7

Automatic Performance Diagnostics

This chapter describes Oracle Database automatic features for performance diagnosing and tuning.

This chapter contains the following topics:

> **See Also:**
>
> *Oracle Database 2 Day + Performance Tuning Guide* for information about using Oracle Enterprise Manager Cloud Control (Cloud Control) to diagnose and tune the database with the Automatic Database Diagnostic Monitor

Overview of the Automatic Database Diagnostic Monitor

The Automatic Workload Repository (AWR) stores performance related statics for an Oracle database. The Automatic Database Diagnostic Monitor (ADDM) is a diagnostic tool that analyzes the AWR data on a regular basis, locates root causes of any performance problems, provides recommendations for correcting the problems, and identifies non-problem areas of the system. Because AWR is a repository of historical performance data, ADDM can analyze performance issues after the event, often saving time and resources in reproducing a problem.

In most cases, ADDM output should be the first place that a DBA looks when notified of a performance problem. ADDM provides the following benefits:

- Automatic performance diagnostic report every hour by default
- Problem diagnosis based on decades of tuning expertise
- Time-based quantification of problem impacts and recommendation benefits
- Identification of root cause, not symptoms
- Recommendations for treating the root causes of problems
- Identification of non-problem areas of the system
- Minimal overhead to the system during the diagnostic process
- Accurate diagnosis of performance issues due to misconfiguration, incorrect usage, etc.
- Identification of objects that are candidates for in-memory placement as well as objects that are in-memory but not utilized for in-memory scans.

- Detection of hard disks in use instead of flash, low cell offloading, I/O workload throttling or imbalance across disks or cells in the system, which improve performance diagnosis capabilities particularly for Exadata.
- Diagnosability into failed tasks and associated error messages through DBA accessible views

Tuning is an iterative process, and fixing one problem can cause the bottleneck to shift to another part of the system. Even with the benefit of ADDM analysis, it can take multiple tuning cycles to reach acceptable system performance. ADDM benefits apply beyond production systems; on development and test systems, ADDM can provide an early warning of performance issues.

> **Note:**
>
> Data visibility and privilege requirements may differ when using ADDM features with pluggable databases (PDBs). For information about how manageability features, including ADDM features, work in a multitenant container database (CDB), see *Oracle Multitenant Administrator's Guide*.

ADDM Analysis

An ADDM analysis can be performed on a pair of AWR snapshots and a set of instances from the same database. The pair of AWR snapshots define the time period for analysis, and the set of instances define the target for analysis.

If you are using Oracle Real Application Clusters (Oracle RAC), then ADDM has three analysis modes:

- Database

  In Database mode, ADDM analyzes all instances of the database.
- Instance

  In Instance mode, ADDM analyzes a particular instance of the database.
- Partial

  In Partial mode, ADDM analyzes a subset of all database instances.

If you are not using Oracle RAC, then ADDM can only function in Instance mode because only one instance of the database exists.

An ADDM analysis is performed each time an AWR snapshot is taken and the results are saved in the database. The time period analyzed by ADDM is defined by the last two snapshots (the last hour by default). ADDM will always analyze the specified instance in Instance mode. For non-Oracle RAC or single instance environments, the analysis performed in the Instance mode is the same as a database-wide analysis. If you are using Oracle RAC, then ADDM also analyzes the entire database in Database mode, as described in "Using ADDM with Oracle Real Application Clusters".

After ADDM completes its analysis, you can view the ADDM results using either Cloud Control, or `DBMS_ADDM` package subprograms, or `DBA_ADDM_*` and `DBA_ADVISOR_*` views.

ADDM analysis is performed top down, first identifying symptoms, and then refining them to reach the root causes of performance problems. The goal of the analysis is to reduce a single throughput metric called `DB time`. `DB time` is the fundamental measure of database

performance, and is the cumulative time spent by the database in processing user requests. It includes wait time and CPU time of all non-idle user foreground sessions. `DB time` is displayed in the `V$SESS_TIME_MODEL` and `V$SYS_TIME_MODEL` views.

By reducing `DB time`, the database is able to support more user requests using the same resources, which increases throughput. The problems reported by ADDM are sorted by the amount of `DB time` they are responsible for. System areas that are not responsible for a significant portion of `DB time` are reported as non-problem areas.

The types of problems that ADDM considers include the following:

- CPU bottlenecks - Is the system CPU bound by Oracle Database or some other application?
- Undersized Memory Structures - Are the Oracle Database memory structures, such as the SGA, PGA, and buffer cache, adequately sized?
- I/O capacity issues - Is the I/O subsystem performing as expected?
- High-load SQL statements - Are there any SQL statements which are consuming excessive system resources?
- High-load PL/SQL execution and compilation, and high-load Java usage
- Oracle RAC specific issues - What are the global cache hot blocks and objects; are there any interconnect latency issues?
- Sub-optimal use of Oracle Database by the application - Are there problems with poor connection management, excessive parsing, or application level lock contention?
- Database configuration issues - Is there evidence of incorrect sizing of log files, archiving issues, excessive checkpoints, or sub-optimal parameter settings?
- Concurrency issues - Are there buffer busy problems?
- Hot objects and top SQL for various problem areas

> **Note:**
>
> This is not a comprehensive list of all problem types that ADDM considers in its analysis.

ADDM also documents the non-problem areas of the system. For example, wait event classes that are not significantly impacting the performance of the system are identified and removed from the tuning consideration at an early stage, saving time and effort that would be spent on items that do not impact overall system performance.

> **See Also:**
>
> - *Oracle Database Reference* for information about the `V$SESS_TIME_MODEL` and `V$SYS_TIME_MODEL` views
> - "Time Model Statistics" for a discussion of time model statistics and `DB time`
> - *Oracle Database Concepts* for information about server processes

Using ADDM with Oracle Real Application Clusters

If you are using Oracle RAC, then run ADDM in Database analysis mode to analyze the throughput performance of all instances of the database. In Database mode, ADDM considers DB time as the sum of the database time for all database instances. Using the Database analysis mode enables you to view all findings that are significant to the entire database in a single report, instead of reviewing a separate report for each instance.

The Database mode report includes findings about database resources (such as I/O and interconnect). The report also aggregates findings from the various instances if they are significant to the entire database. For example, if the CPU load on a single instance is high enough to affect the entire database, then the finding appears in the Database mode analysis, which points to the particular instance responsible for the problem.

> **See Also:**
>
> *Oracle Real Application Clusters Administration and Deployment Guide* for information about using ADDM with Oracle RAC

Using ADDM in a Multitenant Environment

Starting with Oracle Database 12c, ADDM is enabled by default in the root container of a multitenant container database (CDB). Starting with Oracle Database 19c, you can also use ADDM in a pluggable database (PDB).

> **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.

In a CDB, ADDM works in the same way as it works in a non-CDB, that is, the ADDM analysis is performed each time an AWR snapshot is taken on a CDB root or a PDB, and the ADDM results are stored on the same database system where the snapshot is taken. The time period analyzed by ADDM is defined by the last two snapshots (the last hour by default).

After ADDM completes its analysis, you can view the ADDM results using any of the following methods:

- Using Enterprise Manager Cloud Control (Cloud Control)
- Using the `DBA_ADDM_*` and `DBA_ADVISOR_*` views

Note:

- ADDM is enabled by default in a CDB root.
- ADDM does not work in a PDB by default, because automatic AWR snapshots are disabled by default in a PDB. To use ADDM in a PDB, you must enable automatic AWR snapshots in the PDB.
- A user whose current container is the CDB root can view ADDM results for the entire CDB. The ADDM results can include information about multiple PDBs. ADDM results related to a PDB are not included if the PDB is unplugged. The ADDM results stored on the CDB root cannot be viewed when the current container is a PDB.
- ADDM results on a PDB provide only PDB-specific findings and recommendations. A user whose current container is a PDB can view ADDM results for the current PDB only. The ADDM results exclude findings that apply to the CDB as a whole, for example, I/O problems relating to the buffer cache size.
- Enabling AWR snapshots on a PDB does not change the ADDM report on the CDB root.
- AWR data on a PDB cannot be accessed from the CDB root.

PDB-Level ADDM Restrictions

Unlike in a non-CDB, ADDM does not report the following issues in a PDB, because these issues apply to a CDB as a whole and do not apply to an individual PDB:

- I/O problems due to:
  - undersized buffer cache
  - undersized streams pool
  - excessive temporary writes
  - excessive checkpoint writes
  - excessive undo writes
  - excessive PQ checkpoint writes
  - excessive truncate writes
  - excessive tablespace DDL checkpoint
  - I/O capacity limit
- SQL hard parsing issues due to:
  - cursor aging
  - out-of-memory failed parse
- SGA sizing issues

ADDM also does not report the following issues in a PDB, because these issues cannot be resolved at a PDB level:

- Cluster messaging related issues, such as network latency, congestion, contention, and lost blocks
- Log file switch waits on archiving and on checkpoint incomplete

- Too many free-buffer waits
- Contention on log buffer waits
- Waits due to CPU bottleneck
- Operating system VM paging
- Session slot wait event
- CPU quantum wait event
- RMAN related wait events, such as PQ queued wait event, PGA limit wait event, and I/O queue wait event

> **See Also:**
>
> - "Enabling ADDM in a Pluggable Database" for information about how to enable ADDM in a PDB
> - "ADDM Views" for more information about the ADDM views `DBA_ADDM_*` and `DBA_ADVISOR_*`
> - "Diagnosing Database Performance Problems with ADDM" for information about how to run ADDM in an Oracle database using the `DBMS_ADDM` package subprograms
> - *Oracle Database 2 Day + Performance Tuning Guide* for information about how to run ADDM in an Oracle database using Cloud Control

Enabling ADDM in a Pluggable Database

ADDM does not work in a pluggable database (PDB) by default, because automatic AWR snapshots are disabled by default in a PDB. To use ADDM in a PDB, you must enable automatic AWR snapshots in the PDB by setting the `AWR_PDB_AUTOFLUSH_ENABLED` initialization parameter to `TRUE` and AWR snapshot interval greater than 0.

To enable ADDM in a PDB:

1. The `AWR_PDB_AUTOFLUSH_ENABLED` initialization parameter has a default value of `TRUE`. If it is not set to `TRUE`, use the following command to change it:

   ```
   SQL> ALTER SYSTEM SET AWR_PDB_AUTOFLUSH_ENABLED=TRUE;
   ```

2. Set the AWR snapshot interval greater than 0 in the PDB using the command as shown in the following example:

   ```
   SQL> EXEC dbms_workload_repository.modify_snapshot_settings(interval=>60);
   ```

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

Real-Time ADDM Analysis

Introduced in Oracle Enterprise Manager Cloud Control (Cloud Control) 12*c*, Real-Time ADDM helps you to analyze and resolve problems in unresponsive or hung databases that traditionally require you to restart the database. Real-Time ADDM runs through a set of predefined criteria to analyze the current performance of the database. After analyzing the problem, Real-Time ADDM helps you to resolve the identified issues—such as deadlocks, no response, shared pool contention, and other exception situations—without having to restart the database.

This section describes Real-Time ADDM and contains the following topics:

- Real-Time ADDM Connection Modes
- Real-Time ADDM Triggers
- Real-Time ADDM Trigger Controls

> **See Also:**
>
> *Oracle Database 2 Day + Performance Tuning Guide* for information about using Real-Time ADDM with Cloud Control

Real-Time ADDM Connection Modes

Depending on the database state, Real-Time ADDM uses two different types of connection modes when connecting to the database using Cloud Control:

- Normal connection

  In this mode, Real-Time ADDM performs a normal JDBC connection to the database. This mode is intended to perform extensive performance analysis of the database when some connectivity is available.

- Diagnostic connection

  In this mode, Real-Time ADDM performs a latch-less connection to the database. This mode is intended for extreme non-responsive situations when a normal JDBC connection is not possible.

Real-Time ADDM Triggers

Real-Time ADDM proactively detects transient database performance issues. To do this, Real-Time ADDM runs automatically every 3 seconds and uses in-memory data to diagnose any performance spikes in the database.

Real-Time ADDM triggers an analysis automatically when a performance problem is detected, as described in the following steps:

1. Every 3 seconds, the manageability monitor process (MMON) performs an action to obtain performance statistics without lock or latch.
2. The MMON process checks these statistics and triggers a Real-Time ADDM analysis if any of the issues listed in Table 7-1 are detected.
3. The MMON child process creates the report and stores it in the AWR.

   To view metadata for the report, use the `DBA_HIST_REPORTS` view.

Table 7-1 lists the issues and conditions that trigger a Real-Time ADDM analysis.

Table 7-1 Triggering Issues and Conditions for Real-Time ADDM

Issue	Condition
High load	Average active sessions are greater than 3 times the number of CPU cores
I/O bound	I/O impact on active sessions based on single block read performance
CPU bound	Active sessions are greater than 10% of total load and CPU utilization is greater than 50%
Over-allocated memory	Memory allocations are over 95% of physical memory
Interconnect bound	Based on single block interconnect transfer time
Session limit	Session limit is close to 100%
Process limit	Process limit is close to 100%
Hung session	Hung sessions are greater than 10% of total sessions
Deadlock detected	Any deadlock is detected

See Also:

Oracle Database Reference for information about the `DBA_HIST_REPORTS` view

Real-Time ADDM Trigger Controls

To ensure that the automatic triggers do not consume too many system resources and overwhelm the system, Real-Time ADDM employs the following controls:

- Duration between reports

  If a Real-Time ADDM report was created in the past 5 minutes by the automatic trigger, then no new reports will be generated.

- Oracle RAC control

  Automatic triggers are local to the database instance. For Oracle RAC, only one database instance can create a Real-Time ADDM report at a given time because a lock is required and a query is performed by the MMON child process before the report is actually generated.

- Repeated triggers

  An automatic trigger for any issue must have an impact of 100% or higher than the previous report with the same triggering issue within the past 45 minutes. For example, if a

report is triggered for active sessions with an impact of 8 sessions, then in order for another report to trigger within the next 45 minutes, there must be at least 16 active sessions. In this case, the reported problem with the database is becoming more severe over time. On the other hand, if the same report is being generated once every 45 minutes, then the database is experiencing a persistent problem that has a consistent impact.

- Newly identified issues

  If a new issue is detected (that was not previously detected within the past 45 minutes), then a new report is generated. For example, if a report is triggered for 8 active sessions and a new deadlock issue is detected, then a new report is generated regardless of the new active sessions load.

ADDM Analysis Results

In addition to problem diagnostics, ADDM recommends possible solutions. ADDM analysis results are represented as a set of findings. See Example 7-1 for an example of an ADDM analysis result. Each ADDM finding can belong to one of the following types:

- Problem findings describe the root cause of a database performance problem.
- Symptom findings contain information that often lead to one or more problem findings.
- Information findings are used for reporting information that are relevant to understanding the performance of the database, but do not constitute a performance problem (such as non-problem areas of the database and the activity of automatic database maintenance).
- Warning findings contain information about problems that may affect the completeness or accuracy of the ADDM analysis (such as missing data in AWR).

Each problem finding is quantified by an impact that is an estimate of the portion of `DB time` caused by the finding's performance issue. A problem finding can be associated with a list of recommendations for reducing the impact of the performance problem. The types of recommendations include:

- Hardware changes: adding CPUs or changing the I/O subsystem configuration
- Database configuration: changing initialization parameter settings
- Schema changes: hash partitioning a table or index, or using automatic segment-space management (ASSM)
- Application changes: using the cache option for sequences or using bind variables
- Using other advisors: running SQL Tuning Advisor on high-load SQL or running Segment Advisor on hot objects

A list of recommendations can contain various alternatives for solving the same problem; you do not have to apply all the recommendations to solve a specific problem. Each recommendation has a benefit which is an estimate of the portion of `DB time` that can be saved if the recommendation is implemented. Recommendations are composed of actions and rationales. You must apply all the actions of a recommendation to gain the estimated benefit. The rationales are used for explaining why the set of actions were recommended and to provide additional information to implement the suggested recommendation.

Reviewing ADDM Analysis Results: Example

Consider the following section of an ADDM report in Example 7-1.

Example 7-1 Example ADDM Report

```
FINDING 1: 31% impact (7798 seconds)
-----------------------------------
SQL statements were not shared due to the usage of literals. This resulted in
additional hard parses which were consuming significant database time.

RECOMMENDATION 1: Application Analysis, 31% benefit (7798 seconds)
  ACTION: Investigate application logic for possible use of bind variables
    instead of literals. Alternatively, you may set the parameter
    "cursor_sharing" to "force".
  RATIONALE: SQL statements with PLAN_HASH_VALUE 3106087033 were found to be
    using literals. Look in V$SQL for examples of such SQL statements.
```

In Example 7-1, the finding points to a particular root cause, the usage of literals in SQL statements, which is estimated to have an impact of about 31% of total `DB time` in the analysis period.

The finding has a recommendation associated with it, composed of one action and one rationale. The action specifies a solution to the problem found and is estimated to have a maximum benefit of up to 31% `DB time` in the analysis period. Note that the benefit is given as a portion of the total `DB time` and not as a portion of the finding's impact. The rationale provides additional information on tracking potential SQL statements that were using literals and causing this performance issue. Using the specified plan hash value of SQL statements that could be a problem, a DBA could quickly examine a few sample statements.

When a specific problem has multiple causes, ADDM may report multiple problem and symptom findings. In this case, the impacts of these multiple findings can contain the same portion of `DB time`. Because the performance issues of findings can overlap, the sum of the impacts of the findings can exceed 100% of `DB time`. For example, if a system performs many reads, then ADDM might report a SQL statement responsible for 50% of `DB time` due to I/O activity as one finding, and an undersized buffer cache responsible for 75% of `DB time` as another finding.

When multiple recommendations are associated with a problem finding, the recommendations may contain alternatives for solving the problem. In this case, the sum of the recommendations' benefits may be higher than the finding's impact.

When appropriate, an ADDM action may have multiple solutions for you to choose from. In the example, the most effective solution is to use bind variables. However, it is often difficult to modify the application. Changing the value of the `CURSOR_SHARING` initialization parameter is much easier to implement and can provide significant improvement.

Setting Up ADDM

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

The `CONTROL_MANAGEMENT_PACK_ACCESS` parameter should be set to `DIAGNOSTIC` or `DIAGNOSTIC+TUNING` to enable automatic database diagnostic monitoring. The default setting is `DIAGNOSTIC+TUNING`. Setting `CONTROL_MANAGEMENT_PACK_ACCESS` to `NONE` disables ADDM.

The `STATISTICS_LEVEL` parameter should be set to the `TYPICAL` or `ALL` to enable automatic database diagnostic monitoring. The default setting is `TYPICAL`. Setting `STATISTICS_LEVEL` to `BASIC` disables many Oracle Database features, including ADDM, and is strongly discouraged.

> **See Also:**
>
> *Oracle Database Reference* for information about the `CONTROL_MANAGEMENT_PACK_ACCESS` and `STATISTICS_LEVEL` initialization parameters

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

To determine the correct setting for the DBIO_EXPECTED parameter:

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

   Note that this measurement is for random I/O, which includes seek time if you use standard hard drives. Typical values for hard drives are between 5000 and 20000 microseconds.

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

   For example, if the measured value if 8000 microseconds, you should execute the following command as SYS user:

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

Diagnosing Database Performance Problems with ADDM

To diagnose database performance problems, first review the ADDM analysis results that are automatically created each time an AWR snapshot is taken. If a different analysis is required (such as a longer analysis period, using a different `DBIO_EXPECTED` setting, or changing the analysis mode), you can run ADDM manually as described in this section.

ADDM can analyze any two AWR snapshots (on the same database), as long as both snapshots are still stored in AWR (have not been purged). ADDM can only analyze instances that are started before the beginning snapshot and remain running until the ending snapshot. Additionally, ADDM will not analyze instances that experience significant errors when generating AWR snapshots. In such cases, ADDM will analyze the largest subset of instances that did not experience these problems.

The primary interface for diagnostic monitoring is Cloud Control. Whenever possible, run ADDM using Cloud Control, as described in *Oracle Database 2 Day + Performance Tuning Guide*. If Cloud Control is unavailable, then run ADDM using the `DBMS_ADDM` package. To run the `DBMS_ADDM` APIs, the user must be granted the `ADVISOR` privilege.

Running ADDM in Database Mode

For Oracle RAC configurations, you can run ADDM in Database mode to analyze all instances of the databases. For single-instance configurations, you can still run ADDM in Database mode; ADDM will behave as if running in Instance mode.

To run ADDM in Database mode, use the `DBMS_ADDM.ANALYZE_DB` procedure:

```
BEGIN
DBMS_ADDM.ANALYZE_DB (
   task_name           IN OUT VARCHAR2,
   begin_snapshot      IN     NUMBER,
   end_snapshot        IN     NUMBER,
   db_id               IN     NUMBER := NULL);
END;
/
```

The `task_name` parameter specifies the name of the analysis task that will be created. The `begin_snapshot` parameter specifies the snapshot number of the beginning snapshot in the analysis period. The `end_snapshot` parameter specifies the snapshot number of the ending snapshot in the analysis period. The `db_id` parameter specifies the database identifier of the database that will be analyzed. If unspecified, this parameter defaults to the database identifier of the database to which you are currently connected.

The following example creates an ADDM task in database analysis mode, and executes it to diagnose the performance of the entire database during the time period defined by snapshots 137 and 145:

```
VAR tname VARCHAR2(30);
BEGIN
  :tname := 'ADDM for 7PM to 9PM';
  DBMS_ADDM.ANALYZE_DB(:tname, 137, 145);
END;
/
```

Running ADDM in Instance Mode

To analyze a particular instance of the database, you can run ADDM in Instance mode. To run ADDM in Instance mode, use the `DBMS_ADDM.ANALYZE_INST` procedure:

```
BEGIN
DBMS_ADDM.ANALYZE_INST (
   task_name           IN OUT VARCHAR2,
   begin_snapshot      IN     NUMBER,
   end_snapshot        IN     NUMBER,
   instance_number     IN     NUMBER := NULL,
   db_id               IN     NUMBER := NULL);
END;
/
```

The `task_name` parameter specifies the name of the analysis task that will be created. The `begin_snapshot` parameter specifies the snapshot number of the beginning snapshot in the analysis period. The `end_snapshot` parameter specifies the snapshot number of the ending snapshot in the analysis period. The `instance_number` parameter specifies the instance number of the instance that will be analyzed. If unspecified, this parameter defaults to the instance number of the instance to which you are currently connected. The `db_id` parameter specifies the database identifier of the database that will be analyzed. If unspecified, this parameter defaults to the database identifier of the database to which you are currently connected.

The following example creates an ADDM task in instance analysis mode, and executes it to diagnose the performance of instance number 1 during the time period defined by snapshots 137 and 145:

```
VAR tname VARCHAR2(30);
BEGIN
  :tname := 'my ADDM for 7PM to 9PM';
  DBMS_ADDM.ANALYZE_INST(:tname, 137, 145, 1);
```

```
END;
/
```

Running ADDM in Partial Mode

To analyze a subset of all database instances, you can run ADDM in Partial mode. To run ADDM in Partial mode, use the `DBMS_ADDM.ANALYZE_PARTIAL` procedure:

```
BEGIN
DBMS_ADDM.ANALYZE_PARTIAL (
   task_name           IN OUT VARCHAR2,
   instance_numbers    IN     VARCHAR2,
   begin_snapshot      IN     NUMBER,
   end_snapshot        IN     NUMBER,
   db_id               IN     NUMBER := NULL);
END;
/
```

The `task_name` parameter specifies the name of the analysis task that will be created. The `instance_numbers` parameter specifies a comma-delimited list of instance numbers of instances that will be analyzed. The `begin_snapshot` parameter specifies the snapshot number of the beginning snapshot in the analysis period. The `end_snapshot` parameter specifies the snapshot number of the ending snapshot in the analysis period. The `db_id` parameter specifies the database identifier of the database that will be analyzed. If unspecified, this parameter defaults to the database identifier of the database to which you are currently connected.

The following example creates an ADDM task in partial analysis mode, and executes it to diagnose the performance of instance numbers 1, 2, and 4, during the time period defined by snapshots 137 and 145:

```
VAR tname VARCHAR2(30);
BEGIN
  :tname := 'my ADDM for 7PM to 9PM';
  DBMS_ADDM.ANALYZE_PARTIAL(:tname, '1,2,4', 137, 145);
END;
/
```

Displaying an ADDM Report

To display a text report of an executed ADDM task, use the `DBMS_ADDM.GET_REPORT` function:

```
DBMS_ADDM.GET_REPORT (
   task_name           IN VARCHAR2
  RETURN CLOB);
```

The following example displays a text report of the ADDM task specified by its task name using the `tname` variable:

```
SET LONG 1000000 PAGESIZE 0;
SELECT DBMS_ADDM.GET_REPORT(:tname) FROM DUAL;
```

Note that the return type of a report is a `CLOB`, formatted to fit line size of 80. For information about reviewing the ADDM analysis results in an ADDM report, see "ADDM Analysis Results".

Re-Running Failed ADDM Tasks

The main purpose of Automatic Database Diagnostic Monitoring (ADDM) is to run automatically at the end of each AWR snapshot (default is every hour.) If an ADDM task fails, the failure is written to the database trace logs.

In addition, the `DBA_ADDM_PENDING_AUTOTASKS` view tracks the ADDM tasks being executed. The views are accessible at both CDB and PDB levels and contain information about the tasks executed at each level. ADB and Cloud users can see information about the tasks that were executed in their PDB, obtain more diagnostic information on failed tasks, and manually re-execute those tasks. Once a task is run successfully, the entry is removed, otherwise failed tasks remain in the view. When the corresponding AWR snapshots age out, any associated failed tasks age out.

The `FAILED_AUTO_TASKS_REPORT` function returns a report about the ADDM tasks that had failed and are visible in the `DBA_ADDM_PENDING_AUTOTASKS` views for the corresponding PDB level. The report can be generated to visualize either all failed tasks or only those that meet certain criteria, such as a time range, or identifier. The information in the report helps identify and diagnose the existence of failed ADDM tasks and their possible root causes, and thus useful to a DBA.

The `REEXECUTE_FAILED_AUTO_TASKS` procedure re-executes the tasks registered as failed in the `DBA_ADDM_PENDING_AUTOTASKS` view. To re-run a task, the user specifies the range for the snapshots or a time interval. The duration of the re-execution is time constrained by a user-specified parameter. Once a task has executed successfully, its corresponding row is removed from the `DBA_ADDM_PENDING_AUTOTASKS` view.

FAILED_AUTO_TASKS_REPORT Function

This function creates a plain text, user-readable report for the failed tasks registered in the `DBA_ADDM_PENDING_AUTOTASKS` views.

Syntax

```
DBMS_ADDM.FAILED_AUTO_TASKS_REPORT(
   instance_number      IN   NUMBER :=NULL,
   begin_snapshot       IN   NUMBER :=NULL,
   end_snapshot         IN   NUMBER :=NULL,
   dbid                 IN   NUMBER :=NULL)
```

Parameters

Table 7-2 FAILED_AUTO_TASKS_REPORT Function Parameters

Parameter	Description
`instance_number`	Instance number for the tasks to be reported.
`begin_snapshot`	Earliest begin snapshot ID for the tasks to be reported.
`end_snapshot`	Latest end snapshot ID for the tasks to be reported.
`dbid`	Database ID for the tasks to be reported.

REEXECUTE_FAILED_AUTO_TASKS Procedure

This procedure re-executes the tasks registered in the DBA_ADDM_PENDING_AUTOTASKS views.

This API can be called from both the CDB and PDB level. The user specifies the range for the snapshots or a time interval. The duration of the re-execution is time constrained by a user-specified parameter or a default value. Once a task has executed successfully, its corresponding row is removed from the DBA_ADDM_PENDING_AUTOTASKS view.

Syntax

```
DBMS_ADDM.REXECUCUTE_FAILED_AUTO_TASKS(
   instance_number     IN   NUMBER :=NULL,
   begin_snapshot      IN   NUMBER :=NULL,
   end_snapshot        IN   NUMBER :=NULL,
   dbid                IN   NUMBER :=NULL,
   time_budget_in_sec  IN   NUMBER :=NULL,
   max_attempts        IN   NUMBER :=NULL);
```

Parameters

Table 7-3 REEXECUTE_FAILED_AUTO_TASKS Function Parameters

Parameter	Description
instance_number	Instance number for the tasks to be re executed.
begin_snapshot	Earliest begin snapshot ID for the tasks to be re executed.
end_snapshot	Latest end snapshot ID for the tasks to be re executed.
dbid	Database ID for the tasks to be re executed.
time_budget_in_sec	Forced time out to run the procedure after exiting.
max_attempts	Maximum number of attempts to re execute a task.

ADDM Views

Typically, you should view ADDM analysis using Cloud Control or DBMS_ADDM package subprograms.

However, you can also get ADDM information using the DBA_ADDM_* and DBA_ADVISOR_* views. This group of views includes:

- DBA_ADVISOR_FINDINGS

  This view displays all the findings discovered by all advisors. Each finding is displayed with an associated finding ID, name, and type. For tasks with multiple executions, the name of each task execution associated with each finding is also listed.

- DBA_ADDM_FINDINGS

  This view contains a subset of the findings displayed in the related DBA_ADVISOR_FINDINGS view. This view only displays the ADDM findings discovered by all advisors. Each ADDM finding is displayed with an associated finding ID, name, and type.

- DBA_ADDM_PENDING_AUTOTASKS

This view tracks the automatic ADDM tasks that are being executed. The views are accessible at both CDB and PDB levels and contain information about the tasks executed at each level. ADB and Cloud users can see information about the tasks that were executed in their PDB, obtain more diagnostic information on failed tasks, and manually re-execute those tasks. Once a task is executed successfully, the entry is removed from the view. Failed tasks automatically age out and are removed when the corresponding AWR snapshots age out.

- `DBA_ADVISOR_FINDING_NAMES`

  This view lists all finding names registered with the advisor framework.

- `DBA_ADVISOR_RECOMMENDATIONS`

  This view displays the results of completed diagnostic tasks with recommendations for the problems identified in each execution. The recommendations should be reviewed in the order of the `RANK` column, as this relays the magnitude of the problem for the recommendation. The `BENEFIT` column displays the benefit to the system you can expect after the recommendation is performed. For tasks with multiple executions, the name of each task execution associated with each advisor task is also listed.

- `DBA_ADVISOR_TASKS`

  This view provides basic information about existing tasks, such as the task ID, task name, and when the task was created. For tasks with multiple executions, the name and type of the last or current execution associated with each advisor task is also listed.

See Also:

- *Oracle Database Reference* for more information about `DBA_ADDM_*` and `DBA_ADVISOR_*` views
- "Displaying an ADDM Report"

ADDM Spotlight

Once the Automatic Database Diagnostic Monitor (ADDM) has been configured, you can use the ADDM Spotlight to consolidate the recommendations from ADDM for your Oracle database.

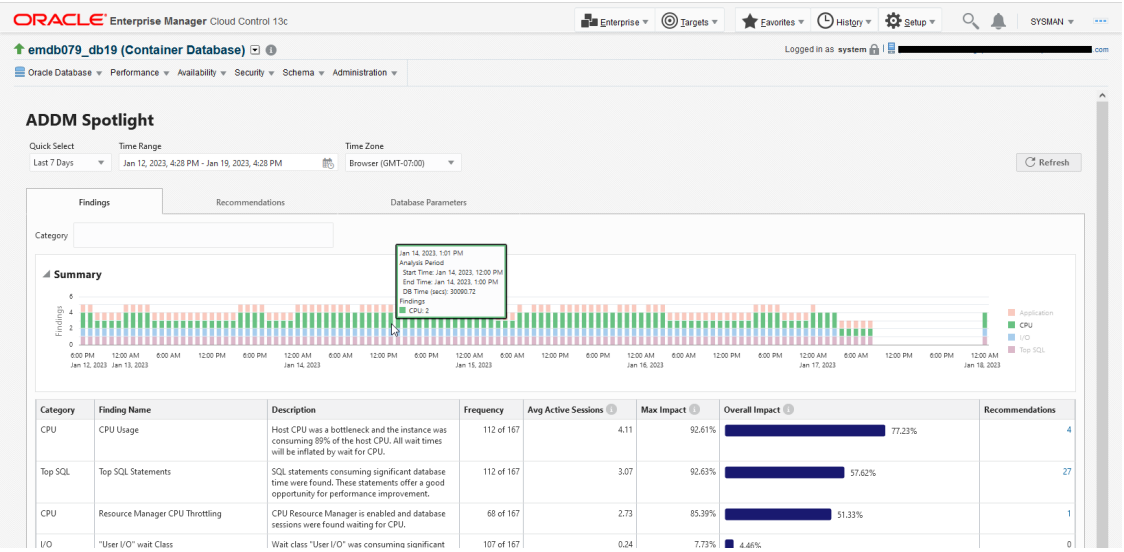
ADDM Spotlight provides a visual representation of critical performance categories, such as `CPU Usage`, `Top SQL Statements`, `Resource Manager CPU Throttling`, and `"User I/O" wait Class`. It rates the overall impact on system performance and provides recommendations. It helps locate root causes of any performance problems, provides recommendations for correcting the problems, and identifies problem areas of the system.

While in Enterprise Manager for your database, you select from the Performance drop-down menu the `ADDM Spotlight` item. The default time range is the last 24 hours when first entering the ADDM Spotlight. You can use the `Quick Select` pull-down menu to specify other common ranges, such as the last 7 days. Or you can specify a specific time range with time zone.

The `Findings` tab of ADDM Spotlight displays a graphic summary of impacts identified by category for that time range. Each point in time in the graph has a bar whose height represents the number of findings from all categories. Each bar is further subdivided by color-coded impact categories whose size on each individual bar is its number of findings. When you mouse-over a colored area of a specific bar, the tooltip's detailed information about that

category's impact at the database time is displayed, as shown in Figure 7-1. Within the `Findings` tab is a summary table showing each finding that was reported during the period. For each finding, it shows the category, frequency, impact, and number of recommendations.

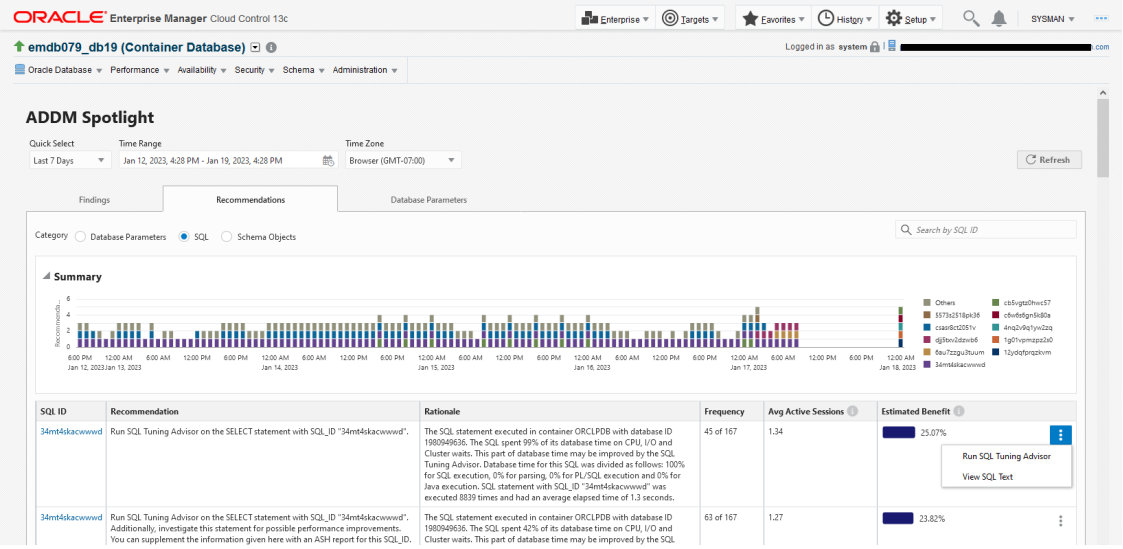
Figure 7-1 The Findings tab of ADDM Spotlight



The `Recommendations` tab of ADDM Spotlight has radio buttons along the top for specifying the category: `Database Parameters`, `SQL`, and `Schema Objects`. Each point in time in the graphic summary for the category has a bar whose height represents the count of performance impacts. Further, each bar is subdivided by color-coded impact of specific items in that category.

For example, if the `SQL` category is selected as shown in Figure 7-2, the color-coded areas of each bar represents the count of individual SQL commands. The table below the summary graph provides details by `SQL ID`, the recommendation for improving its performance, the frequency of its occurrence, and the estimated benefit of implementing the recommendation for that SQL ID. Similarly, the `Database Parameters` and the `Scheme Objects` categories display recommendations for those respective items. The information provided for the `SQL` category is useful for subsequent actions, such as those performed with the `SQL Tuning Advisor`.

Figure 7-2 The Recommendations Tab of ADDM Spolight



If the `Database Parameters` tab of ADDM Spotlight is selected as shown in Figure 7-3, the table shows information abot the values of those database parameters, such as `Parameter Name`, `Begin Value`, `End Value`, `Default`, and `Changed`. Filter checkboxes for `High Impact`, `Non-Default Values`, `Changes`, and `Recommendations` help focus the database parameter table on areas of interest. The `Changes` column has a pop-up that shows the change history for that database parameter.

Figure 7-3 The Database Parameters Tab of ADDM Spotlight