



# CHAPTER 9

Performance Management  
in Oracle Database 10g



Oracle Database 10g Release 1 revolutionizes the performance diagnostic and tuning as we all know it. All the manual data collection and analysis methods we discussed in previous chapters have now been fully automated and are part of the database. The mechanism and the intelligent architecture called Manageability Infrastructure is employed by Oracle Database 10g to collect various statistical data. Not only does it satisfy all the requirements for a fast and accurate root cause analysis discussed in Chapter 4, but it also offers remedial solutions in terms of recommendations, advisories, and server-generated early warning alerts.

In this chapter we discuss the components of the Manageability Infrastructure that make this automatic diagnosis and tuning possible. First, we will discuss the various types of database statistics gathered by Oracle Database 10g.

## Database Statistics

Oracle Database 10g gathers and analyzes performance-related statistical data to diagnose problems. The data is captured using lightweight data capture methods that do not add any measurable load to the system. It reports top problems and offers corrective actions, or advisories, for resolving them. It also reports nonproblematic areas, so you can focus only on problematic areas.

The collected statistical data can be broadly categorized into the following types:

- Time model statistics
- Wait model statistics
- Operating system statistics
- Additional SQL statistics
- Database metrics

## Time Model Statistics

Time model statistics are new in Oracle Database 10g. As we mentioned in Chapter 2, OWI only reports the wait time for events that a session waited on. Time model statistics provide the breakdown of the time a session spent in various steps, such as hard parsing, soft parsing, SQL execution, PL/SQL execution, Java execution, and so on, while performing the actual task. These statistics are displayed by the V\$SESS\_TIME\_MODEL view. Summarized time model statistics at the system level are displayed by V\$SYS\_TIME\_MODEL as shown in the following example:

```
select stat_name, value
from v$sys_time_model;
```

STAT_NAME	VALUE
DB time	835243622
DB CPU	633280130
background elapsed time	3737809876
background cpu time	1869951797
sequence load elapsed time	122400
parse time elapsed	192685706
hard parse elapsed time	151503406
sql execute elapsed time	828428484
connection management call elapsed time	856270
failed parse elapsed time	243612
failed parse (out of shared memory) elapsed time	0
hard parse (sharing criteria) elapsed time	861810
hard parse (bind mismatch) elapsed time	798655
PL/SQL execution elapsed time	94173710
inbound PL/SQL rpc elapsed time	0
PL/SQL compilation elapsed time	94186909
Java execution elapsed time	0

17 rows selected.

The most important time model statistic is the *DB time*. It shows the total time spent by the sessions in database calls. It is equivalent to the sum of CPU time and wait times of all sessions not waiting on events classified by the *Idle* wait class. However, it is timed separately. The following breakdown of the time model statistics shows which statistics are subsets:

1. background elapsed time
  2. background cpu time
1. DB time
  2. DB CPU
  2. connection management call elapsed time
  2. sequence load elapsed time
  2. sql execute elapsed time
  2. parse time elapsed
    3. hard parse elapsed time
      4. hard parse (sharing criteria) elapsed time
      5. hard parse (bind mismatch) elapsed time
  3. failed parse elapsed time
    4. failed parse (out of shared memory) elapsed time
2. PL/SQL execution elapsed time
2. inbound PL/SQL rpc elapsed time
2. PL/SQL compilation elapsed time
2. Java execution elapsed time

If the session spends less time in database calls, it is performing better. Your tuning goal should be to reduce the overall *DB time* for the session.

## Wait Model Statistics

By now, wait model statistics are nothing new to you. Oracle Database 10g Release 1 tracks over 800 wait events to report time spent by the session waiting on those events. These are classified in 12 wait classes. This classification allows easier high-level analysis of the wait events. The classification is based on the solution that normally applies to correcting a problem with the wait event.

## Operating System Statistics

Operating systems statistics provide information about system resources utilization such as CPU, memory, and file systems. In Oracle versions prior to Oracle Database 10g, some of these statistics were not available from within the database. You had to issue OS commands or use OS level tools to gather machine-level statistics to investigate hardware-related issues. Oracle Database 10g captures such statistics within the database and reports them in the view V\$OSSTAT, as shown next:

```
select stat_name, value
from v$osstat;
```

STAT_NAME	VALUE
NUM_CPUS	1
IDLE_TICKS	22201887
BUSY_TICKS	3385285
USER_TICKS	2101041
SYS_TICKS	1284244
IOWAIT_TICKS	78316
AVG_IDLE_TICKS	22201887
AVG_BUSY_TICKS	3385285
AVG_USER_TICKS	2101041
AVG_SYS_TICKS	1284244
AVG_IOWAIT_TICKS	78316
OS_CPU_WAIT_TIME	9.2061E+11
RSRC_MGR_CPU_WAIT_TIME	0
IN_BYTES	123883520
OUT_BYTES	0
AVG_IN_BYTES	123883520
AVG_OUT_BYTES	0

17 rows selected.

## Additional SQL Statistics

Additional SQL statistics provide information at the statement level for wait class time, PL/SQL execution, Java execution, and sampled bind variables. Oracle Database 10g also introduces a new hash value, `SQL_ID`, as a character string for the SQL statement, which is more unique than in earlier versions of Oracle Database.

## Database Metrics

As you all know, almost all of the database statistics reported by various `V$` views are cumulative since the instance startup. As we discussed in Chapter 2, you have to take snapshots at various intervals to find the rate of change in the statistics values reported by these views. In performance diagnostics, this rate of change, or metric, is more important than the cumulative value of the statistics. In Oracle Database 10g, these metrics are readily available for a variety of units, such as time, database calls, and transactions. Most of these metrics are maintained at a one-minute interval and are exposed via various `V$` views. Metrics history is exposed via various `V$` metric history views. A few of the metric views are listed in Table 9-1.

## New Background Processes

In Oracle Database 10g, two new background processes are responsible for collecting, sampling, and maintaining the statistical data. These processes are the `MMON` and `MMNL`.

---

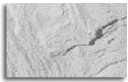
<code>V\$METRICNAME</code>	Lists the metric ID, metric name with its metric group name, and group ID. There are a total of 10 metric groups for over 180 different metrics.
<code>V\$EVENTMETRIC</code>	Displays values of the wait event metrics.
<code>V\$WAITCLASSMETRIC</code>	Displays values of the wait event class metrics.
<code>V\$SESSMETRIC</code>	Displays values of the metrics for the session-level statistics.
<code>V\$SYSMETRIC</code>	Displays values of the metrics for the system-level statistics.
<code>V\$FILEMETRIC</code>	Displays values of the file metrics.

---

**TABLE 9-1.** *Metric Views (Not a Complete List)*

The MMON (Manageability Monitor) process is responsible for various manageability tasks such as taking snapshots of various statistical information at prescribed time intervals and issuing alerts when metric values exceed defined thresholds. The MMON process can spawn multiple slave processes to perform these tasks.

The MMNL (the Manageability Monitor—Lightweight) process is responsible for computing various metrics and taking snapshots of active sessions at every second. We'll discuss this further in the section "Active Session History."



#### NOTE

*The database instance does not crash when MMON or MMNL process is terminated for whatever reason. The PMON process will restart the failed process. Relevant messages will be written to the alert log file.*

So, how does Oracle Database 10g actually collect, report, and maintain the performance statistics data? This is collectively done by the Automatic Workload Repository.

## Automatic Workload Repository

The Oracle Database kernel keeps numerous statistics that can be used to identify performance problems. Unfortunately, many database statistics are not stored on the disk. This missing history data imposes a great challenge to performance practitioners to identify and fix the problem that occurred in the past. Historical data and statistics are very important for database performance monitoring and capacity planning.

Oracle Database 10g enhances the data collection mechanism with the introduction of the Automatic Workload Repository and Active Session History, which replace the previous performance data gathering tools such as Statspack and utlbstat/utlestat. The major differences between the current repositories from previous such tools include the following:

- Earlier tools had no automated way of interpreting the collected data and results produced by supplied reports. The raw data and the formatted output from the Statspack tool required manual interpretation.
- There is no proactive monitoring in the earlier tools. The DBA could tune the database or solve the problem only after its occurrence. The Automatic Database Diagnostics Monitor (ADDM) tool in Oracle Database 10g, detects a problem before it strikes and advises possible solutions.

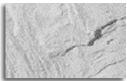
- ADDM works at a fine-grained level to detect problems at the database segment level. For example, if there is a hot block or hot object causing performance problems, ADDM identifies the object and provides tuning advice, while this information would not be directly captured in Statspack report.
- Problems such as excessive logins and logoffs, ITL waits, and RAC-related service issues were not captured in the Statspack report, but ADDM captures this information and offers tuning advice.

The Automatic Workload Repository (AWR) captures all the data that Statspack captured in the earlier versions of the Oracle RDBMS. In addition, it captures the new statistical data described in the preceding sections. AWR is the performance data warehouse of Oracle Database 10g. This data resides in the SGA and is also stored on disk.

The captured data is displayed via available views and AWR report, similar to that generated by Statspack. However, accessing this information is much simpler and easier with Oracle Enterprise Manager (EM).

## Repository Snapshots

By default, out-of-box, AWR goes to work. Using MMON, the new background process, it takes snapshots of database statistics every 60 minutes. The data is stored in a set of tables under the SYS schema in the new mandatory tablespace, SYSAUX. Like Statspack, a unique identifier called SNAP\_ID identifies each snapshot. By default, data older than seven days is purged. Many of these tables are partitioned by range, using the DBID, the unique database identifier, and the SNAP\_ID column as their partitioning key. This partitioning strategy is very helpful in data access and maintenance operations.



### NOTE

*Even if you do not have Oracle Partitioning option installed, Oracle Database 10g will create required partitioned tables and indexes for its internal use. You will still need to license and install Oracle Partitioning for your use.*

In Oracle Database 10g Release 1 there are 140 tables that store the AWR data, and they are accessible via 67 views. The table names are in the format WRI\$\_\*, WRH\$\_\* and WRM\$\_\*. “WR” stands for “Workload Repository,” “I” stands for “internal,” “H” stands for “historical” data, and “M” stands for “metadata.” There are several views that are built upon these base tables. The view names are in the

format `DBA_HIST_*`; the asterisk (\*) represents what statistics the view or table shows. For example, the view `DBA_HIST_LATCH` shows *latch*-related statistics for historical snapshots. This view is based on the base table `WRH$_LATCH`. You can use the following SQL code to list the names of these tables and views:

```
select table_name
from   dba_tables
where  owner = 'SYS'
and    table_name like 'WR%';

select view_name
from   dba_views
where  owner = 'SYS'
and    view_name like 'DBA\_HIST\_%' escape '\';
```

## Snapshot Baselines

AWR allows creation of snapshot baselines to capture performance statistics during a particular time frame. Generally, you will create a baseline for snapshots taken when a typical workload was processed and the performance was acceptable. When the same workload is processed later and the performance is not acceptable, a new baseline for this time frame can be created to compare against the previous baseline. The difference in the performance statistics can shed light on the cause of slow performance. The baseline snapshots are also called Preserved Snapshot Sets.

## Using EM to Manage AWR

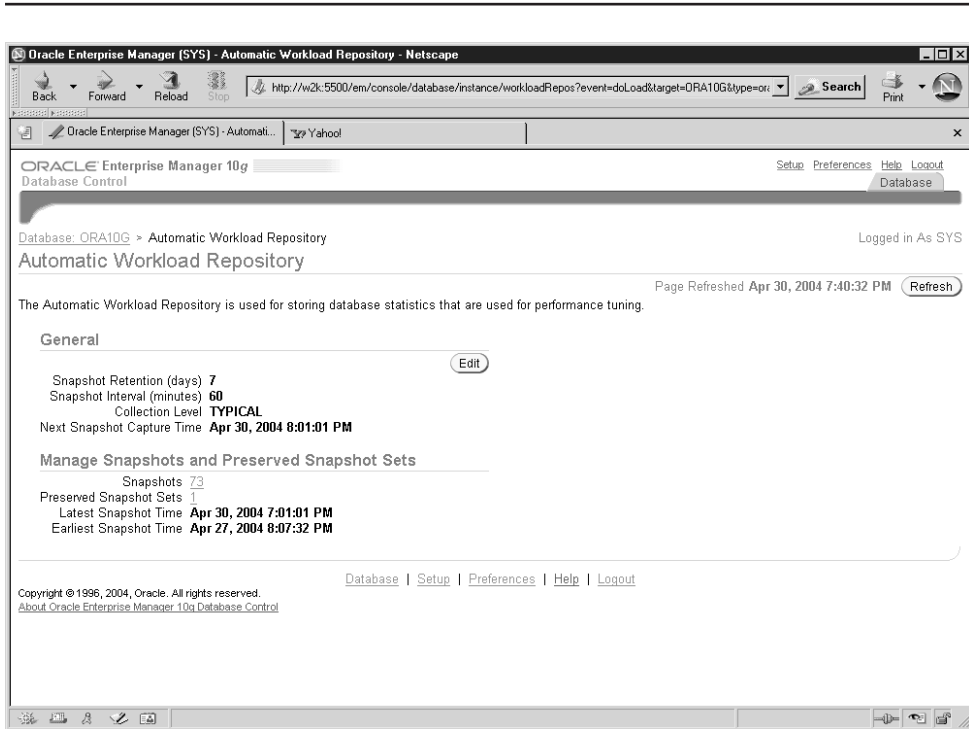
Oracle EM is the primary tool to interact with AWR. However, you can also use Oracle-supplied package procedures to manage AWR functionality. The manual procedures are described in the next section.

You can access AWR from the Database Control home page of Oracle Enterprise Manager. From this page, first click the Administration link to access the Administration page and then click the Automatic Workload Repository link under the Workload heading toward the bottom of the page.

The Automatic Workload Repository page is shown in Figure 9-1. On this page you can view, edit, and manage AWR snapshots.

The General information block shows current snapshot settings and the next snapshot time. The Edit button takes you to a new page where you can change the snapshot interval and snapshot retention and also drill down to change the statistics collection level.

Under the Manage Snapshots and Preserved Snapshot Sets heading you see the total number of available snapshots and preserved snapshots that are used in baselines and the date and time of the earliest and latest available snapshot.



**FIGURE 9-1.** *Workload Repository home page*

Clicking the number shown after Snapshots takes you to the Snapshots home page, where you can view information about snapshots. Figure 9-2 shows the home page for Snapshots.

Using the pull-down Actions menu you can perform following tasks:

- Create Preserved Snapshot Set (baseline creation)
- View Report (by comparing statistics from two snapshots)
- Create ADDM task (to perform analysis against a set of snapshots)
- Delete Snapshot Range
- Compare Timelines (using two sets of snapshots from two time periods)
- Create SQL Tuning Set (to track and tune SQL statements)

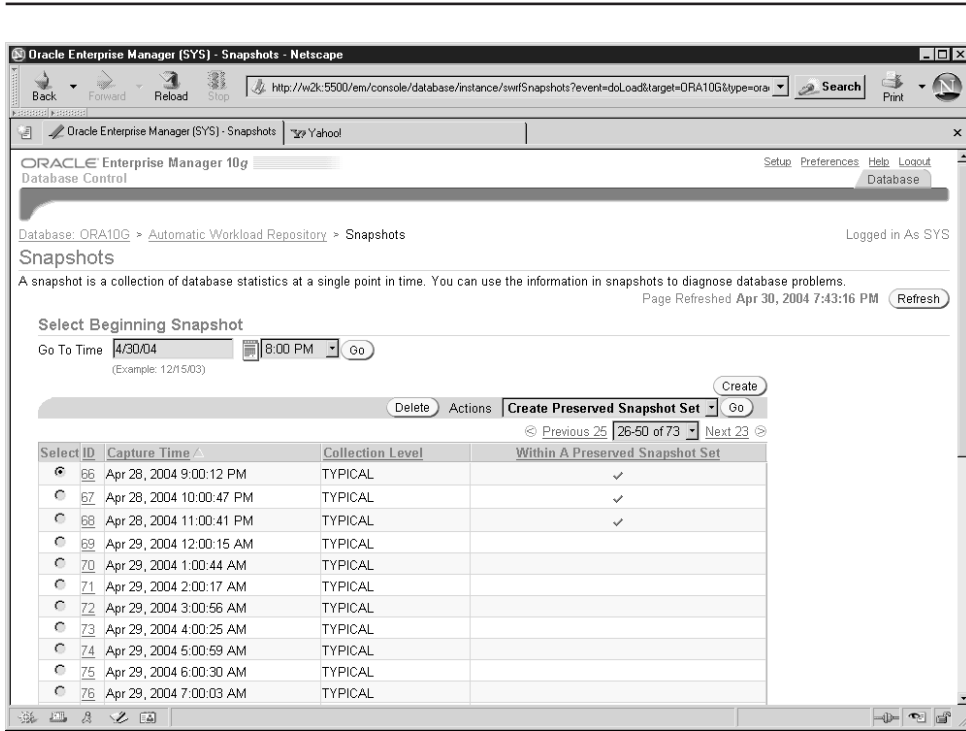
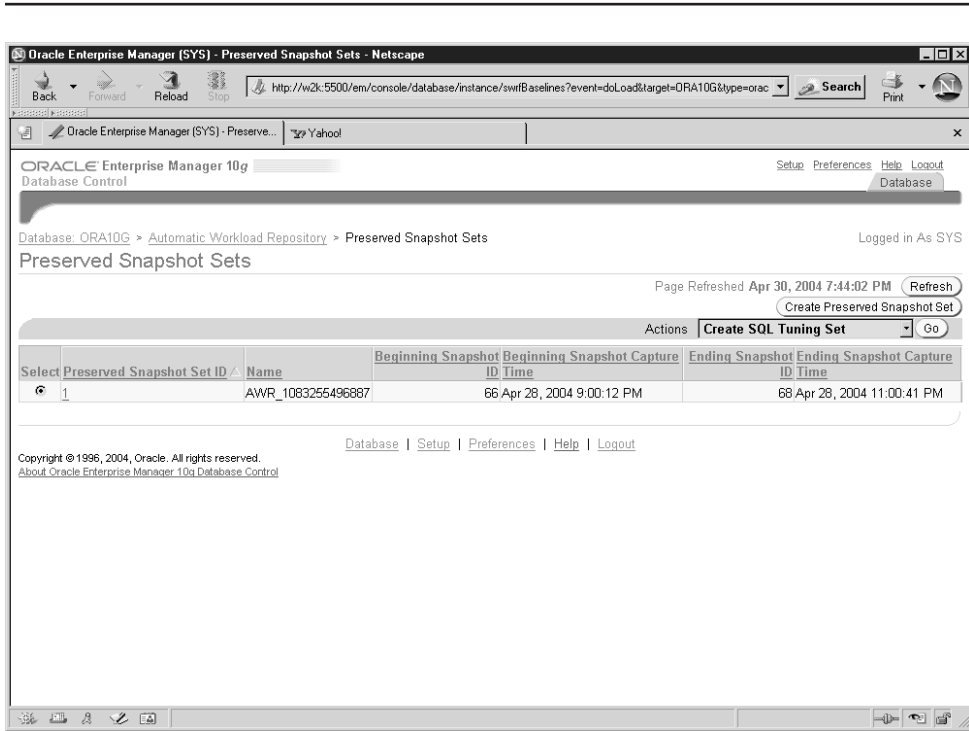


FIGURE 9-2. Snapshots home page

You can click the Create button to manually create a new snapshot. The Delete button deletes the specified snapshot. The Go button initiates the selected task from the pull-down Actions menu. We found the Go, Create, and Delete buttons on this page a bit confusing at first, perhaps because of their positions on the page. The column Within a Preserved Snapshot Set shows a checkmark for the snapshot IDs that are part of a baseline or the Preserved Snapshot Set.

Back on the AWR home page (Figure 9-1), clicking the number shown next to Preserved Snapshot Sets takes you to the home page for Preserved Snapshots Sets, as shown in Figure 9-3. This page lists the details of the Preserved Snapshot Sets, or the baselines, established in the AWR. The pull-down Actions menu offers you various tasks that can be performed against these baseline snapshots. Please note that the Preserved Snapshot Set ID is different from the snapshot ID. The former applies to the baselines or the preserved AWR snapshots only.



**FIGURE 9-3.** Preserved Snapshot Sets (baselines) home page

## Manually Managing AWR

The Oracle-supplied package `DBMS_WORKLOAD_REPOSITORY` provides several routines, procedures, and functions to manually interact with the AWR. These routines allow you to create snapshots on demand, drop a range of snapshots, create a baseline snapshot, drop a baseline snapshot, and—as we discussed in the preceding sections—change snapshot intervals and data retention period, and so on.

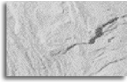
### Modifying Snapshot Settings

The snapshot frequency, or interval, and the data retention can be changed as shown next, where the interval is changed to 30 minutes and data retention to 15 days:

```

begin
    dbms_workload_repository.modify_snapshot_settings (
        interval => 30,
        retention => 15 * 1440 );
end;
/
PL/SQL procedure successfully completed.

```

**NOTE**

*Both the values interval and retention must be specified in terms of minutes. That's why the retention time of 15 days is multiplied by 1440 minutes per day.*

The value specified for *interval* must range from 10 minutes to 52560000 minutes (100 years). Oracle will generate an error (ORA-13511) if the value is outside this range. However, the value zero has a special meaning: when the interval is set to zero, Oracle will disable the mechanism of taking automatic and manual snapshots. In this case, Oracle simply generates a very large number (40150 days) for this parameter.

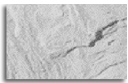
Similarly, the *retention* parameter value must range from 1400 minutes (1 day) to 52560000 minutes (100 years). Oracle will generate an error (ORA-13510) if the value is outside this range. The value zero has a special meaning here, also. When the retention is set to zero, Oracle will retain the snapshot forever. In this case, Oracle uses a larger number (40150 days) for this parameter.

You can see the current values of these parameters as shown next. The column data is displayed in the days and hours format (+DDDDD HH:MI:SS.S):



```
col snap_interval for a20
col retention for a20
select snap_interval,
       retention
from   dba_hist_wr_control;
```

```
SNAP_INTERVAL          RETENTION
-----
+00000 00:30:00.0      +00015 00:00:00.0
```

**NOTE**

*The AWR tables reside in the SYSAUX tablespace. AWR purges snapshot data daily once the retention time has been reached. If AWR detects that the SYSAUX tablespace is running short on available space, it will perform an “emergency purge,” effectively reducing the retention and will write relevant messages to both the trace file and the alert log. However, the space layer code will initiate a server-generated alert because of low space, probably long before an emergency purge is required.*

## Creating and Dropping Snapshots

You may need to create manual snapshots when the automatic snapshot interval is too large for diagnosing performance issues with a particularly smaller workload or job. In this case, you may want to take manual snapshots before running the job and take another one after it completes.

You can use the `CREATE_SNAPSHOT` routine to create snapshots on demand. This routine can be executed as a procedure or as a function. When it is executed as a function, it returns the `SNAP_ID` of the snapshot just created. The optional parameter *flush\_level* controls the level of the statistics that is captured in, or flushed to, the repository tables for this snapshot. The default is `TYPICAL`. But if all of the detailed statistics are required, you must set the *flush\_level* to `ALL`.

The following examples show the use of `CREATE_SNAPSHOT` as a procedure and a function:

```

REM - As a Procedure
begin
    dbms_workload_repository.create_snapshot();
end;
/
PL/SQL procedure successfully completed.

REM - As a Function
select dbms_workload_repository.create_snapshot('ALL') from dual;

DBMS_WORKLOAD_REPOSITORY.CREATE_SNAPSHOT('ALL')
-----
                                           1931

```

On the other hand, the `DROP_SNAPSHOT_RANGE` procedure can be used to permanently drop a range of snapshots. You must know the beginning and ending `SNAP_ID` for this range, though. The procedure has two required parameters, *low\_snap\_id* and *high\_snap\_id*. The optional parameter, *dbid*, defaults to the local database ID. In the following example, the `SNAP_ID` range from 1981 to 2004 is dropped:

```

begin
    dbms_workload_repository.drop_snapshot_range (
        low_snap_id => 1981,
        high_snap_id => 2004 );
end;
/
PL/SQL procedure successfully completed.

```

As shown next, the DBA\_HIST\_SNAPSHOT view will list the information on available snapshots. In this example, we selected only the columns needed for the procedure just discussed:

```
col snap_id for 999999
col begin_interval_time for a26
col end_interval_time for a26
select snap_id, dbid, begin_interval_time, end_interval_time
from dba_hist_snapshot
order by 1 desc;
```

SNAP_ID	DBID	BEGIN_INTERVAL_TIME	END_INTERVAL_TIME
2397	2847681843	29-MAR-04 06.00.47.194 PM	29-MAR-04 06.30.33.990 PM
2396	2847681843	29-MAR-04 05.30.58.107 PM	29-MAR-04 06.00.47.194 PM
2395	2847681843	29-MAR-04 05.00.07.715 PM	29-MAR-04 05.30.58.107 PM
2394	2847681843	29-MAR-04 04.30.21.575 PM	29-MAR-04 05.00.07.715 PM
2393	2847681843	29-MAR-04 04.00.32.191 PM	29-MAR-04 04.30.21.575 PM
2392	2847681843	29-MAR-04 03.30.43.423 PM	29-MAR-04 04.00.32.191 PM
2391	2847681843	29-MAR-04 03.00.57.354 PM	29-MAR-04 03.30.43.423 PM
2390	2847681843	29-MAR-04 02.30.06.223 PM	29-MAR-04 03.00.57.354 PM
2389	2847681843	29-MAR-04 02.00.17.503 PM	29-MAR-04 02.30.06.223 PM

### Creating and Dropping Baseline (Preserved Snapshot Set)

The baseline, or the Preserved Snapshot Set, can be created using the CREATE\_BASELINE routine in the DBMS\_WORKLOAD\_REPOSITORY package. The routine can be executed as a function or a procedure. The required parameters are the *start\_snap\_id*, *end\_snap\_id*, and *baseline\_name*. The optional parameter, *dbid*, defaults to the local database ID. When executed as a function, this routine returns the baseline ID or the Preserved Snapshot Set ID. The following examples show the use of CREATE\_BASELINE routine as a function and a procedure:

```
REM - As a Procedure
begin
    dbms_workload_repository.create_baseline(
        start_snap_id => 2305,
        end_snap_id => 2310,
        baseline_name => 'Good Nightly Batch');
end;
/
PL/SQL procedure successfully completed.

REM - As a Function
REM - Using 2200 for start_snap_id, 2205 for the end_snap_id and
```

```

REM - 'Good Special Batch' for the baseline_name.

select dbms_workload_repository.create_baseline(
                                2200, 2205, 'Good Special Batch')
from dual;

DBMS_WORKLOAD_REPOSITORY.CREATE_BASELINE(2200,2205,'GOODSPECIALBATCH')
-----
3

```

Oracle assigns a unique baseline ID to the newly created baseline. The snapshots that fall in the range of the start and end snap IDs used for the baseline will be preserved as long as the as baseline exists in the AWR. Those snapshots can only be purged when the baseline is dropped.

Oracle generates an error (ORA-13506) when the supplied snapshots IDs are not valid. Also, the supplied baseline name must be unique, or an error (ORA-13528) will be reported.

The procedure `DROP_BASELINE` drops the specified baseline. You must supply the *baseline\_name*. The procedure has an option to either retain the associated snapshots or drop them. When *cascade* is set to *true*, all the snapshot IDs for the baseline are dropped. The default value for this parameter is *false*, meaning associated snapshot IDs will be preserved and only the baseline will be dropped. The optional parameter, *dbid*, defaults to the local database ID. In the following example, the baseline titled 'Routine Jobs' is dropped along with its associated snapshot IDs:

```

begin
    dbms_workload_repository.drop_baseline('Routine Jobs', true);
end;
/
PL/SQL procedure successfully completed.

```

## AWR Reports

AWR contains a couple of SQL scripts to produce Workload Repository reports that resemble the Statspack report. However, the Workload Repository report contains much more information than the Statspack report did, since it reports all the new database statistics we discussed in the preceding sections; for example, the Time Model Statistics and Operating System Statistics. In addition, the report contains statistics at Service Name and Module Name levels.

The Oracle-supplied scripts *awrrpt.sql* and *awrrpti.sql* generate reports either in HTML or text format for the specified range of snapshot IDs. The output report from these scripts is essentially the same, but the latter script allows you to specify a particular instance in a multi-instance database environment. You need DBA privilege to run these scripts. These scripts are in the `$ORACLE_HOME/rdbms/admin` directory.

## AWR Views

You can view the AWR statistics using various views. Table 9-2 lists a few of these views and briefly describes their contents.

In addition to storing performance statistics data in the AWR tables in SYSAUX tablespace, Oracle Database 10g also takes frequent snapshots of the active sessions and stores that information for historical analysis. This mechanism is called Active Session History.

## Active Session History

In Oracle Database 10g, the current session activity can be obtained from the V\$SESSION or V\$SESSION\_WAIT views. The view V\$SESSION\_WAIT\_HISTORY provides information about the last 10 wait events that the session encountered. However, for troubleshooting a performance issue, these views do not provide enough historical information about what the session did. Oracle Database 10g solves this problem with the introduction of Active Session History (ASH).

As the name suggests, Oracle provides historical information on active sessions. It samples all active sessions every second to track their state and stores this information in a buffer in the SGA.

---

View Name	Description
DBA_HIST_ACTIVE_SESS_HISTORY	Displays history of the contents of the V\$ACTIVE_SESSION_HISTORY view.
DBA_HIST_BASELINE	Displays information on the baselines stored in AWR.
DBA_HIST_DATABASE_INSTANCE	Displays database and instance information.
DBA_HIST_SNAPSHOT	Displays information on AWR snapshots.
DBA_HIST_SQL_PLAN	Displays information on SQL execution plans.
DBA_HIST_WR_CONTROL	Displays parameter settings that control AWR properties.

---

**TABLE 9-2.** *AWR Views (Not a Complete List)*

The historical information is displayed by the V\$ACTIVE\_SESSION\_HISTORY view. This view is similar to a join between V\$SESSION and V\$SESSION\_WAIT views with historical data for active sessions.

The other important difference between the V\$ACTIVE\_SESSION\_HISTORY and V\$SESSION\_WAIT\_HISTORY is the archiving of the data. ASH contents are written to the database tables by AWR. This view can roughly be called the “flashback session” view as it helps perform spot analysis of the session when diagnosing problems that may have occurred in the immediate past. However, this information is not guaranteed to be available in this view at all times. In a very active database with numerous active sessions, the internal buffer can get full faster. Oracle will sample available information and write to an AWR table that can be viewed via the DBA\_HIST\_ACTIVE\_SESSION\_HISTORY view. The view V\$ACTIVE\_SESSION\_HISTORY acts as a single point of reference for various pieces of information such as waits events, accessed objects, time spent on CPU, details about the SQL statement such hash value, and the SQL execution plan.

## What Is an Active Session?

We mentioned in the previous section that ASH samples active sessions. But what is an active session? The status ACTIVE is not to be confused with V\$SESSION.STATE, which has a binary value of ACTIVE or INACTIVE. ASH considers a session ACTIVE if the user call to the RDBMS kernel falls in any of the following categories and collects the session activity data:

- PARSE or EXECUTE or FETCH operations
- Waiting for the I/O to complete
- Waiting for the message or buffer from remote instance (in RAC)
- On CPU
- Not waiting for recursive session
- If it is a parallel slave, not waiting for PX\_IDLE event
- Any other wait that does not fall in the *Idle* wait class

## Components of ASH

ASH does not require any initialization parameter setting or any installation script be run. It is enabled by default after creating or upgrading to an Oracle Database 10g.

The new background process, MMNL (the lightweight version of MMON), is responsible for writing the sampled data to the in-memory circular buffer in the fixed

area of the SGA. The buffer contents are further sampled and written to the AWR table, `WRH$_ACTIVE_SESSION_HISTORY`, on every AWR flush every hour by default. The MMNL process will also flush the buffer contents to the AWR tables whenever the buffer gets full. This process does not request any latches to update the buffer contents and can keep up with database activity without any problems.

The ASH in-memory buffer, by default, has an upper limit of 30MB for its size. The minimum size is 1MB. The size of the ASH in-memory buffer depends on factors such as the number of CPUs, the size of shared pool, the value set for `SGA_TARGET`, and some arbitrary rounding of numbers. In Oracle Database 10g Release 1, the following formula derives the buffer size; however, it may change in the future releases:

```
max (min (#of CPUs * 2MB, 5% of SHARED_POOL_SIZE, 30MB), 1MB)
```

The view `V$ACTIVE_SESSION_HISTORY` is based on the `X$KEWASH` and `X$ASH` structures. The `X$ASH` structure contains the sampled details of every active session. The `X$KEWASH` structure contains the details about the number of samples taken in the instance.

There are a few hidden initialization parameters that change the default behavior of ASH. Do not rush to start using those. Always get approval from Oracle Support before using such parameters.

The dynamic parameter, `_ASH_ENABLE`, when set to `FALSE` will disable ASH functionality, and the view `V$ACTIVE_SESSION_HISTORY` will not be populated anymore. The `_ASH_DISK_WRITE_ENABLE` defaults to `TRUE` to flush the in-memory ASH data to disk. Setting it to `FALSE` will disable writing this data to disk. So, if for some reason you do not want to store ASH data to AWR but want to keep it in the memory, you can set this parameter to `FALSE`. You can also increase the buffer size by setting the `_ASH_SIZE` parameter to a larger value than 30MB.

## **V\$ACTIVE\_SESSION\_HISTORY View**

Table 9-3 describes the columns in the `V$ACTIVE_SESSION_HISTORY` and where appropriate relates those to columns already available in other `V$` views.

The `V$ACTIVE_SESSION_HISTORY` view can be considered a fact table in a data warehouse with its columns as the dimensions of the fact table. The contents are in the memory; so accessing those by these columns is very fast. You can find out almost anything about any sessions' activity from this view. You can quickly answer questions such as: how many sessions waited on a particular wait event in the last five minutes and for how long? What objects sessions are waiting on the most and what for? It is important to note that this information is based on sampled data that is captured every second. As such, it will be very close to being accurate and sufficient for your analysis.

---

Column Name	Type	Description
SAMPLE_ID	NUMBER	ID of the sample snapshot.
SAMPLE_TIME	TIMESTAMP(3)	Time at which the sample was taken.
SESSION_ID	NUMBER	Session identifier, maps to V\$SESSION.SID.
SESSION_SERIAL#	NUMBER	Session serial number, maps to V\$SESSION.SERIAL#.
USER_ID	NUMBER	Oracle user identifier, maps to V\$SESSION.USER#.
SQL_ID	VARCHAR2(13)	SQL identifier of the SQL statement.
SQL_CHILD_NUMBER	NUMBER	Child number of the SQL statement.
SQL_PLAN_HASH_VALUE	NUMBER	Hash value of the SQL plan, maps to V\$SQL.PLAN_HASH_VALUE.
SQL_OPCODE	NUMBER	SQL operation code, maps to V\$SESSION.COMMAND.
SERVICE_HASH	NUMBER	Service hash, maps to V\$ACTIVE_SERVICES.NAME_HASH.
SESSION_TYPE	VARCHAR2(10)	BACKGROUND or FOREGROUND.
SESSION_STATE	VARCHAR2(7)	State: WAITING or ON CPU.
QC_SESSION_ID	NUMBER	Query coordinator ID for parallel query.
QC_INSTANCE_ID	NUMBER	Query coordinator instance ID.
EVENT	VARCHAR2(64)	If SESSION_STATE = WAITING, the event for which the session was waiting for at the time of sampling. If SESSION_STATE = ON CPU, the event for which the session last waited upon before being sampled.
EVENT_ID	NUMBER	Identifier of the resource or event, maps to V\$EVENT_NAME.EVENT_ID.
EVENT#	NUMBER	Number of the resource, maps to V\$EVENT_NAME.EVENT#.
SEQ#	NUMBER	Sequence number, uniquely identifies the wait, maps to V\$SESSION.SEQ#.

---

**TABLE 9-3.** V\$ACTIVE\_SESSION\_HISTORY View

---

Column Name	Type	Description
P1	NUMBER	First additional wait parameter.
P2	NUMBER	Second additional wait parameter.
P3	NUMBER	Third additional wait parameter.
WAIT_TIME	NUMBER	It is 0 if the session was waiting, maps to V\$SESSION.WAIT_TIME.
TIME_WAITED	NUMBER	If SESSION_STATE = WAITING, the time that the session actually spent waiting for that EVENT will be 0 until it finishes waiting.
CURRENT_OBJ#	NUMBER	Object ID of the object if the session is waiting for some I/O-related events or for some enqueue waits, maps to V\$SESSION.ROW_WAIT_OBJ#.
CURRENT_FILE#	NUMBER	File number of the file if the session was waiting for some I/O-related events or for some enqueue waits, maps to V\$SESSION.ROW_WAIT_FILE#.
CURRENT_BLOCK#	NUMBER	ID of the block if the session was waiting for I/O-related events or for some enqueue waits, maps to V\$SESSION.ROW_WAIT_BLOCK#.
PROGRAM	VARCHAR2(48)	Name of the operating system program, maps to V\$SESSION.PROGRAM.
MODULE	VARCHAR2(48)	Name of the executing module when sampled, as set by the procedure DBMS_APPLICATION_INFO.SET_MODULE.
ACTION	VARCHAR2(32)	Name of the executing module when sampled, as set by the procedure DBMS_APPLICATION_INFO.SET_ACTION.
CLIENT_ID	VARCHAR2(64)	Client identifier of the session; maps to V\$SESSION.CLIENT_ID.

---

**TABLE 9-3.** V\$ACTIVE\_SESSION\_HISTORY View (continued)

Such online analysis is possible because of the ASH in-memory buffer. The following example shows how to find what sessions waited in the last five minutes, for what wait events, for how long, and how many times they waited:

```

select session_id, event, count(*), sum(time_waited)
from v$active_session_history
where session_state = 'WAITING'
and time_waited > 0
and sample_time >= (sysdate - &HowLongAgo/(24*60))
group by session_id, event;

```

Enter value for howlongago: 5

old 5: and sample\_time >= (sysdate - &HowLongAgo/(24\*60))

new 5: and sample\_time >= (sysdate - 5/(24\*60))

SESSION_ID	EVENT	COUNT(*)	SUM(TIME_WAITED)
126	db file scattered read	276	16958032
131	db file scattered read	270	17728709
131	log file switch completion	1	418071
133	class slave wait	1	5125049
133	db file sequential read	4	151610
138	db file scattered read	5	354926
138	db file sequential read	20	974258
138	log file switch completion	1	418261
138	control file sequential read	1	27706
153	null event	1	45580
153	db file sequential read	26	6900220
153	control file sequential read	4	202271
166	control file parallel write	8	1896634
166	control file sequential read	1	55883
167	log file parallel write	8	359185
167	control file single write	1	30063
168	db file parallel write	9	362689

17 rows selected.

When querying the V\$ACTIVE\_SESSION\_HISTORY view, Oracle has to acquire all the usual latches for statement parsing, accessing the buffers, etc. If there is a parse latch related problem in a hung database, you may not be able to access the wealth of information in the V\$ACTIVE\_SESSION\_HISTORY view that can help you diagnose the problem. However, there is another way to access this in-memory ASH information, and that is what we will discuss in the next section.

## The ASHDUMP: Dumping ASH Circular Buffer Contents to Trace File

The contents of the ASH buffer can be dumped to a trace file using the event ASHDUMP. These contents can then be loaded into a database table. The structure of this table resembles the V\$ACTIVE\_SESSION\_HISTORY view.

You can produce the ASHDUMP trace file using the following *oradebug* command sequence after connecting as sysdba:

```
oradebug setmypid
oradebug unlimit
oradebug dump ashdump 10
oradebug tracefile_name
```

You can also use the ALTER SESSION command to produce an immediate dump of the ASH buffer as shown next:

```
alter session set events 'immediate trace name ashdump, level 10';
```

The trace file will be in your UDUMP directory. The contents of the trace file can be loaded into a table in other database. In the first few lines, the trace file lists all the column names for the data. The trace data is displayed as comma-separated values for the respective columns. This information can be used to employ SQL\*Loader to load the rest of the contents of the trace file to the table for further analysis.

The following example shows the contents of the trace file from an ASHDUMP. The header information, typically found in trace files, is removed for clarity.

```
<<<ACTIVE SESSION HISTORY - PROCESS TRACE DUMP HEADER BEGIN>>>
DBID, INSTANCE_NUMBER, SAMPLE_ID, SAMPLE_TIME, SESSION_ID, SESSION_SERIAL#,
USER_ID, SQL_ID, SQL_CHILD_NUMBER, SQL_PLAN_HASH_VALUE, SERVICE_HASH,
SESSION_TYPE, SQL_OPCODE, QC_SESSION_ID, QC_INSTANCE_ID, CURRENT_OBJ#,
CURRENT_FILE#, CURRENT_BLOCK#, EVENT_ID, SEQ#, P1, P2, P3, WAIT_TIME, TIME_
WAITED, PROGRAM, MODULE, ACTION, CLIENT_ID
<<<ACTIVE SESSION HISTORY - PROCESS TRACE DUMP HEADER END>>>

<<<ACTIVE SESSION HISTORY - PROCESS TRACE DUMP BEGIN>>>
2847681843,1,6033130,"04-13-2004
07:40:58.006572000",41,597,5,"2xbhdwsp8a0zd",0,0,3427055676,1,62,0,0,16314,
1,27521,2652584166,135,1,27709,1,121,0,"sqlplus@hptest (TNS V1-
V3)","SQL*Plus","",""
2847681843,1,6033129,"04-13-2004
07:40:56.976572000",41,597,5,"2xbhdwsp8a0zd",0,0,3427055676,1,62,0,0,16314,
1,27521,2652584166,135,1,27709,1,121,0,"sqlplus@hptest (TNS V1-
V3)","SQL*Plus","",""
. . . . .
. . . . .
```

```

2847681843,1,6032838,"04-13-2004
07:35:57.196572000",49,3,0,"6q766vsk5290x",0,0,165959219,2,47,0,0,429496729
5,0,0,866018717,189,300,0,0,3007426,0,"oracle@hptest (MMON)", "", "", ""
2847681843,1,6032837,"04-13-2004
07:35:56.166572000",49,3,0,"6q766vsk5290x",0,0,165959219,2,47,0,0,429496729
5,0,0,866018717,189,300,0,0,3007426,0,"oracle@hptest (MMON)", "", "", ""
<<<ACTIVE SESSION HISTORY - PROCESS TRACE DUMP END>>>

```

Although this process of dumping the ASH buffer, loading the trace file data into a table, and then troubleshooting the cause of the problem may sound a bit cumbersome, it may occasionally be useful when you have a hung system. You don't need to wait for the disaster to strike. You can experiment by producing the ASHDUMP trace file using the ALTER SESSION command and keeping those scripts ready to load the data to your own ASH table!

In the preceding sections we discussed how Oracle Database 10g captures performance data using AWR and ASH snapshots, how to manage the data capture process, and how you can view the data. Behind the scenes, Oracle Database 10g is doing a lot more with this data, and that is the topic of the next section.

## Automatic Database Diagnostic Monitor (ADDM)

Automatic Database Diagnostic Monitor (ADDM, pronounced "Adam") is not just another piece of software that you purchase from Oracle Corporation and install on your database server and workstation.

ADDM is a holistic self-diagnostic mechanism built into the Oracle Database 10g. It is an integral part of the kernel. It automatically examines and analyzes the snapshot data captured into AWR to proactively determine any major issues with the system, and in many cases it recommends corrective actions with quantified expected benefits. It is constantly monitoring and diagnosing your system.

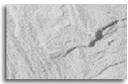
The goal of ADDM is to identify the areas of the system that are consuming the most *DB time*—time spent in the database calls. It uses the wait model and time model statistics to find where time is being spent in the database. It drills down to the root cause of the problem using a tree-structured set of rules. These rules have a proven track record and have been used successfully over the past several years by Oracle Corporation in performance tuning engagements.

ADDM can detect and report many problems including the following types of problems:

- CPU bottlenecks due to Oracle and non-Oracle applications
- I/O subsystem capacity issues

- High-load SQL statements that may be consuming excessive system resources
- High-load PL/SQL compilation and execution consuming excessive system resources
- High-load Java applications consuming excessive system resources
- Undersized memory structures, such as SGA, buffer cache, and log buffer
- Poor connection management
- RAC-related issues with global cache management and interconnect latency
- Concurrent data access issues resulting in buffer busy waits
- Database configuration issues such as log file sizing, archiving or suboptimal parameter settings

In addition to reporting the problematic areas of the system, ADDM also reports the areas it treats as nonproblematic, so you don't spend time analyzing items that don't impact overall system performance. ADDM also recommends possible solutions to the common problems it detects. Again, ADDM recommended solutions are targeted towards achieving lower *DB time*.



**NOTE**

*Because the recommendations from ADDM are generated by a set of predefined rules, those may or may not be applicable to all the situations. You may find some of these recommendations unsuitable to your environments. According to Oracle, ADDM does not target the tuning of individual user response times. Use tracing techniques to tune for individual user response times.*

## ADDM Setup

Although the Automatic Database Diagnostic Monitoring is enabled by default, there are a couple of parameters that you need to be aware of.

The initialization parameter `STATISTICS_LEVEL` must be set either to `TYPICAL` or `ALL` to enable ADDM functionality. It defaults to `TYPICAL`; setting it to `BASIC` will disable ADDM and many other features.

The other parameter is not an initialization parameter. It is a special ADDM-related task parameter, `DBIO_EXPECTED`, which ADDM uses to analyze the performance of the I/O subsystem. The value for the `DBIO_EXPECTED` parameter defines the average time it takes to read a single database block in microseconds. The default value for this parameter is 10 milliseconds (10,000 microseconds). If you think your I/O subsystem response time is significantly different, you may want to change this default value.

First, you must find out the average read time for random reads of a single database block for your hardware. Convert that to microseconds. For this example, let's say it comes out to be 30,000 microseconds (somewhat slow disks).

Second, use the following procedure to set the `DBIO_EXPECTED` value to 30,000:

```

REM - Run this as SYS user
begin
  dbms_advisor.set_default_task_parameter (
    'ADDM', 'DBIO_EXPECTED', 30000);
end;
/
PL/SQL procedure successfully completed.

```

The value for the `DBIO_EXPECTED` parameter is saved in an internal table. You need to execute the preceding procedure to change it. The following SQL script shows how you can interrogate the current value of this parameter:

```

col parameter_name for a20
col parameter_value for a20
select advisor_name,
       parameter_name,
       parameter_value,
from   dba_advisor_def_parameters
where  parameter_name like 'DBIO%';

```

ADVISOR_NAME	PARAMETER_NAME	PARAMETER_VALUE
ADDM	DBIO_EXPECTED	30000

## Using EM to Access ADDM

The Oracle EM Database Control is the primary interface to the ADDM. In the following series of steps you can see the results of ADDM analysis.

Figure 9-4 and Figure 9-5 show the top and bottom portion of the Database home page. Information pertaining to the database instance, host CPU usage, active session,

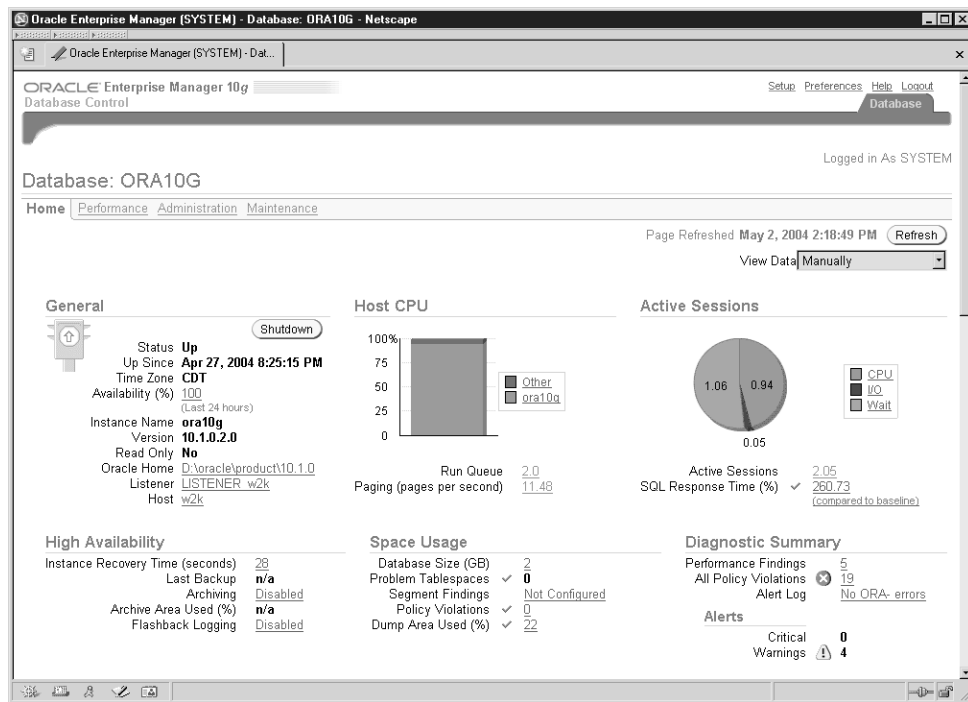


FIGURE 9-4. Database home page top portion

space usage, diagnostic summary, alerts, and performance analysis is shown on the Database home page. It also contains links to access other components that perform several other tasks.

In Figure 9-4, the number of ADDM performance findings for the last ADDM analysis period is shown under Diagnostic Summary toward the bottom right corner.

In Figure 9-5, those performance findings are listed under the Performance Analysis. For each finding ADDM also provided a few recommendations that are shown on the right.

The screenshot displays the Oracle Enterprise Manager (OEM) Database home page. At the top, there is a table listing performance findings:

Impact (%)	Finding	Recommendations
100	Host CPU was a bottleneck and the instance was consuming 88% of the host CPU. All wait times will be inflated by wait for CPU.	2 SQL Tuning 1 Host Configuration
94.39	SQL statements consuming significant database time were found.	2 SQL Tuning
48.5	SQL statements were found waiting for row lock waits.	1 Application Analysis
1.26	Individual database segments responsible for significant user I/O wait were found.	1 Segment Tuning

Below the findings table, the 'Performance Analysis' section shows the period start time as May 2, 2004 1:00:19 PM and a duration of 60.72 minutes. The 'Job Activity' section indicates that jobs are scheduled to start no more than 7 days ago, with 0 suspended and 0 problem executions. The 'Critical Patch Advisories' section shows 0 advisories and notes that Oracle MetaLink credentials are not configured. The 'Related Links' section provides links to various database management tools and metrics.

FIGURE 9-5. Database home page bottom portion

The first performance finding states that there was CPU contention. By clicking the link associated with this finding, you can drill down to the details, as shown in Figure 9-6.

For the CPU contention issue, in this case, ADDM wants you to consider adding more CPUs to the server or adding more database instances, preferably on other servers, to service the load. It also identified two SQL statements that may need investigating.

Oracle Enterprise Manager (SYSTEM) - Performance Finding Details - Netscape

Oracle Enterprise Manager (SYSTEM) - Perf...

ORACLE Enterprise Manager 10g Database Control

Database: ORA10G > Advisor Central > Automatic Database Diagnostic Monitor (ADDM) > Performance Finding Details

Performance Finding Details

Database Time (minutes) 125.23 Period Start Time May 2, 2004 1:00:19 PM Period Duration (minutes) 60.72  
 Task Owner SYS Task Name ADDM:3842429532\_1\_156 Average Active Sessions 2.06

Finding Host CPU was a bottleneck and the instance was consuming 88% of the host CPU. All wait times will be inflated by wait for CPU.  
 Impact (minutes) 125.23  
 Impact (%) 100

Recommendations

Select Item(s) and... Run SQL Tuning Advisor

Select All | Select None | Show All Details | Hide All Details

Select Details	Category	Benefit (%)
<input type="checkbox"/>	Hide Host Configuration	100
Action Consider adding more CPUs to the host or increasing the number of instances serving the database.		
<input checked="" type="checkbox"/>	Hide SQL Tuning	22.81
SQL Text UPDATE KIRTI.ZIPCODES A SET A.CITY = (SELECT DISTINCT B.CUST_CITY FROM SH.CUSTOMERS B WHERE A.ZIP = TO_NUMBER(SUBSTR(B.CUST_POSTAL_CODE,1,5)))		
Action Run SQL Tuning Advisor on the SQL statement with SQL_ID "9z9v01n7x6k1z". Run Advisor Now		
<input checked="" type="checkbox"/>	Hide SQL Tuning	13.85
SQL Text UPDATE KIRTI.ZIPCODES A SET A.STATE = (SELECT DISTINCT SUBSTR(B.CUST_STATE PROVINCE,1,2) FROM SH.CUSTOMERS B WHERE A.ZIP = TO_NUMBER(SUBSTR(B.CUST_POSTAL_CODE,1,5)))		
Action Run SQL Tuning Advisor on the SQL statement with SQL_ID "bn121rq097a7". Run Advisor Now		

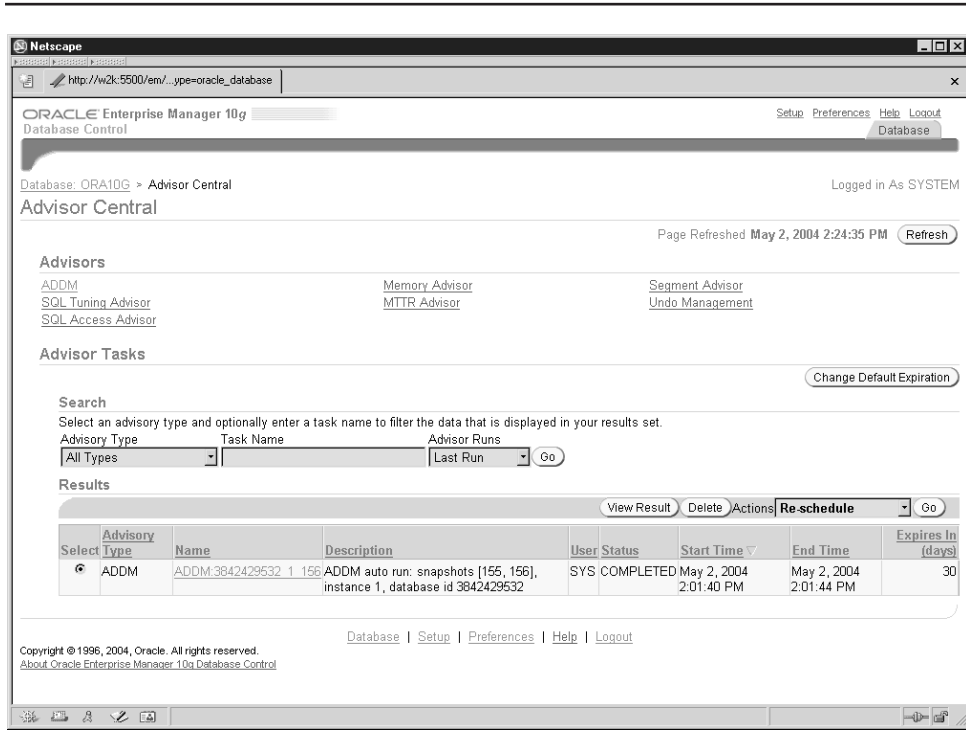
Additional Information

Host CPU

FIGURE 9-6. Performance Finding Details and Recommendations

Back on the Database home page, as shown in Figure 9-5, if you click the Advisor Central link under the heading Related Links at the bottom of the page, you are presented with the Advisor Central home page, as shown in Figure 9-7.

The Advisor Central home page offers a variety of advisories on almost any component of the database, from SQL tuning to undo management. It readily shows the details of the automatic advisory tasks that ADDM performed when the last AWR snapshot was taken. The latest advisor task is shown on this page. You can

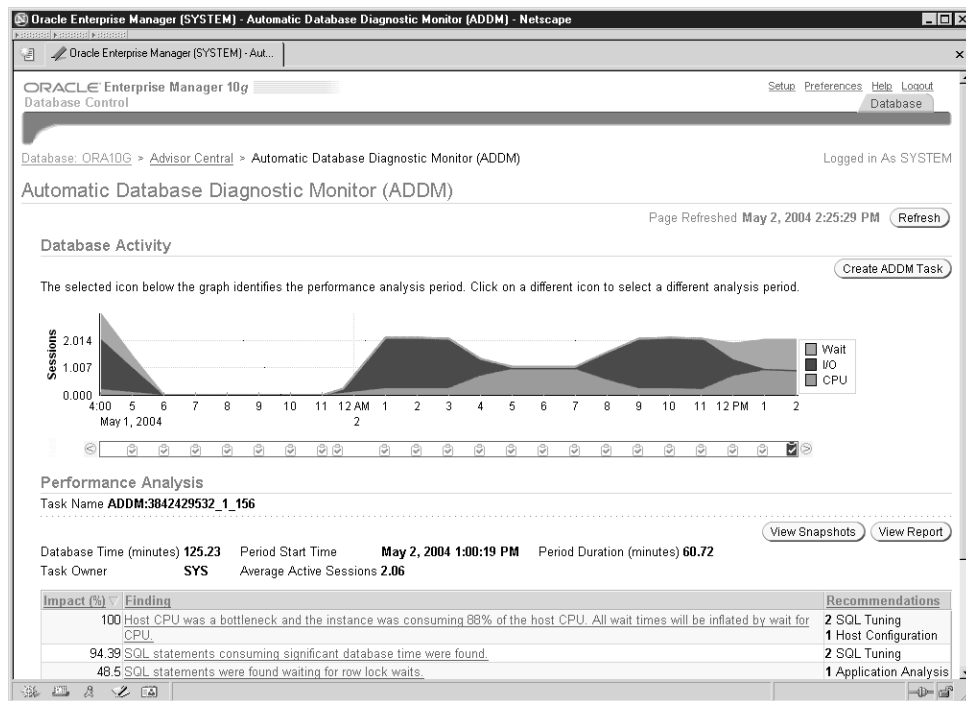


**FIGURE 9-7.** *Advisor Central home page*

review reports of the previous tasks using the pull-down Advisor Runs menu. You can select from the last run, the last 24 hours, or the last 7 days.

Clicking the name of the advisor task takes you to the Automatic Database Diagnostic Monitor (ADDM) page, as shown in Figure 9-8.

This page shows the database activity over the past several hours and additional information on the advisor task. To view the ADDM report for this particular advisor task, click the View Report button. The detailed ADDM report for the selected task will be displayed, as shown in Figure 9-9.

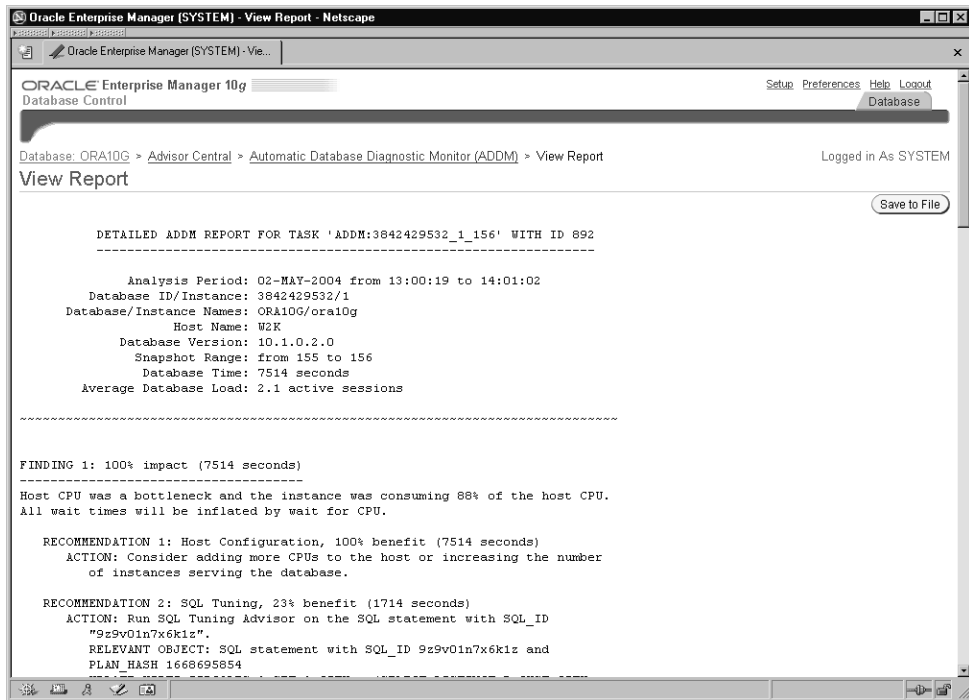


**FIGURE 9-8.** Automatic Database Diagnostic Management

All this information was readily captured and analyzed by Oracle Database 10g. It was easily accessible, and most importantly, ADDM analyzed the system load and offered recommendations before its findings became a real problem.

Even if you do not look at the ADDM findings immediately, you can access them later because they are stored in the database. ADDM data (and all ADDM advisor framework data) is stored for 30 days by default.

However, many a times DBAs will have to perform real-time problem diagnosis. How many times have you received calls from users stating the database is slow?



**FIGURE 9-9.** Viewing the ADDM report

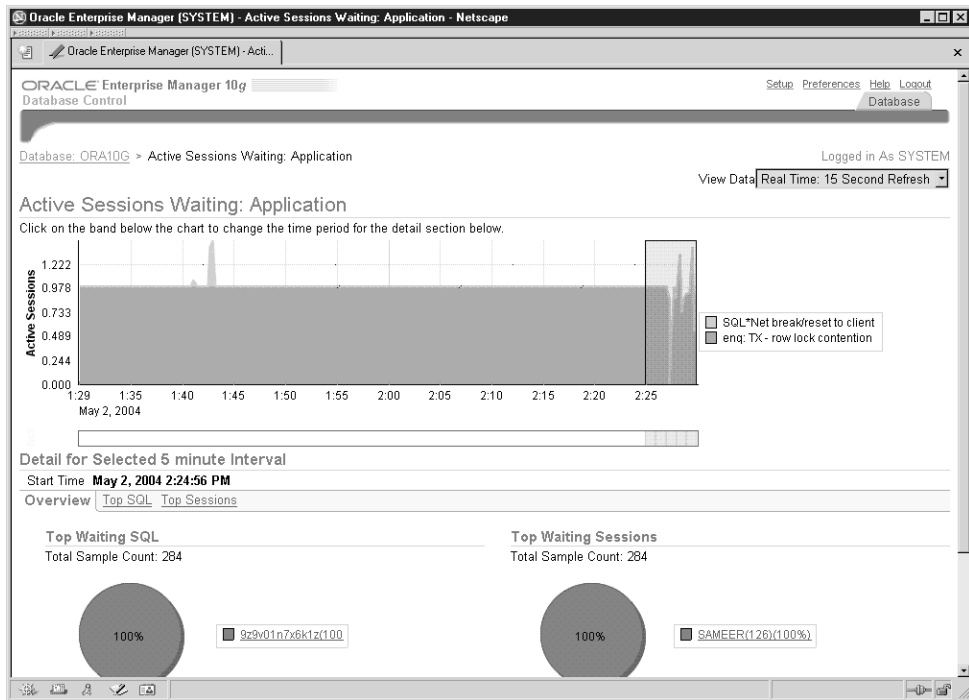
With the Performance page of EM Database Control, you can now easily find out what's going on in your system. From the Database Control home page (Figure 9-4), click the Performance link to access the Performance home page, as shown in Figure 9-10.

On the Performance page, you can see how the CPU and memory resources are being used to make sure those are not the bottlenecks. You can assess the database health from the Sessions: Waiting and Working graph that shows how the CPU is being used by the sessions and if there are any sessions waiting for the resources.



FIGURE 9-10. Performance home page

This graph provides quite a bit of information. It shows the average number of active sessions on the Y axis broken down by the Wait class and CPU. The X axis shows the time. The data is refreshed every 15 seconds by default. The graph uses various colors to indicate different wait classes. The larger the block of color, the worse the problem. Clicking the legend of the color scheme on the right, which is broken down by wait class, will take you to the drill-down page showing the active sessions waiting for that wait class. In the example in Figure 9-10, the Application wait class was the prominent color block. Clicking the Application legend takes you to the



**FIGURE 9-11.** *Active Sessions Waiting: Application (wait class)*

Active Sessions Waiting: Application page, as shown in Figure 9-11. You see that the wait event is *eng: TX row lock contention*, so there is a locking problem in this particular application.

You can drill down by clicking the link under the Top Waiting SQL get the to SQL statement details, as shown in Figure 9-12. There you have it. This is the SQL that is waiting for the lock to be released.

The screenshot shows the Oracle Enterprise Manager interface for a SQL session. The SQL text is as follows:

```

UPDATE kirti.zipcodes a
SET a.city = (SELECT DISTINCT b.cust_city
FROM sh.customers b
WHERE a.zip = to_number(substr(b.cust_postal_code, 1,
5)))

```

The execution plan is displayed below the SQL text. It shows the following operations:

Operation	Object	Object Type	Order	Rows	KB	Cost	Time (seconds)	CPU Cost	IO Cost	Object Node
UPDATE STATEMENT				5		2				
UPDATE				4						
TABLE ACCESS FULL	KIRTI.ZIPCODES	TABLE	1	1	0.029	2	1	7121		2
SORT UNIQUE				3	368	5.75	331	4	51649920	327
TABLE ACCESS FULL	SH.CUSTOMERS	TABLE	2	555	8.672	330	4	40563560		327

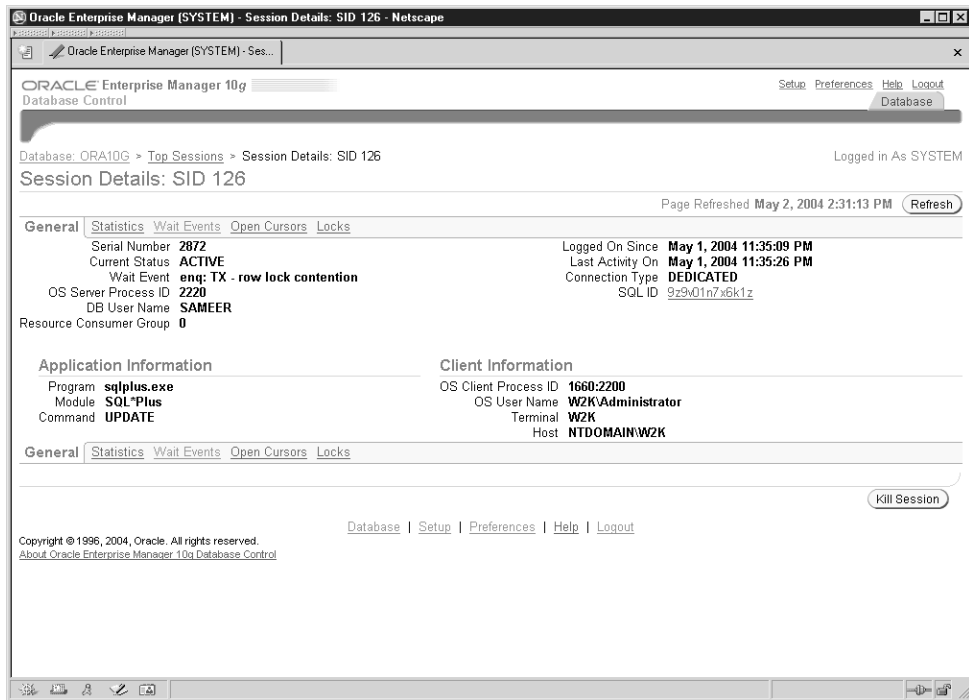
FIGURE 9-12. SQL Details showing SQL text and explain plan

You can also drill down by clicking the link under the Top Waiting Sessions to find the details of the waiting session, as shown in Figure 9-13. The session has been waiting on the *enq: TX row lock contention* wait event.

By clicking the Wait Events link on the Session Details page, you can see the historical wait event the session encountered. In the example shown in Figure 9-14, the session has been waiting on the *enq: TX row lock contention* for quite some time.

## Manually Running ADDM Report

Running the ADDM report from within the Oracle EM is the preferred and the simplest method. However, you can use Oracle-supplied scripts and package procedures to generate the ADDM diagnosis report. You need to know any two AWR snapshots to produce such a report. The snapshots must be available in AWR and there must not be any database restarts between those snapshots.



**FIGURE 9-13.** Session Details showing general information

There are two scripts in the \$ORACLE\_HOME/rdbms/admin directory that can generate the ADDM diagnosis report: *addmrpt.sql* and *addmrpti.sql*. The former generates the report for the local database instance, while the latter can generate the report for other instances (in the RAC environment, for example). In addition, the Oracle-supplied package DBMS\_ADVISOR has procedures (API) to generate the ADDM diagnosis report.

We will briefly discuss how to use these methods to generate the ADDM diagnosis report.

To run the scripts or use the API scripts you must have the ADVISOR privilege.

When running *addmrpt.sql*, you will be prompted to provide the beginning and ending snapshot ID from a list of available snapshots and a report name of your choice. Oracle will generate the ADDM diagnosis report for the specified range of snapshot IDs. The report can also be run in a noninteractive mode. The script has instructions in it to show you how to do that.

Oracle Enterprise Manager (SYSTEM) - Session Details: SID 126 - Netscape

ORACLE Enterprise Manager 10g Database Control

Database: ORA10G > Top Sessions > Session Details: SID 126

Logged in As SYSTEM

View Data Real Time: Manual Refresh

Collected From Target May 2, 2004 2:31:44 PM

Wait Class	Wait Event	P1	P1 Text	P2	P2 Text	P3	P3 Text	Wait Time (Centiseconds)
Application	enq: TX - row lock contention	1415053318	name mode 65536 usn<<16   slot 1798 sequence					308
Application	enq: TX - row lock contention	1415053318	name mode 65536 usn<<16   slot 1798 sequence					310
Application	enq: TX - row lock contention	1415053318	name mode 65536 usn<<16   slot 1798 sequence					310
Application	enq: TX - row lock contention	1415053318	name mode 65536 usn<<16   slot 1798 sequence					307
Application	enq: TX - row lock contention	1415053318	name mode 65536 usn<<16   slot 1798 sequence					311
Application	enq: TX - row lock contention	1415053318	name mode 65536 usn<<16   slot 1798 sequence					309
Application	enq: TX - row lock contention	1415053318	name mode 65536 usn<<16   slot 1798 sequence					307
Application	enq: TX - row lock contention	1415053318	name mode 65536 usn<<16   slot 1798 sequence					307
Application	enq: TX - row lock contention	1415053318	name mode 65536 usn<<16   slot 1798 sequence					308
Application	enq: TX - row lock contention	1415053318	name mode 65536 usn<<16   slot 1798 sequence					309

Copyright © 1996, 2004, Oracle. All rights reserved.  
About Oracle Enterprise Manager 10g Database Control

**FIGURE 9-14.** *Session Details showing session waits*

To generate the ADDM diagnosis report using the DBMS\_ADVISOR package directly needs a bit more setup, as follows:

1. Create an advisor task of ADDM type, using the CREATE\_TASK procedure.
2. Set the START\_SNAPSHOT and END\_SNAPSHOT parameters to run the just-created task using the SET\_TASK\_PARAMETER procedure. The ADDM diagnosis report will be generated for the range of these snapshots.
3. Execute the task using the EXECUTE\_TASK procedure to generate the diagnosis report.
4. View the generated ADDM diagnosis report using GET\_TASK\_REPORT procedure.

---

DBA_ADVISOR_FINDINGS	Displays all the findings discovered by ADDM.
DBA_ADVISOR_LOG	Displays current state of all tasks in the database such as task progress, error messages, and execution times. There is one row for each task.
DBA_ADVISOR_RATIONALE	Displays the rationales for all recommendations.
DBA_ADVISOR_RECOMMENDATIONS	Displays the results of all completed diagnostic tasks with recommendations for the detected problems. The recommendations are ranked in the RANK column. The BENEFIT column shows the expected benefit after carrying out recommended actions.
DBA_ADVISOR_TASKS	Displays information about all existing tasks in the database.

---

**TABLE 9-4.** *ADDM Views (Not a Complete List)*



**NOTE**

*The Oracle Database Performance Guide 10g Release 1 has an excellent example in Chapter 6 in the section “Running ADDM Using DBMS\_ADVISOR APIs” that demonstrates the use of the ADMS\_ADVISOR package to generate and view the ADDM diagnosis report.*

## ADDM Views

Oracle EM is the preferred interface to view all ADDM-related information. However, the DBMS\_ADVISOR views can be used to view this information. Table 9-4 lists a few of these views and their description.

## In a Nutshell

With rich features such as AWR, ASH, and ADDM, along with the Web-based Oracle Enterprise Manager, Oracle Database 10g revolutionizes database performance monitoring.

The mundane tasks of database monitoring and troubleshooting slow-running jobs are now completely automated in Oracle Database 10g. With all the historical performance data now permanently stored in the Automatic Workload Repository, and with the intelligent mechanism to analyze it, Oracle Database 10g knows more about your database behavior than you do!

With these powerful tools, successful performance monitoring will no longer be restricted to an elite group of consultants and experts.