CERTIFICATION OBJECTIVES

- Use Oracle Enterprise Manager Express, OUI, and DBCA
- Implement Real-Time Database Operation Monitoring
- Perform Emergency Monitoring and Real-Time ADDM
- Generate a Compare Period ADDM Report
- Understand ASH, ADR, and Network Enhancements

✓ Two-Minute Drill

Q&A Self Test
Chapter 1: Enterprise Manager and Database Monitoring

Oracle Database 12c contains numerous great features (some new and some enhanced) that help Oracle DBAs perform their everyday tasks. This chapter discusses the following new monitoring-related features:

- **Oracle Enterprise Manager Express 12c** A new lightweight database monitoring tool.
- **Real-Time Database Operation Monitoring** Helps you monitor the performance of active and recently completed SQL statements by determining where the statement is spending most of its time.
- **Emergency database monitoring** Lets you log in to the database in a diagnostic mode to check who is blocking users and why a database is hanging.
- **Compare Period ADDM** Provides intelligent reporting with an analysis that maps root causes to performance changes.
- **ASH Analytics** Refers to the enhanced functionality available to view ASH data.
- **Automatic Diagnostic Repository (ADR) enhancements** These include the new DDL logging capability and the new DDL and DEBUG log files.
- **Network-related enhancements** New parameters that enable you to set up network compression (this is part of the Advanced Compression Option, which you must license separately).

We’ll start by covering the new lightweight database monitoring tool—Oracle Enterprise Manager Express 12c, also called simply EM Database Express. This tool replaces the familiar Oracle Database Control.
CERTIFICATION OBJECTIVE

Use Oracle Enterprise Manager Database Express, OUI, and DBCA (Exam Objective 1.01)

Oracle Database introduces a new monitoring and management tool named Enterprise Manager Database Express, which replaces the Enterprise Manager Database Control. There are also some changes in the Database Configuration Assistant, specifically regarding the creation of a new type of database named Multitenant Container Database and Pluggable Databases, which you'll learn about in detail in Chapters 2 and 3.

Let’s start with a review of the new EM Database Express tool.

EM Database Express

In previous releases of the Oracle Database, you could use the built-in Database Control, which was really a subset of the much broader Oracle Grid Control monitoring and management tool. In Oracle Database 12c, Oracle has replaced the Database Control tool with the brand-new tool Oracle Enterprise Manager Database Express 12c. Oracle prefers that you use the much more powerful Oracle Enterprise Manager Cloud Control 12c for your monitoring and management needs. OEM Cloud Control 12c isn’t merely a monitoring tool—it has the capability to manage your entire enterprise, including web servers, applications servers, middleware, and hosts. In addition to monitoring and managing capabilities, Cloud Control 12c offers change management and configuration capabilities as well. However, Enterprise Manager Database Express 12c will serve your basic Oracle database management and monitoring needs. In this context, it's important to understand that the new Database Express tool is really lightweight and doesn’t have the range of features offered by the OEM Database Control tool in the earlier releases.

Enterprise Manager Database Control is no longer available in Oracle Database 12c. You must use Enterprise Manager Cloud Control 12c or the new Enterprise Manager Database Express 12c to manage Oracle Database 12c databases.

As with the older Oracle Enterprise Manager Database Control, EM Database Express only lets you manage a single type of target—the Oracle Database. You can
use Oracle Enterprise Manager Cloud Control 12c to manage all types of targets in your environment, including listeners, application servers, hosts, Exadata and Exalogic, and Oracle Fusion Middleware, as well as applications such as Siebel, PeopleSoft, and Oracle E-Business, and even other database platforms such as Microsoft SQL Server. Oracle Enterprise Manager Database Express 12c is designed to help you perform basic administrative tasks such as user, performance, memory, and space management.

**EM Database Express Architecture**
Enterprise Manager (EM) Database Express provides out-of-the-box management capabilities for working with a single Oracle database or cluster. You can monitor, configure, administer, and diagnose a single database with EM Express. EM Express offers far fewer features than the old Database Control. It also uses far less memory and is really a very small tool, taking up less than 100MB space, with minimal memory and CPU demands on the server. The lightweight nature of the tool is due to the fact that the browser handles the rendering, leaving the database to handle just the SQL calls.

EM Express uses a web-based console which communicates with the built-in web server provided by XML DB. This means that EM Express requires that the Oracle XML DB components are installed. This is done automatically when you install Oracle Database 12c.

The EM Express Servlet receives requests for reports and processes them, returning the data in the form of XML pages that are rendered by the web browser built into the XML DB. EM Express's web-based console then shows the reports to the user.

In order to use EM Express, ensure that the initialization parameter DISPATCHERS has a minimum of one dispatcher configured for the XMLDB service with the TCP protocol.

**Configuring EM Database Express**
When you create a database through the Oracle Database Configuration Assistant (DBCA) or choose to create a database when you install Oracle Database 12c, EM Database Express comes configured for your use. All you need to do is to simply point to the correct URL and you're done. If you're creating a database on your own, then of course you'll need to configure the EM Database Express tool—and it's actually pretty simple!
Use Oracle Enterprise Manager Database Express, OUI, and DBCA (Exam Objective 1.01)

First, you must configure either an HTTP or an HTTPS listener port for your database by invoking the `DBMS_XDB_CONFIG` package, as shown here:

```sql
SQL> exec DBMS_XDB_CONFIG.setHTTPsPort(5500);
```

Port 5500 happens to be the default port for EM Database Express (it was the default port number for Oracle Database 11g Database Control as well). You can invoke the `setHTTPPort` procedure to set a HTTP port and not a HTTPS port, as we did here. If you did that, you’d use the HTTP protocol, as in `http://localhost.localdomain:5500/em`. The Oracle installer will by default execute the `setHTTPsPort` procedure.

**It’s also easy to configure EM Database Express through the Oracle Database Configuration Assistant (DBCA), with just a single click, during the creation of a database.**

**Connecting to EM Database Express** You connect to the EM Express Database Express console by going to `https://hostname:5500/em`. For example, in my case, I use the following to connect:

https://alapati.localhost:5500/em

Of course, this example uses the default port (5500), but you can pick any other acceptable port number. Figure 1-1 shows the EM Database Express 12c Home page.

![EM Database Express Home page](image-url)
Granting Access to Non-Administrative Users  You can grant access to EM Database Express to non-administrative users such as application developers for performing such tasks as creating and modifying tables, indexes, views, and so on.

You must grant the EM_EXPRESS_BASIC or the EM_EXPRESS_ALL role to these users in order for them to be able to log in to EM Express. The EM_EXPRESS_BASIC role allows the users to access the EM Express in a read-only mode. This role includes the familiar role SELECT_CATALOG_ROLE, which is the same role that grants similar privileges to users in both OEM Database Control and OEM Cloud Control 12c. The EM_EXPRESS_ALL role grants users the ability to access all the functionality of EM Express.

Dealing with Multiple Instances on a Server  As you know, EM Express helps you manage a single database instance. Therefore, you must use a different port for each database instance, even if they all run on the same server. However, you can definitely use the tool to manage the different instances of a RAC system individually—you just can't see all instances on the tool. You can do only that with Oracle Cloud Control 12c.

If you create a database with the Oracle Database Configuration Assistant (Oracle DBCA), the tool presents the EM Express URL during the database installation and creation.

You can find out the port being used by an instance by executing one of the following two SQL statements, depending on whether you are using the HTTP or the HTTPS protocol (with HTTPS being the preferable protocol, of course):

```
SQL> select dbms_xdb_config.gethttpport from dual;  (for http)
SQL> select dbms_xdb_config.gethttpsport from dual;  (for https)
```

Here's an example:

```
SQL> select DBMS_XDB_CONFIG.GETHTTPSPORT from dual;
GETHTTPSPORT
----------
    5500
SQL>
```

Exploring EM Database Express
The following sections briefly describe the main pages of the EM Database Express. Most of these will be already familiar to you from using the Database Control and the Oracle Grid Control tools.
The Home Page  The Home page shows you the status of the database instance and its uptime. You can view the following from the Home page:

- Incidents—Last 24 Hours
- Performance
- Resources
- SQL Monitor

An important thing to remember here is that you can’t start and stop the database from the EM Database Express interface the way you used to be able to with the old Database Control tool. In fact, you can’t perform any actions that change the database state, such as enabling (or disabling) the archive log mode.

The Oracle EM Database Express tool works for both a single instance database and a cluster database.

EM Express Menus  You can access four menu items from Oracle EM Database Express:

- Configuration  Initialization parameters, memory, database feature usage, and current database properties
- Storage  Tablespaces, undo management, redo and archive logs, and control files
- Security  Users, roles, and profiles
- Performance  Performance Hub and SQL Tuning Advisor

As with Database Control, when you select any action in any of the four main menu areas, the tool creates a SQL command for that purpose and lets you view the command or copy and paste it.

Database Configuration Assistant: New Features  DBCA in Oracle Database 12c lets you create the two new types of databases introduced in this release:

- Pluggable databases (PDBs)
- Multitenant container databases (CDBs)
Chapter 1: Enterprise Manager and Database Monitoring

When you are managing a multitenant container database, DBCA lets you perform the following PDB operations:

- Create a PDB
- Delete a PDB
- Unplug a PDB

Chapters 2 and 3 are devoted to the topics of pluggable databases and multitenant container databases.

Oracle SQL*Developer Enhancements

In Oracle Database 12c, you can use the SQL*Developer tools to perform numerous Oracle database administration actions. In order to perform these DBA tasks in the new DBA Navigator in Oracle SQL*Developer, you must first connect to the instances using a DBA connection. The DBA Navigator has nodes for all defined database connections. Of course, you must have either DBA privileges or specific privileges for performing DBA operations on a database. The DBA tasks you can perform include the following:

- Database startup/shutdown
- Database configuration
- Data Pump export/import
- RMAN backup/recovery
- Security configuration (users, role, profiles, and so on)
- Storage configuration (archive logs, tablespaces, and so on)

CERTIFICATION OBJECTIVE

Implement Real-Time Database Operation Monitoring (Exam Objective 1.02)

Oracle Database 11g introduced Real-Time SQL Monitoring, which helps you learn where the SQL statement is in its execution plan and where it’s spending its time. You can also learn the breakdown of time and resources for all statements that recently completed.
Implement Real-Time Database Operation Monitoring (Exam Objective 1.02)

The goal of SQL Monitoring is to help find the resource-intensive (expensive) code and find out what actually is going on when a SQL statement or a PL/SQL function is executing. The 12c release takes this further by introducing Real-Time Database Operation Monitoring. This tool helps you monitor complex situations such as when you have numerous concurrent jobs running. You can use real-time monitoring capabilities to track the progress of various operations and create active reports. This capability will monitor the operations, package data for analyzing offline, and create the reports without accessing the production system.

You may consider Real-Time Database Operation Monitoring as a superset of all types of database monitoring, with the 11g version Real-Time SQL Monitoring feature being a subset of it. This feature offers more diagnostic options than Real-Time SQL Monitor, ASH, and other tools. Using this feature means you can avoid spending too much time using traditional diagnostic tools such as SQL Trace and DBMS_MONITOR.

**Definition of a Database Operation**

A database operation is one or more SQL or PL/SQL statements running in a single session or being concurrently executed in multiple sessions. You can classify database operations into two types: simple and composite.

A simple database operation consists of a single SQL statement or a single PL/SQL procedure or function. A composite database operation consists of multiple SQL statements and/or PL/SQL procedures and functions during a time interval. The code may be executed in a single session or in multiple concurrent sessions.

A composite DB operation can be one of the following types:

- **Single session** Single session operations can be one of two types. The first type is where only a single session exists for the entire duration of the operation, such as a parallel operation where a set of SQL statements is coordinated by a single session. The other type of single session database operation is where multiple sessions exist, but at any given time, only one session is executing. This can occur, for example, in an application pool, where there’s only a single active user session at any time.

- **Multiple concurrent sessions** Operations that are part of jobs such as ETL batch jobs can be considered a single database operation, although these operations typically involve multiple concurrently executing sessions.

*Idle time* in this context is defined as the time for which a session belongs to a database operation, without executing a SQL statement or a PL/SQL procedure or function.
How Real-Time Database Monitoring Helps You

Real-time database monitoring is designed to help a DBA in numerous situations, including the following:

- Monitoring batch jobs to send alerts when jobs exceed set times
- Diagnosing background processes in order to identify expensive SQL statements
- Diagnosing changes to jobs over time by enabling the DBA to compare current and previous executions of operations
- Monitoring and comparing execution times after a database upgrade

Identifying a Database Operation

You identify a database operation uniquely with the help of two pieces of information: the database operation name that you assign the operation and the database operation execution ID, which helps identify different executions of the same database operation. The database assigns each database operation that you begin an execution ID, but you can also specify your own execution ID if you wish.

In order to identify a SQL or a PL/SQL statement that’s being monitored in the V$SQL_MONITOR view, Oracle uses the following set of values:

- A SQL identifier for the statement (SQL_ID)
- A timestamp when the SQL statement began executing (SQL_EXEC_START)
- An internal identifier to ensure the uniqueness of the primary key (SQL_EXEC_ID)

In addition to these three attributes, you can add your own attributes to help identify a DB operation; for example, you can use the attribute db_name, to which you can assign the value PROD to indicate a production database.

In order to monitor a database operation, you must name the operation as well as associate begin and end points to the operation. The latter is also called “bracketing the database operation.” You can name a database operation in one of the following two ways:

- Insert a start point and an end point for the operation by invoking the BEGIN_OPERATION and END_OPERATION procedures from the DBMS_SQL_MONITOR package.
Implement Real-Time Database Operation Monitoring (Exam Objective 1.02)

- For Java and OCI applications, set a tag using the OCI calls `OCIAttrSet` and `OCIAppCtxSet` and the Java call `setClientInfo`. Alternatively, you can set the tag from the OS environment variable `ORA_DBOP`.

Note that unlike the explicit calls, which clearly define a beginning and end of the operation, applications using tagging don’t have explicit methods to end a database operation. The tag should be set to NULL and sent to the database.

Enabling Database Operation Monitoring

You can enable database operation monitoring at the statement, database operation, or system level. Once you enable monitoring, the database collects and refreshes metadata once every second for the database operations and stores the data in the AWR. You can subsequently view the data through the `V$SQL_MONITOR` view. This section covers how to set monitoring at the various levels.

Statement Level

Use the `MONITOR` hint to enable database operation monitoring at the statement level. By explicitly declaring the `MONITOR` hint, you tell the instance to monitor that SQL statement. Make sure you’ve set the `CONTROL_MANAGEMENT_PACK_ACCESS` parameter to `DIAGNOSTIC+TUNING` before you specify the `MONITOR` hint in a SQL statement, which means that database monitoring should be enabled at the system level. The `MONITOR` hint forces SQL monitoring. Here’s the syntax for the `MONITOR` hint:

```
SELECT /*+ MONITOR */ column_list FROM dual;
```

And here’s an example:

```
SQL> select * /*+ monitor */ from sales order by time_id;
```

You can disable database monitoring for a specific SQL statement by executing the query with the `NO_MONITOR` hint:

```
SQL> select /*+no_monitor*/ from sales order by time_id;
```

Database Operation Level

Even after you enable database operation monitoring, the default behavior is for the database to start tracing an operation only if it’s sufficiently expensive. You can force the monitoring of a database operation when it starts by setting the
FORCED_TRACKING attribute to Y when you start a database operation with the DBMS_SQL_MONITOR.BEGIN_OPERATION function. Here's an example:

```
DBMS_MONITOR.BEGIN_OPERATION (  
    dbop_name IN VARCHAR2,initialization parameter to TYPICAL.
    dbop_eid IN NUMBER :=NULL,
    forced_tracking IN VARCHAR2   :=  Y,
    attribute_list IN VARCHAR2 :=NULL)
RETURN NUMBER;
```

Note that when you set the FORCED_TRACKING attribute to Y, all SQL statements are monitored.

You must license Oracle’s SQL Tuning Pack in order to use the Real-Time Database Monitoring feature. Also, the CONTROL_MANAGEMENT_PACK_ACCESS initialization parameter must be set to DIAGNOSTIC+TUNING, which is the parameter’s default value.

System Level
You must perform the two following actions to enable monitoring at the system level:

- Set the STATISTICS_LEVEL initialization parameter to TYPICAL, which is the parameter’s default value, or to ALL.
- Set the value of CONTROL_MANAGEMENT_PACK_ACCESS to DIAGNOSTIC+TUNING. You do this because database monitoring is a feature of the Oracle Database Tuning Pack.

By default, provided you’ve set CONTROL_MANAGEMENT_PACK_ACCESS appropriately, SQL monitoring is enabled, but not for all operations. The instance will only monitor the expensive statements. This means that whenever you issue a SQL statement that runs in parallel, or when any SQL statement takes more than five seconds of CPU or I/O time, SQL monitoring automatically kicks in.

SQL Monitoring is a feature of the Oracle Tuning Pack.
A Database Operation Monitoring Example

Let's look at a simple example to see how you identify, begin, and end a database operation when monitoring it with this new feature. First, you must identify the database operation with an operation name and execution ID, which together uniquely identify a specific execution of a database operation.

An operation name denotes a piece of code; this name will be retained when the code is executed concurrently by multiple sessions or even by multiple users. What distinguishes the different operations is the execution ID, which is unique for each invocation of the code.

It’s important to understand that although the execution ID is unique for each invocation of the code, it may not be so between operations. If a process spawns several sessions associated with the same DB operation, the instance will generate a single execution ID for the operation and each session would set the same execution ID by calling the BEGIN_OPERATION procedure.

Note that it’s also possible for different executions of a database operation with the same name to involve different SQL or PL/SQL code. This can occur due to the conditions in the PL/SQL code or if the code is executed by different users.

Once you identify the database operation with an operation name and an execution ID, you perform the following steps to set up and complete the database operation:

- Start the database operation by executing DBMS_SQL_MONITOR.BEGIN_OPERATION.
- End the database operation by executing DBMS_SQL_MONITOR.END_OPERATION.

Because different code paths can be followed through conditions in PL/SQL code, the actual SQL statements or PL/SQL functions can be different for different executions of the same database operation.

Here's an example that shows how to set the operation name and the execution ID for a database operation to monitor:

```sql
SQL> VAR dbop_eid number
SQL> EXEC :dbop_eid :=dbms_sql_monitor.begin_operation('ORA.HR.select1',forced_tracking =>'Y');
PL/SQL procedure successfully completed.
SQL> exec dbms_sql_monitor.end_operation('ORA.HR.select1', :dbop_eid);
PL/SQL procedure successfully completed.
SQL>
```
Chapter 1: Enterprise Manager and Database Monitoring

Oracle recommends that you follow the format of <component>.<subcomponent>.<operation_name> when naming database operations. You do this in order to avoid collisions with similarly named operations, because all the operation names you assign share the same namespace. For operations that occur within the database, the recommended component name is ORA. In the preceding example, the operation is a refresh of a materialized view. Therefore, the component name is ORA, the subcomponent’s name is MV, and the operation is a refresh, leading us to name the operation ORA.MV.refresh.

Monitoring Load Database Operations

The database operations monitoring feature also lets you monitor large database data load operations. Here’s an example that shows how to track a data load operation:

```
SQL> VAR dbop_eid number
SQL> EXEC :dbop_eid :=dbms_sql_monitor.begin_operation('ORA.HR.load',forced_tracking =>'Y');
SQL> exec dbms_sql_monitor.end_operation('ORA.HR.load', :dbop_eid);
```

The operation name in this example is "load," indicating that it tracks a data load operation. Following this, you can monitor the load database operation, including multiple bulk load statements.

Monitoring the Progress of a Database Operation

You can use either the Oracle Enterprise Manager or query V$ dynamic views to track both currently executing and recently completed database operations.

Using Oracle Enterprise Manager

You can use either OEM Cloud Control 12c or the new EM Database Express to view active and recently finished database operations:

- If you’re using OEM Cloud Control, click the Performance menu and then the SQL Monitoring option to view the progress of database operations.
- If you’re using EM Database Express, you could view the list of monitored SQL and DB operations under the Monitored SQL menu, shown in the Database Home page. You can view the monitored SQL and database operations when you click Monitored SQL, as shown in Figure 1-2. You can also view monitored SQL and database operations in the Performance menu on the Home page by clicking Performance Hub. Figure 1-3 shows this page.
Implement Real-Time Database Operation Monitoring (Exam Objective 1.02)

The instance uses the `DBMS_SQL_MONITOR.REPORT_SQL_MONITOR` and `DBMS_SQL_MONITOR.REPORT_SQL_MONITOR_LIST` functions to report the execution details of active database operations. You can see a summary at the top of the execution details page and either an activity graph or a metrics graph at the bottom of the page.
Chapter 1: Enterprise Manager and Database Monitoring

Querying Database Views for Database Operation Monitoring

The following database views help you monitor database operations:

- **V$SQL_MONITOR**  Contains global high-level information about the top SQL statements in an operation. Each entry in this view is dedicated to a single SQL statement.

- **V$SQL_MONITOR_SESSTAT**  Shows statistics for all sessions involved in the database operation. This is a view primarily meant for use by the report generator, and the statistics are stored in XML format. Oracle recommends using V$SESSTAT instead of this view.

- **V$SQL_PLAN_MONITOR**  Shows statistics for each step in the execution plan of a monitored SQL statement.

Of course, you can also use familiar views such as V$ACTIVE_SESSION_HISTORY, V$SESSION, V$SESSION_LONGOPS, V$SQL, and V$SQL_PLAN to get additional information about all monitored database operations. You can query the V$SQL_MONITOR view to get the status of currently executing and recently completed database operations. Following is a query that shows how to do this:

```
SQL> select dbop_name, dbop_exec_id, status
    from v$sql_monitor
    where dbop_name is not null;
```

Note that each row in the V$SQL_MONITOR view pertains to an aggregation of execution statistics for a specific SQL statement across multiple sessions and multiple executions. Both the V$SQL_MONITOR and the V$ACTIVE_SESSION_HISTORY views have two new columns added to them in this release, DBOP_NAME and DBOP_EXEC_ID, to help monitor database operations.

As in the previous releases, you can also use the V$SQL, V$SQL_PLAN, and V$SESSION_LONGOPS views to monitor the recently completed and active sessions.

If you want to review completed database operations, you can do so by viewing the reports stored in AWR through the following views:

- **DBA_HIST_REPORTS**
- **DBA_HIST_REPORTS_DETAILS**
- **DBA_HIST_ACTIVE_SESS_HISTORY**
Perform Emergency Monitoring and Real-Time ADDM (Exam Objective 1.03)

Here’s an example that shows how to query the DBA_HIST_ACTIVE_SESS_HISTORY view:

```
SQL> select session_id, DBOP_NAME, DBOP_EXEC_ID
       from DBA_HIST_ACTIVE_SESS_HISTORY
       where DBOP_NAME is not null;
```

Using Functions to Report on Database Operations

The DBMS_SQL_MONITOR package also contains two functions—the REPORT_SQL_MONITOR_LIST_XML and REPORT_SQL_MONITOR_XML functions—to report on database operations in an XML format. In addition, you can use the REPORT_SQL_MONITOR_LIST and REPORT_SQL_MONITOR functions to report on database operations in a CLOB. By default, the REPORT_SQL_MONITOR function generates a text report for the last execution that was monitored. You can specify several parameters to specify the execution, the report’s level of details, and the report type.

Following is an example showing how to report using the DBMS_SQL_MONITOR.REPORT_SQL_MONITOR_LIST_XML function:

```
SQL> select DBMS_SQL_MONITOR.REPORT_SQL_MONITOR_LIST_XML() from dual;
```

CERTIFICATION OBJECTIVE

Perform Emergency Monitoring and Real-Time ADDM (Exam Objective 1.03)

Oracle Database 12c introduces two new features—Emergency Monitoring and Real-Time ADDM—to help you troubleshoot a database undergoing a severe performance deterioration, such as a database hang. These diagnostic techniques take you beyond the monitoring capabilities in earlier releases, and allow troubleshooting even when you can’t connect to a database in the normal manner.

Emergency Monitoring

Professional DBAs knows what a production performance emergency is—you’re getting complaints from users that the database is really slow or that their jobs are “hanging.” You can, of course, run an ADDM or an AWR report to diagnose the
Chapter 1: Enterprise Manager and Database Monitoring

situation, but it may not even be possible to run the report—in fact, sometimes you can’t log in to the database when you find yourself in a “database hang” situation. Even if were to log in, it’s not really advisable to run resource-intensive diagnostic reports as you may very well end up making matters worse.

Shutting down the instance is one option to get out of the predicament, but as we all know, this option is not ideal and you’d only use it as a last resort. In Oracle Database 12c, you have an option to monitor the database in these extreme circumstances by connecting to the instance and performing a lightweight analysis of the current instance performance, even when things hit rock bottom performance-wise.

In a production system, a blocking session can queue up a large number of sessions behind it once in a while and bring the whole instance to a grinding halt. It’s not immediately apparent which of the sessions is the main blocker, and you end up either killing a very large number of sessions or restarting the instance.

In Oracle Database 12c, you can run Emergency Monitoring before you decide to restart a slow instance plagued by extremely poor performance. By switching to Emergency Monitoring, you connect to the database in a diagnostic mode, even when you can’t connect to the instance in the normal manner. The whole idea behind this capability is that a DBA ought to be able to make a lightweight connection to quickly review performance, without requiring too many resources such as enqueues and latches.

Currently (11g database) you must enable the “memory access mode” explicitly to enable a lightweight connection to the database for troubleshooting. When you enable the memory access mode, a collector process reads performance data from the SGA. In Oracle Database 12c, you don’t need to explicitly switch to the memory access mode and there’s also no collector process.

When you launch Emergency Monitoring, the agent connects directly to the SGA to get performance data, bypassing the SQL layer. The performance page will show data such as ASH and the Hang Analysis table, refreshed in real time. Once you resume normal performance monitoring, the agent stops collecting the performance data from the SGA.

Unlike in Oracle Database 11g, you don’t have to enable and disable the memory access mode for the data collection agents to directly connect to the SGA.

Emergency Monitoring relies on the historical data stored in the ASH buffers to populate the performance pages. ASH buffers are designed to save performance data for 60 minutes, but this isn’t guaranteed and depends also on the level of instance
activity and resource contention. You can do the following by turning on Emergency Monitoring:

- View the Emergency performance page refreshed with real-time performance data.
- View the ASH data and Hang Analysis table that shows blocking and blocking sessions. This page is also current because it’s refreshed in real time.

The Hang Analysis table helps you pinpoint blocking sessions that are causing the database to hang. You can kill the root blocker or, in the worst situation, restart your database from here.

Emergency Monitoring will most likely point out the main blocking session, which you can then kill. If no blocking session is identified, of course, you may have to resort to restarting the instance.

**Real-Time ADDM**

As explained in the previous section, Emergency Monitoring may help resolve a hung database by identifying the root blocking session, which you can kill. However, Emergency Monitoring isn’t always successful; plus, it doesn’t help you identify the root cause of the performance problem. In severe situations, because either a long-running job has consumed all resources or a blocking session is stopping other sessions from continuing work, the database is frozen in place. In these situations, before you resort to shutting down the instance, you can use the new Real-Time ADDM feature to resolve the production issue.

Real-Time ADDM analysis is available through Oracle Enterprise Manager 12c. Unlike Emergency Monitoring, which is designed for you to log in to the database during a database hang and resolve the issue, Real-Time ADDM offers the possibility of performing a root-cause analysis. The feature is designed to help you resolve critical issues such as database hangs without having to restart the database.

_You can use Real-Time ADDM to get to the root cause of a performance problem, which Emergency Monitoring doesn’t help you with._

Real-Time ADDM is similar to normal ADDM, with the difference being that Real-Time ADDM analyses only recent data, specifically that pertaining to the past 10 minutes, to help you diagnose and resolve a system experiencing severe resource contention. As with normal ADDM, Real-Time ADDM provides you with actionable recommendations to resolve the problem.
 Unlike regular ADDM, which accesses unpurged AWR snapshots, Real-Time ADDM accesses just the ASH recent activity from the SGA. Thus, the focus of Real-Time ADDM is helping with currently occurring performance problems, and not the analysis of less recent or historical performance, which is what you use normal ADDM for.

When a database is frozen due to severe resource contention, you can still analyze performance because Real-Time ADDM can collect current performance data directly from the SGA. When you encounter situations such as users unable to connect because the database has run out of space to write to the default audit-tracing location, Real-Time ADDM can help you. When you have a long-running or runaway query that hogs all the resources, Real-Time ADDM can identify the query for you.

You can invoke Real-Time Monitoring in a RAC environment, where ADDM will collect recent performance data from the SGA of each instance in the cluster.

In order to use Real-Time ADDM, you can connect in a normal mode if you can, and if you can’t, use the diagnostic mode to connect.

You can invoke Real-Time ADDM for a RAC database.

Real-Time Connection Modes

If it’s possible to connect to the database, Real-Time ADDM performs a normal database connection to perform routine performance analysis. If it can’t make a normal database connection, such as when the database hangs, Real-Time ADDM performs a latch-less diagnostic connection.

Real-Time ADDM Triggers

In order to proactively detect performance issues, Real-Time ADDM uses the background process MMON, which runs every three seconds using in-memory data, so it doesn't have to use any latches or locks. If MMON finds any of the following triggering issues, the MMON slave process creates a real-time analysis and stores it in the AWR:

- **High load** Average active sessions greater than three times the number of CPU cores
- **I/O** Active sessions impacted by I/O based on the single block read performance
Perform Emergency Monitoring and Real-Time ADDM (Exam Objective 1.03)

- **CPU bound**  Active sessions greater than 10 percent of total load and CPU usage greater than 50 percent
- **Hung sessions**  Hung sessions are more than 10 percent of the total number of sessions

You can view the reports with the help of the `DBA_HIST_REPORTS` and `DBA_HIST_REPORTS_DETAILS` views.

To ensure that the automatic performance triggers don’t cause a performance problem of their own, new reports won’t be generated if a Real-Time ADDM report was generated in the past five minutes by an automatic trigger. To avoid multiple triggers all pointing to performance problems with the same level of severity, a new automatic trigger is required to have an impact of 100 percent or higher when compared to the immediately preceding report within the past 45 minutes. This applies to reports triggered by the same triggering issue.

### Producing a Real-Time ADDM Report

In addition to using the Real-Time ADDM functionality through Enterprise Manager Cloud Control 12c, you can use the new function `REAL_TIME_ADDM_REPORT` from the `DBMS_ADDM` package to obtain a Real-Time ADDM report, as shown here:

```
SQL> select dbms_addm.real_time_addm_report() from dual;
```

This query, which uses the `DBMS_ADDM` function `REAL_TIME_ADDM_REPORT`, will get you a real-time ADDM report for the last five minutes. Alternatively, you can get a Real-Time ADDM report from EM Database Express by doing the following:

- Click the Performance tab on the home page of EM Database Express.
- Click Performance Hub.
- Click Current ADDM Findings.

Figure 1-4 shows a typical current ADDM finding, which looks pretty similar to a regular ADDM report’s findings in its format. The Hang Data tab shows the Hang Analysis page, which lists Final Blockers and Blocked Sessions, including the
SID/SERIAL# for the blocker session so you can kill it. The Top Activity page shows the top SQLs during the past 10 minutes. As mentioned earlier, unlike regular ADDM, Real-Time ADDM focuses on current and very recent periods.

**Combining Emergency Monitoring and Real-Time ADDM**

You may be wondering when and how to use the two new monitoring features—Emergency Monitoring and Real-Time ADDM. If you’re experiencing a slowdown, first use Emergency Monitoring to try and quickly identify the blocking session or something else that’s slowing the instance. If you don’t find anything through this means, you may want to restart the instance. If you really want to avoid shutting down the instance, employ Real-Time ADDM to see if you can get at the root cause and also obtain recommendations for fixing the problem. If you still aren’t successful, of course, your only alternative is to restart the instance.

Emergency Monitoring helps you view both ASH data and hang data and try to identify blocking sessions that are leading to a hanging database. Real-Time ADDM analyzes a lot more—it uses hang analysis data as well as key metrics such as I/O. It lets you view both hang data as well as what’s called the Top Activity Snapshot in Oracle Cloud Control 12c. Finally, Real-Time ADDM is designed to offer you findings and recommendations, similar to what normal ADDM does.
CERTIFICATION OBJECTIVE

Generate a Compare Period ADDM Report (Exam Objective 1.04)

In Oracle Database 11g, when you wish to analyze a case of performance degradation, the AWR Compare Periods Report comes in very handy. Using this capability, it’s very easy to compare performance between a “bad” time period and a “good” baseline time period. Instead of producing two separate AWR reports for the two periods separately and performing a tedious and error-prone manual comparison of the AWR statistics and different time periods, the AWR Compare Periods Report provides a single report that highlights the differences in performance between the two periods.

When you create an AWR Compare Periods Report, you select your own set of snapshots from among the set of preserved snapshot sets, and the report runs a comparison between the two sets of snapshots. You can get either a text or HTML report that compares the two periods and shows differences in critical areas such as wait events, response times of top queries, OS statistics, and instance activity.

In addition to letting you compare performance between two different time periods, the AWR Compare Periods Reports capability also lets you compare performance between DB Replay capture and replay—and even between two different DB replays.

Regardless of the nature of the comparison—two different time periods or two different DB replays—you still need to analyze the huge volumes of performance metrics summarized in the report and isolate the root causes of the performance differentials between them. The reports don’t map the root causes to performance changes.

How Compare Period ADDM Helps

Oracle Database 12c offers you something that takes you further in your search for the reason for a performance change—the Compare Period ADDM Report. This report performs a cause-to-effect analysis, making it simpler to understand why performance deviated from a base time.

The Compare Period ADDM follows this methodology:

1. The analysis first determines the potential causes, such as system changes that may have resulted in the performance deviance between two different time periods.
2. It runs an ADDM analysis for the two time periods to measure the performance deviance between those periods.
3. Using rule sets, it maps the system changes to the performance deviations.
Unlike the AWR Compare Periods Report, which tells you what exactly is the difference in performance between two periods, the Compare Period ADDM Report tells you what you can attribute the changes to.

**Generating a Compare Period ADDM Report**

You generate a Compare Period ADDM Report by defining a comparison period, which you compare with a prior base period. The comparison period could be one that represents the database following key changes, such as optimizer changes or memory-, storage-, and CPU-related changes. It can even represent a period that involves performance following the addition of new RAC nodes.

You can certainly use DB Replay to produce a Compare Period ADDM Report, with the capture period representing the base period and the replay period representing the comparison period.

**SQL Commonality**

A key factor that determines the level of compatibility, and hence the reliability of the findings when you compare the base and the comparison periods, is the level of workload compatibility between the two periods. One of the big problems with comparing two different time periods in a database, as we all know, is that it’s not often the case that the two periods are highly compatible. The degree to which the SQL statements, the workloads, and the applications are similar between the two periods you’re comparing is called SQL commonality. The higher the SQL compatibility, the more reliable and useful your findings will be.

Because you’re testing identical workloads when you test with Real Application Testing with a capture/replay of the same workload, you’re going to get 100-percent SQL compatibility when you perform your tests using Oracle’s Real Application Testing feature. However, rest easy, because Oracle assures you that if there is a high degree of compatibility (such as an 80 or 90 percent level), it’s good enough to derive valid and reliable findings from the Compare Periods ADDM Report.

**Using Cloud Control 12c to Generate Reports**

As with most database management tasks, the easiest way to generate a Compare Period ADDM Report is through Oracle Cloud Control 12c. To do this, click the Performance tab in the home page for the database and go to Run Compare Period ADDM. Here, you must perform the following steps:

1. Select a comparison period by selecting Begin Time and End Time values in the boxes. This is the period where performance is poor.
2. Select a base period, which represents a period with the same workload but good performance. Here you have three options, which really translate to how the database will select the AWR snapshots to use for the comparison:

- Option 1 lets you select an offset of one snapshot. That is, your snapshot for a recent period will be the one just preceding the snapshot representing the base period. Or, you can choose an offset of one day, using the previous day for comparison (same times). You may also select a base period from one week earlier.

- Option 2 lets you select a baseline, such as the system_moving_window baseline or a custom baseline representing acceptable performance.

- Option 3 lets you select a customized time period without regard to snapshots or baselines.

3. Run the comparison report for the two periods.

The reports produced by EM Cloud Control 12c are highly useful, especially when compared with the AWR Compare Periods reports, because they show the differences between the two periods at a glance. Each report has three major sections, each accessible with a tab: Configuration, Findings, and Resource.

The Configuration page of the report shows, at the very top left, the level of SQL Commonality between the periods being compared. Oracle suggests that the SQL Commonality be at least 80 percent.

Oracle recommends that you don't rely on comparisons where the SQL Commonality is less than 80 percent. Of course, you need to use your own judgment here to see if even reports with a lesser level of compatibility can be used for gaining some knowledge about the performance issue at hand.

The Findings page of the report shows the factors that caused the performance degradation (or enhancement). It'll also show the impact of those factors, such as a change in SGA, in terms of a percentage. You can customize the report by setting the percentage of the impact for something to be presented in a report.

The Resource page compares the performance of key resources such as CPU, memory, I/O, and the interconnect (in a RAC system).
Using DBMS_ADDM to Generate Reports

You can use the DBMS_ADDM package's COMPARE_INSTANCES function to compare two periods within the same instance. Here's an example:

```sql
SQL> select dbms_addm.compare_instances (
    2    base_dbid => 1351551762,
    3    base_instance_id => 1,
    4    base_begin_snap_id => 121,
    5    base_end_snap_id => 122,
    6    Comp_dbid => 1351551762,
    7    comp_instance_id => 1,
    8    comp_begin_snap_id => 198,
    9    comp_end_snap_id => 199,
   10    report_type => 'XML')
```

The COMPARE_INSTANCES function produces a Compare Period ADDM Report that compares the performance of a single instance over two time periods or of two different instances over two time periods. In addition to the COMPARE_INSTANCES function, the DBMS_ADDM package offers the following functions as well:

- **COMPARE_DATABASES** Generates a report for a database-wide comparison of performance. You can compare the performance of a single database over two time periods or of two different databases over two time periods.
- **COMPARE_CAPTURE_REPLAY_REPORT** Compares the performance for a Workload Capture with that of a Workload Replay.
- **COMPARE_REPLAY_REPLAY_REPORT** Compares the performance during one Workload Replay to that during another Workload Replay.

**CERTIFICATION OBJECTIVE**

Understand ASH, ADR, and Network Enhancements (Exam Objective 1.05)

Oracle Database 12c contains some enhancements in the areas of Automatic Session History (ASH), the Automatic Diagnostic Repository (ADR), as well as some key network-related enhancements.
Diagnosing Performance Issues Using Active Session History (ASH) Analytics

The ASH enhancements in Oracle Database 12c pertain to how Enterprise Manager (Cloud Control) displays ASH information. In Enterprise Manager 11g, the Top Activity page displays most of the ASH data. There are several limitations to this page, including limited drilldown capabilities and the inability to send reports offline to other users. In the new release, a brand-new page called the ASH Analytics page has better capabilities, such as filtering dimensions and drilling down into a load map view to show wait events.

ADR Enhancements

Automatic Diagnostic Repository, introduced in Oracle Database 11.1, provides a unified directory structure and consistent diagnostic formats for all Oracle products. The ADR, which is a file-based repository, stores valuable diagnostic data such as the alert logs, trace files, and dump files, as well as incident dumps, packages, and Health Monitor reports.

In Oracle Database 11g, a new log directory under ADR contains two important subdirectories—ddl and debug.

The DDL Log

In Oracle Database 11g, you can set up DDL logging by setting the initialization parameter enable_ddl_logging to true. When you do this, any DDL changes made will be logged in the alert log file. In Oracle Database 12c, when you enable DDL logging, the instance records the DDL logging information in an exclusive DDL logging file that has the same format as your regular alert log.

If you wish to enable DDL logging for your database by setting the parameter enable_ddl_logging to true, you must first license the Oracle Change Management Pack.

If you set the enable_ddl_logging parameter to true, the instance will record the DDL changes in the log.xml file. If you don’t set this parameter to true, Oracle won’t record these changes to the log.xml or the alert log.

The DDL log file is named log.xml and is created in the $ADR_HOME/log/ddl directory; it contains DDL statements that are extracted from the alert log file:

```
SQL> alter system set enable_ddl_logging=true;
System altered.
SQL> conn hr/hr
```
Chapter 1: Enterprise Manager and Database Monitoring

Connected.
SQL> create table test1(name varchar2(20), id number);
Table created.
SQL> alter table test1 add (address varchar2(40));
Table altered.
SQL> drop table test1;
Table dropped.
SQL>
You must license the Oracle Change Management Pack if you decide to enable DDL logging in your databases. When you set `enable_ddl_logging` to `true`, the parameter is licensed as part of the Change Management Pack.

Note that if you’ve never enabled the DML logging parameter before, there’s no prior log.xml file, but the instance will create one to record your DDL changes. If the log.xml file exists, the instance will append the new DDL change information to that file. Once you perform these or any other set of DDL operations, you can view the contents of the log.xml file directly by using an operating system command such as `more`, for example, as shown here:

```bash
[oracle@localhost ddl]$ pwd
/u01/app/oracle/diag/rdbms/orcl/orcl2/log/ddl
[oracle@localhost ddl]$ ls
log.xml
[oracle@localhost ddl]$ more log.xml
<msg time='2013-08-03T13:38:17.176-04:00' org_id='oracle' comp_id='rdbms'
  msg_id='opiexe:4181:2946163730' type='UNKNOWN' group='diag_adl'
  level='16' host_id='localhost.localdomain' host_addr '::1'
  version='1'>
  <txt>create table test1(name varchar2(20), id number)
</txt>
</msg>
<msg time='2013-08-03T13:38:31.660-04:00' org_id='oracle' comp_id='rdbms'
  msg_id='opiexe:4181:2946163730' type='UNKNOWN' group='diag_adl'
  level='16' host_id='localhost.localdomain' host_addr '::1'
  version='1'>
  <txt>alter table test1 add (address varchar2(40))
</txt>
</msg>
<msg time='2013-08-03T13:38:41.458-04:00' org_id='oracle' comp_id='rdbms'
  msg_id='opiexe:4181:2946163730' type='UNKNOWN' group='diag_adl'
  level='16' host_id='localhost.localdomain' host_addr '::1'
  version='1'>
  <txt>drop table test1
</txt>
</msg>
<msg time='2013-08-03T14:00:40.521-04:00' org_id='oracle' comp_id='rdbms'

Understand ASH, ADR, and Network Enhancements (Exam Objective 1.05)

If you don’t enable DDL logging, not only will you not see a DDL log file, but the database won’t record DDL statements in the alert log either.

However, a great way to get a clear and readable DDL log file is to invoke ADRCI and issue the `show log` command, as shown here:

```bash
[oracle@localhostddl]$ adrci
ADRCI: Release 12.1.0.1.0 - Production on Sat Aug 3 13:41:38 2013
Copyright (c) 1982, 2013, Oracle and/or its affiliates. All rights reserved.

ADR base = "/u01/app/oracle"
adrci> show log
ADR Home = /u01/app/oracle/diag/rdbms/orcl/orcl2:
*************************************************************************
Output the results to file: /tmp/utsout_6350_139845_1.ado
adrci>
```

When you issue this command, the instance will open the log.xml file in an editor (such as vi in Unix/Linux and Notepad in Windows) and show the contents, which are in the following format:

```
2013-08-03 13:38:17.176000 -04:00
create table test1(name varchar2(20), id number)
2013-08-03 13:38:31.660000 -04:00
alter table test1 add (address varchar2(40))
2013-08-03 13:38:41.458000 -04:00
drop table test1
```

The Debug Log

The debug log is a file stored in the `$ADR_HOME/log/debug` directory and contains entries that describe unusual events. These are events that aren’t serious enough to warrant a full-fledged incident package or a mention in the alert log, but are deemed to be usual enough to be saved for potential use later on. Oracle recommends that DBAs not use these debug files because their main purpose is to serve as diagnostic aids to Oracle Support personnel.
Chapter 1: Enterprise Manager and Database Monitoring

The debug log has the same format as your regular database alert log and contains useful information pertaining to potential issues, and is designed to reduce the diagnostic data written to the alert log and trace files.

Network-Related Enhancements

The enhancements in the network area mainly consist of new parameters to allow you to configure network compression, which is available as part of Oracle’s Advanced Compression Option. In addition, there are changes in the default values for the parameter DEFAULT_SDU_SIZE.

Network Compression

In a system constrained by network bandwidth, you can increase performance by compressing data flowing through the network, which increases the network’s bandwidth by reducing the amount of bits the network needs to transmit through its pipes.

*Network compression is available as part of the Oracle Advanced Compression Option.*

Setting Up Network Compression

You control network compression by setting the following three parameters in the sqlnet.ora file for an instance:

- **SQLNET.COMPRESSION** Set this parameter to ON to turn on network data compression. By default, network compression isn’t enabled. Setting this parameter to OFF will turn off network compression.

- **SQLNET.COMPRESSION_LEVELS** Determines which levels are used at both ends of the network. A level of LOW tells the network to use low CPU by using a low compression ratio. A HIGH setting translates to a directive to use high CPU by using a high compression ratio. The default setting for this parameter is LOW.

- **SQLNET.COMPRESSION_THRESHOLD** This parameter determines the size of the minimum data size for compression to kick in. The default is 1,024 bytes.
SDU Size
The session data unit (SDU) size refers to the size of data packets that are transmitted over a network. Oracle Net compresses data into buffers the size of the SDU before transmitting them. Larger SDU sizes can potentially improve network performance. You should set the SDU size so it's about 70 bytes larger than the common message size.

Before adjusting the size of the SDU, you must tune the application to require less network usage. If your network is very fast, you don’t need to change the SDU size. Same case if your applications are conveying only small packets of data across the network.

You can set the DEFAULT_SDU_SIZE parameter in the sqlnet.ora file to control the SDU size. Optimal size for this parameter depends on the size of the messages and network characteristics. In Oracle Database 11g, the default value for this parameter is 8KB. In Oracle Database 12c, the default SDU size for a client and a dedicated server is still 8KB, but the maximum has been raised from 64KB to 2MB. Higher SDU sizes require more memory, and you should consider changing the default SDU size only when most of the messages are sized larger (or smaller) than 8,192 bytes.

CERTIFICATION SUMMARY
This chapter explained the new database-monitoring tool Enterprise Manager Database Express, which replaces OEM Database Control in this release. You learned about how to perform real-time SQL monitoring. You learned how to monitor your database during an emergency, such as a severe database hang. The chapter explained Real-Time ADDM and Compare Period ADDM, which help you identify the root cause of a performance problem. You learned about the new diagnostic logs added to ADR and the changes made to improve network performance.
TWO-MINUTE DRILL

Use Oracle Enterprise Manager Express, OUI, and DBCA (Objective 1.01)

- You can use EM Database Express to monitor a single instance or all instances that belong to a RAC database.
- EM Database Express has a very small footprint—it’s a lightweight management solution.
- EM Database Express uses a web-based console.
- In order to configure the port for a database instance, you must use the DBMS_XDB_CONFIG package’s setHTTPsPort or setHTTPPort procedure.
- Use the DBMS_XDB_CONFIG.getHTTPsPort (and getHTTPPORT) procedures to find the port used for a database instance.
- EM Database Express is available only when the database is open.
- You can’t start the database with EM Database Express, change its state, or set the archive log mode with EM Database Express.

Implement Real-Time Database Operation Monitoring (Objective 1.02)

- Real-Time Database Operation Monitoring extends Real-Time SQL Monitoring of Oracle Database 11g.
- You can monitor batch jobs, diagnose background processes, as well as diagnose changes to jobs over time.
- A database operation is an operation the database server runs to perform tasks.
- A database operation can be a simple operation that consists of a single SQL or PL/SQL statement, or a composite operation that consists of the activity of one or more sessions between two points in time.
- A simple database operation consists of a single SQL statement or PL/SQL function or procedure.
- A composite database operation consists of the activity of one or more sessions.
- A composite database operation can consist of a single-session at any given time or multiple concurrent sessions.
You can name a database by using bracketing, which involves a set of begin and end points using the `DBMS_SQL_MONITOR` package.

The second way to name a database operation is by setting a tag—this is for Java and OCI applications.

Applications using tagging don’t have an explicit method for ending an operation.

You can enable database monitoring at the statement, database, or system level.

**Perform Emergency Monitoring and Real-Time ADDM (Objective 1.03)**

- The purpose of Emergency Monitoring is to enable instance troubleshooting and avoid an instance shutdown if you can.
- Emergency Monitoring is designed for situations where you can’t log in because of a database hang or you can’t analyze the database (with AWR snapshots).
- Emergency Monitoring provides a lightweight mechanism to connect and check for blockers and the reasons for a database hang.
- The agent connects directly to the SGA and accesses the SGA directly, bypassing the SQL Retrieval Layer.
- The analysis will show you ASH data and the Hang Analysis table of top blocking and blocked sessions and deadlocks.
- The analysis populates the Emergency Performance page with historical data collected from ASH buffers.
- Real-Time ADDM works in a similar fashion as regular ADDM in analyzing instance performance.
- Real-Time ADDM differs from regular ADDM in that it uses recent ASH activity, and not stored AWR snapshots.
- Real-Time ADDM provides the root causes of performance deviations from the norm, which Emergency Monitoring does not.
- You can use real-time monitoring for both single-instance and Oracle RAC databases.
- You can connect in the normal mode; if you can’t do this, you can connect in the diagnostic mode.
- Real-Time ADDM collects data from the SGA.
- Real-Time ADDM provides you recommendations and possible actions.
Generate a Compare Period ADDM Report (Objective 1.04)

- Compare Period ADDM helps you when you know what has changed (database upgrade, for example) but you don’t know exactly why the performance changed.
- Unlike the AWR Compare Periods Report, the Compared Period ADDM Report provides root-cause analysis.
- The analysis maps the system changes that caused the performance changes to the effects, using a rule set.
- You compare a current “comparison” period to a past “base” period.
- Workload compatibility measures the similarity of the two workloads, by considering the SQL statements and workload.
- The similarly of the SQL statements running in the base and reference periods is called SQL Commonality.
- Oracle suggests you can rely on the findings of the Compare Period ADDM Report if the SQL Commonality is 80 to 90 percent.
- You have three options to define the base period—one snapshot offset, a system moving window, and a customized period.
- Using various procedures from the DBMS_ADDM package, you can perform an instance-level or a database-level performance comparison.
- You can also compare a Workload Capture to a Workload Replay, or one Workload Replay to another Workload Replay.

Understand ASH, ADR, and Network Enhancements (Objective 1.05)

- You can set the enable_ddl_logging parameter to true to log DDL changes to the new log.xml file under the $ADR_HOME/log/ddl directory.
- The new debug log file records unusual events that occur. It’s created in the $ADR_HOME/log/debug directory.
- When you set the enable_ddl_logging parameter to true, the parameter enable_ddl_logging is licensed as part of the Change Management Pack.
- Use the SHOW LOG command to view the DDL log file’s contents.
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.

Use Oracle Enterprise Manager Express, OUI, and DBCA (Objective 1.01)

1. You can use Enterprise Manager Database Express to manage:
   A. All instances of a RAC database
   B. A single-instance database
   C. An entire Oracle environment
   D. A single instance that belongs to a RAC database

2. If you have multiple instances on a server, you need:
   A. To configure a single port to access all the instances
   B. To configure a single HTTP port but multiple HTTPS ports
   C. To configure a single HTTPS port but multiple HTTP ports
   D. To configure a different port for each database instance

3. You can do the following with EM Database Express:
   A. Shut down the database
   B. Start up the database
   C. Monitor users and security
   D. Configure archive log mode for the database

Implement Real-Time Database Operation Monitoring (Objective 1.02)

4. Real-Time Database Operation Monitoring can do the following:
   A. Monitor the progress of an operation
   B. Package raw data for offline analysis
   C. Create active reports that require access to the production system
   D. Create active reports that don’t require access to production systems

5. Monitoring database operations is allowed for sets of:
   A. SELECT and DML operations
   B. SELECT statements only
   C. DML statements only
   D. DDL operations only
Chapter 1: Enterprise Manager and Database Monitoring

6. You can define a set of SQL and PL/SQL statements between two points that include the following:
   A. Single sessions with just a single session for the entire duration of the database operation
   B. Multiple concurrent sessions
   C. Multiple sessions with no more than one session running at any given time
   D. Single sessions only

7. You can identify a database operation with:
   A. Just the execution ID
   B. Just the operation name
   C. A system-generated operation number
   D. Both the operation name and execution ID

8. You can monitor database operations with:
   A. EM Database Express
   B. EM Cloud Control
   C. V$SQL_MONITOR
   D. DB Control

Perform Emergency Monitoring and Real-Time ADDM (Objective 1.03)

9. Switching to Emergency Monitoring lets you:
   A. Connect to the instance in diagnostic mode
   B. Connect to the instance in normal mode
   C. View the Hang Analysis table but not ASH data
   D. View ASH data and the Hang Analysis table

10. When using Emergency Monitoring, you:
    A. Must first enable the memory access mode
    B. Don’t have to enable the memory access mode
    C. Must disable the memory access mode
    D. Must enable the Emergency Monitoring mode

11. Emergency Monitoring:
    A. Can be used along with Real-Time Monitoring
    B. Provides a root-cause analysis
    C. Can’t be used along with Real-Time Monitoring
    D. Needs special licensing of Oracle Packs
12. The difference between Real-Time ADDM and regular ADDM lies in the fact that:
   A. Real-Time ADDM uses AWR snapshots from the last 10 minutes.
   B. Real-Time ADDM uses recent ASH activity details from the SGA.
   C. Regular ADDM uses unpurged AWR snapshots.
   D. Regular ADDM uses AWR snapshots from the last 10 minutes.

13. In Real-Time Database Monitoring, you can use the `REAL_TIME_ADDM_REPORT` procedure (DBMS_ADDM package) to get:
   A. A CLOB containing a Real-Time ADDM report for the past five minutes
   B. A text-based Real-Time ADDM report for the past five minutes
   C. A CLOB containing a Real-Time ADDM report for the past hour
   D. An XML-based Real-Time ADDM report for the past hour

14. With Real-Time Database Monitoring, you can view:
   A. Findings only, without recommendations
   B. Recommendations only
   C. Just the list of the blocking and blocked sessions leading to a database hang
   D. Findings, recommendations, hang data, and a Top Activity Snapshot

15. In order to perform Real-Time Database Monitoring, you must:
   A. Log in only in diagnostic mode.
   B. Log in only in normal mode.
   C. Log in using normal mode if you can; if you can’t, log in using diagnostic mode.
   D. Log in using the Real-Time Monitoring Mode.

Generate a Compare Period ADDM Report (Objective 1.04)

16. The difference between the Compare Period ADDM Report and an AWR Compare Periods Report is that
   A. The output format is different in the two reports; otherwise, they provide the same information.
   B. The AWR Compare Periods Report performs a cause-to-effect analysis.
   C. The Compare Period ADDM Report performs a cause-to-effect analysis.
   D. Compare Period ADDM Report can’t compare performance beyond one hour in the recent past.
17. Which of the following is (are) true?
   A. SQL Commonality must be close to 100 percent between the two periods for the results to be valid.
   B. SQL Commonality must be at least 80 to 90 percent for the analysis to be reliable.
   C. As long as the SQL statements are similar, it doesn’t matter what the level of SQL Commonality is.
   D. You must always invoke Database Replay to get a Compare Period ADDM Report because you can get a 100-percent SQL Commonality only with Database Replay.

18. The DBMS_ADDM package’s COMPARE_INSTANCES procedure:
   A. Generates a Compare Period ADDM Report for a single instance
   B. Generates a Compare Period ADDM Report for all instances of a database
   C. Can compare performance for a Workload Capture to that of a Workload Replay
   D. Generates a Compare Period ADDM Report that contains a root-cause analysis of a performance problem

19. The three options you have to define a base period when creating a Compare Period ADDM Report are:
   A. System moving window, customized period, and database-generated window
   B. System moving window, offset of two snapshots, customized period
   C. System moving window, offset of one snapshot, database-generated window
   D. System moving window, offset of one snapshot, customized period

Understand ASH, ADR, and Network Enhancements (Objective 1.05)

20. If you set the enable_ddlLogging parameter to true, which of the following is true in Oracle Database 12c?
   A. DDL commands can be logged in the alert log and in the DDL log log.xml file.
   B. DDL commands are logged in the log/ddl directory under ADR, in the log.xml file.
   C. DDL commands aren’t logged in the alert log.
   D. DDL commands are logged in the $ADR_HOME/log/debug directory.

21. The SHOW LOG command in ADRCI shows:
   A. The contents of the DDL log file in text format
   B. The contents of the DDL log file in XML format
   C. The contents of the DEBUG file in text format
   D. The contents of the DEBUG file in XML format
22. The debug log in the ADRCI:
   A. Logs both DDL changes and unusual events
   B. Records serious events that used to be written to the alert log in earlier database releases
   C. Records unusual events that aren’t serious enough to warrant writing to the alert log
   D. Is included in IPS incident packages

23. Compression of data in the network layer:
   A. Increases the bandwidth used
   B. Reduces the bandwidth used
   C. Reduces the number of bits that are transmitted
   D. Increases the number of bits that are transmitted

24. You don’t need to modify the SDU size when:
   A. You have a low-speed network.
   B. Your requests return large amounts of data from the server.
   C. Your requests return small amounts of data from the server.
   D. You can’t tune the application to reduce network usage.

25. You set the `DEFAULT_SDU_SIZE` parameter in:
   A. The init.ora file or the SPFILE
   B. The listener.ora file
   C. The tnsnames.ora file
   D. The sqlnet.ora file
SELF TEST ANSWERS

Use Oracle Enterprise Manager Express, OUI, and DBCA (Objective 1.01)

1. ☑ B and D. EM Database Express can be used to monitor/manage a single-instance database. It can also be used to manage and monitor an instance that is part of an Oracle RAC database. ☒ A and C are incorrect. A is incorrect because the EM Database Express can't manage/monitor all instances of a RAC database. The tool is limited to a single instance. C is incorrect because you can't use EM Database Express to monitor and manage an entire Oracle environment—it's Oracle EM Cloud Control 12c that can help you manage your entire environment.

2. ☑ D. You must configure a separate port for each instance. You can't monitor more than one instance through a single port. ☒ A, B, and C are incorrect. A is incorrect because you need a separate port for each instance. B and C are incorrect because regardless of whether it's an HTTP or an HTTPS port, you need a dedicated port for each instance.

3. ☑ C. EM Database Express allows you to manage Oracle security, including the management of users. ☒ A, B, and D are incorrect because you can't perform any of these actions via EM Database Express.

Implement Real-Time Database Operation Monitoring (Objective 1.02)

4. ☑ A, B, and D. Real-Time Database Monitoring does all of these. ☒ C is incorrect because you don't have to access the production system to create an active report—you can do so by using the packaged data and then analyze it offline.

5. ☑ A. You can monitor both SELECT and DML operations. ☒ B, C, and D are incorrect. B and C are incorrect because you can monitor both of these, not just one of them. D is incorrect because you can't monitor DDL operations.

6. ☑ A, B, and C. You can create single sessions, multiple concurrent sessions, and multiple sessions with just one session running at a time. ☒ D is incorrect because database operations can span multiple concurrent sessions.
7. ☑ D. The operation name and the execution ID together uniquely identify an operation.
   × A, B, and C are incorrect. A and B are incorrect because neither the operation name
   nor the execution ID can uniquely identify an operation by itself. C is incorrect because the
   instance doesn’t generate any operation numbers of its own.

8. ☑ A, B, and C. You can monitor database operations with all of these.
   × D is incorrect because there’s no Enterprise Manager DB Control in Oracle Database 12c.

Perform Emergency Monitoring and Real-Time ADDM (Objective 1.03)

9. ☑ A, B, and D. A and B are correct because you can connect in the normal mode if you are
    able; otherwise, you can do so in the diagnostic mode. D is correct because you can view both
    ASH data and the Hang Analysis table in Emergency Database Monitoring.
   × C is incorrect because you can view both ASH data and the Hang Analysis table.

10. ☑ B. You don’t have to enable the memory access mode. You can automatically do the
    Emergency Monitoring.
   × A, C, and D are incorrect because you don’t have to do any of these.

11. ☑ A and B. A is correct because you can use Emergency Monitoring and Real-Time
    Monitoring together. B is correct because Emergency Monitoring does provide a root-cause
    analysis.
   × C and D are incorrect. C is incorrect because you can use both Emergency Monitoring and
    Real-Time Monitoring together—they aren’t mutually exclusive tools. D is incorrect because you
    don’t need special licensing to use Emergency Monitoring.

12. ☑ B and C. Real-Time ADDM uses recent ASH activity from the SGA. C is correct because
    regular ADDM uses preserved AWR snapshots.
   × A and D are incorrect. A is incorrect because Real-Time ADDM doesn’t use AWR
    snapshots—it uses recent ASH data. D is incorrect because regular ADDM could use any
    preserved historical AWR snapshots, not just those from the last 10 minutes.

13. ☑ A. You can get a CLOB containing an XML-based report covering the past five minutes.
    × B, C, and D are incorrect. B is incorrect because the output is a CLOB containing an
    XML-based report. C and D are incorrect because the report covers just the past five minutes,
    not one hour.

14. ☑ D. You get all of these with Real-Time Database Monitoring.
    × A, B, and C are incorrect because none of these three choices enumerates all the things
    that you get with Real-Time Database Monitoring.
15. ☑ C. You can log in normally as well as in the diagnostic mode.
   ☑ A, B, and D are incorrect. A and B are incorrect because you can log in under either a
diagnostic or a normal mode. D is incorrect because there really isn’t a Real-Time Monitoring
mode.

Generate a Compare Period ADDM Report (Objective 1.04)

   ☑ A, B, and D are incorrect. A is incorrect because the reports provide different types of
information. B is incorrect because the AWR Compare Periods Report has no cause-to-effect
analysis. D is incorrect because the Compare Period ADDM Report can compare historical
periods much older than one hour in the past.

17. ☑ B. Oracle states that an 80-to-90-percent compatibility of the SQL statements is good
enough to drive useful information.
   ☑ A, C, and D are incorrect. A is incorrect because you don’t need 100-percent SQL
Compatibility. C is incorrect became you can’t just say that the SQL statements need to
be similar—they need to very similar, something close to an 80-to-90-percent level of
compatibility. D is incorrect because you don’t have to always invoke Database Replay to obtain
a Compare Period ADDM Report.

18. ☑ A, C, and D. The COMPARE_INSTANCES procedure can do all of these.
   ☑ B is incorrect because you need to invoke the COMPARE_DATABASES procedure to
generate a Compare Period ADDM Report for all instances of a database.

19. ☑ C. You can create a Compare Period Report by defining your base period with one of these
three options: a system moving window, an offset of one snapshot, a database-generated window.
   ☑ A, B, and D are incorrect because they contain invalid/nonexistent options.

Understand ASH, ADR, and Network Enhancements (Objective 1.05)

20. ☑ B and C. B is correct because when you set this parameter, the instance will log all DDL
commands in the new log.xml file. C is correct because once you set this parameter, the instance
logs any DDL commands in the alert log.
   ☑ A and D are incorrect. A is incorrect because the instance won’t write DDL changes to the
alert log any longer after you set this parameter. D is incorrect because the debug directory is for
logging unusual events, not DDL changes.

21. ☑ B. The show log command shows the DDL log file in XML format.
   ☑ A, C, and D are incorrect because they either mention the wrong file (debug) or the wrong
format (text).
22. ✓ C and D. C is correct because the purpose of the debug log is to record events that may indicate problems but aren’t quite serious enough to write to the alert log. This log exists primarily for Oracle Support’s diagnostic purposes. D is correct because the debug log is automatically included in the IPS incident packages.
✗ A and B are incorrect. A is incorrect because the debug log doesn’t record DDL changes. B is incorrect because the debug log doesn’t record serious events.

23. ✓ B and C. Compressing data that passes through the network reduces the bandwidth that’s used because it reduces the number of bits that are transmitted.
✗ A and D are incorrect. A is incorrect because compression reduces bandwidth usage. D is incorrect because compression reduces the amount of bits transmitted.

24. ✓ C. You really don’t need to modify the default SDU size if your data requests return small amounts of data.
✗ A, B, and D are incorrect. A is incorrect because you don’t need to modify the SDU size when you have a fast network, not the other way around. B is incorrect because large data being returned means you should actually look into raising the SDU size. D is incorrect because you may have to modify SDU size if you can’t tune the application.

✗ A, B, and C are incorrect files for setting the DEFAULT_SDU_SIZE parameter.