Oracle® Database

Upgrade Guide

10*g* Release 1 (10.1)

Part No. B10763-02

June 2004

ORACLE®

Oracle Database Upgrade Guide, 10*g* Release 1 (10.1)

Part No. B10763-02

Copyright © 2002, 2004, Oracle. All rights reserved.

Primary Author: Tony Morales

Contributors: Nipun Agarwal, Sanjay Agarwal, Rick Anderson, Rae Burns, Ben Chang, Lakshminaray Chidambaran, Eugene Chong, George Claborn, David Colello, Jay Davison, Alan Downing, Sreenivas Gollapudi, Brajesh Goyal, Tom Graves, Michael Hartstein, Thuvan Hoang, Wei Huang, Robert Jenkins, Sanjeev Jhala, Christopher Jones, Mark Jungerman, Sanjay Kaluskar, Garrett Kaminaga, Vishwanath Karra, Mark Kennedy, Susan Kotsovolos, Viswanathan Krishnamurthy, Muralidhar Krishnaprasad, Paul Lane, Gordon Larimer, Simon Law, Jing Liu, Juan Loaiza, J. Bill Lee, Bill Maimone, Raghu Mani, Shailendra Mishra, Valarie Moore, Ari Mozes, Kannan Muthukkaruppan, Subramanian Muralidhar, Ravi Murthy, Karuna Muthiah, Mark Niebur, Naresh Pamnani, Greg Pongracz, Franco Putzolu, N. C. Ramesh, Paul Raveling, Ann Rhee, Ajay Sethi, Carol Sexton, Helen Slattery, James Stamos, Debbie Steiner, Alex Tsukerman, Randy Urbano, Guhan Viswanathan, Steven Wertheimer, Rick Wessman, Andrew Witkowski, Lik Wong, Aravind Yalamanchi, Qin Yu

The Programs (which include both the software and documentation) contain proprietary information; they are provided under a license agreement containing restrictions on use and disclosure and are also protected by copyright, patent, and other intellectual and industrial property laws. Reverse engineering, disassembly, or decompilation of the Programs, except to the extent required to obtain interoperability with other independently created software or as specified by law, is prohibited.

The information contained in this document is subject to change without notice. If you find any problems in the documentation, please report them to us in writing. This document is not warranted to be error-free. Except as may be expressly permitted in your license agreement for these Programs, no part of these Programs may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose.

If the Programs are delivered to the United States Government or anyone licensing or using the Programs on behalf of the United States Government, the following notice is applicable:

U.S. GOVERNMENT RIGHTS Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are "commercial computer software" or "commercial technical data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the Programs, including documentation and technical data, shall be subject to the licensing restrictions set forth in the applicable Oracle license agreement, and, to the extent applicable, the additional rights set forth in FAR 52.227-19, Commercial Computer Software--Restricted Rights (June 1987). Oracle Corporation, 500 Oracle Parkway, Redwood City, CA 94065

The Programs are not intended for use in any nuclear, aviation, mass transit, medical, or other inherently dangerous applications. It shall be the licensee's responsibility to take all appropriate fail-safe, backup, redundancy and other measures to ensure the safe use of such applications if the Programs are used for such purposes, and we disclaim liability for any damages caused by such use of the Programs.

Oracle is a registered trademark of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

The Programs may provide links to Web sites and access to content, products, and services from third parties. Oracle is not responsible for the availability of, or any content provided on, third-party Web sites. You bear all risks associated with the use of such content. If you choose to purchase any products or services from a third party, the relationship is directly between you and the third party. Oracle is not responsible for: (a) the quality of third-party products or services; or (b) fulfilling any of the terms of the agreement with the third party, including delivery of products or services and warranty obligations related to purchased products or services. Oracle is not responsible for any loss or damage of any sort that you may incur from dealing with any third party.

Contents

4 After Upgrading a Database

5 Compatibility and Interoperability

Send Us Your Comments

Oracle Database Upgrade Guide, 10*g* Release 1 (10.1)

Part No. B10763-02

Oracle welcomes your comments and suggestions on the quality and usefulness of this publication. Your input is an important part of the information used for revision.

- Did you find any errors?
- Is the information clearly presented?
- Do you need more information? If so, where?
- Are the examples correct? Do you need more examples?
- What features did you like most about this manual?

If you find any errors or have any other suggestions for improvement, please indicate the title and part number of the documentation and the chapter, section, and page number (if available). You can send comments to us in the following ways:

- Electronic mail: infodev_us@oracle.com
- FAX: (650) 506-7227 Attn: Server Technologies Documentation Manager
- Postal service:

  Oracle Corporation
  Server Technologies Documentation
  500 Oracle Parkway, Mailstop 4op11
  Redwood Shores, CA 94065
  USA

If you would like a reply, please give your name, address, telephone number, and electronic mail address (optional).

If you have problems with the software, please contact your local Oracle Support Services.

Preface

This manual guides you through the process of planning and executing database upgrades on the Oracle Database. In addition, this manual provides information about compatibility, about upgrading applications to the new Oracle Database 10*g* release, and about important changes in the new release, such as initialization parameter changes and data dictionary changes.

Oracle Database Upgrade Guide contains information that describes the features and functionality of the Oracle Database (also known as the standard edition) and the Oracle Database Enterprise Edition products. The Oracle Database and the Oracle Database Enterprise Edition have the same basic features. However, several advanced features are available only with the Enterprise Edition, and some of these are optional. For example, to use application failover, you must have the Enterprise Edition with the Real Application Clusters option.

> **See Also:** *Oracle Database New Features* for information about the differences between the Oracle Database and the Oracle Database Enterprise Edition and the features and options that are available to you.

This preface contains these topics:

Audience

Oracle Database Upgrade Guide is intended for database administrators (DBAs), application developers, security administrators, system operators, and anyone who plans or executes Oracle Database upgrades.

To use this document, you need to be familiar with the following:

- Relational database concepts
- Your current release of the Oracle Database
- Your operating system environment

Documentation Accessibility

Our goal is to make Oracle products, services, and supporting documentation accessible, with good usability, to the disabled community. To that end, our documentation includes features that make information available to users of assistive technology. This documentation is available in HTML format, and contains markup to facilitate access by the disabled community. Standards will continue to evolve over time, and Oracle is actively engaged with other market-leading technology vendors to address technical obstacles so that our documentation can be accessible to all of our customers. For additional information, visit the Oracle Accessibility Program Web site at

`http://www.oracle.com/accessibility/`

Accessibility of Code Examples in Documentation

JAWS, a Windows screen reader, may not always correctly read the code examples in this document. The conventions for writing code require that closing braces should appear on an otherwise empty line; however, JAWS may not always read a line of text that consists solely of a bracket or brace.

Accessibility of Links to External Web Sites in Documentation

This documentation may contain links to Web sites of other companies or organizations that Oracle does not own or control. Oracle neither evaluates nor makes any representations regarding the accessibility of these Web sites.

Structure

This document contains:

Chapter 1, "Introduction"

This chapter provides an overview of the database upgrade process, as well as information about running multiple releases of the Oracle Database.

Chapter 2, "Preparing to Upgrade"

This chapter describes the steps to complete before upgrading a database.

Chapter 3, "Upgrading a Database to the New Oracle Database 10g Release"

This chapter guides you through the process of upgrading a database to the new release of the Oracle Database.

Chapter 4, "After Upgrading a Database"

This chapter describes the actions to complete after upgrading a database to the new release of the Oracle Database.

Chapter 5, "Compatibility and Interoperability"

This chapter contains information about compatibility and interoperability between different releases of the Oracle Database, including detailed information about the `COMPATIBLE` initialization parameter.

Chapter 6, "Upgrading Your Applications"

This chapter provides general information about upgrading your applications and tools for use with the new release of the Oracle Database.

Chapter 7, "Downgrading a Database Back to the Previous Oracle Database Release"

This chapter guides you through the process of downgrading a database that has been upgraded to the new release of the Oracle Database.

Chapter 8, "Data Copying Using Export/Import"

This chapter provides an overview of using the Export and Import utilities to copy data between different releases of the Oracle Database.

Appendix A, "Initialization Parameter and Data Dictionary Changes"

This appendix lists changes to initialization parameters and the data dictionary across different releases of the Oracle Database. This appendix also discusses compatibility issues with certain initialization parameters.

Appendix B, "Migrating from Server Manager to SQL*Plus"

This appendix guides you through the process of modifying your Server Manager line mode scripts for use with SQL*Plus.

Related Documents

For more information, see these Oracle resources:

- *Oracle Database Concepts* for a comprehensive introduction to the concepts and terminology used in this manual
- *Oracle Database Administrator's Guide* for information about administering the Oracle Database
- *Oracle Database SQL Reference* for information on Oracle's SQL commands and functions
- *Oracle Database Utilities* for information about the utilities bundled with the Oracle Database, including Export, Import, and SQL*Loader
- *Oracle Net Services Administrator's Guide* for information about Oracle Net Services

Many of the examples in this book use the sample schemas, which are installed by default when you select the Basic Installation option with an Oracle Database installation. Refer to *Oracle Database Sample Schemas* for information on how these schemas were created and how you can use them yourself.

Printed documentation is available for sale in the Oracle Store at

`http://oraclestore.oracle.com/`

To download free release notes, installation documentation, white papers, or other collateral, please visit the Oracle Technology Network (OTN). You must register online before using OTN; registration is free and can be done at

`http://otn.oracle.com/membership/`

If you already have a username and password for OTN, then you can go directly to the documentation section of the OTN Web site at

`http://otn.oracle.com/documentation/`

Conventions

This section describes the conventions used in the text and code examples of this documentation set. It describes:

- Conventions in Text
- Conventions in Code Examples
- Conventions for Windows Operating Systems

Conventions in Text

We use various conventions in text to help you more quickly identify special terms. The following table describes those conventions and provides examples of their use.

Convention	Meaning	Example
Bold	Bold typeface indicates terms that are defined in the text or terms that appear in a glossary, or both.	When you specify this clause, you create an **index-organized table**.
Italics	Italic typeface indicates book titles or emphasis.	*Oracle Database Concepts* Ensure that the recovery catalog and target database do *not* reside on the same disk.
`UPPERCASE monospace (fixed-width) font`	Uppercase monospace typeface indicates elements supplied by the system. Such elements include parameters, privileges, datatypes, RMAN keywords, SQL keywords, SQL*Plus or utility commands, packages and methods, as well as system-supplied column names, database objects and structures, usernames, and roles.	You can specify this clause only for a `NUMBER` column. You can back up the database by using the `BACKUP` command. Query the `TABLE_NAME` column in the `USER_TABLES` data dictionary view. Use the `DBMS_STATS.GENERATE_STATS` procedure.
`lowercase monospace (fixed-width) font`	Lowercase monospace typeface indicates executable programs, filenames, directory names, and sample user-supplied elements. Such elements include computer and database names, net service names and connect identifiers, user-supplied database objects and structures, column names, packages and classes, usernames and roles, program units, and parameter values. *Note:* Some programmatic elements use a mixture of UPPERCASE and lowercase. Enter these elements as shown.	Enter `sqlplus` to start SQL*Plus. The password is specified in the `orapwd` file. Back up the datafiles and control files in the `/disk1/oracle/dbs` directory. The `department_id`, `department_name`, and `location_id` columns are in the `hr.departments` table. Set the `QUERY_REWRITE_ENABLED` initialization parameter to `true`. Connect as `oe` user. The `JRepUtil` class implements these methods.
`lowercase italic monospace (fixed-width) font`	Lowercase italic monospace font represents placeholders or variables.	You can specify the *`parallel_clause`*. Run *`old_release`*`.SQL` where *`old_release`* refers to the release you installed prior to upgrading.

Conventions in Code Examples

Code examples illustrate SQL, PL/SQL, SQL*Plus, or other command-line statements. They are displayed in a monospace (fixed-width) font and separated from normal text as shown in this example:

```
SELECT username FROM dba_users WHERE username = 'MIGRATE';
```

The following table describes typographic conventions used in code examples and provides examples of their use.

Convention	Meaning	Example
`[ ]`	Anything enclosed in brackets is optional.	`DECIMAL (`*`digits`*` [ , `*`precision`*` ])`
`{ }`	Braces are used for grouping items.	`{ENABLE | DISABLE}`
`|`	A vertical bar represents a choice of two options.	`{ENABLE | DISABLE}` `[COMPRESS | NOCOMPRESS]`
`...`	Ellipsis points mean repetition in syntax descriptions.	`CREATE TABLE ... AS `*`subquery`*`;`
	In addition, ellipsis points can mean an omission in code examples or text.	`SELECT `*`col1`*`, `*`col2`*`, ... , `*`coln`*` FROM employees;`
Other symbols	You must use symbols other than brackets ([ ]), braces ({ }), vertical bars (|), and ellipsis points (...) exactly as shown.	`acctbal NUMBER(11,2);` `acct    CONSTANT NUMBER(4) := 3;`
Italics	Italicized text indicates placeholders or variables for which you must supply particular values.	`CONNECT SYSTEM/`*`system_password`* `DB_NAME = `*`database_name`*

`UPPERCASE`	Uppercase typeface indicates elements supplied by the system. We show these terms in uppercase in order to distinguish them from terms you define. Unless terms appear in brackets, enter them in the order and with the spelling shown. Because these terms are not case sensitive, you can use them in either UPPERCASE or lowercase.	`SELECT last_name, employee_id FROM employees;`
		`SELECT * FROM USER_TABLES;`
		`DROP TABLE hr.employees;`

`lowercase`	Lowercase typeface indicates user-defined programmatic elements, such as names of tables, columns, or files.
	Note: Some programmatic elements use a mixture of UPPERCASE and lowercase. Enter these elements as shown.
		`SELECT last_name, employee_id FROM employees;`
		`sqlplus hr/hr`
		`CREATE USER mjones IDENTIFIED BY ty3MU9;`

Conventions for Windows Operating Systems

The following table describes conventions for Windows operating systems and provides examples of their use.

Convention	Meaning	Example
Choose **Start** > *menu item*	How to start a program.	To start the Database Configuration Assistant, choose **Start** > **Programs** > **Oracle** - *HOME_NAME* > **Configuration and Migration Tools** > **Database Configuration Assistant**.
File and directory names	File and directory names are not case sensitive. The following special characters are not allowed: left angle bracket (<), right angle bracket (>), colon (:), double quotation marks ("), slash (/), pipe (|), and dash (-). The special character backslash (\) is treated as an element separator, even when it appears in quotes. If the filename begins with \\, then Windows assumes it uses the Universal Naming Convention.	c:\winnt"\"system32 is the same as C:\WINNT\SYSTEM32

Convention	Meaning	Example
`C:\>`	Represents the Windows command prompt of the current hard disk drive. The escape character in a command prompt is the caret (^). Your prompt reflects the subdirectory in which you are working. Referred to as the *command prompt* in this manual.	`C:\oracle\oradata>`
Special characters	The backslash (\) special character is sometimes required as an escape character for the double quotation mark (") special character at the Windows command prompt. Parentheses and the single quotation mark (') do not require an escape character. Refer to your Windows operating system documentation for more information on escape and special characters.	`C:\>exp HR/HR TABLES=employees QUERY=\"WHERE job_id='SA_REP' and salary<8000\"`
HOME_NAME	Represents the Oracle home name. The home name can be up to 16 alphanumeric characters. The only special character allowed in the home name is the underscore.	`C:\> net start Oracle`*HOME_NAME*`TNSListener`
ORACLE_HOME and *ORACLE_BASE*	In releases prior to Oracle8i release 8.1.3, when you installed Oracle components, all subdirectories were located under a top level *ORACLE_HOME* directory. The default for Windows NT was `C:\orant`. This release complies with Optimal Flexible Architecture (OFA) guidelines. All subdirectories are not under a top level *ORACLE_HOME* directory. There is a top level directory called *ORACLE_BASE* that by default is `C:\oracle\product\10.1.0`. If you install the latest Oracle release on a computer with no other Oracle software installed, then the default setting for the first Oracle home directory is `C:\oracle\product\10.1.0\db_`*n*, where *n* is the latest Oracle home number. The Oracle home directory is located directly under *ORACLE_BASE*. All directory path examples in this guide follow OFA conventions. Refer to *Oracle Database Installation Guide for Windows* for additional information about OFA compliances and for information about installing Oracle products in non-OFA compliant directories.	Go to the *ORACLE_BASE*`\`*ORACLE_HOME*`\rdbms\admin` directory.

1
Introduction

This chapter provides an overview of the database upgrade process, as well as information about running multiple releases of the Oracle Database.

This chapter covers the following topics:

Overview of the Database Upgrade Process

This section includes an overview of the major steps required to upgrade an existing Oracle Database to the new Oracle Database 10*g* release. These procedures transform an existing Oracle Database system (including associated applications) into an Oracle Database 10*g* system. Oracle Database 10*g* is compatible with all earlier Oracle Database releases. Therefore, databases upgraded using the procedures described in this book can work in the same manner as in earlier releases and, optionally, can leverage new Oracle Database 10*g* functionality.

Careful planning and use of Oracle Database 10*g* tools can ease the process of upgrading a database to the new Oracle Database 10*g* release. Oracle Database 10*g* supports the following methods for upgrading a database:

- Use the Database Upgrade Assistant (DBUA).

  The Database Upgrade Assistant can be launched by the Oracle Universal Installer, depending upon the type of installation that you select, and provides a graphical user interface (GUI) that guides you through the upgrade of a database. During installation, you can choose to not use the Database Upgrade Assistant, instead choosing to launch it as a standalone tool at any time in the future to upgrade a database.

  Note: The Database Upgrade Assistant is the preferred method of upgrading a database; Oracle highly recommends using the Database Upgrade Assistant to upgrade to the new Oracle Database 10*g* release.

- Perform a manual upgrade

A manual upgrade provides a command line upgrade of a database, using SQL scripts and utilities.

- Perform a full or partial export from your database, followed by a full or partial import into a new Oracle Database 10*g* database.

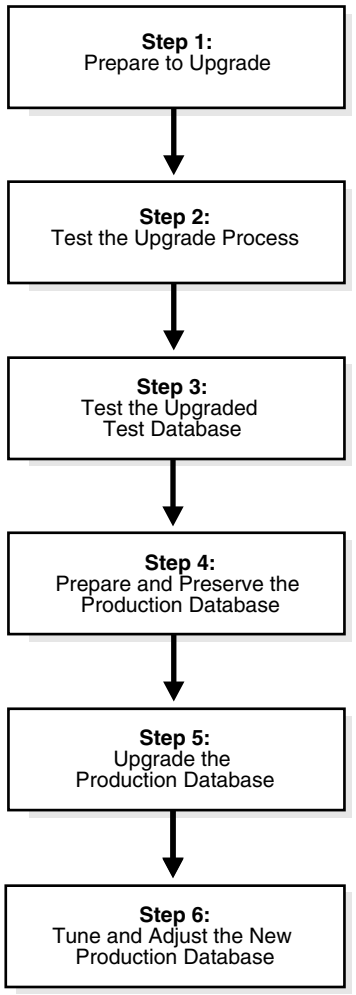
  Export/Import can copy a subset of the data in a database. Export/Import leaves the database unchanged, and makes a copy of the data.

- Copy data from a database into a new Oracle Database 10*g* database using the SQL*Plus `COPY` command or the `AS` clause of the `CREATE TABLE` SQL statement.

  Data copying can copy a subset of the data in a database. Data copying leaves the database unchanged, and makes a copy of the data.

Before you upgrade a database using any of these methods, you should understand the major steps in the upgrade process. These steps are illustrated in Figure 1–1.

Figure 1–1 Major Upgrade Steps



Note: The upgrade steps apply to all operating systems, with the possible exception of a few operating system-specific details identified in your operating system-specific Oracle documentation.

The following sections contain a brief outline of the major steps in the upgrade process. For detailed instructions, refer to the appropriate chapters later in this book.

Step 1: Prepare to Upgrade

- Become familiar with the features of the new Oracle Database 10*g* release.
- Determine the upgrade path to the new Oracle Database 10*g* release.
- Choose an upgrade method.
- Choose an Oracle home directory for the new Oracle Database 10*g* release.
- Prepare a backup strategy.
- Develop a testing plan.

Step 2: Test the Upgrade Process

- Perform a test upgrade using a test database. The test upgrade should be conducted in an environment created for testing and should not interfere with the actual production database.

Step 3: Test the Upgraded Test Database

- Perform the tests you planned in Step 1 on the test database and on the test database that was upgraded to the new Oracle Database 10*g* release.
- Compare results, noting anomalies between test results on the test database and on the upgraded database.
- Investigate ways to correct any anomalies you find and then implement the corrections.
- Repeat Step 1, Step 2, and the first parts of Step 3, as necessary, until the test upgrade is completely successful and works with any required applications.

Chapter 2, "Preparing to Upgrade" provides detailed information about Steps 1 through 3.

Step 4: Prepare and Preserve the Production Database

- Prepare the current production database as appropriate to ensure that the upgrade to the new Oracle Database 10*g* release will be successful.
- Schedule the downtime required for backing up and upgrading the production database.
- Perform a full backup of the current production database.

Step 5: Upgrade the Production Database

- Upgrade the production database to the new Oracle Database 10*g* release.
- After the upgrade, perform a full backup of the production database and perform other post-upgrade tasks.

Chapter 3 describes Steps 4 and 5 when using the Database Upgrade Assistant or when performing a manual upgrade. Chapter 4 describes the backup procedure after the upgrade and other post-upgrade tasks.

Step 6: Tune and Adjust the New Production Database

- Tune the new Oracle Database production database. The new Oracle Database production database should perform as good as, or better than, the database prior to the upgrade. Chapter 4 describes these tuning adjustments.
- Determine which features of the new Oracle Database 10*g* release you want to use and update your applications accordingly.
- Develop new database administration procedures as needed.
- Do not upgrade production users to the new Oracle Database until all applications have been tested and operate properly. Chapter 6 describes considerations for updating applications.

During the upgrade, multi-versioning can be a useful feature because you can keep multiple copies of the same database on one computer. You can use the existing release as your production environment while you test the new release.

Role of the Database Administrator During the Upgrade

Typically, the database administrator (DBA) is responsible for ensuring the success of the upgrade process. The DBA is usually involved in each step of the process, except for steps that involve testing applications on the upgraded database.

The specific DBA duties typically include the following:

- Meeting with everyone involved in the upgrade process and clearly defining their roles
- Performing test upgrades
- Scheduling the test and production upgrades
- Performing backups of the production database
- Completing the upgrade of the production database
- Performing backups of the newly upgraded Oracle Database production database

Role of the Application Developer During the Upgrade

The application developer is responsible for ensuring that applications designed for the current database work correctly with the upgraded Oracle Database. Application developers often test applications against the upgraded Oracle Database and decide which new features of Oracle Database 10*g* should be used.

Before upgrading the production database, the DBA or application developer should install an Oracle Database test database. Then, the application developer can test and modify the applications, if necessary, until they work with their original (or enhanced Oracle Database) functionality.

The following references provide information about identifying differences in the upgraded Oracle Database that could affect particular applications. Application developers can use these differences to guide modifications to existing applications:

- Chapter 5, "Compatibility and Interoperability" describes compatibility and interoperability issues that may result because of differences in releases of Oracle.
- Chapter 6, "Upgrading Your Applications" describes the changes required to enable existing applications to access an Oracle Database and provides guidance for upgrading applications to take advantage of Oracle Database functionality.

- Appendix A, "Initialization Parameter and Data Dictionary Changes" lists obsolete and deprecated initialization parameters and data dictionary views.
- *Oracle Net Services Administrator's Guide* provides instructions for upgrading Net8 to Oracle Net Services.
- *Oracle Database New Features* describes the features available in the new Oracle Database 10*g* release
- *Oracle Real Application Clusters Installation and Configuration Guide* and *Oracle Database SQL Reference* contain descriptions of changes and new Oracle Database functionality.
- *Oracle Database Application Developer's Guide - Fundamentals*, *Oracle Database Application Developer's Guide - Large Objects*, and *Oracle Streams Advanced Queuing User's Guide and Reference* provide information about planning and implementing applications.

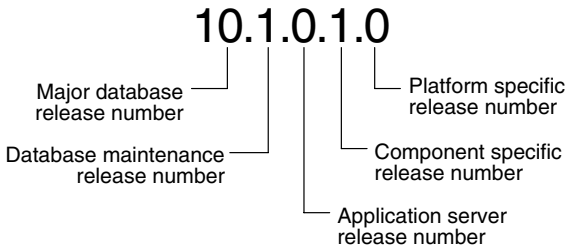
Oracle Database includes features that aid in upgrading existing applications to Oracle Database, for example:

- Oracle Net Services and Net8 support communication between Oracle Database releases.
- The programming interface is unchanged between Oracle Database releases.
- Oracle's backward compatibility accommodates small incompatibilities between different releases.

Oracle Release Numbers

This book describes moving between different **releases** of the Oracle database server. Figure 1–2 describes what each part of a release number represents.

Figure 1–2 Example of an Oracle Release Number



Note: Starting with Oracle9*i* release 9.2, maintenance releases of Oracle are denoted by a change to the second digit of a release number. In previous releases, the third digit indicated a particular maintenance release.

See Also: *Oracle Database Administrator's Guide* for more information about Oracle release numbers

When a statement is made in this book about a major database release number, the statement applies to all releases within that major database release. References to

Oracle Database include release 10.1; references to Oracle9*i* include release 9.0.1 and release 9.2.

Similarly, when a statement is made in this book about a maintenance release, the statement applies to all component-specific and platform-specific releases within that maintenance release. So, a statement about release 9.2 applies to release 9.2.0.1, release 9.2.0.2, and all other platform-specific releases within release 9.2.

Running Multiple Oracle Releases

You can run different releases of the Oracle Database on the same computer at the same time. However, each release can only access a database that is consistent with its release number. For example, if you have Oracle9*i* and Oracle Database 10*g* installed on the same computer, then the Oracle9*i* database server can access Oracle9*i* databases but not Oracle Database 10*g* databases, and the Oracle Database 10*g* database server can access Oracle Database 10*g* databases but not Oracle9*i* databases. The following sections provide general information about running multiple releases of the Oracle Database.

Caution: It is not possible to install release 10.1 products into an existing Oracle home. This functionality was only available for certain previous releases and has not been continued. An Oracle Database release must be installed in a new Oracle home that is separate from previous releases of Oracle. Also, you cannot have more than one release per Oracle home. Oracle recommends that you adopt an Optimal Flexible Architecture (OFA) when creating multiple Oracle homes. See "Using Optimal Flexible Architecture (OFA)" on page 1-7 for more information.

See Also: Your operating system-specific Oracle documentation for more information about running multiple releases of Oracle on your operating system. Restrictions may apply on some operating systems.

Install Databases in Multiple Oracle Homes on the Same Computer

You can install Oracle8, Oracle8*i*, Oracle9*i*, and Oracle Database 10*g* databases in multiple (separate) Oracle homes on the same computer and have Oracle8, Oracle8*i*, Oracle9*i*, and Oracle Database 10*g* clients connecting to any or all of the databases.

Install Databases in Multiple Oracle Homes on Separate Computers

You can install Oracle8, Oracle8*i*, Oracle9*i*, and Oracle Database 10*g* databases in multiple (separate) Oracle homes on separate computers and have Oracle8, Oracle8*i*, Oracle9*i*, and Oracle Database 10*g* clients connecting to any or all of the databases.

Upgrade a Database to the Current Release

You can upgrade an Oracle8, Oracle8*i*, Oracle9*i*, or Oracle Database 10*g* database to the new Oracle Database 10*g* release and have Oracle8, Oracle8*i*, Oracle9*i*, and Oracle Database 10*g* clients connecting to the upgraded database. You cannot upgrade the database in the same Oracle home.

Upgrade Clients to the Current Release

You can upgrade any or all of your Oracle8, Oracle8*i*, Oracle9*i*, or Oracle Database 10*g* clients to the new Oracle Database 10*g* release. You can also upgrade your Oracle8,

Oracle8*i*, Oracle9*i*, or Oracle Database 10*g* database to the new Oracle Database 10*g* release at a later date.

Using Optimal Flexible Architecture (OFA)

Oracle recommends the Optimal Flexible Architecture (OFA) standard for your Oracle Database installations. The OFA standard is a set of configuration guidelines for efficient and reliable Oracle databases that require little maintenance.

OFA provides the following benefits:

- Organizes large amounts of complicated software and data on disk to avoid device bottlenecks and poor performance
- Facilitates routine administrative tasks, such as software and data backup functions, which are often vulnerable to data corruption
- Alleviates switching among multiple Oracle databases
- Adequately manages and administers database growth
- Helps to eliminate fragmentation of free space in the data dictionary, isolates other fragmentation, and minimizes resource contention.

If you are not currently using the OFA standard, then switching to the OFA standard involves modifying your directory structure and relocating your database files.

See Also:

- Your operating system-specific Oracle documentation for more information about OFA
- *Oracle Database Administrator's Guide* for information about relocating database files

Converting Databases to 64-bit Oracle Database Software

If you are installing 64-bit Oracle Database 10*g* software but were previously using a 32-bit Oracle Database installation, then your databases will automatically be converted to 64-bit during the upgrade to Oracle Database 10*g*.

Rolling Upgrades

The term **rolling upgrade** refers to upgrading different databases or different instances of the same database (in a Real Application Clusters environment) one at a time, without stopping the database.

See Also:

- *Oracle High Availability Architecture and Best Practices* for information about RAC rolling upgrades and for information about upgrading with Logical Standby Database
- *Oracle Streams Concepts and Administration* for information about online database upgrade and maintenance with Streams

Deinstalling Options

If you want to deinstall old options when you upgrade to the new Oracle Database release, then use the Oracle Universal Installer to deinstall them. You can deinstall

them before or after you upgrade, but you must use the release of the installer that corresponds with the items you want to remove.

For example, if you are running release 9.0.1 with Oracle Text installed, and you decide that you do not need this option when you upgrade to the new Oracle Database release, then you should deinstall Oracle Text in one of the following ways:

- Before you upgrade to the new Oracle Database release, use the release 9.0.1 Oracle Universal Installer to deinstall Oracle Text. Then, do not install Oracle Text when you install the new Oracle Database release.
- When you upgrade to the new Oracle Database release, install and upgrade Oracle Text. Then, use the Oracle Universal Installer in the new Oracle Database release to deinstall Oracle Text.

Note: After you deinstall an option, extraneous data dictionary tables may remain in the database.

See Also: Your operating system-specific Oracle documentation for information about using the Oracle Universal Installer

2

Preparing to Upgrade

This chapter describes the steps to complete before upgrading a database to the new Oracle Database 10*g* release. This chapter covers in detail Steps 1 through 3 of the upgrade process, which were outlined in "Overview of the Database Upgrade Process" on page 1-1.

This chapter covers the following topics:

- Prepare to Upgrade
- Test the Upgrade Process
- Test the Upgraded Test Database

> **See Also:** *Oracle Net Services Administrator's Guide* for information about upgrade considerations for Oracle Net Services

> **Note:** Some aspects of upgrading are operating system-specific. See your operating system-specific Oracle documentation for additional information about preparing to upgrade.

Prepare to Upgrade

Complete the following tasks to prepare to upgrade:

- Become Familiar with the Features of the New Oracle Database 10g Release
- Determine the Upgrade Path to the New Oracle Database 10g Release
- Choose an Upgrade Method
- Choose an Oracle Home Directory for the New Oracle Database 10g Release
- Prepare a Backup Strategy
- Develop a Testing Plan

Become Familiar with the Features of the New Oracle Database 10*g* Release

Before you plan the upgrade process, become familiar with the features of the new Oracle Database 10*g* release. *Oracle Database New Features* is a good starting point for learning the differences between Oracle Database releases. Also, check specific books in the Oracle Database 10*g* documentation set to find information about new features for a certain component; for example, see *Oracle Real Application Clusters Installation and Configuration Guide* for changes in Real Application Clusters.

Note: Oracle Database 10*g* training classes are an excellent way to learn how to take full advantage of the functionality available with the Oracle Database. Connect to the following Web page for more information:

```
http://education.oracle.com/
```

Determine the Upgrade Path to the New Oracle Database 10*g* Release

The path that you must take to upgrade to the new Oracle Database 10*g* release depends on the release number of your current database. Table 2–1 contains the required upgrade path for each release of the Oracle Database. Use the upgrade path and the specified documentation to upgrade 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	Direct upgrade is *not* supported. Complete the following steps to upgrade to the new Oracle Database 10*g* release: **1.** Upgrade to an intermediate Oracle Database release using the instructions in the intermediate release' s documentation. **2.** Upgrade the intermediate release database to the new Oracle Database 10*g* release using the instructions in Chapter 3, "Upgrading a Database to the New Oracle Database 10g Release".
8.0.6	Direct upgrade is supported. Upgrade to the new Oracle Database 10*g* release using the instructions in Chapter 3, "Upgrading a Database to the New Oracle Database 10g Release".
8.1.5 8.1.6	Direct upgrade is *not* supported. Complete the following steps to upgrade to the new Oracle Database 10*g* release: **1.** Upgrade to an intermediate Oracle Database release using the instructions in the intermediate release's documentation. **2.** Upgrade the intermediate release database to the new Oracle Database 10*g* release using the instructions in Chapter 3, "Upgrading a Database to the New Oracle Database 10g Release".
8.1.7 9.0.1 9.2	Direct upgrade is supported. Upgrade to the new Oracle Database 10*g* release using the instructions in Chapter 3, "Upgrading a Database to the New Oracle Database 10g Release".

If a direct upgrade is not supported from the release number of your database, then you must first upgrade your database to an intermediate Oracle release. The database can then be upgraded from this intermediate release to the new Oracle Database 10*g* release.

Note: Depending on your current release, you may need to upgrade through multiple intermediate releases in order to upgrade to the new Oracle Database 10*g* release.

For example, if your current release is release 8.1.6, then you will need to first upgrade to release 8.1.7 using the instructions in *Oracle8i Migration* for release 8.1.7. The release 8.1.7 database can then be upgraded to the new Oracle Database 10*g* release using the instructions in this book.

Choose an Upgrade Method

Choose one of the upgrade methods outlined in "Overview of the Database Upgrade Process" on page 1-1 to upgrade your database to the new Oracle Database 10*g* release. The following sections describe each of the upgrade methods in detail.

Database Upgrade Assistant

The Database Upgrade Assistant (DBUA) interactively steps you through the upgrade process and configures the database for the new Oracle Database 10*g* release. The Database Upgrade Assistant automates the upgrade process by performing all of the tasks normally performed manually. The Database Upgrade Assistant makes appropriate recommendations for configuration options such as tablespaces and redo logs. You can then act on these recommendations.

For example, the Database Upgrade Assistant recommends sizing information for the new `SYSAUX` tablespace, which is required in Oracle Database 10*g*.

Before the Upgrade The Database Upgrade Assistant performs the following pre-upgrade steps:

- It checks for any invalid user accounts or roles
- It checks for any invalid datatypes
- It checks for any desupported character sets
- It checks for adequate resources, including rollback segments, tablespaces, and free disk space
- It optionally backs up all necessary files

The Database Upgrade Assistant does not begin the upgrade until it completes all of the pre-upgrade steps.

During the Upgrade The Database Upgrade Assistant automatically modifies or creates new required tablespaces, invokes the appropriate upgrade scripts, archives the redo logs, and disables archiving during the upgrade phase.

While the upgrade is running, the Database Upgrade Assistant shows the upgrade progress for each component. The Database Upgrade Assistant writes detailed trace and log files and produces a complete HTML report for later reference. To enhance security, the Database Upgrade Assistant automatically locks new user accounts in the upgraded database. The Database Upgrade Assistant then proceeds to create new configuration files (parameter and listener files) in the new Oracle home.

Real Application Clusters Support The Database Upgrade Assistant is fully compliant with the Real Application Clusters (RAC) environment. In a RAC environment, the Database Upgrade Assistant upgrades all the database and configuration files on all nodes in the cluster.

Support for Silent Mode The Database Upgrade Assistant supports a silent mode of operation where no user interface is presented to the user. Silent mode allows you to use a single command for the upgrade.

Manual Upgrade

A manual upgrade consists of running SQL scripts and utilities from a command line to upgrade a database to the new Oracle Database 10*g* release.

While a manual upgrade gives you finer control over the upgrade process, it is more susceptible to error if any of the upgrade or pre-upgrade steps are either not followed or are performed out of order. The Database Upgrade Assistant performs all necessary pre-upgrade and upgrade steps.

Before the Upgrade When manually upgrading a database, you must perform the following pre-upgrade steps:

- Analyze the database using the Pre-Upgrade Information Tool. The Upgrade Information Tool is a SQL script that ships with the new Oracle Database 10*g* release, and must be run in the environment of the database being upgraded.

  The Upgrade Information Tool displays warnings about possible upgrade issues with the database. It also displays information about required initialization parameters for the new Oracle Database 10*g* release. Before starting up the new Oracle Database 10*g* release, make the necessary adjustments to the database.
- Perform a backup of the database.
- Add free space to any tablespaces in the database that require additional space, and drop and re-create any redo log files whose size is insufficient for the upgrade.
- Adjust the parameter file for the upgrade, removing obsolete initialization parameters and adjusting initialization parameters that might cause upgrade problems.

Depending on the release of the database being upgraded, you may need to perform additional pre-upgrade steps.

After the Upgrade View the status of the upgrade using the Post-Upgrade Status Tool. The Upgrade Status Tool is a SQL script that ships with the new Oracle Database 10*g* release, and must be run in the environment of the new Oracle Database 10*g* release.

Export/Import

Unlike the Database Upgrade Assistant or a manual upgrade, the Export/Import utilities physically copy data from your current database to a new database. The current database's Export utility copies specified parts of the database into an export dump file. Then, the Import utility of the new Oracle Database 10*g* release loads the exported data into a new database. However, the new Oracle Database 10*g* database must already exist before the export dump file can be copied into it.

When importing data from an earlier release, the Oracle Database 10*g* Import utility makes appropriate changes to data definitions as it reads earlier releases' export dump files.

The following sections highlight aspects of Export/Import that may help you to decide whether to use Export/Import to upgrade your database.

Export/Import Effects on Upgraded Databases The Export/Import upgrade method does not change the current database, which enables the database to remain available throughout the upgrade process. However, if a consistent snapshot of the database is required (for data integrity or other purposes), then the database must run in restricted mode or must otherwise be protected from changes during the export procedure. Because the current database can remain available, you can, for example, keep an existing production database running while the new Oracle Database 10*g* database is being built at the same time by Export/Import. During the upgrade, to maintain complete database consistency, changes to the data in the database cannot be permitted without the same changes to the data in the new Oracle Database 10*g* database.

Most importantly, the Export/Import operation results in a completely new database. Although the current database ultimately contains a copy of the specified data, the upgraded database may perform differently from the original database. For example, although Export/Import creates an identical copy of the database, other factors, such as disk placement of data and unset tuning parameters, may cause unexpected performance problems.

Export/Import Benefits Upgrading using Export/Import offers the following benefits:

- Defragments the data - you can compress the imported data to improve performance.
- Restructures the database - you can create new tablespaces or modify existing tables, tablespaces, or partitions to be populated by imported data.
- Enables the copying of specified database objects or users - you can import only the objects, users, and other items that you wish.
- Serves as a backup archive - you can use a full database export as an archive of the current database.

Time Requirements for Export/Import Upgrading an entire database by using Export/Import can take a long time, especially compared to using the Database Upgrade Assistant or performing a manual upgrade. Therefore, you may need to schedule the upgrade during non-peak hours or make provisions for propagating to the new database any changes that are made to the current database during the upgrade.

Data Copying

You can copy data from one Oracle Database to another using database links. For example, you can copy data from one database table to another with the SQL*Plus `COPY` command, or you can create new tables and fill the tables with data by using the `INSERT INTO` statement and the `CREATE TABLE ... AS` statement.

Copying data and Export/Import offer the same advantages for upgrading. Using either method, you can defragment data files and restructure the database by creating new tablespaces or modifying existing tables or tablespaces. In addition, you can copy only specified database objects or users.

Copying data, however, unlike Export/Import, enables the selection of specific rows of tables to be placed into the new database. Copying data is thus a good method for copying only part of a database table. In contrast, using Export/Import, you can copy only entire tables.

Choose an Oracle Home Directory for the New Oracle Database 10*g* Release

You must choose an Oracle home directory for the new Oracle Database 10*g* release that is separate from the Oracle home directory of your current release. You cannot install the new Oracle Database software into the same Oracle home directory as your current release.

Using separate installation directories enables you to keep your existing software installed along with the new Oracle Database software. This method enables you to test the upgrade process on a test database before replacing your production environment entirely.

Prepare a Backup Strategy

The ultimate success of your upgrade depends heavily on the design and execution of an appropriate backup strategy. To develop a backup strategy, consider the following questions:

- How long can the production database remain inoperable before business consequences become intolerable?
- What backup strategy should be used to meet your availability requirements?
- Are backups archived in a safe, offsite location?
- How quickly can backups be restored (including backups in offsite storage)?
- Have recovery procedures been tested successfully?

Your backup strategy should answer all of these questions and include procedures for successfully backing up and recovering your database.

> **See Also:** *Oracle Database Backup and Recovery Basics* for information on database backups

Develop a Testing Plan

You need a series of carefully designed tests to validate all stages of the upgrade process. Executed rigorously and completed successfully, these tests ensure that the process of upgrading the production database is well understood, predictable, and successful. Perform as much testing as possible before upgrading the production database. Do not underestimate the importance of a test program.

The testing plan must include the following types of tests.

Upgrade Testing

Upgrade testing entails planning and testing the upgrade path from your current database to the new Oracle Database, whether you use the Database Upgrade Assistant, perform a manual upgrade, or use Export/Import or other data-copying methods.

Regardless of the upgrade method you choose, you must establish, test, and validate an upgrade plan.

Minimal Testing

Minimal testing entails moving all or part of an application from the current database to the new Oracle Database and running the application without enabling any new database features. Minimal testing is a very limited type of testing that may not reveal potential issues that may appear in a "real-world" production environment. However, minimal testing will immediately reveal any application startup or invocation problems.

Functional Testing

Functional testing is a set of tests in which new and existing functionality of the system are tested after the upgrade. Functional testing includes all database, networking, and application components. The objective of functional testing is to verify that each component of the system functions as it did before upgrading and to verify that new functions are working properly.

Integration Testing

Integration testing examines the interaction of each component of the system. Consider the following factors when you plan your integration testing:

- Pro*C/C++ applications running against a new Oracle Database instance should be tested to ensure that there are no problems with the new software.
- Graphical user interfaces should be tested with other components.
- Subtle changes in the new Oracle Database, such as datatypes, data in the data dictionary (additional rows in the data dictionary, object type changes, and so on) can have an effect all the way up to the front-end application, regardless of whether or not the application is directly connected to a new Oracle Database instance.
- If the connection between two components involves Net8 or Oracle Net Services, then those connections should also be tested and stress tested.

Performance Testing

Performance testing of the new Oracle Database compares the performance of various SQL statements in the new Oracle Database with the statements' performance in the current database. Before upgrading, you should understand the performance profile of the application under the current database. Specifically, you should understand the calls the application makes to the database kernel.

For example, if you are using Real Application Clusters, and you want to measure the performance gains realized from using cache fusion when you upgrade to the new Oracle Database 10*g* release, then make sure you record your system's statistics before upgrading. For cache fusion, record the statistics from the `V$SYSSTAT` and `V$INSTANCE_CACHE_TRANSFER` views. Doing so enables you to compare pre-cache fusion and post-cache fusion performance statistics.

For best results, run the SQL scripts `utlbstat.sql` and `utlestat.sql` to collect `V$SYSSTAT` statistics for a specific period. Use a collection timeframe that most consistently reflects peak production loads with consistent transaction activity levels. To obtain data from `V$LOCK_ACTIVITY` and `V$LOCK_CLASS_PING`, use a `SELECT *` statement at the beginning and end of the statistics collection period. Repeat this process after cache fusion is running on the new Oracle Database release and evaluate your system's performance as described in *Oracle Real Application Clusters Deployment and Performance Guide*.

> **See Also:** *Oracle Database Performance Tuning Guide* for information about tuning. To thoroughly understand the application's performance profile under the source database, enable the SQL trace facility and profile with TKPROF.

Volume and Load Stress Testing

Volume and load stress testing tests the entire upgraded database under high volume and loads. Volume describes the amount of data being manipulated. Load describes the level of concurrent demand on the system. The objective of volume and load testing is to emulate how a production system might behave under various volumes and loads.

Volume and load stress testing is crucial, but is commonly overlooked. Oracle has found that customers often do not conduct any kind of volume or load stress testing. Instead, customers often rely on benchmarks that do not characterize business applications. Benchmarks of the application should be conducted to uncover problems

relating to functionality, performance, and integration, but they cannot replace volume and load stress testing.

After you upgrade the database, you should test the data to ensure that all data is accessible and that the applications function properly. You should also determine whether any database tuning is necessary. If possible, you should automate these testing procedures.

The testing plan should reflect the work performed at the site. You should test the functionality and performance of all applications on the production databases. Gather performance statistics for both normal and peak usage.

Specific Pre-Upgrade and Post-Upgrade Tests

Include the following tests in your testing plan:

- Timing tests
- Data dictionary growth observations
- Database resource usage observations, such as rollback and temporary segment usage

Collecting this information will help you compare the current database with the new Oracle Database.

Use EXPLAIN PLAN on both the previous and new databases to determine the execution plan Oracle follows to execute each SQL statement. Use the `INTO` clause to save this information in tables.

After upgrading, you can compare the execution plans of the new Oracle Database with the execution plans of the current database. If there is a difference, then execute the statement on the new Oracle Database and compare the performance with the performance of the statement executed on the current database.

> **See Also:** *Oracle Database Performance Tuning Guide* for more information about EXPLAIN PLAN.

Test the Upgrade Process

Create a test environment that will not interfere with the current production database. Your test environment will depend on the upgrade method you have chosen:

- If you plan to use the Database Upgrade Assistant or perform a manual upgrade, then create a test version (typically a subset) of the current production database to test the upgrade.
- If you plan to use Export/Import, then export and import small test pieces of the current production database.

Practice upgrading the database using the test environment. The best upgrade test, if possible, is performed on an exact copy of the database to be upgraded, rather than on a downsized copy or test data.

> **Caution:** Do not upgrade the actual production database until after you successfully upgrade a test subset of this database and test it with applications, as described in the next step.

Make sure you upgrade any OCI and precompiler applications that you plan to use with your new Oracle Database. Then, you can test these applications on a sample

database before upgrading your current production database. See "Upgrading Precompiler and OCI Applications" on page 6-2 for more information.

Test the Upgraded Test Database

Perform the planned tests on the current database and on the test database that you upgraded to the new Oracle Database release. Compare the results, noting anomalies. Repeat the test upgrade as many times as necessary.

Test the newly upgraded Oracle Database test database with existing applications to verify that they operate properly with a new Oracle Database. You also might test enhanced functionality by adding features that use the available Oracle Database functionality. However, first make sure that the applications operate in the same manner as they did in the current database.

> **See Also:** Chapter 6, "Upgrading Your Applications" for more information on using applications with Oracle Database

3

Upgrading a Database to the New Oracle Database 10*g* Release

This chapter guides you through the process of upgrading a database to the new Oracle Database 10*g* release. This chapter covers the following topics:

- System Considerations and Requirements
- Install the Release 10.1 Oracle Software
- Upgrade the Database Using the Database Upgrade Assistant
- Upgrade the Database Manually

> **See Also:** Some aspects of upgrading are operating system-specific. See your operating system-specific Oracle documentation for additional instructions about upgrading on your operating system.

System Considerations and Requirements

The following sections discuss system considerations and requirements.

Upgrading a Cluster Database

If you are upgrading a cluster database, then most of the actions described in this chapter should be performed on only one node of the system. So, perform the actions described in this chapter on only one node unless instructed otherwise in a particular step.

Gather Optimizer Statistics Before the Upgrade

When upgrading to Oracle Database 10*g*, optimizer statistics will be collected for dictionary tables that lack statistics. This statistics collection could be time consuming for databases with a large number of dictionary tables, but it will only occur for those tables that lack statistics or are significantly changed during the upgrade.

For databases that are upgraded from Oracle9*i*, it is possible to decrease the downtime during the upgrade by collecting statistics for the dictionary prior to the upgrade. The following two scripts collect statistics for dictionary objects in Oracle9*i*.

This process should be tested on a test database just like any other aspect of the upgrade. Also, some schemas referenced in these scripts may not exist if some database components have not been installed.

```
--
-- This script collect stats for system component schemas.
-- The stats collection may give error if a particular component
```

```
-- schema does not exist in the database. This can happen
-- if a component is not installed or if it is invalid.
--
-- This script must be run connected AS SYSDBA using SQL*Plus.
--

spool gdict

grant analyze any to sys;

exec dbms_stats.gather_schema_stats('WMSYS',options=>'GATHER', estimate_percent =>
DBMS_STATS.AUTO_SAMPLE_SIZE, method_opt => 'FOR ALL COLUMNS SIZE AUTO', cascade =>
TRUE);

exec dbms_stats.gather_schema_stats('MDSYS',options=>'GATHER', estimate_percent =>
DBMS_STATS.AUTO_SAMPLE_SIZE, method_opt => 'FOR ALL COLUMNS SIZE AUTO', cascade =>
TRUE);

exec dbms_stats.gather_schema_stats('CTXSYS',options=>'GATHER', estimate_percent
=> DBMS_STATS.AUTO_SAMPLE_SIZE, method_opt => 'FOR ALL COLUMNS SIZE AUTO', cascade
=> TRUE);

exec dbms_stats.gather_schema_stats('XDB',options=>'GATHER', estimate_percent =>
DBMS_STATS.AUTO_SAMPLE_SIZE, method_opt => 'FOR ALL COLUMNS SIZE AUTO', cascade =>
TRUE);

exec dbms_stats.gather_schema_stats('WKSYS',options=>'GATHER', estimate_percent =>
DBMS_STATS.AUTO_SAMPLE_SIZE, method_opt => 'FOR ALL COLUMNS SIZE AUTO', cascade =>
TRUE);

exec dbms_stats.gather_schema_stats('LBACSYS',options=>'GATHER', estimate_percent
=> DBMS_STATS.AUTO_SAMPLE_SIZE, method_opt => 'FOR ALL COLUMNS SIZE AUTO', cascade
=> TRUE);

exec dbms_stats.gather_schema_stats('OLAPSYS',options=>'GATHER', estimate_percent
=> DBMS_STATS.AUTO_SAMPLE_SIZE, method_opt => 'FOR ALL COLUMNS SIZE AUTO', cascade
=> TRUE);

exec dbms_stats.gather_schema_stats('DMSYS',options=>'GATHER', estimate_percent =>
DBMS_STATS.AUTO_SAMPLE_SIZE, method_opt => 'FOR ALL COLUMNS SIZE AUTO', cascade =>
TRUE);

exec dbms_stats.gather_schema_stats('ODM',options=>'GATHER', estimate_percent =>
DBMS_STATS.AUTO_SAMPLE_SIZE, method_opt => 'FOR ALL COLUMNS SIZE AUTO', cascade =>
TRUE);

exec dbms_stats.gather_schema_stats('ORDSYS',options=>'GATHER', estimate_percent
=> DBMS_STATS.AUTO_SAMPLE_SIZE, method_opt => 'FOR ALL COLUMNS SIZE AUTO', cascade
=> TRUE);

exec dbms_stats.gather_schema_stats('ORDPLUGINS',options=>'GATHER', estimate_
percent => DBMS_STATS.AUTO_SAMPLE_SIZE, method_opt => 'FOR ALL COLUMNS SIZE AUTO',
cascade => TRUE);

exec dbms_stats.gather_schema_stats('SI_INFORMTN_SCHEMA',options=>'GATHER',
estimate_percent => DBMS_STATS.AUTO_SAMPLE_SIZE, method_opt => 'FOR ALL COLUMNS
SIZE AUTO', cascade => TRUE);

exec dbms_stats.gather_schema_stats('OUTLN',options=>'GATHER', estimate_percent =>
DBMS_STATS.AUTO_SAMPLE_SIZE, method_opt => 'FOR ALL COLUMNS SIZE AUTO', cascade =>
```

```
TRUE);

exec dbms_stats.gather_schema_stats('DBSNMP',options=>'GATHER', estimate_percent
=> DBMS_STATS.AUTO_SAMPLE_SIZE, method_opt => 'FOR ALL COLUMNS SIZE AUTO', cascade
=> TRUE);

exec dbms_stats.gather_schema_stats('SYSTEM',options=>'GATHER', estimate_percent
=> DBMS_STATS.AUTO_SAMPLE_SIZE, method_opt => 'FOR ALL COLUMNS SIZE AUTO', cascade
=> TRUE);

exec dbms_stats.gather_schema_stats('SYS',options=>'GATHER', estimate_percent =>
DBMS_STATS.AUTO_SAMPLE_SIZE, method_opt => 'FOR ALL COLUMNS SIZE AUTO', cascade =>
TRUE);


spool off

--
-- This script creates the stats table, 'dictstattab' and
-- exports the stats for the RDBMS component schemas into it.
-- The export will give error if a particular component
-- schema does not exist in the database. This can happen
-- if a component is not installed or if it is invalid.
--
-- This will be useful incase you want to import the stats back
-- example:
-- Following stmt imports the stats for SYS schema after
-- deleting the existing stats.
-- exec dbms_stats.delete_schema_stats('SYS');
-- exec dbms_stats.import_schema_stats('SYS','dictstattab');

--
-- This script must be run connected AS SYSDBA using SQL*Plus.
--

spool sdict

grant analyze any to sys;

exec dbms_stats.create_stat_table('SYS','dictstattab');

exec dbms_stats.export_schema_stats('WMSYS','dictstattab',statown => 'SYS');
exec dbms_stats.export_schema_stats('MDSYS','dictstattab',statown => 'SYS');
exec dbms_stats.export_schema_stats('CTXSYS','dictstattab',statown => 'SYS');
exec dbms_stats.export_schema_stats('XDB','dictstattab',statown => 'SYS');
exec dbms_stats.export_schema_stats('WKSYS','dictstattab',statown => 'SYS');
exec dbms_stats.export_schema_stats('LBACSYS','dictstattab',statown => 'SYS');
exec dbms_stats.export_schema_stats('OLAPSYS','dictstattab',statown => 'SYS');
exec dbms_stats.export_schema_stats('DMSYS','dictstattab',statown => 'SYS');
exec dbms_stats.export_schema_stats('ODM','dictstattab',statown => 'SYS');
exec dbms_stats.export_schema_stats('ORDSYS','dictstattab',statown => 'SYS');
exec dbms_stats.export_schema_stats('ORDPLUGINS','dictstattab',statown => 'SYS');
exec dbms_stats.export_schema_stats('SI_INFORMTN_SCHEMA','dictstattab',statown =>
'SYS');
exec dbms_stats.export_schema_stats('OUTLN','dictstattab',statown => 'SYS');
exec dbms_stats.export_schema_stats('DBSNMP','dictstattab',statown => 'SYS');
exec dbms_stats.export_schema_stats('SYSTEM','dictstattab',statown => 'SYS');
exec dbms_stats.export_schema_stats('SYS','dictstattab',statown => 'SYS');

spool off
```

Upgrading Your Operating System

If required, upgrade your operating system before continuing with your database upgrade.

See Also:

- The Oracle Installation Guide for your platform to determine whether you need to upgrade your operating system

  Your operating system-specific Oracle documentation for information on how to perform an operating system upgrade

Migrating Data to a Different Operating System

When using the Database Upgrade Assistant or when performing a manual upgrade, you *cannot* migrate data in a database on one operating system to a database on another operating system. For example, you cannot migrate data in an Oracle9*i* database on Solaris to an Oracle Database 10*g* database on Windows 2000 using the Database Upgrade Assistant. However, you normally can use Export/Import to migrate data between databases on different operating systems.

Install the Release 10.1 Oracle Software

Complete the following steps to install the release 10.1 software:

1. If your operating system is UNIX, then make sure you are logged in as a user with write permission to the Oracle home and Oracle base directories, as well as all of their subdirectories.
2. Follow the instructions in your Oracle operating system-specific documentation to prepare for installation and start the Oracle Universal Installer.

   If you are upgrading a cluster database, then see *Oracle Real Application Clusters Installation and Configuration Guide* for additional installation instructions.
3. At the Welcome screen of the Oracle Universal Installer, click Next. The File Locations screen appears.

   If you need help at any screen or want to consult more documentation about the Oracle Universal Installer, then click the Help button to open the online help.
4. At the File Locations screen, complete the following steps:
   a. Do not change the text in the Source field. This is the location of files for installation.
   b. Enter the name of a new Oracle home in the Destination Name field.
   c. Enter the complete path of the Oracle home directory where you want to install the new release in the Destination Path field. Click the Browse button to navigate to the directory.

      Note: You must install the new Oracle Database release in a new Oracle home that is separate from the old release.

   d. Click Next.

The Installation Types screen appears.

5. At the Installation Types screen, complete the following steps:
   a. Select Enterprise Edition, Standard Edition, or Custom Installation.
   b. Click Next.

   If you chose Enterprise Edition or Standard Edition, then the Database Configuration screen appears.

   If you chose Custom Installation, then the Available Product Components screen appears.
6. If the Database Configuration screen appears, then complete the following steps:
   a. Select Software Only.
   b. Click Next.

   Note: Normally, you should not install a starter database if you are upgrading an existing database.

7. If the Available Product Components screen appears, then complete the following steps:
   a. Choose the product components you want to install.
   b. Click Next.

      Make sure you install all of the options you installed with the previous database, assuming you do not want to discontinue use of a particular option. For example, if you installed Oracle Text in the previous database, then you should install Oracle Text in the new Oracle Database.
8. If you are installing Real Application Clusters, then, at the Cluster Node Selection screen, select the nodes onto which you want the software installed. Then, click Next.
9. At the Create Database screen, select the No option, indicating that you do not want to create a database because you are upgrading an existing database. Then, click Next.
10. Respond to the remaining screens that enable you to specify your custom installation settings, until you reach the Upgrading an Existing Database screen.
11. At the Upgrading an Existing Database screen, complete the following steps:
    a. To upgrade a database using the Database Upgrade Assistant, select the Upgrade an Existing Database check box.

       To upgrade a database manually, or to start the Database Upgrade Assistant independently after installation is complete, do not select the Upgrade an Existing Database check box.
    b. Click Next.
12. At the Summary screen, make sure all of the settings and choices are correct for your installation. Then, click Install. The Oracle Universal Installer performs the installation.

When installation is complete, one or more assistants may be started. If you chose to run the Database Upgrade Assistant during installation, then you are ready to proceed

with the upgrade when the Database Upgrade Assistant is started. See "Upgrade the Database" on page 3-8.

When installation has completed successfully, click the Exit button to close the Oracle Universal Installer.

Upgrade the Database Using the Database Upgrade Assistant

The following sections guide you through the process of upgrading a database using the Database Upgrade Assistant:

- Starting the Database Upgrade Assistant
- Database Upgrade Assistant Command Line Options
- Upgrade the Database
- Using the Database Upgrade Assistant in Silent Mode

Starting the Database Upgrade Assistant

If you installed the new Oracle Database 10*g* release and specified that you are upgrading an existing database, then the Database Upgrade Assistant is started automatically. See "Upgrade the Database" on page 3-8. However, if you did not specify that you are upgrading an existing database, then you can start the Database Upgrade Assistant independently after installation is complete.

Complete the following steps to start the Database Upgrade Assistant:

1. In the environment of the new Oracle Database 10*g* release, start the Database Upgrade Assistant.

   On UNIX platforms, enter the following command at a system prompt:

   ```
   dbua
   ```

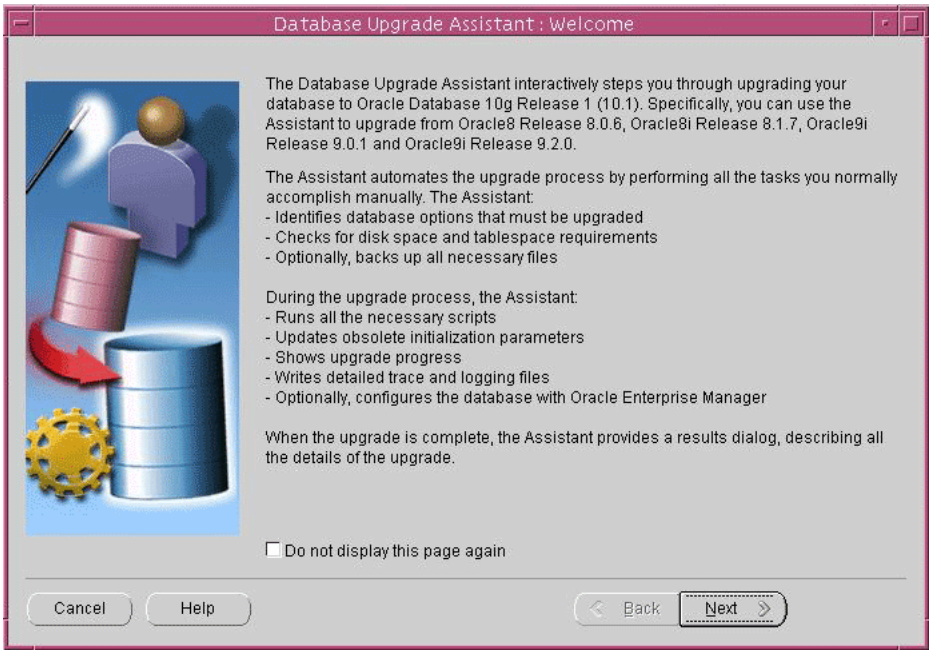
   Note: The `dbua` executable is usually located in *ORACLE_HOME*`/bin`.

   On Windows operating systems, choose:

   ```
   Start > Programs > Oracle - HOME_NAME > Configuration and Migration Tools >
   Database Upgrade Assistant
   ```

When the Database Upgrade Assistant starts, its Welcome screen appears. Figure 3–1 shows the Welcome screen of the Database Upgrade Assistant.

Figure 3–1 Welcome Screen of the Database Upgrade Assistant



Database Upgrade Assistant Command Line Options

The Database Upgrade Assistant supports several command line options. You can specify all valid options from the command line using the following syntax:

```
dbua [ -silent ] [ -dbName SID ]
 [ -disableUpgradeScriptLogging ] [ -backupLocation directory ]
 [ -postUpgradeScripts script [, script ] ... ]
 [ -initParam parameter=value [, parameter=value ] ... ]
 [ -emConfiguration { LOCAL | CENTRAL | NOBACKUP | NOEMAIL | NONE }
 -dbsnmpPassword password -sysmanPassword password
 [ -hostUserName hostname -hostUserPassword password
 -backupSchedule hh:mm
 ]
 [ -smtpServer server_name -emailAddress address ]
 -centralAgent location
 ]
 [ -recoveryAreaDestination directory ] [ -h | -help ]
```

Table 3–1 describes the various options and their parameters that are supported by the Database Upgrade Assistant.

Table 3–1 Database Upgrade Assistant Command Line options

Option	Description
`-silent`	Specifies that the Database Upgrade Assistant should operate in silent mode. See "Using the Database Upgrade Assistant in Silent Mode" on page 3-15.
`-dbName` *SID*	Specifies the system identifier (SID) of the database to upgrade

Table 3–1 (Cont.) Database Upgrade Assistant Command Line options

Option	Description
`-disableUpgradeScriptLogging`	This option disables the detailed log generation for running SQL scripts during the upgrade process. This is enabled by default. To enable the log generation, do not specify this option.
`-backupLocation` *directory*	Specifies a directory to back up your database before the upgrade starts
`-postUpgradeScripts` *script* `[,` *script* `] ...`	Specifies a comma-separated list of SQL scripts. Specify complete pathnames. The scripts will be executed at the end of the upgrade.
`-initParam` *parameter*`=`*value* `[,` *parameter*`=`*value* `] ...`	Specifies a comma-separated list of initialization parameter values of the form *name=value*
`-emConfiguration { LOCAL | CENTRAL | NOBACKUP | NOEMAIL | NONE }`	Specifies Enterprise Manager management options: ■ `LOCAL` - Database is locally managed by Enterprise Manager ■ `CENTRAL` - Database is centrally managed by Enterprise Manager ■ `NOBACKUP` - Automatic daily backups of the database are not enabled ■ `NOEMAIL` - E-mail notifications are not enabled ■ `NONE` - Database is not managed by Enterprise Manager
`-dbsnmpPassword` *password*	Specifies the `DBSNMP` user password
`-sysmanPassword` *password*	Specifies the `SYSMAN` user password
`-hostUserName` *hostname*	Specifies the host user name for the Enterprise Manager backup job
`-hostUserPassword` *password*	Specifies the host user password for the Enterprise Manager backup job
`-backupSchedule` *hh*`:`*mm*	Specifies the daily backup schedule in the form *hh:mm* (hours and minutes)
`-smtpServer` *server_name*	Specifies the outgoing mail (SMTP) server for E-mail notifications
`-emailAddress` *address*	Specifies the E-mail address for E-mail notifications
`-centralAgent` *location*	Specifies the Enterprise Manager central agent location
`-recoveryAreaDestination` *directory*	Specifies the destination directory for all recovery files
`-h | -help`	Displays usage help for the Database Upgrade Assistant

For example, the following command selects a database named `ORCL` for an upgrade:

```
dbua -dbName ORCL
```

Upgrade the Database

When the Welcome screen of the Database Upgrade Assistant appears (Figure 3–1 on page 3-7), you are ready to proceed with the upgrade. Complete the following steps to upgrade the database:

1. At the Welcome screen of the Database Upgrade Assistant, make sure the database being upgraded meets the specified conditions. Then, click Next.

   If you need help at any screen or want to consult more documentation about the Database Upgrade Assistant, then click the Help button to open the online help.

2. At the Selecting a Database Instance screen, select the database you want to upgrade from the Available Databases table. Then, click Next.

   You can select only one database at a time. If you are running the Database Upgrade Assistant from a user account that does not have `SYSDBA` privileges, then you must enter the user name and password credentials to enable `SYSDBA` privileges for the selected database.

   The database you select must already be started. The Database Upgrade Assistant analyzes the database, performing pre-upgrade checks and displaying warnings as necessary:

   - It checks for any redo log files whose size is less than 4 MB. If such files are found, then the Database Upgrade Assistant gives the option to drop/create new redo log files.
   - It checks the parameter file for any obsolete or deprecated initialization parameters

3. At the Creating the SYSAUX Tablespace screen, specify the attributes for the `SYSAUX` tablespace, which is added automatically to all new Oracle Database 10*g* databases you create. Then, click Next.

   See Also: *Oracle Database Administrator's Guide* for more information about the `SYSAUX` tablespace

   Many of the attributes of the `SYSAUX` tablespace are set automatically and cannot be modified. For example, the `SYSAUX` tablespace is set to use Automatic Segment-Space Management. However, you can specify the location of the datafile, the default size of the `SYSAUX` tablespace, and its autoextend attributes.

   Note: If you specify an existing datafile for the `SYSAUX` tablespace, then you must select Reuse Existing File Name. Otherwise, the Database Upgrade Assistant alerts you to the fact that the file already exists.

4. At the Recompiling Invalid Objects screen, decide whether you want the Database Upgrade Assistant to recompile all invalid PL/SQL modules after the upgrade is complete. Then, click Next.

   When you upgrade a database to the new Oracle Database 10*g* release, many of the PL/SQL modules in your database will become invalid. As a result, all existing PL/SQL modules in an `INVALID` state must be recompiled, such as packages, procedures, types, and so on.

   By default, the Oracle Database recompiles invalid PL/SQL modules as they are used. For example, if an invalid PL/SQL module is called, it will first be recompiled before it is actually executed. The time it takes to recompile the module can result in poor performance as you begin to use your newly upgraded database.

   To eliminate these performance issues, select Recompile invalid objects at the end of upgrade. When you select this option, the Database Upgrade Assistant

recompiles all the invalid PL/SQL modules immediately after the upgrade is performed. This will ensure that you will not experience any performance issues later, as you begin using your newly upgraded database.

> **Note:** Selecting Recompile invalid objects at the end of upgrade is equivalent to running the *ORACLE_HOME*`/rdbms/admin/utlrp.sql` script, which is used to recompile stored PL/SQL and Java code.

The task of recompiling all the invalid PL/SQL modules in your database can take a significant amount of time and increase the time it takes to complete your database upgrade. If you have multiple CPUs, then you can reduce the time it takes to perform this task by taking advantage of parallel processing on your available CPUs. If you have multiple CPUs available, then the Database Upgrade Assistant automatically adds an additional section to the Recompile Invalid Objects screen and automatically determines the number of CPUs you have available.

The Database Upgrade Assistant also provides a recommended degree of parallelism, which determines how many parallel processes are used to recompile your invalid PL/SQL modules. Specifically, the Database Upgrade Assistant sets the degree of parallelism to one less than the number of CPUs you have available. For example, if you have three CPUs available for processing, then the Database Upgrade Assistant selects 2 from the Degree of Parallelism menu. You can adjust this default value by selecting a new value from the Degree of Parallelism menu.

5. At the Choosing a Database Backup Procedure screen, specify whether or not you want the Database Upgrade Assistant to back up your database for you. Then, click Next. If you choose not to use the Database Upgrade Assistant for your backup, then Oracle assumes you have already backed up your database using your own backup procedures.

   > **Note:** Oracle strongly recommends that you back up your database before the upgrade. If errors occur during the upgrade, then you may need to restore the database from the backup.

   If you use the Database Upgrade Assistant to back up your database, then the Database Upgrade Assistant will make a copy of all your database files in the directory you specify in the Backup Directory field. The Database Upgrade Assistant will perform this cold backup automatically after it shuts down the database and before it begins performing the upgrade procedure. The cold backup will not compress your database files and the backup directory must be a valid file system path. You cannot specify a raw device for the cold backup files.

   In addition, the Database Upgrade Assistant creates a batch file in the specified directory. You can use this batch file to restore the database files:

   - On Windows operating systems, the file is called *db_name*`_restore.bat`.
   - On UNIX platforms, the file is called *db_name*`_restore.sh`.

6. At the Management Options screen, you have the option of setting up your database so it can be managed with Enterprise Manager. Enterprise Manager provides Web-based management tools for managing individual database instances, as well as central management tools for managing your entire Oracle

environment, including multiple databases, hosts, application servers, and other components of your network.

a. When you run the Database Upgrade Assistant, the assistant checks to see if the Oracle Management Agent has been installed on the host computer. If the assistant locates an Oracle Management Agent, select the Grid Control option and select an Oracle Management Service from the drop-down list. When you finish installing the Oracle Database, the database will automatically be available as a managed target within the Oracle Enterprise Manager Grid Control.

b. If you are not centrally managing your Oracle environment, you can still use Enterprise Manager to manage your database. When you install an Oracle Database, you automatically install the Oracle Enterprise Manager Database Control, which provides Web-based features for monitoring and administering the single-instance or cluster database you are installing.

To configure the database so it can be managed with the Oracle Enterprise Manager Database Control, select the Database Control option.

c. When you select the Database Control management option, you can configure Enterprise Manager so that E-mail notifications will be enabled immediately upon installation.

Select Enable Email Notifications if you want the SYSMAN user (the default Super Administrator and owner of the Management Repository schema) to receive email notification when a metric for a specified condition reaches a critical or warning threshold. For example, Enterprise Manager can send an email when a target goes down or when there are database space usage problems.

d. If you select the Database Control management option, you can also enable automatic daily backups of your entire database.

Select Enable Daily Backups to use the Oracle-suggested backup strategy to back up your entire database with a minimum amount of configuration. Later, you can use Enterprise Manager to customize your backup strategy further.

When you select this option, Enterprise Manager will be configured to back up your database, based on the scheduled start time you enter on this page, immediately after you finish installing the Oracle Database. Enterprise Manager will back up the database to the Flash Recovery Area that you specify later on the Recovery Configuration screen of the Database Upgrade Assistant.

e. After you have made your choices, click **Next**.

7. At the Database Credentials screen, secure your database with passwords for the Enterprise Manager accounts. You can set a single password, which will be applied to each of the listed Enterprise Manager user accounts, or enhance the security of the accounts by providing unique passwords for each user.

8. At the Recovery Configuration screen, specify a flash recovery area and enable archiving. When you are managing your database, it is important to configure the database so you can recover your data in the event of a system failure.

The Flash Recovery Area can be used to recover data that would otherwise be lost during a system failure; this location is also used by Enterprise Manager if you have enabled local management and daily backups on the Management Options screen shown previously in the Database Upgrade Assistant.

9. At the Network Configuration for the database screen, you have two tabs:

The Listeners tab is displayed if you have more than one listener in the release 10.1 Oracle home. Select the listeners in the release 10.1 Oracle home for which you would like to register the upgraded database.

The Directory Service tab shows up if you have directory services configured in the release 10.1 Oracle home. You can select to either register or not register the upgraded database with the directory service.

10. At the Summary screen, make sure all of the specifications are correct. If anything is incorrect, then click Back until you can correct the specification. If everything is correct, then click Finish.

    The Database Upgrade Assistant lists the initialization parameters that will be set for the database during the upgrade. The `COMPATIBLE` initialization parameter will be set to at least `9.2.0`.

    See Also: Chapter 5, "Compatibility and Interoperability" for information about setting the `COMPATIBLE` initialization parameter after the upgrade

11. A Progress dialog appears and the Database Upgrade Assistant begins to perform the upgrade.

    You may encounter error messages with Ignore and Abort choices. If other errors appear, then you must address them accordingly. If an error is severe and cannot be handled during the upgrade, then you have the following choices:

    - If Ignore is presented as a choice in the message, then clicking the button will ignore the error and proceed with the upgrade. The errors ignored are logged and shown later in the summary.

      This causes the Database Upgrade Assistant to display the step as skipped and move on to the next step in the upgrade, ignoring this and any dependent steps. After the upgrade is complete, you can fix the problem, restart the Database Upgrade Assistant, and complete the skipped steps.

    - If Ignore is not presented as a choice in the message, then you need to abort the process by clicking the Abort button.

      This will abort the upgrade process. The Database Upgrade Assistant prompts you to restore the database if the database backup was taken by the Database Upgrade Assistant.

      After the database has been restored, you need to correct the cause of the error and restart the Database Upgrade Assistant to perform the upgrade again.

      If you do not want to restore the database, then the Database Upgrade Assistant leaves the database in its present state so that you can proceed with a manual upgrade.

    After the upgrade has completed, the following message is displayed in the Progress dialog:

    ```
    Upgrade has been completed. Click the "OK" button to see the results of the
    upgrade.
    ```

    Click the OK button.

12. At the Checking Upgrade Results screen, you can examine the results of the upgrade, manage the passwords in the upgraded database, and, if necessary, restore the original database settings.

The upgrade results summary includes a description of the original and upgraded databases and changes made to the initialization paremeters. The upgrade results also include a Step Execution Summary that describes the steps performed during the database upgrade. For each step in the process, the summary provides the step name, the log file for the step, and the status. In some cases, you can click the status to display details about the execution step. The Step Execution Summary also includes the directory where the various log files are stored after the upgrade. You can examine any of these log files to obtain more details about the upgrade process.

> **Note:** An HTML version of the Upgrade Results is also saved in the log files directory.

The Password Management section of the screen allows you to unlock and set passwords for various users in the newly upgraded database. Click Configure Database Passwords to display the Password Management dialog box. The Password Management dialog box allows you to change the default password for a user after you upgrade the database. For security reasons, all users are locked except for the following users:

- `SYS`
- `SYSTEM`

If you have enabled Local Management with Enterprise Manager, then the `SYSMAN` and `DBSNMP` accounts are also unlocked. These accounts provide Enterprise Manager with access to the database so it can gather monitoring data and so you can perform administration tasks with Enterprise Manager.

If you have enabled Central Management with Enterprise Manager, then the `DBSNMP` account is unlocked, as well as the `SYS` and `SYSTEM` user accounts.

> **Note:** To prevent unauthorized use of the database, Oracle recommends that you change all user passwords immediately after you upgrade your database.

If you are not satisfied with the upgrade results, then click Restore. Depending on the method you used to back up your database, the Restore operation performs one of two tasks:

- If you used the Database Upgrade Assistant to back up your database, then clicking Restore will restore the original database and the original database settings from the backup.
- If you used your own backup procedure to back up the database, then clicking Restore will restore only the original database settings. To restore the database itself, you must restore the backup you created with your own backup utilities.

If you are satisfied with the upgrade results, then click Exit to quit the Database Upgrade Assistant and use your newly upgraded database. The Database Upgrade Assistant removes the entry of the upgraded database from the old `listener.ora` file and reloads the listener of the old database.

a. The Database Upgrade Assistant modifies the SID_DESC entry for the upgraded database in the Oracle Database `listener.ora` file in one of the following ways:

A simple case: Suppose the old `listener.ora` has the following SID_DESC entry:

```
...
   (SID_DESC =
      (SID_NAME = ORCL)
   )
...
```

If the database name is SAL, the domain name is COM, and the Oracle home is `/oracle/product/9.2`, then the assistant adds the following entry:

```
...
   (SID_DESC =
       (GLOBAL_DBNAME = sal.com)
          (ORACLE_HOME = /oracle/product/9.2)
          (SID_NAME = SAL)
    )
...
```

A more complicated case: Suppose the old `listener.ora` has the following SID_DESC entry:

```
...
   (SID_DESC =
       (GLOBAL_DBNAME = an_entry)
       (SID_NAME = ORCL)
    )
...
```

If *an_entry* does not match the GLOBAL_DBNAME of the migrated database, and if the database name is SAL, the domain name is COM, and the Oracle home is `/oracle/product/9.2`, then the assistant adds the following entry:

```
...
   (SID_DESC =
       (GLOBAL_DBNAME = sal.com)
          (ORACLE_HOME = /oracle/product/9.2)
          (SID_NAME = SAL)
    )
...
```

This entry is the same as the entry in the simple case, but the Database Upgrade Assistant also adds the entry *an_entry* to the `SERVICE_NAMES` parameter in the `listener.ora` file. Therefore, the Database Upgrade Assistant changes the `SERVICE_NAMES` parameter to the following:

```
SERVICE_NAMES = sal.com, an_entry
```

b. The Database Upgrade Assistant removes the entry of the upgraded database from the old `listener.ora` file.

c. The Database Upgrade Assistant reloads the `listener.ora` file in both the old and new Oracle Database environments.

13. At the **Changes in Default Behavior** screen, the Database Upgrade Assistant displays some changes in behavior of Oracle Database 10*g* from that of previous releases. In some cases the default values of some initialization parameters have changed. In other cases some new behavior/requirement has been introduced that may affect current scripts or applications.

14. Complete the procedures described in Chapter 4, "After Upgrading a Database".

WARNING: If you retain the old Oracle software, then never start the upgraded database with the old Oracle software. Only start the database with the executables in the new Oracle Database installation. Also, before you remove the old Oracle environment, make sure you relocate any datafiles in that environment to the new Oracle Database environment. See the *Oracle Database Administrator's Guide* for information about relocating datafiles.

Using the Database Upgrade Assistant in Silent Mode

When invoked with the `-silent` command line option, the Database Upgrade Assistant operates in silent mode. In silent mode, the Database Upgrade Assistant does not present a user interface. It also writes any messages (including information, errors, and warnings) to a log file.

For example, the following command upgrades a database named `ORCL` in silent mode:

```
dbua -silent -dbName ORCL
```

Upgrade the Database Manually

The following sections guide you through the process of performing a manual upgrade:

- Analyze the Database to be Upgraded
- Backup the Database
- Upgrade the Database
- Troubleshooting the Upgrade
- Abandoning the Upgrade

Analyze the Database to be Upgraded

Before you upgrade your database to the new Oracle Database 10*g* release, you must run the Pre-Upgrade Information Tool. The Upgrade Information Tool is a SQL script that ships with the new Oracle Database 10*g* release, and must be copied to and run from the environment of the database being upgraded. Complete the following steps to run the Upgrade Information Tool:

1. Log in to the system as the owner of the Oracle home directory of the new Oracle Database 10*g* release.
2. Copy the following file from the *ORACLE_HOME*`/rdbms/admin` directory of the new Oracle Database 10*g* release to a directory outside of the Oracle home, such as the temporary directory on your system:
   - `utlu101i.sql`

   Make a note of the new location of this file.
3. Log in to the system as the owner of the Oracle home directory of the database being upgraded.
4. Change to the directory outside of the Oracle home directory that you copied files to in Step 2.

5. Start SQL*Plus.
6. Connect to the database instance as a user with `SYSDBA` privileges.
7. Set the system to spool results to a log file for later analysis:

   ```
   SQL> SPOOL info.log
   ```

8. Run `utlu101i.sql`:

   ```
   SQL> @utlu101i.sql
   ```

9. Turn off the spooling of script results to the log file:

   ```
   SQL> SPOOL OFF
   ```

   Then, check the spool file and examine the output of the upgrade information tool. You named the spool file in Step 7; the suggested name was `info.log`.

The following example displays the output of the Upgrade Information Tool for a release 9.0.1.0.0 database named `TEST`.

```
Oracle Database 10.1 Upgrade Information Tool    MM-DD-YYYY HH:MM:SS
.
**********************************************************************
Database:
---------
--> name: TEST
--> version: 9.0.1.0.0
--> compatibility: 9.0.0
WARNING: Database compatibility must be set to 9.2.0 prior to upgrade.
.
**********************************************************************
Logfiles: [make adjustments in the current environment]
--------------------------------------------------
-- The existing log files are adequate. No changes are required.
.
**********************************************************************
Tablespaces: [make adjustments in the current environment]
--------------------------------------------------------
--> SYSTEM tablespace is adequate for the upgrade.
.... owner: SYS
.... minimum required size: 466 MB
--> CWMLITE tablespace is adequate for the upgrade.
.... owner: OLAPSYS
.... minimum required size: 13 MB
--> DRSYS tablespace is adequate for the upgrade.
.... owner: CTXSYS
.... minimum required size: 8 MB
.
**********************************************************************
Options: [present in existing database]
---------------------------------------
--> Partitioning
--> Spatial
WARNING: Listed option(s) must be installed with Oracle Database 10.1
.
**********************************************************************
Update Parameters: [Update Oracle Database 10.1 init.ora or spfile]
-------------------------------------------------------------------
WARNING: --> "shared_pool_size" needs to be increased to at least "100663296"
WARNING: --> "pga_aggregate_target" needs to be increased to at least "25165824"
WARNING: --> "large_pool_size" needs to be increased to at least "8388608"
```

```
--> "java_pool_size" is already at "67108864" calculated new value is "67108864"
.
**************************************************************************
Deprecated Parameters: [Update Oracle Database 10.1 init.ora or spfile]
--------------------------------------------------------------------------
-- No deprecated parameters found. No changes are required.
.
**************************************************************************
Obsolete Parameters: [Update Oracle Database 10.1 init.ora or spfile]
--------------------------------------------------------------------------
--> "undo_suppress_errors"
.
**************************************************************************
Components: [The following database components will be upgraded or installed]
--------------------------------------------------------------------------------
--> Oracle Catalog Views         [upgrade]
--> Oracle Packages and Types    [upgrade]
--> JServer JAVA Virtual Machine [upgrade]
--> Oracle XDK for Java          [upgrade]
--> Oracle Java Packages         [install]
--> Oracle Workspace Manager     [upgrade]
--> OLAP Catalog                 [upgrade]
--> Oracle interMedia            [upgrade]
--> Spatial                      [upgrade]
--> Oracle Text                  [upgrade]
--> Oracle Ultra Search          [upgrade]
.
**************************************************************************
.
**************************************************************************
SYSAUX Tablespace: [Create tablespace in Oracle Database 10.1 environment]
--------------------------------------------------------------------------
--> New "SYSAUX" tablespace
.... minimum required size for database upgrade: 500 MB
Please create the new SYSAUX Tablespace AFTER the Oracle Database
10.1 server is started and BEFORE you invoke the upgrade script.
.
**************************************************************************
```

The sections which follow describe the output of the Upgrade Information Tool.

Database

This section displays global database information about the current database, such as the database name, release number, and compatibility level. A warning is displayed if the `COMPATIBLE` initialization parameter needs to be adjusted before the database is upgraded.

Logfiles

This section displays a list of redo log files in the current database whose size is less than 4 MB. For each log file, the file name, group number, and recommended size is displayed. New files of at least 4 MB (preferably 10 MB) need to be created in the current database. Any redo log files less than 4 MB must be dropped before the database is upgraded.

Tablespaces

This section displays a list of tablespaces in the current database. For each tablespace, the tablespace name, owner, and minimum required size is displayed. In addition, a

message is displayed if the tablespace is adequate for the upgrade. If the tablespace does not have enough free space, then space must be added to the tablespace in the current database. Tablespace adjustments need to be made before the database is upgraded.

Options

This section displays a list of options in the current database that must be available in the new Oracle Database 10*g* release before the database is upgraded.

Update Parameters

This section displays a list of initialization parameters in the parameter file of the current database that must be adjusted before the database is upgraded. The adjustments need to be made to the parameter file after it is copied to the new Oracle Database 10*g* release.

> **See Also:** Appendix A, "Initialization Parameter and Data Dictionary Changes" for more information about changes to initialization parameters in the new Oracle Database 10*g* release

Deprecated Parameters

This section displays a list of initialization parameters in the parameter file of the current database that are deprecated in the new Oracle Database 10*g* release.

> **See Also:** "Deprecated Initialization Parameters" on page A-1 for a list of initialization parameters that are deprecated in the new Oracle Database 10*g* release

Obsolete Parameters

This section displays a list of initialization parameters in the parameter file of the current database that are obsolete in the new Oracle Database 10*g* release. Obsolete initialization parameters need to be removed from the parameter file before the database is upgraded.

> **See Also:** "Obsolete Initialization Parameters" on page A-2 for a list of initialization parameters that are obsolete in the new Oracle Database 10*g* release

Components

This section displays a list of database components in the new Oracle Database 10*g* release that will be upgraded or installed when the current database is upgraded.

SYSAUX Tablespace

This section displays the minimum required size for the `SYSAUX` tablespace, which is required in Oracle Database 10*g*. The `SYSAUX` tablespace must be created after the new Oracle Database 10*g* release is started and BEFORE the upgrade scripts are invoked.

Backup the Database

After cleanly shutting down the instance following the analysis of the database, you should perform a full backup of the database. Complete the following steps:

1. Sign on to RMAN:

```
rman "target / nocatalog"
```

2. Issue the following RMAN commands:

```
RUN
{
    ALLOCATE CHANNEL chan_name TYPE DISK;
    BACKUP DATABASE FORMAT 'some_backup_directory%U' TAG before_upgrade;
    BACKUP CURRENT CONTROLFILE TO 'save_controlfile_location';
}
```

Caution: If you encounter any problems with the upgrade, or later wish to abandon the upgrade, then you will need to restore the database from this backup. Therefore, make sure you back up your database now as a precaution.

See Also: *Oracle Database Backup and Recovery Basics* for more information about backing up a database

Upgrade the Database

After analyzing and backing up the database to be upgraded, you are ready to proceed with the manual upgrade. Complete the following steps to upgrade the database:

1. Copy configuration files from the Oracle home of the database being upgraded to the new Oracle Database 10*g* Oracle home:

   a. If your parameter file resides within the old environment's Oracle home, then copy it to the new Oracle home. By default, Oracle looks for the parameter file in *ORACLE_HOME*`/dbs` on UNIX platforms and in *ORACLE_HOME*`\database` on Windows operating systems. The parameter file can reside anywhere you wish, but it should not reside in the old environment's Oracle home after you upgrade to Oracle Database 10*g*.

   b. If your parameter file is a text-based initialization parameter file with either an `IFILE` (include file) or a `SPFILE` (server parameter file) entry, and the file specified in the `IFILE` or `SPFILE` entry resides within the old environment's Oracle home, then copy the file specified by the `IFILE` or `SPFILE` entry to the new Oracle home. The file specified in the `IFILE` or `SPFILE` entry contains additional initialization parameters.

   c. If you have a password file that resides within the old environment's Oracle home, then move or copy the password file to the new Oracle Database 10*g* Oracle home.

      The name and location of the password file are operating system-specific. On UNIX platforms, the default password file is *ORACLE_HOME*`/dbs/orapw`*sid*. On Windows operating systems, the default password file is *ORACLE_HOME*`\database\pwd`*sid*`.ora`. In both cases, *sid* is your Oracle instance ID.

   d. If you are upgrading a cluster database and your `init`*db_name*`.ora` file resides within the old environment's Oracle home, then move or copy the `init`*db_name*`.ora` file to the new Oracle home.

Note: If you are upgrading a cluster database, then perform this step on all nodes in which this cluster database has instances configured.

2. Adjust your parameter file in the new Oracle Database 10*g* release by completing the following steps:

   a. Remove obsolete initialization parameters and adjust deprecated initialization parameters. Certain parameters are obsolete in the new Oracle Database 10*g* release, while other parameters have become deprecated. Remove all obsolete parameters from any parameter file that will start a release 10.1 instance. Obsolete parameters may cause errors in the new Oracle Database 10*g* release. Also, alter any parameter whose syntax has changed in the new Oracle Database 10*g* release.

      The Upgrade Information Tool displays any deprecated parameters and obsolete parameters it finds in the **Deprecated Parameters** and **Obsolete Parameters** sections, respectively.

      See Also: Appendix A, "Initialization Parameter and Data Dictionary Changes" for a list of initialization parameters that have been deprecated or have become obsolete

   b. Make sure the `COMPATIBLE` initialization parameter is properly set for the new Oracle Database 10*g* release. The Upgrade Information Tool displays a warning in the **Database** section if `COMPATIBLE` is not properly set.

   c. Make sure the `SHARED_POOL_SIZE` initialization parameter is set to at least 96 MB (for 32-bit platforms) or to at least 144 MB (for 64-bit platforms).

   d. Make sure the `PGA_AGGREGATE_TARGET` initialization parameter is set to at least 24 MB.

   e. Make sure the `LARGE_POOL_SIZE` initialization parameter is set to at least 8 MB.

   f. Make sure the `JAVA_POOL_SIZE` initialization parameter is set to at least 48 MB.

   g. Make sure the `DB_DOMAIN` initialization parameter is set properly.

      See Also: "The DB_DOMAIN Parameter" on page A-8 for more information about setting this initialization parameter

   h. On Windows operating systems, change the `BACKGROUND_DUMP_DEST` and `USER_DUMP_DEST` initialization parameters that point to RDBMS80 or any other environment variable to point to the following directories instead:

Initialization Parameter	Change Setting To
`BACKGROUND_DUMP_DEST`	*`ORACLE_BASE`*`\oradata\`*`DB_NAME`*
`USER_DUMP_DEST`	*`ORACLE_BASE`*`\oradata\`*`DB_NAME`*`\archive`

      In the settings, substitute the complete Oracle base path for *`ORACLE_BASE`* and substitute the database name for *`DB_NAME`*.

   i. Make sure all path names in the parameter file are fully specified. You should not have relative path names in the parameter file.

   j. If the parameter file contains an `IFILE` entry, then change the `IFILE` entry in the parameter file to point to the new location of the include file that you specified in Step 1. b. Then, edit the file specified in the `IFILE` entry in the same way that you edited the parameter file in Step a through Step i.

k. If you are upgrading a cluster database, then modify the `init`*`db_name`*`.ora` file in the same way that you modified the parameter file.

Make sure you save all of the files you modified after making these adjustments.

> **Note:** If you are upgrading a cluster database, then perform this step on all nodes in which this cluster database has instances configured.

3. If you are upgrading a cluster database, then set the `CLUSTER_DATABASE` initialization parameter to `false`. After the upgrade, you must set this initialization parameter back to `true`.

4. Shut down the instance:

```
SQL> SHUTDOWN IMMEDIATE
```

If your operating system is Windows, then complete the following steps:

a. Stop the `OracleService`*`SID`* Oracle service of the database you are upgrading, where *`SID`* is the instance name. For example, if your *`SID`* is `ORCL`, then enter the following at a command prompt:

```
C:\> NET STOP OracleServiceORCL
```

b. Delete the Oracle service at a command prompt using ORADIM. The following table lists the command to run for each Oracle Database release:

Oracle Database Release	Enter at a Command Prompt
8.0	`C:\> ORADIM80 -DELETE -SID` *`SID`*
8.1 and higher	`C:\> ORADIM -DELETE -SID` *`SID`*

For example, if your Oracle Database release is release 8.0.6 and your *`SID`* is `ORCL`, then enter the following command:

```
C:\> ORADIM80 -DELETE -SID ORCL
```

If your Oracle Database release is release 8.1.7 and your *`SID`* is `ORCL`, then enter the following command:

```
C:\> ORADIM -DELETE -SID ORCL
```

c. Create the new Oracle Database 10*g* service at a command prompt using the ORADIM command of the new Oracle Database release:

```
C:\> ORADIM -NEW -SID SID -INTPWD PASSWORD -MAXUSERS USERS
     -STARTMODE AUTO -PFILE ORACLE_HOME\DATABASE\INITSID.ORA
```

This syntax includes the following variables:

Variable	Description
SID	The same SID name as the SID of the database you are upgrading.
PASSWORD	The password for the new release 10.1 database instance. This is the password for the user connected with `SYSDBA` privileges. The -INTPWD option is not required. If you do not specify it, then operating system authentication is used, and no password is required.

Variable	Description
USERS	The maximum number of users who can be granted `SYSDBA` and `SYSOPER` privileges.
ORACLE_HOME	The release 10.1 Oracle home directory. Ensure that you specify the full pathname with the -PFILE option, including drive letter of the Oracle home directory.

For example, if your *SID* is ORCL, your *PASSWORD* is TWxy579, the maximum number of *USERS* is 10, and the *ORACLE_HOME* directory is `C:\ORA92`, then enter the following command:

```
C:\> ORADIM -NEW -SID ORCL -INTPWD TWxy579 -MAXUSERS 10
     -STARTMODE AUTO -PFILE C:\ORA92\DATABASE\INITORCL.ORA
```

5. If your operating system is UNIX, then make sure that the following environment variables point to the new release 10.1 directories:
   - `ORACLE_HOME`
   - `PATH`
   - `ORA_NLS10`
   - `LD_LIBRARY_PATH`

   Note: If you are upgrading a cluster database, then perform this step on all nodes in which this cluster database has instances configured.

   See Also: Your operating system-specific Oracle Database installation documents for information about setting other important environment variables on your operating system.

6. Log in to the system as the owner of the Oracle home directory of the new Oracle Database 10*g* release.
7. At a system prompt, change to the *ORACLE_HOME*`/rdbms/admin` directory.
8. Start SQL*Plus.
9. Connect to the database instance as a user with `SYSDBA` privileges.
10. Start up the instance by issuing the following command:

    ```
    SQL> STARTUP UPGRADE
    ```

    You may need to use the `PFILE` option to specify the location of your initialization parameter file.

    The following are common errors that may occur when attempting to start up the new Oracle Database 10*g* release.

    - If the `COMPATIBLE` initialization parameter is set below `9.2.0`:

      ```
      ORA-00401: the value for parameter compatible is not supported by this
      release
      ```

    - If the `CLUSTER_DATABASE` initialization parameter is set to `true` instead of `false`:

```
ORA-39701: database must be mounted EXCLUSIVE for UPGRADE or DOWNGRADE
```

- If the `STARTUP` command was issued without the `UPGRADE` keyword:

  ```
  ORA-39700: database must be opened with UPGRADE option
  ```

- If a redo log's size is less than 4 MB:

  ```
  ORA-00336: log file size xxxx blocks is less than minimum 8192 blocks
  ```

If errors appear listing obsolete initialization parameters, then make a note of the obsolete initialization parameters and continue with the upgrade. Then, remove the obsolete initialization parameters the next time you shut down the database.

11. Create a `SYSAUX` tablespace. In Oracle Database 10*g*, the `SYSAUX` tablespace is used to consolidate data from a number of tablespaces that were separate in previous releases.

The `SYSAUX` tablespace must be created with the following mandatory attributes:

- `ONLINE`
- `PERMANENT`
- `READ WRITE`
- `EXTENT MANAGEMENT LOCAL`
- `SEGMENT SPACE MANAGEMENT AUTO`

The Upgrade Information Tool provides an estimate of the minimum required size for the `SYSAUX` tablespace in the **SYSAUX Tablespace** section. Table 3–2 can be used to determine an optimal size for the `SYSAUX` tablespace.

Table 3–2 Guidelines for sizing the SYSAUX tablespace

Factor	Small	Medium	Large
Number of CPUs	2	8	32
Number of concurrently active sessions	5	20	100
Number of user objects (tables and indexes)	500	5,000	50,000
Estimated SYSAUX size at steady state with default config	500 MB	2 GB	5 GB

The following SQL statement would create a 500 MB `SYSAUX` tablespace for the database:

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

See Also: *Oracle Database Administrator's Guide* for more information about the `SYSAUX` tablespace

12. Set the system to spool results to a log file for later verification of success:

```
SQL> SPOOL upgrade.log
```

13. Run u*old_release*.sql, where *old_release* refers to the release you had installed prior to upgrading. See Table 3–3 to choose the correct script. Each script provides a direct upgrade from the release specified in the "Old Release" column. The "Old Release" is the release from which you are upgrading.

    To run a script, enter the following:

    ```
    SQL> @uold_release.sql
    ```

Table 3–3 Upgrade Scripts

Old Release	Run Script
8.0.6	u0800060.sql
8.1.7	u0801070.sql
9.0.1	u0900010.sql
9.2	u0902000.sql

> **See Also:** "Determine the Upgrade Path to the New Oracle Database 10g Release" on page 2-2 if the old release you had installed prior to upgrading is not listed in Table 3–3

Make sure you follow these guidelines when you run the script:

- You must use the version of the script supplied with the new release 10.1 installation.
- You must run the script in the new release 10.1 environment.
- You only need to run one script, even if your upgrade spans more than one release. For example, if your old release was 8.1.7, then you only need to run u0801070.sql.

The upgrade script creates and alters certain data dictionary tables. It also upgrades or installs the following database components in the new release 10.1 database:

- Oracle Database Catalog Views
- Oracle Database Packages and Types
- JServer JAVA Virtual Machine
- Oracle Database Java Packages
- Oracle XDK
- Oracle Real Application Clusters
- Oracle Workspace Manager
- Oracle interMedia
- Oracle XML Database
- OLAP Analytic Workspace
- Oracle OLAP API
- OLAP Catalog
- Oracle Text
- Spatial

- Oracle Data Mining
- Oracle Label Security
- Messaging Gateway

14. Turn off the spooling of script results to the log file:

```
SQL> SPOOL OFF
```

Then, check the spool file and verify that the packages and procedures compiled successfully. You named the spool file in Step 12; the suggested name was `upgrade.log`. Correct any problems you find in this file and rerun the appropriate upgrade script if necessary. You can rerun any of the scripts described in this chapter as many times as necessary.

15. Run `utlu101s.sql`, specifying the `TEXT` option:

```
SQL> @utlu101s.sql TEXT
```

The Post-upgrade Status Tool displays the status of the database components in the upgraded database. The Upgrade Status Tool displays output similar to the following:

```
Oracle Database 10.1 Upgrade Status Tool MM-DD-YYYY HH:MM:SS
--> Oracle Database Catalog Views       Normal successful completion
--> Oracle Database Packages and Types  Normal successful completion
--> JServer JAVA Virtual Machine        Normal successful completion
--> Oracle XDK                          Normal successful completion
--> Oracle Database Java Packages       Normal successful completion
--> Oracle Real Application Clusters    Normal successful completion
--> Oracle interMedia                   Normal successful completion
--> Oracle Text                         Normal successful completion
```

16. Shut down and restart the instance to reinitialize the system parameters for normal operation. The restart will also perform release 10.1 initialization for JServer JAVA Virtual Machine and other components.

```
SQL> SHUTDOWN IMMEDIATE
SQL> STARTUP
```

Cleanly shutting down and restarting the instance flushes all caches, clears buffers, and performs other housekeeping activities. These measures are an important final step to ensure the integrity and consistency of the newly upgraded Oracle Database.

Also, if you encountered a message listing obsolete initialization parameters when you started the database in Step 10, then remove the obsolete initialization parameters from the parameter file now.

17. Run `utlrp.sql` to recompile any remaining stored PL/SQL and Java code.

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

Verify that all expected packages and classes are valid:

```
SQL> SELECT count(*) FROM dba_objects WHERE status='INVALID';
SQL> SELECT distinct object_name FROM dba_objects WHERE status='INVALID';
```

18. Exit SQL*Plus.

Your database is now upgraded to the new Oracle Database 10*g* release. Complete the procedures described in Chapter 4, "After Upgrading a Database".

WARNING: If you retain the old Oracle software, then never start the upgraded database with the old software. Only start the database with the executables in the new release 10.1 installation directory. Also, before you remove the old Oracle Database environment, make sure you relocate any datafiles in that environment to the new Oracle Database environment. See the *Oracle Database Administrator's Guide* for information about relocating datafiles.

Troubleshooting the Upgrade

There are three resources that generally require increases for a new Oracle Database release:

- `SYSTEM` tablespace
- Shared memory
- Rollback segments

If you run out of one of these resources during the upgrade, then increase the resource allocation and re-run the appropriate upgrade scripts.

SYSTEM Tablespace

Typically you will receive one of the following messages during the upgrade if your `SYSTEM` tablespace size is insufficient:

```
ORA-01650: unable to extend rollback segment string by string in tablespace string
ORA-01651: unable to extend save undo segment by string for tablespace string
ORA-01652: unable to extend temp segment by string in tablespace string
ORA-01653: unable to extend table string.string by string in tablespace string
ORA-01654: unable to extend index string.string by string in tablespace string
ORA-01655: unable to extend cluster string.string by string in tablespace string
```

Shared Memory

You will require larger shared memory pool sizes, particularly if you have JServer in the database. The error message will indicate which shared memory initialization parameter needs to be increased.

```
ORA-04031: unable to allocate string bytes of shared memory
("string","string","string","string")
```

Refer to *Oracle Database Reference* for information about shared memory initialization parameters.

Public Rollback Segment

If you are using rollback segments, then you need to have a single large (70 MB) `PUBLIC` rollback segment online while the upgrade scripts are being run. Smaller public rollback segments should be taken offline during the upgrade. Typically you will get the following error if your rollback segment size is insufficient:

```
ORA-01562: failed to extend rollback segment number string
```

Abandoning the Upgrade

If you completed the steps in "Backup the Database" on page 3-18 to back up your database, then the easiest way to abandon the upgrade is to restore that backup. Complete the following steps:

1. Log in to the system as the owner of the Oracle home directory of the previous release.
2. Sign on to RMAN:

```
rman "target / nocatalog"
```

3. Issue the following RMAN commands:

```
STARTUP NOMOUNT
RUN
{
    REPLICATE CONTROLFILE FROM 'save_controlfile_location';
    ALTER DATABASE MOUNT;
    RESTORE DATABASE FROM TAG before_upgrade
    ALTER DATABASE OPEN RESETLOGS;
}
```

4

After Upgrading a Database

This chapter guides you through the procedures to perform after you have completed an upgrade of your database. This chapter covers the following topics:

Tasks to Complete After Upgrading Your Database

Complete the following tasks after you have upgraded your database.

Back Up the Database

> **Note:** You do not need to perform this step if the Database Upgrade Assistant was used to upgrade the database.

Make sure you perform a full backup of the production database.

> **See Also:** *Oracle Database Backup and Recovery Basics* for details about backing up a database

Change Passwords for Oracle-Supplied Accounts

> **Note:** You do not need to perform this step if the Database Upgrade Assistant was used to upgrade the database.

Depending on the release from which you upgraded, there may be new Oracle-supplied accounts. Oracle recommends that you lock all Oracle-supplied accounts except for `SYS` and `SYSTEM`, and expire their passwords, thus requiring new passwords to be specified when the accounts are unlocked.

You can view the status of all accounts by issuing the following SQL statement:

```
SQL> SELECT username, account_status
        FROM dba_users
        ORDER BY username;
```

To lock and expire passwords, issue the following SQL statement:

```
SQL> ALTER USER username PASSWORD EXPIRE ACCOUNT LOCK;
```

Upgrading from the Standard Edition to the Enterprise Edition

If you are using the Standard Edition of the Oracle Database and want to upgrade to the Enterprise Edition, then complete the following steps:

1. Ensure that the release number of your Standard Edition server software is the same release as the Enterprise Edition server software.

   For example, if your Standard Edition server software is release 9.2.0.1.0, then you should upgrade to release 9.2.0.1.0 of the Enterprise Edition.
2. Shut down your database.
3. If your operating system is Windows, then stop all Oracle services, including the `OracleService`*`SID`* Oracle service, where *`SID`* is the instance name.
4. Deinstall the Standard Edition server software.
5. Install the Enterprise Edition server software using the Oracle Universal Installer.

   Select the same Oracle home that was used for the de-installed Standard Edition. During the installation, be sure to select the Enterprise Edition. When prompted, choose Software Only from the Database Configuration screen.
6. Start up your database.

Your database is now upgraded to the Enterprise Edition.

Upgrading and Tablespace Alerts

An upgraded Oracle Database 10*g* database has the Tablespace Alerts disabled (the thresholds are set to null). Tablespaces in the database that are candidates for monitoring need to be identified and the appropriate threshold values set.

The default threshold values (for a newly created Oracle Database 10*g* database) are:

- 85% full warning
- 97% full critical

Migrate Your Oracle Managed Files

If you are upgrading from an Oracle9`i` release earlier than release 9.0.1.2.0, then you must migrate your Oracle Managed Files. In Oracle9`i` releases earlier than release 9.0.1.2.0, Oracle sometimes incorrectly considered non-OMF files as OMF. This resulted in the following error when adding a datafile, control file, or log file to the database:

```
ORA-01276: Cannot add a file with an Oracle Managed Files file name
```

Also, Oracle sometimes incorrectly deleted the operating system files associated with a tablespace or redo log when dropping the tablespace or redo log.

Starting with release 9.0.1.2.0, the format of Oracle Managed Files file names on Windows and UNIX operating systems has changed. OMF files created in earlier Oracle9`i` releases will not be recognized as OMF files unless they are renamed to conform to the new OMF file name format.

In Oracle9i releases earlier than release 9.0.1.2.0, a file was considered an Oracle-managed file if its base file name contained:

- An ora_ prefix
- A .dbf, .tmp, .log, or .ctl extension

In release 9.0.1.2.0 and later, a file is considered an Oracle-managed file if its base file name contains:

- An o1_mf_ prefix
- A .dbf, .tmp, .log, or .ctl extension
- An underscore (_) immediately preceding the extension

You can migrate old OMF datafiles, tempfiles, and log files by renaming them in the file system and in the control file. Complete the following steps:

1. Find the OMF files by issuing the following SQL statements:

```
SQL> SELECT name FROM v$datafile;
SQL> SELECT name FROM v$tempfile;
SQL> SELECT member FROM v$logfile;
```

2. Shut down the instance:

```
SQL> SHUTDOWN IMMEDIATE
```

3. Rename the files in the file system:
   - Change ora_ to o1_mf_
   - Add _ before the extension

   For example, for a file named ora_tbs1_2ixfh90q.dbf, the new name would be o1_mf_tbs1_2ixfh90q_.dbf.
4. Mount the database.
5. Rename the files in the control file. For example:

```
SQL> ALTER DATABASE RENAME FILE 'old_filename' TO 'new_omf_filename';
```

6. Open the database.

OMF control files can be migrated by renaming them in the file system and in the CONTROL_FILES initialization parameter. Complete the following steps:

1. Find the OMF files by examining the CONTROL_FILES initialization parameter.
2. Shut down the instance:

```
SQL> SHUTDOWN IMMEDIATE
```

3. Rename the files in the file system:
   - Change ora_ to o1_mf_
   - Add _ before the extension

   For example, for a file named ora_cmr7t90p.ctl, the new name would be o1_mf_cmr7t90p_.ctl.
4. Modify the CONTROL_FILES initialization parameter to reference the new names.
5. Mount and open the database.

Migrate Your Initialization Parameter File to a Server Parameter File

> **Note:** You do not need to perform this step if the Database Upgrade Assistant was used to upgrade the database.

If you are currently using a traditional initialization parameter file, perform the following steps to migrate to a server parameter file:

1. If the initialization parameter file is located on a client machine, transfer the file from the client machine to the server machine.

   > **Note:** If you are using Real Application Clusters, then you must combine all of your instance-specific initialization parameter files into a single initialization parameter file. Instructions for doing this, and other actions unique to using a server parameter file for cluster databases, are discussed in:
   >
   > - *Oracle Real Application Clusters Installation and Configuration Guide*
   > - *Oracle Real Application Clusters Administrator's Guide*

2. Create a server parameter file using the `CREATE SPFILE` statement. This statement reads the initialization parameter file to create a server parameter file. The database does not have to be started to issue a `CREATE SPFILE` statement.
3. Start up the instance using the newly-created server parameter file.

See Also:

- *Oracle Database Administrator's Guide* for more information about creating server parameter files
- *Oracle Database SQL Reference* for information about the `CREATE SPFILE` statement

Migrate Tables from LONGs to LOBs

`LOB` datatypes (`BFILE`, `BLOB`, `CLOB`, and `NCLOB`) can provide many advantages over `LONG` datatypes. See *Oracle Database Concepts* for information about the differences between `LONG` and `LOB` datatypes.

In Oracle9i release 9.0.1 and later, the `ALTER TABLE` statement can be used to change the datatype of a `LONG` column to `CLOB` and that of a `LONG RAW` column to `BLOB`.

In the following example, the `LONG` column named `long_col` in table `long_tab` is changed to datatype `CLOB`:

```
SQL> ALTER TABLE Long_tab MODIFY ( long_col CLOB );
```

After using this method to change `LONG` columns to LOBs, all the existing constraints and triggers on the table will still be usable. However, all the indexes, including Domain indexes and Functional indexes, on all columns of the table will become unusable and will have to be rebuilt using an `ALTER INDEX ... REBUILD` statement. Also, the Domain indexes on the `LONG` column will have to be dropped before changing the `LONG` column to a LOB.

See Also: *Oracle Database Application Developer's Guide - Large Objects* for information about modifying applications to use LOB data

Modify Your listener.ora File

You need to modify your `listener.ora` file only if one of the following conditions is true:

- You did not use the Database Upgrade Assistant to upgrade your database.
- You used the Database Upgrade Assistant to upgrade your database but chose not to have the `listener.ora` file updated automatically.

If neither of these conditions is true, then skip this section. If one of these conditions is true, then you need to modify your `listener.ora` file.

> **See Also:** *Oracle Net Services Administrator's Guide* for information about modifying your `listener.ora` file.

Upgrade Your Standby Database

The following procedures contain information about upgrading your current release of the Oracle Database to the new Oracle Database release for a configuration that includes one or more standby databases.

Prepare to Upgrade

If multiple standby databases exist, then repeat the steps in this section for each standby database to be upgraded:

1. Check for the existence of nologging operations. If nologging operations have been performed, then the standby will need to be updated. Refer to *Oracle Data Guard Concepts and Administration* for further details.
2. Make note of any tablespaces or datafiles that need recovery due to offline immediate. Tablespaces or datafiles should be recovered and either brought online or taken offline prior to upgrading.

Upgrade the Production Site

Install the new Oracle Database release on production sites and follow the instructions in Oracle Database for upgrading the production database.

Make the following additional adjustments to your parameter file before the upgrade:

- Do not enable remote archiving within the production database's parameter file if it was not already enabled. If remote archiving is enabled, then set the remote destination to defer.
- Cancel managed recovery on the standby database if running.
- If upgrading from release 8.1.7 or earlier and running Real Application Clusters Guard, make sure to comment out the `PARALLEL_SERVER` initialization parameter and set `CLUSTER_DATABASE = true` on the production site.

Ensure that all archived redo logs have been applied to the standby prior to the upgrade.

After the upgrade is complete, switch logfiles to archive any redo that remains in the last log:

```
SQL> ALTER SYSTEM SWITCH LOGFILE;
```

Manually transfer archive logs from the upgrade from the primary archive destination on the production site to the standby archive destination on the standby host.

Shut down the standby database and listener

Start up and mount the standby database.

Place the standby database in managed recovery mode. At the SUGGESTION prompt, type AUTO to apply all of the archive logs generated during the upgrade process.

Verify that the standby database has been recovered to the last log that was transferred to the standby host. Resolve any archive log gaps between the production and the standby.

Re-enable remote archiving on the primary site by changing the standby destination from defer to enable.

Place standby into a recovery state.

Upgrade or Downgrade the Oracle Data Guard Broker

If you need to upgrade or downgrade Oracle Data Guard broker to a different release, then see *Oracle Data Guard Broker* for the following release scenarios:

- Upgrading from release 9.0.1 to release 10.1
- Upgrading from release 9.2 to release 10.1
- Downgrading from release 10.1

Upgrading Oracle Text

> **See Also:** *Oracle Text Application Developer's Guide* for information about upgrading your applications from previous releases of Oracle Text

Supplied Knowledge Bases

The Supplied Knowledge Bases have been moved to the Oracle Database 10*g* Companion CD and are not immediately available after an upgrade to Oracle Database 10*g*. Any Text features dependent on the Supplied Knowledge Bases which were available before the upgrade will not function after the upgrade. To re-enable such features, you must install the Supplied Knowledge Bases from the Companion CD.

After an upgrade, all user-extensions to the Supplied Knowledge Bases must be regenerated. These changes affect all databases installed in the given `ORACLE_HOME`.

> **See Also:**
>
> - *Oracle Text Application Developer's Guide* for information about Supplied Knowledge Bases
> - The post-installation tasks section of the Database Installation Guide and the Companion CD Installation Guide for information about installing products from the Companion CD

Copy Files from the Previous ORACLE_HOME to the New ORACLE_HOME

After an upgrade to Oracle Database 10*g*, copy the following files from the previous `ORACLE_HOME` to the new `ORACLE_HOME`:

- Stemming user-dictionary files
- User-modified `KOREAN_MORPH_LEXER` dictionary files
- `USER_FILTER` executables

These files affect all databases installed in the given ORACLE_HOME.

> **See Also:** *Oracle Text Reference* for more information about these files

Add New Features as Appropriate

Oracle Database New Features describes many of the new features available in the new Oracle Database release. Determine which of these new features can benefit the database and applications; then, develop a plan for using these features.

It is not necessary to make any immediate changes to begin using your new Oracle Database. You may prefer to introduce these enhancements into your database and corresponding applications gradually.

Chapter 6, "Upgrading Your Applications" describes ways to enhance your applications so that you can take advantage of new Oracle Database features. However, before you implement new Oracle Database features, test your applications and successfully run them with the upgraded database.

Develop New Administrative Procedures as Needed

After familiarizing yourself with new Oracle Database features, review your database administration scripts and procedures to determine whether any changes are necessary.

Coordinate your changes to the database with the changes that are necessary for each application. For example, by enabling integrity constraints in the database, you may be able to remove some data checking from your applications.

Adjust Your Parameter File for the New Release

> **Note:** You do not need to perform this step if the Database Upgrade Assistant was used to upgrade the database.

Each release of the Oracle Database introduces new initialization parameters, deprecates some initialization parameters, and makes some initialization parameters obsolete. You should adjust your parameter file to account for these changes and to take advantage of new initialization parameters that may be beneficial to your system.

> **See Also:**
>
> - *Oracle Database Reference* for a list of the new initialization parameters in release 9.2, and for information about each parameter
> - Appendix A, "Initialization Parameter and Data Dictionary Changes" for lists of obsolete and deprecated initialization parameters in release 9.2

The COMPATIBLE initialization parameter controls the compatibility level of your database. Set the COMPATIBLE initialization parameter based on the compatibility level you want for your new database.

> **See Also:** "Setting the COMPATIBLE Initialization Parameter" on page 5-3 for information

Tasks to Complete Only After Upgrading a Release 8.1.7 or Lower Database

Complete the following additional tasks only if you upgraded your database from release 8.1.7 or lower. These tasks are *not* required if you upgraded from release 9.0.1.

Upgrade User NCHAR Columns

If you upgraded from a version 8 release and your database contains user tables with `NCHAR` columns, you must upgrade the `NCHAR` columns before they can be used in the Oracle Database.

The following steps convert your `NCHAR` columns from the old format and character set to the new Oracle Database format. In addition, if your old National Character Set was UTF8, it will remain UTF8 in the Oracle Database. However, your National Character Set will be converted to AL16UTF16 if it was not UTF8 in the old release.

You can override the default upgrade selection of the National Character Set. That is, a version 8 UTF8 National Character Set can be converted to an Oracle Database AL16UTF16 National Character Set or a version 8 non-UTF8 National Character Set can be converted to an Oracle Database UTF8 National Character Set.

You will encounter the following error when attempting to use the `NCHAR` columns in the Oracle Database until you perform the steps in this section:

```
ORA-12714: invalid national character set specified
```

To upgrade user tables with `NCHAR` columns, perform the following steps:

1. Log in to the system as the owner of the Oracle home directory.
2. At a system prompt, change to the *ORACLE_HOME*`/rdbms/admin` directory.
3. Start SQL*Plus.
4. Connect to the database instance as a user with `SYSDBA` privileges.
5. If the instance is running, shut it down using `SHUTDOWN IMMEDIATE`:

   ```
   SQL> SHUTDOWN IMMEDIATE
   ```

6. Start up the instance in `RESTRICT` mode:

   ```
   SQL> STARTUP RESTRICT
   ```

   You may need to use the `PFILE` option to specify the location of your initialization parameter file.

7. Run `utlnchar.sql`:

   ```
   SQL> @utlnchar.sql
   ```

   Alternatively, to override the default upgrade selection, run `n_switch.sql`:

   ```
   SQL> @n_switch.sql
   ```

8. Shut down the instance:

   ```
   SQL> SHUTDOWN IMMEDIATE
   ```

9. Exit SQL*Plus.

Migrate Your Server Manager Line Mode Scripts to SQL*Plus

The Oracle Database no longer supports the use of Server Manager. If you run SQL scripts using Server Manager line mode, you must modify these scripts so that they are compatible with SQL*Plus. Appendix B, "Migrating from Server Manager to SQL*Plus" contains instructions for modifying your Server Manager line mode scripts to work with SQL*Plus.

Tasks to Complete Only After Upgrading a Release 8.0.6 Database

Complete the following additional tasks only if you upgraded your database from release 8.0.6. These tasks are *not* required if you upgraded from release 8.1.7 or higher.

Avoid Problems with Parallel Execution

Starting with release 8.1, parallel execution message buffers can be allocated from the large pool. In past releases, this allocation was from the shared pool. To avoid problems resulting from this change, you may need to adjust the following initialization parameters in your initialization parameter file:

- `SHARED_POOL_SIZE`
- `LARGE_POOL_SIZE`

> **See Also:** "Parallel Execution Allocated from Large Pool" on page A-9 for information about adjusting these parameters.

Normalize Filenames on Windows Operating Systems

You only need to normalize filenames if you are running the Oracle Database on a Windows operating system. You do not need to perform these steps on UNIX platforms.

The control file and the recovery catalog both store filenames so that they can access files that are required by the database, such as:

- Datafiles
- Control files
- Online and archived redo logs used by Oracle
- Datafile copies and on-disk backup pieces used by Recovery Manager

In releases prior to release 8.1.6 on Windows operating systems, a flawed filename normalization mechanism allowed two different filenames to refer to the same physical file. For example, because of this flaw, the Oracle Database may not record the fully specified pathname for a file in the control file. That is, the Oracle Database may record only `dbfile1.dbf` instead of `c:\oracle\oradata\dbfile1.dbf`. If this happens, then, in subsequent statements that modify `c:\oracle\oradata\dbfile1.dbf`, the Oracle Database might conclude that this file is different than `dbfile1.dbf`.

Also, because of this behavior, SQL statements and Recovery Manager commands that refer to existing files must be specified exactly as they were originally entered or they are not recognized. An example of a SQL statement that refers to existing files is the `ALTER DATABASE RENAME FILE` statement.

In release 8.1.6 and higher, the flawed filename normalization mechanism is corrected. However, existing filenames in the control file and recovery catalog must be normalized with the new filename normalization mechanism.

> **Note:** Do not perform the following procedure on Oracle releases prior to release 8.1.6.

To normalize these filenames, complete the following steps:

1. Using SQL*Plus, connect to the database as a user with `SYSDBA` privileges.
2. Shut down the database using `SHUTDOWN NORMAL` or `SHUTDOWN IMMEDIATE`:

   ```
   SQL> SHUTDOWN IMMEDIATE
   ```

3. Make an operating system backup of your control file.

   > **See Also:** *Oracle Database Backup and Recovery Basics* for more information about operating system backups

4. Run `STARTUP MOUNT` to mount the database without opening it:

   ```
   SQL> STARTUP MOUNT
   ```

5. Run the `DBMS_BACKUP_RESTORE.RENORMALIZEALLFILENAMES` procedure to normalize the filenames in your control file:

   ```
   SQL> EXECUTE DBMS_BACKUP_RESTORE.RENORMALIZEALLFILENAMES;
   ```

6. When the `DBMS_BACKUP_RESTORE.RENORMALIZEALLFILENAMES` procedure has completed successfully, open the database:

   ```
   SQL> ALTER DATABASE OPEN;
   ```

7. Exit SQL*Plus.
8. Log in to Recovery Manager and connect to a target database and recovery catalog.

   For example, if the network service name for the target database is `TGT_DB` and the network service name for the recovery catalog database is `CAT_DB`, then you can enter the following, substituting the appropriate schema names and passwords:

   ```
   rman target sys/password@tgt_db catalog rcat_schema/rcat_password@cat_db
   ```

9. Issue the `RENORMALIZE CATALOG` command to normalize the filenames in the recovery catalog for this target database:

   ```
   RMAN> renormalize catalog;
   ```

   > **Note:** The `RENORMALIZE CATALOG` command is not considered part of the Recovery Manager syntax and is not documented in the *Oracle Database Backup and Recovery Advanced User's Guide*. The command is only intended for use on databases upgraded from a release prior to release 8.1.6 on Windows platforms.

10. Repeat Steps 8 through 9 for each release 8.1.6 or higher target database registered in this recovery catalog.

Your filenames are now normalized.

Note: If you need to restore a control file for a point-in-time recovery from a backup that was taken before you completed the filename normalization procedure described above, then first restore the backup control file, then perform Steps 1 to 7, and finally perform the recovery.

Rebuild Unusable Function-Based Indexes

During an upgrade, some function-based indexes may become unusable. To find these indexes, issue the following SQL statement:

```
SELECT owner, index_name, funcidx_status
   FROM dba_indexes WHERE funcidx_status = 'DISABLED';
```

Rebuild the unusable function-based indexes listed.

Upgrade Materialized Views

Note: The word "snapshot" is synonymous with the word "materialized view".

Materialized views upgraded from release 8.0 or imported from a release 8.0 database cannot use the new summary management features available in release 8.1 and higher. If you want to use these new features, then complete the following steps for each materialized view and for each materialized view imported from release 8.0:

1. GRANT QUERY REWRITE privileges to the owner of the materialized view. Only local materialized views are available for query rewrite.

   If the materialized view references any schema objects outside its owner's schema, then you must issue a `GRANT GLOBAL QUERY REWRITE` statement.

2. Issue the `ALTER MATERIALIZED VIEW ... ENABLE QUERY REWRITE` statement on the materialized views you want to upgrade.

   For example, on a materialized view named `SSORDERS`, issue the following statement:

   ```
   ALTER MATERIALIZED VIEW ssorders ENABLE QUERY REWRITE;
   ```

In addition, if you do not `ENABLE QUERY REWRITE` on a materialized view, then the `ATOMIC=FALSE` option of the `DBMS_MVIEW.REFRESH` procedure may not work unless you issue an `ALTER MATERIALIZED VIEW ... COMPILE` statement on the materialized view. For example, for a materialized view named `SSCUST`, issue the following statement:

```
ALTER MATERIALIZED VIEW sscust COMPILE;
```

You do not need to issue this statement if you have issued any other `ALTER MATERIALIZED VIEW` statement on the materialized view, such as the `ALTER MATERIALIZED VIEW ... ENABLE QUERY REWRITE` statement.

Upgrade Your Queue Tables

The following release 8.1 and higher AQ enhancements are available only if you upgrade your existing queue tables:

- Addition of the original message ID column for propagated messages
- Addition of a sender's ID column
- Queue and system level privileges
- Rule based subscriptions
- Separate storage of history management information, which was stored in a varray in release 8.0

To upgrade an existing queue table, run the `DBMS_AQADM.MIGRATE_QUEUE_TABLE` procedure, specifying 8.1 for the option. For example, for a queue table named `tb_queue` owned by user `scott`, run the following procedure:

```
EXECUTE dbms_aqadm.migrate_queue_table (
    queue_table => 'scott.tb_queue',
    compatible => '8.1');
```

To create a new queue table that is compatible with release 8.1 and higher, connect as the owner of the queue table and run the `DBMS_AQADM.CREATE_QUEUE_TABLE` procedure, specifying 8.1 for the `COMPATIBLE` option, as in the following example:

```
EXECUTE dbms_aqadm.create_queue_table(
    queue_table => 'scott.tkaqqtpeqt',
    queue_payload_type =>'message',
    sort_list => 'priority,enq_time',
    multiple_consumers => true,
    comment => 'Creating queue with priority and enq_time sort order',
    compatible => '8.1');
```

Upgrade the Recovery Catalog

See Also: *Oracle Database Backup and Recovery Advanced User's Guide* for information about upgrading the recovery catalog

Upgrade Statistics Tables Created by the DBMS_STATS Package

If you created statistics tables using the DBMS_STATS.CREATE_STAT_TABLE procedure, then upgrade these tables by executing the following procedure:

```
EXECUTE DBMS_STATS.UPGRADE_STAT_TABLE('scott', 'stat_table');
```

where SCOTT is the owner of the statistics table and STAT_TABLE is the name of the statistics table. Execute this procedure for each statistics table.

Test the Database and Compare Results

Test the new Oracle Database using the testing plan you developed in "Develop a Testing Plan" on page 2-6. Compare the results of the test with the results obtained with the original database and make certain the same, or better, results are achieved.

Generally, the performance of the new Oracle Database should be as good as, or better than, the performance of the previous database. If you notice any decline in database performance with the new Oracle Database, then make sure the initialization parameters are set properly, because improperly set initialization parameters can impede performance.

Tune the Upgraded Database

If you want to improve the performance of the upgraded database, then tune the database. Actions you used to tune your previous database and applications should not impair the performance of the upgraded Oracle Database.

> **See Also:** *Oracle Database Performance Tuning Guide* for tuning information

5

Compatibility and Interoperability

This chapter describes compatibility and interoperability issues that may arise because of differences between Oracle Database releases. These differences may affect general database administration and existing applications.

This chapter covers the following topics:

What Is Compatibility?

When you upgrade to a new release of the Oracle Database, certain new features may make your database incompatible with your previous release. Your upgraded Oracle database becomes incompatible with your previous release under the following conditions:

- A new feature stores any data on disk (including data dictionary changes) that cannot be processed with your previous release.
- An existing feature behaves differently in the new environment as compared to the old environment. This type of incompatibility is classified as a **language incompatibility**.

The COMPATIBLE Initialization Parameter

The Oracle Database enables you to control the compatibility of your database with the `COMPATIBLE` initialization parameter. By default, when the `COMPATIBLE` initialization parameter is not set in your parameter file, it defaults to `10.0.0` for Oracle Database 10*g* release 10.1. You cannot use new Oracle Database 10*g* features that would make your upgraded database incompatible until the `COMPATIBLE` initialization parameter is set to this value.

Table 5–1 illustrates the default value and the range of values of the `COMPATIBLE` initialization parameter in the new Oracle Database 10*g* release and in each release supported for upgrading to the new Oracle Database 10*g* release.

Table 5–1 The COMPATIBLE Initialization Parameter

Oracle Database Release	Default Value	Minimum Value	Maximum Value
Oracle8 release 8.0.6	8.0.0	8.0.0	8.0.6
Oracle8*i* release 8.1.7	8.0.0	8.0.0	8.1.7
Oracle9*i* release 9.0.1	8.1.0	8.1.0	9.0.1
Oracle9*i* release 9.2	8.1.0	8.1.0	9.2.0
Oracle Database 10*g* release 10.1	10.0.0	9.2.0	10.0.0

Downgrading and Compatibility

Before upgrading to the new Oracle Database 10*g* release, the COMPATIBLE initialization parameter must be set to 9.2.0, which is the lowest possible setting for Oracle Database 10*g* release 10.1. Only a subset of Oracle Database 10*g* features are available while the COMPATIBLE initialization parameter is set to this value.

After upgrading to the new Oracle Database 10*g* release, you can set the COMPATIBLE initialization parameter to match the release number of the new release. Doing so enables you to use all of the features of the new release, but prevents you from downgrading back to your previous release. If you want to downgrade, then you must leave the COMPATIBLE initialization parameter set to 9.2.0 after the upgrade.

> **See Also:** Chapter 7, "Downgrading a Database Back to the Previous Oracle Database Release" for more information about downgrading

How the COMPATIBLE Initialization Parameter Operates

The COMPATIBLE initialization parameter operates in the following way:

- It controls the behavior of your database. For example, if you run a release 10.1 database with the COMPATIBLE initialization parameter set to 9.2.0, then the release 10.1 database generates release 9.2 compatible database structures on disk. Therefore, the COMPATIBLE initialization parameter enables or disables the use of features. If you try to use any new features that make the database incompatible with the COMPATIBLE initialization parameter, then an error is displayed. However, any new features that do not make incompatible changes on disk are enabled.
- It makes sure that the database is compatible with its setting. If the database becomes incompatible with its setting, then the database does not start and terminates with an error. If this happens, then you must set the COMPATIBLE initialization parameter to an appropriate value for the database.

> **See Also:** *Oracle Database Concepts* for more information about database structures

Compatibility Level

The compatibility level of your database corresponds to the value of the COMPATIBLE initialization parameter. For example, if you set the COMPATIBLE initialization parameter to 10.0.0, then the database runs at 10.0.0 compatibility level.

Checking the Current Value of the COMPATIBLE Initialization Parameter

To check the current value of the COMPATIBLE initialization parameter, issue the following SQL statement:

```
SQL> SELECT name, value, description FROM v$parameter
```

```
WHERE name = 'compatible';
```

When to Set the COMPATIBLE Initialization Parameter

You should set the COMPATIBLE initialization parameter at a specific point during the upgrade or downgrade process. Follow the procedure in the appropriate chapter and set the COMPATIBLE initialization parameter only when you are instructed to do so.

> **Note:** Once the upgrade or downgrade is complete, you can change the setting of the COMPATIBLE initialization parameter as necessary.

Setting the COMPATIBLE Initialization Parameter

Complete the steps in one of the following sections to set the COMPATIBLE initialization parameter:

Raising the COMPATIBLE Initialization Parameter

Complete the following steps to set the COMPATIBLE initialization parameter to a higher value:

1. Perform a backup of your database before you raise the COMPATIBLE initialization parameter (optional).

   Raising the COMPATIBLE initialization parameter may cause your database to become incompatible with earlier releases of the Oracle Database, and a backup ensures that you can return to the earlier release if necessary.

   > **See Also:** *Oracle Database Backup and Recovery Basics* for more information about performing a backup

2. If you are using a server parameter file, then complete the following steps:

   a. Update the server parameter file to set or change the value of the COMPATIBLE initialization parameter.

      For example, to set the COMPATIBLE initialization parameter to 10.0.0, issue the following statement:

      ```
      SQL> ALTER SYSTEM SET COMPATIBLE = '10.0.0' SCOPE=SPFILE;
      ```

   b. Shut down and restart the instance.

3. If you are using an initialization parameter file, then complete the following steps:

   a. Shut down the instance if it is running:

      ```
      SQL> SHUTDOWN IMMEDIATE
      ```

   b. Edit the initialization parameter file to set or change the value of the COMPATIBLE initialization parameter.

      For example, to set the COMPATIBLE initialization parameter to 10.0.0, enter the following in the initialization parameter file:

      ```
      COMPATIBLE = 10.0.0
      ```

   c. Start the instance using STARTUP.

What Is Interoperability?

Interoperability is the ability of different releases of the Oracle Database to communicate and work together in a distributed environment. A distributed database system can have different releases of the Oracle Database, and all supported releases of the Oracle Database can participate in a distributed database system. However, the applications that work with a distributed database must understand the functionality that is available at each node in the system.

Note: Since this book documents upgrading and downgrading between different releases of the Oracle Database, this definition of interoperability is appropriate. However, other Oracle Database documentation may use a broader definition of the term **interoperability**; for example, in some cases, interoperability may describe communication between different hardware platforms and operating systems.

Compatibility and Interoperability Issues Introduced in Oracle Database 10*g* Release 10.1

The following sections describe compatibility and interoperability issues introduced in Oracle Database 10*g* release 10.1. If you are upgrading to the new Oracle Database 10*g* release, then the sections which follow discuss actions you can take to prevent problems resulting from these issues.

SQL Optimizer

This section describes compatibility and interoperability issues relating to the SQL Optimizer in Oracle Database 10*g*.

Rule-Based Optimizer Desupported

Starting with Oracle Database 10*g* release 10.1, the cost-based optimizer (CBO) is now enabled by default. The rule-based optimizer is no longer supported in release 10.1. As a result, `rule` and `choose` are no longer supported as `OPTIMIZER_MODE` initialization parameter values and a warning is displayed in the alert log if `OPTIMIZER_MODE` is set to either of these values.

> **See Also:** *Oracle Database Performance Tuning Guide* for more information about the cost-based optimizer

Optimizer Statistics

Collection of optimizer statistics is now automatically performed by default for all schemas (including `SYS`), for pre-existing databases upgraded to Oracle Database 10*g*, and for newly created Oracle Database 10*g* databases. Gathering optimizer statistics on stale objects is scheduled by default to occur daily during the maintenance window.

> **See Also:** *Oracle Database Performance Tuning Guide* for more information about optimizer statistics

COMPUTE STATISTICS Clause of CREATE INDEX

In earlier releases, the `COMPUTE STATISTICS` clause of `CREATE INDEX` could be used to start or stop the collection of statistics on an index. This clause has been deprecated. Oracle Database 10*g* now automatically collects statistics during index

creation and rebuild. This clause is supported for backward compatibility and will not cause errors.

SKIP_UNUSABLE_INDEXES

In earlier releases, `SKIP_UNUSABLE_INDEXES` was a session parameter only. In Oracle Database 10*g* release 10.1 and later, it is now an initialization parameter and defaults to `true`. The `true` setting disables error reporting of indexes and index partitions marked UNUSABLE. This setting allows all operations (inserts, deletes, updates, and selects) on tables with unusable indexes or index partitions.

See Also: `SKIP_UNUSABLE_INDEXES` in *Oracle Database Reference*

SQL

Starting with Oracle Database 10*g* release 10.1, CLOB <-> NCLOB implicit conversion in SQL and PL/SQL is allowed.

Starting with release 10.1, name resolution for synonyms has changed. If the base object of a synonym does not exist, the SQL compiler now tries looking up `PUBLIC`.base_object.

Starting with release 10.1, VPD policies are attached to synonyms rather than the base objects.

Invalid Synonyms After an Upgrade

Starting with Oracle Database 10*g* release 10.1, if a synonym (public or private) is pointing to an object that does not exist or is invalid, then the synonym will be invalid after the upgrade.

Manageability

Database performance statistics are now automatically collected by the Automatic Workload Repository (AWR) database component for databases upgraded to Oracle Database 10*g* and for newly created Oracle Database 10*g* databases. This data is stored in the `SYSAUX` tablespace, and is used by the database for automatic generation of performance recommendations.

See Also: *Oracle Database Performance Tuning Guide*

If you currently use Statspack for performance data gathering, then refer to the Statspack README (*`ORACLE_HOME`*`/rdbms/admin/spdoc.txt`) for directions on using Statspack in Oracle Database 10*g* to avoid conflict with the AWR.

Transaction and Space

Starting with Oracle Database 10*g* release 10.1, dropped objects are now moved to the recycle bin where the space is only reused when it is needed. This allows an object to be undropped using the `FLASHBACK DROP` feature.

See Also: *Oracle Database Administrator's Guide*

Starting with release 10.1, automatic tuning of undo retention is enabled by default. The `UNDO_SUPPRESS_ERRORS` initialization parameter has been deprecated. Errors generated when executing rollback segment operations while in automatic undo management mode will always be suppressed.

Starting with release 10.1, the default AUTOEXTEND NEXT size is larger for Oracle-managed files (OMF).

See Also: *Oracle Database SQL Reference*

Recovery and Data Guard

Starting with Oracle Database 10*g* release 10.1, the LOG_ARCHIVE_START initialization parameter has been deprecated. Archiving is now automatically started when the database is placed in ARCHIVELOG mode.

Starting with release 10.1, the LOG_PARALLELISM initialization parameter has been deprecated. Log file parallelism is now automatically enabled.

Starting with release 10.1, the default value for the RECOVERY_PARALLELISM initialization parameter now defaults to allow parallel recovery.

Starting with release 10.1, the default value for the parallel clause in the ALTER DATABASE RECOVER DATABASE statement has changed to PARALLEL.

See Also: *Oracle Database SQL Reference*

Starting with release 10.1, the default buffer size for the ASYNC attribute of the LOG_ARCHIVE_DEST_*n* initialization parameter has increased from 2,048 blocks to 61,440 blocks.

Starting with release 10.1, the default values of the parameters MAX_SGA and MAX_SERVERS as set by the DBMS_LOGSTDBY.APPLY_SET() procedure have changed.

See Also: *PL/SQL Packages and Types Reference*

Starting with release 10.1, the default values for the Data Guard broker properties ApplyParallel, AsyncBlocks, and LogXptMode have changed.

See Also: *Oracle Data Guard Concepts and Administration*

Starting with release 10.1, the default behavior of the STARTUP SQL*Plus command and the ALTER DATABASE MOUNT and ALTER DATABASE OPEN SQL statements have changed for physical standby databases. The commands now automatically detect that the database is a physical standby and thus the STANDBY DATABASE and READ ONLY options are made default.

See Also: *Oracle Database SQL Reference*

RMAN

Starting with Oracle Database 10*g* release 10.1, RMAN now creates an empty file when restoring a file from backup and no backup of the file exists. RMAN backup of archived logs now automatically backs up logs that were created before the last resetlogs. Such logs were previously ignored.

Starting with release 10.1, RMAN now continues to run the remaining portions of a backup or restore job when it encounters an error. RMAN now tries to restore from an alternate backup if it finds the targeted backup is corrupt.

CREATE DATABASE

In Oracle Database 10*g*, a SYSAUX tablespace is always created at database creation time or whenever a database is upgraded. The SYSAUX tablespace serves as an

auxiliary tablespace to the SYSTEM tablespace. Because SYSAUX is the default tablespace for many Oracle features and products that previously required their own tablespaces, it reduces the number of tablespaces that a DBA must maintain.

> **See Also:** *Oracle Database Administrator's Guide* for more information about the SYSAUX tablespace

Starting with release 10.1, the minimum and default logfile sizes have increased. The minimum size is now 4 MB. The default size is 50 MB, unless using Oracle-managed files (OMF) in which case the default is 100 MB.

Real Application Clusters

In Oracle Database 10*g*, there is now an automated high availability (HA) framework for Real Application Clusters. The framework provides detection, recovery, restart, and notification services.

> **See Also:** *Oracle Real Application Clusters Administrator's Guide* for more information

Materialized Views

Starting with Oracle Database 10*g* release 10.1, some privilege name changes have been made. The new names appear in all data dictionary views, but both the old and new names are accepted by the GRANT and REVOKE SQL statements.

- CREATE SNAPSHOT changed to CREATE MATERIALIZED VIEW
- CREATE ANY SNAPSHOT changed to CREATE ANY MATERIALIZED VIEW
- ALTER ANY SNAPSHOT changed to ALTER ANY MATERIALIZED VIEW
- DROP ANY SNAPSHOT changed to DROP ANY MATERIALIZED VIEW

Change Data Capture

Starting with Oracle Database 10*g* release 10.1, the interfaces in DBMS_CDC_SUBSCRIBE and DBMS_CDC_PUBLISH now take a subscription name parameter instead of a subscription handle.

> **See Also:** *PL/SQL Packages and Types Reference*

Starting with release 10.1, subscriber views are now managed automatically. There is no longer any need to call the DBMS_CDC_SUBSCRIBE and DBMS_CDC_PUBLISH interfaces PREPARE_SUBSCRIBER_VIEW() and DROP_SUBSCRIBER_VIEW().

Starting with release 10.1, the computation of synchronous Change Data Capture's RSID$ column has been changed to facilitate joining a subscriber view to itself in order to show both old and new values in the same row. The RSID$ values for the UO and UN rows associated with the same update operation are now the same. To revert to the Oracle9*i* behavior where UN RSID$ value is UO RSID$ value + 1 for the same update operation, set event 10983 to level 4.

Change in the Default Archival Processing to Remote Archive Destinations

Starting with Oracle Database 10*g* release 10.1, the default archival processing to remote destinations has changed so that archiver processes on the primary database will completely and successfully archive the local online redo log files before

transmitting the redo data to remote standby destinations. This default behavior is equivalent to setting the LOG_ARCHIVE_LOCAL_FIRST initialization parameter to true, which is also new in release 10.1. Note that this new default archival processing is relevant only when log transport services are defined to use archiver processes (ARC*n*), not the log writer process (LGWR), when the archiver processes are writing to remote destinations, and when the remote standby destination is not a mandatory destination.

Prior to release 10.1, the default behavior was to transmit redo data to the standby destination at the same time the online redo log file was being archived to the local online redo log files. You can achieve this behavior by setting the LOG_ARCHIVE_LOCAL_FIRST initialization parameter to false. This archival processing is also relevant only when log transport services are defined to use archiver processes (ARC*n*), not the log writer process (LGWR), when the archiver processes are writing to remote destinations, and when the remote standby destination is not a mandatory destination.

The benefit of the new default behavior is that local archiving, and hence, processing on the primary database, are not affected by archival to non-mandatory, remote destinations. Because local archiving is now disassociated with remote archiving, sites that may have policies to delete archived redo log files on the primary database immediately after backing them up must make sure that the standby destinations have received the corresponding redo data before deleting the archived redo log files on the primary database. You can query the V$ARCHIVED_LOG view to verify that the redo data has been received on standby destinations.

Note: Any value specified for the LOG_ARCHIVE_LOCAL_FIRST initialization parameter is ignored for mandatory destinations (configured with the MANDATORY attribute of the LOG_ARCHIVE_DEST_*n* initialization parameters).

See Also: *Oracle Data Guard Concepts and Administration* for complete information about setting up archival to remote destinations

Compatibility and Interoperability Issues Introduced in Oracle9*i* Release 9.2

The following sections describe compatibility and interoperability issues introduced in Oracle9*i* release 9.2. If you are upgrading to the new Oracle Database 10*g* release from a release earlier than release 9.2, then the sections which follow discuss actions you can take to prevent problems resulting from these issues.

- Locally Managed SYSTEM Tablespace
- New AnyData DAtatypes
- Dictionary Managed Tablespaces
- Change in Compatibility for Automatic Segment-Space Managed Tablespaces
- Compatibility and Object Types
- Oracle Managed Files
- Oracle OLAP
- Log Format Change with Parallel Redo

- Oracle Dynamic Services
- Oracle Syndication Server

Locally Managed SYSTEM Tablespace

Starting with Oracle9*i* release 9.2, the `SYSTEM` tablespace can be locally managed. The `SYSTEM` tablespace can be migrated from dictionary managed format to locally managed format using the `DBMS_SPACE_ADMIN.TABLESPACE_MIGRATE_TO_LOCAL` procedure.

Before the `SYSTEM` tablespace can be migrated to locally managed format, you should ensure the following:

- The database has a default temporary tablespace which is not `SYSTEM`
- There are not any rollback segments in dictionary managed tablespaces
- There is at least one online rollback segment in a locally managed tablespace, or an undo tablespace (if using automatic undo management mode) should be online.
- All tablespaces other than the tablespace containing the undo space (undo tablespace or the tablespace containing the rollback segment) and the default temporary tablespace are in read-only mode.
- There is a complete backup of the system.
- The system is in restricted mode.

The following query determines whether the `SYSTEM` tablespace is locally managed:

```
SQL> SELECT ts# FROM ts$
         WHERE ts# = 0 AND bitmapped <> 0;
```

If `0` rows are returned, then the `SYSTEM` tablespace is dictionary managed. Otherwise, the `SYSTEM` tablespace is locally managed.

New AnyData DAtatypes

Starting with Oracle9*i* release 9.2, persistent storage of `AnyData` values of the following datatypes is allowed:

- `TIMESTAMP`
- `TIMESTAMP WITH TIME ZONE`
- `TIMESTAMP WITH LOCAL TIME ZONE`
- `INTERVAL YEAR TO MONTH`
- `INTERVAL DAY TO SECOND`
- `NCHAR`
- `NVARCHAR2`
- `NCLOB`

Dictionary Managed Tablespaces

Starting with Oracle9*i* release 9.2, dictionary managed tablespaces are deprecated. Once the `SYSTEM` tablespace has been migrated from dictionary managed format to locally managed format, existing dictionary managed tablespaces are read-only. That is, they cannot be made read-write once the `SYSTEM` tablespace is locally managed.

Once the SYSTEM tablespace is locally managed (either due to a new installation or SYSTEM tablespace migration), new dictionary managed tablespaces cannot be created.

Change in Compatibility for Automatic Segment-Space Managed Tablespaces

Starting with Oracle9*i* release 9.0.1.3.0, the compatibility requirement for automatic segment-space managed tablespaces has been changed from 9.0.0 when first introduced in Oracle9*i* release 9.0.1 to 9.0.1.3. If you are upgrading from an Oracle9*i* release earlier than release 9.0.1.3.0 and the database contains any automatic segment-space managed tablespaces, then the COMPATIBLE initialization parameter will need to be set to 9.0.1.3 or higher in order to open the database. The existing tablespaces need not be dropped.

Compatibility and Object Types

Starting with Oracle9*i* release 9.2, object types support user-defined constructors using the CONSTRUCTOR keyword that cannot be referred to from PL/SQL programs in previous releases of the Oracle Database. Specifically, such programs will fail to compile with an error.

Oracle Managed Files

Starting with Oracle9*i* release 9.0.1.2.0, the naming scheme used by the Oracle Database to keep track of Oracle Managed Files has changed. As a result, existing Oracle Managed Files created in Oracle9*i* releases earlier than release 9.0.1.2.0 will appear to the Oracle Database to be regular operating system files. See "Migrate Your Oracle Managed Files" on page 4-2 for information on migrating your Oracle Managed Files to the new naming scheme.

Oracle OLAP

The OLAP API client provided with Oracle9*i* release 9.0.1 is not compatible with newer Oracle Database releases; similarly, the OLAP API client provided with Oracle9*i* release 9.2 is not compatible with earlier Oracle Database releases.

The procedure that an application uses to make a connection through the OLAP API has changed in release 9.2. Connections in previous releases relied on CORBA software, but in release 9.2 and later, connections are made through Java Database Connectivity (JDBC). Consequently, programs created using the OLAP API client provided with release 9.0.1 will not execute in later releases, and programs created using the OLAP API client provided with release 9.2 will not execute in earlier Oracle releases.

To upgrade OLAP API applications designed to run in release 9.0.1, application developers must use the OLAP API client provided with release 9.2 and revise the code for making a connection and for creating a MetadataProvider.

For information about using the OLAP API in release 9.2 to perform these actions, see the *Oracle OLAP Developer's Guide to the OLAP API* and the online Oracle OLAP API Reference help.

Log Format Change with Parallel Redo

Starting with Oracle9*i* release 9.2, the parallel redo feature generates redo logs using a new format. Previous releases of the Oracle Database cannot apply parallel redo

generated logs. However, when previous Oracle Database releases detect that release 9.2 parallel redo is being applied, the following error is displayed:

```
ORA-00303: cannot process Parallel Redo
```

Release 9.2 can process Oracle9*i* release 9.0.1 and earlier format logs as well as release 9.2 parallel redo format logs.

Oracle Dynamic Services

Starting with Oracle9*i* Database release 9.2, Oracle Dynamic Services has been Deprecated. Oracle Dynamic Services, an XML-based broker for the creation, aggregation, and deployment of services from various content sources, was released with Oracle9*i* Database release 9.0.1 along with the documentation, *Oracle Dynamic Services User's and Administrator's Guide*.

Starting with Oracle9*i*AS release 2 (9.0.2), Oracle is delivering an integrated, J2EE-compliant Web Services platform. Oracle Dynamic Services has been integrated with Oracle9*i*AS Web Services as the XML/HTML Stream Processing Tool.

> **See Also:** *Oracle9i Application Server Web Services Developer's Guide* for more information

Oracle9*i*AS release 2 (9.0.2) provides a standards-based, fully integrated J2EE and Web services deployment platform. The current Dynamic Services functionality has been integrated into the Oracle9*i*AS platform, and the Dynamic Services terminal release is being delivered with Oracle9*i* Database release 9.2.

Oracle Syndication Server

Starting with Oracle9*i* Database release 9.2, Oracle Syndication Server has been Deprecated. Oracle Syndication Server, designed to deliver file system and database content to Information and Content Exchange (ICE)-compliant subscribers, was released with Oracle9*i* Database release 9.0.1 along with the documentation, *Oracle Syndication Server User's and Administrator's Guide*.

Starting with Oracle9*i*AS release 2 (9.0.2), Oracle Syndication Server has become a feature of Oracle9*i*AS. The current Syndication Server functionality has been integrated into this platform, and the Syndication Server terminal release is being delivered with Oracle9*i* Database release 9.2.

Oracle9*i*AS Syndication Server is automatically installed with the Oracle9*i*AS Portal install. The current release of the *Oracle Syndication Server User's and Administrator's Guide* can be found with the Oracle9*i*AS Portal documentation on the Oracle9*i*AS release 2 (9.0.2) Documentation CD-ROM.

Compatibility and Interoperability Issues Introduced in Oracle9*i* Release 9.0.1

The following sections describe compatibility and interoperability issues introduced in Oracle9*i* release 9.0.1. If you are upgrading to the new Oracle Database 10*g* release from a release earlier than release 9.0.1, then the sections which follow discuss actions you can take to prevent problems resulting from these issues.

The STARTUP Command

This section describes compatibility and interoperability issues related to the SQL*Plus `STARTUP` command.

Change in Default Parameter File Selection

When the `STARTUP` command is issued without the `PFILE` option, the Oracle Database attempts to start up the instance using a default parameter file. Starting with Oracle9*i* release 9.0.1, the search criteria for selecting the default parameter file has changed to facilitate the use of a server parameter file.

In previous releases of the Oracle Database, the `STARTUP` command looked for an initialization parameter file with the name *`ORACLE_HOME`*`/dbs/init`*`SID`*`.ora`, where *`SID`* is the instance name.

In release 9.0.1 and later, the process of selecting a default parameter file is as follows:

- The `STARTUP` command first looks for a server parameter file with the name *`ORACLE_HOME`*`/dbs/spfile`*`SID`*`.ora`, where *`SID`* is the instance name.
- The `STARTUP` command next looks for a server parameter file with the name *`ORACLE_HOME`*`/dbs/spfile.ora`.
- If the `STARTUP` command cannot find a server parameter file, it defaults to the behavior of the `STARTUP` command in previous releases, and looks for an initialization parameter file with the name *`ORACLE_HOME`*`/dbs/init`*`SID`*`.ora`.

> **See Also:** *Oracle Database Administrator's Guide* for more information about server parameter files

Tablespaces and Datafiles

This section describes compatibility and interoperability issues related to tablespaces and datafiles.

CREATE TABLESPACE: New Behavior

In Oracle8*i*, the default type of tablespace that is created is dictionary managed if the `EXTENT MANAGEMENT` clause is not specified in the `CREATE TABLESPACE` statement.

Starting with Oracle9*i* release 9.0.1, the default for the `EXTENT MANAGEMENT` clause is locally managed. The default storage clause is parsed to determine whether to use `AUTOALLOCATE` or `UNIFORM` allocation policy for this tablespace.

In addition, there was another change made to disallow assigning permanent locally managed tablespaces as a user's temporary tablespace. In Oracle8*i*, an error would be signalled only when a temporary segment had to be created in the tablespace.

Default Temporary Tablespaces

Oracle strongly recommends using a default temporary tablespace for the database. The default temporary tablespace should be created using the `CREATE TEMPORARY TABLESPACE` statement.

Undo Tablespaces

Oracle Database instances can run in one of two undo space management modes:

- Automatic undo management mode
- Manual undo management mode

All instances of the same database must run in the same undo space management mode.

The instance is started in manual undo management mode if the UNDO_MANAGEMENT initialization parameter is not specified.

In the manual undo management mode, CREATE, ALTER, and DROP operations on undo tablespaces are allowed. Rollback segments can coexist with undo tablespaces. That is, rollback segments can exist while running in automatic undo management mode and undo tablespaces can exist while running in manual undo management mode. Undo tablespaces cannot be brought online unless the instance is running in automatic undo management mode.

In automatic undo management mode, DROP ROLLBACK SEGMENT operations are allowed. Rollback segments cannot be brought online.

> **See Also:** *Oracle Database Administrator's Guide* for more information about managing undo space.

Datatypes

This section describes compatibility and interoperability issues relating to datatypes.

Datetime and Interval Datatypes

When a database is upgraded to Oracle9*i* release 9.0.1 or later, the database time zone is set to the time zone of the environment variable ORA_SDTZ. If ORA_SDTZ is not set, the database time zone is set to the time zone of the operating system clock. If the time zone of the operating system clock is not set or is not valid, the database time zone defaults to UTC.

old Oracle Database DATE data with time portion can be migrated to either TIMESTAMP to support fractional seconds or TIMESTAMP WITH LOCAL TIME ZONE to support time zone adjustments in addition to fractional seconds without having legacy data rewritten. An ALTER TABLE statement must be explicitly issued to modify a DATE column to a TIMESTAMP column or a TIMESTAMP WITH LOCAL TIME ZONE column.

Database Character Sets

In Oracle8*i* and earlier releases, the SQL NCHAR datatypes (NCHAR, NVARCHAR2, and NCLOB) will be limited to the Unicode character set encoding (UTF8 and AL16UTF16) only. Any other character sets that were available under the NCHAR data type, including Asian character sets (such as JA16SJISFIXED), will no longer be supported.

Before migrating your SQL NCHAR data to the new Unicode NCHAR, Oracle recommends that you analyze your SQL NCHAR data, using the Character Set Scanner for the identification of possible invalid character set conversion or data truncation.

> **See Also:** *Oracle Database Globalization Support Guide* for more information about the Character Set Scanner

When you upgrade to Oracle Database 10g, the value of the National Character Set of the upgraded database is set based on the value of the National Character Set of the database being upgraded.

If the old National Character Set is UTF8, then the new National Character Set will be UTF8. Otherwise, the National Character Set is changed to AL16UTF16.

During the upgrade, the existing NCHAR columns in the data dictionary are changed to use the new Oracle Database format and, if the National Character Set has been changed to AL16UTF16, the dictionary NCHAR columns will be converted to the AL16UTF16 character set.

> **Note:** NCHAR columns in user tables are not changed during the upgrade. To change NCHAR columns in user tables, see "Upgrade User NCHAR Columns" on page 4-8.

AL24UTFFSS Character Set Desupported

The AL24UTFFSS Unicode character set has been desupported in Oracle9*i* release 9.0.1 and later. AL24UTFFSS was introduced in Oracle7 as the Unicode character set supporting the UTF-8 encoding scheme based on the Unicode 1.1 standard, which is now obsolete. In release 9.0.1 and later, The Unicode database character sets AL32UTF8 and UTF8, include the Unicode enhancements based on the Unicode 3.1 standard.

The migration path for existing AL24UTFFSS databases is to upgrade your database character set to UTF8 prior to upgrading your Oracle Database. As with all migrations to a new database character set, Oracle recommends you use the Character Set Scanner for data analysis before attempting to migrate your existing database character set to UTF8.

> **See Also:** *Oracle Database Globalization Support Guide* for more information about the Character Set Scanner

User-Defined Datatypes

This section describes compatibility and interoperability issues relating to user-defined datatypes.

Subtypes and Non-Final Types

Types created in Oracle8*i* release 8.1 and earlier are considered to be FINAL types. Thus, they cannot be used as supertypes in Oracle9*i* release 9.0.1 and later. However, an ALTER statement can be explicitly used to change the type to be NOT FINAL.

Release 8.1 Clients Accessing a Release 9.0.1 or Higher Server Any transfer involving data of non-final types will return an error. Release 8.1 clients cannot access a release 9.0.1 or higher server if the type has been altered to non-final on the server.

Release 9.0.1 and Higher Clients Accessing a Release 8.1 Server Since the release 8.1 server can have only non-final types, no errors occur.

Oracle Replication

If you plan to use CHAR column length semantics in Oracle Database 10g, or if your replication database contains tables with NCHAR or NVARCHAR2 columns, then this section contains considerations for upgrading a replication environment to Oracle Database 10g.

CHAR Column Length Semantics

If you plan to use CHAR column length semantics in a replication database after you upgrade it to Oracle Database 10g, then all of the databases participating with that

database in the replication environment must also use CHAR column length semantics. In this case, Oracle recommends that you upgrade all of the databases participating in the replication environment at the same time. This applies to both master sites and materialized view sites in your replication environment.

If you cannot upgrade all of the databases in your replication environment at the same time, then you can only use CHAR column length semantics in your Oracle Database if all of the databases prior to Oracle9*i* are using a single-byte character set. Otherwise, do not switch to CHAR column length semantics in the Oracle Database until all of the other databases in the replication environment are upgraded to Oracle Database 10g.

NCHAR or NVARCHAR2 Columns

If your replication database contains tables with NCHAR or NVARCHAR2 columns, then Oracle recommends that you upgrade all of the databases participating in the replication environment at the same time. This applies to both master sites and materialized view sites in your replication environment. In Oracle Database 10g, all columns specified as NCHAR or NVARCHAR2 datatype are stored in Unicode format.

If you cannot upgrade all of the databases in your replication environment at the same time, then interoperability is only supported if all of the databases prior to Oracle9*i* are using a fixed width national character set. If any of the databases prior to Oracle9*i* are using a variable width character set, then you must convert these databases to fixed width character sets before you upgrade any of the other databases in the replication environment to Oracle Database 10g.

See Also:

- *Oracle Database Advanced Replication* for more information about replication support for column length semantics and Unicode
- *Oracle Database Globalization Support Guide* for general information about column length semantics and Unicode
- *Oracle8i National Language Support Guide* for information about converting character sets in release 8.1

Compatibility and Interoperability Issues Introduced in Oracle8*i* Release 8.1

The following sections describe compatibility and interoperability issues introduced in Oracle8*i* release 8.1.*x*. If you are upgrading to the new Oracle Database 10*g* release from a release earlier than release 8.1, then the sections which follow discuss actions you can take to prevent problems resulting from these issues.

Applications

You do not need to modify existing applications that do not use new release 10.1 features. Existing applications should achieve the same, or enhanced, functionality on release 10.1. To increase the likelihood that applications running against your release 10.1 database will continue to work if you downgrade to a previous release, you can leave the COMPATIBLE initialization parameter set to 9.2.0 after the upgrade.

However, the COMPATIBLE initialization parameter only restricts the use of release 10.1 features that change the formatting on disk, not the use of other release 10.1 features. Therefore, a setting of 9.2.0 does not guarantee that applications developed in release 10.1 will run correctly if the database is downgraded to a previous release.

> **See Also:** Chapter 6, "Upgrading Your Applications" for more information about upgrading applications

General Compatibility and Interoperability Issues for Applications

This section describes general compatibility and interoperability issues for applications.

Index-Organized Tables Accessed by Applications If a table accessed by an application changes from a regular table to an index-organized table, then the application may require changes. The possible changes depend on whether the application uses physical rowids or universal rowids (UROWIDs).

Whether an application requires changes depends on the kind of host variables the application is using to bind or define rowid values:

- If the application uses release 8.0 or higher OCI rowid descriptors (OCIROWID * for Pro*C and SQL-ROWID for Pro*COBOL), then the application should continue to function properly without any changes.
- If the application always performs `DESCRIBE` on the host variables, then the application should continue to function properly without any changes. Make sure the application can accommodate the new SQLT_RDD datatype.
- If the application uses SQLT_RID host variables, then you must rewrite the application to use VARCHAR host variables or rowid descriptors. Rowid descriptors are preferred.
- If the application uses CHARACTER host variables, then the behavior also depends on the size of the host variables. If the size can accommodate the primary key and if the variable is a variable length string, then the application should continue to function properly without any changes. However, if the application uses a fixed size 18 character string, then you must change the application to use longer variable strings or OCI descriptors.

For applications using UROWIDs, VARCHAR host variables may no longer be large enough to hold the rowids. If so, then change the application to increase the variable maximum size or change the application to use OCI rowid descriptors. OCI rowid descriptors are preferred because they are opaque and resize automatically.

Change in Behavior for ANALYZE TABLE VALIDATE STRUCTURE Statement Starting with release 8.1, the `ANALYZE TABLE VALIDATE STRUCTURE` statement no longer stops running at the first error. Modify any applications that depend on this behavior to account for this change.

OCI Applications

This section describes compatibility and interoperability issues relating to OCI applications.

> **See Also:** *Oracle Call Interface Programmer's Guide* for more information.

Shared Structures and Interoperability Shared structures are not supported on Oracle7 clients linked with release 8.1 libraries. To take advantage of shared structures, applications must be written with the release 8.1 or higher OCI and must be communicating with a release 8.1 or higher Oracle database server.

A release 8.1 OCI client accessing a release 8.0 Oracle database server only partially realizes the benefits of shared structures, and shared structures are not supported if both the client and the Oracle database server are release 8.0 or lower.

Using Batch Error Mode for Statement Execution Starting with release 8.1, OCI applications can use the batch error mode when executing array DMLs using *OCIStmtExecute*. To do this, both the OCI and server libraries must be release 8.1 or higher.

You can modify existing applications to use batch error mode by setting the mode parameter to OCI_BATCH_ERRORS and adding new code required for this functionality. Then, recompile and relink the application with the release 8.1 client libraries.

Support for Client Notification Starting with release 8.1, client notification is supported in OCI applications using the publish/subscribe interface. Client notification enables applications to take advantage of Database Event Publication and Advanced Queuing features. To use the client notification feature, client applications must link with release 8.1 or higher client libraries.

Support for the LISTEN Call with the Advanced Queuing Option Starting with release 8.1, the LISTEN call is supported in OCI applications. The LISTEN call is available with the Advanced Queuing Option and can be used to monitor a set of queues for a message. To use the LISTEN call, client applications must link with release 8.1 or higher client libraries.

Precompiler Applications

This section describes compatibility and interoperability issues relating to precompiler applications.

> **See Also:** *Pro*C/C++ Programmer's Guide* and *Pro*COBOL Programmer's Guide* for more information.

Connecting With SYSDBA Privileges in Pro*C/C++ `SYSDBA` privileges are no longer available by default when you issue the `CONNECT` statement in Pro*C/C++. In release 8.0, the following `CONNECT` statement connected with `SYSDBA` privileges in Pro*C/C++:

```
EXEC SQL CONNECT :sys IDENTIFIED BY :sys_passwd;
```

In release 8.1 and higher, issue the following `CONNECT` statement to connect with `SYSDBA` privileges in Pro*C/C++:

```
EXEC SQL CONNECT :sys IDENTIFIED BY :sys_passwd IN SYSDBA MODE;
```

Connecting With SYSDBA Privileges in Pro*COBOL `SYSDBA` privileges are no longer available by default when you issue the `CONNECT` statement in Pro*COBOL. In release 8.0, the following `CONNECT` statement connected with `SYSDBA` privileges:

```
EXEC SQL
    CONNECT :sys IDENTIFIED BY :SYS-PASSWD
END-EXEC.
```

In release 8.1 and higher, issue the following `CONNECT` statement to connect with `SYSDBA` privileges:

```
EXEC SQL
    CONNECT :sys IDENTIFIED BY :SYS-PASSWD IN SYSDBA MODE
END-EXEC.
```

PL/SQL Backward Compatibility and Precompilers PLSQL_V2_COMPATIBILITY backward compatibility behavior is available in the precompiler environment by setting the DBMS precompiler command line option as follows:

```
... DBMS=Oracle7
```

PL/SQL Applications

This section includes compatibility and interoperability issues for PL/SQL applications.

> **See Also:** *PL/SQL User's Guide and Reference* for more information

Integrated SQL Analysis Syntax and semantic analysis of SQL statements in PL/SQL programs is now integrated with the SQL engine. As a result, any new SQL feature that is available through SQL*Plus or OCI is also available in PL/SQL.

In Oracle Database, syntax and semantic analysis of SQL statements is also a little stricter than in previous releases. PL/SQL catches additional errors in SQL statements during compilation itself, rather than throwing a runtime exception for invalid SQL syntax. As a result, you may see compile-time errors with the PL/SQL:ORA- prefix in PL/SQL programs that had compiled successfully in previous releases. The new error messages point to problems in the SQL statement that must be fixed before the program can be compiled successfully.

If you are unable to immediately modify a SQL statement to satisfy the new stricter checks, Oracle provides an event to temporarily assist you in migrating PL/SQL code to Oracle Database:

```
ALTER SESSION SET events = '10933 trace name context forever, level 512';
```

This event is provided only for temporary migration assistance. Oracle strongly discourages long-term use of this event, and this event will be desupported in the next major release of Oracle.

If you are upgrading from release 8.1.7 and this event exists in your parameter file, then, as a temporary workaround, change all occurrences of this event from `event = '10933 trace name context forever, level 512'` to `event = '10933 trace name context forever, level 1024'`.

Default Value of Parameter for Functions or Procedures in the Spec and Body Do Not Match In previous releases, PL/SQL quietly ignored this error and used the default value specified in the spec (ignoring the possibly different value in the body). Also, if there is no default value specified in the spec, and a default value is specified in the body, then the default value in the body is ignored.

In Oracle Database, PL/SQL will flag such discrepancies as errors. It is recommended to fix the code, if such errors are reported, to avoid any possible future bugs.

If you are unable to immediately modify the PL/SQL code, then Oracle provides an event to temporarily restore the old compiler behavior:

```
ALTER SESSION SET events = '10932 trace name context level 32768'
```

This event is provided only for temporary migration assistance. Oracle strongly discourages long-term use of this event, and this event will be desupported in the next major release of Oracle.

Compatibility and Object Types In Oracle Database, object types that are qualified as `NOT FINAL`, `NOT INSTANTIABLE`, a subtype, or a SQLJ type cannot be referred to from

PL/SQL programs in earlier releases of Oracle. Specifically, such programs will fail to compile with an error.

PL/SQL V2 Compatibility Mode The PL/SQL V2 compatibility mode is available in PL/SQL release 8.0 and higher. This mode is enabled by the `PLSQL_V2_COMPATIBILITY` initialization parameter.

You can set PL/SQL V2 compatibility mode in any one of the following three ways:

- Add the following line to your initialization parameter file:

  ```
  PLSQL_V2_COMPATIBILITY = true
  ```

- Issue the following SQL statement:

  ```
  ALTER SYSTEM SET PLSQL_V2_COMPATIBILITY = true;
  ```

- Issue the following SQL statement:

  ```
  ALTER SESSION SET PLSQL_V2_COMPATIBILITY = true;
  ```

The `PLSQL_V2_COMPATIBILITY` initialization parameter provides compatibility between PL/SQL release 8.0 and higher and PL/SQL V2 in the following situations:

- The PL/SQL V2 compiler allows a record type or index table type to be referenced before its definition in the source. PL/SQL release 8.0 and higher strictly requires that the type definition precede reference to the type in the source. However, when you enable PL/SQL V2 compatibility mode, PL/SQL release 8.0 and higher behaves the same as PL/SQL V2 regarding type definitions.
- The PL/SQL V2 compiler allows the following illegal syntax:

  ```
  return variable-expression
  ```

  This syntax is incorrect and should be changed to the following:

  ```
  return variable-type
  ```

  The PL/SQL release 8.0 and higher compiler issues an error when it encounters the illegal syntax. However, when you enable PL/SQL V2 compatibility mode, PL/SQL release 8.0 and higher behaves the same as PL/SQL V2 and does not issue an error.
- In PL/SQL V2 it is possible to modify or delete elements of an index table passed in as an IN parameter, as in the following example:

  ```
  function foo (x IN table_t) is
  begin
  x.delete(2);
  end;
  ```

  This use of an IN parameter is incorrect. PL/SQL release 8.0 and higher correctly enforces the read-only semantics of IN parameters and does not let index table methods modify index tables passed in as IN parameters. However, when you enable PL/SQL V2 compatibility mode, PL/SQL release 8.0 and higher behaves the same as PL/SQL V2 and allows the parameter.
- PL/SQL V2 allows the passing (as an OUT parameter) of fields of IN parameters that are records, but PL/SQL release 8.0 and higher does not allow this type of passing. However, when you enable PL/SQL V2 compatibility mode, PL/SQL release 8.0 and higher behaves the same as PL/SQL V2 and allows this type of passing.

- The PL/SQL V2 compiler permits fields of OUT parameters that are record variables to be used in expression contexts (for example, in a dot-qualified name on the right-hand side of an assignment statement).

  This use of OUT parameters should not be permitted. PL/SQL release 8.0 and higher does not permit OUT parameters to be used in expression contexts. However, when you enable PL/SQL V2 compatibility mode, PL/SQL release 8.0 and higher behaves the same as PL/SQL V2 in this regard.

- PL/SQL V2 allows OUT parameters in the `FROM` clause of a `SELECT` list. PL/SQL release 8.0 and higher does not allow this use of OUT parameters. However, when you enable PL/SQL V2 compatibility mode, PL/SQL release 8.0 and higher behaves the same as PL/SQL V2 in this regard.

Tablespaces and Datafiles

This section describes compatibility and interoperability issues related to tablespaces and datafiles.

Transportable Tablespace

There are compatibility issues when you transport a tablespace between two databases.

> **See Also:** *Oracle Database Administrator's Guide* for information about these compatibility issues.

Tempfiles

Release 8.1 introduced tempfiles. The information about tempfiles is in different static data dictionary views and dynamic performance views than the information about datafiles. To view information about tempfiles, consult the `DBA_TEMP_FILES` static data dictionary view and the following dynamic performance views:

- `V$TEMPFILE`
- `V$TEMP_EXTENT_MAP`
- `V$TEMP_EXTENT_POOL`
- `V$TEMP_SPACE_HEADER`
- `V$TEMPSTAT`
- `V$TEMP_PING`

Oracle automatically assigns numbers to both datafiles and tempfiles. Two datafiles cannot share the same number; similarly, two tempfiles cannot share the same number. However, a tempfile and a datafile can share the same number.

> **See Also:** *Oracle Database SQL Reference* for information about tempfiles

Data Dictionary

This section describes possible compatibility and interoperability issues resulting from data dictionary changes.

> **See Also:** Appendix A, "Initialization Parameter and Data Dictionary Changes" for more information about obsolete and deprecated dictionary views

Data Dictionary Protection

The data dictionary protection mechanism introduced in release 8.0 may cause problems in any applications that create user tables in the SYS schema and access them using the '*ANY*' privileges. For example, the user must have DELETE CATALOG ROLE to use the DELETE statement to purge the audit records in the AUD$ table.

Creating and accessing user tables in SYS schema is not secure. Therefore, applications are expected to move the objects to a different schema. Use the O7_DICTIONARY_ACCESSIBILITY initialization parameter for temporary compatibility. However, this parameter is only for interim use.

Applications should not attempt to connect to user SYS without SYSDBA privileges. Instead of connecting to user SYS and sharing the password, grant DBA privilege to a normal user, who will connect to the database as a user with SYSDBA privileges to connect to SYS schema.

In Oracle Database, a user can be granted the SELECT ANY DICTIONARY privilege. A user with this privilege can access objects in the SYS schema regardless of the setting of O7_DICTIONARY_ACCESSIBILITY.

Schema Objects

This section describes compatibility and interoperability issues relating to schema objects.

Bitmap Index Protection

In releases prior to release 8.1, it was possible to unintentionally invalidate bitmap indexes by issuing certain SQL statements. The most common causes of bitmap index invalidation were the following types of statements:

- ALTER TABLE statements that define a primary key on an existing table that did not previously have a primary key.
- ALTER TABLE statements that define a NOT NULL constraint on a column that did not previously have this constraint.

Oracle Database eliminates these unintentional invalidations.

Datatypes

This section describes compatibility and interoperability issues relating to datatypes.

Large Objects (LOBs)

This section describes compatibility and interoperability issues relating to LOBs.

Varying-Width Character Sets for CLOBs and NCLOBs Release 8.0 did not allow users other than SYSTEM to create tables with the CLOB or NCLOB datatype if the database character set was varying-width. Release 8.1 and higher supports CLOB and NCLOB datatypes in tables with a varying-width character set, and the data is stored as UCS2 (2-byte fixed-width unicode).

LOB Index Clause If you used the LOB index clause to store LOB index data in a tablespace separate from the tablespace used to store the LOB, then the index data will be relocated to reside in the same tablespace as the LOB if you perform either of the following actions in release 8.1 and higher:

- Perform an Export/Import on the LOB

- Exchange the LOB into a partitioned table

If you create a new table in release 8.1 and higher and specify a tablespace for the LOB index for a non-partitioned table, then the tablespace specification will be ignored and the LOB index will be located in the same tablespace as the LOB.

To check the storage of LOB indexes, issue the following SQL statement connected as a user with SYSDBA privileges:

```
SQL> SELECT index_name, index_type, tablespace_name
         FROM dba_indexes
         WHERE index_type = 'LOB';
```

Oracle ROWIDs

This section describes compatibility and interoperability issues related to rowids.

UROWID Datatype Release 8.1 introduced the UROWID (universal rowid) datatype. Clients prior to release 8.1 can access columns of UROWID datatype using character host variables only; other types of variables are not supported.

NCHAR and NLS Environment Variables and Compatibility

You should set NLS_LANG to your environment as follows:

- Set the ORA_NLS33 environment variable for the release 8.0 and higher environment

Verify that the client has the correct NLS character set environment variables.

User-Defined Datatypes

This section describes compatibility and interoperability issues relating to user-defined datatypes.

New Format for User-Defined Datatypes

Release 8.1 introduced a new format for user-defined datatypes. The new format can result in significant performance improvements over the format used in release 8.0. You can use release 8.0 user-defined datatypes in a release 8.1 or higher database without causing compatibility problems. However, the database will not realize the performance gains possible with the new format.

Release 8.1 and Higher Clients Accessing Release 8.0 User-Defined Datatypes The user-defined datatypes format is negotiated as part of the compatibility exchange between the client and server. If you are using a release 8.0 server, then release 8.1 and higher clients can access the database, but they are set to release 8.0.

Release 8.0 Clients Accessing Release 8.1 or Higher User-Defined Datatypes When a release 8.0 client accesses a server with release 8.1 or higher user-defined datatypes, the database converts the user-defined datatypes to release 8.0 format. Consequently, the release 8.0 client can access the data, but performance gains may not be realized.

Nested Tables

Release 8.0 clients do not support the following release 8.1 and higher nested table features:

- Collection locators

- User-specified storage for collection columns, including storage of nested table data in an index-organized table

Therefore, access fails with an incompatibility error when a release 8.0 client attempts to access a release 8.1 or higher server and a nested table is specified to be returned as a locator, or the storage for the nested table is user-specified.

Varrays Stored as LOBs

Release 8.0 clients do not support specifications of storage parameters for storing varrays as LOBs. Therefore, access fails with an incompatibility error when a release 8.0 client attempts to access a release 8.1 or higher server where there is a specification of storage parameters for storing a varray as a LOB.

SQL and PL/SQL

This section describes compatibility and interoperability issues relating to SQL and PL/SQL.

> **See Also:** *Oracle Database SQL Reference* and *PL/SQL User's Guide and Reference* for more information about SQL and PL/SQL

Functions GREATEST_LB, LEAST_UB, and TO_LABEL Desupported

Starting with release 8.1, the built-in PL/SQL functions `GREATEST_LB`, `LEAST_UB`, and `TO_LABEL` are no longer supported.

SQL Scripts utlchain.sql and utlchn1.sql

The Oracle Database installation includes the following two scripts for creating a table that stores migrated and chained rows: `utlchain.sql` and `utlchn1.sql`. The `utlchn1.sql` script can be run on index-organized tables as well as regular tables, while the `utlchain.sql` script can be run only on regular tables, but not on index-organized tables.

In Oracle8*i* release 8.1 and later, you must always run the `utlchn1.sql` script.

SQL Scripts utlexcpt.sql and utlexpt1.sql

The Oracle Database installation includes the following two scripts for creating a table that stores exceptions from enabling constraints: `utlexcpt.sql` and `utlexpt1.sql`. The `utlexpt1.sql` script can be run on index-organized tables as well as regular tables, while the `utlexcpt.sql` script can be run only on regular tables, but not on index-organized tables.

In Oracle8*i* release 8.1 and later, you must always run the `utlexpt1.sql` script.

Advanced Queuing (AQ)

This section includes compatibility and interoperability issues for AQ.

> **See Also:** *Oracle Streams Advanced Queuing User's Guide and Reference* for more information about AQ. The sections below only provide compatibility and interoperability information about new AQ features, while *Oracle Streams Advanced Queuing User's Guide and Reference* provides detailed information about using them.

Interoperability and the Sender's ID Column

In release 8.1 and higher, the sender's ID is mapped as an additional attribute in the message properties. This new attribute is ignored when there is communication between release 8.0 and release 9.0.1 and higher databases.

For OCI applications, the sender's ID attribute is available as a new attribute in the message properties descriptor. Release 8.1 and higher OCI clients use a new RPC code to send and receive the message properties to and from the server.

Rule Based Subscriptions

When you migrate a queue table from release 8.0 to release 8.1 or higher using the `DBMS_AQADM.MIGRATE_QUEUE_TABLE` procedure, any existing subscribers are upgraded automatically to subscribers with null rules.

Oracle Optimizer

Oracle Database contains a significant number of optimizer enhancements that are either new or have not been enabled by default in previous releases.

Upgrading an existing application to Oracle Database could therefore result in a large number of changes in execution plans. For a mature application, changes in behavior may introduce an element of risk. Customers who wish to minimize execution plan changes can do so by means of the `OPTIMIZER_FEATURES_ENABLE` initialization parameter.

Setting the value of this parameter to an earlier release, for example, release 8.1.7, makes Oracle use only those optimizer features that were enabled by default in that release, something that will reduce the likelihood of changes in execution plans when upgrading from that release.

The Oracle Plan Stability feature can also be used to preserve old behavior when upgrading to a new release.

Real Application Clusters

Support for different releases of Oracle within a Real Application Clusters environment is operating system-specific. See your operating system-specific Oracle documentation for information about whether or not the coexistence of different releases within a Real Application Clusters environment is supported on your operating system.

INSTANCES Keyword in PARALLEL Clause

The `INSTANCES` keyword can be used in release 8.1 and higher, but it will be interpreted differently than in past releases. In release 8.0, the `INSTANCES` keyword could be used in the `PARALLEL` clause of statements such as the following:

- `ALTER CLUSTER`
- `ALTER DATABASE ... RECOVER`
- `ALTER INDEX ... REBUILD`
- `ALTER TABLE`
- `CREATE CLUSTER`
- `CREATE INDEX`
- `CREATE TABLE`

It also could be used in hints. The INSTANCES keyword was used to specify the number of Oracle Parallel Server instances to use in a parallel operation.

Also starting with release 8.1, the syntax for specifying degree in a PARALLEL clause has changed. You can specify degree simply by placing a number after PARALLEL, as in the following example:

```
ALTER TABLE emp PARALLEL 5;
```

However, the DEGREE keyword remains valid if you choose to use it. The preceding syntax is equivalent to the following statement:

```
ALTER TABLE emp PARALLEL (DEGREE 5 INSTANCES 1);
```

Regardless of the syntax, the value you specify is the number of query threads used in a parallel operation. Neither syntax will affect how many instances are used to execute a query. The system will determine how many instances to use based on the instances available and the load on each of the instances. So, either syntax will produce the same result.

Continuing to Use the INSTANCES Keyword in Release 8.1 and Higher You can still use the old syntax to specify both INSTANCES and DEGREE in release 8.1 and higher, but Oracle interprets it as single keyword that specifies the degree. Therefore, the obsolete command syntax is still accepted in release 8.1 and higher, but its interpretation may be different than in past releases. Table 5–2 illustrates the way in which Oracle interprets the possible settings of INSTANCES and DEGREE if you continue to use the obsolete syntax. The columns in Table 5–2 represent the following:

- The **Degree** column indicates the setting for the DEGREE keyword in the PARALLEL clause.
- The **Instances** column indicates the setting for the INSTANCES keyword in the PARALLEL clause.
- The **8.1 Degree** column indicates Oracle's interpretation of the degree in release 8.1 and higher based on the DEGREE and INSTANCES settings.

Table 5–2 Conversion of INSTANCES Keyword in Release 8.1

Degree	Instances	8.1 Degree
Unset or 1	Unset or 1	1
x	DEFAULT	x
x	Unset or 1	x
Unset or 1	DEFAULT	DEFAULT
DEFAULT	y	y
Unset or 1	y	y
DEFAULT	Unset or 1	DEFAULT
x	y	x * y

In the table, x and y are variables representing an integer value.

If you leave a keyword unset, then Oracle uses 1 as its value.

The following scenarios illustrate the way Oracle may behave differently in release 8.1 and higher because of these interpretations:

- Setting DEGREE to x and INSTANCES to 1 does not guarantee that parallel operations use only one instance.
- Setting DEGREE to 1 and INSTANCES to y does not guarantee that parallel operations use only one query thread per instance.
- Setting DEGREE to x and INSTANCES to y does not guarantee either setting. Instead, Oracle attempts to set the degree to x multiplied by y.

Oracle recommends that you discontinue use of the INSTANCES keyword to avoid unexpected behavior. Also, consider using the PARALLEL_INSTANCE_GROUP initialization parameter.

> **See Also:** *Oracle Database SQL Reference* for more information about the PARALLEL clause and *Oracle Database Reference* for information about the PARALLEL_INSTANCE_GROUP initialization parameter.

Database Security

This section describes compatibility and interoperability issues relating to database security.

Enterprise User Management

This section includes compatibility and interoperability issues related to enterprise user management. This functionality is part of the Oracle Advanced Security feature.

> **Note:** The Oracle Security Server (OSS) component no longer exists in Oracle8*i*; most of its functionality has been integrated into Oracle Advanced Security. Oracle does not provide a tool to migrate from OSS to Oracle Advanced Security.

Interoperability with Release 8.1.5 and Release 8.0 Release 8.1.5 and 8.0 servers cannot share global users and roles with release 8.1.6 and higher servers. In addition, current user database links between release 8.1.5 and release 8.1.6 and higher are not supported. Current user database links between release 8.0 and release 8.1.6 and higher are not supported.

Database Backup and Recovery

This section describes compatibility and interoperability issues related to database backup and recovery.

Recovery Manager

> **See Also:**
>
> - *Oracle Database Recovery Manager Reference* for compatibility and interoperability issues relating to Recovery Manager
> - *Oracle Database Backup and Recovery Advanced User's Guide* for information about upgrading the recovery catalog

Recovery Manager Commands

Release 8.1 of Recovery Manager introduced changes to some Recovery Manager commands. However, all commands used in prior releases will continue to work with release 8.1 and higher of Recovery Manager.

For example, the CLONE command is changed to the DUPLICATE command, but the CLONE command will continue to work. Also, the CLONE option of the ALLOCATE and CONNECT commands is now the AUXILIARY option, but the CLONE option will continue to work. Similarly, the CLONENAME keyword in the COPY and SET commands is now AUXNAME, but the CLONENAME keyword will continue to work.

Datafile Backups

A datafile backup taken with release 8.0 and higher can be restored and recovered with any later Oracle release, if a direct upgrade path between the release that backed-up the file and the release that recovers the file is supported. See Table 2–1, " Upgrade Paths" on page 2-2.

Standby Database

The following compatibility restrictions apply to standby databases:

- The primary and standby databases should run on the same operating system. In addition, Oracle recommends that the primary and standby databases run on the same release of the operating system.
- The primary and standby databases must run the same maintenance release of Oracle. For example, if your primary database is running release 8.1.6, then the standby database can run any production 8.1 release, such as release 8.1.5, 8.1.6, or 8.1.7. However, in this case, the standby database cannot run release 8.0.

> **See Also:** Your operating system-specific Oracle documentation for more information about operating system requirements for standby database.

Fast-Start On-Demand Rollback and Fast-Start Parallel Rollback

As part of the recovery process, after a session or instance is abnormally terminated, Oracle rolls back uncommitted transactions. Oracle8*i* introduced two features to improve rollback performance: fast-start on-demand rollback and fast-start parallel rollback.

When a dead transaction holds a row lock on a row that another transaction needs, fast-start on-demand rollback automatically recovers the data block required by the new transaction. Other data blocks and transactions that do not block any new transaction's progress are rolled back in the background.

Fast-start parallel rollback improves background rollback performance by recovering each dead transaction using multiple server processes. Fast-start parallel rollback recovers each dead transaction using multiple server processes only if the following conditions are met:

- There are enough server processes to allocate one or more processes to each dead transaction.

> **See Also:** *Oracle Database Concepts* for more information about fast-start on-demand rollback.

Archiving of Redo Logs

Release 8.1 and higher enables you to archive online redo log files to multiple destinations, including to a local disk-based file or to a specified standby database. The compatibility and interoperability issues described in this section may arise because of this functionality.

Re-Archiving Previously Archived Online Redo Logs Prior to release 8.1, it was possible to re-archive an online redo log that already had been successfully and fully archived. In addition, it was possible to re-archive redo log files to successfully archived destinations.

Starting with release 8.1, the following restrictions apply:

- Successfully archived online redo logs cannot be re-archived.
- Successfully archived destinations cannot be re-archived.

Archive Operation Error Detection Behavior Prior to release 8.1, when any error was detected, an archive operation stopped immediately, reported the error to the alert log, and signaled the error to the user.

Starting with release 8.1, an archive operation does not stop processing unless all of the archive destinations cannot be processed. An error at one or more destinations does not stop the archive operation; the archive operation only stops if all archive destinations cannot be processed. Specifically, archiving to a mandatory is retried once, and archiving failure on the retry halts processing.

LogMiner

LogMiner runs in a release 8.1 or higher instance and can analyze redo log files from any database that meets the following criteria:

- Has the same DBCS (Database Character Set) as the analyzing Oracle instance
- Is running on the same hardware platform as the analyzing Oracle instance

LogMiner does not require a mounted database to analyze redo log files. However, to fully translate the contents of the redo log files, LogMiner requires access to a LogMiner dictionary (catalog). LogMiner uses the dictionary to translate internal object identifiers and data types to object names and external data formats. You can use the PL/SQL package DBMS_LOGMNR_D to extract a database dictionary into an external file for later use in analyzing redo log files. Without a dictionary, LogMiner returns the internal object identifiers and presents data as hex bytes.

Analyzing Archived Redo Log Files from Other Databases You can run LogMiner on an instance of a database while analyzing redo log files from a different database. To analyze archived redo log files from other databases, LogMiner must:

- Access a dictionary file that is both created from the same database as the redo log files and created with the same database character set
- Run on the same hardware platform that generated the log files, although it does not need to be on the same system

Oracle Media Management API and Proxy Copy

Starting with Oracle Media Management API version 2, proxy copy functionality is supported. If a Recovery Manager proxy backup is attempted, and Oracle is linked with Oracle Media Management API release 1.1, or a version 2 that does not support proxy copy functionality, then Recovery Manager will return an error and the backup will fail.

Distributed Databases

This section describes compatibility and interoperability issues related to distributed databases.

Materialized Views

Prior to release 8.1, an Oracle materialized view always consisted of a materialized view base table and a view on the base table. For example, creating a materialized view `SNAP_EMP` creates a view `SNAP_EMP` and a base table normally called `SNAP$_SNAP_EMP`. In release 8.1 and higher, most materialized views will have only a base table with the same name as the materialized view. The view will not be created.

A view will be added to the materialized view under the following conditions:

- The materialized view was imported from a database prior to release 8.1, such as release 8.0.
- The materialized view requires hidden columns (that is, rowid materialized views and fast-refreshable materialized views that contain subqueries).

Oracle Replication

The following compatibility restrictions apply to a replicated environment:

- If you have a replicated environment with different releases of the Oracle Database, then you cannot replicate data that is incompatible on the lower release. For example, in a replicated environment with a database at 9.2.0 compatibility level and another database at 8.1.0 compatibility level, you cannot replicate data between them if the data is incompatible with release 8.1.
- To improve performance and protect data integrity, a number of Advanced Replication packages that were external prior to release 8.1 have been internalized in release 8.1 and higher. *Oracle Database Advanced Replication* contains a list of these internalized packages.

If one or more of your master sites is a release prior to release 8.1, then the `GENERATE_80_COMPATIBLE` flag must be unset or set to TRUE in the following procedures:

- `GENERATE_REPLICATION_SUPPORT`
- `CREATE_SNAPSHOT_REPOBJECT`
- `GENERATE_SNAPSHOT_SUPPORT`

Heterogeneous Services Agents

This section describes compatibility and interoperability issues related to Heterogeneous Services agents.

Interoperability Between Servers of Different Releases Servers at release 8.0.3 and higher can connect to and use Heterogeneous Services agents of any other server at release 8.0.3 and higher. In a connection between servers of different releases, the functionality is limited to that of the lower release.

Multithreaded Service Agents Starting with release 8.1, multithreaded Heterogeneous Services agents are supported. If you have existing agents and you want to take advantage of the multithreaded features, then create the agent initialization file and explicitly start the agents using the Agent Control Utility.

> **See Also:** *Oracle Database Heterogeneous Connectivity Administrator's Guide* for general information about Heterogeneous Services, and for information about creating the agent initialization file and starting the agents using the Agent Control utility.

Net8

This section describes compatibility and interoperability issues relating to Net8.

Service Naming and Connection Load Balancing

Release 8.1 and higher supports service naming and connection load balancing for services that include more than one database instance. Each service can include multiple instances, and each instance can include multiple handlers. This support enables clients to access a service rather than a specific database instance, and logically separates the service name from any particular instance name.

To support services that include multiple instances, use the following new parameters in connect descriptors:

- SERVICE_NAME
- INSTANCE_NAME

The new parameters enable connection load balancing by taking requests through the following process:

1. A client program specifies the name of the service to which it wants to connect.
2. The TNS Listener finds the least loaded instance in the service.
3. The TNS Listener finds the least loaded handler in the instance.
4. The TNS Listener redirects the client to the optimal handler, or passes the client connection to the handler, if necessary.

To use connection load balancing, perform the following actions:

- Discontinue the use of the SID parameter in connect descriptors.
- Use the `SERVICE_NAMES` and `INSTANCE_NAME` initialization parameters in your parameter file.
- Use the `SERVICE_NAME` parameter in the `tnsnames.ora` file.

Note: Before configuring the TNS Listener to handle two or more instances with the same instance name, make sure no client programs use connections based on the SID parameter.

See Also: *Oracle Net Services Administrator's Guide* for more information about using connection load balancing and the `SERVICE_NAME` parameter.

6

Upgrading Your Applications

This chapter describes upgrading your current applications and covers the following topics:

- Overview of Upgrading Applications
- Upgrading Precompiler and OCI Applications
- Upgrading SQL*Plus Scripts
- Upgrading Oracle Forms or Oracle Developer Applications

Overview of Upgrading Applications

You do not need to modify existing applications that do not use features available in the new Oracle Database 10*g* release. Existing applications running against a new Oracle Database function the same as they did in prior releases and achieve the same, or enhanced, performance.

Many new features and enhancements are available after upgrading to the new Oracle Database 10*g* release. Some of these features provide added functionality, while others provide improved performance. Before you upgrade your applications, you should review these new features to decide which ones you want to use.

> **See Also:** *Oracle Database New Features* for information about the features available in the new Oracle Database 10*g* release

Compatibility Issues for Applications

There may be compatibility issues between different releases of the Oracle Database that could affect your applications. These compatibility issues result from differences in the Oracle Database in various releases. Also, in each new release of the Oracle Database, new Oracle reserved words may be added, changes may be made to initialization parameters, and changes may be made to the data dictionary.

When you upgrade your Oracle Database to a new release, make sure that your applications do not use any Oracle reserved words, that your applications are compatible with the initialization parameters of the database, and that your applications are compatible with the data dictionary of the database. Finally, a new release of the Oracle Database software may require certain operating system releases or the application of certain patch sets.

See Also:

- "Applications" on page 5-15 for information about compatibility issues that relate to applications
- Appendix A, "Initialization Parameter and Data Dictionary Changes" for information about initialization parameter changes and data dictionary changes
- *Oracle Database SQL Reference* for a complete list of Oracle reserved words
- Your operating system-specific Oracle documentation for information about operating system requirements

Net8 and Oracle Net Services work with various Oracle Database releases. Thus, Oracle8, Oracle8*i*, Oracle9*i*, and Oracle Database 10*g* can communicate by using Net8 and Oracle Net Services.

Upgrading Precompiler and OCI Applications

The upgrade path is very similar for precompiler and OCI applications. This section guides you through your upgrade options for these applications and notes differences between precompiler and OCI applications whenever necessary.

Create a test environment before you upgrade your production environment. Your test environment should include your upgraded application and the new Oracle Database. Also, your test environment should provide a realistic test of your application.

See Also: *Pro*C/C++ Programmer's Guide*, *Pro*COBOL Programmer's Guide*, and *Oracle Call Interface Programmer's Guide* for more information about using these programming environments.

Understanding Software Upgrades and Your Client/Server Configuration

To understand your options for upgrading precompiler and OCI applications, you first need to understand the type of software upgrade you are performing and your client/server configuration.

Types of Software Upgrades

Two types of upgrades are possible for Oracle Database client and server software.

Major Database Release Upgrade The upgrade changes the first digit of the release number.For example, upgrading from Oracle9*i* to Oracle Database 10*g* is a major database release upgrade.

Database Maintenance Release Upgrade The upgrade changes the second digit of the release number. For example, upgrading from release 9.0.1 to release 9.2 is a database maintenance release upgrade.

Note: Starting with release 9.2, maintenance releases of the Oracle Database are denoted by a change to the second digit of a release number. In previous releases, the third digit indicated a particular maintenance release.

Possible Client/Server Configurations

Your precompiler and OCI applications run on the client in a client/server environment, where the Oracle Database server is the server. You may use one or more of the following client/server configurations in your environment.

Different Computers The client software and the server software are on different computers, and they are connected through a network. The client and server environments are separate.

Different Oracle Home Directories on the Same Computer The client software and the server software are on the same computer, but they are installed in different Oracle home directories. Again, the client and server environments are separate.

Same Oracle Home The client software and server software are installed in the same Oracle home on the same computer. In this case, any upgrade of the server software is also an upgrade of the client software.

> **See Also:** *Oracle Database Concepts* and *Oracle Database Heterogeneous Connectivity Administrator's Guide* for more information about client/server environments

Compatibility Rules for Applications When Upgrading Client/Server Software

This section covers compatibility rules that apply when you upgrade Oracle Database client or server software. The rules are based on the type of software upgrade you are performing and on your client/server configuration.

The following sections contain compatibility rules for the following type of upgrades:

- Upgrading the Oracle Database Server Software
- Upgrading the Oracle Database Client Software

> **Note:** This section uses the terms introduced in "Understanding Software Upgrades and Your Client/Server Configuration" on page 6-2.

Upgrading the Oracle Database Server Software

The following rules apply when you upgrade the Oracle Database server software.

If You Do Not Change the Client Environment, Then You Do Not Need to Relink. If your client and server are on different computers or are in different Oracle home directories on the same computer, and you upgrade the Oracle Database server software without changing the client software, then you do not need to precompile, compile, or relink your applications. In these cases, the client software is separate from the server software and will continue to function against the server.

However, if your applications are using the same Oracle home as the Oracle Database server, then your server upgrade also upgrades your client software, and you must follow the rules in "Upgrading the Oracle Database Client Software" on page 6-4.

Note: It is possible to upgrade the Oracle Database server software but not install the new precompiler or OCI client software when you are using the same Oracle home for both. In this case, the client software is not upgraded. However, such a configuration is not recommended.

Applications Can Run Against Newer or Older Oracle Database Server Releases When you run a precompiler or OCI application against a database server, Oracle recommends that the release of the database server software be equal to or higher than the client software release, but this configuration is not strictly required. For example, if your Oracle Database client software is release 8.1.7, then your Oracle Database server software *should* be release 8.1.7 or higher to run a precompiler application on the client against the server.

Upgrading the Oracle Database Client Software

Oracle recommends that you upgrade your client software to match the current server software. For example, if you upgrade your Oracle Database server to release 9.2, then Oracle recommends upgrading the client software to release 9.2 as well. Keeping the server and client software at the same release number ensures the maximum stability for your applications. In addition, the latest Oracle Database client software may provide added functionality and performance enhancements that were not available with previous releases.

The following rules apply when you upgrade the Oracle Database client software.

Applications Can Be Linked with Newer Libraries The code generated by precompiler applications can be linked with a release of the client library that is equal to or higher than the server release. In addition, release 8.0 and higher SQLLIB function calls cannot be mixed in the same application and the same transaction.

OCI applications can be linked with a version of the OCI runtime library that is equal to or higher than the version of the OCI library with which the application was developed.

Statically-Linked Applications Do Not Need to be Relinked For statically-linked applications, when you perform any type of upgrade of the client software, you do not need to relink your precompiler and OCI applications. However, relinking is recommended because it may improve performance.

Dynamically-Linked Applications Do Not Need To Be Relinked When you perform an upgrade of your client software, you do not need to relink your dynamically-linked precompiler and OCI applications. However, relinking is recommended because it may improve performance.

Upgrading Options for Your Precompiler and OCI Applications

You have the following four options for upgrading your precompiler and OCI applications:

- **Option 1:** Leave the application unchanged. Do not relink, precompile, or compile the application, and do not change the application code. The application will continue to work against Oracle Database 10*g*.
- **Option 2:** Relink the application with the new Oracle Database 10*g* libraries. Do not precompile or compile the application and do not change the application code.

- **Option 3:** Precompile and/or compile and then relink the application using the new Oracle Database 10*g* software. Do not change the application code.
- **Option 4:**Change the application code to use new Oracle Database 10*g* features. Then, precompile and/or compile and then relink the code.

These options are listed in order of increasing difficulty and increasing potential benefits. That is, Option 1 is the least difficult option, but it offers the least potential benefits, while Option 4 is the most difficult option, but it offers the most potential benefits. These options are discussed in the following sections.

Option 1: Leave the Application Unchanged

You can leave the application unchanged, and it will continue to work with the new Oracle Database 10*g* release. The major advantage to this option is that it is simple and easy. In addition, this option requires the least amount of administration, because you do not need to upgrade all of your client computers. If you have a large number of client computers, then avoiding the administrative costs of upgrading all of them can become very important.

The major disadvantage to this option is that your application cannot use the features that are available in the new Oracle Database 10*g* release. In addition, your application cannot leverage some of the possible performance benefits of the new Oracle Database 10*g* release.

Option 2: Relink the Application with the New Oracle Database 10*g* Libraries

You can relink the application with the new Oracle Database 10*g* libraries, without making any code changes and without recompiling. By relinking, your application may benefit from performance improvements that are available only with the new libraries. Remember that you should always relink the application in a test environment before you relink in your production environment.

Note: On operating systems that do not support shared libraries, you must relink your application if you want to include the new libraries in the executable.

Option 3: Precompile or Compile the Application Using the New Software

You can precompile or compile the application with the new Oracle Database 10*g* software, without making any code changes. This option requires that you install the new Oracle Database client software on each client computer. However, you only need to precompile or compile, and relink your application once, regardless of the number of clients you have. The advantages, however, can be quite large.

By recompiling, you perform a syntax check of your application code. Some problems in the application code that were not detected by previous releases of the Oracle software may emerge when you precompile or compile with the new Oracle software. Therefore, precompiling and compiling with the new software often helps you detect and correct problems in the application code that may have gone unnoticed before.

Also, recompiling affords maximum stability for your application, because you are sure that it works with the new Oracle software. Further, your environment is ready for new development using the latest tools and features available. In addition, you may benefit from performance improvements that are available with the new Oracle software only after you recompile and relink.

Option 4: Change the Application Code to Use New Oracle Database 10*g* Features

You can make code changes to your application to take advantage of new Oracle Database 10*g* features. This option is the most difficult, but it can provide the most potential benefits. You gain all of the advantages described in Option 3. In addition, you also benefit from changes to your application that may leverage performance and scalability benefits available with Oracle Database 10*g*. Further, you can add new features to your application that are available only with the new Oracle Database 10*g* release.

Become familiar with the new Oracle Database 10*g* features by reading *Oracle Database New Features*. Also, consult the Oracle documentation for your development environment so that you understand how to implement the features you want to use. For the precompilers, see *Pro*C/C++ Programmer's Guide* and *Pro*COBOL Programmer's Guide*. For OCI, see *Oracle Call Interface Programmer's Guide*.

When you have decided on the new features you want to use, change the code of your application to use these features. Follow the appropriate instructions in the following sections based on your development environment:

- Changing Precompiler Applications
- Changing OCI Applications

Changing Precompiler Applications Complete the following steps to change your precompiler application to use Oracle Database 10*g* features:

1. If you want to take advantage of the new Oracle Database 10*g* features, then incorporate code for the new Oracle Database 10*g* functionality into the existing application.
2. Precompile the application using the Oracle precompiler.
3. Compile the application.
4. Relink the application with the Oracle Database 10*g* runtime library, SQLLIB, which is included with the precompiler.

Changing OCI Applications Complete the following steps to change your OCI application to use Oracle Database 10*g* features:

1. Incorporate the new Oracle Database 10*g* OCI calls into the existing application.
2. Compile the application.
3. Relink the application with the Oracle Database 10*g* runtime library.

Upgrading SQL*Plus Scripts

To use SQL*Plus release 8.0 and higher, a release 8.0 or higher database, and PL/SQL release 8.0 and higher functionality, complete the following steps:

1. Make the following changes to SQL*Plus release 3.x scripts to convert them into scripts that are compatible with SQL*Plus release 8.0 and higher:
   a. If a script contains the line `SET COMPATIBILITY V7`, then change it to `SET COMPATIBILITY NATIVE`, or remove the line so that the default setting is used. In Oracle Database 10*g*, the default setting is `NATIVE`.
   b. Check any `login.sql` and `glogin.sql` files and change any `SET COMPATIBILITY V7` line found to `SET COMPATIBILITY NATIVE`.

2. To use new Oracle Database 10*g* functionality, change existing SQL scripts to use the new Oracle Database 10*g* syntax. Existing SQL scripts run unchanged on Oracle Database 10*g*, and require no modification, if they do not use new Oracle Database 10*g* functionality.

   See Also:

   - *SQL*Plus User's Guide and Reference* to learn about new functionality in SQL*Plus
   - *Oracle Database SQL Reference* for more information about upgrading SQL scripts

   Note: No changes to PL/SQL packages, procedures, or functions should be required.

Upgrading Oracle Forms or Oracle Developer Applications

Forms applications run the same on Oracle8, Oracle8*i*, Oracle9*i*, and Oracle Database 10*g*. However, review the new features described in *Oracle Database New Features* to determine whether any of the new Oracle Database 10*g* features would be beneficial to your applications or might otherwise affect them. Information about the ways in which the Oracle Database 10*g* features interact with forms and developer applications is provided in the Oracle Developer documentation set. Also, the Oracle Developer documentation for your operating system contains instructions for upgrading your forms or developer applications.

Note: New releases of Oracle Developer may introduce new reserved words that are specific to Oracle Developer. Code changes may be required if your application uses any of these new reserved words.

7

Downgrading a Database Back to the Previous Oracle Database Release

This chapter guides you through the process of downgrading a database back to the previous Oracle Database release. This chapter covers the following topics:

- Supported Releases for Downgrading
- Check for Incompatibilities
- Perform a Full Offline Backup
- Downgrade the Database

> **See Also:** Some aspects of downgrading are operating system-specific. See your operating system-specific Oracle documentation for additional instructions about downgrading on your operating system.

Supported Releases for Downgrading

In Oracle Database 10*g* release 10.1, downgrading is only supported back to release 9.2. However, if the release number of your Oracle9*i* database is lower than release 9.2.0.3, then you should install the latest patch release for release 9.2; downgrading is not supported to releases prior to release 9.2.0.3.

In addition, if the database being downgraded has Messaging Gateway installed, then you should install Messaging Gateway Bundled Patch 2 or later.

> **Note:** You do not need to first upgrade your previous database to release 9.2.0.3.0 or later, but the release 9.2.0.3.0 or later software must be installed before the downgrade from release 10.1.

Check for Incompatibilities

Check the compatibility level of your database to see if your database might have incompatibilities that prevent you from downgrading. If the compatibility level of your release 10.1 database is `10.0.0` or higher, then you will not be able to downgrade. Your compatibility level is determined by the setting of the `COMPATIBLE` initialization parameter. Check your `COMPATIBLE` initialization parameter setting by issuing the following SQL statement:

```
SQL> SELECT name, value, description FROM v$parameter
        WHERE name='compatible';
```

If you are downgrading to release 9.2 and the COMPATIBLE initialization parameter is set to 9.2.0, then you will be able to downgrade.

See Also: "Downgrading and Compatibility" on page 5-2

Perform a Full Offline Backup

Perform a full offline backup of your release 10.1 database before you downgrade.

See Also: *Oracle Database Backup and Recovery Basics* for more information

Downgrade the Database

Make sure your database is compatible with the release to which you are downgrading before you perform the downgrade steps in this section.

Complete the following steps to downgrade your release 10.1 database to the previous Oracle Database release:

1. Log in to the system as the owner of the release 10.1 Oracle home directory.
2. At a system prompt, change to the *ORACLE_HOME*/rdbms/admin directory.
3. Start SQL*Plus.
4. Connect to the database instance as a user with SYSDBA privileges.
5. Start up the instance in DOWNGRADE mode:

   ```
   SQL> STARTUP DOWNGRADE
   ```

   You may need to use the PFILE option to specify the location of your initialization parameter file.
6. Set the system to spool results to a log file for later verification of success:

   ```
   SQL> SPOOL downgrade.log
   ```

7. Run d*old_release*.sql, where *old_release* refers to the release to which you are downgrading. See Table 7–1 to choose the correct script. Each script provides a direct downgrade to the release specified in the "Downgrading To" column.

   To run a script, enter the following:

   ```
   SQL> @dold_release.sql
   ```

Table 7–1 Downgrade Scripts

Downgrading To	Run Script
9.2	d0902000.sql

Note: This is the only release supported for downgrading.

The following are notes about running the script:

- You must use the version of the script included with release 10.1.

- You must run the script in the release 10.1 environment.
- The script downgrades all Oracle Database components in the database.

If you encounter any problems when you run the script, or any of the scripts in the remaining steps, then correct the causes of the problems and rerun the script. You can rerun any of the scripts described in this chapter as many times as necessary.

If the downgrade for a component fails, then an `ORA-39709` error will be displayed and the downgrade will not complete. All components must be successfully downgraded before the Oracle Database data dictionary is downgraded.

8. Turn off the spooling of script results to the log file:

```
SQL> SPOOL OFF
```

Then, check the spool file and verify that there were no errors generated during the downgrade. You named the spool file in Step 6; the suggested name was `downgrade.log`. Correct any problems you find in this file and rerun the appropriate downgrade script if necessary.

9. Shut down the instance:

```
SQL> SHUTDOWN IMMEDIATE
```

If you are downgrading a cluster database, then shut down all instances.

10. Exit SQL*Plus.

11. If your operating system is UNIX, then change the following environment variables to point to the directories of the release to which you are downgrading:

- `ORACLE_HOME`
- `PATH`
- `ORA_NLS33`
- `LD_LIBRARY_PATH`

Note: If you are downgrading a cluster database, then perform this step on all nodes in which this cluster database has instances configured.

See Also: Your operating system-specific Oracle Database 10*g* installation documents for information about setting other important environment variables on your operating system

12. If your operating system is Windows, then complete the following steps:

a. Stop all Oracle services, including the `OracleService`*`SID`* Oracle service of the release 10.1 database, where *`SID`* is the instance name.

For example, if your *SID* is ORCL, then enter the following at a command prompt:

```
C:\> NET STOP OracleServiceORCL
```

See Also: Your *Administrator's Guide* for Windows for information about stopping services

b. Delete the Oracle service at a command prompt by issuing the ORADIM command. For example, if your *SID* is ORCL, then enter the following command:

```
C:\> ORADIM -DELETE -SID ORCL
```

c. Create the Oracle service of the database to which you are downgrading at a command prompt using the ORADIM command.

```
C:\> ORADIM -NEW -SID SID -INTPWD PASSWORD -MAXUSERS USERS
     -STARTMODE AUTO -PFILE ORACLE_HOME\DATABASE\INITSID.ORA
```

This syntax includes the following variables:

Variable	Description
SID	is the same SID name as the SID of the database being downgraded.
PASSWORD	is the password for the database instance. This is the password for the user connected with `SYSDBA` privileges. The -INTPWD option is not required. If you do not specify it, then operating system authentication is used, and no password is required.
USERS	is the maximum number of users who can be granted `SYSDBA` and `SYSOPER` privileges.
ORACLE_HOME	is the Oracle home directory of the database to which you are downgrading. Ensure that you specify the full pathname with the -PFILE option, including drive letter of the Oracle home directory.

For example, if you are downgrading to release 9.2, if your *SID* is ORCL, your *PASSWORD* is TWxy579, the maximum number of *USERS* is 10, and the *ORACLE_HOME* directory is `C:\ORANT`, then enter the following command:

```
C:\> ORADIM -NEW -SID ORCL -INTPWD TWxy579 -MAXUSERS 10
     -STARTMODE AUTO -PFILE C:\ORANT\DATABASE\INITORCL.ORA
```

13. Copy configuration files from the release 10.1 Oracle home directory to the Oracle home of the release to which you are downgrading:

a. Copy your parameter file from the release 10.1 Oracle home to the Oracle home of the release to which you are downgrading. By default Oracle looks for the parameter file in *ORACLE_HOME*`/dbs` on UNIX platforms and in *ORACLE_HOME*`\database` on Windows operating systems. The parameter file can reside anywhere you wish, but it should not reside in the release 9.2 Oracle home.

b. If your parameter file has an `IFILE` (include file) entry and the file specified in the `IFILE` entry resides within the release 10.1 Oracle home directory, then copy the file specified by the `IFILE` entry to the Oracle home of the release to which you are downgrading. The file specified in the `IFILE` entry contains additional initialization parameters. After you copy this file, edit the parameter file to point to its new location.

c. If you have a password file that resides within the release 10.1 Oracle home directory, then move or copy the password file to the Oracle home of the release to which you are downgrading. The name and location of the password file are operating system-specific. On UNIX platforms, the default password file is *ORACLE_HOME*`/dbs/orapw`*sid*. On Windows operating systems, the default password file is *ORACLE_*

HOME\database\pwd*sid*.ora. On both UNIX platforms and Windows operating systems, *sid* is your Oracle instance ID.

Note: If you are downgrading a cluster database, then perform this step on all nodes in which this cluster database has instances configured.

- Set the CLUSTER_DATABASE initialization parameter to false. After the downgrade, you must set this initialization parameter back to true.

---

14. At a system prompt, change to the *ORACLE_HOME*/rdbms/admin directory of the previous release.
15. Start SQL*Plus.
16. Connect to the database instance as a user with SYSDBA privileges.
17. Start up the instance:

```
SQL> STARTUP MIGRATE
```

18. Set the system to spool results to a log file for later verification of success:

```
SQL> SPOOL reload.log
```

19. Run catrelod.sql:

```
SQL> @catrelod.sql
```

The catrelod.sql script reloads the release 9.2 version of all of the database components in the downgraded database.

20. Turn off the spooling of script results to the log file:

```
SQL> SPOOL OFF
```

Then, check the spool file and verify that the packages and procedures compiled successfully. You named the spool file in Step 18; the suggested name was reload.log. Correct any problems you find in this file and rerun the appropriate script if necessary.

ORA-22308: operation not allowed on evolved type errors in the spool file may safely be ignored.

21. Shut down and restart the instance for normal operation:

```
SQL> SHUTDOWN IMMEDIATE
SQL> STARTUP
```

You may need to use the PFILE option to specify the location of your initialization parameter file.

Note: If you are downgrading a cluster database, then perform this step on all nodes in which this cluster database has instances configured.

---

22. Run utlrp.sql:

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

The `utlrp.sql` script recompiles all existing PL/SQL modules that were previously in an `INVALID` state, such as packages, procedures, types, and so on.

23. Exit SQL*Plus.

Your database is now downgraded.

8

Data Copying Using Export/Import

This chapter guides you through the process of using the Export and Import utilities to copy data from one Oracle Database to another in which the database release numbers are different. This chapter covers the following topics:

- Export and Import Requirements
- Upgrade the Database Using Export/Import

> **See Also:** *Oracle Database Utilities* for detailed information about the Export and Import utilities

> **Note:** This chapter discusses data copying using the original Export and Import utilities. Data Pump Export and Data Pump Import cannot be used in releases prior to Oracle Database 10*g* release 10.1.

Export and Import Requirements

Dump files created by the Export utility can be imported into all future releases of the Oracle Database. For example, an Oracle8 export dump file can be imported by the oracle8*i*, Oracle9*i*, and Oracle Database 10*g* Import utilities.

Export dump files, however, are *not* downward compatible with the Import utilities of previous Oracle Database releases. That is, exported data *cannot* be imported by the Import utilities of previous Oracle Database releases. For example, an Oracle9*i* export dump file cannot be imported by an Oracle8*i* Import utility, and an Oracle Database 10*g* export dump file cannot be imported by an Oracle9*i* Import utility.

Table 8–1 shows which Export and Import releases to use when moving data between different releases of the Oracle Database.

Table 8–1 Using Different Releases of Export and Import

Export from	Import to	Use Export Utility for	Use Import Utility For
Release 10.1	Release 10.1	Release 10.1	Release 10.1
Release 10.1	Release 9.2	Release 9.2	Release 9.2
Release 10.1	Release 9.0.1	Release 9.0.1	Release 9.0.1
Release 10.1	Release 8.1.7	Release 8.1.7	Release 8.1.7
Release 10.1	Release 8.0.6	Release 8.0.6	Release 8.0.6
Release 9.2	Release 10.1	Release 9.2	Release 10.1

Table 8–1 (Cont.) Using Different Releases of Export and Import

Export from	Import to	Use Export Utility for	Use Import Utility For
Release 8.1.7	Release 10.1	Release 8.1.7	Release 10.1
Release 8.0.6	Release 10.1	Release 8.0.6	Release 10.1
Release 7.3.4	Release 10.1	Release 7.3.4	Release 10.1

Export/Import Usage on Data Incompatible with a Previous Release

When you export data to a previous release, data that is incompatible with the previous release either is not exported at all or is exported with the loss of some features. For example, the new floating point datatypes in Oracle Database 10*g* are not exported by the Oracle9*i* Export utility. In general, if you need to export data to a previous release, then first remove as many incompatibilities with the previous release as possible before you export the data.

Upgrade the Database Using Export/Import

To upgrade a database using the Export/Import utilities, complete the following steps:

1. Export data from the current database using the Export utility shipped with the current database. See the current database's utilities documents for information about using the Export utility on the current database.

   To ensure a consistent export, make sure the current database is not available for updates during and after the export. If the current database will be available to users for updates after the export, then, prior to making the current database available, put procedures in place to copy the changes made in the current database to the new Oracle Database after the import is complete.

2. Install the new Oracle Database software. Installation is operating system-specific. Installation steps for Oracle Database are covered in your operating system-specific Oracle documentation.

3. If the new Oracle Database will have the same name as the current database, then shut down the current database before creating the new Oracle Database.

4. Create the new Oracle Database.

   See Also: *Oracle Database Administrator's Guide* for information about creating an Oracle Database

5. Start SQL*Plus in the new Oracle Database environment.

6. Connect to the database instance as a user with `SYSDBA` privileges.

7. Start an Oracle Database instance using `STARTUP`.

8. Pre-create tablespaces, users, and tables in the new database to improve space usage by changing storage parameters. When you pre-create tables using SQL*Plus, either run the database in the original database compatibility mode or make allowances for the specific data definition conversions that occur during import.

> **Note:** If the new Oracle Database will be created on the same computer as the source database, and you do not want to overwrite the source database datafiles, then you must pre-create the tablespaces and specify `IGNORE=Y` and `DESTROY=N` when you import.

9. Use the Import utility of the new Oracle Database to import the objects exported from the current database. Include the `LOG` parameter to save the informational and error messages from the import session to a file.

   > **See Also:** *Oracle Database Utilities* for a complete description of the Import utility.

10. After the import, check the import log file for information about which imports of which objects completed successfully and, if there were failures, which failed.

    > **See Also:** *Oracle Database Utilities* and the Oracle Database `README.doc` file for error handling information.

11. Use further Import scenarios (see *Oracle Database Utilities*) or SQL scripts that create the database's objects to clean up incomplete imports (or possibly to start an entirely new import).
12. If changes are made to the current database after the export, then make sure those changes are propagated to the new Oracle Database prior to making it available to users. See Step 1 on page 8-2 for more information.
13. Complete the procedures described in Chapter 4, "After Upgrading a Database".

A

Initialization Parameter and Data Dictionary Changes

This appendix lists changes to initialization parameters and the data dictionary across different releases of the Oracle Database. This appendix also discusses compatibility issues with certain initialization parameters.

This appendix covers the following topics:

- Initialization Parameter Changes
- Compatibility Issues with Initialization Parameters
- Static Data Dictionary View Changes
- Dynamic Performance View Changes

> **Note:** Some of the initialization parameters listed in this appendix are operating system-specific. See your operating system-specific Oracle documentation for more information about these initialization parameters.

Initialization Parameter Changes

The following sections list changes to initialization parameters across different releases of the Oracle Database:

- Deprecated Initialization Parameters
- Obsolete Initialization Parameters

> **See Also:** The "What's New in Oracle Database Reference" section of *Oracle Database Reference* for a list of new initialization parameters in Oracle Database 10*g*

Deprecated Initialization Parameters

The following sections list initialization parameters that have been deprecated. To get a list of all deprecated initialization parameters, issue the following SQL statement:

```
SQL> SELECT name FROM v$parameter
         WHERE isdeprecated = 'TRUE';
```

A deprecated parameter behaves the same way as a regular parameter, except that a warning message is displayed at instance startup if a deprecated parameter is

specified in the parameter file. In addition, all deprecated parameters are logged to the alert log at instance startup:

- Initialization Parameters Deprecated in Release 10.1
- Initialization Parameters Deprecated in Release 9.2
- Initialization Parameters Deprecated in Release 9.0.1

Initialization Parameters Deprecated in Release 10.1

The following initialization parameters were deprecated in release 10.1:

`BUFFER_POOL_KEEP` (replaced by `DB_KEEP_CACHE_SIZE`)
`BUFFER_POOL_RECYCLE` (replaced by `DB_RECYCLE_CACHE_SIZE`)
`GLOBAL_CONTEXT_POOL_SIZE`
`LOCK_NAME_SPACE`
`LOG_ARCHIVE_START`
`MAX_ENABLED_ROLES`
`PARALLEL_AUTOMATIC_TUNING`
`PLSQL_COMPILER_FLAGS` (replaced by `PLSQL_CODE_TYPE` and `PLSQL_DEBUG`)

Initialization Parameters Deprecated in Release 9.2

The following initialization parameters were deprecated in release 9.2:

`DRS_START` (replaced by `DG_BROKER_START`)

Initialization Parameters Deprecated in Release 9.0.1

The following initialization parameters were deprecated in release 9.0.1:

`FAST_START_IO_TARGET` (replaced by `FAST_START_MTTR_TARGET`)
`MTS_CIRCUITS` (replaced by `CIRCUITS`)
`MTS_DISPATCHERS` (replaced by `DISPATCHERS`)
`MTS_MAX_DISPATCHERS` (replaced by `MAX_DISPATCHERS`)
`MTS_MAX_SERVERS` (replaced by `MAX_SHARED_SERVERS`)
`MTS_SERVERS` (replaced by `SHARED_SERVERS`)
`MTS_SESSIONS` (replaced by `SHARED_SERVER_SESSIONS`)
`PARALLEL_SERVER` (replaced by `CLUSTER_DATABASE`)
`PARALLEL_SERVER_INSTANCES` (replaced by `CLUSTER_DATABASE_INSTANCES`)

Obsolete Initialization Parameters

The following sections list initialization parameters that have been made obsolete:

- Initialization Parameters Obsolete in Release 10.1
- Initialization Parameters Obsolete in Release 9.2
- Initialization Parameters Obsolete in Release 9.0.1
- Initialization Parameters Obsolete in Release 8.1

> **Note:** An attempt to start a release 10.1 database using one or more of these obsolete initialization parameters will succeed, but a warning will be returned and recorded in the alert log.

Initialization Parameters Obsolete in Release 10.1

The following initialization parameters were made obsolete in release 10.1:

```
DBLINK_ENCRYPT_LOGIN
HASH_JOIN_ENABLED
LOG_PARALLELISM
MAX_ROLLBACK_SEGMENTS
MTS_CIRCUITS
MTS_DISPATCHERS
MTS_LISTENER_ADDRESS
MTS_MAX_DISPATCHERS
MTS_MAX_SERVERS
MTS_MULTIPLE_LISTENERS
MTS_SERVERS
MTS_SERVICE
MTS_SESSIONS
OPTIMIZER_MAX_PERMUTATIONS
ORACLE_TRACE_COLLECTION_NAME
ORACLE_TRACE_COLLECTION_PATH
ORACLE_TRACE_COLLECTION_SIZE
ORACLE_TRACE_ENABLE
ORACLE_TRACE_FACILITY_NAME
ORACLE_TRACE_FACILITY_PATH
PARTITION_VIEW_ENABLED
PLSQL_NATIVE_C_COMPILER
PLSQL_NATIVE_LINKER
PLSQL_NATIVE_MAKE_FILE_NAME
PLSQL_NATIVE_MAKE_UTILITY
ROW_LOCKING
SERIALIZABLE
TRANSACTION_AUDITING
UNDO_SUPPRESS_ERRORS
```

Initialization Parameters Obsolete in Release 9.2

The following initialization parameters were made obsolete in release 9.2:

```
DISTRIBUTED_TRANSACTIONS
MAX_TRANSACTION_BRANCHES
PARALLEL_BROADCAST_ENABLED
STANDBY_PRESERVES_NAMES
```

Initialization Parameters Obsolete in Release 9.0.1

The following initialization parameters were made obsolete in release 9.0.1:

```
ALWAYS_ANTI_JOIN
ALWAYS_SEMI_JOIN
DB_BLOCK_LRU_LATCHES
DB_BLOCK_MAX_DIRTY_TARGET
DB_FILE_DIRECT_IO_COUNT
GC_DEFER_TIME
GC_RELEASABLE_LOCKS
GC_ROLLBACK_LOCKS
HASH_MULTIBLOCK_IO_COUNT
```

```
INSTANCE_NODESET
JOB_QUEUE_INTERVAL
OPS_INTERCONNECTS
OPTIMIZER_PERCENT_PARALLEL
SORT_MULTIBLOCK_READ_COUNT
TEXT_ENABLE
```

Initialization Parameters Obsolete in Release 8.1

The following initialization parameters were made obsolete in release 8.1:

```
ALLOW_PARTIAL_SN_RESULTS
ARCH_IO_SLAVES
B_TREE_BITMAP_PLANS
BACKUP_DISK_IO_SLAVES
CACHE_SIZE_THRESHOLD
CLEANUP_ROLLBACK_ENTRIES
CLOSE_CACHED_OPEN_CURSORS
COMPATIBLE_NO_RECOVERY
COMPLEX_VIEW_MERGING
DB_BLOCK_CHECKPOINT_BATCH
DB_BLOCK_LRU_EXTENDED_STATISTICS
DB_BLOCK_LRU_STATISTICS
DB_FILE_SIMULTANEOUS_WRITES
DELAYED_LOGGING_BLOCK_CLEANOUTS
DISCRETE_TRANSACTIONS_ENABLED
DISTRIBUTED_RECOVERY_CONNECTION_HOLD_TIMEFAST_FULL_SCAN_ENABLED
ENT_DOMAIN_NAME
FREEZE_DB_FOR_FAST_INSTANCE_RECOVERY
GC_LATCHES
GC_LCK_PROCS
JOB_QUEUE_KEEP_CONNECTIONS
LARGE_POOL_MIN_ALLOC
LGWR_IO_SLAVES
LM_LOCKS
LM_PROCS
LM_RESS
LOCK_SGA_AREAS
LOG_ARCHIVE_BUFFER_SIZE
LOG_ARCHIVE_BUFFERS
LOG_BLOCK_CHECKSUM
LOG_FILES
LOG_SIMULTANEOUS_COPIES
LOG_SMALL_ENTRY_MAX_SIZE
MTS_RATE_LOG_SIZE
MTS_RATE_SCALE
OGMS_HOME
OPS_ADMIN_GROUP
OPTIMIZER_SEARCH_LIMIT
PARALLEL_DEFAULT_MAX_INSTANCES
PARALLEL_MIN_MESSAGE_POOL
PARALLEL_SERVER_IDLE_TIME
PARALLEL_TRANSACTION_RESOURCE_TIMEOUT
PUSH_JOIN_PREDICATE
REDUCE_ALARM
```

```
ROW_CACHE_CURSORS
SEQUENCE_CACHE_ENTRIES
SEQUENCE_CACHE_HASH_BUCKETS
SHARED_POOL_RESERVED_MIN_ALLOC
SNAPSHOT_REFRESH_KEEP_CONNECTIONS
SNAPSHOT_REFRESH_PROCESSES
SORT_DIRECT_WRITES
SORT_READ_FAC
SORT_SPACEMAP_SIZE
SORT_WRITE_BUFFER_SIZE
SORT_WRITE_BUFFERS
SPIN_COUNT
TEMPORARY_TABLE_LOCKS
USE_ISM
```

Compatibility Issues with Initialization Parameters

The lists of deprecated and obsolete initialization parameters earlier in this appendix show changes to initialization parameters across different releases of the Oracle Database. However, certain initialization parameter changes require special attention because they may raise compatibility issues for your database. These parameter changes are described in this section.

Change in Behavior for SESSION_CACHED_CURSORS

In previous Oracle Database releases, the number of SQL cursors cached by PL/SQL was determined by the `OPEN_CURSORS` initialization parameter. Starting with Oracle Database 10*g* release 10.1, the number of cached cursors is determined by the `SESSION_CACHED_CURSORS` initialization parameter.

See Also: `SESSION_CACHED_CURSORS` in *Oracle Database Reference*

New default value for DB_BLOCK_SIZE

In Oracle Database 10*g*, the default value of `DB_BLOCK_SIZE` is operating system specific, but is typically 8 KB (8192 bytes). In previous Oracle Database releases, the default value was 2 KB (2048 bytes). If `DB_BLOCK_SIZE` is not specified in the parameter file when upgrading to the new Oracle Database 10*g* release, then you will receive an error when attempting to start up your Oracle Database. Add the following to your parameter file:

```
DB_BLOCK_SIZE = 2048
```

If `DB_BLOCK_SIZE` is specified in the parameter file, then the Oracle Database will use this value instead of the default value of 8 KB.

OPTIMIZER_MAX_PERMUTATIONS and OPTIMIZER_FEATURES_ENABLE

Starting with Oracle Database 10*g*, the `OPTIMIZER_MAX_PERMUTATIONS` initialization parameter has been made obsolete. If you are upgrading from Oracle9*i* and have `OPTIMIZER_FEATURES_ENABLE` set to `8.1.7` or lower and `OPTIMIZER_MAX_PERMUTATIONS` explicitly set to `2000` in the parameter file, then the release 8.1.7 default of `80000` will be used when you start up the release 10.1 database.

Setting `OPTIMIZER_FEATURES_ENABLE` to `9.0.0` or higher will set the default to `2000`.

Change in Behavior for LOG_ARCHIVE_FORMAT

Starting with Oracle Database 10*g* release 10.1, if the COMPATIBLE initialization parameter is set to 10.0.0 or higher, then archive log file names must contain each of the elements %s (sequence), %t (thread), and %r (resetlogs ID) to ensure that all archive log file names are unique. If the LOG_ARCHIVE_FORMAT initialization parameter is set in the parameter file, then make sure the parameter value contains the %s, %t, and %r elements.

New Default Value for PGA_AGGREGATE_TARGET

Starting with Oracle Database 10*g* release 10.1, Automatic PGA Memory Management is now enabled by default (unless PGA_AGGREGATE_TARGET is explicitly set to 0 or WORKAREA_SIZE_POLICY is explicitly set to MANUAL). PGA_AGGREGATE_TARGET defaults to 20% of the size of the SGA, unless explicitly set. Oracle recommends tuning the value of PGA_AGGREGATE_TARGET after upgrading.

See Also: *Oracle Database Performance Tuning Guide*

Change in Behavior for SHARED_POOL_SIZE

In previous releases, the amount of shared pool memory that was allocated was equal to the value of the SHARED_POOL_SIZE initialization parameter plus the amount of internal SGA overhead computed during instance startup. Starting with Oracle Database 10*g* release 10.1, the value of SHARED_POOL_SIZE must now also accommodate this shared pool overhead.

Shared Server Parameters

Starting with Oracle Database 10*g* release 10.1, the recommended way to turn on shared server mode is to set SHARED_SERVERS to a value greater than 0. This can be done at startup or dynamically after the instance is started. If shared server mode is turned off by setting SHARED_SERVERS to 0, then this only affects new clients (that is, no new clients can connect in shared mode; clients that are already connected in shared mode continue to be serviced by shared servers).

In previous releases, the recommended way to turn on shared server mode was to set DISPATCHERS. If SHARED_SERVERS was changed to 0 and shared server clients were still connected, client requests would hang.

Prior to release 10.1, the following shared server parameters could not be changed dynamically:

- MAX_SHARED_SERVERS
- MAX_DISPATCHERS
- SHARED_SERVER_SESSIONS
- CIRCUITS

Starting with release 10.1, these shared server parameters are dynamically modifiable.

New Default Value for DISPATCHERS

Starting with release 10.1, the default for DISPATCHERS is '(PROTOCOL=TCP)'. DISPATCHERS is given this default value if it is not set or if it is set to '' and SHARED_SERVERS is set to 1 or higher.

In previous releases, there was no default value for DISPATCHERS.

New Default Value for SHARED_SERVERS

Starting with release 10.1, if DISPATCHERS is set such that the total number of dispatchers is equal to 0, then SHARED_SERVERS defaults to 0. If DISPATCHERS is set such that the total number of dispatchers is greater than 0, then SHARED_SERVERS defaults to 1 as in previous releases.

In previous releases, if DISPATCHERS was set such that the number of dispatchers is equal to 0, then SHARED_SERVERS defaulted to 1.

New Default Value for MAX_SHARED_SERVERS

Starting with release 10.1, there is no preset default for MAX_SHARED_SERVERS. The maximum number of shared servers varies depending on the number of free process slots. If MAX_SHARED_SERVERS is not set or is set to a value greater than or equal to PROCESSES, then PMON will not spawn any more shared servers if the number of free process slots is either 2 (if PROCESSES is less than 24) or is less than 1 / 8, unless the existing servers are involved in a deadlock situation. If the existing servers are involved in a deadlock situation, then no matter the transaction load, a new server will be spawned if there is a free process slot.

In previous releases, the default for MAX_SHARED_SERVERS is 20, or 2 * SHARED_SERVERS, whichever is greater, subject to the condition that MAX_SHARED_SERVERS does not exceed PROCESSES.

Starting with release 10.1, SHARED_SERVERS can be set higher than MAX_SHARED_SERVERS, in which case the number of servers will remain constant at the level set for SHARED_SERVERS. This is to allow the range SHARED_SERVERS - MAX_SHARED_SERVERS to be changed without having to change these parameters in a specific order.

In previous releases, SHARED_SERVERS cannot be set higher than MAX_SHARED_SERVERS.

New Default Value for SHARED_SERVER_SESSIONS

Starting with release 10.1, there is no preset default for SHARED_SERVER_SESSIONS. That is, if SHARED_SERVER_SESSIONS is not specified, then shared server sessions can be created as needed and as permitted by the session limit.

In previous releases, the default for SHARED_SERVER_SESSIONS was the maximum number of virtual circuits (CIRCUITS), or the maximum number of database sessions (SESSIONS) - 5, whichever is smaller.

New Default Value for CIRCUITS

Starting with release 10.1, there is no preset default for CIRCUITS. That is, if CIRCUITS is not specified, then circuits can be created as needed and as permitted by dispatcher constraints and system resources.

In previous releases, the default for CIRCUITS was the maximum number of database sessions (SESSIONS) if shared server mode was enabled, 0 otherwise.

New Default Value for MAX_DISPATCHERS

Starting with release 10.1, there is no preset default for MAX_DISPATCHERS. MAX_DISPATCHERS no longer limits the number of dispatchers; the user can increase the number of dispatchers via the DISPATCHERS parameter as long as there are free process slots and system resources.

In previous releases, the default for MAX_DISPATCHERS was 5, or the total number of dispatchers specified via the DISPATCHERS parameter, whichever was greater.

New Default Value for DB_BLOCK_CHECKSUM

Starting with Oracle9*i* release 9.0.1, the DB_BLOCK_CHECKSUM initialization parameter has a new default value. In previous releases, the default value was false, but in Oracle9*i* release 9.0.1 and higher, the default value is true.

See Also: DB_BLOCK_CHECKSUM in *Oracle Database Reference*

Maximum Number of Job Queue Processes

In Oracle9*i*, the maximum number of job queue processes that can be spawned per instance is 1000. In previous releases, the maximum number was 36. The JOB_QUEUE_PROCESSES initialization parameter controls the number of job queue processes.

See Also: JOB_QUEUE_PROCESSES in *Oracle Database Reference*

SORT_AREA_SIZE and SORT_DIRECT_WRITES Parameters

The SORT_DIRECT_WRITES initialization parameter is obsolete in Oracle8*i* release 8.1 and higher. If you had SORT_DIRECT_WRITES set to FALSE or AUTO in a previous release, then the sort buffers were kept in the buffer cache whenever possible. Because SORT_DIRECT_WRITES is obsolete in Oracle8*i* release 8.1 and higher, the sort buffers could go directly to disk if you do not adjust your SORT_AREA_SIZE initialization parameter.

You should increase the value of SORT_AREA_SIZE if either of the following conditions were true in a previous release:

- SORT_DIRECT_WRITES was set to FALSE.
- SORT_DIRECT_WRITES was set to AUTO, and SORT_AREA_SIZE was set to 640 KB or less.

If either of these conditions were true in a previous release, then increase the value of SORT_AREA_SIZE for better performance.

New Default Value for LOG_CHECKPOINT_TIMEOUT

Starting with Oracle8*i* release 8.1.5, the LOG_CHECKPOINT_TIMEOUT initialization parameter has a new default value. In previous releases, the default value was zero seconds, but in Oracle8*i* release 8.1.5 and higher, the default value is 1800 seconds.

See Also: LOG_CHECKPOINT_TIMEOUT in *Oracle Database Reference*

The O7_DICTIONARY_ACCESSIBILITY Parameter

The O7_DICTIONARY_ACCESSIBILITY initialization parameter controls whether to continue Oracle7 data dictionary behavior. Use of this initialization parameter is only a temporary expedient. Starting with Oracle9*i* release 9.0.1, the default value of this initialization parameter is false.

See Also: "Data Dictionary Protection" on page 5-21 for more information.

The DB_DOMAIN Parameter

Starting with Oracle8*i* release 8.1, if the DB_DOMAIN initialization parameter is not set, then it is set to null by default. In previous releases of the Oracle Database, the default setting was the following:

WORLD

A null setting for DB_DOMAIN may cause database connection problems in some environments. If you are upgrading from Oracle8 release 8.0.6, then make sure the DB_DOMAIN initialization parameter in your parameter file is set to one of the following:

- WORLD
- a valid domain setting for your environment

If DB_DOMAIN is not set in your current database, then set it to WORLD before you upgrade.

If DB_DOMAIN is set to a valid domain for your environment in your current database, then retain the setting in your parameter file when you upgrade.

Parallel Execution Allocated from Large Pool

Starting with Oracle8*i* release 8.1, parallel execution message buffers are allocated from the large pool whenever PARALLEL_AUTOMATIC_TUNING is set to true. In previous releases, this allocation was from the shared pool. If you are upgrading from Oracle8 release 8.0.6, and you choose to set PARALLEL_AUTOMATIC_TUNING to true, then you can avoid problems by modifying the settings for the following initialization parameters:

- SHARED_POOL_SIZE
- LARGE_POOL_SIZE

Typically, you should reduce the setting of SHARED_POOL_SIZE and raise the setting of LARGE_POOL_SIZE to avoid problems. Alternatively, you can reduce the setting of SHARED_POOL_SIZE and let the Oracle Database calculate the setting of LARGE_POOL_SIZE. The Oracle Database calculates a default LARGE_POOL_SIZE only if PARALLEL_AUTOMATIC_TUNING is set to true and LARGE_POOL_SIZE is not set.

If PARALLEL_AUTOMATIC_TUNING is not set or is set to false, and if LARGE_POOL_SIZE is not set, then the value of LARGE_POOL_SIZE defaults to 0.

> **See Also:** *Oracle Database Reference* and *Oracle Database Performance Tuning Guide* for more information about other effects of the PARALLEL_AUTOMATIC_TUNING initialization parameter.

The following scenarios illustrate the behavior that results from various initialization parameter settings when you upgrade.

Retaining Parameter Settings without Modifications

You do not alter the parameters from their previous settings:

Table A–1 Retaining Parameter Settings without Modifications

Parameter	Setting
PARALLEL_AUTOMATIC_TUNING	Unset (defaults to FALSE).
SHARED_POOL_SIZE	Set to a large value, including the space required for parallel execution.
LARGE_POOL_SIZE	Unset (defaults to zero).

These settings are the most common scenario. In this case, you already have accounted for the space required for parallel execution in the shared pool.

Using PARALLEL_AUTOMATIC_TUNING

You alter the parameters from their previous settings to the following settings:

Table A–2 Using PARALLEL_AUTOMATIC_TUNING

Parameter	Setting
PARALLEL_AUTOMATIC_TUNING	Set to true.
SHARED_POOL_SIZE	Set to a small value that accounts for all clients except parallel execution.
LARGE_POOL_SIZE	Unset (defaults to a large value that includes the space required for parallel execution).

In this case, parallel execution allocates buffers from the large pool based on the Oracle Database's automatic calculation. Buffer allocation is more efficient, and failures to allocate are isolated from the clients of the shared pool.

Using PARALLEL_AUTOMATIC_TUNING and Setting LARGE_POOL_SIZE

You alter the parameters from their previous settings to the following settings:

Table A–3 Using PARALLEL_AUTOMATIC_TUNING and Setting LARGE_POOL_SIZE

Parameter	Setting
PARALLEL_AUTOMATIC_TUNING	Set to true.
SHARED_POOL_SIZE	Set to a small value that accounts for all clients except parallel execution.
LARGE_POOL_SIZE	Set to a value that includes the space required for parallel execution.

In this case, parallel execution allocates buffers from the large pool. After initial testing with LARGE_POOL_SIZE not set, you determined that the default calculation for LARGE_POOL_SIZE did not reflect your requirements for the large pool. Therefore, you decided to manually set LARGE_POOL_SIZE. After you set LARGE_POOL_SIZE properly, buffer allocation is more efficient, and failures to allocate are isolated from the clients of the shared pool.

Using PARALLEL_AUTOMATIC_TUNING without Modifying SHARED_POOL_SIZE

You alter the parameters from their previous settings to the following settings:

Table A–4 Using PARALLEL_AUTOMATIC_TUNING without Modifying SHARED_POOL_SIZE

Parameter	Setting
PARALLEL_AUTOMATIC_TUNING	Set to true.
SHARED_POOL_SIZE	Set to a large value, including the space required for parallel execution.
LARGE_POOL_SIZE	Unset (defaults to a large value that includes the space required for parallel execution).

In this case, parallel execution allocates buffers from the large pool, but because you did not modify SHARED_POOL_SIZE, it is likely that the SGA will be unnecessarily large, causing performance problems. Therefore, avoid setting PARALLEL_

AUTOMATIC_TUNING to true without modifying the settings of SHARED_POOL_SIZE and LARGE_POOL_SIZE appropriately.

Archive Log Destination Parameters

Oracle8*i* release 8.1 and higher supports new archive log destination parameters. After you upgrade, you can dynamically convert from the old pre-Oracle8*i* parameters (LOG_ARCHIVE_DEST and LOG_ARCHIVE_DUPLEX_DEST) to the new Oracle8*i* release 8.1 and higher parameters (LOG_ARCHIVE_DEST_*n* and LOG_ARCHIVE_DEST_STATE_*n*). You can also dynamically revert to the old parameters.

Note: In Oracle9*i*, the number of archive log destinations was increased from 5 to 10.

Changing to the New Archive Log Destination Parameters

After you determine the new archive destinations, associated states, and options, complete the following steps to change from the old archive log destination parameters to the new ones:

1. Use ALTER SYSTEM to set LOG_ARCHIVE_MIN_SUCCEED_DEST to 1.
2. Use ALTER SYSTEM to set LOG_ARCHIVE_DUPLEX_DEST to null.
3. Use ALTER SYSTEM to set LOG_ARCHIVE_DEST to null.
4. Use ALTER SYSTEM to set any LOG_ARCHIVE_DEST_STATE_*n* parameters to 'defer' or 'enable' as required. Although enable is the default, Oracle recommends that you explicitly set a state for each destination.
5. Use ALTER SYSTEM to set at least one LOG_ARCHIVE_DEST_*n* parameter to a value specifying a local destination.
6. Use ALTER SYSTEM to set other LOG_ARCHIVE_DEST_*n* parameters as required.
7. Use ALTER SYSTEM to set LOG_ARCHIVE_MIN_SUCCEED_DEST to the required value.

For example, assume there are the following two destinations:

- /oracle/dbs/arclog
- /backup/dbs/arclog

Both destinations are mandatory (minimum succeed destination count is 2). The new destinations are the following:

- /oracle/dbs/arclog (local)
- stndby1 (a standby database)
- /backup/dbs/arclog
- /backup2/dbs/arclog

The first destination, the standby destination, and either of the backup destinations are mandatory (minimum succeed destination count is 3).

With these assumptions, issue the following SQL statements to change your old archive log destination parameters to the new ones:

```
ALTER SYSTEM SET LOG_ARCHIVE_MIN_SUCCEED_DEST = 1;

ALTER SYSTEM SET LOG_ARCHIVE_DUPLEX_DEST = ' ';
```

```
ALTER SYSTEM SET LOG_ARCHIVE_DEST = ' ';

ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_1 = 'enable';
ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_2 = 'enable';
ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_3 = 'enable';
ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_4 = 'enable';

ALTER SYSTEM SET LOG_ARCHIVE_DEST_1 = 'LOCATION=/oracle/dbs/arclog MANDATORY';
ALTER SYSTEM SET LOG_ARCHIVE_DEST_2 = 'SERVICE=stndby1 MANDATORY';
ALTER SYSTEM SET LOG_ARCHIVE_DEST_3 = 'LOCATION=/backup/dbs/arclog OPTIONAL';
ALTER SYSTEM SET LOG_ARCHIVE_DEST_4 = 'LOCATION=/backup2/dbs/arclog OPTIONAL';
ALTER SYSTEM SET LOG_ARCHIVE_MIN_SUCCEED_DEST = 3;
```

Changing Back to the Old Archive Log Destination Parameters

Complete the following steps to change back to the old archive log destination parameters:

1. Use `ALTER SYSTEM` to set `LOG_ARCHIVE_MIN_SUCCEED_DEST` to 1.
2. Use `ALTER SYSTEM` to set all `LOG_ARCHIVE_DEST_`*n* parameters to NULL.
3. Use `ALTER SYSTEM` to set the `LOG_ARCHIVE_DEST` parameter to a value specifying a local destination.
4. Use `ALTER SYSTEM` to set the `LOG_ARCHIVE_DUPLEX_DEST` parameter as required.
5. Use `ALTER SYSTEM` to set `LOG_ARCHIVE_MIN_SUCCEED_DEST` to the required value.

For example, assume there are the following two destinations:

- `/oracle/dbs/arclog` (LOG_ARCHIVE_DEST_1)
- `/backup/dbs/arclog` (LOG_ARCHIVE_DEST_4)

Both destinations are mandatory. The new destinations and minimum succeed count are the same.

With these assumptions, issue the following SQL statements to change your new archive log destination parameters to the old ones:

```
ALTER SYSTEM SET LOG_ARCHIVE_MIN_SUCCEED_DEST = 1;

ALTER SYSTEM SET LOG_ARCHIVE_DEST_4 = ' ';
ALTER SYSTEM SET LOG_ARCHIVE_DEST_1 = ' ';

ALTER SYSTEM SET LOG_ARCHIVE_DEST = '/oracle/dbs/arclog';

ALTER SYSTEM SET LOG_ARCHIVE_DUPLEX_DEST = '/backup/dbs/arclog';

ALTER SYSTEM SET LOG_ARCHIVE_MIN_SUCCEED_DEST = 2;
```

Possible Errors During the Transition in Parameters

When you follow the procedures described previously in this section for changing your archive destination parameters, you may encounter the following error messages in your log files if archiving is enabled:

- In the Alert log - "Archiving not possible: No available destinations"
- In the Trace log - "ARCH: INCOMPLETE, no available destinations"

You will not encounter these errors if archiving is disabled. The errors may occur during the procedure when there are no valid archive destinations. However, when the transition in parameters is complete, the errors should cease. You should *not* disable archiving during the transition to avoid these errors.

Static Data Dictionary View Changes

The following sections list changes to static data dictionary views across different releases of the Oracle Database:

- Deprecated Static Data Dictionary Views
- Obsolete Static Data Dictionary Views
- Static Data Dictionary Views with Renamed Columns
- Static Data Dictionary Views with Dropped Columns
- Static Data Dictionary Views with Columns That May Return Nulls

> **See Also:** The "What's New in Oracle Database Reference" section of *Oracle Database Reference* for a list of new static data dictionary views in Oracle Database 10*g*

Deprecated Static Data Dictionary Views

The following sections list static data dictionary views that have been deprecated:

- Static Data Dictionary Views Deprecated in Release 10.1
- Static Data Dictionary Views Deprecated in Release 9.2
- Static Data Dictionary Views Deprecated in Release 9.0.1
- Static Data Dictionary Views Deprecated in Release 8.1

Static Data Dictionary Views Deprecated in Release 10.1

The following static data dictionary views were deprecated in release 10.1:

`ALL_STORED_SETTINGS` (replaced by `ALL_PLSQL_OBJECT_SETTINGS`)
`DBA_STORED_SETTINGS` (replaced by `DBA_PLSQL_OBJECT_SETTINGS`)
`USER_STORED_SETTINGS` (replaced by `USER_PLSQL_OBJECT_SETTINGS`)

Static Data Dictionary Views Deprecated in Release 9.2

The following static data dictionary views were deprecated in release 9.2:

`ALL_RULESETS` (replaced by `ALL_RULE_SETS`)
`DBA_RULESETS` (replaced by `DBA_RULE_SETS`)
`USER_RULESETS` (replaced by `USER_RULE_SETS`)

Static Data Dictionary Views Deprecated in Release 9.0.1

The following static data dictionary views were deprecated in release 9.0.1:

`ALL_REGISTERED_SNAPSHOTS` (replaced by `ALL_REGISTERED_MVIEWS`)
`ALL_SNAPSHOT_LOGS` (replaced by `ALL_BASE_TABLE_MVIEWS` and `ALL_MVIEW_LOGS`)
`ALL_SNAPSHOT_REFRESH_TIMES` (replaced by `ALL_MVIEW_REFRESH_TIMES`)

DBA_REGISTERED_SNAPSHOT_GROUPS (replaced by DBA_REGISTERED_MVIEW_GROUPS)
DBA_REGISTERED_SNAPSHOTS (replaced by DBA_REGISTERED_MVIEWS)
DBA_SNAPSHOT_LOG_FILTER_COLS (replaced by DBA_MVIEW_LOG_FILTER_COLS)
DBA_SNAPSHOT_LOGS (replaced by DBA_BASE_TABLE_MVIEWS and DBA_MVIEW_LOGS)
DBA_SNAPSHOT_REFRESH_TIMES (replaced by DBA_MVIEW_REFRESH_TIMES)
USER_REGISTERED_SNAPSHOTS (replaced by USER_REGISTERED_MVIEWS)
USER_SNAPSHOT_LOGS (replaced by USER_BASE_TABLE_MVIEWS and USER_MVIEW_LOGS)
USER_SNAPSHOT_REFRESH_TIMES (replaced by USER_MVIEW_REFRESH_TIMES)

Static Data Dictionary Views Deprecated in Release 8.1

The following static data dictionary views were deprecated in release 8.1:

ALL_SNAPSHOTS (replaced by ALL_MVIEWS)
ALL_SUMMARIES (replaced by ALL_MVIEW_ANALYSIS)
ALL_SUMMARY_AGGREGATES (replaced by ALL_MVIEW_AGGREGATES)
ALL_SUMMARY_DETAIL_TABLES (replaced by ALL_MVIEW_DETAIL_RELATIONS)
ALL_SUMMARY_JOINS (replaced by ALL_MVIEW_JOINS)
ALL_SUMMARY_KEYS (replaced by ALL_MVIEW_KEYS)
DBA_SNAPSHOTS (replaced by DBA_MVIEWS)
DBA_SUMMARIES (replaced by DBA_MVIEW_ANALYSIS)
DBA_SUMMARY_AGGREGATES (replaced by DBA_MVIEW_AGGREGATES)
DBA_SUMMARY_DETAIL_TABLES (replaced by DBA_MVIEW_DETAIL_RELATIONS)
DBA_SUMMARY_JOINS (replaced by DBA_MVIEW_JOINS)
DBA_SUMMARY_KEYS (replaced by DBA_MVIEW_KEYS)
USER_SNAPSHOTS (replaced by USER_MVIEWS)
USER_SUMMARIES (replaced by USER_MVIEW_ANALYSIS)
USER_SUMMARY_AGGREGATES (replaced by USER_MVIEW_AGGREGATES)
USER_SUMMARY_DETAIL_TABLES (replaced by USER_MVIEW_DETAIL_RELATIONS)
USER_SUMMARY_JOINS (replaced by USER_MVIEW_JOINS)
USER_SUMMARY_KEYS (replaced by USER_MVIEW_KEYS)

Obsolete Static Data Dictionary Views

The following sections list static data dictionary views that have been made obsolete:

- Static Data Dictionary Views Obsolete in Release 10.1

Static Data Dictionary Views Obsolete in Release 10.1

The following static data dictionary views were made obsolete in release 10.1:

ALL_ Views	DBA_ Views	USER_ Views
ALL_SOURCE_TAB_COLUMNS	DBA_SOURCE_TAB_COLUMNS	USER_SOURCE_TAB_COLUMNS

Static Data Dictionary Views with Renamed Columns

The following sections list static data dictionary views with renamed columns:

- Static Data Dictionary Views with Renamed Columns in Release 9.0.1

Static Data Dictionary Views with Renamed Columns in Release 9.0.1

The static data dictionary view columns listed in Table A–5 were renamed in release 9.0.1:

Table A–5 Static Data Dictionary Views with Renamed Columns in Release 9.0.1

Static Data Dictionary View	Pre-Release 9.0.1 Column Name	Release 9.0.1 and Higher Column Name
DBA_RSRC_PLAN_DIRECTIVES	MAX_ACTIVE_SESS_TARGET_P1	ACTIVE_SESS_POOL_P1
DBA_RSRC_PLANS	MAX_ACTIVE_SESS_TARGET_MTH	ACTIVE_SESS_POOL_MTH

Static Data Dictionary Views with Dropped Columns

The following sections list static data dictionary views with dropped columns:

- Static Data Dictionary Views with Dropped Columns in Release 9.0.1
- Static Data Dictionary Views with Dropped Columns in Release 8.1

Static Data Dictionary Views with Dropped Columns in Release 9.0.1

The following static data dictionary view columns were dropped in release 9.0.1:

Static Data Dictionary View	Dropped Columns
DBA_RSRC_PLAN_DIRECTIVES	MAX_ACTIVE_SESS_TARGET_P1
DBA_RSRC_PLANS	MAX_ACTIVE_SESS_TARGET_MTH

Static Data Dictionary Views with Dropped Columns in Release 8.1

The following static data dictionary view columns were dropped in release 8.1:

Static Data Dictionary Views	Dropped Columns
DBA_AUDIT_OBJECT	OBJECT_LABEL
USER_AUDIT_OBJECT	SESSION_LABEL
DBA_AUDIT_SESSION	SESSION_LABEL
USER_AUDIT_SESSION	
DBA_AUDIT_STATEMENT	SESSION_LABEL
USER_AUDIT_STATEMENT	
DBA_AUDIT_TRAIL	OBJECT_LABEL
USER_AUDIT_TRAIL	SESSION_LABEL
DBA_CONTEXT	ATTRIBUTE
ALL_IND_COLUMNS	COLUMN_EXPRESSION
DBA_IND_COLUMNS	
USER_IND_COLUMNS	
ALL_JOBS	CLEARANCE_HI
DBA_JOBS	CLEARANCE_LO
USER_JOBS	CURRENT_SESSION_LABEL

Static Data Dictionary Views	Dropped Columns
ALL_REFS	HAS_REFERENTIAL_CONS
DBA_REFS	REFERENTIAL_CONS_NAME
USER_REFS	

Static Data Dictionary Views with Columns That May Return Nulls

Starting with release 8.1, the static data dictionary view columns listed in Table A–6 may return nulls; in previous releases, these columns could not return nulls. If an application requires non-null values for one or more of these columns, then modify the application accordingly:

Table A–6 Static Data Dictionary Views with Columns That May Return Nulls in Release 8.1

Static Data Dictionary Views	Columns	Explanation
DBA_DATA_FILES	AUTOEXTENSIBLE BLOCKS BYTES INCREMENT_BY MAXBLOCKS MAXBYTES	These columns return a null if the data file is offline and therefore not readable.
ALL_IND_COLUMNS DBA_IND_COLUMNS USER_IND_COLUMNS	COLUMN_NAME	This column returns a null if an index is on a function instead of a column. In this case, there is no column to list.
ALL_IND_PARTITIONS DBA_IND_PARTITIONS USER_IND_PARTITIONS	INITIAL_EXTENT MAX_EXTENT MIN_EXTENT NEXT_EXTENT PCT_INCREASE	These columns return a null if the index is partitioned using a composite method and no default value was specified for the partition.
ALL_OBJECT_TABLES DBA_OBJECT_TABLES USER_OBJECT_TABLES	TABLESPACE_NAME	This column returns a null in if an object table is partitioned or if it is a temporary table.
ALL_SEGMENTS DBA_SEGMENTS USER_SEGMENTS	BLOCKS BYTES EXTENTS NEXT_EXTENT PCT_INCREASE	The BLOCKS, BYTES, and EXTENTS columns return a null if the segment header cannot be read because the file is offline or if there is some other corruption. The NEXT_EXTENT and PCT_INCREASE columns return a null if the tablespace storing the segment is locally managed and uses the AUTOALLOCATE option, because the system chooses the extent sizes, and the algorithm cannot be explained in terms of NEXT_EXTENT and PCT_INCREASE.

Table A–6 (Cont.) Static Data Dictionary Views with Columns That May Return Nulls in Release 8.1

Static Data Dictionary Views	Columns	Explanation
ALL_TAB_PARTITIONS DBA_TAB_PARTITIONS USER_TAB_PARTITIONS	INITIAL_EXTENT MAX_EXTENT MIN_EXTENT NEXT_EXTENT PCT_INCREASE	These columns return a null if the table is partitioned using a composite method and no default value was specified for the partition.
ALL_TABLESPACES DBA_TABLESPACES USER_TABLESPACES	NEXT_EXTENT PCT_INCREASE	These columns return a null if the tablespace is locally managed and uses the AUTOALLOCATE option, because the system chooses the extent sizes, and the algorithm cannot be explained in terms of NEXT_EXTENT and PCT_INCREASE.
ALL_TRIGGERS DBA_TRIGGERS USER_TRIGGERS	TABLE_NAME	This column returns a null if the trigger is a system trigger. In this case, the base object type of the trigger will be SCHEMA or DATABASE, instead of TABLE or VIEW.

Dynamic Performance View Changes

The following sections list changes to dynamic performance views (V$ views) across different releases of the Oracle Database:

- Deprecated Dynamic Performance Views
- Obsolete Dynamic Performance Views
- Dynamic Performance Views with Renamed Columns
- Dynamic Performance Views with Dropped Columns

> **See Also:** The "What's New in Oracle Database Reference" section of *Oracle Database Reference* for a list of new dynamic performance views in Oracle Database 10*g*

Deprecated Dynamic Performance Views

The following sections list dynamic performance views that have been deprecated:

- Dynamic Performance Views Deprecated in Release 10.1
- Dynamic Performance Views Deprecated in Release 9.2
- Dynamic Performance Views Deprecated in Release 9.0.1

Dynamic Performance Views Deprecated in Release 10.1

The following dynamic performance views were deprecated in release 10.1:

GV$CACHE
GV$CACHE_TRANSFER
GV$CLASS_CACHE_TRANSFER (replaced by GV$INSTANCE_CACHE_TRANSFER)
GV$FALSE_PING
GV$FILE_CACHE_TRANSFER (replaced by GV$INSTANCE_CACHE_TRANSFER)
GV$GC_ELEMENTS_WITH_COLLISIONS
GV$LOCK_ACTIVITY
GV$TEMP_CACHE_TRANSFER (replaced by GV$INSTANCE_CACHE_TRANSFER)

V$CACHE
V$CACHE_LOCK
V$CACHE_TRANSFER
V$CLASS_CACHE_TRANSFER (replaced by V$INSTANCE_CACHE_TRANSFER)
V$FALSE_PING
V$FILE_CACHE_TRANSFER (replaced by V$INSTANCE_CACHE_TRANSFER)
V$GC_ELEMENTS_WITH_COLLISIONS
V$LOCK_ACTIVITY
V$TEMP_CACHE_TRANSFER (replaced by V$INSTANCE_CACHE_TRANSFER)

Dynamic Performance Views Deprecated in Release 9.2

The following dynamic performance views were deprecated in release 9.2:

GV$SORT_USAGE (replaced by GV$TEMPSEG_USAGE)
V$SORT_USAGE (replaced by V$TEMPSEG_USAGE)

Dynamic Performance Views Deprecated in Release 9.0.1

The following dynamic performance views were deprecated in release 9.0.1:

GV$BSP (replaced by GV$CR_BLOCK_SERVER)
GV$CLASS_PING (replaced by GV$CLASS_CACHE_TRANSFER)
GV$DLM_ALL_LOCKS (replaced by GV$GES_ENQUEUE)
GV$DLM_CONVERT_LOCAL (replaced by GV$GES_CONVERT_LOCAL)
GV$DLM_CONVERT_REMOTE (replaced by GV$GES_CONVERT_REMOTE)
GV$DLM_LATCH (replaced by GV$GES_LATCH)
GV$DLM_LOCKS (replaced by GV$GES_BLOCKING_ENQUEUE)
GV$DLM_MISC (replaced by GV$GES_STATISTICS)
GV$DLM_RESS (replaced by GV$GES_RESOURCE)
GV$DLM_TRAFFIC_CONTROLLER (replaced by GV$GES_TRAFFIC_CONTROLLER)
GV$FILE_PING (replaced by GV$FILE_CACHE_TRANSFER)
GV$LOCK_ELEMENT (replaced by GV$GC_ELEMENT)
GV$LOCKS_WITH_COLLISIONS (replaced by GV$GC_ELEMENTS_WITH_
COLLISIONS)
GV$MAX_ACTIVE_SESS_TARGET_MTH (replaced by GV$ACTIVE_SESS_POOL_MTH)
GV$MTS (replaced by GV$SHARED_SERVER_MONITOR)
GV$PING (replaced by GV$CACHE_TRANSFER)
GV$TEMP_PING (replaced by GV$TEMP_CACHE_TRANSFER)
V$BSP (replaced by V$CR_BLOCK_SERVER)
V$CLASS_PING (replaced by V$CLASS_CACHE_TRANSFER)
V$DLM_ALL_LOCKS (replaced by V$GES_ENQUEUE)
V$DLM_CONVERT_LOCAL (replaced by V$GES_CONVERT_LOCAL)
V$DLM_CONVERT_REMOTE (replaced by V$GES_CONVERT_REMOTE)
V$DLM_LATCH (replaced by V$GES_LATCH)
V$DLM_LOCKS (replaced by V$GES_BLOCKING_ENQUEUE)
V$DLM_MISC (replaced by V$GES_STATISTICS)
V$DLM_RESS (replaced by V$GES_RESOURCE)
V$DLM_TRAFFIC_CONTROLLER (replaced by V$GES_TRAFFIC_CONTROLLER)
V$FILE_PING (replaced by V$FILE_CACHE_TRANSFER)
V$LOCK_ELEMENT (replaced by V$GC_ELEMENT)
V$LOCKS_WITH_COLLISIONS (replaced by V$GC_ELEMENTS_WITH_COLLISIONS)
V$MAX_ACTIVE_SESS_TARGET_MTH (replaced by V$ACTIVE_SESS_POOL_MTH)
V$MTS (replaced by V$SHARED_SERVER_MONITOR)
V$PING (replaced by V$CACHE_TRANSFER)

V$TEMP_PING (replaced by V$TEMP_CACHE_TRANSFER)

Obsolete Dynamic Performance Views

The following sections list dynamic performance views that have been made obsolete:

- Dynamic Performance Views Obsolete in Release 10.1
- Dynamic Performance Views Obsolete in Release 9.2
- Dynamic Performance Views Obsolete in Release 9.0.1
- Dynamic Performance Views Obsolete in Release 8.1

Dynamic Performance Views Obsolete in Release 10.1

The following dynamic performance views were made obsolete in release 10.1:

GV$ Views	V$ Views
GV$COMPATIBILITY	V$COMPATIBILITY
GV$COMPATSEG	V$COMPATSEG
GV$MLS_PARAMETERS	V$MLS_PARAMETERS
GV$MTS	V$MTS

Dynamic Performance Views Obsolete in Release 9.2

The following dynamic performance views were made obsolete in release 9.2:

GV$ Views	V$ Views
GV$LOADCSTAT	V$LOADCSTAT
GV$LOADTSTAT	V$LOADTSTAT

Dynamic Performance Views Obsolete in Release 9.0.1

The following dynamic performance views were made obsolete in release 9.0.1:

GV$ Views	V$ Views
GV$TARGETRBA	V$TARGETRBA

Dynamic Performance Views Obsolete in Release 8.1

The following dynamic performance views were made obsolete in release 8.1:

GV$ Views	V$ Views
GV$CURRENT_BUCKET	V$CURRENT_BUCKET
GV$RECENT_BUCKET	V$RECENT_BUCKET

Dynamic Performance Views with Renamed Columns

The following sections list dynamic performance views with renamed columns:

- Dynamic Performance Views with Renamed Columns in Release 9.2
- Dynamic Performance Views with Renamed Columns in Release 9.0.1

Dynamic Performance Views with Renamed Columns in Release 9.2

The dynamic performance view columns listed in Table A–7 were renamed in release 9.2:

Table A–7 Dynamic Performance Views with Renamed Columns in Release 9.2

Dynamic Performance View	Pre-Release 9.2 Column Name	Release 9.2 and Higher Column Name
GV$ARCHIVE_DEST and V$ARCHIVE_DEST	MANIFEST	REGISTER
	REGISTER	REMOTE_TEMPLATE
GV$DATABASE and V$DATABASE	STANDBY_MODE	PROTECTION_MODE
GV$LOGMNR_CALLBACK and V$LOGMNR_CALLBACK	CALLBACK_STATE	STATE
	CALLBACK_TYPE	TYPE
	CALLBACK_CAPABILITY	CAPABILITY
GV$LOGMNR_REGION and V$LOGMNR_REGION	ID	MEMSTATE
	CURRENT_STATE	STATE

Dynamic Performance Views with Renamed Columns in Release 9.0.1

The dynamic performance view columns listed in Table A–8 were renamed in release 9.0.1:

Table A–8 Dynamic Performance Views with Renamed Columns in Release 9.0.1

Dynamic Performance View	Pre-Release 9.0.1 Column Name	Release 9.0.1 and Higher Column Name
GV$RSRC_CONSUMER_GROUP and V$RSRC_CONSUMER_GROUP	SESSIONS_QUEUED	QUEUE_LENGTH

Dynamic Performance Views with Renamed Columns in Release 8.1

The dynamic performance view columns listed in Table A–9 were renamed in release 8.1:

Table A–9 Dynamic Performance Views with Renamed Columns in Release 8.1

Dynamic Performance View	Pre-Release 8.1 Column Name	Release 8.1 and Higher Column Name
GV$DISPATCHER_RATE and V$DISPATCHER_RATE	NUM_LOOPS_TRACKED	TTL_LOOPS
	NUM_MSG_TRACKED	TTL_MSG
	NUM_SVR_BUF_TRACKED	TTL_SVR_BUF
	NUM_CLT_BUF_TRACKED	TTL_CLT_BUF
	NUM_BUF_TRACKED	TTL_BUF
	NUM_IN_CONNECT_TRACKED	TTL_IN_CONNECT
	NUM_OUT_CONNECT_TRACKED	TTL_OUT_CONNECT
	NUM_RECONNECT_TRACKED	TTL_RECONNECT

Dynamic Performance Views with Dropped Columns

The following sections list dynamic performance views with dropped columns. If an application requires one or more of these columns, then modify the application accordingly:

- Dynamic Performance Views with Dropped Columns in Release 9.2
- Dynamic Performance Views with Dropped Columns in Release 9.0.1
- Dynamic Performance Views with Dropped Columns in Release 8.1

Dynamic Performance Views with Dropped Columns in Release 9.2

The following dynamic performance view columns were dropped in release 9.2:

Dynamic Performance View	Dropped Columns
GV$DATABASE and V$DATABASE	STANDBY_MODE
GV$LOGMNR_CALLBACK and V$LOGMNR_CALLBACK	FUNC_NAME
	CALLBACK_ID
	CALLBACK_RESULT_SIZE
	CALLBACK_STATE
	CALLBACK_TYPE
	CALLBACK_CAPABILITY
	NUMBER_INVOKED
GV$LOGMNR_REGION and V$LOGMNR_REGION	ID
	CURRENT_STATE

Dynamic Performance Views with Dropped Columns in Release 9.0.1

The following dynamic performance view columns were dropped in release 9.0.1:

Dynamic Performance View	Dropped Columns
GV$LOGMNR_CONTENTS and V$LOGMNR_CONTENTS	PH1_NAME
	PH1_REDO
	PH1_UNDO
	PH2_NAME
	PH2_REDO
	PH2_UNDO
	PH3_NAME
	PH3_REDO
	PH3_UNDO
	PH4_NAME
	PH4_REDO
	PH4_UNDO
	PH5_NAME
	PH5_REDO
	PH5_UNDO
GV$RSRC_CONSUMER_GROUP and V$RSRC_CONSUMER_GROUP	SESSIONS_QUEUED

Dynamic Performance Views with Dropped Columns in Release 8.1

The following dynamic performance view columns were dropped in release 8.1:

Dynamic Performance View	Dropped Columns
V$ARCHIVE_DEST	ARCMODE
V$DLM_LATCH	IMM_GETS
	LATCH_TYPE
	TTL_GETS
V$DLM_LOCKS	RESOURCE_NAME
V$SESSION_LONGOPS	APPLICATION_DATA_1
	APPLICATION_DATA_2
	APPLICATION_DATA_3
	COMPNAM
	CURRENT_TIME
	MSG
	OBJID
	OPID
	STEPID
	STEPSOFAR
	STEPTOTAL
	UPDATE_COUNT

B

Migrating from Server Manager to SQL*Plus

This appendix guides you through the process of modifying your Server Manager line mode scripts to work with SQL*Plus. Server Manager is not supported in Oracle9i release 9.0.1 and later. If you run SQL scripts using Server Manager line mode, then you will need to change these scripts so that they are compatible with SQL*Plus, and then run them using SQL*Plus.

This appendix covers the following topics:

- Startup Differences
- Commands
- Syntax Differences

> **See Also:** *SQL*Plus User's Guide and Reference* for detailed information about SQL*Plus

> **Note:** For brevity, Server Manager line mode is referred to as Server Manager in the rest of this appendix.

Startup Differences

The methods for starting Server Manager and SQL*Plus are different, and your SQL scripts must be modified to properly start SQL*Plus. The following sections explain the startup differences and provide options for starting SQL*Plus.

Starting Server Manager

To start Server Manager, enter the name of the Server Manager program at a system prompt; the name of this program is operating system-specific. After you start up Server Manager, connect using the `CONNECT` command, as in the following example:

```
CONNECT hr/hr
```

Starting SQL*Plus

The following sections describe various ways to start SQL*Plus.

Starting SQL*Plus with the NOLOG Option

If you want SQL*Plus to behave in the same way as Server Manager, then use the `NOLOG` option when you start SQL*Plus, as in the following example:

```
sqlplus /nolog
```

SQL*Plus starts and you can use the CONNECT command to connect as a user.

Starting SQL*Plus with Connect Information

Another option for starting SQL*Plus is to enter the connect information when you start the program. For example, to start SQL*Plus and connect as user hr with password hr, enter the following:

```
sqlplus hr/hr
```

SQL*Plus starts and connects as user hr.

Starting SQL*Plus without Options or Connect Information

To start SQL*Plus without options or connect information, enter the following:

```
sqlplus
```

SQL*Plus prompts you for a user name and password. When you enter a valid user name and password, SQL*Plus starts and connects as the user you specified at the prompts. In your SQL scripts, however, you may not want to prompt the user to enter a user name and password.

Commands

Server Manager and SQL*Plus share certain commands that behave the same in both programs. Other commands, however, behave differently in SQL*Plus than they do in Server Manager. To successfully migrate from Server Manager to SQL*Plus, you need to understand these differences and similarities. The following sections include information about modifying your SQL scripts to use commands that are interpreted correctly by SQL*Plus.

Commands Introduced in SQL*Plus Release 8.1

Table B–1 lists Server Manager commands that are available in SQL*Plus release 8.1 and higher. You can use these commands in SQL scripts that you run with SQL*Plus.

Note: If you run SQL scripts containing any of these commands in Oracle7 or release 8.0, then you must use Server Manager to run these scripts. Versions of SQL*Plus before SQL*Plus release 8.1 will not run scripts containing these commands.

Table B–1 Commands Introduced in SQL*Plus Release 8.1

Command	Description
ARCHIVE LOG	Starts or stops automatic archiving of online redo log files, manually (explicitly) archives specified redo log files, or displays information about archives.
RECOVER	Performs media recovery on one or more tablespaces, one or more datafiles, or the entire database.
SET AUTORECOVERY	ON causes the RECOVER command to automatically apply the default filenames of archived redo log files needed during recovery. No interaction is needed when AUTORECOVERY is set to ON, provided the necessary files are in the expected locations with the expected names.

*Table B–1 (Cont.) Commands Introduced in SQL*Plus Release 8.1*

Command	Description
SET INSTANCE	Changes the default instance for your session to the specified instance path. Does not connect to a database. The default instance is used for commands when no instance is specified.
SET LOGSOURCE	Specifies the location from which archive logs are retrieved during recovery. The default value is set by the LOG_ARCHIVE_DEST initialization parameter. Issuing the SET LOGSOURCE command without a pathname restores the default location.
SHOW AUTORECOVERY	Shows whether autorecovery is enabled.
SHOW INSTANCE	Shows the connect string for the default instance. SHOW INSTANCE returns the value LOCAL if you have not used SET INSTANCE or if you have used the LOCAL option of the SET INSTANCE command.
SHOW LOGSOURCE	Shows the current setting of the archive log location. Displays DEFAULT if the default setting is in effect, as specified by the LOG_ARCHIVE_DEST initialization parameter.
SHOW PARAMETERS	Displays the current values of one or more initialization parameters. The SHOW PARAMETERS command, without any string following the command, displays all initialization parameters.
SHOW SGA	Displays information about the current instance's System Global Area.
SHUTDOWN	Shuts down a currently running Oracle instance, optionally closing and dismounting a database. **Note:** The STARTUP and SHUTDOWN commands in SQL*Plus release 8.1 are not supported against an Oracle7 server.
STARTUP	Starts an Oracle instance with several options, including mounting and opening a database. **Note:** The STARTUP and SHUTDOWN commands in SQL*Plus release 8.1 are not supported against an Oracle7 server.

Commands Common to Server Manager and SQL*Plus

The commands listed in Table B–2 are available in both Server Manager and SQL*Plus, and have been available in both programs in past releases of Oracle. You do not need to alter these commands in your SQL scripts to use SQL*Plus.

Note: There may be minor formatting differences in the output for these commands in the two programs.

*Table B–2 Server Manager Commands Corresponding to Existing SQL*Plus Commands*

Command	Description
CONNECT	Connects to a database using the specified user name.
DESCRIBE	Describes a function, package, package body, procedure, table, view, or object type. For example, for a table, displays the definitions of each column in the table.
REMARK	Enters a comment, typically in SQL script files.
SET COMPATIBILITY	Sets compatibility mode to V7, V8, or NATIVE. The compatibility mode setting affects the specification of character columns, integrity constraints, and rollback segment storage parameters. NATIVE matches the version of the database.
SET ECHO	Controls whether the START command lists each command in a command file as the command is executed. ON lists the commands; OFF suppresses the listing.
SET NUMWIDTH	Sets the default width for displaying numbers.

*Table B–2 (Cont.) Server Manager Commands Corresponding to Existing SQL*Plus Commands*

Command	Description
`SET SERVEROUTPUT`	Controls whether to display the output (that is, `DBMS_OUTPUT.PUT_LINE`) of stored procedures or PL/SQL blocks in SQL*Plus. `OFF` suppresses the output of `DBMS_OUTPUT.PUT_LINE`; `ON` displays the output.
`SET TERMOUT`	Controls the display of output generated by commands executed from a command file. `OFF` suppresses the display so that you can spool output from a command file without seeing the output on the screen. `ON` displays the output.
`SHOW ALL`	Lists all of the system variables set by the `SET` command in alphabetical order, except ERRORS, PARAMETERS, and SGA.
`SHOW ERRORS`	Shows the errors generated from the last compilation of a procedure, package, or function, if any.
`SPOOL`	Stores query results in an operating system file and, optionally in SQL*Plus, sends the file to a printer. **Note:**The extension of spool files may differ between SQL*Plus and Server Manager. To ensure an extension, specify it when you issue the `SPOOL` command. Also, SQL*Plus may format white space in terminal output using tab characters in place of repeated spaces. Use `SET TAB OFF` in SQL*Plus to prevent this replacement. Server Manager never outputs tab characters.

SQL*Plus Equivalents for Server Manager Commands

Table B–3 lists the SQL*Plus commands that correspond to Server Manager commands with different names. If you are using any of these Server Manager commands in SQL scripts, then modify the scripts to use the SQL*Plus commands instead.

*Table B–3 SQL*Plus Equivalents for Server Manager Commands*

Server Manager Commands	SQL*Plus Commands	Description
SET CHARWIDTH SET DATEWIDTH SET LONGWIDTH	COLUMN FORMAT	You can use the COLUMN FORMAT command in SQL*Plus to set the column width of character columns, date columns, and number columns. In your SQL scripts, replace the SET CHARWIDTH, SET DATEWIDTH, and SET LONGWIDTH Server Manager commands with the SQL*Plus COLUMN FORMAT command. Use COLUMN FORMAT for all character columns to be changed. There is no equivalent command to change all character columns with one command. For example, suppose you have the following entry in a SQL script: SET CHARWIDTH 5 This command sets the width for all character columns to 5 in Server Manager. To specify that a particular column, such as first_name, display with a width of 5 characters, enter the following SQL*Plus command: COLUMN first_name FORMAT A5 Use COLUMN FORMAT for all character columns to be changed. There is no equivalent command to change all character columns with one command. Use COLUMN FORMAT for all date columns to be changed. There is no equivalent command to change all date columns with one command. Use SET LONG to specify how much of the LONG column to fetch and display.
SET STOPONERROR	WHENEVER SQLERROR WHENEVER OSERROR	Use the WHENEVER SQLERROR and WHENEVER OSERROR commands to direct SQL*Plus to either exit or continue whenever a SQL error or operating system error occurs. Use these commands in your SQL scripts instead of the Server Manager SET STOPONERROR command. For both WHENEVER SQLERROR and WHENEVER OSERROR, the EXIT clause directs SQL*Plus to exit, while the CONTINUE clause directs SQL*Plus to continue. Other terms and clauses are also available for these commands.

Possible Differences in the SET TIMING Command

The SET TIMING command is available in both Server Manager and SQL*Plus, but this command may function differently in the two programs on some operating systems. Check your operating system-specific Oracle documentation for more information. If the SET TIMING command functions differently in these two programs on your operating system, then modify your SQL scripts so that this command functions properly with SQL*Plus.

Server Manager Commands Unavailable in SQL*Plus

The following Server Manager commands are unavailable in SQL*Plus release 8.1 and higher:

- SET MAXDATA
- SET RETRIES

Remove these commands from your SQL scripts.

Syntax Differences

The following sections explain the syntax differences between Server Manager and SQL*Plus. Modify your SQL scripts to conform with SQL*Plus syntax conventions before you attempt to run your scripts using SQL*Plus.

Comments

SQL*Plus recognizes the following types of comments:

- the SQL*Plus REMARK command (or REM)
- the SQL comment delimiters, /* ... */
- the ANSI/ISO comments, --

The *SQL*Plus User's Guide and Reference* provides detailed information about using these types of comments in SQL*Plus scripts.

Server Manager supports these types of comments, but the behavior is different for some of them. Also, certain types of comments are available in Server Manager, but not in SQL*Plus. The sections below discuss each type of comment and the syntax differences between Server Manager and SQL*Plus.

REMARK Command (or REM)

In general, the `REMARK` command works the same in Server Manager and SQL*Plus, and you do not need to change the occurrences of the `REMARK` command in your SQL scripts. There is, however, one difference: SQL*Plus interprets a hyphen that terminates a `REMARK` command differently than Server Manager. See "Hyphens Used as Dividing Lines" on page B-8 for information about this difference.

SQL Comment Delimiters, /* ... */

In Server Manager, the SQL comment delimiters can be placed after a semicolon (;), but in SQL*Plus, placing a SQL comment delimiter after a semicolon is not allowed. Except for this one difference, SQL comment delimiters work the same in Server Manager and SQL*Plus.

If your SQL scripts contain any SQL comment delimiters placed after a semicolon , then either move the comment to its own line, or remove the semicolon and place a slash (/) on the next line to end the SQL statement.

For example, suppose you have the following Server Manager code in one of your SQL scripts:

```
SELECT * FROM hr.employees
    WHERE job_id LIKE '%CLERK'; /* Includes only clerks. */
```

In SQL*Plus, replace this code with either of the following entries:

```
SELECT * FROM hr.employees
    WHERE job_id LIKE '%CLERK';
/* Includes only clerks. */

SELECT * FROM hr.employees
    WHERE job_id LIKE '%CLERK' /* Includes only clerks. */
    /
```

ANSI/ISO Comments, --

In Server Manager, the ANSI/ISO comments can be placed after a semicolon (;), but in SQL*Plus, placing an ANSI/ISO comment after a semicolon is not allowed. Except for this one difference, ANSI/ISO comments work the same in Server Manager and SQL*Plus.

If your SQL scripts contain any ANSI/ISO comments that are placed after a semicolon, then either move the comment to its own line, or remove the semicolon and place a slash (/) on the next line to end the SQL statement.

For example, suppose you have the following Server Manager code in one of your SQL scripts:

```
SELECT * FROM hr.employees
    WHERE job_id LIKE '%CLERK'; -- Includes only clerks.
```

In SQL*Plus, replace this code with either of the following entries:

```
SELECT * FROM hr.employees
    WHERE job_id LIKE '%CLERK';
-- Includes only clerks.

SELECT * FROM hr.employees
    WHERE job_id LIKE '%CLERK' -- Includes only clerks.
    /
```

Server Manager Pound (#) Comments

Server Manager supports the use of the pound sign (#) to indicate a comment line. If your scripts contain these comments, then change the '#' to '--' to run the scripts using SQL*Plus.

For example, suppose you have the following Server Manager code in one of your SQL scripts:

```
# This statement returns only clerks.
SELECT * FROM hr.employees
    WHERE job_id LIKE '%CLERK';
```

In SQL*Plus, replace this code with the following entry:

```
-- This statement returns only clerks.
SELECT * FROM hr.employees
    WHERE job_id LIKE '%CLERK';
```

Blank Lines

Server Manager ignores blank lines within SQL statements, but when SQL*Plus encounters a blank line the default behavior is to stop recording the statement and return to the prompt.

Both products allow blank lines between distinct SQL statements. This section only applies to blank lines between clauses of SQL statements.

In SQL*Plus, the `SET SQLBLANKLINES` command alters the way blank lines are handled. When `SQLBLANKLINES` is set to `OFF`, the default setting, and there is a SQL statement containing a blank line, SQL*Plus buffers the statement at the blank line, returning to the prompt without executing the statement. This behavior allows interactive users to abort and buffer an unwanted SQL command, or to perform other SQL*Plus commands before executing or editing this buffered SQL command.

If any of your SQL scripts contain blank lines within SQL statements, then either set SQLBLANKLINES to ON, or remove the blank lines before you run these scripts using SQL*Plus.

For example, suppose you have the following SQL statement in one of your SQL scripts:

```
SELECT employee_id, first_name, last_name, salary, commission_pct

    FROM hr.employees

    WHERE job_id LIKE '%MAN';
```

Either set SQLBLANKLINES to ON, or delete the blank lines:

```
SELECT employee_id, first_name, last_name, salary, commission_pct
    FROM hr.employees
    WHERE job_id LIKE '%MAN';
```

If you do not remove the blank lines or set SQLBLANKLINES to ON, then SQL*Plus will treat each blank line of code as a command terminator.

The value of SQLBLANKLINES does not affect blank lines in PL/SQL blocks. These are always treated as part of the block and do not return to the SQL*Plus prompt.

Interactive users can terminate SQL or PL/SQL statements by entering a period on a line by itself, regardless of the value of SQLBLANKLINES.

The Hyphen Continuation Character

SQL*Plus supports the use of a hyphen as a continuation character for long SQL statements or SQL*Plus commands. For example, you can use the continuation character in the following way:

```
SELECT employee_id, first_name, last_name FROM hr.employees -
WHERE job_id LIKE '%MAN';
```

Server Manager does not support the use of a hyphen as a continuation character, but you may use hyphens for other purposes in your SQL scripts. If you do, then SQL*Plus may interpret a hyphen as a continuation character, which can cause unexpected output.

The following sections provide scenarios in which SQL*Plus interprets the use of hyphens in SQL scripts as continuation characters, when the hyphens were meant for another purpose. Check your SQL scripts for the use of hyphens and modify them to avoid scenarios similar to those described below.

Hyphens Used as Dividing Lines

Your SQL scripts may use a long row of hyphens following a REMARK command as a dividing line in the code. Consider the following sample lines from a SQL script:

```
Rem ------------------------------------------------------------------------
SELECT employee_id, first_name, last_name, job_id
    FROM hr.employees;
```

In this statement, SQL*Plus interprets the first line of the SELECT statement as a continuation of the previous line, which is a REMARK comment. Therefore, the FROM line is interpreted as the first line of a SQL statement, and SQL*Plus returns the following error:

```
unknown command beginning "FROM hr..." - rest of line ignored.
```

If you use hyphens as dividing lines in your SQL scripts, then remove the REM command preceding the hyphens before you run the scripts using SQL*Plus.

Hyphens Used as Minus Signs

Because the hyphen is the same keyboard character as the minus sign, you may have a hyphen at the end of a line. Consider the following sample lines from a SQL script:

```
CREATE TABLE xx (
    a int,
    b int,
    c int);

INSERT INTO xx VALUES (10, 20, 30);

SELECT a + b -
    c FROM xx;
```

SQL*Plus interprets the 'c' as an alias because the minus symbol is interpreted as a continuation character:

```
SELECT a + b c FROM xx;
```

Therefore, SQL*Plus returns the following unexpected output:

```
         C
----------
        30
```

Server Manager, however, interprets this code as the following:

```
SELECT a + b - c FROM xx;
```

Therefore, Server Manager returns the following expected output:

```
A+B-C
----------
         0
```

Make sure you do not have a minus sign at the end of a line in your SQL scripts.

Ampersands

SQL*Plus interprets an ampersand (&) as a substitution variable, whereas Server Manager interprets an ampersand as a normal string. If the text following the ampersand does not have a defined value, then SQL*Plus interprets it as an undefined value and prompts the user for input, even if the ampersand is enclosed in a comment. Therefore, ampersands can cause unexpected output in SQL*Plus.

If you have SQL scripts that use ampersands as normal text strings, then you have two options:

- Use the SET ESCAPE command to place an escape character before each ampersand.
- Use the SET DEFINE OFF command to disable the recognition of substitution variables.

> **Note:** Do not use the `SET DEFINE OFF` command if you have other, valid substitution variables; if you do, then the other variables will not be recognized.

For example, the following SQL statement prompts the user for input in SQL*Plus:

```
CREATE TABLE "Employees & Managers" (
    Employees varchar(16),
    Managers varchar(16));

Enter value for managers:
```

Using the SET ESCAPE Command

To avoid the user prompt, you can use the `SET ESCAPE` command to set an escape character. Then, place the escape character before the ampersand. A backslash (\) is often used as an escape character.

To avoid the prompt in the preceding example by using the `SET ESCAPE` command, change the entry to the following:

```
SET ESCAPE \

CREATE TABLE "Employees \& Managers" (
    Employees varchar(16),
    Managers varchar(16));
```

Using the SET DEFINE OFF Command

To avoid the prompt in the preceding example by using the `SET DEFINE OFF` command, change the entry to the following:

```
SET DEFINE OFF

CREATE TABLE "Employees & Managers" (
    Employees varchar(16),
    Managers varchar(16));
```

CREATE TYPE and CREATE LIBRARY Commands

SQL*Plus treats the `CREATE TYPE` and `CREATE LIBRARY` commands as PL/SQL blocks. Therefore, in SQL*Plus, you must use a slash (/) on a separate line to end these commands, while Server Manager allows you to end these commands with a semicolon (;).

If you end any `CREATE TYPE` or `CREATE LIBRARY` command with a semicolon in your SQL scripts, then remove the semicolon and place a slash (/) on the next line. For example, the following SQL statements are not recognized by SQL*Plus:

```
CREATE OR REPLACE TYPE sys.dummy AS OBJECT (data CHAR(1));
CREATE OR REPLACE LIBRARY DBMS_SPACE_ADMIN_LIB TRUSTED AS STATIC;
```

Edit these statements in the following way before you run them with SQL*Plus:

```
CREATE OR REPLACE TYPE sys.aq$_dummy_t AS OBJECT (data CHAR(1))
/
CREATE OR REPLACE LIBRARY DBMS_SPACE_ADMIN_LIB TRUSTED AS STATIC
/
```

COMMIT Command

SQL*Plus requires that the COMMIT command be terminated either with a semicolon (;) or a slash (/), but Server Manager allows the COMMIT command with no terminator. Therefore, if you use the COMMIT command in your SQL scripts without a terminator, then edit these scripts to include a terminator.

For example, suppose you have the following COMMIT command in a SQL script:

```
commit
```

Include a terminator for the command, as shown in either of the following examples:

```
commit;

commit
/
```

Index

Numerics

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

Q

R

S

T

U

V

W