UPGRADING YOUR DATABASES TO 11g: AVOIDING PITFALLS AND ENSURING SUCCESS

Maria Anderson, I.S.P., OCP, Petro-Canada

INTRODUCTION

Oracle databases, and the applications they support, continue to become more mission critical to organizations making it a challenge to perform timely database upgrades. Database upgrades must be tested exhaustively and the expectation is often that a production upgrade will be performed flawlessly. To ensure a successful database upgrade, extensive planning and testing are essential, as is a good understanding of the entire upgrade process. This paper will cover the basics of a database upgrade in a UNIX environment and, in addition, walk through the specific steps involved in an upgrade to Oracle Database 11*g*. In conclusion, a discussion of lessons learned will provide information about what works well throughout the upgrade process.

WHY UPGRADE?

The initiative to upgrade an Oracle database usually comes from one of three sources: (1) the DBA is aware that end of life support is fast approaching and, therefore, feels it prudent to upgrade to a supported release of the database, (2) an application software upgrade requires the latest features, or (3) someone in the organization (developer, manager, etc.) read an article about the cool features in the latest Oracle database release and they want it now! Regardless of how a database upgrade is initiated, the fact is that the entire process requires extensive planning and exhaustive testing to be successful. Even when these two factors are taken into account, there may still be an impact on the applications supported by the upgraded database(s).

Considering what *can* go wrong with a database upgrade, the question remains why upgrade? Large organizations are typically conservative and averse to unnecessary change, so at times it can be challenging for a DBA to initiate an upgrade project. One important reason to upgrade is supportability. As one falls farther and farther behind a supported database release, it begins to impact other components of an organization's infrastructure. Soon, operating system upgrades become a problem because the database release is too far behind and not certified. Following that, hardware becomes incompatible, as well as application releases especially if they are vendor-supported as opposed to in-house. When an upgrade is finally performed (after two or three releases of database software), the chances of having performance problems and application incompatibilities are far greater. While organizations tend to shy away from being on the 'bleeding edge' of technology, staying with a supported database release makes good business sense. An additional benefit of upgrading is to take advantage of the new features introduced in the latest release. While no organization will utilize every new feature in a database release, even one or two may prove to be time-saving features.

Oracle Database 11*g* now provides tools to help mitigate the risk of applications being adversely affected by a database upgrade. In addition to planning and testing, you can now more reliably predict the impact of upgrading by utilizing Real Application Testing. Real Application Testing includes Database Replay and SQL Performance Analyzer. At the time this paper was written, these tools were only available with Oracle 11*g* but there is the possibility that Oracle will backport these tools for use with older releases.

DETERMINE THE UPGRADE PATH

There are a number of options for upgrading a database. As you will see in the next section, there are direct and indirect methods of upgrading. Whether you choose one method over the other can often depend on what release the current database is at. A database must be at release 9.2.0.4, 10.1.0.2 or 10.2.0.1 or higher for a direct upgrade to be supported. A

database at a lower release must be first upgraded to one of these releases, and then to 11*g*. Refer to the table below to determine whether a direct upgrade is supported for your database …

Table 2–1 Upgrade Paths

Current Release	Upgrade Path
7.3.3 and lower 7.3.4 8.0.3 8.0.4 8.0.5 8.0.6 8.1.5 8.1.6 8.1.7.4 9.0.1.4	Direct upgrade is *not* supported. Upgrade to an intermediate Oracle Database release before you can upgrade to Oracle Database 11*g* Release 1 (11.1), as follows: ▪ 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 When upgrading to an intermediate Oracle Database release, follow the instructions in the intermediate release's documentation. Then, upgrade the intermediate release database to Oracle Database 11*g* Release 1 (11.1) using the instructions in Chapter 3, "Upgrading to the New Release".
9.2.0.4 10.1.0.2 10.2.0.1	Direct upgrade to Oracle Database 11*g* Release 1 (11.1) is supported from 9.2.0.4 or higher, 10.1.0.2 or higher, and 10.2.0.1 or higher. Note that Oracle Clusterware release 10.2.0.x must be at release 10.2.0.3 (or higher), before you attempt to upgrade it to Oracle Clusterware 11*g* release 1 (11.1). See "Upgrading an Oracle Real Application Clusters (Oracle RAC) Database" on page 3-2. For release 9.2.0.3 you must first upgrade to an intermediate Oracle Database release, as follows: 9.2.0.3 (or lower) -> 9.2.0.8 -> 11.1 To upgrade to Oracle Database 11*g* Release 1 (11.1), follow the instructions in Chapter 3, "Upgrading to the New Release".

Table 1: Supported upgrade paths [1]

METHODS OF UPGRADING

There are a number of different methods available for upgrading databases to 11*g*. The method chosen depends on several factors: the release and size of the database, allowed time for a scheduled outage, and how quickly one will need to back out if there are issues encountered during the upgrade.

The method encouraged by Oracle Corporation is a direct upgrade using the Database Upgrade Assistant (DBUA). The DBUA provides a graphical interface which allows an existing database to be upgraded to 11*g* in place. This means that a database currently at release 9.2.0.4 or 10.2.0.1, for example, will become an 11.1 database after it is upgraded successfully.

Another option for performing a database upgrade in place is to go through the same steps the DBUA would perform, but do it manually. This method involves running SQL scripts and allows you more control than the DBUA, but it is prone to error. If a step is forgotten during the process, it could jeopardize the upgrade and cause errors or a failed upgrade. For both direct upgrade methods, the only way to back out of an upgrade is to either downgrade the database or restore it from a backup

[1] Source: Oracle Database11g Release 1 (11.1) Upgrade Guide, Oracle Technology Network (http://download.oracle.com/docs/cd/B28359_01/server.111/b28300/toc.htm).

taken previous to the upgrade itself. Refer to Table 1 in the previous section to determine whether a direct upgrade is possible.

Other methods involve indirectly upgrading a database; that is, keeping the original database at its current release but using Oracle utilities (export/import or Data Pump) to create a new database at release 11.1. This method means that there will be two copies of a database after the upgrade. The basic steps involved are as follows:

- Create a new skeleton 11*g* database either on the same server as the database to be upgraded or on another server. Create the appropriate application tablespaces, users and roles.
- Stop all activity on the source database by shutting down applications. If write activity is not stopped on the database, after the import is completed there must be a method in place to copy the changed data to the newly created 11*g* database.
- Shut down the database and then start it up in restricted mode.
- Export the required application schemas from the database using either export (9.x or lower) or Data Pump export (10.1 or higher).
- Shut down the original database.
- Import the schemas into the newly created 11*g* database. Make all appropriate TNS changes and connect applications to the new database.

These indirect methods will work for relatively small databases, and they offer an easy back-out plan. If something unexpected happens during the import causing the upgrade to fail, simply restart the original database and allow applications to connect to it again. This method does require extra disk space since there are now two copies of the same database and it can take considerably longer to perform an upgrade in this manner, but has the added benefit of making a back-out much easier. Of course, this may be your only choice if a direct upgrade is not possible.

Another type of indirect upgrade is to perform a data copy between the original database and a newly created 11*g* database. This, again, requires the 11*g* database to be pre-created with empty tablespaces. A database link is created in the 11*g* database which connects to a user account in the original database. Using this database link, you would issue a series SQL commands such as :

```
create table as select …
… and/or …
insert into … select …
```

This method is quite cumbersome to use and you may not be able to copy across all object types. After all tables are copied across, you must recreate indexes and ensure all constraints were applied. It can be easy to miss something; plus this method would take considerably longer than upgrading in place.

One final method of upgrading is to use transportable tablespaces. In this case, instead of dumping the contents of a schema using export or Data Pump export, a physical copy of the datafiles are made and 'attached' to a newly created 11*g* database. This is definitely faster than using export/import or data copy and may be a good alternative for larger databases. There is a good article in the 2008 Q1 Select magazine by Lei Zeng[2] that describes this method of upgrading in a very concise and logical manner.

[2] Zeng, Lei "Using Transportable Tablespaces to Perform a Database Upgrade", Select Journal, Vol. 15 Number 1, 2008.

DOCUMENTATION

Prior to planning a database upgrade project, read through both the new features guide and the upgrade guide of the release you want to upgrade to. These are both good sources of information and can be found on the Oracle Technology Network (OTN) at the following location:

http://www.oracle.com/pls/db111/homepage

The Oracle Database 11*g* Release 1 (11.1) New Features Guide provides very good information on the new features available in Oracle 11*g*. Besides planning and implementing a successful database upgrade project, it would be beneficial to also suggest some useful and time-saving new features to your business users or clients.

The Oracle Database 11*g* Release 1 (11.1) Upgrade Guide is a wealth of information on the different methods of upgrading a database, as well as the type of testing that must be performed to ensure an upgrade project completes successfully. Both the new features guide and the upgrade guide can be found on OTN. Another useful resource on OTN is the Oracle 11*g* home page located here …

http://www.oracle.com/technology/products/database/oracle11g/index.html

Besides white papers and links to the official 11*g* documentation, there are also podcasts on 11*g* and links to blogs. The forums on OTN are very valuable providing a venue to ask questions about 11*g* and see what issues other DBAs and developers are experiencing.

There are already several Oracle 11*g* books written by industry experts that are available and worth reading. These types of books not only provide useful technical information, but they often contain real-life experiences that can make your upgrade project go smoothly. Finally, there is the IOUG website and Select Journal. Both of these sources provide articles and previous conference papers that discuss tried and true methods for planning and implementing upgrades.

PLANNING AND TESTING: THE KEYS TO A SUCCESSFUL UPGRADE

Regardless of how successful a database upgrade is technically, there are myriad other considerations to be taken into account before even embarking on an upgrade project. There are application and business considerations, as well as compatibility and supportability considerations. As a senior DBA, it is simply not enough to be technically competent; one must also take into consideration the business impact of a project like this and mitigate risks.

First, if this information is not readily available, put together a list of databases that are scheduled for an upgrade. This does not need to be a high tech application; it can simply be a spreadsheet. Ensure that other documentation about each database is captured such as current release, the applications they support and a contact person for each application. This information may already be available in the organization via a change management application which would be easier than recreating it.

Second, determine which databases have low visibility in the organization and which are critical to business functions. Plan to upgrade databases with the lowest visibility first and leave the mission critical databases for last after several upgrades in production have been completed successfully. Devise a high level plan for how the database upgrade will be tested prior to being performed in production. This will lessen concerns among the business users and application support teams about possible disruptions to their applications.

Third, spend some time researching any open and resolved Oracle 11*g* bugs on Metalink and the OTN Forums. It is better to proactively gather information about potential issues and how they may affect applications, rather than scrambling to gather information when you stumble upon the problem in the middle of an upgrade. If there are bugs/issues discovered, determine whether there are patches available or whether they are unresolved. The discovery of a bug by someone else may save you the time and effort of working through a support request with Oracle. Something unexpected like this could delay your entire upgrade project.

Fourth, approach each business unit to explain that a database upgrade is due and determine a date. There may be some resistance, in which case it would be beneficial to come prepared to such a meeting. This is when reading up on new features might come in handy, as well as discussing the high level plan for testing that was put together earlier. Ensure there is understanding and agreement that testing requires technical resources, but also business and application support team members. Often an explanation of Oracle's lifetime support policy may help to clarify why a database upgrade is crucial in order to remain on a supported release. This is a good time to engage the Change Management team and make them aware of the tentative dates chosen. Your Change Management representatives should work with you to notify all users and schedule the production changes.

Finally, a more detailed plan should flow from the high-level plan created earlier. It may be likely that other DBAs on the team will be recruited to assist with the upgrades if there are a large number of them to perform. If this is the case, a more formal project plan is required so that all DBAs perform the upgrades in the same manner. Devise a detailed plan for how test or non-production environments will be refreshed from production so that the upgrades can be tested several times. Remember to add contingency for both time and effort to the project plan, since each environment will be unique and there could be unanticipated issues. In a worst case scenario, some upgrades will be backed out and performed again at a later date.

DEVELOP A DATABASE UPGRADE TEST PLAN

Now that you have business approval to proceed with the database upgrade project, how will you sufficiently test upgrades prior to attempting them in production environments? There are several phases to testing and they are all crucial to the success of your project.

UPGRADE TESTING

This is the first phase of testing and involves creating an environment solely for the purpose of testing the database upgrade, as well as application function. If the database is not too large, create a copy of the production environment in this testing environment. If the database is very large, create a subset of the database in this environment. Create part, or all, of the application in this environment and connect it to the test database. It is important at this stage to devise a solid test plan with the business users and support team members. This plan will also be used throughout the upgrade process and for the production upgrade to ensure all application functionality is good. Note that each functional testing plan will differ depending on the application or business area. Ensure that key components of the application function in this test environment as they do in production (or as close as possible since it may not be possible to test all functionality in a non-production environment).

At this point, upgrade the database to 11*g* but do not introduce any new database functionality with the upgrade. Connect the application to the database again and work through the original test plan; ensure that all application functionality is still present. Make note of any functions or process that no longer function as they currently do in the production environment or are considerably slower, and research a resolution; also make note of processes that are faster after the upgrade. At this stage, it is important to note which systems (databases and applications) integrate with the database you are attempting to upgrade. If at all possible, it would be prudent to test all the integration points but this may not be possible. At the very least, document interfaces and data integration points that go to and from the database targeted for upgrade.

Functional Testing

Assuming that the first phase of testing was successful and any issues were resolved, the next phase involves introducing new 11*g* database functionality with the upgrade. Once again, work through the test plan with the business users and application support team members. As before, note what does not work at all (or as well) but also make note of what has improved. As new database features are introduced, work through the test plan and document the results. It is possible that the DBA may need to assist the application support team members with tuning SQL, or researching any database bugs uncovered as a result of introducing new functionality.

Performance Testing

It is clearly not sufficient to just ensure that all application and database features work after an upgrade. Database and application performance must be taken into account. If you don't take this into account prior to an upgrade, you may be dealing with many angry users the first business day after you perform a production upgrade. Starting with Oracle 11.1, Real Application Testing is available to help you determine which database features have slowed due to the upgrade. Real Application Testing is not only a useful toolset when performing a database upgrade, but it can also highlight potential problems when making operating system changes, parameter changes, or even applying patch sets. Real Application Testing includes the following tools:

- Database Replay
- SQL Performance Analyzer (SPA)

Database replay offers the ability to capture a production system's workload and then replay it back in a test or other non-production environment. This provides not only a realistic sampling of the SQL executed in a production system, but also a good indication of the workload. After the workload is replayed, database replay provides analysis and reporting to highlight potential issues. Below is a graphical representation of how database replay works:

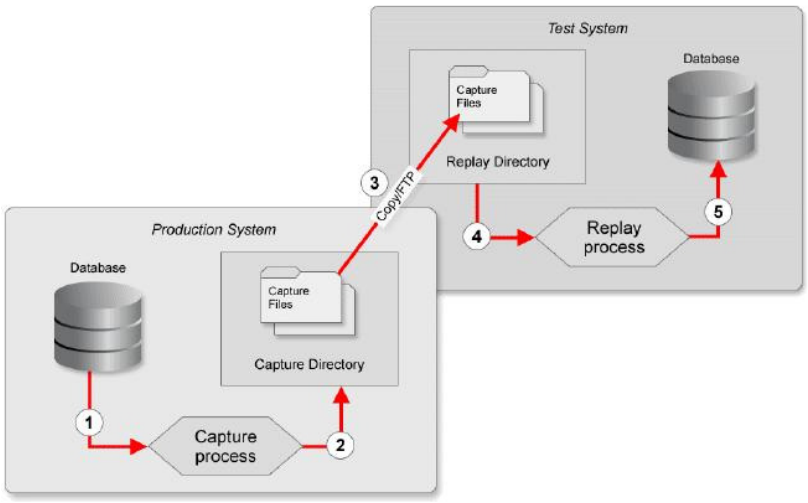


Figure 1: Graphical representation of Oracle Database 11g database replay.[3]

Database replay is more representative of the activity in a production environment as opposed to using most standard stress or load testing tools available on the market. With those tools, you must provide the SQL which the load testing software will

[3] Source: "Oracle Database 11g: The Top New Features for DBAs and Developers", Arup Nanda, Oracle Technology Network (http://www.oracle.com/technology/pub/articles/oracle-database-11g-top-features/11g-replay.html)

use to simulate load. The SQL you provide may or may not be representative of the true workload within the production database. Database replay takes the guessing out of determining a representative workload. The workload captured can be customized to exclude certain types of SQL, such as database management scripts that run frequently. Together with flashback database, database replay can provide an invaluable tool to mitigate the risk in performing database upgrades.

SQL Performance Analyzer (SPA) is a slightly different tool, also currently only available with 11.1, that can assist in determining which SQL statements may suffer performance degradation with an upgrade. The difference between the SQL performance analyzer and database replay is that you choose the SQL statements from a production database that are important, and the analyzer determines the impact of changes on the SQL statements. It provides an analysis of the SQL statement both before and after an upgrade along with recommendations.

While Real Application Testing is not currently available for releases of the database prior to 11*g*, this does not mean that there are no options available to determine SQL performance. SQL Plan Management is available to keep plan changes to a minimum when working with a new database optimizer. The interesting thing here is that you can capture SQL into a SQL Tuning Set (STS) from a 10*g* database and then bulk load the execution plans or pre-existing baselines into the upgraded 11*g* database. With SQL plan management, the optimizer manages these plans and ensures that only known or verified plans are used. If there is a new plan found for a SQL statement, it will not be used until a performance increase over the current plan is verified. This reduces the risk of introducing a new plan that has not been tested or demonstrated to improve performance.

STRESS TESTING

Even after performing all the other types of testing, stress testing can reveal problems that only manifest under load. Unfortunately, this type of testing is often overlooked and deemed not necessary when, in fact, the opposite is true. The perception and experience of your user community is important and if they are unable to log into an application or perform business-critical functions because the database cannot respond to the load of a regular business day, this is a problem.

The previous sections described the various, but necessary, types of testing the DBA must take into account when performing database upgrades. The bottom line is: time spent planning and testing is never wasted!

COMPATIBILITY CONSIDERATIONS

The default compatibility parameter setting for Oracle 11.1 is 11.0.0. You can, however, upgrade a database to 11.1 with a minimum setting of this parameter to 10.0.0. If you neglect to set this parameter, it will be set by default to 11.0. Oracle recommends setting this parameter to 10.0.0 so that you are able to downgrade your database should the upgrade fail. If compatibility is set to 11.0, you will not be able to downgrade. The only option here is to restore your database.

If you set compatibility to 10.0.0 and upgrade your database, a limited number of new 11*g* new features are enabled not including any that make database structural changes. This makes a database downgrade possible. After the database upgrade has been tested, then the compatibility parameter can be set to 11.0 and the database restarted. Remember … once this parameter is changed and the database restarted, you cannot go back to 10.0.0 so be certain that the upgrade has been tested sufficiently.

Table 5–1 The COMPATIBLE Initialization Parameter

Oracle Database Release	Default Value	Minimum Value	Maximum Value
Oracle Database9*i* Release 2 (9.2)	8.1.0	8.1.0.0.0	9.2.0.n.n
Oracle Database 10*g* Release 1 (10.1)	10.0.0	9.2.0.0.0	10.1.0.n.n
Oracle Database 10*g* Release 2 (10.2)	10.2.0	9.2.0.0.0	10.2.0.n.n
Oracle Database 11*g* Release 1 (11.1)	11.0.0	10.0.0.0.0	11.0.0.n.n

Table 2: COMPATIBLE initialization parameter settings for Oracle 9i through 11g. [4]

Another compatibility consideration is the release of the Oracle client that applications use to connect to the database. Upgrading a database may also require upgrading the client software on application servers that connect to the database. It may also mean an upgrade of the Oracle client on workstations that connect directly to the database. This may translate into upgrading only a few servers with a newer release of the Oracle client, or it could mean many. This must be taken into consideration when planning database upgrades. Oracle Database 11*g* is supported with client releases 11.1.0, 10.2.0 and 10.1.0. It is also supported with the 9.2 client but fixes are only possible for clients with Extended Support. For more information on the compatibility between the Oracle database release and the client, refer to Doc ID # 207303.1 on Metalink.

CREATE A TECHNICAL IMPLEMENTATION PLAN (TIP)

A technical implementation plan is a very detailed document that contains the step-by-step instructions for performing a change, in this case a database upgrade. It contains emergency contact information for all technical people involved in the change, and a pre-implementation section with tasks that must be completed prior to the actual upgrade. The bulk of the document contains detailed steps (with actual commands) for upgrading a database and, finally, there is a post-implementation section with tasks that must be completed *after* the upgrade has been completed successfully.

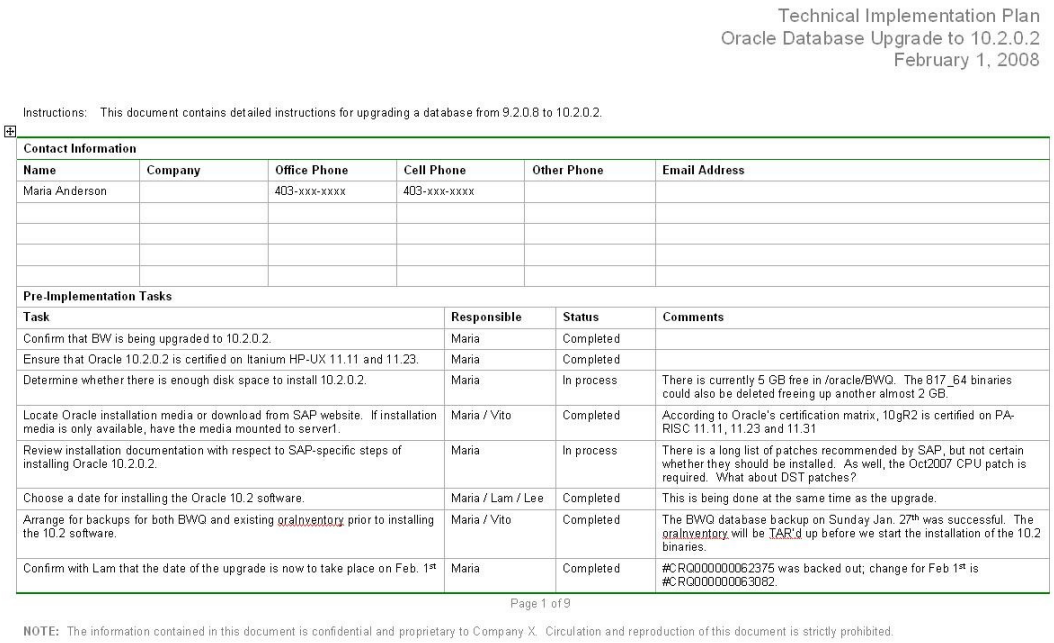
Typically, there is one plan for each production upgrade performed since there may be slightly different steps to follow pre-implementation and post-implementation depending on the applications supported by a database. As each upgrade is performed, the plan is updated with actual times and information. The latest TIP will then serve as a template for the next upgrade and should contain all the latest newly discovered information about upgrading.

Other added benefits of creating a document like this are …

- It serves as an audit trail for compliance requirements
- It supplements change management documentation
- Provides detailed technical instructions in the event that another DBA must step in to perform the change at the last minute

An example of a TIP (pages 1 and 2) appears below:

[4] Source: Oracle Database11g Release 1 (11.1) Upgrade Guide, Oracle Technology Network (http://download.oracle.com/docs/cd/B28359_01/server.111/b28300/toc.htm).



Technical Implementation Plan
Oracle Database Upgrade to 10.2.0.2
February 1, 2008

Instructions: This document contains detailed instructions for upgrading a database from 9.2.0.8 to 10.2.0.2.

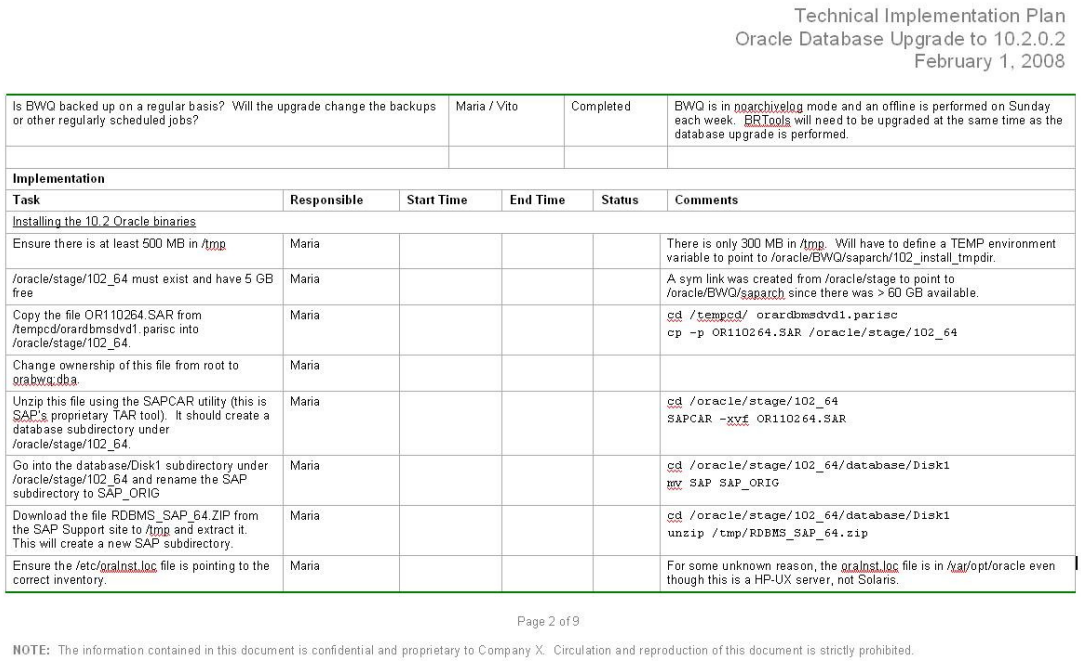
Contact Information					
Name	**Company**	**Office Phone**	**Cell Phone**	**Other Phone**	**Email Address**
Maria Anderson		403-xxx-xxxx	403-xxx-xxxx		

Pre-Implementation Tasks			
Task	**Responsible**	**Status**	**Comments**
Confirm that BW is being upgraded to 10.2.0.2.	Maria	Completed	
Ensure that Oracle 10.2.0.2 is certified on Itanium HP-UX 11.11 and 11.23.	Maria	Completed	
Determine whether there is enough disk space to install 10.2.0.2.	Maria	In process	There is currently 5 GB free in /oracle/BWQ. The 817_64 binaries could also be deleted freeing up another almost 2 GB.
Locate Oracle installation media or download from SAP website. If installation media is only available, have the media mounted to server1.	Maria / Vito	Completed	According to Oracle's certification matrix, 10gR2 is certified on PA-RISC 11.11, 11.23 and 11.31
Review installation documentation with respect to SAP-specific steps of installing Oracle 10.2.0.2.	Maria	In process	There is a long list of patches recommended by SAP, but not certain whether they should be installed. As well, the Oct2007 CPU patch is required. What about DST patches?
Choose a date for installing the Oracle 10.2 software.	Maria / Lam / Lee	Completed	This is being done at the same time as the upgrade.
Arrange for backups for both BWQ and existing oraInventory prior to installing the 10.2 software.	Maria / Vito	Completed	The BWQ database backup on Sunday Jan. 27th was successful. The oraInventory will be TAR'd up before we start the installation of the 10.2 binaries.
Confirm with Lam that the date of the upgrade is now to take place on Feb. 1st	Maria	Completed	#CRQ0000000062375 was backed out; change for Feb 1st is #CRQ0000000063062.

Page 1 of 9

NOTE: The information contained in this document is confidential and proprietary to Company X. Circulation and reproduction of this document is strictly prohibited.

Figure 2: Example of a technical implementation plan (TIP) – page 1.



Technical Implementation Plan
Oracle Database Upgrade to 10.2.0.2
February 1, 2008

Is BWQ backed up on a regular basis? Will the upgrade change the backups or other regularly scheduled jobs?	Maria / Vito	Completed	BWQ is in noarchivelog mode and an offline is performed on Sunday each week. BRTools will need to be upgraded at the same time as the database upgrade is performed.	

Implementation					
Task	**Responsible**	**Start Time**	**End Time**	**Status**	**Comments**
Installing the 10.2 Oracle binaries					
Ensure there is at least 500 MB in /tmp	Maria				There is only 300 MB in /tmp. Will have to define a TEMP environment variable to point to /oracle/BWQ/saparch/102_install_tmpdir.
/oracle/stage/102_64 must exist and have 5 GB free	Maria				A sym link was created from /oracle/stage to point to /oracle/BWQ/saparch since there was > 60 GB available.
Copy the file OR110264.SAR from /tempcd/orardbmsdvd1.parisc into /oracle/stage/102_64.	Maria				cd /tempcd/ orardbmsdvd1.parisc cp -p OR110264.SAR /oracle/stage/102_64
Change ownership of this file from root to orabwq:dba.	Maria				
Unzip this file using the SAPCAR utility (this is SAP's proprietary TAR tool). It should create a database subdirectory under /oracle/stage/102_64.	Maria				cd /oracle/stage/102_64 SAPCAR -xvf OR110264.SAR
Go into the database/Disk1 subdirectory under /oracle/stage/102_64 and rename the SAP subdirectory to SAP_ORIG	Maria				cd /oracle/stage/102_64/database/Disk1 mv SAP SAP_ORIG
Download the file RDBMS_SAP_64.ZIP from the SAP Support site to /tmp and extract it. This will create a new SAP subdirectory.	Maria				cd /oracle/stage/102_64/database/Disk1 unzip /tmp/RDBMS_SAP_64.zip
Ensure the /etc/oraInst.loc file is pointing to the correct inventory.	Maria				For some unknown reason, the oraInst.loc file is in /var/opt/oracle even though this is a HP-UX server, not Solaris.

Page 2 of 9

NOTE: The information contained in this document is confidential and proprietary to Company X. Circulation and reproduction of this document is strictly prohibited.

Figure 3: Example of a technical implementation plan (TIP) – page 2.

PREPARING TO UPGRADE

Besides sufficient planning and exhaustive testing, a DBA must also take into consideration the impact of upgrades on the existing database infrastructure and monitoring environment. With each major database release, there are likely changes that will first require modifications to the existing environment. For example, if you use a RMAN catalog for database backup and recovery then the catalog will have to be upgraded prior to upgrading your first database. Other examples include using OEM

Grid Control; you must ensure that the agent can still collect the same information from your database as before or whether the agent must be upgraded. An additional possibility is that it may be necessary to update any monitoring scripts that are currently being used in the environment. With Oracle 11*g* come changes in the location of the alert log and trace files. In the past, they were stored in locations identified by the initialization parameters background_dump_dest, user_dump_dest and core_dump_dest. Starting with Oracle 11*g*, there is now a new initialization parameter called diagnostic_dest which contains all trace files and alert logs. If you have scripts that monitor alert logs for ORA- errors, you may need to modify these scripts first. In addition, there are some data dictionary view changes with Oracle 11*g* which may impact scripts or jobs that currently rely on data dictionary views to retrieve database or session information.

Some Oracle 11*g* adjustments that must be taken into consideration prior to an upgrade:

- As of Oracle 10*g*, the CONNECT has changed and now only contains the 'create session' privilege. This means all users with this role in your database will have to be identified and explicitly granted privileges they used to have via the CONNECT role if upgrading from a release prior to 10*g*.
- As of Oracle 10.2, the limit for FAILED_LOGIN_ATTEMPTS for the DEFAULT profile is 10. Prior to this release, the default was UNLIMITED.
- Passwords in Oracle 11*g* are now case-sensitive. This may affect any password expiry and security procedures that are currently in place within the database.
- During an upgrade from Oracle 9.2 or 10.1, any passwords in database links are encrypted. If you must downgrade after a failed upgrade to the original release, database links with encrypted passwords are dropped so they will not exist after the downgrade. Be sure to query the SYS.LINK$ table and document information on any existing database links prior to upgrading.
- Oracle 11*g* uses version 4 time zone files. This may affect data using the TIMESTAMP WITH TIME ZONE data type. If you have not updated your existing time zone files to version 4, you must do so prior to upgrading. If you are using a time zone file version greater than 4, then after installing Oracle 11*g* you must apply the appropriate patch. The bottom line is that the time zone file versions must match between the database being upgraded and Oracle 11*g*.

As with each major database release, there are some initialization parameters that have been deprecated or made obsolete. These parameters will be identified when you run the upgrade preparation script (described later in this paper) and are using the DBUA to perform an upgrade. If, however, you have chosen to perform a manual upgrade you must be sure to remove these parameters.

Deprecated initialization parameters in Oracle 11*g*:

- BACKGROUND_DUMP_DEST (replaced by DIAGNOSTIC_DEST)
- COMMIT_WRITE
- CORE_DUMP_DEST (replaced by DIAGNOSTIC_DEST)
- INSTANCE_GROUPS
- LOG_ARCHIVE_LOCAL_FIRST
- PLSQL_DEBUG (replaced by PLSQL_OPTIMIZE_LEVEL)
- PLSQL_V2_COMPATIBILITY
- REMOTE_OS_AUTHENT
- STANDBY_ARCHIVE_DEST
- TRANSACTION_LAG attribute (of the CQ_NOTIFICATION$_REG_INFO object)
- USER_DUMP_DEST (replaced by DIAGNOSTIC_DEST)

Obsolete initialization parameters in Oracle 11*g*:

- DDL_WAIT_FOR_LOCKS
- LOGMNR_MAX_PERSISTENT_SESSIONS
- PLSQL_COMPILER_FLAGS

Note that if you are upgrading from 9.2.0.4 to 11*g*, deprecated and obsolete parameters for Oracle 10*g* must also be taken into consideration.

One of the last, but most important, pre-upgrade tasks to consider is the backup strategy currently in place for your databases. Will the backup you perform prior to a database upgrade be sufficient to recover from? Keep in mind that if you are performing an upgrade in place and if the downgrade does not rollback the upgrade as expected, your only option is to restore and recover the database. You will need a solid backup strategy in place before this happens. After the upgrade is completed, will the current backup strategy be sufficient going forward, or should it be re-evaluated? These are all questions that must be answered prior to starting a database upgrade project.

UPGRADING TO 11g

Now that there is a solid plan in place for upgrading databases and testing the upgrade, the first step toward upgrading is to install the Oracle 11*g* database software. Preferably, you will install Oracle 11*g* for the first time in a non-production environment so as to familiarize yourself with the installation process. Not much has changed between an installation of Oracle 10*g* and 11*g*, however, a production environment is never a good place to try *anything* the first time!

The following sections take the reader through some high-level steps of upgrading a database in place to Oracle 11*g*. Other indirect methods of upgrading (e.g., using export/import, data copy, transportable tablespaces, etc.) are not demonstrated. There is also a brief section on installing Oracle 11*g* software. A word of caution: the steps documented here are very high-level. Do not rely solely on this document to capture *all* the necessary steps of a successful installation and upgrade. Always refer to Oracle's official documentation, as well as Metalink, for complete technical requirements and instructions.

REQUIREMENTS FOR INSTALLING 11g SOFTWARE

Prior to installing Oracle 11*g* on a server, you must refer to the specific installation guide for the platform on which you are installing. The Installation Guides can be found on OTN and contain valuable information such as hardware and software requirements, configuring kernel parameters, etc. The installation guides can be found here …

http://www.oracle.com/pls/db111/homepage

The first document to read is the Readme document located here …

http://download.oracle.com/docs/cd/B28359_01/readmes.111/b28280/toc.htm

This document contains information on known compatibility issues with 11*g*, default behavior changes, in addition to specific information for each Oracle option. Next, you will notice that there is a quick installation guide and a full installation guide available for each platform. Start with the quick installation guide and read through the steps of installing 11*g*. Sometimes, this document is all you require to successfully install the software. Should you encounter issues, however, the full installation guide can be helpful in resolving issues.

Another good source of information is the Oracle 11*g* Release 1 (11.1) Universal Installer and OPatch User's Guide for Windows and UNIX. This guide does not pertain just to installing Oracle 11*g* software, but discusses the OUI, OPatch utility and cloning. The OUI and OPatch have become important components of Oracle database software so it is in your best interest to understand how they work, as well as how to troubleshoot issues when they occur. The OUI and OPatch User's Guide can be found here …

http://download.oracle.com/docs/cd/B28359_01/em.111/b31207/toc.htm

EXAMPLE ORACLE 11g INSTALLATION ON SOLARIS 10

Prior to installing new Oracle software on an existing server, it is a good idea to set a new entry into the oratab file with a fictitious SID and the new Oracle home. Prior to starting a new installation, use oraenv to set environment variables to this new SID using the new Oracle home. Because a new installation updates the global inventory, it is a good practice to backup the existing global inventory, as well as existing Oracle binaries just in case the installation fails.

The Oracle 11*g* installation process is somewhat improved over 10*g*. After starting the Oracle Universal Installer, you will see the Welcome screen. Note that the option to create a starter database has been deselected in this example …

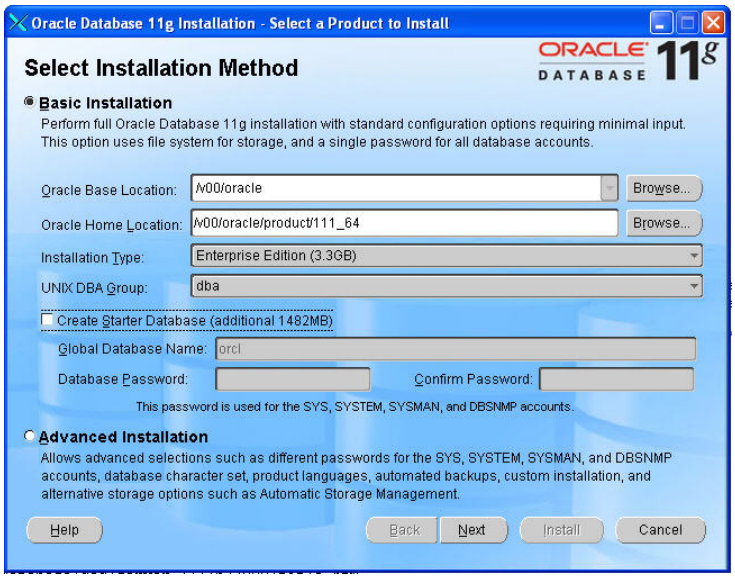


Figure 4: Oracle 11g Installation welcome screen

The next screen performs operating system checks to ensure that all requirements are satisfied prior to the installation …

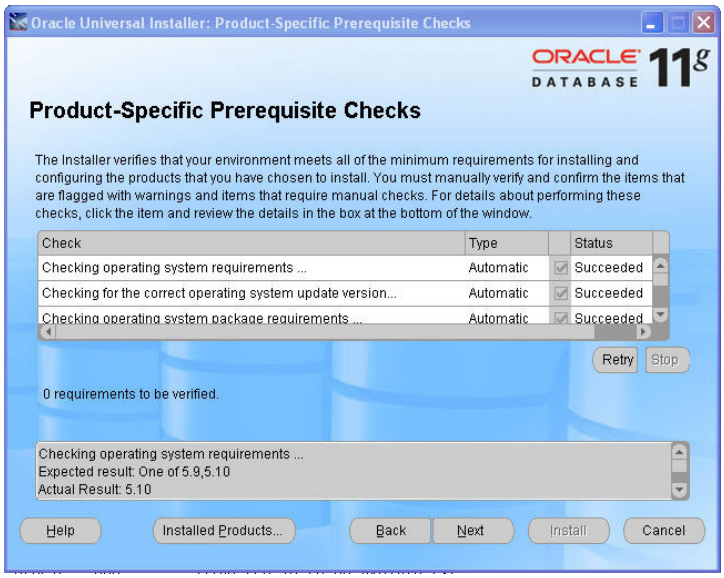


Figure 5: Operating system prerequisite checks

It will also output the results into the bottom pane of this screen. The results will look something like this …

```
Checking operating system requirements ...
Expected result: One of 5.9,5.10
Actual Result: 5.10
Check complete. The overall result of this check is: Passed
=====================================================
Checking for the correct operating system update version...
Check complete. The overall result of this check is: Passed
=====================================================

Checking operating system package requirements ...
Checking for SUNWbtool; found CCS tools bundled with SunOS(SUNWbtool).  Passed
Checking for SUNWcsl; found Core Solaris, (Shared Libs)(SUNWcsl). Passed
Check complete. The overall result of this check is: Passed
=====================================================

Checking kernel parameters
Expected result: BIT_SIZE=64
Actual Result: BIT_SIZE=64
Check complete. The overall result of this check is: Passed
=====================================================

Checking physical memory requirements ...
Expected result: 922MB
Actual Result: 32640MB
Check complete. The overall result of this check is: Passed
=====================================================

Checking available swap space requirements ...
Expected result: 24480MB
Actual Result: 41441MB
Check complete. The overall result of this check is: Passed
=====================================================

Checking PATH environment variable...
Check complete. The overall result of this check is: Passed
=====================================================
```

```
Checking for sufficient diskspace in TEMP location...
Check complete. The overall result of this check is: Passed
=============================================================

Checking LD_LIBRARY_PATH environment variable...
Check complete. The overall result of this check is: Passed
=============================================================

Validating ORACLE_BASE location (if set) ...
Check complete. The overall result of this check is: Passed
=============================================================

Checking Oracle Home path for spaces...
Check complete. The overall result of this check is: Passed
=============================================================

Checking for proper system clean-up....
Check complete. The overall result of this check is: Passed
=============================================================

Checking for Oracle Home incompatibilities ....
Actual Result: NEW_HOME
Check complete. The overall result of this check is: Passed
=============================================================
```

If any requirements failed, stop the installation and resolve the issue. You can restart the installation at a later time. If all requirements are met, the next screen is a summary prior to starting the installation …

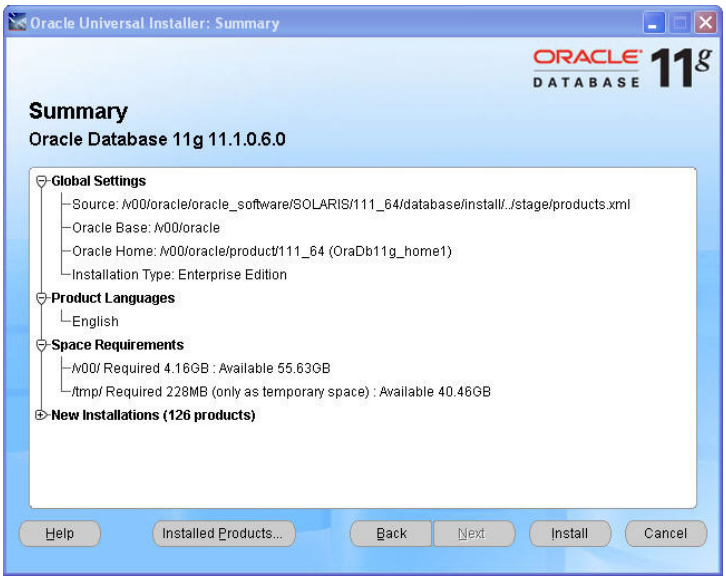


Figure 6: Pre-installation summary

If the 'Install' button is clicked, the installation commences. Note that this phase can take some time but progress is shown as the installation moves through the various phases …

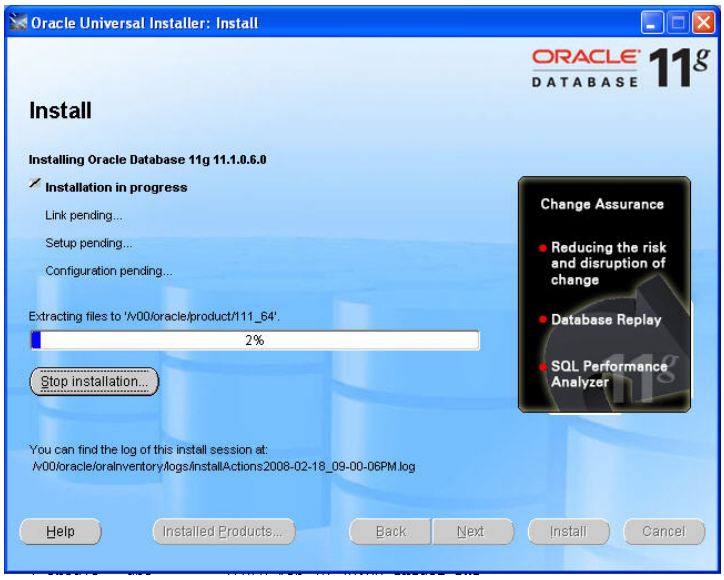


Figure 7: Oracle 11g installation in progress

If the installation goes well, there should be a successful message displayed …

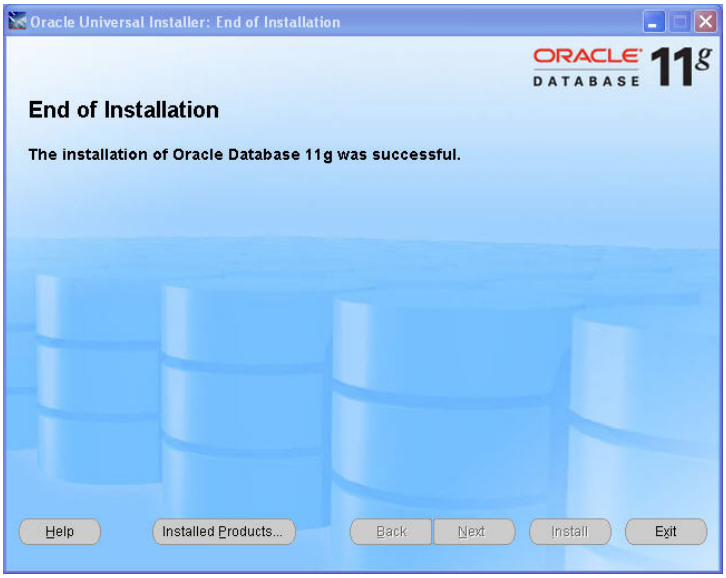


Figure 8: Successful Oracle 11g installation

If the 'Installed Products' button is clicked, you should see the Oracle 11*g* home as well as others that were installed previously …

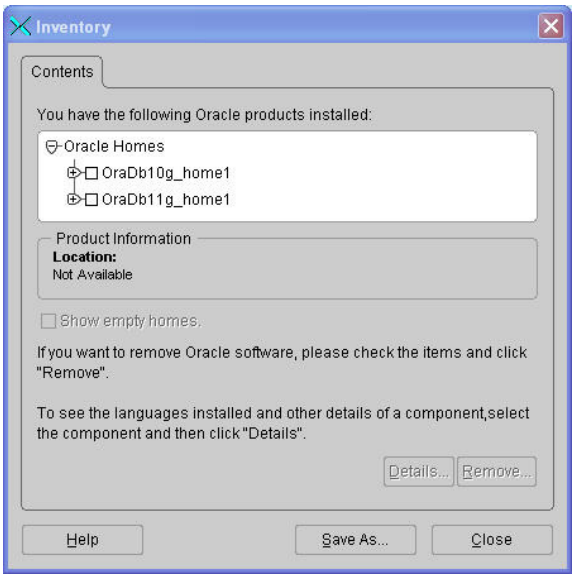


Figure 9: Different Oracle homes installed on a server

PRE-UPGRADE STEPS

The next logical step is to run the pre-upgrade script. Copy the utlu111i.sql script found in the new $ORACLE_HOME/rdbms/admin to a location outside of the Oracle home (I generally create a subdirectory under $ORACLE_BASE/admin/<SID> called upgrade for scripts like this). Log into the database as SYS (or another user with SYSDBA privileges) and run the script from the new location – make sure you spool the output. This script analyzes the database and provides very important information on what needs to change prior to attempting an upgrade. The pre-upgrade script contains information for the following sections:

- Database
- Logfiles
- Tablespaces
- Updated parameters
- Deprecated parameters
- Obsolete parameters
- Components
- Miscellaneous warnings
- SYSAUX tablespace

If you prefer, it can also be run on a non-production database as long as the environment mirrors production. Here is an example of the output …

```
Oracle Database 11.1 Pre-Upgrade Information Tool    02-19-2008 21:03:45
.
**********************************************************************
Database:
**********************************************************************
--> name:          ORATEST
--> version:       10.2.0.3.0
--> compatible:    10.2.0.3.0
```

```
--> blocksize:     8192
--> platform:      Solaris[tm] OE (64-bit)
--> timezone file: V3
.
**********************************************************************
Tablespaces: [make adjustments in the current environment]
**********************************************************************
--> SYSTEM tablespace is adequate for the upgrade.
.... minimum required size: 725 MB
.... AUTOEXTEND additional space required: 245 MB
--> UNDOTBS1 tablespace is adequate for the upgrade.
.... minimum required size: 453 MB
.... AUTOEXTEND additional space required: 428 MB
--> SYSAUX tablespace is adequate for the upgrade.
.... minimum required size: 423 MB
.... AUTOEXTEND additional space required: 173 MB
--> TEMP tablespace is adequate for the upgrade.
.... minimum required size: 61 MB
.... AUTOEXTEND additional space required: 41 MB
--> EXAMPLE tablespace is adequate for the upgrade.
.... minimum required size: 78 MB
.
**********************************************************************
Update Parameters: [Update Oracle Database 11.1 init.ora or spfile]
**********************************************************************
-- No update parameter changes are required.
.
**********************************************************************
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]
**********************************************************************
--> "background_dump_dest" replaced by  "diagnostic_dest"
--> "user_dump_dest" replaced by  "diagnostic_dest"
--> "core_dump_dest" replaced by  "diagnostic_dest"
.
**********************************************************************
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 interMedia            [upgrade]  VALID
--> Spatial                      [upgrade]  VALID
--> Data Mining                  [upgrade]  VALID
--> Expression Filter            [upgrade]  VALID
--> Rule Manager                 [upgrade]  VALID
--> Oracle OLAP API              [upgrade]  VALID
.
**********************************************************************
Miscellaneous Warnings
**********************************************************************
WARNING: --> Database is using an old timezone file version.
.... Patch the 10.2.0.3.0 database to timezone file version 4
```

```
.... BEFORE upgrading the database.  Re-run utlu111i.sql after
.... patching the database to record the new timezone file version.
WARNING: --> Database contains stale optimizer statistics.
.... Refer to the 11g Upgrade Guide for instructions to update
.... statistics prior to upgrading the database.
.... Component Schemas with stale statistics:
....   SYS
....   CTXSYS
WARNING: --> Database contains schemas with objects dependent on network packages.
.... Refer to the 11g Upgrade Guide for instructions to configure Network ACLs.
WARNING: --> EM Database Control Repository exists in the database.
.... Direct downgrade of EM Database Control is not supported. Refer to the
.... 11g Upgrade Guide for instructions to save the EM data prior to upgrade.
.

PL/SQL procedure successfully completed.
```

Read through the output of this script to determine what must change prior to the upgrade. In some cases, an outage may be required to restart the database for changes to take effect so some planning ahead may be required.

USING THE DATABASE UPGRADE ASSISTANT

After reviewing the output of the pre-upgrade script and making any suggested changes in the current environment, it's time to attempt an upgrade in place using the Database Upgrade Assistant (DBUA). Once again, this example is using a Solaris 10 server and upgrading a 10.2.0.3 database to release 11.1.0.6.

Navigate to the new 11*g* Oracle home and then into the bin subdirectory. In this example, the 11*g* database upgrade assistant is located in /v00/oracle/product/111_64/bin …

```
cd /v00/oracle/product/111_64/bin
./dbua
```

The welcome screen should appear …

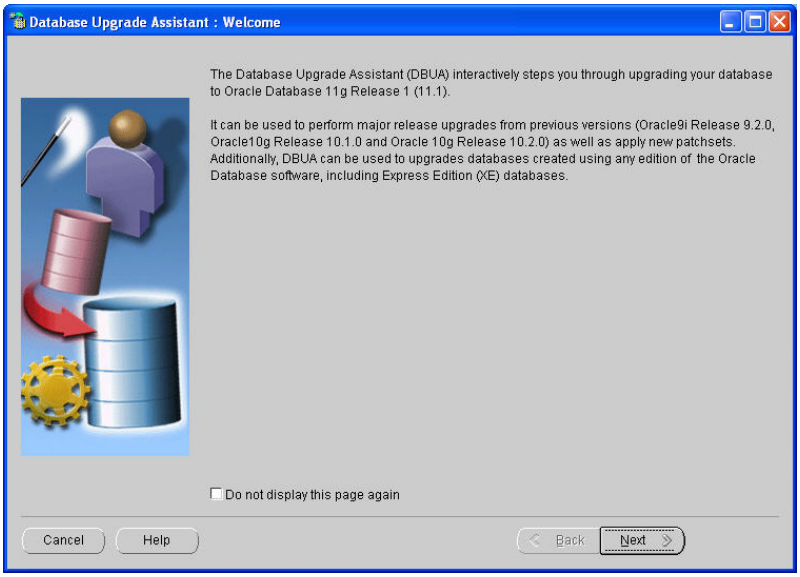


Figure 10: DBUA welcome screen

After choosing 'Next', a screen with the list of available databases to upgrade on that server will display. This list is compiled from entries in the oratab file on the server. Choose the database you wish to upgrade …

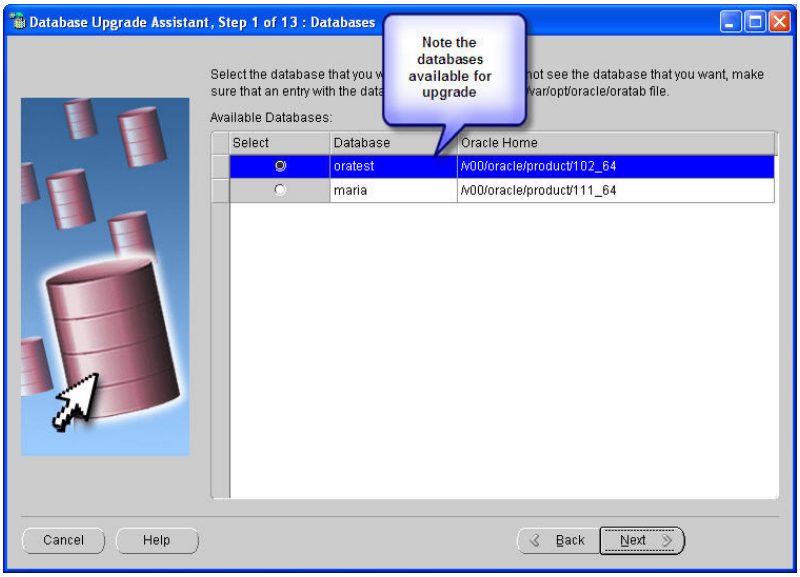


Figure 11: DBUA list of databases to upgrade

The next screen asks where you would like to create the diagnostic destination. Remember that the once familiar background_dump_dest, user_dump_dest and core_dump_dest initialization parameters have now been replaced with diagnostic_dest. The default is to put this in $ORACLE_BASE …

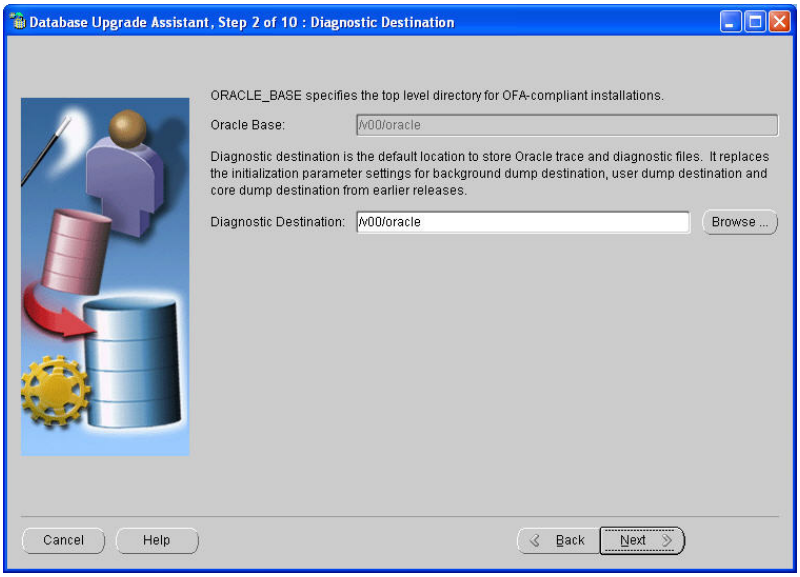


Figure 12: Location of diagnostic destination

Next you will be asked whether you would like to move datafiles during the upgrade …

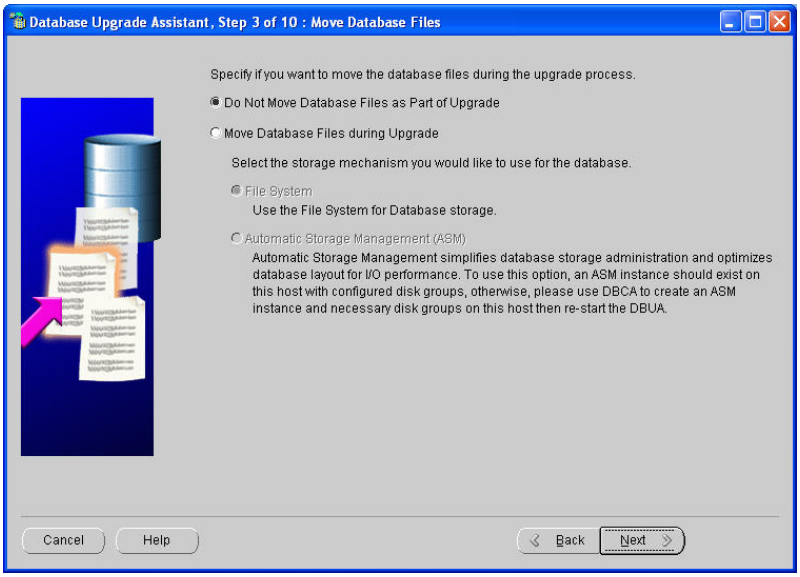


Figure 13: Option to move database files

The next two screens ask whether you wish to configure a Flashback Recovery area and whether you would like to configure the database for OEM. Choose the option most appropriate for your environment. Next, you will receive a confirmation screen to run utlrp.sql to recompile invalid objects. The parallelism is automatically calculated based on the number of CPUs detected. Accept the default and click 'Next' …

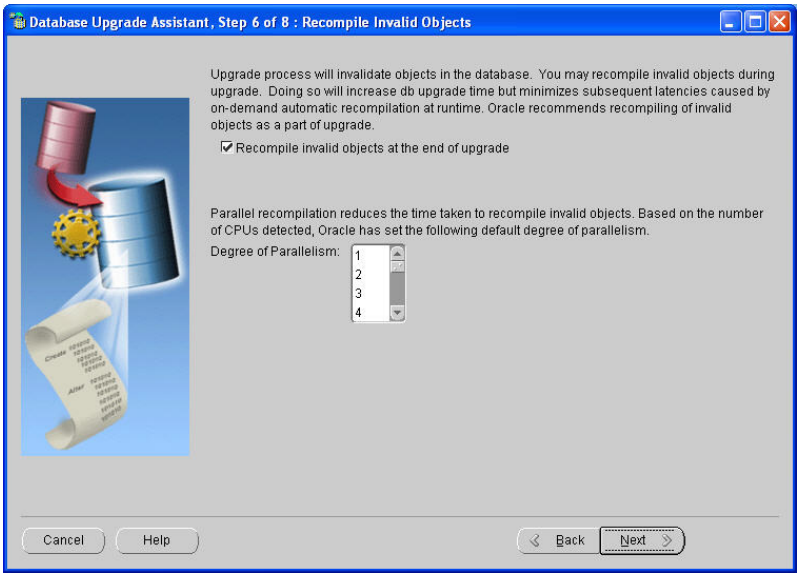


Figure 14: Option to recompile invalid objects during upgrade

The next screen is very important – it provides the opportunity to backup the database prior to starting the upgrade. Ensure that you either take a successful backup on your own prior to starting this process, or let the DBUA backup the database …

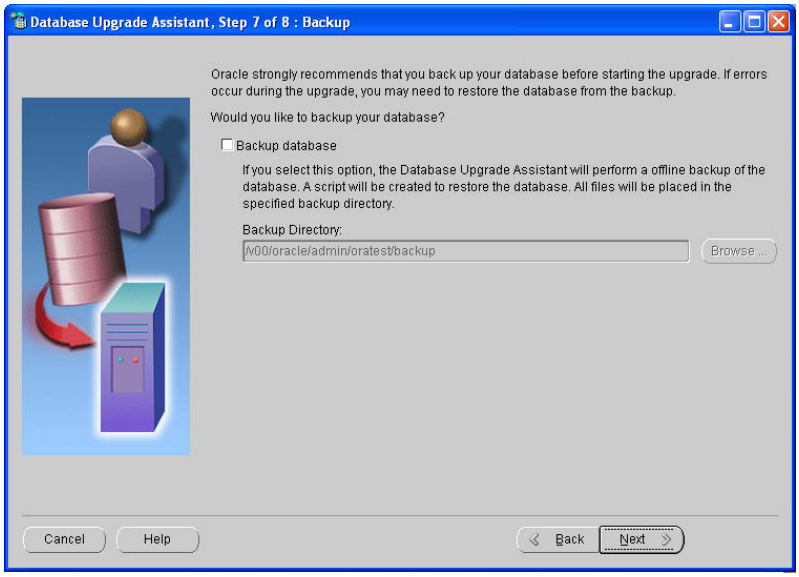


Figure 15: Option for DBUA to take a backup of the database

The final screen presents a summary of the upgrade process. Click 'Finish' to commence the upgrade …

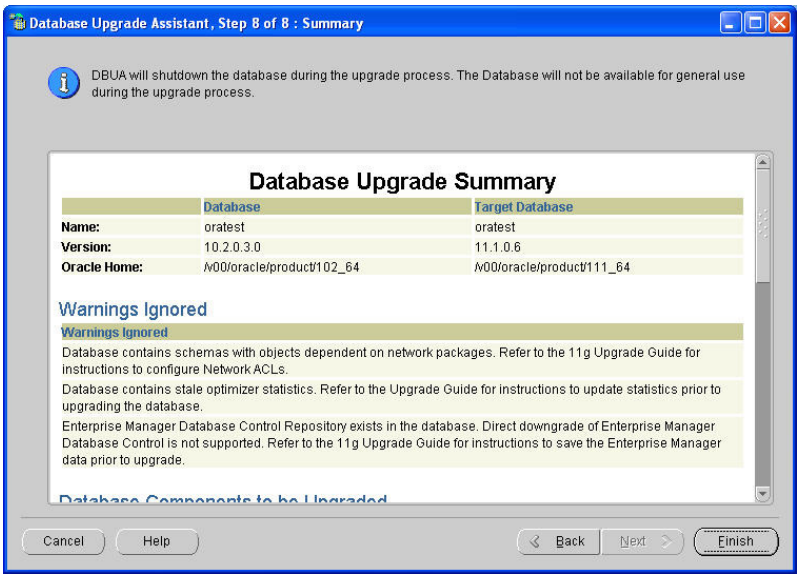


Figure 16: Upgrade summary

When the upgrade process commences, you will see a progress screen that looks similar to this …

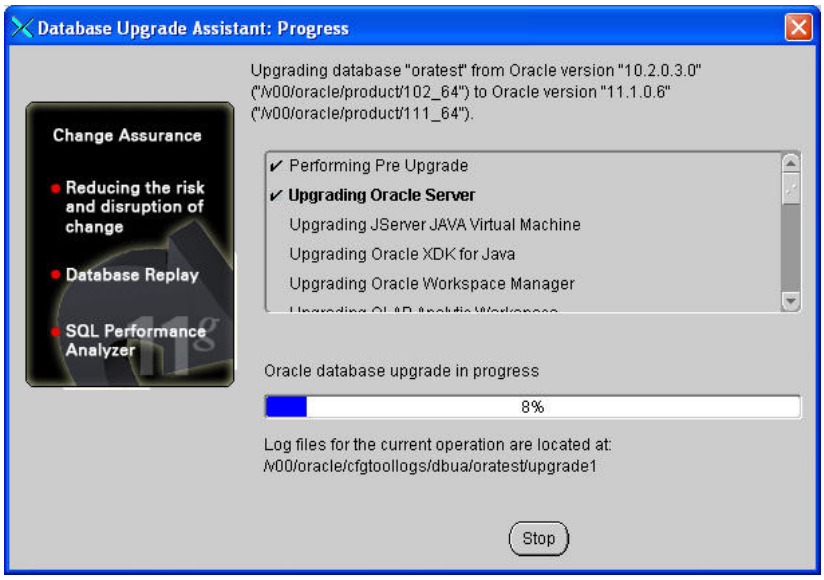


Figure 17: Oracle 11g upgrade progressing

When the upgrade has completed successfully, you will see a summary screen with details on the components upgraded.

PERFORMING A MANUAL DATABASE UPGRADE

While Oracle recommends using the DBUA, there are times when this is not possible due to slow network connectivity or other factors. In cases like this, if you have direct connectivity to the database server you can run the appropriate scripts instead of using the DBUA. This process affords you more control over the process, however, is more error-prone than using the DBUA. This is when having a solid, very detailed technical implementation plan (TIP) will ensure no steps are missed.

The steps to perform a manual upgrade in place to Oracle 11*g* are as follows …

1. Backup the database to be upgraded. The DBUA will prompt you to backup your database or do this for you, but there is no reminder to backup when the manual upgrade method is used. This step is crucial so ensure that a good solid (preferably offline) backup is taken.
2. As in the previous section prior to starting the DBUA, copy the utlu111i.sql script found in the new $ORACLE_HOME/rdbms/admin to a location outside of the Oracle home. A good place to put this script is in a subdirectory under $ORACLE_BASE/admin/<SID> called upgrade. Log into the database as SYS (or another user with SYSDBA privileges) and run the script from the new location – ensure you spool the output. Make any required changes suggested by this script.
3. Copy configuration files from the current $ORACLE_HOME/dbs to the new $ORACLE_HOME/dbs directory. If your database is using a spfile, create a pfile as follows …

```
sqlplus /nolog
connect / as sysdba
create pfile from spfile;
```

4. Copy only the pfile to the new $ORACLE_HOME, as well as the password file if one is currently being used.

5. Remove any obsolete and adjust deprecated initialization parameters in the pfile now in the new $ORACLE_HOME/dbs directory. This information was provided to you by running the pre-upgrade script in step 2. Remember to also set the compatibility parameter if it is not set. Remember that if it is not set, it will default to 11.0.0 and you will not be able to downgrade your database.
6. If you are upgrading from a release prior to 10*g* and are still using rollback segments in the database, this should be converted to using automatic undo. You can do this after the upgrade is completed, but I would recommend that this be done prior to starting the upgrade.
7. At this point, you are ready to start the upgrade. Shut down the database:

   ```
   shutdown immediate;
   ```

   Do not shut down with the abort option. If you must perform a 'shutdown abort', start up the database again and shut down immediate. Remember to also stop the listener and any cron jobs or interfaces that may attempt to connect to the database.
8. Modify the oratab file (/var/opt/oracle/oratab on Solaris, /etc/oratab on other UNIX or Linux platforms) to point to the new $ORACLE_HOME.
9. Set the $ORACLE_SID correctly, i.e., to the database you are upgrading. Ensure that $ORACLE_HOME points to the new Oracle 11*g* software location. Also, your $PATH environment variable should only contain references to the new $ORACLE_HOME.
10. Navigate to the new $ORACLE_HOME/rdbms/admin directory.
11. Log into SQL*Plus and start the database to be upgraded as follows (note the startup command):

    ```
    SQL> connect / as sysdba
    Connected to an idle instance.
    SQL> startup upgrade;
    ORACLE instance started.

    Total System Global Area 1620664320 bytes
    Fixed Size                  2089064 bytes
    Variable Size             385884056 bytes
    Database Buffers         1207959552 bytes
    Redo Buffers               24731648 bytes
    Database mounted.
    Database opened.
    ```

12. The UPGRADE keyword enables you to open the database using earlier Oracle software. It also restricts logins to the database with 'AS SYSDBA' privileges, and disables certain system triggers. When doing upgrades in the past with older releases of the Oracle software, an underscore parameter used to have to be set (_SYSTEM_TRIG_ENABLED = FALSE) but this is no longer necessary.
13. If you are upgrading from a previous release to 10*g*, you will need to create a SYSAUX tablespace with the following mandatory attributes:
    - ONLINE
    - PERMANENT
    - READ WRITE
    - EXTENT MANAGEMENT LOCAL
    - SEGMENT SPACE MANAGEMENT AUTO

    The pre-upgrade script that was executed in step 2 would have provided an estimate for how large to make SYSAUX. Here is an example …

```
SQL> CREATE TABLESPACE sysaux DATAFILE '/v01/oradata/oratest/sysaux01.dbf'
         SIZE 500M REUSE
         EXTENT MANAGEMENT LOCAL
         SEGMENT SPACE MANAGEMENT AUTO
         ONLINE;
```

14. Spool the contents of the upgrade to a log file for review later.

```
spool upgrade.log
```

15. Run the catupgrd.sql script …

```
@catupgrd.sql
```

The catupgrd.sql script determines other scripts that must be called and runs them. It also modifies existing data dictionary tables and views as required.

16. The last thing the catupgrd.sql does is shutdown the database. If, for some reason, the script does not do this restart the database normally …

```
sqlplus /nolog
connect / as sysdba
shutdown immediate;
startup;
```

This restart flushes all caches and clears buffers. This is an important step of the upgrade.

17. If you encountered a message when the database was started about obsolete and/or deprecated parameters, create a pfile from the spfile, edit the file and restart the database. Create an spfile file from the pfile.
18. Review the log file to ensure there were no errors during the upgrade process.
19. Run the Post-Upgrade Status Tool to ensure all components of the database have been upgraded to 11.1.

```
sqlplus /nolog
connect / as sysdba
@utlu111s.sql
```

The output will look similar to this:

```
 SQL> @utlu111s.sql
.
Oracle Database 11.1 Post-Upgrade Status Tool           02-23-2008 15:38:35
.
Component                               Status         Version  HH:MM:SS
.
Oracle Server
.                                        VALID      11.1.0.6.0  01:15:32
JServer JAVA Virtual Machine
.                                        VALID      11.1.0.6.0  00:34:01
Oracle Workspace Manager
.                                        VALID      11.1.0.6.0  00:03:36
OLAP Analytic Workspace
.                                        VALID      11.1.0.6.0  00:02:41
OLAP Catalog
.                                        VALID      11.1.0.6.0  00:02:58
Oracle OLAP API
```

```
.                                                  VALID      11.1.0.6.0  00:01:11
Oracle Enterprise Manager
.                                                  VALID      11.1.0.6.0  00:41:45
Oracle XDK
.                                                  VALID      11.1.0.6.0  00:02:05
Oracle Text
.                                                  VALID      11.1.0.6.0  00:02:50
Oracle XML Database
.                                                  VALID      11.1.0.6.0  00:13:25
```

< … snip …>

If any components show a status of INVALID, it is possible that some database objects remained invalid at the end of the upgrade. Running utlrp.sql to recompile objects may resolve this. If required, you can rerun the upgrade by following the same steps as above …

- shutdown the database and restart it up with the UPGRADE option
- run catupgrd.sql and then utlu111s.sql

20. Log into SQL*Plus as "/ as sysdba" and run the catuppst.sql script, and then run the utlrp.sql. These two scripts can be run concurrently if desired:

```
sqlplus /nolog
connect / as sysdba
@catuppst.sql
@utlrp.sql
```

At this point, the upgrade has been completed. The database is now an Oracle 11.1 database, but there are some critical post-upgrade steps that must be performed prior to releasing the upgraded database for testing.

POST-UPGRADE STEPS

All environment variables for this database that once pointed to the old $ORACLE_HOME must now always point to the Oracle 11*g* home. Be sure to modify all files that reference $ORACLE_HOME, $PATH, $LD_LIBRARY_PATH, or $SHLIB_PATH. These variables must now reference the Oracle 11*g* software location. Never start the upgraded database with the old software.

Modify the listener.ora file and start the listener from the 11*g* $ORACLE_HOME. This will allow applications and/or application servers to now connect to the database. Test connectivity from applications to the database, as well as other interfaces to and from the upgraded database. Now is a good time to go through the test plan that was devised earlier in the planning stages with the business users and application support teams.

With respect to password case-sensitivity, user passwords must be reset to take advantage of this feature. If you wish for this to take effect reset passwords as appropriate, however, there must be a procedure in place to notify the users that this has been done.

In terms of your database management infrastructure, there may be changes necessary to monitor your newly upgraded 11*g* database. Database management scripts may need to be changed, for example. Or if a RMAN catalog is being used, this will likely have to be upgraded prior to backing up your first Oracle 11*g* database.

Lastly, if there is agreement that the database upgrade was successful, ensure that another full backup is taken before the system is fully released to the user community.

TECHNICAL ANOMALIES

While testing this upgrade from 10.2.0.3 to 11.1.0.6, there were a few interesting technical glitches discovered.

- This upgrade seems to take longer than previous upgrades to complete, both via the DBUA and manually running scripts. Of course, this is dependent on how many options are installed but, during testing, this upgrade took 3 to 4 hours on a Solaris 10 server dedicated solely to testing this upgrade.
- The utlrp.sql script to recompile invalid objects has changed. It automatically calculates whether to run the script serially or in parallel depending on the number of CPUs and the number of parallel processes set per CPU. For this script to complete successful, the number of sessions for the database had to be set to at least 300.
- If one or more database components does not upgrade successfully, after identifying the root cause and resolving the issue, you can re-run the upgrade either manually or via the DBUA. A first attempt during testing revealed a component that did not upgrade properly (status was INVALID). The issue was resolved and the catupgrd.sql script was re-run, but this did not upgrade the component that had failed during the first upgrade. A database restore and recovery to the previous release (10.2.0.3) was required to restart the upgrade from the beginning.

DOWNGRADING A DATABASE

Sadly, there are occasions when, even though the upgrade itself was successful, you must back out due to a discovered bug or other issue after upgrading to a newer release. Luckily, it is possible to downgrade a database from Oracle 11*g*. Note that you must downgrade to the database release that you upgraded from. For example, if you upgraded from 10.1 to 11.1 then you must downgrade to 10.1 – you cannot downgrade to 10.2. Even within these releases there are restrictions with respect to downgrading. If the release number of your Oracle 10g Release 1 (10.1) database is lower than 10.1.0.5, then you should install the latest patch for Oracle Database 10g Release 1 (10.1) prior to downgrading. Similarly, if the release number of your Oracle 10g Release 2 (10.2) database is lower than 10.2.0.3, then you should install the latest patch release for 10.2 prior to downgrading. Keep in mind that if the COMPATIBLE parameter has been set to 11.0.0 or higher, a downgrade is not possible at all (see section above on Compatibility Considerations).

The basic steps to downgrading a database are as follows:

1. Shut down the database and listener.
2. Navigate to $ORACLE_HOME/rdbms/admin.
3. Shut down the database and restart it with the DOWNGRADE option. The ORACLE_HOME variable should have been set to the 11*g* software by default after the upgrade.

```
sqlplus /nolog
connect / as sysdba
startup downgrade;
```

4. Run the downgrade script. Remember to spool the contents …

```
spool downgrade.log
@catdwgrd.sql
```

5. After the script has completed, stop spooling and shut down the database. Inspect the log file for errors and, if any are found, the downgrade script can be re-run.

6. Change all variables referencing the Oracle 11*g* software back to the originating software path. Ensure these variables are set correctly. You may also have to copy back the pfile if it is no longer in the old $ORACLE_HOME/dbs directory.
7. Navigate to the $ORACLE_HOME/rdbms/admin directory of the previous release.
8. Restart the database with the UPGRADE option …

```
sqlplus /nolog
connect / as sysdba
startup upgrade;
```

9. Run the catrelod.sql and spool the output to a file …

```
spool reload.log
@catrelod.sql
```

10. Review the log and re-run the script if anything fails.
11. Shut down the database normally and then restart it. Check for any invalid objects or components; if there are any run the utlrp.sql script to recompile the invalid objects.
12. If there were any database links prior to the upgrade, these may have to be recreated if they were dropped during the downgrade (this depends on the release). Start up the listener and allow applications to reconnect to the database. The database has now been downgraded to its previous release. Work through the test plan that was created for the upgrade to ensure that applications function as before.

There are circumstances where a downgrade is not possible; in these cases, a database restore and recovery is the only option. One example of a situation like this is a failed upgrade. If the database is left in an inconsistent state after an upgrade, or it was difficult to determine whether the upgrade was completely successful, it is best to restore the database to a point in time prior to the upgrade. The same holds true for a failed downgrade. When in doubt, restore and recover the database to be sure!

SUMMARY

There are many good reasons to maintain your Oracle databases at the latest release, not the least of which is technical support when required from Oracle. Staying current also means having the opportunity to take advantage of the many new features introduced in the latest releases by Oracle. Some of these new features may save your business users (and you) time and effort in managing the environment. Staying at a supported release, however, means regular database upgrades. The key to a successful database upgrade is extensive planning and exhaustive testing in conjunction with application support teams and business users. A technically successful upgrade does not necessarily translate into success for your user community. Prior to approaching your business users to suggest an upgrade, research the release that you are planning to upgrade their databases to and ensure you have a solid plan for addressing any known bugs or issues. Document the upgrade process with a technical implementation plan (TIP) and create one TIP for each production upgrade. This alone will save countless hours as the upgrade project progresses.

Ensuring that your project is well organized and follows a solid, well-tested plan means a smoother upgrade. All it takes is one miserably failed upgrade due to a lack of preparedness or carelessness to make your business users wary the next time they are approached to upgrade their databases. Planning and testing, as well as working closely with your business users, are critical skills that senior DBAs must possess. If these simple rules are followed, your professionalism and thoroughness will speak for itself when you must approach your business users to plan the next major database upgrade.

REFERENCES

Alapati, Sam R. and Kim, Charles, Oracle Database 11*g*: New Features for DBAs and Developers, Apress, 2007.

Bryla, Bob and Loney, Kevin. Oracle Database 11*g* DBA Handbook, Oracle Press, 2008.

Oracle Metalink (http://metalink.oracle.com).

Oracle Database 11*g* Release 1 (11.1) Readme, Part Number B28280-06, Oracle Technology Network (http://download.oracle.com/docs/cd/B28359_01/readmes.111/b28280/toc.htm)

Oracle Database 11*g* Release 1 (11.1) Quick Installation Guide for Solaris Operating System, Part Number B32313-01, Oracle Technology Network (http://download.oracle.com/docs/cd/B28359_01/install.111/b32313/toc.htm)

Oracle Database 11*g* Release 1 (11.1) Upgrade Guide, Part Number B28300-02, Oracle Technology Network (http://download.oracle.com/docs/cd/B28359_01/server.111/b28300/toc.htm)

Oracle Database 11*g* Release 1 (11.1) New Features Guide, Part Number B28279-02, Oracle Technology Network (http://download.oracle.com/docs/cd/B28359_01/server.111/b28279/toc.htm)

Oracle Database 11*g* Release 1 (11.1) Administrator's Guide, Part Number B28310-03, Oracle Technology Network (http://download.oracle.com/docs/cd/B28359_01/server.111/b28310/toc.htm)

Zeng, Lei, "Using Transportable Tablespaces to Perform a Database Upgrade", Select Journal, Volume 15 Number 1, 2008.