3 Automatic Database Management

CERTIFICATION OBJECTIVES

3.01	Use Automatic Database Diagnostic Monitor
3.02	Use Automatic Shared Memory Management
3.03	Use Automatic Optimizer Statistics Collection
3.04	Use Automatic Undo Retention Tuning
✓	Two-Minute Drill
Q&A	Self Test

You have probably heard a lot of talk about Oracle Database 10*g* being a self-managing database. Far from being hype, Oracle Database 10*g* offers you a dazzling array of self-managing features that will help you perform difficult tuning chores effortlessly. Not only that, the database now provides you with ways to implement your performance fixes. As practicing Oracle DBAs, we all know how difficult it can be to figure out what is causing a performance problem. Accurate and quick diagnosis of performance issues is critical to the success of any Oracle database professional. Oracle DBAs routinely undertake tuning exercises that involve running SQL scripts, poring over extended trace results, using the EXPLAIN PLAN statement, using the TKPROF utility, and/or employing other tuning aids.

For the first time, Oracle Database 10*g* helps you automatically diagnose performance problems. Diagnosing and fixing performance problems may be as simple as going to the OEM Database Control and selecting one or more of the recommendations. Often, it is hard for you to reproduce a performance problem after it has already occurred. Do you recall how often the Oracle*MetaLink* service asks you to see if you can "reproduce" a performance problem? In Oracle Database 10*g*, the database collects and saves all the statistical data you need for a performance diagnosis in the *Automatic Workload Repository* (AWR).

A new diagnosis tool, the *Automatic Database Diagnostic Monitor* (ADDM) analyzes this data regularly, to provide you with detailed information about the root cause of performance problems, as well as recommendations for how to fix the problem. The ADDM relies on years of Oracle Corporation's performance methods, thus making it a sort of an expert system or a self-diagnostic engine that is built right into Oracle Database 10*g*. Since the performance statistics collection mechanism relies on the Oracle System Global Area (SGA), it is very accurate and efficient. The ADDM uses a new statistics collection method, where statistics are read directly from the SGA structures by a new background process called the Manageability Monitor process (MMON). Hence, there is none of the performance overhead of a session populating a `V$` view.

Previous versions of Oracle provided you with ways to manage the critical SGA on a dynamic basis. The ability to dynamically modify SGA component sizes means that you don't need to restart an instance after you make the changes. Now, Oracle has gone a step further and provides you with the option of using *Automatic Shared Memory Management*. With automatic SGA management, you can realize the twin tasks of conserving memory while improving database performance by automatically provisioning SGA memory based on database workload, not tedious guesswork.

Practicing Oracle DBAs know how critical it is to collect timely database statistics so the Oracle optimizer performs efficiently. In Oracle Database 10*g*, there is a new feature, the *automatic optimizer statistics collection*, in which Oracle itself automatically schedules the statistics collection, using the new database Scheduler feature. DBAs have always been beset with the problem of which tables to analyze and how often. Now, in Oracle Database 10*g*, you simply trust Oracle to decide for you.

In Oracle9*i*, you first encountered A*utomatic Undo Management* (AUM). In Oracle Database 10*g*, there are enhancements to this feature, including the interesting new option of guaranteed undo retention. Using this new undo feature, you can virtually rule out any ORA-1555 (snapshot too old) errors.

In this chapter, our focus is on the exciting automatic management features introduced in the Oracle Database 10*g* Server. Let's start our discussion of Oracle Database 10*g*'s automatic management features with a description of the ADDM feature.

- Using the Automatic Database Diagnostic Monitor (ADDM)
- Using Automatic Shared Memory Management
- Using automatic optimizer statistics collection
- Using automatic undo retention tuning

CERTIFICATION OBJECTIVE 3.01

Using the Automatic Database Diagnostic Monitor (ADDM)

As an Oracle DBA, you know how difficult it is sometimes to figure out why your database performance isn't up to par. You may need to hunt down the causes, but too often, you are likely to mistake the symptoms for the causes. Correctly diagnosing performance problems is the key to efficient performance tuning.

Traditionally, organizations have spent considerable amounts of effort on performance tuning, which usually tends to be quite laborious and not an exact science. Oracle Database 10*g* changes all that. It provides you with very powerful and accurate automatic performance-tuning capabilities. The heart of the new automatic performance tuning is the new statistics collection facility, the Automatic Workload Repository (AWR), which automatically saves crucial performance information in the new mandatory SYSAUX tablespace.

By default, the AWR collects new performance statistics in the form of a *snapshot* on an hourly basis and saves the snapshots for seven days before purging them. These snapshots of database activity include resource-intensive SQL statements. The Automatic Database Diagnostic Monitor (ADDM) runs automatically every hour, after the AWR takes a new snapshot. The ADDM uses the AWR performance snapshots to locate the root causes for poor performance and provides recommendations for improving performance.

The AWR saves all historical performance data, so you don't need to worry about being able to reproduce a performance problem. Every time the AWR takes a snapshot of the database, which is hourly by default, the ADDM runs automatically and analyzes the key data gathered by the AWR. You can then go to the OEM Database Control to view the results, or even view them from a SQL*Plus session with the help of an Oracle-supplied SQL script.

The ADDM runs automatically, although you can also manually invoke the tool to investigate problems that occur in between the scheduled snapshots. As the DBA, you are relieved of the responsibility of catching a problem at the right time to collect statistic, since the ADDM automatically analyzes performance data for you.

exam watch

The results of the AWR statistics collection, including the snapshots, are stored in the SYSAUX tablespace. Oracle stores the ADDM analyses in the same tablespace as well.

The Goal of the ADDM

The entire rationale behind the work of the ADDM is to reduce a key database metric called *DB time*, which stands for database time (in microseconds) spent *actually processing user's requests*. DB time is the total time the database is spending on processing user requests.

The DB Time Variable

The DB time variable includes only the cumulative amount of time spent on actual database calls (at the user level) and doesn't include time spent on background processes. DB time includes both the wait time and processing time (CPU time). DB time doesn't include the idle time incurred by your processes. For example, if you spend 30 minutes connected to the database and you're idle for 28 of those minutes, then DB time is only 2 minutes.

If a problem is contributing to inappropriate or excessive DB time, the ADDM automatically flags it as an issue that needs your attention. If there is a problem in your system, but it doesn't contribute significantly to the DB time variable, the ADDM will simply ignore the problem. Thus, the entire functioning of the ADDM revolves around the single mantra: *reduce DB time*. By relentlessly focusing on the reduction of database time (DB time), the ADDM's aim is to increase the *throughput* of your database, thus serving more users with the same amount of resources.

Problems That the ADDM Diagnoses

The ADDM analyzes the AWR snapshots periodically and comes up with performance recommendations, usually quantified in terms of expected benefit of various actions. Following are some of the key problems that the ADDM diagnoses:

- Configuration issues
- Improper application usage
- Expensive SQL statements
- I/O performance issues
- Locking issues
- Excessive parsing
- CPU bottlenecks
- Undersized memory allocation
- Connection management issues, such as excessive logon/logoff statistics

You may be wondering why you shouldn't just use the well-known STATSPACK utility to gather performance-related data. For one thing, STATSPACK has too much information, not all of which is critically important to fixing critical performance problems that are occurring *right now*. The ADDM uses a sophisticated, new time statistics model in Oracle Database 10*g*, which is highly effective in determining where time is spent in the database. This new time statistics model enables Oracle to focus on only the most critical performance problem areas. DB time is the key metric against which the ADDM judges all performance problems. If a problem exceeds the threshold for DB time, the ADDM tags it as a top performance issue; otherwise, it leaves it alone as nonproblem area.

The New Time Model

The ADDM bases most of its performance recommendations on the basis of *time model statistics*, the most important of which is the new DB time statistic. The DB time statistic, as I explained in the previous section, represents the true workload of your database, because it shows the total time spent in making database calls. In addition to DB time, time model statistics also provide timings relating to logon statistics and parse activity. Using decades of its performance-tuning expertise, Oracle has come up with a new and more accurate time model to accurately diagnose performance issues.

Two new database views, V$SESS_TIME_MODEL and V$SYS_TIME_MODEL, help you to manage these new time-based performance statistics.

The V$SYS_TIME_MODEL view provides the accumulated time statistics for various operations in the *entire* database. This view shows time in terms of the number of microseconds the database has spent on a specific operation. The following query demonstrates the kind of operations for which the V$SYS_TIME_MODEL view holds statstics.

```
SQL> select  stat_name ,value from V$SYS_TIME_MODEL
STAT_NAME                                                             VALUE
---------------------------------------------------------------- ----------
DB time                                                          3175312459
DB CPU                                                            473486465
background elapsed time                                          7152613400
background cpu time                                               445371822
sequence load elapsed time                                          2780225
parse time elapsed                                               1003246942
hard parse elapsed time                                           969141891
sql execute elapsed time                                         2921215454
connection management call elapsed time                            49093303
failed parse elapsed time                                            278729
failed parse (out of shared memory) elapsed time                          0
hard parse (sharing criteria) elapsed time                          7446158
hard parse (bind mismatch) elapsed time                             3152674
PL/SQL execution elapsed time                                     327980460
inbound PL/SQL rpc elapsed time                                           0
PL/SQL compilation elapsed time                                   267986126
Java execution elapsed time                                               0
17 rows selected.
SQL>
```

The V$SESS_TIME_MODEL view is similar to the V$SYS_TIME_MODEL view and provides the same types of time statistics, but at a *session* level, instead of the system level.

Benefits Provided by the ADDM

The ADDM bases its recommendations on a holistic approach, with time spent on database activities as its main focus. Here are some of the important benefits of using the ADDM:

- It identifies the root causes of performance problems, instead of merely focusing on the symptoms. The ADDM will automatically capture highly resource-intensive SQL statements.
- It produces automatic performance diagnostic reports at periodic intervals.
- You'll experience very little performance overhead when using the tool. A typical ADDM analysis takes only three or four seconds.
- The ADDM points out nonproblem areas, so you don't waste your efforts poking around in areas with little bang for the buck.
- Performance diagnosis is based on decades' worth of Oracle's expert tuning knowledge.

Types of ADDM Recommendations

The ADDM may propose several recommendations for the same performance problem. The recommendations may include the following:

- **Hardware changes** ADDM may recommend that you add more CPUs to your system. It may also recommend that you change the way you configure your I/O subsystem.
- **Database and application changes** The ADDM may find, for example, that your database is performing an excessive amount of parses due to the failure to use bind variables. In a case like this, it may recommend that you change your initialization parameter `CURSOR_SHARING` to a setting of `FORCE`, rather than rewrite your application code. In some other cases, the ADDM may recommend that you go ahead and rewrite the application code so you use bind variables.
- **Space configuration changes** The ADDM may sometimes make recommendations like switching to the new Automatic Storage Management (ASM) to fix certain performance problems.
- **Using management advisors** The ADDM may recommend several changes that you can implement immediately to improve database performance.

However, in some cases, it may recommend that you use a management advisor—like the SQL Tuning Advisor, Undo Advisor, or Segment Advisor—to gain an in-depth knowledge of the performance problems. For example, the ADDM may tag certain high-load SQL statements as candidates for automatic SQL tuning, using the Automatic Tuning Optimizer (I'll explain this in Chapter 5).

Automatic Management of the ADDM

Oracle manages the ADDM with the help of a brand-new background process in Oracle Database 10*g* databases: the MMON. The MMON process schedules the automatic running of the ADDM. Each time the AWR takes a snapshot (every 60 minutes, by default), the MMON process asks the ADDM to analyze the interval between the last two snapshots it gathered. This is the default behavior of the ADDM performance analysis.

Where does the ADDM store its analysis results? Not surprisingly, the ADDM stores the results in the AWR itself. You can use the OEM Database Control to view the ADDM's performance analysis and action recommendations.

Configuring the ADDM

It might surprise you to find out that you don't need to go through an arduous setup or configuration exercise for using the amazingly powerful ADDM feature. Oracle enables the ADDM feature by default. Your only task is to make sure that the initialization parameter `STATISTICS_LEVEL` is set to `TYPICAL` or `ALL`, in order for the AWR to gather its cache of performance statistics. If you set `STATISTICS_LEVEL` to

exam watch

You must set the initialization parameter `STATISTICS_LEVEL` to `TYPICAL` or `ALL` in order to use the ADDM. If you set `STATISTICS_LEVEL` to `BASIC`, you cannot use the ADDM. In addition, you won't be able to use several other important Oracle diagnostic features. For example, if `STATISTICS_LEVEL` is set to `BASIC`, you must explicitly set the `TIMED_STATISTICS` parameter to `TRUE` in order to collect timed performance statistics. Oracle strongly recommends the use of the `TYPICAL` setting, which it deems sufficient for all your day-to-day needs. `BASIC` cripples your tuning efforts, while `ALL` will have a performance impact on your system.

BASIC, you can still manually use the AWR to collect statistics by using the DBMS_WORKLOAD_REPOSITORY package. However, you will not be able to collect several important types of performance statistics.

You can control the amount of statistics collected by the AWR by adjusting either or both of two variables:

- **Snapshot interval** The default snapshot interval is 60 minutes, and Oracle recommends that you use the default interval for your everyday needs. Remember that flushing statistics involves a performance hit, however minor. Oracle's assumption is that once an hour is frequent enough for diagnosis and infrequent enough not to influence performance. The DBA must decide whether he or she agrees with this assumption.
- **Snapshot-retention period** The snapshot-retention period is the length of time for which the AWR retains all snapshots. The default retention period is 7 days. After the snapshot-retention period expires, Oracle will automatically purge the outdated snapshots from the AWR.

You can modify the length of the snapshot interval and snapshot retention by using the INTERVAL and the RETENTION parameters of the MODIFY_SNAPSHOT_SETTINGS of the DBMS_WORKLOAD_REPOSITORY package. Chapter 4 provides examples showing you how to manage AWR historical data retention by modifying the INTERVAL and RETENTION parameters.

on the job

The ADDM runs automatically after each AWR snapshot. If you don't like the default (60-minute) interval, you can change it.

Determining Optimal I/O Performance

How does the ADDM know what is *optimal* I/O performance? If your I/O system performs at a certain speed, your system can read a database block in a specific number of milliseconds. The ADDM makes the critical assumption that the average time to read a database block is 10 milliseconds. The DBIO_EXPECTED parameter (not an initialization parameter) indicates your I/O performance, and by default, Oracle assumes this parameter's value is 10 milliseconds.

You can find out the current value of the DBIO_EXPECTED parameter by querying the DBA_ADVISOR_DEF_PARAMETERS view in the following way:

```
SQL> select parameter_value
  2  from dba_advisor_def_parameters
where advisor_name='ADDM'
```

```
  4* AND parameter_name='DBIO_EXPECTED';
PARAMETER_VALUE
----------------
10000
SQL>
```

What do you do if you think your I/O subsystem is fast, and performs a database block in only 6 milliseconds, or is slow and takes longer? You can use the DBMS_ADVISOR package to change the default value of the DBIO_EXPECTED parameter, as I'll explain later in this chapter, in the "Using the DBMS_ADVISOR Package to Manage the ADDM" section.

Running the ADDM

The new Oracle background process, MMON, schedules the ADDM to run every time the AWR collects its most recent snapshot. Thus, Oracle will automatically generate ADDM reports throughout the day.

You can also perform an ad hoc ADDM analysis any time, to find out details about a performance problem that's currently occurring in the database. Remember that the AWR will be taking periodic snapshots of database performance statistics throughout the day. You can request that the ADDM analyze the data that falls between any two snapshots. Note that the two beginning and ending snapshots don't need to be consecutive. The only requirements regarding the selection of the AWR snapshots are the following:

- The snapshots must be clean, without any errors.
- There shouldn't be a database shutdown during the two snapshots.

The easiest way to view the ADDM's findings is to use the OEM Database Control. You can get to the ADDM by clicking the Advisor Central link first, and then choosing ADDM. Once you reach the ADDM page, you can view the latest performance findings or start a new ADDM task.

exam watch

Oracle runs the ADDM automatically every hour, following the AWR snapshot collection. You may, however, run it manually whenever you choose. You may want to run the ADDM manually either because an alert recommends that you do so or because you want an ADDM analysis across multiple snapshots.

You can also obtain complete ADDM reports by running the Oracle-provided SQL script addmrpt.sql, which you'll find in your ORACLE_HOME/rdbms/admin directory. As explained earlier, the AWR will take snapshots of database performance at specified intervals. Assuming the default of one-hour intervals, you'll have as many snapshots in the AWR as the number of hours since you started the database instance.

In the following example, the database was started sometime after 10:00 A.M. and there are four consecutive hourly snapshots in the system. Note that there is a snapshot captured at 9:57 A.M., but I brought down the database after that. You can compare two snapshots only if you don't shut down the database in between. Why is this so? The AWR holds only cumulative database statistics. Obviously, once you shut down the database, all the cumulative data will lose its meaning.

In this example, I show how to get the ADDM report for the period between 10:00 A.M. and 1:00 P.M. To do so, I need to specify the snapshot numbers pertaining to the 10:00 A.M. and 1:00 P.M. snapshot collection times. The addmrpt.sql script provides this information. In the script, notice that the snapshot IDs 258 (captured at 10:00 A.M.) and 261 (captured at 1:00 P.M.) will contain the performance statistics for the period between 10:00 AM and 1:00 PM. Therefore, I provide the snapshot IDs 258 and 261 in response to the prompts for the beginning and ending snapshot IDs.

```
SQL> @c:\oracle\product\10.1.0\Db_1\RDBMS\ADMIN\addmrpt.sql
   DB Id    DB Name      Inst Num Instance
----------- ------------ -------- ------------
  877021568 NINA                1 nina
Specify the number of days of snapshots to choose from
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Entering the number of days (n) will result in the most recent
(n) days of snapshots being listed.  Pressing <return> without
specifying a number lists all completed snapshots.
Listing the last 3 days of Completed Snapshots
                                                            Snap
Instance     DB Name        Snap Id    Snap Started    Level
------------ ------------ --------- ------------------ -----
nina         NINA      257 21 Apr 2004 09:27      1
                                258 21 Apr 2004 10:00      1
                                259 21 Apr 2004 11:00      1
                                260 21 Apr 2004 12:00      1
                                261 21 Apr 2004 13:00      1
Specify the Begin and End Snapshot Ids
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Enter value for begin_snap: 258
Begin Snapshot Id specified: 258
Enter value for end_snap: 261
End   Snapshot Id specified: 261
```

```
Specify the Report Name
~~~~~~~~~~~~~~~~~~~~~~~
The default report file name is addmrpt_1_258_261.txt.  To use this name
press <return> to continue, otherwise enter an alternative.
Enter value for report_name:
Using the report name addmrpt_1_258_261.txt
Running the ADDM analysis on the specified pair of snapshots ...
```

For space reasons, I am not showing the entire ADDM report. The textual report that the addmprt.sql generates is identical to the detailed report that you can get using the OEM Database Control's ADDM page. You'll see this report later in this chapter, in the section "Viewing Detailed ADDM Reports."

on the job

The ADDM doesn't always offer you a direct recommendation for a performance problem that it encounters. A key aspect of an ADDM analysis is that. In many cases, it recommends that you use one of Oracle's built-in advisors, like the SQL Tuning Advisor, to analyze a complex performance situation.

The ADDM Analysis

The ADDM presents the results of its analysis to you in a standard format each time. Each ADDM analysis finding consists of the following four components:

- The definition of the problem itself
- The root cause of the performance problem
- Recommendation(s) to fix the problem
- The rationale for the proposed recommendations

By navigating to the Performance Details page of the ADDM using the OEM Database Control, you can see all the problems and the recommendations for fixing them. For each problem in the report, ADDM displays each of its performance findings in the form of three columns: the Impact column, the Finding column, and the Recommendations column.

The Impact column lists the performance problems in the order of their impact on their system. The Impact column is thus very important, because you can start working on fixing the most serious problem that is affecting current database performance. You may think parsing issues are more serious than, say I/O-related issues, but if the Impact column ranks I/O problems as number one, you should attend to the I/O problems first.

on the job

Oracle enables the ADDM by default, as long as you set the `STATISTICS_LEVEL` parameter to `TYPICAL` or `ALL`. You can shut the ADDM down by simply setting the `STATISTICS_LEVEL` parameter to `BASIC`. However, remember that this step will also disable many automatic performance tuning and statistics gathering activities of Oracle Database 10g.

Let's look at a typical ADDM problem analysis and list of recommendations. This analysis is from the OEM Database Control's ADDM page.

Problem	Recommendation
The buffer cache was undersized causing significant additional read I/O.	Increase SGA target by increasing the value of parameter `SGA_TARGET` by 256M.
The throughput of the I/O subsystem was significantly lower than expected.	Consider increasing the throughput of the I/O subsystem. Oracle's recommended solution is to stripe all datafiles using the *same* methodology. You might also need to increase the number of disks for better performance. Alternatively, consider using Oracle's ASM solution.
Hard parsing of SQL statements was consuming significant database time.	Here, the ADDM did not make any recommendations. By clicking the Additional Information button, I found that ADDM didn't see any reason to tinker with the parsing issue, for the following reasons: Hard parses due to cursor environment mismatch were not consuming significant database time. Hard parsing SQL statements that encountered parse errors was not consuming significant database time. Parse errors due to inadequately sized shared pool were not consuming significant database time. Hard parsing due to cursors getting aged out of shared pool was not consuming significant database time. Hard parses due to literal usage and cursor invalidation were not consuming significant database time.
Database writers (DBWR) were unable to keep up with the demand for free buffers.	Consider increasing the number of database writers (DBWR) by setting the parameter `DB_WRITER_PROCESSES`.
Time spent on the CPU by the instance was responsible for a substantial part of database time.	Tune the PL/SQL block with SQL_ID "2b064ybzkwf1y." Refer to the "Tuning PL/SQL Applications Chapter of Oracle's "PL/SQL User's Guide and Reference."

Viewing Detailed ADDM Reports

If you want to view a Detailed ADDM Report for any given ADDM task, all you need to do is click the View Report button on the ADDM main page in the Database Control. You can then view the results, or if you wish, save the results to a file or print the report.

Note that in addition to the information about impact, problem, and recommendations that you can gain from the main ADDM page, the detailed report includes a listing of the symptoms that led to each particular finding. In addition, for some problems, the ADDM report also includes a Rationale section that details the reasoning for its action recommendations.

Here is an example of a detailed report:

```
DETAILED ADDM REPORT FOR TASK 'ADDM:877021568_1_259' WITH ID 1052
          -----------------------------------------------------------------
               Analysis Period: 21-APR-2004 from 10:00:08 to 11:00:40
         Database ID/Instance: 877021568/1
      Database/Instance Names: NINA/nina
                    Host Name: NTL-ALAPATISAM
             Database Version: 10.1.0.2.0
               Snapshot Range: from 258 to 259
                Database Time: 357 seconds
        Average Database Load: .1 active sessions
-----------------------------------------------------------------------------
FINDING 1: 87% impact (311 seconds)
-----------------------------------------------------------------------------
The throughput of the I/O subsystem was significantly lower than expected.
    RECOMMENDATION 1: Host Configuration, 87% benefit (311 seconds)
      ACTION: Consider increasing the throughput of the I/O subsystem.
         Oracle's recommended solution is to stripe all data file using the
         SAME methodology. You might also need to increase the number of disks
         for better performance. Alternatively, consider using Oracle's
         Automatic Storage Management solution.
    SYMPTOMS THAT LED TO THE FINDING:
      Wait class "User I/O" was consuming significant database time. (100%
      impact [467 seconds])
```

Notice that Recommendation 1 is shown as providing an "87% benefit." *Benefit* here refers to the reduction in DB time due to the recommendation's implementation. Thus, the ADDM is telling you that you can save a maximum of up to 87 percent of the total DB time by following the proposed solution.

At the end of the Detailed ADDM Report, you'll see a section called Additional Information, which usually shows insignificant wait information. Here is a typical list of findings under the Additional Information heading:

```
Wait class "Administrative" was not consuming significant database time.
Wait class "Application" was not consuming significant database time.
Wait class "Cluster" was not consuming significant database time.
Wait class "Commit" was not consuming significant database time.
Wait class "Concurrency" was not consuming significant database time.
```

Using the DBMS_ADVISOR Package to Manage the ADDM

The new DBMS_ADVISOR package helps you manage the attributes of the ADDM tool, as well as perform jobs like creating tasks and retrieving ADDM reports using SQL. The DBMS_ADVISOR package is part of the Server Manageability Suite of advisors, which is a set of rule-based expert systems that identify and resolve performance problems of several database components. This set of advisors helps you manage performance issues relating to various database components in Oracle Database 10*g*. The ADDM is one of this set of advisors, and you'll learn more about the other advisors in later chapters.

on the job

The DBMS_ADVISOR package requires the ADVISOR privilege.

The following are a few of the main procedures and functions of the DBMS_ADVISOR package. Note that these program components apply not just to the ADDM, but also to all the other database advisors.

- You use the CREATE_TASK procedure to create a new advisor task.
- The SET_DEFAULT_TASK procedure helps you modify default values of parameters within a task.
- The DELETE_TASK procedure deletes a specific task from the repository.
- The EXECUTE_TASK procedure executes a specific task.
- The GET_TASK_REPORT displays the most recent ADDM report.
- The SET_DEFAULT_TASK_PARAMETER procedure modifies a default task parameter.

You can use the SET_DEFAULT_TASK procedure to modify default values of ADDM parameters like DBIO_EXPECTED (discussed earlier in the "Determining Optimal I/O Performance" section). The following example illustrates the technique:

```
SQL> sho user
USER is "SYS"
SQL>  exec DBMS_ADVISOR.SET_DEFAULT_TASK_PARAMETER(-
>  'ADDM', 'DBIO_EXPECTED', 6000);
PL/SQL procedure successfully completed.
```

The GET_TASK_REPORT procedure in the DBMS_ADVISOR package enables you to get XML, text, or HTML reports for a specified task, including an ADDM task. Here is the structure of the GET_TASK_REPORT procedure:

```
DBMS_ADVISOR.GET_TASK_REPORT (
   task_name      IN VARCHAR2,
   type           IN VARCHAR2 := 'TEXT',
   level          IN VARCHAR2 := 'TYPICAL',
   section        IN VARCHAR2 := 'ALL',
   owner_name     IN VARCHAR2 := NULL)
RETURN CLOB;
```

The possible values for the TYPE parameter are TEXT, XML, and HTML. The possible values for the LEVEL parameter, which stands for the initialization parameter STATISTICS_LEVEL, are TYPICAL, ALL, and BASIC. Remember that setting the STATISTICS_LEVEL parameter to BASIC will mean that you can't use the ADDM tool.

Here's an example that shows how to use the GET_TASK_REPORT procedure to obtain an ADDM report (I am abbreviating the ADDM report shown in the output; it is the same report you saw earlier through the OEM Database Control and the use of SQL scripts):

```
SQL>  select DBMS_ADVISOR.GET_TASK_REPORT('ADDM:877021568_1_252')
  2  from dba_advisor_tasks
  3  where task_id = (select max(t.task_id)
  4  from dba_advisor_tasks t,
  5  dba_advisor_log l
  6  where t.task_id = l.task_id AND
  7  t.advisor_name='ADDM' AND
  8* l.status='COMPLETED');
DBMS_ADVISOR.GET_TASK_REPORT('ADDM:877021568_1_252')
----------------------------------------------------------------------------
          DETAILED ADDM REPORT FOR TASK 'ADDM: 877021568_1_252' WITH ID 1002
----------------------------------------------------------------------------
Analysis Period: 14-APR-2004 from 05:00:37 to 06:00:07
         Database ID/Instance: 877021568/1
The analysis of I/O performance is based on the default assumption that the
average read time for one database block is 10000 micro-seconds.
DBMS_ADVISOR.GET_TASK_REPORT('ADDM:877021568_1_252')
SQL>
```

To produce this ADDM report, you don't need to specify any snapshots. How then, does Oracle know which snapshots it should consider for preparing its report? Well, when you use the DBMS_ADVISOR package to produce an ADDM report, Oracle will always use the data that it collects between the two most recent snapshots.

exam watch

Remember that there are three ways to retrieve an ADDM analysis report: you can use the OEM Database Control interface, the SQL*Plus interface (by running the addmrpt.sql script), or the `DBMS_ADVISOR` package (by running the `GET_TASK_REPORT` procedure).

Using ADDM-Related Dictionary Views

Following are some of the new views that'll help you in dealing with the ADDM:

- The `DBA_ADVISOR_RECOMMENDATIONS` view shows the results of analyzing all the recommendations in the database.
- The `DBA_ADVISOR_FINDINGS` view shows the findings of all the advisors in your database.
- The `DBA_ADVISOR_RATIONALE` view shows the rationale behind all the recommendations.

CERTIFICATION OBJECTIVE 3.02

Using Automatic Shared Memory Management

Every DBA knows how hard it is sometimes to adjust the SGA, which is the memory that Oracle assigns to every instance to hold data and control information. You may have a situation where online transaction processing (OLTP) transactions dominate the database all day, and you run heavy-duty batch jobs during the night. In cases like this, you may need more allocation for the buffer cache during the daytime and an increase in the large pool component of the SGA for the nightly batch jobs.

You can, of course, dynamically change several SGA components, as well as use scripts to change SGA allocations before and after batch jobs, but the fact remains that it is you, the DBA, who is directly responsible for adjusting the SGA components to match the needs of the database instance. Problems like the ORA-4031 (out of shared pool memory) error are all too common, forcing you to juggle with the manual tuning parameters. You may also face a situation where you might be assigning too much SGA memory, thus wasting precious resources, or too little memory, which affects database performance.

exam watch

In Oracle Database 10g, the database enables the Automatic PGA Memory Management feature by default. However, if you set the `PGA_AGGREGATE_TARGET` parameter to 0 or the `WORKAREA_SIZE_POLICY` parameter to `MANUAL`, Oracle doesn't use Automatic PGA Memory Management.

In Oracle Database 10g, for the first time, you can make the often-tricky issue of shared memory management completely automatic. This is one of the more significant enhancements of Oracle Database 10g, and it contributes significantly to Oracle's goal of automatic database self-management. Oracle will automatically allocate and deallocate memory for each of the memory pools, based on changing database workloads. Oracle will use internal views and statistics to decide on the best way to allocate memory among the SGA components. Automatic Shared Memory Management provides you the following benefits:

- Less chance of running out of shared pool memory
- Optimal use of available memory
- Significant performance improvement because memory allocation keeps step with fluctuations in the database workload

Before we delve into the new Automatic Shared Memory Management feature of Oracle Database 10g, let's quickly review the manual memory management feature, which is still available for you to use.

Manual Shared Memory Management

Oracle will still let you manage the shared memory components manually. Under traditional shared memory management, you need to set several initialization parameters for the various components of the SGA. These dynamic memory parameters are `DB_CACHE_SIZE`, `SHARED_POOL_SIZE`, `LARGE_POOL`, `JAVA_POOL_SIZE`, and `STREAMS_POOL_SIZE`.

You can limit the size of the total amount of memory used by Oracle by setting the `SGA_MAX_SIZE` parameter in your initialization file. When you do this, Oracle will limit the sum of the various components of the SGA to the value you specify. If you don't specify an explicit `SGA_MAX_SIZE` parameter, it defaults to the sum of exam watch

All SGA automatic components behave the same as in previous versions, except the shared pool component, which you set with the initialization parameter `SHARED_POOL_SIZE`. In Oracle Database 10g, the value of the `SHARED_POOL_SIZE` parameter includes the internal overhead allocations for metadata such as the various data structures for sessions and processes. You must, therefore, make sure to increase the size of the `SHARED_POOL_SIZE` parameter when you are upgrading to Oracle Database 10g. You can find the appropriate value of your new `SHARED_POOL_SIZE` parameter before the upgrade by using the following query:

```
Select SUM(bytes)/1024/1024 from V$SGASTAT
Where pool = 'shared pool';
```

the actual size of all the SGA components. If you set the `SGA_MAX_SIZE` to a value smaller than the sum of all the SGA components, Oracle will automatically bump up the `SGA_MAX_SIZE` parameter's value to the sum of the memory assigned to all the components.

Automatic Memory Management

In the manual memory management model, you need to calibrate the individual components of the SGA. To let Oracle automatically manage the memory allocations to the individual components of the SGA, you need to set the new initialization parameter `SGA_TARGET`. By default, the `SGA_TARGET` parameter is set to zero. Once you provide a nonzero value for the `SGA_TARGET` parameter, Automatic Shared Memory Management is enabled.

on the job

In order to use Automatic Shared Memory Management, you should first set the initialization parameter `STATISTICS_LEVEL` to its default value of `TYPICAL` or `ALL`. Oracle doesn't populate the `V$SHARED_POOL_ADVICE` and the `V$DB_CACHE_ADVICE` views if the `STATISTICS_LEVEL` parameter is set to `BASIC`.

The new Oracle Database 10g background process MMAN performs all the memory resizing necessary for the Automatic Shared Memory Management feature. The MMAN process constantly monitors the workload of the database and adjusts the size of the individual memory components accordingly.

Two Sets of SGA Parameters

As you know, Oracle's SGA is not one big chunk of memory. Rather, it consists of several specific components like the buffer cache and shared pool. When you use Automatic Shared Memory Management by setting the `SGA_TARGET` parameter to a nonzero value, the database doesn't manage all of the shared memory components. Although we call it Automatic Shared Memory Management, the SGA has both an automatic and a manual set of components.

Under Automatic Shared Memory Management, the database manages the following four major components of the SGA, also known as the *auto-tuned SGA parameters*:

- Buffer cache (`DB_CACHE_SIZE`)
- Shared pool (`SHARED_POOL_SIZE`)
- Large pool (`LARGE_POOL_SIZE`)
- Java pool (`JAVA_POOL_SIZE`)

It is important to understand that even under Automatic Shared Memory Management, you still need to configure any SGA component other than the four auto-tuned components. Following are the *manually sized components* of the SGA:

- Redo Log Buffer
- The KEEP and RECYCLE buffer caches (if specified)
- The nonstandard block size buffer caches (if specified)
- The new Streams pool SGA component
- The new Oracle Storage Management (OSM) buffer cache, which is meant for the optional ASM instance

Note that in addition to the automatic and manual components, Oracle also assigns a certain amount of memory to the fixed SGA, which contains database and instance state information useful to the background processes. The fixed SGA doesn't contain any user data.

It is important to understand that the `SGA_TARGET` parameter shows the *sum of all SGA components*, not just the automatically managed memory components. Interestingly, even under Automatic Shared Memory Management, the manually sized components get the first crack at the SGA allocated by the `SGA_TARGET` parameter. Oracle will first subtract the total value of all the manually sized memory components from `SGA_TARGET`, and then allocate the remainder of the memory

exam watch

Know that the `SGA_TARGET` parameter includes the entire memory of the SGA, including the "overhead component"—memory for the internal and fixed SGA.

among the four auto-tuned memory components—shared pool, default buffer cache, large pool, and Java pool. Let's use a simple example to demonstrate this extremely important point.

Let's say that you set the `SGA_TARGET` parameter to 1000MB. You want to use multiple block sizes in your database, so you then set the following values (both of these parameters belong to the manually sized group of SGA parameters) for the `DB_nK_CACHE_SIZE` parameters:

```
DB_4K_CACHE_SIZE=100MB
DB_8K_CACHE_SIZE=200MB
```

In this case, you'll have a total of 700MB (`SGA_TARGET` – (`DB_4K_CACHE_SIZE` + `DB_8K_CACHE_SIZE`)) left for Oracle to automatically allocate among the four auto-tuned SGA parameters.

Once you set the `SGA_TARGET` variable to a nonzero value, the database will automatically manage shared memory (only the four automatically tuned components). But how does Oracle know how much memory to allocate for each of the four auto-tuned memory components? Well, the default values for these four components begin at zero. Oracle uses an internal memory-tuning algorithm, based on database workload, to allocate memory to each of the auto-tuned memory components. Oracle will gradually increase the memory allocated to each component as necessary over time, eventually stabilizing their level at an optimal allocation. Oracle recommends that you try not to set a minimum for any these components, since that would interfere with the database's ability to allocate memory optimally.

If Oracle is automatically managing your SGA, can you influence the sizes of the automatically tuned SGA components? If you know that you will have serious problems if you start your database instance with a zero-valued shared pool, for example, you can set specific sizes for any of the four auto-tuned components. Oracle will ensure that the memory allocation to these components will never fall below the minimum allocations you made. For example, if you set the `BUFFER_CACHE` parameter to 100M and the `SHARED_POOL` parameter to 400M, Automatic Shared Memory Management will then use these two values as minimum levels for the two parameters.

For example, let's say you assign the following values in the init.ora (or SPFILE): `SGA_TARGET=900M` and `SHARED_POOL_SIZE=400M`. Then Oracle will never allocate less than 400MB of memory for the shared pool. Oracle will have 500MB

(900 – 400) of SGA left for the other three auto-tuned parameters (of course, if you assign any memory for the manually tuned parameters, you'll be left with even less than 500MB of SGA).

In summary, the two sets of SGA parameters work as follows:

- Oracle may assign more, but not less than, the minimum values you assign for the auto-tuned parameters.
- Oracle cannot change the values of the manually sized components.

SGA_TARGET Parameter Size Limits

You set the initial size of the SGA_TARGET parameter in the initialization file. If this is the first time you are using Automatic Shared Memory Management, you can quickly arrive at a good approximation for the SGA_TARGET parameter by summing the value of all the SGA components using the V$SGA view, as shown here:

```
SQL> select sum(value) from v$sga;
SUM(VALUE)
------------------
 184549376
SQL>
```

Once you start the instance with a certain value for SGA_TARGET, you can increase or decrease its size dynamically by using the `alter system` command:

```
SQL> alter system set sga_target=600M;
System altered.
SQL>
```

How high can you raise the SGA_TARGET parameter? The SGA_MAX_SIZE parameter sets an upper bound on the value of the SGA_TARGET parameter. If you haven't specified an SGA_MAX_SIZE value, you can increase the value of the

exam watch

The new SGA_TARGET initialization parameter has two major components: the automatically sized components and the manually sized components. In addition, you have a third minor component, which is the small amount of SGA memory Oracle provides for internal allocations like fixed SGA (you can consider this an overhead component).

SGA_TARGET parameter up to the maximum your particular operating system will allow. Note that you may set the value of the SGA_TARGET parameter *greater* than SGA_MAX_SIZE at startup time. In this case, Oracle automatically raises the value of the SGA_MAX_SIZE parameter to match the higher value of the SGA_TARGET parameter.

What is the minimum allowable value for the SGA_TARGET parameter? If you set any of the manually sized components, you can lower the SGA_TARGET value to the sum of the manually size parameters plus the sizes(s) of any auto-tuned parameters for which you may have set minimum values. In addition, Oracle takes into account factors like the number of CPUs on your server to arrive at the minimum allowable value for the SGA_TARGET parameter.

Once you start your instance with a specific SGA_TARGET value, you can dynamically switch to a manual memory management mode by simply setting the value of the SGA_TARGET parameter to zero using the alter system command. When you do this, the size of the four auto-tuned shared memory components will remain at their present levels. Even if you had started the instance with a specific value for some of the auto-tuned components, the current values are the ones that the database will continue to assign to those parameters, once you decide to set SGA_TARGET to zero. Of course, if you restart the instance, Oracle will assign the component values that you specify in the init.ora or the SPFILE.

Although the SGA_MAX_SIZE value acts as the upper limit of the SGA_TARGET parameter, not all operating systems are alike in this regard. On several UNIX platforms that don't support dynamic shared memory, Oracle recommends you do not set the SGA_MAX_SIZE parameter, since those platforms use the entire physical memory specified by SGA_MAX_SIZE immediately after instance startup. In these cases, it doesn't make any sense to set the SGA_TARGET to a value smaller than the value specified for the SGA_MAX_SIZE parameter. In other platforms (Oracle specifies SUN Solaris and Windows as examples), you can limit the total physical SGA use to the value set by the SGA_TARGET parameter.

on the job

Oracle recommends that you do not manually set any of the (four) automatically sized components, since it reduces the database's ability to adapt to database workload changes.

Let's say you have the following situation:

```
SGA_MAX_SIZE=1024M
SGA_TARGET=512M
DB_CACHE_SIZE=128M
```

You can raise the SGA_TARGET setting up to a maximum of 1024M. You can lower the SGA_TARGET value, but the DB_CACHE_SIZE value cannot go below 128M, since that is the minimum value for this parameter. That is, you can reduce the SGA_TARGET value until one or more of the auto-tuned SGA components reach their minimum size.

The SGA_TARGET Parameter and the SGA Components

When you change the value of the SGA_TARGET parameter, say increase it from 600M to 700M, all the manually configured SGA components will retain their old values. Any changes you make to the SGA_TARGET parameter will affect only the automatically configured SGA components.

The setting of the SGA_TARGET parameter influences the size of the various SGA components. If you set the SGA_TARGET to zero explicitly, or just omit this parameter altogether from your init.ora or SPFILE file, you must then configure the four auto-tuned components yourself—in fact, these components aren't auto-tuned anymore! You specify sizes for these components in the init.ora (or SPFILE) file, and you can use the `alter system` command to adjust their values after the instance starts up.

Under Automatic Shared Memory Management, Oracle allocates minimum values for all four auto-tuned memory components when the instance starts and adjusts them as necessary. For example, suppose that you set SGA_TARGET to 125M and you don't manually size any auto-tuned or manually tuned memory components. You can use the V$SGA_DYNAMIC_COMPONENTS view to see the values Oracle *currently* assigns to the auto-tuned components.

```
SQL> select component,current_size
  2* from v$sga_dynamic_components
COMPONENT                     CURRENT_SIZE
---------------------------------------------------------------- ------------
shared pool                     37748736
large pool                      4194304
java pool                       4194304
DEFAULT buffer cache            75497472
KEEP buffer cache                  0
RECYCLE buffer cache               0
DEFAULT 2K buffer cache            0
DEFAULT 4K buffer cache            0
DEFAULT 8K buffer cache            0
DEFAULT 16K buffer cache           0
DEFAULT 32K buffer cache           0
streams pool                       0
```

```
OSM Buffer Cache                    0
13 rows selected.
SQL>
```

As you can see, Oracle assigns initial values for all four auto-tuned parameters using the SGA_TARGET value of 125M. You can also see that all manually sized components have a value of zero (since no allocations have been made for these components in the example).

on the job

Oracle doesn't recommend that you set any of the auto-tuned parameters, as it reduces the ability of Oracle to optimally allocate the SGA among the various components.

If you look in the V$PARAMETER view, you may see different values for the auto-tuned parameters. The values shown in that view are the minimum values of these parameters, not the actual current values.

Let's briefly review the important points regarding setting the sizes of the SGA_TARGET parameter and the components of the SGA when you use Automatic Shared Memory Management.

- You can increase the size of the SGA_TARGET parameter until you reach the SGA_MAX_SIZE parameter's value.
- If you increase the size of SGA_TARGET, you can allocate the additional memory only among the auto-tuned parameters.
- If you decrease the size of SGA_TARGET, Oracle will reduce the size of one or more of the auto-tuned parameters. Oracle will not change the size of the manually tuned SGA components.
- You can reduce the SGA_TARGET parameter's size until you reach the minimum size for any of the auto-tuned components. You may specify this minimum size, or Oracle may specify it based on the number of CPUs and other factors.
- If you dynamically disable automatic SGA management (by setting SGA_TARGET=0), the value of the auto-tuned parameters will not be set to zero. These parameters will retain their current values. If the current values are higher than any manually set minimums, the current values, not the minimum values set in the initialization files, will prevail.
- If you assign a minimum value for the auto-tuned components, that will act as the *lower bound* for the SGA_TARGET parameter.

- If you don't specify a minimum value for an auto-tuned SGA component, you'll see *zero values* for this parameter in the V$PARAMETER view. The default value of the four auto-tuned parameters is zero. In addition, the value of the ISDEFAULT column will be TRUE.
- If you specify a minimum value for any auto-tuned parameter, you'll see that value in the V$PARAMETER view.
- If you decrease the size of any manually tuned component, Oracle will give the additional memory released to one or more auto-tuned components.
- If you increase the size of any manually tuned components, memory is reduced from one or more auto-tuned components.
- If you dynamically increase the size of one of the auto-tuned parameters, the component's size goes up immediately; the additional memory comes from one of the other auto-tuned components. On the other hand, if you decrease the size of one of the auto-tuned components, the component's size will not go down. The component's size stays at the current level and will go down only if Oracle deems it is good to lower the size later on.

SPFILE and Automatic Memory Management

You can store your initialization parameters in the traditional init.ora file or the newer parameter file, SPFILE. Oracle recommends that you use the SPFILE because of the inherent benefits that come with its use. Automatic Shared Memory Management is a good example of why the SPFILE is a superior way of managing your initialization parameters, compared to the init.ora file.

Under Automatic Shared Memory Management, the database determines the ideal allocations of memory for the four automatic components. It does this with the help of internal algorithms that continually analyze the nature of the database workload. After you first incorporate Automatic Shared Memory Management, Oracle doesn't know the ideal levels for these components. It arrives at these after a period of gradual calibration based on the nature of your workload.

What happens if you shut down the database instance? Well, if you are using the init.ora file for specifying your initialization parameters, Oracle must go through the laborious process of analyzing the workload again. If you use the SPFILE instead, Oracle *remembers* the sizes of the four auto-tuned parameters across the instance shutdown. Thus, when you restart the instance, you won't start from scratch; Oracle will start with the values the auto-tuned memory parameters had before you shut down the instance.

on the job

Use the SPFILE (rather than the init.ora file) if you want Oracle to remember the sizes of the automatically tuned memory components across an instance shutdown.

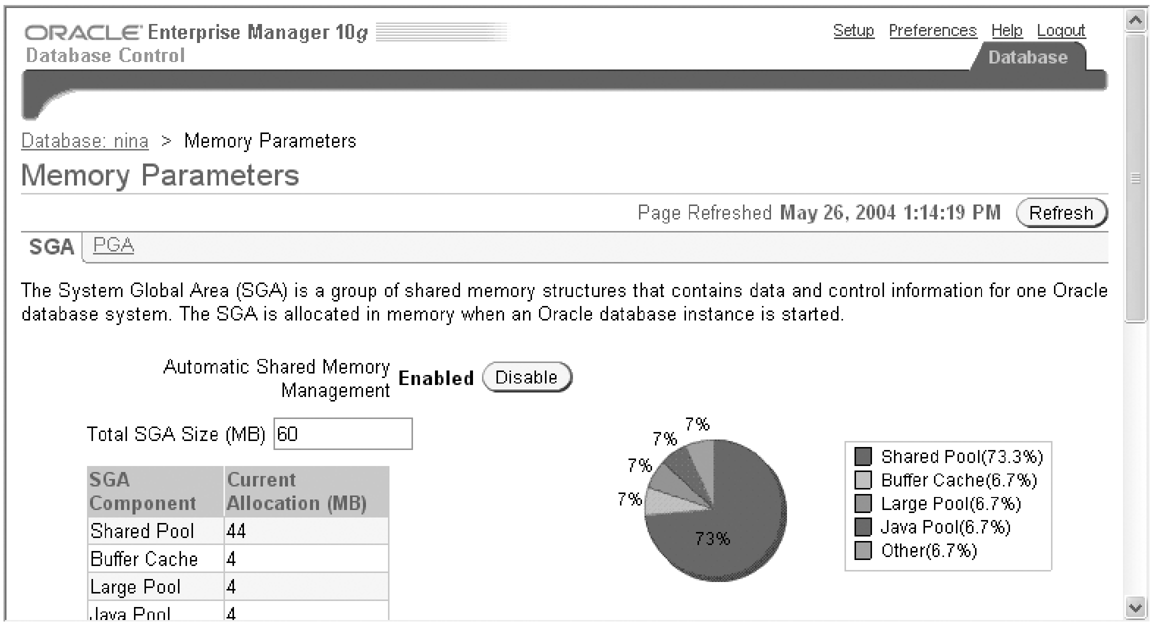
Automatic SGA Management with OEM Database Control

You can use the OEM Database Control to configure Automatic Shared Memory Management in your database, using the following steps:

1. Click the Administration link in the Database Control home page.
2. Under the Instance heading, click the Memory Parameters button.
3. Select Enable as your choice for the Automatic Shared Memory Management option.

Figure 3-1 shows the Database Control page for modifying the SGA management options.

FIGURE 3-1 Using the Database Control for specifying SGA management options


ORACLE Enterprise Manager 10g
Database Control
Setup Preferences Help Logout
Database
Database: nina > Memory Parameters
Memory Parameters
Page Refreshed May 26, 2004 1:14:19 PM
Refresh
SGA PGA
Disable
7%
7%
7%
7%
73%

The System Global Area (SGA) is a group of shared memory structures that contains data and control information for one Oracle database system. The SGA is allocated in memory when an Oracle database instance is started.

Automatic Shared Memory Management: Enabled

Total SGA Size (MB): 60

SGA Component	Current Allocation (MB)
Shared Pool	44
Buffer Cache	4
Large Pool	4
Java Pool	4

■	Shared Pool(73.3%)
□	Buffer Cache(6.7%)
▨	Large Pool(6.7%)
■	Java Pool(6.7%)
▨	Other(6.7%)

CERTIFICATION OBJECTIVE 3.03

Using Automatic Optimizer Statistics Collection

In an Oracle database, the query optimizer plays a critical role in executing SQL statements in the most efficient manner, using the least resources. You can execute a given SQL statement in several ways, and it is the query optimizer's job to provide the database with the fastest and most efficient way to perform a database query.

To arrive at the "best" plan of execution for any SQL statement, the optimizer first looks at the available access paths, join orders, and so on, and selects several candidate execution plans for the query. Next, it figures out the cost of the alternative execution plans, based on their usage of I/O, CPU, and memory. For this step, the optimizer uses optimizer statistics—crucial statistics that tell the optimizer about the data distribution and storage characteristics of tables and indexes in the database. Finally, it compares the cost of the alternative plans and picks the one with the least cost.

on the job

Oracle recommends that you let the database collect optimizer statistics automatically.

The optimizer relies on details about various objects to figure out the best plan of execution, which usually is the execution plan with the least cost (I/O and CPU cost mostly). The statistics that the Oracle optimizer relies on are called o*ptimizer statistics*, which include the following items:

- Table statistics, like the number of rows in a table and the average row length
- Column statistics, like distinct values in a column and data distribution patterns
- Index statistics include the number of levels in the index B-Trees
- System statistics, including CPU and I/O performance

Oracle stores all of these optimizer statistics in its data dictionary for the optimizer's use. Since tables and indexes may change constantly in terms their data and other properties, it is essential that you regularly refresh the optimizer statistics so they do not become stale, and thus misleading.

Prior to Oracle8*i*, DBAs relied on the `analyze table` statements to gather statistics for the Oracle optimizer. In Oracle8*i*, you had access to the new `DBMS_STATS` package, which made the job of collecting statistics easier and more comprehensive.

You used the DBMS_STATS package to collect the latest statistics for cost-based optimization. Even with the availability of the DBMS_STATS package, it was still your responsibility, as a DBA, to schedule the statistics collection jobs.

As you are aware, the Oracle optimizer cannot function correctly if you don't feed it correct and up-to-date statistics. Poor statistics mean nonoptimal execution plans, leading to degradation in query performance. In Oracle Database 10*g*, for the first time, you can *automate* the optimizer statistics collection process by allowing Oracle to collect the statistics for you.

Before we go into the details of how automatic statistics collection works, remember that statistics collection is just one part of query optimization. The choices you make regarding the optimizer mode and any optimizer hints that you may use have a significant bearing on the query optimizer's behavior. Thus, you need to focus on three things—the optimizer mode, optimizer hints, and the collection of optimizer statistics—in order to set up efficient query optimization. Let's first quickly review the concepts of optimizer mode and optimizer hints, before going on to discuss how Oracle automatically collects the optimizer statistics for you.

exam watch

If you set the initialization parameter STATISTICS_LEVEL to BASIC, you disable the monitoring feature, and thus turn off the automatic collection of optimizer statistics.

Optimizer Mode and Hints

In Oracle Database 10*g*, you can set the following goals for the query optimizer, using the initialization parameter OPTIMIZER_MODE:

- **ALL_ROWS** This is the default value. Using the ALL_ROWS goal will ensure the best throughput, which means it will minimize resource use. When you use the ALL_ROWS setting, Oracle uses a cost-based strategy for all SQL statements in the session, regardless of the presence of statistics. The ALL_ROWS setting will lead to the selection of an execution plan that will return the full result set quickly.
- **FIRST_ROWS_*n*** This value will ensure the minimization of response time for returning the first *n* rows in the query output. Oracle uses a cost-based approach regardless of the presence of optimizer statistics.

on the job

FIRST_ROWS_*n will get something back quickly, although it may take longer to retrieve the full set of data. Therefore, the *FIRST_ROWS* setting will favor the use of indexes.

- **FIRST_ROWS** If you use this value, Oracle will use a combination of heuristics (rules of thumb) and a cost-based approach to get you the first few rows in the query output. Note that the FIRST_ROWS setting exists only for backward compatibility. Oracle recommends that you use the FIRST_ROWS_*n* setting instead.

The OPTIMIZER_MODE settings determine the way the query optimizer will perform optimization throughout the database. However, at times, due to lack of accurate statistics, the optimizer can be mistaken in its estimates, leading to poor execution plans. In cases like this, you can override this database optimization setting at the individual SQL statement level, by using optimizer hints. Oracle Database 10*g* also provides the new SQL profile feature, which enables you to collect auxiliary information using sampling and partial execution techniques, thereby avoiding the use of optimizer hints. Chapter 5 discusses the SQL profile feature in detail.

Now that we have reviewed optimizer modes and optimizer hints, let's turn to a discussion of the last factor that determines how a query optimizer works—optimizer statistics collection.

How Automatic Optimizer Statistics Collection Works

Remember that regular collection of statistics is vital for the Optimizer to produce correct and efficient execution plans. Oracle Database 10*g* introduces automatic optimizer statistics collection, and Oracle recommends that you let the database automatically gather the statistics rather than manually collect statistics yourself.

It's very easy to enable automatic statistics collection in Oracle Database 10*g*—Oracle automatically starts collecting statistics when you create the database. All you need to do to make sure the automatic statistics collection process works is to ensure that the STATISTICS_LEVEL initialization parameter is set to TYPICAL or ALL. Oracle will use the DBMS_STATS package to collect optimizer statistics on an automatic basis.

We'll look at how Oracle sets up the automatic statistics collection process in the following sections.

Using the Scheduler to Run DBMS_GATHER_STATS_JOB

Oracle automatically creates a database job called GATHER_STATS_JOB at database creation time. You can verify that this automatic statistics collection job exists by running the following query:

```
SQL> select job_name
  2  from dba_scheduler_jobs
  3* where job_name like 'GATHER_STATS%';
JOB_NAME
-----------------------------------------
GATHER_STATS_JOB
SQL>
```

Oracle schedules the GATHER_STATS_JOB job for automatic execution using the new Scheduler tool. In Oracle Database 10g, the Scheduler replaces and enhances the old job scheduling capability that used the DBMS_JOBS package, and I explain it in detail in Chapter 7.

The Oracle Scheduler has two default operation windows:

- The weeknight window covers the time between 10:00 P.M. and 6:00 A.M., Monday through Friday.
- The weekend window covers the time between 12:00 A.M. Saturday and 12:00 A.M. Monday.

Together, the weeknight and the weekend windows are known as the *maintenance* window. Of course, you can change the default timings of the maintenance window as necessary. Oracle automatically schedules the GATHER_STATS_JOB job to run when the maintenance window opens. Even if the job doesn't complete before the maintenance window is over, the job will run to completion.

If you want to stop the automatic gathering of statistics, you may do so by disabling GATHER_STATS_JOB, as shown here:

```
SQL> begin
  2  dbms_scheduler.disable('gather_stats_job');
  3  end;
  4  /
PL/SQL procedure successfully completed.
SQL>
```

The GATHER_STATS_JOB job calls the procedure DBMS_STATS.GATHER_DATABASE_STATS_JOB_PROC to gather the optimizer statistics. The job collects statistics only for objects that fall into one of the following classes:

- Objects with missing statistics
- Objects with stale statistics

The GATHER_DATABASE_STATS_JOB_PROC procedure is similar to the GATHER_DATABASE_STATS procedure of the DBMS_STATS package. The significant difference is that GATHER_DATABASE_STATS_JOB_PROC sets priorities based on the DML activity in each table. The procedure will analyze the objects that have had the most DML first, so that even if it doesn't finish before the window closes, the tables that require new statistics the most will have been analyzed.

Using the Database Control to Manage the GATHER_STATS_JOB Schedule

You can use the OEM Database Control to change the current schedule of the GATHER_STATS_JOB schedule. Here are the steps:

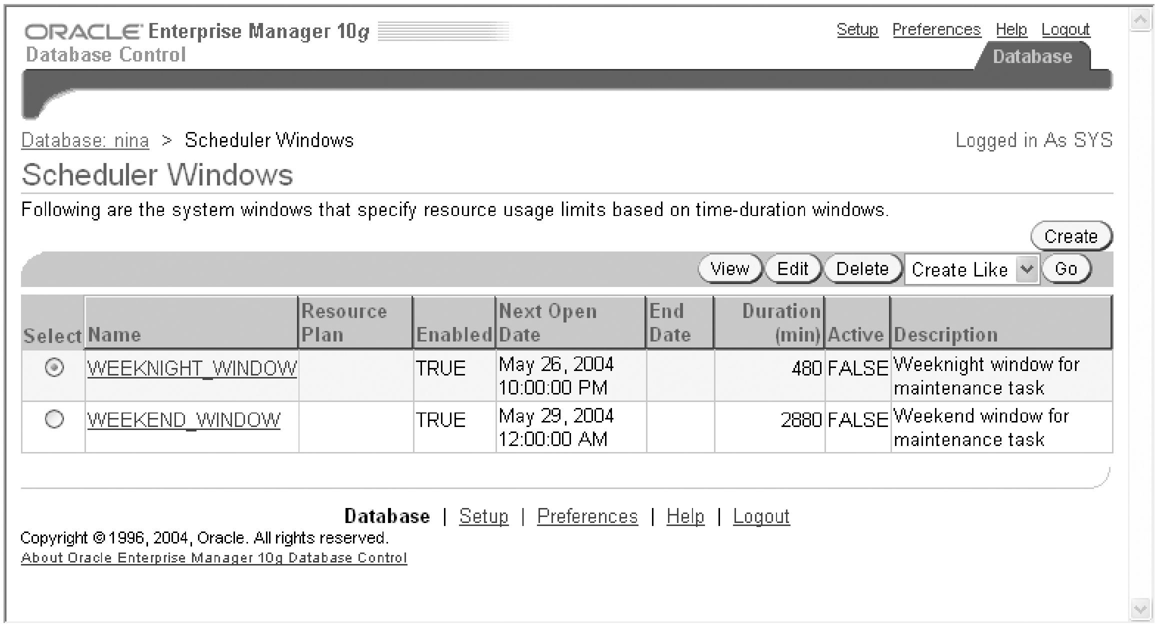
1. From the Database Control home page, click the Administration tab.
2. Go to the Scheduler Group and click the Windows Link
3. Click the Edit button. You'll then be able to edit the weeknight or the weekend window timings.

Figure 3-2 shows the Scheduler Windows page of the Database Control, where you can modify your operating windows for the Scheduler utility.

exam watch

Remember that the GATHER_STATS_JOB job collects statistics for an object only if there are no statistics for that object or if the collected statistics have become stale. Oracle considers statistics as being stale when the database modifies a significant proportion, usually 10 percent of a table's rows.

FIGURE 3-2 The Scheduler Windows page of the OEM Database Control



Database: nina > Scheduler Windows

Logged in As SYS

Scheduler Windows

Following are the system windows that specify resource usage limits based on time-duration windows.

Select	Name	Resource Plan	Enabled	Next Open Date	End Date	Duration (min)	Active	Description
●	WEEKNIGHT_WINDOW		TRUE	May 26, 2004 10:00:00 PM		480	FALSE	Weeknight window for maintenance task
○	WEEKEND_WINDOW		TRUE	May 29, 2004 12:00:00 AM		2880	FALSE	Weekend window for maintenance task

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

Table Monitoring

Oracle Database 10*g* uses an automatic table-monitoring mechanism (enabled by default when `STATISTICS_LEVEL` is set to `TYPICAL` or `ALL`) to determine whether a database object needs fresh statistics. Suppose that there are no changes in any table data in your database over a certain period. In this case, all the previously collected table statistics are still up-to-date, and you don't need to collect statistics again for this table. On the other hand, if a table is going through numerous update, insert, and/or delete operations after the statistics are collected, the statistics are said to become *stale*, since they don't represent the true distribution of data in the table.

You cannot use the `ALTER_DATABASE_TAB_MONITORING` and `ALTER_SCHEMA_TAB_MONITORING` procedures of the `DBMS_STATS` package to turn table monitoring on and off at the database and schema level, respectively, because these subprograms are deprecated in Oracle Database 10*g*. Oracle now automatically performs the functions previously taken care of by these subprograms.

Manual Collection of Optimizer Statistics

Oracle Database 10*g* also allows you to gather statistics manually using the DBMS_STATS package. The following are some situations when you *must* use manual rather than automatic methods of collecting statistics:

- When you use external tables
- When you need to collect system statistics
- To collect statistics on fixed objects, such as the dynamic performance tables (for dynamic tables, you should use the GATHER_FIXED_OBJECTS_STATS procedure to collect optimizer statistics)
- Immediately after you run a bulk load job, since this will make your automatically collected statistics unrepresentative

Let's look at how you can use the DBMS_STATS package to perform several tasks involving the collection and management of optimizer statistics.

Using the DBMS_STATS Package

The DBMS_STATS package helps you view, collect, and modify optimizer statistics. The DBMS_STATS package has the following important procedures:

- GATHER_TABLE_STATS collects all table statistics.
- GATHER_INDEX_STATS collects all index statistics.
- GATHER_SCHEMA_STATS collects statistics for all objects in a schema.
- GATHER_DATABASE_STATS collects statistics for all database objects.
- GATHER_DICTIONARY_STATS collects statistics for all data dictionary objects.
- GATHER_SYSTEM_STATS collects system statistics.

Here's a simple example showing how to use the DBMS_STATS package to collect an entire schema's statistics:

```
EXECUTE DBMS_STATS.GATHER_SCHEMA_STATS ('SALAPATI', DBMS_STATS.AUTO_SAMPLE_SIZE);
```

In the example, the AUTO_SAMPLE_SIZE parameter instructs Oracle to determine the ideal sample size for each object, based on its size and other characteristics. By setting the OPTIONS parameter (not shown in this example) to GATHER_STALE or GATHER_AUTO, you can ensure that Oracle collects new statistics whenever it encounters stale statistics for objects. (Note that GATHER_AUTO is the same as GATHER_EMPTY plus GATHER_STALE.)

You can also use the DBMS_STATS package to delete, import, restore, and set optimizer statistics that you have previously collected.

How often should you execute the DBMS_STATS package to collect statistics? If your database performs only a small amount of DML activities, you may collect statistics at relatively longer intervals, say weekly or monthly. However, if your database objects go through constant change on a daily basis, you need to schedule the statistics collection jobs much more frequently, say daily or even more often. One of the best Oracle Database 10*g* new features is that with the combination of GATHER_AUTO, AUTO_SAMPLE_SIZE, and scheduled GATHER_DATABASE_STATS_JOB procedure, you can just let Uncle Oracle decide what and how much to analyze, analyzing the important objects first.

Handling Volatile Tables by Locking Statistics

Let's say you have a table that is truncated a few times during the day, each time getting new rows by way of fresh data insertions. Also, let's suppose you have another table that is subject to numerous deletions throughout the day. Let's assume you have a nightly Scheduler job that performs automatic statistics collection. Clearly, in the cases I just described, the nightly statistics collection for these tables would be, in all likelihood, somewhat unrepresentative. What do you do under such circumstances?

Oracle Database 10*g*'s solution is to let you "lock" optimizer statistics for tables and schemas—in essence, freezing the most representative optimizer statistics, so the optimizer always uses these, rather than the unrepresentative statistics caused by excessive insert and delete operations. When you lock statistics in this manner, you prevent the automatic collection of statistics for the locked objects or schema. Thus, the statistics you've locked will always be seen as the true statistics for the table, regardless of data changes.

exam watch

When you lock a table's statistics, Oracle automatically locks all the related statistics, such as column statistics and index statistics, as well.

Use the following procedures in the DBMS_STATS package to lock and unlock table and schema statistics.

- LOCK_TABLE_STATISTICS
- UNLOCK_TABLE_STATISTICS

- `LOCK_SCHEMA_STATISTICS`
- `UNLOCK_SCHEMA_STATISTICS`

Here's an example where I lock the statistics for the test table in my schema:

```
SQL> execute DBMS_STATS.LOCK_TABLE_STATS('salapati','test');
PL/SQL procedure successfully completed.
SQL
```

on the job

If you have a volatile table without any statistics, you can lock the statistics.

You can lock a table with statistics or without statistics, using the `LOCK_TABLE_STATS` procedure in both cases. You can also override statistics locking if necessary.

Locking Tables Without Statistics You can lock a table without any statistics, by setting the statistics of a table to NULL. To set the statistics to NULL, you need to first delete any existing statistics, and then lock the table. Here's how to do this:

```
BEGIN
  DBMS_STATS.DELETE_TABLE_STATS('HR','EMPLOYEES');
  DBMS_STATS.LOCK_TABLE_STATS('HR',' EMPLOYEES');
END;
/
```

on the job

The `LOCK*` procedures either freeze the current statistics in place or keep the statistics NULL (no collection of statistics will take place after deleting the current statistics).

Locking Tables with Statistics Instead of setting statistics to NULL when you lock a table, you can always save the same set of statistics, regardless of any table row insert or delete operations. If you have a set of statistics that you consider representative of the table when it's fully loaded, you can lock the table with these statistics in place, thus preventing the excessive insertions and deletions from skewing the optimizer statistics. In this case, use just the `LOCK_TABLE_STATS` procedure, as shown in the following example.

```
BEGIN
    DBMS_STATS.LOCK_TABLE_STATS('HR',' EMPLOYEES');
END;
```

Overriding Statistics Locking In some cases, you may want Oracle to override any existing locks you have imposed on the optimizer statistics. You can do so by using the new FORCE argument with several procedures in the DBMS_STATS package. For the following procedures, the default is FORCE=FALSE. When you set FORCE= TRUE, they will behave as follows:

- DELETE_SCHEMA_STATS will delete the statistics even if they are locked.
- IMPORT_SCHEMA_STATS will ignore the statistics lock and import statistics anyway.
- RESTORE_SCHEMA_STATS will restore statistics even if they are locked.
- SET_SCHEMA_STATISTICS will set the values even if the statistics are locked.

Restoring Historical Optimizer Statistics

Suppose your newly collected optimizer statistics disappoint you, and you think an older version of the statistics was giving you a lot better performance. What do you do under such circumstances?

Fortunately, Oracle lets you automatically save all old statistics whenever your refresh the statistics. Thus, it's a simple matter to ask Oracle to revert to using an older set of "good" statistics. You can restore any type of statistics by using the appropriate RESTORE_*_STATS procedure. For example, the RESTORE_TABLE_STATS procedure is used to restore table statistics. Similarly, the RESTORE_DICTIONARY_STATS procedure helps you restore an older version of the dictionary table statistics. Oracle also has procedures for restoring statistics at the schema, database, and system levels.

exam watch

You can't restore any statistics you collect using the `analyze` command. You also can't restore any user-defined statistics.

Two database views are critically useful when you want to restore older statistics: DBA_OPTSTAT_OPERATIONS and DBA_TAB_STATS_HISTORY. The DBA_OPTSTAT_OPERATIONS view contains a history of all optimizer statistics collections, as shown in the following query:

```
SQL> select operation,end_time
  2  from dba_optstat_operations;
```

```
OPERATION                                       END_TIME
gather_database_stats(auto)     19-APR-04 10.04.59.321000 PM -05:00
set_system_stats                19-APR-04 01.44.53.098000 PM -05:00
set_system_stats                19-APR-04 01.44.53.769000 PM -05:00
set_system_stats                19-APR-04 01.44.53.832000 PM -05:00
SQL>
```

The DBA_TAB_STATS_HISTORY view contains a record of all changes made to table statistics. By default, the DBA_TAB_STATS_HISTORY view saves the statistics history for 31 days. Therefore, you can restore statistics to any time within the previous 31 days.

How long can you retain old statistics? As just stated, by default, Oracle will save your statistics for a period of 31 days. However, by using the ALTER_STATS_HISTORY_RETENTION procedure of the DBMS_STATS package, you can change the default value of the statistics history retention interval.

If you have set your STATISTICS_LEVEL parameter to TYPICAL or ALL, Oracle will automatically purge the old statistics. To perform a manual purge of the statistics, you need to use the PURGE_STATS procedure of the DBMS_STATS package.

CERTIFICATION OBJECTIVE 3.04

Using Automatic Undo Retention Tuning

Oracle databases use undo records to save the actions of transactions. Oracle refers to the records collectively as *undo*. Oracles uses the undo information to roll back, or undo, a transaction if necessary. For example, you may have inserted or deleted data, but now you want to roll back the changes to return the database to how it was before you made the change.

Undo data can help you perform key tasks like the following:

- Perform a rollback when you don't want to commit changes
- Provide read consistency, by preserving the before image of data so a user sees a consistent view of data, even when another user is changing the same data
- Aid during a database recovery process, by undoing any uncommitted changes applied to datafiles by the redo logs
- Facilitate the flashback features that rely on undo information to function

Until the Oracle9*i* database, Oracle used rollback segments to manage undo information. Rollback segments are complex to administer, with segments encountering the dreaded ORA-1555 (snapshot too old) errors on a regular basis during long transactions. While Oracle Database 10*g* still enables you to use traditional rollback segments, that feature has been deprecated in this version. In Oracle Database 10*g*, you can use either a manual mode of undo management (involving rollback segments), or Automatic Undo Management (AUM). Oracle recommends, however, that you use the AUM feature, wherein Oracle will be in charge of maintaining the undo segments. You don't have the headaches of managing rollback segments anymore. In addition, you can now control the amount of time the database retains important undo information before it overwrites the data.

exam watch

If you don't specify the mode of undo management, you will be using manual undo management, with the help of rollback segments. This is because the default undo management mode is manual.

Automatic Undo Management Configuration

To enforce Automatic Undo Management, you need to configure the following initialization parameters:

- **`UNDO_MANAGEMENT`** The default value for this parameter is `MANUAL`, which means you are going to use the traditional rollback segments to manage undo. If you want to use Automatic Undo Management, you need to specify `AUTO` as the value for this parameter.
- **`UNDO_TABLESPACE`** If you choose Automatic Undo Management, you should specify a separate tablespace to hold the undo information by using this parameter. If you don't do this, the undo will be stored in the SYSTEM tablespace.
- **`UNDO_RETENTION`** This parameter specifies the duration of time for which the database should retain the undo information. The default for this parameter is 900 seconds.

The `UNDO_RETENTION` and `UNDO_TABLESPACE` parameters are crucial for managing your undo data. On a simple level, it is easy to see what factors will determine how you set these parameters. If you have a large amount of undo (because of a large amount of database changes like insert and delete operations), you'll need a correspondingly large undo tablespace to hold all the undo information. If you have several long-running SQL queries, your undo retention time should be correspondingly long.

The Undo Advisor

The easiest way to manage undo activity is to access the Undo Advisor from the OEM Database Control. You can get to the Undo Advisor page by clicking the Undo Management link on the Advisor Central page of the OEM Database Control. Once you are on the Undo Management page, you can set parameters for undo management. You can perform the following undo management tasks from the Database Control interface:

- Enable/disable Automatic Undo Management
- Specify/modify undo tablespaces
- Ask for undo tablespace size recommendations
- Ask for undo retention period recommendations
- Experiment with alternative undo retention period settings

You can use the Undo Advisor to help you configure the optimal undo retention time and your undo tablespace size. The Undo Advisor bases its undo management recommendations on the following criteria:

- Longest running query (in minutes)
- Average undo generation rate (KB/minute)
- Maximum undo generation (KB/minute)

Undo Tablespace and Undo Retention Management

Undo management involves managing the undo tablespace and undo retention issues. Let's look at these undo management issues in the following sections.

Managing the Undo Tablespace

Oracle provides you two ways of making sure you don't run out of room for undo in your undo tablespace.

- Oracle alerts you when your undo tablespace is going to run out of space, just as it alerts you about space problems in other tablespaces (for more on tablespace and other alerts, see Chapter 4).
- Oracle alerts you if your system has long-running queries that may result in the ORA-1555 (snapshot too old) error.

Managing Undo Retention

Oracle automatically tunes the undo retention period by constantly collecting statistics on query length as well as undo generation rate. As noted earlier, the default value of the `UNDO_RETENTION` parameter is 900 seconds. So, even if you don't explicitly ask Oracle to retain undo information for a specific period, once you specify the `UNDO_MANAGEMENT=AUTO` parameter, Oracle will automatically start retaining your undo data for 15 minutes (900 seconds).

Here is how undo retention actually works in practice (assume that you left the undo retention interval at the default value of 900 seconds):

- If your undo tablespace has enough space, Oracle will retain undo data for at least 900 seconds.
- If your undo tablespace doesn't have enough free space, Oracle may choose not to retain the undo information for 900 seconds; that is, Oracle will let new undo records write over the older records, even before the older records are 900 seconds old.

Why does Oracle behave in this way? It will shorten the retention interval when there are new DML operations in the database and there isn't enough free space in the undo tablespace to accommodate these new DML operations. Rather than cause the DML operations to fail due to the lack of undo space, Oracle simply chooses the "lesser evil" of overwriting some of the old redo information.

Using the Retention Guarantee Option

If you want to absolutely, positively guarantee that Oracle retains the undo information for the length of time you chose by specifying the `UNDO_RETENTION` parameter's value, you can do so by using a new Oracle Database 10*g* feature: *retention guarantee*. By default, Oracle disables the retention guarantee feature. You can enable the guarantee feature at database creation time, at the undo tablespace creation time, or by using the `alter tablespace` command.

You can check the current retention setting for your undo tablespace by issuing the following query:

```
SQL> select tablespace_name,retention
  2  from dba_tablespaces;
TABLESPACE_NAME            RETENTION
-------------------------------------------
SYSTEM                     NOT APPLY
UNDOTBS1                   NOGUARANTEE
SYSAUX                     NOT APPLY
…
SQL>
```

The `NOGUARANTEE` value under the retention column for the UNDOTBS1 tablespace shows the default value of the `UNDO_RETENTION` parameter. By default, there is *no* guarantee of undo retention. You can guarantee undo retention for the UNDOTBS1 tablespace by using the following command:

```
SQL> alter tablespace UNDOTBS1 retention guarantee;
Tablespace altered.
SQL>
```

If you wish to enforce the retention guarantee feature right from the beginning, you can create your undo tablespace with this feature built in, as shown here:

```
SQL> create undo tablespace new_undo
          datafile 'C:\oracle\product\10.1.0\data\new_iundo_01.dbf'
          size 10M autoextend on
          retention guarantee;
```

exam watch

Does Automatic Undo Management mean that you don't need to worry about the usual ORA-1555 (snapshot too old) errors, because Oracle reuses undo segments with unexpired undo data? Well, it all depends on how much space there is in your undo tablespace. If the available free space is the undo tablespace isn't enough for your transaction-activity levels, Oracle may overwrite unexpired undo data, causing the snapshot too old errors. The only way to eliminate this error is to use the `RETENTION GUARANTEE` clause, which guarantees that Oracle will never overwrite any undo data that is within the undo retention period. One of the main reasons why you may use the retention guarantee feature is to enable the success of flashback features in your database, which depend critically on the retention of necessary undo information.

INSIDE THE EXAM

In the Automatic Shared Memory Management section, you must clearly understand which are the automatically tunable and the manually tunable SGA components. You must also know the background process that coordinates the sizing of the memory components. What is the difference between the behavior of the `SHARED_POOL_SIZE` component in Oracle Database 10*g* and in the older versions of Oracle?

There will be a couple of questions on the AWR and the ADDM. You must understand the new time model and the importance of the DB time metric. What are the different ways in which you can get an ADDM report? How do you get ADDM reports for custom intervals? What does an ADDM report contain?

Expect a question regarding the automatic collection of optimizer statistics by Oracle. How does Oracle prioritize its statistics collection? What procedure do you use to restore older optimizer statistics?

on the job

If you specify the `RETENTION GUARANTEE` clause, you run the risk of your DML operations failing due to lack of undo space. Therefore, don't use this clause unless you must guarantee the availability of undo data (for example, for the flashback query feature).

CERTIFICATION SUMMARY

In this chapter, you first learned about the AWR and how it collects database performance statistics in the form of regular snapshots. Then you learned how the ADDM uses these snapshots to automatically analyze database performance and make recommendations to improve it if necessary.

The chapter introduced you to the new Automatic Shared Memory Management feature and the use of the `SGA_TARGET` initialization parameter. You learned about how Oracle can automatically manage the four auto-tuned parameters.

You learned about how Oracle Database 10*g* can automatically gather key optimizer statistics for you. You reviewed the Automatic Undo Management feature and learned the role of the parameters `UNDO_TABLESPACE` and `UNDO_RETENTION`. Finally, you learned about the new undo retention guarantee feature, which lets you ensure the retention of undo data.

✓ TWO-MINUTE DRILL

Using the Automatic Database Diagnostic Monitor (ADDM)

- ❑ The Automatic Workload Repository (AWR) collects database performance statistics on a regular basis and stores them in the SYSAUX tablespace.
- ❑ By default, the AWR collects its snapshots on an hourly basis and stores them for a period of seven days.
- ❑ The ADDM uses the new time statistics model to analyze database performance.
- ❑ The `V$SYS_TIME_MODEL` and `V$SESS_TIME_MODEL` views show the time statistics on a system and session level, respectively.
- ❑ The key time model metric, DB time, covers the actual time spent processing user database calls.
- ❑ The main goal of the ADDM tool is to reduce the DB time metric.
- ❑ The ADDM addresses the root causes of performance problems, not just symptoms.
- ❑ The AWR automatically purges the old snapshots after the default interval of seven days.
- ❑ The ADDM may recommend changes in I/O or memory, as well as database and application configuration.
- ❑ The ADDM may also recommend the invocation of other management advisors in order to analyze a problem in depth.
- ❑ The MMON background process helps manage the ADDM and schedules the ADDM.
- ❑ Oracle enables ADDM by default. Just make sure to set the `STATISTICS_LEVEL` initialization parameter to `TYPICAL` or `ALL`.
- ❑ By adjusting either or both of two variables—snapshot interval and data retention period—you can control the amount of data that AWR maintains.
- ❑ The `DBIO_EXPECTED` parameter indicates how fast your I/O system performs.
- ❑ You may use the `DBMS_ADVISOR` package to change the default value of the `DBIO_EXPECTED` parameter.
- ❑ You can view the ADDM analysis reports in three different ways: through the Database Control, through SQL statements, and with the help of database packages.

- ❑ The easiest way to view the ADDM reports is through the OEM Database Control.
- ❑ You can obtain ADDM reports by using the SQL script addmrpt.sql.
- ❑ The ADDM reports have three main components: impact, findings, and recommendations.
- ❑ You need the ADVISOR privilege to use the `DBMS_ADVISOR` package, which helps you manage the ADDM.
- ❑ The `GET_TASK_REPORT` procedure of the `DBMS_ADVISOR` package enables you to produce ADDM reports.

Using Automatic Shared Memory Management

- ❑ To use the Automatic Shared Memory Management feature, you must set the `STATISTICS_LEVEL` parameter to `TYPICAL` or `ALL`.
- ❑ There are two sets of SGA parameters: auto-tuned and manually sized.
- ❑ The auto-tuned set of parameters consists of the buffer cache, shared pool, large pool, and Java pool.
- ❑ The default value of the `SGA_TARGET` parameter is zero.
- ❑ To automate SGA management, set the `SGA_TARGET` to a value greater than zero.
- ❑ Oracle subtracts the value of the manually sized parameter (and the fixed SGA) from the SGA first. It allocates the remainder of the SGA to the four auto-tuned parameters.
- ❑ The default values of all four auto-tuned parameters are zero.
- ❑ You can dynamically increase or decrease the value of the `SGA_TARGET` parameter.
- ❑ The `SGA_MAX_SIZE` parameter sets the limit on how high you can set `SGA_TARGET`.
- ❑ Even under automatic SGA management, you can set the sizes for the four individual auto-tuned parameters.
- ❑ Oracle recommends that you don't set any auto-tuned parameters yourself.
- ❑ Use the SPFILE if you want Oracle to remember the sizes of auto-tuned parameters across instance shutdowns.

Using Automatic Optimizer Statistics Collection

- ❑ Oracle enables automatic optimizer statistics collection by default.
- ❑ You must set the STATISTICS_LEVEL parameter to TYPICAL or ALL to use Oracle's automatic optimizer statistics collection capability.
- ❑ Oracle uses the GATHER_STATS_JOB job to run the automatic statistics collection process.
- ❑ The GATHER_STATS_JOB job collects statistics only if they are stale or not available.
- ❑ The Scheduler runs the GATHER_STATS_JOB job during the maintenance window.
- ❑ The database uses the table-monitoring feature to decide whether it should collect new statistics.
- ❑ You can lock table and schema statistics using the LOCK_TABLE_STATISTICS and LOCK_SCHEMA_STATISTICS procedures from the DBMS_STATS package.
- ❑ If you lock a table's statistics, Oracle locks all associated statistics automatically as well.
- ❑ You can lock a table without statistics, by setting the statistics to NULL.
- ❑ Use the FORCE argument to override any locks on statistics.
- ❑ You can restore any old statistics by using the RESTORE_*_ procedures of the DBMS_STATS package.
- ❑ By default, Oracle saves optimizer statistics for 31 days.

Using Automatic Undo Retention Tuning

- ❑ Oracle uses manual undo management by default.
- ❑ Set UNDO_MANAGEMENT=AUTO to use Automatic Undo Management.
- ❑ The default value for the UNDO_RETENTION parameter is 900 seconds.
- ❑ If you run out of fresh room in the undo tablespace, Oracle writes over unexpired undo data.
- ❑ Use the undo RETENTION GUARANTEE clause to ensure Oracle retains undo information for the time set by the UNDO_RETENTION parameter.
- ❑ Your DML operations may fail on occasion if you choose to use the undo RETENTION GUARANTEE clause.

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.

Using the Automatic Database Diagnostic Monitor (ADDM)

1. Where does the ADDM save its analysis results?

A. In the OEM repository
B. In the ADDM tablespace
C. In the SYSTEM tablespace
D. In the SYSAUX tablespace

2. What is the key goal of the ADDM?

A. To reduce DB time
B. To reduce DB idle time
C. To reduce DB active time
D. To reduce throughput

3. In response to a performance bottleneck, what will the ADDM do?

A. Always recommend the use of management advisors
B. Always suggest its own recommendations
C. Sometimes recommend the use of management advisors
D. Propose a single recommendation to fix the problem

4. If your disk read time is 2000 microseconds, what will you need to do?

A. Use the `DBMS_ADVISOR` package to lower the value of the `DBIO_EXPECTED` parameter
B. Use the `DBMS_ADVISOR` package to raise the value of the `DBIO_EXPECTED` parameter
C. Use the `DBA_ADVISOR` package to lower the value of the `DBIO_EXPECTED` parameter
D. Use the Database Control to lower the value of the `DBIO_EXPECTED` parameter

5. To retrieve the ADDM reports using SQL, what do you need to do?

A. Run the addmrpt.sql SQL script
B. Use the `DBA_ADDM` view
C. Use the `DBA_ADVISOR` view
D. Use the `DBMS_ADDM` package

Using Automatic Shared Memory Management

6. To enable Automatic Shared Memory Management, what does the DBA need to do?

A. Set the `STATISTICS_LEVEL` parameter to `BASIC`

B. Set the `STATISTICS_LEVEL` parameter to `TYPICAL` or `ALL` and set `SGA_TARGET_SIZE` to a nonzero value

C. Set the `STATISTICS_LEVEL` parameter to `TYPICAL` or `ALL` and remove the `SGA_TARGET` parameter

D. Set the `STATISTICS_LEVEL` parameter to `TYPICAL` or `ALL` and set `SGA_TARGET` to zero

7. If you set the value of the `SGA_TARGET` parameter higher than the value of the `SGA_MAX_SIZE` parameter at instance startup, what will happen?

A. The instance will not start.

B. `SGA_TARGET` will become equal to the `SGA_MAX_SIZE` value.

C. The database will ignore the `SGA_TARGET` parameter.

D. `SGA_MAX_SIZE` is automatically raised, so it is equal to the `SGA_TARGET` value.

8. In order to turn off automatic SGA management, what should the DBA do?

A. Set the `SGA_MAX_SIZE` parameter to zero

B. Set the `SGA_TARGET` parameter to zero

C. Set the `SGA_TARGET` parameter equal to the `SGA_MAX_SIZE` parameter

D. Remove the `SGA_MAX_SIZE` parameter

9. You are using automatic SGA management, with `SGA_TARGET` set at 500M. If you set the `DB_KEEP_CACHE_SIZE` to 100M, approximately how much of the SGA memory can Oracle assign to the auto-tuned parameters?

A. 500MB

B. 600MB

C. 400MB

D. Oracle cannot assign any memory to the auto-tuned parameters under the given circumstances.

10. If you set the sizes of all four auto-tuned parameters, which of the following is true?

A. You can't use the Automatic Shared Memory Management feature.

B. Oracle can raise the sizes of the parameters under Automatic Shared Memory Management.

C. Oracle can lower the sizes of the parameters under Automatic Shared Memory Management.

D. Oracle can raise the sizes of the parameters under manual shared memory management.

Using Automatic Optimizer Statistics Collection

11. How do you verify that the automatic statistics collection job is running?

A. Query the `DBA_JOBS` view
B. Query the `DBA_SCHEDULER_JOBS` view
C. Query the `DBA_SCHEDULER` view
D. Query the `GATHER _DATABASE_STATS_JOB_PROC` view

12. When should the DBA consider locking table statistics?

A. When the table's data is extremely stable
B. When the table's data is extremely volatile
C. When the table's data changes by less than 10 percent on a regular basis
D. When the table is never truncated

13. In order to override the locking of statistics, what should you use?

A. `DBMS_STATS.DELETE_STATS` procedure
B. `DBMS_STATS.IMPORT_SCHEMA_STATS` procedure
C. `DBMS_STATS.DELETE_STATS` procedure with the `OVERRIDE` option
D. `DBMS_STATS.DELETE` procedure with the `FORCE` option

14. By default, the `DBA_TAB_STATS_HISTORY` view saves statistics history for how long?

A. 31 days
B. 30 days
C. 1 hour
D. 7 days

15. What does the `LOCK_*` procedure of the `DBMS_STATS` package help you do?

A. Lock current statistics
B. Make current statistics NULL
C. Override current statistics with the latest statistics
D. Lock statistics from a previous period

Using Automatic Undo Retention Tuning

16. If you use the retention guarantee feature, you are guaranteeing that the database will never do what?

A. Overwrite unexpired undo data
B. Overwrite expired undo data

C. Overwrite data older that the undo retention period
D. Overwrite data more recent than the undo retention period

17. To ask Oracle to retain undo information for a specific period, what must you do?

A. Use the manual undo management mode
B. Specify a certain value for the `UNDO_RETENTION` parameter
C. Use the `RETENTION GUARANTEE` clause
D. Use the `RETENTION NO GUARANTEE` clause

17. If you wish to use Automatic Undo Management, you must do which of the following?

A. Specify an undo tablespace
B. Use the retention guarantee feature
C. Specify the `UNDO_RETENTION` parameter
D. Set the value of the `UNDO_MANAGEMENT` parameter to `AUTO`

18. By default, Oracle retains your undo information for how long?

A. 31 days
B. 15 minutes
C. 30 minutes
D. 7 days

20. By default, how does Oracle handle undo?

A. Disables the retention guarantee feature
B. Enables the retention guarantee feature
C. Enables the Automatic Undo Management feature
D. Enables the retention guarantee feature for the flashback query feature

LAB QUESTION

You are currently using manual shared memory management. Your init.ora parameter file looks like this:

```
db_cache_size=80M
java_pool_size=10M
large_pool_size=10M
shared_pool_size=100M
```

What do you need to do to switch to Automatic Shared Memory Management? Ensure that your SGA allocation is the same size as before. Show the answer using the manual SQL*Plus method.

SELF TEST ANSWERS

Using the Automatic Database Diagnostic Monitor (ADDM)

1. ☑ **D.** The ADDM facility stores all its analysis reports in the SYSAUX tablespace, just as the AWR facility does.
☒ **A, B,** and **C** provide wrong destinations for the ADDM reports.

2. ☑ **A.** Reducing the DB time metric is the fundamental goal of the ADDM.
☒ **B** and **C** are misleading answers. **D** is wrong because the ADDM's goal is exactly the opposite.

3. ☑ **C.** When ADDM encounters performance problems, it may propose several recommendations that you can directly implement, in addition to making the optional recommendation to run other management advisors.
☒ **A** is wrong because the ADDM doesn't necessarily ask you to invoke a management advisor. **B** is wrong since the ADDM may also suggest that you use management advisors. **D** is wrong since the ADDM doesn't limit itself to providing a single solution for a performance problem.

4. ☑ **A.** By default, ADDM assumes that your `DBIO_EXPECTED` parameter has a value of 10000 microseconds. Since your system disk reading speed is actually lower than this, you must reset this parameter by lowering the value of the `DBIO_EXPECTED` parameter.
☒ **B** is wrong because you need to lower, not raise, the value of the `DBIO_EXPECTED` parameter. **C** is wrong since you can't use the `DBA_ADVISOR` package to adjust the `DBIO_EXPECTED` parameter. **D** is wrong since you can't use the Database Control to adjust the `DBIO_EXPECTED` parameter.

5. ☑ **A.** To retrieve the ADDM report, you must use the Oracle-supplied addmrpt.sql script.
☒ **B** is wrong because there is no `DBA_ADDM view`. **C** is wrong because the `DBA_ADVISOR` view can't help you produce an ADDM report. **D** is wrong because there is no PL/SQL package named `DBMS_ADDM`.

Using Automatic Shared Memory Management

6. ☑ **B.** You must set the `STATISTICS_LEVEL` parameter to `TYPICAL` or `ALL` and set the `SGA_TARGET` parameter to a nonzero value in order for automatic SGA management to come into force.
☒ **A** is wrong since setting the `STATISTICS_LEVEL` parameter to `BASIC` ensures that you won't have automatic SGA management in your database. **C** and **D** are wrong since removing

the SGA_TARGET parameter or explicitly setting it to zero mean the same thing, and the default value of the parameter is zero anyway. Setting the SGA_TARGET parameter to zero means you can't use Automatic Shared Memory Management.

7. ☑ **D.** The value of the SGA_MAX_SIZE parameter will be bumped up to the value of the SGA_TARGET parameter.
☒ **A** is incorrect since the instance will start without any problem. **B** is incorrect because it implies that the value of the SGA_TARGET parameter will be lowered to match the size of the SGA_MAX_SIZE parameter. Actually as answer D indicates, it's the other way around—the SGA_MAX_SIZE parameter's value will be raised to match the SGA_TARGET parameter's value. **C** is incorrect since the database doesn't ignore the higher SGA_TARGET parameter value, but uses it.

8. ☑ **B.** If you set the SGA_TARGET parameter value to zero, you'll disable automatic memory management.
☒ **A, C,** and **D** have no bearing on Automatic Shared Memory Management.

9. ☑ **C.** When you use Automatic Shared Memory Management, Oracle will first deduct the sum of manually sized parameters from the SGA_TARGET parameter. Oracle can assign only the remaining memory to the auto-tuned parameters.
☒ **A** and **B** are wrong, based on the analysis for the correct answer. **D** is wrong because you won't prevent Oracle from assigning memory to the auto-tuned parameters when you set the size of one or more manually tuned memory parameters.

10. ☑ **B.** Even under Automatic Shared Memory Management, you can set the sizes of the individual auto-tuned components. Oracle will treat this as the minimum values for these parameters, and can raise them, but won't be able to lower them.
☒ **A** is wrong because you can use Automatic Shared Memory Management, even when you set the sizes for one or more auto-tuned parameters. **C** is wrong because, as the correct answer explains, Oracle can only *raise*, not *lower*, any values you set for the auto-tuned parameters. **D** is wrong because Oracle cannot change any of the shared memory components when you use manual shared memory management.

Using Automatic Optimizer Statistics Collection

11. ☑ **B.** The new Scheduler facility runs the GATHER_DATABASE_STATS_JOB_PROC procedure, belonging to the DBMS_STATS package, to gather optimizer statistics. The job name is GATHER_STATS_JOB, which Oracle lists in the DBA_SCHEDULER_JOBS view. The new DBA_SCHEDULER_JOBS view is similar to the old DBA_JOBS view, and it provides information about all scheduled jobs in the database. You can query this view to see if the GATHER_DATABASE_STATS_JOB_PROC procedure is scheduled for running. The DBA_JOBS view exists in the Oracle Database 10*g*, but it won't tell you anything about the automatically scheduled jobs.

☒ **C** and **D** are wrong because there are no such views. **A** is wrong because the DBA_JOBS view will not have any information about the new Scheduler facility.

12. ☑ **B.** You should consider locking down a table's statistics when a table's data is extremely volatile.
☒ **A, C,** and **D** are all reasons why you *won't* need to lock down a table's statistics.

13. ☑ **D.** Any time you need to override any statistics, you should use the FORCE argument for the relevant DBMS_STATS package. Therefore, you can figure out that **D** is the right answer, since it's the only one that uses the FORCE option.
☒ **A, B,** and **C** are wrong since none of these alternatives use the FORCE option.

14. ☑ **A.** By default, Oracle saves optimizer statistics for 31 days before purging them. You can always change this default duration.
☒ **B, C,** and **D** provide the wrong period.

15. ☑ **A.** All the LOCK_* procedures enable you to lock current statistics.
☒ **B** is wrong since the LOCK_* procedures don't make the statistics NULL. **C** is wrong since the procedures don't override current statistics. **D** is wrong because the procedures lock the current statistics, not those from a previous period.

16. ☑ **A** and **D. A** is correct because the RETENTION GUARANTEE clause guarantees that you'll never overwrite unexpired data in the undo segments. **D** is correct because you won't overwrite data more recent than the undo retention period.
☒ **B** is wrong because Oracle may very well overwrite expired undo data, whether or not you use the RETENTION GUARANTEE clause. **C** is wrong because there is no guarantee that Oracle won't overwrite data older than the data retention period.

Using Automatic Undo Retention Tuning

17. ☑ **B.** UNDO_RETENTION isn't really a mandatory parameter when you use Automatic Undo Management. Even when you don't specify this parameter, Oracle will automatically use a default value for the parameter. However, if you wish the database to retain undo information for a specific length of time, you must use the UNDO_RETENTION parameter.
☒ **A** is wrong since using manual undo management isn't necessary to retain undo information for a specific period**. C** and **D** are invalid since the retention guarantee and the RETENTION NO GUARANTEE options aren't necessary to ask Oracle to retain undo information for a specific period, the UNDO_RETENTION parameter will suffice for this purpose.

18. ☑ **D.** You must set the UNDO_MANAGEMENT parameter to AUTO if you want Oracle to manage the undo information.
☒ **A, B,** and **C** are incorrect since you don't need to specify anything other than the

UNDO_MANAGEMENT=AUTO parameter to use Automatic Undo Management. As important as they are, all these alternatives mention factors that are optional, not mandatory for Automatic Undo Management.

19. ☑ **B.** By default, Oracle retains undo data for 600 seconds (15 minutes).
☒ **A, C,** and **D** provide incorrect values.

20. ☑ **A.** Oracle disables the retention guarantee feature by default. You can enable it by using the `alter system` command, as demonstrated in this chapter.
☒ **B** is wrong since Oracle disables the option by default. **C** is wrong since Oracle enables manual undo management by default. **D** is incorrect since Oracle doesn't enable the retention guarantee feature by default for any purpose. Oracle recommends that you use the retention guarantee feature sparingly, such as only when you use the flashback query feature.

LAB ANSWER

First, find out the current SGA size using the following command:

```
SQL> select sum(value) from v$sga;
SUM(VALUE)
------------------
 218103808
```

Next, issue the following command, to switch to Automatic Shared Memory Management:

```
SQL> alter system set SGA_TARGET=218103808;
System altered.
SQL>
```

Although you are now technically using Automatic Shared Memory Management, since you have set minimum values for all four auto-tuned SGA parameters in your init.ora file, Oracle can't modify the sizes of the components. You therefore must set the sizes of all four auto-tuned parameters to zero, by issuing the following set of commands:

```
SQL> alter system set db_cache_size=0;
System altered.
SQL> alter system set shared_pool_size=0;
System altered.
SQL> alter system set java_pool_size=0;
System altered.
SQL> alter system set large_pool_size=0;
System altered.
SQL>
```