CHAPTER 1

Installing, Upgrading, and Managing Change

The best way to start reviewing the new features and changes offered by the Oracle Database 11*g* release is by first installing the software. As a DBA, you must also be wondering what it takes to upgrade from your current version of Oracle (8*i*, 9*i*, or 10*g*) to the Oracle Database 11*g* version. Well, this chapter discusses the changes in the Oracle installation procedures as well as the database upgrade process and the revolutionary new Oracle feature called Real Application Testing that helps you anticipate potential problems inherent in both software and application upgrades.

Oracle Database 11*g* introduces several new features related to installing the server software. These new features include several changes in the install options, new components you can install, an enhanced Optimal Flexible Architecture (OFA) to lay out your datafiles and the flash recovery area. Some of the older components such as iSQL*Plus are no longer included in the Oracle 11*g* release, while newer components have been added. We'll review the new installation options, as well as several new initialization parameters, in this chapter.

Once you install the new Oracle 11*g* binaries, your attention will naturally turn to upgrading your current Oracle databases running on older versions of the Oracle server software. There are several changes in both the manual upgrade method and the Database Upgrade Assistant (DBUA). The pre-upgrade information tool has been revised to provide more information.

Change management is one of the top priorities of the Oracle Database 11*g* release. Organizations typically face considerable problems when making changes in their production systems, be it an upgrade to a newer release of the database software or code changes in the applications. Simulated workloads often fail to accurately represent the true production database workloads. Oracle Database 11*g* provides two powerful solutions, Database Replay and SQL Performance Analyzer (as part of a broader feature called Real Application Testing). We devote considerable attention to the Database Replay and SQL Performance Analyzer features in this chapter. Finally, we discuss several interesting new features in database software patching.

This chapter covers the following main topics:

- New features in the server installation
- Oracle Database 11*g* installation
- New features in database creation
- Database upgrade new features

- Real Application Testing
- Database software patching

New Features in Server Installation

The installation process for the Oracle server software is essentially the same in the Oracle Database 11*g* and 10g versions. Invoking the Oracle Universal Installer (invoked by `runInstaller` on Unix/Linux and `setup` on Windows) remains the same, and the Oracle Universal Installer performs the same operating system checks as in the older versions. There are, however, a few important changes when installing Oracle Database 11*g*, which we summarize in the following sections.

Changes in the Optimal Flexible Architecture

The Oracle 11*g* installation process contains changes in the way you specify the Oracle base, the Oracle home, and the flash recovery area. In addition, there is a new infrastructure called the *automatic diagnostic repository*, which serves as a consolidated location for all database diagnostic information.

Choosing the Oracle Base Location

The Oracle base directory is the top-level directory for installing Oracle software, and the OFA-recommended path for this directory is `/mount_point/app/<oracle software owner>`. For example, a typical Oracle base directory path is `/u01/app/oracle`, where `oracle` is the Oracle software owner. The Oracle base is recommended as an environment variable, as in the earlier Oracle versions, but in future versions Oracle is likely to make this a mandatory variable. The Oracle Universal Installer now provides a list box for you to edit or select the Oracle base. The Oracle Universal Installer automatically derives the default Oracle home location from the Oracle base location you provide. The Oracle home directory is a subdirectory of the Oracle base directory, and that's where you install all your Oracle software. You can edit the location offered by the Oracle Universal Installer if you want to specify a different directory as the Oracle home location. Oracle recommends you specify the same Oracle base for multiple Oracle homes created by a user.

Choosing the Datafile and Flash Recovery Area Locations

In Oracle Database 11*g*, by default, all datafiles are located one level below the Oracle base. The flash recovery area is also one level below the Oracle base, and Oracle recommends you create this on a disk that is separate from the ones hosting the datafiles. In Oracle Database 10*g*, by contrast, both the flash recovery area and the datafiles are located in the Oracle home directory. The datafile location and the flash recovery area in an Oracle Database 11*g* release database then would look like the following, assuming you chose `/u01/app/oracle` as your Oracle base location:

```
/u01/app/oracle/oradata
/u01/app/oracle/flash_recovery_area
```

The Oracle Universal Installer will warn you if you don't put the datafiles and the flash recovery area in separate locations.

Automatic Diagnostic Repository

The automatic diagnostic repository (ADR) is a new Oracle Database 11*g* feature, meant for the consolidation of all diagnostic data, including various trace files. The goal of the ADR is to provide a single directory location for all error data you'll need for diagnosing and resolving problems, thus leading to faster error resolution and troubleshooting. The ADR is simply a directory location that you specify through the new initialization parameter `diagnostic_dest`. The ADR replaces the traditional use of the diagnostic directories such as `bdump`, `cdump`, and `udump`, where you had to go to manually seek out the necessary trace file and error files during troubleshooting. The ADR uses standard methods to store diagnostic data not only for the Oracle database but also for other Oracle products. The diagnostic data is then read by special automatic diagnostic tools to provide a quick turnaround time for troubleshooting problems related to various Oracle products.

Under the ADR, you have the different directories such as `cdump`, `alert`, and so on. The alert log that you're used to viewing in the vi editor on Unix is now an XML-based file. You can read this file using the new `adrci` command-line tool. We discuss the ADR in detail in Chapter 2.

If you choose to use the ADR, you must give the Oracle Universal Installer a directory location for the ADR base. To consolidate diagnostic data, Oracle recommends you choose the same ADR base for all Oracle products.

■**Note** If `ORACLE_BASE` is not set, warnings will appear in the alert log. Although `ORACLE_BASE` is a recommended environment variable, this variable will become a requirement in future releases.

By default, the ADR's base directory for storing diagnostic data is set to the Oracle base location. However, you can set an alternate location for the ADR by setting a value for the new initialization parameter `diagnostic_dest`. The ADR directory has the name `$ORACLE_BASE/diag` and contains several subdirectories, the most important of which is the `rdbms` directory. In the `rdbms` directory, diagnostic files are organized by database name and instance name. For example, for a database with the database name `orcl` and an instance name of `orcl1`, the trace files, including the alert log in the traditional text format, are located in the following directory (the Oracle base is `/u01/app/oracle`):

```
/u01/app/oracle/diag/rdbms/orcl/orcl1/diag
```

As this directory structure indicates, you can store the diagnostic data for multiple databases (as well as other Oracle products) under the same ADR base. For more on ADR, please see Chapter 2, which discusses the new fault diagnosability infrastructure.

Changes in the Install Options

There are several important install option changes for Oracle Database 11g, as summarized here:

- The Oracle Configuration Manager, which gathers configuration information pertaining to the software stored in the Oracle home directories, is integrated with the Oracle Universal Installer as an optional component.

- The Oracle Data Mining option is selected by default with the Enterprise Edition installation and is installed automatically when you run the `catproc.sql` script after creating the database.

- The Oracle XML DB option has been removed, since it isn't an optional component any longer. The Database Configuration Assistant installs and configures it. When you manually run the `catproc.sql` script, the XML DB is created automatically.
- Oracle Database Vault is an optional component, and to install this option, you must choose the Custom installation option during installation.

As with any other release, the Oracle11*g* database version deprecates certain components available in older releases. The most important of the deprecated components are as follows:

- iSQL*Plus
- Oracle Workflow
- Oracle Enterprise Manager Java Console
- Oracle Data Mining Scoring Engine
- Raw storage support (installer only)

New Oracle Database 11*g* Components

In Oracle Database 11*g*, the following new components are available while installing the server software:

- *Oracle Application Express (APEX)*: Oracle's browser-based rapid application development tool, formerly known as Oracle HTML DB, is enhanced in Oracle Database 11*g* with prepackaged applications for blogs, storefronts, and discussion forums. There are also new reporting capabilities and support for drag-and-drop forms layout. APEX is now part of the base Oracle CD instead of the companion CD.
- *Oracle SQL Developer*: Oracle's free database development productivity tool is a graphical version of SQL*Plus and is enhanced with new tuning capabilities in Oracle Database 11*g*. These enhancements include database activity reporting and expanded support for version control and visual query building. SQL Developer is automatically installed when you choose to perform a template-based database installation by choosing an installation option such as General Purpose and Transaction Processing.
- *Oracle Real Application Testing*: This component, which is automatically installed with the Enterprise Edition installation, consists of two new features, Data Replay and the SQL Performance Analyzer, both of which we discuss later in this chapter.
- *Oracle Configuration Manager (OCM)*: This is offered as an optional component during the server installation. The OCM collects information about software configuration in the Oracle home directories and uploads it to the Oracle configuration repository.
- *Oracle Warehouse Builder*: This is an enterprise business intelligence design tool and is installed as part of the Oracle Database server software.
- *Oracle Database Vault*: This tool, which enables you to secure business data, is installed with the Oracle Database 11*g* as an optional component, instead of being a component of the companion CD as in previous releases. The Oracle Database Vault installation means you now have a baseline security policy for the database. Security-related initialization parameters are given default values following the installation of the Oracle Database Vault.

Role and Privilege Changes

If you are using automatic storage management (ASM), you can now optionally create an additional OS-level group while installing the software or even after the installation. In addition, there is a new optional system privilege in Oracle Database 11*g* exclusively for ASM administration. If you're migrating from a database release older than Oracle Database 10*g* (10.2), you must also be aware of the changes made to the `connect` role.

New Privileges Group and Database Role for ASM

In Oracle Database 11*g*, there is a clear-cut demarcation between database administration and ASM administration. Previously, you performed all ASM administration as a user with the `sysdba` privilege. There is a new system privilege called `sysasm`, which you should grant to the user who needs to perform ASM administrative tasks. Users will need the `sysasm` privileges to create an ASM instance or cluster using OS authentication. In prior versions of Oracle, you created the `dba` and `oper` operating system groups when installing Oracle software. In Oracle Database 11*g*, you can optionally create a third operating system group called the `osasm` group. Oracle recommends you grant ASM access only to members of the `osasm` group.

■ **Note** There is a myriad of ASM enhancements for 11*g*, and thus we dedicate a chapter to reviewing the new features in managing ASM. Please refer to Chapter 9 for the new ASM features.

Both the new system privilege `sysasm` and the new operating system group `osadm` are optional in Oracle Database 11*g*. However, in future releases, Oracle may restrict access to ASM to members of the `osadm` operating system group in addition to requiring all ASM administrators to have the `sysasm` system privilege.

Deprecation of the connect Role

The `connect` role was deprecated in the Oracle Database 10.2 release. In fact, the role now has only the `create session` privilege, unlike in releases prior to Oracle Database 10.2, when it also had privileges other than `create session`. If you're upgrading to Oracle Database 11*g* from a release older than Oracle Database 10.2, any users with the `connect` role will cease to have all privileges other than the `create session` privilege.

After upgrading to Oracle Database 11*g* from release 9.2 or release 10.1, the `connect` role will have only the `create session` privilege; the other privileges granted to the `connect` role in earlier releases will be revoked during the upgrade. To identify which users and roles in your database are granted the `connect` role, use the following query:

```
SQL> select grantee from dba_role_privs
     where granted_role = 'CONNECT';
```

The upgrade script automatically takes care of adjusting the privileges of all Oracle-supplied users (such as `sys`, `system`, `outln`, and `dbsnmp`). For all other users with the `connect` role, you must explicitly grant all the privileges that were part of the old `connect` role after the upgrade is completed.

Note In previous versions, it was sometimes a difficult process to switch a database manually from Database Control to Grid Control. In Oracle Database 11*g*, you can simply use the new EMCP API to switch a database from Database Control to Grid Control.

Installing Oracle Database 11*g*

The Oracle Universal Installer steps for installing the Oracle Database 11*g* release software are similar to the steps for the Oracle 10*g* release. There are a few changes, however, which we'll highlight when we show the installation steps in this section. You use the `runInstaller` executable to invoke the GUI-based Oracle Universal Installer. If you've downloaded the server software from the Oracle web site, you must first uncompress the downloaded file. This will create a directory named `database`, under which you'll find the `runInstaller` script. Start the installation process by moving to the `database` directory and typing the following:

```
$ ./runInstaller
```

If you're installing from a DVD, invoke the installer by supplying the full path for the database directory on the DVD:

```
$ /<directory_path>/runInstaller
```

If you pass the minimal operating system requirements, the Oracle Universal Installer will open. Once the Oracle Universal Installer GUI shows up, the following are the steps in the installation process:

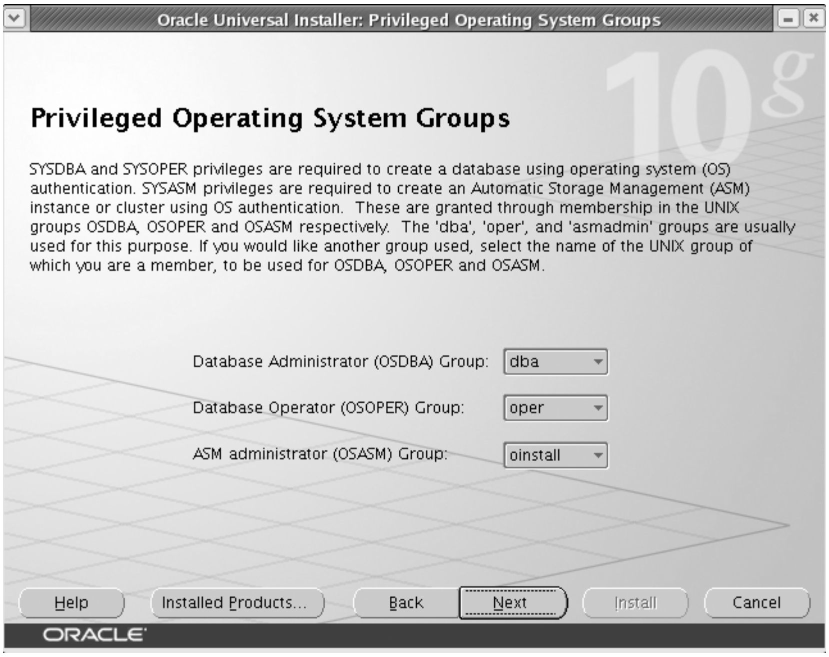
1. On the Select Installation Method page, you can select either Basic Installation or Advanced Installation. Select Advanced Installation, and click Next.

Tip Set the `TMP` and `TMPDIR` environment variables if `/tmp` is too small for the installation.

2. Select Installation Type. You're given three choices—Enterprise Edition, Standard Edition, and Custom. Choose Enterprise Edition, and click Next.
3. On the Install Location page, specify the path for the Oracle base and Oracle home locations, which is where the Oracle Universal Installer will install the database files. Click Next.
4. On the Product-Specific Prerequisite Checks page, the Oracle Universal Installer will verify that your environment meets the minimum requirements for installing the various products you want to install. These checks include the kernel parameters, swap space requirements, validation of the Oracle base location, and network configuration requirements. It's a good idea to go ahead and fix any warnings produced by the Oracle Universal Installer at this stage, say by updating the kernel on a Linux system, although you can get away with not doing so in most cases since the Oracle Universal Installer offers you the choice of continuing despite a warning. Once you pass the requirement checks, click Next.

5. Select Configuration Option. You can choose to create a database, configure ASM, or just install the Oracle 11*g* binaries. For the last option, choose Install Software Only, and click Next.

6. You'll see the Privileged Operating System Groups page next, as shown in Figure 1-1. This step is new in Oracle Database 11*g*. In addition to the `sysdba` and `sysoper` privileges you're familiar with, Oracle now recommends you create the new system privilege called `sysasm` for enabling the management of ASM. Oracle also recommends you create a new Unix/Linux group now, called `osasm`, for ASM administrators.


Oracle Universal Installer: Privileged Operating System Groups

Privileged Operating System Groups

SYSDBA and SYSOPER privileges are required to create a database using operating system (OS) authentication. SYSASM privileges are required to create an Automatic Storage Management (ASM) instance or cluster using OS authentication. These are granted through membership in the UNIX groups OSDBA, OSOPER and OSASM respectively. The 'dba', 'oper', and 'asmadmin' groups are usually used for this purpose. If you would like another group used, select the name of the UNIX group of which you are a member, to be used for OSDBA, OSOPER and OSASM.

Database Administrator (OSDBA) Group: dba

Database Operator (OSOPER) Group: oper

ASM administrator (OSASM) Group: oinstall

Figure 1-1. *The Privileged Operating System Groups page*

7. On the Summary page, the Oracle Universal Installer summarizes the installation, including the names of all the components it will install. Click Next after reviewing the summary.

8. On the Install page, you'll see the progress of the installation. Once the installation finishes successfully, exit the Oracle Universal Installer by first clicking Exit and then clicking Yes.

If you choose to create a new starter database (step 5), the Oracle Universal Installer invokes the Database Configuration Assistant (DBCA) to create the database. We discuss creating a new database with the DBCA in the following section. You'll see the new features we discuss there (such as specifying the automatic memory configuration details) if you choose to create a new database during installation itself by choosing the Create a Database option here. By default, Oracle includes the new Secure Configuration option, which configures the database with auditing options as well as password policy and expiration settings. If you want, you can disable the new enhanced security controls during the installation.

Also, if you choose to create the starter database during the installation, you'll get an option to configure the Oracle Configuration Manager. You'll need to configure the Oracle Configuration Manager in order to associate the database configuration information with Oracle MetaLink. You can then link your service requests to MetaLink with your configuration information. Chapter 2 explains more about the Oracle Configuration Manager in the context of the new Support Workbench feature.

New Features in Database Creation

You can create a new Oracle database either with manual commands at the SQL prompt or with the help of the Database Configuration Assistant (DBCA). Oracle Database 11*g* contains some changes in both methods of creating new databases. Some changes such as new initialization parameters are, of course, common to both techniques, so first we'll cover the new initialization parameters in Oracle Database 11*g*.

New Initialization Parameters

When you create a new Oracle Database 11*g* version database, you'll want to know about some important changes regarding the Oracle initialization parameters. There are both new initialization parameters and some deprecated parameters. A few significant new initialization parameters in Oracle 11*g* affect the implementation of certain key new Oracle 11*g* features. You don't necessarily have to set any of these new parameters when you're upgrading to Oracle Database 11*g*—or even when creating a new Oracle 11*g* database, for that matter.

You can now create a traditional text initialization parameter file or a server parameter file from the current values of the initialization parameters in memory. Chapter 3 shows you how to do this. Another new Oracle Database 11*g* feature is that the initialization parameter settings are recorded in the alert log in a way supposed to make it easy for you to copy and paste those settings if you want to create a new parameter file. When you start up the instance, Oracle writes all initialization parameter values to the alert log in valid syntax, so you can copy and paste this into a new initialization parameter file if you want.

Oracle's wonderful compatibility feature means that your 9*i* or 10*g* database will work under the Oracle 11*g* software with nary a change. The lowest possible setting you can use for the `compatible` initialization parameter is 10.0.0 before you upgrade to the Oracle Database 11*g* release. The default value is 11.1.0, and the maximum value is 11.1.0.n.n. When you set the `compatible` parameter to the minimum value of 10.0.0, the newly upgraded database can avail of only a small subset of the new features of Oracle Database 11*g*. However, some of these new initialization parameters control several of the most important innovations of Oracle 11*g*, topics that we discuss later this book. Scan through the following sections to see which initialization parameters you must use, if you want to adopt key new Oracle 11*g* features. All these parameters will be explained in more detail in the relevant chapters later in this book. The following sections are a quick summary of the most important initialization parameter changes in Oracle 11*g*.

Memory-Related Parameters

One of the biggest changes in Oracle Database 11*g* is the new *automatic memory management* feature under which both the major components of Oracle's memory allocation—the shared global area (SGA) and the program global area (PGA)—will be automatically expanded and shrunk, based on the needs of the instance. All you need to do is set the values for two memory-related parameters, `memory_target` and `memory_max_target`:

- The `memory_target` parameter sets the system-wide usable memory and lets Oracle tune both the SGA and PGA, changing the values of the SGA and the PGA automatically based on the demands of the running Oracle instance. You can dynamically change the value for this parameter using the `alter system` command.
- The `memory_max_target` parameter sets the maximum value of memory Oracle can use. That is, the value you set for the `memory_max_target` parameter is the maximum value up to which you can adjust the `memory_target` parameter's value.

You can enable automatic memory management by setting the value of the `memory_target` parameter and the `memory_max_target` parameter in the initialization parameter file when creating a new database. You can also add them later to the initialization parameter file after database creation, but you have to bounce the database for automatic memory management to take effect. Here's an example showing how to specify the new memory-related parameters, if you started your database with an initialization parameter file:

- `memory_max_target` = 500MB
- `memory_target` = 350MB
- `sga_target` = 0
- `pga_aggregate_target` = 0

This set of initialization parameters ensures that the server allocates 350MB of memory to Oracle right away. Oracle will allocate this memory among the SGA and the PGA. You can dynamically change the value of the `memory_target` parameter up to the maximum of 500MB set by the `memory_max_target` parameter. Note that you must set both the `sga_target` and `pga_aggregate_target` initialization parameters to 0 if you don't want to set any minimum values for the sizes of the SGA and the PGA. For testing purposes on both Linux and Windows servers, you can use as little as 120MB as the value for the `memory_target` parameter.

We discuss the automatic memory management feature in detail in Chapter 3.

PL/SQL Native Compilation Parameter

In Oracle Database 11*g*, it's easier than ever to enable PL/SQL native compilation, which offers greater performance benefits. In Oracle Database 10*g*, you had to use the initialization parameter `plsql_native_library_dir` to set a directory, as well as specify the `plsql_native_library_sbdir_count` parameter to enable native compilation of PL/SQL code. In addition, you also had to use the `spnc_commands` file in the `plsql` directory under the Oracle home. In Oracle Database 11*g* you use just one initialization parameter, `plsql_code_type`, to turn on native compilation. You don't need a C compiler, and you don't have to manage any file system DLLs either. You don't need to create any directories or use the `spnc_commands` file. You can also adopt native compilation for Java. We explain native compilation in more detail in Chapter 4.

■Note Real native compilation is rumored to be twice as fast as C native. The Whetstone benchmark runs 2.5 times faster under real native compilation.

The diagnostic_dest Parameter

The initialization parameter `diagnostic_dest` is new in Oracle 11*g*. This parameter points to the location of the new automatic diagnostic repository. The `diagnostic_dest` parameter replaces the `user_dump_dest`, `background_dump_dest`, and `core_dump_dest` initialization parameters in past releases. The database ignores any values you set for these parameters if you've also set the `diagnostic_dest` parameter.

If your Oracle base location is `/u01/app/oracle`, your database name is `orcl`, and the instance name is also `orcl`, then by default the ADR home directory will take the following form:

```
/u01/app/oracle/diag/rdbms/orcl/orcl
```

■Note You can set a different ADR for each instance of a Real Application Cluster (RAC). Oracle recommends that you do so, specifying the same value for the parameter for each instance.

In the `orcl` directory, you'll find various subdirectories such as `alert`, `incident`, and `trace`. The `trace` directory is where the `user_dump_dest` and `core_dump_dest` files used to go in earlier releases.

You can specify that the ADR be located in a nondefault location by setting the `diagnostic_dest` initialization parameter. The basic directory for the ADR, known as the ADR home, will have the following directory structure:

```
<diagnostic_dest>/diag/rdbms/<dbname>/<instname>
```

If you set the `diagnostic_dest` parameter to `/u05/app/oracle`, the database name is `orcl`, and the instance name is `orcl1`, then the following would be your ADR home directory:

```
/u05/app/oracle/diag/rdbms/orcl/orcl1
```

Chapter 2 discusses the ADR in detail.

New Result Cache–Related Parameters

You're familiar with the caching of queries in the shared pool component of the SGA. In Oracle Database 11*g*, Oracle has gone quite a bit further and caches the actual results of queries in memory. The caching is done in a new component of the SGA called the *result cache*. Result caching dramatically improves performance for frequently run queries when there are few changes in data.

For a table or view to be considered for result caching, you must alter the table using the `result_cache_mode` clause. You must set the new initialization parameter `result_cache_mode` to the appropriate value if you want the database to consider all queries or to only those queries that involve tables and views for which you have set the `result_cache` option. You can also cache PL/SQL function results in addition to SQL query results. In addition to the `result_cache_mode` option, there are several result cache–related initialization parameters, such as the `result_cache_max_result`, `result_cache_max_size`, and `result_cache_remote_expiration` parameters. You can also cache query results on the client side. When you use client-side query caching, you can specify the new parameters `client_result_cache_size` and `client_result_cache_lag`. We discuss all these new parameters as part of the result cache feature in Chapter 4.

Note When you set the compatible parameter to 11.0.0 or greater, the server parameter file is written in a new format to comply with Oracle's HARD initiative, which helps prevent writing corrupted data to disk.

Parameter to Control DDL Lock Timeout

One of the important new features of the Oracle 11*g* database is the new DDL locking duration control feature. The `ddl_lock_timeout` parameter lets you specify the length of time a DDL statement will wait for a DML lock. This feature comes in handy when you want to perform online reorganization, where a DML lock by a user may prevent a DDL operation from succeeding. You can practically specify that a DDL statement wait forever by setting the parameter to the maximum allowed value, which is 1,000,000 seconds. We discuss the `ddl_lock_timeout` parameter in detail in Chapter 3.

SecureFiles-Related Parameter

The new Oracle SecureFiles feature is a major revamping of the implementation of Large Objects (LOBs). By using the new initialization parameter `db_securefile`, you can specify whether to treat a LOB file as a SecureFiles file. Please see Chapter 12 for a detailed discussion of SecureFiles.

Note The `job_queue_processes` parameter has been moved to the basic initialization parameters list in Oracle Database 11*g*. Although this is not a big deal on the face of it, the point is that Oracle is saying you don't have to worry about setting a value for the `job_queue_processes` parameter in most cases, since it's classified as a basic initialization parameter. The `job_queue_processes` parameter can take a value from 0 to 1000. If you set it to 0, you can run `DBMS_SCHEDULER` jobs, but not the `DBMS_JOB`-based jobs. If you set the parameter to any value from 1 to 1000, both `DBMS_JOB`-based and `DBMS_SCHEDULER`-based jobs will run.

The db_ultra_safe Parameter

The new parameter `db_ultra_safe` sets default values for parameters, such as the `db_block_checking` parameter, that control protection levels. To be precise, you can control three corruption-checking parameters—`db_block_checking`, `db_block_checksum`, and `db_lost_write_protect`—by specifying values for the `db_ultra_safe` parameter.

The db_ultra_safe_parameter can take three values—off, data only, and data and index. By default, the db_ultra_safe parameter is set to off, meaning that any values you set for any of the three parameters won't be overridden. If you set the db_ultra_safe parameter value to data only, the following will be true:

- db_block_checking will be set to medium.
- db_lost_write_protect will be set to typical.
- db_block_checksum will be set to full.

If you set the db_ultra_safe parameter value to data and index, the following will be true:

- db_block_checking will be set to full.
- db_lost_write_protect will be set to typical.
- db_block_checksum will be set to full.

Security-Related Parameters

There are two important security-related initialization parameters that are new in Oracle Database 11*g*. The first parameter, sec_case_sensitive_logon, lets you enable and disable password case-sensitivity in the database. By default, password case-sensitivity is enabled in Oracle Database 11*g*.

The other new security-related initialization parameter is the parameter sec_max_failed_login_attempts, which specifies the maximum number of times a client can make a connection attempt to a server. The default value of this parameter is 10.

Please see Chapter 5 for more details about the new case-sensitive password feature as well as the sec_max_failed_login_attempts parameter.

Optimizer-Related Parameters

There are several important new optimizer-related initialization parameters that are intended to support powerful new features such as SQL Plan Management, private statistics, and invisible indexes. We discuss all these features in subsequent chapters and merely introduce the relevant new initialization parameters in this section.

Oracle Database 11*g* replaces the old plan stability feature with the new SQL Plan Management feature. A change in the execution plan of an important SQL statement can potentially degrade performance. To avoid this performance degradation, the database selects optimal SQL plan baselines and prevents the optimizer from changing the execution plan of a statement until the new plan is found to be definitely superior to the existing SQL baseline plan (lower cost). You can enable automatic SQL plan capture so the database can capture and maintain SQL plan history using information from the optimizer.

By default, automatic plan capture is disabled, and you can enable it by setting the optimizer_capture_sql_plan_baselines parameter to true. Chapter 4 contains a detailed discussion of the SQL Plan Management feature.

Use the new initialization parameter `optimizer_use_sql_baselines` to enable the use of SQL plan baselines that are stored in what's called the SQL *management base*. If you enable SQL plan baselines, the cost optimizer will search in the SQL management base for a SQL plan baseline for the SQL statement being currently compiled. If there is a SQL plan outline available, the cost optimizer will select the baseline plan with the least cost.

A third new optimizer-related parameter, `optimizer_private_statistics`, allows you to specify the use of private statistics during the compilation of SQL statements. Please refer to Chapter 4 for details about the major optimizer-related new features.

Finally, the new parameter `optimizer_use_invisible_indexes` lets you enable and disable the use of invisible indexes, a significant new feature that we'll explain in Chapter 3.

DBCA Enhancements

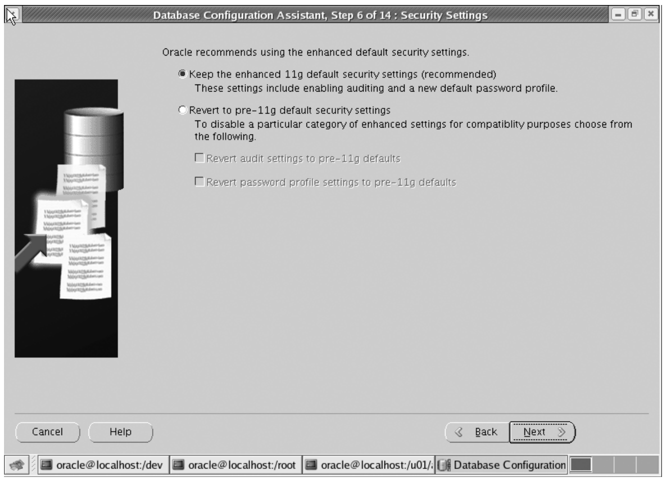
The DBCA provides an alternative to the manual creation of a new Oracle database. In Oracle Database 11*g*, you should be aware of a couple of changes when you're creating a new database with the help of the DBCA. These changes pertain to security settings and the choice of memory allocation for the new database.

We summarize the changes in the DBCA by listing all the steps required to create a new database with the DBCA. Most of the steps are identical to the steps you followed in the Oracle Database 10*g* release, but there are two new steps and a couple of modified steps. Let's review the database creation steps when you use the DBCA to create a new Oracle 11*g* database:

1. On the DBCA Operations page, select the Create a Database option.
2. On the Database Templates page, select one of the following database types: Data Warehouse, General Purpose, or Transaction Processing.
3. On the Database Identification page, select the database name and the system identifier (SID).
4. On the Management Options page, select Database Control or Grid Control.
5. On the Database Credential page, specify passwords for accounts such as `sys` and `system`.
6. On the Security Settings page, choose the security settings for the new database (this is new in Oracle Database 11*g*). DBCA provides secure database configuration features by default. You can turn off security configuration in the new database if you want. Here are the important features related to secure database configuration:

   - Audit settings
   - Password profiles
   - Revoking grants to the `public` role

   You'll find more about secure database configuration in Chapter 6. Figure 1-2 shows the new Security Settings page.


Database Configuration Assistant, Step 6 of 14 : Security Settings

Oracle recommends using the enhanced default security settings.

[✓ Keep the enhanced 11g default security settings (recommended)]
These settings include enabling auditing and a new default password profile.

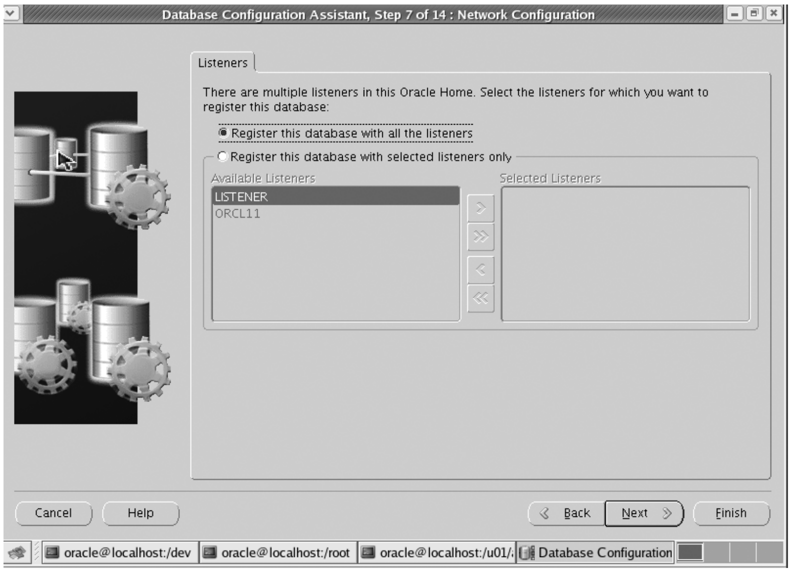
Revert to pre-11g default security settings
To disable a particular category of enhanced settings for compatiblity purposes choose from the following.

☐ Revert audit settings to pre-11g defaults

☐ Revert password profile settings to pre-11g defaults

Figure 1-2. *DBCA's new Security Settings page*

7. On the Network Configuration page, select the listener(s) for which you plan to register the new database. (This is new in Oracle Database 11*g*.) Figure 1-3 shows the new Network Configuration page.


Database Configuration Assistant, Step 7 of 14 : Network Configuration

Listeners

There are multiple listeners in this Oracle Home. Select the listeners for which you want to register this database:

[✓ Register this database with all the listeners]

Register this database with selected listeners only

Available Listeners: LISTENER, ORCL11

Selected Listeners

Figure 1-3. *DBCA's new Network Configuration page*

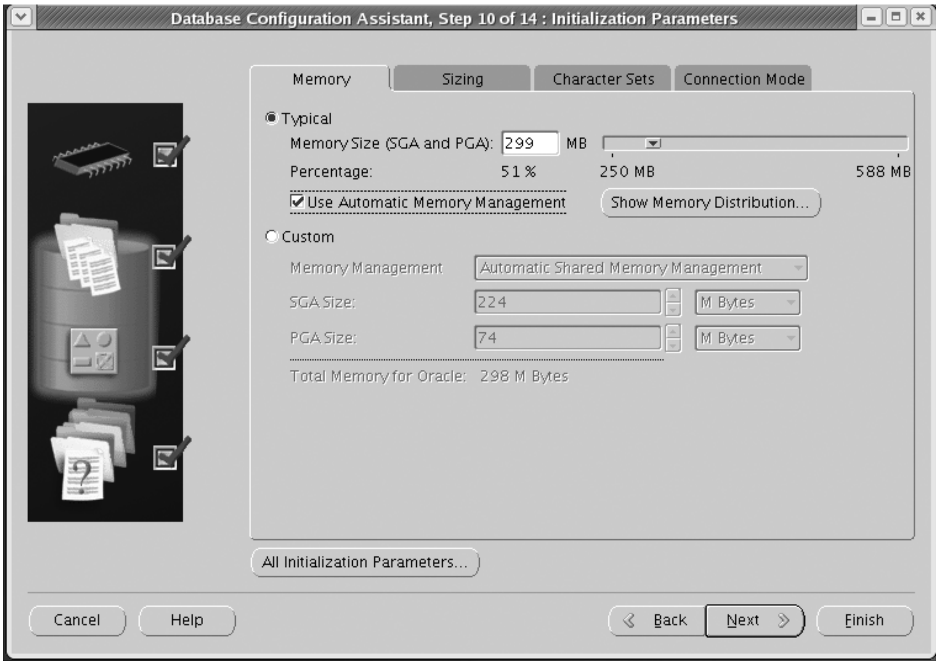
8. On the Storage options page, select the type of storage mechanism.

9. On the Database File Locations page, specify the Oracle software home and the directory for the database files, or select the Oracle-Managed Files (OMF) option for database files.

10. On the Recovery Configuration page, specify the archivelog/noarchivelog choice and the flash recovery area location.

11. On the Database Content page, specify the sample schemas and custom scripts to be run after database creation.

12. On the Initialization Parameters page, alter the default settings for various initialization parameters, such as memory, character sets, and so on. This screen lets you select among the three types of memory allocation—automatic memory management, automatic shared memory management, or manual shared memory management. (This option has been modified in Oracle Database 11*g*.) Figure 1-4 shows this page.


Database Configuration Assistant, Step 10 of 14 : Initialization Parameters

Memory | Sizing | Character Sets | Connection Mode

[✓ Typical]
Memory Size (SGA and PGA): 299 MB
Percentage: 51 % 250 MB 588 MB
[✓] Use Automatic Memory Management
Custom
Memory Management: Automatic Shared Memory Management
SGA Size: 224 M Bytes
PGA Size: 74 M Bytes
Total Memory for Oracle: 298 M Bytes

Figure 1-4. *The DBCA's New Initialization Parameters page*

13. On the Database Storage page, make changes in the storage structure of the database.

14. On the Database Creation Options page, choose from three options: Create Database, Save As a Database Template, or Generate Database Creation Scripts.

In step 6, you can choose to use the new enhanced default security settings, or you can disable the security controls if you want. The default security controls are part of the new Secure Configuration option, which configures database auditing and password policy and expiration settings. Thus, the default configuration includes the Secure Configuration option, but you can check the Disable Security Settings box to disable the enhanced security controls. The new database is then installed with the default security options for Oracle Database 10*g* Release 2. You can configure the Secure Configuration option later by invoking the DBCA and altering the security settings of the database. Oracle strongly recommends you adopt the new Secure Configuration option. Note that if you install Oracle Database Vault, you can't change the Secure Configuration option using the DBCA.

In step 12, you get a chance to modify the default initialization parameter settings. The only real change in this section pertains to the allocation of Oracle memory. As in Oracle Database 10*g*, you can choose between Typical, which requires little or no configuration on your part, and Custom, which requires more configuration. If you choose the Typical method of memory allocation, Oracle automatically tunes memory for the instance using automatic memory management. The amount of memory Oracle allocates to the instance will be a percentage of the overall physical memory on the server. If you choose the Custom method, you must specify the memory you want to allocate to the Oracle instance, and you must also choose from one of the following three types of memory allocation:

- Automatic memory management (new in Oracle Database 11*g*)
- Automatic shared memory management
- Manual shared memory management

You select automatic management by first selecting the Typical option on the Initialization Parameters page and then selecting the Use Automatic Memory Management option. Automatic memory management, as explained earlier in this chapter, is new to Oracle Database 11*g*, and you can enable it through the new unitization parameters `memory_target` and `memory_max_target`. We explain automatic memory management in detail in Chapter 3.

New Oracle Background Processes

Some new Oracle background processes in Oracle Database 11*g* can help with some of the new features introduced as part of this release. Here are the important new background processes in Oracle Database 11*g*:

- *FBDA*: The flashback data archive process archives data from all tables that are enabled for flashback archive. The process stores all pre-change images of the table rows in the flashback archive following a DML operation such as an update or a deletion. Flashback archiving is an important Oracle Database 11*g* new feature that we discuss in Chapter 3.
- *SMCO*: The space management coordinator process is in charge of coordinating the work of space management–related tasks such as space reclamation, for example.
- *RCBG*: The result cache background process supports the new result cache feature.

You'll get a chance to review the key new background processes in more detail when we discuss the new features that use these background processes.

■**Note** On Windows servers, the Volume Shadow Copy Service infrastructure lets you make snapshots, called *shadow copies*. When you install Oracle Database 11*g*, the Oracle Shadow Copy Service Writer is automatically installed.

New Oracle-Supplied PL/SQL Packages

Several new Oracle-supplied PL/SQL packages provide support for various new features. We describe the most important packages briefly here; you'll find explanations regarding the use of each of these packages in later chapters:

- `DBMS_CONNECTION_POOL`: Supports the new database resident connection pooling feature (explained in Chapter 3)
- `DBMS_SQLPA`: Supports the new SQL Performance Analyzer feature, which we explain later in this chapter
- `DBMS_ADDM`: Facilitates the management of the Automatic Database Diagnostic Monitor
- `DBMS_SPM`: Supports the new SQL plan management feature (explained in Chapter 4)
- `DBMS_AUTO_TASK_ADMIN`: Supports the automated maintenance task (explained in Chapter 3)
- `DBMS_COMPARISION`: Lets you compare the database objects in two different databases (explained in Chapter 3)
- `DBMS_SQLDIAG`: Lets you manually invoke the new SQL Test Case Builder (explained in Chapter 2)
- `DBMS_HM`: Supports the new database health management feature. The package helps you run Health Monitor checks and retrieve the resulting reports. We discuss the Health Monitor in Chapter 2.
- `DBMS_RESULT_CACHE`: Supports the new result cache feature (explained in Chapter 4)
- `DBMS_WORKLOAD_CAPTURE` and `DBMS_WORKLOAD_REPLAY`: Support the new Database Replay feature, which we discuss later in this chapter

Of course, as with all new database releases, several of the older Oracle-supplied packages have been updated. Please refer to the *Oracle P/L SQL Packages* manual for full details on all Oracle packages that have been updated in Oracle Database 11*g*.

■ **Note** In Oracle Database 10*g*, manual undo management using rollback segments was deprecated, but the default value for the `undo_management` initialization parameter was still `manual`. In Oracle Database 11*g*, by default the value of this parameter is set to `auto`. If you create a new database with the DBCA, Oracle automatically creates the `undo` tablespace for you. You can still operate a database in the manual undo management mode, but Oracle strongly recommends using automatic undo management owing to its benefits, and we concur.

Upgrading to Oracle Database 11*g*

Upgrading to the new Oracle Database 11*g* release is similar to the upgrade process in 10*g*, but it contains more sophisticated pre-upgrade checks as well as simplified error management. The changes in Oracle 11*g* simplify the database upgrade process while making it faster. Since both the DBUA and the manual upgrade processes utilize the same scripts, such as the `utlu111i.sql` and `utlu111s.sql` scripts, the improvements are common to both methods. The pre-upgrade and the post-upgrade scripts in Oracle 11*g* work the same way as they did in the Oracle 10*g* version.

Upgrading and the Compatibility Factor

The default compatibility value for Oracle Database 11*g* version 11.1 is 11.1. However, you can upgrade to Oracle Database 11*g* with a minimum compatibility value of 10.0.0. Before you perform the database upgrade, you must set the compatibility level for your new database by setting a value for the initialization parameter `compatible`. If you omit this parameter altogether from the new initialization parameter file, it defaults to 11.1.0. Oracle recommends you use the minimum value for the compatible parameter during the upgrade (10.0.0). This way, if your upgrade doesn't go well for some reason and you have to back out of the upgrade process, the upgraded database won't become incompatible with your previous (10*g.x*) release.

If you upgrade to 11.1 and keep the `compatible` parameter at 10.0.0.0, only a small portion of the new features, limited to those that don't make incompatible database structures on disk, will be allowed. The vast majority of the 11*g* features will make these permanent changes, however, so the lower compatibility level disables those new features. Once you confirm that the upgrade did finish successfully, you can change the value of the `compatible` parameter to 11.1 or greater, depending on the software release you installed. Just remember that once you do this and restart the database, you can't downgrade to the previous release.

Before you increase the setting of the `compatible` initialization parameter, back up the database, and either edit the initialization parameter (`compatible=11.1.0`, for example) or, if you're using the server parameter file, use the following statement to make the change:

```
SQL> alter system set compatible ='11.1.0' scope=spfile;
```

Once you change the value of the `compatible` parameter, shut down the database (`shutdown immediate`), and restart it with the `startup` command. To go back to the old compatibility level after this if something doesn't work right, you must return to the backup you made of the pre-upgrade database.

Upgrade Path to Oracle 11*g*

Whether you can directly upgrade your current Oracle database to Oracle 11*g* or have to perform an upgrade to an intermediate release first depends on your current Oracle database release. Oracle supports a direct upgrade to Oracle Database 11*g* Release 1, if you're migrating from a *9.2.04 or newer* release of the Oracle database software. If you want to upgrade to Oracle Clusterware 11*g* release 1 (11.1), then you must upgrade from an Oracle Clusterware release 10.2.0.3 or newer. Here's a summary of the upgrade path to Oracle 11*g* for Oracle database releases older than 9.2.04.

- 7.3.3 (or lower) ➤ 7.3.4 ➤ 9.2.0.8 ➤ 11.1
- 8.0.5 (or lower) ➤ 8.0.6 ➤ 9.2.0.8 ➤ 11.1
- 8.1.7 (or lower) ➤ 8.1.7.4 ➤ 9.2.0.8 ➤ 11.1
- 9.0.1.3 (or lower) ➤ 9.0.1.4 ➤ 9.2.0.8 ➤ 11.1
- 9.2.0.3 (or lower) ➤ 9.2.0.8 ➤ 11.1

■ **Note** In general, Oracle's logic is that, for a direct upgrade, Oracle will support whatever version of the database software is supported at the time of general availability of the new product. This also includes compatibility information regarding database links and communications between the versions of Oracle.

As you can see, some of the older versions have to be migrated to multiple intermediate releases before you can upgrade to Oracle Database Release 1 (11.1). As in the Oracle Database 10*g* release, you can upgrade with the help of Oracle-provided scripts or use the DBUA to simplify matters. For certain databases, you can also consider using the Data Pump Export and Import utilities as well as the `create table as select` (CTAS) statement to copy all or part of a database into a new database created with the Oracle Database 11*g* server software.

You can upgrade an Oracle 8*i*, Oracle 9*i*, or Oracle Database 10*g* client to the Oracle 11.1 release. You can use the Oracle 11.1 client to access any Oracle 8*i*, Oracle 9*i*, Oracle Database 10*g*, and Oracle Database 11*g* (11.1) databases.

In the following sections, we'll review the changes in the manual upgrade first and then look at upgrading with the DBUA.

A Manual Upgrade Example

A manual upgrade of an Oracle database involves executing a set of Oracle-provided SQL scripts that are stored in the `$ORACLE_HOME/rdbms/admin` directory. The first script you run before starting the upgrade process is the `utlu111i.sql` script, also called the Pre-Upgrade Information Tool. In Oracle Database 11*g*, this utility provides more information than before to help you during the upgrade process.

If you're upgrading from Oracle Database 10*g* to Oracle Database 11*g*, you must copy three scripts (`utlu11i.sql`, `utilu111s.sql`, and `utlu111x.sql`) from the 11*g* database's file system (`$ORACLE_HOME/rdbms/admin/`) to a staging directory on the 10*g* database's file system.

Before you can actually run the upgrade script provided by Oracle, you must gather information regarding the upgrade requirements by running the Pre-Upgrade Information Tool (invoked by the `utlu111i.sql` script). Once you make sure you've taken care of all the warnings and recommendations made by the Pre-Upgrade Information Tool, you can upgrade databases to the Oracle Database 11*g* release using the same type of scripts as in the previous releases. In this case, the database upgrade scripts are named in the following way:

- `utlu111i.sql`: This is the previously described pre-upgrade script.
- `catupgrd.sql`: This is the actual upgrade script and is similar to the script in previous releases. The major change now is that it has been restructured to support parallel upgrades of the database.
- `utlu111s.sql`: This is the upgrade status utility script, which you invoke after completing the database upgrade.

Here's a brief summary of the manual direct upgrade process from a 10.2 Oracle release database to the Oracle 11.1 release. Remember that Oracle supports a direct upgrade to the Oracle Database 11*g* release from a 9.2.0.4, 10.1, or 10.2 release database. You must log in as the owner of the Oracle 11.1 release Oracle home directory.

Note Your database upgrade may take a long time since the database will collect optimizer statistics for any dictionary tables that are missing statistics or whose statistics were significantly changed during the upgrade. You can speed up the upgrade process by gathering statistics for these tables by using the `DBMS_STATS.GATHER_DICTIONARY_STATS` procedure before starting the upgrade.

1. Log in as the owner of the Oracle 11.1 release Oracle home directory, and copy the `utlu111.i sql` file from the `$ORACLE_HOME/rdbms/admin` directory to another directory such as `/tmp`.

2. Log in as the owner of the Oracle home directory of the database you want to upgrade, and run the `utlu111.i sql` script (from the `/tmp` directory, where you copied it to) to get the pre-upgrade information such as the necessary initialization parameter changes, tablespace space requirements, and so on. Spool the results of the script execution so you can review the output later. For example, the following is a run of the `utlu111i.sql` script on one of our own systems:

```
SQL> spool upgrade.log
SQL> @utlu111i.sql
Oracle Database 11.1 Upgrade Information Utility    03-23-2007 13:36:14
.
**********************************************************************
Database:
**********************************************************************
--> name:       ORCL
--> version:    10.2.0.1.0
--> compatible: 10.2.0.1.0
--> blocksize:  8192
.
**********************************************************************
Tablespaces: [make adjustments in the current environment]
**********************************************************************
--> SYSTEM tablespace is adequate for the upgrade.
.... minimum required size: 593 MB
.... AUTOEXTEND additional space required: 123 MB
--> UNDOTBS1 tablespace is adequate for the upgrade.
.... minimum required size: 454 MB
.... AUTOEXTEND additional space required: 429 MB
--> SYSAUX tablespace is adequate for the upgrade.
.... minimum required size: 306 MB
.... AUTOEXTEND additional space required: 66 MB
--> TEMP tablespace is adequate for the upgrade.
.... minimum required size: 61 MB
.... AUTOEXTEND additional space required: 41 MB
.
**********************************************************************
```

```
Update Parameters: [Update Oracle Database 11.1 init.ora or spfile]
**********************************************************************
WARNING: --> "streams_pool_size" needs to be increased to at least 50331648
WARNING: --> "session_max_open_files" needs to be increased to at least 20
.
**********************************************************************
Renamed Parameters: [Update Oracle Database 11.1 init.ora or spfile]
**********************************************************************
-- No renamed parameters found. No changes are required.
.
**********************************************************************
Obsolete/Deprecated Parameters: [Update Oracle Database 11.1 init.ora or spfile]
**********************************************************************
-- No obsolete parameters found. No changes are required
.
**********************************************************************
Components: [The following database components will be upgraded or installed]
**********************************************************************
--> Oracle Catalog Views                         [upgrade]          VALID
--> Oracle Packages and Types                    [upgrade]          VALID
--> JServer JAVA Virtual Machine                 [upgrade]          VALID
--> Oracle XDK for Java                          [upgrade]          VALID
--> Oracle Workspace Manager                     [upgrade]          VALID
--> OLAP Analytic Workspace                      [upgrade]          VALID
--> OLAP Catalog                                 [upgrade]          VALID
--> EM Repository                                [upgrade]          VALID
--> Oracle Text                                  [upgrade]          VALID
--> Oracle XML Database                          [upgrade]          VALID
--> Oracle Java Packages                         [upgrade]          VALID
--> Oracle OLAP API                              [upgrade]          VALID
--> Oracle interMedia                            [upgrade]          VALID
--> Spatial                                      [upgrade]          VALID
--> Expression Filter                            [upgrade]          VALID
--> Rule Manager                                 [upgrade]          VALID
.
**********************************************************************
Miscellaneous Warnings
**********************************************************************
WARNING: --> Database contains stale optimizer statistics.
.... Refer to the 10g Upgrade Guide for instructions to update
.... statistics prior to upgrading the database.
.... Component Schemas with stale statistics:
....   SYS
.

PL/SQL procedure successfully completed.

SQL> spool off
```

Note that the `utlu111i.sql` script (also called the Pre-Upgrade Information Tool) will estimate the size requirements for the system and sysaux tablespaces. However, these tablespace size estimates may sometimes be unrealistically small. To avoid problems during the upgrade, Oracle recommends you set one datafile in each of these tablespaces to extend automatically during the upgrade (you can do this by using the `autoextend on maxsize unlimited` clause). The Pre-Upgrade Information Tool shows that in this particular case, there are no changes necessary before the upgrade to the Oracle 11*g* release. In general, the Pre-Upgrade Information Tool will recommend the following:

- Removing obsolete initialization parameters
- Adjusting initialization parameters
- Adding space to key tablespaces such as system and sysaux

3. Shut down the database cleanly (with the `shutdown immediate` command). If you're using a Windows system, stop the Oracle service first using the `net stop` command, and then delete the service using the `oradim` utility. Use the `oradim` utility from the Oracle Database 11*g* release to create a new Oracle Database 11*g* release instance.
4. Back up the database before starting the upgrade.
5. Make the necessary initialization parameter changes, remove any parameters shown as obsolete by the Pre-Upgrade Information Tool from the current `init.ora` file, and move that file to the new Oracle home under 11*g*. Ensure that the initialization parameter `compatible` is set to 11.1. If you are using a password file, you must move it to the new Oracle 11*g* Oracle home.
6. Change the environment variables `ORACLE_HOME`, `PATH`, and `LD_LIBRARY_PATH` so they point to the new Oracle Database 11*g* Release 1 (11.1) directories. Also, make sure you set the `ORACLE_SID` variable correctly to point to the database you're upgrading.

■ **Note** If you take the sysaux tablespace offline after an upgrade, it could lead to potential performance issues, because the database upgrade script moves any SQL profiles you may have from the system tablespace to the sysaux tablespace when you upgrade from Oracle 10.1 to Oracle 11.1.

7. Log in to the server as the owner of the Oracle home directory for the new Oracle Database 11*g* release. Start SQL*Plus after first changing to the new 11*g* `$ORACLE_HOME/rdbms/admin` directory.
8. Start up the database in upgrade mode as shown here after logging in as a user with the `sysdba` privilege:

```
SQL> startup upgrade pfile=$ORACLE_HOME/dbs/initorcl.ora
```

We're assuming you're upgrading from a 10.*x* release to the Oracle Database 11*g* release. If you're upgrading from an Oracle Database 9*i* (Release 2) version, you must first create a sysaux tablespace.

9. Set up spooling so you have a log of the upgrade:

```
SQL> spool upgrade.log
```

10. Run the upgrade script, `catupgrd.sql`, located in the `$ORACLE_HOME/rdbms/admin` directory, to upgrade the database to the Oracle 11*g* release:

```
SQL> @catupgd.sql
```

The `catupgd.sql` script calls various upgrade scripts and shuts down the database after finishing the upgrade. It may upgrade or install several database components.

11. Once the upgrade script finishes successfully, restart the database in normal mode (not in upgrade mode) in order to initialize the system parameters:

```
SQL> startup
```

Starting up the database after the completion of the database upgrade clears the buffers and caches and ensures the integrity and consistency of the upgraded database.

12. Run the `utl111s.sql` script to view the results of the upgrade:

```
SQL> @utlu111s.sql
.
Oracle Database 11.1 Upgrade Status Utility          03-23-2007 15:03:58
.
Component                         Status      Version    HH:MM:SS
.
Oracle Server                     VALID      11.1.0.1.0  00:14:01
JServer JAVA Virtual Machine      VALID      11.1.0.1.0  00:11:08
Oracle Workspace Manager          VALID      11.1.0.0.0  00:00:40
OLAP Analytic Workspace           VALID      11.1.0.1.0  00:00:25
OLAP Catalog .                    VALID      11.1.0.1.0  00:00:50
Oracle OLAP API                   VALID      11.1.0.1.0  00:00:31
Oracle Enterprise Manager         VALID      11.1.0.1.0  00:08:06
Oracle XDK                        VALID      11.1.0.1.0  00:00:58
Oracle Text                       VALID      11.1.0.1.0  00:00:45
Oracle XML Database               VALID      11.1.0.1.0  00:09:29
Oracle Database Java Packages     VALID      11.1.0.1.0  00:01:00
Oracle interMedia                 VALID      11.1.0.1.0  00:16:11
Spatial                           VALID      11.1.0.1.0  00:04:43
Oracle Expression Filter          VALID      11.1.0.1.0  00:00:13
Oracle Rules Manager              VALID      11.1.0.1.0  00:00:11
.
Total Upgrade Time: 01:13:55

PL/SQL procedure successfully completed.

SQL>
```

The `utlu111s.sql` script, also called the Post-Upgrade Status Tool, shows the status of various database components following the upgrade. In the example shown here, all database components have migrated successfully, with a `VALID` status. If you see any errors during the upgrade process, you can rerun the `catupgrd.sql` script as many times as necessary. Most of the time, any errors are because of problems such as an inadequate shared memory or lack of room in a tablespace. If you see a status of `INVALID` for any of the database components, running `utlrp.sql` as shown in the next step will most likely change the status to `VALID`.

13. Execute the `utlrp.sql` script to recompile the stored PL/SQL and Java code:

```
SQL> @utlrp.sql
```

You can run the following script to check there aren't any invalid objects left in the newly upgraded database:

```
SQL> select count(*) from dba_invalid_objects;
```

Once you ascertain that there aren't any invalid objects, you have finished upgrading your database to the Oracle Database 11*g* release. If the upgrade should fail for some reason, you must restore the pre-upgrade database from the backup you made before starting the upgrade.

If you have to rerun the upgrade, you can do so by shutting down the database and restarting the database in the upgrade mode (`startup upgrade`), as shown in step 7. You must run the `catupgrd.sql` script after this and check the status of the components in the end by running the `utlu111s.sql` script. If you want to cancel the upgrade altogether because of any reason, just restore the database from the backup made at the outset of the upgrade process.

At this point, you must reset any passwords you want (see the following note) and set thresholds for tablespace alerts, because the upgraded database disables all tablespace alerts (the alert threshold is set to `null`).

■ **Note** In previous releases of the Oracle databases, passwords weren't case-sensitive. If you want to take advantage of the new Oracle Database 11*g* password case-sensitivity feature, you must reset the user passwords after the database upgrade is completed. Use the `alter user` statement for each user in the database to reset their passwords to a case-sensitive format. See Chapter 5 for a discussion of the new case-sensitive passwords.

Upgrading with the DBUA

You can use the DBUA to effortlessly upgrade a pre-Oracle Database 11*g* database to the 11*g* release. The DBUA now uses the new pre-upgrade script for estimating disk space, providing warnings, and so on. The DBUA works pretty similarly to the way it did in the Oracle Database 10*g* release. You just have an additional screen now that asks you to specify the diagnostic directory location. If you specify that you're performing a database upgrade after installing the Oracle Database 11*g* software, the DBUA starts up automatically after you install Oracle Database 11*g*. Otherwise, you can manually invoke DBUA when you want to upgrade a database to the Oracle Database 11*g* release.

For the past several major releases, Oracle has committed a huge amount of time and resources to the fine-tuning of the DBUA. The DBUA has become a reliable option and the “one-stop shop” for simplified upgrades, especially if you have lot of special database options installed in the database such as Oracle Text, Java, Advanced Replication, and Streams. The DBUA lets you upgrade both the database instance and the ASM instance simultaneously, whereas with a manual method, you must upgrade the ASM separately from the database.

Performance Testing the Upgrade

Whether you use the manual method or the DBUA to upgrade, you can use several new Oracle Database change management features to test the performance of the upgrade to the new release:

- *SQL Performance Analyzer (SPA)*: You can use the SQL Performance Analyzer to predict the impact of any change, such as an upgrade to a new release, on the SQL workload as captured by a SQL tuning set. We discuss the SPA later in this chapter.
- *Database Replay*: You can use the Database Replay feature to test the impact of the database upgrade on the production workload by capturing and replaying the production workload on a test system before you actually perform the upgrade on your production database. We discuss the Database Replay feature later in this chapter.
- *SQL Plan Management*: The SQL Plan Management feature relies on the use of SQL plan baselines, which represent efficient execution plans. When you adopt SQL Plan Management pursuant to a database upgrade, only those SQL plans are used that don’t result in a performance regression. We discuss the SPA in detail in Chapter 3.

Of course, besides the new change management tools described here, you can also use traditional volume and load testing to predict the impact of the database upgrade under real-life conditions.

Downgrading After an Upgrade to 11*g*

You can downgrade to the major release from which you upgraded to 11*g*. For example, if you upgraded from 10.1 to 11.1, you can downgrade only to 10.1, but not to the 10.2 release. Once you upgrade to Oracle Database 11*g* Release 1(11.1), you can downgrade back to release 10.2 or 10.1, whichever you’ve upgraded from. If your original Oracle database release number is lower than 10.2.0.4, however, you must install the latest patch set for release 10.1.

Here are the steps for downgrading from the Oracle Database 11*g* Release 1(11.1) (you must be in the 11.1 environment before starting the downgrade process):

1. Start the instance in downgrade mode:

   ```
   SQL> startup downgrade pfile=spfileorcl11.ora
   ```

2. Spool the results of the downgrade process:

   ```
   SQL> spool downgrade.log
   ```

3. Execute the downgrade script, `catdwgrd.sql`, located in the `$ORACLE_HOME/rdbms/admin` directory.

   ```
   SQL> @catdwgrd.sql
   ```

Once all components are successfully downgraded, the database downgrade will be completed.

4. Turn off spooling once downgrading of the database is completed:

```
SQL> spool off
```

If you find any errors in the spool file downgrade log, you can rerun the downgrade script `catdwgrd.sql` as many times as necessary.

5. Shut down the instance:

```
SQL> shutdown immediate;
```

6. Restart the database after changing all environment variables to point to the correct directories for the release to which you're downgrading.

Rolling Upgrade Enhancements

A *rolling upgrade* lets you upgrade a database to a new server release or apply patches, all while the database is online. Obviously, a rolling upgrade lets you avoid downtime, because it means you can apply patches to a running, "live" database. In Oracle Database 10*g*, Oracle offered rolling upgrades to Oracle Data Guard (and logical standby databases) and Oracle Streams. In addition, you could also apply individual patches online for ORAC databases. In Oracle Database 11*g*, Oracle takes the rolling upgrade capability further by extending it to ASM-based databases. You can now perform a live upgrade of any ASM software from the Oracle Database 11.1 release to a newer release (11.2 and so on).

The rolling upgrade of ASM databases means you can continue to use all the features of a clustered ASM environment, even while you are performing a rolling upgrade of one of the nodes in the cluster. Please see Chapter 9 for more on rolling upgrades in Oracle Database 11g.

You can now also perform an Oracle Clusterware rolling upgrade, but you're limited to applying patch set upgrades. During the patch set upgrade, all instances of the Oracle RAC installation except the instance being upgraded continue to be available to users, thus reducing the total downtime during the upgrade to the new patch set release. We discuss the Oracle Clusterware upgrade in the next section.

■ **Note** Oracle Database 11*g* uses parallelism and delays the compilation of PL/SQL objects during the database upgrade process, making the upgrade to the 11*g* release faster than earlier upgrades in earlier releases.

Oracle Clusterware Upgrade

Oracle Clusterware is the heartbeat of the RAC ecosystem. Oracle continues to make the upgrade procedures for Clusterware simpler with every release. The upgrade procedures for 10*g* to 11*g* is seamlessly straightforward. If you are familiar with the upgrade process from 10*g* Release 2 to 10*g* Release 2 patch set 3, you'll find they are identical to the upgrade procedures here.

The Clusterware upgrade is done in place. Unlike the ASM or database upgrade process where you install the binaries into a separate directory and run the DBUA utility, the Clusterware upgrade occurs in the source Clusterware's `ORACLE_HOME`. Since you are running RAC for high-availability considerations, you should perform rolling upgrades of the Clusterware stack. The following sections provide detailed instructions with screen shots of the Clusterware upgrade procedures from 10g Release 2 to 11g. Furthermore, the upgrade is performed in a rolling upgrade fashion.

Preparation for Upgrade

Before you can run the installation and upgrade your clusterware, you must first prepare your environment. First, the clusterware should be shut down completely. Next, you must run the supplied `preupdate.sh` script to reset the file permissions. This script attempts to shut down Clusterware also, but the recommendation is to perform a clean shutdown manually prior to this step.

You can comment out the last three lines of `/etc/inittab` so that the CRS daemons will not try to restart:

```
#h1:35:respawn:/etc/init.d/init.evmd run >/dev/null 2>&1 </dev/null
#h2:35:respawn:/etc/init.d/init.cssd fatal >/dev/null 2>&1 </dev/null
#h3:35:respawn:/etc/init.d/init.crsd run >/dev/null 2>&1 </dev/null
```

Next, you will tell the kernel to reread its `inittab` by issuing the `init q` command:

```
 [root@rac1 upgrade]# init q
```

Note Please exercise caution when you issue the `init q` command. The Q key is right below the 1 key, and if you are a fast and clumsy typist (like one of the authors), you can accidentally initiate an `init 1`, which tells the operating system to boot in single-user mode.

Now, you need to manually stop all the CRS background processes:

```
[root@rac1 upgrade]# /etc/init.d/init.crs stop
Shutting down Oracle Cluster Ready Services (CRS):
Stopping resources. This could take several minutes.
Error while stopping resources. Possible cause: CRSD is down.
Shutdown has begun. The daemons should exit soon.
```

Once you receive confirmation that the shutdown sequence has begun, you can validate that the processes are down using this script:

```
[root@rac1 upgrade]# ps -ef |grep -v grep| egrep -i "crs|css"
```

There should be nothing returned from this script.

Execute the Pre-update Script

In the upgrade directory from the CD software, run the `preupdate.sh` script as root. The upgrade directory is below the `clusterware` directory.

```
cd /home/oracle/software/11g/clusterware/upgrade
[root@rac1 upgrade]# ./preupdate.sh -crshome /apps/oracle/product/
CRS -crsuser oracle
Stopping resources. This could take several minutes.
Error while stopping resources. Possible cause: CRSD is down.
Oracle CRS stack is down now.
```

You will notice that the CRS shutdown attempt is made. Also, behind the scenes, it is changing the file and directory permissions to Oracle so that you can perform an upgrade. If this script is not executed prior to the upgrade, you will receive an error that resembles this:

```
Checking Cluster Synchronization Services (CSS) status ...
Actual Result: CSS Stack is running on the following nodes : rac1.
Check complete. The overall result of this check is: Failed <<<<
Problem: The cluster is not in a state suitable for upgrade.  One or more of the
following conditions have not been met: The Cluster Synchronization Service
(CSS) is running on one or more nodes in the cluster and/or appropriate
directories within the Oracle Home are not writable.
Recommendation: To upgrade Oracle 10g Release Clusterware, you must shutdown
CSS on all nodes, and ensure that specific directories within the
 Oracle 10g Release Clusterware Home are writable on all nodes.  In the upgrade/
 directory at the root of the Oracle Clusterware 11g Release 1 media
 there is a shell script 'preupdate.sh' that should be run as root on each node
 that you wish to upgrade prior to launching the installer.  This script will
 shutdown the Oracle 10g Release Clusterware stack on that node and prepare the
Oracle 10g Release Clusterware Home such that the upgrade can
take place.  The script can be invoked using the following command: 'preupdate.sh
-crshome <Oracle 10g Release Clusterware Home location> -crsuser
<Oracle 10g Release Clusterware user name>'.
```

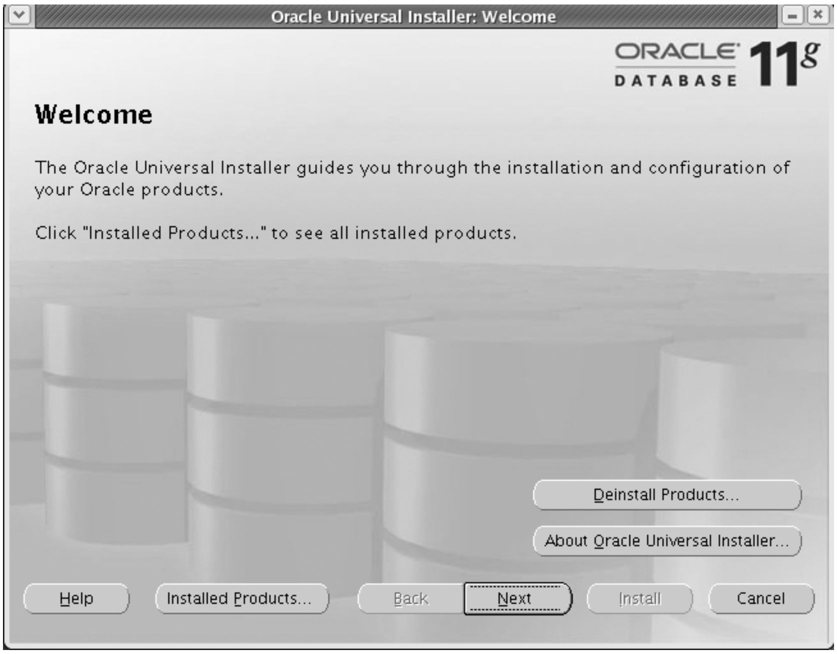
Install and Upgrade Using runInstaller

Now let's proceed with the actual upgrade. The upgrade process starts by executing the `runInstaller` executable from the CD software:

```
$./runInstaller
```

This will take you to the Welcome screen, as depicted in Figure 1-5.

When you click Next, it will take you to the Specify Home Details page where you can confirm the source Clusterware home that needs to be upgraded to Oracle 11*g* Clusterware, as shown in Figure 1-6.

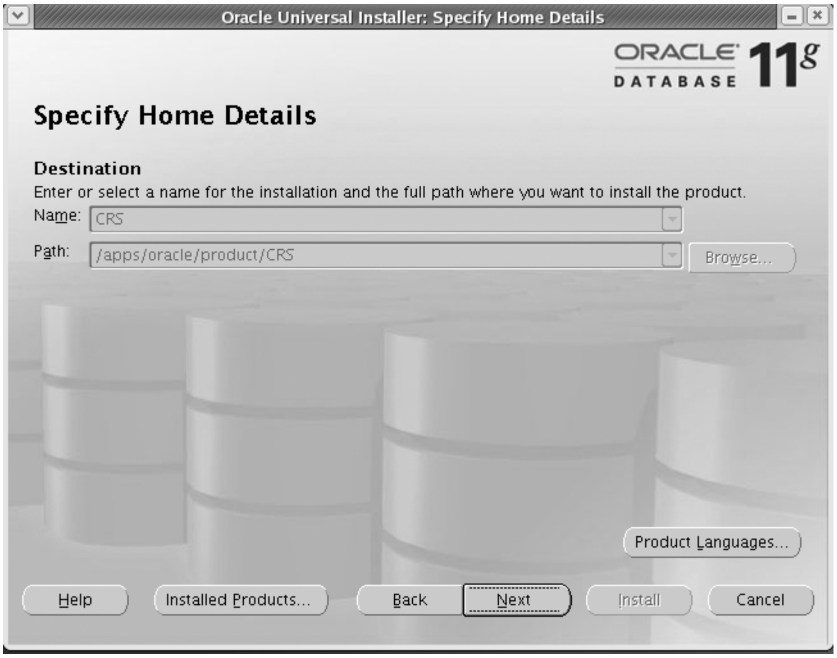

Oracle Universal Installer: Welcome

Welcome

The Oracle Universal Installer guides you through the installation and configuration of your Oracle products.

Click "Installed Products..." to see all installed products.

Figure 1-5. *Clusterware upgrade, Welcome screen*


Oracle Universal Installer: Specify Home Details

Specify Home Details

Destination

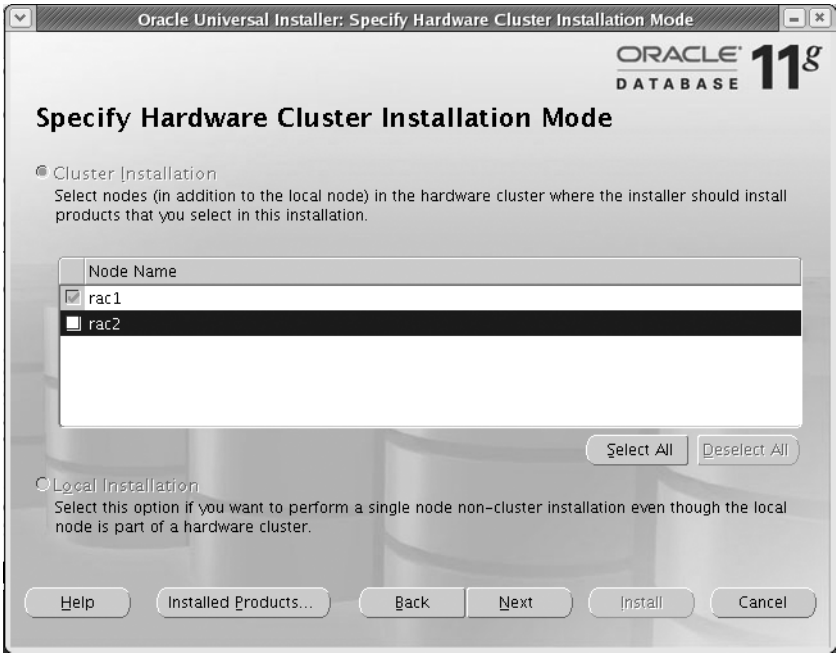
Enter or select a name for the installation and the full path where you want to install the product.

Name: CRS

Path: /apps/oracle/product/CRS

Figure 1-6. *Clusterware upgrade, Specify Home Details page*

When you click Next, you will be directed to the screen shown in Figure 1-7 to select the nodes you want to upgrade. For rolling upgrades, you should select only the node you are upgrading.


Oracle Universal Installer: Specify Hardware Cluster Installation Mode

ORACLE DATABASE 11g

Specify Hardware Cluster Installation Mode

● Cluster Installation

Select nodes (in addition to the local node) in the hardware cluster where the installer should install products that you select in this installation.

	Node Name
✓	rac1
☐	rac2

○ Local Installation

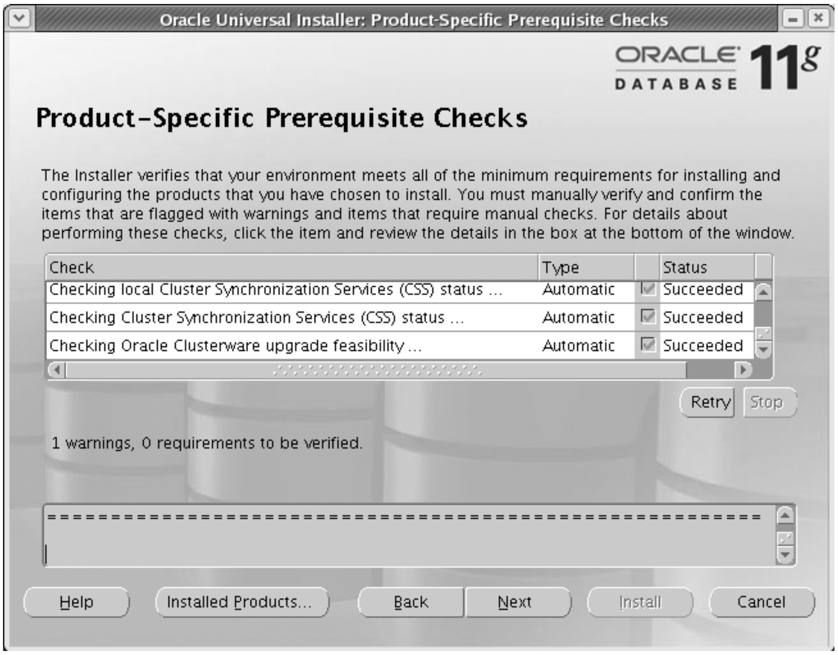
Select this option if you want to perform a single node non-cluster installation even though the local node is part of a hardware cluster.

Figure 1-7. *Clusterware upgrade, Specify Hardware Cluster Installation Mode page*

Note Optionally, prior to running the clusterware upgrade, you can single out the specific node by using the `updateNodeList` option: `./runInstaller -updateNodeList "CLUSTER_NODES=rac1" -local ORACLE_HOME=/apps/oracle/product/CRS`.

Once you pick the node(s), you can continue the upgrade with the system prerequisite checks, as shown in Figure 1-8. In this section, it is critical you make sure the overall status is Succeeded!

Once all the patches, swap space, memory, networks, CSS, OCR, voting disk, and so on are validated, you can continue. The upgrade process will take you to the Summary page, as shown in Figure 1-9.


Oracle Universal Installer: Product-Specific Prerequisite Checks

Product-Specific Prerequisite Checks

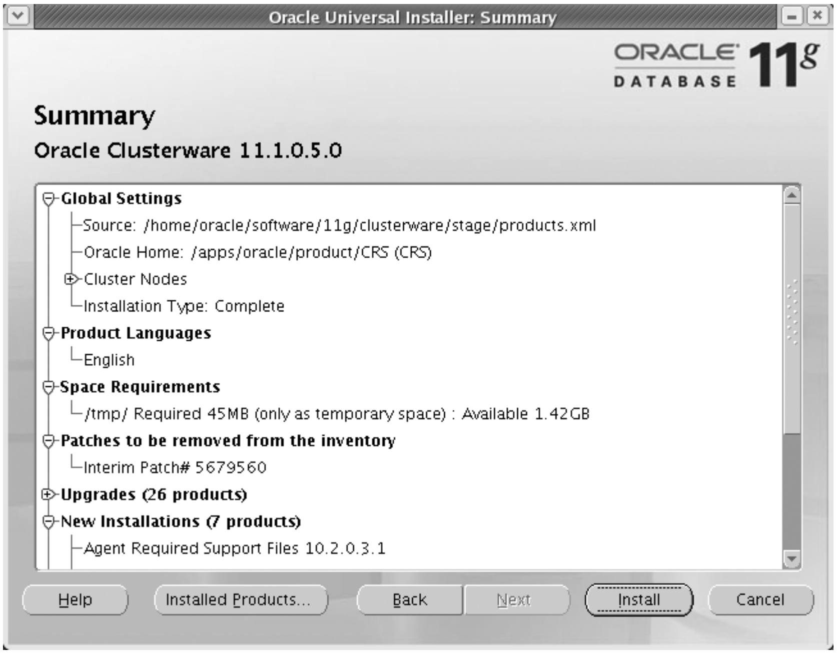
The Installer verifies that your environment meets all of the minimum requirements for installing and configuring the products that you have chosen to install. You must manually verify and confirm the items that are flagged with warnings and items that require manual checks. For details about performing these checks, click the item and review the details in the box at the bottom of the window.

Check	Type		Status
Checking local Cluster Synchronization Services (CSS) status ...	Automatic	✓	Succeeded
Checking Cluster Synchronization Services (CSS) status ...	Automatic	✓	Succeeded
Checking Oracle Clusterware upgrade feasibility ...	Automatic	✓	Succeeded

1 warnings, 0 requirements to be verified.

```
===========================================================
```

Figure 1-8. *Clusterware upgrade, Product-Specific Prerequisite Checks page*


Oracle Universal Installer: Summary

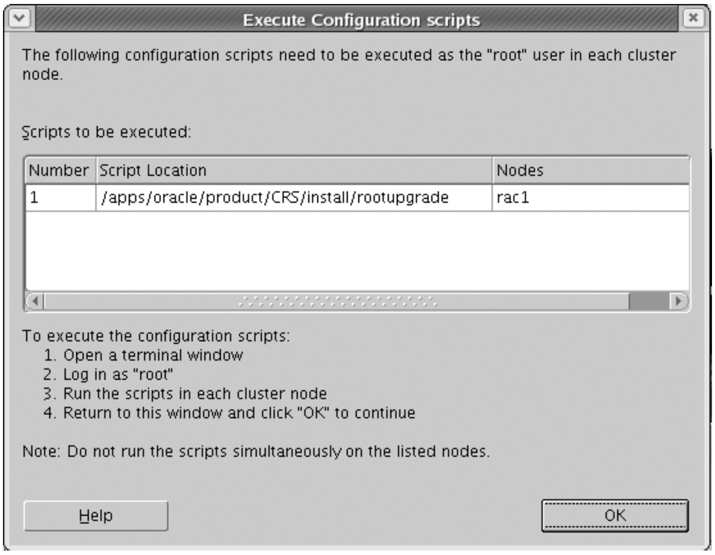
Summary

Oracle Clusterware 11.1.0.5.0

```
Global Settings
  Source: /home/oracle/software/11g/clusterware/stage/products.xml
  Oracle Home: /apps/oracle/product/CRS (CRS)
  Cluster Nodes
  Installation Type: Complete
Product Languages
  English
Space Requirements
  /tmp/ Required 45MB (only as temporary space) : Available 1.42GB
Patches to be removed from the inventory
  Interim Patch# 5679560
Upgrades (26 products)
New Installations (7 products)
  Agent Required Support Files 10.2.0.3.1
```

Figure 1-9. *Clusterware upgrade, Summary page*

Once the installation is complete, you will be directed to the dialog box shown in Figure 1-10 to run the `rootupgrade` script from the `$ORA_CRS_HOME/install` directory.



The following configuration scripts need to be executed as the "root" user in each cluster node.

Scripts to be executed:

Number	Script Location	Nodes
1	/apps/oracle/product/CRS/install/rootupgrade	rac1

To execute the configuration scripts:
1. Open a terminal window
2. Log in as "root"
3. Run the scripts in each cluster node
4. Return to this window and click "OK" to continue

Note: Do not run the scripts simultaneously on the listed nodes.

Figure 1-10. *Clusterware upgrade, running -rootupgrade*

As prompted, execute the `rootupgrade` script. The script will yield the following output:

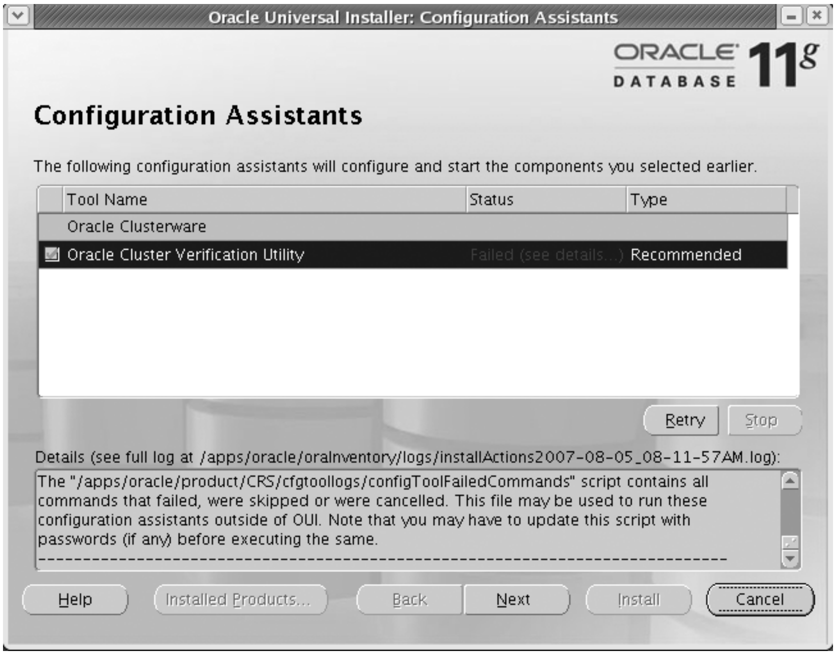
```
[root@rac1 install]# ./rootupgrade
Checking to see if Oracle CRS stack is already up...

copying ONS config file to 11.1 CRS home
/bin/cp: `/apps/oracle/product/CRS/opmn/conf/ons.config' and `/apps/oracle/p
/apps/oracle/product/CRS/opmn/conf/ons.config was copied successfully to /ap
WARNING: directory '/apps/oracle/product' is not owned by root
WARNING: directory '/apps/oracle' is not owned by root
WARNING: directory '/apps' is not owned by root
Oracle Cluster Registry configuration upgraded successfully
Adding daemons to inittab
  ____    ____          ____        ____
 /    \ |    \   /\    /     |     /
|      ||    /  /  \  |      |    |
|      ||---    ----  |      |    |---
|      ||   \  /    \ |      |    |
 \____/ |    \/      \ \____ |____ \____

Attempting to start Oracle Clusterware stack
```

Note You may have to reboot the box if the clusterware does not autostart after the upgrade.

Your clusterware may not autostart and may fail with the cluster verify utility (`cluvfy`). This is an optional requirement, and you can ignore the failure. You should confirm that there is nothing wrong if you get this error. Figure 1-11 shows the Configuration Assistants page, with the Failed status for the Oracle Cluster Verification Utility.



The following configuration assistants will configure and start the components you selected earlier.

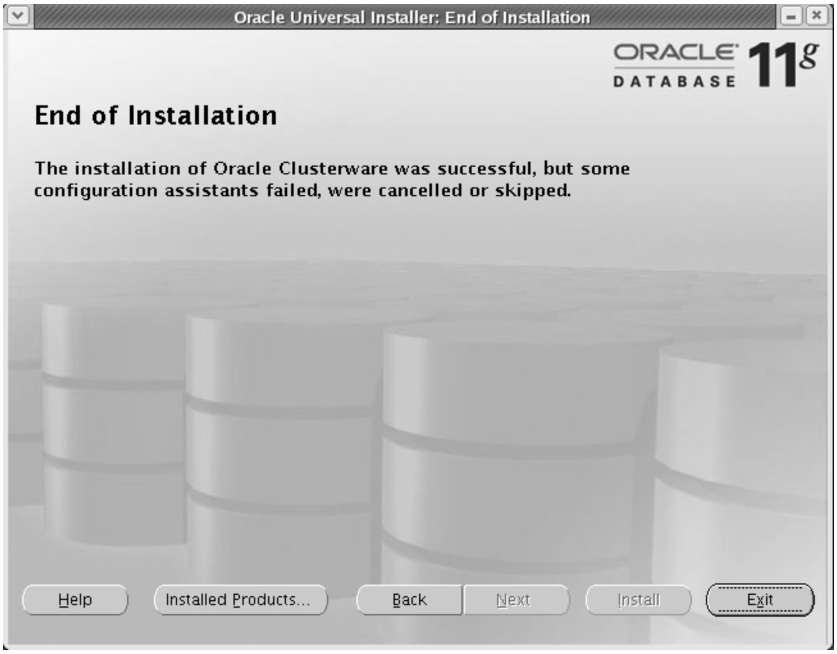
Tool Name	Status	Type
Oracle Clusterware		
Oracle Cluster Verification Utility	Failed (see details...)	Recommended

Details (see full log at /apps/oracle/oraInventory/logs/installActions2007-08-05_08-11-57AM.log):

```
The "/apps/oracle/product/CRS/cfgtoollogs/configToolFailedCommands" script contains all
commands that failed, were skipped or were cancelled. This file may be used to run these
configuration assistants outside of OUI. Note that you may have to update this script with
passwords (if any) before executing the same.
---------------------------------------------------------------------------
```

Figure 1-11. *Clusterware upgrade, Configuration Assistants page*

A simple bounce of the server can resolve this issue. The Next button will take you to the final screen of the upgrade, as shown in Figure 1-12.



End of Installation

The installation of Oracle Clusterware was successful, but some configuration assistants failed, were cancelled or skipped.

Figure 1-12. *Clusterware upgrade, End of Installation page*

Once the successful upgrade is complete and the CRS stack is back up and running, you can validate your rolling clusterware upgrade:

```
crs > crsctl query crs softwareversion
Oracle Clusterware version on node [rac1] is [11.1.0.5.0]
rac1.dbaexpert.com:/home/oracle
crs > crsctl query crs activeversion
Oracle Clusterware active version on the cluster is [10.2.0.3.0]
```

Upgrade Remaining Nodes of the RAC Environment

Repeat the same steps we showed here, starting from the “Prepare for Upgrade” section for every node in the cluster. Once the Clusterware upgrade is complete across all RAC nodes in the cluster, you can confirm that the Oracle Clusterware active version is the same as the software version:

```
crs > crsctl query crs activeversion
Oracle Clusterware active version on the cluster is [11.1.0.5.0]
rac1.dbaexpert.com:/apps/oracle/general/sh
crs > crsctl query crs softwareversion
Oracle Clusterware version on node [rac1] is [11.1.0.5.0]
```

The output shows the Oracle Clusterware version for each RAC node.

Real Application Testing

Adopting new technologies is too often a double-edged sword, making you more efficient and thus providing a competitive advantage while simultaneously introducing uncertainty and potential instability into key production systems. Change assurance, which involves making sure major changes such as new software releases and database upgrades don't negatively impact performance, has always been a prime concern for Oracle application developers and database administrators. Even if you can simulate real production workloads, the effort is just that—a simulation, not the real deal. In a world that is technologically advancing at a mind-numbing pace, you need to know which of the technologies have the potential to benefit you; therefore, you need to perform real testing with real data in real conditions.

Oracle Database 11*g* places considerable emphasis on the proactive testing of changes by making change assurance one of the cornerstones of the new release. It does this through the Real Application Testing feature; this feature encompasses two components, Database Replay and the SQL Performance Analyzer, that dramatically reduce the risks inherent in adopting changes by offering you a realistic method of testing changes using real-life workloads. The tools unearth problems and give you the opportunity to fix them before you actually introduce the changes into your production systems. Here's a brief summary of the two key components of Real Application Testing in Oracle Database 11*g*:

- *Database Replay*: You can use the Database Replay feature to test the impact of the database upgrade on the production workload by capturing and replaying the production workload on a test system before you actually perform the upgrade on your production database. Using the Database Replay reports from a test server, you can fix potential problems before they occur on the production database. We discuss the Database Replay feature later in this chapter.

- *SQL Performance Analyzer (SPA)*: You can use the SQL Performance Analyzer to predict the impact of any change, such as an upgrade to a new release, on the SQL workload as captured by a SQL tuning set. By knowing ahead of the actual upgrade about any adverse impact on performance and the root cause for it, you can prevent it from actually occurring in a production database after a database upgrade. We discuss the SPA later in this chapter.

In addition to the Database Replay and the SQL Performance Analyzer features, there's also a third new feature pertaining to change management, called SQL Plan Management, which replaces the stored outlines feature in earlier releases. The SQL Plan Management feature relies on the use of SQL plan baselines, which represent efficient execution plans. When you adopt SQL Plan Management pursuant to a database upgrade, only those SQL plans are used that don't result in a performance regression. We discuss the SPA in detail in Chapter 3.We discuss Database Replay and the SQL Performance Analyzer in detail in the following sections.

Database Replay

One of the major problems you face during an Oracle server software upgrade process or an application upgrade is the difficulty in simulating the actual production workload on the test databases. This is also true when you're moving to a totally new database configuration, say from a regular operating system file system to automatic storage management. Even if you use sophisticated testing suite software, it's not easy to accurately reproduce the true workload of a production database. Consequently, you are forced to test in an unrealistic setting and take your chances when you move to the new release of the server software or the application. It's not at all uncommon for DBAs and developers to bemoan the fact that the testing folks couldn't adequately "stress test" the changes before they were approved for the switch to production.

■ **Note** You can capture the workload for a single-instance Oracle database and replay it over a test system running in Oracle RAC configuration to get an idea about the performance improvement.

The Database Replay feature provides a solution to the vexing problem of reproducing production conditions in the testing and migration environments. By making it significantly easier to test potential database changes, Oracle Database 11*g* lowers the cost of database upgrades as well as other major changes such as operating system and storage system upgrades. Testing that usually would take months when done by scripts and traditional load simulation tools can be done at dazzlingly fast speeds now with the Database Replay feature. Database Replay lets you capture the actual workload on a production system and analyze it the same way by replaying it on a test system. The goal is to replicate the production environment *in toto* on a test system. Since the characteristics of the original workload such as concurrency and timing are maintained during the replay, you're essentially working with the same type of resource contention and other characteristics. This lets you easily identify any negatives that'll be potentially introduced by making the application, system, or software changes. The goal is to make sure the change you're making, such as a database upgrade, gets you only desirable results. Note that your test system must be running on the same or a newer version of the Oracle Database compared to the production system.

The key fact you must understand here is that Database Replay captures only the workload at the database level and ignores all client, application, and middle-tier interactions. The replay will capture the impact of any system changes that affect the database, such as an upgrade of the database itself, an upgrade of the operating system, or a switch to a new disk storage system. The production database is backed up and restored on a test system with identical configuration and environment. Your goal is to replicate the production system with the same application state as the original.

You can use Database Replay for testing the following types of changes:

- Database upgrades
- Operating system upgrades
- Storage system changes
- Configuration changes such as switching to an Oracle Real Application Cluster

Using Database Replay to replay a production workload consists of four steps:

1. Workload Capture records the production database workload.
2. Workload Preprocessing makes the captured workload replayable by converting it into replay files.

3. Workload Replay replays the production workload in the test database with actual timings, following the changes you want to test.

4. Analysis and Reporting deals with error reporting as well as reports on data divergence and performance divergence between the production and test environments. You can also enlist the ADDM for a deeper performance analysis.

Oracle provides an interface called the Workload Replay Client to let you interface with the Workload Replay feature using the command line. To access the Workload Replay Client, type `wrc` at the operating system command line, as shown here:

```
$ wrc
Workload Replay Client: Release 11.1.0.6.0 - Production on Sat Aug 25 21:45:01 2007
Copyright (c) 1982, 2007, Oracle.  All rights reserved.

FORMAT:
=======
 wrc [user/password[@server]] [MODE=mode-value] KEYWORD=value

Example:
========
   wrc  REPLAYDIR=.
   wrc  scott/tiger@myserver REPLAYDIR=.
   wrc  MODE=calibrate REPLAYDIR=./capture
 The default privileged user is: SYSTEM

Mode:
=====
wrc can work in different modes to provide additional functionalities.
The default MODE is REPLAY.

Mode        Description
----------------------------------------------------------------
REPLAY      Default mode that replays the workload in REPLAYDIR
CALIBRATE   Estimate the number of replay clients and CPUs
            needed to replay the workload in REPLAYDIR.
LIST_HOSTS  List all the hosts that participated in the capture
            or replay.

Options (listed by mode):
=========================

MODE=REPLAY (default)
---------------------

Keyword     Description
----------------------------------------------------------------
USERID      username (Default: SYSTEM)
```

```
PASSWORD    password (Default: default password of SYSTEM)
SERVER      server connection identifier (Default: empty string)
REPLAYDIR   replay directory (Default:.)
WORKDIR     work directory (Default:.)
DEBUG       FILES, STDOUT, NONE  (Default: NONE)
            FILES  (write debug data to files at WORKDIR)
            STDOUT (print debug data to stdout)
            BOTH   (print to both files and stdout)
            NONE   (no debug data)
CONNECTION_OVERRIDE  TRUE, FALSE (Default: FALSE)
            TRUE   All replay threads connect using SERVER,
                   settings in DBA_WORKLOAD_CONNECTION_MAP will be ignored!
            FALSE  Use settings from DBA_WORKLOAD_CONNECTION_MAP
SERIALIZE_CONNECTS  TRUE, FALSE (Default: FALSE)
            TRUE   All the replay threads will connect to
                   the database in a serial fashion one after
                   another. This setting is recommended when
                   the replay clients use the bequeath protocol
                   to communicate to the database server.
            FALSE  Replay threads will connect to the database
                   in a concurrent fashion mimicking the original
                   capture behavior.

MODE=CALIBRATE
,,,

MODE=LIST_HOSTS
. . .
$
```

The first thing to note is that `wrc` can work in different modes to perform various tasks. The default mode is `REPLAY`, which we'll use to replay the workload that we capture. However, it may be necessary to run `wrc` in `calibrate` mode first to estimate the number of replay clients and servers (CPUs) needed to replay the captured workload. By default, the `connection_override` parameter is set to `false`, meaning that all replay threads connect using the server settings in the `DBA_WORKLOAD_CONNECTION_MAP` table.

Each thread of the multithreaded `wrc` (replay client) program submits a workload for a single session for replaying. You must first connect with the replay client to the test system before starting the workload replay, as we show later in this chapter. The Workload Replay Client will connect to the replay system and send database requests that are part of the captured database workload. You can manipulate several options of the `wrc` to control replay behavior, including scaling the login time and think time up and down. You can put these options to use for load and stress testing the production system on the test database.

Types of Data Captured in the Workload

The workload you capture from the production system consists of the following types of SQL calls:

- Data Manipulation Language (DML) statements
- Data Definition Language (DDL) statements
- Session control calls such as `alter session`
- System control calls such as `alter system`

Any background activities and scheduled jobs aren't part of the data capture. Client requests such as a direct path load of data from external files using SQL*Loader, database links, external tables, Oracle Streams, non-SQL-based object access, and distributed transactions are the types of data that *aren't* captured as part of the database workload.

Oracle recommends you use the Oracle Enterprise Manager (Database Control or Grid Control) to capture and replay the workload data. However, you can also use APIs available through the new Oracle packages `DBMS_WORKLOAD_CAPTURE` and `DBMS_WORKLOAD_REPLAY` to perform the workload capture and replay that are at the heart of the Database Replay feature. You use the procedures from the `DBMS_WORKLOAD_CAPTURE` package to capture the production workload and the procedures from the `DBMS_WORKLOAD_REPLAY` package to replay the captured production workload on the test system. You must have either the `dba` privilege or the `execute_catalog_role` privilege to use the two packages. In this chapter, we show you how to use the various procedures of these two key new Oracle packages to work with the Database Replay feature.

Before you can capture the workload, you must create a directory that'll hold the captured workload data. You must also back up the database before the capture process begins so you can replay the captured data on a system with an identical application state as that of the source database. You can use filters (with the `add_filter` procedure) to avoid capturing some user sessions such as the Oracle Enterprise Manager sessions or DBA actions, which aren't deemed part of the database workload for the purposes of change management.

■ **Tip** Add the workload directory as part of your database OFA file system standards: `mkdir -p $ORACLE_BASE/admin/$ORACLE_SID/workload`.

We briefly describe how to perform each of the four key steps of Database Replay in the following sections.

Capturing the Workload

During the workload capture phase, Database Replay will capture all requests made by external clients to the Oracle database. These external client requests primarily include calls made by applications such as SQL*Plus and middle-tier components. Information pertaining to calls made by the external clients, such as SQL text and bind values, are collected and stored in what are known as *capture files* in a location chosen by you (in binary format).

Before you can start capturing the workload, you must do the following:

1. Back up the production database so you can restore it on the test system later for replaying the workload. The goal is to reproduce the application state of the production system by restoring the database before you replay the captured workload. You can use RMAN, a point-in-time recovery, Data Pump Export/Import, or a snapshot standby to back up the production database before starting the data capture.
2. Decide whether you want to restart the production database. You don't have to restart the production database before starting the workload capture, but note that any transactions that are underway when you begin the workload capture may not be captured.
3. If you decide to restart the production database, you must do so in restricted mode after logging in as the user `sys`. Once you start the capture process as shown here, you can open the database for all user sessions.
4. Create a directory with enough space to store the captured workload. Use the `create directory` SQL statement to create the directory object.
5. Determine whether some of the user sessions, such as DBA sessions, don't need to be captured as part of the workload. You can use *workload filters* to eliminate these sessions from the workload, as shown in the following example, which uses the `add_filter` procedure to filter parts of the workload from the data capture:

   ```
   SQL> exec dbms_workload_capture.add_filter (fname => 'my_filter1',-
      > fattribute => ‘user’, fvalue => 'prod_dba');
   ```

You can remove any filters you create by executing the `delete_filter` procedure.

Use the `start_capture` procedure of the `DBMS_WORKLOAD` package to capture the database workload during a representative period of operation. Here's a simple example showing how to capture a database workload:

```
SQL> exec dbms_workload_capture.start_capture(name => -
   > ' aug20_peak', dir=> 'test_dir',duration=> 240);
```

The `name` parameter specifies the name of this workload capture process. The `dir` parameter refers to the directory object pointing to the directory you created for storing the captured workload data. Although you can stop the workload capture with the procedure `stop_capture`, here we use the `duration` parameter instead to limit the workload capture to 4 hours (240 minutes). If you choose to stop the workload capture before the end of the 4-hour duration, you can do so in the following way:

```
SQL> execute dbms_workload_capture.finish_capture ();
```

The execution of the `finish_capture` procedure ends the workload capture. It's a good idea at this point to generate a workload capture report to validate your data capture. Validation includes ascertaining that you did capture the correct workload and didn't miss any critical part of the production workload by accident. Of course, you must also make sure you can replay the captured data on the test server later. Execute the `DBMS_WORKLOAD_CAPTURE.GET_CAPTURE_INFO` procedure to generate a workload capture report, as shown here:

```
declare
   capture_id    number;
   capture_rpt   clob;
begin
   capture_id  :=  dbms_workload_capture.get_capture_info(dir => 'test_dir');
   capture_rpt  := dbms_workload_capture.report (capture_id => capture_id,
                   format => dbms_workload_capture.type_text);
end;
```

The procedure shown here produces a report in a text format. The report shows you a profile of both the workload that was captured and a profile of the workload that was excluded by using filters or that couldn't be captured because of version limitations. You also get statistics such as the number of logins and transactions the workload captured. Although the workload capture report is useful in many ways, it's the workload replay report that you'll see later that's really of more concern to you, because it tells you about the data and performance differences between the original system and the replay system.

Once you finish capturing the production system workload, it's time to prepare the workload for the replay on the test system, as shown in the next section.

Preprocessing the Data

Before you can replay the production data on a test server, you must first preprocess the data and initialize it. Preprocessing the workload entails converting the data into replay files, as well as creating the metadata necessary for replaying those files. Of course, you must preprocess the workload on a system that's running the same version of the Oracle Database as the one from which you captured the database workload.

Preprocessing essentially involves creating *replay files* from the processed data, which you can then replay on the test system. You process the workload in the test database where you're planning the replay. You can preprocess the data on the production system itself, but since preprocessing is a resource-intensive process, you're better off doing it on the test system. Processing the workload sets the stage for replaying the workload; it does this by transforming the workload into replay files.

To preprocess the workload data, you must first move the captured workload from the production system to a directory on the test system. So, create a directory object for this directory on the test system beforehand. Use the `process_capture` procedure to process the captured workload data:

```
SQL> exec dbms_workload_replay.process_capture (capture_dir => 'test_dir');
```

The `process_capture` procedure has only a single parameter, `capture_dir`, which points to the directory object holding the captured workload data. Once you preprocess the workload data, you can replay it on the same or a newer release of the Oracle database.

Performing the System Change

Now that you have moved the captured workload from the pre-change production system to the test system, make the system changes that it's your goal to test. For example, if you're planning to test a database upgrade, perform that upgrade now so you can replay the captured workload from the pre-change database to test the database upgrade.

Replaying the Workload

Before you start the workload replay, restore the test database so it reflects the same application state as the production database. You must also ensure that all external linkages and objects such as database links, external tables, directory objects, and URLs present in the production system are present in the test system as well. Oracle recommends you reset the system time on the test replay system to the time when you started the workload capture to avoid invalid data sets when dealing with workloads containing time-sensitive data, as well as to eliminate any potential job-scheduling issues.

You may want to start the database you restored in a restricted mode to avoid accidental changes to data during the workload replay. Replaying the workload requires several steps, which we summarize in the following sections.

Resolving External References

You must resolve all external references such as database links that are part of the captured workload. You must either disable or reconfigure all external references in the workload, if necessary, to make them accessible from the test system. These external references include database links, external tables, directory objects, and URLs.

Initializing the Replay Data

The next step is to *initialize* the replay data. Initializing the data, which you perform through the `initialize_replay` procedure, loads the metadata into tables, which will be used by the workload replay process:

```
SQL> exec dbms_workload_replay.initialize_replay(replay_name =>
     'test_replay',replay_dir => 'test_dir');
```

The `replay_name` parameter refers to the name of the replay. The `replay_dir` parameter points to the directory containing the captured workload that you want to replay on the test system.

Remapping External Connections

Once you initialize the workload data, you can query `dba_workload_connection_map` to view the connection mappings, if any external connections were made by the source (production) database users during the workload capture. You must now remap the connection strings that were utilized during the workload capture so the individual user sessions can connect to the external databases. You remap the external connection strings using the `remap_connection` procedure. The replay clients must connect to the replay system in order to replay the workload. On single-instance systems (not Real Application Clusters), the connection strings between the capture and the replay systems are mapped one-to-one.

Here's how you remap the external connections using the `remap_connection` procedure:

```
SQL> exec dbms_workload_replay.remap_connection (connection_id => 111,
     replay_connection => 'prod1:1521/mydb');
```

The `connection_id` parameter signifies a connection from the workload capture (generated during replay data initialization), and the `replay_connection` parameter (an optional parameter) specifies the new connection string to be used during the workload replay. If you leave the `replay_connection` parameter's value set to `null` (the default value), all replay sessions will connect to the default host, which is determined by the replay client's runtime environment.

■ **Note** Although Database Replay is a new Oracle Database 11*g* feature, Oracle will be porting the `change capture` portion of this feature back to the Oracle Database 10*g* Release 2. You can replay any workload you capture on an Oracle Database 10.2.0.4 release or newer on an Oracle Database 11*g* based database. So if you are running the 10.2 version, you can use the Database Replay feature to check out the impact of an upgrade to the 11.*x* release of the database once the relevant code is ported to Oracle Database 10*g* Release 2.

Starting the Replay Clients

Before you can replay the workload, you must ensure that the replay clients connect to the test database. Each thread of the replay client (started by the executable `wrc`) submits a workload from a session. Once you process the workload data, you must start the replay client(s). The replay client connects to the database and drives the replay of the workload data. Before you can start the replay clients, make sure the replay directory contains the (preprocessed) workload files, the replay clients have access to that directory, and the replay user has the correct credentials to connect.

The `wrc` executable is run by default in `replay` mode. However, first run the `wrc` executable in `calibrate` mode to estimate the number of replay clients and hosts to replay your captured workload. Each replay client can handle multiple user sessions. If, for example, the captured workload was generated by 200 user sessions and a single host can handle only about 120 sessions, you'll need two hosts, with a replay client running on each of those hosts. This means you must install `wrc` on both the hosts. Here's how you run the `wrc` executable in `calibrate` mode:

```
$ wrc system/<system_password> mode=calibrate replay_dir=./test_dir
```

Once you figure out the number of hosts and replay clients you need to replay the actual workload, you then start the necessary replay clients by executing the `wrc` executable in the (default) `replay` mode, as shown here:

```
$ wrc system/<system_password> mode=replay replay_dir=./test_dir
```

By default, the `connection_override` parameter of the `wrc` executable is set to the value of `false`, meaning that all replay threads will simply use the connection mappings in the `DBA_WORKLOAD_CONNECTION_MAP` view to connect.

Preparing the Workload Replay

Before you start the actual workload replay, you must specify various workload replay options. The following are the three replay options you can configure to control the database replay:

- `Mode`: By default, a workload replay is performed in *synchronization* mode. If you think the workload is primarily composed of independent transactions, you must disable synchronization mode explicitly, because data divergence during the replay under these circumstances isn't something to worry about.
- `Connection Time Scale`: You can use the `connect_time_scale` parameter to calibrate the time between the workload capture and the session connect time during the replay. Use this parameter to adjust the number of concurrent users higher or lower during the workload replay.
- `Replay Speed`: To correct for longer time for completion of user calls during the workload replay, use the `think_time_auto_correct` parameter. The `think_time_auto_correct` parameter helps you adjust the concurrency level during the workload replay. To correct for elapsed time between user calls during the replay, use the `think_time_scale` parameter. If the replay is progressing slower than the data capture, by default the think time component of the replay will be automatically reduced (`think_time_auto_correct` is set to `true` by default).

Use the `prepare_replay` procedure to prepare the workload for the replay on the test system, including setting the replay options discussed earlier, as shown here:

```
SQL> dbms_workload_replay.prepare_replay (replay_name =>'replay1',
     replay_dir => 'test_dir', synchronization= FALSE);
```

Only the `synchronization` parameter needs explanation. By setting the `synchronization` parameter to the value of `false` (the default value is `true`), you're saying that the commit order in the workload capture may or may not be preserved during the replay. You can do this in a situation where there are numerous independent transactions that don't have to follow a certain commit order.

Once you've completed all the requirements for the workload replay, it's time to start the workload relay. Use the `start_replay` procedure to do this, as shown here. After making sure you have started at least one `wrc` replay client, issue the following command to replay the workload:

```
SQL> exec dbms_workload_replay.start_replay();
```

The `start_replay` procedure doesn't require any parameters. You can cancel the replay operation using the `cancel_replay` procedure, if you want, as shown here:

```
SQL> exec dbms_workload_replay.cancel_replay();.
```

The `cancel_replay` procedure directs all replay clients (`wrc`) to stop submitting workload from the captured sessions.

At the end of the workload replay, all AWR snapshots that correspond to the replay time period are automatically exported. If the export of the snapshots fails for some reason, you can manually export them by executing the `export_awr` procedure. You can then import these snapshots into the AWR schema owned by the user `sys` by executing the `import_awr` procedure.

Analyzing Workload Capture and Replay

Once you replay the workloads on the test system, you must analyze the data replay by generating a workload replay report. To measure the data and performance differences between the capture and the replay systems as well as a list of any errors during the workload replay, generate the workload replay report, as shown here:

```
declare
      cap_id     number;
      rep_id     number;
      rep_rpt    clob;
begin
    cap_id  :=  dbms_workload_replay.get_replay_info (dir => 'test_dir');
    select max(id)
     into rep_id
     from dba_workload_replays
     where capture_id = cap_id;
     rep_rpt  :=  dbms_workload_replay.report(replay_id  =>  rep_id,
                  format => dbms_workload_replay.type_text);
end;
/
```

The get_ replay_info function returns the history of the workload capture and all the workload replays made based on the specified directory. You can associate the capture_id value returned by this function with the capture_id column of the DBA_WORKLOAD_REPLAYS table. The get_replay_info function also inserts a row into the DBA_WORKLOAD_REPLAY table for every replay you make from the specified replay directory. Note that we chose text as the value for the replay_type parameter. You can also specify HTML and XML as the values for this parameter.

Here is a sample report from the execution of the replay_report procedure:

```
Error Data

(% of total captured actions)
New errors:
7.3%
Not reproduced old errors:
1.0%
Mutated errors:
1.0%

Data Divergence

Percentage of row count diffs:
5.0%
Average magnitude of difference (% of captured):
2.5%
Percentage of diffs because of error (% of diffs):
25.5%

Result checksums were generated for 10% of all actions(% of checksums)
Percentage of failed checksums:
0.0%
```

```
Percentage of failed checksums on same row count:
0.0%

Replay Specific Performance Metrics

Total time deficit (-)/speed up (+):
12 min
Total time of synchronization:
24 min
Average elapsed time difference of calls:
0.2 sec
Total synchronization events:
3284218772
```

Careful examination of the report by looking for both data and performance divergence will give you a good idea about the potential issues subsequent to major changes such as a database upgrade to a new release. Database Replay's reporting tools let you get the following types of information:

- Data divergence, which is reflected in differences in the number of rows returned by queries.
- Errors during the workload replay.
- Performance comparison between the workload capture and workload replay. In this example, the total time deficit is shown to be 12 minutes after the change was implemented. Obviously, things are taking longer to process after the changes were made to the database, and you must, of course, investigate this further.
- Performance statistics as captured by AWR reports.

Note that a performance divergence could be both desirable as well as undesirable, since it's entirely possible that the replay system uses a more recent version of the database, thus providing improved performance. A data divergence is said to occur when the source and the replay systems return different number of rows in response to identical SQL queries or DML statements. Of course, any such data divergence merits serious investigation.

Oracle claims that Database Replay offers significant benefits when compared to third-party tools such as LoadRunner for testing large applications. Database Replay offers the great advantage that it tests virtually 100 percent of the actual production workload, compared to a third-party tool's artificially simulated workload that at best uses less than 10 percent of the actual workload. Database Replay is also much faster when compared to a third-party tool, completing a replay of a complex application in days compared to months. In one study (*Oracle Database 11g: Real Application testing and Manageability Overview*, available at `http://technet.oracle.com`), Oracle claims that it takes only half a month to test an entire e-business suite with Database Replay, compared to a total time of seven-and-a-half months with LoadRunner.

Oracle provides several `DBA_WORKLOAD_*` views to help monitor and manage the Workload Replay feature. For example, the `DBA_WORKLOAD_REPLAYS` view shows you all the workloads that have been replayed in a database. You can monitor the workload replay by using new views such as `DBA_WORKLOAD_CAPTURES` and `DBA_WORKLOAD_FILTERS`.

The SQL Performance Analyzer

One of the major problems you encounter during an Oracle database upgrade is the difficulty of predicting the impact of a database upgrade on the functionality as well as the performance of the database. In Oracle Database 11*g*, you can perform a what-if analysis of potential database changes and their impact on SQL performance using the powerful SQL Performance Analyzer. The SQL Performance Analyzer (SPA) performs fine-grained performance analysis of individual SQL statements and provides suggestions for solving the potential performance degradation of SQL statements consequent to a database upgrade, say, from the Oracle Database 11.1 release to the Oracle Database 11.2 release. The SPA provides these recommendations by comparing performance before the upgrade and following the upgrade. You thus have the opportunity to catch those SQL queries whose performance is likely to worsen following a database upgrade and tune them so you can avoid the performance degradation. Essentially, the SPA lets you compare and analyze two versions of workload performance in order to identify those SQL statements that are likely to be affected by the changes you are making.

■**Note** You can use a SQL workload that you captured into an STS on an Oracle Database 10.2.0.1 (and newer) release with the SQL Performance Analyzer feature in an Oracle Database 11*g* database.

Once the SPA identifies SQL execution plan modifications and performance regression brought about by changes such as a shift to a different optimizer level, for example, you can use the SQL Tuning Advisor to retain the original execution plans or tune the SQL statements that have regressed following a database change.

In addition to using the SQL Performance Analyzer for gauging the potential SQL performance changes after a database upgrade, you can also use it to analyze any changes in the database that can potentially impact SQL performance, such as the following:

- Database and application upgrades
- Hardware changes
- Operating system changes
- Initialization parameter changes
- SQL tuning actions such as the creation of SQL profiles
- Schema changes

You first need to decide whether you're going to run the SQL Performance Analyzer analysis on the production system or on an identically configured test system. If you decide to go with a test system, you can use RMAN's `duplicate` command to create your test version of the production database. Your test system, should you choose to go that route, must use the same database version and initialization parameters as the production database. If you use a test system for the replay, you can capture the SQL workload on the production system, import it into a test system, and run the SQL Performance Analyzer there to compare the pre-upgrade and post-upgrade SQL performance. We recommend you use a test system to avoid the extra load on the production system.

In this example, we'll show you how to use the SQL Performance Analyzer to help predict SQL performance changes following the upgrade of a database from the Oracle 10.2 release to the Oracle 11.1 release. We assume you're using a test database to run the analysis instead of running it on the production system. You must configure the test database as similarly as possible to the production system to get the most out of the SQL Performance Analyzer. Oracle recommends you use Oracle Enterprise Manager to run the SQL Performance Analyzer. However, we show you how to run the tool using the PL/SQL procedures in the new package `DBMS_SQLPA`, which provides the interface for the SQL Performance Analyzer. When you use this package to run the SQL Performance Analyzer, you create a SQL Performance Analyzer task to contain the results of the SQL replay trials you perform. You also use several new procedures in the `DBMS_SQLTUNE` package to conduct your performance analysis with the SQL Performance Analyzer.

■ **Note** You can run the SQL Performance Analyzer on the production database from which you collect the SQL workload or on a similarly configured test database. Running the job on the production database entails some additional resource usage but gives you the most representative results. If performance is a big concern, then use a test database to run the analysis.

You can store the SQL workload that you capture in a SQL tuning set (STS). An STS is a database object that includes the SQL text of one or more SQL statements along with information pertaining to their execution environment, the execution plan, and the execution statistics. You can also specify that the STS include the execution plans and row source statistics for the SQL statements in the STS. Using an STS lets you collect and store the SQL information in a persistent database object, which you can modify and select from later on, just as you would data from a database table. An STS also makes it easy for you to export the SQL workload from the production system to the test system and provide the workload as input to the SQL Performance Analyzer.

You can use any of the following sources to load statements into an STS:

- The automatic workload repository (AWR)
- A cursor cache
- Another STS

Once you capture the SQL statements that comprise the SQL workload on the production system in a SQL tuning set, you follow these steps to perform a SQL Performance Analyzer task:

1. *Measure the pre-change SQL workload performance*: The SQL Performance Analyzer executes the SQL statements that are part of the STS that you create and generates the execution plan and execution statistics for those statements. The SPA stores the execution information in the STS.
2. *Make the system changes*: After the first run of the SQL Performance Analyzer, you make the changes that you want to test on the testing system. For example, you may want to test the migration to a different release of the database by installing the new release and changing the initialization parameter `optimizer_features_enable` to the new version.

3. *Measure the post-change SQL workload performance*: The SQL Performance Analyzer collects the SQL performance data again, after you make the database change you're testing.

4. *Compare the SQL performance*: The SQL Performance Analyzer will compare the SQL performance before and after the changes you made. The goal is to identify changes in the execution plans of the SQL statements that are part of the captured workload. Statistics such as the execution times, buffer gets, and disk reads are some of the metrics that serve as the basis for comparing SQL performance.

Capturing the Production System SQL Workload

Your first task in running the SQL Performance Analyzer is to capture a representative SQL workload from the production system so you can analyze its performance before and after the database upgrade. The SQL workload consists of not just the SQL statements but additional environmental information such as the SQL bind variable values and the execution frequency of the SQL statements. As mentioned earlier, you can use the AWR, a cursor cache, or an STS to load the production database SQL workload into an STS. The following sections cover the steps you must follow to capture the production system SQL workload into an STS.

Creating the SQL Tuning Set

The first thing you must do is create a new STS, which you can then execute on the production system, to capture the workload statistics:

```
SQL> exec dbms_sqltune.create_sqlset(sqlset_name => 'upgrade_set',
         description  => '11g upgrade workload';
```

The name of the new STS is `upgrade_set`, which we'll use to store the captured SQL workload on the production system. The parameter `sqlset_owner` in the `create_sqlset` procedure defaults to the name of the current schema owner. Note that the `create_sqlset` procedure creates just an empty STS, which you must load with the necessary SQL statements later.

Loading the SQL Tuning Set

Once you create your STS, you must load the SQL statements into it by using the `load_sqlset` procedure of the `dbms_sqltune` package, as shown here:

```
declare
  mycur dbms_sqltune.sqlset_cursor;
begin
  open  mycur for
    select value (P)
    from table (dbms_sqltune.load_sqlset(
      'parsing_schema_name <> ''SYS'' AND elapsed_time > 2500000',
      null,null,null,null,1,null,
      'ALL')) P;
    dbms_sqltune.load_sqlset(sqlset_name => 'upgrade_set',
                             populate_cursor => cur);

end;
/
PL/SQL procedure successfully completed.
SQL>
```

Now that you have captured a representative SQL workload from the production system, it's time to export this STS to the test system where you'll conduct the performance analysis.

Transporting the SQL Tuning Set

You must first create a staging table using the `create_stgtab_sqlset` procedure before you can export the STS you created in the previous step. This staging table is where you export the STS from the production database and subsequently import it from there to the test database. Here are the steps in transporting the STS you captured from the production system:

```
SQL> exec dbms_sqltune.create_stgtb_sqlset ( table_name => 'stagetab');
```

The previous code creates a new staging table named `stagetab`.

Once you create your staging table, it's time to export the STS into that table by using the `pack_stgtab_sqlset` procedure:

```
SQL> exec dbms_sqltune.pack_stgtab_sqlset(sqlset_name => 'upgrade_set',
         staging_table_name => 'stagetab');
```

After the export of the STS from the production system is complete, you're ready to import the STS into the test system.

Import the STS into the Test System

After you use the Data Pump Export utility to export the staging table `stagetab` (which contains the STS), use the Data Pump Import utility to load the staging table into your test system. After you import the staging table, you must run the `unpack_stgtab_sqlset` procedure in order to import the STS into the replay database:

```
SQL> exec dbms_sqltune.unpack_stgtab_sqlset (sqlset_name = '%',
          replace => true, staging_table_name => ('stagetab');
```

Once you successfully import the STS into the replay database, your next step is to create a SQL Performance Analyzer task, as explained in the next section.

Creating a SQL Performance Analyzer Task

To compare the execution of the captured production workload on the pre- and post-upgrade test environments, you must first create a SQL Performance Analyzer task, using the `DBMS_SQLPA` package. Use the `create_analysis_task` procedure to create a tuning task for the STS you created earlier:

```
SQL> exec dbms_sqlpa.create_analysis_task(sqlset_name => 'upgrade_set',
          task_name=> 'spa_task1');
```

Since you're specifying an STS as the source of the SQL workload, make sure you have already created and loaded the STS before creating the SPA task as shown here.

Analyzing the Pre-change SQL Workload

To get the best results from using the SQL Performance Analyzer, your test system must resemble the production system as closely as possible. After installing the Oracle Database 11 release software, you need to set the `optimizer_features_enable` parameter to match the production system version on which you captured the SQL STS, as shown here:

```
optimizer_features_enable=10.2.0
```

By setting the value of the `optimizer_features_enable` parameter to 10.2.0, you can generate SQL performance statistics for a 10.2 database you're about to upgrade to the 11*g* version. Now you are all set to capture the pre-upgrade SQL performance data. Use the `execute_analysis_task` procedure to analyze the SQL workload, as shown here:

```
SQL> exec dbms_sqlpa.execute_analysis_task (task_name => 'spa_task1',
          execution_type => 'test execute',
          execution_name= 'before_change');
```

The key parameter here is the `execution_type` parameter, which can take the values `explain plan`, which generates explain plans for the SQL statements in the SQL workload, and `test execute`, which actually executes all SQL statements in the workload. In our example, we specify `test execute` as the value for the `execution_type` parameter, because we want to execute all the SQL statements in our STS. Note that only DML queries are executed so as to prevent an adverse impact on the data.

Analyzing the Post-upgrade SQL Workload

To compare the effect of a system change, you must first make that change on the test system. In this example, we're trying to figure out the impact of a database upgrade from the 10.2 release to the 11.1 release. To test the impact of the database upgrade, you must first set the value of the initialization parameter `optimizer_features_enable` to the 11*g* version:

```
optimizer_features_enable=11.1
```

Once you do this, you can collect the SQL performance data post-upgrade by running the same `execute_analysis_task` procedure as you did before you made the change:

```
SQL> exec dbms_sqlpa.execute_analysis_task (task_name => 'spa_task1',
         execution_type => 'test execute', execution_name='after_change')
```

The final step is to compare the SQL performance between the two runs of the `execute_analysis_task` procedure.

Comparing the SQL Performance

You must execute the `execute_analysis_task` for a third and final time in order to compare the SQL performance before and after the database upgrade. You must understand how to set the values for the `execute_type` parameters for this third and final execution of the execute_tuning_task procedure. Since two executions of execute_analysis_task, both of the execution type test execute, already exist, you must now pass the value compare performance to the execution_type parameter so the procedure can analyze and compare the SQL performance data from the two workload analyses you conducted.

Here's how you invoke the execute_analysis_task procedure to compare SQL performance before and after the system change:

```
SQL> exec dbms_sqltune.execute_analysis_task (task_name =>

         'spa_task1',
         execution_type => 'compare performance',
         execution_params =>
         dbms_advisor.arglist('comparision_metric',
                          'disk_reads',);
```

In this example, we used disk_reads as the value for the comparision_metric parameter, which lets you specify a variety of statistics to measure the performance impact of the system changes you make. Other possible values for this parameter include elapsed_time, optimizer_cost, direct_write, parse_time, buffer_gets, and so on.

Generating the SQL Performance Analyzer Report

Once you finish running the comparison analysis, it's time to generate a report showing the results. You can generate the results of the SQL performance comparison in report form by executing report_analysis_task, as shown here:

```
var report clob;
exec :report := dbms_sqlpa.report_analysis_task('spa_task1', 'text',
                'typical','summary');
set long 100000 longchunksize 100000 linesize 120
print :report
```

In this example, we chose to generate a text report (HTML and XML are the other options) and summary as the value for the section parameter (all is the other option).

Analyzing the Performance Report

The SQL Performance Analyzer report in this example uses the disk_reads comparison metric to evaluate the database change you made (different optimizer level). The report contains three main sections: general information, result summary, and result details. The result summary section shows overall performance statistics pertaining to the entire SQL workload as a whole, indicating whether performance will improve or degrade after the database changes you're putting in place.

The report provides detailed execution statistics for each of the SQL statements in the STS you provide and summarizes the findings by saying whether the performance of a statement has regressed and whether its SQL execution plan has changed. Wherever the report indicates a performance regression, it also will contain a root cause analysis and recommendations to improve the execution plan of that SQL statement. You can then tune the regressed statements using the recommendations.

Note Oracle Database 11*g* deprecates the use of stored outlines to preserve execution plans for SQL statements. The stored outlines feature is replaced by a newer and more powerful feature called SQL Plan Management that employs SQL plan baselines, which are sets of known good execution plans for SQL statements. Under this, the optimizer maintains a history of execution plans for SQL statements. When an execution plan changes, SQL Plan Management evaluates the new plan during the next maintenance window, and if it finds that the new plan is indeed better, it will replace the old plan with the new execution plan.

The SQL Performance Analyzer offers benefits such as a low overhead for data capture and integration with the cost optimizer when compared to third-party tools that perform similar tasks. In addition, the SQL Performance Analyzer offers the additional advantage that it's integrated with other Oracle tools such as the SQL Tuning Advisor and the SQL Plan Management feature. You can use both the SQL Tuning Advisor and the new SQL Plan Management feature to carry out the SQL Performance Analyzer recommendations made after the system change. You can place the new execution plans generated as a result of the system change you implemented in the SQL plan baseline repository. Thus, you can ensure that the optimizer will pick only previously validated execution plans. Following this, the database can automatically validate the plans seeded by you in the SQL plan baseline, or you can manually validate them yourself. Chapter 4 discusses the SQL plan baselines feature.

Database Software Patching

Once the Oracle Database 11*g* server software is installed, you might want to upgrade your databases to the new release. Before you go live on your 11*g* production database environment, check for any available patch set release. Patch sets don't provide additional functionality, but they provide bug fixes and don't need certification for installation on the system. Similarly, check for any critical path updates, which are the security updates made available by Oracle on a quarterly basis. Applying critical path updates secures your new database from any significant security vulnerabilities that were discovered by Oracle. Just as important, you should make sure to check the bug database and carefully consider critical bugs and pertinent bugs that may cause database outage situations.

New Features in Database Control for Patching

A software patch is a software update to fix defects in the software (bug fixes). A patch is thus supposed to fix bugs but doesn't involve any new functionality as such. Periodically Oracle releases maintenance releases, which are known as *patch sets*. Patch sets are a set of integrated product fixes. For example, if you're using Oracle Database 11*g* Release 11.1.0.1 software, a new patch set might be termed 11.1.0.3. In Oracle Database 10*g*, Database Control and Grid Control provided the crucial patch advisory. However, the Patch Prerequisite Check feature wasn't available in Database Control. In Oracle Database 11*g*, Database Control is enhanced by providing the Parch Prerequisite Check feature. In addition, Database Control now has the Software Library feature as well. You can stage a patch once in the Software Library and use it for multiple deployments.

Enterprise Manager now searches proactively for patches that are relevant to a specific customer's environment. Your installation and your database feature usage determine the search for patches. A daily patch job will run to correlate with the patch metadata on MetaLink. When the patch job finds relevant patches for your environment, it sends you an alert and enables you to apply that patch. As soon as a new one-off patch becomes available, you will be alerted by the proactive MetaLink patch advisory if the patch is relevant to your database environment. Thus, you can count on reliable deployment with this automation of patching.

The Provisioning Pack lets you automatically deploy software, patches, and applications. You can do the following things with the Provisioning Pack:

- Bare-metal provisioning of operating systems and software images
- Cloning installations and software images, for example, RAC or CRS
- Patching
- Using the Deployment Procedure Manager

OEM Database Control simplifies the staging and application of new patches and patch sets. Database Control can automatically stage a new patch set by proactively searching for it and downloading it from Oracle MetaLink to your server directories. Database Control supports all types of patches, including critical patch updates (CPUs), interim patches, and patch sets. Here are some of the key features concerning patching in Oracle Database 11*g*:

- Live update of MetaLink best practices
- Support for sudo
- Support for Pluggable Authentication Modules (PAM)–based authentication

In Oracle Database 10*g*, Database Control had a limited amount of patch management capabilities. In the Oracle Database 10*g* release, you could do only two patch-related activities through Database Control:

- Applying a patch
- Viewing a patch cache

In Oracle Database 11*g*, you still have the two links that are similar to the patching links in Oracle Database 11*g*. The Stage Patch link corresponds to the Apply Patch link. The View Patch Cache link is the same as in the previous release. However, the Database Software Patching page in Oracle Database 11*g* is much more detailed and offers more tools. On the Database Control home page, click Software and Support to reach the Database Software Patching page. You can go to the following pages from that page.

- *Patch Advisor page*: The Patch Advisor page shows you the currently applicable patches to your installation. The page contains patch advisories in two sections.

■ **Note** You can't use feature-based patch application in Database Control 11*g*, since you won't get any feature usage–based patch advisories.

- *Critical security patches*: Critical security patches (also called *critical patch updates*) are periodic patch releases by Oracle to fix major security holes.
- *Patch recommendations by feature*: Lists all patch recommendations based on the database feature.

- *Patch Cache page*: Any patches you download from the Oracle MetaLink to the Enterprise Manager are saved in the Enterprise Manager repository. You can view the current patches in the repository by clicking the View Patch Cache link, which takes you to the Patch Cache page. This page shows, in a tabular format, all the patches you downloaded from MetaLink. The advantage of the using the patch cache is that you need to download a patch only once and stage it to multiple destinations. If you haven't downloaded a patch, that's OK too, since Enterprise Manager will automatically download the necessary path from MetaLink when the patch job runs.
- *Oracle Patch Prerequisite Checker page*: Click Path Prerequisites in the Database Software Patching section to get to the Oracle Patch Prerequisite Checker page. Here, you can select the software updates from MetaLink or the Software Library. You can stage the updates to a staging location and run prerequisite checks on those updates.
- *Stage Patch page*: This page lets you select patches from MetaLink, based on search criteria you provide.
- *Apply Patch page*: This page lets you select the patches to apply. You can select patches from MetaLink for the Software Library.

Emergency Hot Patching (Online Database Patching)

You can use the online database patching feature in Oracle Database 11*g* to apply emergency, one-off patches to the database server software, while the database is online. This means you can now patch Oracle executables with no downtime. This feature is especially useful for diagnostic patches. Some one-off patches can be done online now.

■ **Hint** Patches can now be installed with the `runInstaller` executable instead of `opatch`. This was one of the enhancements added to 10.2.0.3.

The new online patching capability is integrated with OPatch. Besides installing a new patch, you can uninstall a patch without bringing the database offline. Similarly, you can also enable and disable one-off patches online. You can use the online patching capability to patch many one-off patches online. There is also a subset of online upgradeable patches for Real Application Cluster environments. Oracle plans eventually to enable the online patching of its periodically released critical path updates.

Database Change Management Pack

The Change Management Pack offers you the capability to compare database object definitions before and after changes in the databases. The pack enables you to capture and compare database object definitions from different points in time. Using the Change Management Pack, you can easily associate application modules with the database schema objects. After an application upgrade, for example, you can assess the impact of the changes on the dependent database objects so you can modify the application modules in accordance with the changes in the database schema. In addition, you can track changes to initialization parameters as well as authorization and storage settings.

You can use the following types of sources to compare the database objects:

- Two databases
- Two baselines
- A database and a baseline

Please refer to the Chapter 4 to learn more about dictionary baselines and dictionary comparisons.

Software and Database Cloning

Oracle Database 11*g* offers gold-image cloning (*gold images* are tested and approved software images) of software on the same server or cluster. Note the following points about software cloning:

- You can clone multiple Oracle homes in parallel.
- You can pre-patch images to a level you select.
- You have a choice of the Software Library or the host itself to get the source image.

You can optionally perform configuration tasks following the cloning job, such as creating a database. In Oracle Database 10*g*, Database Control offered you only the following two sources for cloning a database:

- A running database instance in archivelog or noarchivelog mode
- A saved working directory from a previous cloning operation

In Oracle Database 11*g*, Database Control provides you with vastly enhanced cloning capabilities. You have the following four source types you can choose from now:

- Copy an online database files over Oracle Net, without any prior backups.
- Copy an online database’s files via staging areas.
- Use RMAN whole-database backups.
- Use a special backup made by a previous database cloning operation.

You can get to the Clone Oracle Home page by clicking the Software and Support link on the Database Control home page and then clicking the Clone Oracle Home link in the Configuration section.