6

Gathering Database Statistics

This chapter describes how to gather database statistics for Oracle Database and contains the following topics:

- About Gathering Database Statistics
- Managing the Automatic Workload Repository
- Generating Automatic Workload Repository Reports
- Generating Performance Hub Active Report

About Gathering Database Statistics

Oracle Database automatically persists the cumulative and delta values for most of the statistics at all levels (except the session level) in the Automatic Workload Repository (AWR). This process is repeated on a regular time period and the results are captured in an AWR snapshot. The delta values captured by the snapshot represent the changes for each statistic over the time period.

A statistical baseline is a collection of statistic rates usually taken over a time period when the system is performing well at an optimal level. Use statistical baselines to diagnose performance problems by comparing statistics captured in a baseline to those captured during a period of poor performance. This enables you to identify specific statistics that may have increased significantly and could be the cause of the problem. AWR supports the capture of baseline data by enabling you to specify and preserve a pair or range of AWR snapshots as a baseline.

A metric is typically the rate of change in a cumulative statistic. You can measure this rate against a variety of units, including time, transactions, or database calls. For example, the number database calls per second is a metric. Metric values are exposed in some `V$` views, where the values are the averages over a fairly small time interval, typically 60 seconds. A history of recent metric values is available through `V$` views, and some data is also persisted by AWR snapshots.

The following sections describe various Oracle Database features that enable you to more effectively gather database statistics:

- Automatic Workload Repository
- Snapshots
- Baselines
- Space Consumption

> **Note:**
>
> - Data visibility and privilege requirements may differ when using AWR features with pluggable databases (PDBs). For information about how manageability features, including the AWR features, work in a multitenant container database (CDB), see *Oracle Multitenant Administrator's Guide*.
> - License for Oracle Diagnostic Pack is required to use the AWR features described in this chapter.

Automatic Workload Repository

AWR collects, processes, and maintains performance statistics for problem detection and self-tuning purposes. This gathered data is stored both in memory and in the database, and is displayed in both reports and views.

The statistics collected and processed by AWR include:

- Object statistics that determine both access and usage statistics of database segments
- Time model statistics based on time usage for activities, displayed in the `V$SYS_TIME_MODEL` and `V$SESS_TIME_MODEL` views
- Some of the system and session statistics collected in the `V$SYSSTAT` and `V$SESSTAT` views
- SQL statements that are producing the highest load on the system, based on criteria such as elapsed time and CPU time
- Active Session History (ASH) statistics, representing the history of recent sessions activity

> **See Also:**
>
> - "About Database Statistics" for information about the various types of database statistics
> - *Oracle Database Reference* for more information about the views `V$SYS_TIME_MODEL`, `V$SESS_TIME_MODEL`, `V$SYSSTAT`, and `V$SESSTAT`

Snapshots

Snapshots are sets of historical data for specific time periods that are used for performance comparisons by Automatic Database Diagnostic Monitor (ADDM). By default, Oracle Database automatically generates snapshots of the performance data once every hour and retains the statistics in AWR for 8 days. You can also manually create snapshots or change the snapshot retention period, but it is usually not necessary.

AWR compares the difference between snapshots to determine which SQL statements to capture based on the effect on the system load. This reduces the number of SQL statements that must be captured over time. After the snapshots are created, ADDM analyzes the data captured in the snapshots to perform its performance analysis.

> **See Also:**
>
> "Managing Snapshots" for information about managing snapshots

Baselines

A baseline is a set of snapshots from a specific time period that is preserved for comparison with other snapshots when a performance problem occurs. The snapshots contained in a baseline are excluded from the automatic AWR purging process and are retained indefinitely.

There are several types of available baselines:

- Fixed Baselines
- Moving Window Baselines
- Baseline Templates

Fixed Baselines

A fixed baseline corresponds to a fixed, contiguous time period in the past that you specify. Before creating a fixed baseline, carefully consider the time period you choose as a baseline, because the baseline should represent the system operating at an optimal level. In the future, you can compare the baseline with other baselines or snapshots captured during periods of poor performance to analyze performance degradation over time.

> **See Also:**
>
> "Managing Baselines" for information about managing fixed baselines

Moving Window Baselines

A moving window baseline corresponds to all AWR data that exists within the AWR retention period. The database can use AWR data in the entire AWR retention period to compute metric threshold values.

Oracle Database automatically maintains a system-defined moving window baseline. The default window size for the system-defined moving window baseline is the current AWR retention period, which by default is 8 days. You can resize the moving window baseline by changing the number of days in the moving window to a value that is equal to or less than the number of days in the AWR retention period. Therefore, to increase the size of a moving window, you must first increase the AWR retention period accordingly.

> **See Also:**
>
> "Resizing the Default Moving Window Baseline" for information about resizing a moving window baseline

Baseline Templates

Baseline templates enable you to create baselines for a contiguous time period in the future. There are two types of baseline templates:

- Single Baseline Templates
- Repeating Baseline Templates

> **See Also:**
>
> "Managing Baseline Templates" for information about managing baseline templates

Single Baseline Templates

Use a single baseline template to create a baseline for a single contiguous time period in the future. This is useful if you know beforehand of a time period that you intend to capture in the future. For example, you may want to capture AWR data during a system test that is scheduled for the upcoming weekend. In this case, you can create a single baseline template to automatically capture the time period when the test occurs.

Repeating Baseline Templates

Use a repeating baseline template to create and drop baselines based on a repeating time schedule. This is useful if you want Oracle Database to automatically capture a contiguous time period on an ongoing basis. For example, you may want to capture AWR data during every Monday morning for a month. In this case, you can create a repeating baseline template to automatically create baselines on a repeating schedule for every Monday, and automatically remove older baselines after a specified expiration interval, such as one month.

Space Consumption

The space consumed by AWR is determined by several factors:

- Number of active sessions in the database at any given time
- Snapshot interval

  The snapshot interval determines the frequency at which snapshots are captured. A smaller snapshot interval increases the frequency, which increases the volume of data collected by AWR.
- Historical data retention period

  The retention period determines how long this data is retained before being purged. A longer retention period increases the space consumed by AWR.

By default, Oracle Database captures snapshots once every hour and retains them in the database for 8 days. With these default settings, a typical system with an average of 10 concurrent active sessions can require approximately 200 to 300 MB of space for its AWR data.

To reduce AWR space consumption, increase the snapshot interval and reduce the retention period. When reducing the retention period, note that several Oracle Database self-managing features depend on AWR data for proper functioning. Not having enough data can affect the validity and accuracy of these components and features, including:

- Automatic Database Diagnostic Monitor (ADDM)
- SQL Tuning Advisor
- Undo Advisor
- Segment Advisor

If possible, Oracle recommends that you set the AWR retention period large enough to capture at least one complete workload cycle. If your system experiences weekly workload cycles, such as OLTP workload during weekdays and batch jobs during the weekend, then you do not need to change the default AWR retention period of 8 days. However, if your system is subjected to a monthly peak load during month-end book closing, then you may need to set the retention period to one month.

Under exceptional circumstances, you can disable automatic snapshot collection by setting the snapshot interval to 0. Under this condition, the automatic collection of the workload and statistical data is stopped, and most of the Oracle Database self-management functionality is not operational. In addition, you cannot manually create snapshots. For this reason, Oracle strongly recommends against disabling automatic snapshot collection.

> **Note:**
>
> Oracle Database uses the `SYSAUX` tablespace to store AWR data by default.

> **See Also:**
>
> "Modifying Snapshot Settings" for information about changing the default values for the snapshot interval and retention period

Managing the Automatic Workload Repository

This section describes how to manage AWR features of Oracle Database and contains the following topics:

- Enabling the Automatic Workload Repository
- Managing Snapshots
- Managing Baselines
- Managing Baseline Templates
- Transporting Automatic Workload Repository Data to Another System
- Using Automatic Workload Repository Views
- Managing Automatic Workload Repository in a Multitenant Environment
- Managing Automatic Workload Repository in Active Data Guard Standby Databases

> **See Also:**
>
> "Automatic Workload Repository" for a description of AWR

Enabling the Automatic Workload Repository

Gathering database statistics using AWR is enabled by default and is controlled by the `STATISTICS_LEVEL` initialization parameter.

The database can be configured so that the AWR reports use the database time zone instead of the operating system time zone. Refer to TIME_AT_DBTIMEZONE.

To enable statistics gathering by AWR:

- Set the `STATISTICS_LEVEL` parameter to `TYPICAL` or `ALL`.

  The default setting for this parameter is `TYPICAL`.

Setting `STATISTICS_LEVEL` to `BASIC` disables many Oracle Database features, including AWR, and is not recommended. If `STATISTICS_LEVEL` is set to `BASIC`, you can still manually capture AWR statistics using the `DBMS_WORKLOAD_REPOSITORY` package. However, because in-memory collection of many system statistics—such as segments statistics and memory advisor information—will be disabled, the statistics captured in these snapshots may not be complete.

> **See Also:**
>
> *Oracle Database Reference* for information about the `STATISTICS_LEVEL` initialization parameter

Managing Snapshots

By default, Oracle Database generates snapshots once every hour, and retains the statistics in the workload repository for 8 days. When necessary, you can manually create or drop snapshots and modify snapshot settings.

This section describes how to manage snapshots and contains the following topics:

- User Interfaces for Managing Snapshots
- Creating Snapshots
- Dropping Snapshots
- Modifying Snapshot Settings

> **See Also:**
>
> "Snapshots" for information about snapshots

User Interfaces for Managing Snapshots

The primary interface for managing snapshots is Oracle Enterprise Manager Cloud Control (Cloud Control). Whenever possible, you should manage snapshots using Cloud Control.

If Cloud Control is unavailable, then manage snapshots using the `DBMS_WORKLOAD_REPOSITORY` package in the command-line interface. The DBA role is required to invoke the `DBMS_WORKLOAD_REPOSITORY` procedures.

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

Creating Snapshots

By default, Oracle Database automatically generates snapshots once every hour. However, you may want to manually create snapshots to capture statistics at times different from those of the automatically generated snapshots.

Creating Snapshots Using the Command-Line Interface

To manually create snapshots, use the `CREATE_SNAPSHOT` procedure. The following example shows a `CREATE_SNAPSHOT` procedure call.

```
BEGIN
  DBMS_WORKLOAD_REPOSITORY.CREATE_SNAPSHOT ();
END;
/
```

In this example, a snapshot is created immediately on the local database instance. To view information about an existing snapshot, use the `DBA_HIST_SNAPSHOT` view.

> **Note:**
>
> You can specify value for the `flush_level` parameter of the `CREATE_SNAPSHOT` procedure to either TYPICAL or ALL. The default value for the flush level is TYPICAL.
>
> The flush level signifies the breadth and depth of the AWR statistics to be captured. If you want to capture all the AWR statistics, then set the flush level to ALL. If you want to skip few AWR statistics, such as, SQL statistics, segment statistics, and files and tablespace statistics for performance reasons, then set the flush level to TYPICAL.

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

Dropping Snapshots

By default, Oracle Database automatically purges snapshots that have been stored in AWR for over 8 days. However, you may want to manually drop a range of snapshots to free up space.

Dropping Snapshots Using the Command-Line Interface

To manually drop a range of snapshots, use the `DROP_SNAPSHOT_RANGE` procedure. The following example shows a `DROP_SNAPSHOT_RANGE` procedure call.

```
BEGIN
  DBMS_WORKLOAD_REPOSITORY.DROP_SNAPSHOT_RANGE (low_snap_id  => 22,
                                                high_snap_id => 32,
                                                dbid         => 3310949047);
END;
/
```

In the example, snapshots with snapshot IDs ranging from 22 to 32 are dropped immediately from the database instance with the database identifier of `3310949047`. Any ASH data that were captured during this snapshot range are also purged.

Tip:

To determine which snapshots to drop, use the `DBA_HIST_SNAPSHOT` view to review the existing snapshots

See Also:

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

Modifying Snapshot Settings

You can adjust the interval, retention period, and number of top SQL to flush for snapshot generation, but note that this can affect the precision of the Oracle Database diagnostic tools.

Modifying Snapshot Settings Using the Command-Line Interface

You can modify snapshot settings using the following parameters of the `DBMS_WORKLOAD_REPOSITORY.MODIFY_SNAPSHOT_SETTINGS` procedure:

Parameter	Description
`INTERVAL`	This setting affects how often the database automatically generates snapshots.
`RETENTION`	This setting affects how long the database stores snapshots in AWR.

Parameter	Description
TOPNSQL	This setting affects the number of top SQL to flush for each SQL criteria (elapsed time, CPU time, parse calls, sharable memory, and version count). This setting is not affected by the statistics/flush level and overrides the system default behavior for AWR SQL collection. It is possible to set the value for this setting to MAXIMUM to capture the complete set of SQL in the shared SQL area, though doing so (or by setting the value to a very high number) may lead to possible space and performance issues because there will be more data to collect and store.

The following example shows how to modify snapshot settings using the DBMS_WORKLOAD_REPOSITORY.MODIFY_SNAPSHOT_SETTINGS procedure:

```
BEGIN
  DBMS_WORKLOAD_REPOSITORY.MODIFY_SNAPSHOT_SETTINGS( retention  => 43200,
                                                     interval   => 30,
                                                     topnsql    => 100,
                                                     dbid       => 3310949047);
END;
/
```

In this example, snapshot settings for the database with the database identifier of 3310949047 are modified as follows:

- The retention period is specified as 43200 minutes (30 days).
- The interval between each snapshot is specified as 30 minutes.
- The number of top SQL to flush for each SQL criteria is specified as 100.

To get information about the current snapshot settings for your database, use the DBA_HIST_WR_CONTROL view as shown in the following example:

```
SQL> select snap_interval, retention from DBA_HIST_WR_CONTROL;

SNAP_INTERVAL               RETENTION
---------------------------------------
+00000 01:00:00.0        +00008 00:00:00.0
```

The snap_interval and retention values are displayed in the format:

```
+[days] [hours]:[minutes]:[seconds].[nanoseconds]
```

> **See Also:**
>
> - *Oracle Database PL/SQL Packages and Types Reference* for more information about the DBMS_WORKLOAD_REPOSITORY.MODIFY_SNAPSHOT_SETTINGS procedure
> - *Oracle Database Reference* for more information about the DBA_HIST_WR_CONTROL view

Managing Baselines

By default, Oracle Database automatically maintains a system-defined moving window baseline. When necessary, you can manually create, drop, or rename a baseline and view the baseline threshold settings. Additionally, you can manually resize the window size of the moving window baseline.

This section describes how to manage baselines and contains the following topics:

- User Interface for Managing Baselines
- Creating a Baseline
- Dropping a Baseline
- Renaming a Baseline
- Displaying Baseline Metrics
- Resizing the Default Moving Window Baseline

> **See Also:**
>
> "Baselines" for information about baselines

User Interface for Managing Baselines

The primary interface for managing baselines is Oracle Enterprise Manager Cloud Control (Cloud Control). Whenever possible, manage baselines using Cloud Control.

If Cloud Control is unavailable, then manage baselines using the `DBMS_WORKLOAD_REPOSITORY` package in the command-line interface. The DBA role is required to invoke the `DBMS_WORKLOAD_REPOSITORY` procedures.

> **See Also:**
>
> - *Oracle Database 2 Day + Performance Tuning Guide* for more information about managing baselines using Cloud Control
> - *Oracle Database PL/SQL Packages and Types Reference* for information about the `DBMS_WORKLOAD_REPOSITORY` package

Creating a Baseline

By default, Oracle Database automatically maintains a system-defined moving window baseline. However, you may want to manually create a fixed baseline that represents the system operating at an optimal level, so you can compare it with other baselines or snapshots captured during periods of poor performance.

To create baselines using command-line interface, use the `CREATE_BASELINE` procedure as shown in the following example:

```
BEGIN
    DBMS_WORKLOAD_REPOSITORY.CREATE_BASELINE (start_snap_id => 270,
                                             end_snap_id   => 280,
                                             baseline_name => 'peak baseline',
                                             dbid          => 3310949047,
                                             expiration    => 30);
END;
/
```

In this example, a baseline is created on the database instance with the database identifier of `3310949047` with the following settings:

- The start snapshot sequence number is 270.
- The end snapshot sequence number is 280.
- The name of baseline is `peak baseline`.
- The expiration of the baseline is 30 days.

Oracle Database automatically assigns a unique ID to the new baseline when the baseline is created.

Tip:

To determine the range of snapshots to include in a baseline, use the `DBA_HIST_SNAPSHOT` view to review the existing snapshots

See Also:

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

Dropping a Baseline

To conserve disk space, consider periodically dropping a baseline that is no longer being used. The snapshots associated with a baseline are retained indefinitely until you explicitly drop the baseline or the baseline has expired.

To drop a baseline using command-line interface, use the `DROP_BASELINE` procedure as shown in the following example:

```
BEGIN
  DBMS_WORKLOAD_REPOSITORY.DROP_BASELINE (baseline_name => 'peak baseline',
                                          cascade       => FALSE,
                                          dbid          => 3310949047);
END;
/
```

In the example, the baseline `peak baseline` is dropped from the database instance with the database identifier of `3310949047` and the associated snapshots are preserved.

> **Tip:**
>
> To determine which baseline to drop, use the `DBA_HIST_BASELINE` view to review the existing baselines.

> **Tip:**
>
> To drop the associated snapshots along with the baseline, set the `cascade` parameter to `TRUE`.

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

Renaming a Baseline

To rename a baseline using command-line interface, use the `RENAME_BASELINE` procedure. The following example shows a `RENAME_BASELINE` procedure call.

```
BEGIN
    DBMS_WORKLOAD_REPOSITORY.RENAME_BASELINE (old_baseline_name => 'peak baseline',
                                             new_baseline_name => 'peak mondays',
                                             dbid              => 3310949047);
END;
/
```

In this example, the name of the baseline on the database instance with the database identifier of `3310949047` is renamed from `peak baseline` to `peak mondays`.

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

Displaying Baseline Metrics

To display the summary statistics for metric values in a baseline period using the command-line interface, use the `SELECT_BASELINE_METRICS` function:

```
DBMS_WORKLOAD_REPOSITORY.SELECT_BASELINE_METRICS (baseline_name IN VARCHAR2,
                                                 dbid          IN NUMBER DEFAULT NULL,
                                                 instance_num  IN NUMBER DEFAULT NULL)
   RETURN awr_baseline_metric_type_table PIPELINED;
```

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

Resizing the Default Moving Window Baseline

By default, Oracle Database automatically maintains a system-defined moving window baseline. The default window size for the system-defined moving window baseline is the current AWR retention period, which by default is 8 days. In certain circumstances, you may want to modify the window size of the default moving window baseline.

To modify the window size of the default moving window baseline using the command-line interface, use the `MODIFY_BASELINE_WINDOW_SIZE` procedure as shown in the following example:

```
BEGIN
    DBMS_WORKLOAD_REPOSITORY.MODIFY_BASELINE_WINDOW_SIZE (window_size => 30,
                                                          dbid        => 3310949047);
END;
/
```

In this example, the default moving window is resized to 30 days on the database instance with the database identifier of `3310949047`.

> **Note:**
>
> The window size must be set to a value that is equal to or less than the value of the AWR retention setting. To set a window size that is greater than the current AWR retention period, you must first increase the value of the `retention` parameter as described in "Modifying Snapshot Settings".

> **See Also:**
>
> - "Moving Window Baselines" for information about moving window baselines
> - *Oracle Database PL/SQL Packages and Types Reference* for information about the `DBMS_WORKLOAD_REPOSITORY` package

Managing Baseline Templates

Baseline templates enable you to automatically create baselines to capture specified time periods in the future. This section describes how to manage baseline templates and contains the following topics:

- User Interfaces for Managing Baseline Templates
- Creating a Single Baseline Template
- Creating a Repeating Baseline Template

- Dropping a Baseline Template

> **See Also:**
>
> "Baseline Templates" for information about baseline templates

User Interfaces for Managing Baseline Templates

The primary interface for managing baseline templates is Oracle Enterprise Manager Cloud Control (Cloud Control). Whenever possible, manage baseline templates using Cloud Control.

If Cloud Control is unavailable, then manage baseline templates using the `DBMS_WORKLOAD_REPOSITORY` package in the command-line interface. The DBA role is required to invoke the `DBMS_WORKLOAD_REPOSITORY` procedures.

> **See Also:**
>
> *Oracle Database 2 Day + Performance Tuning Guide* for more information about managing baseline templates using Cloud Control

Creating a Single Baseline Template

You can use a single baseline template to create a baseline during a single, fixed time interval in the future. For example, you can create a single baseline template to generate a baseline that is captured on April 2, 2012 from 5:00 p.m. to 8:00 p.m.

To create a single baseline template using command-line interface, use the `CREATE_BASELINE_TEMPLATE` procedure as shown in the following example:

```
BEGIN
    DBMS_WORKLOAD_REPOSITORY.CREATE_BASELINE_TEMPLATE (start_time    => '2012-04-02
17:00:00 PST',
                                                       end_time      => '2012-04-02
20:00:00 PST',
                                                       baseline_name =>
'baseline_120402',
                                                       template_name =>
'template_120402',
                                                       expiration    => 30,
                                                       dbid          => 3310949047);
END;
/
```

In this example, a baseline template named `template_120402` is created that will generate a baseline named `baseline_120402` for the time period from 5:00 p.m. to 8:00 p.m. on April 2, 2012 on the database with a database ID of `3310949047`. The baseline will expire after 30 days.

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

Creating a Repeating Baseline Template

You can use a repeating baseline template to automatically create baselines that repeat during a particular time interval over a specific period in the future. For example, you can create a repeating baseline template to generate a baseline that repeats every Monday from 5:00 p.m. to 8:00 p.m. for the year 2012.

To create a repeating baseline template using command-line, use the `CREATE_BASELINE_TEMPLATE` procedure as shown in the following example:

```
BEGIN
    DBMS_WORKLOAD_REPOSITORY.CREATE_BASELINE_TEMPLATE (day_of_week => 'monday',
                                                       hour_in_day => 17,
                                                       duration => 3, expiration => 30,
                                                       start_time => '2012-04-02
17:00:00 PST',
                                                       end_time => '2012-12-31 20:00:00
PST',
                                                       baseline_name_prefix =>
'baseline_2012_mondays_',
                                                       template_name =>
'template_2012_mondays',
                                                       dbid => 3310949047);
END;
/
```

In this example, a baseline template named `template_2012_mondays` is created that will generate a baseline on every Monday from 5:00 p.m. to 8:00 p.m. beginning on April 2, 2012 at 5:00 p.m. and ending on December 31, 2012 at 8:00 p.m. on the database with a database ID of `3310949047`. Each of the baselines will be created with a baseline name with the prefix `baseline_2012_mondays_` and will expire after 30 days.

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

Dropping a Baseline Template

Periodically, you may want to remove baselines templates that are no longer used to conserve disk space.

To drop a baseline template using command-line, use the `DROP_BASELINE_TEMPLATE` procedure as shown in the following example:

```
BEGIN
  DBMS_WORKLOAD_REPOSITORY.DROP_BASELINE_TEMPLATE (template_name =>
'template_2012_mondays',
```

```
                                                dbid          => 3310949047);
END;
/
```

In this example, the baseline template named `template_2012_mondays` is dropped from the database instance with the database identifier of `3310949047`.

> **Tip:**
>
> To determine which baseline template to drop, use the `DBA_HIST_BASELINE_TEMPLATE` view to review the existing baseline templates.

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

Transporting Automatic Workload Repository Data to Another System

Oracle Database enables you to transport AWR data between systems. This is useful in cases where you want to use a separate system to perform analysis of AWR data, so as to reduce the overhead caused by performance analysis on a production system.

To transport AWR data from one system to another, first export the AWR data from the database on the source system, and then import it into the database on the target system.

This section contains the following topics:

- "Exporting AWR Data"
- "Importing AWR Data"

Exporting AWR Data

The `awrextr.sql` script exports AWR data for a range of snapshots from the database into a Data Pump export file. After it is created, you can transport this dump file to another database where you can import the exported AWR data. To run the `awrextr.sql` script, you must be connected to the database as the `SYS` user.

To export AWR data:

1. At the SQL prompt, enter:

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

   A list of the databases in the AWR schema is displayed.
2. Specify the database from which AWR data needs to be exported:

   ```
   Enter value for db_id: 1377863381
   ```

   In this example, the database with the database identifier of `1377863381` is specified.
3. Specify the number of days for which you want to view all the snapshot IDs:

```
Enter value for num_days: 2
```

In this example, all the snapshots captured in the last 2 days are displayed.

4. Define the range of snapshots for which AWR data needs to be exported by specifying the beginning and the ending snapshot IDs:

```
Enter value for begin_snap: 30
Enter value for end_snap: 40
```

In this example, the snapshot ID of 30 is specified as the beginning snapshot, and the snapshot ID of 40 is specified as the ending snapshot.

A list of directory objects is displayed.

5. Specify the directory object pointing to the directory where the export dump file needs to be stored:

```
Enter value for directory_name: DATA_PUMP_DIR
```

In this example, the directory object `DATA_PUMP_DIR` is specified that points to the directory `ORACLE_HOME/rdbms/log`, where `ORACLE_HOME` is `/u01/app/oracle/product/database_release_number/dbhome_1`.

6. Specify a name for the export dump file without the file extension. By default, the file extension of `.dmp` is used.

```
Enter value for file_name: awrdata_30_40
```

In this example, an export dump file named `awrdata_30_40.dmp` is created in the directory specified in the directory object `DATA_PUMP_DIR`:

```
Dump file set for SYS.SYS_EXPORT_TABLE_01 is:
/u01/app/oracle/product/database_release_number/dbhome_1/rdbms/log/awrdata_30_40.dmp
Job "SYS"."SYS_EXPORT_TABLE_01" successfully completed at 08:58:20
```

Depending on the amount of AWR data that must be exported, the AWR export operation may take a while to complete. After the dump file is created, you can use Data Pump to transport the file to another system.

> **See Also:**
>
> *Oracle Database Utilities* for information about using Data Pump

Importing AWR Data

After the export dump file is transported to the target system, import the exported AWR data using the `awrload.sql` script. The `awrload.sql` script creates a staging schema where the snapshot data is transferred from the Data Pump file into the database. The data is then transferred from the staging schema into the appropriate AWR tables. To run the `awrload.sql` script, you must be connected to the database as the `SYS` user.

To import AWR data:

1. At the SQL prompt, enter:

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

A list of directory objects is displayed.

2. Specify the directory object pointing to the directory where the export dump file is located:

```
Enter value for directory_name: DATA_PUMP_DIR
```

In this example, the directory object `DATA_PUMP_DIR` is specified that points to the directory where the export dump file is located.

3. Specify the name of the export dump file without the file extension. By default, the file extension of `.dmp` is used.

```
Enter value for file_name: awrdata_30_40
```

In this example, the export dump file named `awrdata_30_40.dmp` is selected.

4. Specify the name of the staging schema where the AWR data needs to be imported:

```
Enter value for schema_name: AWR_STAGE
```

In this example, a staging schema named `AWR_STAGE` is created.

5. Specify the default tablespace for the staging schema:

```
Enter value for default_tablespace: SYSAUX
```

In this example, the `SYSAUX` tablespace is specified.

6. Specify the temporary tablespace for the staging schema:

```
Enter value for temporary_tablespace: TEMP
```

In this example, the `TEMP` tablespace is specified.

7. First the AWR data is imported into the `AWR_STAGE` schema and then it is transferred to the AWR tables in the `SYS` schema:

```
Processing object type TABLE_EXPORT/TABLE/CONSTRAINT/CONSTRAINT
Completed 113 CONSTRAINT objects in 11 seconds
Processing object type TABLE_EXPORT/TABLE/CONSTRAINT/REF_CONSTRAINT
Completed 1 REF_CONSTRAINT objects in 1 seconds
Job "SYS"."SYS_IMPORT_FULL_03" successfully completed at 09:29:30
... Dropping AWR_STAGE user
End of AWR Load
```

Depending on the amount of AWR data that must be imported, the AWR import operation may take a while to complete. After AWR data is imported, the staging schema will be dropped automatically.

Using Automatic Workload Repository Views

Typically, you would view AWR data using Oracle Enterprise Manager Cloud Control (Cloud Control) or AWR reports. However, you can also view historical data stored in the AWR using the following `DBA_HIST` views.

> **Note:**
>
> In a multitenant environment, these DBA_HIST views can also be interchanged with the AWR_ROOT views and AWR_PDB views at the CDB level and the PDB level respectively. For example, you can use the AWR_PDB_ACTIVE_SESS_HISTORY view for retrieving the AWR data about the active session history at the PDB level, which is equivalent to the DBA_HIST_ACTIVE_SESS_HISTORY view in an independent database in a non-multitenant environment. The AWR_PDB views will not show any AWR data, if the PDB level snapshots have not been collected.

Table 6-1 DBA_HIST Views

DBA_HIST View	Description
DBA_HIST_ACTIVE_SESS_HISTORY	Displays the history of the contents of the in-memory active session history for recent system activity.
DBA_HIST_BASELINE	Displays information about the baselines captured on the system, such as the time range of each baseline and the baseline type.
DBA_HIST_BASELINE_DETAILS	Displays details about a specific baseline.
DBA_HIST_BASELINE_TEMPLATE	Displays information about the baseline templates used by the system to generate baselines.
DBA_HIST_CON_SYS_TIME_MODEL	Displays historical system time model statistics, including OLAP timed statistics.
DBA_HIST_CON_SYSMETRIC_HIST	Displays the historical information about the system metric values.
DBA_HIST_CON_SYSMETRIC_SUMM	Displays history of the statistical summary of all the metric values in the system metrics for the long duration (60 seconds) group.
DBA_HIST_CON_SYSSTAT	Displays historical system statistics, including OLAP kernel statistics.
DBA_HIST_CON_SYSTEM_EVENT	Displays historical information about the total waits for an event.
DBA_HIST_DATABASE_INSTANCE	Displays information about the database environment.
DBA_HIST_DB_CACHE_ADVICE	Displays historical predictions of the number of physical reads for the cache size corresponding to each row.
DBA_HIST_DISPATCHER	Displays historical information for each dispatcher process at the time of the snapshot.
DBA_HIST_DYN_REMASTER_STATS	Displays statistical information about the dynamic remastering process.
DBA_HIST_IOSTAT_DETAIL	Displays historical I/O statistics aggregated by file type and function.
DBA_HIST_RSRC_PDB_METRIC	Displays historical information about the Resource Manager metrics for pluggable databases (PDBs) for the past one hour.
DBA_HIST_RSRC_METRIC	Displays historical information about the Resource Manager metrics for consumer groups for the past one hour.
DBA_HIST_SHARED_SERVER_SUMMARY	Displays historical information for shared servers, such as shared server activity, common queues and dispatcher queues.
DBA_HIST_SNAPSHOT	Displays information on snapshots in the system.
DBA_HIST_SQL_PLAN	Displays the SQL execution plans.
DBA_HIST_WR_CONTROL	Displays the settings for controlling AWR.
DBA_HIST_WR_SETTINGS	Displays the settings and metadata of the AWR.
DBA_HIST_PROCESS_WAITTIME	Displays CPU and wait time for a process type.

> **See Also:**
>
> *Oracle Database Reference* for more information about the `DBA_HIST` views

Managing Automatic Workload Repository in a Multitenant Environment

A centralized Automatic Workload Repository (AWR) stores the performance data related to CDB and PDBs in a multitenant environment.

CDBs and individual PDBs can store, view, and manage AWR data. You can take an AWR snapshot at the CDB level or at the PDB level.

> **Note:**
>
> A multitenant container database is the only supported architecture in Oracle Database 21c and later releases. While the documentation is being revised, legacy terminology may persist. In most cases, "database" and "non-CDB" refer to a CDB or PDB, depending on context. In some contexts, such as upgrades, "non-CDB" refers to a non-CDB from a previous release.

This section contains the following topics:

Categorization of AWR Data in a Multitenant Environment

In a multitenant environment, AWR data falls into different categories.

The categories are as follows:

- General AWR data

  This data has no security implications. It is safe to be shared among all tenants in a CDB. This data is accessible by all PDBs and is captured in both CDB-level and PDB-level snapshots. Examples of general AWR data include the names of statistics, latches, and parameters.

- AWR data for a CDB

  This category aggregates data for all tenants in a CDB. This data contains the status of the database as a whole and is useful only for the CDB administrator. This data is captured only in the CDB-level snapshots.

- AWR data for individual PDBs

  This data describes the individual PDBs in a CDB. It shows container-specific data that represents the contribution of each individual PDB to the whole database instance. Therefore, this data is useful for both the CDB and the PDB administrators. This data is captured in both CDB-level and PDB-level snapshots.

AWR Data Storage and Retrieval in a Multitenant Environment

This section describes the process of managing snapshots, and exporting and importing AWR data in a multitenant environment.

Managing Snapshots

Starting with Oracle Database 12*c* Release 2 (12.2), you can take an AWR snapshot at a CDB-level, that is, on a CDB root, as well as at a PDB-level, that is, on an individual PDB. By default, the CDB-level snapshot data is stored in the `SYSAUX` tablespace of a CDB root and the PDB-level snapshot data is stored in the `SYSAUX` tablespace of a PDB.

A CDB-level snapshot contains information about the CDB statistics as well as all the PDB statistics, such as ASH, SQL statistics, and file statistics. The CDB administrator can perform CDB-specific management operations, such as setting AWR data retention period, setting snapshot schedule, taking manual snapshots, and purging snapshot data for a CDB root.

A PDB-level snapshot contains the PDB statistics and also some global statistics that can be useful for diagnosing the performance problems related to the PDB. The PDB administrator can perform PDB-specific management operations, such as setting AWR data retention period, setting snapshot schedule, taking manual snapshots, and purging snapshot data for a PDB.

The CDB-level and PDB-level snapshot operations, such as creating snapshots and purging snapshots, can be performed in either the *automatic mode* or the *manual mode*.

The automatic snapshot operations are scheduled, so that they get executed automatically at a particular time. The `AWR_PDB_AUTOFLUSH_ENABLED` initialization parameter enables you to specify whether to enable or disable automatic snapshots for all the PDBs in a CDB or for individual PDBs in a CDB. The automatic snapshot operations are enabled by default for a CDB and (in Oracle Database 23ai) for a PDB. To enable automatic snapshots for a PDB, the PDB administrator must connect to that PDB, set the value for the `AWR_PDB_AUTOFLUSH_ENABLED` parameter to `true`, and set the snapshot generation interval to a value greater than 0.

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

The manual snapshot operations are explicitly initiated by users. The automatic snapshots and manual snapshots capture the same AWR information. Oracle recommends to generally use manual snapshots for a PDB. You should enable automatic snapshots only selectively for a PDB for performance reasons.

The primary interface for managing snapshots is Oracle Enterprise Manager Cloud Control (Cloud Control). If Cloud Control is not available, then you can use the procedures in the `DBMS_WORKLOAD_REPOSITORY` package to manage snapshots. The Oracle DBA role is required to use the procedures in the `DBMS_WORKLOAD_REPOSITORY` package. The SQL procedures to create, drop, and modify snapshots for a CDB root and a PDB are the same as that for a non-CDB. These SQL procedures perform their operations on the local database by default, if the target database information is not provided in their procedure call.

Note:

- The PDB-level snapshots have unique snapshot IDs and are not related to the CDB-level snapshots.
- The plugging and unplugging operations of a PDB in a CDB do not affect the AWR data stored on a PDB.
- The CDB administrator can use the *PDB lockdown profiles* to disable the AWR functionality for a PDB by executing the following SQL statement on that PDB:

  ```
  SQL> alter lockdown profile profile_name disable
  feature=('AWR_ACCESS');
  ```

  Once the AWR functionality is disabled on a PDB, snapshot operations cannot be performed on that PDB.

  The AWR functionality can be enabled again for a PDB by executing the following SQL statement on that PDB:

  ```
  SQL> alter lockdown profile profile_name enable
  feature=('AWR_ACCESS');
  ```

- Snapshot data is stored in the `SYSAUX` tablespace of a CDB and a PDB by default. Starting with Oracle Database 19c, you can specify any other tablespace to store snapshot data for a CDB and a PDB by modifying snapshot settings.

Note:

A multitenant container database is the only supported architecture in Oracle Database 21c and later releases. While the documentation is being revised, legacy terminology may persist. In most cases, "database" and "non-CDB" refer to a CDB or PDB, depending on context. In some contexts, such as upgrades, "non-CDB" refers to a non-CDB from a previous release.

See Also:

- "Creating Snapshots"
- "Dropping Snapshots"
- "Modifying Snapshot Settings"
- *Oracle Database Security Guide* for more information about the PDB lockdown profiles

Exporting and Importing AWR Data

The process of exporting and importing AWR data for a CDB root and a PDB in a multitenant environment is similar to the process of exporting and importing AWR data for a non-CDB.

> **See Also:**
>
> - "Exporting AWR Data" for information about exporting AWR data from an Oracle database
> - "Importing AWR Data" for information about importing AWR data into an Oracle database

Viewing AWR Data in a Multitenant Environment

You can view the AWR data in a multitenant environment using various Oracle Database reports and views.

AWR Reports

The primary interface for generating AWR reports is Oracle Enterprise Manager Cloud Control (Cloud Control). Whenever possible, generate AWR reports using Cloud Control.

> **See Also:**
>
> *Oracle Database 2 Day + Performance Tuning Guide* for more information about generating AWR report using Cloud Control

If Cloud Control is unavailable, then you can generate the AWR reports by running SQL scripts as described below. The DBA role is required to run these scripts.

- You can generate a CDB-specific AWR report from a CDB root that shows the *global system data* statistics for the whole multitenant environment. You can generate this AWR report using the SQL scripts described in the section "Generating an AWR Report for the Local Database".
- You can generate a PDB-specific AWR report from a PDB that shows the statistics related to that PDB. You can generate this AWR report using the SQL scripts described in the section "Generating an AWR Report for the Local Database".
- You can generate a PDB-specific AWR report from a CDB root that shows the statistics related to a specific PDB. You can generate this AWR report using the SQL scripts described in the section "Generating an AWR Report for a Specific Database".

AWR Views

The following table lists the Oracle Database views for accessing the AWR data stored on the CDB root and the individual PDBs in a multitenant environment.

> **See Also:**
>
> "Using Automatic Workload Repository Views" for more information about these AWR views

Table 6-2 Views for Accessing AWR Data in a Multitenant Environment

Views	Description
DBA_HIST Views	• The DBA_HIST views show the AWR data present only on the CDB root. • When the DBA_HIST views are accessed from a CDB root, they show all the AWR data stored on the CDB root. • When the DBA_HIST views are accessed from a PDB, they show the subset of the CDB root AWR data, which is specific to that PDB.
DBA_HIST_CON Views	• The DBA_HIST_CON views are similar to the DBA_HIST views, but they provide more fine grained information about each container, and thus, they have more data than the DBA_HIST views. • The DBA_HIST_CON views show the AWR data present only on the CDB root. • When the DBA_HIST_CON views are accessed from a CDB root, they show all the AWR data stored on the CDB root. • When the DBA_HIST_CON views are accessed from a PDB, they show the subset of the CDB root AWR data, which is specific to that PDB.
AWR_ROOT Views	• The AWR_ROOT views are available starting with Oracle Database 12*c* Release 2 (12.2) and are available only in the Multitenant environment. • The AWR_ROOT views are equivalent to the DBA_HIST views. • The AWR_ROOT views show the AWR data present only on the CDB root. • When the AWR_ROOT views are accessed from a CDB root, they show all the AWR data stored on the CDB root. • When the AWR_ROOT views are accessed from a PDB, they show the subset of the CDB root AWR data, which is specific to that PDB.
AWR_PDB Views	• The AWR_PDB views are available starting with Oracle Database 12*c* Release 2 (12.2). • The AWR_PDB views show the local AWR data present on a CDB root or a PDB. • When the AWR_PDB views are accessed from a CDB root, they show the AWR data stored on the CDB root. • When the AWR_PDB views are accessed from a PDB, they show the AWR data stored on that PDB.
CDB_HIST Views	• The CDB_HIST views show the AWR data stored on the PDBs. • When the CDB_HIST views are accessed from a CDB root, they show the union of the AWR data stored on all the PDBs. • When the CDB_HIST views are accessed from a PDB, they show the AWR data stored on that PDB.

Managing Automatic Workload Repository in Active Data Guard Standby Databases

Automatic Workload Repository (AWR) data can be captured for Active Data Guard (ADG) standby databases. This feature enables analyzing any performance-related issues for ADG standby databases.

AWR snapshots for ADG standby databases are called *remote snapshots*. A database node, called *destination*, is responsible for storing snapshots that are collected from remote ADG standby database nodes, called *sources*.

A destination can be either an ADG primary database or a non-ADG database. If a destination is an ADG primary database, then it is also a source database, and its snapshots are *local snapshots*.

A source is identified by a unique name or *source name* by which it is known to a destination.

You can assign a name to a destination node or a source node during its configuration. Otherwise, the value of the initialization parameter `DB_UNIQUE_NAME` is assigned as a name for a node.

Each source must have two database links, a destination-to-source database link and a source-to-destination database link. These database links are configured for each source during the ADG deployment. You must manually reconfigure these database links after certain ADG events, such as failovers, switchovers, and addition and removal of hosts, so that the database applications continue functioning properly after these events.

You can take the remote snapshots either automatically at scheduled time intervals or manually. The remote snapshots are always started by the destination node. After the destination initiates the snapshot creation process, sources *push* their snapshot data to the destination using database links. The snapshot data or *AWR data* stored on the destination can be accessed using AWR reports, Oracle Database import and export functions, and user-defined queries. The Automatic Database Diagnostic Monitor (ADDM) application can use the AWR data for analyzing any database performance-related issues.

AWR snapshot at an ADG Standby is enabled by default (Oracle Database 23ai). There is no need to run `dbms_workload_repository.enable_snapshot_service()` to enable it. It enables automatic snapshots for CDB root and all PDBs at ADG.

The snapshot retention is 8 days by default. To change the retention time, use `dbms_workload_repository.modify_snapshot_settings(retention)`.

After a switch-over, snapshots work correctly for both Primary and Standby.

Destination Database Responsibilities

A destination database manages the following tasks:

- Registering sources
- Assigning unique identifier for each source
- Creating database links between destination and sources
- Scheduling and initiating automatic snapshots for sources
- Managing destination workload by coordinating snapshots among sources
- Managing snapshot settings for each source
- Assigning identifiers to newly generated snapshots
- Partitioning the AWR tables
- Storing the performance data in the local AWR
- Purging the AWR data of destination and sources

Source Database Responsibilities

A source database manages the following tasks:

- Storing its performance data in the local AWR
- Sending its AWR data to the destination
- Responding to service requests from the destination
- Extracting the AWR data from the destination

Major Steps for Managing AWR in ADG Standby Databases

The following are the major steps for managing AWR in ADG standby databases:

1. Configuring the Remote Management Framework (RMF)
2. Managing Snapshots for Active Data Guard Standby Databases
3. Viewing AWR Data in Active Data Guard Standby Databases

> **Note:**
>
> Before you start configuring AWR for ADG environment, make sure that the database links for all the ADG standby databases are already configured during the ADG deployment.

Configuring the Remote Management Framework (RMF)

The Remote Management Framework (RMF) is an architecture for capturing performance statistics (AWR data) in an Oracle database.

> **Note:**
>
> RMF can be used only for ADG standby databases and standalone databases.

The RMF *topology* is a centralized architecture that consists of all the participating database nodes along with their metadata and connection information. The RMF topology has one database node, called *destination*, which is responsible for storing and managing performance data (AWR data) that is collected from the database nodes, called *sources*. A *candidate destination* is a source that can be configured in such way that it can replace the original destination, when the original destination is unavailable or is downgraded. A topology can have only one destination, and one or more candidate destinations.

Each database node in a topology must be assigned a unique name. This can be done using the procedure `DBMS_UMF.configure_node()` during configuring a node. If the name for a node is not provided in this procedure, then the value of the initialization parameter `DB_UNIQUE_NAME` is used as the name for a node.

The database nodes in a topology communicate with each other using database links. The database links between destination to source and source to destination must be created for each ADG standby database during the ADG deployment.

A *service* is an application running on a topology. For example, an AWR service running on a topology enables remote AWR snapshots for all the database nodes in that topology.

The RMF APIs are the PL/SQL procedures and functions that can be used to configure the RMF topology. The RMF APIs are declared in the PL/SQL package `DBMS_UMF`.

> **Note:**
>
> - The `SYS$UMF` user is the default database user that has all the privileges to access the system-level RMF views and tables. All the AWR related operations in RMF can be performed only by the `SYS$UMF` user. The `SYS$UMF` user is locked by default and it must be unlocked before deploying the RMF topology.
> - You need to provide password for the `SYS$UMF` user when creating database links in the RMF topology. If the password for the `SYS$UMF` user is changed, all the database links in the RMF topology must be recreated.

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

Setting Up the RMF Topology

You need to set up the RMF topology for collecting performance statistics for an Oracle database.

The following are the prerequisites for setting up the RMF topology:

- You must create destination to source and source to destination database links for all the database nodes to be registered in the RMF topology. This setup should be done during the ADG deployment.

The following are the steps for setting up the RMF topology:

1. Configure database nodes to add to the topology.
2. Create the topology.
3. Register database nodes with the topology.
4. (Optional) Register database links between the nodes in the topology. This configuration is required when a destination becomes unavailable and a candidate destination needs to connect to the remaining nodes in the topology using database links.

Example for Setting Up the RMF Topology

In this example, the three database nodes `T`, `S0`, and `S1` are added to the topology `Topology_1`. Node `T` is the destination node and nodes `S0` and `S1` are the source nodes. Node `S1` is a candidate destination, that is, when the original destination `T` is not available, node `S1` becomes the new destination. The AWR service is enabled for all the sources in the topology.

Assume that the following database links are already created during the ADG deployment:

- `DBLINK_T_to_S0`: Database link from `T` to `S0`.
- `DBLINK_T_to_S1`: Database link from `T` to `S1`.
- `DBLINK_S0_to_T`: Database link from `S0` to `T`.
- `DBLINK_S0_to_S1`: Database link from `S0` to `S1`.
- `DBLINK_S1_to_T`: Database link from `S1` to `T`.

- `DBLINK_S1_to_S0`: Database link from `S1` to `S0`.

The following is a sample code for setting up the RMF topology:

```
/* Configure the nodes T, S0, and S1 by executing these procedures */

/* Execute this procedure on node T */
SQL> exec DBMS_UMF.configure_node ('T');

/* Execute this procedure on node S0 */
SQL> exec DBMS_UMF.configure_node ('S0', 'DBLINK_S0_to_T');

/* Execute this procedure on node S1 */
SQL> exec DBMS_UMF.configure_node ('S1', 'DBLINK_S1_to_T');

/* Execute all the following procedures on the destination node T */

/* Create the topology 'Topology_1' */
SQL> exec DBMS_UMF.create_topology ('Topology_1');

/* Register the node S0 with the topology 'Topology_1' */
SQL> exec DBMS_UMF.register_node ('Topology_1',
                                  'S0',
                                  'DBLINK_T_to_S0',
                                  'DBLINK_S0_to_T',
                                  'TRUE'  /* Set it as a source */,
                                  'FALSE' /* Set it as not a candidate destination */);

/* Register the node S1 with the topology 'Topology_1' */
SQL> exec DBMS_UMF.register_node ('Topology_1',
                                  'S1',
                                  'DBLINK_T_to_S1',
                                  'DBLINK_S1_to_T',
                                  'TRUE'  /* Set it as a source */,
                                  'TRUE'  /* Set it as a candidate destination */);

/* Register the database links between the nodes S0 and S1 in the topology 'Topology_1'.
 * When destination T is unavailable at the time of failover, the source S0 can connect
 * to the candidate destination S1 using this database link.
 */
SQL> exec DBMS_UMF.create_link ('Topology_1',
                                'S0',
                                'S1',
                                'DBLINK_S0_to_S1',
                                'DBLINK_S1_to_S0');

/* Enable the AWR service on the node S0 in the topology 'Topology_1' */
SQL> exec DBMS_WORKLOAD_REPOSITORY.register_remote_database(node_name=>'S0');

/* Enable the AWR service on the node S1 in the topology 'Topology_1' */
SQL> exec DBMS_WORKLOAD_REPOSITORY.register_remote_database(node_name=>'S1');
```

> **Note:**
>
> The AWR service can be disabled for a node using the procedure:
>
> ```
> SQL> exec
> DBMS_WORKLOAD_REPOSITORY.unregister_remote_database(node_name)
> ```

Managing ADG Role Transition

An ADG role transition occurs when the ADG Primary or original destination fails (*failover* event) or when an ADG standby database or candidate destination takes over the role of the ADG Primary during the maintenance phase (*switchover* event).

Oracle recommends that you perform the following configuration steps before making the role change, that is, before making the candidate destination as the new destination due to the failover or switchover event:

1. Create database links between the sources and the candidate destination. This configuration must be done for all the sources by executing the following procedure on each source:

   ```
   SQL> EXEC DBMS_UMF.CREATE_LINK (topology name,
                                   source name,
                                   candidate destination name,
                                   source to candidate destination database link,
                                   candidate destination to source database link);
   ```

   > **Note:**
   >
   > Oracle recommends that you create database links among all the nodes in a topology to avoid any unanticipated issues that may arise at the time of role change.

2. Take an AWR snapshot on the candidate destination.

   > **Note:**
   >
   > To generate an AWR report for the candidate destination after the role change, take at least one snapshot for the candidate destination before the role change.

3. Restart the candidate destination as well as all the sources.

After completing the preceding configuration steps, you can make the role change by executing the following procedure on the candidate destination:

```
SQL> EXEC DBMS_UMF.SWITCH_DESTINATION(topology name, force_switch=>FALSE);
```

> **Note:**
>
> Oracle recommends that you do not take any snapshots for the sources during the role transition period. After the role change process is complete by executing the `DBMS_UMF.SWITCH_DESTINATION` procedure, you can take snapshots for the sources. If you want to generate AWR reports for the sources after the role change, then you must choose only those snapshots that were taken after the role change.

Getting the Details of Registered RMF Topologies

The RMF views described below show the configuration information about all the registered RMF topologies in a multi-database environment.

Table 6-3 RMF Views

RMF View	Description
`DBA_UMF_TOPOLOGY`	Shows all the registered topologies in a multi-database environment. Each topology has a topology name, a destination ID, and topology state. To enable RMF, the topology state of at least one topology should be `ACTIVE`.
`DBA_UMF_REGISTRATION`	Shows all the registered nodes in all the topologies in a multi-database environment.
`DBA_UMF_LINK`	Shows all the registered database links in all the topologies in a multi-database environment.
`DBA_UMF_SERVICE`	Shows all the registered services in all the topologies in a multi-database environment.

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

Managing Snapshots for Active Data Guard Standby Databases

The AWR snapshots for ADG standby databases are called *remote* snapshots. Similar to local AWR snapshots, remote AWR snapshots can be generated automatically at scheduled time intervals or can be generated manually. The *Push-on-Demand* mechanism is used for generating remote snapshots, where the snapshots generation process is initiated by the destination, which then instructs the sources to start *pushing* the snapshot data to the destination over database links. The destination periodically initiates automatic snapshots based on the snapshot time interval configured for each of the sources.

> **Note:**
>
> The destination is responsible for purging the expired remote snapshots based on the snapshot data or *AWR data* retention settings for individual sources. Purging of locally generated snapshots occurs as part of the regularly scheduled purging process. By default, Oracle Database automatically purges snapshots that have been stored in AWR for over 8 days. The partitioning of AWR table for remote snapshots is done in the same way as that of the local snapshots.

Creating, Modifying, and Deleting Remote Snapshots

The APIs for creating, modifying, and deleting *remote* snapshots are same as that for the *local* snapshots.

> **Note:**
>
> For creating *remote* snapshots, you can also use the `DBMS_WORKLOAD_REPOSITORY.CREATE_REMOTE_SNAPSHOT` API. This API works similar to the *local* snapshot creation API `DBMS_WORKLOAD_REPOSITORY.CREATE_SNAPSHOT`, but it takes the additional parameter of RMF topology name.

> **See Also:**
>
> - "Creating Snapshots"
> - "Modifying Snapshot Settings"
> - "Dropping Snapshots"
> - *Oracle Database PL/SQL Packages and Types Reference* for the syntax of the `DBMS_WORKLOAD_REPOSITORY.CREATE_REMOTE_SNAPSHOT` API.

Managing Baselines for Remote Snapshots

The APIs for managing baselines for *remote* snapshots are same as that for the *local* snapshots.

> **See Also:**
>
> "Managing Baselines"

Exporting and Importing Remote Snapshots

> **Note:**
>
> You cannot execute the AWR export and import scripts related to remote snapshots on an ADG standby database, that is, on a source database. Always execute these scripts on a destination database.

The process of exporting and importing AWR data for *remote* snapshots is same as that for the *local* snapshots. Starting with Oracle Database 12*c* Release 2 (12.2), the AWR data export and import scripts `awrextr.sql` and `awrload.sql` use the *source name* identifier to distinguish snapshots originating from a particular source. A source name is stored in a dump file during an export operation and is used as a default source name during an import operation.

> **See Also:**
>
> "Transporting Automatic Workload Repository Data to Another System" for information about exporting and importing AWR data for local snapshots.

Exporting Remote Snapshots Using the awrextr.sql Script

The process of exporting *remote* snapshots is similar to exporting *local* snapshots using the `awrextr.sql` script described in the section "Exporting AWR Data" with the following differences:

- The default export log file directory is the same as that of the dump file, but you can also specify any other directory for an export log file.
- The `.dmp` suffix can be specified to the name of the dump file to export.
- The export script displays the values of `SOURCE_DBID` and `SOURCE_NAME` columns of AWR tables before prompting for the *Mapped Database ID* value to export.

Importing Remote Snapshots Using the awrload.sql Script

The process of importing *remote* snapshots is similar to importing *local* snapshots using the `awrload.sql` script described in the section "Importing AWR Data" with the following differences:

- The default import log file directory is the same as that of the dump file, but you can also specify any other directory for an import log file. This is particularly useful when the dump file resides in a read-only directory.
- The `.dmp` suffix can be specified to the name of the dump file to import.
- The import script uses the values of `SOURCE_DBID` and `SOURCE_NAME` columns present in the dump file to determine the appropriate *Mapped Database ID* to use for storing the snapshot data in AWR.

> **Note:**
>
> The snapshot import operation is not affected by the version of the Oracle database from which the snapshot dump was generated.

Viewing AWR Data in Active Data Guard Standby Databases

You can view the AWR data stored in the ADG standby databases using Oracle supplied AWR views and AWR reports.

Viewing AWR Data Using AWR Views

You can view the historical data stored in AWR using the *DBA_HIST* views described in the section "Using Automatic Workload Repository Views".

> **Note:**
>
> Starting with Oracle Database 12*c* Release 2 (12.2), the view `DBA_HIST_DATABASE_INSTANCE` contains the column `DB_UNIQUE_NAME` to support AWR for ADG standby databases. The column `DB_UNIQUE_NAME` stores the unique identifier of a source by which it is known to the destination.

Viewing AWR Data Using AWR Reports

You can view the performance statistics related to ADG standby databases using AWR reports. The primary interface for generating AWR reports is Oracle Enterprise Manager Cloud Control (Cloud Control). Whenever possible, generate AWR reports using Cloud Control. If Cloud Control is unavailable, then generate AWR reports using the Oracle supplied SQL scripts. The DBA role is required to run these scripts.

The AWR data can be queried for a particular source using the source name-mapped database ID pair. The mapped database ID is similar to the database identifier (DBID) that is used by AWR to identify a database instance and is stored in the `DBID` column in the AWR tables. The AWR DBID value is derived as follows for the ADG standby databases:

- For a destination, the AWR DBID value is the value of `V$DATABASE.CON_DBID`.
- For a source, the AWR DBID value is the value of `DBMS_UMF.GET_NODE_ID_LOCAL()` or the value of the column `NODE_ID` in the `DBA_UMF_REGISTRATION` view.

As snapshot IDs are not unique across sources, the pair of snapshot ID-mapped database ID identifies a snapshot for a particular source.

> **See Also:**
>
> "Generating an AWR Report for a Specific Database" for information about generating AWR reports using Oracle supplied SQL scripts.

Generating Automatic Workload Repository Reports

An AWR report shows data captured between two snapshots (or two points in time). AWR reports are divided into multiple sections. The content of the report contains the workload profile of the system for the selected range of snapshots. The HTML report includes links that can be used to navigate quickly between sections.

> **Note:**
>
> If you run a report on a database that does not have any workload activity during the specified range of snapshots, then calculated percentages for some report statistics can be less than 0 or greater than 100. This result means that there is no meaningful value for the statistic.

This section describes how to generate AWR reports and contains the following topics:

- User Interface for Generating an AWR Report
- Generating an AWR Report Using the Command-Line Interface

User Interface for Generating an AWR Report

The primary interface for generating AWR reports is Oracle Enterprise Manager Cloud Control (Cloud Control). Whenever possible, generate AWR reports using Cloud Control.

If Cloud Control is unavailable, then generate AWR reports by running SQL scripts. The DBA role is required to run these scripts.

> **See Also:**
>
> *Oracle Database 2 Day + Performance Tuning Guide* for more information about generating AWR reports using Cloud Control

Generating an AWR Report Using the Command-Line Interface

This section describes how to generate AWR reports by running SQL scripts in the command-line interface. The DBA role is required to run these scripts. Click on an appropriate task link in the following table for the detailed steps to generate the required AWR report.

Table 6-4 SQL Scripts for Generating AWR Reports

Task	SQL Script	Description
Generating an AWR Report for the Local Database	`awrrpt.sql`	Generates an AWR report in HTML or text format that displays statistics from a range of snapshot IDs in the local database instance.
Generating an AWR Report for a Specific Database	`awrrpti.sql`	Generates an AWR report in HTML or text format that displays statistics from a range of snapshot IDs in a specific database instance.

Table 6-4 (Cont.) SQL Scripts for Generating AWR Reports

Task	SQL Script	Description
Generating an AWR Report for the Local Database in Oracle RAC	`awrgrpt.sql`	Generates an AWR report in HTML or text format that displays statistics from a range of snapshot IDs in the local database instance in an Oracle RAC environment.
Generating an AWR Report for a Specific Database in Oracle RAC	`awrgrpti.sql`	Generates an AWR report in HTML or text format that displays statistics from a range of snapshot IDs in a specific database instance in an Oracle RAC environment.
Generating an AWR Report for a SQL Statement on the Local Database	`awrsqrpt.sql`	Generates an AWR report in HTML or text format that displays statistics for a particular SQL statement from a range of snapshot IDs in the local database instance.
Generating an AWR Report for a SQL Statement on a Specific Database	`awrsqrpi.sql`	Generates an AWR report in HTML or text format that displays statistics for a particular SQL statement from a range of snapshot IDs in a specific database instance.

Generating an AWR Report for the Local Database

The `awrrpt.sql` SQL script generates an HTML or text report that displays statistics from a range of snapshot IDs.

To generate an AWR report on the local database instance using the command-line interface:

1. At the SQL prompt, enter:

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

2. Specify whether you want an HTML or a text report:

   ```
   Enter value for report_type: text
   ```

   In this example, a text report is chosen.

3. Specify the number of days for which you want to list snapshot IDs.

   ```
   Enter value for num_days: 2
   ```

   A list of existing snapshots for the specified time range is displayed. In this example, snapshots captured in the last 2 days are displayed.

4. Specify a beginning and ending snapshot ID for the workload repository report:

   ```
   Enter value for begin_snap: 150
   Enter value for end_snap: 160
   ```

   In this example, the snapshot with a snapshot ID of 150 is selected as the beginning snapshot, and the snapshot with a snapshot ID of 160 is selected as the ending snapshot.

5. Enter a report name, or accept the default report name:

   ```
   Enter value for report_name:
   Using the report name awrrpt_1_150_160
   ```

In this example, the default name is accepted and an AWR report named `awrrpt_1_150_160` is generated.

Generating an AWR Report for a Specific Database

The `awrrpti.sql` SQL script generates an HTML or text report that displays statistics from a range of snapshot IDs using a specific database instance. This script enables you to specify a database identifier and instance for which the AWR report will be generated.

To generate an AWR report on a specific database instance using the command-line interface:

1. At the SQL prompt, enter:

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

2. Specify whether you want an HTML or a text report:

   ```
   Enter value for report_type: text
   ```

   In this example, a text report is chosen.

   A list of available database identifiers and instance numbers are displayed:

   ```
   Instances in this Workload Repository schema
   ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
      DB Id    Inst Num DB Name      Instance     Host
   ----------- -------- ------------ ------------ ------------
    3309173529        1 MAIN         main         examp1690
    3309173529        1 TINT251      tint251      samp251
   ```

3. Enter the values for the database identifier (`dbid`) and instance number (`inst_num`):

   ```
   Enter value for dbid: 3309173529
   Using 3309173529 for database Id
   Enter value for inst_num: 1
   ```

   > **Note:**
   >
   > For an ADG standby database, the value for `dbid` can be determined as follows:
   >
   > - For a Destination node, use the value of `v$database.con_dbid` .
   > - For a Source node, use the value of `dbms_umf.get_node_id_local()`.

4. Specify the number of days for which you want to list snapshot IDs.

   ```
   Enter value for num_days: 2
   ```

   A list of existing snapshots for the specified time range is displayed. In this example, snapshots captured in the last 2 days are displayed.

5. Specify a beginning and ending snapshot ID for the workload repository report:

   ```
   Enter value for begin_snap: 150
   Enter value for end_snap: 160
   ```

   In this example, the snapshot with a snapshot ID of 150 is selected as the beginning snapshot, and the snapshot with a snapshot ID of 160 is selected as the ending snapshot.

6. Enter a report name, or accept the default report name:

```
Enter value for report_name:
Using the report name awrrpt_1_150_160
```

In this example, the default name is accepted and an AWR report named `awrrpt_1_150_160` is generated on the database instance with a database ID value of `3309173529`.

Generating an AWR Report for the Local Database in Oracle RAC

The `awrgrpt.sql` SQL script generates an HTML or text report that displays statistics from a range of snapshot IDs using the current database instance in an Oracle Real Application Clusters (Oracle RAC) environment.

> **Note:**
>
> In an Oracle RAC environment, Oracle recommends generating an HTML report (instead of a text report) because it is much easier to read.

To generate an AWR report for Oracle RAC on the local database instance using the command-line interface:

1. At the SQL prompt, enter:

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

2. Specify whether you want an HTML or a text report:

   ```
   Enter value for report_type: html
   ```

   In this example, an HTML report is chosen.

3. Specify the number of days for which you want to list snapshot IDs.

   ```
   Enter value for num_days: 2
   ```

   A list of existing snapshots for the specified time range is displayed. In this example, snapshots captured in the last day are displayed.

4. Specify a beginning and ending snapshot ID for the workload repository report:

   ```
   Enter value for begin_snap: 150
   Enter value for end_snap: 160
   ```

   In this example, the snapshot with a snapshot ID of 150 is selected as the beginning snapshot, and the snapshot with a snapshot ID of 160 is selected as the ending snapshot.

5. Enter a report name, or accept the default report name:

   ```
   Enter value for report_name:
   Using the report name awrrpt_rac_150_160.html
   ```

   In this example, the default name is accepted and an AWR report named `awrrpt_rac_150_160.html` is generated.

Generating an AWR Report for a Specific Database in Oracle RAC

The `awrgrpti.sql` SQL script generates an HTML or text report that displays statistics from a range of snapshot IDs using specific databases instances running in an Oracle RAC

environment. This script enables you to specify database identifiers and a comma-delimited list of database instances for which the AWR report will be generated.

> **Note:**
>
> In an Oracle RAC environment, Oracle recommends generating an HTML report (instead of a text report) because it is much easier to read.

To generate an AWR report for Oracle RAC on a specific database instance using the command-line interface:

1. At the SQL prompt, enter:

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

2. Specify whether you want an HTML or a text report:

   ```
   Enter value for report_type: html
   ```

   In this example, an HTML report is chosen.

   A list of available database identifiers and instance numbers are displayed:

   ```
   Instances in this Workload Repository schema
   ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
      DB Id    Inst Num DB Name      Instance     Host
   ----------- -------- ------------ ------------ ------------
    3309173529        1 MAIN         main         examp1690
    3309173529        1 TINT251      tint251      samp251
    3309173529        2 TINT251      tint252      samp252
   ```

3. Enter the value for the database identifier (`dbid`):

   ```
   Enter value for dbid: 3309173529
   Using 3309173529 for database Id
   ```

4. Enter the value for the instance numbers (`instance_numbers_or_all`) of the Oracle RAC instances you want to include in the report:

   ```
   Enter value for instance_numbers_or_all: 1,2
   ```

5. Specify the number of days for which you want to list snapshot IDs.

   ```
   Enter value for num_days: 2
   ```

   A list of existing snapshots for the specified time range is displayed. In this example, snapshots captured in the last 2 days are displayed.

6. Specify a beginning and ending snapshot ID for the workload repository report:

   ```
   Enter value for begin_snap: 150
   Enter value for end_snap: 160
   ```

   In this example, the snapshot with a snapshot ID of 150 is selected as the beginning snapshot, and the snapshot with a snapshot ID of 160 is selected as the ending snapshot.

7. Enter a report name, or accept the default report name:

   ```
   Enter value for report_name:
   Using the report name awrrpt_rac_150_160.html
   ```

In this example, the default name is accepted and an AWR report named `awrrpt_rac_150_160.html` is generated on the database instance with a database ID value of `3309173529`.

Generating an AWR Report for a SQL Statement on the Local Database

The `awrsqrpt.sql` SQL script generates an HTML or text report that displays statistics of a particular SQL statement from a range of snapshot IDs. Run this report to inspect or debug the performance of a SQL statement.

To generate an AWR report for a SQL statement on the local database instance using the command-line interface:

1. At the SQL prompt, enter:

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

2. Specify whether you want an HTML or a text report:

   ```
   Enter value for report_type: html
   ```

   In this example, an HTML report is chosen.

3. Specify the number of days for which you want to list snapshot IDs.

   ```
   Enter value for num_days: 1
   ```

   A list of existing snapshots for the specified time range is displayed. In this example, snapshots captured in the previous day are displayed.

4. Specify a beginning and ending snapshot ID for the workload repository report:

   ```
   Enter value for begin_snap: 146
   Enter value for end_snap: 147
   ```

   In this example, the snapshot with a snapshot ID of 146 is selected as the beginning snapshot, and the snapshot with a snapshot ID of 147 is selected as the ending snapshot.

5. Specify the SQL ID of a particular SQL statement to display statistics:

   ```
   Enter value for sql_id: 2b064ybzkwf1y
   ```

   In this example, the SQL statement with a SQL ID of `2b064ybzkwf1y` is selected.

6. Enter a report name, or accept the default report name:

   ```
   Enter value for report_name:
   Using the report name awrrpt_1_146_147.html
   ```

   In this example, the default name is accepted and an AWR report named `awrrpt_1_146_147` is generated.

Generating an AWR Report for a SQL Statement on a Specific Database

The `awrsqrpi.sql` SQL script generates an HTML or text report that displays statistics of a particular SQL statement from a range of snapshot IDs using a specific database instance.This script enables you to specify a database identifier and instance for which the AWR report will be generated. Run this report to inspect or debug the performance of a SQL statement on a specific database and instance.

To generate an AWR report for a SQL statement on a specific database instance using the command-line interface:

1. At the SQL prompt, enter:

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

2. Specify whether you want an HTML or a text report:

```
Enter value for report_type: html
```

In this example, an HTML report is chosen.

A list of available database identifiers and instance numbers are displayed:

```
Instances in this Workload Repository schema
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
   DB Id    Inst Num DB Name      Instance     Host
------------ -------- ------------ ------------ ------------
 3309173529        1 MAIN         main         examp1690
 3309173529        1 TINT251      tint251      samp251
```

3. Enter the values for the database identifier (`dbid`) and instance number (`inst_num`):

```
Enter value for dbid: 3309173529
Using 3309173529 for database Id
Enter value for inst_num: 1
Using 1 for instance number
```

4. Specify the number of days for which you want to list snapshot IDs.

```
Enter value for num_days: 1
```

A list of existing snapshots for the specified time range is displayed. In this example, snapshots captured in the previous day are displayed.

5. Specify a beginning and ending snapshot ID for the workload repository report:

```
Enter value for begin_snap: 146
Enter value for end_snap: 147
```

In this example, the snapshot with a snapshot ID of 146 is selected as the beginning snapshot, and the snapshot with a snapshot ID of 147 is selected as the ending snapshot.

6. Specify the SQL ID of a particular SQL statement to display statistics:

```
Enter value for sql_id: 2b064ybzkwf1y
```

In this example, the SQL statement with a SQL ID of `2b064ybzkwf1y` is selected.

7. Enter a report name, or accept the default report name:

```
Enter value for report_name:
Using the report name awrrpt_1_146_147.html
```

In this example, the default name is accepted and an AWR report named `awrrpt_1_146_147` is generated on the database instance with a database ID value of `3309173529`.

Generating Performance Hub Active Report

Performance Hub feature of EM provides an active report with a consolidated view of all performance data for a specified time period. The report is fully interactive; its contents are saved in a HTML file, which you can access offline using a web browser.

> **See Also:**
>
> *Oracle Database 2 Day DBA* for more information about Performance Hub feature of EM.

This section describes how to generate Performance Hub active report and contains the following topics:

- Overview of Performance Hub Active Report
- Command-Line User Interface for Generating a Performance Hub Active Report
- Generating a Performance Hub Active Report Using a SQL Script
- Performance Workload Analysis

Overview of Performance Hub Active Report

Performance Hub active report enables you to view all performance data available for a time period that you specify. Different tabs are available in the Performance Hub, depending on whether real-time or historical data is selected for the time period. When real-time data is selected, more granular data is presented, because real-time data for the last hour is displayed. When historical data is selected, more detailed data is presented, but the data points are averaged out to the Automatic Workload Repository (AWR) interval for the selected time period.

This section describes Performance Hub active report and contains the following topics:

- About Performance Hub Active Report Tabs
- About Performance Hub Active Report Types

About Performance Hub Active Report Tabs

Performance Hub active report contains interactive tabs that enable you to view and navigate through performance data categorized into various performance areas.

The tabs contained in a Performance Hub active report include the following:

- Summary

  The Summary tab provides an overview of system performance, including resource consumption, average active sessions, and load profile information. This tab is available for real-time data as well as historical data.

- Activity

  The Activity tab displays ASH analytics. This tab is available for real-time data as well as historical data.

- Workload

The Workload tab displays metric information about the workload profile, such as call rates, logon rate, and top SQL. This tab is available for real-time data as well as historical data.

- RAC

  The RAC tab displays metrics specific to Oracle RAC, such as the number of global cache blocks received and the average block latency. This tab is only available in Oracle RAC environments. This tab is available for real-time data as well as historical data.

- Monitored SQL

  The Monitored SQL tab displays information about monitored SQL statements. This tab is available for real-time data as well as historical data.

- ADDM

  The ADDM tab displays information for ADDM analysis tasks and Real-Time ADDM analysis reports. This tab is available for real-time data as well as historical data.

- Current ADDM Findings

  The Current ADDM Findings tab displays a real-time analysis of system performance for the past 5 minutes. This tab is only available if the specified time period for the Performance Hub active report is within the past hour. This tab is available only for real-time data.

- Database time

  The Database Time tab displays wait events by category for various metrics. This tab is available only for historical data.

- Resources

  The Resources tab displays operating system and I/O usage statistics. This tab is available only for historical data.

- System Statistics

  The System Statistics tab displays database and system statistics. This tab is available only for historical data.

About Performance Hub Active Report Types

You can choose the level of details displayed within each tab of the Performance Hub active report by selecting the report type.

The available report types for the Performance Hub active report include the following:

- Basic

  Only the basic information for all the tabs is saved to the report.

- Typical

  In addition to the information saved in the basic report type, the SQL Monitor information for the top SQL statements contained in the Monitored SQL tab and the ADDM reports are saved to the report.

- All

  In addition to the information saved in the typical report type, the SQL Monitor information for all SQL statements contained in the Monitored SQL tab and all detailed reports for all tabs are saved to the report.

Command-Line User Interface for Generating a Performance Hub Active Report

You can generate a Performance Hub active report using the command-line interface in one of two ways:

- Using a SQL script, as described in "Generating a Performance Hub Active Report Using a SQL Script".
- Using the `DBMS_PERF` package, as described in *Oracle Database PL/SQL Packages and Types Reference*.

Generating a Performance Hub Active Report Using a SQL Script

This section describes how to generate Performance Hub active report by running the `perfhubrpt.sql` SQL script in the command-line interface. The DBA role is required to run this script.

To generate a Performance Hub active report:

1. At the SQL prompt, enter:

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

2. Specify the desired report type:

   ```
   Please enter report type: typical
   ```

   For information about the available report types, see "About Performance Hub Active Report Types".

3. Enter the value for the database identifier of the database you want to use:

   ```
   Please enter database ID: 3309173529
   ```

   To use the local database, enter a null value (the default value). If you specify a database identifier for a database other than the local database, then the Performance Hub active report is generated from imported AWR data.

4. Enter the value for the instance number of the database instance you want to use:

   ```
   Please enter instance number: all instances
   ```

   To specify all instances, enter `all instances` (the default value).

5. Enter the desired time period by specifying an end time and a start time in the format of `dd:mm:yyyy hh:mi:ss`:

   ```
   Please enter end time in format of dd:mm:yyyy hh24:mi:ss: 03:04:2014 17:00:00
   Please enter start time in format of dd:mm:yyyy hh24:mi:ss: 03:04:2014 16:00:00
   ```

6. Enter a report name, or accept the default report name:

   ```
   Enter value for report_name: my_perfhub_report.html
   ```

   In this example, a Performance Hub active report named `my_perfhub_report` is generated on all database instances with a database ID value of `3309173529` for the specified time period from 4:00 p.m. to 5:00 p.m on April 3, 2014.

Performance Workload Analysis

Workload Analysis

The **Workload Analysis** is a tool to quickly identify issues with the workload by comparing a given workload with another reference, which could be fixed or rolling. The analysis itself can be a one time analysis, for instance comparing it two periods, or the analysis can be a re-occurring task that is scheduled with various flexible parameters. The time period for the analysis is selected from the **View Data** drop-down with the default ranges of **Last 24 Hours**, **Last 7 Days**, **Last 30 Days**, and **All**.

Scheduled Analysis

The **Scheduled Analysis** tab of the **Workload Analysis** page displays five boxes with counters to alert the number of differences to:

- **Changed Workloads**
- **Monitored Workloads**
- **Regressed Metrics**
- **Improved Metrics**
- **Unchanged Metrics**

The counter overview provides insight into how changes to workloads improved or regressed for the time period.

The **Create Analysis Task** for the **Scheduled Analysis** requires that you to provide additional information about the re-occurring task. Here are some settings, some of them optional, needed for establishing a re-occurring analysis in the **Workload Capture** form.

- **SQL Tuning Set Name Prefix**
- **Load SQL Statements using AWR Snapshot**,
  - Specify Custom Time Range with a Start Time and End Time
  - Quick Select from Snapshots created in the past with specified units of hours or days, and only up to 7 days

  Total number of SQL Statements Captured
  - Capture All
  - Capture Top N
- **Filter Option**
- **Workload Comparison**
  - **Subsequent Comparisons**: Compare using a rolling reference, or compare using a fixed reference
  - **Optional Initial reference workload**
  - **Comparison Metric** such as **CPU Time** , **Elapsed Time** , **Buffer Gets**, **Disk Reads**, **Direct Writes**, **Physical I/Os**, and **Optimizer Cost** .
  - **Change Threshold (%)**
  - **Consumer Resource Group**
  - **Days to retain purge results**

- **Schedule**
  - **Time zone**: Local (America/Denver), Database (GMT), or UTC

    Start Date: Immediate, or Later specified with date field

    End Date: None, or Specified Date

    Repeat Every: Select units first.
    * (1-24) Hours
    * (1-7) Days
    * (1-4) Weeks
    * (1-12) Months

    Repeat On:Specified Date, or Custom Time

When an analysis task is created, it appears under the **Scheduled Analysis**. The arrow next to the Analysis name in the table expands to a list of various instances when that scheduled analysis was run and links to the resulting Comparison Report (or **Workload Analysis Report**)

One-Time Analysis

The **One-Time Analysis** tab of the **Workload Analysis** page displays five boxes with counters to alert the number of differences to:

- **Analysis With Difference**
- **Total Analyses**
- **Regressed Metrics**
- **Improved Metrics**
- **Unchanged Metrics**

The counter overview provides insight into how changes to workloads improved or regressed for the time period.

Create Analysis Task is similar for **One-Time Analysis**. The task requires

- A unique name for the task
- Select the **Reference Workload** by name or searching for them.
- Select the **Compared Workload** by name or searching for them.
- The task needs one or more comparison metrics,such as **CPU Time** , **Elapsed Time** , **Buffer Gets**, **Disk Reads**, **Direct Writes**, **Physical I/Os**, and **Optimizer Cost** .

When an analysis task is created, it appears under the **One-Time Analysis**. The arrow next to the Analysis name in the table expands to a list of various instances when that scheduled analysis was run and links to the resulting Comparison Report (or **Workload Analysis Report**)

Workload Analysis Report

The **Workload Analysis Report** is available after an analysis task has been run, whether regularly or one-time. Each analysis has its own report.

The information in the report varies according to the criteria established from **Create Analysis Task** . The reports contain data about the reference and/or comparison followed by the summary using a comparison metric, such as **CPU Time** , **Elapsed Time** , **Buffer Gets**, **Disk Reads**, **Direct Writes**, **Physical I/Os**, **Optimizer Cost** .

The **Summary** area lists, for instance, the **Workload Impact** by **Overall**, **Common SQL**, **Improved SQL**, **Regressed SQL**, **Missing SQL**, and **New SQL**:

The **Breakdown** area provides graphs for various performance topics, such as:

- **CPU Time** is a graph with the Before and After to the workload.
- **SQL Statement by Performance** is a graph with Improved, Regressed, Unchanged, Missing, and New categories and a bar indicating their number.
- **SQL Statement by Plan Change** shows the New Plan and Same Plan impacts side by side.

Top SQL Statements By Workload Impacts area is a table for each impact by:

- **Impact**
- **SQL ID**
- **SQL Test**
- **Plan Change**
- **Net Absolute Impact on Workload (%)**
- **CPU Time Change (ms)**