Oracle® Database

Real Application Testing User's Guide

10*g* Release 2 (10.2)

E12024-01

January 2008

Oracle Database Real Application Testing User’s Guide, 10*g* Release 2 (10.2)

E12024-01

Primary Author: Immanuel Chan

Contributors: Pete Belknap, Supiti Buranawatanachoke, Romain Colle, Karl Dias, Leonidas Galanis, Prabhaker Gongloor, Aneesh Khandelwal, Matthew McKerley, Mughees Minhas, Adnan Qazi, Venkateshwaran Venkataramani, Yujun Wang, Khaled Yagoub

Contents

Preface

Oracle's 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 before production deployment, Oracle Real Application Testing minimizes the risk of instabilities associated with changes.

Database Replay enables you to replay a full production workload on a test system to assess the overall impact of system changes. SQL Performance Analyzer enables you to assess the impact of system changes on SQL response time on a given SQL workload.

In this release, Oracle Real Application Testing only supports the following partial functionality:

- For Database Replay, only the capability to capture a database workload is supported
- For SQL Performance Analyzer, only the capability to capture a SQL workload into a SQL tuning set is supported

This functionality is provided is so that you can use this option to test database upgrades to Oracle Database 11*g* and subsequent releases.

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

This preface contains the following topics:

- Audience
- Documentation Accessibility
- Related Documents
- Conventions

Audience

This document provides information about how to assure the integrity of database changes using Oracle Database Real Application Testing. This document is intended for database administrators, application designers, and programmers who are responsible for performing real application testing on Oracle Database.

Documentation Accessibility

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

`http://www.oracle.com/accessibility/`

Accessibility of Code Examples in Documentation

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

Accessibility of Links to External Web Sites in Documentation

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

TTY Access to Oracle Support Services

Oracle provides dedicated Text Telephone (TTY) access to Oracle Support Services within the United States of America 24 hours a day, 7 days a week. For TTY support, call 800.446.2398. Outside the United States, call +1.407.458.2479.

Related Documents

For more information about some of the topics discussed in this document, see the following documents in the Oracle Database Release 10.2 documentation set:

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

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.

Convention	Meaning
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

1 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 recreating 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.

Capturing the production workload eliminates the need to develop simulation workloads or script, 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.

Database Replay performs workload capture of external client workload at the database level and has negligible performance overhead. You can use Database Replay to test any significant system changes, including:

- Database and operating system upgrades
- Configuration changes, such as conversion of a database from a single instance to an Oracle Real Application Clusters (RAC) environment
- Storage, network, and interconnect changes
- Operating system and hardware migrations

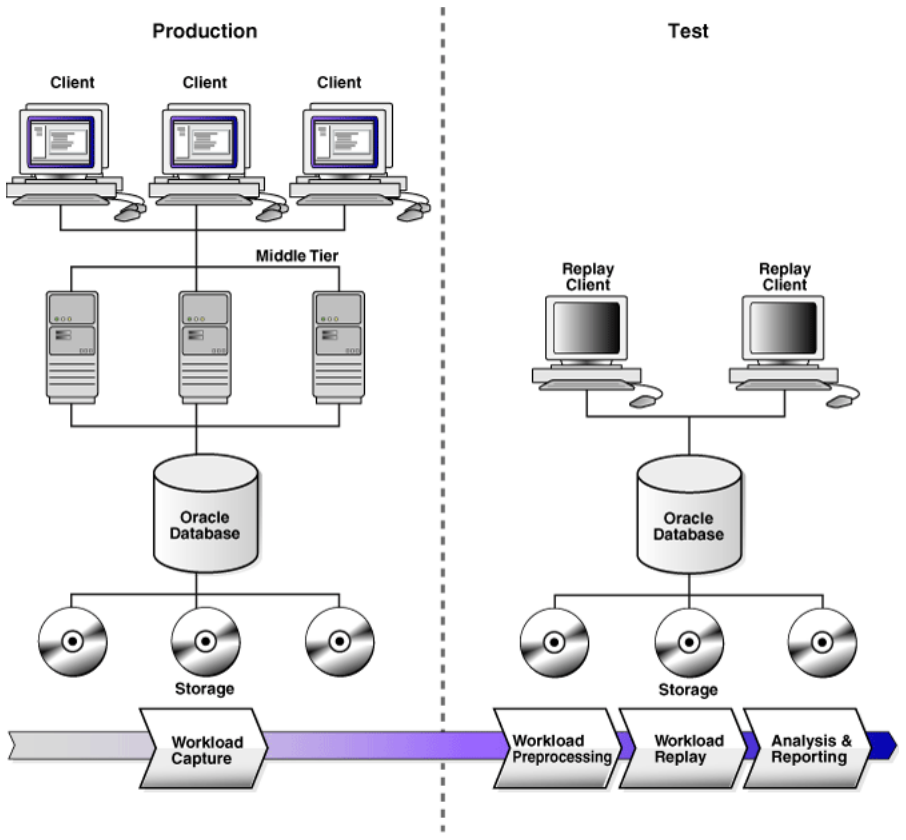
This chapter describes how to use the Database Replay feature of Oracle Database and contains the following sections:

- Overview of Database Replay
- Capturing a Database Workload
- Analyzing Workload Capture

Overview of Database Replay

By capturing a workload on the production system and replaying it on a test system, Database Replay enables you to realistically test system changes by essentially recreating the production workload environment on a test system.

Figure 1–1 Database Replay Architecture



Using Database Replay requires four main steps, as shown in Figure 1–1:

- Workload Capture
- Workload Preprocessing
- Workload Replay
- Analysis and Reporting

Note: Only workload capture is currently supported in this release. Captured workloads can be preprocessed and replayed on Oracle Database 11*g* Release 1 (11.1) and subsequent releases.

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. These capture files are platform independent and can be transported to another system. You can specify the start time and duration for the workload capture, as well as the location to store the capture files. 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.

For information about how to capture a workload on the production system, see "Capturing a Database Workload" on page 1-4.

Workload Preprocessing

Once the workload has been captured, the information in the capture files need to 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 another 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: *Oracle Database Performance Tuning Guide* 11*g* Release 1 (11.1) for more information about preprocessing 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 recreating 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 recreate 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 logically similar to the data in the capture system to minimize data divergence and enable reliable analysis of the replay.

See Also: *Oracle Database Performance Tuning Guide* 11*g* Release 1 (11.1) for more information about replaying a captured workload

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 report summary provides 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 DB time, average active sessions, and user calls—between the workload capture and the workload replay is also provided. 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.

For application-level validation, you should 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 similar number of orders are also processed during replay.

For information about how to generate and analyze workload capture reports, see "Analyzing Workload Capture" on page 1-14.

Capturing a Database Workload

This section describes how to capture a database workload on the production system. The primary tool for capturing database workloads is Oracle Enterprise Manager. If for some reason Oracle Enterprise Manager is unavailable, you can capture database workloads using APIs.

This section contains the following topics:

- Enabling and Disabling Workload Capture
- Prerequisites for Capturing a Database Workload
- Workload Capture Options
- Workload Capture Restrictions
- Capturing a Database Workload Using Enterprise Manager
- Monitoring Workload Capture Using Enterprise Manager
- Capturing a Database Workload Using APIs
- Monitoring Workload Capture Using Views

Enabling and Disabling Workload Capture

Before a workload can be captured, workload capture needs to be enabled on the system where you plan to capture the workload. By default, workload capture is not enabled in Oracle Database 10*g* Release 2 (10.2). You can enable or disable workload capture by specifying the `PRE_11G_ENABLE_CAPTURE` initialization parameter.

To enable workload capture, 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.

See Also: *Oracle Database Reference* for more information about the `PRE_11G_ENABLE_CAPTURE` initialization parameter

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 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.

See Also:

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

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
- Defining the Workload Filters
- Setting Up the Capture Directory

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 data divergence when the workload is replayed. Any subsequent transactions with dependencies on the incomplete transactions may also generate errors during replay.

Before restarting the database, determine an appropriate time to shut down the production database prior to the workload capture time period 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 at 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_RESTRICTED`, 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.

Note: *Oracle Database Administrator's Guide* for information about restricting access to an instance at startup

Only one workload capture can be performed at any given time. For Oracle Real Application Clusters (RAC), workload capture is performed for the entire database. To enable a clean state before starting to capture the workload, all the instances need to be restarted. You can do this by:

1. Shutting down all the instances.
2. Restarting one of the instances.
3. Starting workload capture.
4. Restarting the rest of the instances.

Defining the Workload Filters

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. 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. For example, if the system where the workload will be replayed is running Oracle Enterprise Manager (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. You can use either inclusion filters or exclusion filters in a workload capture, but not both.

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.

For Oracle RAC, consider using a shared file system. Alternatively, you can set up capture directory paths that resolve to separate physical directories on each instance, but you will need to collect the capture files created in each of these directories into a single directory before preprocessing the workload capture.

Workload Capture Restrictions

The following types of client requests are not captured in a workload in the current release:

- Direct path load of data from external files using utilities such as SQL*Loader
- Shared server requests (Oracle MTS)
- Oracle Streams
- Advanced Replication streams
- 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)
- Remote `DESCRIBE` and `COMMIT` operations

Capturing a Database Workload Using Enterprise Manager

This section describes how to capture a database workload using Enterprise Manager.

To capture a database workload using Enterprise Manager:

1. On the Administration page, under Real Application Testing, click **Database Replay**.

   The Database Replay page appears.

2. In the Go to Task column, click the icon that corresponds to the Capture Workload task.

   The Capture Workload: Plan Environment page appears.

3. Verify that all prerequisites are met before proceeding.

   For information about the prerequisites, see "Prerequisites for Capturing a Database Workload" on page 1-5. For each verified prerequisite, check the box in the Acknowledge column. Once all prerequisites are verified, click **Next**.

   The Capture Workload: Options page appears.

4. Select the workload capture options.

- Under Database Restart Options, select whether the database will be restarted before workload capture.

  It is strongly recommended that the database be restarted before capturing a workload to enable a clean state for workload capture. Otherwise, potential problems may arise when replaying the workload. For more information, see "Restarting the Database" on page 1-6.

- Under Workload Filters, 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 Another Row and enter the filter name, session attribute type, and attribute value in the corresponding fields. For more information, see "Defining the Workload Filters" on page 1-6.

After selecting the desired workload capture options, click **Next**. The Capture Workload: Parameters page appears.

5. Define the parameters for the workload capture.

   - Under Workload Capture Parameters, in the Capture Name field, enter a name for the workload capture. In the Directory Object list, select the directory where the captured workload will be stored. You must select a directory that does not already contain a workload capture. For more information, see "Setting Up the Capture Directory" on page 1-7.

     To create a new directory object, click **Create Directory Object**. The Create Directory Object page appears. In the Name field, enter a name for the directory object. In the Path field, enter the path to the directory object. To test if the directory exists in the file system, click **Test File System**. If the directory does not exist, it will need to be created first.

   - Under Database Shutdown Parameters, select the type of database shutdown method to perform. This option only appears if the database will be restarted before workload capture. The types of available database shutdown methods include:

     – Immediate

       An immediate shutdown will roll back all active transactions and disconnect all connected users prior to shutting down the database.

     – Transactional

       A transactional shutdown will first complete all active transactions and then disconnect the connected user prior to shutting down the database.

     – Abort

       An abort shutdown will shut down the database instantaneously by aborting all active transactions.

   - Under Database Startup Parameters, select if the database will restart using the current default server parameter file (spfile) or a specific parameter file (pfile). To select a pfile, enter the fully qualified name for the pfile. This option only appears if the database will be restarted before workload capture.

   After defining the parameters for the workload capture, click **Next**. The Capture Workload: Schedule page appears.

6. Under Job Parameters, define the parameters for the job:

- In the Job Name field, enter a name for the job name or accept the system generated name.
- In the Description field, enter an optional description of the job.

7. Under Job Schedule, specify a start time and duration for the workload capture:
   - Under Start, select whether the job will run immediately by selecting **Immediately**, or at a later time by selecting **Later** and specifying the desired time using the Date and Time fields.
   - Under Capture Duration, specify how long the job will run by selecting **Duration** and specifying the desired duration using the Hours and Minutes fields. To not specify a capture duration, select **Not Specified**. If a capture duration is unspecified, the job must be stopped manually.
8. Under Job Credentials, enter the host and database login credentials:
   - Under Host Credentials, enter the username and password for the host system.
   - Under Database Credentials, enter the username and password for the database that will used for the workload capture. The user needs the `DBA` privilege in order to capture the workload.

   Click **Next**. The Capture Workload: Review page appears.
9. Review the job settings for the workload capture that have been defined. To run the job, click **Submit**. To make changes, click **Back**. To cancel the workload capture without saving changes, click **Cancel**.
10. Depending on the job settings that have been defined:
    - If the job is scheduled to start immediately and the database will be restarted, the Confirmation: Restart Database page appears. To restart the database, click **Yes**. The Information: Restart Database page appears while the database is being restarted. Once the database is restarted, the workload capture begins automatically. Click **Refresh**. The View Workload Capture page appears.
    - If the job is scheduled to start immediately but the database will not be restarted, the workload capture begins automatically and the Database Replay page appears with a confirmation that the job has been successfully created.
    - If the job is scheduled to start at a later time, the Database Replay page appears with a confirmation that the job has been successfully created.

    Once workload capture begins, you can monitor the capture process using the View Workload Capture page, as described in "Monitoring an Active Workload Capture" on page 1-10.

Monitoring 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 or delete a completed workload capture

If for some reason Oracle Enterprise Manager is unavailable, you can monitor workload capture using views, as described in "Monitoring Workload Capture Using Views" on page 1-14.

This section contains the following topics:

- Monitoring an Active Workload Capture
- Stopping an Active Workload Capture
- Managing 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. On the Administration page, under Real Application Testing, click **Database Replay**.

   The Database Replay page appears.

2. Under Active Capture and Replay, select the workload capture you want to monitor and click **View**.

   The View Workload Capture page appears.

3. Under Summary, information about the workload capture is displayed.
4. To view the workload profile, click the **Workload Profile** tab.

   Under Average Active Sessions, the 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).

   Under Comparison, various statistics for the workload capture are displayed, including database time, average active sessions, user calls, transactions, connects, and application errors. The statistics for the total session activity are displayed in the Total column, and the statistics for the captured session activity are displayed in the Capture column. The Percentage of Total column displays the percentage of total session activity that are being captured in the workload.

   To view the workload capture report, click **View Workload Capture Report**.

5. To view workload filters used by the workload capture, click the **Workload Filters** tab.

   Details about the workload filters used by the workload capture are displayed, including the workload filter name, type, session attribute, and value.

6. To return to the Database Replay page, click **OK**.

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. On the Administration page, under Real Application Testing, click **Database Replay**.

   The Database Replay page appears.

2. Under Active Capture and Replay, select the workload capture you want to stop and click **Stop**.

   The Confirmation page appears.

3. To confirm that you want to stop the workload capture, click **Yes**.

   The Export AWR Data page appears.

4. To export the Automatic Workload Repository (AWR) data, click **Yes**.

   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. If you choose not to export AWR data, click **No**. You can still export AWR data from a completed workload capture at a later time from the View Workload Capture History page.

   > **See Also:** *Oracle Database Performance Tuning Guide* for information about the Automatic Workload Repository

Managing a Completed Workload Capture

This section describes how to manage a completed workload capture using Enterprise Manager.

To manage a completed workload capture:

1. On the Administration page, under Real Application Testing, click **Database Replay**.

   The Database Replay page appears.

2. Click **View Workload Capture History**.

   The View Workload Capture History page appears.

3. To delete a workload capture, select the workload capture and click **Delete**.

4. To export AWR data for a workload capture, select the workload capture and click **Export AWR Data**.

   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.

5. To view details about a workload capture, select the workload capture and click **View**.

   The View Workload Capture page appears.

6. Under Summary, information about the workload capture is displayed.

7. To view the workload profile, click the **Workload Profile** tab.

   Under Average Active Sessions, the 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 will be shown only when there is Active Session History (ASH) data available for the capture period.

   > **See Also:** *Oracle Database Performance Tuning Guide* for information about Active Session History

   Under Comparison, various statistics for the workload capture are displayed, including database time, average active sessions, user calls, transactions, connects, and application errors. The statistics for the total session activity are displayed in the Total column, and the statistics for the captured session activity are displayed in the Capture column. The Percentage of Total column displays the percentage of total session activity that are being captured in the workload.

To view the workload capture report, click **View Workload Capture Report**.

8. To view workload filters used by the workload capture, click the **Workload Filters** tab.

   Details about the workload filters used by the workload capture are displayed, including the workload filter name, type, session attribute, and value.

9. To return to the Database Replay page, click **OK**.

Capturing a Database Workload Using APIs

This section describes how to capture a database workload using APIs. Capturing a database workload using the `DBMS_WORKLOAD_CAPTURE` package involves:

- Adding and Removing Workload Filters
- Starting a Workload Capture
- Stopping a Workload Capture
- Exporting AWR Data for Workload Capture

Adding and Removing Workload Filters

This section describes how to add and remove workload filters. For information about using workload filters, see "Defining the Workload Filters" on page 1-6.

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 uses the `fname` required parameter, which specifies the name of the filter to be removed.

Starting a Workload Capture

Before starting a workload capture, you must first complete the prerequisites for capturing a database workload, as described in "Prerequisites for Capturing a Database Workload" on page 1-5. You should also review the workload capture options, as described in "Workload Capture Options" on page 1-5.

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_RESTRICTED` prior to 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" on page 1-6.

To start the workload capture, use the `START_CAPTURE` procedure:

```
BEGIN
  DBMS_WORKLOAD_CAPTURE.START_CAPTURE (name => 'dec06_peak',
                          dir => 'dec06',
                          duration => 600);
END;
/
```

In this example, a workload named `dec06_peak` will be captured for 600 seconds and stored in the operating system defined by the database directory object named `dec06`.

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` optional 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.

Stopping a Workload Capture

To stop the 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.

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 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. The `EXPORT_AWR` procedure uses the `capture_id` required parameter, which specifies the ID of the capture whose AWR snapshots will be exported. This procedure will work 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.

Monitoring Workload Capture Using Views

This section summarizes the views that you can display to monitor workload capture. 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 used for workload captures defined in the current database.

See Also: *Oracle Database Reference* for information about these views

Analyzing Workload Capture

This section describes how to generate and analyze workload capture reports. 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.

This section contains the following topics:

- Generating a Workload Capture Report Using Enterprise Manager
- Generating a Workload Capture Report Using APIs
- Using a Workload Capture Report

Generating a Workload Capture Report Using Enterprise Manager

The workload capture report contains captured workload statistics, information about the top session activities that were captured, and any workload filters used during the capture process.

To generate a workload capture report using Enterprise Manager:

1. On the Administration page, under Real Application Testing, click **Database Replay**.

   The Database Replay page appears.

2. Click **View Workload Capture History**.

   The View Workload Capture History page appears.

3. Select the workload capture for which you want to run a workload capture report and click **View**.

   The View Workload Capture page appears.

4. To view the workload capture report, click **View Workload Capture Report**.

   The Report window opens while the report is being generated.

5. Once the report is generated, you can save the report by clicking **Save to File**.

   For information about how to use a workload capture report, see "Using a Workload Capture Report" on page 1-15.

Generating a Workload Capture Report Using APIs

The workload capture report contains captured workload statistics, information about the top session activities that were captured, and any workload filters used during the capture process.

To generate a report on the latest workload capture, use the `DBMS_WORKLOAD_CAPTURE.GET_CAPTURE_INFO` procedure and the `DBMS_WORKLOAD_CAPTURE.REPORT` function:

```
DECLARE
  cap_id         NUMBER;
  cap_rpt        CLOB;
BEGIN
  cap_id  := DBMS_WORKLOAD_CAPTURE.GET_CAPTURE_INFO(dir => 'dec06');
  cap_rpt := DBMS_WORKLOAD_CAPTURE.REPORT(capture_id => cap_id,
                           format => DBMS_WORKLOAD_CAPTURE.TYPE_TEXT);
END;
/
```

In this example, the `GET_CAPTURE_INFO` procedure retrieves all information regarding the workload capture in the dec06 directory and returns the appropriate `cap_id` for the workload capture. The `REPORT` function generates a text report using the `cap_id` that was returned by the `GET_CAPTURE_INFO` procedure.

The `GET_CAPTURE_INFO` procedure uses the `dir` required parameter, which specifies the name of the workload capture directory object.

The `REPORT` 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` parameter required parameter specifies the report format. Valid values include DBMS_WORKLOAD_CAPTURE.TYPE_TEXT and DBMS_WORKLOAD_REPLAY.TYPE_HTML.

For information about how to use a workload capture report, see "Using a Workload Capture Report" on page 1-15.

> **See Also:** *Oracle Database PL/SQL Packages and Types Reference* for information about the `DBMS_WORKLOAD_CAPTURE` package

Using a Workload Capture Report

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 are 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 workload that was not captured due to version limitations
- Profile of the uncaptured workload that were excluded using defined filters
- Profile of the uncaptured workload that consists of background process or scheduled jobs

2 SQL Performance Analyzer

System changes that affect SQL execution plans, such as upgrading a database or adding new indexes, can severely impact SQL performance. As a result, DBAs spend considerable time identifying and fixing SQL statements that have regressed due to a change.

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 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 executions plan details along with tuning recommendations. As a result, DBAs can remedy any negative outcome before their end users are affected and can validate, with significant time and cost savings, that the system change to the production environment will result in net improvement.

You can use SQL Performance Analyzer to analyze the SQL performance impact of any type of system changes. Examples of common system changes for which you can use SQL Performance Analyzer include:

- Database upgrade
- Configuration changes to the operating system, hardware, or database
- Database initialization parameter changes
- Schema changes, for example, adding new indexes or materialized views
- Gathering optimizer statistics
- Validating SQL tuning actions, for example, creating SQL profiles

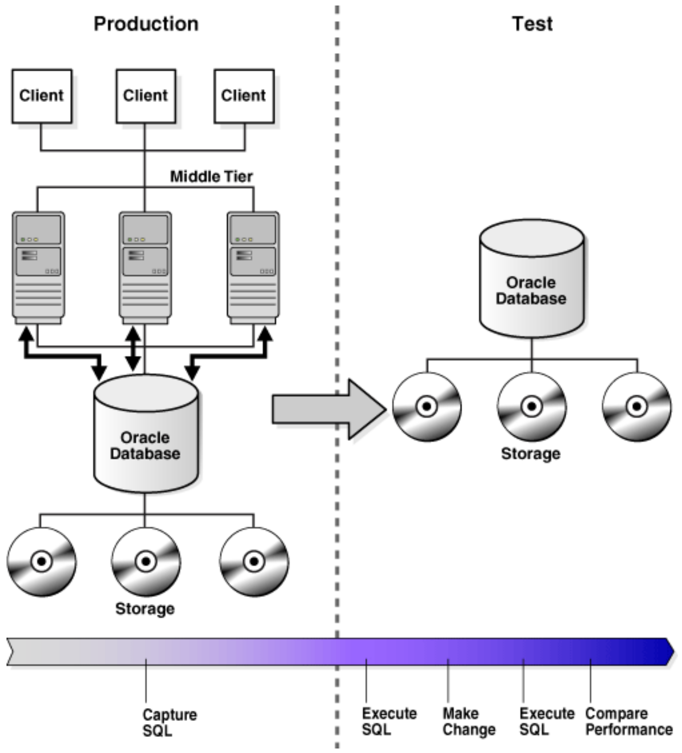
This chapter contains the following sections:

- Overview of SQL Performance Analyzer
- Capturing a SQL Workload
- Transporting a SQL Workload

Overview of SQL Performance Analyzer

As illustrated in Figure 2–1, SQL Performance Analyzer evaluates the impact of system changes on SQL performance through five main steps.

Figure 2–1 SQL Performance Analyzer Workflow



The steps of the SQL Performance Analyzer workflow are as follows:

1. Capture the SQL workload.

   You must first capture the set of SQL statements that represents the typical SQL workload on your production system in a SQL Tuning Set (STS). Later, you can conduct the SQL Performance Analyzer analysis on the same database where the workload was captured or on a different database. Because the analysis is resource-intensive, you would typically capture the workload on a production database and perform the analysis on a test database that closely resembles the production system. For more details about how to perform these actions, see "Capturing a SQL Workload" on page 2-3.

2. Measure the performance of the workload before the change.

   SQL Performance Analyzer executes the SQL statements captured in the SQL Tuning Set and generates execution plans and execution statistics for each statement. Only queries and the query part of DML statements are executed to avoid any side effect on the database. SQL Performance Analyzer executes SQL statements sequentially and in isolation from each other without any respect to their initial order of execution and concurrency. However, you can customize the order in which SQL Performance Analyzer executes the SQL queries. For example, you can start with the most expensive SQL statements in terms of response time.

3. Make a 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, optimizer statistics refresh, and so on.

4. Measure the performance of the workload after the change.

After you have made the planned change, SQL Performance Analyzer re-executes the SQL statements and produces execution plans and execution statistics for each SQL statement a second time. This execution result represents a new set of performance data that SQL Performance Analyzer uses for subsequent comparison.

5. Compare performance.

   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.

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

Capturing a SQL Workload

To capture a SQL workload that can be used with the SQL Performance Analyzer, you must capture a representative set of SQL statements on the production system and store them in a SQL Tuning Set. A SQL tuning set (STS) is a database object that is used to manage SQL workloads. The SQL tuning set can be used to store one or more SQL statements along with their execution context, including the text of the SQL, parsing schema under which the SQL statement can be compiled, bind values needed to execute the SQL statement, execution plan, number of times the SQL statement was executed, and so on.

You can load SQL statements into a SQL Tuning Set from different sources, including the cursor cache, Automatic Workload Repository (AWR), another SQL tuning set, and from a table or a view.

Note: Only SQL workload capture is currently supported in this release. Captured SQL workloads can be executed, and their performance can be measured and compared, on Oracle Database 11*g* Release 1 (11.1) and subsequent releases. For more information, see *Oracle Database 2 Day + Performance Tuning Guide* 11*g* Release 1 (11.1) or *Oracle Database Performance Tuning Guide* 11*g* Release 1 (11.1).

This section contains the following topics:

- Capturing a SQL Workload Using the Enterprise Manager
- Capturing a SQL Workload Using APIs

Capturing a SQL Workload Using the Enterprise Manager

To capture a SQL workload into a SQL tuning set using Oracle Enterprise Manager:

1. On the Database Performance page, under Additional Monitoring Links, click **SQL Tuning Sets**.

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

2. Click **Create**.

The Create SQL Tuning Set: Options page appears.

3. On the Create SQL Tuning Set: Options page:

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

   b. Optionally, in the Description field, enter a description of the SQL tuning set.

   c. Click **Next**.

   The Create SQL Tuning Set: Load Methods page appears.

4. On the Create SQL Tuning Set: Load Methods page, select a load method to collect and load SQL statements into the SQL tuning set:

   - You can load active and running SQL statements from the cursor cache into the SQL tuning set incrementally over a specified period of time.

     To incrementally capture active SQL statements from the cursor cache, select **Incrementally capture active SQL statements over a period of time from the cursor cache**.

     In the Duration field, specify the duration within which the SQL statement will collected.

     In the Frequency field, specify the frequency over which the active SQL statements will be collected repeatedly from the cursor cache.

   - To load SQL statements from another data source, select **Load SQL statements one time only**.

     Select the desired data source:

     – Cursor cache

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

       To use SQL statements from the cursor cache, from the Data Source list, select **Cursor Cache**.

     – AWR snapshots

       You can load SQL statements captured in AWR snapshots. This is useful when you want to collect SQL statements for specific snapshot periods of interest.

       To use SQL statements captured in AWR snapshots, from the Data Source list, select **AWR Snapshots**.

       From the AWR Snapshots list, select the period containing the AWR snapshots you want to include. You can choose to include only AWR snapshots captured in the last day, week, or month. Alternatively, you can choose to include all AWR snapshots.

     – Preserved snapshot sets

       You can load SQL statements captured in preserved snapshot sets. This is useful when you want to collect SQL statements from previously preserved snapshots that are representative of a time period during known performance levels.

To use SQL statements captured in preserved snapshot sets, from the Data Source list, select **Preserved Snapshot Sets**.

In the Preserved Snapshot Sets field, enter the name of the preserved snapshot set containing the AWR snapshots you want to include, or click the Search icon to select it from the Search and Select: Preserved Snapshot Sets window.

– User-defined workload

You can load SQL statements into a SQL tuning set by importing SQL statements from a table or view. This is useful if the workload you want to use is not currently running on the database or captured in existing AWR snapshots or preserved snapshot sets. There are no restrictions on which schema the workload resides in, the name of the table, or the number of tables that you can define. The only requirement is that the table must have a `sql_text` column that stores the text of the SQL statements.

To use SQL statements from a table or view, from the Data Source list, select **User-Defined Workload**. In the User-Defined Workload field, enter the name of the table or view you want to use, or click the Search icon to select it from the Search and Select: User Defined Workload window.

Click **Next**.

The Create SQL Tuning Set: Filter Options page appears.

5. After the load method is selected, you can apply filters to reduce the scope of the SQL statements found in the SQL tuning set:

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

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

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

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

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

   d. Click **Next**.

      The Create SQL Tuning Set: Schedule page appears.

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

   a. On the Create SQL Tuning Set: Schedule page, under Job Parameters, enter a name in the **Job Name** field if you do not want to use the system-generated job name.

   b. In the **Description** field, enter a description of the job.

   c. Under Scheduling, select **Immediately** to run the job immediately after it has been submitted, or **Later** to run the job at a later time as specified using the Time Zone, Date, and Time fields.

   d. Click **Next**.

The Create SQL Tuning Set: Review page appears.

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

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

8. Click **Submit**.

   The SQL Tuning Sets page appears.

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

   See Also: *Oracle Database 2 Day + Performance Tuning Guide* for information about creating a SQL tuning set using Enterprise Manager

Capturing a SQL Workload Using APIs

To capture a SQL workload into a SQL tuning set using APIs:

1. Use the `DBMS_SQLTUNE.CREATE_SQLSET` procedure to create an empty SQL tuning set in the database.

   For example, the following procedure creates a SQL tuning set named `STS_080125`:

```
BEGIN
  DBMS_SQLTUNE.CREATE_SQLSET(
    sqlset_name => 'STS_080125',
    description  => 'SQL tuning set for 2008/01/25');
END;
/
```

2. Populate the SQL tuning set with selected SQL statements. The data sources for populating a SQL tuning set include the cursor cache, workload repository, or another SQL tuning set.

   In the following example, the `CAPTURE_CURSOR_CACHE_SQLSET` procedure is used to load `STS_080125` with active SQL statements incrementally captured from the cursor cache every 5 minutes over a one-hour period.

```
EXEC DBMS_SQLTUNE.CAPTURE_CURSOR_CACHE_SQLSET( -
    sqlset_name => 'STS_080125', -
    time_limit => 3600, -
    repeat_interval  => 300);
```

See Also:

- *Oracle Database Performance Tuning Guide* for information about creating a SQL tuning set using APIs
- *Oracle Database PL/SQL Packages and Types Reference* for information about the `DBMS_SQLTUNE` package

Transporting a SQL Workload

After a SQL workload is captured, you can transport it to another system, such as a test system running Oracle Database 11*g*, where its performance can be measured and compared using SQL Performance Analyzer.

This section contains the following topics:

- Exporting a SQL Workload Using Enterprise Manager
- Transporting a SQL Workload Using APIs

Exporting a SQL Workload Using Enterprise Manager

To transport a SQL workload using Enterprise Manager, you need to first export the SQL tuning set(s) containing the SQL workload from the production system where it was captured, then import the SQL tuning set(s) into another system where it can be analyzed by SQL Performance Analyzer.

Note: For Enterprise Manager, only exporting a SQL workload is currently supported in this release. Exported SQL workloads can be imported using Enterprise Manager on Oracle Database 11*g* Release 1 (11.1) and subsequent releases, or by using APIs. For more information, see *Oracle Database 2 Day + Performance Tuning Guide* 11*g* Release 1 (11.1).

To export a SQL tuning set:

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

   The Advisor Central page appears.

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

   The SQL Tuning Advisor Links page appears.

3. Click **SQL Tuning Sets**.

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

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

   The Export SQL Tuning Set page appears.

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

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

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

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

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

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

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

   By default, `SYSAUX` is used.

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

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

10. Under Schedule, select:

    - **Immediately** to run the job immediately after it has been submitted.

- **Later** to run the job at a later time as specified by selecting or entering values in the Time Zone, Date, and Time fields.

11. Click **OK**.

    The SQL Tuning Sets page appears.

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

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

    See Also: *Oracle Database 2 Day + Performance Tuning Guide* for information about transporting a SQL tuning set using Enterprise Manager

Transporting a SQL Workload Using APIs

To transport a SQL workload using APIs, you need to first move the SQL tuning set(s) containing the SQL workload from the production system where it was captured to a staging table, then export the SQL tuning set(s) from the staging table into another system where it can be analyzed by SQL Performance Analyzer.

To transport a SQL Tuning Set:

1. Use the `CREATE_STGTAB_SQLSET` procedure to create a staging table where the SQL tuning sets will be exported.

   The following example shows how to create a staging table named `staging_table`. Table names are case-sensitive.

```
BEGIN
  DBMS_SQLTUNE.CREATE_STGTAB_SQLSET( table_name => 'staging_table' );
END;
/
```

2. Use the `PACK_STGTAB_SQLSET` procedure to export SQL tuning sets into the staging table.

   The following example shows how to export a SQL tuning set named `STS_080125` to the staging table.

```
BEGIN
  DBMS_SQLTUNE.PACK_STGTAB_SQLSET(
      sqlset_name  => 'STS_080125',
      staging_table_name => 'staging_table');
END;
/
```

3. Move the staging table to the system where the SQL tuning sets will be imported using the mechanism of choice (such as datapump or database link).

4. On the system where the SQL Tuning Sets will be imported, use the `UNPACK_STGTAB_SQLSET` procedure to import SQL Tuning Sets from the staging table.

   The following example shows how to import SQL Tuning Sets contained in the staging table.

```
BEGIN
  DBMS_SQLTUNE.UNPACK_STGTAB_SQLSET(
      sqlset_name  => '%',
      replace  => TRUE,
      staging_table_name => 'staging_table');
END;
```

```
/
```

See Also:

- *Oracle Database Performance Tuning Guide* for information about transporting a SQL tuning set using APIs
- *Oracle Database PL/SQL Packages and Types Reference* for information about the `DBMS_SQLTUNE` package