Oracle7™ Server Migration

Release 7.3

Part No. A32540–1
February, 1996

ORACLE®

Oracle7™ Server Migration, Release 7.3

Part No. A32540–1

Primary Author: Sanford A. Dreskin
Contributors: John Frazzini, Irene Hu, Bill Lee, Ravi Narayanan, Greg Pongracz, Toumas Pystynen, Brian Quigley, Hari Sankar, Tim Smith, Harry Sun, Peter Vasterd, Raghu Viswanatahan

This software was not developed for use in any nuclear, aviation, mass transit, medical, or other inherently dangerous applications. It is the customer's responsibility to take all appropriate measures to ensure the safe use of such applications if the programs are used for such purposes.

Oracle Corporation, 500 Oracle Parkway, Redwood City, CA 94065.

Preface

This manual is designed to aid database administrators and system programmers in the planning and implementation of migration, upgrade, and downgrade procedures. Detailed, procedural steps are provided to guide you through each of the various migration, upgrade, and downgrade operations. The use of tools and features involved with each operation is also explained.

The process of transforming one version of the Oracle database into a later version of the Oracle database is called *migration*. Thus, the transformation of an Oracle, Version 6 database into an Oracle, Version 7 database is described as "the *migration* of an Oracle, Version 6 database to an Oracle, Version 7 database". For specific information on migrating from Version 6 to any of the Version 7 releases, see Chapter 7 "Migrating from Version 6 to Version 7".

The process of transforming one release to another release of the same version of the Oracle database is called *upgrading* or *downgrading*. Thus, the process of transforming a Release 7.0 database into a Release 7.1 database is described as "*upgrading* Release 7.0 to Release 7.1". The reverse process, of transforming a Release 7.1 database into a Release 7.0 database, is described as "*downgrading* Release 7.1 into Release 7.0". For specific information on upgrading and downgrading between Releases 7.0, 7.1, 7.2, and 7.3, see Chapter 9 "Upgrading and Downgrading between Oracle7 Releases".

Audience

This manual is written for database administrators, application programmers, security administrators, and system operators who are responsible for planning and executing migration, upgrading, and downgrading operations.

Knowledge Assumed of the Reader

Readers of this manual are assumed to be familiar with either Version 5, Version 6, Release 7.0, Release 7.1 or Release 7.2 of the Oracle7 Server and with the operating system environment under which their particular Oracle version or release is running. Readers should also be familiar with the new features provided in each of the Oracle7 Server releases, as described in Appendices A, B, C, and D of this manual, and the Oracle7 Server README.DOC files for each of the Oracle7 releases.

How *Oracle7 Server Migration* Is Organized

This section contains the following topics:

- Part I: Generic Migration, Upgrading, and Downgrading Procedures
- Part II: Migrating, Upgrading, and Downgrading between Specific Versions and Releases
- Part III: Appendices A, B, C, D, E, and F

This manual is divided into the following parts and chapters:

Part I: Generic Migration, Upgrading, and Downgrading Procedures

Chapter 1: Migration Overview

This chapter describes migration procedures and the responsibilities of database administrators and application programmers.

Chapter 2: Preparing to Migrate

This chapter describes the steps to take before you begin migrating the database.

Chapter 3: The Migration Utility

This chapter describes the process of migrating your database using the Migration Utility.

Chapter 4: Migrating Using Export/Import

This chapter describes the process of migrating your database using the Export and Import utilities.

Chapter 5: After Migrating the Database

This chapter describes the steps to take after you have converted your database using either the Migration Utility or Export/Import.

Part II: Migrating, Upgrading, and Downgrading between Specific Versions and Releases

Chapter 6: System Requirements for Migration This chapter describes the system requirements that must be satisfied to ensure successful migration, upgrading, and downgrading operations.

Chapter 7: Migrating from Version 6 to Version 7

This chapter describes how to migrate an Oracle6 database to any of the Oracle7 releases using either the Migration Utility or Export/Import.

Chapter 8: Migrating Version 6 Applications

This chapter describes how to run Oracle6 applications against the various Oracle7 releases.

Chapter 9: Upgrading and Downgrading between Oracle7 Releases

This chapter describes how to upgrade and downgrade between the various Oracle7 releases.

Part III: Appendices A, B, C, D, E, and F

Appendix A: Summary of Changes in Oracle7, Release 7.0

This appendix describes the differences between Oracle Version 6 and Oracle Release 7.0.

Appendix B: Summary of Changes in Oracle7, Release 7.1

This appendix describes the differences between Oracle7, Release 7.0 and Oracle7, Release 7.1.

Appendix C: Summary of Changes in Oracle7, Release 7.2

This appendix describes the differences between Oracle7, Release 7.1 and Oracle7, Release 7.2.

Appendix D: Summary of Changes in Oracle7, Release 7.3

This appendix describes the differences between Oracle7, Release 7.2 and Oracle7, Release 7.3.

Appendix E: Migration Utility Errors and Messages

This appendix lists the errors you may see when using the Migration Utility. The probable cause and corrective action are given for each error.

Appendix F: Operating System–Specific Information

This appendix lists references in this manual to your operating system–specific documentation.

Conventions Used in this Manual

This section contains the following topics:

- Special Icons
- Text of the Manual
- Code Examples

Special Icons

Special icons alert you to particular information within the body of this manual:

Suggestion: The light bulb highlights suggestions and practical tips that could save time, make procedures easier, and so on.

Warning: The warning symbol highlights text that warns you of actions that could be particularly damaging or fatal to your operations.

OSDoc

Additional Information: The OSDoc icon refers you to the Oracle operating system–specific documentation for additional information.

Text of the Manual

The following sections describe the conventions used in the text of this manual.

UPPERCASE Characters

Uppercase text is used to call attention to command keywords, object names, parameters, filenames, and so on.

For example, “If you create a private rollback segment, the name must be included in the ROLLBACK_SEGMENTS parameter of the parameter file.”

Italicized Characters

Italicized words within text indicate the definition of a word, book titles, or emphasized words.

An example of a definition is the following: “A *database* is a collection of data to be treated as a unit. The general purpose of a database is to store and retrieve related information, as needed.”

An example of a reference to another book is the following: “For more information, see *Oracle7 Server Tuning*.”

An example of an emphasized word is the following: “You *must* back up your database regularly.”

Code Examples

SQL, Server Manager line mode, and SQL*Plus commands/statements appear separated from the text of paragraphs in a monospaced font. For example:

```
INSERT INTO emp (empno, ename) VALUES (1000, 'SMITH');
ALTER TABLESPACE users ADD DATAFILE 'users2.ora' SIZE 50K;
```

Example statements may include punctuation, such as commas or quotation marks. All punctuation in example statements is required. All example statements terminate with a semicolon (;). Depending on the application, a semicolon or other terminator may or may not be required to end a statement.

Uppercase words in example statements indicate the keywords within Oracle SQL. When issuing statements, however, keywords are not case sensitive.

Lowercase words in example statements indicate words supplied only for the context of the example. For example, lowercase words may indicate the name of a table, column, or file.

Your Comments Are Welcome

We value and appreciate your comments as an Oracle user and reader of the manuals. As we write, revise, and evaluate, your opinions are the most important input we receive. At the back of this manual is a Reader's Comment Form which we encourage you to use to tell us both what you like and what you dislike about this (or other) Oracle manuals. If the form has been used, or you would like to contact us in California, please use the following address.

Oracle7 Server Documentation Manager
Oracle Corporation
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Contents

PART

I

Generic Migration, Upgrading, and Downgrading Procedures

CHAPTER

1

Migration Overview

This chapter gives you an overview of the migration processes and procedures for migrating Oracle databases.

Oracle migration processes and procedures transform existing versions or releases of Oracle databases (including their applications) into different versions or releases. All Oracle7 Server releases are upwardly compatible with all earlier Oracle versions and releases. Therefore, databases transformed using the migration processes described in this book work in the same manner as in earlier versions and, optionally, permit the use of functionality available with the new release.

You must perform several preparatory steps in the migration process before you begin to migrate the data in your current, production database. Also, once you have migrated your current database, you should perform several additional steps that deal with testing and adding the functionality available with the new version or release.

The topics covered in this chapter are

- Migrating a Database from One Version to Another
- Overview of the Migration Process
- Role of the Database Administrator During Migration
- Role of the Application Developer During Migration
- Downgrading to a Previous Database
- Migrating from Single Instance to Parallel Server

For information about upgrading and downgrading between the Version 7 releases, see Chapter 9 “Upgrading and Downgrading between Oracle7 Releases”.

If you are migrating to Trusted Oracle, see the *Trusted Oracle7 Server Administrator’s Guide*.

Migrating a Database from One Version to Another

Careful planning and proper tools can greatly reduce the complexity of migrating a database. Oracle provides several tools, such as the Export/Import and Migration utilities, to make the migration process as simple as possible.

> **Note:** The Migration Utility can be used only to transform an earlier version to a later version of the Oracle database. *The Migration Utility cannot be used for reverse migration*. For example, the Migration Utility cannot be used to transform a Version 7 database into a Version 6 database.

The following Oracle features aid in the migration process:

- SQL*Net can be used with different versions and releases of Oracle. For example, Version 6, Release 7.0, Release 7.1, and Release7.2 databases can communicate with Release 7.3 using SQL*Net.
- The programming interface (for example, in the Oracle Precompilers and Oracle Call Interface) remains unchanged between different versions of Oracle. For example, you can use either SQL*Net or relink applications designed for a Version 6 database to run them with any of the Oracle7 releases.
- A backward compatibility mode is provided when there are small incompatibilities between the functionality of different Oracle releases. For more information about upgrading and downgrading between the Version 7 releases, see Chapter 9 “Upgrading and Downgrading between Oracle7 Releases”.
- Many new features and enhancements are automatically available after migration to a more recent version or release. Several of these features, such as shared SQL areas, reduced instruction costs, and the multi–threaded server architecture, should result in improved performance.

Overview of the Migration Process

This section contains the following topics:

- Step 1: Prepare to Migrate
- Step 2: Rehearse the Migration Process
- Step 3: Test Your Applications
- Step 4: Preserve the Source Database
- Step 5: Migrate the Source Database
- Step 6: Make Initial Adjustments to the New Production Database

Before attempting to migrate your database, you should understand the requirements and procedures involved with each step of the migration process. The following list is a step–by–step overview of the migration process. The details of each step are discussed in later chapters. This list is a guideline for approaching migration. Note that the migration steps presented are generic; they do not depend on a specific operating system.

Step 1: Prepare to Migrate

There are several preparatory steps that you are advised to follow. Although the following steps are not mandatory, they will ensure that the final migration proceeds smoothly:

- Become familiar with the features of the database to which you are migrating (the target database) and compare the new target database features with the features of the database to be migrated (the source database which, in most cases, is the current production database).
- Estimate the system resources that will be required for the successful migration of your database to the new version of Oracle. See Chapter 6 “System Requirements for Migration”.
- Decide on the migration method you need to use (the Migration Utility, Export/Import, or copying data using database links). Refer to the section “Choose a Migration Method” in Chapter 3 “Preparing to Migrate”.
- Develop a plan for testing the database after migrating to the target database. Perform the tests on the source database and record the results for later comparison with target database results.

Step 2: Rehearse the Migration Process

Rehearse the migration of your database with *a subset of the source database.* If you are using the migration utility, create a "test" version of the source database; then migrate the test database. At this point, *do not migrate the entire production (source) database.*

Step 3: Test Your Applications

Test your applications with the Release 7.x database. Be certain that your applications run correctly with Release 7.x before migrating your production database.

Step 4: Preserve the Source Database

There are a few, remaining preparation steps that will protect the contents of your source database and ensure a completely successful migration.

- Shut down the source database using normal shutdown procedures. Be certain that there are no uncommitted transactions and there is no outstanding redo information in the redo log files.
- Make a complete backup of the source database. Be certain to back up all datafiles, control files, and initialization (INIT.ORA) files, and any scripts that create objects in the source database. Although there should not be any outstanding redo information in the redo log files, you should back up these files as well, in case any datafiles in the source database are lost or unreadable. You should keep this backup until you are using the new target database as your production database.
- Delete or update any obsolete or changed parameters in your initialization (INIT.ORA) files.
- Update any SQL scripts that you currently use to create tables or clusters.

Step 5: Migrate the Source Database

Migrate the source database to the format of the target database using the migration method that you chose (such as Export/Import or the Migration Utility).

Attention: Both the Migration Utility and Export/Import methods require the installation of the target database during this step.

- Make a full, offline backup of the target database, including redo log files, after successfully opening the target database.
- If you have been working with a test database and are certain that your applications work with the target database, repeat Steps 3 through 5 using your source (production) database.

- If you have been working with your source database and are certain that your applications work with the target database, continue with Step 6.

Step 6: Make Initial Adjustments to the New Production Database

The basic migration steps have now been completed. However, there are several remaining steps that will ensure that the migrated database is ready to serve as your new production database.

- Test the newly created target database using the testing plan developed in Step 1. Compare the results with the results from the source database.
- Make tuning adjustments as needed, to ensure that the target database performance is as good as, or better than, that of the source database.
- Determine which target database new features are appropriate to use with your data and update your applications accordingly.
- Develop new database administration procedures as needed.

After completing this process, your database should be completely transformed into a new production database.

Role of the Database Administrator During Migration

Typically, the database administrator is responsible for ensuring the success of the migration process. Specific duties of the database administrator might include scheduling the migration process, making backups of the database to be migrated and the new database, and performing the actual migration procedure. The database administrator should arrange to meet with everyone involved in the migration process to define clearly everyone's roles during migration.

The database administrator typically becomes involved in each step of the process, except for steps that are concerned with testing applications in the target database and selecting new features of the target database, which are generally performed by application developers.

Role of the Application Developer During Migration

While the database administrator is responsible for migration of the database, the application developer is responsible for ensuring that applications designed for the source database work in the same way using the target database.

Before the migration process begins, the database administrator or application developer should install a target database for testing so that applications can be tested and modified, if necessary, to work with the same (or enhanced) functionality. Migration of production users to the target database should not begin until all applications have been tested and operate properly.

The application developer should note changes in the target database that may affect particular applications. Many of these changes are described in Appendices A, B, C, and D of this manual. *Oracle7 Parallel Server Concepts & Administration* and *Oracle7 Server SQL Reference* give additional descriptions of changes in Release 7.3.

Appendices A, B, C, and D of this manual also list changed data dictionary views upon which an application might depend. The application developer should use all of these sources to identify any modifications that need to be made in existing applications. In addition, Chapter 8 “Migrating Your Version 6 Applications”, describes the changes necessary to enable applications accessing a Version 6 database to access a Release 7.x database in a backwards compatibility mode, as well as how to upgrade these applications to take advantage of new Release 7.x functionality.

Downgrading to a Previous Database

You might want to return (downgrade) to a previous database after migrating to the target database. If you have not entered any new data in the target database, you can restore a complete backup of a previous database and open it again. Be certain to use a complete backup that contains the original initialization parameters that were used in the previous database. Once you have entered data in a target database, you cannot reverse the methods of migration described in this manual.

Warning: You must disable certain features of the target database before you return to a previous database. For more information about setting the COMPATIBLE parameter and other downgrading issues, see Chapter 9 “Upgrading and Downgrading between Oracle 7 Releases”

There are, however, the following methods of sending table data from a target database to a source database:

- You can use the SQL*Plus COPY command to copy the data from the target database tables into the tables in the earlier version or release database.
- You can create the table again in the earlier version or release database (using CREATE TABLE AS SELECT) by selecting the data in the target database table through a distributed query from the source database to the target database using a database link.
- You can use SQL*Plus to create non–Oracle text files from the target database and then use SQL*Loader to load this data back into a source database.

For more information on the COPY command, see the *SQL*Plus User's Guide and Reference* manual.

For more information on performing a Version 6 export of Release 7.x files, see *Oracle7 Server Utilities.*

For more information on the AS clause of the CREATE TABLE command, see *Oracle7 Server SQL Reference.*

For more information about upgrading and downgrading between the Version 7 releases, see Chapter 9 "Upgrading and Downgrading between Oracle7 Releases".

Migrating from Single Instance to Parallel Server

You may wish migrate from a single instance Oracle7 database to a multi–instance Oracle7 database which uses the parallel server option.

For more information about migrating from single instance to parallel server, see *Oracle7 Parallel Server Concepts & Administration*.

CHAPTER

2 Preparing to Migrate

This chapter describes all tasks that must be performed before you attempt to migrate your production database. This chapter also provides a detailed discussion of Steps 1 through 4 of the migration process (previously outlined in Chapter 1 “Migration Overview”.

The specific topics presented in this chapter are

- Step1: Prepare to Migrate
- Step 2: Rehearse the Migration Process
- Step3: Test Yor Applications
- Step 4: Preserve the Source Database

The information presented in this chapter is generic and, as such, applies to all Oracle versions and releases. For information on migration, upgrading, and downgrading procedures for specific Oracle versions and releases, see Chapter 7 “Migrating from Version 6 to Version 7”, Chapter 8 “Migrating Version 6 Applications”, and Chapter 9 “Upgrading and Downgrading between Oracle7 Releases”.

Step 1: Prepare to Migrate

This section contains the following topics:

- Become Familiar with the Features of the Target Database
- Estimate the System Requirements
- Choose a Migration Method
- Comparing Migration Methods
- Develop a Testing Plan

Become Familiar with the Features of the Target Database

Before you begin to plan the migration process, you should be familiar with the new features of the target database to which you wish to migrate. A good starting point for learning how a specific, Oracle7, Release 7.x database differs from Version 6 or another Oracle7 release are Appendices A, B, C, and D of this manual. These appendices list the changes in each of the Oracle7 releases and give specific references for more information about each new feature. If you are using the Parallel Server option, you should also see *Oracle7 Parallel Server Concepts & Administration* for changes in the Parallel Server.

You should also take an Oracle training class to learn how to take full advantage of the functionality available with the Oracle7 releases.

Estimate the System Requirements

Estimate the system resources that will be required for the successful migration of your database to the new version of Oracle. The configuration requirements for both the operating system and the hardware must be considered. See Chapter 6 “System Requirements for Migration”.

Choose a Migration Method

There are several methods for migrating or upgrading to a target database:

- You can use the Migration Utility.
- You can perform a full or partial database import into a target database from a full or partial database export of a source database.
- You can copy data from a source database to a target database using the COPY command or the AS clause of the CREATE TABLE command
- You can use the Oracle Installer for your specific platform, which has a migration option.

OSDoc

Additional Information: For more information on using the Oracle Installer method, see the operating system–specific Oracle documentation for your specific platform.

Each method is appropriate for different circumstances. The following sections describe each method, the amount of time and space required by each method, and the situations where each method is appropriate.

Warning: Whichever method you choose, you need to ensure that the character encoding scheme used for data in the target database is correctly specified. All character data in the target database is assumed to be in the specific character encoding scheme specified when the database was created with the CREATE DATABASE command. For more information on NLS, see *Oracle7 Server Reference.*

The Migration Utility

The Migration Utility is a program that converts some of the files and structures in your source database to target database format. The primary advantages of the Migration Utility are its speed and ease of use. The Migration Utility takes significantly less time than Export/Import and consists of a few easy steps. However, you cannot selectively migrate datafiles. Thus, the Migration Utility is useful if you want to migrate an entire source database to a target database quickly.

The Migration Utility does not require a significant amount of temporary space. The utility requires only that you have enough extra room in the SYSTEM tablespace to hold the data dictionaries of both the source and target databases simultaneously.

When you use the Migration Utility, the entire database is converted, including database files, rollback segments, and the control file. At any point before actually migrating your source database (Step 5), you can still open and access data with the source database. However, once you have performed Step 5 and migrated the source database to the target database with the Migration Utility, you can only go back to the source database by restoring a full backup of the source database.

If you decide to use the Migration Utility as the method for migrating your database, see Chapter 4 “The Migration Utility” for detailed information about the Migration Utility and its use.

Export/Import

You must create the target database before using the Export/Import method of migration.

The Export utility copies the data in your source database to an export file, from which the Import utility can load the data into a target database. An important distinction between Export/Import and the Migration Utility is that the physical data in your database is copied to a new location with Export/Import, while the Migration Utility changes only the file headers and the definitions of the data in the files where they currently reside.

The Export/Import method of migration provides some additional advantages.

- You can defragment the data. A full database import can compact the data and improve performance.
- You can restructure your database; that is, you can create new tablespaces, or modify existing tables or tablespaces.
- You can migrate only certain database objects or users. You can selectively import only objects and users that need to be imported at a given time.

Since the Export/Import method of migration does not change the source database, it is available at any point during the migration process. This allows you to keep a source database running in parallel with the target database without requiring the restoration of a backup. (However, you should not change the source database unless you make exactly the same changes to the target database.) Also, a full export can serve as an archive of your source database.

Export/Import Limitations

The Export/Import method has the following limitations:

- For a large database, a full database Export/Import can take a significant amount of time.

OSDoc

Additional Information: The time required for Export/Import migration depends on the characteristics of your operating system. For more information, see your operating system–specific Oracle documentation.

- For a large database, a full database Export/Import can require a substantial amount of temporary storage space for the data.
- Because you should not make changes to the source database after performing the export, your applications are unavailable until the migration is completed.

If you decide to use Export/Import as the method for migrating your database, see Chapter 5 “Migrating Using Export/Import” for detailed information about using Export/Import.

Copying Data

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

The copy data method of migration has the same advantages as Export/Import, which are:

- You want to defragment your data files.
- You want to restructure your database; that is, you want to create new tablespaces, or modify existing tables or tablespaces.
- You want to migrate only certain database objects or users, or even a subset of rows.

One reason to use the Copying Data method of migration, and not Export/Import, is to allow the selection of only certain rows of tables that are to be placed into your target database. The Export/Import utilities can only load entire tables and not selected rows. For example, if you wanted to create a new EMP table that contains a subset of the data in your existing EMP table (for example, employees in departments 10, 20 and 30), you would use the following SQL statement in Oracle7, Release 7.x:

```
CREATE TABLE NEW_EMP
   (EMPNO         NUMBER(4),
   ENAME          VARCHAR2(10),
   JOB            VARCHAR2(9),
   MGR            NUMBER(4),
   HIREDATEDATE,
   SAL            NUMBER(7,2),
   COMM           NUMBER(7,2),
   DEPTNO         NUMBER(2)) AS SELECT EMPNO, ENAME, JOB, MGR,
                  HIREDATE, SAL, COMM, DEPTNO
   FROM EMP@V6DB  WHERE DEPTNO IN (10, 20, 30);
```

The Copying Data method also requires less space than the other migration methods. Therefore, use the Copying Data migration method if you are migrating only a portion of your database. You do not need to allocate large amounts of extra space for temporary files or for Export/Import files. You simply need your source database and the target database online.

The COPY command is also useful if you are working with large cluster tables. The SQL*Plus COPY command allows you to move different portions of the cluster in parallel using SQL*Net.

For more information about copying data from one database to another, refer to the CREATE TABLE command in *Oracle7 Server SQL Reference* and the COPY command in the *SQL*Plus User's Guide and Reference.*

Comparing Migration Methods

The following table summarizes the advantages and disadvantages of the four migration methods:

Migration Method	Advantages	Disadvantages
Migration Utility	Automatic and requires minimal interaction by the DBA. Relatively fast no matter what the size of the database because the data dictionary objects are the only objects that are changed.	Can only be used for forward, version–to–versionmigrations. Cannot be used for reverse migration between versions of the Oracle Server. For example, cannot be used for reverse migration from Oracle7 to Oracle6. Cannot be used for release–to–releasemigration. Cannot be used for migration to Trusted Oracle7.
Export/Import	Ability to migrate specific segments of a database. Can be used for reverse migration between versions of the Oracle Server, for example, for reverse migration from Oracle7 to Oracle6. Can be used for release–to–release migration in upgrade/downgrade operations. Migrated data can be compacted for improved performance. Database can be restructured with modified or new tablespaces.	May be slow, depending on size of the database. For large databases, several Gb in size, may take several hours. Requires large amounts of disk space for export files and offline backup.
Copying Data	Datafiles can be defragmented. Database can be restructured with modified or new tablespaces. Ability to migrate specific segments of a database. Can be used for release–to–releasemigration.	Same as for Export/Import.

Table 2 – 1 Comparing Migration Methods

Develop a Testing Plan

A carefully designed series of tests, that will validate all stages of the migration process, will ensure the success of the migration operation. The importance of such a test program should not be underestimated.

Testing Before Migration

A rigorous testing program, performed before moving your production system to a target database, will ensure that the final migration process will be well understood and predictable. *Failure to perform a rigorous testing program is risky and may lead to unpredictable results.* Therefore, as much testing as possible should be completed before migrating to a target database. Such preparation testing must include the following types of tests: migration, minimal, functionality, integration, performance, and stress.

Migration Testing

Migration Testing involves planning and testing the migration path from the source database to the new, target database. You may choose to perform a full database export under the source database and then perform a full database import to move your data to the target database. You may also choose to use the Migration Utility. These methods are discussed in Chapter 4 “The Migration Utility” and Chapter 5 “Migrating Using Export/Import”.

Regardless of what migration method you choose, *you must establish, test, and validate a migration plan.*

Minimal Testing

Minimal testing involves moving all, or portions, of your application on the source database to the target database and running the application without enabling any new, target database features. Minimal testing is a very limited type of testing that does not reveal potential issues you may encounter under a production environment. However, any application startup or invocation problems will be revealed immediately.

Functional Testing

Functional testing is a set of tests in which new and existing functionality of the system are tested after migrating. Functional testing includes all components of the RDBMS system, networking, and application components. The objective of functional testing is to determine if each component of the system functions as it did before migrating.

Integration Testing

Integration testing tests the interaction of each component of the system.

- Pro*C/C++ applications running against the target database instance should be tested to ensure that there are no problems with the new software.
- GUI interfaces should be tested with other components.
- Subtle changes in the target database, such as datatypes, data in the data dictionary (additional rows in the data dictionary, object type changes, and so forth) can have an effect all the way up to the front–end application, regardless of whether the application is directly connected to the Oracle7 instance or not.
- If the connection between two components involves SQL*Net, that connectivity should also be tested as well as stress tested.

Performance Testing

Performance testing of a target database compares the performance of various SQL statements in the target database with the statements' performance in the source database. Before migrating to the target database, you should understand the performance profile of your application under the source database. Specifically, you should understand the calls the application makes to the database kernel.

Suggestion: To thoroughly understand your application's performance profile under the source database, enable SQL_TRACE and profile with TKPROF. For more information, see *Oracle7 Server Tuning*.

Volume/Load Stress Testing

Volume and load stress testing tests the entire, newly migrated 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. Benchmarks that do not characterize business applications are, instead, relied upon heavily. Benchmarks of the application should be conducted to uncover unknown problems relating to functionality, performance and integration.

Once the source database has been successfully migrated to the target database, you should test the data to ensure that all data is accessible and that your applications function properly. You should also determine if any database tuning is necessary. If possible, you should automate these testing procedures.

Your testing plan should be representative of the work performed at your site. You should test the functionality and performance of all of your applications on your source production databases. Gather performance statistics for both normal and peak usage.

Specific Pre– and Post–Migration Tests

The following tests can assist you in tuning your SQL statements after you migrate to the target database:

- Gather performance statistics for each of your SQL statements. You should be able to write scripts to extract these statements from your applications and run them against your production database.
- Use EXPLAIN PLAN to determine the execution plan Oracle follows to execute each SQL statement. Use the INTO parameter to save this information in a table.

After migrating, you can compare performance of the target database with your source database test results by completing the following steps:

1. Use EXPLAIN PLAN to determine the execution plan for each statement. Save this information in a new table.
2. Compare the target database execution plans to the source database execution plans. If you notice a difference, you should execute the command in the target database and compare this performance to the performance of the source database.

Step 2: Rehearse the Migration Process

You should perform a test migration (or test upgrade) before beginning the actual migration or upgrade procedure so that you can resolve any potential problems before actually migrating or upgrading your production database. For example, if you are migrating a Version 6 database and you plan on using Export/Import, you can use your actual Version 6 database.

If you plan on using the Migration Utility, you need to create a test version of the source database on which to perform the test migration.

OSDoc

Additional Information: Refer to your operating system–specific Oracle documentation for information on how to configure another database so that no operating system variables defined for your production database are affected by the test database.

To rehearse the migration of the database, perform Steps 2 through 4 of the migration process described in Chapter 1 “Migration Overview” and later, in this chapter, using either a test database or a subset of the source database, depending upon which migration method you choose.

Step 3: Test Your Applications

After migrating a test database, you should use the test database to ensure that your existing applications operate properly with the target database. You can also begin enhancing source database applications by adding target database functionality to the applications. However, it is suggested that you first make sure that the applications operate in the same manner as they did in the source database. For more information on using your applications with Oracle7, Release 7.x, see Chapter 8 “Migrating Version 6 Applications”.

Step 4: Preserve the Source Database

There are several tasks to perform before starting the migration procedure. The following list summarizes the steps to take before implementing any of the migration procedures:

- Shut down the source (original production) database.
- Make a complete backup of the source database.
- Update your initialization (INIT.ORA) file.
- Change any SQL scripts that you currently use to create objects.
- Update datatypes.
- Update database administration scripts.
- Update database link names.
- Move constraint identifiers.
- Change unique indexes to UNIQUE or PRIMARY KEY integrity constraints.

Shut Down the Source (original production) Database

The source (original production) database must be shut down normally so that there is no outstanding redo information, and no uncommitted transactions.

If you migrate using the Migration Utility, all source datafiles that are online when the target database is opened are automatically converted to the target database file format. Files that are offline when the target database is opened remain in the source database file format.

You do not need to convert all of the database files to target database format immediately. The remaining files are converted when they are brought online in the target database.

Rollback segments are converted as they are accessed by the target database. Thus, all rollback segments that are in tablespaces that are online when the database is first opened in the target database are converted. If a source database rollback segment is in a tablespace that is offline when the target database is opened, it is converted the first time it is brought online in the target database.

Back up the Source (original production) Database

After shutting down the database, you should make a full backup of the source database before proceeding with migration. Be certain to back up all datafiles, control files, parameter files, online redo log files, and any script files used to create objects in the source database.

Suggestion: You should make a backup of the online redo log files even though there should be no outstanding redo information. This allows you to easily restore your source database if necessary.

Change Parameter Files (INIT.ORA Files)

Certain initialization parameters are obsolete in new Oracle versions and releases. Obsolete parameters may cause errors if used with a new Oracle version or release database. You must remove all obsolete parameters from any parameter file that starts a new Oracle version or release instance. You must also alter any parameter whose syntax has changed in a new Oracle version or release. Refer to Appendices A, B, C, and D for lists of obsolete and changed parameters for Release 7.0, Release 7.1, Release 7.2, and Release 7.3, respectively.

Change SQL Scripts

All SQL scripts that you built that created objects in the source database should be changed to conform to the syntax of the Oracle version to which you will migrate.

Update Datatypes

Datatypes change between Oracle versions and releases. Therefore, you must make appropriate changes to your applications before attempting to use them with a new Oracle version or release. Refer to Appendices

A, B, C, and D for lists of changed datatypes in Release 7.0, Release 7.1, Release 7.2, and Release 7.3, respectively. Also, see Chapter 7 "Migrating from Version 6 to Version 7" for specific information on datatype changes between Version 6 and Version 7.

Update Database Administration Scripts

Use the CREATE USER and GRANT CREATE SESSION commands to update database administration scripts. Refer to Appendices A, B, C, and D for lists of changed database administration scripts in Release 7.0, Release 7.1, Release 7.2, and Release 7.3, respectively. Also, see Chapter 7 "Migrating from Version 6 to Version 7" for specific information on changes to database administration scripts between Version 6 and Version 7.

Update Database Link Names

The naming convention for database links may change when you migrate, upgrade, or downgrade between Oracle versions and releases. See Chapter 7 "Migrating from Version 6 to Version 7" for specific information on updating database link names.

Move Constraint Identifiers

The constraint clause for the CREATE TABLE command has new syntax in the Oracle7 releases. See Chapter 7 "Migrating from Version 6 to Version 7" for specific information on constraint identifiers.

Change Unique Indexes to UNIQUE or PRIMARY KEY Integrity Constraints

Integrity constraints can be used in Oracle7 releases to enforce uniqueness among column values. Because unique indexes might not be supported in future versions of Oracle, you should begin using UNIQUE or PRIMARY KEY integrity constraints instead of unique indexes.

Indexes can now be validated using the ANALYZE command. You should begin using this command, because the VALIDATE INDEX command might not be supported by future versions of Oracle. See Chapter 7 "Migrating from Version 6 to Version 7" for more information on unique indexes and Primary Key integrity constraints.

CHAPTER

3 The Migration Utility

This chapter guides you through the migration procedure using the Migration Utility with a step–by–step approach. In particular, this chapter describes the use of Step 5 “Migrate the Source Database”.

Topics covered in this chapter include the following:

- Overview of the Migration Utility
- Blocksize Considerations
- Migrating to a Different Computer Architecture
- Errors During Migration
- Step 5: Migrate the Source Database (using the Migration Utility)

Overview of the Migration Utility

The Migration Utility is intended only for the migration of earlier to later versions of the Oracle database, for example, from Version 6 to any of the Version 7 releases; it is *not* intended for release–to–release upgrades such as Release 7.1 to Release 7.2.

> **Note:** The Migration Utility cannot be used for reverse migration such as downgrading from Release 7.1 to Version 6.

Structures can be converted in the following ways:

- Some structures are converted by the Migration Utility.
- Some structures are converted by the ALTER DATABASE command.
- Some structures are converted automatically by the target database itself.
- Some file structures do not need to be converted at all.

The Migration Utility also outputs a binary file that is used to build a new target database control file. The target database control file is created when execution of the ALTER DATABASE CONVERT command is completed.

The Migration Utility uses the character set specified by the LANGUAGE parameter in the initialization file (the INIT.ORA file) of the source database as the character set for all target databases. Since this cannot be changed after migration is complete, you must ensure that LANGUAGE specifies the correct character set before running the Migration Utility.

To run the Migration Utility, you need DBA privileges. You do not interact with the Migration Utility. Once the utility is invoked, it performs the following migration activities automatically:

- creates the database user, MIGRATE, as a temporary owner of the target database data dictionary table. If you already have a user named MIGRATE in your database, you should export this user's objects before migration, then import the objects after migration.
- runs the conversion script MIGRATE.BSQ that creates the target database data dictionary base tables
- creates a new bootstrap segment for use by the target database
- creates CONVERT.ORA, which creates the new control file for the target database (the name of this file may vary on some operating systems)

The Migration Utility can be run multiple times and does not prevent you from returning to the source database. However, if the Migration Utility has been run, you must recreate the source database catalog views (that were dropped automatically by the Migration Utility) before returning to the source database.

Warning: You cannot return to the source database after the ALTER DATABASE CONVERT command has been run. If you wish to return the source database after the ALTER DATABASE CONVERT command has been run, you must restore the source database.

Blocksize Considerations

Before executing the STARTUP NOMOUNT command under the target database, check the value of the DB_BLOCK_SIZE parameter in the INIT.ORA file. Change the value of DB_BLOCK_SIZE, if necessary, to match the blocksize of the source database.

Warning: The migrated database will not perform correctly if the value of DB_BLOCK_SIZE of the migrated database does not match the blocksize of the source database *exactly*.

Migrating to a Different Computer Architecture

The Migration Utility cannot migrate a database to a computer system that uses a different architecture. For example, the Migration Utility will not successfully migrate a database from a 16–bit platform to a 32–bit platform. The only migration procedure that will work correctly in such a situation is the Export/Import method.

Errors During Migration

During the migration procedure, you may encounter errors. Errors can be due to performing migration steps out of order, not performing certain migration steps, not fulfilling the prerequisites for migration, or encountering certain conversion irregularities.

For more information about specific errors encountered during the migration process and the action to take for each error, see Appendix E.

Step 5: Migrate the Source Database (using the Migration Utility)

Before proceeding, you should review the migration steps outlined in Chapter 1 “Migration Overview”. The following steps expand on Step 5 (see page 1 – 5) of the migration procedure and explain how to migrate your source database using the Migration Utility.

1. Install and run the Migration Utility.

OSDoc

Additional Information: This task is operating system specific. Refer to your operating system–specific Oracle documentation.

The following parameters can be passed to the Migration Utility:

ALLOW_ OFFLINE	when TRUE, allows migration of release 6.0.34.3 databases and earlier with offline tablespaces. Be certain that these tablespaces were taken offline cleanly before using this option; see Chapter 3 “Preparing to Migrate” for details.
CFILE	specifies the filename of control file (if not using the expected default).
CHECK_ ONLY	when TRUE, performs space calculations without performing migration.
CNVFILE	specifies CONVERT.ORA filename (if not using the expected default).
DB_NAME	specifies the name of the database to migrate.
DUMPCF	a diagnostic tool (primarily for Oracle Worldwide Customer Support).
ECHO	when TRUE, echoes the commands issued by the MIGRATE.BSQ script.
MIGFILE	specifies MIGRATE.BSQ filename (if not using the expected default).
NEW_ DATABASE	specifies a new name for your migrated database. **Note**: Some databases use the default name of “DEFAULT”. A different, more meaningful name should be chosen.
NO_SPACE_ CHECK	when TRUE, does not perform space check before migration.
PFILE	specifies the name of the parameter file (if not using the expected default, INIT.ORA).
SPOOL	specifies filename of file to which to spool output.

Warning: The Migration Utility creates a binary file that creates the control file. The name of this file is operating system dependent. *Do not alter this file in any way.*

2. Install the target database.

OSDoc

Additional Information: Installation of the target database is operating system specific. Refer to your operating system–specific Oracle documentation. Read the README.DOC file included with your target database installation for any late additions or modifications to the product.

3. Issue the following command to connect to the target database instance:

```
SVRMGR> CONNECT INTERNAL
```

   Note: You must be connected as INTERNAL to perform the rest of the migration.

4. Start an instance for the target database, but do not mount the target database. Issue the following command:

```
SVRMGR> STARTUP NOMOUNT
```

5. Issue the following command:

```
SVRMGR> ALTER DATABASE CONVERT;
```

   This command converts the file headers and builds a new target database control file. After this process is complete, the data dictionary tables in the source database no longer exist in the target database.

6. Open the target database by issuing the following command:

```
SVRMGR> ALTER DATABASE OPEN RESETLOGS;
```

   All datafiles and rollback segments that are online when the target database is opened are converted to the new format of the target database. Refer to Step 1 “Prepare to Migrate” and the comments on shutting down the source (original production) in Chapter 1 “Migration Overview” for a description of database file and rollback segment conversion.

7. Once you open the target database and start an instance, you can drop the MIGRATE user created temporarily to hold the data dictionary. Issue the following statement to drop this user and reclaim the space allocated to it:

```
DROP USER migrate CASCADE;
```

Objects in this user's schema are no longer needed once the conversion is complete. You can also delete the binary file that creates the target database data dictionary base tables.

8. Run the CATALOG.SQL script for your new target database to create the data dictionary views on the target database base tables.

   ```
   SVRMGR> @catalog
   ```

   Run the target database CATPROC.SQL script which will run all of the scripts required for, or used within, PL/SQL.

   ```
   SVRMGR> @catproc
   ```

   If you wish to create additional data dictionary structures, see *Oracle7 Server Reference* for a complete list and description of available scripts.

   If you purchased any Oracle options, such as the distributed option, the parallel server option, or the advanced replication option, you must also run the appropriate script to install each option. These scripts are described in *Oracle7 Server Reference.*

CHAPTER

4

Migrating Using Export/Import

This chapter guides you through the migration procedure using the Export/Import method. If you are exporting from Version 5, be certain to read the "Notes for Version 5" section on page 4 – 5.

The Export/Import method can also be used to upgrade and downgrade releases of the same version. For example, the transformation of a Release 7.1 database into a Release 7.2 database can be accomplished using the Export/Import method. For more information on upgrading and downgrading, see Chapter 9 "Upgrading and Downgrading between Oracle7 Releases".

Topics covered in this chapter include

- Export/Import
- Data Definition Conversion
- Space Requirements for Export/Import
- Time Requirements for Export/Import
- Step 5: Migrate the Source Database (using Export/Import)

If you are exporting from, or importing to, a Trusted Oracle database, there are additional features of which you must be aware. For more information about Export/Import and migration, see the *Trusted Oracle7 Server Administrator's Guide*.

Export/Import

The Export/Import method of migrating a database involves the following two steps:

- Export the data from the database that you wish to migrate (the source database).
- Import the exported data into the database to which you wish to migrate (the target database).

For example, if you wish to migrate from a Version 6 database to a Version 7 database, you would first export the desired data from the Version 6 database. Then you would import the exported data into the Version 7 database.

To use the Export/Import method, the Oracle Corporation recommends that you use the version of the Export utility shipped with the version or release of the *source* database. Once you have exported the desired data, you must then use the version of the Import utility shipped with the version or release of the *target* database. You should use the Export and Import utilities for migration only after you have carefully read Part I of *Oracle7 Server Utilities.*

The Export/Import method of migration allows you to selectively choose tables and users to migrate. A possibly advantageous side effect of using the Export/Import method of migration is that data is automatically compressed.

Data Definition Conversion

When importing data from one version to another version, the Import utility makes changes to data definitions as it reads in the export file. For more information on Version 6 to Version 7 data definitions, see "Data Definition Conversion" in Chapter 7 "Migrating from Version 6 to Version 7".

Space Requirements for Export/Import

The amount of space required for an export depends upon the amount of data you are exporting. Examine the views USER_SEGMENTS or DBA_SEGMENTS to determine the amount of space occupied by the data. These views give you the number of segments allocated, but keep in mind that some segments can be allocated but unused. Refer to the *Oracle7 Server Administrator's Guide* for more information on estimating space usage.

Time Requirements for Export/Import

The Export/Import method of migration may require several hours. Therefore, you may need to schedule your migration during non–peak, production hours. The time required to complete a migration will, of course, increase for databases that contain large amounts of data or a large number of indexes.

OSDoc

Additional Information: The time required for Export/Import migration depends on the characteristics of your operating system. For more information, see your operating system–specific Oracle documentation.

Step 5: Migrate the Source Database (using Export/Import)

This section contains the following reference to Oracle, Version 5:

- Notes for Version 5

The following Export/Import steps are completely general and can be applied to any Oracle version or release.

> **Note:** Once an Export/Import migration has started, the database being migrated is unavailable for all production tasks.

To migrate an Oracle database using Export/Import, perform the following steps:

1. Export all desired objects from the original database *using the Export utility shipped with the original database.*
2. Install the target database. A default database may be created in this step. If so, you can proceed to Step 4; otherwise, continue with Step 3.

OSDoc

Additional Information: Installation of a target database is operating system specific. See your operating system–specific Oracle documentation. Read the README.DOC file included with your target database installation for any late additions or modifications to the product.

3. Create the target database. All issues involved in database creation are covered thoroughly in the *Oracle7 Server Administrator's Guide*.

4. Open the target database and start an instance. From the Server Manager prompt, issue the following commands:

```
SVRMGR> CONNECT INTERNAL
SVRMGR> STARTUP
```

5. You should pre–create tables, tablespace, and users in the target database as necessary, for example, to improve space usage by changing storage parameters. When you pre–create tables using SQL*Plus or Server Manager, you must either run in the original database compatibility mode or specifically make allowances for the data definition conversions outlined on the previous page that occur during import.

6. Import the data and tables from the original database export using the Import utility shipped with target database. Refer to *Oracle7 Server Utilities* for a complete description of the Import utility.

7. After migrating, you may discover that your tables were not imported properly. Views and synonyms may not be created in the correct order when dependencies exist (for example, when a view is based on a synonym). You may have to perform one of the following procedures to finish importing correctly:

   – If all of the rows for all tables were not successfully imported, repeat the import until it completes successfully. Be certain that IGNORE=Y and ROWS=N. IGNORE=Y causes Import to overlook "object already exists" errors and ROWS=N indicates that you do not want to import the rows of table data.

   – If some tables were imported successfully, while others were not even created, repeat the import with IGNORE=N and ROWS=Y to ignore the "object already exists" errors and re–import the rows.

   – If the tables had some, but not all rows imported, drop the incomplete tables and repeat the previous step.

For more information on the use of the Export and Import utilities, see *Oracle7 Server Utilities*.

Notes for Version 5

If you are currently using Version 5 of Oracle, it is recommended that you upgrade first to Version 6. If you choose to export your Version 5 files and import them directly to an Oracle7 release, you should be aware of the following.

- Column level grants are not imported and must be regranted after import.
- The EXPVEW and EXPTAB views are exported from Version 5, even though you run the DROPCAT5.SQL script prior to export. These views are not created when you perform the import to an Oracle7 release. Ignore the resulting error messages.

CHAPTER

5

After Migrating the Database

This chapter discusses the steps to perform after you have completed migrating from one version to another version or upgrading from one release to another release using either the Migration Utility or the Export/Import method of migration. This chapter elaborates on Step 6 of the migration process, discussed in Chapter 1 “Migration Overview”.

Topics in this chapter include the following:

- Back Up the Target Database
- Migrate Your Production Database
- Test the Database and Compare Results
- Tune the Target Database
- Add New Features as Appropriate
- Develop New Administrative Procedures as Needed

Back Up the Target Database

This section contains the following topics:

- Questions You Should Ask
- Offline Backup
- Online Backup

Questions You Should Ask

The ultimate success of any migration is strongly dependent upon the design and execution of an appropriate backup strategy. You must give careful attention to answering questions such as

- How long can the production database be inoperable before intolerable business consequences would result?
- How is the data in the database derived; for example, from input batch jobs, user input, or formulated by other sources?
- How flexible must my strategy be to accommodate change?
- Should backups be archived in an offsite, safe location?

Answers to these and other appropriate questions will help you formulate a robust backup strategy.

After completing the migration or upgrade of your database, you should make a complete backup of the new database. Make sure to back up all datafiles, control files, online redo log files, parameter files, and SQL scripts that create objects in the new database.

There are two types of backups that you may use. The first is the offline backup. The second is the online backup.

Offline Backup

Offline backups require a complete shutdown of the new (target) database, which may be intolerable for environments that must remain constantly active, that is, seven days a week and 24 hours each day. Offline backups are, therefore, commonly performed during time periods when system utilization is at a minimum.

Online Backup

Online backups can be conducted while the database instance is running and active. However, the database *must* be running in ARCHIVLOG mode and the DBA must use the ALTER_TABLESPACE <*tablespace_name*> BEGIN / END BACKUP command. See the *Oracle7 Server Administrator's Guide* for more information on online backups.

The online backup allows considerable flexibility in the backup strategy. Some of the major benefits are

- Not all datafiles must be backed up at the same time.
- Portions of a database (for example, selected tablespaces) can be backed up during off–peak hours.

Issue the following command when a tablespace is backed up:

```
ALTER SYSTEM SWITCH LOGFILE
```

which forces a log switch, thereby archiving off the log containing the end–backup marker in the last archived redo log.

Migrate Your Production Database

You should not migrate or upgrade your production database until you have successfully migrated a sample database and tested your applications to ensure that they work properly with the Release 7.x database. For more information on upgrading your applications, see Chapter 8 “Migrating Version 6 Applications” and Chapter 9 “Upgrading and Downgrading between Oracle7 Releases”.

Test the Database and Compare Results

Test the Release 7.x database using the testing plan you developed in Step 1 of the migration procedure, described in Chapter 1 “Migration Overview”. Compare the results of the test with the results obtained with the original database and make certain the same results are achieved.

Generally, performance with any Release 7.x database should be as good as, or better than, performance with the source database. If you notice any decline in database performance with Release 7.x, you should check the initialization parameters to ensure that they are set properly. For additional information on tuning the database, see *Oracle7 Server Tuning*.

You might want to run your source database and the new (target) database in parallel for a time, until you are certain that your new database is correctly configured and functioning properly. Any updates would need to be made to both the original and the new database. Depending upon how your updates are performed, you might be able to automate this process by editing your scripts to modify both versions of the database.

Tune the Target Database

Most methods used to tune a source database and related applications have either the same effect or are unnecessary for a target database. Any actions you took to tune your source database and applications should not impair the performance of a target database.

Add New Features as Appropriate

Appendices A, B, C, and D of this manual describe many of the new features available in Oracle7 releases. You should determine which of these new features can benefit your application and develop a plan for incorporating them. Target databases offer many new features that can affect not only your database design, but your application design as well. You do not, however, have to make any immediate changes to begin using your target database. You may prefer to introduce these enhancements into your database and corresponding applications gradually.

Chapter 8 “Migrating Version 6 Applications” describes how you can enhance your applications to begin taking advantage of these new features. You should already have tested your applications and successfully run them with the target database in a special V6 compatibility mode.

Develop New Administrative Procedures as Needed

After familiarizing yourself with the new target database features, you should review your database administration scripts and procedures to determine if any changes are necessary.

Completion of the final migration steps described in this chapter will allow you to begin taking advantage of new functionality provided by the target database. You need to coordinate your changes to the database with the changes that need to be made to each application. For example, by enabling integrity constraints in the database, you may be able to remove some of this data checking from your applications.

PART

II

Migrating, Upgrading, and Downgrading between Specific Versions and Releases

CHAPTER

6

System Requirements for Migration

This chapter discusses the system requirements that must be satisfied to ensure successful migration, upgrading, and downgrading operations. When migrating from an early version, such as Oracle Version 6, to a Release 7.x database, it is necessary to consider the configuration requirements for both the operating system and hardware.

The topics presented in this chapter are

- General Memory Requirements
- Shared Global Area (SGA)
- Shared Pool Area (SPA)
- The Buffer Cache
- The Redo Log Buffer Cache
- The Program Global Area (PGA)
- Oracle Executables
- Concurrent Access
- Disk Capacity and I/O
- Block Size
- CPU Resources
- Oracle7 Parallel Server
- Distributed Transactions and Network Considerations

General Memory Requirements

Oracle7 requires at least 16 megabytes of RAM for a minimal configuration. However, optimum use of Oracle7, in conjunction with the entire Oracle product suite of Developer/2000 tools, network utilities, and special applications, requires 32 megabytes or more. (Minimum recommended RAM for an Oracle Version 6, minimum configuration was 16 megabytes.) Your memory requirements will be based on the following factors:

- total size of the Shared Global Area (SGA)
- average functional size of the Program Global Area (PGA)
- client/server configuration
- average number of concurrent connections
- multi–threaded shared (MTS) configuration
- cached backups (supplied by vendors such as Sun, HP, etc.)
- applications

OSDoc

Additional Information: See your operating system–specific Oracle documentation for more information on memory requirements specific to your operating system.

The Shared Global Area (SGA)

The total size of the SGA is dependent upon several factors that are controlled within the initialization file (also known as the INIT.ORA file on most platforms). In Oracle7 (Release 7.0 and higher), the system parameters having the most impact on the total size of the SGA are

- DB_BLOCK_BUFFERS
- DB_BLOCK_SIZE
- LOG_BUFFER
- SHARED_POOL_SIZE

Figure 6 – 1 below is a simplified block diagram of the Oracle7 SGA.

OSDoc

Additional Information: On UNIX platforms, the Oracle Corporation recommends that the entire SGA be placed in one, shared memory segment. See your operating system–specific Oracle documentation for more information.

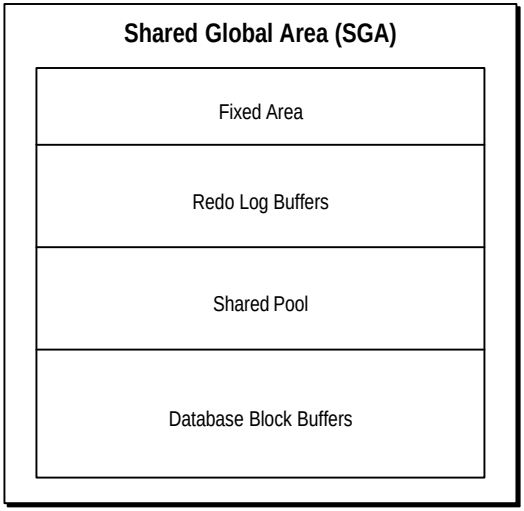

Figure 6 – 1 Shared Global Area

The Shared Pool Area (SPA)

The Shared Pool Area (SPA) is an enhancement that was introduced in the Oracle7 releases; it is governed by the parameter SHARED_POOL_SIZE. The SGA of an Oracle7 database requires additional memory to accommodate the SPA. The default shared pool size is approximately 3.5 megabytes. It is not recommended that the shared pool size be reduced during an Oracle7 installation.

The Buffer Cache

The DB_BLOCK_BUFFERS parameter specifies the number of database buffers comprising the buffer cache of an Oracle7 instance. The size of each buffer is defined by the system parameter DB_BLOCK_SIZE. Thus, the size of the buffer cache of the SGA is

```
buffer cache size = (DB_BLOCK_BUFFERS * DB_BLOCK_SIZE)
```

For example, suppose that DB_BLOCK_SIZE is 4096 and DB_BLOCK_BUFFERS is set to 5000. Then,

```
buffer cache size = (4096 * 5000) = 20,480,000 bytes
```

The value for DB_BLOCK_SIZE is set at the time of database creation and remains at that value for the life of the database. *You cannot change the value of DB_BLOCK_SIZE once the database has been created.* Note that DB_BLOCK_SIZE, DB_BLOCK_BUFFERS, and the above calculations are the same as they were for Oracle Version 6. The recommended minimum DB_BLOCK_SIZE should be equal to, or greater than, 4096 bytes. See page 6 – 10 for more information on block size.

The Redo Log Buffer Cache

The LOG_BUFFER system parameter specifies the size (in bytes) of the redo log buffer. The size of the redo log buffer is typically small; the information in this buffer is not scanned for reuse as it is for data blocks. In addition, when the redo log buffer reaches two–thirds of its capacity, it is flushed to disk. The redo log buffer is also flushed to disk when a commit occurs to ensure that the changes are permanent. It is recommended that the size of the redo log buffer be set within a range of 8192 bytes to 262144 bytes.

The Program Global Area (PGA)

The process architecture for systems not configured for multi–threaded shared (MTS) servers is essentially the same as that for Version 6. Most operating systems start (or fork) a shadow process, which is a database server process for the application session, when an Oracle application is invoked. Figure 6 – 2 shows a SQL*Plus application process, its Oracle shadow process, and an approximation of its PGA. SQL*Plus can only have one cursor open at a time, so its memory utilization is simplified. The circles shown in Figure 6 – 2 represent processes; the squares represent memory allocation.

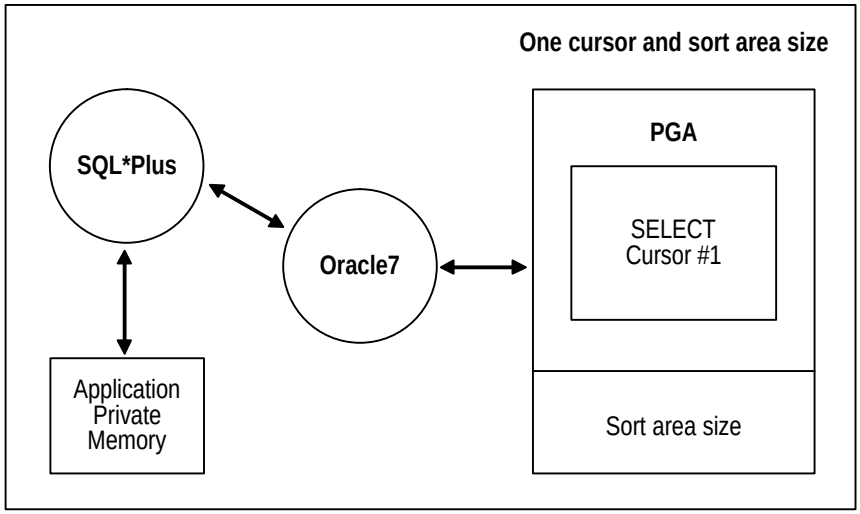

Figure 6 – 2 Process Global Area

Figure 6 – 2 shows the cursor for the SELECT SQL statement. Under Version 6, the cursor is synonymous with context area. In Oracle7, if the SELECT cursor computes a join, sort, or uses an aggregate function such as SUM(), AVG(), and so forth, the sort area is allocated based on a fraction of the value (in bytes) specified by the system parameter SORT_AREA_SIZE. The size of the sort area increases up to the value specified. All cursors for a given Oracle7 session use the same sort area to achieve more efficient use of memory than is possible in Oracle Version 6.

Figure 6 – 3 illustrates an Oracle7 session with multiple cursors.

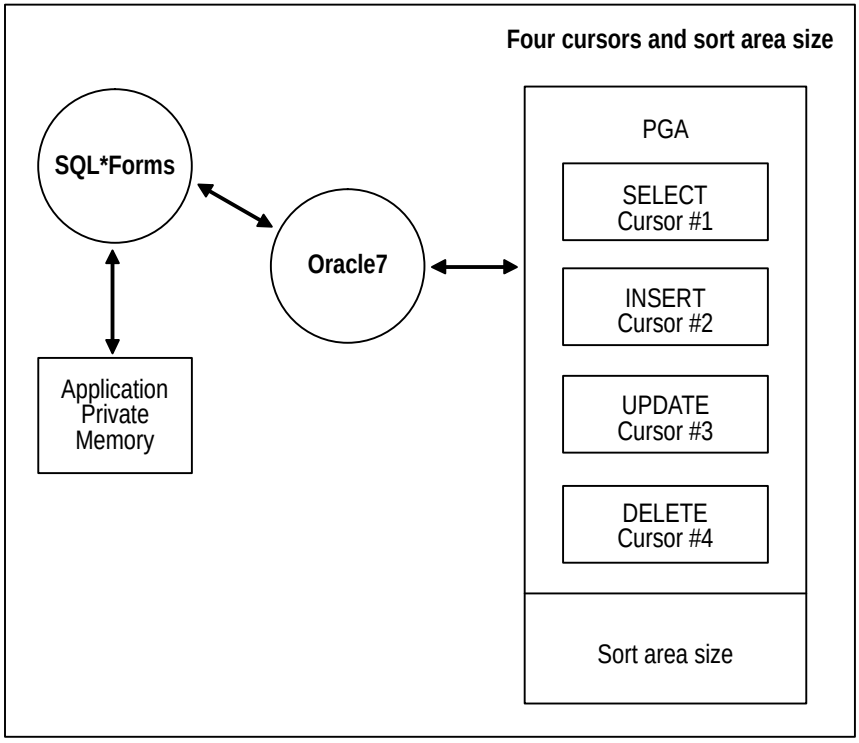

Figure 6 – 3 Oracle7 Session with Multiple Cursors

The PGA size is affected by the sort area, the number of open cursors, and the value of SORT_AREA_SIZE. In Oracle7, the cursor state, SQL text string, and execution plan reside in the Shared Pool. The memory needed for cursors in the PGA is approximately 4 KB per cursor. The Sort Area Size is 64 KB, by default, on most operating systems. Figure 6 – 3 shows a SQL*Forms application with multiple cursors and one sort area.

Any given Oracle application session may have from one to several hundred cursors opened. However, only one sort area is used. Only one cursor can be executing within an Oracle7 session; it remains blocked until the execution / fetch is complete.

As illustrated in Figure 6 – 3, memory is consumed fairly quickly by having too many open cursors. For example, SQL*Forms, Version 3 applications that use SQL*Forms, Release 2.3 style triggers are likely to exceed OPEN_CURSORS. Also note that each Oracle Forms (previously referred to as SQL*Forms) block, based on database tables, will automatically open four cursors: INSERT, UPDATE, DELETE, and SELECT, provided the block attributes are defined. If you have 10 such SQL*Forms blocks, you will have 40 open cursors, in addition to any cursors that may be opened with SQL*Forms triggers.

One final item you should consider is the private memory allocated by the front–end application. When the application connects to an Oracle7 instance, whether it be SQL*Net, multi–threaded shared servers, or a local TWO–TASK connection, the connection will result in the application allocating additional memory for the Local Data Area (LDA). In addition, the application will allocate private data structures for accomplishing its tasks. Figure 6 – 2 and Figure 6 – 3 provide a schematic representation of the application private memory area.

Oracle7 Executables

The size of Oracle7 executables is 1.5 to 2 times larger than Version 6 Oracle executables. (This is due to new functionality added to Oracle7.) The size of the executables also depends on which options you choose to include in your Oracle7 environment, such as distributed transactions, Oracle Parallel Server, and various SQL*Net adapters. You should adjust your system memory to accommodate the inclusion of such options when migrating from Version 6 to Oracle7.

The use of shared objects on many platforms provides some reduction in the size of client applications. However, the size of some shared objects (or libraries) can be quite large due to the new functionality of the Oracle7 releases.

Concurrent Access

The remaining issue for determining the memory size of an Oracle7 system is concurrent access and how that access is accomplished. In Oracle7, you have the following connect options:

1. Use local connections through the TWO–TASK architecture (same as Version 6).
2. Use remote connections through SQL*Net (same as Version 6).
3. Use multi–threaded shared servers for both local and remote connections.
4. Use TP monitors.

Option 1 requires more memory than options 2 or 3. In option 1, if both the client application and its Oracle server (or shadow) process reside on the same machine, memory is required for both. For example, 100 client application processes connected to Oracle7 result in 100 additional Oracle server processes on the system, totaling 200 in all.

For option 2, only the Oracle processes reside on the system since the clients are connected remotely. Thus, you need to pay attention to the size of the Oracle server processes and the size of the SGA.

Option 3, using the multi–threaded shared servers, is new in Oracle7. The multi–threaded shared servers feature allows the processes of several local and/or remote clients to connect to a single dispatcher process instead of having a dedicated Oracle shadow process. While not meant as a performance enhancement, multi–threaded shared servers allow more concurrent connections on an Oracle7 server, thereby improving throughput. Since multiple clients can connect to a single dispatcher, the memory utilization for concurrent user connections decreases. For further information on the multi–threaded shared server feature of Oracle7, refer to the September 1992 IOUW paper #243 entitled “Taking Advantage of the Multi–Threaded Server”.

Option 4, use of TP monitors, is an alternative for systems requiring a high number of users (greater than several hundred) all performing OLQP/OLTP type transactions. Such transactions are usually short–lived and do not require the user to make a direct connection to the database. All transactions are performed with messages routed by the TP (Transaction Processor) monitor service. The TP layer provides named services and coordinates service requests with various DBMS systems, including Oracle. The requirements for using TP monitors will vary greatly and will not be discussed in this manual. Please consult the appropriate TP monitor vendor for system requirements.

In summary, you should be able to obtain a good estimate of your system memory requirements, for a single system, by considering the following factors:

- the total size of the SGA
- the average number of open cursors and cursors that may cause sorts for a given Oracle application session
- the average size of the Oracle shadow processes that will include open cursors and sort areas
- the peak number of concurrent users on the system
- the average memory size of the Oracle front–end application

Disk Capacity and Input/Output (I/O)

All Oracle7 releases perform most efficiently when datafiles are spread out over several disk devices. If you are planning to migrate an existing Oracle Version 6 database to an Oracle7, Release 7.x database, the database can be migrated in place without requiring additional disk space for the database files. However, for any expansion plans you may have, it may be necessary to examine the current disk resources to determine if additional disks will be required to meet future administration and performance requirements.

For sites implementing a very large database (VLDB), approximately 50 gigabytes or larger, see the Oracle publication *The OFA Standard, Oracle7 for Open Systems, Part Number A19308–1*, which provides device and file naming conventions for better administration.

In addition, to distribute I/O requests across disks, consider adding disk controllers when adding disk drives to your hardware configuration. Placing all of your disk drives onto one or two controllers can lead to I/O bottlenecks.

If your migration plan requires continuous operation, even during media failures, you must configure your hardware and operating system to support volume/disk mirroring. A minimum configuration, for an operating system that supports mirroring, should mirror the redo log files, the rollback segment tablespace datafiles, and all data tablespace datafiles. Such a configuration, of course, requires additional disk drives and controllers; however, it also provides further flexibility in system availability and administration.

Keep in mind that mirroring volumes can encompass several disk devices per volume. Ensure that you have sufficient disk drives to support the entire volume, if you wish to mirror Oracle datafiles. Consult your hardware vendor for further information.

Sites that cannot afford to be down due to media failure or time–consuming backups should use triple mirrors.

Do not use all disk drives (or file systems) on the system for Oracle database files. Some drives should be reserved for user files, for placing small export files, or for maintenance operations that require a significant amount of space.

Use of the Oracle Parallel Server requires that each node have its own copy of Oracle7 software. Thus, many platforms, such as IBM RS6000, NCR 3XXX series, Pyramid MIS Server, Sequent Symmetry, Sun SPARKCenter, and others, may require additional private disks.

Block Size

The Oracle7 releases require a minimum block size of 1024 bytes. The use of long VARCHAR2 datatypes requires specification of special block sizes.

- Create a database block size of 4 KB or greater if you plan to use VARCHAR2 columns of length greater than about 1000 characters.
- The following table lists the supported, recommended, and default block sizes for desktop platforms using Release 7.2:

Minimum block size	1 KB
Minimum block size recommended	2 KB
Default block size for a new database	4 KB

CPU Resources

The Oracle7 releases are well suited for symmetric multi–processors (SMP) and massively parallel processors (MPP) systems. SMP systems are tightly coupled; the memory is shared among a pool of CPUs. MPP systems are loosely coupled; only message buffers and disk drives (or disk controlling processes) are shared. An MPP system is composed of separate and independent nodes.

Each Oracle7, Release 7.x server process (or shared server for MTS configuration) performs the requested task without going through a centralized database server. Thus, all Oracle7 releases rely on the operating system to handle process scheduling. Various, unrelated database tasks can be performed on different CPUs (for SMP systems) or on separate nodes on MPP systems.

Thus, for applications with high Online Transaction Processing (OLTP), a single CPU system can become CPU bound because the CPU resource becomes exhausted. SMP and MPP systems can distribute the workload across CPUs. The CPU utilization should be monitored to determine if additional CPU bandwidth is required.

Oracle Parallel Server

The Oracle7 Parallel Server technology allows multiple, homogeneous systems to share the same database. This is accomplished by hardware and software technology that allows disk volumes to be shared. You must configure the Distributed Lock Manager (DLM) provided by your hardware vendor if you intend to use the Oracle7 Parallel Server. The DLM configuration is dependent on the configuration of the Oracle7 Parallel Server database.

Oracle RDBMS releases earlier than Release 6.0.35 (or 6.2) did not contain the generic Oracle Parallel Server (OPS) feature. The Oracle7 releases can be configured with the OPS option. Install the OPS option only if your hardware and operating system support an Oracle–certified Distributed Lock Manager.

Distributed Transactions and Network Considerations

While Oracle, Version 6 supported distributed queries, the Oracle7 releases support distributed queries and DLM transactions protected by two–phase commits. If your applications require distributed transactions, it may be necessary to examine the network costs for performance for such applications. While distributed transactions is an important feature and provides further flexibility for the application, it also adds to the existing network traffic and may degrade performance. If the application heavily utilizes distributed transactions, it might be a good idea to install an additional network interface card or use a different, infrequently used subnet.

In addition, distributed queries that cause large tables to be brought over to the local server through SQL*Net may also cause performance degradations on the network. The application performing such queries should first be examined to see if other possible solutions can be found.

If the users of your applications connect to the database server from PC clients or other types of workstations using SQL*Net, the application should be tested to determine how much traffic it will introduce to the existing network. The amount of traffic can indicate if network bottlenecks will occur. If input/output (I/O) bottlenecks occur, either modify the application to reduce network traffic, use an existing, infrequently used subnet, or install additional network interface cards to distribute the network load.

CHAPTER

7

Migrating from Version 6 to Version 7

This chapter describes the techniques that are available to migrate a Version 6 database to a Version 7 database. The specific topics covered in this chapter are

- Migrating a Version 6 Database – General Comments
- Preparing to Migrate from Version 6 to Version 7
- Using the Migration Utility to Migrate from Version 6 to a Version 7 Release
- Using Export/Import to Migrate from Version 6 to a Version 7 Release
- Preserving Your Version 6 Database
- Tune the New, Release 7.x Database
- Enabling and Disabling Release 7.x Features
- Add New Features as Appropriate
- Develop New Administrative Procedures as Needed
- Downgrading to a Previous, Version 6 Database

Migrating a Version 6 Database

This section contains the following topics:

- General Comments
- SQL92 Compatibility

General Comments

Careful planning and proper tools can greatly reduce the complexity of migrating a Version 6 database. Oracle provides several tools, such as the Export/Import and Migration utilities, to make the migration process as simple as possible.

The Migration Utility can be used only to transform an earlier version to a later version of the Oracle database. *The Migration Utility cannot be used for reverse migration*. For example, the Migration Utility cannot be used to transform a Version 7 database to a Version 6 database.

In addition, the following features of Oracle aid in the Version 6 to Version 7 migration process:

- Version 6 can communicate with any of the Oracle7 releases using SQL*Net.
- The programming interface (for example, in the Oracle Precompilers and Oracle Call Interface) are the same for both Version 6 and Version 7. You can use either SQL*Net or relink applications designed for a Version 6 database to run them with any of the Oracle7 releases.
- A backward compatibility mode is provided when there are small incompatibilities between the functionality of Version 6 and Release 7.x databases. For information about upgrading and downgrading between the Version 7 releases, see Chapter 9 “Upgrading and Downgrading between Oracle7 Releases”.

SQL92 Compatibility

Although Oracle7 SERIALIZABLE mode is compatible with SQL92 and offers many benefits, compared with read–locking implementations, it does not provide semantics identical to such systems. Application developers should take into account the fact that reads in Oracle7 do not block writes as they do in other systems. For example, transactions that check for database consistency at the application level require coding techniques such as the use of SELECT FOR UPDATE. This issue should be considering when applications using SERIALIZABLE mode are ported to ORacle7 from other environments.

For more information about SERIALIZABLE isolation, see the *Oracle7 Server Application Developer’s Guide.*

Preparing to Migrate from Version 6 to Version 7

Become Familiar with the Features of the Release 7.x Target Database

The features and functionality of Version 6 and the Release 7.x databases are significantly different. You should pay particular attention to the following topics when migrating a Version 6 database:

- VARCHAR2 datatypes
- padded vs. non–padded comparison semantics
- integrity constraints
- privilege and resource management
- redo log files
- recovery strategies
- multi–threaded server configuration
- optimization
- character encoding scheme used for the database
- parallel query option
- symmetric replication

Using the Migration Utility to Migrate from Version 6 to a Version 7 Release

This section contains the following topics:

- General Considerations for Version 6 Migration
- Space Requirements
- Data Dictionary Conversion when using the Migration Utility
- Migrating to Trusted Oracle7
- Step 5: Migrate the Version 6 Database (using the Migration Utility)

General Considerations for Version 6 Migration

The Migration Utility is a program that converts some of the files and structures in your Version 6 database to Oracle7, Release 7.x format. The primary advantages of the Migration Utility are its speed and ease of use. The Migration Utility takes significantly less time than exporting/importing, and the migration process using the utility consists of a few easy steps. However, you cannot selectively migrate datafiles. Thus, the Migration Utility is useful if you want to migrate an entire Version 6 database to an Oracle7, Release 7.x database quickly.

When you use the Migration Utility, the entire database is converted, including database files, rollback segments, and the control file. At any point before actually migrating your source database (Step 5), you can still open and access data with Oracle Version 6. However, once you have performed Step 5 and migrated the Version 6 database to Oracle7, Release 7.x with the Migration Utility, you can only go back to Version 6 by restoring a full backup of the database. If you decide to use the Migration Utility as the method for migrating your database, see Chapter 4 “The Migration Utility” for detailed information about the Migration Utility and its use.

The following Version 6 structures must be converted before they can be used in any Oracle7 release:

- datafiles (file header only)
- data dictionary information
- control files

The Migration Utility creates a new data dictionary based on the Version 6 data dictionary. When the data dictionary is converted, all objects and users defined in the new dictionary automatically conform to all Oracle7 release specifications. Conversion of datafiles occurs later in the migration process. Only the file headers are changed; data blocks have the same format in Oracle7 releases as in Version 6.

Space Requirements

The Migration Utility requires that the SYSTEM tablespace have enough free space to hold the Release 7.x data dictionary and the existing Version 6 data dictionary concurrently. If necessary, add space to the SYSTEM tablespace.

The space required to hold the Release 7.x data dictionary varies depending on how many objects are in the database. Typically, a Release 7.x data dictionary will be approximately one and a half times as large as your Version 6 data dictionary.

If you do not allocate enough space in the SYSTEM tablespace, the Migration Utility will not complete the migration. An error will be returned stating how much additional space needs to be allocated.

To determine how much space is needed, run the Migration Utility with the CHECK_ONLY option set to TRUE. This causes the utility to do the space check only, without actually building the Release 7.x data dictionary. The database must be opened for this option to work.

Data Dictionary Conversion when using the Migration Utility

The following changes are made to information stored in the data dictionary:

- Columns defined in Version 6 as type CHAR are defined as type VARCHAR2 in all Oracle7 releases.
- Users with CONNECT, RESOURCE, or DBA privileges in Version 6 have the respective CONNECT, RESOURCE, or DBA role in all Oracle7 releases.
- The passwords for SYSTEM and SYS are reset to MANAGER and CHANGE_ON_INSTALL respectively.
- Integrity constraints defined in Version 6 are initially disabled in Oracle7 releases (except NOT NULL constraints, which are enabled after migration). After migration, you can manually enable these constraints as needed, using the ENABLE clause of the ALTER TABLE command.

Warning: If any row in a Version 6 table is inconsistent with the constraints of Version 7, in other words, a Version 6 row violates one or more Version 7 integrity constraints, the integrity constraints remain disabled. Oracle returns an error message indicating that the integrity constraints are disabled.

For more information, see *Oracle7 Server SQL Reference.*

Migrating to Trusted Oracle7

The Migration Utility cannot migrate a Version 6 database to Trusted Oracle7. One of the methods of migrating a Version 6 database to Trusted Oracle is to migrate from Version 6 to Oracle7 using the Migration Utility, and then to migrate this database to Trusted Oracle. For information about migrating from Oracle7 to Trusted Oracle7, see *Trusted Oracle7 Server Administrator's Guide.*

Step 5: Migrate the Version 6 Database (using the Migration Utility)

Before proceeding, you should review the migration steps outlined in Chapter 1 “Migration Overview”. The following steps expand on Step 5 of the migration procedure and explain how to migrate your source database using the Migration Utility.

1. Install and run the Migration Utility.

OSDoc

Additional Information: This task is operating system specific. Refer to your operating system–specific Oracle documentation.

There are several parameters that can be passed to the Migration Utility; these are outlined below. They are discussed in more detail in your operating system–specific Oracle documentation.

ALLOW_ OFFLINE	when TRUE, allows migration of release 6.0.34.3 databases and earlier with offline tablespaces. Be certain that these tablespaces were taken offline cleanly before using this option; see Chapter 3 “Preparing to Migrate” for details.
CFILE	specifies the filename of control file (if not using the expected default).
CHECK_ ONLY	when TRUE, performs space calculations without performing migration.
CNVFILE	specifies CONVERT.ORA filename (if not using the expected default).
DB_NAME	specifies the name of the database to migrate.
DUMPCF	a diagnostic tool (primarily for Oracle Worldwide Customer Support).
ECHO	when TRUE, echoes the commands issued by the MIGRATE.BSQ script.
MIGFILE	specifies MIGRATE.BSQ filename (if not using the expected default).
NEW_ DATABASE	specifies a new name for your migrated database. **Note**: Some databases use the default name of “DEFAULT”. A different, more meaningful name should be chosen.
NO_SPACE_ CHECK	when TRUE, does not perform space check before migration.
PFILE	specifies the name of the parameter file (if not using the expected default, INIT.ORA).
SPOOL	specifies filename of file to which to spool output.

The Migration Utility creates a user, MIGRATE, to hold data dictionary information temporarily. If you have a user by that name, you need to drop all objects owned by this user. You should first either copy the user's objects to another user or export the user's objects.

Warning: Do not attempt to open your database with Version 6 of Oracle after running the Migration Utility. If you open the Version 6 database before proceeding with the remaining steps, the migration process (specifically, Step 6, described in Chapter 3 "The Migration Utility") will fail with an error.

The Migration Utility drops the Version 5 and Version 6 views (such as those created by CATEXP6.SQL) and catalogs (such as those created by CATALOG.SQL). You can usually run the Migration Utility multiple times losing only these views and catalogs, as well as user–defined views and synonyms. It should only be necessary to restore your database if you wish to return to the original database after having executed the ALTER DATABASE CONVERT command. Refer to Appendix E for more information about errors encountered during the migration process.

The Migration Utility creates a binary file that creates the control file. The name of this file is operating system dependent. Do not alter this file in any way.

2. Install Release 7.x.

OSDoc

Additional Information: Installation of Release 7.x is operating system specific. Refer to your operating system–specific Oracle documentation. Read the README.DOC file included with your Release 7.x installation for any late additions or modifications to the product.

3. Issue the following command to connect to the Release 7.x instance:

```
SVRMGR> CONNECT INTERNAL
```

Note: You must be connected as INTERNAL to perform the rest of the migration. The connect procedure is different from Version 6. You must now issue CONNECT INTERNAL *before* STARTUP.

4. Start a Release 7.x instance, but do not mount the database. Issue the following command:

```
SVRMGR> STARTUP NOMOUNT
```

5. Issue the following command:

```
SVRMGR> ALTER DATABASE CONVERT;
```

   This command converts the file headers and builds a new Release7.x control file. After this process is complete, the Version 6 data dictionary tables no longer exist in the Release 7.x database.

6. Open the Release 7.x database by issuing the following command:

```
SVRMGR> ALTER DATABASE OPEN RESETLOGS;
```

   All datafiles and rollback segments that are online when the Release 7.x database is opened are converted to Release 7.x format. Refer to the section in this chapter "Preserving Your Version 6 Database" for more information on shutting down the source (original production) database and database file and rollback segment conversion.

7. Once you open the Release 7.x database and start an instance, you can drop the MIGRATE user created temporarily to hold the data dictionary. Issue the following statement to drop this user and reclaim the space allocated to it:

```
DROP USER migrate CASCADE;
```

   Objects in this user's schema are no longer needed once the conversion is complete. You can also delete the binary file that creates the Oracle7 data dictionary base tables.

8. Run the Release 7.x CATALOG.SQL script to create the data dictionary views on the Release 7.x base tables.

```
SVRMGR> @catalog
```

   Run the Release 7.x CATPROC.SQL script which will run all of the scripts required for, or used within, PL/SQL.

```
SVRMGR> @catproc
```

   If you wish to create additional data dictionary structures, see *Oracle7 Server Reference* for a complete list and description of available scripts.

   You might want to run the CATALOG6.SQL script, which contains Version 6 data dictionary views, if you have not yet migrated your applications and the applications rely on obsolete or changed Version 6 data dictionary views. Version 6 views with the same name as Release 7.x views have a "6" appended to the name to prevent naming conflicts. Refer to Appendices A, B, C, and D for lists of changed and obsolete data dictionary views.

If you purchased any Oracle options, such as the distributed option, the parallel server option, or the advanced replication option, you must also run the appropriate script to install each option. These scripts are described in *Oracle7 Server Reference.*

Using Export/Import to Migrate from Version 6 to a Version 7 Release

This section contains the following topics:

- Basic Export/Import Steps
- Export/Import Limitations
- Data Definition Conversion when using Export/Import
- Space Requirements for Export/Import
- Time Requirements for Export/Import
- Step 5: Migrate the Version 6 Database (using Export/Import)

Basic Export/Import Steps

The Export/Import method of migrating a database involves the following two steps:

1. The first step is to export the data from the database that you wish to migrate.
2. The second step is to import the exported data into the database to which you wish to migrate.

You must create the target database (the Release 7.x database) before using the Export/Import method of migration.

The Export Utility copies the data in your source database to an export file, from which the Import utility can load the data into an Oracle7, Release 7.x database. An important distinction between Export/Import and the Migration Utility is that the physical data in your database is copied to a new location with Export/Import, while the Migration Utility changes only the file headers and the definitions of the data in the files where they currently reside.

The Export/Import method of migration provides some additional advantages.

- You can defragment the data. A full database import can compact the data and improve performance.
- You can restructure your database; that is, you can create new tablespaces, or modify existing tables or tablespaces.
- You can migrate only certain database objects or users. You can selectively import only objects and users that need to be imported at a given time.

Since the Export/Import method of migration does not change the Version 6 database, your Version 6 database is available at any point during the migration process. This allows you to keep a Version 6 database running in parallel with an Oracle7, Release 7.x database without requiring the restoration of a backup. (However, you should not change the Version 6 database unless you make exactly the same changes to the Oracle7, Release 7.x database.) Also, a full export can serve as an archive of your Version 6 database.

Export/Import Limitations

The Export/Import method has the following limitations:

- For a large database, a full database Export/Import can take a significant amount of time.
- For a large database, a full database Export/Import can require a substantial amount of temporary storage space for the data.
- Because you should not make changes to the Version 6 database after performing the export, your application is unavailable until the migration is completed.

If you decide to use Export/Import as the method for migrating your database, see Chapter 5 “Migrating Using Export/Import” for detailed information about using Export/Import.

When migrating a Version 6 database to a Version 7 database, you would first export the desired data from the Version 6 database. Then you would import the exported data into the Version 7 database.

To use the Export/Import method, the Oracle Corporation recommends that you use the version of the Export Utility shipped with the version or release of the source database, which, in this case, would be Version 6. Once you have exported the desired data, you must then use the version of the Import Utility shipped with the version or release of the destination database. You should use the Export and Import utilities for migration only after you have carefully read Part I of *Oracle7 Server Utilities*.

Data Definition Conversion when using Export/Import

When importing data from one version to another version, the Import Utility makes changes to data definitions as it reads in the export file. For example, when importing data from Version 6 to an Oracle7 release, the Import utility makes the following changes to data definition statements in the export file:

- Columns defined as CHAR from a Version 6 export file are changed to VARCHAR2.
- Users with CONNECT, RESOURCE, or DBA privileges in Version 6 are granted the respective CONNECT, RESOURCE, or DBA roles in the Release 7.x database.
- Constraints defined on tables in Version 6 are created in Release 7.x, but they are initially disabled (except NOT NULL constraints, which are enabled after import). After migration, you can manually enable these constraints as needed, using the ENABLE clause of the ALTER TABLE command. The SQL syntax for integrity constraints in Version 6 is different from the syntax in a Release 7.x database. Import automatically adjusts the declaration so that integrity constraints are properly imported in a Release 7.x database (unless V6 compatibility mode is in effect).

Space Requirements for Export/Import

The amount of space required for an export depends upon the amount of data you are exporting. Examine the views USER_SEGMENTS or DBA_SEGMENTS to determine the amount of space occupied by the data. These views give you the number of segments allocated, but keep in mind that some segments can be allocated but unused. Refer to the *Oracle7 Server Administrator's Guide* for more information on estimating space usage.

Time Requirements for Export/Import

The Export/Import method of migration may require several hours. Therefore, you may need to schedule your migration during non–peak, production hours. The time required to complete a migration will, of course, increase for databases that contain large amounts of data or a large number of indexes.

> **Note:** Once an Export/Import migration has started, the database being migrated is unavailable for all production tasks.

Step 5: Migrate the Version 6 Database (using Export/Import)

The following Export/Import steps are completely general and can be applied to any Oracle version or release.

To migrate an Oracle database using Export/Import, perform the following steps:

1. Export all desired objects from the original database *using the Export utility shipped with the original database*. Refer to *Oracle7 Server Utilities* for a complete description of how to use the Export utility for Version 6.
2. Install the Oracle7, Release 7.x database. A default database may be created in this step. If so, you can proceed to Step 4; otherwise, continue with Step 3.

   

   OSDoc

   Additional Information: Installation of an Oracle7 release is operating system specific. See your operating system–specific Oracle documentation. Read the README.DOC file included with your Release 7.x installation for any late additions or modifications to the product.
3. Create the Release 7.x database. All of the issues involved in database creation are covered thoroughly in the *Oracle7 Server Administrator's Guide*.
4. Open the Release 7.x database and start an instance. From the Server Manager prompt, issue the following commands:

   ```
   SVRMGR> CONNECT INTERNAL
   SVRMGR> STARTUP
   ```
5. You should pre–create tables, tablespace, and users in Release 7.x as necessary, for example, to improve space usage by changing storage parameters. When you pre–create tables using SQL*Plus or Server Manager, you must either run in the original database compatibility mode or specifically make allowances for the data definition conversions outlined earlier in this chapter under "Data Definition Conversion when using Export/Import".
6. Import the data and tables from the original database export using the Import utility shipped with Release 7.x. Refer to *Oracle7 Server Utilities* for a complete description of how to use the Import utility.

7. After migrating, you may discover that your tables were not imported properly. Views and synonyms may not be created in the correct order when dependencies exist (for example, when a view is based on a synonym). You may have to perform one of the following procedures to finish importing correctly:

   – If all of the rows for all tables were not successfully imported, repeat the import until it completes successfully. Be certain that IGNORE=Y and ROWS=N. IGNORE=Y causes Import to overlook “object already exists” errors and ROWS=N indicates that you do not want to import the rows of table data.

   – If some tables were imported successfully, while others were not even created, repeat the import with IGNORE=N and ROWS=Y to ignore the “object already exists” errors and re–import the rows.

   – If the tables had some, but not all rows imported, drop the incomplete tables and repeat the previous step.

For more information on the use of the Export and Import utilities, see *Oracle7 Server Utilities*.

Preserving Your Version 6 Database

This section contains the following topics:

- Shut Down the Source (original production) Database
- Back up the Source (original production) Database
- Change Parameter Files (INIT.ORA Files)
- Change SQL Scripts
- Update Datatypes
- Update Database Administration Scripts
- Update Database Link Names
- Move Constraint Identifiers
- Change Unique Indexes to UNIQUE or PRIMARY KEY Integrity Constraints

Shut Down the Source (original production) Database

The Version 6 database must be shut down normally so that there is no outstanding redo information, and no uncommitted transactions.

If you migrate using the Migration Utility, all source datafiles that are online when the target database is opened are automatically converted to the target database file format. Files that are offline when the target database is opened remain in the source database file format.

You do not need to convert all of the database files to Oracle7, Release 7.x format immediately. The remaining files are converted when they are brought online in Oracle7, Release 7.x.

Rollback segments are converted as they are accessed by the Oracle7, Release 7.x database. Thus, all rollback segments that are in tablespaces that are online when the database is first opened in Oracle7, Release 7.x are converted. If a source database rollback segment is in a tablespace that is offline when the Oracle7, Release 7.x database is opened, it is converted the first time it is brought online in Oracle7, Release 7.x.

Back up the Source (original production) Database

After shutting down the database, you should make a full backup of the source database before proceeding with migration. Be certain to back up all datafiles, control files, parameter files, online redo log files, and any script files used to create objects in the source database.

Suggestion: You should make a backup of the online redo log files even though there should be no outstanding redo information. This allows you to easily restore your source database if necessary.

You can only migrate with offline tablespaces that were taken offline cleanly, using the NORMAL or TEMPORARY options; otherwise (for example, if you had performed an ALTER TABLESPACE OFFLINE IMMEDIATE), you receive an error. For 6.0.34.3, or later, databases you can simply bring these tablespaces online and then back offline with no other changes, and repeat the migration procedure.

Before release 6.0.34.3, offline tablespaces may appear to have outstanding save undo, even if they were taken offline cleanly. If you are certain that your offline tablespaces do not have outstanding save undo, you can set the ALLOW_OFFLINE migration parameter to TRUE to continue the migration.

Warning: When ALLOW_OFFLINE is TRUE, any offline tablespaces that were not cleanly taken offline will be lost after migration. You should not use this parameter with a release 6.0.34.3, or later, database.

Change Parameter Files (INIT.ORA Files)

Certain initialization parameters are obsolete in all Oracle7, releases. You must remove all obsolete parameters from any parameter file that starts an Oracle7, Release 7.x instance. Obsolete parameters may cause errors if used with an Oracle7, Release 7.x database. You must also alter any parameter whose syntax has changed in Oracle7, Release 7.x. Refer to Appendices A, B, C, and D for lists of obsolete and changed parameters for Release 7.0, Release 7.1, Release 7.2, and Release 7.3, respectively.

Change SQL Scripts

All SQL scripts that you built that created objects in the source database should be changed to conform to Oracle7, Release 7.x syntax.

VARCHAR is synonymous with VARCHAR2 in all Oracle7 releases. However, VARCHAR may have different properties in a future version of Oracle. If any of your scripts use VARCHAR, you should update them to use VARCHAR2.

In addition, there are two new reserved words in the SQL language: ROWLABEL and VARCHAR2. If any of the objects in your database have names that match either of the new reserved words, you must take the following action, depending upon the object type:

- If the reserved word is used as a column heading, you must copy the data to a new table with a different column name.
- If the reserved word is used as the name of a table, sequence, synonym, or view, you must rename the object.

If you want to use a reserved word as an object name, the reserved word must be enclosed in double quotes (for example, “ROWLABEL”). Refer to *Oracle7 Server SQL Reference* for more information on object naming.

Update Datatypes

Attention: The Version 6 CHAR datatype is equivalent to the new Oracle7 VARCHAR2 datatype, except for a difference in maximum length. VARCHAR2 datatype values are variable–length character strings with a maximum length of 2000. Oracle compares VARCHAR2 values using non–padded comparison semantics. In all Oracle7 releases, values with the CHAR datatype are fixed–length character strings with a maximum length of 255 characters. Oracle7 compares CHAR values using blank–padded comparison semantics.

Update Database Administration Scripts

For database administration scripts, you should use the Version 7 CREATE USER and GRANT CREATE SESSION commands instead of the Version 6 GRANT commands for creating users and assigning resource quotas. The GRANT syntax continues to provide similar effects in all Oracle7 releases as in Version 6, but future versions of Oracle may not support this functionality.

Warning: You must assign resource quotas to each user using the QUOTA option of the CREATE USER or ALTER USER commands. There is no equivalent to granting a resource to PUBLIC. Resource quotas are not assigned in the GRANT command in Oracle7 releases. You should remove any resource quotas in GRANT statements.

Granting the RESOURCE role in an Oracle7 release permits you to have the UNLIMITED TABLESPACE privilege, so the following statement is equivalent in Version 6 and all Oracle7 releases:

```
GRANT RESOURCE TO user1;
```

Note: If you use any Server Manager or SQLDBA scripts to start instances or open the database, you must CONNECT AS SYSOPER or CONNECT AS SYSDBA (this command was CONNECT INTERNAL under Version 6) before issuing the STARTUP command in Oracle7 releases.

To comply with ANSI/ISO SQL standards, two dashes, "– –", are now recognized as starting a comment. Although it is unlikely that you are using two dashes outside a quoted string, you should be aware that all Oracle7 releases treat the text following "– –" as a comment.

Update Database Link Names

Oracle7 releases assume that any characters following the @ sign in a database link are part of the database name. For instance, in Version 6, to select the column ENAME from the EMP table at BOSTON, you could have typed the following statement:

```
SELECT emp@boston.ename FROM emp@boston
```

With Oracle7 releases, you must change this SELECT statement to the following statement:

```
SELECT ename FROM emp@boston
```

Move Constraint Identifiers

The constraint clause for the CREATE TABLE command has new syntax. The optional constraint identifier (CONSTRAINT *name*) has moved from the end of the clause to the beginning of the clause. For example, the following example shows a constraint clause used in a Version 6 SQL statement:

```
CREATE TABLE emp
   (column definitions...
   FOREIGN KEY (deptno) REFERENCES dept (deptno)
      CONSTRAINT deptno_fk)
```

In Oracle7 releases, this same statement must be written as shown below.

```
CREATE TABLE emp
   (column definitions...
   CONSTRAINT deptno_fk
      FOREIGN KEY (deptno) REFERENCES dept (deptno))
```

Version 6 scripts that *do not* contain a CONSTRAINT identifier require no modification and run with all Oracle7 releases. However, Version 6 scripts that *do* contain a CONSTRAINT identifier must be modified, as shown in the previous example, before running the scripts with an Oracle7 release. See *Oracle7 Server SQL Reference* for a description of the CONSTRAINT clause, Chapter 4 “The Migration Utility” for comments on integrity constraints, and
Chapter 9 “Upgrading and Downgrading between Oracle7 Releases” for a discussion of compatibility mode.

Change Unique Indexes to UNIQUE or PRIMARY KEY Integrity Constraints

Integrity constraints can be used in Oracle7 releases to enforce uniqueness among column values. Because unique indexes might not be supported in future versions of Oracle, you should begin using UNIQUE or PRIMARY KEY integrity constraints instead of unique indexes.

Indexes can now be validated using the ANALYZE command. You should begin using this command, because the VALIDATE INDEX command might not be supported by future versions of Oracle.

Tune the New, Release 7.x Database

Most methods used to tune a Version 6 database and related applications have either the same effect or are unnecessary for Release 7.x. Any actions you took to tune your source database and applications should not impair the performance of Release 7.x.

Release 7.x includes several new features designed to improve performance of your database and related applications. Some of these features, such as the multi–threaded server configuration and shared SQL areas, require little or no action on your part to implement. Other features, such as hashed clusters, direct path loader, cost–based optimization, and even roles, have the potential to improve performance. For additional information on tuning the database, see *Oracle7 Server Tuning*.

Enabling and Disabling Release 7.x Features

After migrating to an Oracle7, Release 7.x database, certain steps must be followed to properly enable new Release 7.x features.

Once you have migrated to a Release 7.x database, you may wish to downgrade to your previous Version 6 database. Before you downgrade to your previous Version 6 database, you must disable some of the features that were enabled for operation with the Release 7.x database.

For more information on enabling and disabling Release 7.x features, see Chapter 9 “Upgrading and Downgrading Between Oracle7 Releases”.

Add New Features as Appropriate

Appendices A, B, C and D of this manual describe many of the new features available in Oracle7 releases. You should determine which of these new features can benefit your application and develop a plan for incorporating them. Release 7.x databases offer many new features that can affect not only your database design, but your application design as well. You do not, however, have to make any immediate changes to begin using your Release 7.x database. You may prefer to introduce these enhancements into your database and corresponding applications gradually.

Chapter 8 “Migrating Version 6 Applications” describes how you can enhance your applications to begin taking advantage of these new features. You should already have tested your applications and successfully run them with Release 7.x in a special V6 compatibility mode.

Develop New Administrative Procedures as Needed

After familiarizing yourself with the new Release 7.x features, you should review your database administration scripts and procedures to determine if any changes are necessary. Some topics to consider are

- managing mirrored online redo log files
- administering the multi–threaded server
- using new Server Manager features, such as KILL SESSION and DISABLE RESTRICTED SESSION
- distributing your database to improve performance
- determining optimal rollback segment size
- reclaiming unused space in datafiles using the Release 7.2 “Resizeable Datafiles” enhancement
- secure connections for privileged users
- read–only tablespaces

Completion of the final migration steps described in this chapter will allow you to begin taking advantage of Release 7.x functionality. You need to coordinate your changes to the database with the changes that need to be made to each application. For example, by enabling integrity constraints in the database, you may be able to remove some of this data checking from your applications.

Downgrading to a Previous, Version 6 Database

You might want to return (downgrade) to a Version 6 database after migrating to Release 7.x. If you have not entered any new data with Release 7.x, you can restore a complete backup of the Version 6 database and open it again. Be certain to use a complete backup that contains the original initialization parameters that were used in the Version 6 database. Once you have entered data with Release 7.x, you cannot reverse the methods of migration described in this manual.

Warning: You must disable certain features of the 7.x database before you return to a Version 6 database. For more information about setting the COMPATIBLE parameter, see Chapter 9, "Upgrading and Downgrading between Oracle7 Releases".

However, there are alternative methods of sending table data from a Release 7.x database to a Version 6 database. For more information about upgrading and downgrading between the Version 7 releases, see Chapter 9 "Upgrading and Downgrading between Oracle7 Releases".

- You can use the SQL*Plus COPY command to copy the data from the Release 7.x tables into the tables in the earlier version or release database.
- You can create the table again in the earlier version or release database (using CREATE TABLE AS SELECT) by selecting the data in the Release 7.x table through a distributed query from the source database to Release 7.x using a database link.
- You can run the CATEXP6.SQL file on your Release 7.x database. Then perform a Version 6 export of the Release 7.x data and import that data into the Version 6 database. To see which of your Release 7.x database objects will not be exported over SQL*Net when using a source database export, run UTLEXP6.SQL. Note that you cannot import a Release 7.x export into a Version 6 database.
- You can use SQL*Plus to create non–Oracle text files from the Release 7.x database and then use SQL*Loader to load this data back into a Version 6 database.

For more information about performing a Version 6 export of Release 7.x files, see *Oracle7 Server Utilities*. For more information about the COPY command, see the *SQL*Plus User's Guide and Reference*. For more information about the AS clause of the CREATE TABLE command, see *Oracle7 Server SQL Reference*.

Each of these methods of replacing data have some restrictions. These restrictions, outlined below, hold only for replacing data in Version 6 databases. Not all restrictions apply to all methods. For more information about restrictions, see Chapter 8 “Migrating Version 6 Applications”.

- If a table has columns of character–type data (CHAR, which blank–pads data), any trailing spaces are retained when the data is copied to Version 6. Use an UPDATE statement, similar to the example shown below, if you want to remove these trailing spaces after copying the data to Version 6.

  ```
  UPDATE emp SET ename = RTRIM(ename)
  ```

- If a table has columns of type VARCHAR2 in Release 7.x, the Version 6 table must have the same columns of type CHAR.
- If columns of type VARCHAR2 in Release 7.x contain any rows with data longer than 255 characters, you must either truncate this data to 255 characters or fewer or store the data in multiple fields before copying the data.
- Only tables can be copied. Synonyms, sequences, views, and any other database objects created or changed in Release 7.x must be re–created in the Version 6 database.

These methods of returning data to Version 6 are relatively simple if few tables have been updated using Release 7.x. However, copying an entire database of tables can be a long and complicated task, so you should decide whether you need to return to Version 6 before many tables are updated using one of the releases of Oracle7.

CHAPTER

8 Migrating Version 6 Applications

This chapter describes how you can use your Oracle tools and applications with Oracle7 releases. Usually, little or no change is required of your applications to achieve the same or enhanced functionality running on a Release 7.x database. The topics included in this chapter are

- Oracle7 Changes and the Programmatic Interfaces
- Migrating Your Oracle Programmatic Interface Applications
- Migrating Your Precompiler Applications
- Migrating Your OCI Applications
- Migrating Your Oracle Forms Applications
- Migrating Your CASE*Dictionary Applications
- Migrating Your SQL*Report (RPT/RPF) Reports
- Migrating Your Oracle Reports and SQL*ReportWriter Applications
- Migrating Your SQL*Plus Scripts
- SQL*Net

If you are migrating applications to Trusted Oracle, there are additional features of which you need to be aware. For more information about migrating your applications to Trusted Oracle, see the *Trusted Oracle7 Server Administrator's Guide*.

Oracle7 Changes and the Programmatic Interfaces

This section contains the following topics:

- Retaining Version 6 Behavior
- Retaining Release 1.5 and Version 6 OCI Behavior

Retaining General Version 6 Behavior

The behavior of Oracle7 has changed in some minor ways from Version 6 to accommodate ANSI/ISO standards. Some of these changes can be averted by specifying option and procedure call settings that preserve Version 6 behavior.

The following changes are always in effect and cannot be avoided. Some minor recoding of applications might be required to assure continued correct behavior.

- In SQL statements, the special keyword USER is considered a literal and, if compared to other literals, uses padded comparison semantics.
- All string literals have both the fixed–length and standard–compare attributes, and ANSI/ISO string comparison rules apply.

Retaining Version 6 Precompiler and OCI Behavior

The following changes can be averted by retaining Version 6 behavior using specific option settings to the precompilers or, for OCI applications, by carefully selecting procedure calls and associated arguments. Most of these changes were introduced for standards compliance.

- CHAR columns are fixed length created by a CREATE TABLE or CREATE CLUSTER statement.

  Note: If you are migrating a Version 6 application to any of the Version 7 releases and plan to use the CREATE TABLE...AS SELECT command, CHAR datatypes will not be correctly changed to VARCHAR2. To avoid this problem, specify a V6 compatibility mode, as follows:

  ```
  SQL> SET COMPATIBILITY V6
  SQL> CREATE TABLE <tablename> AS SELECT ....
  ```

- The syntax for specifying constraint names on table columns has changed. The constraint name must now be specified at the beginning of the CONSTRAINT clause.

- In Version 6, if a NULL value was fetched into a host variable, and no indicator variable was present, you received the error message “ORA–01405: fetched column value is NULL” only if you set MODE=ANSI. In Oracle7, you always receive this message (unless you set DBMS=V6).
- In Version 6, if a value fetched from the database had to be truncated to fit the host variable, and no indicator variable was present, you did not receive an Oracle error message if you set MODE=ANSI. In Oracle7, you never receive a message (unless you set DBMS=V6).
- The DESCRIBE of a fixed–length string returns datatype 96 (CHAR) instead of datatype 1. Datatype 1 is now VARCHAR2. This results from the difference between the Version 6 CHAR datatype and the Version 7 CHAR datatype.
- PCTINCREASE can no longer be specified in the CREATE ROLLBACK SEGMENT statement.
- Illegal MAXEXTENTS storage parameters are no longer allowed.

In the Release 1.5 Precompilers, specifying DBMS=V6 causes SQL statement behavior to remain the same as when working against Oracle, Version 6, and applications need not be recoded to accommodate the changes listed above. Specifying DBMS=V7 to obtain Oracle7 behavior might require slight recoding. For more information, see the *Programmer’s Guide to the Oracle Precompilers.*

In the Oracle7 OCIs, use of the existing OSQL3 parse call, or the new OPARSE call with the LNGFLG parameter set to Version 6, similarly guarantees Version 6–like program behavior. Substituting OPARSE for OSQL3 and specifying LNGFLG=V7 might require slight recoding to handle the changes listed above. For more information, see the *Programmer’s Guide to the Oracle Call Interface.*

Migrating Your Oracle Programmatic Interface Applications

This section contains the following topics:

- New Features of the Oracle Call Interface
- Non–blocking Oracle Call Interface
- SQL*Module
- SQL*Module Default WITH INTERFACE
- Migrating Precompiler and OCI Applications

The Oracle Programmatic Interface family consists of the Oracle Precompilers, the Oracle Call Interface, and the new SQL*Module. The tool that you select to create an application varies depending upon your needs. This section briefly discusses the changes to these tools resulting from the latest release of the database or the tool itself, and the implications of these changes for existing programs.

New Features of the Oracle Call Interface

Upgrading to any Oracle7 release automatically upgrades your Oracle Call Interface (OCI) libraries.

Attention: Existing applications must be relinked with these new libraries. If desired, you can also recode existing applications or code new applications to take advantage of the following new features.

Deferred Database Calls

A new link option reduces the amount of messaging traffic required between OCI clients and Oracle by transparently bundling BIND and DEFINE variable information on the client. Bundled (or deferred) information is only transmitted when required by DESCRIBE, EXECUTE, and FETCH calls.

New OCI Calls

ODESCR replaces ODSC and OBNDRA supplements OBNDRV/OBNDRN calls. OFLNG allows piecewise access to the data stored in Oracle7 LONG and LONG RAW database column types. OPARSE and OEXFET can be used with deferred database linking to realize even greater performance improvements using network optimization.

Non–blocking Oracle Call Interface

Release 7.2 enhances responsiveness of client/server applications by overcoming the limitations of synchronous OCI calls. Applications are now allowed to continue while an OCI call is being processed. Four new functions support non–blocking OCI

- **ONBSET:** This function places a database connection in the non–blocking mode for all subsequent OCI calls on the current connection.

- **ONBTST**: This function tests to see if a database connection is in the non–blocking mode.
- **ONBCLR**: This function places a database connection in the non–blocking mode.
- **OLOG**: This function establishes a non–blocking database connection between an OCI application and an Oracle database.

For more information on non–blocking OCI, see the *Programmer's Guide to the Oracle Call Interface.*

SQL*Module

A new, separately licensed product, SQL*Module, provides the best of both the Precompiler and OCI development environments, while delivering 100% compliance with relevant ANSI and ISO standards.

Like the Precompilers, SQL*Module provides an easy capability to execute Static SQL Data Manipulation Language statements. However, like the OCIs, the mechanism by which the host program executes the statements is a simple host language procedure call. Because no embedded SQL code is present in the application source code, you can continue to use your standard programming tools to edit and debug your code.

You might choose to take advantage of this new tool to create any new applications or even to recode your existing applications. SQL*Module incorporates the MODE and FIPS options of the Release 1.5 Precompilers.

SQL*Module Default WITH INTERFACE

SQL*Module, Release 1.1 does not require a WITH INTERFACE clause to call Oracle stored procedures.

You might want to use the WITH INTERFACE clause as follows:

- Map PL/SQL types to embedded SQL types that are subsequently mapped to host language types.
- Specify which parameters are associated with NULL INDICATORS.
- Specify how error conditions are to be reported to the client; for example, with SQLCODE, SQLSTATE, or both.

If you use the default WITH INTERFACE clause, a default mapping will be automatically applied.

For more information on SQL*Module WITH INTERFACE, see the *SQL*Module User's Guide and Reference.*

Migrating Precompiler and OCI Applications

Before you migrate your Oracle database from Version 6 to Oracle7, migrate any OCI and Precompiler applications that you plan to use with your Oracle7 databases. You can then test these applications on a sample Oracle7 database before migrating your production database.

The amount of effort involved in this migration process is dependent upon the degree to which you want to take advantage of the Programmatic Interfaces and Oracle7. In order of increasing difficulty, you can choose to

- maintain your existing performance and functionality
- boost the performance of your applications
- take advantage of the new Oracle7 database functionality

See the "Migrating Your Precompiler Applications" and "Migrating Your OCI Applications" sections which follow in this chapter for the specific steps required to migrate your Precompiler and OCI applications.

Migrating Your Precompiler Applications

You must complete a subset of the following steps to use your existing precompiler applications with an Oracle7 database. You may optionally begin at any step; however, once you begin you must complete any steps that follow. For instance, if you begin with Step 3, you must complete Steps 4 and 5. Step 5 is required of all applications.

1. If you want your application to be fully ANSI/ISO compliant, you should recode your application according to the ANSI/ISO guidelines published in the *Programmer's Guide to the Oracle Precompilers.*
2. If you want to take advantage of the new features offered by Oracle7, you must recode your applications to reflect the differences between Version 6 and Oracle7 that were outlined previously in this chapter.
3. If you want to take advantage of new features offered by the Release 1.5 Precompiler, you must re–precompile your application using the appropriate option settings as explained below. For a complete explanation of each of the option settings, including those that are now obsolete, see the *Programmer's Guide to the Oracle Precompilers.*

- Use the following option settings to maintain existing functionality:

  ```
  DBMS=V6
  ```

  other existing options=no changes

- Use the following settings to improve performance and take advantage of the new features available with Oracle7:

  ```
  DBMS=V7
  MODE=Oracle
  ```

  other existing options=no changes

- To be fully ANSI compliant, use the following settings:

  ```
  DBMS=V7
  MODE=ANSI
  FIPS=YES
  ```
  (optionally, to receive FIPS warning messages)

  other existing options=no changes

4. Recompile your application with your compiler.
5. (**Required**) Relink your application with Release 1.5 of the Oracle Runtime Library (SQLLIB), which is included with the precompiler. You must complete at least this step to use your applications with Oracle7.

Migrating Your OCI Applications

You should complete the following steps to use your existing OCI applications with an Oracle7 database. Steps 1 and 2 are required. Step 3 ensures that constraints present in Version 6 applications will be properly enabled when run on a Version 7 database. Step 4 is optional and allows you to take advantage of the new functionality offered with Oracle7 releases.

1. Make certain that any applications that use “old” OCI [HLI] calls (as documented in the *Programmer’s Guide to the Oracle Call Interface*) are either
   - recoded using the “new” OCI equivalents
   - relinked with a user–written interface layer to map “old” calls to “new” calls. This may be necessary, for example, where the source code of the original application is no longer available.

2. Relink your applications with Oracle7 of the OCI libraries, using either

   - deferred mode linking, to improve performance of applications
   - non–deferred mode, to maintain existing performance levels

   The OCI libraries are shipped with all Oracle7 releases.

   One implication of deferred linking is that bind and define errors may not be reported to the application immediately after bind and define operations, but may, instead, be reported later at the time of a describe, execute, or fetch call.

3. ENABLE all constraints after a newly migrated Version 6 application is run on a Version 7 database. When a Version 6 OCI application is run on a Version 7 database, the constraints are created, but disabled. For example, if a Version 6 OCI application containing a DDL statement such as

   ```
   ALTER TABLE gpd ADD (PRIMARY KEY (fd1) CONSTRAINT gp)
   ```

   is run on a Version 7 database, the constraint, *gp,* is created, but disabled. If a Version 7 OCI application, containing a similar statement (note the changed syntax) is run, such as

   ```
   ALTER TABLE gpd ADD (CONSTRAINT gp PRIMARY KEY (fd1))
   ```

   the constraint, *gp,* is ENABLED by DEFAULT. For more information, see *Oracle7 Server SQL Reference.*

4. (Optional) Recode your applications to use new calls. The new OBNDRA, ODESCR, OFLNG, ONBSET, ONBTST, ONBCLR, and OLOG calls allow you to take advantage of new features offered by Oracle7, and do not require any further recoding. The new OPARSE and OEXFET calls allow you to improve the performance of your applications. If you use OPARSE with the LNGFLG parameter set to V7, you must also change your code to reflect the differences between Version 6 and Oracle7, outlined previously.

OCI applications written with the non–blocking feature will work in backward mode with Release 7.0 and Release 7.1.

Migrating Your Oracle Forms Applications

This section contains the following topics:

- General Comments
- Taking Advantage of Oracle7 Functionality

General Comments

All Forms releases run with either Oracle, Version 6 or any Oracle7 release. You are not required to regenerate your applications to use them with a Release 7.x database. If you purchased a new Forms release (the newest Forms release is 4.5), install it after the installation of your new 7.x database is completed. Your Forms (Runform) applications run with no modification. You should, however, keep in mind the following issues:

- Forms applications running on an Oracle7 release are functionally identical to those applications running on Version 6. Guidelines for modifying Forms applications to take advantage of Oracle7 functionality are outlined later in this section.
- Forms 4.5 continues to use PL/SQL Version 1, resulting in the following restrictions on your datatypes:
  - NUMBER, DATE, BOOLEAN, and VARCHAR2 fields are supported, but VARCHAR2 fields are limited to 2 Kb.
  - No other PL/SQL Version 2 datatypes can be used in Forms triggers.

Taking Advantage of Oracle7 Functionality

Forms releases greater than 4.0 have been enhanced to allow support of stored procedures, functions, and packages (local and remote). Because Forms creates the access routine for each stored subprogram that your trigger or Forms PL/SQL code references, it is *not* possible to write a form containing calls to stored procedures or functions that will generate against both Oracle, Version 6 and an Oracle7 database.

Once a form includes a reference to at least one stored subprogram, it must be generated against an Oracle7 database. Otherwise, compilation errors result when the names of stored subprograms cannot be resolved by the PL/SQL compiler.

You should review the new features described in Appendices A, B, C, and D of this manual to determine if stored procedures or any of the other new Oracle7 features would be beneficial to your Forms applications. A complete description of how these Oracle7 features interact with Forms applications is provided in the *Oracle Forms 4.5 Reference Manual, Vol.1 and Vol. 2*, the *Oracle Forms 4.5 Developer's Guide* and *Forms 4.5 Advanced Techniques.*

Migrating Your CASE*Dictionary Applications

The DDL generator in CASE Dictionary V5.0.22 generates SQL DDL using Oracle, Version 6 syntax, which is slightly different from the syntax used by the Oracle7 Server. To account for this, SQL*Plus and Server Manager provide a setting to allow V6 syntax to be interpreted properly by the Oracle7 Server, including CHAR to VARCHAR2 translation. CASE*Dictionary 5.0 sets the mode automatically when these scripts are run from within the CASE*Dictionary environment. If you wish to execute your CASE–generated DDL scripts outside of CASE*Dictionary, you must first set your SQL*Plus or Server Manager session to V6 compatibility mode by issuing the following command:

```
SQL> SET COMPATIBILITY V6
```

To create true Oracle7 CHAR columns (fixed length, blank padded), the DDL scripts must be executed outside of the CASE environment without setting V6 compatibility mode. See *Oracle7 Server SQL Reference* for more information on Oracle, Version 6 compatibility mode.

Full support for the generation of DDL syntax and objects introduced in the Oracle7 Server will be available in CASE*Dictionary 5.1.

Migrating Your SQL*Report (RPT/RPF) Reports

Support for reports created using SQL*Report is maintained in Oracle7. Your reports should run against either Version 6 or Oracle7 of the database with no modifications. SQL*Report has not been upgraded to take advantage of any new Oracle7 functionality.

Migrating Your Oracle Reports and SQL*ReportWriter Applications

Support for Oracle Reports and SQL*ReportWriter is maintained in Oracle7 releases. Your reports should run against either Version 6 or Oracle7 databases with no modifications. Oracle7 functionality is supported in Oracle Reports, Version 2.

If you have user–owned Oracle Reports, Version 2 tables and wish to take advantage of the row–level locking available in standard Oracle7, you must create system–owned tables. To install the system tables, use the SQL*Plus script SRW_ICEN.SQL if you are using SQL*ReportWriter Version 1.1, or SRW_ICEN2.SQL if you are using Oracle Reports, Version 2. Then load your existing reports into the new tables.

Migrating Your SQL*Plus Scripts

This section contains the following topics:

- Set Compatibility Mode to V6
- Taking Advantage of Oracle7 Functionality

Set Compatibility Mode to V6

After you have migrated your database from Oracle, Version 6 to an Oracle7 release, you can run your SQL*Plus 3.0 scripts against the Oracle7 release and achieve the same output as if you were running against Version 6 by adding the following line as the first line of your script:

```
SET COMPATIBILITY V6
```

Alternatively, you could add this line to your LOGIN.SQL file.

Taking Advantage of Oracle7 Functionality

If you want SQL*Plus Release 3.1, Oracle7, and PL/SQL Version 2 functionality, you must complete these additional steps:

- You must make the following changes to your SQL*Plus 3.0 scripts to make them SQL*Plus 3.1 scripts:
  - If your scripts contain the line SET COMPATIBILITY V6, change it to SET COMPATIBILITY V7. Also check your LOGIN.SQL file and make this change as needed.
  - If your script relied on the DESCRIBE command's output, be aware that it now displays two new datatypes: ROWLABEL and VARCHAR2.
  - To learn about functionality new to SQL*Plus Release 3.1, refer to the *SQL*Plus User's Guide and Reference.*
- Change all SQL scripts to conform to Oracle7 syntax. Instructions for completing these modifications are described in the section "Change SQL Scripts" in Chapter 7 "Migrating from Version 6 to Version 7". For additional information on SQL, see *Oracle7 Server SQL Reference.*

- No changes to PL/SQL procedures are required.

SQL*Net

All Oracle7 releases use Version 2 of SQL*Net, which may have an impact on migrated applications. Several points are important.

- SQL*Net, Version 1 has been deprecated.
- Both the client and server must use the same version of SQL*Net.
- The Multi–Threaded Server requires SQL*Net, Version 2 at the server side. Therefore, if you want to connect using the Multi–Threaded Server, you must also use SQL*Net, Version 2 at the client side

Perform the following changes to upgrade from SQL*Net Version 1 to Version 2:

- Install SQL*Net, Version 2.
- Re–create each Version 1 connect string as a Version 2 connect descriptor. SQL*Net Version 2 relies on the new syntax outlined in the *SQL*Net V2.0 Administrator's Guide.*
- Relink any precompiler programs that you want to use with SQL*Net Version 2.

For complete instructions on upgrading SQL*Net from Version 1 to Version 2, refer to the *SQL*Net V2.0 Administrator's Guide* and the *SQL*Net Version 2 Migration Guide.*

CHAPTER

9

Upgrading and Downgrading between Oracle7 Releases

This chapter describes how you can upgrade and downgrade between Release 7.0, Release 7.1, Release 7.2, and Release 7.3. The simultaneous availability of different Oracle databases is a highly desirable feature for Oracle customers.

The topics included in this chapter are

- Upgrading to a New Release
- Downgrading to a Previous Release
- Downgrading from Release 7.3 to Release 7.1
- Upgrading and Downgrading by Copying Data
- Upgrading and Downgrading Considerations
- Enabling Release 7.1 Features
- Disabling Release 7.1 Features
- Enabling Release 7.2 Features
- Disabling Release 7.2 Features
- Enabling Release 7.3 Features
- Disabling Release 7.3 Features

- Downgrading from Release 7.2 to 7.1 and the UNRECOVERABLE Parameter
- Downgrading and Hash Clusters
- Downgrading PL/SQL Wrapper Code
- Downgrading after Using Resizeable Datafiles
- PL/SQL Compatibility: Upgrading, Downgrading, and Interoperability
- Compatibility and Migration for Sort Direct Writes
- Migration and Compatibility Issues for Object Groups
- Migration and Compatibility Issues for Synchronous Propagation
- Migration and Compatibility Issues for Sort Big Keys
- Migration and Compatibility Issues for the UNSAFE_NULL_FETCH Command
- Migration and Compatibility Issues for Sort Segment
- Upgrade and Downgrade Issues for Compiled Triggers
- Upgrade and Downgrade Issues for the REMOTE_DEPENDENCIES_MODE Parameter
- Migration and Compatibility Issues for Load Balancing in Listener
- Migration and Compatibility Issues for OBINDPS, ODEFINPS, OGETPI, and OSETPI Functions
- Migration and Compatibility Issues for Thread Safety, OCI
- Migration and Compatibility Issues for Thread Safety, Pro*
- Migration and Compatibility Issues for Fine Grained Locking
- Migration and Compatibility Issues for Buffer Cache LRU Latch Contention
- Upgrade and Downgrade Issues for Histograms
- Migration and Compatibility Issues for Standby Database
- Migration and Compatibility Issues for Direct Path Export
- Upgrading and the Advanced Replication Option
- Advanced Replication Compatibility Between Release 7.3 and Earlier Releases

The information contained in the sections “Upgrading to a New Release” and “Downgrading to a Previous Release” describes *generic* upgrading, downgrading, enabling, and disabling procedures. The sections, “Upgrading and Downgrading Considerations”, and the remaining sections in this chapter present procedures and warnings that must be followed to successfully upgrade, downgrade, or disable specific Release 7.2 features.

If you are upgrading or downgrading to Trusted Oracle, there are additional features of which you need to be aware. For more information, see the *Trusted Oracle7 Server Administrator’s Guide*.

For general, introductory comments about downgrading to a previous release, see Chapter 1, “Migration Overview”.

Upgrading to a New Release

Perform the following steps to upgrade your current Oracle7 database:

1. Execute the SHUTDOWN NORMAL command in Server Manager if you are currently using Oracle7, Release 7.1 or later. Execute SHUTDOWN NORMAL in SQL*DBA if you are currently using an Oracle release earlier than 7.1. This must be done for all instances, if you are running the parallel server.
2. Perform a full offline backup.
3. Install the new release.

OSDoc

Additional Information: Installation is operating system–specific. See your operating system–specific Oracle documentation and the Oracle7 README file for your operating system.

4. If you are upgrading to either Release 7.2 or Release 7.3, change the initialization parameter COMPATIBLE, in the INIT.ORA file, to either 7.2 or 7.3. If you are not upgrading to either Release 7.2 or Release 7.3, proceed directly to Step 5.
5. Relink your applications using the libraries for the new release.
6. Execute the CONNECT INTERNAL command.
7. Execute the STARTUP RESTRICT command in Server Manager.
8. Run the appropriate scripts listed in Table 9 – 1. Each script is an incremental upgrade. To upgrade from your current release to the latest release, start with the script for your release and run each incremental upgrade script until you reach the current release.

   Note: If you are using Trusted Oracle7, then you *must* run all scripts at a session label of DBLOW.

Upgrading From	To	Script to Run
7.0.11	7.0.12	CAT712.SQL
7.0.12	7.0.13	CAT713.SQL
7.0.13	7.0.14	CAT714.SQL
7.0.14	7.0.15	none
7.0.15	7.0.16	none
7.0.16	7.1.1	CAT70101.SQL
7.1.1	7.1.2	CAT70102.SQL
7.1.2	7.1.3	CAT7103.SQL
7.1.3	7.1.4	none
7.1.4	7.1.5	none
7.1.5	7.1.6	CAT7106.SQL
7.1.6	7.2.1	CAT7201.SQL
7.2.1	7.2.2	CAT7202.SQL
7.2.2	7.2.3	CAT7203.SQL
7.2.3	7.3.1	CAT7301.SQL
7.3.1	7.3.2	CAT7302.SQL

Table 9 – 1 Upgrading Scripts

9. Now run the following scripts:

   - CATALOG.SQL
   - CATPROC.SQL (run only if PL/SQL is installed)
   - CATPARR.SQL (run only if the Parallel Server option is installed)
   - CATREP.SQL (run only if the replication option is installed)

For example, to upgrade 7.0.16 to 7.1.4, (with PL/SQL, but no Parallel Server option), you run *only* the following scripts:

- CAT70101.SQL
- CAT70102.SQL
- CAT7103.SQL
- CATALOG.SQL
- CATPROC.SQL

Warning: Remember to use the scripts that are supplied with the release to which you are upgrading.

10. Check to see that the upgrade scripts ran successfully by spooling the results to a file. This may be especially important if you are planning to use the Advanced Replication option. The Advanced Replication option may require significant increase in the size of both allocated tablespace and SHARED_POOL_SIZE. (If SHARED_POOL_SIZE needs to be increased, return to Step 4 and make the necessary change in the INIT.ORA file.)

    Note: If you are using the Advanced Replication option, the Oracle Corporation recommends that you allocate 12 Mb of additional tablespace and 9 Mb to 12 Mb additional space for SHARED_POOL_SIZE.

11. Execute the ALTER SYSTEM DISABLE RESTRICTED SESSION command.

12. When upgrading from 7.0.x to 7.1.2 (or higher), each user who will use EXPLAIN PLAN must run the following script in their schema:

    - UTLXPLAN.SQL

Your database is now upgraded. See pages 9 – 10 and 9 – 12 for information about enabling the new features of the release to which you have now upgraded.

Downgrading to a Previous Release

After installing a new release, you can downgrade to a previous release by performing the following steps:

1. Execute the SHUTDOWN NORMAL command (for all instances, if running the Parallel Server).
2. Perform a full offline backup.
3. Execute the CONNECT INTERNAL command.
4. Execute the STARTUP RESTRICT command.
5. Disable certain features of the current release. See pages 9 – 11 and 9 – 13 for more information on disabling features.

6. Run the appropriate scripts listed in Table 9 – 2. Each script is an incremental downgrade. To downgrade from the latest release to your previous release, start with the script for the latest release and run each incremental downgrade script until you reach your previous release.

Downgrading From	To	Script to Run
7.3.2	7.3.1	CAT7302D.SQL
7.3.1	7.2.3	CAT7301D.SQL
7.2.3	7.2.2	None
7.2.2	7.2.1	None
7.2.1	7.1.6	None
7.1.6	7.1.5	None
7.1.5	7.1.4	None
7.1.4	7.1.3	CAT7102D.SQL
7.1.3	7.1.2	none
7.1.2	7.1.1	none
7.1.1	7.0.16	none
7.0.16	7.0.15	none
7.0.15	7.0,14	none
7.0.14	7.0.13	none
7.0.13	7.0.12	CAT712D.SQL
7.0.12	7.0.11	none

Table 9 – 2 Downgrading Scripts

For example, to downgrade 7.1.3 to 7.0.13, you only run the following script:

- CAT7102D.SQL

7. Execute the ALTER DATABASE RESET COMPATIBILITY command.
8. Execute the ALTER DATABASE BACKUP CONTROLFILE TO TRACE command. For more information about the ALTER DATABASE command, see *Oracle7 Server SQL Reference.*
9. Execute the SHUTDOWN NORMAL command (for all instances, if running the parallel server).
10. Install the release to which you wish to downgrade.

OSDoc

Additional Information: Installation is operating system–specific. See your operating system–specific Oracle documentation and the Oracle7 README file for your operating system.

11. Relink your applications using the libraries for the installed release.
12. Execute the CONNECT INTERNAL command.
13. Execute the STARTUP NOMOUNT RESTRICT command.
14. Execute the CREATE CONTROLFILE command to recreate the control file. For more information about the CREATE CONTROLFILE command, see *Oracle7 Server SQL Reference*.
15. Execute the ALTER DATABASE OPEN command.
16. Run the following scripts (at a session label of DBLOW for Trusted Oracle7):
    - CATALOG.SQL
    - CATPROC.SQL (run only if PL/SQL is installed)
    - CATPARR.SQL (run only if the Parallel Server option is installed)
    - CATREP.SQL (run only if the Replication option is installed)

Warning: Remember to use the scripts that are supplied with the release to which you are downgrading.

17. Execute the ALTER SYSTEM DISABLE RESTRICTED SESSION command.
18. Execute the SHUTDOWN NORMAL command (for all instances, if running the Parallel Server).
19. Restart the database without enabling the new features (set the COMPATIBLE initialization parameter to the release value to which you have downgraded).
20. Perform a full offline backup.

Your database is now downgraded to your previous release.

Downgrading from Release 7.3 to Release 7.1

The logfile and control file formats were changed in Release 7.2 and Release 7.3. Perform the following steps to ensure successful downgrading from Release 7.3 to Release 7.1:

1. Set the COMPATIBLE parameter in INIT.ORA to 7.3.
2. Open the Release 7.3 database.
3. Run the downgrade script CAT7301D.SQL (see Table 9 – 2).
4. Drop all currently active Release 7.2 and Release 7.3 features.
5. Execute the ALTER DATABASE RESET COMPATIBILITY command.
6. Execute the ALTER DATABASE BACKUP CONTROLFILE TO TRACE command.
7. Shutdown the database.
8. Set the COMPATIBLE parameter in INIT.ORA to 7.2.
9. STARTUP NOMOUNT and create the control file using the trace file obtained in Step 6.
10. Shutdown the database.
11. Set the COMPATIBLE parameter in INIT.ORA to 7.2.
12. Open the database.
13. Issue the following command to clear all but the current logfile:

    ```
    ALTER DATABASE CLEAR 'LOGFILE'
    ```

14. Shutdown the database.
15. Install the 7.1.x release to which you wish to downgrade.

Upgrading and Downgrading by Copying Data

The technique of copying data is useful if you want to

- defragment your data files
- restructure your database by creating or modifying tables or tablespaces
- migrate only certain database objects

For more information about copying data from one release to another, see “Copying Data” in Chapter 2.

Upgrading and Downgrading Considerations

This section contains the following topics:

- CAT70102.SQL and the Parallel Query Option
- ORA_TQ_BASE$ and the Parallel Query Option
- CATSVRMG.SQL and Server Manager

CAT70102.SQL and the Parallel Query Option

If you have the 7.1.1 parallel query option and you have already run queries in parallel against a given database, you will receive an error when running CAT70102.SQL on that database because the sequence will already exist. In this case, the error can be ignored.

ORA_TQ_BASE$ and the Parallel Query Option

The ORA_TQ_BASE$ sequence, introduced in 7.1.2, is required if you use the Parallel Query option. The sequence is created when any of the following scripts are run:

- SQL.BSQ (automatically run as part of database creation)
- MIGRATE.BSQ (when migrating from Oracle Version 6)
- CAT70102.SQL (when upgrading Oracle7 to 7.1.3 and installing Server Manager views)

CATSVRMG.SQL and Server Manager

In 7.1.2, two new SQL scripts were added for the Server Manager product. CATSVRMG.SQL is automatically run during database creation by CATALOG.SQL. This script installs several views and one public synonym (SM$VERSION) required by Server Manager to administer the database.

CATNOSVM.SQL is a script that DROPs all objects created by CATSVRMG.SQL, de–installing the Server Manager.

Enabling Release 7.1 Features

After upgrading to Oracle7, Release 7.1, set the COMPATIBLE parameter in the initialization parameter file to 7.1.0 to enable the new Release 7.1 features. Set this parameter *before* starting up the database.

The COMPATIBLE parameter must be set if you want to create read–only tablespaces, or if you want to create multiple triggers of the same type on a single table.

```
COMPATIBLE = 7.1.0
```

Disabling Release 7.1 Features

This section describes how to disable the new features available with Oracle7, Release 7.1.

Before downgrading to an earlier release, you must disable certain features associated with Release 7.1. To disable the new features associated with Oracle7 Release 7.1, complete the following steps:

1. Determine whether the new 7.1 features are in use. Check the following views for use of their corresponding feature:

View	Feature
DBA_TABLESPACES	read–only tablespaces
DBA_TRIGGERS	multiple same–type triggers per table

Table 9 – 3 Features to Disable Before Downgrading

The following SQL statements are examples of queries that can determine which Release 7.1 features have been enabled:

```
SELECT tablespace_name from DBA_TABLESPACES
   WHERE status = 'READ ONLY';
SELECT table_name, trigger_name,
     trigger_type||' '||triggering_event AS type_name
   FROM ALL_TRIGGERS
   WHERE (table_name,trigger_type,triggering_event) =
     (SELECT table_name,trigger_type,triggering_event
        FROM ALL_TRIGGERS
      GROUP BY table_name, trigger_type,triggering_event
        HAVING count(*) > 1)
   ORDER BY table_name, type_name;
```

2. Drop or change any Release 7.1 features so that they are no longer in use.

   - Change the READ ONLY tablespaces to READ WRITE. For example, to make the read–only FLIGHTS tablespace writeable again, you would issue the following command:

     ```
     ALTER TABLESPACE flights READ WRITE;
     ```

   - Ensure that there is only one trigger of each type per table by either merging multiple triggers of the same type into one trigger or deleting the extra triggers.
   - Reset any new initialization parameters to their default values, and remove any changes that you have implemented to take advantage of these parameters.

For example, if you are using the new database connection security features, you must set the ORA_ENCRYPT_LOGIN environment variable and DBLINK_ENCRYPT_LOGIN initialization parameter to FALSE. You must also set the REMOTE_LOGIN_PASSWORDFILE initialization parameter to NONE and return to using CONNECT INTERNAL to perform database administration.

If you have begun using the new SQL syntax, you must set the SQL92_SECURITY initialization parameter to FALSE and remove this new syntax from your SQL statements. Any code written to take advantage of the DBMS_SQL package must also be removed. Any snapshot refresh groups that you have created will be ignored after you downgrade to Release 7.0.

If you are using the parallel query option, you must alter any table definitions to exclude the parallel clause. Also, you must remove any parallel query initialization parameters (for example, PARALLEL_MIN_SERVERS) from your parameter file.

3. Make appropriate changes to any PL/SQL code that uses features introduced with Release 7.1 before downgrading to an earlier release.

   Note: If you used the RESTRICT_REFERENCES pragma in your 7.1 applications, you must remove all calls to user–defined, stored functions before downgrading from Release 7.1 to Release 7.0.

4. Issue the following command:

   ```
   ALTER DATABASE RESET COMPATIBILITY
   ```

5. Shut down the database.

6. Restart the database without enabling the new features (set the COMPATIBLE parameter to 7.0.12 or above, excluding 7.1.0) and install an earlier release of Oracle (Release 7.0).

See page 9 – 6 for information on downgrading to an earlier release.

Enabling Release 7.2 Features

The COMPATIBLE parameter for Release 7.2 must be set to 7.2.0 in the initialization parameter file to permit applications created under a pre–Release 7.2 database to make use of new Release 7.2 features.

Disabling Release 7.2 Features

When downgrading from Release 7.2 to an earlier release, the COMPATIBLE parameter must be set to a value less than 7.2 *before the Release 7.2 database is shut down***.**

> **Note:** Prior releases did not require that the COMPATIBLE parameter be set before shutting down the database.

The following steps should be followed in preparation for downgrading from Release 7.2 to an earlier release.

1. Open the Release 7.2 database.
2. Query V$COMPATIBILITY to see which Release 7.2 features are currently active.
3. Drop or change all currently active Release 7.2 features.
4. Execute the following command:

   ```
   ALTER DATABASE RESET COMPATIBILITY
   ```

5. Set the value of the COMPATIBLE parameter in the initialization parameter file to a value less than 7.2. For more information on the COMPATIBLE parameter, see *Oracle7 Server Reference.*
6. Open the database to which you wish to downgrade. This will succeed if none of the Release 7.2 features are in use; otherwise, it will fail.
7. Clear all log files, except those that are current, by executing the following command:

   ```
   ALTER DATABASE CLEAR LOGFILE <log_name> [<log_name>,...]
   ```

8. Make appropriate changes to any PL/SQL code that uses features introduced with Release 7.2.

   > **Note:** If you used cursor variables in Release 7.2 applications, you must remove all references to cursor variables or change the references to static cursors before downgrading to an earlier release.

9. Cleanly close the 7.2 database.

Enabling Release 7.3 Features

The COMPATIBLE parameter for Release 7.3 must be set to 7.3.0 in the initialization parameter file to permit applications created under a pre–Release 7.3 database to make use of new Release 7.3 features.

Disabling Release 7.3 Features

If you mount the database with the COMPATIBLE initialization parameter set to COMPATIBLE = 7.3.0 the control file is marked as a 7.3 control file and cannot be used by an earlier release. If you wish to run an application that was previously developed under a pre–7.3 release against your 7.3 database, compatibility must be reset and the control file must be recreated. Perform the following steps:

1. Reset the compatibility by executing

   ```
   ALTER DATABASE RESET COMPATIBILITY
   ```

2. Recreate the control file by executing

   ```
   CREATE CONTROLFILE
   ```

For more information about the ALTER DATABASE RESET COMPATIBILITY and CREATE CONTROLFILE commands, see *Oracle7 Server SQL Reference.*

Downgrading from Release 7.2 to 7.1 and the UNRECOVERABLE Parameter

A Release 7.1 database can be recovered from Release 7.2 redo logs that were generated using UNRECOVERABLE table creation in a Release 7.2 database. The redo log information that is generated when the UNRECOVERABLE parameter is used remains compatible with Release 7.1. Downgrading (backward migration) from Release 7.2 to Release 7.1 is, thereby, allowed.

For more information, see *Oracle7 Server SQL Reference* and the *Oracle7 Server Administrator's Guide.*

Downgrading and Hash Clusters

Release 7.2 provides you with improved control over the creation of hash clusters. You can now specify an application–specific SQL expression, with some restrictions, as the hash function for a cluster. Choice of appropriate hash functions reduces collisions, resulting in better performance.

> **Note:** *Hash clusters created under Release 7.2 become inaccessible if the database is downgraded to a pre–Release 7.2 database*. The solution to this problem is to drop the hash cluster first and then downgrade.

For more information, see *Oracle7 Server Concepts.*

Downgrading PL/SQL Wrapper Code

The PL/SQL Wrapper feature consists of two components:

- The PL/SQL Wrapper Compiler, which is a standalone utility that compiles PL/SQL source code files to files in PL/SQL Wrapper format.
- The PL/SQL Wrapper Loader, which is an extension to the PL/SQL compiler that enables the recognition and loading of PL/SQL compilation units in *PL/SQL Wrapper format*.

PL/SQL Wrapper offers the same portability and flexibility as the PL/SQL source code format. In particular, PL/SQL Wrapper provides

- platform independence. Developers do not have to deliver multiple versions of the same unit.
- dynamic loading. You can add new features without having to shut down and relink.
- dynamic (load time) binding of unqualified external references.
- automatic recompilation when depended–on units change specification.

- Code developed using PL/SQL Wrapper cannot be downgraded to a pre–Release 7.2 database. Therefore, to preserve the original PL/SQL source code, execute the following command:

```
WRAP INAME=/mydir/myfile.sql ONAME=mydir/myfile/myfile.plb
```

where INAME specifies the path and name of the original PL/SQL source code and ONAME specifies the path and name of the Wrapper output file.

For more information, see the *PL/SQL User's Guide and Reference.*

Downgrading after Using Resizeable Datafiles

There are several steps that must be performed to ensure that the downgrade process will succeed in returning files for correct operation with a pre–Release 7.2 database.

- If you anticipate the need to downgrade to an earlier release or version, *it is recommended that a backup be taken of the Release 7.x database before using Resizeable Datafiles.*
- You must perform the nine steps shown on page 9 – 13 under "Disabling Release 7.2 Features" before attempting to downgrade after having used the Resizeable Datafiles feature.
- Query the FILEXT$ table to determine which files are currently in the AUTOEXTEND ON mode. (If FILEXT$ does not exist, no datafiles are in the AUTOEXTEND ON mode.)
- Disable the automatic extension feature for each file in AUTOEXTEND ON mode. For example, the following command disables automatic extension for the datafile *filename2*

```
ALTER DATABASE DATAFILE 'filename2'
AUTOEXTEND OFF
```

- Return the size of all files whose size was changed by the AUTOEXTEND ON command to their original creation size, in other words, to the size that the files had before the AUTOEXTEND ON command was executed. The creation size can be found from the CREATE_BYTES column in V$DATAFILE. The CREATE_BYTES and BYTES columns must be equal for all datafiles in V$DATAFILE before the downgrading process is started.

Warning: The size of the files to be downgraded must match *exactly* their original creation size.

- Finally, execute the following command:

```
ALTER DATABASE RESET COMPATIBILITY
```

Note: Once a resize operation is complete, earlier versions of the database cannot read the redo log file for changes that occurred after the point of extension. For example, if a database is resized from 10 Mb to 20 Mb, an earlier version or release of the database will not be able to read the redo log file for changes that occurred in the 10–to–20 Mb extension.

For more information, see the *Oracle7 Server Administrator's Guide* and *Oracle7 Server SQL Reference.*

PL/SQL Compatibility: Upgrading, Downgrading, and Interoperability

This section contains the following topics:

- Upgrading to PL/SQL, Release 2.3
- Downgrading from PL/SQL, Release 2.3
- Interoperability: RPC between PL/SQL, Release 2.2 and PL/SQL, Release 2.3

Upgrading to PL/SQL, Release 2.3

All features that work in PL/SQL, Release 2.2 will continue to work in Release 2.3.

All PL/SQL stored procedures are invalidated in an upgrade from Release 2.2 to Release 2.3, but are automatically recompiled upon the first execution thereafter.

You will have to upgrade to PL/SQL, Release 2.3 to take advantage of the following features:

- FETCH from cursor variable in PL/SQL
- ability to use cursor variables in client–side PL/SQL packages and procedures
- ability to invoke server–side PL/SQL procedures with cursor variables from client–side PL/SQL
- self–contained execution, in the server, of PL/SQL procedures containing cursor variables without reliance on host language bind variables
- weak REF CURSOR types
- REMOTE_DEPENDENCIES_MODE parameter to take advantage of the SIGNATURE mode

- PL/SQL tables of records
- PL/SQL table operations

For new applications, PL/SQL, Release 2.3 requires only IN mode (IN OUT is still permitted) of cursor variable procedure parameters that are used solely in FETCH, CLOSE, or as a source of assignment.

Downgrading from PL/SQL, Release 2.3

All PL/SQL stored procedures are invalidated in a downgrade from Release 2.3, but are automatically recompiled upon the first execution thereafter.

Warning: Programs using features that are new in PL/SQL, Release 2.3 will compile with errors after a downgrade.

Interoperability: RPC between PL/SQL, Release 2.2 and PL/SQL, Release 2.3

All remote procedure calls (RPC) currently supported in the first origin–to–destination configuration shown in Table 9 – 4 are supported in the remaining three origin–to–destination configurations.

Origin	Destination
PL/SQL Release 2.2 client	PL/SQL Release 2.2 server
PL/SQL Release 2.2 client	PL/SQL Release 2.3 server
PL/SQL Release 2.3 client	PL/SQL Release 2.2 server
PL/SQL Release 2.3 client	PL/SQL Release 2.3 server

Table 9 – 4 Remote Procedure Calls from Client–to–Server

All remote procedure calls (RPC) currently supported in the first origin–to–destination configuration shown in Table 9 – 5 are supported in the remaining three origin–to–destination configurations.

Origin	Destination
PL/SQL Release 2.2 server	PL/SQL Release 2.2 server
PL/SQL Release 2.2 server	PL/SQL Release 2.3 server
PL/SQL Release 2.3 server	PL/SQL Release 2.2 server
PL/SQL Release 2.3 server	PL/SQL Release 2.3 server

Table 9 – 5 Remote Procedure Calls from Server–to–Server

Both the Release 2.2 and the Release 2.3 compilers reject server–to–server calls to remote procedures having parameters of ref cursor types.

Client–to–server calls to stored procedures having parameters of ref cursor types are supported only with PL/SQL, Release 2.3 on both client and server.

Remote procedure calls from Release 2.2 client–side PL/SQL to stored procedures having parameters of ref cursor types are not supported, regardless of whether the server on which the stored procedure exists is a Release 7.2 or a Release 7.3 server. Client–side applications that attempt such calls could have undefined behavior. Furthermore, the Release 7.2 server could exhibit undefined behavior at runtime; the Release 7.3 server rejects such calls at runtime.

For more information about PL/SQL, see the *PL/SQL User's Guide and Reference.*

Compatibility and Migration for Sort Direct Writes

Users who upgrade to Release 7.3 will get SORT_DIRECT_WRITES in AUTO mode by default. Because the direct writes use large buffers (typically 32 Kb to 64 Kb), the space map function in the sort splits extents into buffer–sized chunks to exploit large multi–block writes. The non–direct write case requires only 4 Kb. This change in space allocation may result in a 10% to 15% increase in temporary space usage.

For more information about *Sort Direct Writes*, see *Oracle7 Server Reference*, the *Oracle7 Server Administrator's Guide*, Appendix D, "Summary of Changes in Oracle7, Release 7.3", and *Oracle7 Server Tuning*.

Migration and Compatibility Issues for Object Groups

This section contains the following topics:

- Modifications to Deferred RPC Calls for Object Groups
- Modifications to Deferred RPC Tables
- Modifications to Deferred RPC Views
- Modifications to Deferred RPC API
- Changed Semantics
- Snapshot Sites
- RepCat API Compatibility with Release 7.2
- Catalog Compatibility
- REP$WHAT AM I

- Migration to Object Groups
- Downgrading to Repschemas
- Interoperability

Modifications to Deferred RPC Calls for Object Groups

Deferred RPCs are altered in Release 7.3 to remain compatible with Release 7.2. Table 9 – 6 shows the RepCat tables that affect deferred RPCs:

Table	Comments
REPCAT$_REPSCHEMA	The shape of REPCAT$_REPSCHEMA remains the same as it was in Release 7.2, but the value in the sname column is interpreted as the object group name instead of the schema name. An object's group name may or may not be the same as its schema name. However, the shape of REPCAT$_REPPROP, as well as the semantics of its columns, remains unchanged. Therefore, when comparing with REPCAT$REPSCHEMA.SNAME, a group name must be used. When comparing with REPCAT$_REPPROP, a schema name must be used.
REPCAT$_REPPROP	For deferred RPCs to be compatible with object groups, queries involving REPCAT$_REPPROP and REPCAT$_REPSCHEMA found in deferrer RPC code must be altered to satisfy the requirements shown above for REPCAT$_REPSCHEMA.

Table 9 – 6 General Table Modifications for Deferred RPC Calls

Modifications to Deferred RPC Tables

Table 9 – 7 shows the changes to DEF$ CALL in Release 7.3:

Column Name	Type	Constraints	Comments
DEFERRED_TRAN_DB	VARCHAR2(128)	primary$_1$	
DEFERRED_TRAN_ID	VARCHAR2(22)	primary$_2$	
BUFFER_NUMBER	NUMBER	primary$_3$	
CALLNO	NUMBER		
SCHEMANANE	VARCHAR2(30)		
GROUPNAME	VARCHAR2(30)		new column
PACKAGENAME	VARCHAR2(30)		
PROCNAME	VARCHAR2(30)		
ARGCOUNT	NUMBER		
PARM_BUFFER	LONG RAW		
DESTINATION_LIST	CHAR(1)		
ORIGIN_USER_ID	NUMBER		
ORIGIN_USER	VARCHAR2(30)		
DELIVERY_ORDER	NUMBER		
ORIGIN_TRAN_ID	VARCHAR2(22)		
ORIGIN_TRAN_DB	VARCHAR2(128)		
START_TIME	DATE		
DESTINATION_COUNT	INTEGER		
COMMIT_COMMENT	VARCHAR2(50)		

Table 9 – 7 Changes to the DEF$ CALL Table

Modifications to Deferred RPC Views

Only the view, DEFCALL, changes to reflect the new column added to DEF$_CALL in Release 7.3.

Modifications to Deferred RPC API

By adding the new parameter, GNAME, to the end of the following procedures, RPCs become aware of the object groups while remaining compatible with Release 7.2. Current triggers must be altered to include the group name parameter to the procedure call().

- ADD_CALL
- CALL (both)

The following package variable is added to DBMS_DEFER:

CURRENT_CALL_GROUP_NAME VARCHAR2(30)

Changed Semantics

The following semantic changes should be followed:

- Use PL/SQL APIs to create object groups, in addition to replicated schemas. Once the object groups have been created, use the APIs to add objects to them
- Schemas are specified only when adding objects to objects groups.
- Schemas specify object ownership and are not used for replicating groups of objects, as in Release 7.2.
- Use the PL/SQL APIs to replicate object groups, instead of schemas, to databases.
- Specify REPCATLOG operations using object groups instead of schemas.

Snapshot Sites

For snapshot sites, an object group is equivalent to a snapshot refresh group. Release 7.3 offers a parameter to the procedure CREATE_SNAPSHOT_REPGROUP(...) to automatically create a refresh group for the snapshot REPOBJECT group.

RepCat API Compatibility with Release 7.2

Complete forward compatibility exists between Release 7.2. and Release 7.3 applications. In other words, Release 7.2 API are extended to accommodate objects groups in a compatible fashion. However, be certain to disable the Object Groups feature in all applications developed under Release 7.3 before you attempt to downgrade to Release 7.2.

Catalog Compatibility

To accommodate existing Release 7.2 environments, Release 7.3 preserves all existing columns in all base tables and views. The GNAME column is added to appropriate tables and view.

REP$WHAT AM I

REP$WHAT AM I is a package that stores the type, master or snapshot, for each replicated schema in Release 7.2. Since object groups may contain objects from more than one schema, this information is stored at the object level in Release 7.3. For each replicated table whose schema name and object name are different, the package TABLE_NAME$TP is generated. Compatibility is preserved by generating REP$WHAT AM I in the case where the schema name is the same as the group name.

Migration to Object Groups

The upgrade script, CAT7301.SQL will convert an existing Release 7.2 environment to use object groups. Each repschema is converted to an object group with the same name. All existing procedure wrappers, triggers, and packages remain unchanged.

Downgrading to Repschemas

The downgrade script, CAT7301D.SQL will replace object groups with repschemas if the existing Release 7.3 environment is compatible with Release 7.2. A repschema is created if the group name is the same as the schema name of one or more objects in the object group. All objects that cannot be converted to a Release 7.2 environment are removed from the replication environment. Removed objects are not dropped from the database, but are removed from the replication catalog. However, generated objects (procedure wrappers, triggers, and packages) for objects that are no longer replicated, are dropped from the database.

Interoperability

As long as each object group is actually a repschema, a master site can be running either Release 7.2 or Release 7.3, and a snapshot can be running either Release 7.2 or Release 7.3.

For more information about Object Groups, see *Oracle7 Server Distributed Systems, Volume II* and Appendix D, "Summary of Changes in Oracle7, Release 7.3".

Migration and Compatibility Issues for Synchronous Propagation

This section contains the following topics:

- Creating a N–Way Master Configuration
- Adding a Snapshot
- Semantics for Release 7.3 Snapshot Sites
- Downgrading to Release 7.2
- Interoperability

Creating a N–Way Master Configuration

Execute the following six steps to create a N–way master configuration:

1. Create the master object group at the master site.
2. Add master objects to the object group.
3. Add remote replication sites.
4. Generate replication support.
5. Alter the propagation method using the three, new procedures.
6. Unquiesce environments and begin replication.

Adding a Snapshot

Execute the following three steps to create a snapshot site.

1. Create the snapshot object group at the snapshot site site.
2. Add snapshot objects to the object group.
3. Alter the propagation method using the three, new procedures. The default propagation method for each updatable snapshot to its master is asynchronous.

Semantics for Release 7.3 Snapshot Sites

You should declare conflict resolution on the snapshot's master table for each updatable snapshot. Although this is not required, conflict resolution is highly recommended since communication between snapshot and master may be asynchronous. Asynchronous snapshot configurations may cause conflicts.

Downgrading to Release 7.2

A downgrade script is provided to restore asynchronous propagation to all objects at all sites. The method of propagation to all other destinations is changed to asynchronous, as necessary, for each object at all replication sites.

Interoperability

The method of propagation among sites in a replication environment is transparent to applications as well as users. Site A can synchronously replicate an object, FOO, to site B, while site B can asynchronously replicate an object bar to site X. But the method of propagation for each replicated object must be symmetric between any two masters. Thus, an object synchronously replicated from site A to site B implies that the same object is synchronously replicated from site B to site A. Configurations that are not globally synchronous do not gain the full advantages of synchronous replication, for example, there is a finite probability of conflicts.

For more information about *Synchronous Propagation*, see *Oracle7 Server Distributed Systems, Volume II* and Appendix D, "Summary of Changes in Oracle7, Release 7.3".

Migration and Compatibility Issues for Sort Big Keys

Users may transparently migrate old applications to Release 7.3 applications that contain the *Sort Big Keys* feature. Certain queries that previously generated error message ORA–01467, “sort key too long”, will now work. Sorts that use VARCHAR2(2000) key columns will show slight performance degradation due to increased checks for buffer fragmentation. This performance problem can be alleviated by setting the SORT_DIRECT_WRITE initialization parameter.

For more information about the *Sort Big Keys* feature, see the *Oracle7 Server Administrator’s Guide*, *Oracle7 Server SQL Reference*, *Oracle7 Server Tuning*, and Appendix D, “Summary of Changes in Oracle7, Release 7.3”.

Migration and Compatibility Issues for the UNSAFE_NULL_FETCH Command

The UNSAFE–NULL_FETCH command line option is intended to ease migration from Oracle, Version 6 to Oracle7. Users upgrading to Oracle7, who use the precompiler command line option DBMS=V6, will be unaffected by the UNSAFE_NULL_FETCH option because precompiler application using DBMS=V6 maintain full compatibility with Oracle, Version 6.

Users upgrading to Oracle7, who use precompiler command line option DBMS=V7, will encounter new Oracle7 behavior that is different from Oracle, Version 6 behavior. In most cases, the Oracle, Version 6 to Oracle7 change in behavior requires minimal modification of Pro* applications to accommodate the change. Large modification to Pro* applications may be required to handle the consequence of a null value that is FETCHed into a host variable that does not have indicator variables. Using UNSAFE_NULL_FETCH=YES with DBMS=V7 restores the Version 6 behavior for handling FETCHes of null values.

FETCHing null values into host variable that do not have indicator variables is very undesirable. UNSAFE_NULL_FETCH=YES is intended only to allow users a *grace* period during which they can adapt their Pro* applications to the new Oracle7 behavior on null values.

> **Note:** Users are advised to complete the migration to Oracle7 before the release of Oracle8. Precompiler option UNSAFE_NULL_FETCH will be obsoleted at some future time and use of UNSAFE_NULL_FETCH=YES will generate a precompile time error.

For more information about *UNSAFE_NULL_FETCH*, see the *Programmer's Guide to the Pro*Ada Precompiler*, *Pro*COBOL Supplement to the Oracle Precompilers Guide*, *Programmer's Guide to the Oracle Pro*C/C++ Precompiler*, and *Oracle7 Server Tuning*, and Appendix D, "Summary of Changes in Oracle7, Release 7.3".

Migration and Compatibility Issues for Sort Segment

In order to use *Sort Segment*, the COMPATIBLE parameter must be set to 7.3.0.0 or higher. Also, it is not possible to move the database to a pre–7.3 release if there are temporary tablespaces. The temporary tablespaces have to be converted to permanent (or offline) state before any backward migration procedure is attempted.

For more information about *Sort Segment*, see *Oracle7 Parallel Server Concepts & Administration*, *Oracle7 Server Tuning*, the *Oracle7 Server Administrator's Guide*, and Appendix D, "Summary of Changes in Oracle7, Release 7.3".

Upgrade and Downgrade Issues for Compiled Triggers

The following conditions now hold for compiled triggers:

- Non–stored triggers cannot run under Oracle7 releases that support stored triggers. If You must recompile all existing triggers with the ALTER TRIGGER COMPILE command if you are upgrading from a non–stored trigger release to a stored trigger release. This is performed by running an upgrade script.
- Downgrading from a stored trigger release to a non–stored trigger release can be safely done, since all dictionary columns required to support non–stored triggers are maintained in the stored trigger release. The non–stored trigger release does not attempt to search for pcode or dependency information for triggers. Therefore, information generated under the stored trigger release is ignored.

- If a downgrade is done from a stored trigger release to a non–stored trigger release, and is then followed by an upgrade to a stored trigger release, you must delete any stale pcode and/or dependency information generated before the downgrade. Running the ALTER TRIGGER COMPILE command at upgrade time automatically deletes information for triggers that have not been dropped or recreated. However, there may be some data associated with objects that no longer exist, since the non–stored trigger release does not delete the data. To clear such data, the upgrade script cleans out the DEPENDENCY$ table and the IDL$ table of data associated with non–existent objects.

For more information about *Compiled Triggers*, see *Oracle7 Server SQL Reference, Oracle7 Parallel Server Concepts & Administration*, the *Oracle7 Server Application Developer's Guide*, and *Oracle7 Server Administrator's Guide*, and Appendix D, "Summary of Changes in Oracle7, Release 7.3".

Upgrade and Downgrade Issues for the REMOTE_DEPENDENCIES_MODE Parameter

The following suggestions may be useful when using the REMOTE_DEPENDENCIES_MODE parameter:

Suggestion: Client–side users of PL/SQL, Release 2.3 must use REMOTE_DEPENDENCIES_MODE=SIGNATURE to talk to stored procedures and install their applications at client sites; this should be hard–coded at the time of connection to the database using UPI calls.

Suggestion: Server –side users of PL/SQL, Release 2.3 can ignore the REMOTE_DEPENDENCIES_MODE parameter or set it explicitly to REMOTE_DEPENDENCIES_MODE=TIMEST AMP to continue getting Release 7.2 behavior.

Suggestion: If you are a server–side user of PL/SQL, Rel;ease 2.3 and wish to avoid certain types of invalidations, such as the addition of a new procedure at the end of a package, choose the SIGNATURE mode.

For more information about *Remote Dependencies in a PL/SQL Environment*, see *Oracle7 Server Application Developer's Guide*, *Oracle7 Server SQL Reference, PL/SQL User's Guide and Reference*, and Appendix D, "Summary of Changes in Oracle7, Release 7.3".

Migration and Compatibility Issues for Load Balancing in Listener

Pre–7.3 dispatchers cannot contact multiple listeners with the new initialization parameter, MTS_LISTENER_ADDRESS. Therefore, if you wish to use load balancing, you must upgrade to SQL*NET, Release 2.3 and Oracle7, Release 7.3.

For more information about *Load Balancing in Listener*, see *Oracle7 Parallel Server Concepts & Administration* and Appendix D, "Summary of Changes in Oracle7, Release 7.3".

Migration and Compatibility Issues for the OBINDPS, ODEFINPS, OGETPI, and OSETPI Functions

The functions OBINDPS, ODEFINPS, OGETPI, and OSETPI are compatible only with Release 7.3 servers and beyond. If a Release 7.3 application uses one of these calls against a Release 7.2, or earlier, release, error ORA–01551 may be generated. If this happens, you must restart the execution.

For more information about *Piecewise Binds and Defines for String and Raw Data*, see the *Programmer's Guide to the Oracle Call Interface* and Appendix D, "Summary of Changes in Oracle7, Release 7.3".

Migration and Compatibility Issues for Thread Safety, OCI

A new call, OPINIT has been added for handling multi–threading. All other OCI calls remain the same.

For backward compatibility, single–threaded applications will work even if they do not issue the OPINIT call.

If you specify a single–threaded environment in the OPINIT call, or the OPINIT call is skipped, there is no performance overhead for using the multi–threaded version of the OCI library.

The ORLON and OLON call will be supported until Version 8. However, you should use OLOG, even for single–threaded applications.

> **Note:** The OLOG call is required for multi–threaded applications.

For more information about *Thread Safety, OCI*, see the *Programmer's Guide to the Oracle Call Interface* and Appendix D, "Summary of Changes in Oracle7, Release 7.3".

Migration and Compatibility Issues for Thread Safety, Pro*

A new parameter, the runtime context pointer, has been added to the generated SQLLIB calls. The existing entry points to SQLLIB remain unchanged, so you can simply re–link your applications after upgrading. If you precompile your applications with the THREAD=YES option, you will not be able to link against older versions of SQLLIB.

For more information about *Thread Safety, Pro**, see the *Programmer's Guide to the Pro*Ada Precompiler*, the *Pro*COBOL Supplement to the Oracle Precompilers Guide*, the *Programmer's Guide to the Oracle Pro*C/C++ Precompiler*, and Appendix D, "Summary of Changes in Oracle7, Release 7.3".

Migration and Compatibility Issues for Fine Grained Locking

This section contains the following topics:

- Catalog Views
- Interoperability

Catalog Views

The FILE_LOCK view has been renamed. The correct name is V$FILE_LOCK. For releasable locks, the starting value must now be starting lock resource name.

V$CACHE_LOCK: The index value no longer indicates a DLM lock name.

Interoperability

The new locking implementation works correctly in mixed environments with old software installed on other members of the cluster. However, all nodes in the cluster must be running the new code in order to use *Fine Grained Locking*. Therefore, you must perform a brief shutdown of all nodes in the cluster to change the initialization parameters for the locking configuration.

Upgrading the operating system lock manager may also require brief down time. If a new operating system is required to support the recovery OSD interface, you must shutdown before attempting to use *Fine Grained Locking*.

For more information about *Fine Grained Locking*, see *Oracle7 Parallel Server Concepts & Administration* and Appendix D, “Summary of Changes in Oracle7, Release 7.3”.

Migration and Compatibility Issues for Buffer Cache LRU Latch Contention

Changes required by *LRU Latch Scalability* are

- A new initialization parameter, DB_BLOCK_LRU_LATCHES, configures the buffer cache. DB_BLOCK_LRU_LATCHES specifies an advisory upper bound value for the desired number of sets.
- The number of sets used by the instance as a new field in now exported in the V$PARAMETER view. Note that the number of sets displayed is the number of sets used by the system and may not be the same as the value requested by the DB_BLOCK_LRU_LATCHES parameter.

For more information about *LRU Latch Scalability*, see *Oracle7 Server Tuning*, *Oracle7 Parallel Server Concepts & Administration*, and Appendix D, “Summary of Changes in Oracle7, Release 7.3”.

Upgrade and Downgrade Issues for Histograms

The ANALYZE command and the cost–based optimizer will not work unless the proper upgrade and downgrade procedures are followed. To upgrade from Release 7.2 to Release 7.3 you must run the script, CAT7301.SQL. To downgrade from Release 7.3 to Release 7.2, you must run the script, CAT7301D.SQL.

The upgrade script creates new data dictionary tables and catalog views for histograms. The downgrade script deletes these tables and updates the appropriate catalog views.

For more information about *Histograms*, see *Oracle7 Server Tuning* and Appendix D, “Summary of Changes in Oracle7, Release 7.3”.

Migration and Compatibility Issues for Standby Database

Standby Database will only operate on Oracle7, Release 7.3 or higher. The primary and standby databases must be running the same version and release number of the Oracle7 Server.

OSDoc

Additional Information: The primary and standby database must be running the same version and release of the operating system platform. For more information, see your operating system–specific Oracle documentation.

The following comments will aid you in using *Standby Database* correctly:

- The COMPATIBLE initialization parameter must be the same between both the primary and standby databases.
- The data files, log files, and control files of the primary and standby databases must exist on separate physical media. It is not possible, for example, to use the same control file for both the primary and standby databases.
- It is recommended, but not required, that the database identification string be the same, and the data files, log files, and control files have the same names on both the primary and standby databases.

For more information about *Standby Database*, see *Oracle7 Server SQL Reference*, the *Oracle7 Server Administrator's Guide*, *Oracle7 Parallel Server Concepts & Administration*, and Appendix D, "Summary of Changes in Oracle7, Release 7.3".

Migration and Compatibility Issues for Direct Path Export

Direct Path Export uses an encoding format that is different from the encoding format used by the conventional path. Therefore, the Export dump files generated by *Direct Path Export* and the conventional path Export are different.

Dump files generated by both *Direct Path Export* and the conventional path are usable by the Release 7.3 Import utility.

Direct Path Export has the same upward compatibility as the conventional path Export with all future Oracle Server versions and releases. Pre–Release 7.3 dump files are upward compatible with the Release 7.3 Import utility.

Export dump files generated with the conventional path are downward compatible with pre–Release 7.3 versions and releases of the Import utility. However, Export dump files generated with *Direct Path Export* are not downward compatible with pre–Release 7.3 versions and releases of the Import utility. For example, Export dump files generated with *Direct Path Export* are not compatible with the Release 7.2 Import utility. If backward compatibility is important for your operation, run Export in the conventional mode to obtain a compatible dump file.

You can still import data obtained using *Direct Path Export* into pre–Release 7.3 databases. This can be done by first importing your data into a Release 7.3 (or future) database and then exporting the data using the conventional path. The data obtained from the conventional path should be compatible with all pre–Release 7.3 versions and releases of the Import utility.

Upgrade the Import message file to verify the export method (*Direct Path Export* or conventional). Use the Release 7.3 Import utility after upgrading the Import message file (IMPMTB.MSG) to a Release 7.3 database. Upgrading the Import message file allows you to verify the import method using screen messages. If the Import LOG option is turned on, the message will also appear in the Import log file.

Upgrading and the Advanced Replication Option

This section contains the following topics:

- Setting the COMPATIBLE Parameter
- Release 7.3 Replication Triggers and Packages

Setting the COMPATIBLE Parameter

Enable Release 7.3 new features by setting the initialization parameter COMPATIBLE to 7.3.0.0 following the completion of an upgrade to Release 7.3. For more information on enabling Release 7.3 new features, see page 9 – 14.

Setting the INIT.ORA initialization parameter COMPATIBLE to 7.3.0.0 puts the database in Release 7.3 compatibility mode. However, both Release 7.3 master sites and Release 7.3 snapshot sites will still operate normally with pre–Release 7.3 replication triggers and wrappers.

Replication support must be regenerated for all replicated objects if the INIT.ORA parameter, COMPATIBLE, is reset to 7.3.0.0 or above to take advantage of new Release 7.3 features, or if you just want to upgrade to new replication triggers and wrappers.

Note: No adjustments to the Release 7.3 database are necessary if the COMPATIBLE parameter remains at a value less than 7.3.0.0 and only pre–Release 7.3 functionality will be used.

There are three possible upgrade scenarios that you might wish to follow:

- You wish to use new Release 7.3 features at a master definition site.
- You wish to use new, non–replication Release 7.3 features at one or more master sites that are not master definition sites.
- You wish to use new Release 7.3 features at a snapshot site.

Setting the COMPATIBLE Parameter at a Master Definition Site

If the COMPATIBLE parameter of any master definition site is reset to 7.3.0.0 to use new features, perform the following steps:

1. Use DBMS_REPCAT.SUSPEND_MASTER_ACTIVITY() to quiesce all object groups that are registered at the affected master definition site.
2. When all object groups are quiesced and the repcatlog is empty, use DBMS_REPCAT.GENERATE_REPLICATION_SUPPORT() to regenerate replication triggers and packages for all replicated objects such as tables, procedures, packages, and package bodies. If any pre–Release 7.3 snapshot sites exist, or if there is a possibility that pre–Release 7.3 snapshot sites may be added in the future, the GEN_REP2_TRIGGER parameter must be set to TRUE for DBMS_REPCAT.GENERATE_REPLICATION_SUPPORT.
3. If jobqueues are used, wait until repcatlog becomes empty at the master definition site. Otherwise, use DBMS_REPCAT.DO_DEFERRED_REPCAT_ADMIN() first at all non–master definition sites which replicate the object group and then at the master definition site to apply administration requests at all sites.

   

   Warning: This step must be performed twice if jobqueues is not used, because Release 7.3 generates replication support in two phases.
4. Use DBMS_REPCAT.RESUME_MASTER_ACTIVITY() to unquiesce all objects groups and begin normal replication activity. Even though the master definition site is now a 7.3.0.0 compatible Release 7.3 site, new Release 7.3 features will be available only to remote masters that have also reset their compatibility to 7.3.0.0.

Setting the COMPATIBLE Parameter at Master Sites

New Release 7.3 replication features can be used only if the master definition site is in Release 7.3 compatibility mode.

Setting the COMPATIBLE Parameter at a Snapshot Site

If the COMPATIBLE parameter of a snapshot site is reset to 7.3.0.0, to use new replication features, the following steps should be taken:

1. Make sure that the master site does not have valid, outstanding administration requests. If requests exist, wait until the requests are applied and removed from the administration queue at the master site. It is generally best to wait until the master site's administration queue is empty.
2. Use DBMS_REPCAT.GENERATE_SNAPSHOT_SUPPORT() to regenerate triggers and packages for all replicated objects, such as tables, procedures, packages, and package bodies, at the snapshot site. This assumes that the master site is available and that the master object has already generated replication support. Release 7.3 snapshot sites can use new Release 7.3 features even if their master sites are pre–Release 7.3.

   Note: These steps are recommended for both snapshot sites with a pre–Release 7.3 master site and snapshot sites with a Release 7.3 master site.

Release 7.3 Replication Triggers and Packages

Table 9 – 8 shows the new Release 7.3 replication triggers and packages.

Trigger or Package	Comments
$RT Trigger	This is the replication trigger that propagates changes to replicated tables to other sites. In 7.3.x, there are two flavors of this trigger, a pre–Release 7.3 and a Release 7.3 version. The triggers can be distinguished by the REASON column of the following views: DBA_REPGENERATED, ALL_REPGENERATED, and USER_REPGENERATED. A pre–Release 7.3 trigger, denoted by 'REPLICATION TRIGGER', supports only asynchronous propagation. A Release 7.3 trigger, denoted by MIXED REPLICATION TRIGGER, can support both synchronous and asynchronous propagation. For each Release 7.3 trigger generated, an associated $TP package and package body is also generated (see below).
$ST Trigger	This is a pre–Release 7.3 trigger that is generated at master sites if GEN_REP2_TRIGGER=TRUE for DBMS_REPCAT.GENERATE_REPLICATION_SUPPORT(). The purpose of this trigger is for Release 7.3 Masters with pre–Release 7.3 Snapshots. Pre–release 7.3 snapshots do not generate their own replication support, but copy the master site's replication triggers and wrappers. This trigger is created at the master in a disabled state and exists only to be copied by pre–Release 7.3 snapshot sites. Do not enable this trigger at master sites.
$TP Package	This package (and package body) is generated for each Release 7.3 $RT trigger. This package contains the procedures that queue deferred transactions and/or issue synchronous remote procedure calls.

Table 9 – 8 New Release 7.3 Replication Triggers and Packages

> **Note:** A database is said to be in pre–Release 7.3 compatibility mode if the value assigned to its COMPATIBLE parameter (set in INIT.ORA) is less than 7.3.0.0. A database is in Release 7.3 compatibility mode if the value assigned to its COMPATIBLE parameter is 7.3.0.0 or greater.

Downgrading and the Advanced Replication Option

This section contains the following topics:

- Downgrading a Master Definition Site or a Master Site
- Downgrading a Snapshot Site

Downgrade operations are initiated by running one or more of the downgrade scripts shown on page 9 – 7 and by performing the general downgrade steps shown on page 9 – 6.

Two distinct steps are performed by the downgrade scripts:

1. The object group and all objects replicated in the object group are unregistered if the group name is not an existing schema name.
2. The remaining replicated objects are unregistered as replicated objects if the name of the replicated object's owner does not correspond to the object group name.

   Note: There may be situations in which the object group's name is the same as the object owner's name, but the object group includes objects from multiple schemas. These objects are removed only from RepCat. The objects themselves are not deleted from the database.

Downgrading a Master Definition Site or a Master Site

After the completion of the downgrade process, replication triggers are no longer valid because all replication trigger packages were dropped. All procedures, packages, and package body wrappers were also dropped. This effectively blocks transactions against replicated tables until you regenerate replication support. From the master definition site, take the following steps (this should be done if any master site is downgraded):

1. Use DBMS_REPCAT.SUSPEND_MASTER_ACTIVITY() to quiesce all object groups that are being replicated at the downgraded site(s).
2. When all object groups are quiesced and the administration queue is empty, use DBMS_REPCAT.GENERATE_REPLICATION_SUPPORT() to regenerate replication triggers and packages for all replicated objects such as tables, procedures, packages, and package bodies.
3. If jobqueues are used, wait until the administration queue becomes empty at the master definition site. Otherwise use DBMS_REPCAT.DO_DEFERRED_REPCAT_ADMIN(), first at the remote site and then at the master definition site, to execute administration requests at all sites. This step needs to be performed only once because pre–Release 7.3 databases generate replication support in a single step.
4. Use DBMS_REPCAT.RESUME_MASTER_ACTIVITY() to unquiesce all objects groups and begin normal replication activity.

Downgrading a Snapshot Site

Updatable snapshots that use the $ST trigger will continue to operate normally. All Release 7.3 updatable snapshots that relied on a $TP package are no longer updatable. Since pre–Release 7.3 snapshot sites cannot generate replication support, these snapshots need to be unregistered and reregistered. Pre–Release 7.3 snapshots do not generate replication support, but rather copy necessary triggers, procedures, packages and package bodies from the master site. The extraction is performed only once, when the snapshot replicated object is created.

Advanced Replication Compatibility Between Release 7.3 and Earlier Releases

The following are compatibility issues between Release 7.3 and earlier releases:

- Coexistence of pre–Release 7.3 master sites and Release 7.3 master sites is permitted. You need to have a Release 7.3 master definition site to use new Release 7.3 features.
- Two–phased generation of replication support is not supported in pre–Release 7.3 compatibility mode.
- Additional calls to DBMS_REPCAT.DO_DEFERRED_REPCAT_ADMIN() are not required in pre–Release 7.3 compatibility mode.
- Replication behaves exactly as it would in a pre–7.3 release when COMPATIBILITY is set to a pre–Release 7.3 value.
- Many existing replication administration requests have been modified to include additional fields to support new Release 7.3 functionality. Release 7.3 sites will process requests with the new fields. Pre–Release 7.3 masters will ignore the new fields and process requests normally. Listed in Table 9 – 9 are RepCatLog requests that are new to Release 7.3:

Requests	Comments
GENERATE_SUPPORT_PHASE1 and GENERATE_SUPPORT_PHASE2	A Release 7.3 master definition site will send out LOG_REQUEST_GEN_SUPPORT_PHASE1 and LOG_REQUEST_GEN_SUPPORT_PHASE2 requests only to Release 7.3 master sites. Remote pre–Release 7.3 master sites will only receive GENERATE_REPLICATION_SUPPORT requests. These request are only sent to Release 7.3 master sites that have the COMPATIBLE parameter set to 7.3.0.
ALTER_MASTER_ PROPAGATION	This request is only sent to Release 7.3 master sites that have the COMPATIBLE parameter set to 7.3.0..
ADD_MASTER_DATABASE	The new “overloaded” version of this request is only used at Release 7.3 master definition site sites (similar to the already overloaded version used only at master definition site sites in pre–Release 7.3).

Table 9 – 9 New RepCatLog Requests in Release 7.3

The replication administration requests that require regeneration of all replication triggers at all master sites, for example, ADD_MASTER_DATABASE, will check for pre–Release 7.3–compatible triggers at the master definition site and regenerate these triggers as well.

Pre–Release 7.3 snapshot sites will attempt to copy triggers from their master sites. Release 7.3 triggers are incompatible with pre–Release 7.3 triggers. Therefore, DBAs can optionally generate pre–Release 7.3–compatible triggers at Release 7.3 master sites for the pre–Release 7.3 snapshots to copy (set GEN_REP2_TRIGGER to TRUE when calling DBMS_REPCAT.GENERATE_REPLICATION_SUPPORT()). The generated pre–Release 7.3–compatible triggers are disabled at the master sites and are not used at master sites.

PART

III

Appendices A, B, C, D, E, and F

APPENDIX

A

Summary of Changes in Oracle7, Release 7.0

This appendix provides an overview of the changes in the Oracle7 Server. It covers the following topics:

- Terminology
- Functionality Enhancements
- Backup and Recovery Enhancements
- Security Enhancements
- Performance Enhancements
- Administration Enhancements
- SQL*DBA Changes
- Changes to Utilities
- Changes to Views
- Oracle Datatype Changes
- SQL Command Changes
- New Features of the Oracle Precompilers, Release 1.5
- Initialization Parameter Changes
- Data Dictionary Changes
- Version 6 Compatibility
- New and Renamed SQL Scripts
- Other Changes

Terminology

Some new terms have been introduced with Oracle7 that describe Version 6 features. The concepts behind these terms are not new; the new terms are used to better explain the concepts. The following list summarizes the new terms:

initialization parameters and parameter files

Parameters placed in the parameter file are used to specify configuration settings when starting an instance. In Version 6 documentation, *initialization parameters* were commonly referred to as INIT.ORA parameters.

In Version 6 documentation, the file containing initialization parameters was referred to as the INIT.ORA file. In Oracle7 documentation, it is known as the *parameter file*.

schema

A *schema* is a logical space in which a user can create objects. In Version 6, there was no distinction between a *user* and a collection of objects that the user owns. In Oracle7, that distinction is introduced with the term *schema*.

Every user has one schema in which objects can be created. The name of that schema is the same as the user's Oracle username. Objects are uniquely identified by the schema to which the object belongs and the object name. For example, the table EMP created in SCOTT's schema is identified as SCOTT.EMP.

schema objects

In Version 6, it was only necessary to talk about tables, views, and users. In Oracle7, there are tables, views, snapshots, roles, profiles, users, procedures, packages, triggers, and integrity constraints, among other entities. These are known collectively as database *objects*.

server process

A process that handles requests (program interface calls) from user processes. A *server process* can be either dedicated to one user process, or shared among many user processes, depending on the configuration of your database system. In Version 6, these processes were known as "shadow processes".

session	A logical connection by a user to the database. One user can have several *sessions*. The Version 6 term "user process" has been replaced by "user session" in Oracle7.
shared SQL areas	In Version 6, the parsed form of SQL statements were stored in user *context areas*. In Oracle7, they are stored in *shared SQL areas*. See page A – 17 for more information.

Functionality Enhancements

This section contains the following topics:

- Enforced Integrity Constraints
- Enforced Default Values
- Extended National Language Support
- PL/SQL
- Distributed Option
- Parallel Server Option

Enforced Integrity Constraints

Data verification can now occur within Oracle, as well as at the application level. By checking data at the application level, you can ensure that the users of a particular application receive immediate useful and appropriate help and error messages when they access a table. Checking at the database level ensures that no incorrect data can be entered in the table, regardless of which application or tool is used. Data checking at the database level is accomplished by using integrity constraints.

Integrity constraints can be defined to specify logical relationships of data in a database. Version 6 introduced syntactic support for data integrity constraints. Oracle7 supports the enforcement of integrity constraints to ensure that a database contains consistent data. If a Data Manipulation Language statement violates an integrity constraint, the statement is rolled back and an error is returned.

Enabling Constraints

Integrity constraints can be enabled or disabled. When constraints are enabled on existing data, an exception table can be generated to list all rows that violate constraints along with the constraints they violate.

Unique Constraints

In Version 6, the fact that table indexes were unique was frequently used to enforce uniqueness constraints. This is no longer necessary in Oracle7 since the UNIQUE constraint can now be used to guarantee uniqueness.

Delete Cascade

When deleting a master row that is referenced by foreign keys in other tables, you can choose to *cascade* the delete, dropping all foreign key rows, as well as the master row.

Standards Compliance

The integrity constraints implemented in the Oracle7 Server fully comply with the SQL89 Level 2 standard set forth by ANSI X3.135–1989 and ISO 9075–1989.

Along with Oracle7, the Release 1.5 precompiler is 100% compliant with the ANSI SQL89 standard, including integrity features.

In addition, the Release 1.5 precompiler passes 100% of the tests contained in the National Institute of Standards and Technology (NIST) SQL89 test suite. It also contains the FIPS Flagger (required by the Federal Information Processing Standard) to call attention to any non–ANSI–standard extensions that are used.

Enforced Default Values

You can now specify a default value to be used for a column by supplying an expression for the DEFAULT clause of the ALTER or CREATE TABLE statement. If you issue an INSERT statement without providing a column value, this default value will be used. Version 6 introduced syntactic support for default values, whereas Oracle7 supports the use of default values.

Extended National Language Support

National Language Support (NLS) in Oracle7 supports the use of multi–byte character sets used in many Asian languages. It also provides greater flexibility in specifying language or territory, either for a session or for the entire database.

New NLS initialization parameters allow the specification of default date format, currency symbol, number–group separator, and decimal character. These format characteristics can also be specified explicitly in SQL functions to override the default values.

The values of all NLS parameters are defined with systemwide defaults, and can be overridden on a per–session basis. Different sessions connected to the same instance can use different values for NLS parameters. These values can be changed during a session.

For more information on NLS features supported by the Oracle7 Server, see *Oracle7 Server Concepts* and *Oracle7 Server Reference.*

PL/SQL

The PL/SQL programming language now permits a *stored procedure* or *function* to be defined and compiled once, stored in the database, and then executed by multiple users and applications. Stored procedures and functions consist of a set of SQL and PL/SQL statements that are stored in a compiled form in the database. In addition, PL/SQL now allows *stored packages* and database *triggers*.

Packages

Packages provide a method of encapsulating and storing related procedures, functions, cursors, variables, constants, and exception handlers as a unit. Global package variables and constants can be declared and used by any procedure in the package. Between calls, variables and cursors retain their state for each session. Only public data, procedures, and functions can be accessed from outside the package.

The *Oracle7 Server Application Developer's Guide* contains a detailed description of stored procedures, functions, and packages.

Triggers

Triggers are shared procedures that are automatically executed as a result of an insertion into, update of, or deletion from a table. A trigger consists of an event to signal the firing (execution) of the trigger, trigger restriction, and the action to take when the trigger fires. A trigger is implicitly fired by the Oracle7 Server when the triggering event occurs, no matter which user is connected or which application is being used.

The *Oracle7 Server Application Developer's Guide* contains a detailed description of triggers and their uses.

Compilation of Procedural Objects

Procedures, functions, packages, and triggers are automatically recompiled, as needed, whenever they are referenced. All of these procedural objects can also be compiled manually. With either method, all dependent objects are marked for automatic recompilation so that the most recent version of a routine is always used.

PL/SQL Language Changes

PL/SQL in Oracle7 supports remote procedure calls if both the distributed option and PL/SQL are licensed. Remote procedure calls include support for two–phase commit (described in the next section).

Several new commands, functions, and datatypes have been added to the PL/SQL language, and many changes have been made to existing commands. Refer to the changes appendix of the *PL/SQL User's Guide and Reference* for a detailed description of the changes.

Pipes and Alerts	Several new packages supplied with PL/SQL implement process–to–process communication *pipes* and procedural *alerts*. Pipes allow different processes to communicate by sending messages to one another. By registering an alert, a process can be informed when a specified event occurs in the database; for example, when a field reaches a particular value. For more information on pipes and alerts, see the *Oracle7 Server Application Developer's Guide.*
Interactive Output from Stored Procedures	Oracle7 stored procedures can display output interactively using the DBMS_OUTPUT package. Output can be enabled or disabled for debugging. Output lines are automatically displayed by SQL*DBA and SQL*Plus when output is enabled. You can also access debugged output from within your procedures. For more information, see the *Oracle7 Server Application Developer's Guide.*
Distributed Option	In Version 6, only queries were permitted to remote databases. Oracle7 with the distributed option supports all Data Manipulation Language operations, including queries, insertions, updates, and deletions of remote table data. When both the distributed option and PL/SQL are installed, calls to remote procedures are supported, as is the use of table snapshots. You might want to distribute your database to improve performance or accommodate organizational structure. You should investigate and implement any changes that are necessary to the architecture of your applications to allow them to store data where it is used most often. See *Oracle7 Server Distributed Systems, Volume I,* for a complete description of the distributed database features.
Two–Phase Commit	The two–phase commit mechanism transparently maintains data consistency throughout a distributed database by guaranteeing that a distributed transaction either commits at all nodes or rolls back at all nodes, no matter what type of system or network failure occurs. Any inserts, updates, or deletes in a distributed transaction (whether from SQL statements or stored procedures) are protected by the two–phase commit mechanism. In the event of a node failure, two–phase commit automatically coordinates the recovery of all nodes involved in a transaction.
Deadlock Detection and Resolution	The Oracle7 distributed option also detects and resolves distributed deadlock conditions.
Multi–Node Read Consistency	For a single query that spans multiple nodes, read–consistency is guaranteed. For multiple queries, the ability to specify a *read–only* transaction guarantees read–consistency between distributed queries.

Snapshot Capability

If you have both the distributed option and PL/SQL, then you can make read–only copies of master tables (snapshots) at remote sites. Snapshots can reduce network traffic, improve access times, and allow remote sites to continue functioning in the event of network failure or central system failure.

Snapshots can contain a subset of a table's rows, or they can summarize the information in multiple rows. They can also join data from multiple master tables.

A master table can have multiple snapshots in different locations. Snapshots can be refreshed either manually or automatically, using either a complete copy of the master table or a log that contains master table changes.

Global Naming

In Oracle7, every object in a distributed database can be uniquely specified with global names. Following this convention paves the way for the future use of external name servers. With such servers, the work of resolving the path to an external object is done by the server, instead of within the database. To ensure that the current system is correctly configured, the GLOBAL_NAMES parameter can be set, as described in *Oracle7 Server Reference*. The global names capability is designed to work with external name servers, such as the OSF's Distributed Computing Environment (DCE).

If you rename a database with a global name and begin enforcing global naming, the change in database name must also be reflected in the following areas:

- Database links in remote databases that point to the renamed database must be renamed.
- Synonyms in remote databases that point to objects within the renamed database must be updated.
- PL/SQL program units in remote databases that reference data in the renamed database must be recompiled.

You may choose to enable global naming gradually by enforcing it in selected databases in a distributed system. A database with global naming enabled can successfully query a database where global naming is not enabled as long as the links to the database are named properly. A database where global naming is not enabled can successfully query a database where global naming is enabled.

Global names will be required in future releases of Oracle; however, in Oracle7, you can disable global naming. Disabling global naming allows database link names to have no correspondence to global database names, as with earlier versions of Oracle.

DB_DOMAIN Parameter	The Oracle7 initialization parameter DB_DOMAIN replaces the Version 6 parameter DB_DIRECTORY. DB_DOMAIN's default value is ".WORLD". You should change it to a unique value for every system in a distributed network to identify objects uniquely in a distributed system.
Closing Database Links	A database link can now be closed when it is no longer needed without terminating the session by using the ALTER SESSION command. This can decrease your telecommunication charges, and reduce resource consumption at the site of the remote database.
LONGs Supported	In Oracle7, LONG data items can be referenced in distributed queries, updates, and deletes.
Improvements in Distributed Query Processing	If all of the data referenced by a query is located on a remote node, then the Oracle7 query processor evaluates the query at the remote node, returning only the result to the originating node. Multi–node queries are divided into parts, and each part is sent to the remote node for evaluation. This mechanism can dramatically reduce network traffic. In addition, the optimizer can use information on indexes as well as statistics from remote tables to develop the execution plan. Using this information, queries are optimized in the same way that they would be if the tables were local. For example, Oracle7 may be able to perform a nested–loop join, where Version 6 may have been constrained to perform a sort/merge join.
Heterogeneous Distributed Database Systems	New coordination mechanisms make it possible for the Oracle7 Server to operate in a distributed arrangement with non–Oracle7 databases.
Standards Compliance	To interface to TP monitors for transaction coordination, the Oracle7 Server supports the XA interface, as defined in the X/OPEN specification.
Parallel Server Option	The Parallel Server option to the Oracle7 Server supports simultaneous database access from two or more loosely coupled systems to achieve high availability and performance. For a discussion of the new Parallel Server features, see the changes appendix in *Oracle7 Parallel Server Concepts & Administration*.

Backup and Recovery Enhancements

This section contains the following topics:

- Recovery Capabilities
- Parallel Server Recovery Enhancements
- SCN–based Recovery
- Mirrored Online Redo Log Files

Recovery Capabilities

The following changes have been made to recovery features in Oracle7:

- The RECOVER command in SQL*DBA has new options for incomplete recovery.
- Filename format can now be specified for archived redo log files with the initialization parameter LOG_ARCHIVE_FORMAT.
- Each instance running in the parallel server has its own set of online redo log files.
- Checkpoints can now be specified by time interval, and can also be forced on demand.

Refer to *Oracle7 Server Concepts* and the *Oracle7 Server Administrator's Guide* for more information about recovery and recovery structures. See *Oracle7 Server Utilities* for more information about the RECOVER command.

Parallel Server Recovery Enhancements

In Oracle7, it is possible to perform the same tablespace and datafile operations in parallel mode as when running in exclusive mode. For example, the database can remain running while a tablespace is taken offline for backup, or brought back online as part of a recovery operation. For details, see the changes appendix in *Oracle7 Parallel Server Concepts & Administration*.

SCN–based Recovery

System Change Numbers (SCNs) can be used in Oracle7 recovery operations, allowing you to recover up to a specific transaction. Details are given in the *Oracle7 Server Administrator's Guide.*

Mirrored Online Redo Log Files

The Oracle7 Server provides the capability to maintain "mirror images" of the online redo logs. When mirrored online redo log files are configured, the LGWR background process concurrently writes the same redo log information to multiple active online redo log files, thereby eliminating a potential single point of redo log failure.

You should mirror your online redo log files to reduce the chance of losing a log file in the event of media failure. Since mirrored redo log files do not significantly affect performance, the only additional overhead is space to contain the additional log files. See *Oracle7 Server Concepts* for a detailed description of how to configure mirrored redo log files.

Security Enhancements

This section contains the following topics:

- System and Object Privileges
- Roles
- Auditing Changes
- PUBLIC Quotas
- Standards Compliance and Trusted Oracle7

System and Object Privileges

Many new system privileges have been added to Oracle7 to replace the three Version 6 system privileges: CONNECT, RESOURCE, and DBA. For example, statements like the following give users the privilege to create tables in their schemas:

```
GRANT CREATE TABLE TO SCOTT;
```

The new system privileges in Oracle7 allow for more specific control of system operations. The Version 6 system privileges have become predefined roles in Oracle7 for backward compatibility (roles are discussed later in this section).

An object privilege gives a user the right to perform a selected operation on a specific database object. See *Oracle7 Server Concepts* for more information about system and object privileges.

Creating Users

In Version 6, users were created as a side–effect of the GRANT CONNECT command. For Oracle7 it is recommended that you use the new CREATE USER command to create a user.

Users created in this manner have no privileges after creation. The GRANT CREATE SESSION command can then be employed to give the user the CREATE SESSION system privilege. (This privilege is the equivalent of the Version 6 CONNECT system privilege.) For backward compatibility, GRANT CONNECT currently creates a user, but this behavior may not always be supported.

RESTRICTED SESSION Privilege

The Oracle7 privilege CREATE RESTRICTED SESSION limits database access to privileged users. In Oracle7, you grant this privilege to create a special class of users who can use the database when it is not completely open, rather than assigning the DBA role, as was done in Version 6.

Roles

Oracle7 simplifies privilege management with roles. Roles are groups of related privileges that are granted to users or other roles.

You can create roles with specific privileges for classes of users, such as computer operators and data entry clerks. When you assign a role to a user, that user is given all of the associated privileges. In addition, when you change the privileges granted to a role, the change affects all users with that role. Roles can be password–protected, and they can be dynamically enabled.

For more details on roles and privilege management, see *Oracle7 Server Concepts.*

SET ROLE Command

Issuing a SET ROLE command at the start of an application automatically enables the designated role for the user (provided the user has been granted that role) and disables any previously enabled roles. Users can have default roles, and passwords can limit which applications can use each role.

Predefined Roles

Version 6 had three system privileges: CONNECT, RESOURCE, and DBA. Oracle7 defines roles with the same names, containing the equivalent Oracle7 system privileges. These roles can be modified by the system administrator, and new roles can be created.

The roles IMP_FULL_DATABASE and EXP_FULL_DATABASE allow users to perform full and incremental imports and exports. These roles can be assigned to DBAs and non–DBAs.

The predefined roles OSOPER and OSDBA have also been added to Oracle7. The OSOPER role is intended for operators and others who need some of a DBA's privileges to perform system maintenance, but who should not be granted full DBA privileges, as contained in OSDBA.

Table A – 1 lists the predefined roles that are provided for backward compatibility to Version 6.

Role Name	Privileges
CONNECT [1]	ALTER SESSION, CREATE CLUSTER, CREATE DATABASE LINK, CREATE SEQUENCE, CREATE SESSION, CREATE SYNONYM, CREATE TABLE, CREATE VIEW
RESOURCE [1,2]	CREATE CLUSTER, CREATE PROCEDURE, CREATE SEQUENCE, CREATE TABLE, CREATE TRIGGER
DBA [1,3]	All system privileges WITH ADMIN OPTION

Table A – 1 Predefined Roles

[1] created by SQL.BSQ
[2] grantees of the RESOURCE role also receive the UNLIMITED TABLESPACE system privilege as an explicit grant (not part of the RESOURCE role)
[3] grantees of the DBA role also receive the UNLIMITED TABLESPACE system privilege with the WITH ADMIN OPTION as an explicit grant (not as part of the DBA role)

During migration, Version 6 users with the CONNECT, RESOURCE, or DBA system privileges are granted the respective roles in the Oracle7 database. After migration, you should consider how you want to control access to your database and create appropriate roles. You will probably want to have a variety of roles for users and developers of different applications, as well as the various people who administer your system. After creating and assigning these roles, you should revoke the CONNECT, RESOURCE, and DBA roles from users in your database. However, you should not alter or drop the predefined roles.

There are a few small differences between the Version 6 DBA privilege and the Oracle7 DBA role. Views that could be created in Version 6 only because the view owner had DBA privilege cannot be created in Oracle7. The Migration Utility migrates these views, but they are inaccessible to other users because Oracle7 views cannot use privileges granted to the view owner via the DBA role. To access the view, the privileges required for the view's creation must be granted directly to the owner of the view.

Oracle7 Server Concepts gives a more complete description of the new system privileges and strategies for creating roles, as well as how you can use your operating system for role management.

Improved Operating System Security Interface

The operating system can be used to maintain a list of Oracle7 roles, manage the grants and revokes of roles, and, if desired, manage the verification process for roles that require authorization. For details, see *Oracle7 Server Concepts*.

As in Version 6, users can be authenticated by the operating system as well as by the Oracle7 Server. However, in Oracle7 the prefix OPS$ (used to identify operating system authenticated roles) can be changed by adjusting the OS_AUTHENT_PREFIX initialization parameter.

Auditing Changes

Statement execution and the use of system privileges can now be audited on a per–user basis. Table access can be audited, as can the execution of SQL statements. In addition, new Oracle7 constructs such as triggers, procedures, and roles can be audited. To accommodate the new auditing functionality, changes have been made to most of the auditing views defined in CATAUDIT.SQL.

In Version 6, auditing of SQL statements occurred at parse time. In Oracle7, auditing occurs at execution time, since the parsed form of SQL statements are frequently already stored in shared SQL areas. But if an application explicitly controls statement processing in order to execute a statement many times but only parse it once, then only a single audit record is generated.

In Oracle7, table auditing and view auditing options are independent. If a table is accessed through a view, audit records may be generated for the table, for the view, or for both, depending on which auditing options are enabled. If a table is audited, an audit record is generated whenever that table is accessed, whether directly or through a view. An audit record is only generated on the view if auditing is enabled for the view, regardless of the audit–status of the table.

Auditing is discussed in the *Oracle7 Server Administrator's Guide.*

PUBLIC Quotas

Although the UNLIMITED TABLESPACE system privilege can still be granted to PUBLIC, there is no Oracle7 equivalent for the Version 6 action of granting a tablespace quota to PUBLIC. In Oracle7, quotas are assigned as part of the CREATE USER or ALTER USER statements. See *Oracle7 Server SQL Reference* for more details.

Standards Compliance and Trusted Oracle7

The Oracle7 Server is designed to comply with level C2 of the National Computer Security Center (NCSC) standard for computer security, as defined in the Orange Book. Oracle7 provides discretionary access control.

Trusted Oracle7 is designed to comply with level B1. Trusted Oracle7 contains all features and functions of the Oracle7 Server. In addition, it provides Mandatory Access Control (MAC) allowing for the definition of data sensitivity levels (for example *secret*, *restricted*, and *unclassified*) and user clearance up to a particular level. For more information on the Trusted Oracle7 Server, see the *Trusted Oracle7 Server Administrator's Guide*.

Performance Enhancements

This section contains the following topics:

- Multi–Threaded Server Architecture
- Checkpoint Process
- Rule–Based Optimization
- Hashing
- Shared SQL Areas
- TRUNCATE Command
- Additional Improvements

Multi–Threaded Server Architecture

The Multi–Threaded server architecture can reduce system overhead on multi–user systems, allowing you to increase the number of concurrent users. *Oracle7 Server Concepts* contains a full description of the multi–threaded server architecture.

In Version 6, when using the "two task" architecture, a shadow process was associated with each user process to communicate with the database. In Oracle7, dedicated server processes and shared server processes replace shadow processes. Dedicated server processes are used in a normal two–task configuration and have the same functionality as shadow processes. Shared server processes are used when you configure your database to run with dispatcher processes that route requests from user processes to the servers.

In Version 6, shadow processes often had a high percentage of idle time. They were either waiting for the user to enter a request, or for the application to process data. In Oracle7, the shared server versions of those processes use that time to service other users. Thus, only a few shared server processes are needed to serve many users, with no loss in throughput. In Oracle7, the system resources that were previously dedicated to shadow processes are available for additional users.

Checkpoint Process

In Oracle7, the checkpoint process (CKPT) can be enabled. It helps on heavily loaded systems by taking over the work of checkpointing from LGWR. For more information, see *Oracle7 Server Concepts.*

The Version 6 optimizer used only the rule–based method to determine the execution plan for a SQL statement. The rule–based method considers only the syntax of the statement, table indexes, and table clusters when formulating a plan. The Oracle7 optimizer can choose an execution plan with the lowest expected "cost" using statistics collected with the ANALYZE command. *Hints* can also be provided to control the optimization manually.

ANALYZE Command

The ANALYZE command introduced in Oracle7 computes or estimates statistics on tables, clusters, and indexes. Statistics are then stored in the data dictionary for performance tuning and query optimization.

ANALYZE can also identify chained rows and validate the structure of tables and indexes. In this capacity, it replaces the VALIDATE INDEX command. (Although VALIDATE INDEX is supported in Oracle7, it may not be supported in future versions.)

You should familiarize yourself with the statistics that are collected by the ANALYZE command (see *Oracle7 Server SQL Reference* and *Oracle7 Server Tuning*) and how these statistics are used by the cost–based optimization approach. You should then establish a plan for collecting these statistics at appropriate time intervals. Note that you must re–collect statistics for data that has been exported and imported.

Rule–Based Optimization

If you do not collect statistics, or if the initialization parameter OPTIMIZER_MODE is set to RULE, Oracle continues to use the rule–based optimization approach. There are some minor changes in this method between Version 6 and Oracle7:

- You may notice some performance improvements with joins of clustered tables where one table also has a unique index on the cluster keys.
- Where possible, queries are executed entirely at one node.
- Joins in distributed queries might be much faster because the local node is now aware of the indexes and statistics for the remote node and can choose to perform a sort/merge or nested loop join as appropriate.
- Two types of semantically undefined outer–join queries now produce error messages.

If you have manually tuned your SQL statements to force a particular execution plan (for instance, by concatenating null or adding 0 to a numeric column), Oracle7 continues to recognize these constructs, and you should generally notice no difference in the execution time for the statement.

Because future versions of Oracle will not support the rule–based optimization approach, you should begin removing this manual tuning information. In general, you can simply remove this tuning information and use the cost–based optimization method with no new tuning being needed. The cost–based optimization method generally selects an execution path for the untuned SQL statement that is as good as, or better than, the path selected for the tuned statement.

However, there are certain rare cases, such as when you have a non–uniform distribution of data, when you might need to provide "hints" to the cost–based optimizer. Additionally, there are situations where it might be necessary to provide hints to the cost–based optimizer for a query that did not need to be tuned when using the rule–based optimization approach. Once again, these usually involve non–uniform distributions of data.

Oracle7 Server Tuning contains more information on Oracle7's cost–based optimizer.

Refer to *Oracle7 Server SQL Reference* for a complete description of the syntax and function of the ANALYZE command. See the ALL_TABLES, USER_TABLES, and DBA_TABLES views in *Oracle7 Server Reference*.

Hashing

Hash clusters permit more efficient retrieval of data stored in clusters. Hashing may be used instead of an index to locate a row more quickly, usually in one read operation. This is useful when you want to fetch one row. See the *Oracle7 Server Administrator's Guide* for details.

Shared SQL Areas

Shared SQL areas are the memory buffers that hold the parsed form of SQL statements. In Version 6 they were called context areas, and they were stored in each user process's private PGA. Oracle7 stores the parsed form of SQL statements in the SGA and shares them among concurrent users.

When different applications issue identical SQL statements, the shared SQL area used to process the first occurrence of a statement is reused when processing subsequent occurrences of the same statement, regardless of the process that issues it. Shared SQL areas reduce memory usage and save parsing time, without any change to existing applications.

However, the SQL statements must be identical in order to take advantage of this feature. You should develop some guidelines for how SQL statements should be written. This includes spacing, capitalization, bind variables, and object names. For instance, if two identical SQL statements are issued, but one uses a synonym to refer to the table, these statements are stored separately in the SGA.

TRUNCATE Command

The TRUNCATE command quickly deletes all rows in a table or cluster and optionally shrinks the corresponding segment by deallocating unnecessary extents. Removing rows with the TRUNCATE command is faster than removing them with the DELETE command, and TRUNCATE is more convenient when you want to empty a table without removing it. For more information about the TRUNCATE command, see *Oracle7 Server SQL Reference* and the *Oracle7 Server Administrator's Guide*.

Additional Improvements

The following additional improvements have been made to the Oracle7 architecture in order to improve performance:

- shortened critical code paths
- improved latching
- reduced logging
- the ability to use standard Transaction Processing (TP) Monitors (on some platforms)

Administration Enhancements

This section contains the following topics:

- Rollback Segments
- Resource Limits
- Profiles
- User Definitions
- ALTER SYSTEM Command

Rollback Segments

Rollback segments are more efficient and are easier to manage in Oracle7. Not only can rollback segments grow as needed, but they can dynamically shrink to a specifiable optimal size if space in the segment is not being used. Rollback segments can be taken offline and brought back online to facilitate rollback segment management. It is also possible to specify which rollback segment to use for a given transaction. See the *Oracle7 Server Administrator's Guide* for a complete discussion of rollback segments.

PCTINCREASE and MAXEXTENTS Not Applied

PCTINCREASE is no longer applied to the allocation of new rollback segment extents in Oracle7. Because of this change, all extents within a rollback segment are of uniform size, which helps to reduce disk fragmentation. MAXEXTENTS is also no longer specifiable for rollback segments.

Resource Limits

In Oracle7, limits can be set on the system resources available to a user. By explicitly setting resource limits, the database administrator can prevent uncontrolled consumption of valuable system resources.

Resource limits apply on a per call or per session basis. They can be set for

- CPU usage
- logical I/O
- idle time
- connect time
- number of open sessions
- memory usage

Profiles

Resource limits are easily managed with user profiles. A profile is a named set of resource limits that can be assigned to users. Profiles are only needed if resource limits are a requirement of a database's security policy. The database administrator can universally enable or disable the enforcement of profile resource limits at the database level.

User Definitions

In Version 6, users could not be created without granting them connect access. In Oracle7, user definitions can be created without automatically granting access to them, and a user's right to log on can be dropped while leaving the associated schema intact. For more information, see the *Oracle7 Server Administrator's Guide.*

ALTER SYSTEM Command

The SQL command ALTER SYSTEM can be used to change Oracle7 configuration with respect to files, resource limits, multi–threaded server processes, and distributed recovery. For more information, see *Oracle7 Server SQL Reference.*

SQL*DBA Changes

Interactive Menu Interface

Version 6 introduced SQL*DBA to facilitate common database administrator activities, such as monitoring database activity and database recovery. SQL*DBA in Oracle7 is further enhanced with a menu–driven interface to make database administration easier. New monitors have also been introduced.

As in Version 6, any SQL statement can be executed from SQL*DBA, as well as stored procedures and functions. Refer to *Oracle7 Server Utilities* for more information about SQL*DBA's menu interface, commands, and monitoring capabilities.

CONNECT Required before STARTUP or SHUTDOWN

In order to issue a SQL*DBA STARTUP or SHUTDOWN command, you must first CONNECT to the database. This is a new requirement in Oracle7. It was not necessary in Version 6. Accordingly, scripts similar to the following no longer work:

```
SQLDBA> STARTUP
```

For Oracle7, such commands should be embedded in scripts that include a CONNECT clause. The script can then be invoked with a command line similar to the following:

```
SQLDBA> @START_SCRIPT.SQL
```

Also, there are several new options to the STARTUP command; see *Oracle7 Server Utilities* for more information.

Monitors	The SQL*DBA monitor commands no longer take arguments. Instead, fields in the monitor screens take values to customize the monitor displays. Also, the monitor screens are no longer available when using the Version 6–compatible line–mode interface. They are only available when using the Oracle7 window and menu interface.
Starting a Database in Restricted Mode	Rather than using the Version 6 command STARTUP DBA, you use the Oracle7 command STARTUP RESTRICTED to limit database access to users who have been granted the RESTRICTED SESSION privilege.
Controlling Restricted Sessions	Using the SQL command ALTER SYSTEM, restricted session mode can be enabled and disabled without having to restart the database. See *Oracle7 Server SQL Reference* for details.
KILL SESSION Command	The SQL*DBA KILL SESSION command allows the database administrator to terminate user sessions remotely.
DESCRIBE Command	SQL*DBA supports the DESCRIBE command in Oracle7. This command describes tables, views, procedures, functions, and packages.
New Reserved Words	V6 and V7 are now reserved words in SQL*DBA.
Debug Output Supported	Output lines generated in stored procedures and functions are automatically displayed by SQL*DBA when you enable the SERVEROUTPUT function, either from the menu interface or with the SET command. The output routines are described in the *Oracle7 Server Application Developer's Guide.*

Changes to Utilities

This section contains the following topics:

- Import/Export Changes
- SQL*Loader Changes

Import/Export Changes

The Oracle7 Import/Export utilities handle all of the new Oracle7 objects. Among other improvements, error–managing facilities have improved, and messages can be stored in a log file. Major changes include the following:

- You can create an export file containing a read–consistent image of tables and views.
- In order to prevent accidental destruction, database files are no longer automatically reused on a full database import.

- Index creation statements can be routed to a file during import, instead of being processed.
- The default values for Export's GRANTS and CONSTRAINTS parameter and for Import's IGNORE parameter have changed.
- A privileged user can import objects into another user's schema, even if that user does not have the CREATE privileges.
- Single–byte character sets in an export file are automatically translated to the target database's single–byte character set.

For more information, see *Oracle7 Server Utilities.*

SQL*Loader Changes

SQL*Loader's *direct path* greatly reduces data loading times. This path bypasses SQL processing and loads data directly into the database. Other improvements to SQL*Loader include the following:

- SQL functions can be applied to data as it is loaded.
- New datatypes have been added.
- Multi–byte character sets are supported.
- White space and field delimiters can be handled with greater precision.
- New performance features include the use of table truncation, OS–specific specification of file management, and the ability to make use of operating system sorts.
- Parameters can be specified in a separate control file.

For more information, see *Oracle7 Server Utilities.*

Changes to Views

This section contains the following topics:

- Creating a View with Errors
- View Still Valid after Changing Table
- Replacing a View
- SELECT * In View Definitions

Creating a View with Errors

In Version 6, it was not possible to create a view if the view could not be immediately queried. So, for example, it was impossible to create a view if an underlying table did not exist, or if its definition did not match that of the view.

In Oracle7, such views can be created. The underlying table can then be created or modified later. Until the underlying tables are corrected, any attempt to use the view produces an error. This change relaxes the strict sequence of table–before–view creation, permitting greater flexibility in database definition.

When accessing a view that has been created with errors, the view is "compiled" to determine if it has become usable. A new SQL command, ALTER VIEW ... COMPILE, lets you force the recompilation at any time, without waiting for the view to be accessed.

View Still Valid after Changing Table

In Version 6, a view failed if its underlying table was modified or if it was deleted and re–created. Because Oracle7 implements object dependencies, such a view is marked for recompilation when the underlying table changes. The view is then recompiled the next time it is accessed, or after an ALTER VIEW ... COMPILE command, so it functions normally as long as all of the columns it accesses still exist.

Replacing a View

An existing view can be dropped and re–created in one statement using the syntax

```
CREATE OR REPLACE VIEW
```

When a view is replaced in this manner, existing GRANTS and references by other views remain intact.

SELECT * In View Definitions

In Version 6, views created with "SELECT *" would operate only so long as new columns were not added to the underlying table, because the wildcard (*) was expanded dynamically during the select. As a result, the number of columns in the changed table no longer matched the view definition.

Oracle7 adopts SQL's standard behavior of expanding such wildcards when the view is defined. The number of columns is then statically defined, because a statement like

```
SELECT *
```

is transformed into one like

```
SELECT column1, column2, column3
```

As a result, the view remains valid even when additional columns are added to the underlying table. (If there is an error when the view is created, the wildcard is expanded at the first successful compilation of the view.)

Oracle Datatype Changes

This section contains the following topics:

- Conversion of Datatypes
- Character Datatypes
- LONG and LONG RAW Datatypes
- MLSLABEL
- Addition of ROWLABEL Column
- Change in ROWID Format

Conversion of Datatypes

Refer to "Data Dictionary Conversion" on page 7 – 5 for more information on how your existing Version 6 datatypes will be converted to Oracle7 datatypes using different migration options.

Character Datatypes

Version 6 supported one datatype, CHAR, for handling variable–length character strings, and one synonym for that datatype, VARCHAR. Oracle7 uses two datatypes for character strings, as described in the following sections.

CHAR

Values of this datatype are fixed–length character strings with a maximum length of 255. CHAR values are compared using padded comparison semantics. The Oracle7 CHAR datatype is not equivalent to the Version 6 CHAR datatype.

VARCHAR2

Values of this datatype are variable–length character strings of maximum length 2000. VARCHAR2 values are compared using non–padded comparison semantics. The Oracle7 VARCHAR2 datatype is equivalent to the Version 6 CHAR datatype except for the difference in maximum lengths.

VARCHAR

VARCHAR is a synonym for the VARCHAR2 datatype in Oracle7, but future versions may use VARCHAR as a separate datatype, so use of VARCHAR is not recommended. Refer to *Oracle7 Server SQL Reference* for a description of the different comparison semantics.

LONG and LONG RAW Datatypes

The LONG and LONG RAW datatypes can store a variable–length character string of up to 2 gigabytes (2^{31}–1 bytes) in length, as opposed to 64 Kb in Version 6. One piece at a time can be fetched with the new OCI call OFLNG.

MLSLABEL

The MLSLABEL type was added for compatibility with Trusted Oracle. For more information, see the *Trusted Oracle7 Server Administrator's Guide*.

Addition of ROWLABEL Column

ROWLABEL is a new virtual column in Oracle tables. Like ROWID, it is automatically generated when a table is created. It has the datatype MLSLABEL. This column exists for compatibility with Trusted Oracle, which uses it to store the security label of each row in a table. For more information, see the *Trusted Oracle7 Server Administrator's Guide*.

Change in ROWID Format

In order to support more files per database, the number of bits in the ROWID used to identify file number and block id have been reallocated. This change has no effect in the externally visible ROWID.

SQL Command Changes

Many new commands and functions have been added to the SQL language in Oracle7, and many changes have been made to existing commands. New format models and datatypes have also been added, along with new built–in trig and math functions, the UNION ALL operator, and a new comment facility. Two new keywords have been added: ROWLABEL and VARCHAR2. For additional information, see *Oracle7 Server SQL Reference* for a detailed description of the changes.

New Features of the Oracle Precompilers, Release 1.5

This section contains the following topics:

- Datatype Equivalencing
- Bundled Database Calls
- Concise, Faster Generated Code
- More Flexible Error Handling
- Smart Resizing
- Smart Rebinding
- Full Reentrance
- Separate Executables for Each Language
- Standards Compliance
- New Debugging Aid

Release 1.5 of the Oracle Precompilers was introduced with Oracle7, and incorporated many new features designed to enhance precompiler functionality and application performance, as well as allow your applications to access new Oracle7 capabilities. Most users will upgrade to Release 1.5 at the same time they upgrade to Oracle7. Use of Release 1.3 and 1.4 of the Precompilers is supported against Oracle7. However, if you wish to use the enhanced functionality of Oracle7, you must use Release 1.5 of the Precompilers.

The following changes have been introduced since Release 1.3 of the Precompilers. Some changes appeared in Release 1.4 as well, but are listed here for convenience.

Datatype Equivalencing	You can now selectively override the datatypes assigned to host variables in the Declare Section, and use those variables in static embedded SQL statements. In C and Pascal, you can also associate an Oracle datatype with user–defined datatypes. This ability to equivalence datatypes adds flexibility to your applications.
Bundled Database Calls	Whenever possible, only a single bundled database call is made per embedded SQL statement, speeding up communication with Oracle, especially in a networked environment.
Concise, Faster Generated Code	The interface to the Oracle runtime library has been revised, reducing the amount of precompiler–generated code required to execute embedded SQL statements. This code is not only easier to read, but allows applications to run faster.
More Flexible Error Handling	An enhanced SQL WHENEVER statement allows you to embed host–language procedure calls in a precompiled program. If a SQL statement fails, you can have your program transfer control to a routine, and return when it ends.
Smart Resizing	Private SQL areas are automatically resized and the AREASIZE option is now obsolete.
Smart Rebinding	Host variables are rebound only when necessary, making the REBIND option obsolete.
Full Reentrance	Reentrant code is generated automatically for systems that require it and the REENTRANT option is now obsolete.
Separate Executables for Each Language	The precompilers are now distributed as separate executables, each designed for a specific host language. As a result, the HOST option is no longer needed for specifying the host language, but it remains for port–specific and COBOL–specific purposes.
Standards Compliance	The Release 1.5 precompilers have been verified as 100% compliant by the National Institute of Standards and Technology. Additionally, the new FIPS (Federal Information Processing Standard) flagger option issues warning messages whenever it detects the existence of an Oracle SQL extension, or usage of a standards–mandated feature in a non–standard manner.
New Debugging Aid	Now the SQLCA stores more runtime information about the outcome of SQL operations. Specifically, SQLERRD(5) stores a parse error offset to help you debug SQL statements.

Initialization Parameters

The following tables list the obsolete and renamed Version 6 initialization parameters and the new Oracle7 parameters. Refer to *Oracle7 Server Reference* for a complete description of all initialization parameters.

Obsolete Parameters

BG_PRIORITY	FREE_LIST_PROC
CALLS	GC_SORT_LOCKS
CONTEXT_AREA	INSTANCES
CONTEXT_INCR	LANGUAGE
CPU_COUNT	LOG_ALLOCATION
DP_BLOCK_CACHE_ PROTECT	LOG_BLOCKS_DURING_ BACKUP
DP_BLOCK_COMPUTE_ CHECKSUMS	LOG_BUFFERS_DEBUG
DB_BLOCK_HASH_ BUCKETS	LOG_DEBUG_MULTI_ INSTANCE
DB_BLOCK_MAX_ CLEAN_PCT	LOG_ENTRY_PREBUILD_ THRESHOLD
DB_BLOCK_MAX_MOD_PCT	LOG_IO_SIZE
DB_BLOCK_MAX_SCAN_CNT	MESSAGES
DB_BLOCK_MAX_SCAN_PCT	ROW_CACHE_BUFFER_SIZE
DB_BLOCK_MULTIPLE_ HASHCHAIN_LATCHES	ROW_CACHE_ ENQUEUES
DB_BLOCK_TIMEOUT_ WRITE_PCT	ROW_CACHE_ INSTANCE_LOCKS
DB_BLOCK_WRITE_ BATCH	ROW_CACHE_MULTI_ INSTANCE
DB_HANDLES	ROW_CACHE_PCT_FREE
DB_HANDLES_CACHED	SAVEPOINTS
DC_*(All DC_parameters)	SCN_INCREMENT
DDL_LOCKS	SCN_THRESHOLD
ENQUEUE_DEBUG_ MULTI_INSTANCE	SORT_READ_FAC
ENQUEUE_HASH	USE_ROW_ENQUEUES

Obsolete Parameters

ENQUEUE_LOCKS	USER_SESSIONS
FREE_LIST_INST	WAIT_FOR_SYNC

If you use Version 6 National Language Support, you must alter your Oracle7 parameter file to include the appropriate initialization parameters. Additionally, the specification of the NLS_SORT parameter has changed. For more information on NLS_SORT, see *Oracle7 Server Reference.*

Renamed Parameter	**New Name**
MI_BG_PROCS	GC_LCK_PROCS

New Parameters

AUTO_MOUNTING	MTS_MAX_DISPATCHERS
CHECKPOINT_PROCESS	MTS_MAX_SERVERS
COMMIT_POINT_STRENGTH	MTS_SERVERS
CURSOR_SPACE_FOR_TIME	MTS_SERVICE
DB_BLOCK_CHECKPOINT_BATCH	NLS_CURRENCY
DB_DOMAIN	NLS_DATE_FORMAT
DB_MOUNT_MODE	NLS_DATE_LANGUAGE
DISCRETE_TRANSACTIONS_ ENABLED	NLS_ISO_CURRENCY
DISTRIBUTED_LOCK_TIMEOUT	NLS_LANGUAGE
DISTRIBUTED_RECOVERY_ CONNECTION_HOLD_TIME	NLS_NUMERIC_CHARACTERS
DISTRIBUTED_TRANSACTIONS	NLS_TERRITORY
GC_LCK_PROCS	OPEN_COMPATIBLE
GLOBAL_NAMES	OPEN_MOUNTS
INSTANCE_NUMBER	OPTIMIZER_COMP_WEIGHT
LABEL_CACHE_SIZE	OPTIMIZER_OVERRIDE
LICENSE_MAX_SESSIONS	OS_AUTHENT_PREFIX
LICENSE_MAX_USERS	OS_ROLES
LICENSE_SESSIONS_WARNING	RECOVERY_COMPATIBLE
LOG_ARCHIVE_BUFFERS	REMOTE_OS_AUTHENT
LOG_ARCHIVE_BUFFER_SIZE	REMOTE_OS_ROLES

New Parameters

LOG_ARCHIVE_FORMAT	RESOURCE_LIMIT
LOG_CHECKPOINT_TIMEOUT	SHARED_POOL_SIZE
MAX_ENABLED_ROLES	SMALL_TABLE_THRESHOLD
MAX_ROLLBACK_SEGMENTS_MLS	SORT_AREA_RETAINED_SIZE
MLS_LABEL_FORMAT	TEMPORARY_TABLE_LOCKS
MTS_DISPATCHERS	THREAD
MTS_LISTENER_ADDRESS	

Data Dictionary Changes

This section contains the following topics:

- Views
- Dynamic Performance Tables
- Trusted Oracle Views

This section describes the changes to the Oracle7 data dictionary. See *Oracle7 Server Reference* for descriptions of the standard tables and views that are available to developers and lists all tables and views.

Views

Oracle7 introduces a new set of data dictionary views. The views are created by the database administrator under the schema of user SYS. Several data dictionary views from Version 6 have either changed or become obsolete. Others have been renamed. In addition, new views have been added in Oracle7. The following tables summarize the changes:

Obsolete Views

ACCESSIBLE_COLUMNS	DBA_AUDIT_RESOURCE
ACCESSIBLE_TABLES	DBA_CROSS_REPS
CONSTRAINT_COLUMNS	MYPRIVS
CONSTRAINT_DEFS	USER_AUDIT_CONNECT
DBA_AUDIT_CONNECT	USER_AUDIT_RESOURCE
DBA_AUDIT_DBA	USER_CROSS_REFS

Changed Views

ALL_CONSTRAINTS	DBA_ROLLBACK_SEGS
ALL_DEF_AUDIT_OPTS	DBA_TABLES
ALL_INDEXES	DBA_TAB_COLUMNS
ALL_OBJECTS	DBA_USERS
ALL_TAB_COLUMNS	TABLE_PRIVILEGES
ALL_TABLES	USER_AUDIT_TRAIL
COLUMN_PRIVILEGES	USER_CLUSTERS
DBA_AUDIT_EXISTS	USER_CONSTRAINTS
DBA_AUDIT_TRAIL	USER_INDEXES
DBA_CLUSTERS	USER_OBJECTS
DBA_CONSTRAINTS	USER_TABLES
DBA_INDEXES	USER_TAB_COLUMNS
DBA_OBJECTS	USER_USERS

Renamed Views

Old Name	New Name
ALL_COL_GRANTS	ALL_COL_PRIVS
ALL_COL_GRANTS_MADE	ALL_COL_PRIVS_MADE
ALL_COL_GRANTS_RECD	ALL_COL_PRIVS_RECD
ALL_TAB_GRANTS	ALL_TAB_PRIVS
ALL_TAB_GRANTS_MADE	ALL_TAB_PRIVS_MADE
ALL_TAB_GRANTS_RECD	ALL_TAB_PRIVS_RECD
AUDIT_OPTION_MAP	STMT_AUDIT_OPTION_MAP
DBA_COL_GRANTS	DBA_COL_PRIVS
DBA_TAB_GRANTS	DBA_TAB_PRIVS
DBA_SYS_AUDIT_OPTS	DBA_STMT_AUDIT_OPTS
DBA_TAB_AUDIT_OPS	DBA_OBJ_AUDIT_OPS
USER_COL_GRANTS	USER_COL_PRIVS
USER_COL_GRANTS_MADE	USER_COL_PRIVS_MADE
USER_COL_GRANTS_RECD	USER_COL_PRIVS_RECD
USER_TAB_AUDIT_OPTS	USER_OBJ_AUDIT_OPTS
USER_TAB_GRANTS	DBA_TAB_PRIVS

Renamed Views

USER_TAB_GRANTS_MADE	USER_TAB_PRIVS_MADE
USER_TAB_GRANTS_RECD	USER_TAB_PRIVS_RECD

New Views

ALL_DEPENDENCIES	DBMS_ALERT_INFO
ALL_ERRORS	DBMS_LOCK_ALLOCATED
ALL_OBJECT_SIZE	DEPTREE
ALL_SNAPSHOTS	EXCEPTIONS
ALL_SOURCE	GLOBAL_NAME
ALL_TRIGGERS	IDEPTREE
CHAINED_ROWS	INDEX_HISTOGRAM
DBA_2PC_PENDING	INDEX_STATS
DBA_2PC_NEIGHBORS	PLAN_TABLE
DBA_AUDIT_OBJECT	PUBLIC_DEPENDENCY
DBA_AUDIT_SESSION	RESOURCE_COST
DBA_AUDIT_STATEMENT	ROLE_ROLE_PRIVS
DBA_BLOCKERS	ROLE_SYS_PRIVS
DBA_CONSTRAINTS	ROLE_TAB_PRIVS
DBA_DDL_LOCKS	ROLE_TAB_TABS
DBA_DEPENDENCIES	SESSION_PRIVS
DBA_DML_LOCKS	SESSION_ROLES
DBA_ERRORS	USER_AUDIT_OBJECT
DBA_LOCKS	USERT_AUDIT_SESSION
DBA_OBJECT_SIZE	USER_AUDIT_STATEMENT
DBA_PROFILES	USER_DEPENDENCIES
DBA_ROLE_PRIVS	USER_ERRORS
DBA_ROLES	USER_OBJECT_SIZE
DBA_SNAPSHOTS	USER_RESOURCE_LIMITS
DBA_SNAPSHOT_LOGS	USER_ROLE_PRIVS
DBA_SOURCE	USER_SNAPSHOTS
DBA_SYS_PRIVS	USER_SNAPSHOT_LOGS
DBA_TRIGGERS	USER_SOURCE

New Views

DBA_WAITERS	USER_SYS_PRIVS
	USER_TRIGGERS

Additional Views

These are new tables and views that have been created for use by the Oracle Server or Oracle utility programs. They are mentioned for the sake of completeness, but are of little interest to the user or administrator.

CODE_PIECES	LOADER_TAB_INFO
CODE_SIZE	LOADER_TRIGGER_INFO
ERROR_SIZE	PARSED_PIECES
LOADER_COL_INFO	PARSED_SIZE
LOADER_CONSTRAINT_INFO	SOURCE_SIZE
LOADER_INDCOL_INFO	STMT_AUDIT_OPTION_MAP
LOADER_IND_INFO	SYSTEM_PRIVILEGE_MAP
LOADER_PARAM_INFO	TABLE_PRIVILEGE_MAP

Dynamic Performance Tables

This section lists the changes to the dynamic performance tables.

Changed V$ Views

V$ACCESS	V$PROCESS
V$FILESTAT	V$ROLLNAME
V$LATCH	V$ROLLSTAT
V$LOCK	V$SESSION
V$LOGFILE	V$WAITSTAT
V$PARAMETER	

New V$ Views

V$ARCHIVE	V$OPEN_CURSOR
V$BACKUP	V$QUEUES
V$CIRCUITS	V$RECOVERY_LOG
V$DATABASE	V$RECOVER_FILE
V$DATAFILE	V$REQDIST
V$DB_OBJECT_CACHE	V$SESSION_WAIT
V$DISPATCHERS	V$SGASTAT

New V$ Views

V$ENABLEDPRIVS	V$SHARED_SERVERS
V$LIBRARYCACHE	V$SQLAREA
V$LICENSE	V$SQLTEXT
V$LOADCSTAT	V$THREAD
V$LOADSTAT	V$TIMER
V$LOG	V$TYPE_SIZE
V$LOG_HISTORY	V$VERSION
V$LOGHIST	V$VERSION
V$NLS_PARAMETERS	

Trusted Oracle Views

The following views and dynamic tables are new in Trusted Oracle7:

ALL_LABELS	V$SECONDARY
ALL_MOUNTED_DBS	V$SYSLABEL

Version 6 Compatibility

This section contains the following topics:

- Version 6 to Oracle7 Migration Utility
- Release 7.0 Backward Compatibility
- Version 6 SQL Compatibility Mode
- CATALOG6.SQL
- EXPVEW6.SQL
- EXPEOB6.SQL

Version 6 to Oracle7 Migration Utility

The Oracle7 Migration Utility converts a Version 6 database into Oracle7 form. The conversion happens “in place”; the database does not need to be reloaded from backup files. For more information, see Chapter 4 “The Migration Utility”.

Release 7.0 Backward Compatibility

Release 7.0 is completely backward compatible with Version 6 of the Oracle7 Server.

Warning: If the COMPATIBILITY parameter is set to Version 6, the functionality of certain Release 7.0 features will not be available.

Version 6 SQL Compatibility Mode

Each of the Oracle7 utilities makes it possible to select Version 6 SQL command compatibility. In SQL*DBA, for example, the SET COMPATIBILITY command is used. For more information on the SET command, see the SQL*DBA Commands section of *Oracle7 Server Utilities*. For more information on the effects of setting the compatibility mode, see *Oracle7 Server SQL Reference*. The effects include the following:

- CHAR datatypes create variable–length columns in CREATE TABLE statements, instead of the fixed–length columns they would ordinarily create in Oracle7 mode.
- Version 6 syntax for integrity constraints is recognized in CREATE TABLE statements, instead of the new Oracle7 syntax.
- PCTINCREASE cannot be specified for Oracle7 rollback segments. (The specification is allowed in V6 compatibility mode, but is ignored.)
- MAXEXTENTS cannot be specified for Oracle7 rollback segments. (It is allowed in V6 compatibility mode.)

CATALOG6.SQL

This command file augments the Oracle7 data dictionary by re–creating obsolete Version 6 views and views that were replaced in Oracle7. The file DROPCAT6.SQL can be used later to undo these changes to the Oracle7 data dictionary.

EXPVEW6.SQL

Running EXPVEW6.SQL allows remote Version 6 sites to do exports from Oracle7 databases. See *Oracle7 Server Utilities* for more details.

EXPEOB6.SQL

This script lists the objects that are *excluded* when Version 6 sites do exports from an Oracle7 database. See *Oracle7 Server Utilities* for more details.

New and Renamed SQL Scripts

The names of many SQL scripts were changed in order to make them easier to identify. In addition, many new scripts were added to support Oracle7's extended functionality. The new and changed scripts are listed below.

By convention, scripts that begin with CAT are intended to be run by SYS. They create views and tables in the data dictionary. Scripts that begin with DBMS are also run by SYS. They create stored procedures, functions, and packages. Scripts that begin with UTL are intended to be run by individual users. For more information, see the *Oracle7 Server Administrator's Guide* and *Oracle7 Server Reference.*

Renamed Scripts

Old Name	New Name
AUDIT.SQL	CATAUDIT.SQL
BLOCKING.SQL	CATBLOCK.SQL
BSTAT.SQL	CATBSTAT.SQL
DBA_SYN.SQL	CATDBSYN.SQL
DISPIDXS.SQL	UTILDXSD.SQL
ESTAT.SQL	UTLIDXSS.SQL
EXPVEW.SQL	CATEXP.SQL
IDXSTAT.SQL	UTLESTAT.SQL
LOCKTREE.SQL	UTLLOCKT.SQL
MONITOR.SQL	UTLMONTR.SQL
NOAUDIT.SQL	CATNOAUD.SQL
ONEIDXS.SQL	UTILDXSO.SQL
PSVIEWS.SQL	CATPARR.SQL
SAMPLE1.SQL..SAMPLE4.SQL	UTLSAMPL.SQL (combined)
ULVIEW.SQL	CATLDR.SQL
XPLAINPL.SQL	UTLXPLAN.SQL

New Scripts

CATEXP6.SQL	DBMSALRT.SQL	UTLCHAIN.SQL
CATNOPRC.SQL	DBMSLOCK.SQL	UTLDTREE.SQL
CATPROC.SQL	DBMSMAIL.SQL	UTLEXCPT.SQL
CATSTAT.SQL	DBMSPIPE.SQL	UTLMAIL.SQL
	DBMSSNAP.SQL	
	DBMSTRST.SQL	
	DBMSUTIL.SQL	

Other Changes

This section contains the following topics:

- Larger Control Files
- More Data Files
- Fewer Data Blocks
- MAXEXTENTS
- Change in Use of PCTINCREASE
- Assigning Tablespace Quotas
- Accessing Tables During Index Creation
- Blocks in Rollback Segments Dedicated to Transactions
- Messages and Codes
- ALERT Filenames
- TRACE Files

Larger Control Files

Oracle7 control files can be larger than they were in Version 6, although the maximum size is still operating system dependent. On most systems, the maximum has been increased from 500 blocks to 2500 blocks.

OSDoc

Additional Information: Consult your operating system–specific Oracle documentation for more information.

More Data Files

Although the maximum number of operating system files that can be in the database is operating system dependent, the maximum has been increased. In Oracle Version 6, the maximum was typically 255 files. In Oracle7, the maximum is typically 1022 or more.

OSDoc

Additional Information: Consult your operating system–specific Oracle documentation for more information.

Fewer Data Blocks

The number of data blocks per database may have shrunk for your system's implementation of the Oracle7 Server. If it has, then the database blocksize may need to be increased in order to create a database of the same size.

MAXEXTENTS

The default value for MAXEXTENTS depends on the database block size in Oracle7, as well as the maximum value. The details are operating system specific.

OSDoc

Additional Information: Consult your operating system–specific Oracle documentation for more information.

Change in Use of PCTINCREASE

When a new extent is allocated in Oracle7, the size of the new extent is based on the last automatically allocated extent in that segment. As a result, changing PCTINCREASE produces an incremental change in extent size, regardless of how many extents have been previously allocated. (In Version 6, the size of the new extent depended on the total number of allocated extents, which magnified the effects of changes to PCTINCREASE.) For details, consult the *Oracle7 Server Administrator's Guide.*

Assigning Tablespace Quotas

In Version 6, tablespace quotas were assigned with the GRANT command. In Oracle7, tablespace quotas are assigned with the CREATE USER and ALTERUSER commands.

The Version 6 syntax made it appear to be legal to assign tablespace quotas to PUBLIC, although in fact it was not. The Oracle7 syntax precludes that possibility.

Accessing Tables During Index Creation

In Version 6, it was not possible to select from a table while a CREATE INDEX statement was operating. In Oracle7, rows can be selected from tables while indexes are being created.

Blocks in Rollback Segments Dedicated to Transactions

Blocks in Oracle7 rollback segments are not shared by multiple transactions. Instead, each Oracle7 rollback segment block is exclusively dedicated to a single transaction. As a result, you may need to increase the size of your rollback segments if your transactions are very small relative to the rollback segment block size.

Messages and Codes

Because of increased functionality, many new messages have been added to Oracle7, and some Version 6 messages have become obsolete. If your programs use any hardcoded message numbers, please verify that the message number and message text have remained the same for any messages for which your programs test.

In addition, some message numbers have been reused. See *Oracle7 Server Messages* for a listing of the message codes used in Oracle7.

ALERT Filenames

Previously, ALERT filenames had the following format:

`ALERT_`*`instanceName`*`.LOG`

In Oracle7, they usually have the following format:

`nodename_instanceName``_ALERT.LOG`

OSDoc

Additional Information: The format depends on your operating system. Consult your operating system–specific Oracle documentation for details.

TRACE Files

Previously, TRACE filenames had the following format:

`imageName_instanceID_process#``.LOG`

In Oracle7, they usually have the following format:

`nodename_instanceID_imageName``_FG_`*`process#`*`.LOG`

Previously, each background process wrote a trace file at instance startup. This is no longer true with Oracle7.

OSDoc

Additional Information: The format depends on your operating system. Consult your operating system–specific Oracle documentation for details.

APPENDIX

B

Summary of Changes in Oracle7, Release 7.1

This appendix describes changes in Oracle7, Release 7.1. The following topics are:

- Terminology
- Functionality Enhancements
- SQL Syntax Changes
- Changes to Supplied Packages
- SQL*Net and the Multi–Threaded Server
- New Features of the Oracle Precompilers, Release 1.6
- Advanced Replication Option
- Initialization Parameter Changes
- Data Dictionary Views and Dynamic Performance Tables for Release 7.1
- Release 7.1 Backward Compatibility

Terminology

New terms have been introduced with Oracle7, Release 7.1. The new terms and their definitions follow:

Parallelism	Parallelism allows multiple processes to work together to process a single task simultaneously, such as a SQL query, CREATE INDEX, or DIRECT LOAD using SQL*LOADER.
Query Coordinator	Query coordinators separate the execution functions of an information processing task into parallel pieces, distribute the pieces to several query servers, coordinate the results from all of the servers, and send the results back to the user.
Query Server	Query servers operate in parallel to execute processing commands provided by a query coordinator.
Refresh Group	Refresh groups allow you to update a collection of snapshots consistently to a single point in time by creating a *snapshot refresh group*.
Symmetric Replication	Symmetric replication allows multiple copies of data to be maintained at different sites in a distributed environment. Immediate, local access to data is provided and systems are allowed to function autonomously, even when other systems in the distributed environment or network fail.

Functionality Enhancements

This section contains the following topics:

- Server Manager
- Procedural Option
- Symmetric Replication
- Consistent Snapshot Refresh
- Enhancements to SQL and PL/SQL
- Read–Only Tablespaces
- Parallel Recovery
- Improved Security When Connecting to Remote Databases
- Parallel Query Option
- Dynamic SQL Supplied Package
- SQL*Net Oracle Names
- Link to Oracle Office
- Mapping of Trusted Oracle7 Labels during Import
- Referencing PL/SQL Functions in SQL Expressions
- Referencing Sequences in Distributed SQL Statements
- Using PL/SQL Functions
- Restrictions
- Privileges Required

Although primarily a maintenance release, Oracle7, Release 7.1 contains several important functional enhancements. These enhancements include the following:

Server Manager

Oracle has a new database administration tool with a graphical user interface. This tool is documented in the *Oracle Server Manager User's Guide*. Release notes for Server Manager installation and startup are included in the file SVRMGR.TXT.

Procedural Option

Procedural option features previously available only to users of the procedural option, such as stored procedures and triggers, are now available to all users. Although this is no longer a separate option, it is still installed separately. Additionally, you may now have multiple triggers of the same type on a single table.

Symmetric Replication

Oracle's symmetric replication facility, which lets you maintain multiple updatable copies of data at different sites in a distributed environment, is available starting with Oracle7, Release 7.1.6 and is described in *Oracle7 Server Distributed Systems, Volume II*.

Consistent Snapshot Refresh

Snapshot refresh groups now allow you to refresh a collection of related snapshots to a transaction–consistent point in time. For example, this allows you to preserve master detail relationships between snapshots.

Enhancements to SQL and PL/SQL

Oracle7, Release 7.1 conforms to the entry level ANSI SQL92 (ANSI X3.135–1992 and ISO 9075:1992) standard requirements. This enhancement includes the use of the optional AS keyword to define column or expression aliases as well as allowing column or expression aliases to be used in the ORDER BY clause. This enhancement also allows administrators to enforce SELECT privileges on tables for deletes and updates that require a scan of the table.

SQL has been enhanced to allow PL/SQL user–defined functions to be called from expressions in SQL statements.

Read–Only Tablespaces

You can now create read–only tablespaces to avoid backup and recovery of static data. For multimedia or archival applications, read–only tablespaces can be stored on low cost per bit media, such as CD–ROM and WORM drives.

Parallel Recovery

You can now invoke multiple processes to recover your database concurrently, thus decreasing recovery time.

Improved Security When Connecting to Remote Databases

When you log on to a remote database, the password is now encrypted when it is sent over the network. Additionally, remote O/S–authorized DBA and OPER sessions over non–secure connections are now supported, even when a database is not available.

Parallel Query Option

Oracle7 with the new parallel query option improves the performance of bulk operations, such as data loading, queries, and index creation, by using parallel processing on multiple CPUs.

Dynamic SQL Supplied Package

The new DBMS_SQL package allows you to use dynamic SQL in stored procedures and PL/SQL anonymous blocks. This allows you to develop more flexible and general–purpose procedures.

SQL*Net Oracle Names

The SQL*Net Oracle Names feature allows you to use a database link without defining a connect string in the database. This is done by providing a central location for database link definitions.

Link to Oracle Office

The DBMS_OFFICE supplied package allows you to send a message to Oracle Office, in much the same manner that the DBMS_MAIL package, available with Release 7.0 of the Oracle Server, allows you to send a message to Oracle*Mail. The DBMS_OFFICE package is available with Oracle*Office and is described in the *Oracle Office Server Release Notes, Release 2.0.12*.

Mapping of Trusted Oracle7 Labels during Import

Exported Trusted Oracle MLSLABELs can now be imported to Trusted Oracle databases with different labels.

Referencing PL/SQL Functions in SQL Expressions

In Release 7.1 of the Oracle Server, you can include user–written PL/SQL functions in SQL expressions. See the *Oracle7 Server Application Developer's Guide* for more information on this feature.

By using PL/SQL functions in SQL statements, you can do the following:

- Increase user productivity by extending SQL. Expressiveness of the SQL statement increases where activities are too complex, too awkward, or unavailable with SQL.
- Increase query efficiency. Functions in the WHERE clause of a query can filter data using criteria that would otherwise have to be evaluated by the application.
- Manipulate character strings to represent special datatypes (for example, latitude, longitude, or temperature).
- Provide parallel query execution. If the query is parallelized, SQL statements in your PL/SQL function may be executed in parallel also (using the parallel query option).

Referencing Sequences in Distributed SQL Statements

In Oracle7, Release 7.1, the following object types referenced in distributed SQL statement may be located on different databases:

- updated tables
- tables locked by the FOR UPDATE clause of a SELECT statement
- tables from which LONG columns are selected

- sequences

Using PL/SQL Functions

PL/SQL functions must be created as top–level functions or declared within a package specification before they can be named within a SQL statement. PL/SQL functions are created as top–level functions by using the CREATE FUNCTION statement. Packaged functions are specified within a package with the CREATE PACKAGE statement and created in the CREATE PACKAGE BODY statement.

See the *Oracle7 Server Application Developer's Guide* for more information about creating functions and packages.

Restrictions

PL/SQL functions called from SQL statements cannot violate the following restrictions:

- The functions cannot contain OUT or IN OUT parameters; all parameters must be IN parameters.
- The functions cannot update the database.
- Only local functions that are referenced in a SELECT list, VALUES clause of an INSERT statement, or SET clause of an UPDATE statement can update a variable defined in a package.
- Remote functions cannot read or write package state.
- Functions that read or write package state cannot take advantage of the parallel query option.
- Functions cannot be called from a CHECK constraint or DEFAULT clause in a CREATE or ALTER TABLE command.

See the *Oracle7 Server Application Developer's Guide* for more information on PL/SQL functions and packages.

Privileges Required

To call a PL/SQL function from SQL, you must either own or have EXECUTE privileges on the function. To select from a view defined with a PL/SQL function, you are required to have SELECT privileges on the view. No separate EXECUTE privileges are needed to select from the view.

SQL Syntax Changes

This section outlines changes to SQL syntax introduced with Oracle7, Release 7.1 of the Oracle Server.

This section contains the following topics:

- SELECT List
- FIPS SQL Flagging
- SELECT Privileges When Updating or Deleting
- Outer Joins
- ALTER CLUSTER
- ALTER DATABASE
- ALTER SESSION
- ALTER TABLE
- ALTER TABLESPACE
- CREATE INDEX
- CREATE TABLE
- ALTER and CREATE SNAPSHOT and Index Storage

SELECT List

Column aliases in the SELECT list can optionally be separated from their expressions by the new AS keyword. Although the AS keyword adds no functionality, it may make SQL statements easier to read. These column aliases effectively rename the SELECT list items. For example:

```
SELECT empno, ename AS name
    FROM emp
```

For example, the statement

```
SELECT empno, ename, sal+comm AS comp
    FROM emp
    ORDER BY sal+comm
```

which uses an ORDER BY column expression, is equivalent to the following statement which uses an ORDER BY column expression alias:

```
SELECT empno, ename, sal+comm AS comp
    FROM emp
    ORDER BY comp
```

Both statements result in column names of EMPNO, ENAME, and COMP, and both order the results by SAL+COMM. The difference is that the second statement uses the column expression's alias in the ORDER BY clause rather than the column expression itself. This saves typing and can prevent errors, because the expression need not be retyped.

FIPS SQL Flagging

The Federal Information Processing Standard for SQL (FIPS 127–2) requires a way to identify SQL statements that use vendor–supplied extensions. Oracle provides a FIPS flagger through interactive SQL to help you write portable applications.

When SQL flagging is set to on, your SQL statements are checked to see whether they include extensions that go beyond the ANSI/ISO SQL92 standard. If any non–standard constructs are found, the Oracle Server flags them as errors and displays the violating syntax.

SELECT Privileges When Updating or Deleting

In conformance with Entry level SQL92, Oracle7, Release 7.1 allows the security administrator to require that users have SELECT privilege on a table when executing an UPDATE or DELETE statement that references table column values in a WHERE or SET clause. This feature can prevent unauthorized users from learning about the table's data through a covert channel.

Compatibility with Previous Releases

To ensure compatibility with Oracle7, Release 7.0 or Version 6, when migrating to release 7.1, keep the SQL92_SECURITY parameter at FALSE, thus inhibiting this feature.

Outer Joins

Oracle7 is more stringent than Oracle Version 6 in rejecting two types of semantically undefined outer join queries:

- The OR logical operator cannot combine two conditions if either contains the outer join operator (+). Also, a condition cannot compare a column marked with the (+) operator to another expression using the IN comparison operator. Error 1719, outer join operator (+) not allowed in operand of OR or IN, is returned for such invalid conditions. If you have applications that issue queries with such conditions, replace them with equivalent queries that use the UNION or UNION ALL set operators instead.
- If a condition compares a column marked with the (+) operator to a subquery, Oracle7 returns the following message.

```
ORA–01799: a column may not be outer–joined to a subquery
```

Oracle, Version 6 ignored the (+) operator in such conditions. If you have applications that issue queries with such conditions, remove the (+) from them and they will behave in Oracle7 as they did in Oracle, Version 6.

For more information, see *Oracle7 Server SQL Reference.*

ALTER CLUSTER

This command has a PARALLEL clause and a CACHE clause to support the parallel query option. See *Oracle7 Server SQL Reference* for more information on this command.

ALTER DATABASE

This command has a RESET COMPATIBILITY option for compatibility control.

You must have ALTER DATABASE system privilege and your instance must have the database open for you to issue this command.

The RECOVER option of this command has changed to include a PARALLEL clause for use with the parallel recovery feature. See *Oracle7 Server SQL Reference* for more information on this command.

ALTER SESSION

This command has a new SET FLAGGER option to support flagging of SQL extensions that go beyond the SQL92 standard for SQL. The SET FLAGGER option has four additional options: entry, intermediate, full, and off.

This command also has a new option for closing cached cursors used by PL/SQL. Using the ALTER SESSION command with this option overrides the initialization parameter CLOSE_CACHED_OPEN_CURSORS for your current session. For more information about the initialization parameter, see *Oracle7 Server Reference.*

This command also has a new option for specifying the size of the session cursor cache. The syntax is:

```
ALTER SESSION SET SESSION_CACHED_CURSORS = integer
```

The integer specified can be any positive integer, but the maximum value is operating–system dependent.

See *Oracle7 Server SQL Reference* for more information on using the ALTER SESSION command.

ALTER TABLE

This command has a PARALLEL clause and a CACHE clause to support the parallel query option. See *Oracle7 Server SQL Reference* for more information on this command.

ALTER TABLESPACE This command has READ ONLY and READ WRITE options to support read–only tablespaces. See *Oracle7 Server SQL Reference* for more information on this command.

CREATE INDEX This command has a PARALLEL clause to support the parallel query option. See *Oracle7 Server SQL Reference* for more information on this command.

CREATE TABLE This command has a PARALLEL clause and a CACHE clause to support the parallel query option. See *Oracle7 Server SQL Reference* for more information on this command.

ALTER and CREATE SNAPSHOT and Index Storage The creation of a simple snapshot automatically causes an index to be created. The CREATE and ALTER SNAPSHOT commands now support a USING INDEX clause to allow users to specify the tablespace and storage parameters to use for this index. If the USING INDEX clause is not specified, the index is created with the same tablespace and storage parameters as the snapshot.

Changes to Supplied Packages

The distribution mechanism for supplied packages, described in the *Oracle7 Server Application Developer's Guide*, was changed in release 7.1. The public package specifications continue to reside in the DBMS*.SQL files; for example, DBMSPIPE.SQL. The package bodies now reside in separate files, which follow a similar naming convention to the package specification file. The package body files are generally named PRVT*.SQL; for example PRVTPIPE.SQL.

However, the following packages have public package specifications, but the bodies are stored in binary format, which has the extension .PLB:

- DBMSSQL.SQL and PRVTSQL.PLB
- DBMSPEXP.SQL and PRVTPEXP.PLB
- DBMSJOB.SQL and PRVTJOB.PLB
- DBMSDFRD.SQL and PRVTDFRD.PLB
- DBMSSNAP.SQL and PRVTSNAP.PLB

To create a supplied package, you must run both of these files against your database. These scripts should be run automatically for you when you create a new database or upgrade an existing database.

OSDoc

Additional Information: Refer to your operating system–specific Oracle documentation for more information about the location and installation of these packages.

SQL*Net and the Multi–Threaded Server

Oracle7 fully supports SQL*Net, Version 1, either as a client or a server. Therefore, clients running SQL*Net, Version 1 can connect to Oracle7 Servers running SQL*Net, Version 1 for client / server access. Furthermore, Oracle7 Servers can communicate with one another for distributed queries, and (with the distributed option) for distributed update, two–phase commit and snapshots using SQL*Net, Version 1. The multi–threaded server architecture in Oracle7 requires use of SQL*Net, Version 2.

New Features of the Oracle Precompilers, Release 1.6

This section contains the following topics:

- SQL Standards Compliance
- Configuration Files
- EXEC TOOLS Statements
- AUTO_CONNECT Option
- Optional INAME and ONAME Keywords
- Two New SQLLIB Routines
- CHARF Datatype Specifier
- New Pro*FORTRAN Option
- Pro*C / C++, Release 2.0

Release 1.6 of the Oracle Precompilers was introduced with Oracle7, Release 7.1 and incorporated many new features designed to enhance precompiler functionality and application performance, as well as allow your applications to access new Oracle7, Release 7.1 capabilities.

Most users will upgrade to Release 1.6 at the same time they upgrade to Oracle7, Release 7.1. Use of Releases 1.3, 1.4 and 1.5 of the Oracle Precompilers is supported against Oracle7, Release 7.1. However, if you wish to use the enhanced functionality of Oracle7, Release 7.1, you must use the Release 1.6 of the Precompilers. For more information on the Oracle Precompilers, Release 1.6, see the *Programmer's Guide to the Oracle Precompilers.*

The following new features are introduced with the Oracle Precompilers, Release 1.6.

SQL Standards Compliance

The Oracle Precompilers, Release 1.6 supports the latest ANSI/ISO embedded SQL standard (SQL92) by letting you

- use the status variable SQLSTATE
- check the SQLCA warning flag SQLWARN(3) for data exceptions
- declare multiple host variables with the same name, provided their scopes do not overlap
- specify the USAGE IS BINARY clause (Pro*COBOL only)

Configuration Files

The Oracle Precompilers, Release 1.6 can use a configuration file containing preset command–line options. By default, a text file called the *system configuration file* is used. However, you can specify any of several alternative files, called *user configuration files*, on the command line.

EXEC TOOLS Statements

EXEC TOOLS statements support the basic Oracle Toolset (Oracle Forms, Oracle Reports, and Oracle Graphics) by providing a generic way to handle get, set, and exception callbacks from user exits.

AUTO_CONNECT Option

To increase compatibility with other precompilers, the Oracle Precompilers, Release 1.6 allow your program to log on to the default database without using the CONNECT statement. Simply specify the new precompiler option AUTO_CONNECT on the command line.

Optional INAME and ONAME Keywords

When specifying the names of your input and output files on the command line, the keywords INAME and ONAME are optional.

Two New SQLLIB Routines

The Oracle Precompiler runtime library, SQLLIB, contains two new routines: SQLGLS and SQLVCP.

- SQLGLS returns the text and length of the most recently parsed SQL statement. It also returns a two–digit code indicating the SQL operation (such as INSERT or DELETE) performed by the statement.
- SQLVCP returns the actual length (in bytes) of a VARCHAR data field. This is useful when your host–language compiler blank–pads the data field, making its actual length longer than its declared length. To get the actual (possibly padded) length of a VARCHAR data field, just pass the declared length to SQLVCP.

CHARF Datatype Specifier

The Oracle Precompilers, Release 1.6 introduce a new datatype specifier named CHARF. You can use this new datatype in EXEC SQL TYPE and VAR statements to equivalence host–language datatypes to the fixed–length ANSI datatype CHARF, regardless of the DBMS setting.

New Pro*FORTRAN Option

With Pro*FORTRAN, Release 1.6, the new option MULTISUBPROG controls the generation of COMMON statements and BLOCK DATA subprograms.

When MULTISUBPROG = YES, the precompiler generates COMMON statements and BLOCK DATA subprograms like its Release 1.5 predecessor.

When MULTISUBPROG = NO, the precompiler generates no COMMON statements or BLOCK DATA subprograms.

Pro*C/C++, Release 2.0

Pro*C / C++, Release 2.0 offers the following new features:

- preprocessor support
- the ability to declare a host structure that contains host variables
- the ability to declare host variables in a Declare Section
- the ability to use the EXEC SQL WHENEVER DO statement to call functions that take arguments and return a value

  Note: If you are using Pro*C source code that is not ANSI C or not Kernigan and Ritchie, you cannot upgrade directly to Release 2.0.

Advanced Replication Option

Symmetric replication allows multiple copies of data to be maintained at different nodes in a distributed environment. Changes to local copies of data are stored locally, and then periodically forwarded within separate transactions. If a remote node is unavailable, the changes remain in the local node and are propagated when the remote node becomes available. At any given point in time, some replicated copies may contain older data than other copies, but over time, the copies converge to the same value.

The Advanced Replication option, available with Release 7.1.6, supports the following features:

- Immediate, local access to data is provided.
- Nodes in the system can function autonomously, even when other nodes in the distributed environment or network fail.
- A symmetric, update–anywhere replication model ensures that all copies of data can be updated with updates automatically propagated to all other copies.
- Integrated, update conflict detection facilities are provided that automatically invoke application–level conflict resolution routines.
- Full transactional consistency is provided. Updates made by transactions to multiple tables are applied to remote copies of the tables, transactionally, to ensure data consistency and referential integrity between the tables.

For more information, see *Oracle7 Server Distributed Systems, Volume II.*

Initialization Parameter Changes

This section lists initialization parameters that are new or changed with Oracle7, Release 7.1. For a complete list of initialization parameters and their definitions, see *Oracle7 Server Reference.*

CACHE_SIZE_THRESHOLD	PARALLEL_MAX_SERVERS
CLOSE_CACHED_OPEN_ CURSORS	PARALLEL_MIN_SERVERS
COMPATIBLE	PARALLEL_SERVER_IDLE_ TIME
DB_DOMAIN	PRE_PAGE_SGA

DBLINK_ENCRYPT_LOGIN	RECOVERY_PARALLELISM
LOG_CHECKPOINTS_TO_ ALERT	REMOTE_LOGIN_ PASSWORDFILE
MAX_COMMIT_ PROPAGATION_DELAY	SESSION_CACHED_CURSORS
NLS_DATE_FORMAT	SNAPSHOT_REFRESH_ INTERVAL
OPTIMIZER_MODE	SNAPSHOT_REFRESH_KEEP_ CONNECTIONS
PARALLEL_DEFAULT_MAX_ INSTANCES	SNAPSHOT_REFRESH_ PROCESSES
PARALLEL_DEFAULT_MAX_ SCANS	SQL92_SECURITY
PARALLEL_DEFAULT_ SCANSIZE	

Data Dictionary Views and Dynamic Performance Tables for Release 7.1

This section contains the following topics:

- Data Dictionary Views
- Dynamic Performance Tables

Data Dictionary Views

This section lists data dictionary views that were new in Release 7.1.

Data Dictionary Views
DBA_SEGMENTS
DBA_TABLESPACES
PLAN_TABLE
PRODUCT_COMPONENT_VERSION
USER_SEGMENTS
USER_TABLESPACES

Dynamic Performance Tables

This section lists dynamic performance tables that were new in Release 7.1.

Dynamic Performance Tables	
V$COMPATIBILITY	V$OPTION
V$COMPATSEG	V$PQ_SESSTAT
V$CONTROLFILE	V$PQ_SLAVE
V$DATAFILE	V$PQ_SYSSTAT
V$LOCK	V$PWFILW_USERS
V$NLS_VALID_VALUES	

Release 7.1 Backward Compatibility

Release 7.1 is completely backward compatible with Version 6 and Release 7.0 of the Oracle7 Server. As a maintenance release, Release 7.1 is the normal maintenance path for users running Version 6 and Release 7.0 applications.

Warning: If the SQL*DBA COMPATIBILITY parameter is set to Version 6, the functionality of certain Release 7.1 features will not be available.

APPENDIX

C

Summary of Changes in Oracle7, Release 7.2

This appendix provides an overview of the changes in the Oracle7 Server, Release7.2. The topics included in this appendix are:

- Functionality Enhancements
- Recovery Enhancements
- Security Enhancements
- Performance Enhancements
- Administration Enhancements
- Oracle7, Release 7.2 Datatype Changes
- SQL Changes
- New Features of the Oracle Precompilers, Release 1.7
- Pro*C / C++, Release 2.1 New Features
- SQL*Module Default WITH INTERFACE Clause
- Non–blocking Oracle Call Interface (OCI)
- Initialization Parameter Changes
- Data Dictionary Changes
- Standards Compliance
- Release 7.2 Backward Compatibility
- New, Changed, and Obsolete SQL Scripts
- Other Changes

Functionality Enhancements

This section contains the following topics:

- Cursor Variables
- CREATE TABLE...AS SELECT
- UNRECOVERABLE
- CREATE INDEX

Cursor Variables

PL/SQL, Release 2.2 provides a new and convenient declarative mechanism to return multi–row result sets from stored procedures. Using a *cursor variable*, a calling program can obtain a *handle* that enables the retrieval of multiple rows from a stored procedure. Cursor variables allow you to pass cursors as parameters in your application.

Cursor variables are like C or Pascal pointers, which hold the memory location (address) of some object instead of the object itself, allowing you to allocate memory for the object dynamically.

The *cursor variables* feature results in reduced coding complexity by providing a structured way to encapsulate queries in a stored procedure.

For more information, see the *PL/SQL User's Guide and Reference.*

CREATE TABLE...AS SELECT

Table creation and population for CREATE TABLE...AS <subquery> statements can now be performed in parallel. This new functionality is an extension to the parallel query option of Oracle7, Release 7.1. Parallel execution of the CREATE TABLE...AS SELECT operation permits the insertion of the results of a query into a new table in parallel, in addition to executing the query itself in parallel.

The Create Table As Select feature permits the following:

- Summaries of large amounts of data can now be created, in the form of subsets, that can be repeatedly queried by multiple users on an ad hoc basis.
- Successive refinements of large data sets can now be created by applying series of highly complex queries. Typical applications involve large datafiles that contain redundant or irrelevant information.

For more information about CREATE TABLE...AS SELECT, see *Oracle7 Server SQL Reference* and "SQL Changes" on page C – 16.

UNRECOVERABLE

A new keyword, UNRECOVERABLE, allows you to avoid redo logging during the creation of tables or indexes (regardless of whether they are created in parallel). Objects created with the UNRECOVERABLE keyword cannot be recovered in the event of media failure, but the creation time will be significantly shorter.

For more information about UNRECOVERABLE, see *Oracle7 Server SQL Reference* and "SQL Changes" on page C – 16.

CREATE INDEX

CREATE INDEX and CREATE TABLE AS SELECT operations can now be performed faster. Release 7.2 offers you the option to disable logging for the CREATE INDEX and CREATE TABLE AS SELECT operations. The new Release 7.2 option is available for parallel and serial execution of the CREATE INDEX and CREATE TABLE AS SELECT operations.

The syntax for the CREATE INDEX command is

```
CREATE INDEX <index name> ON <tablename> (column name)
```

For more information about CREATE INDEX, see *Oracle7 Server Tuning* and *Oracle7 Server SQL Reference* and "SQL Changes" on page C – 16.

Recovery Enhancements

This section contains the following topics:

- Checksums
- Media Recovery

Checksums

A new feature in Release 7.2 facilitates earlier detection of database corruption. Timely detection of database corruption also aids in diagnosing the causes of corruption.

A new initialization parameter, LOG_BLOCK_CHECKSUM, can be set to write a checksum in the header of every block of the redo log as it is written out of the log buffer. The checksums, if present, are validated when the log is read for archiving or recovery. Similarly, data blocks may also have checksums to detect corruption more easily.

The initialization parameter DB_BLOCK_CHECKSUM enables the checksum option.

For more information, see the *Oracle7 Server Administrator's Guide.*

Media Recovery

A new mechanism avoids performing media recovery if a database crash occurs while one or more files is in online backup mode.

For more information, see the *Oracle7 Server Administrator's Guide.*

Security Enhancements

This section contains the following topics:

- Network Security Enhancements
- Trusted Stored Procedures
- Secure PL/SQL Code (PL/SQL Wrapper)
- Secure Connections with Encrypted Password

Network Security Enhancements

The Oracle7 external authentication mechanism now extends to include support for network authentication services. Distributed installations can now validate database access and roles by taking advantage of network authentication services.

Specific network security features are

- Secure external authentication. The database trusts the external authorization service for authorization, even if the underlying protocol itself is not secure.
- Secure database links. Users who are authenticated by a network service that supports proxies can use the proxy when making remote access with an anonymous database link (one without a specified user name) to make secure connection.
- Network roles. The database recognizes roles granted by network services similar to the current OS roles feature.
- Network SYSDBA/SYSOPER privilege. Remote *internal* connections can be authorized by the network service, rather than using the Oracle7 password file utility in Release 7.1.

For more information, see *Oracle7 Server Concepts.*

Trusted Stored Procedures

The release 7.2, Trusted Stored Procedures enables application developers to

- write procedures that would perform MAC–privileged operations without having to grant the privileges explicitly to all executors of the procedure.
- write procedures that enable unprivileged users to perform MAC–privileged operations in a carefully controlled manner by using software inside the server.

For more information, see the *Trusted Oracle7 Server Administrator's Guide*.

Secure PL/SQL Code (PL/SQL Wrapper)

- Application developers can now deliver PL/SQL code in a precompiled format using the new PL/SQL Wrapper feature. The name of the new executable is *Wrap*. Intellectual property is protected by hiding source code in a *binary source* form.

The PL/SQL Wrapper feature consists of two components:

- The PL/SQL Wrapper Compiler, which is a stand–alone utility that compiles PL/SQL source code files to files in PL/SQL Wrapper format.
- The PL/SQL Wrapper Loader, which is an extension to the PL/SQL compiler that enables the recognition and loading of PL/SQL compilation units in PL/SQL Wrapper format.

PL/SQL Wrapper offers the same portability and flexibility as the PL/SQL source code format. In particular, PL/SQL Wrapper provides

- Platform independence. Developers do not have to deliver multiple versions of the same unit.
- Dynamic loading. You can add new features without having to shut down and relink.
- Dynamic (load time) binding of unqualified external references.
- Automatic recompilation when depended–on units change specification.

For more information, see the *PL/SQL User's Guide and Reference*.

Secure Connections with Encrypted Passwords

To better protect the confidentiality of your password, Oracle7 releases, 7.1 and higher, can be configured to use encrypted passwords for client / server and server / server connections.

You can require that the password used to verify a connection always be encrypted by setting the following values.

- Set the ORA_ENCRYPT_LOGIN environment variable to TRUE on the client machine.
- Set the DBLINK_ENCRYPT_LOGIN server initialization parameter to TRUE.

If enabled at both the client and server, passwords will not be sent across the network "in the clear", but will be encrypted using a modified DES (Data Encryption Standard) algorithm.

The DBLINK_ENCRYPT_LOGIN parameter is used for connections between two Oracle servers (for example, when performing distributed queries). If you are connecting from a client, Oracle checks the ORA_ENCRYPT_LOGIN environment variable.

Oracle releases before release 7.1 are not designed to accept encrypted passwords. If you know that you will attempt connections from a release 7.1 server to earlier versions of the Oracle Server (for example, if you are using a database link for distributed queries) you must set DBLINK_ENCRYPT_LOGIN to FALSE for the connection to succeed. If you will be connecting from a new client to an older version of the server (for example, if you run Server Manager against an earlier version of Oracle) you must set ORA_ENCRYPT_LOGIN to FALSE.

For more information, see *Oracle7 Server Reference* and the *Oracle7 Server Administrator's Guide*.

Performance Enhancements

This section contains the following topics:

- Operational Efficiency and Oracle7 Parallel Query Features
- Hash Clusters
- Loadable Character Sets (NLS)
- National Language Support Enhancements
- Application Registration
- Index Fixed Tables

- Parallel Server Enhancements

Operational Efficiency and Oracle7 Parallel Query Features

Oracle7 offers new parallel features with the parallel query option. These new features include the following:

- parallel direct loads of a single table using SQL*Loader
- parallel index create
- parallel query
- parallel database recovery

For best performance, the Oracle Corporation recommends that your system meet the following requirements:

- You should have sufficient CPU bandwidth provided by a SMP or MPP system.
- Your media (raw volumes or file systems) should be striped across controllers and disks. A minimum of four controllers and disks begin to show performance improvements.

For best performance with SQL*Loader direct parallel load, place database files to be loaded on a separate disk or on striped volumes/file systems. If you are not implementing disk striping, the Oracle Corporation recommends that you only start as many direct parallel load sessions as you have datafiles. For striped disk implementation, start as many direct parallel load sessions as possible, but without causing CPU bottlenecks or I/O saturation.

For best performance with CREATE INDEX ... PARALLEL operations, the previous suggestions also apply. For some SMP systems, the degree of parallelism can be twice the number of CPUs, provided your volume or file system are on separate drives or are striped.

For a full discussion on all of the Oracle7 parallel query options, see *Oracle7 Server Tuning*.

Hash Clusters

Release 7.2 provides you with improved control over the creation of hash clusters. You can now specify an application–specific SQL expression, with some restrictions, as the hash function for a cluster. Choice of appropriate hash functions reduces collisions, resulting in better performance.

For more information, see *Oracle7 Server Concepts.*

Loadable Character Sets (NLS)

Release 7.2 loads character sets upon first reference. Instead of linking all character sets as static data, each character set is read into dynamic memory upon first reference. The size of the executable is thus reduced by eliminating character set data not in use during execution.

There are three new utility programs that facilitate the use of the NLS loadable character set. They are the *NLS Data Installation Utility*, the *NLS Configuration Utility*, and the *NLS Calendar Utility*.

The NLS Data Installation Utility

A default set of NLS objects is included with the NLS loadable character set you have purchased. At some future time, you may receive updates to the data in the form of text files, or you may create your own text–form object files. Such files must be converted into binary format and merged into the existing NLS object set. The *NLS Installation Utility* allows you to perform the necessary conversion to binary format.

The NLS Configuration Utility

Along with the binary object files, a boot file is also generated by the Data Object Installer. The boot file is used by NLS modules to identify and locate all the NLS objects that need to be loaded. There are three types of boot files:

- Installation boot file: This is the boot file that is included as part of the loadable character set.
- System boot file: This is a boot file that is generated by the *Data Installation Utility* and is used by the NLS to load the NLS objects.
- User boot file: This is a boot file that is created by the *Configuration Utility*; it contains only a subset of the System boot file.

The *NLS Configuration Utility* allows you to configure your boot files such that NLS loads only the NLS objects that you require.

NLS Calendar Utility

NLS provides support for a number of calendars in addition to the Gregorian calendar. All supported calendars are defined with data linked directly into the NLS. However, some of the calendars may require the addition of ruler eras (in the case of imperial calendars) or deviation days (in the case of lunar calendars) in the future. The *NLS Calendar Utility* enables you to define additional eras or deviation days in an external file without waiting for a new NLS release. The external file will be automatically loaded by NLS when calendar functions are executed.

For more information about the NLS, see *Oracle7 Server Reference.*

National Language Support Enhancements

Release 7.2 supports certain national language parameters as environment variables that can be altered by issuing appropriate operating system commands. Greater flexibility for multi–lingual applications is, thereby, provided by allowing more granular specification of NLS parameters. The environment variables include

NLS_DATE_FORMAT	NLS_CREDIT
NLS_DATE_LANGUAGE	NLS_MONETARY_CHARACTERS
NLS_SORT	NLS_CALENDAR
NLS_LIST_SEPARATOR	NLS_DISPLAY
NLS_DEBIT	

The UNICODE encoding scheme, UTF2, a variable–width, multi–byte format, is supported with Oracle7, Release 7.2 to support multi–byte character sets.

Release 7.2 supports five additional calendar systems: Japanese Imperial, ROC (Republic of China) Official, Thai Buddha, Arabic Hijrah, and the Arabic / Hebrew display character set.

For more information about NLS environment variables, see *Oracle7 Server Reference.*

Application Registration

A new package, DBMS_APPLICATION_INFO, provides a mechanism for registering, with the database, the name and activity of the application currently running.

For more information, see *Oracle7 Server Tuning.*

Index Fixed Tables

Certain V$ views now have a fast access method that can greatly improve query performance on these views. In previous versions, queries of V$SESSION, V$SESSTAT, and V$SQLAREA could cause performance bottlenecks due to very large sort and scan operations. With Release 7.2, queries using the numeric columns of these tables as the search key can perform significantly faster due to an “index” on the fixed tables’ numeric columns.

You can determine which columns have had indexes created on them by using the V$INDEXED_FIXED_COLUMN view. For more information, see *Oracle7 Server Reference.*

Parallel Server Enhancements

The Oracle7 Parallel Server has new performance enhancements. Descriptions of the functionality of the new features follow.

Space Management

Batch Allocate at Insert: Batch allocation of all of the blocks covered by a PCM (Parallel Cache Management) lock during insert eliminates the need to pre–allocate space for free list groups. Two situations can be significantly affected:

- If the table is partitioned, locks can be assigned to cover one contiguous range of blocks (using the GC_LOCKS_TO_FILES parameter). If a batch allocation has been performed, as described above, an INSERT will allocate all blocks covered by one lock to one free list group. Therefore, pre–allocation of extents will not be necessary, and no pinging will occur.
- In an application in which a lock is configured to cover a range of contiguous blocks, an INSERT will require fewer locks if it allocates all the blocks covered by one lock. There will also be less pinging because two instances will not be INSERTing into blocks covered by the same lock.

Freelist Group Size Determination: Release 7.2 allows you to determine how much of the space you have pre–allocated to a freelist has been used. The allocation of more space than is required is, therefore, prevented. A new package, DBMS_SPACE (in DBMSUTIL.SQL), returns the number of blocks that a table has for each freelist group.

Indexes Support Freelist Groups: Indexes now support freelist groups, but extents cannot be pre–allocated to the freelist groups.

Choosing a Freelist Group for Inserts: A new ALTER SESSION command allows a session to override the value of the INSTANCE parameter.

Locking Issues

Table Locks: A new command, ALTER TABLE <table_name> DISABLE TABLE LOCK, disables all table locks on a *per–table* basis. A corresponding ENABLE command re–enables table locking. The command must wait for all transactions that are active in the system to COMMIT or ROLLBACK.

Lock Recovery: Pre–Release 7.2 procedures recovered locks after an instance crash, one by one and in serial order, using the SMON process. Locks are now recovered by the lock processes after an instance crashes, which is a much faster process than the one used in pre–Release 7.2 databases. Also, multiple lock processes can operate in parallel.

New and Modified Views and Dynamic Performance Views: The following views and dynamic performance views were modified or added to allow better monitoring of PCM (Parallel Cache Monitoring).

FILE_LOCK	V$FALSE_PING
FILE_PING	V$LOCK_ELEMENT
V$BH	V$LOCK_WITH_COLLISIONS
V$CACHE	V$PING
V$_CACH_ACTIVITY	

Ping Efficiency

Early Release of Lock on Block: During a ping, the lock on a block is now released once the block has been written; this feature is now implemented in the DBWR. The buffer is available for reuse immediately after the write of the block completes, instead of waiting for the write of the entire batch of blocks to complete.

Ping Fairness: The number of changes that can be made, per second, to very hot blocks is now greatly increased. Previously, only one access could be made to a block in an instance before the block pinged out. Now, multiple accesses can be made to a block before it is pinged out.

Tuning Tools for the Parallel Server

Ping Per File Statistics: There is now a generic implementation of the ping per file statistics.

Other Caches: Release 7.2 provides statistics on pings in the library and dictionary cache.

Fixed Table Change: The lock element pointer is now added to V$BH (the buffer cache header fixed table). You may now determine which locks cover which blocks. You may then determine when false pinging occurs and which blocks are being falsely pinged.

For more information on Oracle7 Parallel Server enhancements, see *Oracle7 Parallel Server Concepts & Administration.*

Administration Enhancements

This section contains the following topics:

- Resizeable Datafiles
- Job Queues
- Displaying Space Information
- Network Authentication
- Activate SQL_TRACE for Remote Sessions

Resizeable Datafiles

The resizeable datafiles feature allows datafiles to automatically increase in size when more space is needed. The database administrator is thus provided with additional flexibility in the design, operation, and management of datafiles.

Benefits of the resizeable datafiles feature are

- Database files can be dynamically resized. Since datafiles can automatically extend when space is required, the need for manual, database administrator intervention is eliminated.
- Unused space in datafiles can now be reclaimed without any need to re–create an underlying tablespace.
- The need to add files to a database is reduced, which in turn, reduces administrative and performance overhead.

The Export Utility and Resizeable Datafiles

The Export Utility does not export FILEXT$ and does not preserve any information about the automatic extension feature of Resizeable Datafiles. Since FILE$ is exported, the exported size of the datafile reflects the last datafile resize operation. After the datafiles have been imported, the database administrator (DBA) must issue SQL commands to turn on the auto–extension feature.

For more information, see the *Oracle7 Server Administrator's Guide* and *Oracle7 Server SQL Reference.*

Job Queues

Release 7.2 provides a new feature, *job queues*, that enables you to schedule periodic execution of PL/SQL code. For example, job queues are used by Oracle's automatic snapshot refresh.

A new package, DBMS_JOB, can schedule execution of stored PL/SQL procedures and programs written with the Oracle Call Interface, Oracle Precompilers, SQL*Module, or within a stored PL/SQL subprogram.

The DBMS_JOB package allows you to submit, change, and remove jobs in a queue. Job queues are used by the symmetric application facility to refresh snapshots automatically and to schedule communication between master sites. You can use job queues to schedule administrative procedures to be performed at periodic intervals.

For more information, see the *Oracle7 Server Administrator's Guide.*

Displaying Space Information

This new Release 7.2 feature enables you to determine the precise manner in which extents are actually used, *from a space management perspective*. The additional space management information is provided through a set of PL/SQL procedures.

A new PL/SQL package, called DBMS_SPACE, contains the following two procedures:

- UNUSED_SPACE
- FREE_BLOCKS

The DBMS_SPACE package is declared in the DBMSUTIL script that is installed when all other Release 7.2 utility packages are installed. Therefore, both upgrade and downgrade operations (in other words, forward and backward migrations, respectively) are accomplished by running the DBMSUTIL script.

Authorization required to use the DBM_SPACE package is the same as required to perform an ANALYZE command. The caller's authorization performs the security checks.

A complete set of definitions for all of the options in the UNUSED_SPACE and FREE_BLOCKS procedures is found in the *Oracle7 Server Administrator's Guide.*

Network Authentication

Sites migrating OS roles to their network service must remember to use the global database name instead of the SID when assigning names. However, no change to the database–defined roles is necessary.

Sites that wish to move OS authenticated users to their network service must use the same external name. For example, if a database user, SCOTT, is defined as IDENTIFIED EXTERNALLY and the operating system username is SCOTT, the network username cannot be defined as JANE; it must remain SCOTT to match the database username that already exists. If sites are creating new database accounts, the network identity need not match the operating system identity. However, the database username must correspond to the network identity for network authentication to take place.

For more information, see *Oracle7 Server Utilities* and the *Oracle7 Server Administrator's Guide*.

Activate SQL_TRACE for Remote Sessions

A portable mechanism now enables a database administrator to activate SQL_TRACE for a session other than her / his own.

For more information, see *Oracle7 Server Tuning*.

Oracle7, Release 7.2 Datatype Changes

The following table gives the rules for input and output host variables of different Oracle 7.2 datatypes that are national language variables:

Oracle 7.2 External Datatypes	On Input	On Output
ANSI Fixed CHAR (External datatype 96: Pro*COBOL–mode = ANSI	Input host variables are stripped of trailing double–byte spaces.† Pro*COBOL variables that are declared as PIC N take this datatype with mode=ANSI.	Output host variables are blank padded with double–byte spaces Pro*COBOL variables that are declared as PIC N take this datatype with mode=ANSI.
CHARZ (External datatype 97: Pro*C–DBMS = V7.2	Input host variables are stripped of trailing double–byte spaces and must contain a null terminator at the end of the data.† Pro*C CHAR variables take this datatype if the variable is typed to be a NLS variable (using the NLS_CHAR option).	Output host variables are double–byte blank padded and null terminated at the end of the buffer. Pro*C/C++ CHAR variables take this datatype if the variable is typed to be a NLS variable (using the NLS_CHAR option).
VARCHAR (External datatype 9)	Input host variables are not stripped of double–byte spaces. The length component is assumed to be the length of the data in characters, not bytes.	Output host variables are not padded at all. The length of the buffer is set to the length in characters, not bytes.

Table C – 1 Oracle 7.2 National Language Datatypes

† If the input national language host variable contains only spaces, one double byte will be left in the buffer to act as a sentinel.

SQL Changes

This section contains the following topics:

- ALTER DATABASE CLEAR
- ALTER DATABASE DATAFILE datafile END BACKUP
- ALTER ROLLBACK SEGMENT SHRINK
- ALTER SESSION SET INSTANCE
- ALTER TABLE
- CREATE CLUSTER ... HASH IS
- CREATE CLUSTER ... PARALLEL
- CREATE TABLE ... PARALLEL
- CREATE TABLE UNRECOVERABLE
- CREATE INDEX ... PARALLEL
- expr
- INSERT INTO subquery
- TO_CHAR
- UPDATE subquery
- Subquery in FROM Clause

ALTER DATABASE CLEAR

A new SQL command, ALTER DATABASE CLEAR, can reinitialize and optionally not archive an online log. The new command is the equivalent of a drop and add of the log file, except that it is allowed even if there are only two logs for the thread. The command may also be done to the current log of a closed thread. The syntax of the new command is

```
ALTER DATABASE CLEAR [UNARCHIVED] LOGFILE <log_name>
[<log_name>,...] [UNRECOVERABLE DATAFILE]
```

ALTER DATABASE DATAFILE datafile END BACKUP

It is now unnecessary to do complete media recovery when a database was shutdown without finishing an online backup with the ALTER DATABASE...END BACKUP command.

The following command takes the named datafile(s) out of backup mode:

```
ALTER DATABASE DATAFILE ‘<name>’ [‘<name>’...] END BACKUP
```

ALTER ROLLBACK SEGMENT SHRINK

It is now possible to shrink a rollback segment to an optimum size using the following command:

```
ALTER ROLLBACK SEGMENT name SHRINK TO size ;
```

ALTER SESSION SET INSTANCE

In a parallel server environment while connected to one instance it is now possible to mimic that the session is connected to another instance. For example:

```
ALTER SESSION SET INSTANCE = 3;
```

ALTER TABLE

It is now possible to allow or disallow users to use a table lock using the following commands:

```
ALTER TABLE table_name DISABLE TABLE LOCK;
ALTER TABLE table_name ENABLE TABLE LOCK;
```

This feature is of most use in a parallel server environment where a table lock can affect system performance.

CREATE CLUSTER ... HASH IS

The HASH IS option in the CREATE CLUSTER command is changed. The old HASH option specified a column as the hash function for a hash cluster. The new HASH option specifies a SQL expression as the hash function for a hash cluster. It is now possible to use your own PL/SQL functions to calculate the hash key. For example:

```
CREATE CLUSTER cloudy (deptno number(2))
   HASHKEY 20 HASH IS my_hash(deptno);
```

CREATE CLUSTER ... PARALLEL

It is now possible to use parallelism when creating a cluster and when querying the table after creation. For example:

```
CREATE CLUSTER cluz (empno number(6))
   PARALLEL (DEGREE 3);
```

CREATE TABLE ... PARALLEL

It is now possible to use parallelism when creating a table and when querying the table after creation. For example:

```
CREATE TABLE tmp_emp
   PARALLEL (DEGREE 3)
   AS SELECT * FROM emp WHERE deptno = 10;
```

CREATE TABLE UNRECOVERABLE

Release 7.2 provides support for unrecoverable logging during CREATE INDEX and CREATE TABLE AS SELECT operations. If you specify UNRECOVERABLE, log records are not generated for the operation. If you specify RECOVERABLE, log records are generated so that the newly created object can be recovered from database backups taken before the object was created. The syntax for the CREATE TABLE AS SELECT and CREATE INDEX commands is

```
CREATE TABLE '<name>' ['<name>'...] AS SELECT UNRECOVERABLE

CREATE INDEX '<index_name>'
    ON '<table_name>'(column_name)
    UNRECOVERABLE
```

CREATE INDEX ... PARALLEL

It is now possible to use parallelism when creating an index. For example:

```
CREATE INDEX idx ON emp(ename)
    PARALLEL (DEGREE 3);
```

expr

It is now possible to use a user defined PL/SQL function in the same manner as a SQL expression. For example:

```
SELECT my_fun(ename) FROM emp;
```

INSERT INTO subquery

It is now possible to use a subquery in the INTO clause of an insert statement similar to how views are used. For example:

```
INSERT INTO (SELECT * FROM dept)
    VALUES (50, 'DEVELOPMENT', 'BELMONT');
```

TO_CHAR

A number format model using '9's now returns a zero for the value zero. For example:

```
SELECT TO_CHAR(0,'999') num FROM DUAL;

NUM
----
   0
```

UPDATE subquery

It is now possible to use a subquery in an updated statement similar to how views are used. For example:

```
UPDATE (SELECT * FROM dept)
    SET deptno = 50
    WHERE deptno = 60
```

Subquery in FROM Clause

Release 7.2 offers the capability to use a subquery in the FROM clause of a SQL query. Such subqueries provide an "immediate view", specified directly in the query. The need to write relatively more complex queries that would otherwise require a view is thereby eliminated. The syntax is

```
SELECT <column_list>
FROM <table or subquery> [alias] [<table or subquery> [alias]]
```

For more information on new Release 7.2 SQL commands, see *Oracle7 Server SQL Reference* and the *Oracle7 Server Administrator's Guide.*

New Features of the Oracle Precompilers, Release 1.7

This section contains the following topics:

- Cursor Variable Support
- VARCHAR Recognition, C structs, and COBOL Group Items

Release 1.7 of the Oracle Precompilers is introduced with Oracle7, Release 7.2 and incorporates many new features designed to enhance precompiler functionality and application performance, as well as allow your applications to access new Oracle7, Release 7.2 capabilities. Most users will upgrade to Release 1.7 at the same time they upgrade to Oracle7, Release 7.2. Use of Releases 1.3, 1.4 1.5, and 1.6 of the Oracle Precompilers is supported against Oracle7, Release 7.2. However, if you wish to use the enhanced functionality of Oracle7, Release 7.2, you must use Release 1.7 of Pro*COBOL, Pro*FORTRAN, and Pro*ADA.

For more information, see the *Programmer's Guide to the Oracle Pro*C/C++ Precompiler*, the *Pro*COBOL Supplement to the Oracle Precompilers Guide*, and the *SQL*Module User's Guide and Reference.*

The following new features are introduced with the Oracle Precompilers, Release 1.7.

Cursor Variable Support

The new *cursor variable* feature in PL/SQL, Release 2.2 is fully supported in Pro*C, Release 2.1. PL/SQL,

VARCHAR Recognition, C structs, and COBOL Group Items

Pro*C / C++, Release 2.1 and Pro*COBOL, Release 1.7, with the VARCHAR command–line option enabled, recognize two unique data structures as VARCHAR host variables. These data structures consist of a length field and a string field, as shown in the following examples.

The following C struct is recognized by the Pro*C precompiler as a VARCHAR host variable:

```
struct {
        short     len;
        char      arr[length];
}  data_name
```

where data_name is a VARCHAR host variable and length is expressed in single–byte characters. The variable–length string is then referenced as data_name and the length field (len) is referenced as data_name.len.

In addition to single–byte character data, the Pro*COBOL precompiler also supports group items with string fields declared as multi–byte NLS datatypes (PIV N). The following COBOL group item is recognized by the Pro*COBOL precompiler as a VARCHAR host variable:

```
WORKING-STORAGE SECTION.
...
01     data_name.
       49 data_name-LEN PIC S9(4) {COMP|COMP-5}.
       49 data_name-ARR PIC {X(length)|N(length)}.
```

where data_name is a VARCHAR host variable and length is expressed in single–byte (X) or double–byte (N) characters. The variable–length string is then referenced as data_name and the field length (len) is referenced as data_name–LEN. “COMP” and “COMP–5” are used to declare integer datatypes.

The Oracle Precompiler Pro*C / C++, Release 2.1 offers enhanced functionality, performance, level of standards compliance for embedded SQL application development.

Pro*C/C++, Release 2.1 New Features

This section contains the following topics:

- C++ Support and Embedded SQL Programming
- Pro*C / C++, Release 2.1 and Multi Byte NLS Character Data

C++ Support and Embedded SQL Programming

Pro*C / C++, Release 2.1, which is available with the Oracle7 Server, Release 7.2, permits SQL and PL / SQL blocks to be embedded in C++ programs. A new PARSE command line option is available in Pro*C / C++, Release 2.1. Parse functionality permits you to decide how much of a program, outside the EXEC SQL DECLARE SECTION, is parsed. You may choose between parsing the entire program, parsing preprocessor directives and C declarations within an EXEC SQL DECLARE SECTION, or not parsing at all.

The extension to Pro*C / C++ has nothing to do with C++ intrinsically, but rather with grammatical extensions to C. An example of such an extension is that the Microsoft C compiler allows declarations to be amended with the (new) keywords *near* and *far*. Pro*C / C++, Release 2.1 has mechanisms that allow porters to handle special cases, but for cases not handled, you could invoke the Pro*C / C++ option as a workaround.

For more information, see the *Programmer's Guide to the Oracle Pro*C/C++ Precompiler.*

PL / SQL, Release 2.2 provide a new and convenient declarative mechanism to return multi–row result sets from stored procedures. Cursors can be used as regular (bind) variables, passed as parameters in functions and procedures, and returned from functions. This flexible and powerful solution allows you to fetch from a cursor at the client side, while the SQL statement belonging to the cursor is defined in a stored procedure.

For more information, see the *PL/SQL User's Guide and Reference.*

Pro*C/C++, Release 2.1 and Multi Byte NLS Character Data

In Pro*C, Release 2.1, your applications can use national language characters according to the Nippon Telegraph and Telephone Corporation (NTT) Multivendor Integration Architecture (MIA) specifications. (See "Standards Compliance" on page C – 25.)

The NTT MIA enhancement requires runtime support in SQLLIB. All applications that previously used Pro*COBOL 1.7 or Pro*C / C++ 2.1 need to link against the current version of SQLLIB.

For more information, see the *Programmer's Guide to the Oracle Pro*C/C++ Precompiler*, the *Pro*COBOL Supplement to the Oracle Precompilers Guide*, and the *SQL*Module User's Guide and Reference.*

SQL*Module Default WITH INTERFACE Clause

SQL*Module can now generate a default WITH INTERFACE clause for client–side remote procedure calls (RPC) to PL / SQL stored procedures.

The uses of the default WITH INTERFACE clause are as follows:

- Map PL / SQL types to embedded SQL types that are subsequently mapped to host language types.
- Specify which parameters are associated with indicator parameters.
- Specify how error conditions are to be reported to the client; for example, with SQLCODE, SQLSTATE, or both.

For more information, see the *SQL*Module User's Guide and Reference.*

Non–blocking Oracle Call Interface (OCI)

Release 7.2 enhances responsiveness of client / server applications by overcoming the limitations of synchronous, Oracle Call Interface (OCI) calls. Non–blocking OCI calls are now possible.

A new mechanism allows an OCI call to yield control back to the application if the call is blocked. Applications may now respond to events such as mouse clicks or timers while a call is being processed by Oracle, resulting in improved responsiveness.

Non–blocking calls are especially useful in client / server environments in which events from multiple sources, not generated in predictable order, must be handled.

Client side applications that existed before migration to Release 7.2 need to be modified if the non–blocking feature is to be used with the new Release 7.2 database.

For more information on non–blocking OCI, see the *Programmer's Guide to the Oracle Call Interface*.

Initialization Parameter Changes

The following table lists the initialization parameters that are obsolete and new in Release 7.2. See *Oracle7 Server Reference* for a complete description of all initialization parameters.

Obsolete Parameters	
LOG_ALLOCATION	INIT_SQL_FILES
New Parameters	
CPU_COUNT	SD_ALL
DB_BLOCK_CHECKSUM	SD_INHIBIT
LOG_BLOCK_CHECKSUM	SORT_DIRECT_WRITES

If you use Version 6 National Language Support, you must alter your Oracle7 parameter file to include the appropriate initialization parameters. Additionally, the specification of the NLS_SORT parameter has changed. See *Oracle7 Server Reference* for a complete description of National Language Support.

Data Dictionary Changes

This section contains the following topics:

- Views
- Dynamic Performance Tables
- Trusted Oracle7 Views

This section describes the changes to the Oracle7, Release 7.2 data dictionary. See *Oracle7 Server Reference* for descriptions of all standard tables and views that are available to developers.

Views

Oracle7, Release 7.2 introduces changes and additions to data dictionary views. Some pre–Release 7.2 views have been changed. New views have been introduced. The following tables summarize the changes and additions.

Changed Views

ALL_INDEXES	LOADER_CONSTRAINT_INFO
ALL_TABLES	USER_CLUSTERS
DBA_CLUSTERS	USER_INDEXES
DBA_INDEXES	USER_TABLES
DBA_TABLES	

New Views

ALL_CLUSTERS
ALL_CLUSTERS_HASH_EXPRESSIONS
DBA_CLUSTER_HASH_EXPRESSIONS
FILE_LOCK
FILE_PING
USER_CLUSTER_HASH_EXPRESSIONS

Dynamic Performance Tables

This section lists the changes and additions to the dynamic performance tables.

Obsolete Views

V$BACKUP	V$XATRANS$
V$PENDING_XATRANS$	

Changed Views

V$LATCHHOLDER	V$SESSION
V$OPEN_CURSOR	V$SQLAREA
V$FIXED_TABLE	

New Views

V$BACKUP_STATE	V$INSTANCE
V$BH	V$LOCK_ACTIVITY
V$CACHE	V$LOCK_ELEMENT
V$CACHE_LOCK	V$LOCK_WITH_COLLISIONS
V$PING	V$OBJECT_DEPENDENCY
V$DB_PIPES	V$PWFILE_USERS
V$EVENT_NAME	V$SQL
V$FALSE_PING	V$SQLTEXT
V$FIXED_VIEW_DEFINITION	V$SQLTEXT_WITH_NEWLINES
V$INDEXED_FIXED_COLUMN	FILEXT$

Trusted Oracle7 Views The following views and dynamic tables are now in Trusted Oracle7:

New Views

PROCEDURE_MAC_PRIVS

Standards Compliance

Multi–byte NLS Pro*C/C++, Release 2.1 and Pro*COBOL, Release 1.7 are now compatible with some aspects of the Multi–vendor Integration Architecture (MIA) specification developed by the Nippon Telegraph and Telephone Corporation (NTT), Japan. MIA is an attempt by NTT to achieve greater SQL compatibility among its vendors in the Japanese market. Developers can create double–byte National Language Character (NLC) character applications using NLC literals and variables.

For more information, see the *Programmer's Guide to the Oracle Pro*C/C++ Precompiler*, the *Pro*COBOL Supplement to the Oracle Precompilers Guide*, and the *SQL*Module User's Guide and Reference.*

Release 7.2 Backward Compatibility

Release 7.2 is completely backward compatible with Version 6, and Releases 7.0 and 7.1 of the Oracle7 Server. As a maintenance release, Release 7.2 is the normal maintenance path for users running 7.0 and 7.1 applications.

Warning: If the SQL*DBA COMPATIBILITY parameter is set to Version 6, the functionality of certain Release 7.2 features will not be available. For example, if COMPATIBLE is set to Version 6, the only constraint that will be enforced automatically is NOT NULL. The PRIMARY KEY constraint will not be enforced automatically unless it is indexed. All other constraints will not be enforced.

For more information about backward compatibility and downgrading, see Chapter 9, "Upgrading and Downgrading between Oracle7 Releases".

New, Changed, and Obsolete SQL Scripts

New scripts are added and existing scripts are modified or made obsolete to support the extended functionality of Oracle7 Release 7.2. The new and changed scripts are listed below.

By convention, scripts that begin with CAT are intended to be run by SYS. They create views and tables in the data dictionary. Scripts that begin with DBMS are also run by SYS. They create stored procedures, functions, and packages. Scripts that begin with UTL are intended to be run by individual users. For more information, see *Oracle7 Server Reference.*

Obsolete Scripts

UTLOPCRX.SQL	UTLOPEXP.SQL
UTLOPDRX.SQL	UTLOPIMP.SQL
UTLOPENV.SQL	UTLOPTAB.SQL

Changed Scripts

DBMSUTIL.SQL†	TKPQCON*.SQL
TKPQSUIT.TSC	TKPQKPMM.TSC
TKPQSPRF.TSC	TKPQABD.SQL
TKTTAB.TSC	

† The DBMS_SPACE package was added to this script.

New Scripts

CAT70103.SQL	SAMPLE1.SQL
CAT7201.SQL	SAMPLE2.SQL
CAT7202.SQL	SAMPLE3.SQL
DBMSPROBE.SQL	SAMPLE4.SQL
PBREAK.SQL	TKTCTAS.SQL
PBRPH.SQL	TKPQCTAS.TAS
PBUTL.SQL	

Other Changes

XA Library

The Oracle XA library is included with Release 7.2. The Oracle XA library is an external interface that allows global transactions to be coordinated by a transaction manager other than Oracle7. This allows inclusion of non–Oracle entities called resource managers (RM) in distributed, two–phase commit transactions. The Oracle XA library conforms to the X/Open Distributed Transaction Processing (DTP) software architecture's XA interface specification. For more information, see *Oracle7 Server Distributed Systems, Volume I.*

APPENDIX

D

Summary of Changes in Oracle7, Release 7.3

This appendix provides an overview of the changes in the Oracle7 Server, Release 7.3. The topics included in this appendix are:

- Administration Enhancements
- Query Execution Enhancements
- Scalability and Performance Enhancements
- Parallel Server Enhancements
- Serviceability Enhancements
- Tuning Enhancements
- Advanced Replication Enhancements
- Interface Enhancements
- PL/SQL Enhancements
- Data Dictionary Changes
- Initialization Parameter Changes

Administration Enhancements

This section contains the following topics:

- Standby Database
- Resilvering Enhancement
- Media Recovery Usability
- Dynamic Initialization Parameters
- Fast Recreate Index
- Direct Path Export
- Space Management Enhancements
- Sort Direct Writes

Standby Database

Standby Database supports the capability of maintaining a duplicate, or standby, database of your primary, online production database at a remote site. *Standby Database* thus enables recovery from production site disasters.

The operational features of the standby database are as follows:

- The standby database is copied from the primary, or current, production database and is duplicated on physical hardware such as disk or CPU.
- The standby database should be used only as a disaster recovery system.
- The standby database is mounted, but not open, and is in constant recovery mode.
- As the primary database archives its redo logs, they are transferred to the remote site and applied to the standby database.
- In the event of catastrophic failure of the primary database, the standby database can be taken out of recovery mode and opened for online use.

Standby Database requires the use of the following new and / or changed SQL statements:

```
ALTER DATABASE CREATE STANDBY CONTROLFILE AS <filename>
ALTER DATABASE MOUNT STANDBY DATABASE[PARALLEL]
ALTER DATABASE RECOVER STANDBY DATABASE
ALTER DATABASE ACTIVATE STANDBY DATABASE
```

Standby Database requires the use of the following new SQLDBA statement:

```
RECOVER STANDBY DATABASE
```

> **Note:** *Standby Database* will operate only on Oracle7, Release 7.3 or higher of the database.

For more information about *Standby Database*, see *Oracle7 Server SQL Reference*, the *Oracle7 Server Administrator's Guide*, and *Oracle7 Parallel Server Concepts & Administration*.

Resilvering Enhancement

Many operating systems provide mirrored disk support. A typical mirrored disk support system uses several disks that are maintained as identical copies of each other. The failure of any one disk is not catastrophic because the remaining disks continue to operate without any loss of data. The disks are kept in synchronization by performing duplicate writes.

However, it is possible that some of the writes may not be completed due, for example, to a system failure of some sort. All of the disks in the mirrored system will continue to operate properly but would not, in the case of incompleted writes, be exact mirror images of each other. The usual recourse to resolve this problem is to recopy the entire mirror from one disk, which is very expensive and time consuming. In parallel server configurations, where each node in a cluster is responsible for issuing multiple writes to the mirror, the death of any node in the cluster requires the resilvering of all mirrored disks in the cluster.

The *Resilvering Enhancement* eliminates the need for the complete recovery of a mirrored file when there are failures that could have left the mirror out of sync due to writes by Oracle. The *Resilvering Enhancement* also allows Oracle to use other files in a mirror to repair corrupted data that does not produce hardware–detectable errors.

OSDoc

Additional Information: Control of mirroring is port specific. For more information about mirroring and your specific platform, see your operating system–specific Oracle documentation.

Media Recovery Usability

The *Media Recovery Usability* enhancement provides you with fixed tables and views that contain the information regarding a media recovery. There are two new views:

- V$RECOVERY_FILE_STATUS
- V$RECOVERY_STATUS

V$RECOVERY_FILE_STATUS, contains one row for each file that is a potential candidate of the recover command that was issued. For example, if the recover command was RECOVER DATAFILE, then V$RECOVER_FILE_STATUS will contain one row for each datafile named in the command; if the recover command was RECOVER DATABASE, then V$RECOVER_FILE_STATUS will contain one row for each datafile that is online.

V$RECOVERY_STATUS contains

- information about the log that recovery needs next from the database administrator.
- information about why recovery rejected the last log name the user supplied and the status of the most recently processed log.

For more information about the *Media Recovery Usability* enhancement, see the *Oracle7 Server Administrator's Guide.*

Dynamic Initialization Parameters

Dynamic Initialization Parameters allows the modification of initialization parameters while an instance is running. Traditionally, the only way initialization parameters could be modified has been to change their values within the INIT.ORA parameter file, shut down the instance, and restart the instance using the modified parameter file.

Initialization parameters may now be specified *dynamically* using the ALTER SESSION SET and ALTER SYSTEM SET commands.

The ALTER SESSION command can be used to change the value of an initialization parameter for the duration of a session or until the next execution of the ALTER SESSION command. The required syntax is

```
ALTER SESSION SET <parameter name> = <value>
```

The ALTER SYSTEM command can be used to change the *global* value of an initialization parameter. New sessions will see the changed value of the initialization parameter. The required syntax is:

```
ALTER SYSTEM SET <parameter name> = <value>
```

The following are the dynamic initialization parameters available with Oracle7, Release 7.3:

- HASH_AREA_SIZE
- HASH_JOIN_ENABLED
- HASH_MULTIBLOCK_IO_COUNT

For more information about *Dynamic Initialization Parameters*, see *Oracle7 Server Reference* and *Oracle7 Server Tuning*.

Fast Recreate Index

Fast Recreate Index allows users to create an index using an existing index as the data source. Essentially, this allows the user to change an index's storage characteristics, if desired.

The semantics of the CREATE index command remain unchanged with the fast recreate index command. If a user wishes to create a new index where the columns to be indexed are a subset of the columns of an existing index on the same table, the CREATE INDEX command can use the existing index to retrieve the rows of the index for fast operation.

Fast Recreate Index introduces the REBUILD option to the ALTER INDEX DML statement. The REBUILD option allows the user to recreate an existing index. In the process of recreation, the storage characteristics and tablespace where the index resides can be changed. Recreating an existing index also removes intrablock space fragmentation. The syntax of the extended ALTER INDEX command is

```
ALTER INDEX <indexname>   REBUILD
                          [PARALLEL <integer> | NOPARALLEL]
                          [RECOVERABLE | UNRECOVERABLE]
                          [TABLESPACE <tablespace name>]
                          [<extent> specs]
```

All of the clauses after REBUILD are optional.

For more information about *Fast Recreate Index*, see the *Oracle7 Server Administrator's Guide* and *Oracle7 Server Tuning*.

Direct Path Export

Direct Path Export provides a *fast path* for the extraction of data from tables that significantly improves the overall performance of the Export Utility.

The Export Utility now provides two data paths for exporting table data.

- the conventional path
- the direct path

The conventional path uses the SQL statement "SELECT* FROM table" to extract data from database tables. This path is used by all Oracle tools and applications. Data is transferred to the buffer cache; the EXPORT client then writes the data to the EXPORT dump file.

The direct path bypasses the buffer cache and transfers data to the user's private buffer cache. Contention with other users is thereby avoided.

Direct Path Export can be invoked

- from the command line using a PARAMETER file
- from the command line using the EXP command with the DIRECT parameter specified

For more information about *Direct Path Export*, see *Oracle7 Server Utilities*.

Space Management Enhancements

This section describes space management enhancements that are available with Release 7.3.

Unlimited Extents

In Release 7.2 and earlier releases of Oracle7, the number of extents that could be allocated to a single segment was limited by the database block size. The entire extent map had to fit within half of the segment header block. For a 2 Kb block, the maximum number of extents per segment was 121.

The constraint on the number of extents made a number of common database management operations more difficult than they would otherwise have been if no constraint existed. *Unlimited Extents* removes the constraint of an upper limit to the number of possible extents.

The following are changes in space management:

- MAXEXTENTS is no longer limited by the number of extents that fit into a single database block.
- A new keyword, UNLIMITED, is now supported as a valid value for MAXEXTENTS.

- *Unlimited Extents* generates incompatible disk data structures that must be corrected if migration operations are to be performed.
- Dictionary tables cannot be altered to have MAXEXTENTS greater than the allowed block maximum. In other words, dictionary tables cannot be converted to unlimited format.
- Rollback segments have to be offline to be converted to unlimited format from limited format and vice versa.
- Extents for rollback segments in unlimited formats are required to have, at least, 4 blocks.
- The default storage clause for newly created tablespaces is the same as it was in Release 7.2. Existing tablespaces and newly created tablespaces need to have their default storage clauses changed manually.
- Since *Unlimited Extents* allows unlimited extents, old Version 6 SQL scripts may, when run under Release 7.3, have MAXEXTENTS greater than the past allowed maximum. Offending values for existing tables are, therefore, all reset to the allowed Release 7.2 maximum the first time the system is started in Release 7.3 compatible mode.

For more information about *Unlimited Extents*, see the *Oracle7 Server Administrator's Guide*.

Tablespace Coalesce

Tablespace Coalesce improves space management by providing a new command, ALTER TABLESPACE <tablespace> COALESCE. The new command coalesces all available free space (extents) in the tablespace into larger, contiguous extents on a per file basis. A new catalog view, DBA_FREE_SPACE_COALESCED displays statistics pertaining to coalesceable extents for tablespaces.

The syntax for the ALTER TABLESPACE COALESCE command is

```
ALTER TABLESPACE <tablespace> COALESCE;
```

DBA_FREE_SPACE_COALESCED has the following columns:

Column	Comments
TABLESPACE_NAME	the name of the tablespace
TOTAL_EXTENTS	the number of free extents
EXTENTS_COALESCED	the number of free extents that are coalesced
PERCENT_EXTENTS_COALESCED	the percentage of coalesced free extents
TOTAL_BYTES	the total number of free bytes
BYTES_COALESCED	the number of coalesced free bytes
TOTAL_BLOCKS	the number of free Oracle blocks
BLOCKS_COALESCED	the number of coalesced free Oracle blocks
PERCENT_BLOCKS_COALESCED	the percentage of coalesced free Oracle blocks

Table D – 1 Columns in the DBA_FREE_SPACE_COALESCED View

> **Note:** The lower the percentage of coalesced entries or blocks, the more fragmented the space and the greater the need to issue the ALTER TABLESPACE COALESCE command.

For more information about *Tablespace Coalesce*, see *Oracle7 Server SQL Reference* and the *Oracle7 Server Administrator's Guide.*

Deallocation of Unused Space

Deallocation of Unused Space provides the ability to release unused space from a segment and return it to the database system. The DBMS_SPACE package, available in Release 7.2, allowed users to compute that amount of unused space in a specific segment. *Deallocation of Unused Space* provides additional functionality by enabling users to actually release the unused space (or some portion of it).

An ALTER command, to release the unused space, is now provided for each user segment type, TABLE, INDEX, and CLUSTER.

The ALTER syntax for deallocating space from the table segment is

```
ALTER TABLE <tablename> DEALLOCATE UNUSED [KEEP (integer)]
```

The ALTER syntax for deallocating space from the index segment is

```
ALTER INDEX <indexname> DEALLOCATE UNUSED [KEEP (integer)]
```

The ALTER syntax for deallocating space from the cluster segment is

```
ALTER INDEX <clustername> DEALLOCATE UNUSED [KEEP (integer)]
```

For more information about *Deallocation of Unused Space*, see *Oracle7 Server SQL Reference* and the *Oracle7 Server Administrator's Guide.*

Sort Segment

Sort Segment improves concurrency of multiple sort operations. The enhanced performance is not entirely automatic and must be enabled by the user. The user must do the following:

- define temporary tablespaces
- monitor Sort Segment performance

Defining Temporary Tablespaces

A tablespace can be defined as temporary during creation, or it can be made temporary later. The CREATE TABLESPACE command is expanded to include the following options:

```
CREATE TABLESPACE <tablespace> TEMPORARY
CREATE TABLESPACE <tablespace> PERMANENT
```

Specifying TEMPORARY defines the tablespace as a *temporary tablespace*. All sorts in a temporary tablespace share a single Sort Segment and allocate space using the Sort Segment table. However, no permanent objects can be stored in the temporary tablespace.

Specifying PERMANENT allows the permanent objects to be stored in the tablespace. But if this tablespace is used for sorting, no caching is done, so sort performance may suffer.

The default setting is PERMANENT.

A setting for an existing tablespace can be altered as follows:

```
ALTER TABLESPACE <tablespace> TEMPORARY
ALTER TABLESPACE <tablespace> PERMANENT
```

Specifying TEMPORARY makes this tablespace a temporary tablespace. The tablespace may not contain any permanent objects; otherwise, an error is generated.

Specifying PERMANENT allows future creation of permanent objects.

A temporary status is reflected in the new CONTENTS column of the DBA_TABLESPACES view. The column shows a value of “PERMANENT” for permanent tablespaces and a value of “TEMPORARY” for temporary tablespaces.

Monitoring Sort Segment Performance

Sort Segment performance can be monitored using some of the new dynamic performance tables and some new values in the existing dynamic tablespaces.

Table	Comments
V$LATCH	The Sort Extent Pool latch is reflected in this table.
V$LATCHNAME	The Sort Extent Pool latch is reflected in this table. The latch name is “sort extent pool”.
V$SORT_SEGMENTS	This is a new table. It contains information about every Sort Segment created in the given instance.

Table D – 2 Changed and New Tables for Sort Segment

For more information about *Sort Segment*, see *Oracle7 Parallel Server Concepts & Administration, Oracle7 Server Tuning,* and the *Oracle7 Server Administrator’s Guide.*

Sort Direct Writes

Sort Direct Writes provides an automatic tuning method for deriving the size and number of direct write buffers based upon the sort area size. The memory for the buffers is taken from the sort area, so only one tuning parameter is necessary. In addition, an optimizer cost model is provided.

Performance Benefits of Sort Direct Writes

You can set the initialization parameter SORT_DIRECT_WRITES to increase sort performance if memory and temporary space are abundant on your system and you perform many large sorts to disk.

For Release 7.3 and greater, the default value of SORT_DIRECT_WRITES is AUTO. If the initialization parameter is unspecified or set to AUTO, the database automatically allocates direct write buffers if the SORT_AREA_SIZE is ten times the minimum direct write buffer configuration.

Performance Tradeoffs of Sort Direct Writes

Sort Direct Writes causes each Oracle process that sorts to allocate

```
(SORT_WRITE_BUFFERS)*(SORT_WRITE_BUFFER_SIZE)
```

bytes of memory in addition to the memory already allocated for the sort area. You must ensure that your operating system has enough free memory available to accommodate the increased allocation.

Sorts that use direct writes tend to consume more temporary segment space on disk. A good rule of thumb is that the total memory allocated for direct write buffers should be less than one–tenth of the memory allocated for the sort area. If the minimum configuration of the direct write buffers is greater than one–tenth of your sort area, you should not trade sort area for direct write buffers.

Warning: Using the default SORT_DIRECT_WRITES mode of AUTO causes the database to use the one–tenth rule to decide whether to use direct writes and it allocates the direct write buffers out of a portion of the total sort area, ignoring the settings for SORT_DIRECT_WRITE_BUFFERS and SORT_WRITE_BUFFER_SIZE.

Initialization Parameter Files for Sort Direct Writes

The following is a list of the initialization parameters that are used for *Sort Direct Writes*:

SORT_DIRECT_WRITES

Default value for Release 7.2:	FALSE
Default value for Release 7.3:	AUTO

SORT_WRITE_BUFFER_SIZE

Default value:	O/Dependent
Range of values:	32 kilobytes to 64 kilobytes

SORT_WRITE_BUFFERS

Default value:	O/S dependent
Range of values:	2 to 8

Compatibility and Migration of Sort Direct Writes

If you upgrade to Release 7.3, SORT_DIRECT_WRITES is initially set in AUTO mode by default. Because the direct writes use large buffers (typically 32 kilobytes to 64 kilobytes), the space map function in the sort splits extents into buffer–sized chunks in order to exploit large multiblock writes. The non–direct write case uses only 4 kilobytes. This change in space allocation may result in a 10% to 15% increase in temporary space usage.

For more information about *Sort Direct Writes*, see *Oracle7 Server Reference*, the *Oracle7 Server Administrator's Guide*, and *Oracle7 Server Tuning*.

Query Execution Enhancements

This section contains the following topics:

- Hash Join
- Histograms
- Updatable Join Views
- Sort Big Keys

Hash Join

Previous releases and versions of Oracle have employed two join algorithms: Nested loops and Sort–Merge. *Hash Join* improves the performance of join operations, especially in decision support applications. The performance improvement is applicable to both serial queries and parallel queries.

Three, new initialization parameters are available with Release 7.2.2 that must be used with *Hash Join*. The three initialization parameters are session parameters, that is, their values may be altered using the ALTER SESSION command.

Initialization Parameter	Use
HASH_JOIN_ENABLED	A boolean operator. If using hash joins produces less than ideal results, you can turn it off by setting it to FALSE. The default value is TRUE.
HASH_AREA_SIZE	Specifies the maximum amount of memory, in bytes, to be used for the hash join. If not specified, hash join uses twice the SORT_AREA_SIZE value.
HASH_MULTIBLOCK_IO_COUNT	Determines how many blocks hash join should read and write at once. If not specified, hash join uses the value for DB_FILE_MULTIBLOCK_ READ_COUNT.

Table D – 3 Initialization Parameters used with Hash Join

USE_HASH is a “hint” that increases the probability of the optimizer selecting hash join as the optimal method for joining each specified table with another row source. The syntax is

```
USE_HASH (table ...)
```

where *table* is a table to be joined to the row source resulting from joining the previous tables in the join order using a hash join.

For more information about *Hash Join*, see *Oracle7 Server Tuning* and *Oracle7 Server SQL Reference.*

Histograms

Histograms enables Oracle's cost based optimizer to generate better query evaluation plans for Oracle applications, end user applications, and ad hoc queries.

One of the fundamental capabilities of any cost–based optimizer (CBO) is the ability to determine the selectivity of predicates that appear in queries. Oracle's CBO, in releases earlier than 7.3, provided support for accurate selectivity estimates under the assumption that the attribute domains, in other words, a table's columns, were uniformly distributed. However, most attribute domains are not uniformly distributed. *Histograms* enables the CBO to describe the distributions of non–uniform domains by utilizing height balanced histograms on specified attributes.

Histograms are useful only when they reflect the current data distribution of a given column. If the data distribution is not static, the histogram should be updated frequently. The data need not be static as long as the distribution remains constant. Histograms are expensive and should be used only when they substantially improve query plans. Histograms are not useful for columns with the following characteristics:

- All predicates on the column use bind variables.
- The column data is uniformly distributed.
- The column is not used in WHERE clauses of queries.
- The column is unique and is used only with equality predicates.

Histograms has required modification of

- the ANALYZE TABLE syntax to allow specification and management of height balanced histograms,
- the data dictionaries to store the histograms efficiently,
- the cost–based optimizer to utilize the histogram effectively, and
- the stored procedures ANALYZE_OBJECT, ANALYZE_SCHEMA, and ANALYZE_DATABASE in package DBMS_UTILITY to reflect the new analyze options.

Histograms can be viewed using the following views:

- USER_HISTOGRAMS
- ALL_HISTOGRAMS
- DBA_HISTOGRAMS
- TAB_COLUMNS

The ANALYZE command and the cost–based optimizer will not work unless the proper upgrade and downgrade procedures are followed. The following upgrade and downgrade scripts must run:

Script	Use
CAT7301.SQL	upgrade from release 7.2 to release 7.3
CAT7301D.SQL	downgrade from release 7.3 to release 7.2

Table D – 4 Upgrade and Downgrade Scripts

For more information about *Histograms*, see *Oracle7 Server Tuning*.

Updatable Join Views

Updatable Join Views provides support for inserts, updates, and deletes on unambiguous join views.

Basic Concepts

The following three definitions are basic to the use of *Updatable Join Views*:

Join View	A join view is a view with more than one table (or view) in its FROM clause and with none of the following constructs used in it: DISTINCT, AGGREGATION, GROUP_BY, START_WITH, CONNECT_BY, and set operations such as UNION, UNION ALL, MINUS, and INTERSECT.
Join Column	A join column is any column of a table in the FROM clause that is used in a WHERE clause expression that has columns from some other table in the FROM clause.
Key Preserved Table	A table is said to have its keys preserved through a join if every key of the table is also a key of the result of the join. Such a table is called a *key preserved table* (with respect to the join).

Rules for Insert, Update, and Delete on Join Views

The following rules apply to insert, update, and delete operations on join views:

General	Any insert, update, or delete statement on a join view can modify only one underlying base table at a time.
Insert	An insert may not, explicitly or implicitly, refer to the columns of a *non–key preserved* table. If the join is defined with the WITH CHECK OPTION, then it may not be inserted into.
Update	All updatable columns of a join view must map to columns of a *key preserved* table. If the view is defined with the WITH CHECK OPTION clause, then all join columns and all columns of repeated tables are non–updatable.
Delete	Rows from a join view can be deleted provided there is exactly one table in the join whose keys are being preserved. If the view is defined with the WITH CHECK OPTION and the *key preserved* table is repeated, then the rows cannot be deleted from the view.

Table D – 5 Insert, Update, and Delete Rules

New Catalog Views

A new family of catalog views, UPDATABLE_COLUMNS is provided with Release 7.3. The new views are {USER|ALL|DBA}_UPDATABLE_COLUMNS, each of which has the following columns:

Column Name	Comments
OWNER	Owner of this table or view
TABLE_NAME	Name of this table or view
COLUMN_NAME	Name of the column in the table or view
UPDATABLE	Is the column updatable? YES or NO

Table D – 6 New Catalog Views to Support Updatable Join Views

For more information about *Updatable Join Views*, see the *Oracle7 Server Administrator's Guide*, *Oracle7 Server Concepts*, and the *Oracle7 Server Application Developer's Guide*.

Changes to Data Dictionary Tables

The following table shows the changes in data dictionary tables that have been made to support *Compiled Triggers*:

Table	Changes
ERROR$	now contains errors generated while compiling triggers
OBJ$	now keeps the appropriate status of a trigger object
IDL$	now contains pcode and debug code for trigger objects
DEPENDENCY$	now keeps dependencies for trigger objects

Table D – 7 Data Dictionary Table Changes for Compiled Triggers

For more information about *Compiled Triggers*, see *Oracle7 Server SQL Reference*, *Oracle7 Parallel Server Concepts & Administration*, and the *Oracle7 Server Application Developer's Guide*.

Sort Big Keys

Sort Big Keys removes query restrictions that existed in Release 7.2. The following restrictions have been removed in Release 7.3:

- There are no longer any restrictions on the size of SELECT list items or ORDER BY keys that previously existed in Release 7.2.
- Size restrictions are eliminated for all DISTINCT operations and set operators such as MINUS.
- For cases of aggregate operations, the total size of all DISTINCT and non–DISTINCT aggregate workspaces may not exceed 32 kilobytes (this is a cursor memory limit.)

- There is no limit on the combined size of the GROUP BY key and any individual DISTINCT aggregate. However, the total size of the GROUP BY expression and the sum of the sizes of the non–DISTINCT aggregates may not exceed a single sort block, which is a single database block minus some overhead.
- The MIN / MAX operations on VARCHAR expressions that exceed 255 bytes are converted to MIN / MAX DISTINCT.

For more information about *Sort Big Keys*, see the *Oracle7 Server Administrator's Guide*, *Oracle7 Server SQL Reference*, and *Oracle7 Server Tuning*.

Scalability and Performance Enhancements

This section contains the following topics:

- Remote Dependencies in a PL / SQL Environment
- Fast Transaction Rollback and XA Recovery Enhancements
- LRU Latch Scalability
- Serializable Transaction Isolation

Remote Dependencies in a PL/SQL Environment

Remote Dependencies in a PL/SQL Environment provides the following enhancements:

An extra level of flexibility in the model for managing remote dependencies: Prior to Release 7.3, the model for managing remote dependencies between stored procedures was based on timestamps. With the new model for managing remote dependencies in Release 7.3, you now have a way to control the management of remote dependencies. Table D – 8 summarizes these new modes of control.

Improved performance – avoiding unnecessary recompilations: With the SIGNATURE MODE (see Table D – 8), compatible changes to a referenced unit no longer cause the invalidation of those dependent units that are remote, but will continue to cause the invalidation of local dependent units. Unnecessary recompilations of dependent units across the network are thus prevented, which improves performance.

Improved performance – smaller library units: The size of library units has decreased. Package bodies are substantially smaller. Package specifications, especially those containing subprograms with a large number of parameters, are also smaller.

Ability to allow client–side tools, such as Oracle Forms and Oracle Procedure Builder, to upgrade to PL/SQL, Version 2: Client–side tools, such as Oracle Forms and Oracle Procedure Builder are built with PL/SQL Version 1 on the client–side. PL/SQL Version 1 does not inherently support a dependency management model to track dependencies from client–side PL/SQL library units to server–side PL/SQL library units.

While PL/SQL Releases 2.0 through 2.2 do have a strong dependency management model to track such dependencies, the model is too restrictive: client–side applications built with Oracle Forms or Oracle Procedure Builder cannot be installed at a user's site without requiring a recompilation of the client–side PL/SQL library units immediately upon installation. The new *Remote Dependencies* feature, with the SIGNATURE mode, has relaxed some of these restrictions, allowing client–side installations to proceed without requiring a recompilation of library units on the client–side.

REMOTE_ DEPENDENCIES_MODE Parameter

All library units in Oracle7, Release 7.2 (and earlier) databases have timestamps associated with them. Prior to Release 7.3, timestamps were used to control dependencies between procedures across the network. With Release 7.3, all library units continue to have timestamps associated with them. However, timestamp mismatches are now ignored if the *user requests that invalidation be based on signatures by using the REMOTE_DEPENDENCIES_MODE parameter,* which is new with Release 7.3.

The REMOTE_DEPENDENCIES_MODE parameter is relevant only during RPC calls and is applicable on the remote end of the RPC call. The REMOTE_DEPENDENCIES_MODE parameter can be set to have the following values:

Value	Comments
TIMESTAMP	invalidation of remote dependents will happen based on mismatch of timestamps
SIGNATURE	invalidation of remote dependents will happen based on mismatch of signatures

Table D – 8 Values for the REMOTE_DEPENDENCIES_MODE Parameter

The REMOTE_DEPENDENCIES_MODE parameter can be set in any of the following ways:

- as an initialization parameter in the INIT.ORA file. The syntax is

```
REMOTE_DEPENDENCIES_MODE=<value>
```

- at the system level using the ALTER SYSTEM command. The syntax is

```
ALTER SYSTEM SET REMOTE_DEPENDENCIES_MODE=<value>
```

- at the session level using the ALTER SESSION command. The syntax is

```
ALTER SESSION SET REMOTE_DEPENDENCIES_MODE=<value>
```

where *value* can be either TIMESTAMP or SIGNATURE.

For more information about *Remote Dependencies in a PL/SQL Environment*,, see the *Oracle7 Server Application Developer's Guide*, *Oracle7 Server SQL Reference*, and the *PL/SQL User's Guide and Reference*.

Fast Transaction Rollback and XA Recovery Enhancements

The *Fast Transaction Rollback and XA Recovery* enhancements provide the following new functionality:

- The time required for transaction recovery is reduced by recovering multiple transactions in parallel.
- The remaining parts of the database, in other words, data that is not locked by the transactions requiring recovery, are now available very quickly.
- Correct behavior for XA recovery operations is now provided.

Fast Transaction Rollback now allows the database to be opened for connections as soon as cache recovery is completed.

XA Recovery Enhancements

There are two enhancements to XA recovery in Release 7.3:

- An option is added to make the XA_RECOVER call wait for instance recovery.
- The XA Info string has a new clause called OPS_FAILOVER. If OPS_FAILOVER is set to T or t for a given XA resource manager connection, any XA_RECOVER call issued from that connection will wait for instance recovery to compete. The syntax of OPS_FAILOVER is

```
OPS_FAILOVER=T or OPS_FAILOVER=t.
```

The default value for OPS_FAILOVER is

```
OPS_FAILOVER=F or FALSE
```

When OPS_FAILOVER is set to TRUE, the XA_RECOVER call waits until SMON has finished cache recovery, has identified the in–doubt transactions, and added them to the PENDING_TRAN$ table that contains a list of in–doubt transactions.

For more information about *Fast Transaction Rollback* and *XA Recovery*, see the *Oracle7 Server Application Developer's Guide* and *Oracle7 Server Distributed Systems, Volume I*.

LRU Latch Scalability

LRU Latch Scalability provides LRU scalability with large SMP machines. The major benefits are

- low LRU contention under large SMP configurations
- the elimination of the need for non–preemptive operating system scheduling under heavy loads

Changes required by *LRU Latch Scalability* are the following:

- A new initialization parameter, DB_BLOCK_LRU_LATCHES, configures the buffer cache. DB_BLOCK_LRU_LATCHES specifies an advisory upper bound value for the desired number of sets.
- The number of sets used by the instance as a new field is now exported in the V$PARAMETER view. Note that the number of sets displayed is the number of sets used by the system and may not be the same as the value requested by the DB_BLOCK_LRU_LATCHES parameter.

For more information about *LRU Latch Scalability*, see *Oracle7 Server Tuning* and *Oracle7 Parallel Server Concepts & Administration*.

Serializable Transaction Isolation

Serializable Transaction Isolation allows application developers to employ a more flexible tool when designing application transactions that must have a consistent view of their data throughout the duration of those transactions.

The capability of designing application transactions that must have a consistent view of their data throughout the duration of those transactions is already possible for query–only application transactions using the existing SET TRANSACTION READ ONLY command. The new isolation level provided by *Serializable Transaction Isolation* preserves the transaction–consistent view of data that is provided by SET TRANSACTION READ ONLY. *Serializable Transaction Isolation* also allows transactions that use it to execute DML statements and allows such transactions to see their own changes while shielding them from visibility of other transactions' changes either in–flight or committed.

There are modifications to the SET TRANSACTION and ALTER SESSION commands.

The SQL command syntax for the SET TRANSACTION command is extended as follows:

```
SET TRANSACTION ISOLATION_LEVEL SERIALIZABLE
```

or

```
SET TRANSACTION ISOLATION_LEVEL READ COMMITTED
```

The SQL command syntax for the ALTER SESSION command is extended as follows:

```
ALTER SESSION SET ISOLATION_LEVEL=SERIALIZABLE
```

or

```
ALTER SESSION SET ISOLATION_LEVEL=READ COMMITTED
```

For more information about *Serializable Transaction Isolation*, the SET TRANSACTION command, and the ALTER SESSION command, see the *Oracle7 Server Administrator's Guide* and *Oracle7 Server SQL Reference*.

Parallel Server Enhancements

This section contains the following topics:

- Fine Grained Locking
- Instance Registration
- Delayed–Logging Block Clean Out
- Parallel Query Affinity
- Load Balancing in Listener

Fine Grained Locking

Fine Grained Locking provides a more efficient method for locking in a multinode configuration.

Fine Grained Locking provides the following capabilities:

- Parallel server locking configurations are now allowed that can reduce the number of database blocks protected by a single lock, allowing a one–to–one mapping from lock to data block, if desired.

- Configurations of locks that require more locks than can reasonably be held by the instance at any one time are now allowed.
- Instances are now allowed to release locks that are not necessary to protect data currently in the buffer cache.
- The database administrator is now allowed to specify locks that should always be held, regardless of the necessity of holding them to protect data in the cache.

Warning: *Fine Grained Locking* allows the database administrator to configure PCM locks by specifying a set of releasable DBA and hashed locks to protect the blocks in data files. Since releasable locks are expensive (they are acquired and released for each use) certain benchmarks may show a decreased level of performance when run in this mode. However, other types of access to the database will improve with releasable DBA locks. The database administrator should try to configure the locks to match this type of use for each data file.

The parameters that may be used with *Fine Grained Locking* are summarized in the following table.

Parameter	Use
GC_DB_LOCKS	specifies the total number of locks that protect data blocks
GC_FILES_TO_LOCKS	gives the mapping of hashed locks to blocks within each datafile
GC_DEFAULT_LOCKS	specifies the translation to use for files that are not mentioned in GC_FILES_TO_LOCKS
GC_ROLLBACK_LOCKS	specifies the number of hashed locks for each UNDO segment
GC_SAVE_ROLLBACK_LOCKS	similar to GC_ROLLBACK_LOCKS
GC_SEGMENTS	specifies the number of hashed locks to create for the segment header class of blocks
GC_TABLESPACES	specifies the number of hashed locks for save–undo segment headers
GC_ROLLBACK_SEGMENTS	specifies the number of hashed locks for undo segment headers (transaction tables)

Table D – 9 Parameters for the Fine Grained Locking

For more information about *Fine Grained Locking*, see *Oracle7 Parallel Server Concepts & Administration*.

OSDoc

Additional Information: *Fine Grained Locking* is not available on all platforms. For information regarding the use of *Fine Grained Locking* on your platform, see your operating system–specific Oracle documentation.

Instance Registration

Instance Registration provides a simple, straightforward way to retrieve and store information about multiple instances of a database running the Oracle Parallel Server. Previously, there was no generic mechanism in the Oracle Parallel Server that permitted multiple instances to retrieve information about each other.

Parallel query will probably use *Instance Registration* the most. When servers are spawned to parallelize an operation, information is needed about remote instances to determine how many users to start and on which instances to start them. Using *Instance Registration*, parallel query is now able to determine the address of the servers and which instances are most powerful and least loaded.

Instance Registration now provides a generic interface that retrieves and stores information to be used for the following purposes:

- cross instance communication
- detecting the existence and availability of remote servers participating in parallel operations
- load balancing information for parallel operations
- local instance number assignment and identification

For more information about *Instance Registration*, see *Oracle7 Parallel Server Concepts & Administration*.

Delayed–Logging Block Clean Out

Delayed–Logging Block Cleanout provides the following Block Cleanout improvements:

- The current block is not dirtied.
- No redo logs are generated when the current block is cleaned out during reads.
- Block cleanout during current reads is faster and less costly because the block is changed directly and no redo log is generated.
- For the Parallel Server, *Delayed–Logging Block Cleanout* does not cause data or index block pings during current reads because the blocks are not dirtied.

- *Delayed–Logging Block Cleanout* introduces a No–IO Delayed Block Cleanout to be done by DBWR when writing DIRTY, but not yet cleanout, blocks to disk.

The basic idea of *Delayed–Logging Block Cleanout* is to not regenerate redo records when cleaning out the current block during reads. The block is cleaned out in the usual way, but a new no–logging flag is set for the block and for every ITL that has been cleaned out. The block is marked CLEANOUT (a new state for DBWR) but not DIRTY, and no redo record is generated at this time.

> **Note:** *Delayed–Logging Block Cleanout* improves system performance, especially when running OPS.

For more information about *Delayed–Logging Block Cleanout*, see *Oracle7 Parallel Server Concepts & Administration*.

Parallel Query Affinity

Parallel Query Affinity provides the following new functionality:

- a new mechanism for slave allocation that takes into account a balance between disk transfer rate and CPU processing rate for user's queries
- the physical proximity of data is taken into account when assigning work to query slaves on machines that have preferred access to local disks

The following initialization parameters are now obsolete:

- PARALLEL_DEFAULT_SCANSIZE
- PARALLEL_DEFAULT_MAX_SCANS

A new initialization parameter replaces the two obsolete parameters. The new initialization parameter, PARALLEL_MIN_PERCENT, allows the user to specify the minimum fraction of parallel query slaves desired. The user can specify an integer number in the range of 0 to 100 for PARALLEL_MIN_PERCENT.

For more information about *Parallel Query Affinity*, see *Oracle7 Server Tuning*.

Load Balancing in Listener

Prior to Release 7.3, no coherent mechanism was available to manage the load of numerous instances that constituted an Oracle Parallel Server (OPS). *Load Balancing in Listener* now provides load balancing in the SQL*Net network listener among multiple instances.

The user is now provided with the following functionality:

- a GUI–based Network Manager tool that permits multiple users per database instance.
- a new initialization parameter, MTS_LISTENER_ADDRESS, that sets the configuration for each port to which that database will connect.

GUI Network Manager Tool

Oracle Network Manager is a graphical tool that is used for configuring and maintaining a SQL*Net network, including the Listener.

MTS_LISTENER_ ADDRESS Parameter

The MTS_LISTENER_ADDRESS parameter sets the configuration for each port to which the database will connect; it's syntax is

```
MTS_LISTENER_ADDRESS=<addr>
```

where *addr* is an address to which the listener will listen for connection requests for a specific protocol.

For more information about *Load Balancing in Listener*, see *Oracle7 Parallel Server Concepts & Administration*.

Serviceability Enhancements

This section contains the following topics:

- DB_VERIFY
- Transaction Trace Facility

DB_VERIFY

DB_VERIFY, which is available with Release 7.3, is an external command–line utility that performs physical data structure integrity checks on offline databases. Checking is limited, as follows:

- Only physical data structure integrity is verified.
- Only transaction managed blocks are checked.

The benefits of performing an offline check are

- One can ensure that a backup database (or datafile) is valid before being restored.

- An existing database can be checked without being brought online by using an external utility.
- Significant performance improvement can be achieved by using an external tool as compared with the online verification capabilities of pre–7.3 releases.
- The time required for verification is comparable to the time required for copying the file.
- Verification provides informative reports that describe the nature of the problem.

DB_VERIFY provides the following functionality:

- Verifications can be performed on either a file or a piece of a file.
- Verifications perform physical data integrity checking of all cache managed blocks that support such checking.
- Verifications perform minimal logical data structure integrity checking of all cache managed blocks; such checking is limited in scope and does not rely upon the data dictionary.

  Note: If corruption is detected, the statistical information provided by DB_VERIFY should be communicated to Oracle World Wide Support for further analysis.

OSDoc

Additional Information: The name and location of DB_VERIFY is dependent on your operating system. See your operating system–specific Oracle documentation for the location of DB_VERIFY for your system.

For more information about DB_VERIFY, see the *Oracle7 Server Administrator's Guide* and *Oracle7 Server Utilities*.

Transaction Trace Facility

The *Transaction Trace Facility* enhancement provides database users, database administrators, and application developers with information about

- the status of a transaction (active, in doubt, and so forth)
- the session to which a transaction belongs
- the parent of a transaction
- the type of the transaction (initiated, recursive, and so forth)
- the starting time of the transaction
- the starting location of the transaction
- the amount of UNDO information the transaction has generated

- the number of data blocks the transaction has updated
- the objects the transaction has updated
- the number of CR changes the transaction has made

The following dynamic performance tables have been changed to support the *Transaction Trace Facility*:

- V$TRANSACTION
- V$ROLLSTAT
- V$LOCKED_OBJECT

For more information about the *Transaction Trace Facility* and the modifications to the dynamic performance tables, see *Oracle7 Server Reference.*

Tuning Enhancements

This section contains the following topics:

- EXPLAIN PLAN changes
- Oracle TRACE™
- Antijoins

EXPLAIN PLAN changes

The enhancement to EXPLAIN PLAN improves the readability and usefulness of EXPLAIN PLAN output.

A new CHAR column, OTHER_TAG, which describes the function of the SQL text in the OTHER column, has been added to the EXPLAIN PLAN table for Release 7.3. The values for OTHER_TAG are

SERIAL	The SQL is the text of a locally executed, serial query plan.
SERIAL_FROM_REMOTE	The SQL shown will be executed at a remote site.
PARALLEL_COMBINED_ WITH_PARENT	The parent of this operation is a DFO that performs both operations in the parallel execution plan.
PARALLEL_COMBINED_ WITH_CHILD	The child of this operation is a DFO that performs both operations in the parallel execution plan.

PARALLEL_TO_SERIAL	The SQL is the top level of the parallel plan.
PARALLEL_TO_PARALLEL	The SQL is executed and outputs it in parallel.
PARALLEL_FROM_SERIAL	This operation consumes data from a serial operation and outputs it in parallel.

Several new columns have been added for the OPTIMIZER:

- COST
- CARDINALITY
- BYTES

For more information about the changes to EXPLAIN PLAN, see *Oracle7 Server Tuning*.

Oracle TRACE™

Oracle TRACE™, often referred to simply as TRACE™, is a software product that collects performance data for any application—most notably, transaction processing and database applications. It monitors performance by gathering and reporting event–based data from layered products and application programs that contain calls to TRACE™ routines. TRACE™ is designed to operate with minimal performance impact on the system and can be used in both development and production environments.

TRACE™ differs from other collector software in that it is event based, whereas most other collectors are timer based. Timer–based collectors gather data at specified time intervals, at random places within your code. An event–based collector gathers data at predefined locations in your program code when that code is executed.

The advantage of event–based collectors is that you can determine the actual frequency of the execution of events, rather than an average or estimated frequency. Also, event–based collectors give you the ability to collect and report on the resources used by specific events in an application.

TRACE™ users include application developers, application performance analysts, database administrators, system managers, and capacity planners. They use TRACE™ to assist them in pinpointing the reasons for an application's poor performance. General reasons for poor performance can be any of the following:

- data access contention
- poorly or incorrectly designed databases

- not enough servers to handle user requests
- inefficient database queries
- actual use of the application differs from the intended use
- inadequate hardware resources

Finding specific causes for these general problems requires data about the application's resource use and response time. TRACE™ collects a variety of such data from all layers of an application—the user interface, the processing engine, and the database. TRACE™ is unique in that it can collect information from each of these layers, transcending the proprietary and industry standard application programming interfaces (APIs). Each layer that logs TRACE™ information can be tied to the layer above it, which allows you to track a transaction throughout its lifetime.

For more information about Oracle TRACE™ parameters, see *Oracle7 Server Tuning* and *Oracle7 Server Reference.*

Antijoins

An antijoin is a form of join with reverse logic; instead of returning rows when there is a match, (according to the join predicate), between the left and right side, an antijoin returns those rows from the left side for which there is no match on the right. The behavior of an antijoin is exactly that of a NOT EXISTS subquery with the right side of the antijoin corresponding to the subquery.

Release 7.3 introduces *Antijoin*. The following list summarizes the new functionality and important facts about *Antijoin*:

- The hash and sort–merge antijoins are introduced as alternative evaluation techniques available for NOT IN subqueries.
- Hash and sort–merge antijoins can only be invoked explicitly through hints or initialization parameters in the INIT.ORA file.
- The new antijoins are parallelizable.

Restrictions on the Use of Antijoin Methods

Release 7.3 can use hash and sort–merge antijoins to evaluate NOT IN subqueries provided that certain conditions are met. Assume that the subquery predicate is of the form

```
(col_a1, col_a2,   ,col_an) NOT IN
(SELECT col_b1, col_b2,   ,col_bn FROM ...)
```

The following conditions must hold in order for the subquery to be transformed into a hash or sort–merge antijoin:

- All of $col_{a1}...col_{an}$ must be simple references to columns; $col_{b1}...col_{bn}$ must be either simple references to columns or aggregate functions (MIN, MAX, SUM, COUNT, or AVG) applied directly to a simple column provided that the subquery contains a GROUP BY. No other expressions are allowed.
- All of $col_{a1}...col_{an}$ and $col_{b1}...col_{bn}$ must not be NULL. Currently, this is checked for each column by either the presence of a NOT NULL constraint or a NOT NULL predicate on the columns at the topmost logical level of the appropriate WHERE clause ANDed with any other predicates in that Where clause. That is, NOT NULL predicates should be in the WHERE clause of the subquery for columns referred to in $col_{b1}...col_{bn,}$ and in the WHERE clause of the surrounding query for columns $col_{a1}...col_{an.}$
- The subquery must not have any correlation predicates, in other words, predicates that reference anything in surrounding query blocks.
- The WHERE clause of the surrounding query must not have ORs at the topmost logical level. That is, the subquery must not be part of a logical expression that is ORed with some other logical expression.
- Antijoins can be used only with the cost based optimizer.

How to Invoke Antijoin Methods

If invoked by a hint, the hint is put in the NOT IN subquery and must be either of the following antijoins:

- MERGE_AJ for sort merge antijoins
- HASH_AJ for hash antijoins

The antijoin transformation can also be invoked based on the setting of a new initialization parameter, ALWAYS_ANTI_JOIN. If the parameter ALWAYS_ANTI_JOIN is set to either MERGE or HASH, the transformation to the corresponding antijoin type takes place wherever it is legal.

If the antijoin transformation takes place, the antijoin appears as a join in the explain plan output with the word “ANTI” in the options column of the PLAN_TABLE. The right side of the antijoin appears as a view in the query plan.

For more information about *Antijoins*, see *Oracle7 Server Tuning*.

Advanced Replication Enhancements

This section contains the following topics:

- Object Groups
- Synchronous Propagation
- Replicated Table Comparison

Object Groups

Release 7.3 introduces the idea of an *object group* which replaces the schema as the logical unit of distribution in Oracle’s Advanced Replication feature.

An *object group* is a set of replicated objects. The replicated objects may reside in one or more schemas, but any replicated object can belong in, at most, one object group. Instead of replicating schemas, users now replicate *object groups*. Release 7.3 identifies the objects and schemas that need to be replicated. Object groups provide the following benefits:

- Replication administration is simplified by letting users define configurations and execute replication commands at a higher level than was previously possible.
- Instead of replicating individual objects, users are now able to group objects that participate in an application function and replicate them as a group.

Modifications to RepCat Tables

The following table lists the RepCat tables used by Advanced Replication and indicates which tables were changed in Release 7.3.

RepCat Table	Comments
REPCAT$ REPCAT	no change
REPCAT$ REPOBJECT	a new column has been added: GNAME. Its type is VARCHAR2(30)
REPCAT$ REPPROP	no change
REPCAT$ REPSCHEMA	no change
REPCAT$ DDL	no change
REPCAT$ GENERATED	no change
REPCAT$ REPCATLOG	a new column has been added: GNAME. Its type is VARCHAR2(30)

Table D – 10 Advanced Replication Table Changes

Modifications to RepCat Views

Views associated with the modified table now include the GNAME column, as shown in the following table:

RepCat View	Comments
REPCAT_REPCAT	no change; gname=sname
USER_REPCAT	
ALL_REPCAT	
DBA_REPCAT	
REPCAT_REPOBJECT	add gname column; gname=NVL(gname,sname)
USER_REPOBJECT	
ALL_REPOBJECT	
DBA_REPOBJECT	
REPCAT_REPSCHEMA	no change; gname=sname
USER_REPSCHEMA	
ALL_REPSCHEMA	
DBA_REPSCHEMA	
REPCAT_REPPROP	no change
USER_REPPROP	
ALL_REPPROP	
DBA_REPPROP	
REPCAT_REPCATLOG	add gname column; gname=NVL(gname, sname)
USER_REPCATLOG	
ALL_REPCATLOG	
DBA_REPCATLOG	

Table D – 11 Advance Replication View Changes

Changes to RepCat API Procedures

In Release 7.3, the PL/SQL procedures used to create, maintain, and drop repschemas are modified to operate on object groups. There are new Release 7.3 _REPGROUP() procedures that have been added to replace Release 7.2 _REPSCHEMA() procedures. The following procedures, which are new in Release 7.3, check database compatibility:

- CREATE_MASTER_REPGROUP
- COMMENT_ON_REGROUP
- DROP_MASTER_REGROUP
- CREATE_SNAPSHOT_REGROUP
- DROP_SNAPSHOT_REGROUP
- REFRESH_SNAPSHOT_REGROUP

In Release 7.3, The PL/SQL procedures used to create, maintain, and drop repschemas are modified to operate on object groups. The following is a list of Release 7.2 procedures that have been converted to operate on object group names (GNAME) instead of replication schema names (sname).

- ADD_MASTER_DATABASE
- REMOVE_MASTER_DATABASE (both)
- CREATE_MASTER_REPOBJECT
- SUSPEND_MASTER_ACTIVITY
- RESUME_MASTER_ACTIVITY
- RELOCATE_MASTERDEF
- SWITCH_SNAPSHOT_MASTER
- EXECUTE_DDL (both)
- PURGE_MASTER_LOG
- WAIT_MASTER_LOG
- DO_DEFERRED_REPCAT_ADMIN

For more information about *Object Groups*, see *Oracle7 Server Distributed Systems, Volume II*.

Synchronous Propagation

Release 7.3 introduces synchronous propagation of transactions. Every delete, update, or insert on a replicated table triggers a synchronous RPC to each remote, synchronous site. Synchronous propagation utilizes two–phase commit for distributed transactions. In addition to Release 7.2 configurations, in which the method of propagation was globally asynchronous, users can now create configurations with global synchronous communication or mix propagation methods.

New Procedures

Release 7.3 introduces three new procedures that ensure that the method of propagation between any two master sites is symmetric. The three procedures are:

- ALTER_OBJECT_PROPAGATION
- ALTER_GROUP_PROPAGATION
- ALTER_DATABASE_PROPAGATION

ALTER_OBJECT_PROPAGATION

```
ALTER_OBJECT_PROPAGATION  (SNAME IN VARCHAR2,
                           ONAME IN VARCHAR2,
                           TYPE IN VARCHAR2,
                           HOW IN VARCHAR2,
                           DEST_BLINK IN VARCHAR2, := '',
                           SOURCE_DBLINK IN VARCHAR2 := '')
```

ALTER_OBJECT_PROPAGATION alters the propagation method for an object between two sites. If SOURCE_DBLINK is NULL, the local database is assumed to be the source site. If both SOURCE_DBLINK and DEST_DBLINK are NULL, all sites in the object's replication environment are altered. This procedure must be executed from the master definition site if SOURCE_DBLINK is not the local database.

Exceptions:

- MISSINGOBJECT
- NOTQUIESCED
- NONMASTERDEF
- VERSION
- COMMFAILURE
- RECONFIGERROR

ALTER_GROUP_PROPAGATION

```
ALTER_GROUP_PROPAGATION   (gname IN VARCHAR2,
                           HOW IN VARCHAR2,
                           DEST_DBLINK IN VARCHAR2 := '',
                           SOURCE_DBLINK IN VARCHAR2 := '')
```

ALTER_GROUP_PROPAGATION alters the propagation method for all replicated objects in an object group between two sites. If SOURCE_DBLINK is NULL, the local database is assumed to be the source site. If both SOURCE_DBLINK and DEST_DBLINK are NULL, all sites in the group's replication environment are altered. This procedure must be executed from the master definition site if SOURCE_DBLINK is not the local database.

Exceptions:

- MISSINGOBJECT
- MISSINGOBJGROUP (new)
- NOTQUIESCED
- NONMASTERDEF
- VERSION
- COMMFAILURE
- RECONFIGERROR

ALTER_DATABASE_PROPAGATION

```
ALTER_DATABASE_PROPAGATION(HOW IN VARCHAR2,
                           DEST_DBLINK IN VARCHAR2 := '',
                           SOURCE_DBLINK IN VARCHAR2 := '')
```

ALTER_DATABASE_PROPAGATION alters the propagation method for all replicated objects between two sites. If SOURCE_DBLINK is NULL, the local database is assumed to be the source site. If both SOURCE_DBLINK and DEST_DBLINK are NULL, all sites in the replication environment are altered. This procedure must be executed from the master definition site if SOURCE_DBLINK is not the local database. Exceptions are the following:

- MISSINGOBJECT
- MISSINGOBJGROUP (new)
- NOTQUIESCED
- NONMASTERDEF
- VERSION

- COMMFAILURE
- RECONFIGERROR

For more information about *Synchronous Propagation*, see *Oracle7 Server Distributed Systems, Volume II.*

Replicated Table Comparison

Table Comparison of the Advanced Replication option enables you to determine, in a running system, if two replicated tables are either the same or different and, if different, the nature of the difference. Specifically, *Table Comparison* does the following:

- The differences between tables is reported precisely. The exact rows that are different are reported.
- The result of rectifying differences between tables is that each compared table must be exactly equivalent to the reference table. Exact equivalence means that the table at each site has the same *shape* and the same *content*. The shape of a table refers to the number of columns, their column names, and the column data types. The content of a table refers to the number of rows and the actual values for each column on a row.

Table Comparison uses set difference to determine which rows are different. You can improve the performance of table comparison by setting the following initialization parameters in the INIT.ORA file:

Initialization Parameter Setting	Comments
SORT_DIRECT_WRITES=TRUE	If set to TRUE, each sort allocates additional buffers in memory for direct writes.
SORT_WRITE_BUFFERS	Specifies the number of buffers.
SORT_WRITE_BUFFER_SIZE	Specifies the size of the buffers.
SORT_AREA_SIZE=1000000	The system default is 64000. This increases the size of the area where rows are sorted and will improve performance dramatically.

Table D – 12 Initialization Parameter Settings for Table Comparison

For more information about *Table Comparison*, see *Oracle7 Server Distributed Systems, Volume II.*

Interface Enhancements

This section contains the following topics:

- Thread Safety, OCI
- Thread Safety, Pro*
- Piecewise Binds and Defines for String and Raw Data
- Binding/Defining Arrays of Structures in OCI
- UNSAFE_NULL_FETCH, Pro*

Thread Safety, OCI

Thread Safety, OCI allows developers of Oracle applications to use Oracle interfaces or embedded SQL in a multi–threading environment. Implementation of thread safety now makes OCI code reentrant and allows multiple threads of a user program to make OCI calls without having any side effect from one thread to another.

The principal benefits of *Thread Safety, OCI* are

- Multiple threads can use Oracle library calls with the same result as if they were executed serially.
- Users who do not use the thread–safety property do not pay the performance penalty for thread–safe libraries.
- Cursors to statements in a session are shareable serially. This means that connections can be shared among threads; however, two threads should never share the same connection at the same time.

Using Thread Safety, OCI

You must inform the OCI layer that your environment is single–threaded or multi–threaded in order to use the services of the OCI layer. Therefore, you must execute an OCI process initialization call, OPINIT before any other OCI calls are issued. If the OPINIT call is skipped then, for backward compatibility, a single–threaded environment is assumed. The syntax of the OPINIT call is

```
SWORD OPINIT (ub4 MODE);
```

The allowed values for the MODE parameter are

OCIEVDEF	for single–threaded environments
OCIEVTSF	for multi–threaded environments

A new logon call, OLOG, must now be used instead of the ORLON or OLON calls. The syntax of OLOG is

```
SWORD OLOG (struct        cda_def*lda,
                          ub1*hda,
                          text*uid,
                          sword uidl,
                          text*pswd,
                          sword pswdl,
                          text*conn,
                          sword conn1,
                          ub4 MODE);
```

For more information about *Thread Safety*, *OCI*, see the *Programmer's Guide to the Oracle Call Interface*.

Thread Safety, Pro*

*Thread Safety, Pro** enables Pro*C and Pro*Ada application developers to write applications that operate in a preemptive threads environment, for example DCE and OS/2, by providing a thread–safe, Pro* runtime library and generating thread–safe code. The following applications are now possible:

- A multi–threaded application with persistent threads: In this case, multiple threads each establish and maintain one or more connections to the database. Since threads are expected to be persistent, once a thread dies, there is no way for a different thread to resume processing of any pending statements.
- A multi–client configuration with transient threads: In this case, a client is analogous to an application. There may be multiple clients executing database calls. A *non–persistent* thread is used to process a unit of work for the client. Once a thread dies or is suspended, its runtime context may be used by a different thread.

Embedded SQL Statements

New embedded SQL statements that support *Thread Safety, Pro** are summarized in Table D – 13.

Embedded SQL Statement	Definition
EXEC SQL ENABLE THREADS	This statement is required for correct process initialization. It should be called only once and before any threads are spawned. It does nor require any host variables.
SQL_CONTEXT:ctx1	This is a user program variable. It must appear in the DECLARE section for those languages that require a DECLARE section, such as COBOL and FORTRAN. Its scope and visibility are determined by the placement in your program and the host language programming rules.
EXEC SQL CONTEXT:ctx1	This statement is a precompiler directive. It tells the precompiler which runtime context to use on subsequent executable SQL statements.
EXEC SQL CONTEXT ALLOCATE:ctx1	This function initializes the SQLLIB runtime context that is referenced in an EXEC SQL CONTEXT USE statement. In a multi–threaded application, this call should be executed once for each thread. In a multi–client configuration, this call should be executed once per client.
EXEC SQL CONTEXT FREE:ctx1	This function will free all memory associated with a runtime context and put a null pointer in your program variable.

Table D – 13 Embedded SQL Statements

New Command Line Option, THREADS=YES

A new command line option, THREADS=YES is a precompiler option that is required for any program requiring multi–threaded support. It is allowed only on the command line and in a configuration file. If the THREADS=YES option is in effect, the precompiler generates an error if it encounters any executable SQL statements and no context is visible.

For more information about *Thread Safety, Pro**, see the *Programmer's Guide to the Pro*Ada Precompiler* and the *Programmer's Guide to the Oracle Pro*C/C++ Precompiler.*

Piecewise Binds and Defines for String and Raw Data

With Release 7.3, a long column can now be inserted, updated, or fetched, *in pieces*, by an application program.

The following table summarizes the functionality of four new functions that support *Piecewise Binds and Defines for String and Raw Data.*

Function	Definition
OBINDPS (OCI BIND Piecewise)	This function associates the address of a program variable with a placeholder in a SQL or PL/SQL statement.
ODEFINPS (OCI DEFINE Piecewise)	This function defines an output variable for a specified SELECT list item in a SQL query. It registers a context pointer that is returned, at runtime, to the application when TTC has to provide data incrementally to the application.
OGETPI (OCI GET Piece Information)	This function returns information about the next piece needed by, or to be given to, the TTC layer.
OSETPI (OCI SET Piece Information)	This function sets information about the next piece to be provided to, or to be fetched from, the TTC layer.

Table D – 14 New Functions for Support of Piecewise Binds and Defines

Warning: The functions defined in Table D – 14 are only compatible with Release 7.3 servers and beyond. If a Release 7.3 application attempts to use any of the functions shown in Table D – 14 against a Release 7.2 or earlier server, error message ORA–01551 is likely to be generated. At that point, you must restart the execution.

For more information about *Piecewise Binds and Defines for String and Raw Data*, see the *Programmer's Guide to the Oracle Call Interface*.

Binding/Defining Arrays of Structures in OCI

The previous OCI interface for performing multi–row operations forced the user to allocate a set of parallel arrays, one per column being inserted or fetched. The task of the host language programmer was, therefore, complicated because related data that should have been ideally a part of a single array of structures or records was split across several parallel arrays of scalars.

With Release 7.3, you may now place related scalars in a single structure. In other words, you can now perform inserts from, or fetches into, arrays of structures or records.

Two new calls to OCI are provided. The following table summarizes their functionality.

Function	Definition
OBINDPS	binds fields in arrays of structures or records
ODEFINPS	defines the fields

Table D – 15 Functions for Binding/Defining Arrays of Structures in OCI

Compatibility and Migration Issues

The following list summarizes the compatibility and migration issues associated with *Binding/Defining Arrays of Structures in OCI*:

- Existing user applications require no modification to work with *Binding/Defining Arrays of Structures in OCI.*
- There are no interoperability problems since the changes made are only on the client side.
- Since this feature is supported only in the deferred mode, applications that are linked with non–deferred UPI cannot take advantage of this feature.

For more information about *Binding/Defining Arrays of Structures in OCI,* see the *Programmer's Guide to the Oracle Call Interface* and the *Oracle7 Server Application Developer's Guide.*

UNSAFE_NULL_ FETCH, Pro*

UNSAFE_NULL_FETCH provides a new precompiler command line option that allows applications precompiled by Pro*COBOL, Release 1.8, Pro*FORTRAN, Release 1.8, and Pro*C, Release 2.2 with DBMS=V7 to FETCH null values into host variables that lack indicator variables without generating the ORA–01405 error message, "FETCHED column value is NULL", at runtime.

The new precompiler command line option is

```
UNSAFE_NULL_FETCH=YES/NO
```

The default value for option UNSAFE_NULL_FETCH is NO.

Restrictions on the USE of UNSAFE_NULL_FETCH

The following list explains the restrictions on the use of UNSAFE_NULL_FETCH:

- UNSAFE_NULL_FETCH is intended only for use with DBMS=V7. If used with command line option DBMS=V6 or DBMS=NATIVE, the UNSAFE_NULL_FETCH option generates a precompilation error.
- Source code precompiled with DBMS=V6 always behaves as though it has been precompiled with UNSAFE_NULL_FETCH=YES.
- Source code precompiled with DBMS=NATIVE behaves as though UNSAFE_NULL_FETCH=NO when the database is running in V7 mode and as though UNSAFE_NULL_FETCH=YES when the database is running in V6 compatibility mode.
- The UNSAFE_NULL_FETCH option cannot be used in combination with MODE=ANSI, because ANSI requires an error to be returned when a null value is returned to a host variable that does not have an indicator variable. This combination of options results in an error message at precompilation.
- Application code precompiled with UNSAFE_NULL_FETCH=NO generates error message ORA–01405 when FETCHing null values into host variables that do not have indicator variables.
- Application code precompiled with UNSAFE_NULL_FETCH=YES does *not* generate error message ORA–01405 when FETCHing null values into host variables that do not have indicator variables.

The net effect of the previously listed restrictions is that using UNSAFE_NULL_FETCH=YES suppresses error message ORA–01405 that otherwise is generated by the database when running in V7 mode. When UNSAFE_NULL_FETCH=NO, ORA–01405 is not suppressed.

For more information about *UNSAFE_NULL_FETCH*, see the *Programmer's Guide to the Pro*Ada Precompiler*, the *Pro*COBOL Supplement to the Oracle Precompilers Guide*, and the *Programmer's Guide to the Oracle Pro*C/C++ Precompiler*.

PL/SQL Enhancements

This section contains the following topics:

- PL/SQL Tables of Records and Call–by–Reference in PL/SQL
- PL/SQL File I/O
- Fetch from Cursor Variable

PL/SQL Tables of Records and Call–by–Reference in PL/SQL

PL/SQL Tables of Records and *Call–by–Reference in PL/SQL* provide the following new functionality in Release 7.3:

- record types with scalar fields are now supported as elements of PL/SQL tables
- new operators on PL/SQL tables
- call–by–reference parameters for passing PL/SQL tables as arguments to subprograms

The major benefit of *PL/SQL Tables of Records* and *Call–by–Reference in PL/SQL* is the improved capability to handle bulk data and composite data. For example, it is now easier for users to

- move collections of data between client side applications and stored programs
- use PL/SQL tables variables to move data into and out of relational tables
- manipulate PL/SQL tables in stored programs

New Operations on PL/SQL Tables

The following new built–in functions and procedures are provided with Release 7.3:

Function or Procedure	Comments
COUNT	the number of elements the PL/SQL table contains
FIRST	the index of the first element in the PL/SQL table
LAST	the index of the last element of the PL/SQL table
EXISTS	verifies that the PL/SQL table contains an elements at the given index
NEXT	the next (higher) index
PRIOR	the previous (lower) index
DELETE	remove the element of the PL/SQL table at the given index

Table D – 16 Built–in Functions and Procedures in Release 7.3

Syntax for Passing Arguments "by Reference"

The keyword BYREF is supported as a parameter mode, instead of IN/OUT, on parameters to which arguments will be passed by reference.

RPC for PL/SQL Tables of Records

Release 7.3 permits PL/SQL–to–PL/SQL RPC of PL/SQL tables of records from clients (that contain a PL/SQL executor) to server and from server to server.

> **Note:** The record types permitted as elements of index tables are records that do not have fields that are records or index tables.

For more information about *PL/SQL Tables of Records* and *Call–by–Reference in PL/SQL*, see the*Oracle7 Server Application Developer's Guide*.

PL/SQL File I/O

PL/SQL File I/O allows PL/SQL developers to read and write OS files using the same API on both client and server. Specifically, the following capabilities are now provided:

- It is now possible to write warnings, errors, and debug information outside of the database transaction cycle.
- It is now possible to read upgrade scripts on the server from a client Server Manager process and then execute those upgrade scripts.
- A file I/O process is now available that writes processing status and error messages to concurrent log files. The concurrent log files are stored in different product–specific subdirectories of a central directory.

Access to PL/SQL File I/O is provided through the UTL_FILE package. The available file operations are as follows:

- FOPEN function
- IS_OPEN function
- FCLOSE procedure
- FCLOSE_ALL procedure
- GET_LINE procedure
- PUT procedure
- NEW_LINE procedure
- PUT_LINE procedure
- PUTF procedure
- FFLUSH procedure

FUNCTION FOPEN

```
FUNCTION FOPEN     (LOCATION in VARCHAR2
                    FILENAME in VARCHAR2
                    OPEN_MODE in VARCHAR2) RETURN
                   UTL_FILE.FILE_TYPE;
```

where

LOCATION	is the operating system–specific string that specifies the directory or area in which to open the file
FILENAME	is the name of the file, including extension, without any directory information
OPEN_MODE	is a string that specifies how the file is to be opened

Function FOPEN returns a file handle that is used in subsequent file operations.

FUNCTION IS_OPEN

```
FUNCTION IS_OPEN   (FILE in FILE_TYPE) RETURN boolean;
```

where

FILE	is the value returned by FOPEN
FILE_TYPE	the contents of the file handle object (FILE_TYPE) are not visible to the user

Function IS_OPEN tests a file handle to determine if it identifies an open file.

PROCEDURE FCLOSE

```
PROCEDURE FCLOSE   (FILE IN OUT FILE_TYPE)
```

where

FILE	is the value returned by an FOPEN operation

PROCEDURE FCLOSE closes the open file identified by FILE.

PROCEDURE FCLOSE_ALL

```
PROCEDURE FCLOSE_ALL
```

This procedure closes all open files for the session. This is intended as an emergency cleanup procedure, to be used when a PL/SQL program exits on an exception.

PROCEDURE GET_LINE

```
PROCEDURE GET_LINE        (FILE IN FILE_TYPE
                           BUFFER OUT VARCHAR2)
```

where

FILE	is the value returned by an FOPEN operation
BUFFER	holds the read text

PROCEDURE GET_LINE reads a line of text from the open file identified by FILE and places the text in the output BUFFER.

PROCEDURE PUT

```
PROCEDURE PUT             (FILE IN FILE_TYPE
                           BUFFER IN VARCHAR2)
```

where

FILE	is the value returned by an FOPEN operation
BUFFER	is the text to be written

PROCEDURE PUT writes the text string BUFFER to the open file identified by FILE.

PROCEDURE NEW_LINE

```
PROCEDURE NEW_LINE        (FILE IN FILE_TYPE
                           LINES IN NATURAL := 1)
```

where

FILE	is the value returned by an FOPEN operation
LINES	is the number of line terminators to be written

PROCEDURE NEW_LINE writes LINES number of line terminators to the file identified by FILE. Default is a single line terminator.

PROCEDURE PUT_LINE

```
PROCEDURE PUT_LINE        (FILE IN FILE_TYPE
                           BUFFER IN VARCHAR2)
```

where

FILE	is the value returned by an FOPEN operation
LINES	is the text to write

PROCEDURE PUT_LINE writes text string BUFFER to the file identified by FILE, then writes a line terminator, in other words, calls PUT, then NEW_LINE.

PROCEDURE PUTF

```
PROCEDURE PUTF            (FILE IN FILE_TYPE
                           FORMAT IN VARCHAR2
                           ARG1 IN VARCHAR2
                           [ARG5 IN VARCHAR2])
```

where

FILE	is the value returned by an FOPEN operation
FORMAT	is the limited printf style format string
ARG1...ARG5	are the text substitution arguments

PROCEDURE PUTF formats the arguments ARG1...ARG5 according to the FORMAT string. The formatted text is then written to the file identified by FILE.

PROCEDURE FFLUSH

```
PROCEDURE FFLUSH          (FILE IN FILE_TYPE)
```

where

FILE	is the value returned by an FOPEN operation

PROCEDURE FFLUSH writes all pending data to a file. Normally, data written to a file may be buffered until enough bytes have accumulated. PROCEDURE FFLUSH forces the write to occur immediately.

For more information about *PL/SQL File I/O*, see the *PL/SQL User's Guide and Reference* and the *Oracle7 Server Application Developer's Guide*.

Fetch from Cursor Variable

Release 7.3 provides an extension to the cursor variable feature introduced in PL/SQL, Release 2.2. The new functionality available with PL/SQL, Release 2.3 and Oracle7 Server, Release 7.3 is as follows:

- The Cursor Variable feature is now extended to all clients of PL/SQL.
- Complete functionality of PL/SQL cursors for cursor variables is now available.
- A set of semantics is now available that is compatible with existing PL/SQL types and semantics for database cursors. The new semantics set is extensible to semantics of objects and object references, which is a feature anticipated for Object PL/SQL in Oracle8.

- Weak REF CURSOR type is now available.
- FETCH from a cursor variable and explicit cursor attributes are now available.
- Self–contained execution within PL/SQL is now available. (The Release 2.2 requirement for a bound *host* cursor variable is removed.)
- Client–Server communication within PL/SQL is now possible.
- Relaxed parameter mode (IN) on CLOSE of a cursor variable is now possible.

For more information about *Fetch from Cursor Variable*, see the *PL/SQL User's Guide and Reference* and the *Oracle7 Server Application Developer's Guide*.

Data Dictionary Changes

This section contains the following topics:

- Data Dictionary Views
- Dynamic Performance Tables

This section describes the changes to the Oracle7, Release 7.3 data dictionary. See *Oracle7 Server Reference* for descriptions of all standard tables and views that are available to developers.

Data Dictionary Views

The following table shows data dictionary views that are new in Oracle7, Release 7.3.

New Views

ALL_HISTOGRAMS	DEFCALL
ALL_UPDATABLE_COLUMNS	REPCAT$_REPOBJECT
DBA_FREE_SPACE_COALESCED	USER_HISTOGRAMS
DBA_HISTOGRAMS	USER_UPDATABLE_COLUMNS
DBA_UPDATABLE_COLUMNS	

Dynamic Performance Tables

This section lists the dynamic performance tables that are changed or new in Oracle7, Release 7.3.

Changed Views

V$CACHE_LOCK	V$SQLAREA
V$SQL	

New Views

V$LATCH	V$ROLLSTAT
V$LATCHNAME	V$SESSTAT
V$LIBRARYCACHE	V$SORT_SEGMENT
V$LOCKED_OBJECT	V$SYSSTAT
V$RECOVERY_FILE_STATUS	V$TRANSACTION
V$RECOVERY_STATUS	

Initialization Parameter Changes

This section lists initialization parameters that are obsolete, changed, or new in Oracle7, Release 7.3.

Dynamic Initialization Parameters

HASH_AREA_SIZE	HASH_MULTIBLOCK_IO_COUNT
HASH_JOIN_ENABLED	

Obsolete Initialization Parameters

INIT_SQL_FILES	SEQUENCE_CASH_HASH_ BUCKETS
LOG_ENTRY_PREBUILD_ THRESHOLD	SESSION_CACHED_CURSORS
PARALLEL_DEFAULT_MAX_ SCANSIZE	SMALL_TABLE_THRESHOLD
PARALLEL_DEFAULT_ SCANSIZE	

Changed Initialization Parameters

COMPATIBLE	
DB_BLOCK_LRU_STATISTICS	SORT_DIRECT_WRITES
PARALLEL_DEFAULT_MAX_ SCANS	

New Initialization Parameters

ALWAYS_ANTI–JOIN	ORACLE_TRACE_ COLLECTION_SIZE
DB_BLOCKS_LRU_LATCHES	ORACLE_TRACE_ENABLE
DB_FILE_STANDBY_ NAME_CONVERT	ORACLE_TRACE_ FACILITY_NAME
DELAYED_LOGGING_ BLOCK_CLEANOUTS	ORACLE_TRACE_ FACILITY_PATH
GC_DEFAULT_LOCKS	PARALLEL_MIN_PERCENT
GC_RELEASABLE_LOCKS	REMOTE_DEPENDENCIES_ MODE
LOG_FILE_STANDBY_ NAME_CONVERT	SHARED_POOL_ RESERVED_MIN_ALLOC
MAX_TRANSACTION_ BRANCHES	SHARED_POOL_ RESERVED_SIZE
MTS_MULTIPLE_LISTENERS	SORT_WRITE_BUFFER_SIZE
ORACLE_TRACE_ COLLECTION_NAME	SORT_WRITE_BUFFERS
ORACLE_TRACE_ COLLECTION_PATH	UTL_FILE_DIR

APPENDIX

E

Migration Utility Errors and Messages

The Migration Utility can return various error messages and informational messages. This appendix describes the errors that can be encountered when using the Migration Utility. For each error, probable cause and corrective action are given. Informational messages are also listed, although no action is necessary.

Cannot create conversion file, records exceed *num* bytes

Cause An internal error has occurred. A valid control file could not be created from the Version 6 control file.

Action Check the Version 6 control file for corruption, fix any problems, and rerun the Migration Utility.

CFILE – use this init.ora file

Cause This is an informational message that displays information about the CFILE command–line argument.

Action No user action is required.

CHECK_ONLY – estimate V7 catalog space requirement only

Cause This is an informational message that displays information about the CHECK_ONLY command–line argument.

Action No user action is required.

CHECK_ONLY and NO_SPACE_CHECK are mutually exclusive

Cause Two mutually exclusive command–line options were passed to the Migration Utility.

Action Rerun the Migration Utility using only one of these options.

Command–line argument value must be TRUE or FALSE (*str*)

Cause You entered a command–line argument with a value other than TRUE or FALSE.

Action Check the syntax of the command–line argument, correct the statement, and retry the operation.

Command–line arguments must be of the form <keyword>=<value> (str)

Cause You used a command–line argument improperly.

Action Check the syntax of the command–line argument, correct the statement, and retry the operation.

Command–line arguments:

Cause This is an informational message that displays the command–line arguments.

Action No user action is required.

Command name not found (*str*)

Cause An internal error has occurred; possibly the MIGRATE.SQL script is corrupted.

Action Check that the version of the Migration Utility, MIGRATE.BSQ, and of the target Oracle7 software are compatible, and that no corruption exists in MIGRATE.BSQ. Fix any such problems and rerun the migration.

Command not of form CMD(ARG1, ARG2, ...)

Cause An internal error has occurred; possibly the MIGRATE.SQL script is corrupted.

Action Check that the version of the Migration Utility, MIGRATE.BSQ, and of the target Oracle7 software are compatible, and that no corruption exists in MIGRATE.BSQ. Fix any such problems and rerun the migration.

CNVFILE – use this file for convert.ora

Cause This is an informational message that displays information about the CNVFILE command–line argument.

Action No user action is required.

Copy long command must be of form

Cause An internal error has occurred; possibly the MIGRATE.SQL script is corrupted.

Action Check that the version of the Migration Utility, MIGRATE.BSQ, and of the target Oracle7 software are compatible, and that no corruption exists in MIGRATE.BSQ. Fix any such problems and rerun the migration.

Could not find single contiguous extent of *num*bytes for c_file#_block#

Cause You do not have enough contiguous space in your SYSTEM tablespace.

Action Add free space to your SYSTEM tablespace and repeat the migration.

Could not find single contiguous extent of *num*bytes for c_ts#

Cause You do not have enough contiguous space in your SYSTEM tablespace.

Action Add free space to your SYSTEM tablespace and repeat the migration.

Could not translate logical name

Cause An internal error has occurred.

Action Check that the logical is defined correctly and rerun the Migration Utility.

Data type must be long for column *name*

Cause An internal error has occurred; possibly the MIGRATE.SQL script is corrupted.

Action Check that the version of the Migration Utility, MIGRATE.BSQ, and of the target Oracle7 software are compatible, and that no corruption exists in MIGRATE.BSQ. Fix any such problems and rerun the migration.

DBNAME – database name

Cause This is an informational message that displays information about the DBNAME command–line argument.

Action No user action is required.

Dictionary constant not found – *name*

Cause An internal error has occurred; possibly the MIGRATE.SQL script is corrupted.

Action Check that the version of the Migration Utility, MIGRATE.BSQ, and of the target Oracle7 software are compatible, and that no corruption exists in MIGRATE.BSQ. Fix any such problems and rerun the migration.

DUMPCF – dump control file symbolically

Cause This is an informational message that displays information about the DUMPCF command–line argument. You should generally only use this argument when asked to do so by your Customer Support representative.

Action No user action is required.

ECHO – echo all SQL statements to standard out

Cause This is an informational message that displays information about the ECHO command–line argument.

Action No user action is required.

Error calling slgtd

Cause An internal error has occurred; possibly the MIGRATE.SQL script is corrupted.

Action Check that the version of the Migration Utility, MIGRATE.BSQ, and of the target Oracle7 software are compatible, and that no corruption exists in MIGRATE.BSQ. Fix any such problems and rerun the migration.

Error closing file *name*

Cause An internal error has occurred; possibly the MIGRATE.SQL script is corrupted.

Action Check that the version of the Migration Utility, MIGRATE.BSQ, and of the target Oracle7 software are compatible, and that no corruption exists in MIGRATE.BSQ. Fix any such problems and rerun the migration.

File header does not fit in *num bytes*

Cause An internal error has occurred.

Action Check the control file for corruption, fix any problems, and repeat the migration.

Fixed portion of control file does not fit in *num* bytes

Cause An internal error has occurred.

Action Check the control file for corruption, fix any problems, and repeat the migration.

Found NULL SQL statement

Cause An internal error has occurred; possibly the MIGRATE.SQL script is corrupted.

Action Check that the version of the Migration Utility, MIGRATE.BSQ, and of the target Oracle7 software are compatible, and that no corruption exists in MIGRATE.BSQ. Fix any such problems and rerun the migration.

Free space found: *num*

Cause This is an informational message that informs you how much free space you have in your SYSTEM tablespace.

Action Be certain that you have enough free space before running the Migration Utility.

Incomplete write

Cause An internal error has occurred. Data could not be written to disk.

Action Check that the file access permissions are correct, that you have enough space or quota to write this file, and that the disk is not corrupt. Fix any problems and repeat the migration.

Insufficient space for new dictionaries, *num bytes needed, num found*

Cause There is insufficient room in your SYSTEM tablespace for the new data dictionary information.

Action Allocate the additional space required in the SYSTEM tablespace, and rerun the Migration Utility.

MIGFILE – use this file for MIGRATE.BSQ

Cause This is an informational message that displays information about the MIGFILE command–line argument.

Action No user action is required.

Mounting database ...

Cause This is an informational message. The Migration Utility is mounting the Version 6 database.

Action No user action is required.

NEW_DBNAME – new name for the database

Cause This is an informational message that displays information about the NEW_DBNAME command–line argument.

Action No user action is required.

NO_SPACE_CHECK – do not execute the space check

Cause This is an informational message that displays information about the NO_SPACE_CHECK command–line argument.

Action No user action is required. You should be certain that you have adequate space before running the Migration Utility with this option.

Offline tablespaces with outstanding undo, cannot proceed

Cause One or more tablespaces were offline, with outstanding save undo, when you attempted to use the Migration Utility to migrate.

Action If you are using Release 6.0.34.3, or later, complete the following steps:

- Query X$KTTVS and determine which tablespaces are offline and have outstanding transaction information. The tablespace number is listed as the third column of the X$KTTVS table.
- Bring each of these tablespaces online and take them offline without making any changes to the data in the tablespaces.
- Rerun the Migration Utility.

If you are using a pre–6.0.34.3 Release, first make certain that all offline tablespaces have been taken offline cleanly. Then rerun the Migration Utility with ALLOW_OFFLINE set TRUE. Use caution, however, since any tablespaces that were not taken offline cleanly cannot be recovered after migrating to Oracle7.

Opening database ...

Cause This is an informational message. The Migration Utility is opening the Version 6 database.

Action No user action is required.

ORA–*num*: ???

Cause The Migration Utility has received an ORA error and cannot retrieve the message text for the error.

Action Take appropriate action for Oracle error *num*.

Parameter file exceeds *num* bytes

Cause The parameter file for your Version 6 database exceeds the maximum size given.

Action If possible, reduce the size of your parameter file by removing obsolete parameters. Otherwise, contact your Customer Support representative.

Parameter buffer overflow

Cause The initialization parameter file is too large to fit in the buffer.

Action Reduce the size of the parameter file, possibly by removing any obsolete parameters, and repeat the migration.

PFILE – use this init.ora file

Cause This is an informational message that displays information about the PFILE command–line argument.

Action No user action is required.

Seek error in file *name*

Cause An internal error has occurred.

Action An error occurred reading file *name*. Check that the file and disk are not corrupted. Fix any such corruption before restarting the migration.

Short read, *num* bytes requested, *num* bytes read

Cause There is a problem reading the control file.

Action Check the CFILE parameter to make certain that the Migration Utility is using the correct control file.

Shutting down database (abort) ...

Cause An internal error has occurred.

Action Additional error messages should occur to inform you of the cause of the shutdown. Follow the actions suggested for these messages.

Shutting down database ...

Cause This is an informational message. The Migration Utility is shutting down the Version 6 database.

Action No user action is required.

SPOOL – spool output to file

Cause This is an informational message that displays information about the SPOOL command–line argument.

Action No user action is required.

Starting up database ...

Cause This is an informational message. The Migration Utility is starting up a Version 6 instance.

Action No user action is required.

Too many arguments in command (*num* max)

Cause You specified too many arguments on the command–line.

Action Check the syntax of the command and specify fewer command–line options.

Unable to open file *name*

Cause An internal error has occurred, or a database file was offline or not in the expected location when the Migration Utility was started.

Action Check to see that all database files are online before shutting down your Version 6 database. Check that the file exists and has access permissions that allow Oracle to open and read the file. If possible, check that the file is not corrupt and the disks on which the file resides are not corrupt. Fix any problems and restart the Migration Utility.

Unable to allocate buffer space to copy longs

Cause The Migration Utility could not allocate memory to serve as a buffer for copying LONG columns in the database.

Action Make sure enough machine resources are available for the utility and rerun the migration.

Unable to read file *name*

Cause An internal error occurred, or a database file was offline or not in the expected location when the Migration Utility was started.

Action Check to see that all database files are online before shutting down your Version 6 database. Check that the file exists and has access permissions that allow Oracle to open and read the file. If possible, check that the file is not corrupt and the disks on which the file resides are not corrupt. Fix any problems and restart the Migration Utility.

Unable to write file *name*

Cause An internal error occurred.

Action Check that the correct access permissions exist for Oracle to write to the file. Check that the disks to which the file is being written are not corrupt. Fix any such corruption before restarting the migration.

V7 catalog space requirement: *num*

Cause This is an informational message that informs you how much additional space you need in your SYSTEM tablespace to run the Migration Utility successfully.

Action Be certain that you have the specified amount of additional space before running the Migration Utility.

APPENDIX

F

Operating System–Specific Information

This manual occasionally refers to other Oracle documentation that contains detailed information for using Oracle on specific operating systems. Such documentation is usually referred to as, “operating–specific Oracle documentation”, although the exact name may vary among operating systems.

This appendix lists all the references in this manual to specific operating systems and related Oracle documentation. If you are using Oracle on multiple operating system, this appendix can help you ensure that your applications are portable across these operating systems.

The following table lists references to operating system specific information by topic:

Index

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X