Proactive Maintenance

Any proactive action refers to preventive measure. Reactive active refers applying a solution when problem arises.

Oracle 10g DBA has to respond to the problems reactively and proactively. Reactive monitoring involves monitoring a database environment after a performance or management issue has arisen. Most of the new features in Enterprise Manager (EM) Database Control are geared toward proactive monitoring.

The monitoring tools available in EM Database Control collect their information from a variety of sources such as data dictionary views, dynamic performance views, and the operating system. In addition, cost based optimizer statistics are also used in Oracle. All these sources of information are accessed by the Automatic Workload Repository feature.

Automatic Workload Repository

Oracle 10g introduces two new background processes—Memory Monitor (MMON) and Memory Monitor Light (MMNL). These processes work together to collect performance statistics directly from the System Global Area (SGA). The MMON process does most of the work by waking up every 60 minutes and gathering statistical information from the data dictionary views, dynamic performance views, and optimizer and then storing this information in the database.

The tables that store these statistics are called the Automatic Workload Repository (AWR). These tables are owned by the user SYSMAN and are stored in the SYSAUX tablespace.

To activate the AWR feature, you must set the PFILE/SPFILE parameter STATISTICS\_LEVEL to the appropriate value. The values assigned to this parameter determine the depth of the statistics that the MMON process gathers. The following table shows the values that can be assigned to the STATISTICS\_LEVEL parameter.

Specifying Statistics Collection Levels

Collection Level Description

BASIC Disables the AWR and most other diagnostic monitoring and advisory activities. Few database statistics are gathered at each collection interval when operating the instance in this mode.

TYPICAL Activates the standard level of collection activity. This is the default value for AWR and is appropriate for most environments.

ALL Captures all the statistics gathered by the TYPICAL collection level, plus the execution plans and timing information from the operating system.

Once gathered, the statistics are stored in the AWR for a default duration of 7 days. However, both the frequency of the snapshots and the duration for which they are saved in the AWR can be modified.

by using the Oracle-supplied package DBMS\_WORKLOAD\_REPOSITORY. The following SQL command shows the DBMS\_WORKLOAD\_ REPOSITORY package being used to change the AWR collection interval to 1 hour and the retention period to 30 days:

SQL> execute dbms\_workload\_repository.modify\_snapshot\_settings

(interval=>60,retention=>43200);

PL/SQL procedure successfully completed.

The 30-day retention value shown above is expressed in minutes: 60 minutes per hour ? 24 hours per day ? 30 days = 43,200 minutes.

by using the EM Database Control. Choose Administration Automatic Workload Repository Edit on the main screen to open the Edit Settings screen. You can also modify the depth at which statistics are collected by the AWR by clicking the Collection Level link. Clicking this link opens the Initialization Parameters screen in which you can specify any of the three pre-defined collection levels.

Gathering snapshots too frequently requires additional space in the SYSAUX tablespace and adds additional database overhead each time the statistics are collected. Once AWR snapshots are taken and stored in the database, the Automatic Database Diagnostic feature uses the statistics as described in the next section.

Automatic Database Diagnostic Monitoring

Following each AWR statistics collection process, the Automated Database Diagnostic Monitoring (ADDM) feature automatically analyzes the gathered statistics and compares them to the statistics gathered by the previous two AWR snapshots. By comparing the current statistics to these two previous snapshots, the ADDM can easily identify potential database problems such as CPU and I/O bottlenecks, resource- intensive SQL or PL/SQL, lock contention, and the utilization of Oracle’s memory structures within the SGA.

Based on these findings, the ADDM may recommend possible remedies. The goal of these recommendations is to minimize DB Time. DB Time is composed of two types of time measures for non-idle database users: CPU time and wait time. This information is stored as the cumulative time that all database users have spent either using CPU resources or waiting for access to resources such as CPU, I/O, or Oracle’s memory structures. High or increasing values for DB Time indicate that users are requesting increasingly more server resources and may also be experiencing waits for those resources, which can lead to less than optimal performance. In this way, minimizing DB Time is a much better way to measure overall database performance than Oracle’s old ratio-based tuning methodologies.

DB Time is calculated by combining all the times from all non-idle user sessions into one number. Therefore, it is possible for the DB Time value to be larger than the total time that the instance has been running.

Once ADDM completes its comparison of the newly collected statistics to the previously collected statistics, the results are stored in the AWR. You can use these statistics to establish baselines against which future performance will be compared, and you can use deviations from these baseline measures to identify areas that need attention. In this manner, ADDM allows you to not only better detect and alert yourself to potential management and performance problems in the database, but also allows you to take corrective actions to rectify those problems quickly and with little or no manual intervention.

Using EM Database Control to View ADDM Analysis

EM Database Control graphically displays the results of the ADDM analysis on several screens, including: The Performance Findings link under the Diagnostic Summary section of the EM Database Control main screen The Performance tab of the EM Database Control main screen The ADDM screen located by clicking the Advisor Central link at the bottom of the EM Database Control main screen

The EM Database Control Performance Findings Link

The EM Database Control main screen contains a section called Diagnostic Summary. One of the links under this section is called Performance Findings.

ADDM suggests three options for improving the performance of the I/O on this system: Stripe and mirror (also known as SAME) all datafiles across multiple disk drives. Increase the number of physical disk drives. Consider implementing Oracle’s Automatic Storage Management feature.

The SAME, or Stripe and Mirror Everything, methodology refers to a database file configuration strategy that is described in this white paper on the Oracle Technology Network: http://otn.oracle.com/ deploy/availability/pdf/OOW2000\_same\_ppt.pdf.

ADDM has essentially identified the SALES\_HISTORY table as the source of excessive I/O and recommends that you run the Segment Advisor utility against this table to generate recommendations for improving its performance.

The EM Database Control Performance Tab

You can also click the Performance tab on the EM Database Control main screen to view performance data collected by AWR and analyzed by ADDM. ADDM uses its findings to populate the Sessions: Waiting And Working section of the Performance screen, Using this section of the Performance screen, you can drill down into detailed information in 11 areas that have been identified as having an impact on performance, from User I/O thorough CPU Used. By clicking the User I/O link, you can drill down into detailed information about user I/O.

The lines on the graph show which of the events in the graph’s legend experienced the most activity during that snapshot period. The graph output indicates that most of the user I/O activity is experiencing waits for the database event "db file scattered read." This event is caused by the I/O activity that occurs when Oracle experiences a wait while performing a sequential disk read of contiguous blocks from a datafile into the buffer cache—usually when a table is being accessed using a full table scan or fast full index scan.

The Performance screen also contains a Performance Overview section near the bottom that summarizes, in pie graphs, the top SQL and top session wait events identified by ADDM.

Clicking the links in the boxes next to either of these graphs displays details about that item. For example, clicking the link for the SQL statement that experienced the most wait time (35 percent on the graph).

Clicking the link at the bottom of this same screen allows you to view the execution plan for this statement.

If the execution plan for this query shows that a full table scan of the 900,000+ row SALES\_ HISTORY table is occurring, then you can see how it might experience I/O waits while retrieving its rows. To view the tuning recommendations that ADDM has generated for this statement, click the Run SQL Tuning Advisor button at the bottom of the screen.

The Advisor Central Screen

The Advisor Central screen also contains ADDM findings. The link for the Advisor Central screen is at the bottom of the EM Database Control main screen. Click this link to display the Advisor Central screen, the top portion of which is shown in

Click the ADDM link in the Advisors section of this screen to display a graph, shown in

As stated earlier, the ADDM automatically compares the most recent AWR snapshot with the last two AWR snapshots when formulating its recommendations. However, you can use this Create ADDM Task screen to manually select any two AWR snapshot times and formulate ADDM recommendations for activity that occurred between those two points in time. To start this process, click the Period Start Time radio button and then select a start date and time by clicking the point in the graph’s timeline that corresponds to the beginning period that you want to use. Repeat this process to specify the end process time stamp.

Click OK to analyze the database for possible performance problems between the two specified points in time. You can also manually perform an ADDM analysis without the use of EM Database Control by using the addmrpt.sql script located in $ORACLE\_HOME/rdbms/ admin on Unix systems and %ORACLE\_HOME%\rdbms\admin on Windows systems. See Chapter 6 of Oracle Database Performance Tuning Guide 10 g Release 1 (10.1), Part Number B10752-01, for details on how to use this script. The results of this analysis is displayed at the bottom of the ADDM screen that is displayed when the analysis is complete.

Although using EM Database Control to view ADDM results is by far the simplest way to review ADDM recommendations, you can also query the ADDM data dictionary views directly as well. Some of these data dictionary views are discussed in the following section.

Using Data Dictionary Views to View ADDM Analysis

You can use more than 20 data dictionary views to examine the results of ADDM’s activities. Four commonly used ADDM views that store the recommendation information we saw in the EM Database Control pages are described in the following table.

ADDM Data Dictionary Views

View Name Description

DBA\_ADVISOR\_FINDINGS Describes the findings identified by the ADDM analysis

DBA\_ADVISOR\_OBJECTS Describes the objects that are referenced in the ADDM findings and recommendations

DBA\_ADVISOR\_RECOMMENDATIONS Describes the recommendations made based on ADDM findings

DBA\_ADVISOR\_RATIONALE Describes the rationale behind each ADDM finding

The following SQL statement shows a sample query on the DBA\_ADVISOR\_FINDINGS data dictionary view that identifies the type of performance problem that is causing the most impact on the database:

SQL> SELECT task\_id, type, message

2 FROm dba\_advisor\_findings

3 WHERE impact= (select MAX(impact) FROM dba\_advisor\_findings);

TASK\_ID TYPE MESSAGE

------- -------- -------------------------------------------------

164 PROBLEM SQL statements consuming significant database time were found.

The output from this query shows that SQL statements being executed in the database are contributing to the poor database performance. By itself, the DBA\_ADVISOR\_FINDINGS data dictionary view does not identify which SQL statements are consuming the database time. Instead, these are shown in the DBA\_ADVISOR\_OBJECTS data dictionary view and are identified by the TASK\_ID value shown in the query on DBA\_ADVISOR\_FINDINGS. A query on that view, using the TASK\_ID of 164 returned by the ADDM session that had the potential for the greatest database impact, returns the SQL statements shown here:

SQL> SELECT attr4

2 FROM dba\_advisor\_objects

3 WHERE task\_id = 164;

ATTR4

----------------------------------------------------------------------

UPDATE customers SET credit\_limit=credit\_limit\*1.15 WHERE cust\_id=:B1

DELETE FROM sales WHERE time\_id BETWEEN ’01-JAN-00’ and ’01-

JAN-01’; UPDATE sales\_history SET quantity\_sold =

quantity\_sold+10 WHERE

CHANNEL\_ID := B1

SELECT COUNT(\*) FROM Sales\_history;

SELECT DISTINCT channel\_id FROM sales\_history;

This query shows all the SQL statements that were captured by the AWR during the snapshot period and that were used in the ADDM analysis for that same period.

The DBA\_ADVISOR\_ACTIONS data dictionary view shows the ADDM recommendations for each finding. The following query shows the recommendations for correcting the performance issues associated with TASK\_ID 164, which was identified earlier as being the costliest database activity:

SQL> SELECT TRIM(attr1) ATTR1, TRIM(attr2) ATTR2, TRIM(attr3) ATTR3

2 FROM dba\_advisor\_actions

3 WHERE task\_id = 164;

ATTR1 ATTR2 ATTR3

---------- ------ ----------

log\_buffer 262144 15728640

db\_cache\_size 25165824 50331648

undo\_retention 900 363

This output indicates that ADDM recommends that the values for LOG\_BUFFER, DB\_CACHE\_ SIZE, and UNDO\_RETENTION all be changed from their current values to 15,728,640 bytes, 50,331,648 bytes, and 363 seconds, respectively.

If you want to see the rationale behind each of the actions shown in DBA\_ADVISOR\_ACTIONS, query the DBA\_ADVISOR\_RATIONALE data dictionary view. The DBA\_ADVISOR\_RATIONALE view stores the ADDM recommendations that ADDM has formulated based on the AWR data like those stored in DBA\_ADVISOR\_FINDINGS and DBA\_ADVISOR\_OBJECTS. The following example shows a sample query on the DBA\_ADVISOR\_RATIONALE view using the TASK\_ID of 164 identified earlier:

SQL> SELECT message

2 FROM dba\_advisor\_rationale

3 WHERE task\_id = 164;

MESSAGE

---------------------------------------------------------

Buffer cache writes due to small log files were consuming

significant database time.

The buffer cache was undersized causing significant read I/O. The value of "undo retention" was 900 seconds and the longest running query lasted only 330 seconds. This extra retention caused unnecessary I/O.

As you can see from the complexity of these examples, examining the ADDM results via the EM Database Control is much easier than accessing the data dictionary views via SQL. From a practical standpoint, you would run SQL queries against these ADDM views only if the EM Database Control were unavailable.

To gain further insight into the recommendations and information gathered by the ADDM, Oracle 10g also provides several advisor utilities in the EM Database Control.

ADDM Diagnostic Advisors

The ADDM utility also provides several tuning and diagnostic advisors that you can use to examine several common problem areas in your database and then offer suggestions for improving those areas. The diagnostic and tuning advisors include the following: SQL Tuning Advisor SQL Access Advisor Memory Advisor Mean Time To Recover Advisor Segment Advisor Undo Management Advisor

The links to all these advisors are available by clicking the Advisor Central link at the bottom of the EM Database Control main screen.

The SQL Tuning Advisor As you saw earlier, the ADDM utility allows you to drill down and view the actual SQL of the statements that are contributing to increasing DB Times. Once the SQL has been identified, you can use the SQL Tuning Advisor to attempt to formulate more efficient SQL execution plans for the offending SQL.

There are four options for the SQL Tuning Advisor: Top SQL, SQL Tuning Sets, Snapshots, and Preserved Snapshot Sets.

Types of Analysis for the SQL Tuning Advisor Collection Method Current And Recent SQL Activity

Import Workload From SQL Repository

Import SQL From A Table

Advisors

The following examples use the Top SQL option of the SQL Tuning Advisor. You can view the Top SQL statements in two ways, Spot SQL and Period SQL; each are represented by a tab in the Top SQL screen.

AWR automatically assigns a system-generated name to each SQL statement that is recorded. These names are a combination of 13 numbers and lowercase letters.

Spot SQL graphically displays all the resource wait, I/O, and CPU statistics for SQL statements that have been active in the most recent five-minute interval. By examining the graphical output, you can readily identify which SQL statements caused spikes in these three areas.

between two points in time. By examining the graphical output on this screen, you can also easily identify which periods of time experienced spikes in the areas of resource waits, I/O, and CPU.

Regardless of whether you isolate your problem SQL statements using ADDM, Spot SQL, or Period SQL, the process of analyzing that SQL using the SQL Advisor is the same. First, you need to decide which statement you wish to analyze and then click the Run SQL Tuning Advisor button on the SQL Details page after selecting that statement from ADDM, Spot SQL, or Period SQL.

Clicking the Run SQL Tuning Advisor button opens the Schedule Advisor screen. You use this screen to formulate the job that will be submitted to the database when the advisor is actually executed. You specify three elements when running the SQL Tuning Advisor: Description, Scope, and Schedule. Each of these elements is described in the following table.

Elements for Scheduling a SQL Tuning Advisor Job

Link Name Description

Top SQL Allows to identify and tune the most resource-intensive SQL statements

SQL Tuning Sets Allows to group several related SQL statements together for analysis

Snapshots Allows to select a specific snapshot to analyze

Preserved Snapshot Sets Allows to create and analyze a collection of related snapshots

Lets you specify a schema against which to generate SQL tuning recommendations Select the Current And Recent SQL Activity radio button and click Next to display the Recommendation Options screen.

The Advisor can focus its efforts on three options.

To assess whether the SQL statements will benefit from the addition of an index To assess whether the SQL statements will benefit from the addition of a materialized view

To check for both indexes and materialized views

You can specify the depth level of the analysis as either Limited or Comprehensive.

Because Comprehensive tuning analysis can consume a lot of server resources, schedule it when user activity against the database is at its lowest.

Clicking Next after selecting the analysis options displays the scheduling options. Like previous EM Database Control scheduling screens, the time that the analysis will begin, its duration, and its frequency can all be defined on this page.

Finally, clicking Next displays the SQL Access Advisor: Review screen.

This screen displays all the options that you specified before clicking Submit to actually begin the analysis.

When the analysis is complete, you return to the Advisor Central screen and a link to the results of the analysis is displayed at the bottom of the screen.

Click the link in the completed analysis table to display the recommendations that the SQL Access Advisor created for SQL statements that were analyzed.

Clicking Recommendation ID (1) displays the details of the SQL Access Advisor recommendation.

This recommendation indicates that building an index will improve the access path of the SQL statement shown at the bottom of the screen. You can change the default index name of \_IDX$$\_02930001 suggested by the advisor by typing a new name in its place. Additionally, you can also specify the tablespace where the new index should be stored. Clicking OK returns you to the recommendation summary screen. Click the Schedule Implementation button to schedule the actual creation of the index.

The Memory Advisor Both the SQL Tuning and Access advisors focus on identifying and tuning the SQL that is having the greatest impact on increasing overall DB Time. Alternatively, you can use the Memory Advisor to gather more global tuning recommendations about all aspects of Oracle’s memory structures, including the SGA and user memory structures. You can also access the Memory Advisor from the Advisor Central screen.

The sizes of four components of the SGA—the Shared Pool, Buffer Cache, Large Pool, and Java Pool—are summarized in tabular and in graphical form. Clicking the Advice buttons next to the Shared Pool and Buffer Cache values tells the Memory Advisor to formulate tuning recommendations for that memory structure.

This output shows that as the size of the buffer cache is increased from its current size of 24MB to the recommended size of 48MB, the overall physical reads are reduced by 20 percent (a relative decline from 1.0 to 0.80). This would be beneficial because reads from memory are thousands of times faster than reads from disk.

If you enable Automatic Shared Memory Management the ADDM can automatically adjust the size of the Buffer Cache based on its findings.

The main Memory Advisor screen also has a second tab that shows the results of the advisor’s analysis of the PGA (Program Global Area) memory that is allocated to each user process.

Like the SGA screen, the PGA recommendations screen also has an Advice button; click it to view the Memory Advisor’s recommendations for improving PGA performance.

Using cache hit ratios as the primary basis for performance analysis is not recommended. In several situations, high cache hit ratios can exist even when overall performance is poor.

The Mean Time To Recover (MTTR) Advisor The preceding advisors focused primarily on improving database performance by minimizing user waits for I/O, CPU, and other resources with the goal of minimizing each user’s overall DB Time. The Mean Time To Recover (MTTR) Advisor is not concerned with minimizing DB Time, but instead tries to formulate recommendations that minimize the time it takes to perform instance recovery in the case of instance failure. Instance failure can occur when the host server crashes, when any critical SGA background process fails, or if the instance is shut down using the ABORT option. Instance recovery occurs automatically on the first startup following the instance failure. During instance recovery, Oracle uses the undo segments and online redo logs to roll back any uncommitted transactions that were "in flight" when the instance crashed to ensure that all committed transactions are written to disk. As a DBA, you often try to minimize the time it takes to perform this instance recovery so that the database can be brought up quickly.

In previous Oracle releases, you could use the FAST\_START\_MTTR\_TARGET initialization parameter to specify the maximum allowable instance recovery time (in seconds). This parameter must be set to a non-zero value for the new features described next to work.

The MTTR Advisor analyzes the database during regular processing and makes recommendations about database and instance parameters that can be modified in order to meet your instance recovery goals.

If this MTTR is not acceptable because a Service Level Agreement (SLA) requires a 15- second instance recovery time, then you can specify a new MTTR value of 15 in the Desired Mean Time To Recover box.

The MTTR Advisor screen also has Media Recovery and Flash Recovery sections, which are described in more detail in Chapter 10, "Implementing Database Backups" and Chapter 11, "Implementing Database Recovery."

The Segment Advisor You use the Segment Advisor to identify segments that might benefit from a shrink operation. Segments that can be shrunk are those that the Segment Advisor has found to be needing less space than they are currently allocated. By shrinking or compressing these segments, space is returned to the database for use by other objects, and the total number of I/Os needed to access these objects is reduced, potentially improving the performance of SQL statements that access these objects.

You can analyze potentially compressible segments either at the segment level or at the tablespace level. In addition, you can also specify the degree to which the segments are examined at two levels: Limited and Comprehensive. If you select the Schema Objects and Limited options and then click Continue, the Segment Advisor: Schema Objects screen.

Initially, no segments are listed for analysis, but click the Add button to specify which segments you want the Segment Advisor to examine.

You can specify how much time the Segment Advisor can take when analyzing the specified segments and how long to store the results of the analysis in the repository. Click Next on this and subsequent screens to display the familiar job scheduling and submissions screens. Once the analysis of the selected segments or tablespaces is complete, the results are displayed at the bottom of the Advisor Central screen, along with all other submitted job results. Click the Segment Advisor Job link to display the Segment Advisor Task screen.

The output on this screen shows that the SUPPLEMENTARY\_DEMOGRAPHICS table owned by the user SH has been identified as a segment that will benefit from a shrink operation, reducing its allocated space from 4MB to the 1.193MB that the table actually needs to store its data. When shrinking an object identified by the Segment Advisor, there are two shrink options: Compact Segments and Compact Segments And Release Space.

The Compact Segments option compacts the rows in the SUPPLEMENTARY\_DEMOGRAPHICS table, but does not release the newly freed space back to the tablespace. This option allows you to put off the more resource-intensive operation of actually releasing the space until later.

The Compact Segments And Release Space option compacts the space in the SUPPLEMENTARY\_DEMOGRAPHICS table and also releases the unused space back to the tablespace at the same time. Choosing either of these two compression options displays the familiar job submission screen, which submits the compaction job in the background.

If you have a table that has been compressed using the Compact Segments option and thus does not have its space released, you can later release this space using the SHRINK SPACE option of the ALTER TABLE command, for example: ALTER TABLE supplementary\_demographics SHRINK SPACE.

In order for the Segment Advisor to modify segments effectively, you need to enable the ROW MOVEMENT attribute of the affected segments. You can do so using the Options tab in the Edit Table screen or the ALTER TABLE … ENABLE ROW MOVEMENT command.

The Undo Management Advisor The Undo Management Advisor helps you monitor and proactively respond to potential problems in a common trouble area of any transactional database system: undo segments. When a user starts a DML (Data Manipulation Language) transaction, the before-image of the changed data is buffered in the Database Buffer Cache in the SGA. Copies of these buffers are also written to an undo segment, which is stored in the database’s undo tablespace. The before-image data stored in the undo segment is used for three important purposes:

It can be used to restore the original state of the data if the user performing the DML command issues a ROLLBACK command. It provides a read-consistent view of the changed data to other users who access the same data prior to the DML user issuing a COMMIT command. It is used during instance recovery to undo uncommitted transactions that were in progress just prior to an instance failure.

In previous releases of Oracle, undo segments were referred to as rollback segments because they are used to roll back a transaction when a ROLLBACK command is issued. However, this term is now generally used to refer to manually managed undo segments, not the system managed undo segments that Oracle recommends be used in Oracle 10g.

Once a transaction is assigned to an undo segment, the transaction never switches to a different undo segment, even if the original undo segment was not the most appropriate choice. Because of this, undo segment tuning can be one of the most elusive aspects of database administration. Even when undo activity has reached a steady state and no problems are apparent, the right combination of transactions can cause an undo segment error. This can lead to frustrating undo segment–tuning problems that never completely go away. The goals of undo segment tuning usually ensure the following:

That database users always find an undo segment to store their transaction before- images without experiencing a wait That database users always get the read-consistent view that they need to complete their transactions That the database undo segments do not cause unnecessary I/O Every database contains at least one undo segment, which is the SYSTEM undo segment. This undo segment is used only for data dictionary read consistency and transaction control.

The most common undo segment–related error message is ORA-01555: Snapshot Too Old. This error can occasionally occur when some users are running long queries and others are simultaneously modifying the data being queried. This scenario can cause the session running the long-running query to be unable to build a read- consistent view of the database, thus causing the ORA-01555 error message.

The Undo Management Advisor helps prevent undo-related problems in the database by monitoring and analyzing undo activity before making recommendations for improving undo performance.

To open the Undo Management Advisor, click the Undo Management link in the Advisor Central screen in EM Database Control.

To meet this undo retention period, the Undo Management Advisor recommends increasing the size of the undo tablespace to 703MB. For details, click the Undo Advisor button to display the screen.

This output graphically shows how the undo retention time (in minutes along the bottom of the graph) increases to 16 minutes if the size of the undo tablespace increases to 431MB. The graph also shows that the best possible undo retention time that can be obtained with the current undo tablespace size of 200MB is 6 minutes. Implementing these recommendations will minimize database management and performance problems related to undo segments.

ADDM Alerts In addition to monitoring and making recommendations on SQL, memory, mean time to recover, segments, and undo activity, ADDM can also be used to proactively monitor the database for other types of problems related to memory, I/O, and CPU utilization, as well as security and space management. To do so, you use ADDM alerts.

ADDM alerts are also an integral part of the ADDM architecture. They notify you when a management or performance issue occurs and begin taking corrective actions—if you configured such actions. By default, the alert notifications are sent to the EM Database Control main screen.

You can also configure alerts so that they are sent to you via e-mail. To do so, click the Setup link at the top of the EM Database Control screen to display the Setup screen. Click the Notification Methods link on the left to open the Notification Methods screen.

You’ll need to supply three pieces of information: The IP address of your network’s SMTP mail server The name of the user from whom the e-mail address will be sent The e-mail address of the user sending the notification e-mails

smtp.acme.com is the server through which EM Database Control will send the ADDM alert e-mails. The name of the user from whom the e-mail address will be sent is shown as OEM. The e-mail address of the user who will be sending the notification e-mails appears in the From box when you receive an alert notification via e-mail.

After you add the e-mail configuration entries, click Apply to save them.

Click the Test Mail Servers button on the Notification Methods screen to confirm that the configuration you’ve entered is correct.

Once EM Database Control knows how to send notification e-mails when ADDM alert events occur, you need to tell EM Database Control to whom the notification e-mails should be sent. For the user SYS, click the Preferences link at the top of the EM Database Control main screen. To configure e-mail notification information for other users, click the Edit button on the Administrators screen.

You can then enter the e-mail address of the DBA who should receive the notification in the box under the E-mail Address column before clicking Apply to save the change. Click the Test button to send a test e-mail message to the address supplied and confirm the e-mail connectivity between EM Database Control and the DBA’s e-mail address.

You can also set the message format, long or short, at this time. The short format is useful when you are e-mailing the notifications to a text pager or a cell phone.

After you configure the notification methods, alerts are sent to both the EM Database Control main screen and to the e-mail address specified.

An alert is triggered whenever a monitored event occurs or when a specified database threshold, called a metric, is surpassed. Metrics are the statistical performance measurements that are collected and stored in the AWR repository. The ADDM utility then gathers additional database statistics and compares them against the baseline metrics in order to monitor, diagnose, and remedy management problems or poor database performance. There are four default ADDM alerts configured in each database as described in the following table. Oracle 10g has several additional predefined ADDM alerts, which require a small amount of additional configuration before using. These alerts are defined on the Manage Metrics screen of the EM Database Control. To open this screen, click the Manage Metrics link at the bottom of the EM Database Control main screen.

Some of the alerts include the following:

The archive destination is more than 80 percent full. The archive process is hung and returns an error message. The superuser SYS is connecting to the database.

The two alert levels Warning and Critical allow you to achieve greater granularity. For example, you might want two thresholds set up with regard to the archive destination. One might be a warning threshold that triggers an alert when the archive destination is 80 percent full—causing a message to be displayed on the EM Database Control main screen. In addition, you might want to set up a critical threshold so that you receive an e-mail whenever the archive destination device is 90 percent full. In this manner, you can escalate a potential problem from an EM Database Control console message to an e-mail alert as the problem gets worse.

You can also use warning and critical alerts to distinguish between lower severity problems, such as statistics indicating temporary poor performance, and higher severity problems, such as ORA-0600 error messages in the database Alert log. You can achieve this by defining only warning thresholds for lower severity alerts and defining warning and critical alerts for higher severity problems.

Instead of specifying a set value, such as 50 percent and 75 percent for CPU utilization alert levels, you can also raise alerts when CPU utilization exceeds a baseline metric. Baseline metrics are gathered during a period of processing that represents normal database activity. Using these baselines, you can raise alerts when relative performance problems occur. For example, rather than raising an alert when the CPU utilization is 50 percent of the available CPU cycles, you can raise an alert when CPU utilization is 50 percent more than the baseline CPU utilization—which itself could be 85 percent of CPU cycles.

To gather baseline metrics, use the Options link on the Metric Baselines tab in the Manage Metrics screen.

The output shows that no metric baselines have yet been gathered. The AWR can store several baselines, any of which can be used as the basis for the alert system. To gather the first baseline metric, click Create to display the Create Metric Baseline screen.

Using the Create Metric Baseline screen, you can minimally assign a name and date to the baseline statistics that will be gathered. Optionally, you can also assign a time of day, warning, and alert thresholds.

If no value is supplied for Hour Of Day, baseline statistics are gathered for the entire 24- hour period for the date specified.

The Metric Baselines tab in the Manage Metrics screen Click Go to capture the current database metrics and then display them at the bottom of the Create Metric Baseline screen.

Using this screen, you can deselect any metrics that you don’t want to include in the baseline (all metrics are included by default) or modify the baseline values that were used. Once you tailored these metrics to your needs, you can store them by clicking OK at the top of the page, creating the baseline metric called Initial Server Stats.

The first column shows the name of the metric being monitored, the second column displays the warning values, and the third column displays the critical values that were calculated for that threshold. For example, the baseline warning threshold for the metric Current Open Cursors Count is 784, and the critical threshold is 826.

Computation of Thresholds by ADDM: When baseline metrics are gathered, the AWR stores the high value and the low value for each metric. The technique that ADDM uses to compute a baseline threshold depends on the comparison operator that is used for specifying the threshold, and on the associated high and low value.

If the metric comparison operator is a greater than (>), the warning threshold for the metric is computed as Metric High Value \* (1 + Warning Percentage/100). For example, if the high value for the metric Current Open Cursors Count is 500, the 85 percent warning threshold is 500 \* (1 + 85/100) or 925, raising a warning alert whenever the Current Open Cursors Count exceeds 925. Likewise, the critical threshold is 500 \* (1 + 95/100) or 975, raising a critical alert whenever the Current Open Cursors Count exceeds 975.

If the metric comparison operator is a less than (<), the warning threshold for the metric is computed as Metric Low Value \* (1 + Warning %/100). For example, if the low value for the metric Large Pool Free % is 300MB, the 85 percent warning threshold is 300 \* (1 – 85/100) or 45MB, raising a warning alert whenever the percentage of free space in the Large Pool falls below 45MB. Likewise, the critical threshold is 300 \* (1 – 95/100) or 15, raising a critical alert whenever the free space in the Large Pool falls below 15 percent.

Because more than one baseline can be stored in the AWR at one time, EM Database Control gives you a way to choose the baseline that you want to use for alert thresholds. Suppose you have the two baselines.

To use the metrics associated with the Overnight Processing baseline as the basis for ADDM alerts, simply select that option and click the Copy Thresholds From Metric Baseline button. EM Database Control gives you an opportunity to modify selected metric parameters if needed, before you update the threshold values and see the screen. Job Element Description

Scope

Schedule Default ADDM Alerts

Alert Description Tablespace Space Usage Alerts you whenever a tablespace’s free space falls below 15 percent and again when it falls below 3 percent. Snapshot Too Old Alerts you whenever the ORA-01555 error message described earlier) occurs. Recovery Area Low On Free Space Alerts you when the Flash Recovery area is low on free space. Resumable Session Suspended Alerts you whenever an operation that can be resumed goes into a suspended state. Description Description of the SQL statement that is being analyzed. Although this is an optional element for submitting a job, it is useful because otherwise Oracle uses a system-generated name for the SQL statement— 85zq7jwf3x3qg.

The depth to which the advisor should examine the statement. Possible values are Limited and Comprehensive. The deeper the analysis, the greater the potential for uncovering additional tuning options. You can specify the maximum time that the advisor should spend performing a comprehensive analysis.

Specifies when to execute the analysis job. The default value is Immediately, but you can also schedule the job to execute at some time in the future. Description Lets you select the SQL to analyze from what is currently in the SGA

Lets you analyze a SQL tuning set that was created using the SQL Tuning Advisor User-Defined Workload;

Lets you perform a tuning analysis on a workload that is not currently running in the database. After you specify the job submission elements, click OK to begin analyzing the specified statement.

Choosing the Comprehensive level of analysis can be time-consuming and resource intensive.

To review the SQL that will be issued to create this index, click the Show SQL button. If desired, you can modify the generated SQL to rename the index or change its storage parameters. Otherwise, clicking OK submits the job to create the new index. Like the SQL Tuning Advisor, the job can be executed immediately or scheduled to run some time in the future.

Like the SQL identifier, each recommendation is also assigned a system-generated ID made up of a combination of 13 numbers and lowercase letters.

The SQL Access Advisor You’ve seen how you can use the SQL Tuning Advisor to identify and create an index to minimize the DB Time for a particular statement. The SQL Access Advisor provides additional support for finding potential schema modifications that you can use to reduce the amount of I/O, CPU, and wait time for a given SQL statement.

The SQL Access Advisor shows four ways in which you can select the SQL statement to be analyzed. The following table compares these four techniques.

Comparison of Four Techniques for Selecting SQL to Analyze

Performance Monitoring

Although AWR, ADDM, and advisors all help you proactively monitor and manage your databases, you can use additional performance-specific features of EM Database Control to further enhance the performance of your database.

However, you should measure exactly how the system is currently performing before beginning any tuning effort. The baseline metrics described in the previous section are a good example of this type of measurement. Using this benchmark, you can then compare the performance of the system after any tuning changes and evaluate their impact. In addition to these baseline metrics, you can use additional sources of tuning information to monitor and tune database performance. The following section describes these sources.

Sources of Tuning Information

The EM Database Control provides a wealth of information for improving database monitoring and management, but you also need to be aware of several other sources of information about database performance, including:

The Alert log

Background and user trace files

Dynamic performance views

Data dictionary views

The Alert Log

The Oracle Alert log records informational and error messages for a variety of activities that have occurred against the database during its operation. These activities are recorded in chronological order from the oldest to most recent. The Alert log is found in the background\_dump\_dest directory specified in the PFILE/SPFILE.

The Alert Log frequently indicates whether gross tuning problems exist in the database. Tables that are unable to acquire additional storage, sorts that are failing, and problems with undo segments are all examples of tuning problems that can show up as messages in the Alert log. Most of these messages are accompanied by an Oracle error message.

ADDM provides a mechanism for sending an alert whenever Oracle errors are detected in the Alert log. Click the Alert Log Content link on the EM Database Control main screen to see the most recent messages.

Background and User Trace Files

Oracle trace files are text files that contain session information for the process that created them. Trace files can be generated by the Oracle background processes, through the use of trace events, or by user server processes. These trace files can contain useful information for performance tuning and system troubleshooting. Background process trace files are found in the background\_dump\_dest and user\_dump\_dest directories specified in the PFILE/SPFILE.

The 10046 trace event, which can be activated at the instance or session level, is particularly useful for finding performance bottlenecks.

See Note 171647.1 at http://metalink.oracle.com for a discussion of using the 10046 trace event as a tuning technique.

Dynamic Performance Views Oracle 10g contains approximately 350 dynamic performance views.

A partial listing of some of the V$ views that are frequently used in performance tuning:

Name Description

V$SGASTAT Shows information about the size of the SGA’s components.

V$EVENT\_NAME Shows database events that may require waits when requested

by the system or by an individual session. There are

approximately 200 possible wait events.

V$SYSTEM\_EVENT Shows events for which waits have occurred for all sessions

accessing the system.

V$SESSION\_EVENT Shows events for which waits have occurred, individually

identified by session.

V$SESSION\_WAIT Shows events for which waits are currently occurring, individually

identified by session.

V$STATNAME Matches the name to the statistics listed only by number in V$SESSTAT

and V$SYSSAT.

V$SYSSTAT Shows overall system statistics for all sessions, both currently and

previously connected.

V$SESSTAT Shows statistics on a per-session basis for currently connected

sessions.

V$SESSION Shows current connection information on a per-session basis.

V$WAITSTAT Shows statistics related to block contention.

In general, queries that incorporate V$SYSSTAT show statistics for the entire instance since the time it was started. By joining this view to the other relevant views, you get the overall picture of performance in the database. Alternatively, queries that incorporate V$SESSTAT show statistics for a particular session. These queries are better suited for examining the performance of an individual operation or process. The EM Database Control makes extensive use of these views when creating performance- related graphs.

Data Dictionary Views

Depending on the features and options installed, there are approximately 1300 DBA data dictionary views in an Oracle 10g database.

A Sampling of Data Dictionary Views:

Name Description

DBA\_TABLES Table storage, row, and block information

DBA\_INDEXES Index storage, row, and block information

INDEX\_STATS Index depth and dispersion information

DBA\_DATA\_FILES Datafile location, naming, and size information

DBA\_SEGMENTS General information about any space-consuming segment

in the database.

DBA\_HISTOGRAMS Table and index histogram definition information

DBA\_OBJECTS General information about all objects in the database,

including tables, indexes, triggers, sequences, and partitions.

The DBA\_OBJECTS data dictionary view contains a STATUS column that indicates, through the use of a VALID or an INVALID value, whether a database object is valid and ready to be used or invalid and in need of some attention before it can be used. Common examples of invalid objects are PL/SQL code that contains errors or references to other invalid objects and indexes that are unusable due to maintenance operations or failed direct-path load processes. Some invalid objects, such as some types of PL/SQL code, dynamically recompile the next time they are accessed, and they then take on a status of VALID again. But you must manually correct other invalid objects, such as unusable indexes. Therefore, proactive database management techniques dictate that you identify and remedy invalid objects before they cause problems for database users.

Identifying Unusable Objects Using Data Dictionary

One way to identify invalid objects is to query the DBA\_OBJECTS and DBA\_INDEXES data dictionary views to find any invalid objects or unusable indexes and then correct them using the commands shown here:

SQL> SELECT owner, object\_name, object\_type

2 FROM dba\_objects

3 WHERE status = 'INVALID';

OWNER OBJECT\_NAME OBJECT\_TYPE

------ ---------------------- ----------------- -----------

SH P\_UPDATE\_SALES\_HISTORY PROCEDURE SYS DBA\_HIST\_LATCH VIEW

SQL> ALTER VIEW sys.dba\_hist\_filestatxs COMPILE;

View altered.

SQL> ALTER PROCEDURE sh.p\_update\_sales\_history COMPILE; Procedure altered.

SQL> SELECT owner, index\_name, index\_type

2 FROM dba\_indexes

3 WHERE status = 'UNUSABLE';

OWNER INDEX\_NAME INDEX\_TYPE

-------------- ------------------- ----------

HR JOB\_ID\_PK NORMAL

SQL> ALTER INDEX hr.job\_id\_pk REBUILD;

The ALTER … COMPILE command also works on invalid PL/SQL triggers, packages, package bodies, and functions.

When rebuilding an index using the REBUILD command, the amount of space used by the index is temporarily larger than the actual space needed to store the index. Make sure that adequate space exists in the tablespace before starting the rebuild process; up to 1.5 times the size of the original index is a good rule of thumb.

Identifying Unusable Objects Using EM

EM Database Control also offers a mechanism for fixing invalid database objects. To view it, click the Procedures link on the Administration screen in the EM Database Control.

The Procedures screen shows that the status of the P\_UPDATE\_SALES\_HISTORY procedure is currently INVALID. By selecting the Compile option from the Actions drop- down list, you can begin the recompilation process.

Selecting the Compile option, and then clicking Go causes the procedure to recompile and displays the Edit Procedure screen.

Scrolling to the bottom of this screen shows the messages associated with the error.

Once the error condition is corrected (in other words, the missing SALES\_HISTORY\_VIEW view is re-created), the procedure can again be recompiled using the Compile button, after which the successful completion screen is displayed.

Using the Indexes screen, you can also use EM Database Control to recompile indexes that are in an unusable state. Click the Indexes link in the Administration screen to open the Indexes screen.

To begin the recompilation process, select the Reorganize option from the Actions dropdown list. Click Go to display the second screen of the Reorganize Objects Wizard. Click the Set Attributes or Set Attributes By Type button to modify the index’s attributes—such as the tablespace that it will be stored in or its storage parameters— before rebuilding. Click Next to display the third screen of the Reorganize Objects Wizard.

Using this screen, you can control how the index is rebuilt. For example, you can select the rebuild method, either offline or online, that is best suited for your environment. Offline rebuilds are faster but impact application users who need to access the index. Online rebuilds have minimal impact on users but take longer to complete. You can also specify a "scratch" tablespace where Oracle stores the intermediate results during the rebuild process. Redirecting this activity to another tablespace helps minimize potential space issues in the index’s tablespace during the rebuild. You can also specify whether to gather new optimizer statistics when the index build is complete. Click Next on this screen to generate an impact report.

The output indicates that there is adequate space in the EXAMPLE tablespace for the unusable JOBS\_ID\_PK index. Clicking Next displays the job scheduling screen.

Like the earlier job-scheduling example in this chapter, you can use this screen to assign a job description and to specify the start time for the job. Clicking Next submits the job and rebuilds the unusable index according to the parameters you defined.

Storing Database Statistics in the Data Dictionary

Some columns in the DBA views are not populated with data until the table or index referenced by the view is analyzed. For example, the DBA\_TABLES data dictionary view does not contain values for NUM\_ROWS, AVG\_ROW\_LEN, BLOCKS, and EMPTY\_BLOCKS, among others, until the table is analyzed. Likewise, the DBA\_INDEXES view does not contain values for BLEVEL, LEAF\_ BLOCKS, AVG\_LEAF\_BLOCKS\_PER\_KEY, and AVG\_DATA\_BLOCKS\_PER\_KEY, among others, until the index is analyzed. These statistics are useful not only to you, but also are critical for proper functioning of the cost-based optimizer.

The cost-based optimizer (CBO) uses these statistics to formulate efficient execution plans for each SQL statement that is issued by application users. For example, the CBO may have to decide whether to use an available index when processing a query. The CBO can only make an effective guess at the proper execution plan when it knows the number of rows in the table, the size and type of indexes on that table, and how many the CBO expects to be returned by a query. Because of this, the statistics gathered and stored in the data dictionary views are sometimes called optimizer statistics. In Oracle 10g, there are several ways to analyze tables and indexes to gather statistics for the CBO. These techniques are described in the following sections.

Automatic Collection of Statistics

If you created your database using the Database Configuration Assistant GUI tool, your database is automatically configured to gather table and index statistics every day between 10:00 P.M. and 6:00 A.M. However, the frequency and hours of collection can be modified as needed using EM Database Control.

Manual Collection of Statistics

You can also configure automatic statistics collection for manually created databases using manual techniques. Collecting manual statistics is also useful for tables and indexes whose storage characteristics change frequently or that need to be analyzed outside the normal analysis window of 10:00 P.M. and 6:00 A.M. You can collect manual statistics through EM Database Control or using the built-in DBMS\_STATS PL/SQL package.

Manually Gathering Statistics Using EM

You can use the EM Gather Statistics Wizard to manually collect statistics for individual segments, schemas, or the database as a whole. To start the wizard, click the Maintenance link on the EM Database Control screen. This wizard walks you through five steps, beginning with the Introduction screen.

Click Next on the Introduction screen to open Step 2 in the wizard, and select the method to use when gathering the statistics.

As you can see, three primary statistics options are available: Compute, Estimate, and Delete. The Compute option examines the entire table or index when determining the statistics. This option is the most accurate, but also the most costly in terms of time and resources if used on large tables and indexes. The Estimate option takes a representative sample of the rows in the table and then stores those statistics in the data dictionary. The default sample size is 10 percent of the total table or index rows. You can also manually specify your own sample size if desired. You can also specify the sample method, telling EM Database Control to sample based on a percentage of the overall rows, or blocks, in the table or index. The Delete option removes statistics for a table or index from the data dictionary. If you specify a sample size of 50 percent or more, the table or index is analyzed using the Compute method.

After choosing a collection and sampling method, click Next to display the Object Selection screen.

This screen lets you focus your statistics collection by schema, table, index, partition, or the entire database. Click OK to display the statistics summary screen.

Click the Options button to specify the analysis method, sample method, and other options related to the gathering the table statistics, and then click Next to move to the fourth EM Gather Statistics Wizard screen.

Accepting the default values generates a system job ID and runs immediately for one time only. If desired, you can change the frequency and time for the statistics- gathering process. Click Next to display the final screen of the EM Gather Statistics Wizard.

Once the job is complete, it is moved to the Run History tab on the Scheduler Jobs screen where its output can be inspected for job success or failure and any associated runtime messages.

Manually Gathering Statistics Using DBMS\_STATS

You can also call the DBMS\_STATS PL/SQL package directly from a SQL\*Plus session. Some of the options for the DBMS\_STATS package include the following: Back up old statistics before new statistics are gathered. This feature allows you to restore some or all of the original statistics if the CBO performs poorly after updated statistics are gathered.

Gather table statistics much faster by performing the analysis in parallel. Automatically gather statistics on highly volatile tables and bypass gathering statistics on static tables.

The following example shows how the DBMS\_STATS packages can be used to gather statistics on the PRODUCT\_HISTORY table in SH’s schema:

SQL> EXECUTE DBMS\_STATS.GATHER\_TABLE\_STATS (‘SH’,’PRODUCT\_HISTORY’);

You can use the DBMS\_STATS package to analyze tables, indexes, an entire schema, or the whole database. A sample of some of the procedures available within the DBMS\_STATS package are shown hereunder:

Procedures within the DBMS\_STATS Package

Procedure Name Description GATHER\_INDEX\_STATS Gathers statistics on a specified index GATHER\_TABLE\_STATS Gathers statistics on a specified table GATHER\_SCHEMA\_STATS Gathers statistics on a specified schema GATHER\_DATABASE\_STATS Gathers statistics on an entire database

The presence of accurate optimizer statistics has a big impact on two important measures of overall system performance: throughput and response time.

Important Performance Metrics

Throughput is another example of a statistical performance metric. Throughput is the amount of processing that a computer or system can perform in a given amount of time, for example, the number of customer deposits that can be posted to the appropriate accounts in four hours under regular workloads. Throughput is an important measure when considering the scalability of the system. Scalability refers to the degree to which additional users can be added to the system without system performance declining significantly. New features such as Oracle Database 10g’s Grid Computing capabilities make Oracle one of the most scalable database platforms on the market.

Performance considerations for transactional systems usually revolve around throughput maximization.

Another important metric related to performance is response time. Response time is the amount of time that it takes for a single user’s request to return the desired result when using an application, for example, the time it takes for the system to return a listing of all the customers who purchased products that require service contracts. Telling ADDM about Your Server I/O Capabilities Both throughput and response time are impacted by disk I/O activity. In order for ADDM to make meaningful recommendations about the I/O activity on your server, you need to give ADDM a reference point against which to compare the I/O statistics it has gathered. This reference point is defined as the "expected I/O" of the server. By default, ADDM uses an expected I/O rate of 10,000 microseconds (10 milliseconds). This means that ADDM expects that, on average, your server will need 10 milliseconds to read a single database block from disk.

Using operating system utilities, we performed some I/O tests against our large storage area network disk array and found that the average time needed to read a single database block was about 7 milliseconds (7000 microseconds). To give ADDM a more accurate picture of our expected I/O speeds, we used the DBMS\_ADVISOR package to tell ADDM that our disk subsystem was faster than the default 10 millisecond value: EXECUTE DBMS\_ADVISOR.SET\_DEFAULT\_TASK\_PARAMETER('ADDM', 'DBIO\_EXPECTED', 7000);

Without this adjustment, the ADDM might have thought that our I/O rates were better than average (7 milliseconds instead of 10 milliseconds) when in fact they were only average for our system. The effect of this inaccurate assumption regarding I/O would impact nearly every recommendation that the ADDM made and would have almost certainly resulted in sub-par system performance.

Performance tuning considerations for decision-support systems usually revolve around response time minimization.

EM Database Control can be used to both monitor and react to sudden changes in performance metrics like throughput and response time.

Using EM Database Control to View Performance Metrics

EM Database Control provides a graphical view of throughput, response time, I/O, and other important performance metrics. To view these metrics, click the All Metrics link at the bottom of the EM Database Control main screen to display the All Metrics screen.

Click the metric you want to examine to expand the available information.

Click the Database Block Changes (Per Second) link to display details on the number of database blocks that were modified by application users, per second, for any period between the last 24 hours and the last 31 days.

You can also see that the Warning threshold associated with this metric is 85 and that the Critical threshold is 95 block changes per second and that there were two occurrences of exceeding one or both of those thresholds.

EM Database Control also provides a rich source of performance-tuning information on the Performance tab of the EM Database Control main screen. The Performance tab is divided into three sections of information.

Host

Sessions

Instance Throughput

The Host section of the Performance tab shows run queue length and paging information for the host server hardware. The Run Queue Length graph indicates how many processes were waiting to perform processing during the previous one-hour period. The Paging Rate graph shows how many times per second the operating system had to page out memory to disk during the previous one-hour period.

In addition to other metrics, the Sessions: Waiting And Working section of the Performance tab always shows CPU and I/O activity per session for the previous one- hour period.

This portion of the Performance tab graphically depicts the logons and transactions per second and the physical reads and redo activity per second. You can also view these metrics on a per transaction basis instead of per section, by clicking the Per Transaction button below the graph.

Using EM Database Control to React to Performance Issues

Suppose you notice a drop in database performance within the last 30 minutes. Using the EM Database Control Performance tab, you can drill down into the detail of any of the performance metrics summarized on the tab and identify the source of the problem using techniques.