQLPA.EXECUTE_ANALYSIS_TASK` function

Upgrading from Oracle Database 10*g* Release 2 and Newer Releases

You can use SQL Performance Analyzer to test the impact on SQL response time of a database upgrade from Oracle Database 10*g* Release 2 or a newer release to any later release by capturing a SQL tuning set on the production system, then executing it twice remotely over a database link on a test system—first to create a pre-change SQL trial, then again to create a post-change SQL trial.

Before the database upgrade can be tested, ensure that the following conditions are met:

- The production system which you are upgrading from is running Oracle Database 10*g* Release 2 or a newer release.
- Initially, the test system should also be running the same release of Oracle Database.
- The test system must contain an exact copy of the production data found on the production system.

- The hardware configuration must be as similar to the production system as possible.

You will also need to set up a separate SQL Performance Analyzer system running Oracle Database 11*g* Release 2. You will be using this system to run SQL Performance Analyzer. Neither your production data or schema need to be available on this system, since the SQL tuning set will be built using statistics stored in the SQL trace files from the production system. SQL Performance Analyzer tasks will be executed remotely on the test system to generate the execution plan and statistics for the SQL trial over a database link that you specify. The database link must be a public database link that connects to a user with the `EXECUTE` privilege for the `DBMS_SQLPA` package and the `ADVISOR` privilege on the test system. You should also drop any existing `PLAN_TABLE` from the user's schema on the test system.

Once the upgrade environment is configured as described, perform the steps as described in the following procedure to use SQL Performance Analyzer in a database upgrade from Oracle Database 10*g* Release 2 or a newer release to any later release.

1. On the production system, capture the SQL workload that you intend to analyze and store it in a SQL tuning set, as described in "Capturing the SQL Workload".
2. Set up the test system so that it matches the production environment as closely as possible, as described in "Setting Up the Test System".
3. Transport the SQL tuning set to the SQL Performance Analyzer system.

   For information about transporting SQL tuning sets using:

   - Oracle Enterprise Manager, see *Oracle Database 2 Day + Performance Tuning Guide*
   - APIs, see *Oracle Database SQL Tuning Guide*
4. On the SQL Performance Analyzer system, create a SQL Performance Analyzer task using the SQL tuning set as its input source.

   Remotely test execute the SQL statements in the SQL tuning set on the test system over a database link to build a pre-upgrade SQL trial that will be used as a baseline for comparison, as described in "Testing Database Upgrades from Oracle Database 10*g* Release 2 and Newer Releases".
5. Upgrade the test system.
6. Remotely test execute the SQL statements a second time on the upgraded test system over a database link to build a post-upgrade SQL trial, as described in "Testing Database Upgrades from Oracle Database 10*g* Release 2 and Newer Releases".
7. Compare SQL performance and fix regressed SQL.

   SQL Performance Analyzer compares the performance of SQL statements read from the SQL tuning set during the pre-upgrade SQL trial to those captured from the remote test execution during the post-upgrade SQL trial. A report is produced to identify any changes in execution plans or performance of the SQL statements.

   If the report reveals any regressed SQL statements, you can make further changes to fix the regressed SQL, as described in "Tuning Regressed SQL Statements After Testing a Database Upgrade".

   Repeat the process of executing the SQL tuning set and comparing its performance to a previous execution to test any changes made until you are satisfied with the outcome of the analysis.

Testing Database Upgrades from Oracle Database 10*g* Release 2 and Newer Releases

Once the SQL tuning set is transported to the SQL Performance Analyzer system, you can use SQL Performance Analyzer to build a pre-upgrade SQL trial by executing or generating plans of SQL statements in the SQL tuning set on the test system. SQL Performance Analyzer test executes the SQL statements using a database link that you specify by connecting to the test system remotely and generating the execution plans and statistics for the SQL trial. The database link should exist on the SQL Performance Analyzer system and connect to a remote user with privileges to execute the SQL tuning set on the test system.
After the pre-upgrade SQL trial is built, you need to upgrade the test system. Once the database has been upgraded, SQL Performance Analyzer will need to execute or generate plans of SQL statements in the SQL tuning set a second time on the upgraded test system to build a post-upgrade SQL trial. Alternatively, if hardware resources are available, you can use another upgraded test system to execute the second remote SQL trial. This method can be useful in helping you investigate issues identified by SQL Performance Analyzer.

You can run SQL Performance Analyzer to test a database upgrade from Oracle Database 10*g* Release 2 or a newer release using Oracle Enterprise Manager or APIs, as described in the following sections:

- Testing Database Upgrades from Releases 10.2 and Higher Using Cloud Control
- Testing Database Upgrades from Releases 10.2 and Higher Using APIs

Testing Database Upgrades from Releases 10.2 and Higher Using Cloud Control

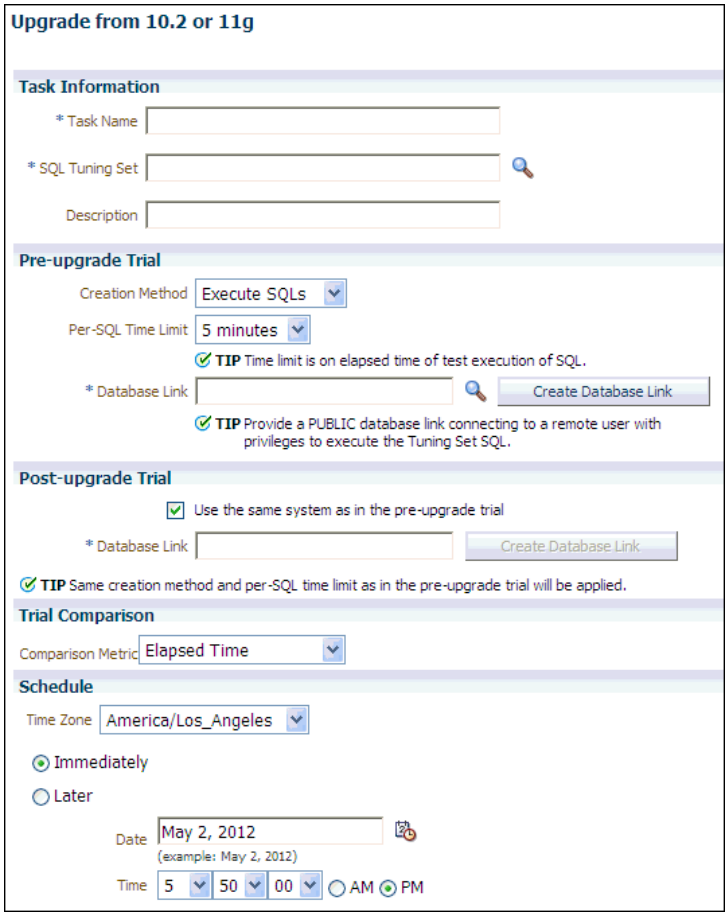
To test a database upgrade from Oracle Database 10*g* Release 2 or a newer release using SQL Performance Analyzer:

1. From the **Performance** menu, select **SQL**, then **SQL Performance Analyzer**.

   If the Database Login page appears, then log in as a user with administrator privileges.

   The SQL Performance Analyzer Home page appears.
2. Under SQL Performance Analyzer Workflows, click **Upgrade from 10.2 or 11g**.

   The Upgrade from 10.2 or higher releases page appears.



3. Under Task Information:
   a. In the Task Name field, enter the name of the task.
   b. In the SQL Tuning Set field, enter the name of the SQL tuning set that was built.

      Alternatively, click the search icon to search for the SQL tuning set using the Search and Select: SQL Tuning Set window.

      The selected SQL tuning set now appears in the SQL Tuning Set field.
   c. In the Description field, optionally enter a description of the task.
4. In the Creation Method field, select:
   - **Execute SQLs** to generate both execution plans and statistics for each SQL statement in the SQL tuning set by actually running the SQL statements remotely on the test system over a public database link.
   - **Generate Plans** to create execution plans remotely on the test system over a public database link without actually running the SQL statements.
5. In the Per-SQL Time Limit list, determine the time limit for SQL execution during the trial by performing one of the following actions:
   - Select **5 minutes**.

     The execution will run each SQL statement in the SQL tuning set up to 5 minutes and gather performance data.
   - Select **Unlimited**.

The execution will run each SQL statement in the SQL tuning set to completion and gather performance data. Collecting execution statistics provides greater accuracy in the performance analysis but takes a longer time. Using this setting is not recommended because the task may be stalled by one SQL statement for a prolonged time period.

- Select **Customize** and enter the specified number of seconds, minutes, or hours.

6. In the Database Link field, enter the global name of a public database link connecting to a user with the `EXECUTE` privilege for the `DBMS_SQLPA` package and the `ADVISOR` privilege on the pre-upgrade system.

   Alternatively, click the search icon to search for and select a database link, or click **Create Database Link** to create a database link using the Create Database Link page.

7. Under Post-upgrade Trial:

   a. Select **Use the same system as in the pre-upgrade trial** to use the same system for executing both the pre-upgrade and post-upgrade trials.

      Oracle recommends using this option to avoid possible errors due to different system configurations. When using this option, you will need to upgrade the test database to the higher database version before the post-upgrade trial is executed.

   b. In the Database Link field, enter the global name of a public database link connecting to a user with the `EXECUTE` privilege for the `DBMS_SQLPA` package and the `ADVISOR` privilege on the post-upgrade system.

8. In the Comparison Metric list, select the comparison metric to use for the comparison analysis:

   - **Elapsed Time**
   - **CPU Time**
   - **User I/O Time**
   - **Buffer Gets**
   - **Physical I/O**
   - **Optimizer Cost**
   - **I/O Interconnect Bytes**

   Optimizer Cost is the only comparison metric available if you generated execution plans only in the SQL trials.

   To perform the comparison analysis by using more than one comparison metric, perform separate comparison analyses by repeating this procedure with different metrics.

9. Under Schedule:

   a. In the Time Zone list, select your time zone code.

   b. 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.

10. Click **Submit**.

    The SQL Performance Analyzer Home page appears.

    In the SQL Performance Analyzer Tasks section, the status of this task is displayed. To refresh the status icon, click **Refresh**.

    If you are using the same system to execute both the pre-upgrade and post-upgrade trials, you will need to upgrade the database after the pre-upgrade trial step is completed. After

the database is upgraded, the post-upgrade trial can be executed. After the task completes, the Status field changes to Completed.

11. Under SQL Performance Analyzer Tasks, select the task and click the link in the Name column.

    The SQL Performance Analyzer Task page appears.

    This page contains the following sections:

    - SQL Tuning Set

      This section summarizes information about the SQL tuning set, including its name, owner, description, and the number of SQL statements it contains.

    - SQL Trials

      This section includes a table that lists the SQL trials used in the SQL Performance Analyzer task.

    - SQL Trial Comparisons

      This section contains a table that lists the results of the SQL trial comparisons

12. Click the icon in the Comparison Report column.

    The SQL Performance Analyzer Task Result page appears.

13. Review the results of the performance analysis, as described in "Reviewing the SQL Performance Analyzer Report Using Oracle Enterprise Manager".

    If regressed SQL statements are found following the database upgrade, tune them as described in "Tuning Regressed SQL Statements After Testing a Database Upgrade".

Testing Database Upgrades from Releases 10.2 and Higher Using APIs

This section describes how to test database upgrades from Oracle Database releases 10.2 and higher using APIs.

To test a database upgrade from releases 10.2 and higher:

1. On the system running SQL Performance Analyzer, create an analysis task.
2. Build the pre-upgrade SQL trial by performing an explain plan or test execute of SQL statements in the SQL tuning set.

   Call the `EXECUTE_ANALYSIS_TASK` procedure using the following parameters:

   - Set the `task_name` parameter to the name of the SQL Performance Analyzer task that you want to execute.
   - Set the `execution_type` parameter to `EXPLAIN PLAN` or `TEST EXECUTE`:
     - If you choose to use `EXPLAIN PLAN`, only execution plans will be generated. Subsequent comparisons will only be able to yield a list of changed plans without making any conclusions about performance changes.
     - If you choose to use `TEST EXECUTE`, the SQL workload will be executed to completion. This effectively builds the pre-upgrade SQL trial using the statistics and execution plans generated from the test system. Using `TEST EXECUTE` is recommended to capture the SQL execution plans and performance data at the source, thereby resulting in a more accurate analysis.
   - Specify a name to identify the execution using the `execution_name` parameter. If not specified, then SQL Performance Analyzer automatically generates a name for the task execution.

- Set the `DATABASE_LINK` task parameter to the global name of a public database link connecting to a user with the `EXECUTE` privilege for the `DBMS_SQLPA` package and the `ADVISOR` privilege on the test system.

The following example executes the SQL Performance Analyzer task named `my_spa_task` and performs a test execute of the SQL statements remotely over a database link:

```
EXEC DBMS_SQLPA.EXECUTE_ANALYSIS_TASK(task_name => 'my_spa_task', -
       execution_type => 'TEST EXECUTE', -
       execution_name => 'my_remote_trial_10g', -
       execution_params => dbms_advisor.arglist('database_link',
                                                'LINK.A.B.C.BIZ.COM'));
```

3. Build the post-upgrade SQL trial by performing an explain plan or test execute using the `EXECUTE_ANALYSIS_TASK` procedure:

- Set the `execution_type` parameter to `EXPLAIN PLAN` or `TEST EXECUTE`:
  - If you choose to use `EXPLAIN PLAN`, only execution plans will be generated. Subsequent comparisons will only be able to yield a list of changed plans without making any conclusions about performance changes.
  - If you choose to use `TEST EXECUTE`, the SQL workload will be executed to completion. This effectively builds the post-upgrade SQL trial using the statistics and execution plans generated from the test system. Using `TEST EXECUTE` is recommended to capture the SQL execution plans and performance data at the source, thereby resulting in a more accurate analysis.
- Set the `DATABASE_LINK` task parameter to the global name of a public database link connecting to a user with the `EXECUTE` privilege for the `DBMS_SQLPA` package and the `ADVISOR` privilege on the test system.

The following example performs a test execute of the SQL statements remotely over a database link:

```
EXEC DBMS_SQLPA.EXECUTE_ANALYSIS_TASK(task_name => 'my_spa_task', -
       execution_type => 'TEST EXECUTE', -
       execution_name => 'my_remote_trial_12c', -
       execution_params => dbms_advisor.arglist('database_link',
                                                'LINK.A.B.C.BIZ.COM'));
```

See Also:

- "Creating an Analysis Task Using APIs"
- "Creating a Pre-Change SQL Trial Using APIs"
- "Creating a Post-Change SQL Trial Using APIs"
- *Oracle Database PL/SQL Packages and Types Reference* to learn about the `DBMS_SQLPA.EXECUTE_ANALYSIS_TASK` function

Tuning Regressed SQL Statements After Testing a Database Upgrade

In some cases, SQL Performance Analyzer may identify SQL statements whose performance regressed after you upgrade the database on the test system.

You can tune the regressed SQL statements by using the SQL Tuning Advisor or SQL plan baselines, as described in Comparing SQL Trials . This involves using APIs to build a subset of a SQL tuning set with only the regressed SQL statements, transport this subset of regressed SQL statements to the remote database, and running the SQL Tuning Advisor on the remote database.

Oracle Enterprise Manager does not provide support for fixing regressions after running SQL Performance Analyzer involving one or more remote SQL trials.

If you are upgrading from Oracle Database 10*g* Release 2 and newer releases, you can also create SQL plan baselines to instruct the optimizer to select existing execution plans in the future.

See Also:

- "Tuning Regressed SQL Statements From a Remote SQL Trial Using APIs"
- "Creating SQL Plan Baselines Using APIs"

Part II
Database Replay

Database Replay enables you to replay a full production workload on a test system to assess the overall impact of system changes.

Part II covers Database Replay and contains the following chapters:

9

Introduction to Database Replay

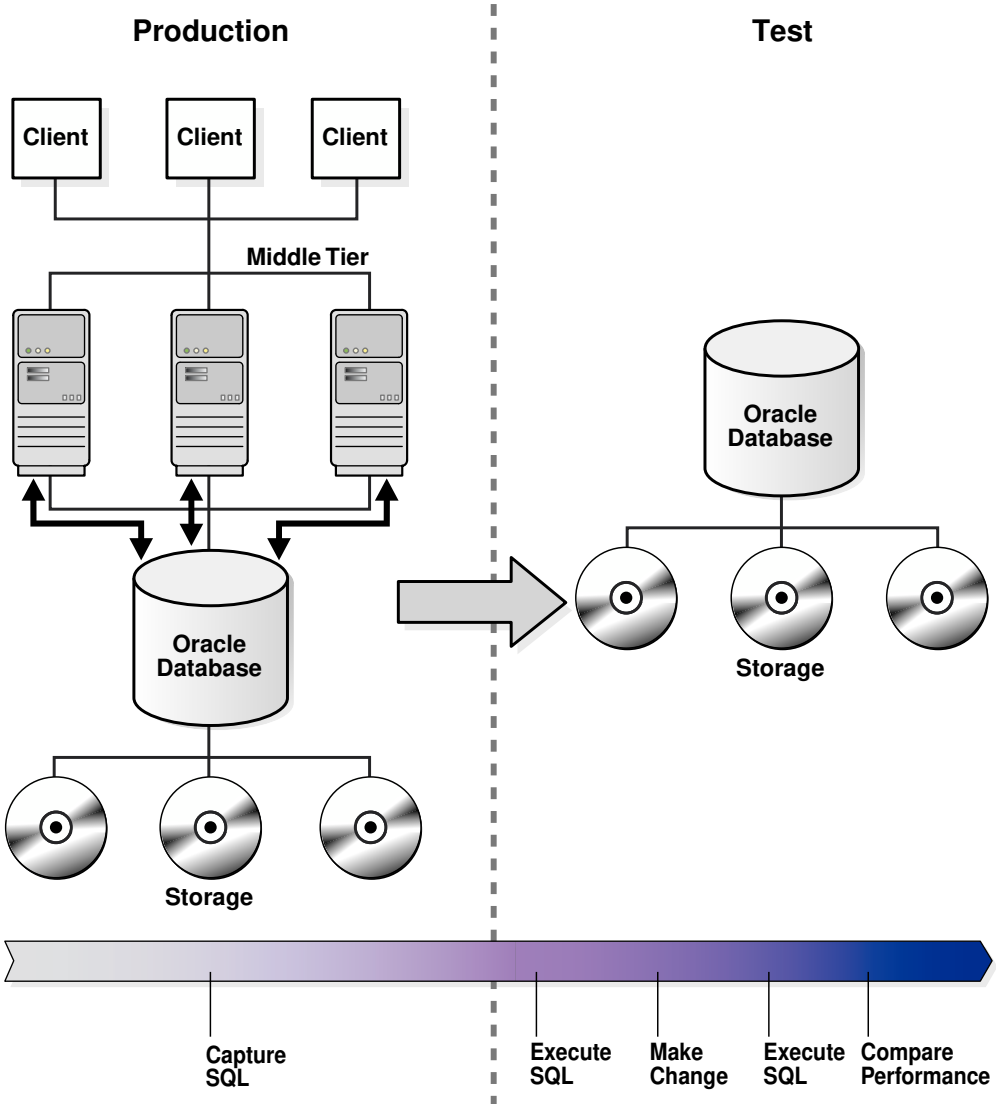
You can use Database Replay to capture a workload on the production system and replay it on a test system with the exact timing, concurrency, and transaction characteristics of the original workload. This enables you to test the effects of a system change without affecting the production system.

Database Replay supports workload capture on a system running Oracle Database 10*g* Release 2 and newer releases. In order to capture a workload on a system running Oracle Database 10*g* Release 2, the database version must be 10.2.0.4 or higher. Workload replay is only supported on systems running Oracle Database 11*g* Release 1 and newer releases.

> **Note:**
>
> To use the workload capture feature on a system running Oracle9*i* Database or an earlier version of Oracle Database 10*g*, refer to My Oracle Support note ID 560977.1 at the URL below for information about the required patches, or contact Oracle Support for more information:
>
> https://support.oracle.com/rs?type=doc&id=560977.1

Analyzing the effect of system changes using Database Replay involves the following steps, as illustrated in Figure 9-1:

Figure 9-1 Database Replay Workflow



1. On the production system, capture the workload into capture files, as described in "Workload Capture".
2. Copy the capture files to the test system and preprocess them, as described in "Workload Preprocessing".
3. On the test system, replay the preprocessed files, as described in "Workload Replay".
4. Using the reports generated by Database Replay, perform detailed analysis of both the workload capture and workload replay, as described in "Analysis and Reporting".

Workload Capture

The first step in using Database Replay is to capture the production workload. Capturing a workload involves recording all requests made by external clients to Oracle Database.

When workload capture is enabled, all external client requests directed to Oracle Database are tracked and stored in binary files—called capture files—on the file system. You can specify the

location where the capture files will be stored. Once workload capture begins, all external database calls are written to the capture files. The capture files contain all relevant information about the client request, such as SQL text, bind values, and transaction information. Background activities and database scheduler jobs are not captured. These capture files are platform independent and can be transported to another system.

> **See Also:**
>
> - Capturing a Database Workload for information about how to capture a workload on the production system

Workload Preprocessing

Once the workload has been captured, the information in the capture files must be preprocessed. Preprocessing creates all necessary metadata needed for replaying the workload. This must be done once for every captured workload before they can be replayed. After the captured workload is preprocessed, it can be replayed repeatedly on a replay system running the same version of Oracle Database. Typically, the capture files should be copied to a test system for preprocessing. As workload preprocessing can be time consuming and resource intensive, it is recommended that this step be performed on the test system where the workload will be replayed.

> **See Also:**
>
> - Preprocessing a Database Workload for information about how to preprocess a captured workload

Workload Replay

After a captured workload has been preprocessed, it can be replayed on a test system. During the workload replay phase, Oracle Database performs the actions recorded during the workload capture phase on the test system by re-creating all captured external client requests with the same timing, concurrency, and transaction dependencies of the production system. Database Replay uses a client program called the replay client to re-create all external client requests recorded during workload capture. Depending on the captured workload, you may need one or more replay clients to properly replay the workload. A calibration tool is provided to help determine the number of replay clients needed for a particular workload. Because the entire workload is replayed—including DML and SQL queries—the data in the replay system should be as logically similar to the data in the capture system as possible. This will minimize replay divergence and enable a more reliable analysis of the replay.

> **See Also:**
>
> - Replaying a Database Workload for information about how to replay a preprocessed workload on the test system

Analysis and Reporting

Once the workload is replayed, in-depth reporting is provided for you to perform detailed analysis of both workload capture and replay.

The workload capture report and workload replay report provide basic information about the workload capture and replay, such as errors encountered during replay and data divergence in rows returned by DML or SQL queries. A comparison of several statistics—such as database time, average active sessions, and user calls—between the workload capture and the workload replay is also provided.

The replay compare period report can be used to perform a high-level comparison of one workload replay to its capture or to another replay of the same capture. A divergence summary with an analysis of whether any data divergence occurred and if there were any significant performance changes is also provided. Furthermore, Automatic Database Diagnostic Monitor (ADDM) findings are incorporated into these reports.

For advanced analysis, Automatic Workload Repository (AWR) reports are available to enable detailed comparison of performance statistics between the workload capture and the workload replay. The information available in these reports is very detailed, and some differences between the workload capture and replay can be expected. Furthermore, Workload Intelligence operates on data recorded during a workload capture to create a model that describes the workload. This model can be used to identify significant patterns in templates that are executed as part of the workload. For each pattern, you can view important statistics, such as the number of executions of a given pattern and the database time consumed by the pattern during its execution.

The SQL Performance Analyzer report can be used to compare a SQL tuning set from a workload capture to another SQL tuning set from a workload replay, or two SQL tuning sets from two workload replays. Comparing SQL tuning sets with Database Replay provides more information than SQL Performance Analyzer test-execute because it considers and shows all execution plans for each SQL statement, while SQL Performance Analyzer test-execute generates only one execution plan per SQL statement for each SQL trial. Moreover, the SQL statements are executed in a more authentic environment because Database Replay captures all bind values and reproduces dynamic session state such as PL/SQL package state more accurately. It is recommended that you run SQL Performance Analyzer test-execute first as a sanity test to ensure SQL statements have not regressed and the test system is set up properly before using Database Replay to perform load and currency testing.

Besides using replay divergence information to analyze replay characteristics of a given system change, you should also use an application-level validation procedure to assess the system change. Consider developing a script to assess the overall success of the replay. For example, if 10,000 orders are processed during workload capture, you should validate that a similar number of orders are also processed during replay.

After the replay analysis is complete, you can restore the database to its original state at the time of workload capture and repeat workload replay to test other changes to the system.

See Also:

- Analyzing Captured and Replayed Workloads for information about how to analyze data and performance divergence using Database Replay reports

Workload Capture and Replay in a PDB

A workload capture can be enabled and a workload replay can be started at the pluggable database (PDB) level.

Before Oracle Database Release 19c, workload capture and replay were strictly for container database (CDB) administrators where capture and replay were started from CDB root. Starting with Oracle Database Release 19c, a workload capture can be enabled for the current pluggable database (PDB). A workload replay can also be started for the current PDB.

> **Note:**
>
> Concurrent workload captures and replays are supported at the PDB level.

10

Capturing a Database Workload

This chapter describes how to capture a database workload on the production system. The first step in using Database Replay is to capture the production workload.
This chapter contains the following sections:

> **See Also:**
>
> "Workload Capture" for more information about how capturing a database workload fits within the Database Replay architecture

Prerequisites for Capturing a Database Workload

Before starting a workload capture, you should have a strategy in place to restore the database on the test system. Before a workload can be replayed, the logical state of the application data on the replay system should be similar to that of the capture system when replay begins. To accomplish this, consider using one of the following methods:

- Recovery Manager (RMAN) `DUPLICATE` command
- Snapshot standby
- Data Pump Import and Export

This will allow you to restore the database on the replay system to the application state as of the workload capture start time.

If the database is protected by Database Vault, then you need to be authorized to use the `DBMS_WORKLOAD_CAPTURE` and `DBMS_WORKLOAD_REPLAY` packages in a Database Vault environment before you can use Database Replay.

See Also:

- *Oracle Database Vault Administrator's Guide* for information about using Database Replay in a Database Vault environment
- *Oracle Database Backup and Recovery User's Guide* for information about duplicating databases with RMAN
- *Oracle Data Guard Concepts and Administration* for information about managing snapshot standby databases
- *Oracle Database Utilities* for information about using Data Pump

Setting Up the Capture Directory

Determine the location and set up a directory where the captured workload will be stored. Before starting the workload capture, ensure that the directory is empty and has ample disk space to store the workload. If the directory runs out of disk space during a workload capture, the capture will stop. To estimate the amount of disk space that is required, you can run a test capture on your workload for a short duration (such as a few minutes) to extrapolate how much space you will need for a full capture. To avoid potential performance issues, you should also ensure that the target replay directory is mounted on a separate file system.
For Oracle RAC, consider using a shared file system. Alternatively, you can set up one capture directory path that resolves to separate physical directories on each instance, but you will need to consolidate the files created in each of these directories into a single directory. For captures on an Oracle RAC database, Enterprise Manager only supports Oracle RAC configured with a shared file system. The entire content of the local capture directories on each instance (not only the capture files) must be copied to the shared directory before it can be used for preprocessing. For example, assume that you are:

- Running an Oracle RAC environment in Linux with two database instances named `host1` and `host2`
- Using a capture directory object named `CAPDIR` that resolves to `/$ORACLE_HOME/rdbms/capture` on both instances
- Using a shared directory that resides in `/nfs/rac_capture`

You will need to login into each host and run the following command:

```
cp -r /$ORACLE_HOME/rdbms/capture/* /nfs/rac_capture
```

After this is done for both instances, the `/nfs/rac_capture` shared directory is ready to be preprocessed or masked.

Workload Capture Options

Proper planning before workload capture is required to ensure that the capture will be accurate and useful when replayed in another environment.

Before capturing a database workload, carefully consider the following options:

Restarting the Database

While this step is not required, Oracle recommends that the database be restarted before capturing the workload to ensure that ongoing and dependent transactions are allowed to be completed or rolled back before the capture begins. If the database is not restarted before the capture begins, transactions that are in progress or have yet to be committed will not be fully captured in the workload. Ongoing transactions will thus not be replayed properly, because only the part of the transaction whose calls were captured will be replayed. This may result in undesired replay divergence when the workload is replayed. Any subsequent transactions with dependencies on the incomplete transactions may also generate errors during replay. On a busy system, it is normal to see some replay divergence, but the replay can still be used to perform meaningful analysis of a system change if the diverged calls do not make up a significant portion of the replay in terms of DB time and other such key attributes.
Before restarting the database, determine an appropriate time to shut down the production database before the workload capture when it is the least disruptive. For example, you may want to capture a workload that begins at 8:00 a.m. However, to avoid service interruption during normal business hours, you may not want to restart the database during this time. In this case, you should consider starting the workload capture at an earlier time, so that the database can be restarted at a time that is less disruptive.

Once the database is restarted, it is important to start the workload capture before any user sessions reconnect and start issuing any workload. Otherwise, transactions performed by these user sessions will not be replayed properly in subsequent database replays, because only the part of the transaction whose calls were executed after the workload capture is started will be replayed. To avoid this problem, consider restarting the database in `RESTRICTED` mode using `STARTUP RESTRICT`, which will only allow the `SYS` user to login and start the workload capture. By default, once the workload capture begins, any database instance that are in `RESTRICTED` mode will automatically switch to `UNRESTRICTED` mode, and normal operations can continue while the workload is being captured.

Only one workload capture can be performed at any given time. If you have a Oracle Real Application Clusters (Oracle RAC) configuration, workload capture is performed for the entire database. Once you enable capture for one of the Oracle RAC nodes, workload capture is started on all database instances (the workload capture process is Oracle RAC aware). Although it is not required, restarting all instances in a Oracle RAC configuration before workload capture is recommended to avoid capturing ongoing transactions.

To restart all instances in a Oracle RAC configuration before workload capture:

1. Shut down all the instances.
2. Restart all the instances.
3. Start workload capture.
4. Connect the application and start the user workload.

> **See Also:**
>
> - *Oracle Database Administrator's Guide* for information about restricting access to an instance at startup

Using Filters with Workload Capture

By default, all user sessions are recorded during workload capture. You can use workload filters to specify which user sessions to include in or exclude from the workload during workload capture. There are two types of workload filters: inclusion filters and exclusion filters. You can use either inclusion filters or exclusion filters in a workload capture, but not both. Inclusion filters enable you to specify user sessions that will be captured in the workload. This is useful if you want to capture only a subset of the database workload.

Exclusion filters enable you to specify user sessions that will not be captured in the workload. This is useful if you want to filter out session types that do not need to captured in the workload, such as those that monitor the infrastructure—like Oracle Enterprise Manager (EM) or Statspack—or other such processes that are already running on the test system. For example, if the system where the workload will be replayed is running EM, replaying captured EM sessions on the system will result in duplication of workload. In this case, you may want to use exclusion filters to filter out EM sessions.

Workload Capture Restrictions

Certain types of user sessions and client requests may sometimes be captured in a workload, but they are not supported by Database Replay. Capturing these session and request types in a workload may result in errors during workload replay.

The following types of user sessions and client requests are not supported by Database Replay:

- Direct path load of data from external files using utilities such as SQL*Loader
- Non-PL/SQL based Advanced Queuing (AQ)
- Flashback queries
- Oracle Call Interface (OCI) based object navigations
- Non SQL-based object access
- Distributed transactions

  Any distributed transactions that are captured will be replayed as local transactions.
- XA transactions

  XA transactions are not captured or replayed. All local transactions are captured.
- `JAVA_XA` transactions

  If the workload uses the `JAVA_XA` package, `JAVA_XA` function and procedure calls are captured as normal PL/SQL workload. To avoid problems during workload replay, consider dropping the `JAVA_XA` package on the replay system to enable the replay to complete successfully.
- Database Resident Connection Pooling (DRCP)
- Workloads using OUT binds
- Multi-threaded Server (MTS) and shared server sessions with synchronization mode set to `OBJECT_ID`
- Migrated sessions

The workload is captured for migrated sessions. However, user logins or session migration operations are not captured. Without a valid user login or session migration, the replay may cause errors because the workload may be replayed by a wrong user.

Typically, Database Replay refrains from capturing these types of non-supported user sessions and client requests. Even when they are captured, Database Replay will not replay them. Therefore, it is usually not necessary to manually filter out non-supported user sessions and client requests. In cases where they are captured and found to cause errors during replay, consider using workload capture filters to exclude them from the workload.

See Also:

- "Using Filters with Workload Capture" for information about using workload capture filters
- "Using Filters with Workload Replay" for information about using workload replay filters

Enabling and Disabling the Workload Capture Feature

Database Replay supports capturing a database workload on a system running Oracle Database 10*g* Release 2 that can be used to test database upgrades to Oracle Database 11*g* and subsequent releases. By default, the workload capture feature is not enabled in Oracle Database 10*g* Release 2 (10.2). You can enable or disable this feature by specifying the `PRE_11G_ENABLE_CAPTURE` initialization parameter.

Note:

It is only necessary to enable the workload capture feature if you are capturing a database workload on a system running Oracle Database 10*g* Release 2.

If you are capturing a database workload on a system running Oracle Database 11*g* Release 1 or a later release, it is not necessary to enable the workload capture feature because it is enabled by default. Furthermore, the `PRE_11G_ENABLE_CAPTURE` initialization parameter is only valid with Oracle Database 10*g* Release 2 (10.2) and cannot be used with subsequent releases.

To enable the workload capture feature on a system running Oracle Database 10*g* Release 2, run the `wrrenbl.sql` script at the SQL prompt:

```
@$ORACLE_HOME/rdbms/admin/wrrenbl.sql
```

The `wrrenbl.sql` script calls the `ALTER SYSTEM` SQL statement to set the `PRE_11G_ENABLE_CAPTURE` initialization parameter to `TRUE`. If a server parameter file (spfile) is being used, the `PRE_11G_ENABLE_CAPTURE` initialization parameter will be modified for the currently running instance and recorded in the spfile, so that the new setting will persist when the database is restarted. If a spfile is not being used, the `PRE_11G_ENABLE_CAPTURE` initialization parameter will only be modified for the currently running instance, and the new setting will not persist when the database is restarted. To make the setting persistent without using a spfile, you will need to manually specify the parameter in the initialization parameter file (`init.ora`).

To disable workload capture, run the `wrrdsbl.sql` script at the SQL prompt:

```
@$ORACLE_HOME/rdbms/admin/wrrdsbl.sql
```

The `wrrdsbl.sql` script calls the `ALTER SYSTEM` SQL statement to set the `PRE_11G_ENABLE_CAPTURE` initialization parameter to `FALSE`. If a server parameter file (spfile) is being used, the `PRE_11G_ENABLE_CAPTURE` initialization parameter will be modified for the currently running instance and also recorded in the spfile, so that the new setting will persist when the database is restarted. If a spfile is not being used, the `PRE_11G_ENABLE_CAPTURE` initialization parameter will only be modified for the currently running instance, and the new setting will not persist when the database is restarted. To make the setting persistent without using a spfile, you will need to manually specify the parameter in the initialization parameter file (`init.ora`).

> **Note:**
>
> The `PRE_11G_ENABLE_CAPTURE` initialization parameter can only be used with Oracle Database 10*g* Release 2 (10.2). This parameter is not valid in subsequent releases. After upgrading the database, you will need to remove the parameter from the server parameter file (spfile) or the initialization parameter file (`init.ora`); otherwise, the database will fail to start up.

Enterprise Manager Privileges and Roles

The Database Replay resource type privileges enable you to view or operate any Database Replay entities. Additionally, you need the target operator privilege for the target from which the workload was captured to access the entities associated with the workload. For a target that does not exist anymore, the Enterprise Manager user who owns the entities or the Enterprise Manager super user can still access the entities.
The two security roles discussed in the following sections make it easier to grant or revoke privileges related to Database Replay entities.

Database Replay Viewer Role

Users who have the Database Replay Viewer role can view any Database Replay entity. By default, no Enterprise Manager user is granted this role. However, the EM_ALL_VIEWER role includes this role by default.
The Database Replay Viewer role consists of the Database Replay Viewer (resource type) privilege.

Database Replay Operator Role

The Database Replay Operator role includes the Database Replay Viewer role and thus its privileges. Users who have the Database Replay Operator role can also edit and delete any Database Replay entity. By default, no Enterprise Manager user is granted this role. However, the EM_ALL_OPERATOR role includes this role by default.
The Database Replay Operator role consists of the following privileges:

- Database Replay Operator (resource type privilege)
- Create new Named Credential (resource type privilege)
- Create new job (resource type privilege)

- Connect to any viewable target (target type privilege)
- Execute Command Anywhere (target type privilege)

To capture or replay a workload on a database target, an Enterprise Manager user needs all the privileges granted by the Database Replay Operator role plus the target operator privilege for the database target.

Capturing a Database Workload Using Enterprise Manager

This section describes how to capture a database workload using Enterprise Manager. The primary tool for capturing database workloads is Oracle Enterprise Manager.

For information about the prerequisites, see "Prerequisites for Capturing a Database Workload".

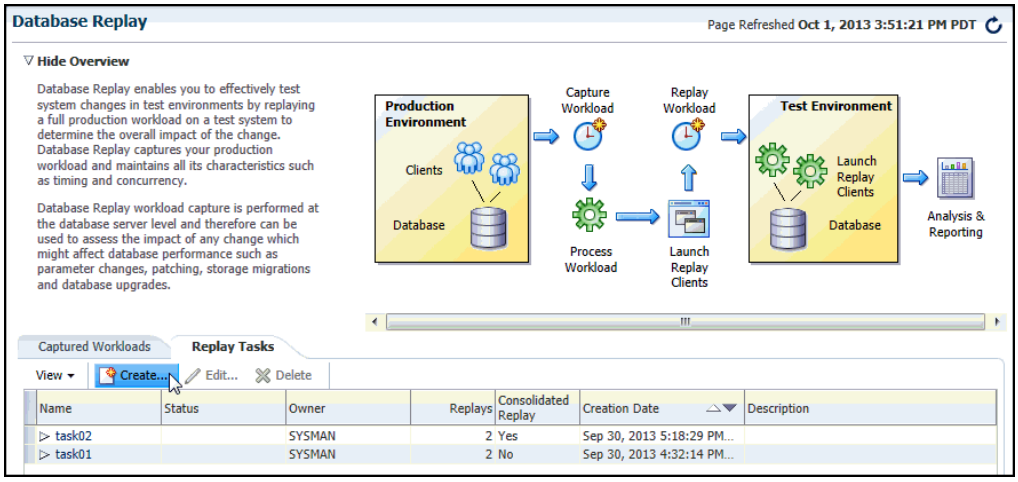
> **Tip:**
>
> If Oracle Enterprise Manager is unavailable, you can capture database workloads using APIs, as described in "Capturing a Database Workload Using APIs".

To capture a database workload using Enterprise Manager:

1. From the Enterprise menu of the Enterprise Manager Cloud Control console, select **Quality Management**, then **Database Replay**.

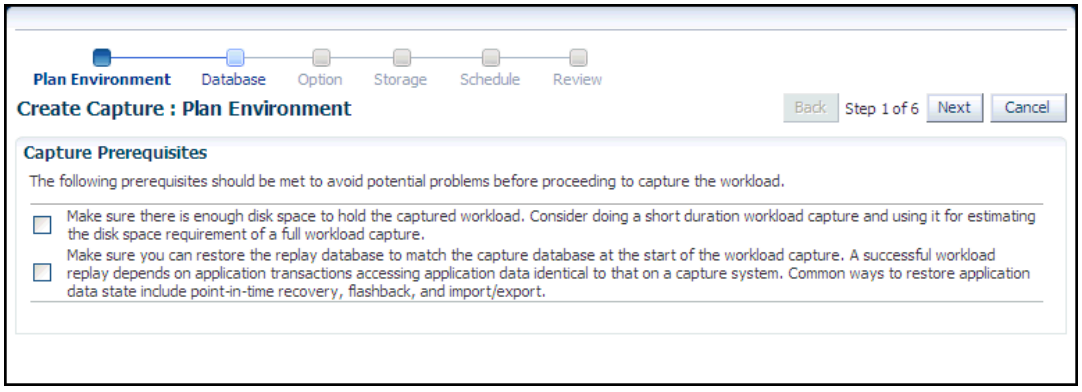
   If the Database Login page appears, log in as a user with administrator privileges.

   The Database Replay page appears.



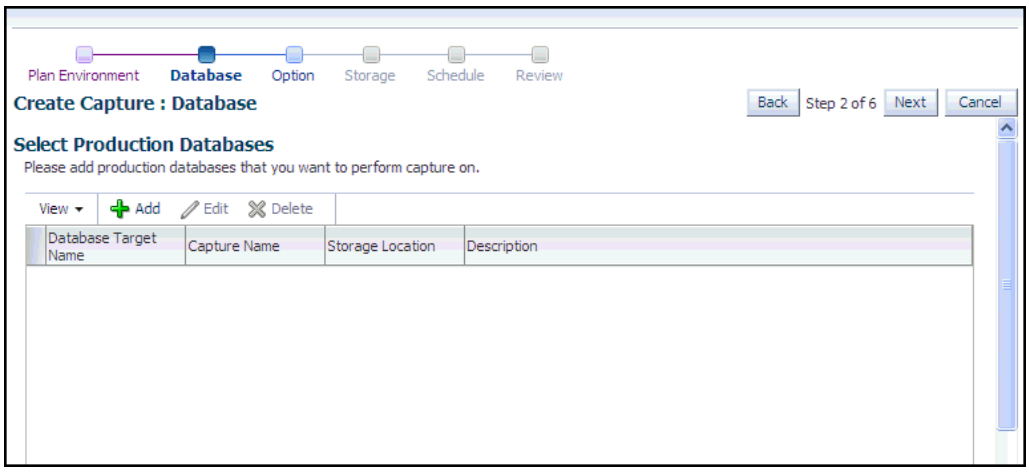
2. From the Database Replay page, click the **Captured Workloads** tab, then click **Create** in the toolbar.

   The Create Capture: Plan Environment page appears.



3. Verify that you have met both prerequisites described on this page, then enable both checkboxes and click **Next**.

   The Create Capture: Database page appears.

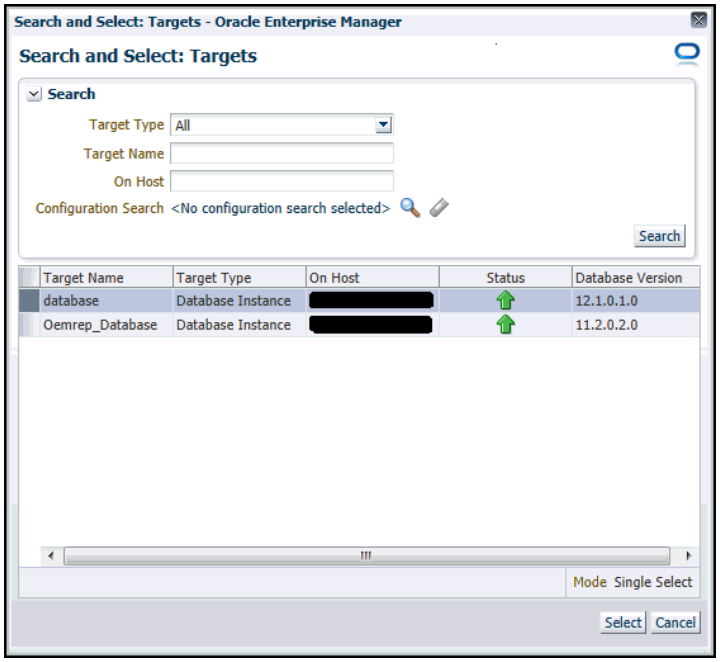


4. Click **Add**.

   The Add pop-up appears.

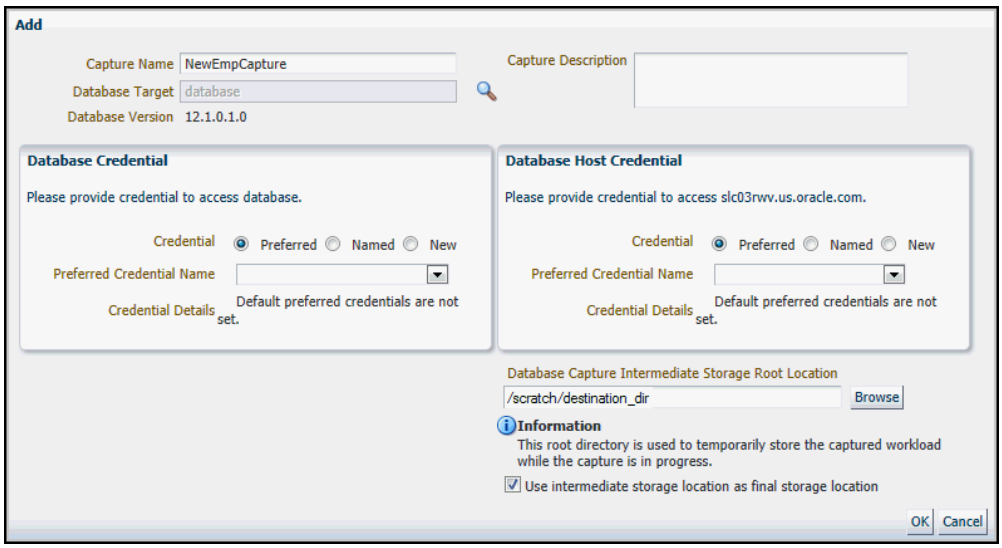
5. Provide a capture name, provide an optional description, then click the Target Database search icon.

   The Search and Select: Targets pop-up appears.



6. Select a Target Type, optionally provide a configuration search, choose a target database from the list, then click **Select**.

   The Add pop-up reappears with added sections for Database Credential and Database Host Credential.



7. Provide database credentials, database host credentials, a Database Capture Intermediate Storage Location, then click **OK**.

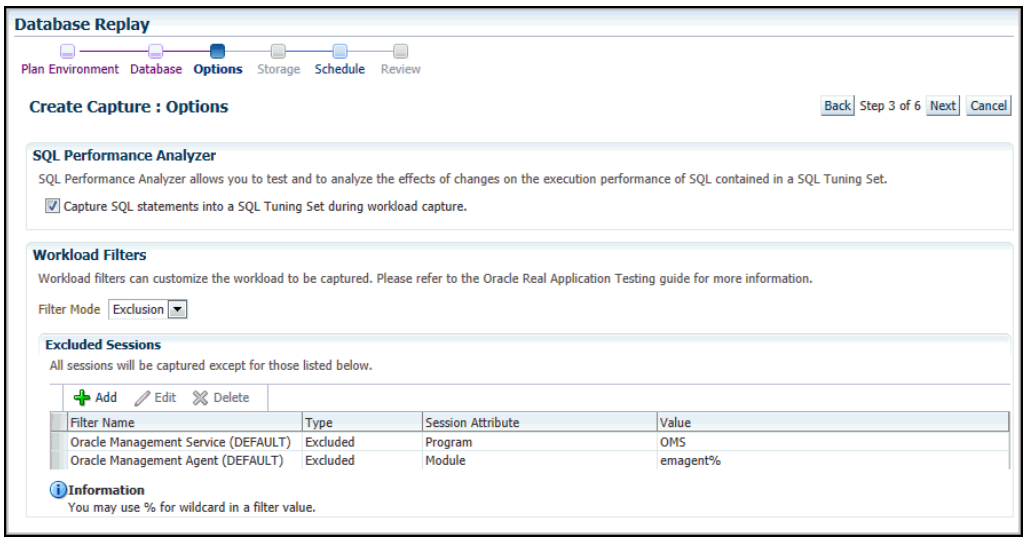
   - After the Capture, the files are copied to the storage location, unless you use the intermediate storage location as the final storage location.

> **Note:**
>
> For captures on an Oracle RAC database, Enterprise Manager only supports Oracle RAC configured with a shared file system.

The selected target database now appears in the list of databases in the Select Production Databases table.

8. Click **Next**.

   The **Create Capture:** Options page appears.



9. Select the workload capture options:

   - Under the SQL Performance Analyzer section, select whether to capture SQL statements into a SQL tuning set during workload capture.

     While Database Replay provides an analysis of how a change affects your entire system, you can use a SQL tuning set in conjunction with the SQL Performance Analyzer to gain a more SQL-centric analysis of how the change affects SQL statements and execution plans.

     By capturing a SQL tuning set during workload capture and another SQL tuning set during workload replay, you can use the SQL Performance Analyzer to compare these SQL tuning sets to each other without having to re-execute the SQL statements. This enables you to obtain a SQL Performance Analyzer report and compare the SQL performance, before and after changes, while running Database Replay.

     > **Note:**
     >
     > Capturing SQL statements into a SQL Tuning Set is the default, and is the recommended workload capture option. Capturing SQL statements into a SQL Tuning Set is not available for Oracle RAC.

     > **Tip:**
     >
     > For information about comparing SQL tuning sets using SQL Performance Analyzer reports, see "Generating SQL Performance Analyzer Reports Using APIs".

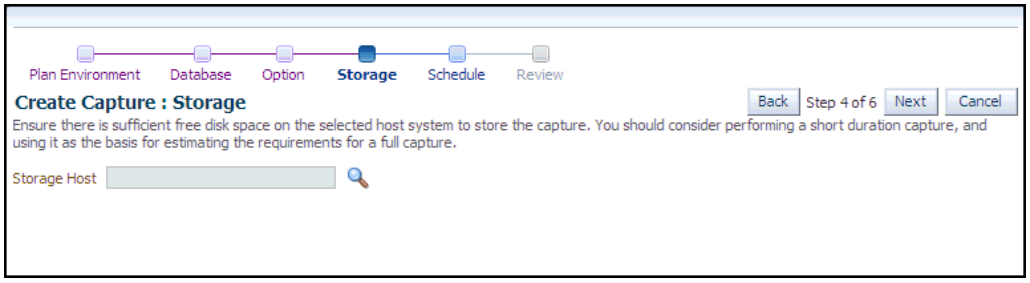
- Under the Workload Filters section, select whether to use exclusion filters by selecting **Exclusion** in the Filter Mode list, or inclusion filters by selecting **Inclusion** in the Filter Mode list.

  To add filters, click **Add** and enter the filter name, session attribute, and value in the corresponding fields.

> **Tip:**
>
> For more information, see "Using Filters with Workload Capture".

After selecting the desired workload capture options, click **Next**.

The Create Capture: Storage page appears.

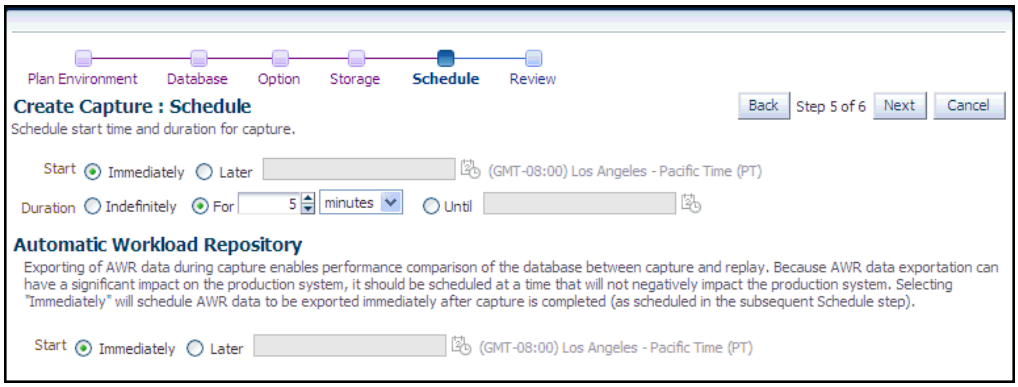


10. Click the Storage Host icon, choose a target from the list, then click **Select**.

    The Storage page now requests Host Credentials and a Storage Location.

11. Provide Host Credentials, click **Browse** to select a Storage Location, select the location and click **OK**, then click **Next**.

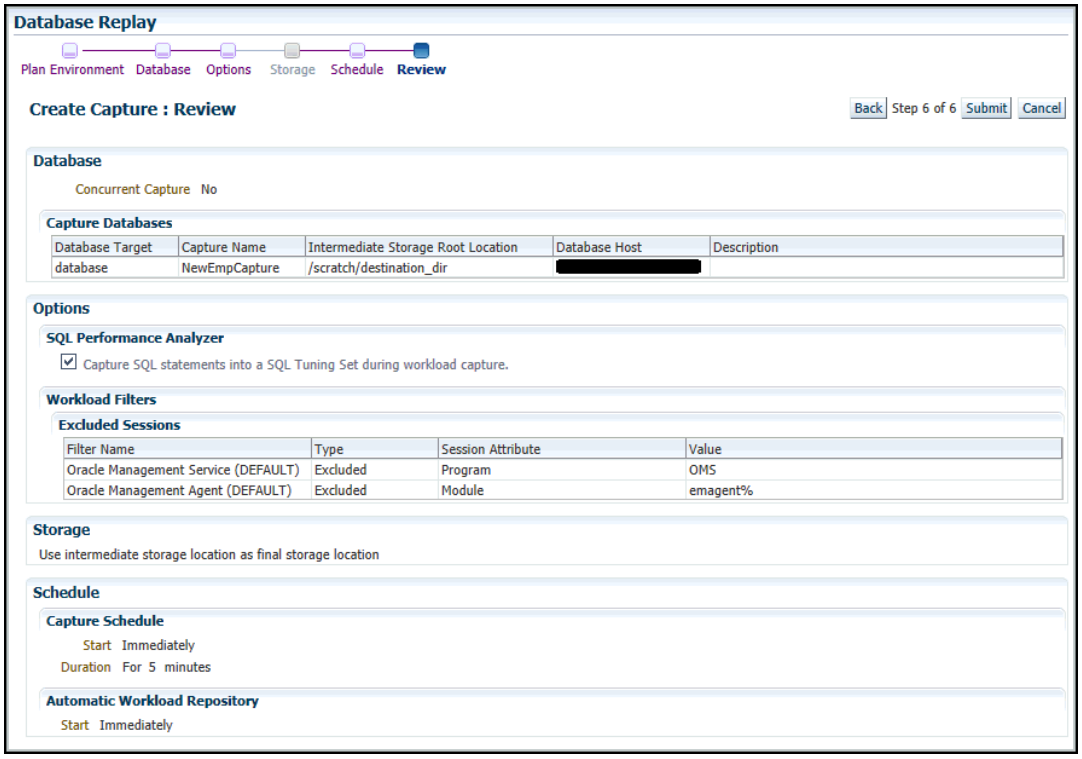
    The Create Capture: Schedule page appears.



12. Schedule the starting time and duration for the capture, schedule the exporting of AWR data, then click **Next**.

    - The default capture duration is 5 minutes. Change the capture duration to capture representative activity over a time period of interest that needs to be tested.

    The Create Capture: Review page appears.



13. If all of the parameters appear as you have intended, click **Submit** to start the capture job.
    - The "Capture SQL statements into a SQL Tuning Set during workload capture" option is enabled by default. Uncheck this option if you do not want to compare SQL tuning sets at the end of the Replay.

    The Database Replay page reappears, displays a message that the capture was created successfully, and displays the status of the capture in the Captures list, such as "Scheduled."

14. For detailed information about the capture, double-click the name of the capture.

    The Capture Summary page appears, and displays several attributes, including the average active sessions, a workload comparison, and related concurrent captures, if any.

> **Tip:**
>
> After capturing a workload on the production system, you need to preprocess the captured workload, as described in Preprocessing a Database Workload .

Capturing Workloads from Multiple Databases Concurrently

Concurrent capture refers to capturing the workload on multiple databases simultaneously.

To capture a concurrent database replay workload:

1. From the Enterprise menu of the Enterprise Manager Cloud Control console, select **Quality Management**, then **Database Replay**.

   If the Database Login page appears, log in as a user with administrator privileges.

   The Database Replay page appears.

2. From the Database Replay page, click the **Captured Workloads** tab, then click **Create** in the toolbar.

   The Create Capture: Plan Environment page appears.

3. Make sure that you have met both prerequisites described on this page, then enable both checkboxes and click **Next**.

   The Create Capture: Database page appears.

   > **Tip:**
   >
   > For information about the prerequisites, see "Prerequisites for Capturing a Database Workload".

4. Click **Add**.

   The Add pop-up appears.

5. Provide a Capture name, provide an optional description, then click the Target Database search icon.

   The Search and Select: Targets pop-up appears.

6. Choose a target database from the list, then click **Select**.

   The Add pop-up reappears with added sections for Database Credential and Database Host Credential.

7. Provide database credentials, database host credentials, a Database Capture Intermediate Storage Location, then click **OK**.

   - After the Capture, the files are copied to the storage location, unless you use the intermediate storage location as the final storage location.

   The selected target database now appears in the list of databases in the Select Production Databases table.

8. Add another database for concurrent capture:

   a. Follow the instructions in steps 4 through 7.

      The Create Capture: Database page reappears, and displays the additional database for capture along with the first database you specified in steps 4 through 7.

   b. Provide a name and optional description for the concurrent capture, then click **Next**.

      The Create Capture: Options page appears.

9. Select the workload capture options:

   - Under the SQL Performance Analyzer section, select whether to capture SQL statements into a SQL tuning set during workload capture.

     While Database Replay provides an analysis of how a change affects your entire system, you can use a SQL tuning set in conjunction with the SQL Performance Analyzer to gain a more SQL-centric analysis of how the change affects SQL statements and execution plans.

     By capturing a SQL tuning set during workload capture and another SQL tuning set during workload replay, you can use the SQL Performance Analyzer to compare these SQL tuning sets to each other without having to re-execute the SQL statements. This enables you to obtain a SQL Performance Analyzer report and compare the SQL performance, before and after changes, while running Database Replay.

> **Note:**
>
> Capturing SQL statements into a SQL tuning set is the default and recommended workload capture option.

- Under the Workload Filters section, select whether to use exclusion filters by selecting **Exclusion** in the Filter Mode list, or inclusion filters by selecting **Inclusion** in the Filter Mode list.

  To add filters, click **Add** and enter the filter name, session attribute, and value in the corresponding fields.

After selecting the desired workload capture options, click **Next**.

The Create Capture: Storage page appears.

10. Click the Storage Host icon, choose a target from the list, then click **Select**.

    The Storage page now requests Host Credentials and a Storage Location.

11. Provide Host Credentials, click **Browse** to select a Storage Location, then click **Next**.

    The Create Capture: Schedule page appears.

12. Schedule the starting time and duration for the capture, schedule the exporting of AWR data, then click **Next**.

    - The default capture duration is 5 minutes. Change the capture duration to capture representative activity over a time period of interest that needs to be tested.

    The Create Capture: Review page appears.

13. If all of the parameters appear as you have intended, click **Submit** to start the Capture job.

    - The "Capture SQL statements into a SQL Tuning Set during workload capture" option is enabled by default. Uncheck this option if you do not want to compare SQL tuning sets at the end of the Replay.

    The Database Replay page reappears, displays a message that the capture was created successfully, and displays the status of the capture in the Captures list, such as "Scheduled."

14. For detailed information about the capture, double-click the name of the capture.

    The Capture Summary page appears, and displays several attributes, including the average active sessions, a workload comparison, and related concurrent captures.

Monitoring a Workload Capture Using Enterprise Manager

This section describes how to monitor workload capture using Enterprise Manager. The primary tool for monitoring workload capture is Oracle Enterprise Manager. Using Enterprise Manager, you can:

- Monitor or stop an active workload capture
- View a completed workload capture

> **Tip:**
>
> If Oracle Enterprise Manager is unavailable, you can monitor workload capture using views, as described in "Monitoring Workload Capture Using Views".

This section contains the following topics:

- Monitoring an Active Workload Capture
- Stopping an Active Workload Capture
- Viewing a Completed Workload Capture

Monitoring an Active Workload Capture

This section describes how to monitor an active workload capture using Enterprise Manager.

To monitor an active workload capture:

1. From the Enterprise menu of the Enterprise Manager Cloud Control console, select **Quality Management**, then **Database Replay**.

   If the Database Login page appears, log in as a user with administrator privileges.

   The Database Replay page appears.

2. From the Captured Workloads tab of the Database Replay page for a capture that has a Status other than Completed, click the name of the desired capture from the Capture table.

   The Summary tab of the Database Replay page appears, showing detailed statistics, a chart for average active sessions that updates dynamically while the capture is in progress, a comparison of data for captured elements versus the same elements not captured, and related concurrent captures, if any.

   - The Not Captured data shown in the Active Average Sessions chart shows the database activity (database sessions) that is not being captured.
   - The values for the Total column in the Comparison section shows all of the captured and uncaptured activity in the database. Filtering, as determined by the Workload Filters you provided in the Options step of the Create Capture wizard, is primarily why some activity is captured or not. Additionally, background activities, database scheduler jobs, and unreplayable calls are not captured.
   - You can click the refresh icon in the upper right corner to update the capture while it is running.

3. To return to the Database Replay page, click the Database Replay breadcrumb.

Stopping an Active Workload Capture

This section describes how to stop an active workload capture using Enterprise Manager.

To stop an active workload capture:

1. From the Enterprise menu of the Enterprise Manager Cloud Control console, select **Quality Management**, then **Database Replay**.

   If the Database Login page appears, log in as a user with administrator privileges.

   The Database Replay page appears.

2. From the Captured Workloads tab of the Database Replay page for a capture that has a Status of Draft, click the name in the Capture table of the capture you want to stop.

   The Capture Summary page appears.

3. Click the **Stop Capture** button.

   The button label changes to Stopping Capture. When the process completes, the Status changes to Stopped.

Viewing a Completed Workload Capture

This section describes how to manage a completed workload capture using Enterprise Manager.

To view a completed workload capture:

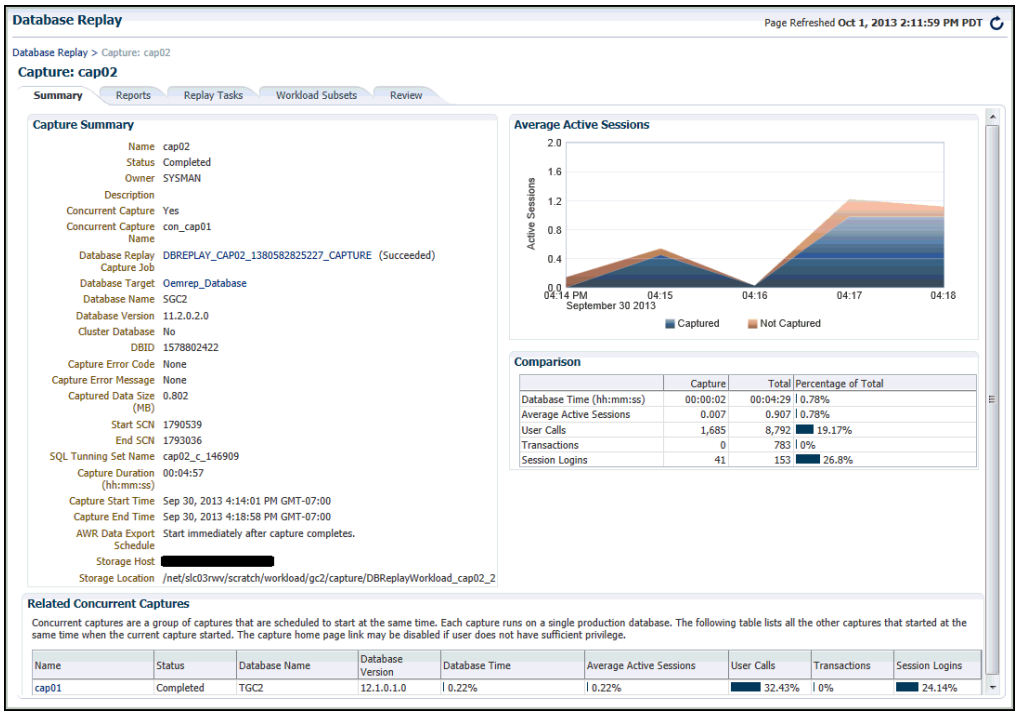
1. From the Enterprise menu of the Enterprise Manager Cloud Control console, select **Quality Management**, then **Database Replay**.

   If the Database Login page appears, log in as a user with administrator privileges.

   The Database Replay page appears.

2. From the Capture Workloads tab of the Database Replay page, click the name of a capture that has a Status of Completed.

   The contents of the Summary tab of the Database Replay page appears as for a capture in progress, except the Average Active Sessions chart shows aggregated data for the capture time period, rather than dynamic data recorded during the capture.



The Average Active Sessions chart provides a graphic display of the captured session activity compared to the uncaptured session activity (such as background activities or filtered sessions). This chart appears only when Active Session History (ASH) data is available for the capture period.

Under Comparison, various statistics for the workload capture are displayed:

- Capture column

  Displays the statistics for the captured session activity.

- Total column

  Displays the statistics for the total session activity.

- Percentage of Total column

  Displays the percentage of total session activity that has been captured in the workload.

3. To return to the Database Replay page, click the Database Replay breadcrumb.

> **Tip:**
>
> See Analyzing Captured and Replayed Workloads for information about accessing workload capture reports.

Importing a Workload External to Enterprise Manager

You can import a workload captured through the PL/SQL interface or through a different Enterprise Manager instance into Enterprise Manager to manage it as you would for a workload originally created within Enterprise Manager. The workload can be in the process of being captured, or the Capture can be completed and the workload stored on a file system. You can also preprocess and replay the workload as you would ordinarily do for a workload created within Enterprise Manager.
This feature is available for Cloud Control Database plug-in 12.1.0.5 and later releases.

To import a database workload external to Enterprise Manager:

1. From the Database Replay page, click the **Captured Workloads** tab, then click **Import** in the toolbar.

   The Import Workload: Source page appears.

2. Select one of the three choices available to import a captured workload, then click **Next**:

   - **Import a completed captured workload from a directory in the file system**

     This option typically applies for a workload created using an API, in which you now want to import it to Enterprise Manager for subsequent processing. In this case, Enterprise Manager may not even be managing the Capture database.

   - **Import a completed captured workload from a database target**

     In this case, Enterprise Manager is probably already managing the Capture database. The Capture could have been done on this database, or it could have been loaded in as would be the case for the option above.

   - **Attach to a Capture running in a database target**

     This option is similar to the option above, except this a Capture that is still in progress, rather than one that has already completed.

   The Import Workload: Database page appears.

3. Click **the search icon next to the Database Target field and select a database from the pop-up that appears.**

> **Note:**
>
> The target version of the database loading the workload must be at least as high as the one you used to capture the workload. For instance, if you captured using Oracle Database 12x, the database you select to read the Capture must be at least version 12x.

- The system now requests database and host credentials.
- In the previous step, if you chose to import a completed captured workload from a directory in the file system, the system also requires a workload location.
- A consolidated Replay has a different directory structure in which there are at least two capture directories. Consequently, if the workload directory contains a consolidated Replay, you need to enable the check box so that Enterprise Manager can be aware of the consolidated Replay during the import operation.

4. Provide the requisite input for the step above, then click **Next**.

   The Import Workload: Workload page appears.

   - If you chose "Import a completed captured workload from a directory in the file system" in step 2, this page provides a Load Workload button.
   - If you chose “Import a completed captured workload from a database target" or “Attach to a Capture running in a database target" in step 2, this page provides a Discover Workload button.

5. Click either **Load Workload** or **Discover Workload**, depending on which button is available in accordance with your selection in step 2.

   The system displays workloads, if found, in the Discovered Workloads table.

6. Either click **Next** to load the workload, or select one of the workloads, then click **Next** to continue importing the workload.

   The Import Workload: Replay page appears only under the following conditions:

   - You chose "Import a completed captured workload from a directory in the file system" in step 2.
   - The workload contains one or more Replays.

7. *Optional:* Select one or more Replays, if desired, provide a **Replay Task Name**, then click **Next**.

   The Import Workload: Review page appears.

8. If everything appears as you have intended, click **Submit**.

   The Database Replay page reappears and displays a message stating that the job was submitted successfully. The Status column for your loaded or imported workload in the Captured Workload table will show In Progress.

> **Tip:**
>
> You can check on the job's progress by clicking on the captured workload name that you just submitted in the Review step. A Capture Summary page appears, and you can click the Database Replay Import Job link to see the progress of the job execution steps.

Creating Subsets from an Existing Workload

For cases in which you have captured a large workload covering a long period of time, you may only want to replay a portion of it to accelerate testing. The Database Replay Workload Subsetting feature enables you to create new workloads by extracting portions of an existing captured workload.

Enterprise Manager provides a wizard to extract a subset of data from an existing workload that you can use for Replay on a test system. Each extracted subset is a legitimate workload that can be replayed on its own or with other workloads in a consolidated Replay.

To perform a Replay, you need to preprocess the workloads.

This feature is available for Cloud Control Database plug-in 12.1.0.5 and later releases.

To extract subsets from a workload:

1. From the Captured Workloads tab of the Database Replay page, select a workload for which you want to extract a subset, then click **Subset**.

   The Subset Workload: Define page appears, showing an Active Sessions History chart for the workload.

2. Select a starting and ending time for the subset you want to extract from the workload:

   a. Click **Add** above the Subsets table at the bottom of the page.

      The Create Subset pop-up appears.

   b. Select either snapshot or calendar times, provide start and end times, then click **OK**.

      > **Note:**
      >
      > Snapshot time is the preferred choice, because not all performance data may be available for the calendar time you select.

      Your selected time period now appears in the Active Sessions chart as a greyed-out segment.

   c. *Optional:* Define one or more additional subsets with different time periods than those you selected in the step above.

   d. *Optional:* In the Advanced Parameters section, indicate whether you want to include incomplete calls after the subset workload ends. The default is to include incomplete calls when the subset workload begins.

      - These parameters enable you to include the calls outside of your defined boundaries. For example, when you specify a starting and ending time as the boundaries for a transaction, the transaction may have started before your indicated start time, and may also have continued after your indicated end time.

3. Click **Next**.

   The Subset Workload: Database page appears.

4. Click **the search icon next to the Database Target field and select a database for subsetting from the Search and Select: Targets pop-up that appears.**

   The system now requests database and host credentials.

5. Provide the requisite input for the step above, then click **Next**.

   The Subset Workload: Location page appears.

   - If the source host and staged database host are the same, the location is pre-populated, so you do not need to provide the location for the source workload files.
   - If the source host and staged database host are not the same, do the following:

     a. Choose whether you want to access the workload files from the host name shown in the Host field, or whether you want to copy the files from the source host to the Destination Host shown.

        Access Directly means that the database to be used to subset the workload can access the original workload directly using the specified file system location. This is typically the case when the original workload is stored at a network shared location.

        Access Copy means that the two hosts are not in the shared network path. You need to provide the source host credentials so that Enterprise Manager can copy the original workload from its current location to the specified location on the subset database host.

     b. Depending on your choice above, either provide the directory location containing the workload files, or provide the location for the destination host.

6. In the Subset field, specify the storage location for each subset, then click **Next**.

   The Subset Workload: Schedule page appears.

7. Indicate when you want to start the subset job, then click **Next**.

   The Subset Workload: Review page appears.

8. If everything appears as you have intended, click **Submit**.

   The Database Replay page reappears and displays a message stating that the job was submitted successfully. The Status column for your subset in the Replay Tasks table will show In Progress.

> **Tip:**
>
> You can check on the job's progress by clicking on the subset name that you just submitted in the Review step. A Capture Summary page appears, and you can click the Database Replay Subset Job link to see the progress of the job execution steps.

Copying or Moving a Workload to a New Location

You can use the copy function for two purposes. The purposes are:

- Duplicate the Capture files from the source to another host and location.
- Move the Capture files to a new host and location, and delete the source files from the original location.

This feature is available for Cloud Control Database plug-in 12.1.0.5 and later releases.

To copy a workload to a new location:

1. From the Captured Workloads tab of the Database Replay page, select a workload you want to copy to a new location, then click **Copy**.

The Copy Workload page appears, and displays the current source location of the workload directory you selected.

2. Provide or change credentials for the storage host, if necessary. The system automatically picks up the previously defined credentials for the current storage host.
3. Leave the After Copy radio button enabled, which is "Keep original workload in the source location."
4. Select the **Storage Host** for the new location of the workload directory.
5. Provide credentials for the new storage host.
6. Select the directory for the new **Destination Location** for the workload.
7. Schedule the job, then click **Submit**.

   The Database Replay page reappears and displays a message stating that the job was submitted and that the storage location has been updated successfully.

> **Tip:**
>
> You can check on the job's progress by going to the Job Activity page, searching for the job name that appeared in the submit message, then clicking its link to access the Job Run page.

Capturing a Database Workload Using APIs

This section describes how to capture a database workload using APIs. You can also use Oracle Enterprise Manager to capture database workloads, as described in "Capturing a Database Workload Using Enterprise Manager".
Capturing a database workload using the `DBMS_WORKLOAD_CAPTURE` package involves:

- Defining Workload Capture Filters
- Starting a Workload Capture
- Stopping a Workload Capture
- Exporting AWR Data for Workload Capture
- Importing AWR Data for Workload Capture

> **See Also:**
>
> - *Oracle Database PL/SQL Packages and Types Reference* for information about the `DBMS_WORKLOAD_CAPTURE` package

Defining Workload Capture Filters

This section describes how to add and remove workload capture filters. For information about using workload filters with workload capture, see "Using Filters with Workload Capture".

To add filters to a workload capture:

- Use the `ADD_FILTER` procedure:

```
BEGIN
  DBMS_WORKLOAD_CAPTURE.ADD_FILTER (
                           fname => 'user_ichan',
                           fattribute => 'USER',
                           fvalue => 'ICHAN');
END;
/
```

In this example, the `ADD_FILTER` procedure adds a filter named user_ichan, which can be used to filter out all sessions belonging to the user name ICHAN.

The `ADD_FILTER` procedure in this example uses the following parameters:

- The `fname` required parameter specifies the name of the filter that will be added.
- The `fattribute` required parameter specifies the attribute on which the filter will be applied. Valid values include PROGRAM, MODULE, ACTION, SERVICE, INSTANCE_NUMBER, and USER.
- The `fvalue` required parameter specifies the value for the corresponding attribute on which the filter will be applied. It is possible to use wildcards such as % with some of the attributes, such as modules and actions.

To remove filters from a workload capture:

- Use the `DELETE_FILTER` procedure:

```
BEGIN
  DBMS_WORKLOAD_CAPTURE.DELETE_FILTER (fname => 'user_ichan');
END;
/
```

In this example, the `DELETE_FILTER` procedure removes the filter named user_ichan from the workload capture.

The `DELETE_FILTER` procedure in this example uses the `fname` required parameter, which specifies the name of the filter to be removed. The `DELETE_FILTER` procedure will not remove filters that belong to completed captures; it only applies to filters of captures that have yet to start.

Starting a Workload Capture

This section describes how to start a workload capture.

Before starting a workload capture, you must first complete the prerequisites for capturing a database workload. The prerequisites are described in "Prerequisites for Capturing a Database Workload". You should also review the workload capture options, as described in "Workload Capture Options".

It is important to have a well-defined starting point for the workload so that the replay system can be restored to that point before initiating a replay of the captured workload. To have a well-defined starting point for the workload capture, it is preferable not to have any active user sessions when starting a workload capture. If active sessions perform ongoing transactions, those transactions will not be replayed properly in subsequent database replays, since only that part of the transaction whose calls were executed after the workload capture is started will be replayed. To avoid this problem, consider restarting the database in restricted mode using `STARTUP RESTRICT` before starting the workload capture. Once the workload capture begins, the database will automatically switch to unrestricted mode and normal operations can continue while the workload is being captured. For more information about restarting the database before capturing a workload, see "Restarting the Database".

To start a workload capture:

- Use the `START_CAPTURE` procedure:

```
BEGIN
  DBMS_WORKLOAD_CAPTURE.START_CAPTURE (name => 'dec10_peak',
                           dir => 'dec10',
                           duration => 600,
                           capture_sts => TRUE,
                           sts_cap_interval => 300,
                           plsql_mode => 'extended',
                           encryption => 'AES256');
END;
/
```

  In this example, a workload named `dec10_peak` will be captured for 600 seconds and stored in the file system directory pointed to by the database directory object named `dec10`. A SQL tuning set will also be captured in parallel with the workload capture.

  The `START_CAPTURE` procedure in this example uses the following parameters:

  - The `name` required parameter specifies the name of the workload that will be captured.
  - The `dir` required parameter specifies a directory object pointing to the directory where the captured workload will be stored.
  - The `duration` parameter specifies the number of seconds before the workload capture will end. If a value is not specified, the workload capture will continue until the `FINISH_CAPTURE` procedure is called.
  - The `capture_sts` parameter specifies whether to capture a SQL tuning set in parallel with the workload capture. If this parameter is set to `TRUE`, you can capture a SQL tuning set during workload capture, then capture another SQL tuning set during workload replay, and use SQL Performance Analyzer to compare the SQL tuning sets without having to re-execute the SQL statements. This enables you to obtain a SQL Performance Analyzer report and compare the SQL performance—before and after the change—while running Database Replay. You can also export the resulting SQL tuning set with its AWR data using the `EXPORT_AWR` procedure, as described in "Exporting AWR Data for Workload Capture".

    This feature is not supported for Oracle RAC. Workload capture filters that are defined using `DBMS_WORKLOAD_CAPTURE` do not apply to the SQL tuning set capture. The default value for this parameter is `FALSE`.
  - The `sts_cap_interval` parameter specifies the duration of the SQL tuning set capture from the cursor cache in seconds. The default value is 300. Setting the value of this parameter below the default value may cause additional overhead with some workloads and is not recommended.
  - The optional `plsql_mode` parameter determines how PL/SQL is handled by DB Replay during capture and replays.

    These two values can be set for the `plsql_mode` parameter:

    * `top_level`: Only top-level PL/SQL calls are captured and replayed, which is how DB Replay handled PL/SQL prior to Oracle Database 12*c* Release 2 (12.2.0.1). This is the default value.
    * `extended`: Both top-level PL/SQL calls and SQL called from PL/SQL are captured. When the workload is replayed, the replay can be done at either top-level or extended level.

– The `encryption` parameter specifies the algorithm used for encrypting capture files.

  The following encryption standards can be used for the `encryption` parameter:

  * NULL - Capture files are not encrypted (the default value)
  * `AES128` - Capture files are encrypted using AES128
  * `AES192` - Capture files are encrypted using AES192
  * `AES256` - Capture files are encrypted using AES256

> **Note:**
>
> To run `START_CAPTURE` with encryption, you must set the password using the `oracle.rat.database_replay.encryption` (case-sensitive) identifier. The password is stored in a software keystore. For more information about creating a software keystore, see *Oracle Database Advanced Security Guide*.

Example: Setting up a password-based software keystore:

The following statement creates a password-based software keystore:

```
ADMINISTER KEY MANAGEMENT CREATE KEYSTORE 'MYKEYSTORE' IDENTIFIED BY password;
```

The following statements open the password-based software keystore and create a backup of the password-based software keystore:

```
ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN IDENTIFIED BY password;

ADMINISTER KEY MANAGEMENT SET KEY IDENTIFIED BY password WITH BACKUP;
```

The following statement adds secret *`secret_key`*, with the tag `DBREPLAY`, for client `'oracle.rat.database_replay.encryption'` to the password-based software keystore. It also creates a backup of the password-based software keystore before adding the secret:

```
ADMINISTER KEY MANAGEMENT ADD SECRET secret_key FOR CLIENT
'oracle.rat.database_replay.encryption' USING TAG 'DBREPLAY'
IDENTIFIED BY password WITH BACKUP;
```

The following statement closes the password-based software keystore:

```
ADMINISTER KEY MANAGEMENT SET KEYSTORE CLOSE IDENTIFIED BY password;
```

> **Note:**
>
> The software keystore must be kept open before running database capture and database replay.

Stopping a Workload Capture

This section describes how to stop a workload capture.

To stop a workload capture:

- Use the `FINISH_CAPTURE` procedure:

```
BEGIN
  DBMS_WORKLOAD_CAPTURE.FINISH_CAPTURE ();
END;
/
```

In this example, the `FINISH_CAPTURE` procedure finalizes the workload capture and returns the database to a normal state.

> **Tip:**
>
> After capturing a workload on the production system, you need to preprocess the captured workload, as described in Preprocessing a Database Workload .

Exporting AWR Data for Workload Capture

Exporting AWR data enables detailed analysis of the workload. This data is also required if you plan to run the Replay Compare Period report or the AWR Compare Period report on a pair of workload captures or replays.

To export AWR data:

- Use the `EXPORT_AWR` procedure:

```
BEGIN
  DBMS_WORKLOAD_CAPTURE.EXPORT_AWR (capture_id => 2);
END;
/
```

In this example, the AWR snapshots that correspond to the workload capture with a capture ID of 2 are exported, along with any SQL tuning set that may have been captured during workload capture.

The `EXPORT_AWR` procedure uses the `capture_id` required parameter, which specifies the ID of the capture whose AWR snapshots will be exported. The value of the `capture_id` parameter is displayed in the `ID` column of the `DBA_WORKLOAD_CAPTURES` view.

> **Note:**
>
> This procedure works only if the corresponding workload capture was performed in the current database and the AWR snapshots that correspond to the original capture time period are still available.

> **See Also:**
>
> *Oracle Database Reference* for information about the `DBA_WORKLOAD_CAPTURES` view

Importing AWR Data for Workload Capture

After AWR data is exported from the capture system, you can import the AWR data into another system, such as a test system where the captured workload will be replayed. Importing AWR data enables detailed analysis of the workload. This data is also required if you plan to run the Replay Compare Period report or the AWR Compare Period report on a pair of workload captures or replays.

To import AWR data:

- Use the `IMPORT_AWR` function, as shown in the following example:

```
CREATE USER capture_awr
SELECT DBMS_WORKLOAD_CAPTURE.IMPORT_AWR (capture_id => 2,
                                         staging_schema => 'capture_awr')
  FROM DUAL;
```

  In this example, the AWR snapshots that correspond to the workload capture with a capture ID of 2 are imported using a staging schema named `capture_awr`.

  The `IMPORT_AWR` procedure in this example uses the following parameters:

  – The `capture_id` required parameter specifies the ID of the capture whose AWR snapshots will be imported. The value of the `capture_id` parameter is displayed in the `ID` column of the `DBA_WORKLOAD_CAPTURES` view.

  – The `staging_schema` required parameter specifies the name of an existing schema in the current database which will be used as a staging area while importing the AWR snapshots from the capture directory to the `SYS AWR` schema.

> **Note:**
>
> This function fails if the schema specified by the `staging_schema` parameter contains any tables with the same name as any of the AWR tables.

> **See Also:**
>
> *Oracle Database Reference* for information about the `DBA_WORKLOAD_CAPTURES` view

Encrypting and Decrypting an Existing Workload Capture Using APIs

This section describes how to encrypt and decrypt an existing workload capture using APIs.

During a workload capture, a variety of information such as connection strings, SQL text, and bind values are saved. This information can be encrypted if it contains sensitive data. You can enable encryption during workload capture as described in Starting a Workload Capture.

Encrypting an Existing Workload Capture

This section describes how to encrypt an existing workload capture.

To encrypt an existing workload capture:

- Use the `ENCRYPT_CAPTURE` procedure:

```
BEGIN
  DBMS_WORKLOAD_CAPTURE.ENCRYPT_CAPTURE(src_dir => 'dec10',
                                        dst_dir => 'dec10_enc',
                                        encryption => 'AES128');
END;
/
```

  The `ENCRYPT_CAPTURE` procedure in this example uses the following parameters:

  - The `src_dir` parameter points to the directory that contains the workload capture to be encrypted.
  - The `dst_dir` parameter points to the directory where the encrypted capture is saved after encryption.
  - The `encryption` parameter specifies the algorithm used for encrypting the workload capture.

> **Note:**
>
> Before running `DBMS_WORKLOAD_CAPTURE.ENCRYPT_CAPTURE`, you must store the `oracle.rat.database_replay.encryption` (case-sensitive) identifier in a software keystore.

> **See Also:**
>
> *Oracle Database PL/SQL Packages and Types Reference*

Decrypting an Encrypted Workload Capture

This section describes how to decrypt an encrypted workload capture.

An encrypted workload capture can be decrypted by using the `DBMS_WORKLOAD_CAPTURE.DECRYPT_CAPTURE` procedure.

To decrypt an encrypted workload capture:

- Use the `DECRYPT_CAPTURE` procedure:

```
BEGIN
  DBMS_WORKLOAD_CAPTURE.DECRYPT_CAPTURE(src_dir => 'dec10_enc',
                                        dst_dir => 'dec10');
```

```
END;
/
```

The `DECRYPT_CAPTURE` procedure in this example uses the following parameters:

- The `src_dir` parameter points to the directory that contains the encrypted capture.
- The `dst_dir` parameter points to the directory where the decrypted capture is saved after decryption.

> **Note:**
>
> Before running `DBMS_WORKLOAD_CAPTURE.DECRYPT_CAPTURE`, you must store the `oracle.rat.database_replay.encryption` (case-sensitive) identifier in a software keystore.

> **See Also:**
>
> *Oracle Database PL/SQL Packages and Types Reference*

Monitoring Workload Capture Using Views

This section summarizes the views that you can display to monitor workload capture. You can also use Oracle Enterprise Manager to monitor workload capture, as described in "Monitoring a Workload Capture Using Enterprise Manager".
To access these views, you need DBA privileges:

- The `DBA_WORKLOAD_CAPTURES` view lists all the workload captures that have been captured in the current database.
- The `DBA_WORKLOAD_FILTERS` view lists all workload filters used for workload captures defined in the current database.

> **See Also:**
>
> - *Oracle Database Reference* for information about the `DBA_WORKLOAD_CAPTURES` view
> - *Oracle Database Reference* for information about the `DBA_WORKLOAD_FILTERS` view

11

Preprocessing a Database Workload

After a workload is captured and setup of the test system is complete, the captured data must be preprocessed. Preprocessing a captured workload creates all necessary metadata for replaying the workload. This must be done once for every captured workload before they can be replayed. After the captured workload is preprocessed, it can be replayed repeatedly on a replay system.
To preprocess a captured workload, you will first need to move all captured data files from the directory where they are stored on the capture system to a directory on the instance where the preprocessing will be performed. Preprocessing is resource intensive and should be performed on a system that is:

- Separate from the production system
- Running the same version of Oracle Database as the replay system

For Oracle Real Application Clusters (Oracle RAC), select one database instance of the replay system for the preprocessing. This instance must have access to the captured data files that require preprocessing, which can be stored on a local or shared file system. If the capture directory path on the capture system resolves to separate physical directories in each instance, you will need to merge them into a single capture directory where the preprocessing will be performed. All directories must have the same directory tree and all files contained in each of these directories must be moved into a directory that has the same relative path to the capture directory.

Typically, you will preprocess the captured workload on the replay system. If you plan to preprocess the captured workload on a system that is separate from the replay system, you will also need to move all preprocessed data files from the directory where they are stored on the preprocessing system to a directory on the replay system after preprocessing is complete.

This chapter contains the following sections:

Preparing a Single Database Workload Using Enterprise Manager

Several tasks are involved in preparing a single workload. For example:

- Creating a database replay task
- Creating a replay from a replay task
- Preparing the test database
- Preprocessing the workload and deploying the replay clients

Before you can preprocess a captured workload, you must first capture the workload on the production system, as described in Capturing a Database Workload .

> **Note:**
>
> Preparing the test database is only required if you have not done so already.

The following sections provide procedures for these tasks.

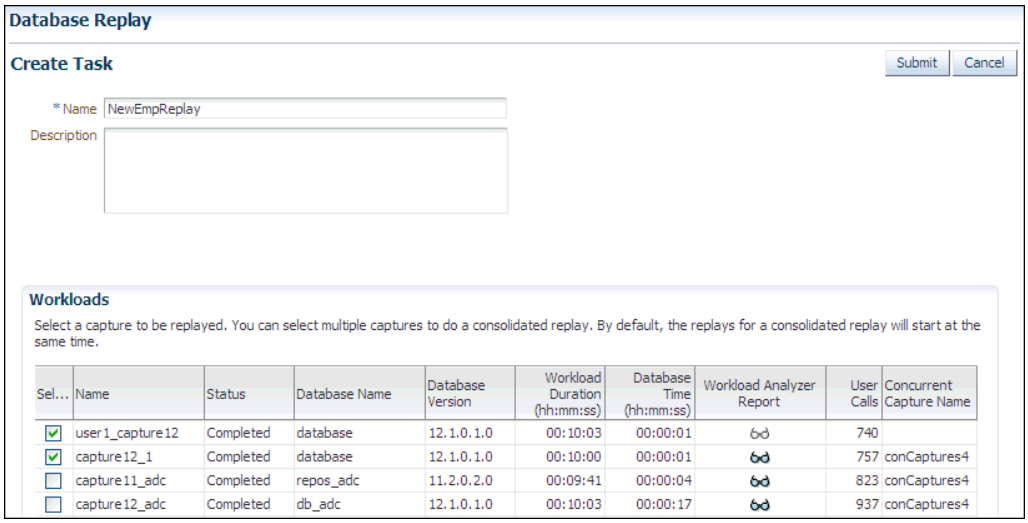
Creating a Database Replay Task

Before creating a database replay task, make sure that the capture that you want to replay has some captured user calls.

To create a database replay task:

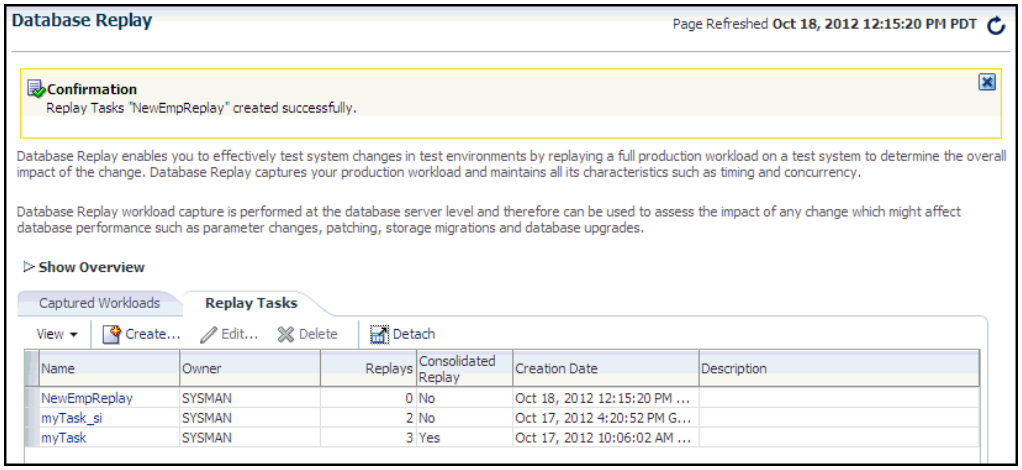
1. From the Database Replay page, click the **Replay Tasks** tab, then click **Create** in the toolbar.

   The Create Task page appears.



2. Provide a **Name** for the task, select a capture to be replayed, then click **Submit**. For consolidated replays, select two or more captures.

   The Database Replay page reappears, and displays your newly created replay task in the table under the Replay Task tab.



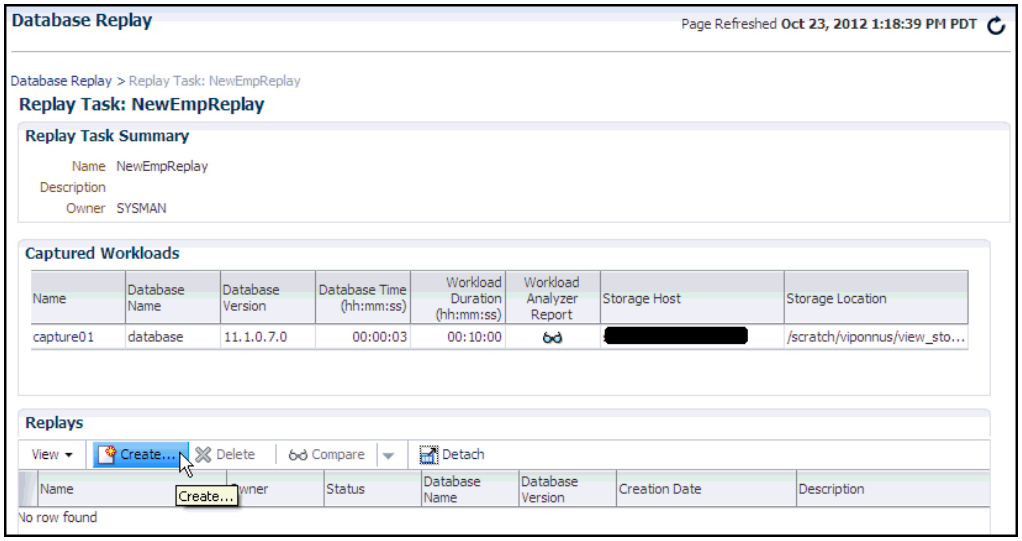
Creating a Replay from a Replay Task

This topic describes how to create a replay from a replay task.

To create the replay:

1. From the Database Replay page, click the **Replay Tasks** tab.
2. Click the link for the desired replay task in the table.

   The Replay Task page for the capture appears.

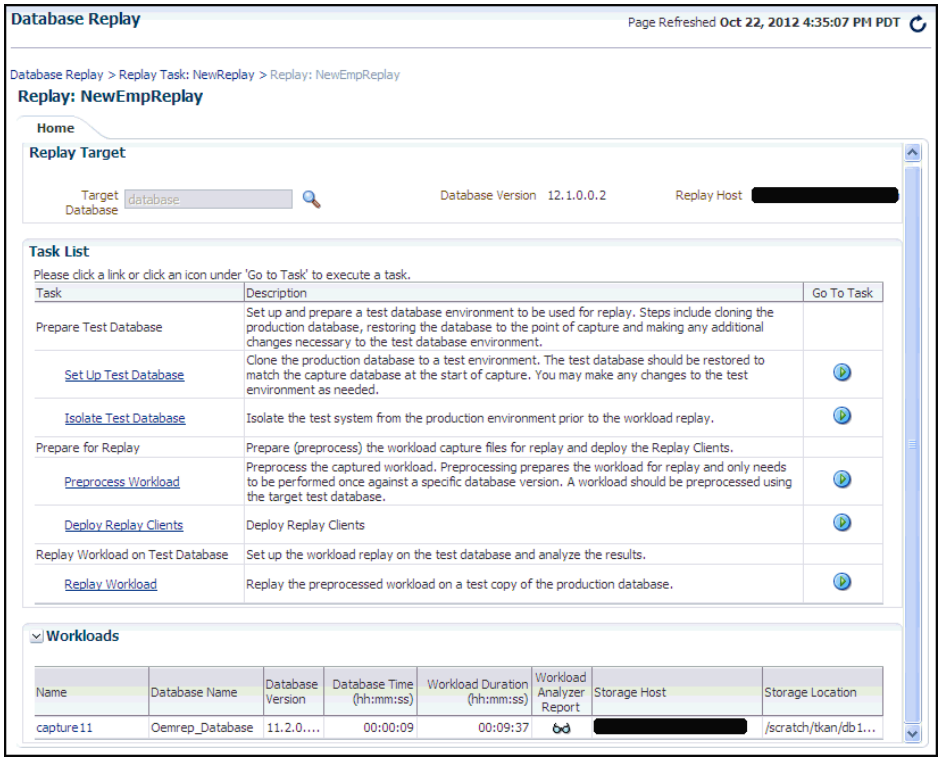


3. Click **Create** in the Replays section.

   The Create Replay pop-up appears.
4. Provide a required Name and optional description, then click the **Target Database** icon.

   The Search and Select: Targets pop-up appears.
5. Choose the desired database, then click **Select**.
6. Click **OK** in the Create Replay pop-up.

   The Database Replay page for your replay appears, which includes a Task List with links to perform the needed tasks.



You can now proceed to the first task in the Task List, described in the next section.

Preparing the Test Database

This topic describes the tasks involved in preparing the test database. For example:

- Setting up the test database
- Isolating the test database

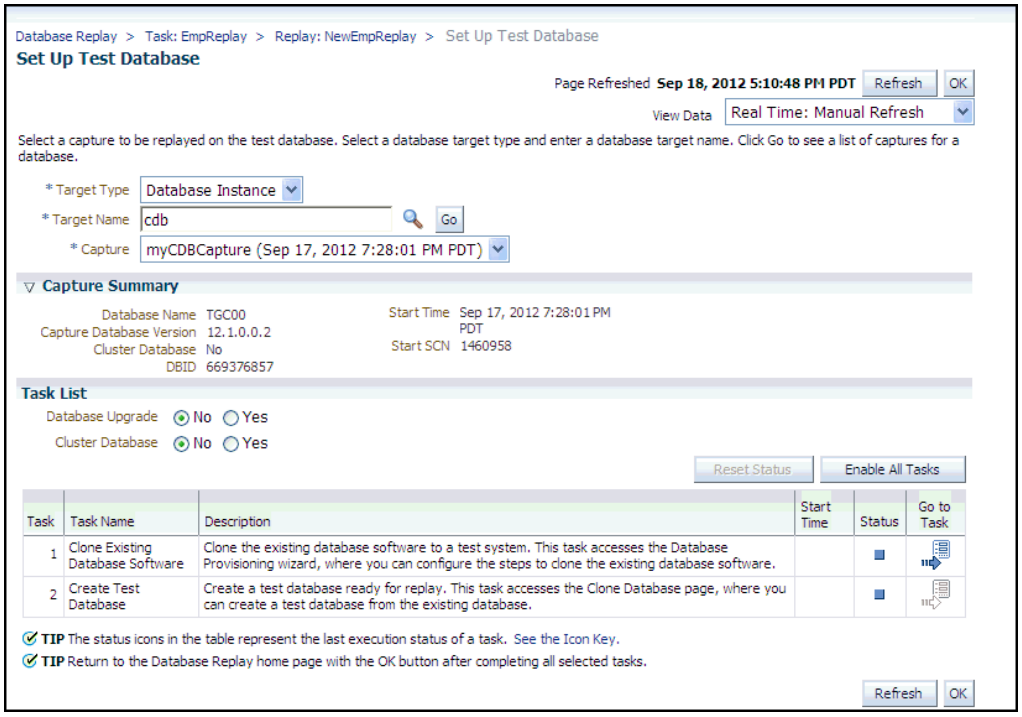
> **Note:**
>
> These tasks are optional. If you have already set up your test database, skip to "Preprocessing the Workload and Deploying the Replay Clients".

The following procedures explain how to perform each of these tasks, which you can do in any order.

To set up the test database:

1. From the Replay page for your particular replay, click the link for the **Set Up Test Database** task.

   The Set Up Test Database page appears.



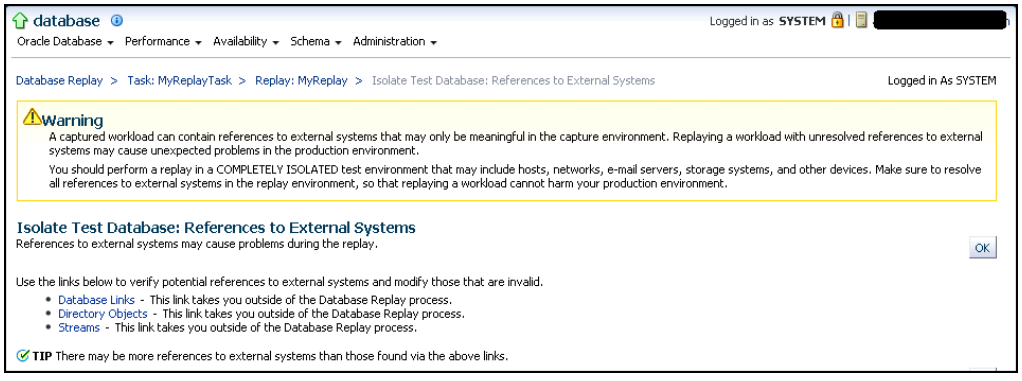
2. Choose whether you want to upgrade the database or not, and indicate whether this is a cluster database.
3. Click the **Go to Task** icon for the Clone Existing Database Software sub-task, or click **Enable All Tasks** if you want to create the test database first.
4. Follow the instructions provided in the online help for the wizards.

   When the tasks are complete, a checkmark appears in the Status column for each task.
5. Click **OK** to return to the Database Replay page.

To isolate the test database:

1. From the Replay page for your particular replay, click the link for the **Isolate Test Database** task.

   A page appears explaining that references to external systems can cause problems during the replay.



2. Use the links provided to verify potential references to external systems, modify those that are invalid, then click **OK**.

   The Replay Summary page reappears.

Preprocessing the Workload and Deploying the Replay Clients

The final preparation for the replay involves preprocessing the workload and deploying the replay clients. For example:

- Preprocessing the workload

  You need to preprocess each captured workload once for each version of the database against which the workload will be replayed. After you preprocess the workload once, you can use it for any subsequent replay tasks and replays without needing to preprocess again, as long as the test database is the same version as the database where the workload was preprocessed.For instance, for a replay task that contains two replays named "MyReplay1" and "MyReplay2," after you have preprocessed "MyReplay1", you can just directly reuse the directory object to replay "MyReplay2."

  The Workload Analyzer report is available after preprocessing.

- Deploying the replay clients

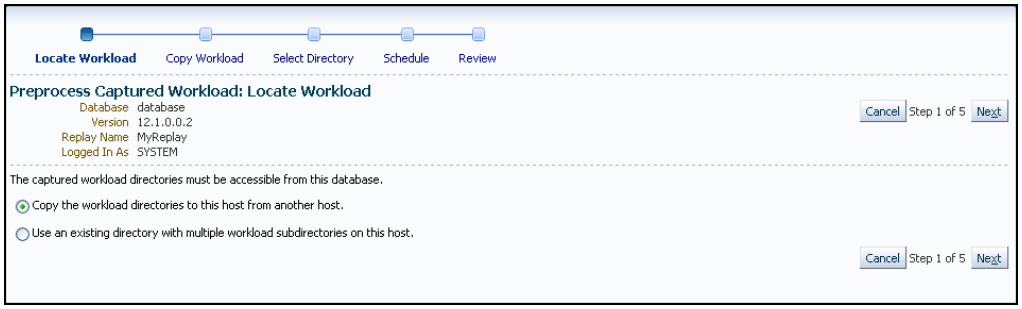
  You do not need to deploy the replay clients to other replay client hosts if these hosts can access the Oracle home of the test database you specified in the Database Target Name field.

The following procedures explain how to accomplish each of these tasks.

To preprocess the workload:

1. From the Replay page for your particular replay, click the link for the **Preprocess Workload** task.

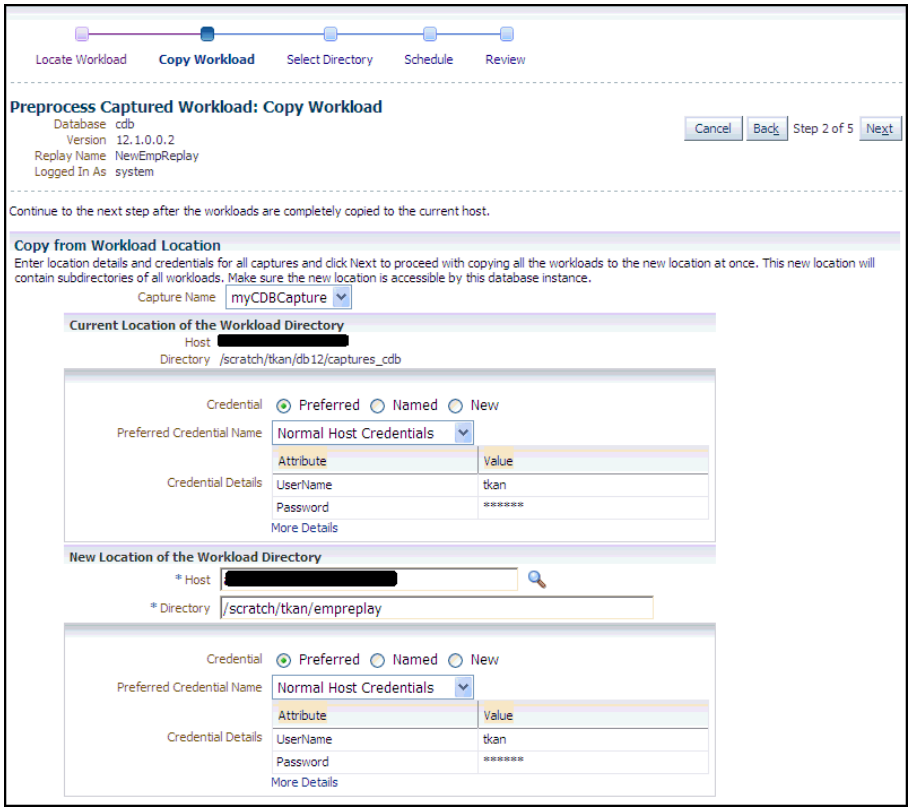
   The Preprocess Captured Workload: Locate Workload page appears.



2. Select the desired workload location option, then click **Next**.

> **Note:**
>
> You initially need to select the copy option.

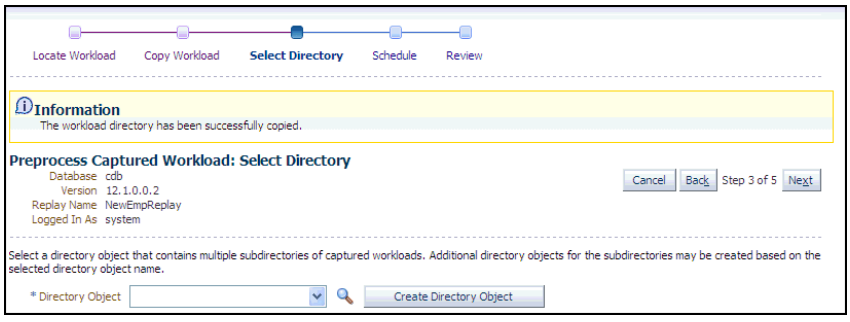
   The Preprocess Captured Workload: Copy Workload page appears.



3. Provide the required credentials and the new location to which the workloads will be copied and preprocessed, then click **Next**.

   - For a consolidated replay, there are multiple source workloads, so multiple source credentials might be needed for the current location of the workload directory. For more information on consolidated replays, see "Using Consolidated Database Replay with Enterprise Manager."

   The system responds by displaying a progress bar graph during processing, then displays the Preprocess Captured Workload: Select Directory page after the copy operation concludes.



4. Specify the Directory Object, or create a new Directory Object that points to the location that contains the workload. If you chose to copy from the workload location to a new location in the previous step, make sure that the directory object points to the exact location you specified in the New Location of the Workload Directory section.

   The system responds by displaying a Capture Summary. You can now expand the Capture Details section to see the workload profile and workload filters. The Capture Summary does not appear for consolidated replays.

   Click **Next** to display the Preprocess Captured Workload: Schedule page.

5. Provide input to schedule the preprocess job:
   a. Provide your own required job name or accept the system-supplied name. The job system automatically names the job in uppercase.
   b. Indicate whether you want the job to run as soon as you submit it, or whether you want it to run at a later time.
   c. Provide the host credentials, which are used to run the preprocess job in the operating system.

   Click **Next** to display the Preprocess Captured Workload: Review page.
6. Check to make sure that the settings are what you intend, then click **Submit**.

   The Database Replay page appears, and assuming that there were no errors in your input, a confirmation message at the top of the page states "Job JOBNAME to prepare the workload has been created successfully."
7. Click the *JOBNAME* link to check the status of the job. The job must succeed before you can proceed to the Replay Workload task.

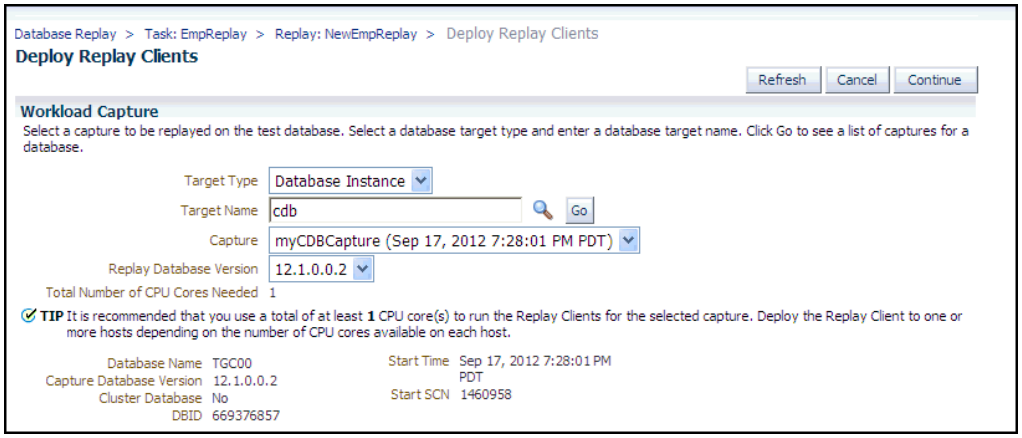
> **Note:**
>
> A message may appear in the Task List stating that you need to install an additional PL/SQL package in the test database to generate a compare period report after the trial. Click **Install PL/SQL Package** to resolve this issue before proceeding to the Replay Workload task.

> **Tip:**
>
> After preprocessing a captured workload, you can replay it on the test system, as described in Replaying a Database Workload .

To deploy the replay clients:

1. From the Replay page for your particular replay, click the link for the **Deploy Replay Clients** task.

   The Deploy Replay Clients page appears.



2. Accept the default values defined for the associated workload capture, or override these values, then click **Continue**.

   The Provision Oracle Database Client wizard appears.

3. Follow the instructions provided in the online help for each step of the wizard.

   After you click Submit in the Review step to run the deployment procedure according to the schedule you have set, the Replay Summary page reappears.

Preprocessing a Database Workload Using APIs

This section describes how to preprocess a captured workload using the `DBMS_WORKLOAD_REPLAY` package. You can also use Oracle Enterprise Manager to preprocess a captured workload, as described in "Preparing a Single Database Workload Using Enterprise Manager".

Before you can preprocess a captured workload, you must first capture the workload on the production system, as described in Capturing a Database Workload .

To preprocess a captured workload:

- Use the `PROCESS_CAPTURE` procedure:

```
BEGIN
  DBMS_WORKLOAD_REPLAY.PROCESS_CAPTURE (capture_dir => 'dec06',
                                        plsql_mode => 'extended');
END;
/
```

  In this example, the captured workload stored in the `dec06` directory will be preprocessed.

  The `PROCESS_CAPTURE` procedure in this example uses the `capture_dir` required parameter, which specifies the directory that contains the captured workload to be preprocessed.

  The optional `plsql_mode` parameter specifies the processing mode for PL/SQL.

  These two values can be set for the `plsql_mode` parameter:

  – `top_level`: Metadata is generated for top-level PL/SQL calls only; this will be the only option for replay. This is the default value.

  – `extended`: Metadata is generated for both top-level PL/SQL calls and the SQL called from PL/SQL. A new directory `ppe_X.X.X.X` (where X's represent the current Oracle version) is created under the capture root directory. Capture must have been done with this same value for the `plsql_mode` parameter. Replay can use either `'TOP_LEVEL'` or `'EXTENDED'`.

    The `extended` value can be set only for workloads that were captured with the `plsql_mode` parameter set to `extended`. If `extended` is specified, but the capture was not executed in `extended` mode, then you will receive an error message.

> **Note:**
>
> To run `PROCESS_CAPTURE` on an encrypted workload capture, you need to set the password using the identifier `oracle.rat.database_replay.encryption` (case-sensitive). The password is stored in a software keystore. You can find whether a workload capture is encrypted or not from `DBA_WORKLOAD_CAPTURES` view.

> **Tip:**
>
> After preprocessing a captured workload, you can replay it on the test system, as described in Replaying a Database Workload .

> **See Also:**
>
> - Starting a Workload Capture for information about the `plsql_mode` parameter for the `START_CAPTURE` procedure for the `DBMS_WORKLOAD_CAPTURE` package
> - *Oracle Database PL/SQL Packages and Types Reference* for additional information about the `DBMS_WORKLOAD_REPLAY.PROCESS_CAPTURE` procedure

Running the Workload Analyzer Command-Line Interface

The Workload Analyzer is a Java program that analyzes a workload capture directory and identifies parts of a captured workload that may not replay accurately due to insufficient data, errors that occurred during workload capture, or usage features that are not supported by Database Replay. The results of the workload analysis are saved to an HTML report named `wcr_cap_analysis.html` located in the capture directory that is being analyzed. If an error can be prevented, the workload analysis report displays available preventive actions that can be implemented before replay. If an error cannot be corrected, the workload analysis report provides a description of the error so it can be accounted for during replay. Running Workload Analyzer is the default option and is strongly recommended.

> **Note:**
>
> If you are preprocessing a workload capture using Oracle Enterprise Manager, then you do not need to run Workload Analyzer in the command-line interface. Oracle Enterprise Manager enables you to run Workload Analyzer as part of the workload preprocessing.

Workload Analyzer is composed of two JAR files, `dbranalyzer.jar` and `dbrparser.jar`, located in the `$ORACLE_HOME/rdbms/jlib/` directory of a system running Oracle Database Enterprise Edition Release 11.2.0.2 or higher. Workload Analyzer requires Java 1.5 or higher and the `ojdbc6.jar` file located in the `$ORACLE_HOME/jdbc/lib/` directory.

To run Workload Analyzer:

1. In the command-line interface, run the following `java` command on a single line:

   ```
   java -classpath
   $ORACLE_HOME/jdbc/lib/ojdbc6.jar:$ORACLE_HOME/rdbms/jlib/dbrparser.jar:
   $ORACLE_HOME/rdbms/jlib/dbranalyzer.jar:
   oracle.dbreplay.workload.checker.CaptureChecker
   <capture_directory> <connection_string>
   ```

   For the `capture_directory` parameter, input the operating system path of the capture directory. This directory should also contain the exported AWR data for the workload capture. For the `connection_string` parameter, input the connection string of an Oracle database that is release 11.1 or higher.

   An example of this command may be:

   ```
   java -classpath
   $ORACLE_HOME/jdbc/lib/ojdbc6.jar:$ORACLE_HOME/rdbms/jlib/dbrparser.jar:
   $ORACLE_HOME/rdbms/jlib/dbranalyzer.jar:
   oracle.dbreplay.workload.checker.CaptureChecker /scratch/capture
   jdbc:oracle:thin:@myhost.mycompany.com:1521:orcl
   ```

2. When prompted, input the username and password of a database user with `EXECUTE` privileges for the `DBMS_WORKLOAD_CAPTURE` package and the `SELECT_CATALOG` role on the target database.

12

Replaying a Database Workload

After a captured workload is preprocessed, it can be replayed repeatedly on a replay system that is running the same version of Oracle Database. Typically, the replay system where the preprocessed workload will be replayed should be a test system that is separate from the production system.
This chapter describes how to replay a database workload on the test system and contains the following sections:

- Steps for Replaying a Database Workload
- Replaying a Database Workload Using Enterprise Manager
- Setting Up the Replay Schedule and Parameters Using Enterprise Manager
- Monitoring Workload Replay Using Enterprise Manager
- Importing a Replay External to Enterprise Manager
- Replaying a Database Workload Using APIs
- Monitoring Workload Replay Using APIs

> **Tip:**
>
> Before you can replay a database workload, you must first:
>
> - Capture the workload on the production system, as described in Capturing a Database Workload
> - Preprocess the captured workload, as described in Preprocessing a Database Workload

Steps for Replaying a Database Workload

Proper preparation of the replay system and planning of the workload replay ensures that the replay will be accurate. Before replaying a database workload, complete the following steps to prepare the replay system and the workload replay:

- Setting Up the Replay Directory
- Restoring the Database
- Resolving References to External Systems
- Connection Remapping
- User Remapping
- Specifying Replay Options
- Using Filters with Workload Replay
- Setting Up Replay Clients

Setting Up the Replay Directory

The captured workload must have been preprocessed and copied to the replay system. A directory object for the directory to which the preprocessed workload is copied must exist in the replay system.

Restoring the Database

Before a workload can be replayed, the application data state on the replay system should be logically equivalent to that of the capture system at the start time of workload capture. This minimizes replay divergence during replay. The method for restoring the database depends on the backup method that was used before capturing the workload. For example, if RMAN was used to back up the capture system, you can use RMAN `DUPLICATE` capabilities to create the test database. For more information, see "Prerequisites for Capturing a Database Workload". After the database is created with the appropriate application data on the replay system, perform the system change you want to test, such as a database or operating system upgrade. The primary purpose of Database Replay is to test the effect of system changes on a captured workload. Therefore, the system changes you make should define the test you are conducting with the captured workload.

Resolving References to External Systems

A captured workload may contain references to external systems, such as database links or external tables. Typically, you should reconfigure these external interactions to avoid impacting other production systems during replay. External references that need to be resolved before replaying a workload include:

- Database links

  It is typically not desirable for the replay system to interact with other databases. Therefore, you should reconfigure all database links to point to an appropriate database that contains the data needed for replay.

- External tables

  All external files specified using directory objects referenced by external tables need to be available to the database during replay. The content of these files should be the same as during capture, and the filenames and directory objects used to define the external tables should also be valid.

- Directory objects

  You should reconfigure any references to directories on the production system by appropriately redefining the directory objects present in the replay system after restoring the database.

- URLs

  URLs/URIs that are stored in the database need to be configured so that Web services accessed during the workload capture will point to the proper URLs during replay. If the workload refers to URLs that are stored in the production system, you should isolate the test system network during replay.

- E-mails

  To avoid resending E-mail notifications during replay, any E-mail server accessible to the replay system should be configured to ignore requests for outgoing E-mails.

> **Tip:**
>
> To avoid impacting other production systems during replay, Oracle strongly recommends running the replay within an isolated private network that does not have access to the production environment hosts.

Connection Remapping

During workload capture, connection strings used to connect to the production system are captured. In order for the replay to succeed, you need to remap these connection strings to the replay system. The replay clients can then connect to the replay system using the remapped connections.

For Oracle Real Application Clusters (Oracle RAC) databases, you can map all connection strings to a load balancing connection string. This is especially useful if the number of nodes on the replay system is different from the capture system. Alternatively, if you want to direct workload to specific instances, you can use services or explicitly specify the instance identifier in the remapped connection strings.

User Remapping

During workload capture, the username of the database user or schema used to connect to the production system is captured. You can choose to remap the captured username to that of a new user or schema.

Specifying Replay Options

After the database is restored, and connections and users are remapped, you can set the appropriate replay options. For example:

- Specifying the Synchronization Method
- Controlling Session Connection Rate
- Controlling Request Rate Within a Session

Specifying the Synchronization Method

The `synchronization` parameter controls the synchronization method used for database replay.

If the parameter is set to `TIME`, the replay will use the same wall-clock timing as the capture. All database session login times will be replayed exactly as the capture. Likewise, all timing between transactions within database sessions will be preserved and replayed as captured. This synchronization method will produce good replays for most workloads.

If this parameter is set to `SCN`, the `COMMIT` order in the captured workload will be observed during replay and all replay actions will be executed only after all dependent `COMMIT` actions have completed. This synchronization method may introduce significant delays for some workloads. If this is the case, it is recommended to use `TIME` as the `synchronization` parameter.

If this parameter is set to `OBJECT_ID`, all replay actions will be executed only after all relevant `COMMIT` actions have completed. Relevant `COMMIT` actions must meet the following criteria:

- Issued before the given action in the workload capture
- Modified at least one of the database objects for which the given action is referencing, either implicitly or explicitly

Setting this parameter to `OBJECT_ID` allows for more concurrency during workload replays for `COMMIT` actions that do not reference the same database objects during workload capture.

Controlling Session Connection Rate

The `connect_time_scale` parameter enables you to scale the elapsed time between the time when the workload capture began and each session connects. You can use this option to manipulate the session connect time during replay with a given percentage value. The default value is 100, which will attempt to connect all sessions as captured. Setting this parameter to 0 will attempt to connect all sessions immediately.

Controlling Request Rate Within a Session

User think time is the elapsed time while the replayed user waits between issuing calls within a single session. To control replay speed, use the `think_time_scale` parameter to scale user think time during replay.
If user calls are being executed slower during replay than during capture, you can make the database replay attempt to catch up by setting the `think_time_auto_correct` parameter to `TRUE`. This will make the replay client shorten the think time between calls, so that the overall elapsed time of the replay will more closely match the captured elapsed time.

If user calls are being executed faster during replay than during capture, setting the `think_time_auto_correct` parameter to `TRUE` will not change the think time. The replay client will not increase the think time between calls to match the captured elapsed time.

Using Filters with Workload Replay

By default, all captured database calls are replayed during workload replay. You can use workload filters to specify which database calls to include in or exclude from the workload during workload replay.

Workload replay filters are first defined and then added to a replay filter set so they can be used in a workload replay. There are two types of workload filters: inclusion filters and exclusion filters. Inclusion filters enable you to specify database calls that will be replayed. Exclusion filters enable you to specify database calls that will not be replayed. You can use either inclusion filters or exclusion filters in a workload replay, but not both. The workload filter is determined as an inclusion or exclusion filter when the replay filter set is created.

Setting Up Replay Clients

The replay client is a multithreaded program (an executable named `wrc` located in the `$ORACLE_HOME/bin` directory) where each thread submits a workload from a captured session. Before replay begins, the database will wait for replay clients to connect. At this point, you need to set up and start the replay clients, which will connect to the replay system and send requests based on what has been captured in the workload.
Before starting replay clients, ensure that the:

- Replay client software is installed on the hosts where it will run
- Replay client software has the same version as the RDBMS database software
- Replay clients have access to the replay directory

- Replay directory contains the preprocessed workload capture
- Replay user has the correct user ID, password, and privileges (the replay user needs the DBA role and cannot be the `SYS` user)
- Replay clients are not started on a system that is running the database
- Replay clients read the capture directory on a file system that is different from the one on which the database files reside

  To do this, copy the capture directory to the system where the replay client will run. After the replay is completed, you can delete the capture directory.

After these prerequisites are met, you can proceed to set up and start the replay clients using the `wrc` executable. The `wrc` executable uses the following syntax:

```
wrc [user/password[@server]] MODE=[value] [keyword=[value]]
```

The parameters `user`, `password` and `server` specify the username, password and connection string used to connect to the replay database. The parameter `mode` specifies the mode in which to run the `wrc` executable. Possible values include `replay` (the default), `calibrate`, and `list_hosts`. The parameter `keyword` specifies the options to use for the execution and is dependent on the mode selected. To display the possible keywords and their corresponding values, run the `wrc` executable without any arguments.

The following sections describe the modes that you can select when running the `wrc` executable:

- Calibrating Replay Clients
- Starting Replay Clients
- Displaying Host Information

Calibrating Replay Clients

Since one replay client can initiate multiple sessions with the database, it is not necessary to start a replay client for each session that was captured. The number of replay clients that need to be started depends on the number of workload streams, the number of hosts, and the number of replay clients for each host.

To estimate the number of replay clients and hosts that are required to replay a particular workload, run the `wrc` executable in calibrate mode.

In calibrate mode, the `wrc` executable accepts the following keywords:

- `replaydir` specifies the directory that contains the preprocessed workload capture you want to replay. If unspecified, it defaults to the current directory.
- `process_per_cpu` specifies the maximum number of client processes that can run per CPU. The default value is 4.
- `threads_per_process` specifies the maximum number of threads that can run within a client process. The default value is 50.

The following example shows how to run the `wrc` executable in calibrate mode:

```
%> wrc mode=calibrate replaydir=./replay
```

In this example, the `wrc` executable is executed to estimate the number of replay clients and hosts that are required to replay the workload capture stored in a subdirectory named `replay` under the current directory. In the following sample output, the recommendation is to use at least 21 replay clients divided among 6 CPUs:

```
Workload Replay Client: Release 12.1.0.0.1 - Production on Fri Sept 30
13:06:33 2011

Copyright (c) 1982, 2011, Oracle.  All rights reserved.

Report for Workload in: /oracle/replay/
-----------------------

Recommendation:
Consider using at least 21 clients divided among 6 CPU(s).

Workload Characteristics:
- max concurrency: 1004 sessions
- total number of sessions: 1013

Assumptions:
- 1 client process per 50 concurrent sessions
- 4 client process per CPU
- think time scale = 100
- connect time scale = 100
- synchronization = TRUE
```

Starting Replay Clients

After determining the number of replay clients that are needed to replay the workload, you need to start the replay clients by running the `wrc` executable in replay mode on the hosts where they are installed. Once started, each replay client will initiate one or more sessions with the database to drive the workload replay.
In replay mode, the `wrc` executable accepts the following keywords:

- `userid` and `password` specify the user ID and password of a replay user for the replay client. If unspecified, these values default to the `system` user.
- `server` specifies the connection string that is used to connect to the replay system. If unspecified, the value defaults to an empty string.
- `replaydir` specifies the directory that contains the preprocessed workload capture you want to replay. If unspecified, it defaults to the current directory.
- `workdir` specifies the directory where the client logs will be written. This parameter is only used with the `debug` parameter for debugging purposes.
- `debug` specifies whether debug data will be created. Possible values include:
  - `on`

    Debug data will be written to files in the working directory
  - `off`

    No debug data will be written (the default value)

> **Note:**
>
> Before running the `wrc` executable in debug mode, contact Oracle Support for more information.

- `connection_override` specifies whether to override the connection mappings stored in the `DBA_WORKLOAD_CONNECTION_MAP` view. If set to `TRUE`, connection remappings stored in the

DBA_WORKLOAD_CONNECTION_MAP view will be ignored and the connection string specified using the server parameter will be used. If set to FALSE, all replay threads will connect using the connection remappings stored in the DBA_WORKLOAD_CONNECTION_MAP view. This is the default setting.

- walletdir points to the location of the auto-login software keystore directory. The default value is an empty string. walletdir is mandatory during the replay of an encrypted workload capture. For a non-encrypted capture, walletdir must not be set and it defaults to an empty string.

> **Note:**
>
> - For an encrypted workload capture, you must set the password using the oracle.rat.database_replay.encryption (case-sensitive) identifier. The password is stored in the auto-login software keystore.
> - During replay of encrypted workload capture, you must set up a separate client-side software keystore. For example:
>
> ```
> rm -rf keystore_location
> mkdir keystore_location
> orapki -wrl keystore_location -create
> orapki -wrl keystore_location -createEntry
> 'oracle.rat.database_replay.encryption' secret_key
> orapki -wrl keystore_location -createSSO
> ```
>
> *secret_key* must match the *secret_key* that was used in the server-side software keystore which was created during the encryption of the workload capture.

The mkstore wallet management command line tool is deprecated with Oracle Database 23ai, and can be removed in a future release.

To manage wallets, Oracle recommends that you use the orapki command line tool.

After all replay clients have connected, the database will automatically distribute workload capture streams among all available replay clients and workload replay can begin. You can monitor the status of the replay clients using the V$WORKLOAD_REPLAY_THREAD view. After the replay finishes, all replay clients will disconnect automatically.

Example: Running wrc executable in replay mode for a non-encrypted capture

The following example shows how to run the wrc executable in replay mode:

```
%> wrc system/password@test mode=replay replaydir=./replay
```

In this example, the wrc executable starts the replay client to replay the workload capture stored in a subdirectory named replay under the current directory.

Example: Running wrc executable in replay mode for an encrypted capture

The following example shows how to run the `wrc` executable in replay mode for an encrypted workload capture:

```
%> wrc system/password@test mode=replay replaydir=./replay walletdir=/tmp/
replay_encrypt_cwallet
```

In this example, the `wrc` executable starts the replay client to replay the workload capture stored in a subdirectory named `replay` under the current directory. `walletdir` points to the location of the auto-login software keystore directory.

Displaying Host Information

You can display the hosts that participated in a workload capture and workload replay by running the `wrc` executable in list_hosts mode.

In list_hosts mode, the `wrc` executable accepts the keyword `replaydir`, which specifies the directory that contains the preprocessed workload capture you want to replay. If unspecified, it defaults to the current directory.

The following example shows how to run the `wrc` executable in list_hosts mode:

```
%> wrc mode=list_hosts replaydir=./replay
```

In this example, the `wrc` executable is executed to list all hosts that participated in capturing or replaying the workload capture stored in a subdirectory named `replay` under the current directory. In the following sample output, the hosts that participated in the workload capture and three subsequent replays are shown:

```
Workload Replay Client: Release 12.1.0.0.1 - Production on Fri Sept 30
13:44:48 2011

Copyright (c) 1982, 2011, Oracle.  All rights reserved.

Hosts found:
Capture:
        prod1
        prod2
Replay 1:
        test1
Replay 2:
        test1
        test2
Replay 3:
        testwin
```

Replaying a Database Workload Using Enterprise Manager

This section describes how to replay a database workload using Enterprise Manager.

Before proceeding, you must already have created a replay task and created a replay from the replay task. To do this, see "Preparing a Single Database Workload Using Enterprise Manager".

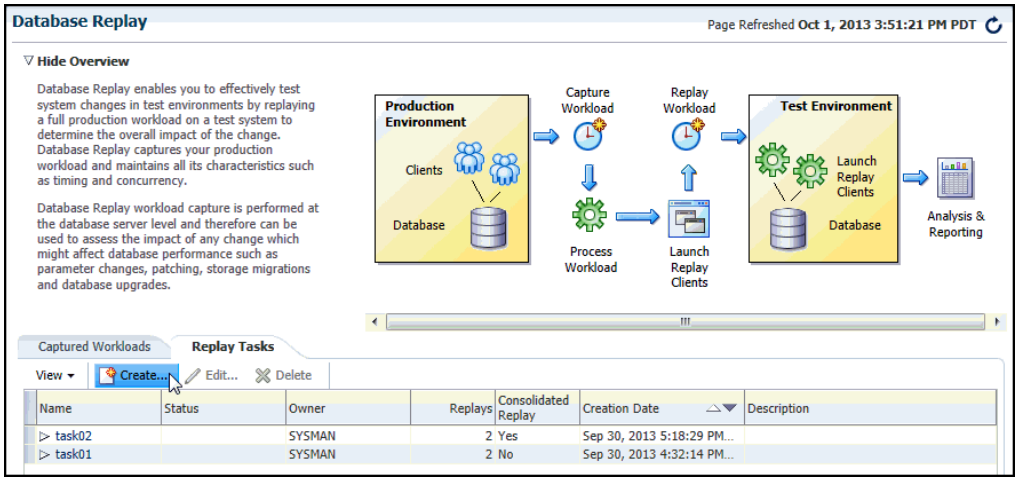
The primary tool for replaying database workloads is Oracle Enterprise Manager. If Oracle Enterprise Manager is unavailable, you can also replay database workloads using APIs, as described in "Replaying a Database Workload Using APIs".

To replay a database workload using Enterprise Manager:

1. From the Enterprise menu of the Enterprise Manager Cloud Control console, select **Quality Management**, then **Database Replay**.

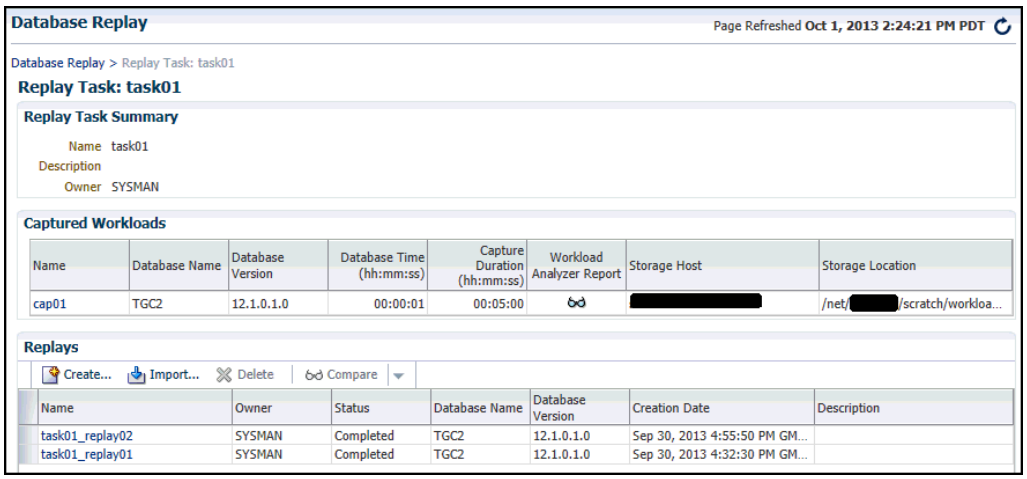
   If the Database Login page appears, log in as a user with administrator privileges.

   The Database Replay page appears.



2. Select the **Replay Tasks** tab, then click the link for the desired replay task in the table.

   The Replay Task page for the replay appears.



3. Click **Create** in the Replays section to create the replay.

   The Create Replay pop-up appears.

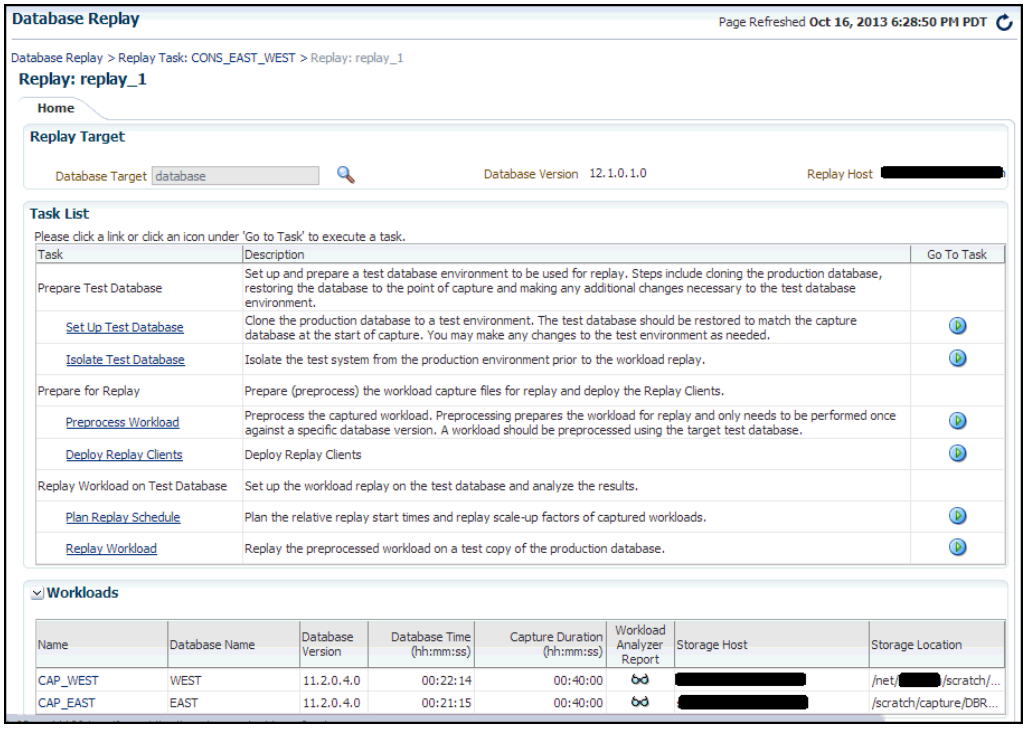
4. Provide a required Name and optional description, then click the **Target Database** icon.

   The Search and Select: Targets pop-up appears.

5. Choose the appropriate database, then click **Select**.

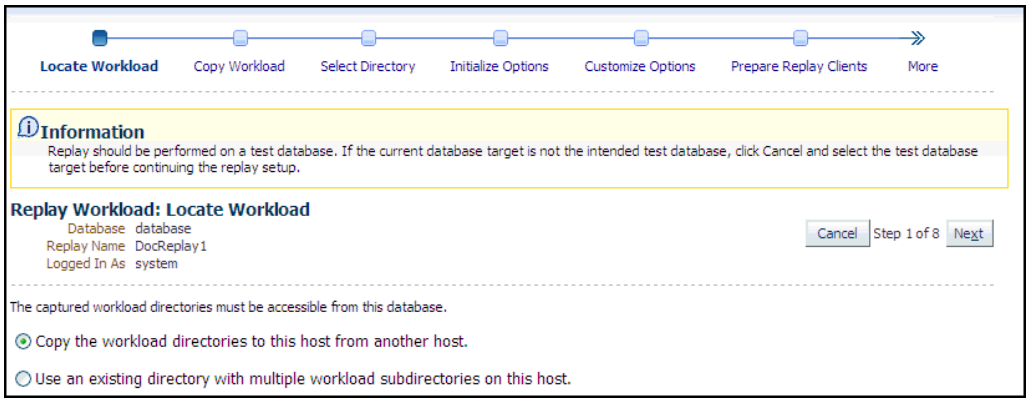
6. Click **OK** in the Create Replay pop-up.

   The Database Replay page for your replay appears, which includes a Task List with a link to perform the replay.



7. Click the link for the **Replay Workload** task.

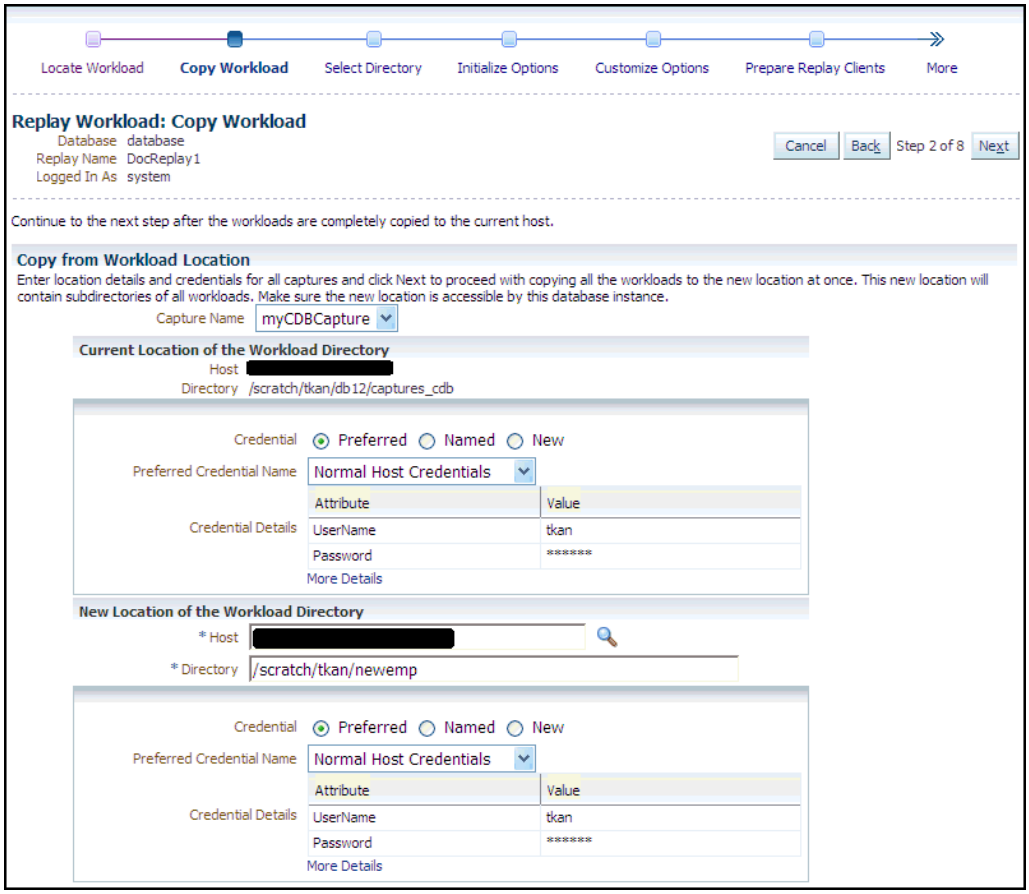
   The Replay Workload: Locate Workload page appears.



8. Select the desired workload location option.

   If you have not previously copied the workload from its storage location to the replay location where the replay clients can access it, select the option to copy the workload. Otherwise, select the option to use the existing replay directory that contains the workload to be replayed.

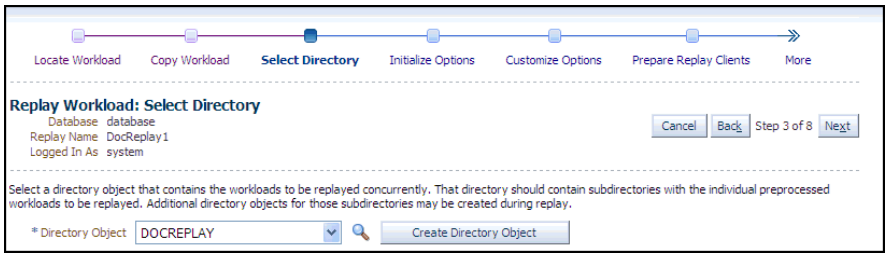
   Click **Next** to display the Replay Workload: Copy Workload page.



9. Provide the required credentials and the new location of the workload directory to which you want to copy the workload, then click **Next**.

   - There are multiple source workloads for a consolidated replay, so multiple source credentials might be needed for the current location of the workload directory. For more information on consolidated replays, see "Using Consolidated Database Replay with Enterprise Manager."

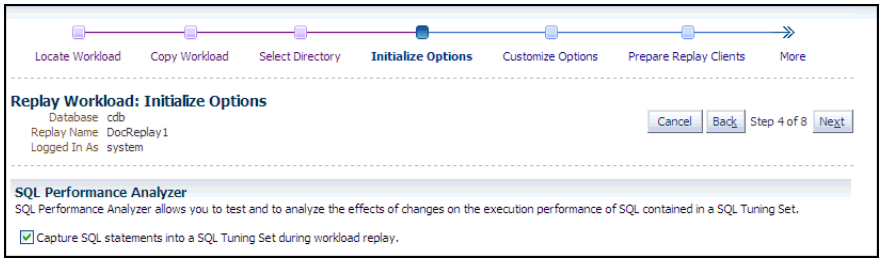
   The system responds by displaying a progress bar graph during processing, then displays the Replay Workload: Select Directory page after the copy operation concludes.



10. Specify the Directory Object, or create a new Directory Object that points to the location that contains the workload. If you chose to copy from the workload location to a new location in the previous step, make sure that the directory object points to the exact location you specified in the New Location of the Workload Directory section.

    The system responds by displaying a Capture Summary. You can expand the Capture Details section to see the workload profile and workload filters. You can also generate a Workload Capture Analyzer Report and Database Capture Report. The Capture Summary does not appear for consolidated replays.

Click **Next** to display the Replay Workload: Initialize Options page.



11. In the SQL Performance Analyzer section, retain or disable the Capture SQL Statements option, which is enabled by default and recommended. You can disable this option if you do not want to compare SQL tuning sets at the end of the replay.

    - The SQL Performance Analyzer can initiate an impact analysis of environmental changes on the performance of SQL statements within a SQL Tuning Set. You can create and analyze SQL Performance Analyzer tasks to test the effects of a database upgrade, initialization parameter change, Exadata simulation, or custom experiments. A task compares the effects of before-trial changes with after-trial changes.

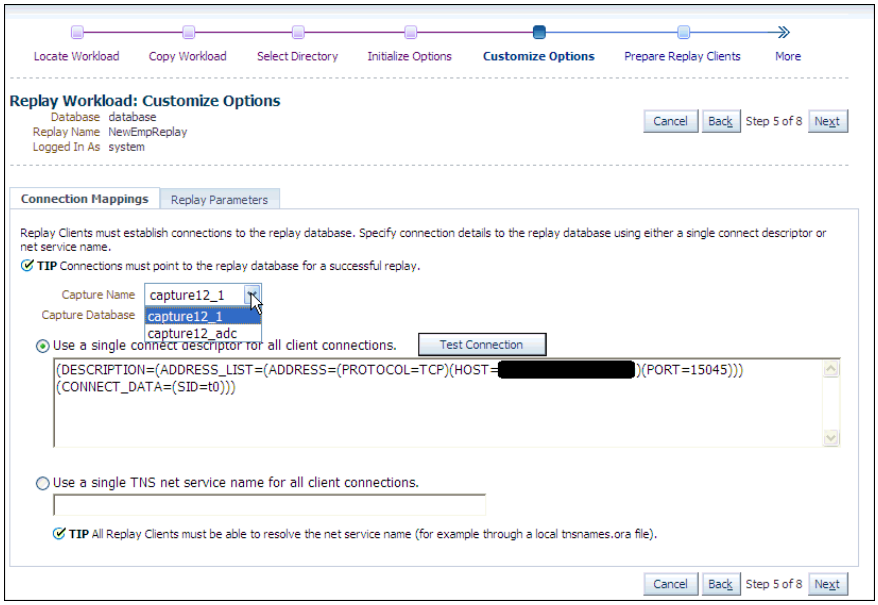
      Although Database Replay provides an analysis of how a change affects your entire system, you can use a SQL tuning set in conjunction with the SQL Performance Analyzer to gain a more SQL-centric analysis of how the change affects SQL statements and execution plans.

      By capturing a SQL tuning set during workload replay, you can use SQL Performance Analyzer to compare this SQL tuning set to another SQL tuning set captured during workload capture, without having to re-execute the SQL statements. This enables you to obtain a SQL Performance Analyzer report and compare the SQL performance, before and after change, while running Database Replay.

    - In the Identify Source section, initial replay options refer to the connection mappings and parameters on the Customize Options page. Connections are captured along with the workload.

      > **Note:**
      >
      > This section does not appear for consolidated replays or Oracle RAC.

Click **Next** to display the Replay Workload: Customize Options page.



12. Remap captured connection strings to connection strings that point to the replay system. Note that you need to remap each capture connection. For example, in the illustration above, you would need to remap the connection for both capture12_1 and capture12_adc.

    (You can remap connections per workload for consolidated replay. There is a Capture Name drop-down to choose the workload.)

    Click the **Connection Mappings** tab. There are several methods you can use to remap captured connection strings. You can choose to:

    - **Use a single connect descriptor for all client connections** by selecting this option and entering the connect descriptor you want to use. The connect descriptor should point to the replay system.

      To test the connection, click **Test Connection**. If the connect descriptor is valid, an Information message is displayed to inform you that the connection was successful.
    - **Use a single TNS net service name for all client connections** by selecting this option and entering the net service name you want to use. All replay clients must be able to resolve the net service name, which can be done using a local `tnsnames.ora` file.
    - **Use a separate connect descriptor or net service name for each client connect descriptor captured in the workload** by selecting this option and, for each capture system value, entering a corresponding replay system value that the replay client will be use. If you selected the "Use replay options from a previous replay" option in the Initialize Options step, the "Use a separate connect descriptor" option is selected, and the previous replay system values appear in the text field below.

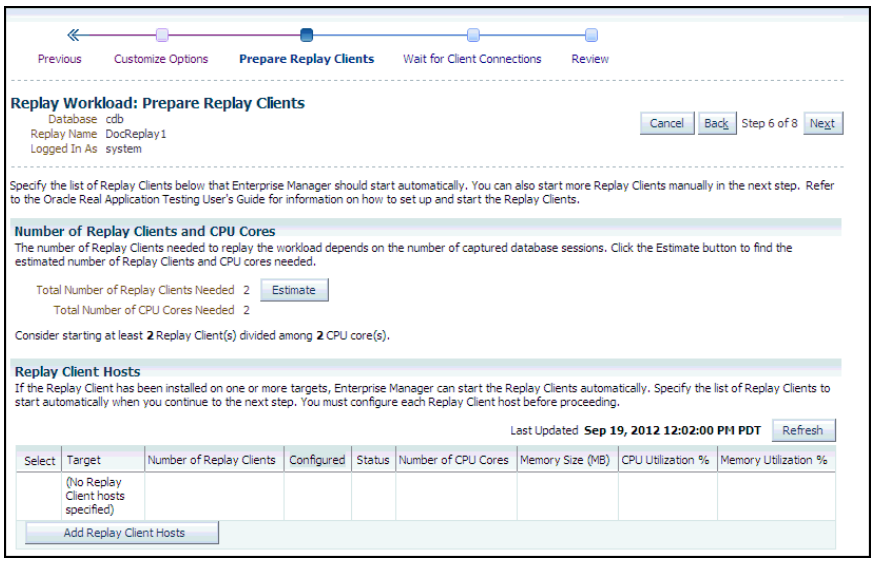
    > **Note:**
    >
    > This option does not apply to consolidated replays.

13. Specify the replay options using the replay parameters, which control some aspects of the replay.

To modify the replay behavior, click the **Replay Parameters** tab and enter the desired values for each replay parameter. Using the default values is recommended. For information about setting the replay parameters, see .

After setting the replay parameters, click **Next**.

The Replay Workload: Prepare Replay Clients page appears.



14. Ensure that the replay clients are prepared for replay:

    Before proceeding, the replay clients must be set up.

    a. Click **Estimate** to determine how many replay clients and CPUs are required for the replay.

    b. Click **Add Replay Client Hosts** to add a host or hosts for the replay clients. (If you do not want to add any replay client hosts, you can still continue and start the replay clients from the command line outside of Enterprise Manager).

       The Search and Select: Replay Client Host pop-up appears.

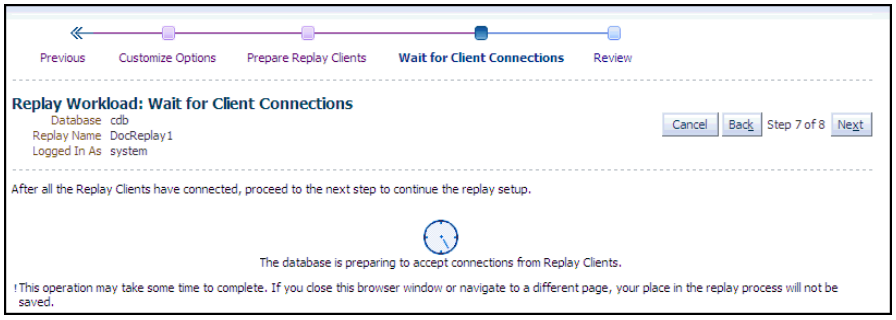
       - Specify a Target Name and then click **Go**, or just click **Go** to display the entire list of available hosts.
       - Choose a host and then click **Select**.

    c. When the host name appears in the Target column of the Replay Client Hosts table, specify the number of replay clients recommended from the estimate results, then make sure that the number of CPUs listed for the host satisfies the minimum recommendation in the estimate results. You should start at least one replay client per captured workload.

    d. In the Configured column, click the **No** link to configure the replay client host in the pop-up that appears.

    e. Click **Apply** after you have finished providing input in the pop-up. The Configured column now displays Yes in the Replay Client Hosts table.

15. Click **Next** to start the replay clients and display the Replay Workload: Wait for Client Connections page.



> **Note:**
>
> If you have reached this step in the process from the Enterprise menu, you also need to enter credentials for the replay job and for the replay result storage host.

- As replay clients are started, the replay client connections are displayed in the Client Connections table.
- The text below the clock changes if a replay client is connected.
- The Client Connections table is populated when at least one replay client is connected.

When all replay clients have connected, enter host credentials at the bottom of the page to start the replay job, then click **Next** to display the Replay Workload: Review page.

16. Review the options and parameters that have been defined for the workload replay.
    - The value for Connected Replay Clients must be at least 1 in order to successfully submit the Replay Workload job.
    - The Submit button is enabled only if at least one replay client is connected.
17. If everything appears as you have intended, click **Submit** to submit the replay job.

    After the replay starts, the Home tab of the Database Replay page for this replay reappears with a system message that states "The workload replay has started."

> **See Also:**
>
> - "Connection Remapping"
> - "Specifying Replay Options"
> - "Setting Up Replay Clients"
> - "Monitoring an Active Workload Replay"

Setting Up the Replay Schedule and Parameters Using Enterprise Manager

The replay schedule feature enables you to scale up the instances of captured workloads to be included in a consolidated replay, and then control the relative playback scheduling of the

instances within the replay. The relative scheduling of the captured workload instances is visually displayed by updating the alignment of the active session chart for each instance. This feature is available for Cloud Control Database plug-in 12.1.0.5 and later releases.

This feature enables you to accomplish the following tasks:

- **Offset instances so that they execute at different time intervals**

  You can adjust the relative replay start time of each workload instance to align the average active sessions peak times for scheduled instances of Capture workloads. This alignment of the workload peaks potentially maximizes the load on the system, so that you can experiment with how the test system responds under different workload conditions.

- **Scale up the replay workload by adding instances of the captured workload**

  Each added instance is replayed independently of the other instances.

  When you scale up a workload by specifying multiple instances of it, the default and recommended configuration is to replay the DML statements in one of the instances. All of the additional instances will only replay the query (ready-only) statements.

  For example, if a workload has a SQL Insert to an employee database, you normally would want to have only one instance that executes the Insert, and the others would bypass user calls that modify the database with this Insert. However, you can override the default setting of an instance by unchecking the Replay Query-only check box to replay all statements in the workload.

To access the Plan Replay Schedule page:

1. From the Database Replay home page, click on the **Replay Tasks** tab.
2. Click on the name of an existing replay task with more than one workload to navigate to the Replay Task page.
3. Click **Create** to create a new replay.
4. Provide the requisite information in the Create Replay pop-up, then click **OK**.

   The new replay appears in the Replays table.
5. Click the name of your new replay in the Replays table.

   A Task List now appears in the Replay page.
6. Click the **Plan Replay Schedule** link.

   The Plan Replay Schedule page appears.

To scale up a replay:

1. In the drop-down next to the Add Workload Instance button, select the Capture workload for which you want to add an instance.
2. Click **Add Workload Instance** to add the instance.

To schedule the time intervals:

1. From the Replay Delay column, either leave the default value of 0 for the first capture instance, or adjust the number of minutes desired before the instance starts to execute.
2. Repeat the step above for each capture instance until you have set all of the values to represent how you want the performance spikes to execute for your testing needs.

To auto-align workload peaks:

1. Click the **Auto-align** button.

To specify query-only instances:

1. For each instance that you want to be query-only, enable the **Replay Query-only** check box.

To review the updated schedule:

1. From the Replay Task tab of the Database Replay page, click the Replay task link containing your scheduled replay.

   The Replay Task page appears.
2. Click the scheduled replay in the Replays table.
3. Select the Review tab in the Replay page.

Monitoring Workload Replay Using Enterprise Manager

This section describes how to monitor workload replay using Enterprise Manager. The primary tool for monitoring workload replay is Oracle Enterprise Manager. Using Enterprise Manager, you can:

- Monitor or stop an active workload replay
- View a completed workload replay

If Oracle Enterprise Manager is unavailable, you can monitor workload replay using APIs and views, as described in "Monitoring Workload Replay Using APIs".

This section contains the following topics:

Monitoring an Active Workload Replay

This section describes how to monitor an active workload replay using Enterprise Manager.

To monitor an active workload replay:

1. From the Database Replay page, click the **Replay Tasks** tab.
2. Click the name of the replay task that contains the replay for which you want to monitor replay progress.
3. From the Replays section of the Replay Task page, click the name of the replay you have submitted for processing in the Create Replay wizard. (The Status column for this replay should show In Progress.)

   The Home tab of the Database Replay page appears, and the Replay Summary shows a Status of Running.

   - The replay line in the Replay Progress chart updates dynamically while the replay is in progress. You can update the chart by clicking the Refresh button.
   - The user calls line in the Replay Progress chart indicates how much of the workload the database has serviced relative to the capture at the same time.
   - Data for the Replay Divergence Summary is not available until the replay completes.

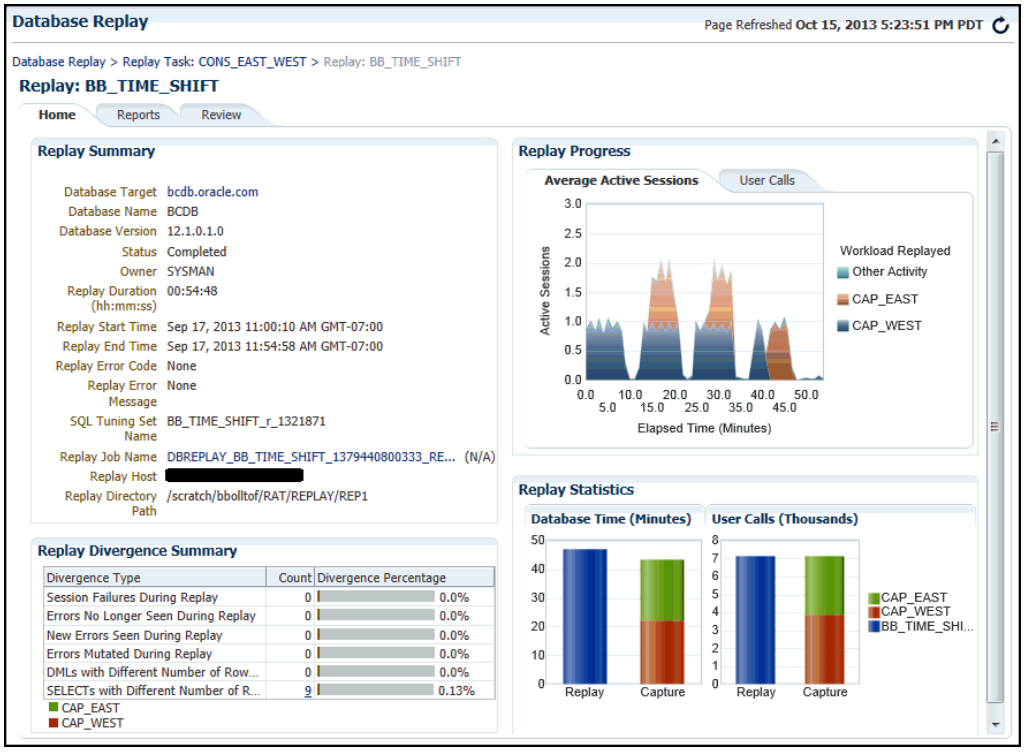
Viewing a Completed Workload Replay

This section describes how to view a completed workload replay using Enterprise Manager.

To view a completed workload replay:

1. From the Database Replay page, click the **Replay Tasks** tab.
2. Click the name of the replay task that contains the completed replay you want to view.
3. From the Replays section of the Replay Task page, click the name of the replay you have submitted for processing in the Create Replay wizard. (The Status column for this replay should show Completed.)

   The Home tab of the Database Replay page appears, and the Replay Summary shows a Status of Completed.



- The replay line in the User Calls chart graphically represents the replay progress during the course of the entire replay from initiation to conclusion.

  The chart shows how much time it has taken to replay the same workload compared to the elapsed time during the workload capture in terms of user calls. If the Replay line is above or to the left of the Capture line, the replay system is processing the workload faster than the capture system.
- Under the Replay Divergence Summary, any errors and data discrepancies between the replay system and the capture system are displayed as diverged database calls during replay. You can use the percentage of total calls that diverged as a measure of the replay quality.

  To view details about the diverged calls, click the link that corresponds to the type of diverged call in the Count column to access the Diverged Calls During Replay page.

The Diverged Calls During Replay page shows the most relevant set of replayed calls that diverged from the workload captured by grouping them based on common attribute values and specified filter conditions. To view details about a particular diverged call—such as the call attributes, SQL text, and bind variables—click the corresponding link in the SQL ID column to bring up the Replay Diverged Statement page.

4. To return to the Database Replay page, click the Database Replay breadcrumb.

> **See Also:**
>
> "Analyzing Captured and Replayed Workloads " for information about accessing workload replay reports

Importing a Replay External to Enterprise Manager

As with importing workloads external to Enterprise Manager, you can import replays into Enterprise Manager to manage them. To import a replay, you import it from a replay task, which can contain one or more workloads and one or more replays. The replay task is at the top of the database replay hierarchy, and serves as the container for these other subordinate components.
A replay to be imported can be running in the test database, or the replay can be completed and the replay directory stored on a file system.

This feature is available for Cloud Control Database plug-in 12.1.0.5 and later releases.

To import a replay external to Enterprise Manager:

1. From the Database Replay page, click the **Replay Tasks** tab, then select the desired replay task.

   The Replay Task page appears.

2. Click **Import** in the Replays section.

   The Import Replay: Source page appears

3. Select one of the three choices available to import a replay, then click **Next**.

   - **Import one or more completed Replays from a directory in the file system**

     This option typically applies for a replay created using an API, in which you now want to import it to Enterprise Manager for subsequent processing. In this case, Enterprise Manager may not be necessarily managing the replay database.

   - **Import one or more completed Replays from a database target**

     In this case, Enterprise Manager is probably already managing the replay database. The replay could have been done on this database, or it could have been loaded in as would be the case for the option above.

   - **Attach to a Replay of this Replay task running in a database target**

     This option is similar to the option above, except this replay is still in progress, rather than one that has already completed.

   The Import Replay: Database page appears.

4. Click **the search icon next to the Database Target field and select a database from the pop-up that appears.**

> **Note:**
>
> The target version of the database reading the replay must be at least as high as the one you used in the replay task. For instance, if the replay task used Oracle Database 12x, the database you select to read the replay must be at least version 12x.

- The system now requests database and host credentials.
- In the previous step, if you chose to import one or more completed replays from a directory in the file system, the system also requires a workload location.
- The replay task can determine if this is a consolidated replay based on the number of workloads contained in the replay task. If this is a consolidated replay, this step asks you to enter a consolidated replay directory for the workload location.

5. Provide the requisite input for the step above, then click **Next**.

   The Import Replay: Replay page appears.

   - If you chose "Import one or more completed Replays from a directory in the file system" in step 3, this page provides a Load Replay button.
   - If you chose “Import one or more completed Replays from a database target" or “Attach to a Replay of this Replay task running in a database target" in step 3, this page provides a Discover Replay button.

6. Click either **Load Replay** or **Discover Replay**, depending on which button is available in accordance with your selection in step 3.

   The system displays one or more replays, if found, in the Discovered Replays table.

7. Either click **Next** to load the replay, or select one or more replays, then click **Next** to continue importing the replays.

   The Import Replay: Review page appears.

8. If everything appears as you have intended, click **Submit**.

   The Database Replay page reappears and displays a message stating that the job was submitted successfully. The Status column in the table for your imported replay will show In Progress.

> **Tip:**
>
> You can check on the job's progress by clicking on the replay name that you just submitted in the Review step. A Replay Summary page appears, and you can click the Database Replay Import Job link to see the progress of the job execution steps.

Replaying a Database Workload Using APIs

This section describes how to replay a database workload using the `DBMS_WORKLOAD_REPLAY` package. You can also use Oracle Enterprise Manager to replay a database workload, as described in "Replaying a Database Workload Using Enterprise Manager".
Replaying a database workload using the `DBMS_WORKLOAD_REPLAY` package is a multi-step process that involves:

- Initializing Replay Data
- Remapping Connections
- Remapping Users
- Setting Workload Replay Options
- Defining Workload Replay Filters and Replay Filter Sets
- Setting the Replay Timeout Action
- Starting a Workload Replay
- Pausing a Workload Replay
- Resuming a Workload Replay
- Cancelling a Workload Replay
- Retrieving Information About Workload Replays
- Loading Divergence Data for Workload Replay
- Deleting Information About Workload Replays
- Exporting AWR Data for Workload Replay
- Importing AWR Data for Workload Replay

See Also:

- *Oracle Database PL/SQL Packages and Types Reference* for information about the `DBMS_WORKLOAD_REPLAY` package

Initializing Replay Data

After the workload capture is preprocessed and the test system is properly prepared, the replay data can be initialized. Initializing replay data loads the necessary metadata into tables required by workload replay. For example, captured connection strings are loaded into a table where they can be remapped for replay.

To initialize replay data:

- Use the `INITIALIZE_REPLAY` procedure:

```
BEGIN
  DBMS_WORKLOAD_REPLAY.INITIALIZE_REPLAY (replay_name => 'dec06_102',
                          replay_dir => 'dec06',
                          plsql_mode => 'top_level');
END;
/
```

  In this example, the `INITIALIZE_REPLAY` procedure loads preprocessed workload data from the `dec06` directory into the database.

  The `INITIALIZE_REPLAY` procedure in this example uses the following parameters:

  – The `replay_name` required parameter specifies a replay name that can be used with other APIs to retrieve settings and filters of previous replays.

- The `replay_dir` required parameter specifies the directory that contains the workload capture that will be replayed.
- The optional `plsql_mode` parameter specifies the PL/SQL replay mode.

  These two values can be set for the `plsql_mode` parameter:

  * `top_level`: Only top-level PL/SQL calls. This is the default value.
  * `extended`: SQL executed inside PL/SQL or top-level PL/SQL if there is no SQL recorded inside. Non-PL/SQL calls will be replayed in the usual manner.

> **Note:**
>
> To run `INITIALIZE_REPLAY` on an encrypted workload capture, you need to set the password using the `oracle.rat.database_replay.encryption` (case-sensitive) identifier. The password is stored in a software keystore. You can find whether a workload capture is encrypted or not from `DBA_WORKLOAD_CAPTURES` view.

> **See Also:**
>
> - "Preprocessing a Database Workload Using APIs" for information about preprocessing a workload capture
> - "Steps for Replaying a Database Workload" for information preparing the test system

Remapping Connections

After the replay data is initialized, connection strings used in the workload capture need to be remapped so that user sessions can connect to the appropriate databases and perform external interactions as captured during replay. To view connection mappings, use the `DBA_WORKLOAD_CONNECTION_MAP` view.

To remap connections:

- Use the `REMAP_CONNECTION` procedure:

```
BEGIN
  DBMS_WORKLOAD_REPLAY.REMAP_CONNECTION (connection_id => 101,
                          replay_connection => 'dlsun244:3434/bjava21');
END;
/
```

  In this example, the connection that corresponds to the connection ID 101 will use the new connection string defined by the `replay_connection` parameter.

  The `REMAP_CONNECTION` procedure in this example uses the following parameters:

  - The `connection_id` required parameter is generated when initializing replay data and corresponds to a connection from the workload capture.
  - The `replay_connection` required parameter specifies the new connection string that will be used during workload replay.

> **See Also:**
>
> "Connection Remapping"

Remapping Users

Aside from remapping connection strings, you can also use a new schema or user instead of the user captured in the workload capture. To view captured users, use the `DBA_WORKLOAD_USER_MAP` view.

To remap users:

- Use the `SET_USER_MAPPING` procedure:

```
BEGIN
  DBMS_WORKLOAD_REPLAY.SET_USER_MAPPING (capture_user => 'PROD',
                          replay_user => 'TEST');
END;
/
```

  In this example, the `PROD` user used during capture is remapped to the `TEST` user during replay.

  The `SET_USER_MAPPING` procedure in this example uses the following parameters:

  – The `capture_user` required parameter specifies the username captured during the time of the workload capture.

  – The `replay_user` required parameter specifies the username to which the captured user is remapped during replay. If this parameter is set to `NULL`, then the mapping is disabled.

> **See Also:**
>
> "User Remapping"

Setting Workload Replay Options

After the replay data is initialized, and connections and users are remapped, you need to prepare the database for workload replay.

To prepare workload replay on the replay system:

- Use the `PREPARE_REPLAY` procedure:

```
BEGIN
  DBMS_WORKLOAD_REPLAY.PREPARE_REPLAY (synchronization => 'OBJECT_ID',
                          capture_sts => TRUE,
                          sts_cap_interval => 300);
END;
/
```

  In this example, the `PREPARE_REPLAY` procedure prepares a replay that has been previously initialized. A SQL tuning set will also be captured in parallel with the workload replay.

The PREPARE_REPLAY procedure uses the following parameters:

- The synchronization required parameter controls the type of synchronization used during workload replay.

  If the parameter is set to TIME, the replay will use the same wall-clock timing as the capture. All database session login times will be replayed exactly as the capture. Likewise, all timing between transactions within database sessions will be preserved and replayed as captured. This synchronization method will produce good replays for most workloads.

  If this parameter is set to SCN (the default value), the COMMIT order in the captured workload will be observed during replay and all replay actions will be executed only after all dependent COMMIT actions have completed. This synchronization method may introduce significant delays for some workloads. If this is the case, it is recommended to use TIME as the synchronization parameter.

  If this parameter is set to OBJECT_ID, all replay actions will be executed only after all relevant COMMIT actions have completed. Relevant COMMIT actions must meet the following criteria:

  * Issued before the given action in the workload capture
  * Modified at least one of the database objects for which the given action is referencing, either implicitly or explicitly

  Setting this parameter to OBJECT_ID can allow for more concurrency during workload replays for COMMIT actions that do not reference the same database objects during workload capture.

- The connect_time_scale parameter scales the elapsed time from when the workload capture started to when the session connects with the specified value and is interpreted as a % value. Use this parameter to increase or decrease the number of concurrent users during replay. The default value is 100.
- The think_time_scale parameter scales the elapsed time between two successive user calls from the same session and is interpreted as a % value. Setting this parameter to 0 will send user calls to the database as fast as possible during replay. The default value is 100.
- The think_time_auto_correct parameter corrects the think time (based on the think_time_scale parameter) between calls when user calls take longer to complete during replay than during capture. This parameter can be set to either TRUE or FALSE. Setting this parameter to TRUE reduces the think time if the workload replay is taking longer than the workload capture. The default value is TRUE.
- The scale_up_multiplier parameter defines the number of times the workload is scaled up during replay. Each captured session will be replayed concurrently for as many times as specified by this parameter. However, only one session in each set of identical replay sessions will execute both queries and updates. The rest of the sessions will only execute queries.
- The capture_sts parameter specifies whether to capture a SQL tuning set in parallel with the workload replay. If this parameter is set to TRUE, you can capture a SQL tuning set during workload replay and use SQL Performance Analyzer to compare it to another SQL tuning set without having to re-execute the SQL statements. This enables you to obtain a SQL Performance Analyzer report and compare the SQL performance —before and after the change—while running Database Replay. You can also export the resulting SQL tuning set with its AWR data using the EXPORT_AWR procedure, as described in "Exporting AWR Data for Workload Replay".

This feature is not supported for Oracle RAC. Workload replay filters that are defined using `DBMS_WORKLOAD_REPLAY` do not apply to the SQL tuning set capture. The default value for this parameter is `FALSE`.

– The `sts_cap_interval` parameter specifies the duration of the SQL tuning set capture from the cursor cache in seconds. The default value is 300. Setting the value of this parameter below the default value may cause additional overhead with some workloads and is not recommended.

For more information about setting these parameters, see "Specifying Replay Options".

> **See Also:**
>
> "Steps for Replaying a Database Workload"

Defining Workload Replay Filters and Replay Filter Sets

This section describes how to add and remove workload replay filters, and how to create and use replay filter sets.
This section contains the following topics:

- Adding Workload Replay Filters
- Deleting Workload Replay Filters
- Creating a Replay Filter Set
- Using a Replay Filter Set

> **See Also:**
>
> "Using Filters with Workload Replay"

Adding Workload Replay Filters

This section describes how to add a new filter to be used in a replay filter set.

To add a new filter:

- Use the `ADD_FILTER` procedure:

```
BEGIN
  DBMS_WORKLOAD_REPLAY.ADD_FILTER (
                           fname => 'user_ichan',
                           fattribute => 'USER',
                           fvalue => 'ICHAN');
END;
/
```

In this example, the `ADD_FILTER` procedure adds a filter named user_ichan, which can be used to filter out all sessions belonging to the user name ICHAN.

The `ADD_FILTER` procedure in this example uses the following parameters:

– The `fname` required parameter specifies the name of the filter that will be added.

- The `fattribute` required parameter specifies the attribute on which the filter will be applied. Valid values include PROGRAM, MODULE, ACTION, SERVICE, USER, and CONNECTION_STRING. You must specify a valid captured connection string that will be used during replay as the CONNECTION_STRING attribute.
- The `fvalue` required parameter specifies the value for the corresponding attribute on which the filter will be applied. It is possible to use wildcards such as % with some of the attributes, such as modules and actions.

Once all workload replay filters are added, you can create a replay filter set that can be used when replaying the workload.

Deleting Workload Replay Filters

This section describes how to delete workload replay filters.

To delete workload replay filters:

- Use the `DELETE_FILTER` procedure:

```
BEGIN
  DBMS_WORKLOAD_REPLAY.DELETE_FILTER (fname => 'user_ichan');
END;
/
```

In this example, the `DELETE_FILTER` procedure removes the filter named user_ichan.

The `DELETE_FILTER` procedure uses the `fname` required parameter, which specifies the name of the filter to be removed.

Creating a Replay Filter Set

After the workload replay filters are added, you can create a set of replay filters to use with workload replay. When creating a replay filter set, all workload replay filters that were added since the previous replay filter set was created will be used.

To create a replay filter set:

- Use the `CREATE_FILTER_SET` procedure:

```
BEGIN
  DBMS_WORKLOAD_REPLAY.CREATE_FILTER_SET (
                            replay_dir => 'apr09',
                            filter_set => 'replayfilters',
                            default_action => 'INCLUDE');
END;
/
```

In this example, the `CREATE_FILTER_SET` procedure creates a replay filter set named replayfilters, which will replay all captured calls for the replay stored in the `apr09` directory, except for the part of the workload defined by the replay filters.

The `CREATE_FILTER_SET` procedure in this example uses the following parameters:

- The `replay_dir` parameter specifies the directory where the replay to be filtered is stored
- The `filter_set` parameter specifies the name of the filter set to create

- The `default_action` parameter determines if every captured database call should be replayed and whether the workload replay filters should be considered as inclusion or exclusion filters.

  If this parameter is set to `INCLUDE`, all captured database calls will be replayed, except for the part of the workload defined by the replay filters. In this case, all replay filters will be treated as exclusion filters, since they will define the part of the workload that will not be replayed. This is the default behavior.

  If this parameter is set to `EXCLUDE`, none of the captured database calls will be replayed, except for the part of the workload defined by the replay filters. In this case, all replay filters will be treated as inclusion filters, since they will define the part of the workload that will be replayed.

Using a Replay Filter Set

Once the replay filter set is created and the replay is initialized, you can use the replay filter set to filter the replay in the `replay_dir` directory.

To use a replay filter set:

- Use the `USE_FILTER_SET` procedure:

```
BEGIN
  DBMS_WORKLOAD_REPLAY.USE_FILTER_SET (filter_set => 'replayfilters');
END;
/
```

  In this example, the `USE_FILTER_SET` procedure uses the filter set named replayfilters.

  The `USE_FILTER_SET` procedure in this example uses the `filter_set` required parameter, which specifies the name of the filter set to be used in the replay.

Setting the Replay Timeout Action

This section describes how to set a timeout action for the workload replay. You can set a replay timeout action to abort user calls that are significantly slower during replay or cause a replay to hang. For example, you may want to set a replay timeout action to abort runaway queries caused by sub-optimal execution plans following a database upgrade.
When a replay timeout action is enabled, a user call will exit with an `ORA-15569` error if it is delayed beyond the conditions specified by the replay timeout action. The aborted call and its error are reported as error divergence.

To set a replay timeout:

- Use the `SET_REPLAY_TIMEOUT` procedure:

```
BEGIN
  DBMS_WORKLOAD_REPLAY.SET_REPLAY_TIMEOUT (
                          enabled => TRUE,
                          min_delay => 20,
                          max_delay => 60,
                          delay_factor => 10);
END;
/
```

  In this example, the `SET_REPLAY_TIMEOUT` procedure defines a replay timeout action that will abort a user call if the delay during replay is more than 60 minutes, or if the delay

during replay is over 20 minutes and the elapsed time is 10 times greater than the capture elapsed time.

The `SET_REPLAY_TIMEOUT` procedure in this example uses the following parameters:

- The `enabled` parameter specifies if the replay timeout action is enabled or disabled. The default value is `TRUE`.
- The `min_delay` parameter defines the lower bound value of call delay in minutes. The replay timeout action is only activated when the delay is over this value. The default value is 10.
- The `max_delay` parameter defines the upper bound value of call delay in minutes. The replay timeout action is activated and issues an error when the delay is over this value. The default value is 120.
- The `delay_factor` parameter defines a factor for the call delays that are between the values of `min_delay` and `max_delay`. The replay timeout action issues an error when the current replay elapsed time is higher than the multiplication of the capture elapsed time and this value. The default value is 8.

To retrieve the replay timeout action setting:

- Use the `GET_REPLAY_TIMEOUT` procedure:

```
DECLARE
  enabled        BOOLEAN;
  min_delay      NUMBER;
  max_delay      NUMBER;
  delay_factor   NUMBER;
BEGIN
  DBMS_WORKLOAD_REPLAY.GET_REPLAY_TIMEOUT(enabled, min_delay, max_delay,
                          delay_factor);
END;
/
```

The `GET_REPLAY_TIMEOUT` procedure in this example returns the following parameters:

- The `enabled` parameter returns whether the replay timeout action is enabled or disabled.
- The `min_delay` parameter returns the lower bound value of call delay in minutes.
- The `max_delay` parameter returns the upper bound value of call delay in minutes.
- The `delay_factor` parameter returns the delay factor.

Starting a Workload Replay

There are tasks to perform before starting a workload replay. For example:

- Preprocess the captured workload, as described in "Preprocessing a Database Workload Using APIs"
- Initialize the replay data, as described in "Initializing Replay Data"
- Specify the replay options, as described in "Setting Workload Replay Options"
- Start the replay clients, as described in "Starting Replay Clients"

> **Note:**
>
> Once a workload replay is started, new replay clients will not be able to connect to the database. Only replay clients that were started before the `START_REPLAY` procedure is executed will be used to replay the captured workload.

To start a workload replay:

- Use the `START_REPLAY` procedure:

```
BEGIN
  DBMS_WORKLOAD_REPLAY.START_REPLAY ();
END;
/
```

> **Note:**
>
> Starting with Oracle Database Release 19c, the `DBA_RAT_CAPTURE_SCHEMA_INFO` view provides `login_schema` and `current_schema` information for SQL statements. During a replay in the extended mode, if you encounter the `ORA-00942: table or view does not exist` error, then use the `DBA_RAT_CAPTURE_SCHEMA_INFO` and `DBA_WORKLOAD_CAPTURE_SQLTEXT` views to find the tables on which the error happened. Then grant the necessary privileges on the tables to the users who hit the error and retry the replay; this may fix the problem.

Pausing a Workload Replay

Pausing a workload replay will halt all subsequent user calls issued by the replay clients until the workload replay is either resumed or cancelled. User calls that are already in progress will be allowed to complete. This option enables you to temporarily stop the replay to perform a change and observe its impact for the remainder of the replay.

To pause a workload replay:

- Use the `PAUSE_REPLAY` procedure:

```
BEGIN
  DBMS_WORKLOAD_REPLAY.PAUSE_REPLAY ();
END;
/
```

Resuming a Workload Replay

This section describes how to resume a workload replay that is paused.

To resume a workload replay:

- Use the `RESUME_REPLAY` procedure:

```
BEGIN
  DBMS_WORKLOAD_REPLAY.RESUME_REPLAY ();
END;
/
```

Cancelling a Workload Replay

This section describes how to cancel a workload replay.

To cancel a workload replay:

- Use the `CANCEL_REPLAY` procedure:

```
BEGIN
  DBMS_WORKLOAD_REPLAY.CANCEL_REPLAY ();
END;
/
```

Retrieving Information About Workload Replays

You can retrieve all information about the workload captures in a replay directory object and the history of the workload replay attempts from the directory. By default, the workload replay divergence data is not loaded, but you can selectively choose to load this data.

To retrieve information about workload replays:

- Call the `DBMS_WORKLOAD_REPLAY.GET_REPLAY_INFO` function.

  The `GET_REPLAY_INFO` function first imports a row into the `DBA_WORKLOAD_CAPTURES` view, which contains information about the workload capture. By default, it then only imports information for replays that have not been previously loaded into the `DBA_WORKLOAD_REPLAYS` view. This function returns the `cap_id` of the capture directory (for a consolidated capture directory, the `cap_id` returned is `0`), which can be associated with the `CAPTURE_ID` column in the `DBA_WORKLOAD_REPLAYS` view to access the information retrieved.

  The `GET_REPLAY_INFO` function uses the following parameters:

  - The `replay_dir` required parameter, which specifies the name of the workload replay directory object.
  - The `load_divergence` optional parameter, which specifies if divergence data is loaded. The default value for this parameter is `FALSE`. To load divergence data, which imports rows for every replay attempt retrieved from the replay directory into the `DBA_WORKLOAD_REPLAY_DIVERGENCE` view, set this parameter to `TRUE`. Alternatively, you can use the `LOAD_DIVERGENCE` procedure to selectively load divergence data for a single replay or all replays in a directory object after the replay information is retrieved, as described in "Loading Divergence Data for Workload Replay".

Example 12-1 Retrieving information about workload replay

The following example shows how to retrieve information about the workload captures and the history of the workload replay attempts for the replay directory object named `jul14`, and to validate that the information is retrieved.

```
DECLARE
  cap_id         NUMBER;
BEGIN
  cap_id := DBMS_WORKLOAD_REPLAY.GET_REPLAY_INFO(replay_dir => 'jul14');
  SELECT capture_id
    FROM dba_workload_replays
   WHERE capture_id = cap_id;
END;
/
```

Loading Divergence Data for Workload Replay

Loading divergence data for workload replay imports rows for every replay attempt retrieved from the replay directory into the `DBA_WORKLOAD_REPLAY_DIVERGENCE` view, which displays information about diverged calls and errors during replay attempts. You can choose to load divergence data for either a single workload replay or all workload replays in a given directory object.

To load divergence data for workload replay:

1. Call the `WORKLOAD_REPLAY.LOAD_DIVERGENCE` procedure using one of the following parameters:
   - The `replay_id` parameter specifies the ID of the workload replay for which you want to load divergence data. Use this parameter if you only want to load divergence data for a single workload replay.
   - The `replay_dir` parameter specifies the name of the directory object (the value is case-sensitive). Use this parameter if you want to load divergence data for all workload replays in a given directory object.
2. To check the loading status of divergence data, query the `DIVERGENCE_LOAD_STATUS` column in the `DBA_WORKLOAD_REPLAYS` view.

   A value of `TRUE` indicates that the divergence data is loaded, and a value of `FALSE` indicates that it has not been loaded.

Example 12-2 Loading divergence data for a single workload replay

The following example shows how to load divergence data for the workload replay with a `replay_id` value of `12`, and to validate that the divergence data is loaded.

```
DECLARE
  rep_id         NUMBER;
BEGIN
  rep_id := DBMS_WORKLOAD_REPLAY.LOAD_DIVERGENCE (replay_id => 12);
  SELECT divergence_load_status
    FROM dba_workload_replays
   WHERE capture_id = rep_id;
END;
/
```

Deleting Information About Workload Replays

You can delete information retrieved for either a single workload replay or all workload replays in a replay directory. Deleted information can be retrieved using the `GET_REPLAY_INFO` function, as described in "Retrieving Information About Workload Replays".

To delete information about workload replays:

1. Call the `DBMS_WORKLOAD_REPLAY.DELETE_REPLAY_INFO` procedure using the `replay_id` parameter.
   - The `replay_id` parameter specifies the ID of the workload replay for which you want to delete replay information. Use this parameter if you only want to delete information for a single workload replay.
2. By default, deleting information about workload replays does not remove the data from disk.

Example 12-3 Deleting information about workload replay

The following example deletes information retrieved about the workload captures and the history of the workload replay attempts for the workload replay with an ID of `2`. The replay data is not deleted from disk, and thus can be retrieved by calling the `GET_REPLAY_INFO` function, as described in "Retrieving Information About Workload Replays".

```
BEGIN
  DBMS_WORKLOAD_REPLAY.DELETE_REPLAY_INFO (replay_id => 2);
END;
/
```

Exporting AWR Data for Workload Replay

Exporting AWR data enables detailed analysis of the workload. This data is also required if you plan to run the AWR Compare Period report on a pair of workload captures or replays.

To export AWR data:

- Use the `EXPORT_AWR` procedure:

  ```
  BEGIN
    DBMS_WORKLOAD_REPLAY.EXPORT_AWR (replay_id => 1);
  END;
  /
  ```

  In this example, the AWR snapshots that correspond to the workload replay with a replay ID of 1 are exported, along with any SQL tuning set that may have been captured during workload replay.

  The `EXPORT_AWR` procedure uses the `replay_id` required parameter, which specifies the ID of the replay whose AWR snapshots will be exported.

> **Note:**
>
> This procedure works only if the corresponding workload replay was performed in the current database and the AWR snapshots that correspond to the original replay time period are still available.

Importing AWR Data for Workload Replay

After AWR data is exported from the replay system, you can import the AWR data into another system. Importing AWR data enables detailed analysis of the workload. This data is also required if you plan to run the AWR Compare Period report on a pair of workload captures or replays.

To import AWR data:

- Use the `IMPORT_AWR` function:

  ```
  CREATE USER capture_awr
  SELECT DBMS_WORKLOAD_REPLAY.IMPORT_AWR (replay_id => 1,
                                          staging_schema => 'capture_awr')
    FROM DUAL;
  ```

In this example, the AWR snapshots that correspond to the workload replay with a capture ID of 1 are imported using a staging schema named `capture_awr`.

The `IMPORT_AWR` procedure in this example uses the following parameters:

– The `replay_id` required parameter specifies the ID of the replay whose AWR snapshots will be import.

– The `staging_schema` required parameter specifies the name of an existing schema in the current database which will be used as a staging area while importing the AWR snapshots from the replay directory to the `SYS AWR` schema.

> **Note:**
>
> This function fails if the schema specified by the `staging_schema` parameter contains any tables with the same name as any of the AWR tables.

Monitoring Workload Replay Using APIs

This section describes how to monitor workload replay using APIs and views. You can also use Oracle Enterprise Manager to monitor workload replay, as described in "Monitoring Workload Replay Using Enterprise Manager".
This section contains the following topics:

- Retrieving Information About Diverged Calls
- Monitoring Workload Replay Using Views

Retrieving Information About Diverged Calls

During replay, any error and data discrepancies between the replay system and the capture system are recorded as diverged calls.

To retrieve information about a diverged call—including its SQL identifier, SQL text, and bind values—call the `GET_DIVERGING_STATEMENT` function using the following parameters:

- Set the `replay_id` parameter to the ID of the replay in which the call diverged
- Set the `stream_id` parameter to the stream ID of the diverged call
- Set the `call_counter` parameter to the call counter of the diverged call

To view these information about a diverged call, use the `DBA_WORKLOAD_REPLAY_DIVERGENCE` view. The following example illustrates a function call:

```
DECLARE
  r                   CLOB;
  ls_stream_id        NUMBER;
  ls_call_counter     NUMBER;
  ls_sql_cd           VARCHAR2(20);
  ls_sql_err          VARCHAR2(512);
  CURSOR c IS
  SELECT stream_id, call_counter
  FROM DBA_WORKLOAD_REPLAY_DIVERGENCE
  WHERE replay_id = 72;
BEGIN
  OPEN c;
  LOOP
```

```
  FETCH c INTO ls_stream_id, ls_call_counter;
  EXIT when c%notfound;
  DBMS_OUTPUT.PUT_LINE (ls_stream_id||''||ls_call_counter);
  r:=DBMS_WORKLOAD_REPLAY.GET_DIVERGING_STATEMENT(replay_id => 72,
  stream_id => ls_stream_id, call_counter => ls_call_counter);
  DBMS_OUTPUT.PUT_LINE (r);
  END LOOP;
END;
/
```

See Also:

- *Oracle Database PL/SQL Packages and Types Reference* for information about the `DBMS_WORKLOAD_REPLAY` package

Monitoring Workload Replay Using Views

This section summarizes the views that you can display to monitor workload replay. You need DBA privileges to access these views.

- The `DBA_WORKLOAD_CAPTURES` view lists all the workload captures that have been captured in the current database.
- The `DBA_WORKLOAD_FILTERS` view lists all workload filters for workload captures defined in the current database.
- The `DBA_WORKLOAD_REPLAYS` view lists all the workload replays that have been replayed in the current database.
- The `DBA_WORKLOAD_REPLAY_DIVERGENCE` view enables you to view information about diverged calls, such as the replay identifier, stream identifier, and call counter.
- The `DBA_WORKLOAD_DIV_SUMMARY` view displays a summary of the replay divergence information in the `DBA_WORKLOAD_REPLAY_DIVERGENCE` view.
- The `DBA_WORKLOAD_REPLAY_FILTER_SET` view lists all workload filters for workload replays defined in the current database.
- The `DBA_WORKLOAD_CONNECTION_MAP` view lists the connection mapping information for workload replay.
- The `V$WORKLOAD_REPLAY_THREAD` view lists information about all sessions from the replay clients.

See Also:

- *Oracle Database Reference* for information about these views

13

Analyzing Captured and Replayed Workloads

This chapter describes how to analyze captured and replayed workloads using various Database Replay reports. This chapter contains the following sections:

- Using Workload Capture Reports
- Using Workload Replay Reports
- Using Replay Compare Period Reports
- Using SQL Performance Analyzer Reports

> **Note:**
>
> After the replay analysis is complete, you can restore the database to its original state at the time of workload capture and repeat workload replay to test other changes to the system once the workload directory object is backed up to another physical location.

Using Workload Capture Reports

Workload capture reports contain captured workload statistics, information about the top session activities that were captured, and any workload filters used during the capture process.

The following sections describe how to generate and utilize workload capture reports:

- Accessing Workload Capture Reports Using Enterprise Manager
- Generating Workload Capture Reports Using APIs
- Reviewing Workload Capture Reports

Accessing Workload Capture Reports Using Enterprise Manager

This section describes how to generate a workload capture report using Oracle Enterprise Manager.

The primary tool for generating workload capture reports is Oracle Enterprise Manager. If for some reason Oracle Enterprise Manager is unavailable, you can generate workload capture reports using APIs, as described in "Generating Workload Capture Reports Using APIs".

To access workload capture reports using Enterprise Manager:

1. From the Enterprise menu of the Enterprise Manager Cloud Control console, select **Quality Management**, then **Database Replay**.

   If the Database Login page appears, log in as a user with administrator privileges.

   The Database Replay page appears.

2. From the Captured Workloads tab of the Database Replay page for a capture that has a Status other than Completed, click the name of the desired capture from the Capture table.

   The Summary tab of the Database Replay page appears.

3. Click the **Reports** tab for access to controls for the Workload Capture report and Workload Capture ASH Analytics report.

   - The Workload Capture report contains detailed output showing the type of data components that were captured and not captured.
   - The Capture ASH Analytics report shows which sessions are consuming the most database time. This report 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.

     The "Other Activity" list choice for the report means that the activity has not been captured.

4. After you access a report, you can save it by clicking **Save**.

> **See Also:**
>
> - "Reviewing Workload Capture Reports" for information about how to interpret the workload capture report
> - *Oracle Database Performance Tuning Guide* for information about ASH (Active Session History)

Generating Workload Capture Reports Using APIs

You can generate a workload capture report using the `DBMS_WORKLOAD_CAPTURE` package. You can also use Oracle Enterprise Manager to generate a workload capture report, as described in "Accessing Workload Capture Reports Using Enterprise Manager".

To generate a report on the latest workload capture:

1. Use the `DBMS_WORKLOAD_CAPTURE.GET_CAPTURE_INFO` procedure.

   The `GET_CAPTURE_INFO` procedure retrieves all information regarding the workload capture and returns the `cap_id` for the workload capture. This function uses the `dir` required parameter, which specifies the name of the workload capture directory object.

2. Call the `DBMS_WORKLOAD_CAPTURE.REPORT` function.

   The `REPORT` function generates a report using the `cap_id` that was returned by the `GET_CAPTURE_INFO` procedure. This function uses the following parameters:

   - The `capture_id` required parameter relates to the directory that contains the workload capture for which the report will be generated. The directory should be a valid directory in the host system containing the workload capture. The value of this parameter should match the `cap_id` returned by the `GET_CAPTURE_INFO` procedure.
   - The `format` required parameter specifies the report format. Valid values include `DBMS_WORKLOAD_CAPTURE.TYPE_TEXT` and `DBMS_WORKLOAD_REPLAY.TYPE_HTML`.

In this example, the `GET_CAPTURE_INFO` procedure retrieves all information regarding the workload capture in the `jul14` directory and returns the `cap_id` for the workload capture. The

`REPORT` function then generates a text report using the `cap_id` that was returned by the `GET_CAPTURE_INFO` procedure.

```
DECLARE
  cap_id          NUMBER;
  cap_rpt         CLOB;
BEGIN
  cap_id  := DBMS_WORKLOAD_CAPTURE.GET_CAPTURE_INFO(dir => 'jul14');
  cap_rpt := DBMS_WORKLOAD_CAPTURE.REPORT(capture_id => cap_id,
                           format => DBMS_WORKLOAD_CAPTURE.TYPE_TEXT);
END;
/
```

See Also:

- "Reviewing Workload Capture Reports" for information about how to interpret the workload capture report
- *Oracle Database PL/SQL Packages and Types Reference* for information about the `DBMS_WORKLOAD_CAPTURE` package

Reviewing Workload Capture Reports

The workload capture report contains various types of information that can be used to assess the validity of the workload capture. Using the information provided in this report, you can determine if the captured workload:

- Represents the actual workload you want to replay
- Does not contain any workload you want to exclude
- Can be replayed

The information contained in the workload capture report is divided into the following categories:

- Details about the workload capture (such as the name of the workload capture, defined filters, date, time, and SCN of capture)
- Overall statistics about the workload capture (such as the total DB time captured, and the number of logins and transactions captured) and the corresponding percentages with respect to total system activity
- Profile of the captured workload
- Profile of the uncaptured workload due to version limitations
- Profile of the uncaptured workload that was excluded using defined filters
- Profile of the uncaptured workload that consists of background process or scheduled jobs

Using Workload Replay Reports

Workload replay reports contain information that can be used to measure performance differences between the capture system and the replay system.

The following sections describe how to generate and review workload replay reports:

- Accessing Workload Replay Reports Using Enterprise Manager

- Generating Workload Replay Reports Using APIs
- Reviewing Workload Replay Reports

Accessing Workload Replay Reports Using Enterprise Manager

This section describes how to generate a workload replay report using Oracle Enterprise Manager.

The primary tool for generating workload replay reports is Oracle Enterprise Manager. If for some reason Oracle Enterprise Manager is unavailable, you can generate workload replay reports using APIs, as described in "Generating Workload Replay Reports Using APIs"

To access workload replay reports using Enterprise Manager:

1. From the Enterprise menu of the Enterprise Manager Cloud Control console, select **Quality Management**, then **Database Replay**.

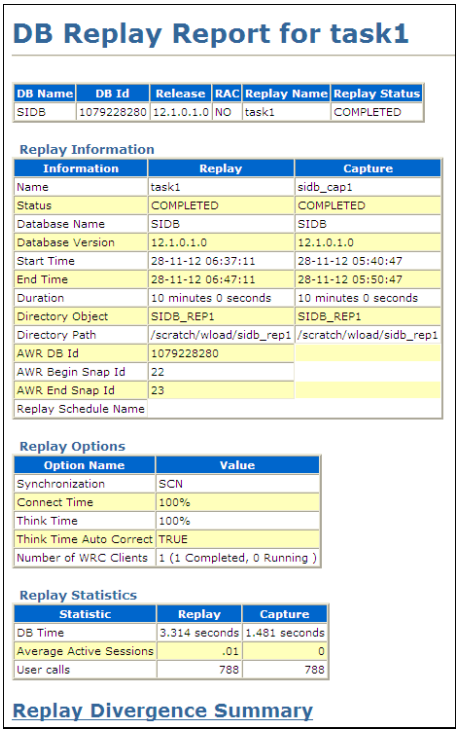
   If the Database Login page appears, log in as a user with administrator privileges.

   The Database Replay page appears.
2. Click the **Replay Tasks** tab, then select a replay for which you want to access reports.
3. Click the **Reports** tab to gain access to individual reports.

   There are several types of reports you can view for a completed workload replay:

   - Database Replay

     Use this report to view complete replay statistics in a tabular format, including replay divergence and the workload profile.



DB Replay Report for task1

DB Name	DB Id	Release	RAC	Replay Name	Replay Status
SIDB	1079228280	12.1.0.1.0	NO	task1	COMPLETED

Replay Information

Information	Replay	Capture
Name	task1	sidb_cap1
Status	COMPLETED	COMPLETED
Database Name	SIDB	SIDB
Database Version	12.1.0.1.0	12.1.0.1.0
Start Time	28-11-12 06:37:11	28-11-12 05:40:47
End Time	28-11-12 06:47:11	28-11-12 05:50:47
Duration	10 minutes 0 seconds	10 minutes 0 seconds
Directory Object	SIDB_REP1	SIDB_REP1
Directory Path	/scratch/wload/sidb_rep1	/scratch/wload/sidb_rep1
AWR DB Id	1079228280	
AWR Begin Snap Id	22	
AWR End Snap Id	23	
Replay Schedule Name		

Replay Options

Option Name	Value
Synchronization	SCN
Connect Time	100%
Think Time	100%
Think Time Auto Correct	TRUE
Number of WRC Clients	1 (1 Completed, 0 Running)

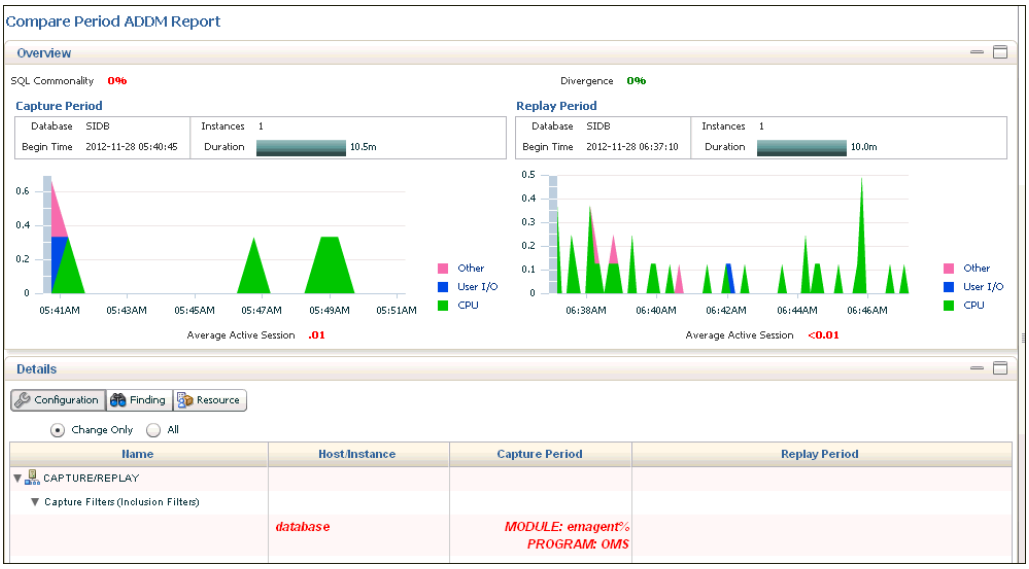
Replay Statistics

Statistic	Replay	Capture
DB Time	3.314 seconds	1.481 seconds
Average Active Sessions	.01	0
User calls	788	788

Replay Divergence Summary

   - Compare Period ADDM

Use this report to perform a high-level comparison of one workload replay to its capture or to another replay of the same capture. Only workload replays that contain at least 5 minutes of database time can be compared using this report.



Examine the following 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

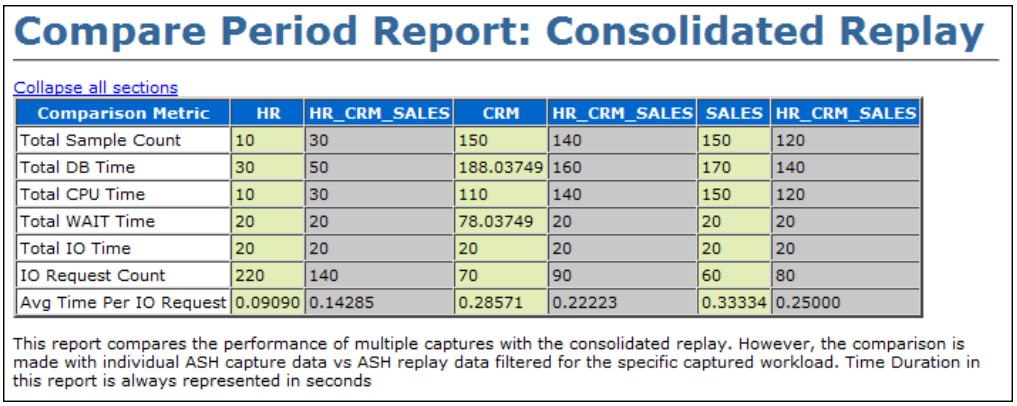
The information shown provides a summary of the division of database time for both time periods, and shows the resource usage for CPU, memory, I/O, and interconnect (Oracle RAC only).

- SQL Performance Analyzer

  Use this report to compare a SQL tuning set from a workload capture to another SQL tuning set from a workload replay, or two SQL tuning sets from two workload replays. Comparing SQL tuning sets with Database Replay provides more information than SQL Performance Analyzer test-execute, because it considers and shows all execution plans for each SQL statement, while SQL Performance Analyzer test-execute generates only one execution plan per SQL statement for each SQL trial.

- Replay Compare Period

  Use this report to compare the AWR data from one workload replay to its capture or to another replay of the same capture. Before running this report, AWR data for the captured or replayed workload must have been previously exported.



Compare Period Report: Consolidated Replay

Collapse all sections

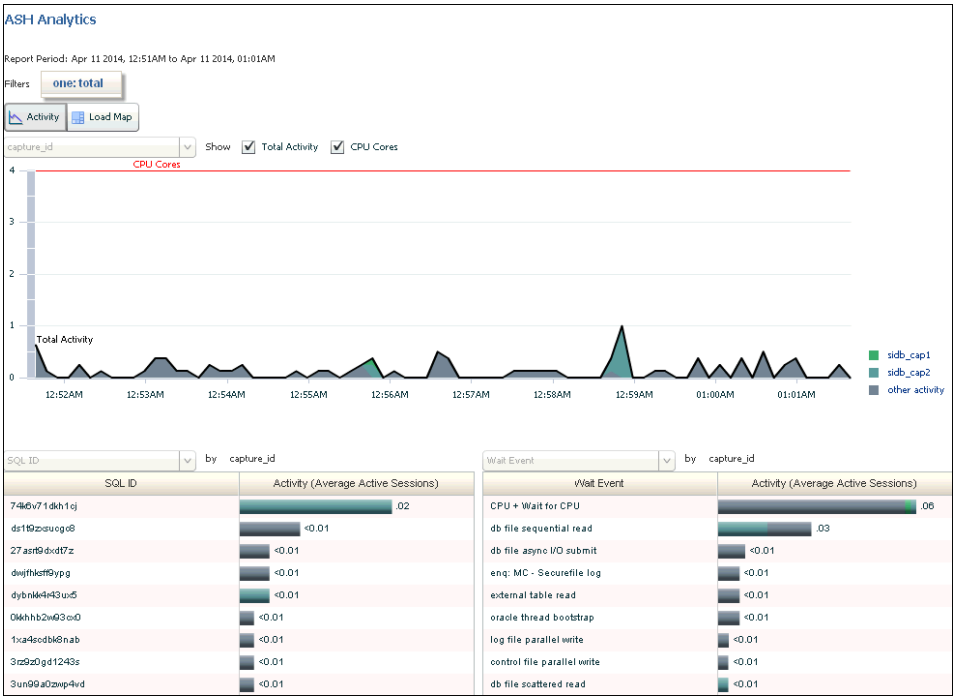
Comparison Metric	HR	HR_CRM_SALES	CRM	HR_CRM_SALES	SALES	HR_CRM_SALES
Total Sample Count	10	30	150	140	150	120
Total DB Time	30	50	188.03749	160	170	140
Total CPU Time	10	30	110	140	150	120
Total WAIT Time	20	20	78.03749	20	20	20
Total IO Time	20	20	20	20	20	20
IO Request Count	220	140	70	90	60	80
Avg Time Per IO Request	0.09090	0.14285	0.28571	0.22223	0.33334	0.25000

This report compares the performance of multiple captures with the consolidated replay. However, the comparison is made with individual ASH capture data vs ASH replay data filtered for the specific captured workload. Time Duration in this report is always represented in seconds

For information about using this report, see "Reviewing Replay Compare Period Reports".

- Replay ASH Analytics

  The Replay ASH Analytics report contains active session history (ASH) information for a specified duration of a workload that was replayed for the category you selected in the drop-down menu. Before running this report, AWR data must have been previously exported from the captured or replayed workload.



The chart shows workload activity breakdown values for wait classes, and provides detailed statistics for the top activity sessions that are adversely affecting the system.

You can optionally 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.

> **See Also:**
>
> *Oracle Database 2 Day + Performance Tuning Guide* for information about how to interpret replay compare period reports

Generating Workload Replay Reports Using APIs

You can generate a workload replay report using the `DBMS_WORKLOAD_REPLAY` package. You can also use Oracle Enterprise Manager to generate a workload replay report, as described in "Accessing Workload Replay Reports Using Enterprise Manager".

To generate a report on the latest workload replay for a workload capture using APIs:

1. Retrieve information about the workload captures and the history of the workload replay attempts from the replay directory object by calling the `DBMS_WORKLOAD_REPLAY.GET_REPLAY_INFO` function, as described in "Retrieving Information About Workload Replays".

   The `GET_REPLAY_INFO` function returns the `cap_id` of a single capture directory (for a consolidated capture directory, the `cap_id` returned is 0).

2. Using the `cap_id` that was returned by the `GET_REPLAY_INFO` function, run a query to return the appropriate `rep_id` for the latest replay of the workload.

3. Call the `DBMS_WORKLOAD_REPLAY.REPORT` function.

   The `REPORT` function generates a report using the `rep_id` that was returned by the `SELECT` statement.

   The `REPORT` function uses the following parameters:

   - The `replay_id` required parameter specifies the directory that contains the workload replay for which the report will be generated. The directory should be a valid directory in the host system containing the workload replay. The value of this parameter should match the `rep_id` returned by the previous query.
   - The `format` parameter required parameter specifies the report format. Valid values include `DBMS_WORKLOAD_REPLAY.TYPE_TEXT`, `DBMS_WORKLOAD_REPLAY.TYPE_HTML`, and `DBMS_WORKLOAD_REPLAY.TYPE_XML`.

In this example, the `GET_REPLAY_INFO` function retrieves all information about the workload captures and the history of all the workload replay attempts from the `jul14` replay directory object. The function returns the `cap_id` of the capture directory, which can be associated with the `CAPTURE_ID` column in the `DBA_WORKLOAD_REPLAYS` view to access the information retrieved.The `SELECT` statement returns the appropriate `rep_id` for the latest replay of the workload. The `REPORT` function then generates a HTML report using the `rep_id` that was returned by the `SELECT` statement.

```
DECLARE
  cap_id          NUMBER;
  rep_id          NUMBER;
  rep_rpt         CLOB;
BEGIN
  cap_id := DBMS_WORKLOAD_REPLAY.GET_REPLAY_INFO(replay_dir => 'jul14');
  /* Get the latest replay for that capture */
  SELECT max(id)
  INTO   rep_id
  FROM   dba_workload_replays
  WHERE  capture_id = cap_id;

  rep_rpt := DBMS_WORKLOAD_REPLAY.REPORT(replay_id => rep_id,
                           format => DBMS_WORKLOAD_REPLAY.TYPE_HTML);
END;
/
```

> **See Also:**
>
> - "Reviewing Workload Replay Reports" for information about how to interpret the workload replay report
> - *Oracle Database PL/SQL Packages and Types Reference* for information about the `DBMS_WORKLOAD_REPLAY` package

Reviewing Workload Replay Reports

After the workload is replayed on a test system, there may be some divergence in what is replayed compared to what was captured. There are numerous factors that can cause replay divergence, which can be analyzed using the workload replay report. The information contained in the workload replay report consists of performance and replay divergence. Performance divergence may result when new algorithms are introduced in the replay system that affect the overall performance of the database. For example, if the workload is replayed on

a newer version of Oracle Database, a new algorithm may cause specific requests to run faster, and the divergence will appear as a faster execution. In this case, this is a desirable divergence.

Data divergence occurs when the results of DML or SQL queries do not match results that were originally captured in the workload. For example, a SQL statement may return fewer rows during replay than those returned during capture.

Error divergence occurs when a replayed database call:

- Encounters a new error that was not captured
- Does not encounter an error that was captured
- Encounters a different error from what was captured

The information contained in the workload replay report is divided into the following categories:

- Details about the workload replay and the workload capture, such as job name, status, database information, duration and time of each process, and the directory object and path
- Replay options selected for the workload replay and the number of replay clients that were started
- Overall statistics about the workload replay and the workload capture (such as the total DB time captured and replayed, and the number of logins and transactions captured and replayed) and the corresponding percentages with respect to total system activity
- Profile of the replayed workload
- Replay divergence
- Error divergence
- DML and SQL query data divergence

Starting with Oracle Database Release 19c, the workload replay report provides information that is required to diagnose a slow workload replay. The information contained in the workload replay report has the following additional sections:

- Replay Sessions
- Synchronization
- Tracked Commits
- Session Failures

Replay Sessions

The Replay Sessions section includes statistics about the ongoing and completed replay sessions, such as top events, top slowest replay sessions, and top fastest replay sessions. The statistics related to the ongoing replay sessions are shown only when the replay is in progress. Use the information in this section to understand the top events for the ongoing and completed replay sessions, the replay sessions that were slower than the capture, and a comparison of the session elapsed time between the capture and replay.

Synchronization

During a workload replay, the execution order of the captured SQL statements is preserved. When a SQL statement's execution is delayed or blocked during the replay, the result is a slower workload replay.

The Synchronization section contains information about the sessions that are blocking the execution of one of the synchronized SQL statements. This information is shown only when a SQL statement's execution is blocked. Use the information in this section to understand why a workload replay is blocked or is slower than the workload capture. This section also includes the top events and the top SQL statements with top events related to sessions that are running synchronized SQL statements.

Tracked Commits

The commits executed during a workload capture and the time consumed between their executions are tracked by the Database Replay. It can detect slow commits and identify the parts of the captured workload that slowed down the workload replay.

The Tracked Commits section includes information about the number of captured commits and the time at which a commit was executed. Use this information to find whether the workload replay is moving slower than the workload capture.

Session Failures

When the workload replay of a user session fails, the replay of the remaining calls in the session are skipped. The Session Failures section includes information about the SQL statement that was executed during a session failure, the file ID and the call counter of the failed session, and the total number of calls that were not executed.

Using Replay Compare Period Reports

Replay compare period reports can be used for several purposes. For example, you can use replay compare period reports to compare the performance of:

- A workload replay to its workload capture
- A workload replay to another replay of the same workload capture
- Multiple workload captures to a consolidated replay

The following sections describe how to generate and review replay compare period reports:

- Generating Replay Compare Period Reports Using APIs
- Reviewing Replay Compare Period Reports

> **See Also:**
>
> - Using Consolidated Database Replay for information about Consolidated Database Replay

Generating Replay Compare Period Reports Using APIs

This section describes how to generate replay compare period reports using the `DBMS_WORKLOAD_REPLAY` package. This report only compares workload replays that contain at least 5 minutes of database time.
To generate replay compare period reports, use the `DBMS_WORKLOAD_REPLAY.COMPARE_PERIOD_REPORT` procedure:

```
BEGIN
  DBMS_WORKLOAD_REPLAY.COMPARE_PERIOD_REPORT (
                           replay_id1 => 12,
                           replay_id2 => 17,
                           format => DBMS_WORKLOAD_CAPTURE.TYPE_HTML,
                           result => :report_bind);
END;
/
```

In this example, the `COMPARE_PERIOD_REPORT` procedure generates a replay compare period report in HTML format that compares a workload replay with a replay ID of 12 with another replay with an ID of 17.

The `COMPARE_PERIOD_REPORT` procedure in this example uses the following parameters:

- The `replay_id1` parameter specifies the numerical identifier of the workload replay after change for which the reported will be generated. This parameter is required.
- The `replay_id2` parameter specifies the numerical identifier of the workload replay before change for which the reported will be generated. If unspecified, the comparison will be performed with the workload capture.
- The `format` parameter specifies the report format. Valid values include `DBMS_WORKLOAD_CAPTURE.TYPE_HTML` for HTML and `DBMS_WORKLOAD_CAPTURE.TYPE_XML` for XML. This parameter is required.
- The result parameter specifies the output of the report.

> **See Also:**
>
> - "Reviewing Replay Compare Period Reports" for information about how to interpret the replay compare period report
> - *Oracle Database PL/SQL Packages and Types Reference* for information about the `DBMS_WORKLOAD_REPLAY` package

Reviewing Replay Compare Period Reports

Reviewing replay compare period reports enables you to determine if any replay divergence occurred and whether there were any significant performance changes.

Depending on the type of comparison that is being made, one of three types of replay compare period reports is generated:

- Capture vs. Replay

  This report type compares the performance of a workload replay to its workload capture.

- Replay vs. Replay

  This report type compares the performance of two workload replays of the same workload capture.

- Consolidated Replay

  This report type compares the performance of multiple workload captures to a consolidated replay. Only the ASH Data Comparison section is available for this report type. For more information about this report type, see "Reporting and Analysis for Consolidated Database Replay".

All replay compare period report types contain information about the most important changes between the two runs that are being compared. Use this information to determine the appropriate action to take. For example, if a new concurrency issue is found, review Automatic Database Diagnostic Monitor (ADDM) reports to diagnose the issue. If a new SQL performance problem is found, run SQL Tuning Advisor to fix the problem.

The information in the replay compare period report is divided into the following sections:

- General Information
- Replay Divergence
- Main Performance Statistics
- Top SQL/Call
- Hardware Usage Comparison
- ADDM Comparison
- ASH Data Comparison

General Information

This section contains metadata about the two runs being compared in the report. Any `init.ora` parameter changes between the two runs are also shown here. Review this section to verify if the system change being tested was performed.

Replay Divergence

This section contains the divergence analysis of the second run relative to the first. If the analysis shows significant divergence, review the full divergence report.

Main Performance Statistics

This section contains a high-level performance statistic comparison across the two runs (such as change in database time). If the comparison shows an insignificant change in database time, then the performance between the two runs are generally similar. If there is a significant change in database time, review the statistics to determine the component (CPU or user I/O) that is causing the greatest change.

Top SQL/Call

This section compares the performance change of individual SQL statements from one run to the next. The SQL statements are ordered by the total change in database time. Top SQL statements that regressed by the most database time are best candidates for SQL tuning.

Hardware Usage Comparison

This section compares CPU and I/O usage across the two runs. The number of CPUs is summed for all instances and CPU usage is averaged over instances.

I/O statistics are shown for data and temp files. A high value for the single block read time (much higher than 10 milliseconds) suggests that the system is I/O bound. If this is the case, then review the total read and write times to determine if the latency is caused by excessive I/O requests or poor I/O throughput.

ADDM Comparison

This section contains a comparison of ADDM analyses across the two runs ordered by the absolute difference in impact. Compare the ADDM results for the two runs to identify possible problems that only existed in one run. If the system change being tested is intended to improve database performance, then verify if the expected improvement is reflected in the ADDM findings.

> **See Also:**
>
> - *Oracle Database Performance Tuning Guide* for information about ADDM analysis

ASH Data Comparison

This section compares the ASH data across the two runs. The begin time and end time of the comparison period are displayed in a table. These times may not match the capture and replay times of the two runs being compared. Instead, these times represent the times when the ASH samples were taken.
The ASH Data Comparison section contains the following subsections:

- Compare Summary
- Top SQL
- Long Running SQL
- Common SQL
- Top Objects

> **See Also:**
>
> - *Oracle Database Performance Tuning Guide* for information about ASH

Compare Summary

This section summarizes the activity during the two runs based on database time and wait time distribution. For example:

- DB Time Distribution indicates how the total database time is distributed across CPU usage, wait times, and I/O requests.

  Figure 13-1 shows the DB Time Distribution subsection of a sample report.

Figure 13-1 DB Time Distribution

(-) DB Time Distribution

Compare Attr	Time Period1	Time Period2
Total Sample Count	2050	7490
Total DB Time	2519.88801	10164.33483
Total CPU Time	2260	9130
Total WAIT Time	259.88801	1034.33483
Total IO Time	10	770
IO Request Count	2924	329819
Avg Time Per IO Request	00000000.00342	00000000.00233

- Wait Time Distribution indicates how the total wait time is distributed across wait events. The top wait event class, event name, and event count are listed for both runs.

  Figure 13-2 shows the Wait Time Distribution subsection of a sample report.

Figure 13-2 Wait Time Distribution

(-) Wait Time Distribution

Time Period (1)

Event(1)	Wait time(1)	Event Count(1)
log file switch (checkpoint incomplete),Configuration	239.88801	12
direct path write temp,User I/O	10	2924
log buffer space,Configuration	10	6

Time Period (2)

Event(2)	Wait time(2)	Event Count(2)
direct path read,User I/O	590	185443
log file switch (checkpoint incomplete),Configuration	114.33483	20
direct path read temp,User I/O	80	132686
free buffer waits,Configuration	70	435
direct path write temp,User I/O	50	73
db file sequential read,User I/O	50	11617
log file switch completion,Configuration	30	44
log buffer space,Configuration	30	34
latch: row cache objects,Concurrency	10	28
undo segment extension,Configuration	10	4

Top SQL

This section displays the top SQL statements for both runs by total database time, CPU time, and wait time.

Long Running SQL

This section displays the top long-running SQL statements for both runs. Each long-running SQL statement contains details about the query, such as the maximum, minimum, and average response times.

Common SQL

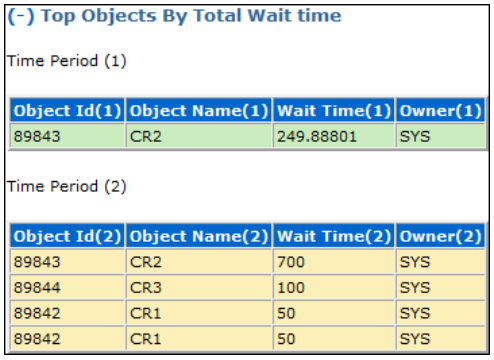
This section extracts the SQL statements that are common in both runs and displays the top common SQL statements by variance in average response time and total database time.

Top Objects

This section contains details about the top objects for both runs by total wait time.

Figure 13-3 shows the Top Objects section of a sample report.

Figure 13-3 Top Objects



(-) Top Objects By Total Wait time

Time Period (1)

Object Id(1)	Object Name(1)	Wait Time(1)	Owner(1)
89843	CR2	249.88801	SYS

Time Period (2)

Object Id(2)	Object Name(2)	Wait Time(2)	Owner(2)
89843	CR2	700	SYS
89844	CR3	100	SYS
89842	CR1	50	SYS
89842	CR1	50	SYS

Using SQL Performance Analyzer Reports

Use the SQL Performance Analyzer report to compare a SQL tuning set from a workload replay to another SQL tuning set from a workload capture, or two SQL tuning sets from two workload replays.

Comparing SQL tuning sets with Database Replay provides more information than SQL Performance Analyzer test-execute because it considers and shows all execution plans for each SQL statement, while SQL Performance Analyzer test-execute generates only one execution plan per SQL statement for each SQL trial.

Generating SQL Performance Analyzer Reports Using APIs

This section describes how to generate a SQL Performance Analyzer report using the `DBMS_WORKLOAD_REPLAY` package.

To generate a SQL Performance Analyzer report, use the `DBMS_WORKLOAD_REPLAY.COMPARE_SQLSET_REPORT` procedure:

```
BEGIN
  DBMS_WORKLOAD_REPLAY.COMPARE_SQLSET_REPORT (
                           replay_id1 => 12,
                           replay_id2 => 17,
                           format => DBMS_WORKLOAD_CAPTURE.TYPE_HTML,
                           result => :report_bind);
END;
/
```

In this example, the `COMPARE_SQLSET_REPORT` procedure generates a SQL Performance Analyzer report in HTML format that compares a SQL tuning set captured during the workload replay with a replay ID of 12 to a SQL tuning set captured during workload replay with an ID of 17.

The `COMPARE_SQLSET_REPORT` procedure in this example uses the following parameters:

- The `replay_id1` parameter specifies the numerical identifier of the workload replay after change for which the reported will be generated. This parameter is required.

- The `replay_id2` parameter specifies the numerical identifier of the workload replay after change for which the reported will be generated. If unspecified, the comparison will be performed with the workload capture.
- The `format` parameter specifies the report format. Valid values include `DBMS_WORKLOAD_CAPTURE.TYPE_HTML` for HTML, `DBMS_WORKLOAD_CAPTURE.TYPE_XML` for XML, and `DBMS_WORKLOAD_CAPTURE.TYPE_TEXT` for text. This parameter is required.
- The result parameter specifies the output of the report.

See Also:

- "Reviewing the SQL Performance Analyzer Report in Command-Line" for information about how to interpret the SQL Performance Analyzer report
- *Oracle Database PL/SQL Packages and Types Reference* for information about the `DBMS_WORKLOAD_REPLAY` package

14

Using Workload Intelligence

Workload Intelligence analyzes the data stored in a captured workload and identifies significant patterns in a workload. This chapter describes how to use Workload Intelligence and contains the following sections:

- Overview of Workload Intelligence
- Analyzing Captured Workloads Using Workload Intelligence
- Example: Workload Intelligence Results

Overview of Workload Intelligence

Workload capture generates a number of binary files—called capture files—that contain relevant information about the captured workload. The information stored in the capture files enables workload replay to realistically reproduce the captured workload at a later time. For each client request that is recorded by workload capture, the captured information includes SQL text, bind values, transaction information, timing data, identifiers of accessed objects, and other information about the request.
Workload Intelligence enables you to use the information stored in capture files in additional ways, including the following:

- Analyze and model the workload
- Discover significant patterns and trends within the workload
- Visualize what was running on the production system during workload capture

This section describes Workload Intelligence and contains the following topics:

- About Workload Intelligence
- Use Case for Workload Intelligence
- Requirements for Using Workload Intelligence

> **See Also:**
>
> - "Workload Capture" for information about workload capture
> - "Workload Replay" for information about workload replay

About Workload Intelligence

Workload Intelligence comprises a suite of Java programs that enable you to analyze the data stored in a captured workload. These Java programs operate on data recorded during a workload capture to create a model that describes the workload. This model can help you identify significant patterns of templates that are executed as part of the workload.

A template represents a read-only SQL statement, or an entire transaction that consists of one or more SQL statements. If two SQL statements (or transactions) exhibit significant similarity, then they are represented by the same template.

Workload Intelligence enables you to better visualize what a captured workload looks like by exploring template patterns and the corresponding SQL statements. For each pattern, you can view important statistics, such as the number of executions of a given pattern and the database time consumed by the pattern during its execution.

Use Case for Workload Intelligence

You can use Workload Intelligence to discover significant patterns in a captured workload.

SQL statements that are executed in a production system are typically not manually inputted by the users, but instead come from one or more applications running on an application server that is connected to the database server. There is usually a finite number of such SQL statements in an application. Even if different bind values are used in every execution of a particular statement, its SQL text essentially remains the same.

Depending on the user input to the application, a code path is executed that includes one or more SQL statements submitted to the database in a given order as defined by the application code. Frequent user actions correspond to application code paths that are regularly executed. Such frequently executed code paths generate a frequent pattern of SQL statements that are executed by the database in a given order. By analyzing a captured workload, Workload Intelligence discovers such patterns and associates them with related execution statistics. In other words, Workload Intelligence uses the information stored in capture files to discover patterns that are generated by significant code paths of applications running in the production system during workload capture. Workload Intelligence does this without requiring any information about the applications.

Using Workload Intelligence to discover significant patterns:

- Enables you to better visualize what was running in the database during workload capture.
- Provides more information that can be used for optimizations.
- Offers a better context because SQL statements are not isolated, but are combined.

Requirements for Using Workload Intelligence

Workload Intelligence uses the information that is stored in the capture files, and does not require the execution of the workload using workload replay. Furthermore, Workload Intelligence does not require any user schema, user data, or connection to the production system. To avoid any overhead in the production system, it is recommended that Workload Intelligence be used on a test system where the capture files have been copied, especially for large captured workloads because running Workload Intelligence may be resource intensive. The necessary Java classes for invoking the Java programs that comprise Workload Intelligence are packed in `$ORACLE_HOME/rdbms/jlib/dbrintelligence.jar`. Two other jar files must be included in the classpath: `$ORACLE_HOME/rdbms/jlib/dbrparser.jar` and `$ORACLE_HOME/jdbc/lib/ojdbc6.jar`.

Workload Intelligence also uses some `SYS` tables and views internally.

Analyzing Captured Workloads Using Workload Intelligence

This section introduces the steps for analyzing captured workloads using Workload Intelligence. For example:

To analyze captured workloads using Workload Intelligence:

1. Create a database user with the appropriate privileges to use Workload Intelligence, as described in "Creating a Database User for Workload Intelligence".
2. Create a new Workload Intelligence job by running the `LoadInfo` Java program, as described in "Creating a Workload Intelligence Job".
3. Generate a model that describes the workload by running the `BuildModel` Java program, as described in "Generating a Workload Model".
4. Identify patterns in templates that occur in the workload by running the `FindPatterns` Java program, as described in "Identifying Patterns in a Workload".
5. Generate a report to display the results by running the `GenerateReport` Java program, as described in "Generating a Workload Intelligence Report".

Creating a Database User for Workload Intelligence

Before using Workload Intelligence, first create a database user with the appropriate privileges.

Example 14-1 shows how to create a database user that can use Workload Intelligence.

Example 14-1 Creating a Database User for Workload Intelligence

```
create user workintusr identified by password;
grant create session to workintusr;
grant select,insert,alter on WI$_JOB to workintusr;
grant insert,alter on WI$_TEMPLATE to workintusr;
grant insert,alter on WI$_STATEMENT to workintusr;
grant insert,alter on WI$_OBJECT to workintusr;
grant insert,alter on WI$_CAPTURE_FILE to workintusr;
grant select,insert,alter on WI$_EXECUTION_ORDER to workintusr;
grant select,insert,update,delete,alter on WI$_FREQUENT_PATTERN to workintusr;
grant select,insert,delete,alter on WI$_FREQUENT_PATTERN_ITEM to workintusr;
grant select,insert,delete,alter on WI$_FREQUENT_PATTERN_METADATA to workintusr;
grant select on WI$_JOB_ID to workintusr;
grant execute on DBMS_WORKLOAD_REPLAY to workintusr;
```

Creating a Workload Intelligence Job

To create a Workload Intelligence job, use the `LoadInfo` program. `LoadInfo` is a Java program that creates a new task to apply the algorithms of Workload Intelligence. The program parses the data contained in a capture directory and stores the relevant information required for running Workload Intelligence in internal tables.
The `LoadInfo` program uses the following syntax:

```
java oracle.dbreplay.workload.intelligence.LoadInfo -cstr connection_string -user
username -job job_name -cdir capture_directory

java oracle.dbreplay.workload.intelligence.LoadInfo -version

java oracle.dbreplay.workload.intelligence.LoadInfo -usage
```

The `LoadInfo` program supports the following options:

- `-cstr`

  Specifies the JDBC connection string to the database where Workload Intelligence stores the information and intermediate results required for execution (for example, `jdbc:oracle:thin@hostname:portnum:ORACLE_SID`)

- `-user`

  Specifies the database username. The user must have certain privileges for using Workload Intelligence.

  For information about creating a database user with the appropriate privileges, see "Creating a Database User for Workload Intelligence".

- `-job`

  Specifies a name that uniquely identifies the Workload Intelligence job.

- `-cdir`

  Specifies the operating system path of the capture directory to be analyzed by Workload Intelligence.

- `-version`

  Displays the version information for the `LoadInfo` program.

- `-usage`

  Displays the command-line options for the `LoadInfo` program.

Example 14-2 shows how to create a workload intelligence job named `wijobsales` using the `LoadInfo` program.

Example 14-2 Creating a Workload Intelligence Job

```
java -classpath $ORACLE_HOME/rdbms/jlib/dbrintelligence.jar:
$ORACLE_HOME/rdbms/jlib/dbrparser.jar:
$ORACLE_HOME/jdbc/lib/ojdbc6.jar:
oracle.dbreplay.workload.intelligence.LoadInfo -job wijobsales -cdir
/test/captures/sales -cstr jdbc:oracle:thin:@myhost:1521:orcl -user workintusr
```

Generating a Workload Model

To generate a workload model, use the `BuildModel` program. `BuildModel` is a Java program that reads data from a captured workload (this data must be generated by the `LoadInfo` program) and generates a model that describes the workload. This model can then be used to identify frequent template patterns that occur in the workload.
The `BuildModel` program uses the following syntax:

```
java oracle.dbreplay.workload.intelligence.BuildModel -cstr connection_string -user
username -job job_name

java oracle.dbreplay.workload.intelligence.BuildModel -version

java oracle.dbreplay.workload.intelligence.BuildModel -usage
```

The `BuildModel` program supports the following options:

- `-cstr`

Specifies the JDBC connection string to the database where Workload Intelligence stores the information and intermediate results required for execution (for example, `jdbc:oracle:thin@hostname:portnum:ORACLE_SID`)

- `-user`

  Specifies the database username. The user must have certain privileges for using Workload Intelligence.

  For information about creating a database user with the appropriate privileges, see "Creating a Database User for Workload Intelligence".

- `-job`

  Specifies a name that uniquely identifies the Workload Intelligence job.

- `-version`

  Displays the version information for the `BuildModel` program.

- `-usage`

  Displays the command-line options for the `BuildModel` program.

Example 14-3 shows how to generate a workload model using the `BuildModel` program.

Example 14-3 Generating a Workload Model

```
java -classpath $ORACLE_HOME/rdbms/jlib/dbrintelligence.jar:
$ORACLE_HOME/rdbms/jlib/dbrparser.jar:
$ORACLE_HOME/jdbc/lib/ojdbc6.jar:
oracle.dbreplay.workload.intelligence.BuildModel -job wijobsales -cstr
jdbc:oracle:thin:@myhost:1521:orcl -user workintusr
```

Identifying Patterns in a Workload

To identify patterns in a workload, use the `FindPatterns` program. `FindPatterns` is a Java program that reads data from a captured workload (this data must be generated by the `LoadInfo` program) and its corresponding workload model (the workload model must be generated by the `BuildModel` program), and identifies frequent template patterns that occur in the workload.
The `FindPatterns` program uses the following syntax:

```
java oracle.dbreplay.workload.intelligence.FindPatterns -cstr connection_string
-user username -job job_name -t threshold

java oracle.dbreplay.workload.intelligence.FindPatterns -version

java oracle.dbreplay.workload.intelligence.FindPatterns -usage
```

The `FindPatterns` program supports the following options:

- `-cstr`

  Specifies the JDBC connection string to the database where Workload Intelligence stores the information and intermediate results required for execution (for example, `jdbc:oracle:thin@hostname:portnum:ORACLE_SID`)

- `-user`

  Specifies the database username. The user must have certain privileges for using Workload Intelligence.

For information about creating a database user with the appropriate privileges, see "Creating a Database User for Workload Intelligence".

- `-job`

  Specifies a name that uniquely identifies the Workload Intelligence job.

- `-t`

  Specifies a threshold probability that defines when a transition from one template to the next is part of the same pattern or the border between two patterns. Valid values include real numbers in the range [0.0, 1.0]. Setting this value is optional; its default value is `0.5`.

- `-version`

  Displays the version information for the `FindPatterns` program.

- `-usage`

  Displays the command-line options for the `FindPatterns` program.

Example 14-4 shows how to identify frequent template patterns in a workload using the `FindPatterns` program.

Example 14-4 Identifying Patterns in a Workload

```
java -classpath $ORACLE_HOME/rdbms/jlib/dbrintelligence.jar:
$ORACLE_HOME/rdbms/jlib/dbrparser.jar:
$ORACLE_HOME/jdbc/lib/ojdbc6.jar:
oracle.dbreplay.workload.intelligence.FindPatterns -job wijobsales -cstr
jdbc:oracle:thin:@myhost:1521:orcl -user workintusr -t 0.2
```

Generating a Workload Intelligence Report

To generate a Workload Intelligence report, use the `GenerateReport` program. `GenerateReport` is a Java program that generates a report to display the results of Workload Intelligence. The Workload Intelligence report is an HTML page that displays the patterns identified in the workload.
The `GenerateReport` program uses the following syntax:

```
java oracle.dbreplay.workload.intelligence.GenerateReport -cstr connection_string
-user username -job job_name -top top_patterns -out filename

java oracle.dbreplay.workload.intelligence.GenerateReport -version

java oracle.dbreplay.workload.intelligence.GenerateReport -usage
```

The `GenerateReport` program supports the following options:

- `-cstr`

  Specifies the JDBC connection string to the database where Workload Intelligence stores the information and intermediate results required for execution (for example, `jdbc:oracle:thin@hostname:portnum:ORACLE_SID`)

- `-user`

  Specifies the database username. The user must have certain privileges for using Workload Intelligence.

  For information about creating a database user with the appropriate privileges, see "Creating a Database User for Workload Intelligence".

- `-job`

Specifies a name that uniquely identifies the Workload Intelligence job.

- `-top`

  Specifies a number that indicates how many patterns to display in the report. The patterns are ordered by different criteria (number of executions, DB time, and length) and only the defined number of top results are displayed. Setting this value is optional; its default value is `10`.

- `-out`

  Specifies the name of the file (in HTML format) where the report is stored. Setting this value is optional; its default value is based on the job name specified in the `-job` option.

- `-version`

  Displays the version information for the `GenerateReport` program.

- `-usage`

  Displays the command-line options for the `GenerateReport` program.

Example 14-5 shows how to generate a Workload Intelligence report using the `GenerateReport` program.

Example 14-5 Generating a Workload Intelligence Report

```
java -classpath $ORACLE_HOME/rdbms/jlib/dbrintelligence.jar:
$ORACLE_HOME/rdbms/jlib/dbrparser.jar:
$ORACLE_HOME/jdbc/lib/ojdbc6.jar:
oracle.dbreplay.workload.intelligence.GenerateReport -job wijobsales -cstr
jdbc:oracle:thin:@myhost:1521:orcl -user workintusr -top 5 -out wijobsales.html
```

Example: Workload Intelligence Results

This section assumes a scenario where Workload Intelligence is used on a captured workload generated by Swingbench, a benchmark used for stress testing Oracle Database.

The most significant pattern discovered by Workload Intelligence consists of the following 6 templates:

```
SELECT product_id, product_name, product_description, category_id, weight_class,
       supplier_id, product_status, list_price, min_price, catalog_url
  FROM product_information
 WHERE product_id = :1;

SELECT p.product_id, product_name, product_description, category_id, weight_class,
       supplier_id, product_status, list_price, min_price, catalog_url,
       quantity_on_hand, warehouse_id
  FROM product_information p, inventories i
 WHERE i.product_id = :1 and i.product_id = p.product_id;

INSERT INTO order_items (order_id, line_item_id, product_id, unit_price, quantity)
     VALUES (:1, :2, :3, :4, :5);

UPDATE orders
   SET order_mode = :1, order_status = :2, order_total = :3
 WHERE order_id = :4;

SELECT /*+ use_nl */ o.order_id, line_item_id, product_id, unit_price, quantity,
       order_mode, order_status, order_total, sales_rep_id, promotion_id,
       c.customer_id, cust_first_name, cust_last_name, credit_limit, cust_email
  FROM orders o, order_items oi, customers c
```

```
 WHERE o.order_id = oi.order_id
   AND o.customer_id = c.customer_id
   AND o.order_id = :1;

UPDATE inventories
   SET quantity_on_hand = quantity_on_hand - :1
 WHERE product_id = :2
   AND warehouse_id = :3;
```

This pattern corresponds to a common user action for ordering a product. In this example, the identified pattern was executed 222,261 times (or approximately 8.21% of the total number of executions) and consumed 58,533.70 seconds of DB time (or approximately 11.21% of total DB time).

Another significant pattern discovered by Workload Intelligence in this example consists of the following 4 templates:

```
SELECT customer_seq.nextval
  FROM dual;

INSERT INTO customers (customer_id, cust_first_name, cust_last_name, nls_language,
                       nls_territory, credit_limit, cust_email, account_mgr_id)
     VALUES (:1, :2, :3, :4, :5, :6, :7, :8);

INSERT INTO logon
     VALUES (:1, :2);

SELECT customer_id, cust_first_name, cust_last_name, nls_language, nls_territory,
       credit_limit, cust_email, account_mgr_id
  FROM customers
 WHERE customer_id = :1;
```

This pattern corresponds to the creation of a new customer account followed by a login in the system. In this example, the identified pattern was executed 90,699 times (or approximately 3.35% of the total number of executions) and consumed 17,484.97 seconds of DB time (or approximately 3.35% of total DB time).

15

Using Consolidated Database Replay

Database Replay enables you to capture a workload on the production system and replay it on a test system. This can be very useful when evaluating or adopting new database technologies because these changes can be tested on a test system without affecting the production system. However, if the new system being tested offers significantly better performance than the existing system, then Database Replay may not accurately predict how much additional workload can be handled by the new system.
For example, if you are consolidating multiple production systems into a single Oracle Exadata Machine, replaying a workload captured from one of the existing systems on Oracle Exadata Machine may result in much lower resource usage (such as host CPU and I/O) during replay because the new system is much more powerful. In this case, it is more useful to assess how the new system will handle the combined workloads from all existing systems, rather than that of a single workload from one system.

Consolidated Database Replay enables you to consolidate multiple workloads captured from one or multiple systems and replay them concurrently on a single test system. In this example, using Consolidated Database Replay will help you to assess how the database consolidation will affect the production system and if a single Oracle Exadata Machine can handle the combined workloads from the consolidated databases.

This chapter describes how to use Consolidated Database Replay and contains the following sections:

Use Cases for Consolidated Database Replay

Consolidated Database Replay enables you to replay multiple workloads captured from one or multiple systems concurrently. During the replay, every workload capture that is consolidated will start to replay when the consolidated replay begins.
Some typical use cases for Consolidated Database Replay include:

- Database Consolidation Using Pluggable Databases
- Stress Testing
- Scale-Up Testing

Each of these use cases can be performed using the procedures described in this chapter. In addition, you can employ various workload scale-up techniques when using Consolidated Database Replay, as described in Using Workload Scale-Up .

Database Consolidation Using Pluggable Databases

One use for Consolidated Database Replay is to assess if the system can handle the combined workload from a database consolidation.

For example, assume that you want to consolidate the databases for the CRM, ERP, and SCM applications by migrating them to pluggable databases (PDBs). You can use Consolidated Database Replay to combine the captured workloads from the three applications and replay them concurrently on PDBs.

> **See Also:**
>
> "Example: Replaying a Consolidated Workload with APIs" for an example of this use case

Stress Testing

Another use for Consolidated Database Replay is for stress testing or capacity planning.

For example, assume that you are expecting the workload for the Sales application to double during the holiday season. You can use Consolidated Database Replay to test the added stress on the system by doubling the workload and replaying the combined workload.

> **See Also:**
>
> "Using Time Shifting" for an example of this use case

Scale-Up Testing

A third use for Consolidated Database Replay is for scale-up testing.

For example, assume that you want to test if your system can handle captured workloads from the Financials application and the Orders application concurrently. You can use Consolidated Database Replay to test the effects of the scaled-up workload on your system by combining the workloads and replaying them simultaneously.

> **See Also:**
>
> - "Using Schema Remapping"
> - "Using Workload Folding"

Steps for Using Consolidated Database Replay

This section describes the steps involved when using Consolidated Workload Replay. It contains the following topics:

Capturing Database Workloads for Consolidated Database Replay

Consolidated Database Replay does not require any special steps for capturing database workloads. The steps for capturing database workloads are exactly the same as for capturing a single workload for Database Replay, as described in Capturing a Database Workload . This section contains the following topics for workload captures that are specific to Consolidated Database Replay:

Supported Types of Workload Captures

Consolidated Database Replay supports multiple workloads captured from one or multiple systems running Oracle Database 9*i* Release 2 (release 9.2.0.8.0) or higher on one or multiple operating systems. For example, you can use workloads captured from one system running Oracle Database 9*i* Release 2 (release 9.2.0.8.0) on HP-UX and another system running Oracle Database 10*g* Release 2 (release 10.2.0.4.0) on AIX.

> **Note:**
>
> Consolidated Database Replay is only available on Oracle Database 11*g* Release 2 (release 11.2.0.2.0) and higher.

Capture Subsets

Consolidated Database Replay enables you to transform existing workload captures into new, smaller pieces of capture subsets. You can then generate new workload captures from the capture subsets that can be used in different use cases, as described in "Use Cases for Consolidated Database Replay".
A **capture subset** is a piece of a workload capture that is defined from an existing workload capture by applying a time range. The time range is specified as an offset from the start of the workload capture. All user workloads captured within the specified time range are included in the defined capture subset.

For example, assume that a workload was captured from 2 a.m. to 8 p.m. and the peak workload is identified to be from 10 a.m. to 4 p.m. You can define a capture subset to represent the peak workload by applying a time range that starts at 8 hours after the start of the workload (or 10 a.m.) and ends at 14 hours after the start of the workload (or 4 p.m.).

However, if a capture subset only contains recorded user workloads that satisfy the specified time range, user logins that occurred before the specified time range are not recorded. If these user logins are required for replay, then the capture subset may not be replayable. For example, if a user session starts at 9:30 a.m. and ends at 10:30 a.m. and the specified time range for the capture subset is 10:00 a.m. to 4:00 p.m., the replay may fail if the user login at

9:30 a.m. is not included in the workload. Similarly, the specified time range may also include incomplete user calls that are only partially recorded if a user sessions starts at 3:30 p.m. but does not complete until 4:30 p.m.

Consolidated Database Replay addresses this problem by including only incomplete user calls caused by the start time of the specified time range. To avoid including the same incomplete user calls twice if the workload capture is folded, incomplete user calls caused by the end time are *not* included by default. Therefore, a capture subset is essentially the minimal number of recorded user calls during a specified time range that are required for proper replay, including the necessary user logins, alter session statements, and incomplete user calls caused by the start time.

> **See Also:**
>
> "Generating Capture Subsets Using APIs"

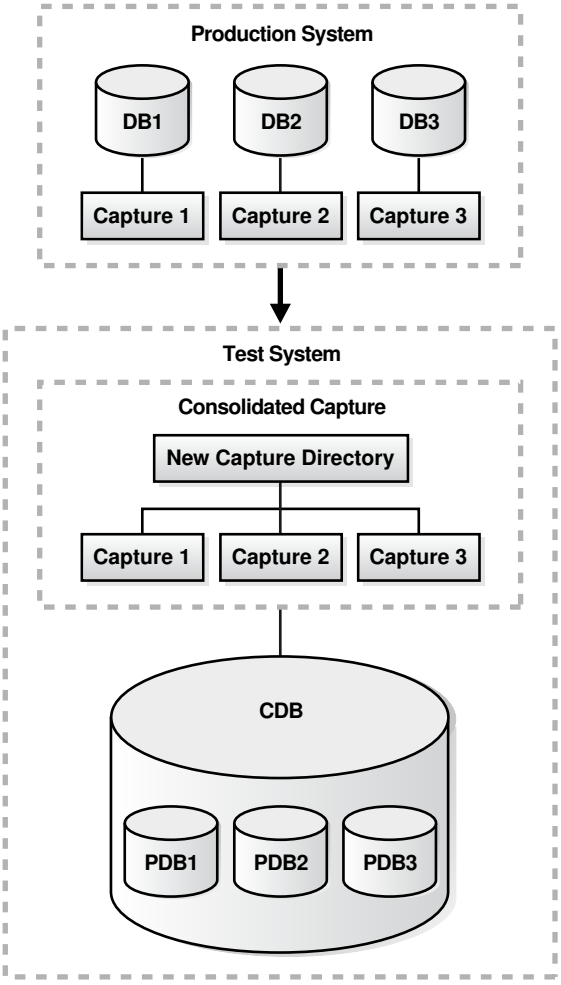
Setting Up the Test System for Consolidated Database Replay

Setting up the test system for Consolidated Database Replay is similar to setting up a test system for Database Replay. However, there are some additional considerations when setting up a replay database for Consolidated Database Replay. See "Steps for Replaying a Database Workload" for more information about setting up a test system for Database Replay.
To minimize divergence during the replay, the test system should contain the same application data and the state of the application data should be logically equivalent to that of the capture system at the start time of each workload capture. However, because a consolidated capture may contain multiple workload captures from different production systems, the test system needs to be set up for all the captures. In this case, it is recommended that the multitenant architecture be used to consolidate multiple databases, so that each database will have equivalent data to its capture system at the capture start time.

For Consolidated Database Replay, all participating workload captures must be placed under a new capture directory on the test system. You can copy all the workload captures into the new capture directory, or create symbolic links pointing to the original workload captures. Before consolidating the workload captures, ensure that the new capture directory has enough disk space to store all participating captures.

Figure 15-1 illustrates how to set up the test system and new capture directory to consolidate three workload captures.

Figure 15-1 Setting Up the Test System for Consolidated Database Replay



See Also:

- "Setting the Consolidated Replay Directory Using APIs"
- *Oracle Database Concepts* for information about the multitenant architecture

Preprocessing Database Workloads for Consolidated Database Replay

Preprocessing a database workload for Consolidated Database Replay is similar to preprocessing a database workload for Database Replay. See " Preprocessing a Database Workload " for information about preprocessing a database workload for Database Replay. For Consolidated Database Replay, preprocess each captured workload into its own directory. Do not combine different workload captures into one directory for preprocessing. Preprocessing of captured workloads must be performed using a database running the same version of Oracle Database as that of the test system where the workloads will be replayed.

Replaying Database Workloads for Consolidated Database Replay

Replaying consolidated workloads using Consolidated Database Replay is quite different from replaying a single database workload using Database Replay.

This section contains the following topics for replaying workloads that are specific to Consolidated Database Replay:

- Defining Replay Schedules
- Remapping Connections for Consolidated Database Replay
- Remapping Users for Consolidated Database Replay
- Preparing for Consolidated Database Replay
- Replaying Individual Workloads

Defining Replay Schedules

A **replay schedule** adds one or multiple workload captures to a consolidated replay and specifies the order in which the captures will start during replay. A replay schedule must be created before a consolidated replay can be initialized. Multiple replay schedules can be defined for a consolidated replay. During replay initialization, you can select from any of the existing replay schedules.
Replay schedules perform two types of operation:

- Adding Workload Captures
- Adding Schedule Orders

> **See Also:**
>
> "Defining Replay Schedules Using APIs"

Adding Workload Captures

The first type of operation performed by a replay schedule is to add the participating workload captures to a replay.

When a workload capture is added to a replay schedule, a unique number is returned to identify the workload capture. A workload capture can be added to a replay schedule more than once, as it will be assigned a different capture number each time it is added. The replay schedule will point to the same capture directory each time to avoid a waste of disk space by copying the capture each time it is added.

> **See Also:**
>
> "Adding Workload Captures to Replay Schedules Using APIs"

Adding Schedule Orders

The second type of operation performed by a replay schedule is to add schedule orders that specify the order in which the participating workload captures will start during replay.

A **schedule order** defines an order between the start of two workload captures that have been added to the replay schedule. Multiple schedule orders can be added to a replay schedule. For example, assume that a replay schedule has three workload captures added. One schedule order can be added to specify that Capture 2 must wait for Capture 1 to complete before starting. Another schedule order can be added to specify that Capture 3 must wait for Capture 1 to complete before starting. In this case, both Capture 2 and Capture 3 must wait for Capture 1 to complete before starting.

It is possible for a replay schedule to not contain any schedule orders. In this case, all participating workload captures in the replay schedule will start to replay simultaneously when the consolidated replay begins.

> **See Also:**
>
> "Adding Schedule Orders to Replay Schedules Using APIs"

Remapping Connections for Consolidated Database Replay

As in the case with replaying a single database workload using Database Replay, captured connection strings used to connect to the production system need to be remapped to the replay system. See "Connection Remapping" for more information.
For Consolidated Database Replay, you need to remap captured connection strings from multiple workload captures to different connection strings during replay.

> **See Also:**
>
> "Remapping Connection Using APIs"

Remapping Users for Consolidated Database Replay

As in the case with replaying a single database workload using Database Replay, usernames of database users and schemas used to connect to the production system can be remapped during replay. See "User Remapping" for more information.
For Consolidated Database Replay, you can choose to remap the captured users from multiple workload captures to different users or schemas during replay.

> **See Also:**
>
> "Remapping Users Using APIs"

Preparing for Consolidated Database Replay

As is the case with replaying a single database workload using Database Replay, replay options are defined during preparation of a replay. See "Specifying Replay Options" for more information.
For Consolidated Database Replay, all participating workload captures in a consolidated replay use the same replay options during replay that are defined during replay preparation.

> **See Also:**
>
> "Preparing for Consolidated Database Replay Using APIs"

Replaying Individual Workloads

It is recommended that each of the participating workloads be replayed individually before replaying the consolidated workload. See "Replaying a Database Workload " for more information.
The individual replays can establish a baseline performance for each workload capture and be used to analyze the performance of the consolidated replay.

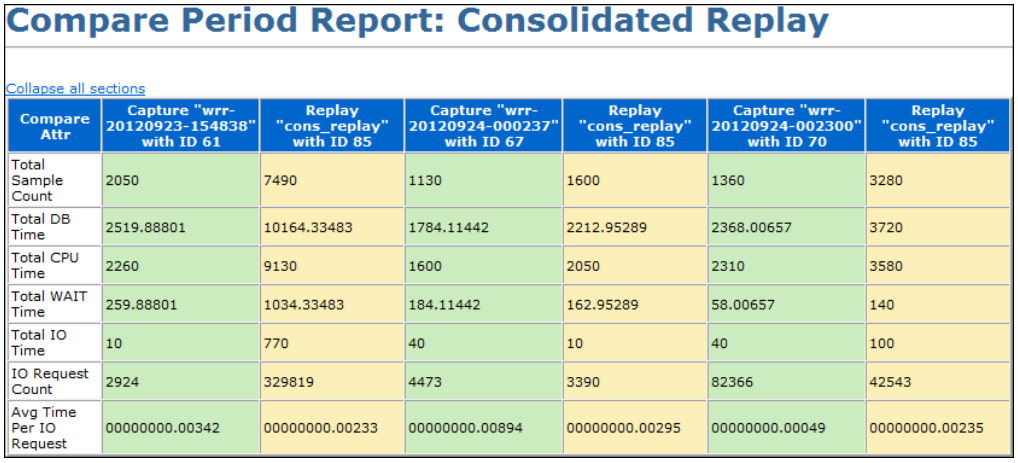
Reporting and Analysis for Consolidated Database Replay

Reporting and analysis for Consolidated Database Replay is performed using the replay compare period report. See "Using Replay Compare Period Reports" for more information.
The replay compare period report for Consolidated Database Replay identifies the Active Session History (ASH) data for each individual workload capture and compares the ASH data from the workload capture to the filtered ASH data from the consolidated replay. Use this report to compare replays of the same consolidated workload capture.

The replay compare period report for Consolidated Database Replay treats the consolidated replay as multiple Capture vs. Replay comparisons. The summary section of the report contains a table that summarizes all individual Capture vs. Replay comparisons. Review the information in this section to gain a general understanding of how the consolidated replay ran.

Figure 15-2 shows the summary section of a sample replay compare period report for Consolidated Database Replay.

Figure 15-2 Compare Period Report: Consolidated Replay



Compare Period Report: Consolidated Replay

Collapse all sections

Compare Attr	Capture "wrr-20120923-154838" with ID 61	Replay "cons_replay" with ID 85	Capture "wrr-20120924-000237" with ID 67	Replay "cons_replay" with ID 85	Capture "wrr-20120924-002300" with ID 70	Replay "cons_replay" with ID 85
Total Sample Count	2050	7490	1130	1600	1360	3280
Total DB Time	2519.88801	10164.33483	1784.11442	2212.95289	2368.00657	3720
Total CPU Time	2260	9130	1600	2050	2310	3580
Total WAIT Time	259.88801	1034.33483	184.11442	162.95289	58.00657	140
Total IO Time	10	770	40	10	40	100
IO Request Count	2924	329819	4473	3390	82366	42543
Avg Time Per IO Request	00000000.00342	00000000.00233	00000000.00894	00000000.00295	00000000.00049	00000000.00235

The rest of the sections in the report resemble the ASH Data Comparison section of the replay compare period report and are formed by joining all Capture vs. Replay reports in the consolidated replay. For a description of this section, see "ASH Data Comparison".

Using Consolidated Database Replay with Enterprise Manager

This section describes how to use Consolidated Database Replay with Enterprise Manager.

The primary tool for replaying consolidated database workloads is Oracle Enterprise Manager. If Oracle Enterprise Manager is unavailable, you can also replay consolidated database workloads using APIs, as described in "Using Consolidated Database Replay with APIs".

The process for replaying a consolidated database workload is nearly identical to that of replaying a single database workload. The differences are documented in the procedures for single replays in the following sections:

- "Creating a Database Replay Task" in Preprocessing a Database Workload
- "Preprocessing the Workload and Deploying the Replay Clients" in Preprocessing a Database Workload
- "Replaying a Database Workload Using Enterprise Manager" in Replaying a Database Workload

The following list provides a summary of the differences between replaying a consolidated database workload versus replaying a single database workload:

- When creating a replay task, you need to select two or more captured workloads from the Select Captures table in the Create Task page.
- The Preprocess Captured Workload: Copy Workload step of the wizard has more than one choice for the Capture Name drop-down, so you may need to enter multiple credentials for the current location of the workload directory.
- The Preprocess Captured Workload: Select Directory step of the wizard does not display a Capture Summary as it does for single replays.
- The Replay Workload: Copy Workload step of the wizard has more than one choice for the Capture Name drop-down, so you may need to enter multiple credentials for the current location of the workload directory.

- The Replay Workload: Select Directory step of the wizard does not display a Capture Summary as it does for single replays.
- The Replay Workload: Initialize Options step of the wizard does not display the Identify Source section.
- The Replay Workload: Customize Options step of the wizard has more than one choice for the Capture Name drop-down in the Connection Mappings tab, so you can remap connections for each captured workload. The option to use a single connect descriptor or net service name is not available.

Using Consolidated Database Replay with APIs

This section describes how to create and replay consolidated workloads using the `DBMS_WORKLOAD_REPLAY` package. You can also create and replay consolidated workloads using Oracle Enterprise Manager, as described in "Using Consolidated Database Replay with Enterprise Manager".
Creating and replay a consolidated workload using APIs is a multi-step process that involves:

- Generating Capture Subsets Using APIs
- Setting the Consolidated Replay Directory Using APIs
- Defining Replay Schedules Using APIs
- Running Consolidated Database Replay Using APIs

> **See Also:**
>
> *Oracle Database PL/SQL Packages and Types Reference* for information about the `DBMS_WORKLOAD_REPLAY` package

Generating Capture Subsets Using APIs

This section describes how to generate capture subsets from existing workload captures using the `DBMS_WORKLOAD_REPLAY` package. For information about capture subsets, see "Capture Subsets".

To generate a capture subset from existing workload captures:

1. Use the `GENERATE_CAPTURE_SUBSET` procedure:

```
DBMS_WORKLOAD_REPLAY.GENERATE_CAPTURE_SUBSET (
   input_capture_dir        IN VARCHAR2,
   output_capture_dir       IN VARCHAR2,
   new_capture_name         IN VARCHAR2,
   begin_time               IN NUMBER,
   begin_include_incomplete IN BOOLEAN   DEFAULT TRUE,
   end_time                 IN NUMBER,
   end_include_incomplete   IN BOOLEAN   DEFAULT FALSE,
   parallel_level           IN NUMBER    DEFAULT NULL);
```

2. Set the `input_capture_dir` parameter to the name of the directory object that points to an existing workload capture.
3. Set the `output_capture_dir` parameter to the name of the directory object that points to an empty directory where the new workload capture will be stored.

4. Set the `new_capture_name` parameter to the name of the new workload capture that is to be generated.
5. Set the other parameters, which are optional, as appropriate.

   For information about these parameters, see *Oracle Database PL/SQL Packages and Types Reference*.

This example shows how to create a capture subset named `peak_wkld` at directory object `peak_capdir` from an existing workload capture at directory object `rec_dir`. The capture subset includes workload from 2 hours after the start of the workload capture (or 7,200 seconds) to 3 hours after the start of the workload capture (or 10,800 seconds).

```
EXEC DBMS_WORKLOAD_REPLAY.GENERATE_CAPTURE_SUBSET ('rec_dir', 'peak_capdir',
     'peak_wkld', 7200, TRUE, 10800, FALSE, 1);
```

> **See Also:**
>
> *Oracle Database PL/SQL Packages and Types Reference* for information about the `GENERATE_CAPTURE_SUBSET` procedure

Setting the Consolidated Replay Directory Using APIs

This section describes how to set the consolidated replay directory on the test system using the `DBMS_WORKLOAD_REPLAY` package. Set the consolidated replay directory to a directory on the test system that contains the workload captures to be consolidated and replayed. For information about setting up the test system, see "Setting Up the Test System for Consolidated Database Replay".

To set the replay directory:

1. Use the `SET_CONSOLIDATED_DIRECTORY` procedure:

   ```
   DBMS_WORKLOAD_REPLAY.SET_CONSOLIDATED_DIRECTORY (
      replay_dir IN VARCHAR2);
   ```

2. Set the `replay_dir` parameter to the name of the directory object that points to the operating system directory containing the workload captures to be used for workload consolidation.

> **Tip:**
>
> The `SET_REPLAY_DIRECTORY` procedure is deprecated and replaced by the `SET_CONSOLIDATED_DIRECTORY` procedure.

This example shows how to set the replay directory to a directory object named `rep_dir`.

```
EXEC DBMS_WORKLOAD_REPLAY.SET_CONSOLIDATED_DIRECTORY ('rep_dir');
```

You can also use the `DBMS_WORKLOAD_REPLAY` package to view the current consolidated replay directory that has been set by the `SET_CONSOLIDATED_DIRECTORY` procedure.

To view the current consolidated replay directory that has been set:

- Use the `GET_REPLAY_DIRECTORY` function:

```
DBMS_WORKLOAD_REPLAY.GET_REPLAY_DIRECTORY RETURN VARCHAR2;
```

If no consolidated replay directory has been set, then the function returns `NULL`.

> **See Also:**
>
> - *Oracle Database PL/SQL Packages and Types Reference* for information about the `SET_REPLAY_DIRECTORY` procedure
> - *Oracle Database PL/SQL Packages and Types Reference* for information about the `GET_REPLAY_DIRECTORY` function

Defining Replay Schedules Using APIs

This section describes how to define replay schedules using the `DBMS_WORKLOAD_REPLAY` package. For information about replay schedules, see "Defining Replay Schedules".

Before defining replay schedules, ensure that the following prerequisites are met:

- All workload captures are preprocessed using the `PROCESS_CAPTURE` procedure on a system running the same database version as the replay system, as described in Preprocessing a Database Workload .
- All capture directories are copied to the replay directory on the replay system
- Replay directory is set using the `SET_REPLAY_DIRECTORY` procedure, as described in "Setting the Consolidated Replay Directory Using APIs".
- Database state is not in replay mode

To define replay schedules:

1. Create a new replay schedule, as described in "Creating Replay Schedules Using APIs".
2. Add workload captures to the replay schedule, as described in "Adding Workload Captures to Replay Schedules Using APIs".
3. Add schedule orders to the replay schedule, as described in "Adding Schedule Orders to Replay Schedules Using APIs".
4. Save the replay schedule, as described in "Saving Replay Schedules Using APIs".

Creating Replay Schedules Using APIs

This section describes how to create replay schedules using the `DBMS_WORKLOAD_REPLAY` package. For information about replay schedules, see "Defining Replay Schedules".

To create a replay schedule:

1. Use the `BEGIN_REPLAY_SCHEDULE` procedure:

```
DBMS_WORKLOAD_REPLAY.BEGIN_REPLAY_SCHEDULE (
   schedule_name  IN VARCHAR2);
```

2. Set the `schedule_name` parameter to the name of this replay schedule.

> **Note:**
>
> The `BEGIN_REPLAY_SCHEDULE` procedure initiates the creation of a reusable replay schedule. Only one replay schedule can be defined at a time. Calling this procedure again while a replay schedule is being defined will result in an error.

This example shows how to create a replay schedule named `peak_schedule`.

```
EXEC DBMS_WORKLOAD_REPLAY.BEGIN_REPLAY_SCHEDULE ('peak_schedule');
```

> **See Also:**
>
> *Oracle Database PL/SQL Packages and Types Reference* for information about the `BEGIN_REPLAY_SCHEDULE` procedure

Adding Workload Captures to Replay Schedules Using APIs

This section describes how to add workload captures to and remove workload captures from replay schedules using the `DBMS_WORKLOAD_REPLAY` package. For information about adding workload captures to replay schedules, see "Adding Workload Captures".

Before adding workload captures to a replay schedule, ensure that the following prerequisite is met:

- A replay schedule to which the workload captures are to be added is created.

  For information about creating a replay schedule, see "Creating Replay Schedules Using APIs".

To add workload captures to a replay schedule:

1. Use the `ADD_CAPTURE` function:

```
DBMS_WORKLOAD_REPLAY.ADD_CAPTURE (
   capture_dir_name    IN VARCHAR2,
   start_delay_seconds IN NUMBER  DEFAULT 0,
   stop_replay         IN BOOLEAN DEFAULT FALSE,
   take_begin_snapshot IN BOOLEAN DEFAULT FALSE,
   take_end_snapshot   IN BOOLEAN DEFAULT FALSE,
   query_only          IN BOOLEAN DEFAULT FALSE)
RETURN NUMBER;

DBMS_WORKLOAD_REPLAY.ADD_CAPTURE (
   capture_dir_name    IN VARCHAR2,
   start_delay_seconds IN NUMBER,
   stop_replay         IN VARCHAR2,
   take_begin_snapshot IN VARCHAR2 DEFAULT 'N',
   take_end_snapshot   IN VARCHAR2 DEFAULT 'N',
   query_only          IN VARCHAR2 DEFAULT 'N')
RETURN NUMBER;
```

This function returns an unique identifier that identifies the workload capture in this replay schedule.

> **See:**
>
> "About Query-Only Database Replay" for information about query-only database replays.

> **Note:**
>
> Query-only database replays are meant to be used and executed in test environments only.
>
> - Do not use query-only database replays on production systems.
> - Divergence is expected during query-only database replays.

2. Set the `capture_dir_name` parameter to the name of the directory object that points to the workload capture under the top-level replay directory.

   The directory must contain a valid workload capture that is preprocessed on a system running the same database version as the replay system.

3. Set the other parameters, which are optional, as appropriate.

   For information about these parameters, see *Oracle Database PL/SQL Packages and Types Reference*.

The following example shows how to add a workload capture named `peak_wkld` to a replay schedule by using the `ADD_CAPTURE` function in a `SELECT` statement.

```
SELECT DBMS_WORKLOAD_REPLAY.ADD_CAPTURE ('peak_wkld')
  FROM dual;
```

You can also use the `DBMS_WORKLOAD_REPLAY` package to remove workload captures from a replay schedule.

To remove workload captures from a replay schedule:

1. Use the `REMOVE_CAPTURE` procedure:

   ```
   DBMS_WORKLOAD_REPLAY.REMOVE_CAPTURE (
      schedule_capture_number IN NUMBER);
   ```

2. Set the `schedule_capture_number` parameter to the unique identifier that identifies the workload capture in this replay schedule.

   The unique identifier is the same identifier that was returned by the `ADD_CAPTURE` function when the workload capture was added to the replay schedule.

> **See Also:**
>
> - *Oracle Database PL/SQL Packages and Types Reference* for information about the `ADD_CAPTURE` function
> - *Oracle Database PL/SQL Packages and Types Reference* for information about the `REMOVE_CAPTURE` procedure

Adding Schedule Orders to Replay Schedules Using APIs

This section describes how to add schedule orders to and remove schedule orders from replay schedules using the `DBMS_WORKLOAD_REPLAY` package. For information about adding schedule orders to replay schedules, see "Adding Schedule Orders".

Before adding schedule orders to a replay schedule, ensure that the following prerequisites are met:

- A replay schedule to which the schedule orders are to be added is created.

  For information about creating a replay schedule, see "Creating Replay Schedules Using APIs".
- All workload captures participating in the schedule order are added to the replay schedule.

  For information about adding workload captures to a replay schedule, see "Adding Workload Captures to Replay Schedules Using APIs".

> **Note:**
>
> Adding schedule orders to a replay schedule is optional. If you do not add a schedule order to a replay schedule, then all workload captures added to the replay schedule will start to replay simultaneously when the consolidated replay begins.

To add schedule orders to a replay schedule:

1. Use the `ADD_SCHEDULE_ORDERING` function:

   ```
   DBMS_WORKLOAD_REPLAY.ADD_SCHEDULE_ORDERING (
      schedule_capture_id IN NUMBER,
      waitfor_capture_id IN NUMBER)
   RETURN NUMBER;
   ```

   This function adds a schedule order between two workload captures that have been added to the replay schedule. If a schedule order cannot be added, it returns a nonzero error code.
2. Set the `schedule_capture_id` parameter to the workload capture that you want to wait in this schedule order.
3. Set the `wait_for_capture_id` parameter to the workload capture that you want to be completed before the other workload capture can start in this schedule order.

You can also use the `DBMS_WORKLOAD_REPLAY` package to remove schedule orders from a replay schedule.

To remove schedule orders from a replay schedule:

1. Use the `REMOVE_SCHEDULE_ORDERING` procedure:

```
DBMS_WORKLOAD_REPLAY.REMOVE_SCHEDULE ORDERING (
   schedule_capture_id IN VARCHAR2,
   wait_for_capture_id IN VARCHAR2);
```

2. Set the `schedule_capture_id` parameter to the workload capture waiting in this schedule order.
3. Set the `wait_for_capture_id` parameter to the workload capture that needs to be completed before the other workload capture can start in this schedule order.

To view schedule orders:

- Use the `DBA_WORKLOAD_SCHEDULE_ORDERING` view.

> **See Also:**
>
> - *Oracle Database PL/SQL Packages and Types Reference* for information about the `ADD_SCHEDULE_ORDERING` function
> - *Oracle Database PL/SQL Packages and Types Reference* for information about the `REMOVE_SCHEDULE_ORDERING` procedure
> - *Oracle Database Reference* for information about the `DBA_WORKLOAD_SCHEDULE_ORDERING` view

Saving Replay Schedules Using APIs

This section describes how to save replay schedules that been defined using the `DBMS_WORKLOAD_REPLAY` package.

Before saving a replay schedule, ensure that the following prerequisites are met:

- A replay schedule that will be saved is created.

  For information about creating a replay schedule, see "Creating Replay Schedules Using APIs".
- All workload captures participating in the schedule order are added to the replay schedule.

  For information about adding workload captures to a replay schedule, see "Adding Workload Captures to Replay Schedules Using APIs".
- Any schedule orders that you want to use are added to the replay schedule (this step is optional).

  For information about adding schedule orders to a replay schedule, see "Adding Schedule Orders to Replay Schedules Using APIs".

To save a replay schedule:

- Use the `END_REPLAY_SCHEDULE` procedure:

```
DBMS_WORKLOAD_REPLAY.END_REPLAY_SCHEDULE;
```

This procedure completes the creation of a replay schedule. The replay schedule is saved and associated with the replay directory. Once a replay schedule is saved, you can use it for a consolidated replay.

To view replay schedules:

- Use the `DBA_WORKLOAD_REPLAY_SCHEDULES` view.

> **See Also:**
>
> - *Oracle Database PL/SQL Packages and Types Reference* for information about the `END_REPLAY_SCHEDULE` procedure
> - *Oracle Database Reference* for information about the `DBA_WORKLOAD_REPLAY_SCHEDULES` view

Running Consolidated Database Replay Using APIs

This section describes how to run Consolidated Database Replay using the `DBMS_WORKLOAD_REPLAY` package. For information about consolidated replay, see "Replaying Database Workloads for Consolidated Database Replay".

Before running Consolidated Database Replay, ensure that the following prerequisites are met:

- All workload captures are preprocessed using the `PROCESS_CAPTURE` procedure on a system running the same database version as the replay system, as described in Preprocessing a Database Workload .
- All capture directories are copied to the replay directory on the replay system
- Replay directory is set using the `SET_REPLAY_DIRECTORY` procedure, as described in "Setting the Consolidated Replay Directory Using APIs".
- Database is logically restored to the same application state as that of all the capture systems at the start time of all workload captures.

To run Consolidated Database Replay:

1. Initialize the replay data, as described in "Initializing Consolidated Database Replay Using APIs".
2. Remap connections strings, as described in "Remapping Connection Using APIs".
3. Remap users, as described in "Remapping Users Using APIs".

   Remapping users is optional.
4. Prepare the consolidated replay, as described in "Preparing for Consolidated Database Replay Using APIs".
5. Set up and start the replay clients, as described in "Setting Up Replay Clients".
6. Start the consolidated replay, as described in "Starting Consolidated Database Replay Using APIs".
7. Generate reports and perform analysis, as described in "Reporting and Analysis for Consolidated Database Replay".

Initializing Consolidated Database Replay Using APIs

This section describes how to initialize the replay data for a consolidated replay using the `DBMS_WORKLOAD_REPLAY` package.

Initializing the replay data performs the following operations:

- Puts the database state in initialized mode for the replay of a consolidated workload.
- Points to the replay directory that contains all workload captures participating in the replay schedule.
- Loads the necessary metadata into tables required for replay.

  For example, captured connection strings are loaded into a table where they can be remapped for replay.

To initialize Consolidated Database Replay:

1. Use the `INITIALIZE_CONSOLIDATED_REPLAY` procedure:

```
DBMS_WORKLOAD_REPLAY.INITIALIZE_CONSOLIDATED_REPLAY (
   replay_name    IN VARCHAR2,
   schedule_name  IN VARCHAR2,
   plsql_mode     IN VARCHAR2 DEFAULT 'TOP_LEVEL');
```

2. Set the `replay_name` parameter to the name of the consolidated replay.
3. Set the `schedule_name` parameter to the name of the replay schedule to use.

   The `schedule_name` parameter is the name of the replay schedule used during its creation, as described in "Creating Replay Schedules Using APIs".

The optional `plsql_mode` parameter specifies the PL/SQL replay mode.

These two values can be set for the `plsql_mode` parameter:

- `top_level`: Only top-level PL/SQL calls. This is the default value.
- `extended`: SQL executed inside PL/SQL or top-level PL/SQL if there is no SQL recorded inside. All captures must have been done in the `'extended'` PL/SQL mode. Non-PL/SQL calls will be replayed in the usual manner.

The following example shows how to initialize a consolidated replay named `peak_replay` using the replay schedule named `peak_schedule`.

```
EXEC DBMS_WORKLOAD_REPLAY.INITIALIZE_CONSOLIDATED_REPLAY ('peak_replay',
     'peak_schedule');
```

> **See Also:**
>
> *Oracle Database PL/SQL Packages and Types Reference* for information about the `INITIALIZE_CONSOLIDATED_REPLAY` procedure

Remapping Connection Using APIs

This section describes how to remap connection strings for a consolidated replay using the `DBMS_WORKLOAD_REPLAY` package. For information about connection remapping, see "Remapping Connections for Consolidated Database Replay".

To remap connection strings:

1. Use the `REMAP_CONNECTION` procedure:

```
DBMS_WORKLOAD_REPLAY.REMAP_CONNECTION (
   schedule_cap_id   IN NUMBER,
   connection_id     IN NUMBER,
   replay_connection IN VARCHAR2);
```

   This procedure remaps the captured connection to a new connection string for all participating workload captures in the replay schedule.

2. Set the `schedule_capture_id` parameter to a participating workload capture in the current replay schedule.

   The `schedule_capture_id` parameter is the unique identifier returned when adding the workload capture to the replay schedule, as described in "Adding Workload Captures to Replay Schedules Using APIs".

3. Set the `connection_id` parameter to the connection to be remapped.

   The `connection_id` parameter is generated when replay data is initialized and corresponds to a connection from the workload capture.

4. Set the `replay_connection` parameter to the new connection string that will be used during replay.

> **See Also:**
>
> *Oracle Database PL/SQL Packages and Types Reference* for information about the `REMAP_CONNECTION` procedure

Remapping Users Using APIs

This section describes how to remap users for a consolidated replay using the `DBMS_WORKLOAD_REPLAY` package. For information about remapping users, see "Remapping Users for Consolidated Database Replay".

Before remapping users, ensure that the following prerequisites are met:

- Replay data is initialized, as described in "Initializing Consolidated Database Replay Using APIs".
- The database state is not in replay mode.

To remap users:

1. Use the `SET_USER_MAPPING` procedure:

```
DBMS_WORKLOAD_REPLAY.SET_USER_MAPPING (
   schedule_cap_id IN NUMBER,
```

```
    capture_user    IN VARCHAR2,
    replay_user     IN VARCHAR2);
```

2. Set the `schedule_capture_id` parameter to a participating workload capture in the current replay schedule.

   The `schedule_capture_id` parameter is the unique identifier returned when adding the workload capture to the replay schedule, as described in "Adding Workload Captures to Replay Schedules Using APIs".

3. Set the `capture_user` parameter to the username of the user or schema captured during the time of the workload capture.

4. Set the `replay_user` parameter to the username of a new user or schema to which the captured user is remapped during replay.

   If this parameter is set to `NULL`, then the mapping is disabled.

This example shows how to remap the `PROD` user used during capture to the `TEST` user during replay for the workload capture identified as `1001`.

```
EXEC DBMS_WORKLOAD_REPLAY.SET_USER_MAPPING (1001, 'PROD', 'TEST');
```

> **See Also:**
>
> *Oracle Database PL/SQL Packages and Types Reference* for information about the `SET_USER_MAPPING` procedure

Preparing for Consolidated Database Replay Using APIs

This section describes how to prepare a consolidated replay using the `DBMS_WORKLOAD_REPLAY` package. For information about preparing consolidated replays, see "Preparing for Consolidated Database Replay".

Before preparing a consolidated replay, ensure that the following prerequisites are met:

- Replay data is initialized, as described in "Initializing Consolidated Database Replay Using APIs".
- Captured connections are remapped, as described in "Remapping Connection Using APIs".
- Users are mapped, as described in "Remapping Users Using APIs".

  Remapping users is optional. However, if you are planning to remap users during replay, then it must be completed before preparing the consolidated replay.

Preparing a consolidated replay performs the following operations:

- Specifies the replay options, such as synchronization mode, session connection rate, and session request rate.
- Puts the database state in replay mode.
- Enables the start of replay clients.

To prepare a consolidated replay:

- Use the `PREPARE_CONSOLIDATED_REPLAY` procedure:

```
DBMS_WORKLOAD_REPLAY.PREPARE_CONSOLIDATED_REPLAY (
   synchronization         IN VARCHAR2 DEFAULT 'SCN',
   connect_time_scale      IN NUMBER   DEFAULT 100,
   think_time_scale        IN NUMBER   DEFAULT 100,
   think_time_auto_correct IN BOOLEAN  DEFAULT TRUE,
   capture_sts             IN BOOLEAN  DEFAULT FALSE,
   sts_cap_interval        IN NUMBER   DEFAULT 300);
```

For information about these parameters and how to set them, see "Specifying Replay Options".

> **See Also:**
>
> *Oracle Database PL/SQL Packages and Types Reference* for information about the `PREPARE_CONSOLIDATED_REPLAY` procedure

Starting Consolidated Database Replay Using APIs

This section describes how to start a consolidated replay using the `DBMS_WORKLOAD_REPLAY` package.

Before starting a consolidated replay, ensure that the following prerequisites are met:

- The consolidated replay is prepared, as described in "Preparing for Consolidated Database Replay Using APIs".
- An adequate number of replay clients are started.

  For information about setting up and starting replay clients, see "Setting Up Replay Clients".

To start a consolidated replay:

- Use the `START_CONSOLIDATED_REPLAY` procedure:

```
DBMS_WORKLOAD_REPLAY.START_CONSOLIDATED_REPLAY;
```

> **See Also:**
>
> *Oracle Database PL/SQL Packages and Types Reference* for information about the `START_CONSOLIDATED_REPLAY` procedure

About Query-Only Database Replay

In a query-only database replay, only the read-only queries of a workload capture are replayed. In other words, in a query-only replay, only `SELECT` statements are sent to the server at replay time. No DML statements are executed during a query-only replay, and the replay does not make any changes to user schemas or data.

> **Note:**
>
> A query-only database replay can be performed with Consolidated Database Replay only.

> **Note:**
>
> Query-only database replays are meant to be used and executed in test environments only.
>
> - Do not use query-only database replays on production systems.
> - Divergence is expected during query-only database replays.

Use Cases for Query-Only Database Replay

You can use query-only database replay to warm up the database buffer cache and to find regressions. For example:

- To warm up the database buffer cache

  In some cases, a workload is captured when the database buffer cache is warm (data blocks are already in the buffer cache). However, when you replay that workload on the test system, the buffer cache will not be warm, and the data blocks will need to be loaded from disk initially. This may make the replay duration longer than the capture duration, and increase the database time.

  To avoid having to warm up the buffer cache, you can perform a query-only replay and then perform the read/write replay without restarting the database and without flushing the buffer cache. Note that you do not have to restart the database after a query-only replay because a query-only replay is read-only.

- To find regressions

  A query-only replay is a good and easy way to find regressions from the read-only part of the workload with concurrency. The read-only part includes `SELECT` (not `SELECT...FOR UPDATE`) statements, PL/SQL without DMLs and DDLs, LOB reads, and so on. It is typically the main part of the workload capture.

Performing a Query-Only Database Replay

You can perform a query-only database replay.

To perform a query-only database replay, follow the instructions in "Using Consolidated Database Replay with APIs". When you use the `ADD_CAPTURE` function to add workload captures to the replay schedule as described in "Adding Workload Captures to Replay Schedules Using APIs", set the `query_only` parameter to `Y`.

Example: Replaying a Consolidated Workload with APIs

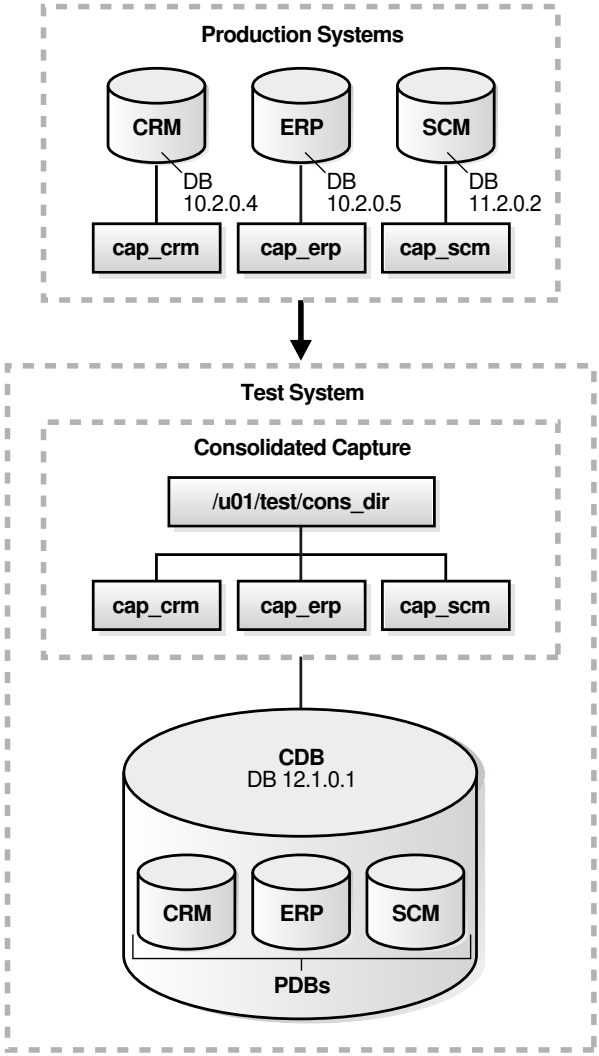
This section assumes a scenario where workloads from three separate production systems running different versions of Oracle Database on various operating systems are being consolidated.

This scenario uses the following assumptions:

- The first workload to be consolidated is captured from the CRM system, which is running Oracle Database 10g Release 2 (release 10.2.0.4) on a Solaris server.
- The second workload to be consolidated is captured from the ERP system, which is running Oracle Database 10g Release 2 (release 10.2.0.5) on a Linux server.
- The third workload to be consolidated is captured from the SCM system, which is running Oracle Database 11*g* Release 2 (release 11.2.0.2) on a Solaris server.
- The test system is set up as a multitenant container database (CDB) running Oracle Database 12*c* Release 1 (release 12.1.0.1).
- The CDB contains three PDBs created from the CRM, ERP, and SCM systems.
- Each PDB contained within the CDB is restored to the same application data state as the CRM, ERP, and SCM systems at the capture start time.

Figure 15-3 illustrates this scenario.

Figure 15-3 Scenario for Consolidating Three Workloads



To consolidate the workloads and replay the consolidated workload in this scenario:

1. On the test system, preprocess the individual workload captures into separate directories:
   - For the CRM workload:
     a. Create a directory object:

     ```
     CREATE OR REPLACE DIRECTORY crm AS '/u01/test/cap_crm';
     ```

     b. Ensure that the captured workload from the CRM system is stored in this directory.
     c. Preprocess the workload:

     ```
     EXEC DBMS_WORKLOAD_REPLAY.PROCESS_CAPTURE ('CRM');
     ```

   - For the ERP workload:
     a. Create a directory object:

     ```
     CREATE OR REPLACE DIRECTORY erp AS '/u01/test/cap_erp';
     ```

     b. Ensure that the captured workload from the ERP system is stored in this directory.
     c. Preprocess the workload:

```
EXEC DBMS_WORKLOAD_REPLAY.PROCESS_CAPTURE ('ERP');
```

- For the SCM workload:

  a. Create a directory object:

  ```
  CREATE OR REPLACE DIRECTORY scm AS '/u01/test/cap_scm';
  ```

  b. Ensure that the captured workload from the SCM system is stored in this directory.

  c. Preprocess the workload:

  ```
  EXEC DBMS_WORKLOAD_REPLAY.PROCESS_CAPTURE ('SCM');
  ```

2. Create a root directory to store the preprocessed workloads:

```
mkdir '/u01/test/cons_dir';
CREATE OR REPLACE DIRECTORY cons_workload AS '/u01/test/cons_dir';
```

3. Copy each preprocessed workload directory into the root directory:

```
cp -r /u01/test/cap_crm /u01/test/cons_dir
cp -r /u01/test/cap_erp /u01/test/cons_dir
cp -r /u01/test/cap_scm /u01/test/cons_dir
```

4. For each workload, create a directory object using the new operating system directory path:

```
CREATE OR REPLACE DIRECTORY crm AS '/u01/test/cons_dir/cap_crm';
CREATE OR REPLACE DIRECTORY erp AS '/u01/test/cons_dir/cap_erp';
CREATE OR REPLACE DIRECTORY scm AS '/u01/test/cons_dir/cap_scm';
```

5. Set the replay directory to the root directory previously created in Step 2:

```
EXEC DBMS_WORKLOAD_REPLAY.SET_REPLAY_DIRECTORY ('CONS_WORKLOAD');
```

6. Create a replay schedule and add the workload captures:

```
EXEC DBMS_WORKLOAD_REPLAY.BEGIN_REPLAY_SCHEDULE ('CONS_SCHEDULE');
SELECT DBMS_WORKLOAD_REPLAY.ADD_CAPTURE ('CRM') FROM dual;
SELECT DBMS_WORKLOAD_REPLAY.ADD_CAPTURE ('ERP') FROM dual;
SELECT DBMS_WORKLOAD_REPLAY.ADD_CAPTURE ('SCM') FROM dual;
EXEC DBMS_WORKLOAD_REPLAY.END_REPLAY_SCHEDULE;
```

7. Initialize the consolidated replay:

```
EXEC DBMS_WORKLOAD_REPLAY.INITIALIZE_CONSOLIDATED_REPLAY ('CONS_REPLAY',
     'CONS_SCHEDULE');
```

8. Remap connections:

   a. Query the `DBA_WORKLOAD_CONNECTION_MAP` view for the connection mapping information:

   ```
   SELECT schedule_cap_id, conn_id, capture_conn, replay_conn
     FROM dba_workload_connection_map;
   ```

   b. Remap the connections:

   ```
   EXEC DBMS_WORKLOAD_REPLAY.REMAP_CONNECTION (schedule_cap_id => 1,
        conn_id => 1, replay_connection => 'CRM');
   EXEC DBMS_WORKLOAD_REPLAY.REMAP_CONNECTION (schedule_cap_id => 2,
        conn_id => 1, replay_connection => 'ERP');
   EXEC DBMS_WORKLOAD_REPLAY.REMAP_CONNECTION (schedule_cap_id => 3,
        conn_id => 1, replay_connection => 'SCM');
   ```

   The `replay_connection` parameter represents the services that are defined on the test system.

   c. Verify the connection remappings:

```
SELECT schedule_cap_id, conn_id, capture_conn, replay_conn
  FROM dba_workload_connection_map;
```

9. Prepare the consolidated replay:

```
EXEC DBMS_WORKLOAD_REPLAY.PREPARE_CONSOLIDATED_REPLAY (
     synchronization => 'OBJECT_ID');
```

10. Start replay clients:

    a. Estimate the number of replay clients that are required:

```
wrc mode=calibrate replaydir=/u01/test/cons_dir/cap_crm
wrc mode=calibrate replaydir=/u01/test/cons_dir/cap_erp
wrc mode=calibrate replaydir=/u01/test/cons_dir/cap_scm
```

    b. Add the output to determine the number of replay clients required.

    You will need to start at least one replay client per workload capture contained in the consolidated workload.

    c. Start the required number of replay clients by repeating this command:

```
wrc username/password mode=replay replaydir=/u01/test/cons_dir
```

    The `replaydir` parameter is set to the root directory in which the workload captures are stored.

11. Start the consolidated replay:

```
EXEC DBMS_WORKLOAD_REPLAY.START_CONSOLIDATED_REPLAY;
```

See Also:

- *Oracle Database Administrator's Guide* for information about configuring a CDB
- *Oracle Database Administrator's Guide* for information about creating PDBs

16

Using Workload Scale-Up

This chapter describes using various workload scale-up techniques with Consolidated Database Replay. It contains the following sections:

- Overview of Workload Scale-Up
- Using Time Shifting
- Using Workload Folding
- Using Schema Remapping

Overview of Workload Scale-Up

Consolidated Database Replay enables you to replay multiple workloads captured from one or multiple systems concurrently. During the replay, every workload capture that is consolidated will start to replay when the consolidated replay begins. Depending on the use case, you can employ various workload scale-up techniques when using Consolidated Database Replay. This section describes the following workload scale-up techniques:

- About Time Shifting
- About Workload Folding
- About Schema Remapping

> **See Also:**
>
> "Use Cases for Consolidated Database Replay" for information about typical use cases for Consolidated Database Replay

About Time Shifting

Database Replay enables you to perform time shifting when replaying captured workloads. This technique is useful in cases where you want to conduct stress testing on a system by adding workloads to an existing workload capture and replaying them together.

For example, assume that there are three workloads captured from three applications: Sales, CRM, and DW. In order to perform stress testing, you can align the peaks of these workload captures and replay them together using Consolidated Database Replay.

> **See Also:**
>
> - "Using Time Shifting" for information about using time shifting
> - "Stress Testing" for information about using Consolidated Database Replay for stress testing

About Workload Folding

Database Replay enables you to perform scale-up testing by folding an existing workload capture. For example, assume that a workload was captured from 2 a.m. to 8 p.m. You can use Database Replay to fold the original workload into three capture subsets: one from 2 a.m. to 8 a.m., a second from 8 a.m. to 2 p.m., and a third from 2 p.m. to 8 p.m. By replaying the three capture subsets together, you can fold the original capture and triple the workload during replay to perform scale-up testing.

See Also:

- "Using Workload Folding" for information about using workload folding
- "Capture Subsets" for information about capture subsets
- "Scale-Up Testing" for information about using Consolidated Database Replay for scale-up testing

About Schema Remapping

Database Replay enables you to perform scale-up testing by remapping database schemas. This technique is useful in cases when you are deploying multiple instances of the same application—such as a multi-tenet application—or adding a new geographical area to an existing application.

For example, assume that a single workload exists for a Sales application. To perform scale-up testing and identify possible host bottlenecks, set up the test system with multiple schemas from the Sales schema.

See Also:

- "Using Schema Remapping" for information about using schema remapping
- "Scale-Up Testing" for information about using Consolidated Database Replay for scale-up testing

Using Time Shifting

This section describes how to use time shifting with Consolidated Database Replay, and assumes a scenario where you want to use time shifting to align the peaks of workloads captured from three applications and replay them simultaneously. The scenario demonstrates how to use time shifting for stress testing. For more information about time shifting, see "About Time Shifting".
This scenario uses the following assumptions:

- The first workload is captured from the Sales application.
- The second workload is captured from the CRM application and its peak time occurs 1 hour before that of the Sales workload.

- The third workload is captured from the DW application and its peak time occurs 30 minutes before that of the Sales workload.
- To align the peaks of these workloads, time shifting is performed by adding a delay of one hour to the CRM workload and a delay of 30 minutes to the DW workload during replay.

To perform time shifting in this scenario:

1. On the replay system which will undergo stress testing, create a directory object for the root directory where the captured workloads are stored:

   ```
   CREATE [OR REPLACE] DIRECTORY cons_dir AS '/u01/test/cons_dir';
   ```

2. Preprocess the individual workload captures into separate directories:
   - For the Sales workload:
     a. Create a directory object:

        ```
        CREATE OR REPLACE DIRECTORY sales AS '/u01/test/cons_dir/cap_sales';
        ```

     b. Ensure that the captured workload from the Sales application is stored in this directory.
     c. Preprocess the workload:

        ```
        EXEC DBMS_WORKLOAD_REPLAY.PROCESS_CAPTURE ('SALES');
        ```

   - For the CRM workload:
     a. Create a directory object:

        ```
        CREATE OR REPLACE DIRECTORY crm AS '/u01/test/cons_dir/cap_crm';
        ```

     b. Ensure that the captured workload from the CRM application is stored in this directory.
     c. Preprocess the workload:

        ```
        EXEC DBMS_WORKLOAD_REPLAY.PROCESS_CAPTURE ('CRM');
        ```

   - For the DW workload:
     a. Create a directory object:

        ```
        CREATE OR REPLACE DIRECTORY DW AS '/u01/test/cons_dir/cap_dw';
        ```

     b. Ensure that the captured workload from the DW application is stored in this directory.
     c. Preprocess the workload:

        ```
        EXEC DBMS_WORKLOAD_REPLAY.PROCESS_CAPTURE ('DW');
        ```

3. Set the replay directory to the root directory:

   ```
   EXEC DBMS_WORKLOAD_REPLAY.SET_REPLAY_DIRECTORY ('CONS_DIR');
   ```

4. Create a replay schedule and add the workload captures:

   ```
   EXEC DBMS_WORKLOAD_REPLAY.BEGIN_REPLAY_SCHEDULE ('align_peaks_schedule');
   SELECT DBMS_WORKLOAD_REPLAY.ADD_CAPTURE ('SALES') FROM dual;
   SELECT DBMS_WORKLOAD_REPLAY.ADD_CAPTURE ('CRM', 3600) FROM dual;
   SELECT DBMS_WORKLOAD_REPLAY.ADD_CAPTURE ('DW', 1800) FROM dual;
   EXEC DBMS_WORKLOAD_REPLAY.END_REPLAY_SCHEDULE;
   ```

   Note that a delay of 3,600 seconds (or 1 hour) is added to the CRM workload, and a delay of 1,800 seconds (or 30 minutes) is added to the DW workload.

5. Initialize the consolidated replay:

```
EXEC DBMS_WORKLOAD_REPLAY.INITIALIZE_CONSOLIDATED_REPLAY ('align_peaks_replay',
                          'align_peaks_schedule');
```

6. Remap connections:

   a. Query the `DBA_WORKLOAD_CONNECTION_MAP` view for the connection mapping information:

   ```
   SELECT schedule_cap_id, conn_id, capture_conn, replay_conn
     FROM dba_workload_connection_map;
   ```

   b. Remap the connections:

   ```
   EXEC DBMS_WORKLOAD_REPLAY.REMAP_CONNECTION (schedule_cap_id => 1,
        conn_id => 1, replay_connection => 'inst1');
   EXEC DBMS_WORKLOAD_REPLAY.REMAP_CONNECTION (schedule_cap_id => 1,
        conn_id => 2, replay_connection => 'inst1');
   EXEC DBMS_WORKLOAD_REPLAY.REMAP_CONNECTION (schedule_cap_id => 2,
        conn_id => 1, replay_connection => 'inst2');
   EXEC DBMS_WORKLOAD_REPLAY.REMAP_CONNECTION (schedule_cap_id => 2,
        conn_id => 2, replay_connection => 'inst2');
   EXEC DBMS_WORKLOAD_REPLAY.REMAP_CONNECTION (schedule_cap_id => 3,
        conn_id => 1, replay_connection => 'inst3');
   EXEC DBMS_WORKLOAD_REPLAY.REMAP_CONNECTION (schedule_cap_id => 3,
        conn_id => 2, replay_connection => 'inst3');
   ```

   The `replay_connection` parameter represents the services that are defined on the test system.

   c. Verify the connection remappings:

   ```
   SELECT schedule_cap_id, conn_id, capture_conn, replay_conn
     FROM dba_workload_connection_map;
   ```

7. Prepare the consolidated replay:

   ```
   EXEC DBMS_WORKLOAD_REPLAY.PREPARE_CONSOLIDATED_REPLAY;
   ```

8. Start replay clients:

   a. Estimate the number of replay clients that are required:

   ```
   wrc mode=calibrate replaydir=/u01/test/cons_dir/cap_sales
   wrc mode=calibrate replaydir=/u01/test/cons_dir/cap_crm
   wrc mode=calibrate replaydir=/u01/test/cons_dir/cap_dw
   ```

   b. Add the output to determine the number of replay clients required.

   You will need to start at least one replay client per workload capture contained in the consolidated workload.

   c. Start the required number of replay clients by repeating this command:

   ```
   wrc username/password mode=replay replaydir=/u01/test/cons_dir
   ```

   The `replaydir` parameter is set to the root directory in which the workload captures are stored.

9. Start the consolidated replay:

   ```
   EXEC DBMS_WORKLOAD_REPLAY.START_CONSOLIDATED_REPLAY;
   ```

Using Workload Folding

This section describes how to use workload folding with Consolidated Database Replay, and assumes a scenario where you want to use workload folding to triple a captured workload. The

scenario demonstrates how to use workload folding for scale-up testing. For more information about workload folding, see "About Workload Folding".
This scenario uses the following assumptions:

- The original workload was captured from 2 a.m. to 8 p.m. and folded into three capture subsets.
- The first capture subset contains part of the original workload from 2 a.m. to 8 a.m.
- The second capture subset contains part of the original workload from 8 a.m. to 2 p.m.
- The third capture subset contains part of the original workload from 2 p.m. to 8 p.m.
- To triple the workload during replay, workload folding is performed by replaying the three capture subsets simultaneously.

To perform workload folding in this scenario:

1. On the replay system where you plan to perform scale-up testing, create a directory object for the root directory where the captured workloads are stored:

```
CREATE OR REPLACE DIRECTORY cons_dir AS '/u01/test/cons_dir';
```

2. Create a directory object for the directory where the original workload is stored:

```
CREATE OR REPLACE DIRECTORY cap_monday AS '/u01/test/cons_dir/cap_monday';
```

3. Create directory objects for the directories where you are planning to store the capture subsets:
   a. Create a directory object for the first capture subset:

```
CREATE OR REPLACE DIRECTORY cap_mon_2am_8am
                            AS '/u01/test/cons_dir/cap_monday_2am_8am';
```

   b. Create a directory object for the second capture subset:

```
CREATE OR REPLACE DIRECTORY cap_mon_8am_2pm
                            AS '/u01/test/cons_dir/cap_monday_8am_2pm';
```

   c. Create a directory object for the third capture subset:

```
CREATE OR REPLACE DIRECTORY cap_mon_2pm_8pm
                            AS '/u01/test/cons_dir/cap_monday_2pm_8pm';
```

4. Create the capture subsets:
   a. Generate the first capture subset for the time period from 2 a.m. to 8 a.m.:

```
EXEC DBMS_WORKLOAD_REPLAY.GENERATE_CAPTURE_SUBSET ('CAP_MONDAY',
                           'CAP_MON_2AM_8AM', 'mon_2am_8am_wkld',
                           0, TRUE, 21600, FALSE, 1);
```

   b. Generate the second capture subset for the time period from 8 a.m. to 2 p.m.:

```
EXEC DBMS_WORKLOAD_REPLAY.GENERATE_CAPTURE_SUBSET ('CAP_MONDAY',
                           'CAP_MON_8AM_2PM', 'mon_8am_2pm_wkld',
                           21600, TRUE, 43200, FALSE, 1);
```

   c. Generate the third capture subset for the time period from 2 p.m. to 8 p.m.:

```
EXEC DBMS_WORKLOAD_REPLAY.GENERATE_CAPTURE_SUBSET ('CAP_MONDAY',
                           'CAP_MON_2PM_8PM', 'mon_2pm_8pm_wkld',
                           43200, TRUE, 0, FALSE, 1);
```

5. Preprocess the capture subsets:

```
EXEC DBMS_WORKLOAD_REPLAY.PROCESS_CAPTURE ('CAP_MON_2AM_8AM');
EXEC DBMS_WORKLOAD_REPLAY.PROCESS_CAPTURE ('CAP_MON_8AM_2PM');
EXEC DBMS_WORKLOAD_REPLAY.PROCESS_CAPTURE ('CAP_MON_2PM_8PM');
```

6. Set the replay directory to the root directory:

```
EXEC DBMS_WORKLOAD_REPLAY.SET_REPLAY_DIRECTORY ('CONS_DIR');
```

7. Create a replay schedule and add the capture subsets:

```
EXEC DBMS_WORKLOAD_REPLAY.BEGIN_REPLAY_SCHEDULE ('monday_folded_schedule');
SELECT DBMS_WORKLOAD_REPLAY.ADD_CAPTURE ('CAP_MON_2AM_8AM') FROM dual;
SELECT DBMS_WORKLOAD_REPLAY.ADD_CAPTURE ('CAP_MON_8AM_2PM') FROM dual;
SELECT DBMS_WORKLOAD_REPLAY.ADD_CAPTURE ('CAP_MON_2PM_8PM') FROM dual;
EXEC DBMS_WORKLOAD_REPLAY.END_REPLAY_SCHEDULE;
```

8. Initialize the consolidated replay:

```
EXEC DBMS_WORKLOAD_REPLAY.INITIALIZE_CONSOLIDATED_REPLAY (
                          'monday_folded_replay', 'monday_folded_schedule');
```

9. Remap connections:

   a. Query the `DBA_WORKLOAD_CONNECTION_MAP` view for the connection mapping information:

```
SELECT schedule_cap_id, conn_id, capture_conn, replay_conn
  FROM dba_workload_connection_map;
```

   b. Remap the connections:

```
EXEC DBMS_WORKLOAD_REPLAY.REMAP_CONNECTION (schedule_cap_id => 1,
     conn_id => 1, replay_connection => 'inst1');
EXEC DBMS_WORKLOAD_REPLAY.REMAP_CONNECTION (schedule_cap_id => 1,
     conn_id => 2, replay_connection => 'inst1');
EXEC DBMS_WORKLOAD_REPLAY.REMAP_CONNECTION (schedule_cap_id => 2,
     conn_id => 1, replay_connection => 'inst2');
EXEC DBMS_WORKLOAD_REPLAY.REMAP_CONNECTION (schedule_cap_id => 2,
     conn_id => 2, replay_connection => 'inst2');
EXEC DBMS_WORKLOAD_REPLAY.REMAP_CONNECTION (schedule_cap_id => 3,
     conn_id => 1, replay_connection => 'inst3');
EXEC DBMS_WORKLOAD_REPLAY.REMAP_CONNECTION (schedule_cap_id => 3,
     conn_id => 2, replay_connection => 'inst3');
```

      The `replay_connection` parameter represents the services that are defined on the test system.

   c. Verify the connection remappings:

```
SELECT schedule_cap_id, conn_id, capture_conn, replay_conn
  FROM dba_workload_connection_map;
```

10. Prepare the consolidated replay:

```
EXEC DBMS_WORKLOAD_REPLAY.PREPARE_CONSOLIDATED_REPLAY;
```

11. Start replay clients:

   a. Estimate the number of replay clients that are required:

```
wrc mode=calibrate replaydir=/u01/test/cons_dir/cap_monday_2am_8am
wrc mode=calibrate replaydir=/u01/test/cons_dir/cap_monday_8am_2pm
wrc mode=calibrate replaydir=/u01/test/cons_dir/cap_monday_2pm_8pm
```

   b. Add the output to determine the number of replay clients required.

      You will need to start at least one replay client per workload capture contained in the consolidated workload.

c. Start the required number of replay clients by repeating this command:

```
wrc username/password mode=replay replaydir=/u01/test/cons_dir
```

The `replaydir` parameter is set to the root directory in which the workload captures are stored.

12. Start the consolidated replay:

```
EXEC DBMS_WORKLOAD_REPLAY.START_CONSOLIDATED_REPLAY;
```

Using Schema Remapping

This section describes how to use schema remapping with Consolidated Database Replay, and assumes a scenario where you want to use schema remapping to identify possible host bottlenecks when deploying multiple instances of an application. The scenario demonstrates how to use schema remapping for scale-up testing. For more information about schema remapping, see "About Schema Remapping".
This scenario uses the following assumptions:

- A single workload exists that is captured from the Sales application.
- To set up the replay system with multiple schemas from the Sales schema, schema remapping is performed by adding the captured workload multiple times into a replay schedule and remapping the users to different schemas.

To perform schema remapping in this scenario:

1. On the replay system where you plan to perform scale-up testing, create a directory object for the root directory where the captured workloads are stored:

```
CREATE OR REPLACE DIRECTORY cons_dir AS '/u01/test/cons_dir';
```

2. Create a directory object for the directory where the captured workload is stored:

```
CREATE OR REPLACE DIRECTORY cap_sales AS '/u01/test/cons_dir/cap_sales';
```

Ensure that the captured workload from the Sales application is stored in this directory.

3. Preprocess the captured workload:

```
EXEC DBMS_WORKLOAD_REPLAY.PROCESS_CAPTURE ('CAP_SALES');
```

4. Set the replay directory to the root directory:

```
EXEC DBMS_WORKLOAD_REPLAY.SET_REPLAY_DIRECTORY ('CONS_DIR');
```

5. Create a replay schedule and add the captured workload multiple times:

```
EXEC DBMS_WORKLOAD_REPLAY.BEGIN_REPLAY_SCHEDULE ('double_sales_schedule');
SELECT DBMS_WORKLOAD_REPLAY.ADD_CAPTURE ('CAP_SALES') FROM dual;
SELECT DBMS_WORKLOAD_REPLAY.ADD_CAPTURE ('CAP_SALES') FROM dual;
EXEC DBMS_WORKLOAD_REPLAY.END_REPLAY_SCHEDULE;
```

6. Initialize the consolidated replay:

```
EXEC DBMS_WORKLOAD_REPLAY.INITIALIZE_CONSOLIDATED_REPLAY (
                          'double_sales_replay', 'double_sales_schedule);
```

7. Remap the users:

```
EXEC DBMS_WORKLOAD_REPLAY.SET_USER_MAPPING (2, 'sales_usr', 'sales_usr_2');
```

8. Prepare the consolidated replay:

```
EXEC DBMS_WORKLOAD_REPLAY.PREPARE_CONSOLIDATED_REPLAY;
```

9. Start replay clients:

   a. Estimate the number of replay clients that are required:

      ```
      wrc mode=calibrate replaydir=/u01/test/cons_dir/cap_sales
      ```

   b. Add the output to determine the number of replay clients required.

      You will need to start at least one replay client per workload capture contained in the consolidated workload.

   c. Start the required number of replay clients by repeating this command:

      ```
      wrc username/password mode=replay replaydir=/u01/test/cons_dir
      ```

      The `replaydir` parameter is set to the root directory in which the workload captures are stored.

10. Start the consolidated replay:

    ```
    EXEC DBMS_WORKLOAD_REPLAY.START_CONSOLIDATED_REPLAY;
    ```

Part III
Workload Analysis

Workload Analysis provides valuable near real-time analyses of executed SQL statements in a production database.

Part III covers Workload Analysis and contains the following chapters:

17

Using Workload Analysis

Workload Analysis helps you identify, quantify, and eliminate the reason for regression or improvements. This chapter contains the following topics:

•

A common reason for database performance regression is regressed SQL statements caused by query plan changes, increased data volumes or increased activity in the database.

Workload Analysis performs an analysis of top queries in the database from two different time points expected to be the same or similar. Regressed statements can then be tuned by using SQL Tuning Advisor or SQL Plan baselines.

Accessing Workload Analysis in Enterprise Manager

You access Workload Analysis in Oracle Enterprise Manager Cloud Control using two methods.

Method 1:

1. Click the **Targets** drop-down list.
2. Select **Databases**.
3. In the **Name** column, select your database name. For example, **rep_database**.
4. From the **Performance** drop-down list, select **Workload Analysis**.
5. In the Database Login screen, select a **Named** or **New** credential, and then click **Login** to access Workload Analysis.

Method 2:

1. Click the **Targets** drop-down list.
2. Expand **Databases** and click **Database Instance**.
3. In the **Target Name** column, click the database name. For example, **rep_database**.
4. From the **Performance** drop-down list, select **Workload Analysis**.
5. In the Database Login screen, select a **Named** or **New** credential, and then click **Login** to access Workload Analysis.

Oracle Database Support for Workload Analysis

Workload Analysis is supported on Oracle Database 19c and later.

- Workload Analysis is supported on single instance Oracle Database, Container Database (CDB), and Oracle RAC database.
- Workload Analysis is not supported on Autonomous Databases.
- For CDBs, Workload Analysis is not supported on `CDB$ROOT`.
- The Workload Analysis menu and the tile are not displayed on `CDB$ROOT`.

- Workload Analysis supports CDBs and pluggable databases (PDBs) but does not support `CDB$ROOT`.
- Workload Analysis is supported on Oracle RAC container and on the RAC PDBs. However, Workload Analysis is not supported on Oracle RAC instances.
- The Workload Analysis menu and the tile are not displayed on an Oracle RAC instance.

Overview of Workload Analysis

Workload Analysis provides near real-time analysis of database top SQL statements to identify changed performance and reason for changed performance using historical execution statistics.

A workload is a set of SQL statements that you run in the database or PDB. It can be limited to a specific application or module in the application using filters or it can span the complete database or PDB. These statements with statistics and execution plans are stored in a SQL Tuning Set (STS).When collecting STS from Automatic Workload Repository (AWR) it is limited to the top N statements that can be modified with `dbms_workload_repository.modify_snapshot_settings(topnsql =>[number])`.

The Workload Analysis feature compares two SQL tuning sets from different time points in a production database as compared to the SQL Performance Analyzer which only analyzes one SQL tuning set in a test database before and after a change. You can compare the 2 SQL tuning sets either based on a certain criteria or based on the top statements for the database.

While the SQL Performance Analyzer helps to analyze performance data at the database level, Workload Analysis helps to analyze performance data at the application level.

If you are using a reference workload, then before you start analyzing performance data using Workload Analysis, create a SQL tuning set for your workload.

There are two types of Workload Analysis options currently available.

- Automated Analysis
- One-Time Analysis

Both automated analysis and one-time analysis have options to display the tasks on the Database Home Page in the Workload Analysis tile.

Using Automated Analysis

Automated Analysis generates reports that run automatically based on a schedule configured by the database administrator.

About Automated Analysis

You can use automated analysis to create a task that compares two SQL tuning sets. It runs automatically on a schedule such as hourly, daily, weekly, or monthly.

Creating an Automated Analysis Task

Create an Automated Analysis task by providing the Workload Capture details, Workload Comparison details, and Schedule the time and date of the task.

These are the automated analysis task options:

- Basic Automated Analysis task
- Advanced Automaed Analysis task.

Creating a Basic Automated Analysis Task

Create a Basic Automated Analysis Task by providing the Workload Capture details.

1. Go to the database main page in Enterprise Manager.
2. From the **Performance** drop-down list, select **Workload Analysis**.
3. Select the **Automated Analysis** tab.
4. Click **Create Analysis Task** to create an automated analysis task for your workload.
5. In the **Task Description**, enter a **Name** and a brief **Description** for the automated analyis task. Select the **Display on Home Page** toggle if you want this task to be displayed on the Database home page tile.
6. From the **Select task creation type** option, select **Basic**.
7. In the **Task Parameters** section, schedule your task in the **Workload capture interval (hours)** drop-down list by selecting any number between 1 and 24.
8. Click **Submit**. The basic analysis task is displayed on the **Results** tab after the workload analysis completes.

For example, if you select 4 hours from the **Workload capture interval (hours)** drop-down list, then the Workload Analysis task creates a SQL tuning set from the Automatic Workload Repository (AWR) snapshots created in the last 4 hours and compares this SQL tuning set with the SQL tuning set that is created for the previous 4 hours.

It compares the SQL tuning set for the 6am -10am time interval with the SQL tuning set that is created for the 2 am - 6 am time interval. The task runs every 4 hours, and creates an SQL tuning set for comparison.

Creating an Advanced Automated Analysis Task

Create an Advanced Automated Analysis Task by providing the Workload Capture details, Workload Comparison details, and Schedule the time and date of the task.

1. Go to the database main page in Enterprise Manager.
2. From the **Performance** drop-down list, select **Workload Analysis**.
3. Select the **Automated Analysis** page.
4. Click **Create Analysis Task** to create an automated analysis task for your workload.
5. In the **Task Description**, enter a **Name** and a brief **Description** for the automated analyis task. Select the **Display on Home Page** toggle if you want this task to be displayed on the Database home page tile.
6. From the **Select task creation type** option, select **Advanced**.

7. Enter information for the following sections:

Workload Capture

In the Workload Capture section, enter information about the SQL tuning set and load SQL statements captured from the Automatic Workload Repository (AWR) snapshots.

- **SQL Tuning Set Name Prefix:** Specify a prefix before the SQL tuning set name.
- **Load SQL Statements Using Automatic Workload Repository (AWR) Snapshots**
  - **Specify Custom Time Range:** You can specify a time range to capture AWR snapshots that you can then use to load SQL statements when you create a new SQL tuning set.
    - **Start Time:** Specify the start time to capture AWR snapshots from the AWR.
    - **End Time:** Specify the end time to capture AWR snapshots from the AWR.
  - **Quick Select From Snapshots Created in the Past:** You can quickly select the AWR snapshots that are captured in the past from the drop down list by specifying the number of hours or days.
- **Total Number of SQL Statements Captured:** Top SQL statements that are available from the AWR.
  - **Capture All:** Select this option to capture all the SQL statements.
  - **Capture Top N:** Select this option to capture a specified number of SQL statements such as top ten or top twenty SQL statements.
  - **Filter Option:** You can add filters for Parsing Scheme Name, SQL Text, SQL ID, or Module using operators.

Workload Comparison

Workload comparison has the following options:

- **Subsequent Comparisons**
  - **Compare using a rolling reference:** You can compare workloads that are captured against the previously captured workloads on a rolling basis. Example: Today's workload against the previous day's workload.
  - **Compare using a fixed reference:** A fixed reference is a static SQL Tuning set captured at a specific time point. For example: From January 1st between time A and B when drawing comparisons, it always uses this reference SQL Tuning set to compare.
- **Optional Initial Reference Workload**
  Specify the SQL tuning set that you can use as an initial reference workload to compare other workloads. This value is optional.
  - **Comparison Metric:** Compares the performance metrics using Elapsed time. You can add multiple comparison metrics such as CPU time, Buffer Gets, Disk Reads, Physical I/O, Direct Writes, and Optimizer Cost.
  - **Change Threshold:** A minimum threshold is required here.
  - **Consumer Reference Group:** You can assign tasks to a particular resource group.
  - **Retention policy (days)*:** You can retain the purged executions by selecting the preferred time range. Analysis results that are older than the retention policy will be deleted.
- **Schedule**

You can schedule the time when you want to run a task by specifying a particular time range and time zone.

- **Start Date:** Specify the start date to run a task.
  - **Immediately:** Select this option to run the task immediately.
  - **Later:** Select this option to run the task on a specified date.
- **End Date:** Specify the end date to run a task.
  - **None:** Select this option if you do not want to end the task on a particular date.
  - **Specified Date:** Select this option to end the task on a specific date.
- **Repeat Every:** Select this drop down option to repeat the schedules based on your time preference such as hourly, daily, weekly, or monthly.

Click **Submit**. The advanced analysis task is displayed on the **Results** tab after the workload analysis completes.

Reviewing the Results of Your Automated Analysis Tasks

Use the Results tab to view the outcome and analyze the task that you created for your database workload.

Table 17-1 Results of Your Automated Analysis Tasks

Item	Description
Analysis	Provides performance analysis and comparison reports of the SQL tuning sets. To view the comparison report, expand the analysis task for which you want the report and click **Workload Analysis Report**. Each time there is a regression or improvement in the automated workload, a report is generated.
Description	The description of the workload captured in the SQL tuning sets.
Reference workload	The workload of an existing or a previous SQL tuning set. Click on the reference workload to get more information about the SQL tuning set.
Analysis Workload	Compare the performance of an existing SQL tuning set with another SQL tuning set. Click on the analysis workload to get more information about the SQL tuning set.
Last Updated On	Date on which the tasks was last updated.
Created By	User who created the task. For example, SYS, SYSTEM, or SYMAN.
Metric Comparisons	By default, Elapsed time performance metrics is used to compare other peformance metrics. Based on a comparison of performance metrics such as CPU time, Buffer Gets, Disk Reads, Physical I/O, Direct Writes, and Optimizer Cost, displays a status if the task has Regressed, Improved, or Unchanged.
Missing SQL	Number of SQL statements that were part of the reference workload, but are no longer present in the compared workload.
New SQL	Number of SQL statements that are run frequently during different reference periods and not during the current reference period.
Previous Results	Displays a report about the previous results of the task such as the name of the reference workload, the compared workload, and other metric comparisons.

Listing Your Automated Analysis Tasks

The Analysis Tasks tab contains the following columns and lists all the Automated Analysis Tasks that you created.

Table 17-2 Listing Your Analysis Tasks

Item	Description
Task	Name of the task.
Description	Description of the task.
Created By	User who created the task. For example, SYS, SYSTEM, or SYSMAN.
Last Run Date	Date when the task was last run.
Next Run Date	Date when the task is automated to run next.
State	Status of the task whether it is scheduled or not. Click on the status for more information about the automated job.
Enabled	If the task is enabled or not.
Previous Executions	Tasks that were executed previously. Click on the task number to get information about the task such as created by, start date, completion date, elapsed time, and current status.
Vertical dots menu	Click on the vertical dots menu to enable, disable, delete, or stop the execution of the task. Click **Display on Home Page** to display an automated analysis task on the database home page.

Reviewing Workload and Metric Summary

Automated Analysis provides a series of panels that give you a brief summary of the workloads and metrics of the analysis tasks that you created.

- **Changed Workloads:** Number of workloads that have changed.
- **Monitored Workloads:** Number workloads that are being monitored.
- **Regressed Metrics:** The improved and regressed metrics is based on comparisons. One SQL tuning set can have several comparisons.
- **Improved Metrics:** Number of SQL tuning sets that have improved.
- **Unchanged Metrics:** Number of SQL tuning sets that are unchanged.

You can also select options from the **View Data** drop-down list in the top-right of the Workload Analysis page to view Workload Analysis data for any time or without any time limit.

Using One-Time Analysis

Use the One-Time Analysis page to create a one-time analysis task that will help you compare the performance characteristics of two similar workloads. For example, to measure the performance of a SQL tuning set before and after a database upgrade or patch.

About One-Time Analysis

One-Time analysis performs a comparison of two workloads. You can do this analysis to validate performance after a known change such as an application upgrade.

Creating a One-Time Analysis Task

Create an Analysis Task for one-time analysis of your workload by providing the workload definition and comparison details.

1. Go to the database main page in Enterprise Manager.
2. From the **Performance** drop-down list, select **Workload Analysis**.
3. Select the **One-Time Analysis** page.
4. Click **Create One-Time Analysis Task** to create a One-Time analysis task or to schedule an analysis task for your workload.
5. In the **Create One-time Analysis Task** window, enter information for the following sections:

Workload

A workload is a SQL tuning set that is captured for a certain time period and includes a set of filtered SQL statements and Data Manipulation Language (DML) statements.

General Option

The options that are available for the various tasks that you can create for One-Time Analysis.

- **Name:** Enter a name for the scheduled task.
- **Description:** Enter a brief description for the scheduled task.

Workload Comparison

Enter information to compare the performance of various SQL tuning sets.

- **Reference Workload:** Search and enter the SQL tuning set to set as a reference point.
- **Compared Workload:** Search and enter a workload so you can compare the performance of an existing SQL tuning set with another SQL tuning set.
- **Comparison Metric:** Enter value/s to be used when generating comparison reports. When selecting multiple metrics each metric generates a compression report based on any of these metrics such as Elapsed time, CPU time, Buffer Gets, Disk Reads, Physical I/O, Direct Writes, and Optimizer Cost.
- **Change Threshold:** Enter threshold in % when a statement is considered to have regressed.

Reviewing the Results of Your One-Time Analysis Task

The One-Time Analysis page displays the results of all the one-time analysis workload tasks that you created. It lists the following information.

Table 17-3 Results of Your One-Time Analysis Tasks

Item	Description
Analysis	Provides a one-time performance analysis and comparison reports of the SQL tuning sets. To view the comparison report, expand the analysis task for which you want the report and click **Worload Analysis Report**. Each time there is a regression or improvement in the scheduled workload, a report is generated.

Table 17-3 (Cont.) Results of Your One-Time Analysis Tasks

Item	Description
Description	The task description of the workload captured in the SQL tuning sets.
Reference workloads	The workload of an existing or a previous SQL tuning set. Click on the reference workload to get more information about the SQL tuning set.
Compared Workload	Compare the performance of an existing SQL tuning set with another SQL tuning set. Click on the compared workload to get more information about the SQL tuning set.
Last Updated On	Date on which the tasks was last updated.
Created By	User who created the task. For example, SYS, SYSTEM, or SYMAN.
Delete	Option to delete a one-time task.
Metric Comparisons	Based on a comparison of performance metrics such as Elapsed time, CPU time, Buffer Gets, Disk Reads, Physical I/O, Direct Writes, and Optimizer Cost, displays a status if the task has Regressed, Improved, or Unchanged.
Missing SQL	Number of SQL statements that were part of the reference workload, but are no longer present in the compared workload.
New SQL	Number of SQL statements that appeared in the compared workload, but were not present in the reference workload.
State	Current status of the task which can be either completed or executing.

Reviewing the Analysis and Metric Summary

One-Time Analysis provides a series of panels that give you a brief analyses and metrics of the analysis tasks that you created.

- **Analyses With Differences:** Displays the number of one-time analyses task with differences.
- **Total Analyses:** Displays the total number of one-time analyses tasks.
- **Regressed Metrics:** The improved and regressed metrics is based on comparisons. An automated workload analysis task can have several comparisons.
- **Improved Metrics:** Number of SQL tuning sets that are improved.
- **Unchanged Metrics:** Number of SQL tuning sets that are unchanged.

You can also select options from the **View Data** drop-down list in the top-right of the database page to view Workload Analysis data for the last 24 hours, 1 week, 30 days, or a combination of all of these.

Reviewing the Automated Workload Analysis Task on the Database Home Page

You can review only one automated workload analysis task on the database home page by selecting a specific workload analysis task on the workload analysis page.

Adding or Removing Tasks on the Home Page

You can add or remove an automated analysis task for your database Home Page using two methods.

Method 1:

1. Go to the database main page in Enterprise Manager.
2. From the **Performance** drop-down list, select **Workload Analysis**.
3. Select the **Automated Analysis** tab.
4. Click **Create Analysis Task** to create an automated analysis task for your workload.
5. In the **Task Definition** section, enter a name and description for the automated analysis task.
6. Use the **Display on home page** toggle bar to display an automated analysis task on the database home page. A home icon appears next to the task confirming that the task is displayed on the database home page.
7. Click **Submit** to add the automated analysis task for your workload.

Method 2:

1. In the Workload Analysis home page, select the **Analysis Tasks** tab.
2. From the **Analysis Tasks** tab, select the task that you want to display on the database home page.
3. From the Vertical dots menu, select **Display on Home Page** to display the selected task on the database home page. The task is displayed on the database home page.
4. From the list of tasks, select the task that you want to remove from the database home page.
5. From the Vertical dots menu, select **Remove from Home Page** to remove the selected task from the database home page.

> **Note:**
>
> The task takes some time to display on the Database home page.

Accessing the Automated Workload Analysis Task

You can view the automated workload analysis tasks on the database home page.

1. Go to the database home page.
2. Click on the **Workload Analysis** tile to access the workload analysis task next to the Workload Analysis metric for which you want the report.

> **Note:**
>
> Do not map multiple EMs to the same database as this feature does not work correctly if you have multiple EMs managing the same Database.

Reviewing the Automated Workload Analysis Tasks

In the automated or one-time workload analysis, a performance analysis and Workload Analysis report of the SQL tuning sets is always generated.

Table 17-4 Automated Workload Analysis Tasks

Item	Description
Task	Name of the task.
Description	The description of the workload captured in the SQL tuning sets.
Reference Period	It is a period between two time points where we capture a SQL tuning set. It is used to compare with an analysis period.
Analysis Period	The time interval when we capture a SQL tuning set.
Summary	Displays the summary of the workload impact for the comparison metric that you choose in the analysis task in the Workload Analysis home page.
SQL Commonality	Measure of similarity between the reference and analysis periods based on the SQL statements from both periods with a range of 0% to 100%.
Overall Performance Change	Displays the total improvements or regression in percentage for the selected comparision metric.
Workload Impact	Percentage of contribution from a category of SQL statements to overall perfornance regression.
Common SQL	Number of SQL statements that are common in both runs and displays the top common SQL statements by variance in average response time and total database time.
Improved SQL	Number of SQL statements whose performance has improved.
Regressed SQL	Number of SQL statements whose performance has regressed.
Missing SQL	Number of SQL statements that were part of the reference workload, but are no longer present in the compared workload.
New SQL	Number of SQL statements that appeared in the compared workload, but were not present in the reference workload.
Elapsed Time	Elapsed time is the total amount of seconds for all the executions of all the statements.
SQL Statements by Performance	Number of SQL statements for each group that are common in both the referenced and analysis periods based on performance.
SQL Statements by Plan Change	Number of SQL statements that are common in both the referenced and analysis periods based on plan change.
View All Results	Displays all the results for the automated workload analysis in the database home page.

Reviewing the Workload Analysis Report

In the automated or one-time workload analysis, a performance analysis and Workload Analysis report of the SQL tuning sets is always generated.

Review this report for data such as Workload Analysis metrics, performance breakdown of SQL statements, and top SQL statements impacted by the workload.

Also, review the information at the top of the page such as Task Name, Reference Workload, Compared Workload, Execution Name, Reference Workload Owner, Compared Workload Owner, Reference SQL Analyzed, and Compared SQL Analyzed.

- Accessing the Workload Analysis Report
- Reviewing the Summary Report
- Example: Workload Analysis Report

Accessing the Workload Analysis Report

You can view the Workload Analysis report for your workload analysis task.

1. In the Workload Analysis home page, expand the analysis task for which you want the report.
2. Click the **Workload Analysis Report** link next to the Workload Analysis metric for which you want the report.
3. A **Workload Analysis Report** page is displayed with the Workload Analysis metrics.
4. Click **Save Report** to download the Workload Analysis report.

Reviewing the Summary Report

The Summary panel displays a summary of the workload impact for the comparison metric that you choose in the analysis task in the Workload Analysis home page.

The UI displays Workload Analysis reports for the following metrics:

- Direct Writes: Direct Writes allow a session to queue an I/O write request and continue processing while the Operating System handles the I/O.
- Physical I/Os: The sum of Direct Writes and Disk Reads when you run a SQL statement.
- Disk Reads: The total amount of Physical Disk reads (disk reads is only done if the data is not available in the memory).
- Buffer Gets: Total number of buffer gets (number of times the database accessed a block) for this SQL statement.
- Optimizer Cost: The cost calculated by the optimizer for the execution plan.
- Elapsed Time: Number of seconds elapsed for an SQL statement.
- CPU Time: Sum of the CPU time for a SQL statement.

Example: Workload Analysis Report

The workload analysis report is a report that compares two SQL tuning sets where it highlights differences to make it possible to identify abnormalities between two execution periods.

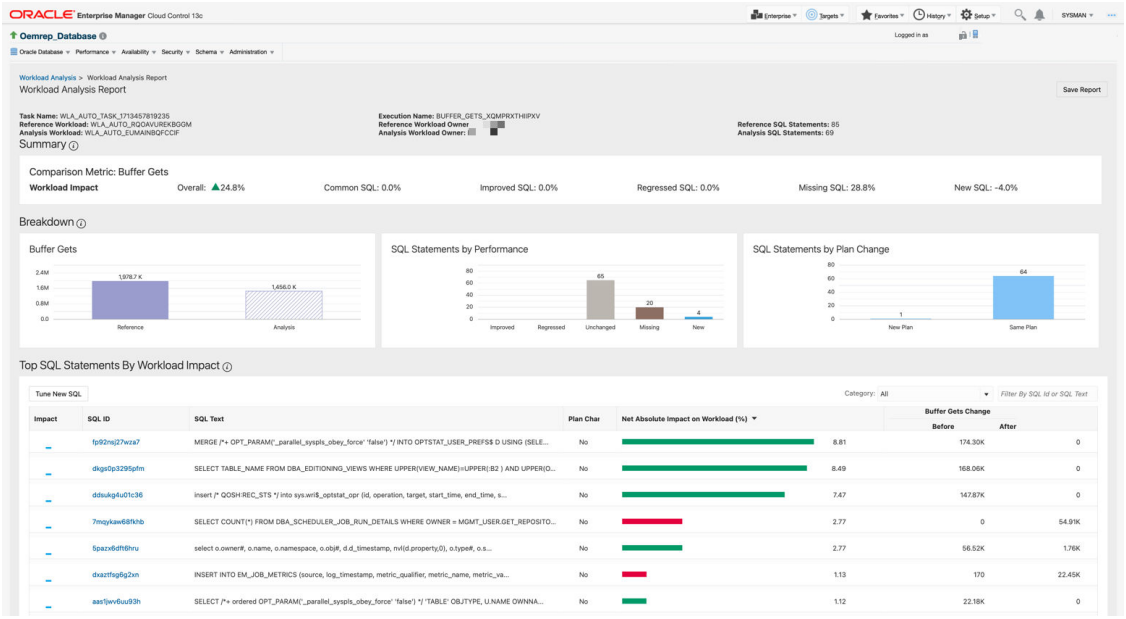
Overview of the Workload Analysis Report

There will always be differences when capturing two different SQL tuning sets and it is up to you to interpret and understand if these differences are abnormal or not.

For example, let's assume that the analyzed workload is expected to be same. You have captured a SQL tuning set last Monday between 9 a.m. and 10 a.m. and you compare it with a workload executed this Monday between 9 a.m. and 10 a.m. The workload is a SQL tuning set captured from Automatic Workload Repository (AWR), hence it will include only top N

statements for each category. The statements that contribute to the performance metrics such as the elapsed time. However, it will not result in new statements to be the top contributor.

Figure 17-1 Overview of the WLA Report



The top section provides you with information of the name of the workloads compared and how many SQL statements are captured for each workload. In this example, there are 70 statements in the reference workload and 75 statements in the compared workload.

This indicates that we have more statements in AWR for the comparison period. This may imply that statements in different categories align more in the reference workload than in the compared workload, or there has been a change to include more statements in the AWR snapshot.

These are the criteria for the number of top SQL statements by workload impact:

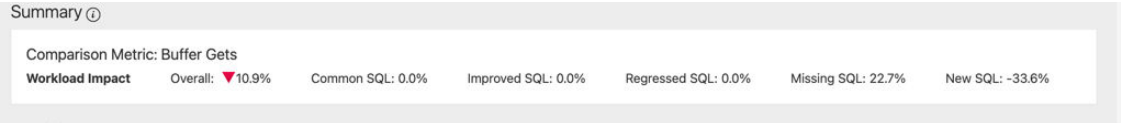
- SQL ordered by Elapsed Time
- SQL ordered by CPU Time
- SQL ordered by User I/O Wait Time
- SQL ordered by Gets
- SQL ordered by Reads
- SQL ordered by Physical Reads (UnOptimized)
- SQL ordered by Executions
- SQL ordered by Parse Calls
- SQL ordered by Sharable Memory
- SQL ordered by Version Count

AWR keeps the top N SQL for each criteria. If none of these overlap, then there are 30 statements * 10 criteria = 300 statements. So instead of 300, there may be around 150. But this may fluctuate, in the range of 170 to 130 and so on. So the delta may either increase or decrease and is dependent on the overlap.

Summary Section

In this example, the report is generated on the Elapsed Time comparison metric. You can see an overall regression of 22.3%.

Figure 17-2 Summary Section



The regression is divided into different categories:

Common SQL: These statements are executed during both time periods.

Improved SQL: What are the total improvements on all statements with less total elapsed time than the reference workload.

Regressed: What is the total regression on all statements with more total elapsed time than the reference workload.

Missing SQL: This is the total improvements on all statements not captured in the compared workload. If there are no statistics collected, they will not contribute to the elapsed time. These statements can be replaced by other statements as top N statements in AWR or they can be replaced by statements due to application upgrades or similar activities.

New SQL: This is the total impact on all new statements captured in the compared workload but not in the reference workload. The reasons for new statements can be the same as it is for missing statements.

In this example, missing and new contributes to approximately 50% of the total workload.

Breakdown

You should always correlate the information from the breakdown section with the data in the summary section. If you have 10% increased elapsed time, then it is important to know other circumstances like the total amount of elapsed time.

Figure 17-3 Breakdown section

In this example, the total elapsed time is approximately 24 seconds. Hence, there is no real value to perform a comparison if the workload is captured for an hour. But if the elapsed time is around 15000 seconds, then a 10% increase can have an impact on application performance.

Elapsed Time

The Elapsed Time tile provides the total amount of elapsed time for all executions of all captured statements within that time period. The label of this tile will change to reflect the comparison metric that you select.

SQL Statements by Performance

This tile provides the number of statements for each group. It is important to correlate this data with the data available in the summary section. In this example, the Missing statements is 12% of the total amount of statements but it contributes to almost 45% of the total elapsed time, and same for new statements which is 19% of all statements but it contributes to 54% of the elapsed time.

In this use case they are high contributors to the elapsed time and should not just disappear or appear without any explanation.

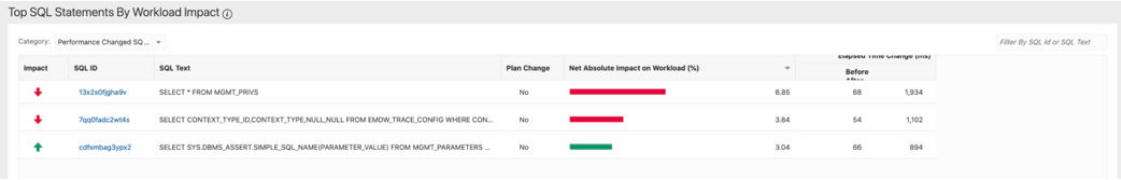
SQL Statements by Plan Changes

This is an important section where you must analyze the SQL details. If the new plan shows performance improvements, then you must determine if a single execution is faster or slower. If the execution is slower, then there is regression. However, there are less executions, which provides us with less elapsed time, and you should investigate this further. If it shows regression, then you must investigate if there is an increased elapsed time for a single execution. However, less elapsed time for a single execution does not require any further investigation.

Top SQL Statements by Workload Impact

In this tile you can add different filters to show only a certain category of SQL statements. By default, this tile will show Performance Changed SQL but it is also possible to show all, only regressed, and a few other categories.

Figure 17-4 Top SQL Statement by Workload

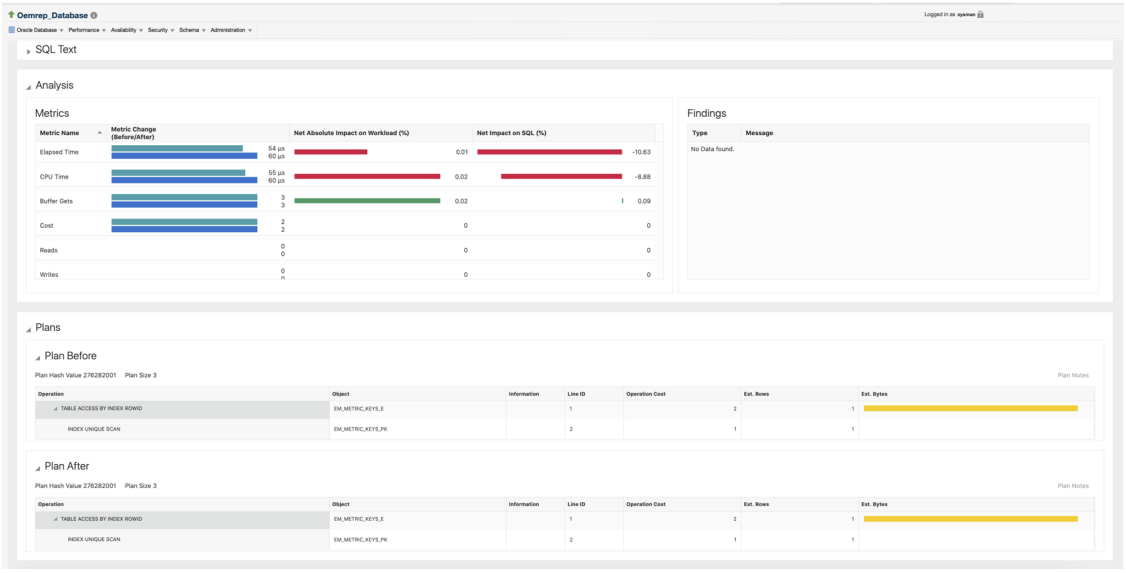


To change the category, select the category from the drop-down list. In the example, you can see 3 SQL statements with performance changes. Note that it is the predefined impact threshold because of which the SQL statements are listed as performance changed. The default threshold is 3% and it is applicable for either impact on workload or impact on SQL statement. Click on the SQL ID to see more details for each individual SQL statement.

SQL Details

SQL details are divided into three different tiles where the analysis and the plan tiles are the most important.

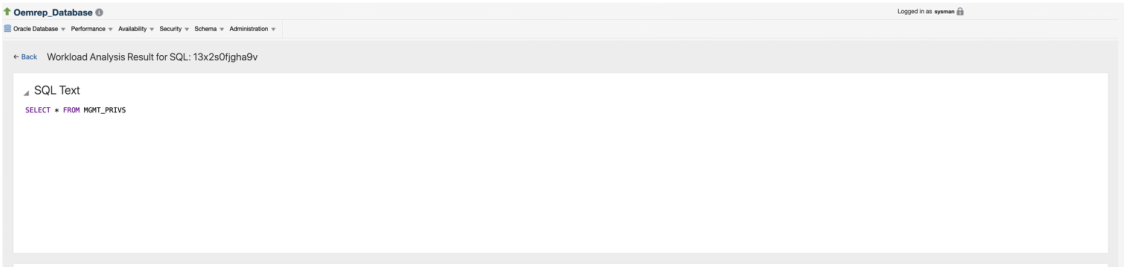
Figure 17-5 SQL Details



SQL Text

This tile is collapsed by default. Expand it to see the complete SQL text.

Figure 17-6 SQL Text



Analysis

Analysis is divided into two tiles: Metrics and Findings.

Figure 17-7 Analysis Metrics

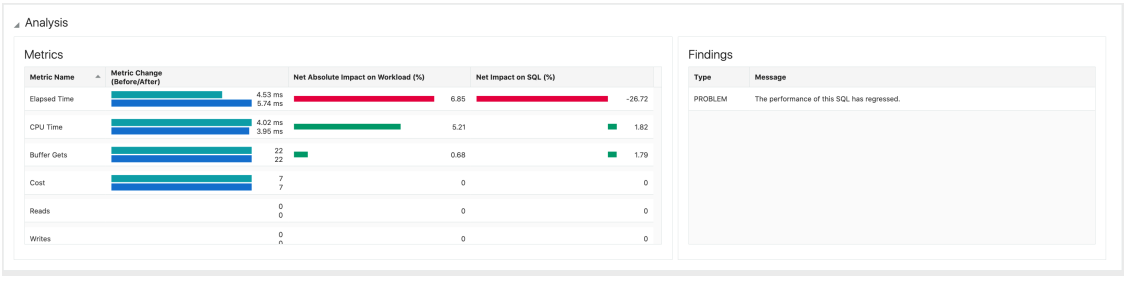
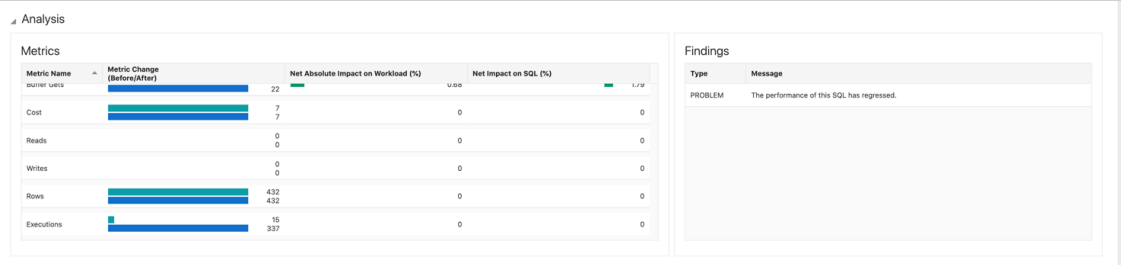
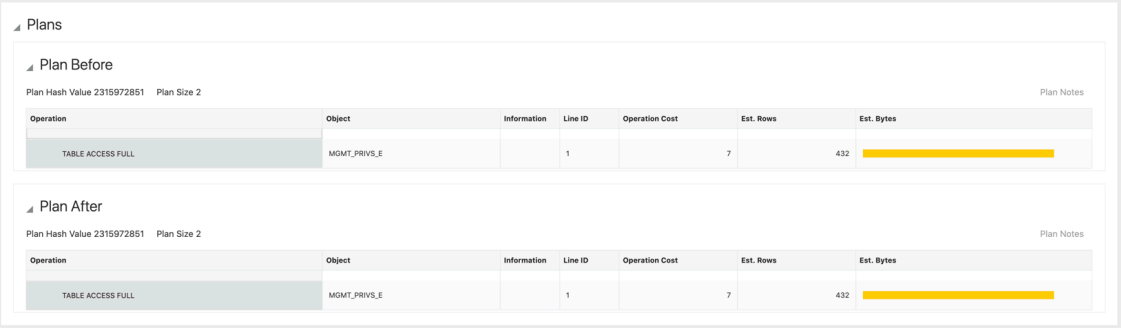


Figure 17-8 Analysis Findings



The Metrics tile includes statistics for all important metrics and displays an overview of all the differences. In this example, the elapsed time has a small change and CPU time has decreased. Buffer Gets is the same, which implies that each execution does more or less the same work. But the major difference is that for Executions instead of 16 executions there are 337 executions. Hence, the load is much higher. This explains why the elapsed time is higher. If the load is higher, then it may have introduced waits. There are no I/Os, which indicates that the CPU is flooded.

Figure 17-9 Plan Changes



Other reasons for performance changes are plan changes. The query has a new more efficient or less efficient plan. The workload analysis report provides you with a Plan Before and Plan After for all common statements.