4 Manageability Infrastructure

CERTIFICATION OBJECTIVES

4.01	Types of Oracle Statistics
4.02	The Automatic Workload Repository (AWR)
4.03	Active Session History
4.04	Server-Generated Alerts
4.05	The Automated Tasks Feature
4.06	The Management Advisory Framework
✓	Two-Minute Drill
Q&A	Self Test

Oracle Corporation has devoted a substantial amount of effort to making Oracle Database 10*g* a truly self-managing database. Self-managing capabilities rely on a manageability infrastructure. This chapter explores several important parts of the new manageability infrastructure.

In this chapter, you'll learn more about the Automatic Workload Repository (AWR), which lies at the foundation of Oracle Database 10*g*'s self-management capabilities. You'll learn how to manage the AWR, by managing the AWR snapshots.

In previous versions of Oracle, you had access to the OEM alert system. Now, in Oracle Database 10*g*, you have an altogether new server-generated alert mechanism to help you manage the database. These server-generated alerts include both *threshold-based* and *event-based* warnings.

The chapter also discusses the new automated tasks feature. You'll also get an introduction to the new set of management advisors that comes with the Oracle Database 10*g*. The advisory framework is a new approach to managing the database. Now you can turn to these advisors for advice on configuring crucial factors like memory, space, and SQL statement tuning.

The chapter contains the following major sections:

- Types of Oracle Statistics
- The Automatic Workload Repository (AWR)
- Active Session History (ASH)
- Server-Generated Alerts
- The Automated Tasks Feature
- The Management Advisory Framework

CERTIFICATION OBJECTIVE 4.01

Types of Oracle Statistics

Oracle DBAs regularly collect several types of performance statistics in order to analyze database bottlenecks and other performance issues. In Oracle Database 10*g*, DBAs now have access to several new types of performance statistics. Besides database statistics at the system and session levels (wait statistics, segment usage statistics, and so on), these statistics include operating system statistics (CPU statistics, disk usage statistics, and memory usage statistics) and network statistics as well.

All performance statistics, regardless of their origin—database, operating system, or network—can be classified into several types according to the basis on which you collect the statistics. Let's briefly look at the various types of performance statistics you can collect in Oracle Database 10*g*.

Cumulative Statistics

Cumulative statistics are the accumulated *total* value of a particular statistic since instance startup. The statistic *total logons*, for example, is a cumulative statistic. Oracle collects cumulative statistics for just about every variable for which it collects statistics. Oracle collects several types of cumulative statistics, including statistics for segments and SQL statements, as well as session- and system-wide statistics. By comparing the *delta values*—the rates of change of the cumulative statistics between a beginning and an ending period—Oracle analyzes database performance during a specific interval of time.

You are familiar with the important dynamic performance views `V$SYSSTAT` and `V$SESSSTAT`. These two views hold many of the important cumulative statistics for the Oracle database. Dynamic performance views are very useful in judging database performance, but unfortunately, when you shut down the database, the data in the dynamic performance views disappears completely! If you wish to track database performance over time, or if you wish to compare the performance impact of database changes, you need to store the performance data in a repository. This is where the Automatic Workload Repository (AWR) comes in.

The AWR takes the difference between the cumulative data between two periods and stores this information in its repository. This difference between two periods is called an *AWR snapshot*. By default, the database will take a performance snapshot every hour, but you can customize this interval.

Database Metrics

Database *metrics* are another important type of Oracle performance statistics. You can define metrics as statistics that measure the *rate of change* in a cumulative performance statistic.

In previous Oracle versions, you needed to compute metrics for various performance statistics by collecting data at various periods, to calculate the rate of change of various statistics. Now, Oracle places precomputed metrics at your fingertips. For example, you may be interested in a metric like the number of transactions per second during peak time. Dynamic performance views hold these metrics, and the AWR can also store them in its repository. All the Oracle management advisors use database metrics for diagnosing performance problems and making tuning recommendations.

You can look at statistics such as the number of user transactions and the number of physical reads in the system as the *base statistics* from which you derive database metrics. Thus, database metrics are *derived statistics*. The background process MMON (Manageability Monitor) updates metric data on a minute-by-minute basis, after collecting the necessary fresh base statistics.

Sample Data

The new Automatic Session History (ASH) feature now automatically collects *session sample data*, which represents a sample of the current state of the active sessions. ASH collects the data in memory, where you can view it with the help of `V$` views. The AWR helps save the ASH data permanently, by collecting it as part of its regular snapshots. I discuss the ASH feature in detail later in this chapter, in the "Automatic Session History" section.

exam watch

MMON is a brand-new Oracle Database 10g background process. This process performs mostly management-related tasks, including issuing database alerts and capturing statistics for recently modified database objects. The MMON process transfers the memory version of AWR statistics to disk on a regular basis (in the form of snapshots). The Manageability Monitor Light (MMNL) process performs lightweight manageability tasks, including computing metrics and capturing session history information for the Automatic Session History (ASH) feature under some circumstances. For example, the MMNL process will flush ASH data to disk if the ASH memory buffer fills up before the one-hour interval that would normally cause MMON to flush it. (Note that although Oracle calls MMNL the Manageability Monitor Light, this process shows up as Manageability Monitor Process 2 when you query the `V$BGPROCESS view`.)

Baseline Data

A good way to evaluate database performance is by comparing database performance statistics from two periods, if you know that the first period reflects "good" performance. The statistics from the period where the database performed well are called *baseline data*. By comparing the current performance with the base period's performance, you can see if the database is faring better or worse. You could also compare individual statistics to see which database component may be the source of your problems.

on the job

The `STATISTICS_LEVEL` initialization parameter is the key determinant of exactly what type of statistics the AWR collects. You can determine the types of statistics collected by using various values for this parameter. If you use `BASIC` as the value for this parameter, the AWR turns off its statistics collection completely. Choosing `TYPICAL` as the value for this parameter will collect what Oracle deems it typically needs for monitoring database behavior. If you set `STATISTICS_LEVEL=ALL`, the AWR collects all available statistics. Oracle recommends that you use the default `STATISTICS_LEVEL =TYPICAL` setting. The `ALL` setting may be too comprehensive most of the time, and if you use the `BASIC` setting, you won't be able to use many of Oracle Database 10g's automatic performance tuning features.

CERTIFICATION OBJECTIVE 4.02

The Automatic Workload Repository (AWR)

The AWR is Oracle Database 10*g*'s brand-new feature for the automatic collection of performance statistics in the database, and it lies at the heart of the new self-tuning capabilities of the Oracle database. The AWR stores its data in the new SYSAUX tablespace and, in fact, is one of the major users of that tablespace. AWR provides performance statistics in two distinct formats:

- A *temporary* in-memory collection of statistics in the SGA, which you can access with the help of the dynamic performance (`V$`) views. You can also view these statistics through the OEM interface.

- A *persistent* type of performance data in the form of regular AWR *snapshots*, which you access either through data dictionary views (DBA_*) or the OEM Database Control. You can use the AWR snapshots for historical comparisons of performance. The new MMON background process performs the transfer of the data from memory to disk.

Oracle DBAs traditionally have needed to maintain special database tables to collect historical performance data. The AWR automatically collects performance statistics for you and maintains historical data for analysis. You can view the data in the snapshots with the help of the V$ views or create reports to examine the data in detail. Various database components and features use the data from these AWR snapshots to monitor and diagnose performance issues. For example, as you saw in Chapter 3, the ADDM relies on these snapshots for the diagnosis of performance problems.

Types of Data Collected by AWR

The AWR facility collects a large number of performance statistics, including the following:

- Base statistics that are also collected in the V$SYSSTAT and V$SESSSTAT views
- New SQL statistics that aid in the identification of resource-intensive SQL statements
- Database object usage statistics that inform you about how the database is currently accessing various objects
- Time statistics, which tell you how much time each database activity is taking
- Wait statistics, which provide information about session waits (in previous versions, you needed to join the V$SESSION view with the V$SESSION_WAIT view to gather information on session waits; now several columns have been added to the V$SESSION view, so you can query the view directly)
- Active Session History (ASH) statistics, which are flushed to the AWR on a regular basis
- Database feature usage statistics that tell you if and how intensively your database is utilizing various features

- The results of various management advisory sessions like the Segment Advisor and the SQL Access Advisor
- Operating system statistics like disk I/O and memory usage within the database

It is important to understand that the AWR isn't a repository where Oracle stores *all* the data for the various performance indicators that it covers. The AWR stores only a *part* of the statistics that it collects in memory. Whenever AWR collects a snapshot, it transfers part of the huge amount of data it holds in memory (SGA) to disk.

AWR Data Handling

AWR automatically generates snapshots to collect performance statistics. A *snapshot* is simply the performance data that is captured at certain point in time. As you recall from the previous chapter, each time the AWR generates a snapshot, the ADDM will analyze the period corresponding to the last two snapshots. You may also create snapshots manually if necessary. By comparing the difference in statistics between snapshots, the AWR knows which SQL statements are contributing significantly to your system load. It will then focus on these SQL statements. AWR stores its data in the SYSAUX tablespace. The space used by AWR depends on the following:

- **Data-retention period** The longer the retention period, the more space used
- **Snapshot interval** The more frequent the snapshots are taken, the more space used
- **Number of active sessions** The higher the number of user sessions, the more data collected by the AWR

By default, the AWR saves the data for a period of seven days, but you can modify this period. Oracle recommends that you retain the AWR data to cover at least one complete workload cycle.

Managing the AWR

Managing the AWR really means managing the regular snapshots that AWR collects from your database. By default, the AWR collects its snapshots every 60 minutes. If you think this isn't an appropriate length of time for your purposes, you can change the default snapshot interval easily by changing the `INTERVAL` parameter.

exam watch

Snapshots provide you values for key performance statistics at a given point in time. By comparing snapshots from different periods, you can compute the rate of change of a performance statistic. Most of the Oracle advisors depend on these AWR snapshots for their recommendations. You identify a given snapshot by its unique snapshot sequence numbers called snap IDs. The default interval for snapshot collection is 60 minutes, and the minimum interval is 10 minutes. You can change the interval between snapshots by adjusting the `INTERVAL` parameter. You can take manual snapshots of the system any time you wish. You can combine manual and automatic snapshots as well.

In order to use the AWR feature well, you need to select a truly representative baseline, which is a pair or range of AWR snapshots. When database performance is slow, you can compare the baseline snapshot statistics with current performance statistics and figure out where the problems lie.

You can manage the AWR snapshots either with the help of the OEM Database Control or with the Oracle-supplied `DBMS_WORKLOAD_REPOSITORY` package. Let's first look at how you can use this package to manage AWR snapshots.

Using the DBMS_WORKLOAD_REPOSITORY Package to Manage AWR Snapshots

You can use the `DBMS_WORKLOAD_REPOSITORY` package to create, drop, and modify snapshots, as well as to create and drop snapshot baselines.

The AWR automatically generates snapshots, but you can create a snapshot manually, if you want to collect snapshots in between those scheduled by the AWR. You can do this by using the `CREATE_SNAPSHOT` procedure, as follows.

```
begin
 dbms_workload_repository.create_snapshot ();
end;
```

In order to drop a range of snapshots, use the `DROP_SNAPSHOT` procedure. When you drop a set of snapshots, Oracle automatically purges the AWR data that is part of this snapshot range. The following example drops all snapshots whose snap IDs fall in the range of 40 to 60.

```
begin
  dbms_workload_repository.drop_snapshot_range (low_snap_id => 40,
    high_snap_id => 60, dbid => 2210828132);
end;
```

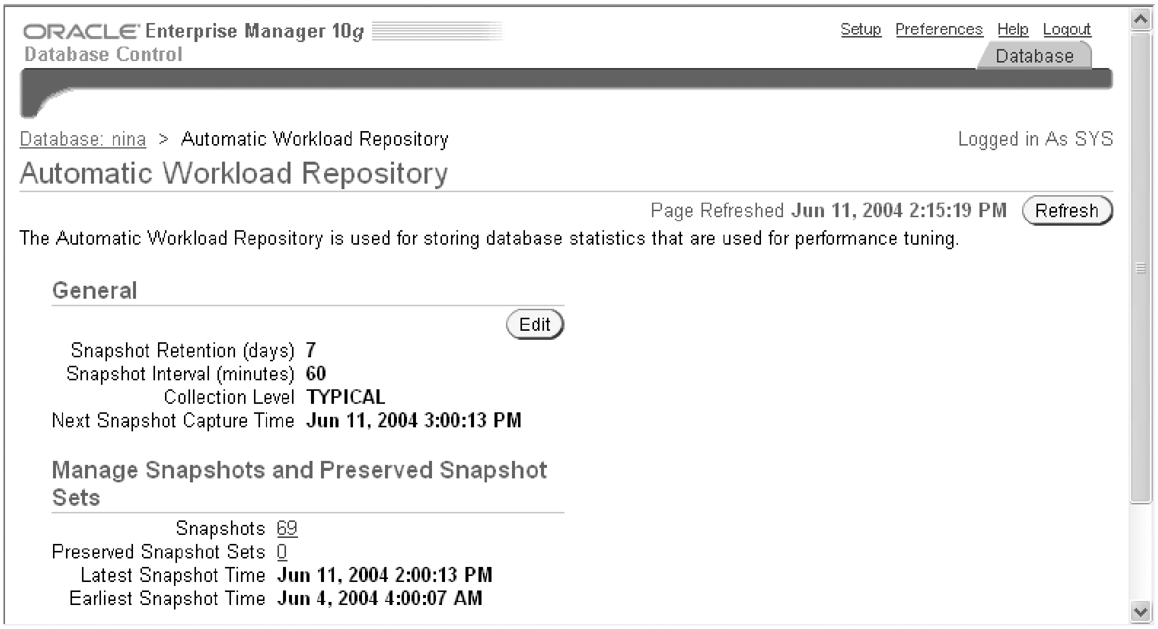
on the job

If you set the snapshot interval to 0, the AWR will stop collecting snapshot data.

Using the Database Control to Manage AWR Snapshots

You can manage AWR snapshots using the AWR page of the OEM Database Control, shown in Figure 4-1. To access this page, from the Database Control home page, click the Administration link, and go the Workload group. Then click the Automatic Workload Repository link. This page has two main sections: the General section and the Manage Snapshots and Preserved Snapshot Sets section.

FIGURE 4-1 The main AWR page



Database: nina > Automatic Workload Repository

Logged in As SYS

Automatic Workload Repository

Page Refreshed Jun 11, 2004 2:15:19 PM

The Automatic Workload Repository is used for storing database statistics that are used for performance tuning.

General

Snapshot Retention (days)	7
Snapshot Interval (minutes)	60
Collection Level	TYPICAL
Next Snapshot Capture Time	Jun 11, 2004 3:00:13 PM

Manage Snapshots and Preserved Snapshot Sets

Snapshots	69
Preserved Snapshot Sets	0
Latest Snapshot Time	Jun 11, 2004 2:00:13 PM
Earliest Snapshot Time	Jun 4, 2004 4:00:07 AM

If you want to change the general settings of the AWR, you can do so by clicking the Edit button in the General section. This will take you to the Edit Settings page, where you can modify the following:

- Snapshot retention intervals
- Snapshot collection intervals
- Snapshot collection levels (Typical or All)

Click the Manage Snapshots and Preserved Snapshot Sets button to get to the Manage Snapshots page. The Manage Snapshots page lists all the snapshots in the AWR. You can click an individual snapshot to view complete details about it, including the capture time and the collection level. If you have established a baseline, you'll also see whether a particular snapshot falls within that baseline. From the Manage Snapshots page, you can do the following:

- Create a snapshot spontaneously (use the Create button)
- View a list of snapshots collected over a specific period
- Establish a range of snapshots to use as a baseline (use the Create Preserved Snapshot Set button)
- Delete a defined range of snapshots from the list of snapshots collected over a period of time (use the Delete Snapshot Range button)

on the job

The range of snapshots you use for a baseline is the same as a preserved snapshot set.

Creating and Deleting AWR Snapshot Baselines

The purpose of using *snapshot baselines* is to have a valid measuring stick for acceptable database performance, as well as to have a reference point for various system statistics. When you say database performance is bad, you must know that it's bad compared to something you clearly know to be good performance.

You define a snapshot baseline on a pair of snapshots, when you know that the period covered by the snapshots represents typical "good" database performance. The baselines will then serve as a valid representative sample to compare with current system database performance. Whenever you create a baseline by defining it over any two snapshots (identified by their snap IDs), the AWR retains the snapshots indefinitely (it won't purge these snapshots after the default period of seven days), unless you decide to drop the baseline itself.

You can create a new snapshot baseline by using the CREATE_BASELINE procedure of the DBMS_WORKLOAD_REPOSITORY package. The snap ID uniquely identifies each snapshot time sequentially. You can get the snap IDs you need to create baselines from the DBA_HIST_SNAPSHOT view. The following example creates a snapshot baseline named peak_time baseline:

```
begin
dbms_workload_repository.create_baseline (start_snap_id => 125,
    end_snap_id => 185, baseline_name => 'peak_time baseline',
    dbid => 2210828132);
end;
```

If you don't specify a name for the baseline, Oracle will assign a system-generated identifier.

You can drop a snapshot baseline by using the DROP_BASELINE procedure of the DBMS_WORKLOAD_REPOSITORY package:

```
begin
  dbms_workload_repository.drop_baseline (baseline_name => 'peak_time
 baseline',
cascade => FALSE, dbid => 2210828132);
end;
```

The CASCADE parameter is FALSE by default. By setting this parameter to TRUE, you can drop the actual snapshots as well.

Purging AWR Snapshots

As you've learned, by default, the AWR runs every hour, and Oracle saves AWR statistics for a default period of seven days. After the seven-day period, Oracle removes the snapshots, starting with the oldest ones first. Oracle estimates that if you have ten concurrent sessions, it will take between 200MB and 300MB of disk space to store the data that it saves over the standard seven-day period. You must therefore ensure that your SYSAUX tablespace has at least this much free space, if you want Oracle to retain the AWR data for the standard default period of seven days. The number of user sessions is a key determinant of the space necessary for the AWR statistics.

on the job

If your SYSAUX tablespace runs out of space, Oracle will automatically delete the oldest set of snapshots, to make room for new snapshots.

In addition to the number of active user sessions, two parameters affect the total statistics retained in the SYSAUX tablespace:

- **RETENTION** As you know, the default retention period for AWR statistics is seven days. The minimum retention period is one day. The longer the retention period, the more space the AWR will need in the SYSAUX tablespace.
- **INTERVAL** By default, the AWR collects data every 60 minutes and the minimum interval value is 10 minutes. You can't collect statistics more frequently than in 10-minute intervals. The more frequently you schedule the AWR, the more data the AWR will collect. And the more infrequent the AWR snapshots, the greater the chance that you may miss short bursts in disk or memory usage that may occur in your database.

You can use the DBMS_WORKLOAD_REPOSITORY package to modify the snapshot settings, as shown here:

```
begin
  DBMS_WORKLOAD_REPOSITORY.MODIFY_SNAPSHOT_SETTINGS( retention => 43200,
                 interval => 30, dbid => 3310949047);
end;
```

Oracle recommends that you make the retention period the same as your database workload cycle. If your database is like many typical OLTP databases, you probably have OLTP transactions throughout the weekdays, with batch jobs scheduled during nights and weekends. If this is the case, your workload is considered to span a week, in which case, the default AWR retention period of seven days is just fine.

Note that Oracle deletes older snapshots in chronological order. In addition, AWR always *retains* baseline snapshots until you remove the baselines themselves.

exam watch

The default period of retention for AWR data is seven days and the minimum is one day. However, if there is no room in the SYSAUX tablespace, that fact will override all other retention settings. Oracle will start deleting snapshots, by first overwriting the oldest ones with new data.

on the Job

If you set the value of the `RETENTION` parameter to zero, you disable the automatic purging of the AWR. If you set the value of the `INTERVAL` parameter to zero, you disable the automatic capturing of AWR snapshots.

Creating AWR Reports

Oracle provides a script named awrrpt.sql (located in the $ORACLE_HOME/rdbms/admin directory) to generate summary reports about the statistics collected by the AWR facility. Don't confuse this report with the ADDM report (created by running addmrpt.sql) that you learned about in Chapter 3. The AWR report doesn't give you recommendations, as does an ADDM report. Rather, it provides information about the various kinds of performance statistics in the repository. The AWR report looks remarkably similar to the traditional STATSPACK reports.

The awrrpt.sql actually calls another script, the awrrpt1.sql script, to generate the AWR report for you. Note that you must have the SELECT ANY DICTIONARY privilege in order to run the awrrpt.sql script.

When you run the awrrpt.sql script, Oracle will ask you to make the following choices for your reports:

- Choose between an HTML or plain text report
- Specify the beginning and ending snap IDs
- Specify the number of days of snapshots to choose from

The AWR reports include the following types of information:

- Load profile
- Instance efficiency percentages (target 100 percent)
- Top-five timed events
- Wait events and latch activity
- Time model statistics
- Operating system statistics
- SQL ordered by elapsed time
- Tablespace and file I/O statistics
- Buffer pool and PGA statistics and advisories

Managing AWR Statistics with Data Dictionary Views

You can manage AWR statistics through the following data dictionary views:

- The `DBA_HIST_SNAPSHOT` view shows all snapshots saved in the AWR.
- The `DBA_HIST_WR_CONTROL` view displays the settings to control the AWR.
- The `DBA_HIST_BASELINE` view shows all baselines and their beginning and ending snap ID numbers.

CERTIFICATION OBJECTIVE 4.03

Active Session History (ASH)

The AWR snapshots are very useful, but Oracle takes these snapshots only every 60 minutes. If you are interested in analyzing a performance problem that happened 10 minutes ago, the AWR snapshots aren't of any help to you. However, you do have a way to get that information. Oracle Database 10*g* now collects the new Active Session History (ASH) statistics (mostly the wait statistics for different events) for all active sessions *every second*, and stores them in a circular buffer in the SGA. Thus, ASH records *very recent session activity (within the past five or ten minutes)*.

on the job

Note that not all the extra statistics in Oracle Database 10g described in this chapter will have a detrimental affect on performance, since the statistics mostly come directly from the SGA, via background processes. The ASH feature uses about 2MB of SGA memory per CPU.

Current Active Session Data

As you are aware, the `V$SESSION` view holds all the session data for all current sessions. The `V$SESSION` view contains 72 columns of information, so it's unwieldy when you are trying to analyze session data. That's why ASH samples the `V$SESSION` view and gets you the most critical wait information from it. Oracle provides the new `V$ACTIVE_SESSION_HISTORY` view, which enables you to access the ASH statistics. The view contains one row for each active session that ASH samples and returns the latest session rows first.

V$ACTIVE_SESSION_HISTORY is where the database stores a sample of all active session data. What is an *active session*? In the V$ACTIVE_SESSION_HISTORY view, there is a column called SESSION_STATE, which indicates whether a session is active. The SESSION_STATE column can take two values: ON CPU or WAITING. A session is defined as an active session in the following cases:

- The session state is ON CPU, meaning that it is actively using the CPU to perform a database chore.
- The session state is WAITING, but the EVENT column indicates that the session isn't waiting for any event in the IDLE class.

Note that the ASH is really a rolling buffer in the SGA; it is an *in-memory* active session history. Thus, in a busy database, older information is frequently overwritten, since ASH collects data every second from the V$SESSION view. ASH stores its data in V$ACTIVE_SESSION_HISTORY, but this view will contain only session samples of the most recent active sessions, because the ASH statistics are overwritten in memory.

Older Active Session History

The new data dictionary view DBA_HIST_ACTIVE_SESSION_HISTORY provides historical information about recent active session history. In other words, this view is nothing but a collection of snapshots from the V$ACTIVE_SESSION_HISTORY view. To put it in simple terms, the DBA_HIST_ACTIVE_SESSION_HISTORY view samples the V$ACTIVE_SESSION_HISTORY view, which itself is a sample of active session data.

How does the database populate the DBA_HIST_ACTIVE_SESSION_HISTORY view? There are two ways to populate the view:

- During the course of the regular (by default, hourly) snapshots performed by the AWR, the MMON background process flushes the ASH data to the AWR. By default, the MMON process performs this data transfer on an hourly basis.

- Oracle may also need to transfer data to the `DBA_HIST_ACTIVE_HISTORY` view in between the regular snapshots, if the memory buffer is full and you can't write new session activity data to it. In this case, the new MMNL background process will perform the flushing of data from the memory buffer to the data dictionary view.

CERTIFICATION OBJECTIVE 4.04

Server-Generated Alerts

Oracle DBAs generally use SQL scripts to alert them when abnormal conditions occur. Oracle Database 10*g* now has a built-in system of alerts, formally called *server-generated alerts*. The Oracle Database 10*g* Server automatically alerts you when certain problems occur in the database.

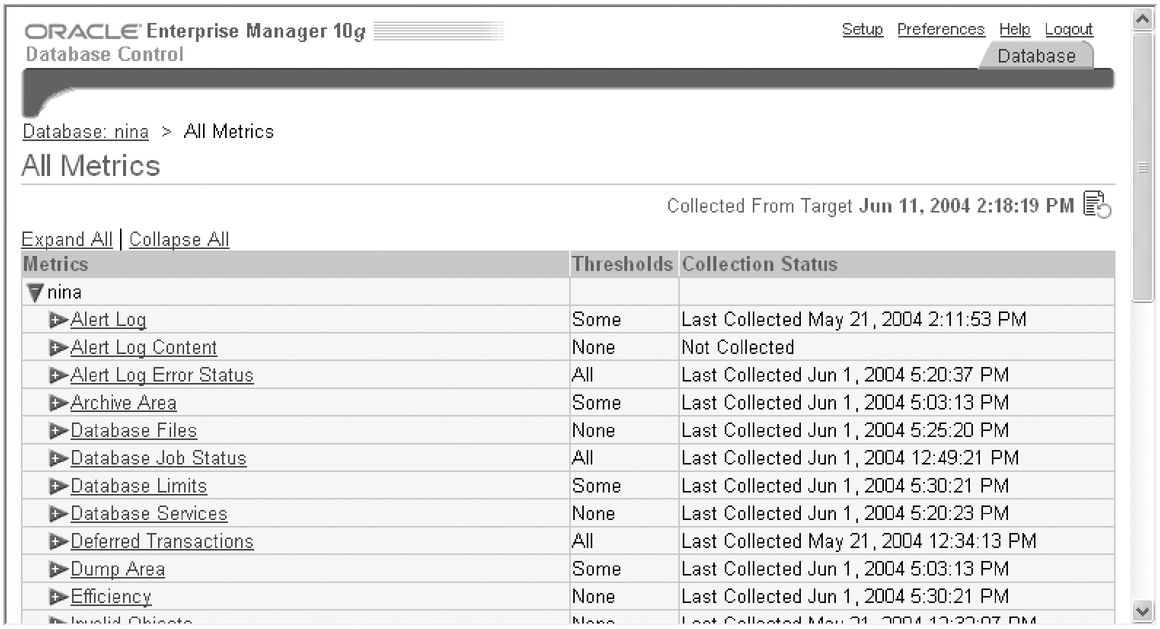
Oracle Database 10*g* uses a set of indicators called *metrics*, which show you the rate of change of a cumulative statistic. A typical example of an Oracle metric is the number of database calls per second. Once a metric crosses a certain *threshold*, Oracle sends an alert to notify you that the target has a problem. Since database metrics are an important part of server-generated alerts, let's first take a look at how you monitor database metrics.

Introduction to Metrics

Metrics are indicators of the health of various database targets. The set of metrics depends on the target you are measuring. For example, key metrics help you decide if the host is healthy and if resources like CPU, memory, and I/O are performing at an acceptable level.

The OEM Database Control's All Metrics page, shown in Figure 4-2, offers an excellent way to view the various metrics. To access this page, from the Database Control home page, click All Metrics under the Related Links heading. From here, you can view all the metrics in your database. For details, click the Expand All link in the left corner of the page. If you want the drill down to the details of any specific metric, just click it.

FIGURE 4-2 The All Metrics page of database control



ORACLE Enterprise Manager 10*g*
Database Control

Setup Preferences Help Logout

Database

Database: nina > All Metrics

All Metrics

Collected From Target **Jun 11, 2004 2:18:19 PM**

Expand All | Collapse All

Metrics	Thresholds	Collection Status
nina		
Alert Log	Some	Last Collected May 21, 2004 2:11:53 PM
Alert Log Content	None	Not Collected
Alert Log Error Status	All	Last Collected Jun 1, 2004 5:20:37 PM
Archive Area	Some	Last Collected Jun 1, 2004 5:03:13 PM
Database Files	None	Last Collected Jun 1, 2004 5:25:20 PM
Database Job Status	All	Last Collected Jun 1, 2004 12:49:21 PM
Database Limits	Some	Last Collected Jun 1, 2004 5:30:21 PM
Database Services	None	Last Collected Jun 1, 2004 5:20:23 PM
Deferred Transactions	All	Last Collected May 21, 2004 12:34:13 PM
Dump Area	Some	Last Collected Jun 1, 2004 5:03:13 PM
Efficiency	None	Last Collected Jun 1, 2004 5:30:21 PM
Invalid Objects	None	Last Collected May 21, 2004 12:32:07 PM

Table 4-1 lists the basic metric groups in Oracle Database 10*g*.

TABLE 4-1 Oracle Database 10*g* Metric Groups

Metric	Description
Event Class Metrics	Metrics collected on the wait event class level; for example, DB_TIME_WAITING
Event Metrics	Metrics collected on various wait events
File Metrics Long Duration	Metrics collected at the file level; for example, AVERAGE_FILE_WRITE_TIME
Service Metrics	Metrics collected at the service level; for example, CPU_TIME_PER_CALL
Session Metrics Short (Long) Duration	Metrics collected at the session level; for example, BLOCKED_USERS
System Metrics Short (Long) Duration	Metrics collected at the system level
Tablespace Metrics Long Duration	Metrics collected at the tablespace level; for example, TABLESPACE_PCT_FULL.

Viewing In-Memory Metrics

The new MMON background process collects database metrics continuously and automatically saves them in the SGA for one hour. You can view all the system-related metrics saved in memory by using views like `V$SYSMETRIC_HISTORY` and `V$SYSMETRIC`. Here, for example, are the typical kinds of system metrics maintained in the `V$SYSMETRIC` view:

- Buffer Cache Hit Ratio
- CPU Usage Per Sec
- Database CPU Time Ratio
- Database Wait Time Ratio
- Disk Sort Per Sec
- Hard Parse Count Per Sec
- Host CPU Utilization (%)
- Library Cache Hit Ratio
- SQL Service Response Time
- Shared Pool Free %

The `V$SERVICEMETRIC` and `V$SERVICEMETRIC_HISTORY` views, on the other hand, provide details about service-level metrics. The `V$SERVICEMETRIC` view, for example, contains the ELAPSEDPERCALL and the CPUPERCALL columns for each service, including the database service. `V$SERVICEMETRIC` lists the metrics for the last minute (a few over the last 15 seconds), and `V$SERVICEMETRIC_HISTORY` gives them per minute over the last hour.

Viewing Saved Metrics

The only way Oracle will save the metric information that is continuously placed in its SGA by the MMON process is through the AWR's snapshot mechanism. As you know, the AWR takes its snapshots every 60 minutes by default. The AWR's data-collection facility includes data pertaining to metrics. The metric data that AWR collects with the help of the MMON background process is permanently stored in the `DBA_HIST_*` views, such as `DBA_HIST_SERVICE_NAME` and `DBA_HIST_SESSMETRIC_HISTORY`.

on the Job

You can view the current values of any metric by using the `V$` views, such as the `V$METRICNAME` and `V$SERVICEMETRIC` views. The MMON process regularly computes these metrics for you. However, the various performance metrics stay in the SGA for only an hour. MMON will automatically flush the metric data from the SGA to the `DBA_HISTORY_*` views on disk. These history views maintain a permanent record of the various metrics. Examples of the history views are `DBA_HIST_SUMMARY_HISTORY`, `DBA_HIST SYSMETRIC_HISTORY`, and `DBA_HIST_METRICNAME`. Each of these views contains snapshots of the corresponding `V$` view. For example, the `DBA_HIST_SYSMETRIC_HISTORY` view contains snapshots of the `V$SYSMETRIC_HISTORY` view.

Database Alerts

A *database alert* tells you that there is a potential problem in your database. The potential problem can occur when a threshold value for a monitored metric is crossed or a database target simply becomes unavailable. For example, an undo tablespace may trigger the Long Query Warning alert, thus alerting the DBA about a long-running query that may fail because the system is running out of free space in the undo tablespace. Oracle calls the threshold alerts *stateful alerts*. Oracle calls the types of alerts issued when a target becomes unavailable *stateless alerts*.

Threshold-Based Alerts and Problem Alerts

You can set threshold values at two different levels: *warning* and *critical*. Thus, there are three situations when a database can send an alert:

- A monitored metric crosses a critical threshold value
- A monitored metric crosses a warning threshold value
- A service or target suddenly becomes unavailable

We can refer to the first two types of the alerts as *threshold-based alerts* and the third type as *problem alerts*. Problem alerts report problems that occur in the database, such as an ORA-1555 (snapshot too old) error. Problem alerts are based on a certain predetermined *event* (usually bad) occurring in the database.

Threshold-based alerts are based on thresholds for certain objects or events in the database; for example, a tablespace may be reaching 95 percent of its total allocated space. Threshold-based alerts thus are dependent on *metrics*, not events. The threshold itself could be an internally set level, or you, the DBA, can set your own alert threshold levels. When a metric crosses a threshold, Oracle automatically generates an alert to you. In addition, the database can undertake remedial action to fix the problem (or potential problem), if you specify a response action.

When you use threshold-based alerts, Oracle further makes a distinction between a warning type alert (severity level 5) and a critical (severity level 1) alert. For example, by default, the database will send you a warning alert when any tablespace hits an 85 percent space use threshold. When the usage reaches the 97 percent level, you get a critical alert.

Default Server-Generated Alerts

Even if you don't explicitly set any alerts in your database, there are several alerts that Oracle enables by default when you create an Oracle Database 10*g* database. These default server-generated alerts may be either problem alerts or threshold alerts. The server-generated alerts are very similar to the OEM alerts in prior versions of the Oracle database.

The following are the default, or out-of-the-box, server-generated alerts:

- Any snapshot too old errors
- Tablespace space usage (warning alert at 85 percent usage; critical alert at 97 percent usage)
- Resumable session suspended
- Recovery session running out of free space

In addition, Oracle automatically set thresholds on all metrics with the object type SYSTEM.

exam watch

The new database alert system has really nothing to do with the alert log of the database. The only time the alert log may be involved is when the database cannot write an alert to the alert queue, in which case, it records this exception in the alert log.

exam watch

Tablespace usage alerts are based on the default values of 85 percent space usage for a warning alert and 97 percent space usage for a critical alert.

How the Server-Generated Alert Mechanism Works

As you are probably aware, the OEM enables the notification of various types of alerts. So, what is this new server-generated alert concept? Well, under the old OEM alert notification system, the OEM was responsible for gathering the metrics and generating the alerts. In the new Oracle Database 10*g* server-generated alert system, it is not the OEM, but rather the *database* itself that collects the metrics that underlie all alerts.

The new MMON process checks all the available metrics to see if any metric crosses a preset threshold. When a database reaches a threshold for one of the monitored variables, Oracle will send you an alert. For example, the Database Control can alert you, using e-mail or a pager notification, when the database reaches any tablespace thresholds.

As described in the previous section, your database comes with a set of default alerts already configured. In addition, you can choose to have other alerts. You may also change the thresholds for the default alerts. You can perform these tasks with the help of the OEM Database Control or with Oracle-supplied PL/SQL packages.

Using the Database Control, you can set up notification rules; for example, you can specify a blackout period for the alerts. When the database issues an alert, you can see that in the Database Control alert window. In addition, you'll receive a notification, if you've configured the system to send you one. The alerts usually are accompanied by a recommendation to fix the problem as well.

exam watch

Make sure you set the `STATISTICS_LEVEL` parameter to `TYPICAL` or `ALL`, in order to user the server-generated alerts feature. You can use either the OEM Database Control or a PL/SQL package to manage system-generated alerts. In addition, you can display alerts directly by subscribing to the alert queue (`ALERT_QUE`).

Managing Alerts

Several new data dictionary views aid in managing both database metrics and server-based alerts. The following sections explain using the various methods to manage alerts. Let's first look at how you can use the OEM Database Control for this purpose.

Using the Database Control to Manage Server Alerts

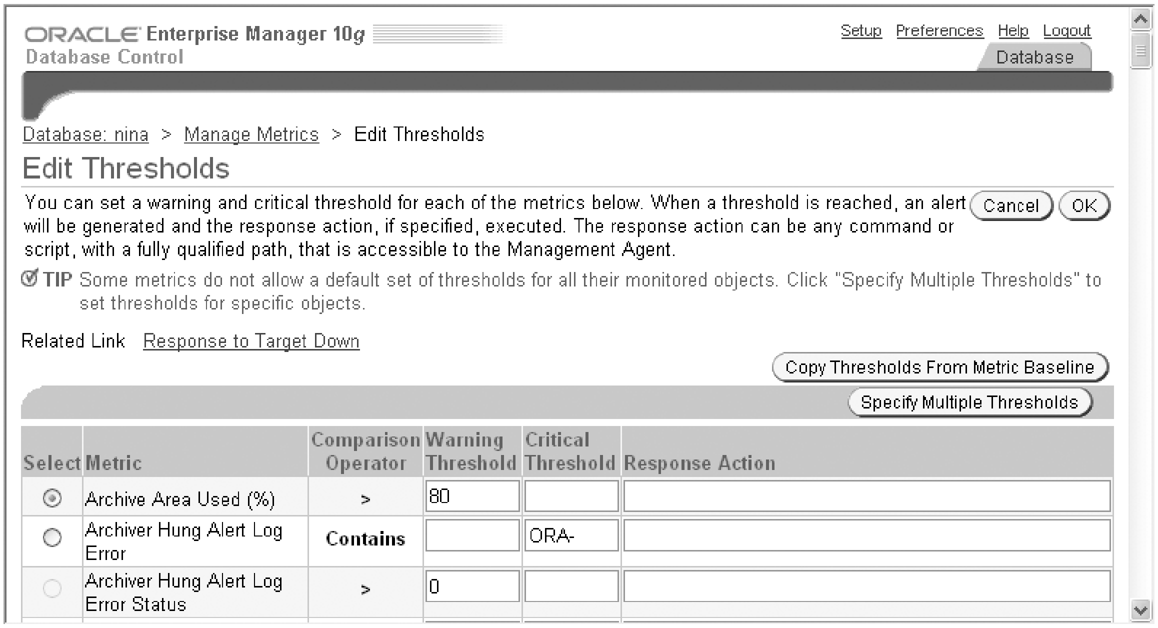
If you've used the Oracle9*i* OEM, you're familiar with the Enterprise Manager alerts, wherein you can configure OEM to alert you when it encounters certain errors in the database, using a pager or e-mail. Server-generated alerts work in a similar fashion. In addition to the capability to send alerts, now you can configure alert thresholds as well.

Setting Alert Thresholds It is very easy to set your own warning and critical thresholds for any database metric. To set alert thresholds, from the Database Control home page, click the Manage Metrics link, which you'll find under the Related Links group. On the Manage Metrics page, click the Edit Thresholds button. You'll see the Edit Thresholds page, as shown in Figure 4-3. For each metric on the Edit Thresholds page, you can set the following:

- **A warning and critical threshold** You can set an arbitrary threshold or compute a threshold based on a set of baselines for a metric. For example, you may specify that the database should generate a threshold alert if a resource use is 15 percent higher than its normal baseline values. You may also specify multiple thresholds.
- **A response action** This action can be a SQL script or an operating system command. Oracle will automatically execute this response action immediately when the alert is generated. Make sure that you provide the complete path to the SQL script or operating system command, so the OEM Management Agent can find it.

Setting Notification Rules When the database needs to send you an alert, it follows any notification rules that you've set up. Notification rules enable you to choose the conditions under which you want to receive a message from the OEM. For example, you many not want to be woken up at 2:00 A.M. just because a tablespace with 100GB allocated space has reached an 80 percent usage level. On the other hand, you would surely want to know immediately when a 200MB tablespace has crossed the critical 97 percent usage level.

FIGURE 4-3 Using the Database Control to set alert thresholds


ORACLE Enterprise Manager 10g
Database Control
Setup Preferences Help Logout
Database
Cancel
OK
Copy Thresholds From Metric Baseline
Specify Multiple Thresholds

Database: nina > Manage Metrics > Edit Thresholds

Edit Thresholds

You can set a warning and critical threshold for each of the metrics below. When a threshold is reached, an alert will be generated and the response action, if specified, executed. The response action can be any command or script, with a fully qualified path, that is accessible to the Management Agent.

TIP Some metrics do not allow a default set of thresholds for all their monitored objects. Click "Specify Multiple Thresholds" to set thresholds for specific objects.

Related Link Response to Target Down

Select	Metric	Comparison Operator	Warning Threshold	Critical Threshold	Response Action
◉	Archive Area Used (%)	>	80		
○	Archiver Hung Alert Log Error	Contains		ORA-	
○	Archiver Hung Alert Log Error Status	>	0		

You can use the Database Control to set notification rules. These rules are set through Preferences. From the Database Control home page, click the Preferences link (at the very bottom of the page) to go to the Preferences page. Then click the Rules link in the Notification section. Select any metric, such as Listener Availability, and click the Edit button. Then you can set notification rules for the selected event, such as the following:

- The precise conditions under which you want receive a notification
- The metrics for which you would like to receive alerts
- The severity conditions (critical and warning levels) under which you would like to receive notification
- The notifications you want to be sent
- E-mail notification or an advanced notification method, if you've configured one

Using the DBMS_SERVER Package to Manage Alerts

Although the OEM Database Control interface provides an easy way to manage database alerts, there may be times when you need to incorporate certain changes inside a PL/SQL program. At times like this, you can use the Oracle-supplied PL/SQL package DBMS_SERVER_ALERT to set up and modify thresholds on various database metrics. The DBMS_SERVER_ALERT package has two main procedures: GET_THRESHOLD and SET_THRESHOLD.

You use the SET_THRESHOLD procedure to define threshold settings for a database metric. This procedure has the following structure:

```
SQL> desc dbms_server_alert.set_threshold
PROCEDURE dbms_server_alert.set_threshold
 Argument Name                  Type                    In/Out Default?
 ------------------------------ ----------------------- ------ --------
 METRICS_ID                     BINARY_INTEGER          IN
 WARNING_OPERATOR               BINARY_INTEGER          IN
 WARNING_VALUE                  VARCHAR2                IN
 CRITICAL_OPERATOR              BINARY_INTEGER          IN
 CRITICAL_VALUE                 VARCHAR2                IN
 OBSERVATION_PERIOD             BINARY_INTEGER          IN
 CONSECUTIVE_OCCURRENCES        BINARY_INTEGER          IN
 INSTANCE_NAME                  VARCHAR2                IN
 OBJECT_TYPE                    BINARY_INTEGER          IN
 OBJECT_NAME                    VARCHAR2                IN
SQL>
```

Here's an example that sets up an automatic alert monitoring of CPU use by each user in the instance:

```
DBMS_SERVER_ALERT.SET_THRESHOLD(
DBMS_SERVER_ALERT.CPU_TIME_PER_CALL, DBMS_SERVER_ALERT.OPERATOR_GE, '8000',
DBMS_SERVER_ALERT.OPERATOR_GE, '10000', 1, 2, 'prod1',
DBMS_SERVER_ALERT.OBJECT_TYPE_SERVICE,
'main.regress.rdbms.dev.us.oracle.com');
```

This SET_THRESHOLD procedure example specifies that a *warning* alert is issued when CPU time exceeds 8000 microseconds for each user call, and a *critical* alert is issued when CPU time exceeds 10,000 microseconds for each user call. The other key arguments are as follows:

- CPU_TIME_PER_CALL specifies the metric identifier.
- The observation period is set to 1 minute. Oracle issues the alert after the threshold is crossed for 1 minute.

- The number of consecutive occurrences is set to 2. This is the number of times the metric value crosses the threshold values before Oracle generates the alert.
- The name of the instance is set to `prod1`.
- The constant `DBMS_ALERT.OBJECT_TYPE_SERVICE` specifies the object type on which the threshold is set. Here, the service name is `main.regress.rdbms.dev.us.oracle.com`.

You use the `GET_THRESHOLD` procedure to find out the current threshold settings for any database metric. You can discover both the warning and critical thresholds by using this procedure.

exam watch

If you don't want Oracle to send any metric-based alerts, simply set the the warning value and the critical value to NULL when you execute the `DBMS_SERVER_ALERT.SET_THRESHOLD` procedure.

Using the Alert Queue

Earlier, you learned how you can view and change threshold settings for the server alert metrics using the `SET_THRESHOLD` and `GET_THRESHOLD` procedures of the `DBMS_SERVER_ALERTS PL/SQL` package. Of course, you may also use the OEM Database Control to manage alerts. In addition, you can use procedures from the `DBMS_AQ` and `DBMS_AQADM` packages for directly accessing and reading alert messages in the alert queue. The SYS user account owns the alert queue and by default, the Database Control is the only user of the alert queue. However, Oracle provides the new `DBMS_AQADM` package, which consists of various procedures that help you directly access the alerts stored in the alert queue (`ALERT_QUE`).

Using the `DBMS_AQADM` package procedures, you can subscribe to the `ALERT_QUE`, set thresholds for alerts, and display alert notifications. Here are some of the main procedures of the `DBMS_AQADM` package that help you manage database alerts:

- Use the `GRANT_SYSTEM_PRIVILEGE` procedure to grant AQ system privileges to users and roles.
- Use the `ADD_SUBSCRIBER` procedure to add a default subscriber to a queue.
- Use the `CREATE_AQ_AGENT` procedure to register an agent.

- Use the ENABLE_DB_ACCESS procedure to grant an AQ Agent-specific database user's privileges.

In addition to the DBMS_AQADM package, Oracle also offers the DBMS_AQ package, which provides a direct interface to the Oracle Streams Advanced Queuing (AQ) feature. You can use the DBMS_AQ procedure to *manage alert notifications*. Here are some of the important procedures of the DBMS_AQ database package.

- Use the REGISTER procedure to register for message notifications.
- Use the DEQUEUE procedure to remove a message from a queue.

Using Data Dictionary Views to Manage Metrics and Alerts

The OEM Database Control is your first stop in the managing of database alerts and the metrics associated with them. There are also several new data dictionary views that provide information about database metrics and alerts. I've already mentioned the V$METRICNAME, V$SYSMETRIC, V$SERVICEMETRIC, and V$SYSMETRIC_HISTORY views earlier in this chapter. Following is a list of some of the other key views:

- V$ALERT_TYPES displays information about server alert types.
- DBA_HIST_SYSMETRIC_HISTORY contains snapshots of V$SYSMETRIC_HISTORY.
- DBA_ALERT_HISTORY provides a history of alerts that are no longer outstanding; that is, all alerts that you have already resolved.
- DBA_OUTSTANDING_ALERTS contains all the stateful (threshold) alerts that have yet to be resolved.
- DBA_THRESHOLDS shows the name as well as the critical and warning values for all thresholds in the database.

I'll describe a couple of the important views in more detail in the following sections.

exam watch

The DBA_OUTSTANDING_ALERTS view shows all alerts that have yet to be resolved. The DBA_ALERT_HISTORY view is where all cleared alerts are moved.

V$ALERT_TYPES The `V$ALERT_TYPES` view provides information about all system alert types. Three columns in this view are noteworthy:

- **STATE** Holds two possible values: *stateful* or *stateless*. Stateful alerts are those alerts that clear automatically when the alert threshold that prompted the alert is cleared. The database considers all the nonthreshold alerts as stateless alerts. A stateful alert first appears in the `DBA_OUTSTANDING_ALERTS` view and goes to the `DBA_ALERT_HISTORY` view when it is cleared. A stateless alert goes straight to `DBA_ALERT_HISTORY`.
- **SCOPE** Classifies alerts into *database wide* and *instance wide*. The only database-level alert is the one based on the Tablespace Space Usage metric. All the other alerts are at the instance level.
- **GROUP_NAME** Oracle aggregates the various database alerts into some common groups. Here are the common alert groups:
    - Space, such as a suspended session, snapshot too old, or tablespace out of space
    - Performance, such as elapsed time or CPU time exceeds set thresholds
    - Configuration-related database alerts

DBA_THRESHOLDS The `DBA_THRESHOLDS` view provides the current threshold settings for all alerts. This view is useful when you want to find out the current threshold settings for any alert. Here is a simple example of how to use the view:

```
SQL> select metrics_name, warning_value, critical_value,
     consecutive_occurrences
     from DBA_THRESHOLDS
     where metrics_name LIKE '%CPU Time%';
```

EXERCISE 4-2

Use the DBMS_SERVER_ALERT Package to Set Alert Thresholds

In the following exercise, you will define a small tablespace and then attempt to create a large table in it. The database will alert you because you'll be crossing a threshold setting for the free space in the tablespace. The example will show you how to set, view, and clear an alert.

1. Create a test the tablespace using the following command:

```
SQL>  create tablespace test datafile 'test01.dbf' size 1M
      extent management local uniform size 200K;
Tablespace created.
```

2. Set your tablespace alert thresholds as follows (warning alert at 75% full and critical at 90% full):

```
SQL> exec dbms_server_alert.set_threshold(-
> dbms_server_alert.tablespace_pct_full,dbms_server_
    alert.operator_ge,'75',-
> dbms_server_alert.operator_ge,'90',1,1,null,-
> dbms_server_alert.object_type_tablespace,'TEST');
PL/SQL procedure successfully completed.
SQL>
```

3. When you create a new table using the following SQL statement, you will set off an alert (because the 'minextents 4' clause for the new table will cause the tablesapce to cross its warning threshold of 75% space full):

```
SQL> create table test_table (name varchar2(30))
     tablespace test
     storage (minextents 4);
Table created.
SQL>
```

4. You can verify the tablespace alert in the following way:

```
SQL>  select reason from dba_outstanding_alerts;
REASON
-----------------------------------------------------
Tablespace [TEST] is [78 percent] full
SQL>
```

5. You can clear the alert by increasing the size of the datafile that is part of the tablespace named small and see what happens to the alert by querying the DBA_OUTSTANDING_ALERTS view. You'll find that the alert is gone from that view, since it has been cleared.

```
SQL> alter tablespace test add datafile 'test02.dbf' size 2M;
Tablespace altered.
SQL>
SQL> select reason from dba_outstanding_alerts;
no rows selected
SQL>
```

6. Where do cleared alerts go? All cleared alerts will show up in the DBA_ALERT_HISTORY. You can verify that the cleared tablespace alert is in that view, by using the following query.

```
SQL> select reason,resolution from dba_alert_history;
REASON                                                      RESOLUTION
------------------------------------------------------------------------
Tablespace [TESTNEW] is [13 percent] full                    cleared
SQL>
```

CERTIFICATION OBJECTIVE 4.05

The Automated Tasks Feature

Most Oracle DBAs are familiar with using the UNIX crontab feature or the Windows AT facility to manage their regularly scheduled jobs. Although you can schedule the jobs, you still must decide when to run these automated tasks. Now, in Oracle Database 10*g*, the database itself will perform some of the routine tasks all by itself, using the new automated tasks feature. For example, in Chapter 3, you learned how the database now collects its own optimizer statistics, using the Scheduler facility. In Chapter 1, I also explained how the DBCA lets you automate the maintenance of backup jobs while you are creating the database.

The new Oracle Scheduler—with its concepts of jobs, job classes, programs, and operation windows—lies at the heart of the automated tasks feature. Therefore, let's start our discussion of the automated tasks feature by briefly looking at the Scheduler.

An Introduction to the Job Scheduler

The Scheduler can perform very complex scheduling tasks for you, unlike a normal crontab-based program. The Scheduler is nothing but a set of PL/SQL functions and procedures, which together provide a facility to schedule and manage database and operating system jobs. You may run PL/SQL and Java stored procedures and functions, C functions, regular SQL scripts, and UNIX or Windows scripts using the Scheduler.

The Scheduler consists of the following four important concepts:

- **Program** A Scheduler *program* consists of metadata about what task the Scheduler will run. A program consists of a specific action, along with specific arguments to run the task. A *task* is any executable program, which you may reuse. You can create several named tasks and store them in the database.
- **Job** A *job* is a user-defined task that you schedule for running one or more times. A *job class* is a group of similar jobs. You create job classes so you can prioritize the jobs in an orderly way, when you are allocating database resources.
- **Schedule** A *schedule* indicates when a job should be run. A schedule tells the database the date, time, and frequency of execution for a job. Schedules have a start date, an end date, and a repeat interval.
- **Window** A *window* is a time-duration for which you may specify certain resource allocations. A window has a start date, an end date, and a duration that specifies how long the window will be open each time it opens. A window is usually associated with a resource plan (created using the Database Resource Manager), which specifies how resources should be allocated among groups of jobs (called j*ob classes*). The Scheduler comes with a default maintenance window group as you saw in Chapter 3. These include the weeknight window (10:00 P.M. to 6:00 A.M., Monday through Friday) and the weekend window (12:00 A.M. Saturday to 12:00 A.M. Monday). Oracle automatically configures these two Scheduler windows when you create any Oracle Database 10*g* database.

You may use the DBMS_SCHEDULER package to manage various scheduling activities. You can create and drop jobs, job classes, programs, and windows using this package. The Scheduler and the DBMS_SCHEDULER package are discussed in detail in Chapter 7.

Managing Automated Tasks

Oracle schedules several maintenance tasks, including the automatic optimizer statistics collection job, during the maintenance window (MAINTENANCE_WINDOW_GROUP). Every Oracle Database 10*g* database comes with the default program GATHER_STATS_PROG, the default job GATHER_STATS_JOB, and default job class AUTO_TASKS_JOB_CLASS.

As you know from Chapter 3, the *job* GATHER_STATS_JOB executes the DBMS_STATS.GATHER_DATABASE_STATS_JOB_PROC procedure in order to collect optimizer statistics for the database. Oracle defines this job on the *program* GATHER_STATS_PROG. The job runs in the AUTO_TASKS_JOB_CLASS *job class*.

You can use the DBMS_SCHEDULER package to create a new job and add it to the automated task schedule. You must first create the SQL or PL/SQL code that will perform the task functions. You can then use the CREATE_JOB procedure to add this task to the automated task list. Here's an example:

```
begin
dbms_scheduler.create_job
(
 job_name         => 'DAILY_BACKUP_JOB_01',
 job_type         => 'STORED PROCEDURE',
 program_name     => 'DAILY_BACKUP_PROGRAM',
 schedule_name     => 'DAILY_BACKUP_WINDOW'
 );
end;
/
```

You can also use the Database Control interface to manage all aspects of the Oracle Scheduler, including creating, deleting, and modifying jobs, job classes, and windows.

CERTIFICATION OBJECTIVE 4.06

The Management Advisory Framework

Oracle Database 10*g* provides you with several server advisors to provide you with automatic performance details about various subsystems of the database. For example, the Undo Advisor tells you what the optimal undo tablespace size might be for your database. Each of these advisors bases its actions on a specific Oracle PL/SQL package like the `DBMS_ADVISOR` package.

The management advisory framework offers you a uniform interface for all Oracle advisors. Some of these advisors, like the PGA Advisor, have been around since Oracle9*i*. What is new is that Oracle has built a Common Manageability Structure in Oracle Database 10*g* to make it easy to manage the advisors. This allows you to use a similar method to invoke all the advisors, and the advisors provide their reports in a consistent fashion as well. All the advisors get their raw data from the AWR. They also store their analysis results in the AWR itself.

The advisory framework's primary function is to help the database improve its performance. The ADDM recommends using the management advisors on an ad hoc basis, whenever a performance problem needs a deeper analysis. DBAs can also use the advisors for performing what-if analyses.

The Advisors

We can group the automatic advisors into the following groups: memory-related, tuning-related, and space-related. In later chapters, you'll see a complete discussion of several of the management advisors. Let's briefly look at the advisors that fall into these three groups.

Memory-Related Advisors

Of the three memory-related advisors, two provide information about the SGA and one provides PGA advice. The following are the memory-related advisors:

- **Buffer Cache Advisor** This advisor tells you the benefits in increasing or decreasing the memory allocation to the `BUFFER_CACHE` parameter. The advisor shows the benefit as a decrease in disk I/O.

on the job

Obviously, if you are using Automatic Shared Memory Management, you don't need the Buffer Cache Advisor or the Library Cache Advisor to tell you how to size these memory components. Oracle itself will do that for you.

- **Library Cache Advisor** This advisor shows you the impact of changing the shared pool component of the SGA.
- **PGA Advisor** This advisor provides recommendations on optimal allocation of Program Global Area (PGA) memory, based on your database workload.

Space-Related Advisors

The two space-related advisors are new to Oracle Database 10*g*:

- **Segment Advisor** This advisor allows you to perform growth trend analyses on various database objects. This advisor also helps you perform object shrinkage, thus helping you reclaim unused space in your database.
- **Undo Advisor** This advisor bases its activities on system usage statistics, including the length of the queries as well as the rate of undo generation. The Undo Advisor facilitates Oracle's Automatic Undo Management feature. It helps you to correctly size your undo tablespace. It also helps you choose the correct undo retention parameters.

Tuning-Related Advisors

The ADDM, of course is the most important all-round tuning advisor in the database. Apart from the ADDM itself, there are two purely SQL tuning-related and performance-related advisors:

- **SQL Tuning Advisor** This advisor analyzes complex SQL statements and recommends ways to improve performance. The SQL Tuning Advisor bases all its work on internal statistics and may include suggestions to collect new statistics as well as SQL code restructuring.
- **SQL Access Advisor** This advisor mainly provides you advice on creating new indexes, materialized views, or materialized view logs. You provide the advisor a representative workload in order to get the advice.

on the job

Most of the database alert messages in the OEM also contain a link to specific management advisors. Thus, you can invoke a management advisory directly from the alert message itself.

Managing the Advisory Framework

You can manage all aspects of the management advisory framework easily, using the Database Control interface. You can also use the `DBMS_ADVISOR` package to create and manage tasks for each of the management advisors.

Using the DBMS_ADVISOR Package

You can invoke any of the management advisors through the OEM interface, using various wizards like the SQL Access Advisor Wizard. However, you may also run any of the advisors using the `DBMS_ADVISOR` package. Before a user can use any of the advisors, you must grant the user the ADVISOR privilege, as shown here:

```
SQL> grant advisor to oe;
Grant succeeded.
SQL>
```

The following are the steps you must follow in order to use the `DBMS_ADVISOR` package to manage various advisors:

- Create a task
- Adjust task parameters
- Perform the analysis
- Review the advisor recommendations

These steps are described in the following sections.

Creating a Task The first step in using an advisor is creating a task. A task is where the advisor stores all its recommendation-related information. You create a task using the `CREATE_TASK` procedure, as shown here:

```
VARIABLE task_id NUMBER;
VARIABLE task_name VARCHAR2(255);
EXECUTE :task_name := 'TEST_TASK';
EXECUTE DBMS_ADVISOR.CREATE_TASK ('SQL Access Advisor', :task_id,
:task_name);
```

Defining the Task Parameters After you create a new task, the next step is to set the parameters for this task. The task parameters control the recommendation process. The parameters you can modify belong to four groups: workload filtering, task configuration, schema attributes, and recommendation options.

exam watch

Use the `DBMS_ADVISOR` package to manage any modules in the advisory framework. You follow identical steps to use any advisor for tuning purposes. These steps are creating a task, adjusting task parameters, performing the analysis, and reviewing the recommendations of the advisor.

Here is an example showing how you can set various task parameters using the SET_TASK_PARAMETER procedure:

```
SQL> EXECUTE DBMS_ADVISOR.SET_TASK_PARAMETER ( -
'TEST_TASK', 'VALID_TABLE_LIST', 'SH.SALES, SH.CUSTOMERS');
```

In this example, the VALID_TABLE_LIST parameter belongs to the workload filtering group of parameters. You are instructing the advisor (the SQL Access Advisor) to exclude all tables from the analysis, except the sales and customers tables, from the SH schema.

The following example uses the STORAGE_CHANGE parameter from the recommendation options group to add 100MB of space to the recommendations.

```
SQL> EXECUTE DBMS_ADVISOR.SET_TASK_PARAMETER('TEST_TASK','STORAGE_CHANGE',-
100000000);
```

Generating the Recommendations To generate a set of recommendations by any advisor, you *execute the task* that you created earlier, using the EXECUTE_ TASK procedure of the DBMS_ADVISOR package. The EXECUTE_TASK procedure will generate recommendations, which consist of one or more actions. For example, excuting the SQL Access Advisor may provide a recommendation to create a materialized view and a materialized view log. Here's how you execute a task named TEST_TASK:

```
SQL> EXECUTE DBMS_ADVISOR.EXECUTE_TASK('TEST_TASK');
```

Viewing the Recommendations You can view the recommendations of the advisor task by using the GET_TASK_REPORT procedure. You may also use the DBA_ADVISOR_RECOMMENDATIONS view to check the recommendations, as shown in the following example:

```
SQL> select rec_id, rank, benefit
       from DBA_ADVISOR_RECOMMENDATIONS WHERE task_name = 'TEST_
TASK';
     REC_ID       RANK    BENEFIT
---------- ---------- ----------
         1          2       2754
         2          3       1222
         3          1       5499
         4          4        594
```

In this example, the RANK column shows how the four recommendations stack up against each other. The BENEFIT column shows the decrease in the execution cost for each of the four recommendations.

Using the Database Control to Manage the Advisory Framework

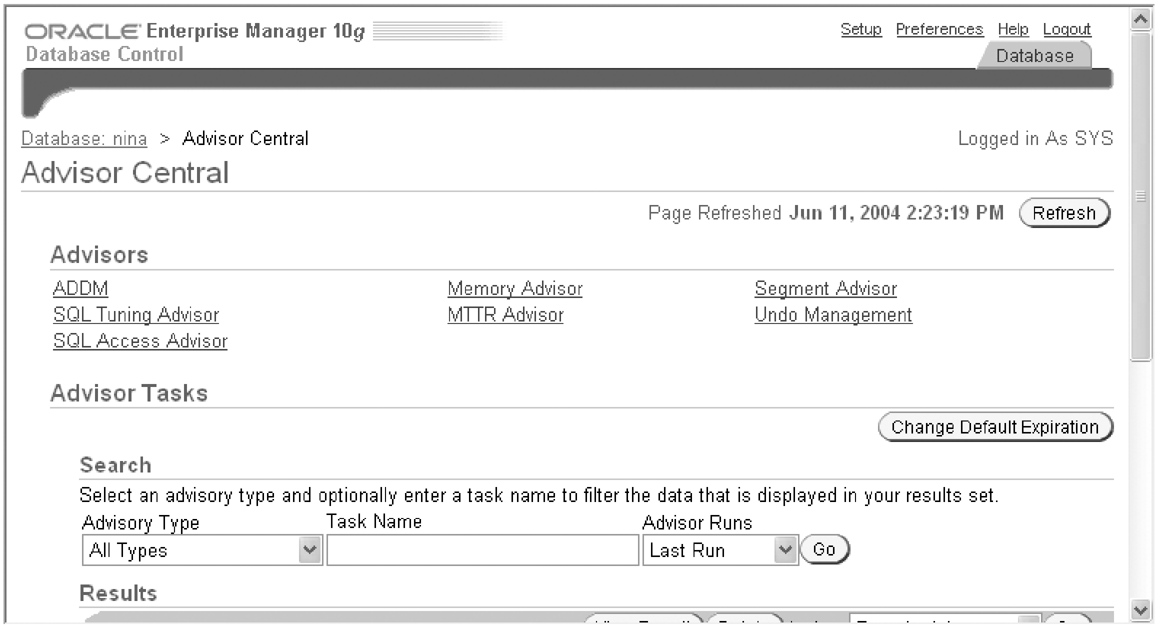
The best way to use the management advisors is through the OEM Database Control. All you need to do is to click the Advisor Central link on the Database Control home page. From the Advisor Central page, shown in Figure 4-4, you can select any of the management advisors in your database.

Using Dictionary Views to Manage the Advisory Framework

Several new data dictionary views provide information about managing tasks, as well recommendations made by the various advisors. Here are the main advisor-related dictionary views:

- DBA_ADVISOR_TASKS
- DBA_ADVISOR_PARAMETERS
- DBA_ADVISOR_FINDINGS
- DBA_ADVISOR_RECOMMENDATIONS
- DBA_ADVISOR_ACTIONS
- DBA_ADVISOR_RATIONALE

FIGURE 4-4 The Advisor Central page in OEM Database Control



Database: nina > Advisor Central

Logged in As SYS

Advisor Central

Page Refreshed Jun 11, 2004 2:23:19 PM Refresh

Advisors

ADDM
SQL Tuning Advisor
SQL Access Advisor
Memory Advisor
MTTR Advisor
Segment Advisor
Undo Management

Advisor Tasks

Change Default Expiration

Search

Select an advisory type and optionally enter a task name to filter the data that is displayed in your results set.

Advisory Type: All Types
Task Name:
Advisor Runs: Last Run
Go

Results

INSIDE THE EXAM

The test looks at your knowledge of the Automatic Workload Repository (AWR) in depth. You must be aware of the in-memory and persistent portions of the AWR and the mechanism that records the AWR statistics on disk. How do you create an AWR report? What are the options during the report creation process? Be aware that the Active Session History (ASH) is a component of the AWR, and that it focuses on current session activity. You must also understand the role of the background processes MMON and MMNL clearly.

You must know all the default settings, like how long the AWR retains information and how often AWR snapshots occur. How do you change these settings?

You must be aware of the various pages of the Database Control that you must traverse in order to manage the automatic alerts in your database. You can expect a question (or more) about custom configuration of alerts using the `DBMS_AQADM` package (create AQ Agent, add subscriber, and so on). The exam tests your knowledge of the types of events that set off server-generated alerts. What are the default tablespace usage alert settings? What are the various nonthreshold, or stateless, alerts?

The exam tests you on your knowledge of the PL/SQL interface for alert threshold settings, which is the `DBMS_SERVER_ALERT` package. You must know exactly what each of the `SET_THRESHOLD` parameters stands for. How do you set the critical and warning thresholds to NULL? You must also know the different components of the `DBMS_ADVISOR` package that help you in creating an advisor tuning session (create task, adjust task parameters, perform analysis, and review the results). The test checks your knowledge of important data dictionary views like `DBA_OUTSTANDING_ALERTS` (where outstanding alerts are stored) and `DBA_ALERT_HISTORY` (where cleared alerts go).

CERTIFICATION SUMMARY

This chapter introduced the new AWR feature, which is central to Oracle Database 10*g*'s performance tuning. You reviewed the different kinds of data stored in the AWR and how to manage the AWR by using the `DBMS_WORKLOAD_REPOSITORY` package. You learned how to create and manage AWR snapshots and baselines. The chapter also explained the ASH statistics, which are a key part of the AWR.

In the server-generated alerts section, you saw how Oracle generates automatic alerts. You learned how to use the Database Control to manage server alerts. The chapter showed you how to set alerts using the DBMS_SERVER_ALERT package.

The chapter introduced you the Oracle's automated tasks feature using the Scheduler. You also learned about the various Oracle management advisors. You reviewed the use of the DBMS_ADVISOR package to manage the advisory framework.

✓ TWO-MINUTE DRILL

Types of Oracle Statistics

- ❑ Oracle collects database statistics at the system and session level, as well as operating system statistics.
- ❑ Cumulative statistics are totals of various statistics since the startup of the instance.
- ❑ Both the `V$SYSSTAT` and `V$SESSTAT` views show cumulative performance statistics.
- ❑ Database metrics measure the rate of a change in a cumulative statistic.
- ❑ Database metrics are statistics that are derived from base statistics.
- ❑ The MMON background process issues database alerts and captures statistics for recently modified objects.
- ❑ The MMNL background process captures ASH history data when the memory buffer is full.
- ❑ ASH data is a sample of the most recent session data for all active sessions.
- ❑ Baseline data helps you to make meaningful comparisons of database performance between two periods.
- ❑ If you set the `STATISTICS_LEVEL` parameter to `BASIC`, the AWR turns off its statistics collection.
- ❑ If you set the `STATISTICS_LEVEL` parameter to `ALL`, the AWR collects all available statistics.
- ❑ If you set the `STATISTICS_LEVEL` to `TYPICAL`, the AWR collects what Oracle considers is a typically needed amount of statistics.

Automatic Workload Repository (AWR)

- ❑ The AWR lies at the heart of the Oracle Database 10*g* self-management framework.
- ❑ The AWR collects statistics in a temporary in-memory format, as well as a persistent component in the form of regular AWR snapshots.
- ❑ The MMON background process performs the AWR snapshots.

- ❑ By default, the AWR collects new snapshots every 60 minutes, but you can change this interval.
- ❑ You can also collect manual snapshots yourself.
- ❑ You uniquely identify a snapshot by its snap ID sequence number.
- ❑ The AWR collects base, database feature, and object usage; ASH; operating system; and other statistics.
- ❑ Managing the AWR snapshots enables you to manage the AWR.
- ❑ You can view AWR snapshot data through `V$ views or the Database Control interface.`
- ❑ The Manage Snapshots section of the AWR page in the Database Control lets you manage snapshots.
- ❑ You can use the `DBMS_WORKLOAD_REPOSITORY` package to create and drop snapshots, as well as to change the settings of the `RETENTION` and `INTERVAL` parameters.
- ❑ A snapshot baseline consists of any pair of AWR snapshots.
- ❑ A *preserved snapshot set* shows the range of snapshots you use for an AWR baseline.
- ❑ You can create and drop baselines using the `DBMS_WORKLOAD_REPOSITORY` package's `CREATE_BASELINE` and `DROP_BASELINE` procedures.
- ❑ By default, the AWR retains snapshot data for seven days, before automatically purging the data. You may modify this default behavior.
- ❑ The minimum retention period for AWR data is one day.
- ❑ The minimum interval value for AWR data collection is ten minutes.
- ❑ If the SYSAUX tablespace fills up, Oracle will automatically purge the oldest set of AWR snapshots.
- ❑ The longer the AWR retention period, the larger you should make your SYSAUX tablespace.
- ❑ The shorter the AWR snapshot interval, the larger you should make your SYSAUX tablespace.
- ❑ Oracle recommends that you set your retention period according to the length of your database workload cycle.

- ❑ If you set the value of the `RETENTION` parameter to zero, you disable the automatic purging of the `AWR`.
- ❑ If you set the `INTERVAL` parameter to zero, you disable the automatic snapshot collection by the `AWR`.
- ❑ Use the $ORACLE_HOME/rdbms/admin/awrrpt.sql script to produce AWR reports.
- ❑ The AWR reports are very similar to the old STATSPACK reports.

Active Session History (ASH)

- ❑ The ASH collects active session statistics and stores them in the SGA.
- ❑ The `V$SESSION` view is sampled every second, and the samples are saved to the `V$ACTIVE_SESSION_HISTORY` view.
- ❑ A database session is termed active, if either it is ON CPU or it is WAITING for an event that isn't from the IDLE class.
- ❑ The `DBA_HIST_ACTIVE_SESSION _HISTORY` view maintains a permanent record of a sample of the ASH data in the `V$ACTIVE_SESSION_HISTORY` view.
- ❑ The background process MMON is responsible for flushing ASH data to disk periodically.
- ❑ If the memory buffer for the ASH is full, the ASH data is flushed to disk by the MMNL background process.

Server-Generated Alerts

- ❑ Database metrics are the foundation for all Oracle alerts.
- ❑ The set of metrics for each alert depends on the targets that are covered. Threshold alerts are prompted when a database metric crosses a preset threshold value.
- ❑ Threshold alerts are also called *stateful alerts*.
- ❑ Error or problem alerts, which are issued because a service or target becomes unavailable, are also called *stateless alerts*.
- ❑ When you fix the problems that prompted an alert, the alert is cleared automatically.
- ❑ Threshold-based alerts are divided into warning and critical types.

❑ Default server-generated alerts may be problem alerts or threshold alerts.

❑ Oracle automatically sets thresholds on all metrics with the object type SYSTEM.

❑ The MMON process computes metrics and decides when a threshold has been crossed for a database metric.

❑ Set the `STATISTICS_LEVEL` parameter to `TYPICAL` if you want to use the server-generated alerts feature.

❑ You can display alerts directly by subscribing as a consumer to the `ALERT_QUE`.

❑ You can view current threshold settings by using the `GET_THRESHOLD` procedure from the `DBMS_SERVER_ALERT` package.

❑ You can set thresholds by using the `SET_THRESHOLD` procedure.

❑ You can prevent any database alert from firing, by setting the critical or warning threshold values to NULL.

❑ The procedures in the `DBMS_AQADM` package enable you to access the alerts in the `ALERT_QUE`.

❑ The `REGISTER, DEQUEUE`, and other procedures of the `DBMS_AQ` package help you manage alert notifications.

❑ Database metrics history is maintained by the `DBA_HIST_*` dictionary views.

❑ The `DBA_THRESHOLD` dictionary view provides current threshold settings for all alerts.

❑ The `DBA_OUTSTANDING_ALERTS` view stores information about all *pending* alerts, and the `DBA_ALERT_HISTORY` view provides a history of all *resolved* alerts.

The Automated Tasks Feature

- ❑ The Oracle Scheduler is at the heart of the automated tasks feature.
- ❑ A *program* consists of metadata about a task.
- ❑ A *job* is any executable program.
- ❑ A *schedule* sets the execution time and frequency for a job.
- ❑ A *window* is a time period during which you schedule a job.
- ❑ By default, Oracle comes with a maintenance window, the `GATHER_STATS_PROG` program, the `GATHER_STATS_JOB` job, and the `AUTO_TASKS_JOB_CLASS` job class.

The Management Advisory Framework

- ❑ The management advisory framework provides performance details about various subsystems of the database.
- ❑ All management advisors get their raw data from the AWR repository and store their analysis in the AWR as well.
- ❑ The main advisors are the ADDM, Buffer Cache Advisor, Library Cache Advisor, SQL Access Advisor, SQL Tuning Advisor, Segment Advisor, and Undo Advisor.
- ❑ You can use the `DBMS_ADVISOR` package to run any of the advisors.
- ❑ To use an advisor, you create a task, define the task parameters, execute the task, and view the recommendations.

SELF TEST

The following questions will help you measure your understanding of the material presented in this chapter. Read all the choices carefully because there might be more than one correct answer. Choose all correct answers for each question.

Types of Oracle Statistics

1. What type of statistic is *total logons*?

A. Cumulative statistic
B. Baseline statistic
C. Database metric
D. Type of sample data

2. Which of the following statements is true?

A. Base statistics are derived from database metrics.
B. Metrics are derived from base statistics.
C. The number of physical reads in the database is a metric.
D. Cumulative statistics are derived from database metrics.

3. Which of the following Oracle background processes is responsible for updating metric statistics?

A. MMAN
B. MMNL
C. MMON
D. MMMN

Automatic Workload Repository (AWR)

4. What is the persistent portion of the AWR represented by?

A. The statistics stored in the SGA by the AWR
B. Statistics shown by the `V$` dynamic performance views
C. The AWR snapshots
D. Active Session History data

5. Which is the following is true about snapshots belonging to a baseline?

A. They are retained permanently in the database (never deleted).
B. They are deleted when you drop the underlying baselines, provided you specify the `CASCADE=>TRUE` option.

C. They are deleted immediately after you use them.

D. They are deleted when you create any new snapshots.

6. Your SYSAUX tablespace is under space pressure. The AWR currently has data ranging from snap ID 101 to 322.Which of the following scenarios is the likeliest outcome?

A. Oracle will delete everything from the SYSAUX tablespace except the information for the snap ID range 101 to 322.

B. Oracle will delete snapshot 101.

C. Oracle will delete snapshot 322.

D. Oracle will send an error message saying that it can't create any more new AWR snapshots.

Active Session History (ASH)

7. Which of the following two statements regarding the `V$ACTIVE_SESSION_HISTORY` and `DBA_HIST_ACTIVE_SESS_HISTORY views` are correct?

A. The `V$ACTIVE_SESSION_HISTORY` is a collection of snapshots from the `DBA_HIST_ACTIVE_SESS_HISTORY` view.

B. The `DBA_HIST_ACTIVE_SESS_HISTORY` view is a collection of snapshots from the `V$ACTIVE_SESSION_HISTORY` view.

C. The `V$ACTIVE_SESSION_HISTORY` view shows only the current active session history.

D. The `V$ACTIVE_SESSION_HISTORY` view shows recent active session history.

8. Which of the following statements regarding how ASH data is flushed are true?

A. The MMON process flushes ASH data to disk whenever the rolling buffer in memory is full.

B. The MMNL process flushes ASH data to disk whenever the rolling buffer in memory is full.

C. The MMON process flushes ASH data to disk automatically every 60 minutes.

D. The MMNL process flushes ASH data to disk automatically every 60 minutes.

9. What will the AWR save?

A. All the ASH data

B. A part of the ASH data

C. All the active session ASH data

D. All the inactive session ASH data

Server-Generated Alerts

10. Stateful alerts are those alerts that pertain to thresholds. Stateless alerts pertain to problem alerts. Which of the following two statements are true?

A. Stateful alerts are first logged in the `DBA_ALERT_HISTORY` view.

- B. Stateful alerts are first recorded in the DBA_OUTSTANDING_ALERTS view.
- C. Stateless alerts are never recorded in the DBA_OUTSTANDING_ALERTS view.
- D. Stateful alerts are never recorded in the DBA_OUTSTANDING_ALERTS view.

11. Threshold alerts are stateful alerts. What happens when these occur?

- A. Oracle will automatically clear the alert when you take care of the alert condition.
- B. You need to manually clear the alert after you take care of the alert condition.
- C. The alert can't be cleared automatically or manually.
- D. The alert is automatically recorded in the DBA_ALERT_HISTORY view.

12. Which three of the following statements are correct?

- A. The snapshot too old alert is a stateful alert.
- B. The snapshot too old alert is a stateless alert.
- C. The snapshot too old alert is an out-of-the-box server-generated alert.
- D. All stateless alerts are recorded directly in the DBA_ALERT_HISTORY view.

The Automated Tasks Feature

13. Which two of the following statements are correct?

- A. The MMON background process saves metrics in the SGA.
- B. The AWR snapshot mechanism saves the in-memory metrics to disk.
- C. The AWR snapshots save metrics in the SGA.
- D. The MMON background process flushes metrics to the disk every hour.

14. You can set up thresholds on metrics using which view?

- A. V$SYSMETRIC
- B. V$DBMS_SERVER
- C. DBA_SERVER
- D. V$DBMS_SERVER_ALERT

14. When is subscribing to the ALERT_QUE necessary?

- A. If you wish to use the server-generated alert system of the OEM Database Control
- B. If you wish to set up a notification system for server-generated alerts
- C. If you wish to set up your own tool to display alerts, instead of using the OEM Database Control interface
- D. For only critical alerts

16. Which response best describes a resource plan?

A. Mandatory for every Job Scheduler window

B. Optional for any Job Scheduler window

C. Can't be used with the Job Scheduler

D. Can only be used if you also use a Job Class item

The Management Advisory Framework

17. Which of the following can you use to manage the database advisory framework?

A. `DBA_ADVISOR`

B. `DBMS_ADVISOR`

C. `V$ADVISOR`

D. `DBMS_MANAGEMENT_ADVISORY`

18. What is the first step in using PL/SQL packages to manage the advisory framework?

A. Create a task report

B. Create an advisory task

C. Execute the advisory task

D. Set the task parameters

19. What privilege do you need to execute any advisor procedures?

A. EXECUTE ANY ADVISOR

B. SELECT_ANY_CATALOG

C. ADVISOR

D. SCHEMA OWNER

20. Which of the following two statements are true regarding the SQL Access Advisor?

A. It advises about the use of indexes and materialized views.

B. It provides SQL tuning advice.

C. It provides both SQL tuning and object access advice.

D. It can be called by the ADDM.

LAB QUESTION

Use the manual method to get the ADDM results for a pair of snapshots gathered by the AWR. After the ADDM analysis, show how you would get the report. (Hint: Use the `DBMS_ADVISOR` package.)

SELF TEST ANSWERS

Types of Oracle Statistics

1. ☑ **A.** The number of total logons is a cumulative statistic, because it counts all the logons since you started the instance.
   ☒ **B, C,** and **D** are wrong answers, since none of them is a cumulative statistic.

2. ☑ **B.** Metrics are derived statistics, and their source is the base statistics.
   ☒ **A** is wrong because base statistics aren't derived from database metrics; it's the other way around. **C** is wrong because the number of physical reads is a cumulative statistic, not a metric, which measures the rate of change, not the absolute size of a statistic. **D** is wrong because cumulative statistics are derived from base statistics.

3. ☑ **C.** The MMON process updates the metric statistics.
   ☒ **A, C,** and **D** point to the wrong or a nonexistent background processes.

Automatic Workload Repository (AWR)

4. ☑ **C.** AWR snapshots capture the persistent portion of the AWR data.
   ☒ **A** is wrong because statistics stored in the SGA aren't persistent. **B** is wrong because the `V$` views show only the in-memory statistics. **D** is wrong because ASH data represents only temporary, in-memory data.

5. ☑ **B.** The set of snapshots belonging to a baseline are automatically deleted when you drop that baseline.
   ☒ **A** is wrong because snapshots can be deleted. **C** is wrong because there is no automatic mechanism to drop the snapshots after using them. **D** is wrong because there is no connection between creating new snapshots and dropping old ones.

6. ☑ **B.** When you are under space pressure, Oracle ignores the default retention policy and deletes the oldest snapshots first, to make room for new data. Snap ID 101 happens to be the oldest in this example.
   ☒ **A** is wrong because Oracle deletes only the old snapshots first, not other data from the SYSAUX tablespace. **C** is wrong because the snapshot with the ID 322 is the most recent, not the oldest snapshot in the AWR. **D** is wrong because while Oracle sends an error message, the message is to inform you that there is a space problem in the SYSAUX tablespace.

Active Session History (ASH)

7. ☑ **B** and **D. B** is correct because the DBA_HIST_ACTIVE_SESS_HISTORY view shows a sample of the contents of the V$ACTIVE_SESSION_HISTORY view. **D** is correct because the V$ACTIVE_SESSION_HISTORY view shows recent active session history.
☒ **A** is wrong because the V$ACTIVE_SESSION_HISTORY view is the source for the DBA_HIST_ACTIVE_SESS_HISTORY view, not the other way around. **C** is wrong since the view shows only *recent,* not *current* active session history.

8. ☑ **B** and **C.** The MMNL process flushes AWR data only when the memory buffers are full. **C** is correct because the MMON background process is responsible for automatic flushing of ASH data every 60 minutes, by default.
☒ **A** is wrong since it is the MMNL process, not MMON, that flushes data to disk when the memory buffers are full. **D** is wrong because the MMNL doesn't flush on schedule to disk—it is the MMON process that performs the periodic flushing to disk.

9. ☑ **B.** The AWR saves only a sample of the ASH data.
☒ **A** is wrong because the AWR doesn't store all of the ASH data. **C** is wrong because this is similar to A. **D** is wrong because the ASH facility doesn't capture inactive session data.

Server-Generated Alerts

10. ☑ **B** and **C. B** is correct because all stateful (problem) alerts are recorded in the DBA_OUTSTANDING_ALERTS view first. **C** is correct because the same isn't true for stateless alerts—they aren't recorded in the DBA_OUTSTANDING_ALERTS view.
☒ **A** is wrong because stateful alerts go into the DBA_ALERT_HISTORY view only after they are resolved, not when they are first reported. **D** is wrong since **B** is correct.

11. ☑ **A.** When you clear a stateful (problem) error, Oracle will clear the alert automatically.
☒ **B** is wrong since you don't need to manually clear alerts. **C** is wrong because the alerts are cleared automatically. **D** is wrong because the alerts are recorded in the DBA_OUTSTANDING_ALERTS view first, not the DBA_ALERT_HISTORY view.

12. ☑ **B, C,** and **D. B** and **C** are correct because the snapshot too old alert is an out-of-the-box, server-generated alert and is considered stateless. **D** is correct because stateless alerts are directly written to the DBA_ALERT_HISTORY view.
☒ **A** is wrong because the error is stateless.

13. ☑ **B** and **D.** The AWR snapshots are the means through which the MMON process flushes metrics to disk on an hourly basis.

☒ **A** is wrong because the MMON process isn't responsible for in-memory statistics. **C** is wrong because the AWR doesn't save metrics in the SGA—it saves them on disk.

The Automated Tasks Feature

14. ☑ **D.** The `DBMS_SERVER_ALERT` view enables you to set up thresholds on metrics, using the `SET_THRESHOLD` procedure.
☒ **A, B,** and **C** point to the wrong views.

15. ☑ **C.** You must use the `ALERT_QUE` mechanism only if you want to set up your own tool to display alerts.
☒ **A** is wrong because you don't need to subscribe to the `ALERT_QUE` mechanism if you wish to use server-generated alerts through the Database Control. The OEM Database Control is already a registered for the `ALERT_QUE`. **B** is wrong because you don't need to set up alert notification. **D** is wrong because the `ALERT_QUE` doesn't have anything to do with critical alerts.

16. ☑ **B.** A resource manager is usually associated with a window, but a window doesn't have to use it.
☒ **A** is wrong because resource plans aren't mandatory. **C** is wrong because you can use resource plans with the Scheduler. **D** is wrong since you don't have to specify a Job Class in order to use a resource plan with the Scheduler.

The Management Advisory Framework

17. ☑ **B.** You must use the `DBMS_ADVISOR` package to manage the advisory framework.
☒ **B, C,** and **D** point to wrong or nonexistent views and packages.

18. ☑ **B.** The first step in using the `DBMS_ADVISOR` package, which helps you manage the advisory framework, is to create an advisory task.
☒ **A, C,** and **D** present the various steps in the wrong sequence.

19. ☑ **C.** You must have the ADVISOR privilege to execute any advisor privileges in the database.
☒ **A, B,** and **D** provide wrong or nonexistent privileges.

20. ☑ **A** and **D. A** is correct because the SQL Access Advisor advises you about the best access to objects, including using indexes. **D** is correct since the ADDM can call any of the management advisors to provide an in-depth analysis of a performance problem.
☒ **B** is wrong since the SQL Access Advisor doesn't provide any direct SQL tuning advice. **C** is wrong for the same reason that B is a wrong answer.

LAB ANSWER

First, define the bind variables:

```
SQL> VARIABLE taskname VARCHAR2(40)
SQL> VARIABLE taskid NUMBER
```

Next, create the advisory task:

```
SQL> exec dbms_advisor.create_task('ADDM', :taskid, :taskname);
```

Then set the task parameters:

```
SQL> exec dbms_advisor.set_taskparameter(:taskname, 'START_
SNAPSHOT',20);
SQL> exec dbms_advisor.set_task_parameter():taskname, 'END_
SNAPSHOT',24);
```

Execute the task:

```
SQL> exec dbms_advisor.execute_task(:taskname);
```

Finally, get the analysis results:

```
     SQL> select dbms_advisor.get_task_report(:taskname)
     from dba_advisor_tasks t
     where t.task_name=:taskname AND
     t.owner=SYS_CONTEXT('USERENV', 'SESSION_USER');
```