Oracle8™

Migration

Release 8.0

December 1997

Part No. A58243-01

ORACLE®

Enabling the Information Age™

Oracle8 Migration

Part No. A58243-01

Release 8.0

Primary Author: Randy Urbano

Contributors: Karleen Aghevli, Bill Bridge, Maria Chien, David Colello, Sandy Dreskin, Jeffrey Hebert, Muralidhar Krishnaprasad, Thomas Kurian, Gordon Larimer, Lefty Leverenz, Tracy Lee, Bill Maimone, Joan Pearson, Elizabeth Pitt, Greg Pongracz, Mary Rhodes, Richard Sarwal, Carol Sexton, Alvin To, Alex Tsukerman, Douglas Utzig, Peter Vasterd, Lik Wong

Contents

5 After Migrating the Database

6 Upgrading Version 7 Applications

7 Migration Issues for the Version 8 ROWIDs

8 Upgrading and Downgrading

A Migration Utility Messages

B Control File Fixed View Changes

C Version 8 INIT.ORA Changes

D New SQL Key and Reserved Words

E General System Requirements for Migration

Index

Send Us Your Comments

Oracle8 Migration, Release 8.0

Part No. A58243-01

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

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

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

- electronic mail - infodev@us.oracle.com
- FAX - telephone number. Attn: Server Technologies Documentation Manager
- postal service:
  Oracle Corporation
  Server Technologies Documentation Manager
  500 Oracle Parkway
  Redwood Shores, CA 94065
  United States

If you would like a reply, please give your name, address, and telephone number below.

Preface

This manual guides you through the process of planning and executing migrations, upgrades, and downgrades for the Oracle database system. It describes basic principles and Oracle product features, and it contains step-by-step instructions for migration, upgrade, and downgrade operations.

The following topics are covered in this preface:

- Audience and Assumed Knowledge
- How Oracle8 Server Migration is Organized
- Conventions Used in This Manual
- Your Comments Are Welcome

Oracle8 Migration contains information that describes the features and functionality of the Oracle8 and the Oracle8 Enterprise Edition products. Oracle8 and Oracle8 Enterprise Edition have the same basic features. However, several advanced features are available only with the Enterprise Edition, and some of these are optional. For example, to use Objects, you must have the Enterprise Edition and the Objects Option.

> **See Also:** *Getting to Know Oracle8 and the Oracle8 Enterprise Edition* for information about the differences between Oracle8 and the Oracle8 Enterprise Edition and the features and options that are available to you.

Audience and Assumed Knowledge

This manual is for database administrators (DBAs), application programmers, security administrators, system operators, and anyone who plans or executes migration, upgrade, or downgrade operations on Oracle software. Users are assumed to be familiar with Version 7 of the Oracle server (Oracle7) and with their operating system environment. Users also are assumed to be familiar with Oracle database management system (DBMS) concepts. The first chapter of *Oracle8 Concepts* provides a comprehensive introduction to the concepts and terminology used in this migration manual.

How Oracle8 Server Migration is Organized

This manual contains the following chapters and appendices:

Chapter 1: Migration Overview

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

Chapter 2: Preparing to Migrate

This chapter describes the steps to complete before migrating the database.

Chapter 3: Migrating Using the Migration Utility

This chapter describes how to migrate a version 7 database to version 8 using the Migration Utility.

Chapter 4: Migrating Using Export/Import

This chapter describes how to migrate a version 7 or version 6 database to version 8 using the Export and Import utilities.

Chapter 5: After Migrating the Database

This chapter describes the steps to complete after migrating the database to version 8.

Chapter 6: Upgrading Version 7 Applications

This chapter describes how to upgrade version 7 applications and tools for use with version 8.

Chapter 7: Migration Issues for the Version 8 ROWIDs

This chapter covers issues associated with the new version 8 ROWIDs in relation to migrating columns containing ROWIDs to version 8.

Chapter 8: Upgrading and Downgrading

This chapter describes the steps to complete to upgrade a database from release 8.0.3 to release 8.0.4. This chapter also covers downgrading a version 8, release 8.0.4 database to release 8.0.3 or to version 7, release 7.3.

Appendix A: Migration Utility Messages

This appendix lists the messages displayed by the Migration Utility and includes an explanation for each message, probable cause(s) of each message, and suggested corrective action for each error condition.

Appendix B: Control File Fixed View Changes

This appendix briefly describes changes from version 7, release 7.3 to the version 8 server Control File Fixed Views.

Appendix C: Version 8 INIT.ORA Changes

This appendix briefly describes Oracle INIT.ORA file initialization parameters important for migration. Specifically, this appendix describes initialization parameters that have been added, changed, or dropped since release 7.3.

Appendix D: New SQL Key and Reserved Words

This appendix lists the keywords and reserved words new to version 8.

Appendix E: General System Requirements for Migration

This appendix discusses system requirements that may be important for successful migration to version 8.

Conventions Used in This Manual

The following conventions are used in this manual:

UPPERCASE Words	Uppercase calls attention to command keywords, object names, parameters, filenames, and so on. For example: "If you create a private rollback segment, its name must be included in the ROLLBACK_SEGMENTS parameter in the PARAMETER file."
Italicized Words	Italicized words indicate the first occurrence and definition of a term, as in the following example: "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." Italicized words also indicate emphasis and book titles.

Code Examples	SQL, Server Manager line mode, and SQL*Plus commands and statements are displayed in a fixed-width font, separated from normal text, as in the following 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. 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 you issue 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 our Oracle manuals. As we write, revise, and evaluate our documentation, your opinions are especially important input for us. Before the preface of each printed manual is a Reader's Comment Form, which we encourage you to use to tell us what you like and dislike about this manual or any other Oracle manual. If you do not find this form, please write your remarks to us in any convenient form. At your earliest convenience, please send your opinions to the following U.S. mail address, fax number, or email.

Server Technologies Documentation Manager
Oracle Corporation
500 Oracle Parkway
Redwood City, CA 94065
U.S.A.
Fax: 650-506-7200

infodev@us.oracle.com

1

Migration Overview

This chapter provides an overview of the major steps required to migrate a pre-version 8 database to version 8.

These migration procedures transform an existing pre-version 8 database system (including associated applications) into a version 8 database system. Version 8 is compatible with all earlier Oracle versions and releases. Therefore, databases transformed using the migration procedures described in this book can work in the same manner as in earlier versions and, optionally, can leverage new version 8 functionality.

Several preparatory steps are required before you migrate the current production database. After migrating a database, you should perform several additional test steps to test the migration. Other procedures enable you to add new version 8 functionality to existing pre-version 8 applications.

The following topics are covered in this chapter:

- Terminology
- Overview of Migration Steps
- Role of the Database Administrator During Migration
- Role of the Application Developer During Migration

Terminology

The following terms are used throughout this document:

Migration is the process of transforming an installed version of an Oracle database into a later version. For example, transforming a version 7 database into version 8 is migrating the database system.

Upgrading is the process of transforming an installed version of an Oracle database from an installed release into a later release of the same version. For example, transforming release 8.0.3 into release 8.0.4 is upgrading.

Downgrading is the process of transforming an installed version of an Oracle database from a later release back into an earlier release. For example, transforming an Oracle database from release 8.0.4 back into release 8.0.3 is downgrading, and transforming an Oracle database from version 8 back into version 7 is downgrading.

> **See Also:** Chapter 8, "Upgrading and Downgrading" for information about upgrading or downgrading.

The *source database* is the database to be migrated to version 8; typically, the source database uses an older version of Oracle software. The *target database* is the database into which you are migrating the source database; typically, the target database uses new version 8 software.

Overview of Migration Steps

Before you perform a database migration, you should understand the major steps in the migration process. These major steps apply to all operating systems, with the possible exception of a few platform-specific details identified in your *Installation Guide*.

Careful planning and use of version 8 tools can ease the process of migrating a database to version 8. The Migration Utility is the easiest way to migrate an entire database, while Export/Import and SQL copy utilities enable piecemeal migration of parts of a database.

The migration process includes the following major steps:

- Step 1: Prepare to Migrate
- Step 2: Test the Migration Process
- Step 3: Test the Migrated Test Database
- Step 4: Prepare and Preserve the Source Database
- Step 5: Migrate the Production Database
- Step 6: Tune and Adjust the New Version 8 Production Database

The following sections contain a brief outline of these steps. The purpose of these descriptions is to familiarize you with the major steps in the migration process. For detailed instructions, refer to the appropriate sections later in this book.

Step 1: Prepare to Migrate

- Become familiar with the features of the version 8 database. See *Getting to Know Oracle8 and the Oracle8 Enterprise Edition* for an overview of these features.
- Estimate and secure the system resources required for the migration.
- Decide which migration method to use, based on considerations involving the current production database, your migration objectives, and the behavior and capabilities of available migration methodologies.
- Develop a plan for testing the migration with a version 8 test database and a plan for testing the migrated version 8 production database.
- Prepare a backup strategy so that you can recover quickly from any unexpected problems or delays.

Step 2: Test the Migration Process

- Perform a test migration using a version 7 test database. The test migration should be conducted in an environment created for migration testing and should not interfere with the actual version 7 production database.

Step 3: Test the Migrated Test Database

- Perform the tests you planned in Step 1 on the pre-migration version 7 test database and on the version 7 test database that was migrated to version 8.
- Compare results, noting anomalies between test results on the pre-migration version 7 test database and on the version 7 test database that was migrated to version 8.
- Investigate ways to correct any anomalies you find and then implement the corrections.
- Repeat Step 1, Step 2, and the first parts of Step 3, as necessary, until the migration is completely successful and works with any required applications.

Chapter 2, “Preparing to Migrate”, provides detailed information about Steps 1 through 3.

Step 4: Prepare and Preserve the Source Database

- Prepare the current production database as appropriate to ensure that its migration to version 8 will be successful.
- Schedule the downtime required for backing up and migrating the pre-version 8 production database to version 8.
- Perform a full backup of the current production database. This step is required only if the Migration Utility is used for the migration.

Step 5: Migrate the Production Database

- Migrate the pre-version 8 production database to version 8.
- After the migration, perform a full backup of the production database.

Chapter 3 describes Steps 4 and 5 using the Migration Utility; Chapter 4 describes Steps 4 and 5 using Export/Import. Chapter 5 describes the backup procedure after the migration.

See Also: *Oracle8 Replication*, Appendix B, "Migration and Compatibility", if you are migrating a pre-version 8 database system that has Advanced Replication installed.

Step 6: Tune and Adjust the New Version 8 Production Database

- Tune the new version 8 production database. The version 8 production database should perform as good as, or better than, the pre-migration Oracle database. Chapter 5 describes these tuning adjustments.
- Determine which new features of the version 8 database are appropriate to use with your data and update your applications accordingly.
- Develop new database administration procedures as needed.
- Do not migrate production users to the version 8 database until all applications have been tested and operate properly. Chapter 6 describes considerations for updating applications.

Role of the Database Administrator During Migration

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

The specific DBA duties typically include:

- meeting with everyone involved in the migration process and clearly defining their roles during migration
- performing test migrations
- scheduling the test and production migration process
- performing backups of the pre-migration version 7 production database
- completing the production database migration
- performing backups of the newly migrated version 8 production database

Users should not have access to the migrated version 8 database until after all applications have been tested and operate properly.

Role of the Application Developer During Migration

The application developer is responsible for ensuring that applications designed for the pre-migration version 7 database work correctly with the migrated version 8 database. Application developers often test applications against the migrated version 8 database and decide which available new features of version 8 should be used.

Before migrating the version 7 production database, the DBA or application developer should install a version 8 test database. Then, the application developer tests and modifies the applications, if necessary, until they work with their original (or enhanced version 8) functionality.

The following references provide information about identifying differences in the migrated version 8 database that could affect particular applications. Application developers can use these differences to guide modifications to existing applications.

- Chapter 6, "Upgrading Version 7 Applications", describes the changes required to enable existing applications (that access a version 7 database) to access a version 8 database and provides guidance for upgrading version 7 applications to take advantage of version 8 functionality.
- *Getting to Know Oracle8 and the Oracle8 Enterprise Edition* describes enhancements in version 8.
- Appendix B lists changed data dictionary views that an application might require and SQL reserved and key words for version 8.
- *Oracle8 Parallel Server Concepts and Administration* and *Oracle8 SQL Reference* contain descriptions of changes and new version 8 functionality.
- For a pre-version 8 database system that has Advanced Replication installed, you must also refer to *Oracle8 Replication*, Appendix B, "Migration and Compatibility".

Version 8 provides features that aid in upgrading existing applications to version 8:

- Net8 and SQL*Net V2 support communication between Oracle versions.
- The programming interface is unchanged between Oracle versions.
- Oracle's backward compatibility accommodates small incompatibilities between different versions and releases.

2

Preparing to Migrate

This chapter covers the steps that must be completed before you migrate a production database. Steps 1 through 3 of the migration process, outlined in Chapter 1, “Migration Overview”, are covered in detail in this chapter:

- Step 1: Prepare to Migrate
- Step 2: Test the Migration Process
- Step 3: Test the Migrated Test Database

The information in this chapter is generic and applies generally to version 7 and version 6 production databases.

> **See Also:** *Oracle8 Replication*, Appendix B, “Migration and Compatibility”, if you are migrating a pre-version 8 database system that has Advanced Replication installed.

Step 1: Prepare to Migrate

This step includes the following actions, which are covered in detail in the following sections:

- Become Familiar with the Features of the New Version 8 Database
- Choose a Migration Method
- Assess System Requirements vs. Resources Available
- Start with Oracle Version 7, Release 7.X or Higher
- Avoid Common Migration Problems
- Prepare a Backup Strategy
- Develop a Testing Plan

Become Familiar with the Features of the New Version 8 Database

Before you plan the migration process, become familiar with the new features of the version 8 database. *Getting to Know Oracle8 and the Oracle8 Enterprise Edition* is a good starting point for learning the differences between a version 8 RDBMS and a release 7.3 RDBMS.

> **See Also:** *Oracle8 Parallel Server Concepts and Administration*, if you are using the Parallel Server option, for changes in Parallel Server.

Note: Version 8 training classes are an excellent way to learn how to take full advantage of the functionality available with version 8.

Choose a Migration Method

Use one of these three methods to migrate a database to version 8:

- Migration Utility, for migrating a version 7 database to version 8. See your platform-specific Oracle documentation for information about the earliest release that the Migration Utility can migrate on your platform. For example, on some platforms, the Migration Utility can migrate only release 7.1.4 and later databases.
- Export (full or partial) of a version 7 (or version 6) source database, followed by a full or partial Import into a version 8 target database.
- Copying data from a source database into a version 8 database using the COPY command or the AS clause of the CREATE TABLE command.

Table 2–1 summarizes these methods and lists their advantages and disadvantages.

Table 2–1 Advantages and Disadvantages of Migration Methods

Migration Method	Advantages	Disadvantages
Migration Utility: For migration of a complete database, version 7 to version 8 Changes datafile headers but leaves actual data unchanged. Does not copy data.	Automatic and requires minimal interaction by the DBA. Relatively fast, whatever the size of the database, because the data dictionary objects are the only objects that are changed. Imposes essentially no limit on the size of the database it can migrate. Usually requires relatively little additional disk space, when compared with other migration methods.	Performs only version 7 to version 8 migrations, and cannot downgrade back to version 7. Cannot perform release-to-release upgrades, for example release 8.0.3 to release 8.0.4. However, upgrades can be accomplished easily with the Oracle Installer. Cannot migrate selected parts of a database—migrates only the entire database.
Export/Import: For migration of parts of the database. Leaves datafile headers and actual data unchanged. Makes new copy of data.	Can migrate version 6 and version 7 databases to version 8. Can migrate specific parts of a database. Can be used to downgrade between versions of Oracle, for example, downgrading from version 8 to version 7. Can be used for release-to-release upgrade or downgrade operations, for example, upgrading from 8.0.3 to 8.0.4. Datafiles can be defragmented, and migrated data compressed, to improve performance. A database can be restructured with modified or new tablespaces, or by the partitioning of tables.	Extremely slow except for very small databases. Time required increases with the amount of data and use of LONG datatypes. Very large databases of several gigabytes may take many hours, and terabyte databases may take days. Requires large amounts of disk space for copying data into export file(s).
Copying Data: For migration of parts of the database. Leaves datafile headers and actual data unchanged. Makes new copy of data.	Datafiles can be defragmented, and migrated data compacted, to improve performance. A database can be restructured with modified or new tablespaces. Can migrate version 6 or version 7 databases to version 8. Can migrate specific parts of a database. Can be used for release-to-release upgrade or downgrade operations, for example, upgrading from 8.0.3 to 8.0.4.	Extremely slow except for very small databases. Time required increases with the amount of data and use of LONG datatypes. Very large databases of several gigabytes may take many hours, and terabyte databases may take days. Requires that both source and target databases be available at once during copying operations.

The following sections describe each of the migration methods in detail, covering the relative amounts of time and space they require and the situations in which they are appropriate.

Migration Utility

The Migration Utility converts files and structures in the version 7 source database to version 8 format, changing only the file headers and, if necessary, the definitions of the data in the files. The Migration Utility does not change the data portions of the datafiles, nor their format and content.

The primary advantages of using the Migration Utility are speed and ease of use. The Migration Utility takes significantly less time than Export/Import, and its use entails a standardized series of specific, easy steps. In addition, the time required to migrate a database with the Migration Utility depends less on the size of the database than on the number of objects in the data dictionary.

The Migration Utility is especially useful for quickly migrating an entire source database. Unlike Export/Import, the Migration Utility cannot selectively migrate specific datafiles. However, for databases with large amounts of data, large datatypes, and some other version 7 features, the Migration Utility may be the only practical tool for migration to version 8.

The Migration Utility requires only enough temporary space in the SYSTEM tablespace to hold both the version 7 (source) and version 8 (target) data dictionaries simultaneously.

The Migration Utility converts the entire database, including database files, rollback segments, and the control file(s). At any point before actually migrating the version 7 database, you can open and access data with the version 7 instance. However, once the Migration Utility has migrated the version 7 source database to version 8, you can go back to version 7 only by restoring a full backup of the version 7 source database.

> **See Also:** Chapter 3, “Migrating Using the Migration Utility”, for detailed information about using the Migration Utility.

Export/Import

Unlike the Migration Utility, the Export/Import utilities physically copy data in the source database to a new database. The source database's Export Utility copies specified parts of the source database into an export file. Then, the version 8 Import Utility loads the exported data into the new version 8 database. However, the new version 8 target database already must exist before the export file can be migrated into it.

The following sections discuss aspects of Export/Import operations that may help you to decide whether to use Export/Import for migrating your database.

> **See Also:** Chapter 4, "Migrating Using Export/Import", and also *Oracle8 Utilities*, for more information about using Export/Import for migration.

Export/Import Effects on Migrated Databases The Export/Import method of migration does not change the source database, enabling the source database to remain available throughout the migration process; however, if a consistent snapshot of the database is required (for data integrity or other purposes), the source database must run in restricted mode or must otherwise be protected from changes during the export procedure. Because the source database can remain available, you can, for example, allow an existing version 7 production database to continue running while the new version 8 database is being built at the same time by Export/Import. During this migration, to maintain complete database consistency, changes to the data in the version 7 database cannot be permitted without the same changes to the data in the version 8 database.

The Export/Import method also can be used to upgrade or downgrade. For example, the transformation of a version 8 database back into a version 7, release 7.3 database can be accomplished using Export/Import.

Most importantly, the Export/Import operation results in a completely new database. Although the source database ultimately contains a copy of the specified data, the migrated database may perform differently from the original source database. As a result of data defragmentation, database restructuring by the DBA, or the upgrade to version 8, expect changes in performance, data growth patterns, shared resource usage, data dictionary size, and object organization.

Careful planning, expert implementation, and rigorous testing are required to take advantage of the possible positive effects of Export/Import on the database; otherwise, the database changes may create problems. If the database was restructured during migration, and the migrated database behaves differently, it may be difficult to determine the cause of the differences.

Export/Import Benefits Data migration by Export/Import offers the following benefits:

- Defragmenting the data; that is, you can compress the imported data to improve performance.
- Restructuring the database; that is, you can create new tablespaces or modify existing tables, tablespaces, or partitions to be populated by imported data.
- Allowing the migration of specified database objects or users; that is, you can import only the objects, users, and other items that you wish.
- Serving as a backup archive; that is, you can use a full database export as an archive of the source database.

Export/Import Limitations Data migration by Export/Import has following limitations:

- Migrating a database by Export/Import requires an expert DBA. The combination of required planning and complicated execution typically requires multiple stagings and a great deal of practice before the final migration can be attempted.
- For a large database, a full database export/import can require a substantial amount of temporary storage space for the export dump file.
- The export may need to be partitioned into multiple jobs if the operating system does not support a file size as large as the database.
- Export/Import creates an entirely new database. To keep the source database in place, and to import its export dump file/data into a version 8 target database, requires the creation of target data files on the system *before* you import.
- Making multiple changes to the database at the same time, such as migrating to version 8 and defragmenting/restructuring the database at the same time, can hinder troubleshooting.
- To keep data in the source database and the target database synchronized during migration, either prohibit any data change in the source or make full provisions to mirror the changes in the migrated data in the target database.

Time Requirements for Export/Import Migrating an entire database by Export/Import can take a long time, especially compared with using the Migration Utility. Therefore, you may need to schedule the migration during non-peak hours or make provisions for propagating to the new target database any changes that are made to the source database during the migration.

The time and system resources (particularly disk space) required for Export/Import migration depend on DBA skill, database size, and the type of data to be migrated, particularly the number, size, and type of indexes that must be rebuilt.

For example, a relatively simple 6-gigabyte, version 7 database was migrated to version 8 using the Migration Utility in about an hour. The same version 7 database was exported, producing a single 2-gigabyte export dump file. To import that one export dump file took 20 hours. The complete migration using the steps described in "Migrate the Pre-Version 8 Source Database Using Export/Import" on page 4-3 took two days.

Consider the following factors related to the extended time required to migrate a database by Export/Import:

- Migration time easily exceeds the non-peak or off-production hours available in a typical daily schedule. Therefore, making the database unavailable for production tasks for the duration of the migration process might be impractical.
- If you make the source database read-only or do not allow changes to be made in it after the export, applications will be unavailable until after the import and final migration steps are completed.
- For larger databases, consider operating parallel export streams to reduce the time and optimize the process.

Data Definition Conversion by Version 8 Import When importing data from an earlier version, the version 8 Import Utility makes appropriate changes to data definitions as it reads earlier versions' export dump files. That is, it handles dump files produced by the Export utilities of Oracle version 6, version 7, and version 8. If the export source database is earlier than version 6, the source database *must* first be upgraded to at least version 6 before the export is performed.

Copying Data

You can copy data from one Oracle database to another Oracle database using database links. For example, you can 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, the CREATE TABLE ... FROM command, and the CREATE TABLE ... AS command.

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

Copying data, however, unlike Export/Import, allows the selection of specific rows of tables to be placed into the target database. Copying data is thus a good method for migrating only part of a database table. In contrast, using the Export/Import utilities to migrate data from version 7 to version 8, you can migrate only entire tables.

For example, to create a new table (NEW_EMP) that contains a subset of the data in an existing table (EMP@V7DB, only the employees in departments 10, 20, and 30), you can use the following SQL statement:

```
CREATE TABLE NEW_EMP
 (EMPNO         NUMBER(4),
 ENAME          VARCHAR2(10),
 JOB            VARCHAR2(9),
 MGR            NUMBER(4),
 HIREDATE       DATE,
 SAL            NUMBER(7,2),
 COMM           NUMBER(7,2),
 DEPTNO         NUMBER(2)) AS SELECT EMPNO, ENAME, JOB, MGR,
                 HIREDATE, SAL, COMM, DEPTNO
 FROM EMP@V7DB WHERE DEPTNO IN (10, 20, 30);
```

Copying data requires less disk space and memory buffer space for migration than Export/Import because copying data requires only that the source database and the target database both are online. There is no need to allocate large amounts of extra space for temporary files or for Export dump files.

The SQL COPY command is useful for working with large clustered tables. Further, the SQL*Plus COPY command can move portions of the cluster in parallel using Net8 (or SQL*Net). For more information about copying data from one database to another, refer to the CREATE TABLE command in the *Oracle8 SQL Reference* and to the COPY command in the *SQL*Plus User's Guide and Reference.*

Assess System Requirements vs. Resources Available

Estimate the system resources required for successful migration. Different migration methods may result in different resource requirements; therefore, if you are not certain of the method you want to use, complete an estimate for each potential method of migrating the existing database to version 8.

Consider the following factors in your estimates:

- configuration requirements for both the operating system and hardware
- the size of the existing production database
- possible size adjustments to it associated with implementing version 8

> **See Also:** Appendix E, "General System Requirements for Migration", and your platform-specific *Installation Guide*, for details about system requirements.

> **Note:** Version 8 binaries may require as much as three times the disk space required by version 7 binaries. To estimate disk space requirements, run the Migration Utility in CHECK_ONLY mode.

After you have chosen a migration method and estimated your requirements, secure the necessary resources for a successful migration.

Start with Oracle Version 7, Release 7.*X* or Higher

If you plan to use the Migration Utility, the earliest release supported by the Migration Utility is platform-specific. For example, on some platforms, the Migration Utility cannot migrate a release lower than version 7, release 7.1.4 (such as version 6, release 7.0, or release 7.1.3). See your platform-specific Oracle documentation for the supported releases on your platform.

If your database release number is lower than the release supported by the Migration Utility on your platform, upgrade or migrate the database to the required release. Use *Oracle7 Server Migration, Release 7.3* to migrate or upgrade the system to the required release. Then, use this *Oracle8 Migration* manual to migrate to version 8.

> **Note:** If you do not use the Migration Utility but instead use Export/Import or data copying, this restriction does not apply. You can use Export/Import or data copying to migrate data directly from a pre-version 8 database (for example, version 6, release 7.0, or release 7.1.3) to version 8.

Avoid Common Migration Problems

You can save time by eliminating common migration problems before you migrate your database. Common problem areas include the following:

- If you use ROWIDs stored in columns or in application code, see Chapter 7, "Migration Issues for the Version 8 ROWIDs". Because the format for ROWIDs is different in version 8, the old ROWIDs are invalid and must be converted.
- Make sure you do not use new reserved words as names for database objects (tables, columns, and so forth). See Appendix D, "New SQL Key and Reserved Words" for information.
- Make sure all Oracle product versions, operating system versions, and third-party software versions ar certified against version 8 on your platform. See your platform-specific *Installation Guide* for information.

Prepare a Backup Strategy

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

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

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

> **See Also:** The *Oracle8 Backup and Recovery Guide* for more information.

Develop a Testing Plan

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

> **WARNING: Failing to test rigorously before migration is risky and may lead to unpredictable results.**

The testing plan must include the following types of tests:

Migration Testing

Migration testing entails planning and testing the migration path from the source database to the migrated database, whether you use the Migration Utility, Export/Import, or other data-copying methods to migrate the production database data to the target database. These methods are discussed in Chapter 3, “Migrating Using the Migration Utility” and Chapter 4, “Migrating Using Export/Import”.

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

Minimal Testing

Minimal testing entails moving all or part of an 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 may not reveal potential issues that may appear in a “real-world” production environment. However, minimal testing will reveal any application startup or invocation problems immediately.

Functional Testing

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

Integration Testing

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

- Pro*C/C++ applications running against 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 version 8 instance or not.
- If the connection between two components involves Net8 or SQL*Net, those connections should also be tested and 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, you should understand the performance profile of the application under the source database. Specifically, you should understand the calls the application makes to the database kernel.

> **See Also:** *Oracle8 Tuning* for information about tuning. To thoroughly understand the application's performance profile under the source database, enable SQL_TRACE and profile with TKPROF.

Volume/Load Stress Testing

Volume and load stress testing tests the entire 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 Corporation has found that customers often do not conduct any kind of volume or load stress testing. Instead, customers often rely on benchmarks that do not characterize business applications. Benchmarks of the application should be conducted to uncover problems relating to functionality, performance, and integration, but they cannot replace volume and load stress testing.

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

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

Specific Pre-Migration and Post-Migration Tests

Include the following tests in your testing plan:

- timing tests
- data dictionary growth observations
- database resource usage observations, such as rollback and temporary segment usage

Collecting this information will help you compare the source database with the migrated target database.

Use EXPLAIN PLAN on both the source and target databases to determine the execution plan Oracle follows to execute each SQL statement. Use the INTO parameter to save this information in tables.

After migrating, you can compare the execution plans of the migrated database with the execution plans of the source database. If there is a difference, execute the command on the migrated database and compare the performance with the performance of the command executed on the source database.

Step 2: Test the Migration Process

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

- If you plan to use the Migration Utility, create a test version (typically a subset) of the source database to test migration.
- If you plan to use Export/Import, export and import small test pieces of the actual version 7 production database.

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

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

Make sure you upgrade any OCI and precompiler applications that you plan to use with your version 8 database. Then, you can test these applications on a sample Oracle database before migrating your production database. See "Upgrading Precompiler and OCI Applications" on page 6-2 for more information.

> **See Also:** Your platform-specific Oracle documentation for information about configuring a test database so that no operating system variables defined for the production database are affected by the test database.

Step 3: Test the Migrated Test Database

Perform the planned tests on the version 7 source database and on the test database that you migrated to version 8. Compare the results, noting anomalies. Repeat Step 1, Step 2, and Step 3 described in this chapter as necessary.

Test the newly migrated version 8 test database with existing applications to verify that they operate properly with a migrated version 8 database. You also might test enhanced functionality by adding features that use the available version 8 functionality. However, first make sure that the applications operate in the same manner as they did in the source database.

> **See Also:** Chapter 6, "Upgrading Version 7 Applications", for more information on using applications with version 8.

3

Migrating Using the Migration Utility

This chapter guides you through the process of migrating a version 7 database to version 8 using the Migration Utility.

> **See Also:** Some aspects of migration are platform-specific. See your platform-specific Oracle documentation for additional instructions about migrating on your platform.

The following topics are covered in this chapter:

- Overview of Migration Using the Version 8 Migration Utility
- System Considerations and Requirements
- Prepare the Version 7 Source Database for Migration
- Install the Version 8 Migration Utility
- Review Version 8 Migration Utility Command Line Options
- Migrate the Version 7 Source Database
- Errors During Migration
- Abandoning the Migration

WARNING: If you are migrating from Oracle8 Enterprise Edition to Oracle8 (formerly Workgroup Server), before you migrate you must modify any applications that use the advanced features of Oracle8 Enterprise Edition so that they do not use these advanced features. See *Getting to Know Oracle8 and the Oracle8 Enterprise Edition* for more information about the differences between the editions.

Overview of Migration Using the Version 8 Migration Utility

Migration converts the data dictionary and structures of a version 7 database into version 8 format. To migrate the database, the DBA first installs and runs the version 8 Migration Utility on the version 7 database. Then, the DBA executes a series of ALTER DATABASE commands on the new version 8 database. The completion of these procedures results in the conversion of the following version 7 structures into structures that can be used in version 8:

- Data files (file header only)
- Data dictionary
- Control file(s)
- Rollback segment(s)

Outline of the Migration Process

The following sections provide an outline of the migration process:

In the Version 7 Environment

- The DBA runs the version 8 Migration Utility, which creates and populates a new data dictionary based on the data dictionary of the version 7 database, and also creates a binary file based on the control file of the version 7 database. This binary file is called the convert file.

> **Note:** The DBA can run the version 8 Migration Utility multiple times (without opening the database in version 8) and still be able to return to the version 7 database. However, running the Migration Utility automatically eliminates the version 7 database catalog views (see "Abandoning the Migration" on page 3-20).

In the Version 8 Environment

- The DBA executes an ALTER DATABASE CONVERT statement, which creates a new control file based on the convert file generated by the Migration Utility, converts all online datafile headers to version 8 format, and mounts the version 8 database.

The file headers of offline datafiles and read-only tablespaces are not updated during migration. The file headers of offline datafiles are converted later when they are brought online, and the file headers of read-only tablespaces are converted when and if they are made read-write sometime after migration; however, they never have to be made read-write.

- The DBA executes an ALTER DATABASE OPEN RESETLOGS statement. This command automatically converts all objects and users defined in the new dictionary to version 8 specifications, and converts all rollback segments to version 8 format.

  If a source database rollback segment is in a tablespace that is offline when the version 8 database is opened, the rollback segment is not converted immediately to version 8 database format. Instead, the rollback segment is converted the first time the tablespace is brought online in version 8.

Using the Migration Utility

This section contains important considerations for using the Migration Utility.

Pre-Version 7 Databases

A version 6 database must be migrated to at least version 7 before it can be migrated to version 8. See your platform-specific Oracle documentation for information about the earliest release that is supported by the Migration Utility on your platform. For example, on some platforms, the Migration Utility can migrate only release 7.1.4 and later databases.

> **See Also:** *Oracle7 Server Migration, Release 7.3* for information about migrating or upgrading a database to a release that can be migrated by the Migration Utility on your platform.

Downgrading

Downgrading is the process of transforming an existing Oracle database to a previous version or release. The Migration Utility *cannot* transform a version 8 database back to version 7. In some situations, you can use another facility to downgrade, such as using Export/Import, restoring from backups, and possibly using other functions.

Databases with Advanced Replication Installed

If the Oracle database system has Advanced Replication installed, refer to *Oracle8 Replication*, Appendix B, "Migration and Compatibility" before running the Migration Utility.

System Considerations and Requirements

The following sections discuss system considerations and requirements for using the Migration Utility.

Space Requirements

Version 8 binaries may require as much as three times the disk space required by version 7 binaries. This requirement may cause you to run out of disk space during migration.

However, the Migration Utility requires relatively little temporary space. It needs only enough extra room in the SYSTEM tablespace to hold the new version 8 data dictionary simultaneously with the existing version 7 data dictionary.

The space required to hold an Oracle data dictionary depends on how many objects are in the database. Typically, a new version 8 data dictionary is approximately 50 per cent larger than its version 7 source data dictionary. If necessary, add space to the SYSTEM tablespace.

The Migration Utility will not complete the migration unless sufficient space is allocated in the SYSTEM tablespace before the migration begins. If there is not enough space, the Migration Utility will return an error specifying the amount of additional disk space required.

To determine disk space requirements for a successful migration, run the version 8 Migration Utility with the CHECK_ONLY command line option set to TRUE. The CHECK_ONLY command line option causes the Migration Utility to assess the amount of disk space required for migration, check the amount of space available, and issue an informational message about the disk space requirements. When the CHECK_ONLY command line option is set to TRUE, the Migration Utility does not build the version 8 data dictionary or perform any other migration processing.

Block Size Considerations

The value of DB_BLOCK_SIZE (a parameter in the INIT.ORA file) in the version 7 database and in the migrated version 8 database *must* be the same. Version 8 requires a minimum block size of 2048 bytes (2KB). Above this amount, integer multiples of your platform's physical block size are acceptable. However, multiples of 2KB, especially powers of 2—that is, 2KB, 4KB, 8KB, 16KB—provide for the most robust operation.

Make sure the version 8 block size setting meets the following criteria:

- Matches the version 7 setting.
- Is at least 2048 bytes (2KB). The version 8 Migration Utility displays an error message if the version 7 block size is less than 2KB.
- Is an integer-multiple of your platform's physical block size, preferably a multiple of 2KB.

Considerations for Replication Environments

You can migrate a version 7 replication environment to version 8. Version 7 sites can co-exist and run successfully with version 8 sites within the replication environment. However, take special care to accommodate the various replication features implemented on each system.

> **See Also:** *Oracle8 Replication*, Appendix B, "Migration and Compatibility", for detailed instructions about migrating systems using replication features.

Migrating to a Different Computer Architecture

The version 8 Migration Utility *cannot* migrate a database to a computer system that has a different architecture. For example, it will not migrate a database from a 32-bit platform to a 64-bit platform or from version 7 on Solaris to version 8 on Windows NT. Typically, different types of operating systems are associated with different architectures. Therefore, the Migration Utility generally will not migrate a database between operating systems. However, you normally can use Export/Import to migrate a database to a new architecture.

Character Encoding Considerations

Successful migration requires that you correctly specify the *character encoding scheme* for data in the version 8 database. All character data in the version 8 database is assumed to be in the character encoding scheme specified in the CREATE DATABASE command that created the database.

> **See Also:** *Oracle8 Reference* for information about National Language Support (NLS) for specifying these character encodings.

The Migration Utility uses the character set of the source version 7 database as the character set for the version 8 database, unless specified otherwise by the language parameter in the INIT.ORA initialization file (see also "NCHAR and NLS Use" on page 6-16 and "NCHAR and NLS Parameters and Compatibility" on page C-4). This character set cannot be changed after migration is complete; therefore, ensure that the correct character set is specified before you run the Migration Utility.

Prepare the Version 7 Source Database for Migration

Complete the following steps before you migrate your version 7 database to version 8:

1. If your database release number is lower than the release supported by the Migration Utility on your platform, upgrade or migrate the database to a supported release. For example, on some platforms, the Migration Utility cannot migrate databases lower than version 7, release 7.1.4 (such as version 6, release 7.0, or release 7.1.3).

   > **See Also:** Your platform-specific Oracle documentation for information about the releases supported by the Migration Utility on your platform. Then, use *Oracle7 Server Migration, Release 7.3* if you need to migrate or upgrade the system to the required release.

2. If the Procedural Option is not installed, use your version 7 installation media to install it. See your platform-specific *Installation Guide* for instructions.

3. Make sure all datafiles and tablespaces are either online or offline normal.

   The version 8 Migration Utility will not proceed, and will display an error, if any datafiles require media recovery. Tablespaces that are not taken offline cleanly must be dropped or brought online before migration. Otherwise, these tablespaces will not be available under version 8 after the migration. Typically, tablespaces that are taken offline by using an ALTER TABLESPACE OFFLINE IMMEDIATE or OFFLINE TEMPORARY command require media recovery.

Note: Tablespaces that are offline when you open the version 8 database remain in version 7 database file format. The offline tablespaces can be brought online at any time after migration, and the file headers are converted to version 8 format at that time. In addition, if you want to avoid large restores in the event of a failure, you can make all tablespaces except SYSTEM and ROLLBACK offline normal; then, you can restore only the datafiles for SYSTEM and ROLLBACK if you need to run another migration.

4. Make sure no user or role has the name MIGRATE, because the version 8 Migration Utility creates this schema and uses it to replace any pre-existing user or role with this name, and finally drops it from the system.

   To check for a user named MIGRATE, enter the following command:

   ```
   Select * from dba_users where username = 'MIGRATE';
   ```

   To check for a role named MIGRATE, enter the following command:

   ```
   Select * from dba_roles where role = 'MIGRATE';
   ```

5. Make sure the SYSTEM rollback segment is large enough by entering the following command:

   ```
   Select count(*) from dba_extents where segment_name = 'SYSTEM' and
   segment_type = 'ROLLBACK';
   ```

6. Make sure the SYSTEM rollback segment is not reaching maxextents by entering the following command:

   ```
   Select max_extents from dba_rollback_segs where segment_name = 'SYSTEM';
   ```

7. Make sure no redo information and no uncommitted transactions are outstanding, and resolve every pending, in-doubt transaction. Specifically, as discussed in the "Manually Overriding In-Doubt Transactions" section in *Oracle8 Distributed Database Systems*, make sure that *no in-doubt transactions* are left in DBA_2PC_PENDING.

8. Make sure your SYSTEM tablespace has enough free space to hold the version 8 data dictionary and the existing version 7 data dictionary concurrently.

   See Also: "Space Requirements" on page 3-4 for more information.

Install the Version 8 Migration Utility

Complete the following steps to install the version 8 Migration Utility executable from your version 8 installation media into the version 7 $ORACLE_HOME directory:

> **See Also:** Your platform-specific *Installation Guide* for detailed information about the Installation Utility.

> **Note:** The specific commands for some of the steps in the following procedure depend on your platform. See your platform-specific Oracle documentation and README file for information.

1. Run the version 8 Installation Utility. The utility name varies, depending on the platform—for example, **orainst** on UNIX systems or **oracleins** on VMS. See your platform-specific Oracle installation documentation for the command.

   > **Note:** If you do not want to use the Installation Utility to install the Migration Utility, you may be able to simply copy the necessary files onto your system from your installation media. See your platform-specific Oracle documentation for information about copying the necessary migration files.

2. On the “Install Type” screen, select “Default or Customer Install”.
3. On the “Select the Installer Activity” screen, select “Install, Upgrade, or De-Install Software”.
4. On the “Select the Installer Option” screen, select “Migrate from ORACLE7 to ORACLE8”.
5. On the “Installation Options: Home Locator” screen, enter the complete path of the version 7 $ORACLE_HOME location.
6. On the “Logging and Status” screen, confirm or change the log file location.
7. On the “Select an ORACLE7 to ORACLE8 Migration Action” screen, select “Install Migration Utility”.

8. On the “Install Source” screen, either select “Install from CD-ROM” or “Install from Staging Area”. If you select “Install from Staging Area”, the “Source Staging Area” screen appears and prompts you to enter the pathname of the source staging area.
9. On the “NLS” screen, select the native language for the installation.
10. On the “Software Asset Manager” screen, select “Migration Utility: Oracle7 to Oracle8 8.0.*X*”, where *X* is the latest Oracle release number.

    The Installation Utility installs the Migration Utility.
11. Check the installation to make sure that the Installation Utility has built the environment to run the Migration Utility on the version 7 database.

    For example, on a UNIX platform, make sure the Installation Utility has performed the following installation actions:

    – The version 8 Migration Utility executable, **mig**, is installed in the $ORACLE_HOME/bin directory on the version 7 environment.

    – Version 8 of the message file, **migus.msb**, is installed in the $ORACLE_HOME/rdbms/mesg directory on the version 7 environment.

    – Version 8 of **migrate.bsq** is installed in the $ORACLE_HOME/dbs directory on the version 7 environment.

    – The required NLS files are installed in the $ORACLE_HOME/migrate/nls/admin/data directory on the version 7 environment.

Review Version 8 Migration Utility Command Line Options

The next task in the migration process is running the version 8 Migration Utility. Before you begin that task, review the following command-line options for the Migration Utility because you may want to use some of them in your migration. In addition, your platform-specific Oracle documentation may contain more information about Migration Utility command line options for your platform.

CHECK_ONLY When TRUE, the Migration Utility performs space use calculations without performing a migration. When FALSE, the Migration Utility performs both space usage calculations and the migration. This command line option is mutually exclusive with NO_SPACE_CHECK. If no CHECK_ONLY command line option is specified, the Migration Utility sets it to FALSE by default.

DBNAME Specifies name of the database to migrate (DB_NAME in INIT.ORA).

MULTIPLIER Specifies the initial size of the version 8 i_file#_block# index relative to the version 7 i_file#_block# index. For example, MULTIPLIER=30 triples the initial size when the index is created. If no MULTIPLIER command line option is specified, the Migration Utility uses the i_file#_block# value of 15, creating an index for version 8 that is 1.5 times larger than the version 7 i_file#_block# index.

NEW_DBNAME Specifies a new name for the migrated database. The default name "DEFAULT" should not be used; choose a more meaningful name.

NLS_NCHAR Specifies the National Language Standard (NLS) NCHAR character set in props$ for the version 8 database, for example W52DEC or US7ASCII. If no NLS_NCHAR command line option is specified, the Migration Utility uses the version 7 database character set.

NO_SPACE_CHECK When TRUE, the Migration Utility does not perform a space usage check before the migration. When FALSE, the Migration Utility performs a space usage check before migration. This command line option is mutually exclusive with CHECK_ONLY. If no NO_SPACE_CHECK command line option is specified, the Migration Utility sets it to FALSE by default.

PFILE Specifies the name of the parameter file. If no PFILE command line option is specified, the Migration Utility uses the default INIT.ORA file.

Note: On UNIX, the pathname must be enclosed by double-quotes masked by a backslash, for example:

```
mig PFILE=\"/tmp/mig/pfile\"
```

SPOOL Specifies the filename for the spool output

Note: On UNIX, the pathname must be enclosed by double-quotes masked by a backslash, for example:

```
mig SPOOL=\"/tmp/mig/spool\"
```

Migrate the Version 7 Source Database

Complete the steps in the following sections, to migrate a version 7 source database to version 8 using the Migration Utility.

Migration Steps in the Version 7 Environment

Complete the following migration steps in the version 7 environment:

1. Review the README.doc file.
2. Make sure you have DBA privileges, which are required to run the version 8 Migration Utility.
3. Refer to your platform-specific documentation to verify the system settings that control the placement of new version 8 database objects you will be creating.

   The following examples illustrate platform-specific variable settings:

   - on a UNIX system, the TWO_TASK environment variable must be unset
   - on an NT system, TWO_TASK must remain set
   - on VMS, the ORA_DFLT_HOSTSTR logical variable must be unset
4. Make sure no other DBA (CONNECT INTERNAL) with RESTRICTED SESSION privilege connects to the database while the Migration Utility is running. “Normal” users should not connect to the database during migration.
5. Make sure the Oracle ORA_NLS33 variable is set correctly.

   For example, on UNIX, with the NLS file installed in the default location ($ORACLE_HOME/migrate/nls/admin/data), you must set ORA_NLS33 to $ORACLE_HOME/migrate/nls/admin/data.
6. If your version 7 sql.bsq file was customized, make the same changes to the migrate.bsq file in the $ORACLE_HOME/dbs directory. Otherwise, move on to Step 7.

   During migration, migrate.bsq creates a number of objects normally created by sql.bsq, but it creates them under the Migrate schema. Later in the migration process, the ownership of these objects is changed to SYS.

7. Shutdown the version 7 database cleanly using the SHUTDOWN NORMAL or SHUTDOWN IMMEDIATE command; *do not use* SHUTDOWN ABORT. The version 7 source database must be shut down cleanly; therefore, no redo information or uncommitted transactions can remain.

   Note: If you do not shut down the version 7 database before migration starts, the Migration Utility will stop and display an error message.

8. Run the version 8 Migration Utility. You can run the Migration Utility by entering the Migration Utility command at the system prompt, or by using the Installation Utility.

   To start the Migration Utility at the system prompt, use the command shown in your platform-specific documentation.

   For example, on a UNIX system, enter the **mig** command. Enter **mig** alone to run the Migration Utility with a default set of options, or enter **mig** followed by one or more selected options.

   See Also: "Review Version 8 Migration Utility Command Line Options" on page 3-9 for information about command line options.

   To run the Migration Utility from the Installation Utility, complete the following steps:

   a. Run the version 8 Installation Utility. Use the correct command for your platform; for example, use **orainst** on a UNIX system, or use **oracleins** on VMS.

      See Also: Your platform-specific *Installation Guide* for information.

   b. On the "Install Type" screen, select "Default or Customer Install".

   c. On the "Select the Installer Activity" screen, select "Install, Upgrade, or De-Install Software".

   d. On the "Select the Installer Option" screen, select "Migrate from ORACLE7 to ORACLE8".

   e. On the "Installation Options: Home Locator" screen, enter the complete path to the version 7 $ORACLE_HOME location.

   f. On the "Logging and Status" screen, confirm or change the log file location.

g. On the “Select an ORACLE7 to ORACLE8 Migration Action” screen, select “Run Migration Utility”.

h. On the “ORACLE_SID” screen, enter the ORACLE_SID for the database you are migrating.

i. On the “Software Asset Manager” screen, select “Migration Utility: Oracle7 to Oracle8 8.0.*X*” (where *X* is the current release number) and choose “Install”.

j. On the “Migrate SID” screen, enter the ORACLE_SID of the database you are migrating.

The Installation Utility runs the Migration Utility.

9. Check the results after running the Migration Utility. The Migration Utility generates informational messages and echoes its progress as it runs the migrate.bsq script (see Appendix A, “Migration Utility Messages” for more information).

The Migration Utility creates a convert file that contains the information of the version 7 control file. Later in the migration process, the convert file is used by ALTER DATABASE CONVERT to create a new control file in version 8.

The name and location of the convert file are platform-specific. For example, on a UNIX platform, the default location is $ORACLE_HOME/dbs in the version 7 environment, and the default filename in this directory is conv*SID*.dbf, where *SID* is your version 7 instance ID.

WARNING: Do not open the version 7 database, which was shut down by the version 8 Migration Utility. To ensure datafile version integrity, the SCNs in the dictionary, the convert file, and file header must all be consistent when the database is converted to version 8. If the version 7 database is opened after running the Migration Utility, the SCN check will fail when the database is converted to version 8, and an ORA-1211 error will be displayed, stating “Oracle7 datafile is not from migration to Oracle8.” Therefore, if the version 7 database is opened, you must rerun the Migration Utility, starting at Step 7.

Preserve the Version 7 Source Database

After successfully running the Migration Utility, perform a cold backup of the version 7 database. This backup serves the following purposes:

- If you wish to return to the version 7 database after executing the ALTER DATABASE CONVERT command on version 8, you can restore the backup and start the version 7 database.
- It can be used as the first version 8 backup for a version 8 recovery.
- If an error occurs at version 8 database convert time (ALTER DATABASE CONVERT or ALTER DATABASE OPEN RESETLOGS), you can restore this backup, fix the problem(s), and continue the conversion process. However, if you restore a backup performed before you ran the Migration Utility, you must rerun the Migration Utility.

> **See Also:** *Oracle8 Backup and Recovery Guide* for information about performing backups.

In addition, perform a backup of the entire version 7 software distribution, including the version 7 home directory ($ORACLE_HOME on UNIX or ORA_ROOT on VMS). Make sure the backup includes the following:

- all of the subdirectories
- control files
- datafiles and online redo log files (in case any datafiles in the version 7 database are lost or unreadable), although these files should not contain any outstanding redo information.
- parameter files
- convert file
- scripts that create objects in the version 7 database
- scripts that could restore the original database, if necessary

Migration Steps in the Version 8 Environment

Complete the following migration steps in the version 8 environment:

1. Install the version 8 software with the version 8 Installation Utility. Choose the Install/Upgrade option to avoid the creation of a new database. The procedure for installing the version 8 database is platform-specific.

   See Also: Your platform-specific version 8 installation documents, and read the additions or modifications in the README.DOC file included with version 8.

2. Make sure that the following environment variables point to the version 8 executables:

   - ORACLE_HOME
   - PATH
   - ORACLE_PATH
   - LD_LIBRARY_PATH

   Note: If ORACLE_HOME points to the version 7 executable, an ORA-223 error is displayed when you run the ALTER DATABASE CONVERT command, stating “conversion data file is invalid or incorrect version”.

3. Make sure the operating system running the version 8 DBMS is the same as the operating system used for running the version 7 DBMS and the version 8 Migration Utility.

4. Adjust the INIT.ORA file for version 8.

   Certain version 7 initialization parameters are obsolete in version 8. Remove all obsolete parameters from any initialization parameter file that starts a version 8 instance. Obsolete parameters may cause errors if used with a version 8 database. Also, alter any parameter whose syntax has changed in version 8; refer to Appendix C, “Version 8 INIT.ORA Changes” for lists of new, changed, and obsolete parameters.

5. Set the COMPATIBLE parameter in your INIT.ORA file.

   See Also: "COMPATIBLE Parameter" on page C-2 for information.

6. Either remove or rename the database's control files, or use the CONTROL_FILES parameter in the INIT.ORA file to specify new control file names.

   The ALTER DATABASE CONVERT command automatically creates new control files. If you do not use the CONTROL_FILES parameter, this command uses the control file names of your pre-migration database and returns an error if the control files already exist. Therefore, in this case, you must remove or rename the control file(s).

   However, if you use the CONTROL_FILES parameter in the INIT.ORA file, the ALTER DATABASE CONVERT command creates the new control file(s) with the names you specify, and you do not need to remove the old control files.

   Note: CONTROL_FILES specifies one or more names of control files, separated by commas. Oracle Corporation recommends using multiple files on different devices or mirroring the file at the operating system level. See the *Oracle8 Administrator's Guide* for more information.

7. If you installed the version 8 executables in a separate $ORACLE_HOME directory, move or copy the convert file from the version 7 environment to the version 8 environment. Skip to Step 9 if you are using the same $ORACLE_HOME.

   On UNIX, the convert file, conv*SID*.dbf (where *SID* is the version 8 instance ID), should reside in $ORACLE_HOME/dbs. If the version 8 instance ID is different from the version 7 instance ID, rename the conv*SID*.dbf file to match the version 8 instance ID.

8. If you installed the version 8 executables in a separate $ORACLE_HOME directory and you have a password file, move or copy the password file from the version 7 to the version 8 environment.

   The name and location of the password file is platform-specific; for example, on UNIX platforms, the default password file is $ORACLE_HOME/dbs/orapw*SID*, but on Windows NT, the default password file is $ORACLE_HOME/database/pwd*SID*.ora. In both cases, *SID* is your Oracle instance ID.

9. Make sure all online data files are accessible and in the correct directories. If you are using a raw disk, log files also must be accessible.

10. Start Server Manager by entering the following:

```
SVRMGRL
```

11. Connect to the version 8 database instance:

```
SVRMGR> CONNECT INTERNAL;
```

12. Start a version 8 database instance without mounting the new version 8 database:

```
SVRMGR> STARTUP NOMOUNT;
```

Note: Starting the database instance in any other mode might corrupt the database.

---

13. Create a new version 8 database control file and convert the file headers of all online tablespaces to version 8 format:

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

Successful execution of this command is the "point of no return" to version 7 for this database. However, if necessary, you can restore the version 7 database from backups.

Note: Control files are considerably larger in version 8 than in version 7. Control files in the tens of kilobytes size range in version 7 could be expanded into the range of tens of megabytes automatically during migration to version 8. This size increase could be important if a control file is on a raw device or if its available disk space is restricted.

If error(s) occur during this step, correct the condition(s) that caused the error(s) and rerun the Migration Utility. Restart at Step 1 on page 3-11; otherwise restore the backup you performed after you ran the Migration Utility.

14. Open the version 8 database with the following command:

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

When the version 8 database is opened, all rollback segments that are online are converted to the new version 8 format.

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

```
SVRMGR> SPOOL CATOUT.LOG
```

16. Run the version 8 database conversion script, CAT8000.SQL:

```
SVRMGR> @CAT8000.SQL
```

The CAT8000.SQL script creates and alters certain system tables and drops the "MIGRATE" user. It also runs the CATALOG.SQL and CATPROC.SQL scripts, which create the system catalog views and all the necessary packages for using PL/SQL.

> **See Also:** *Oracle8 Reference* for a complete list and descriptions of available scripts, if you want to create additional data dictionary structures.

17. If the Oracle system has Advanced Replication installed, run the CATREP8M.SQL catalog script:

```
SVRMGR> @CATREP8M.SQL
```

The CATREP8M.SQL script automatically runs the CATREP.SQL script.

18. If the Oracle system has Parallel Server installed, run the following catalog script supplied with your new release:

```
SVRMGR> @CATPARR.SQL
```

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

```
SVRMGR> SPOOL OFF
```

Then, check the spool file and verify that *every* package and procedure compiled successfully. You named the spool file in Step 15; the suggested name was CATOUT.LOG. Correct any problems you find in this file.

20. Run SHUTDOWN on the version 8 database:

```
SVRMGR> SHUTDOWN NORMAL
```

> **WARNING: Use SHUTDOWN NORMAL or SHUTDOWN IMMEDIATE. Do not use SHUTDOWN ABORT.**

Executing this clean shutdown flushes all caches, clears buffers, and performs other DBMS housekeeping activities. These measures are an important final step to ensure the integrity and consistency of the newly migrated version 8 database.

21. Perform full table scans of the entire database.

> **Note:** This step is recommended but is not required.

The migration process changed your table definitions to the new format, but existing ROWIDs are not changed until they are accessed. You may want to complete this action all at once, instead of waiting for users to access ROWIDs in the tables. An easy way to perform full table scans of the entire database is to conduct a full export of the database with FILE=/dev/null. This export does not produce an export file.

> **Note:** If you have any stored ROWIDs in your tables for use by your applications, you must convert these to version 8 format. See Chapter 7, “Migration Issues for the Version 8 ROWIDs”, for more information.

22. Complete the procedures described in Chapter 5, “After Migrating the Database”.

Errors During Migration

Errors may be caused by the following actions or omissions:

- performing a migration step out of order
- failing to fulfill the prerequisites for migration
- encountering an occasional conversion irregularity

> **See Also:** Appendix A, "Migration Utility Messages" and *Oracle8 Error Messages* for information about errors during migration and about corrective action for each error.

Abandoning the Migration

You can run the version 8 Migration Utility multiple times and still return to the version 7 database. However, running the Migration Utility automatically eliminates the version 7 database catalog views. Therefore, to return to the version 7 database after running the Migration Utility, you must run the version 7 CATALOG.SQL script to restore the version 7 database catalog views.

To abandon the migration, you generally must restore the version 7 database by completing the following steps:

1. Start the version 7 database using Server Manager:

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

2. Drop the user "MIGRATE":

```
SVRMGR> DROP USER MIGRATE CASCADE
```

3. Rerun CATALOG.SQL and CATPROC.SQL:

```
SVRMGR> @CATALOG.SQL
SVRMGR> @CATPROC.SQL
```

4. If Server Manager is installed, run CATSVRMG.SQL:

```
SVRMGR> @CATSVRMG.SQL
```

5. If Parallel Server is installed, run CATPARR.SQL

```
SVRMGR> @CATPARR.SQL
```

6. If Advanced Replication is installed, run CATREP.SQL:

```
SVRMGR> @CATREP.SQL
```

Note: The version 8 Migration Utility upgrades release 7.1 and release 7.2 databases to release 7.3. If the original version 7 production database was release 7.1 or 7.2 and the migration is run but abandoned before the conversion to version 8, the version 7 database will be left with a dictionary that is release 7.3.

4

Migrating Using Export/Import

This chapter provides information about using Export / Import to migrate a database from version 7 to version 8.

The following topics are covered in this chapter:

- Basics of Export / Import
- Additional Export / Import Information Sources
- Additional Options
- Migrate the Pre-Version 8 Source Database Using Export / Import

Basics of Export/Import

To migrate a database using Export/Import, complete the following three basic steps:

1. Export the data from the database you are migrating (the source database). The export physically copies the data to the export dump file.
2. Create the version 8 database into which you will import the exported data (the target database).
3. Import the exported data into the new version 8 target database.

> **See Also:** "Choose a Migration Method" on page 2-3 and *Oracle8 Utilities* for information that can help you to evaluate the choice of Export/Import for migration.

Export Requirements

To migrate or upgrade a database, use the Export Utility shipped with the version or release of the *source* database. After the export, the Import Utility can copy the data from the export dump file into the target database. The target database must be created and operational before the Import Utility can migrate the exported data into the target database.

Import Requirements

To migrate, upgrade, or downgrade a database, use the Import Utility shipped with the version or release of the *target* database.

Additional Export/Import Information Sources

If you plan to migrate a database by using Export/Import, first study *Oracle8 Utilities* for detailed information about using the Export/Import utilities.

Additional Options

Refer to the following sources if you have additional options installed.

> **See Also:** The *Trusted Oracle7 Administrator's Guide* for information about migrating the features of the trusted Oracle database if you are exporting from, or importing to, a Trusted Oracle database (version 7).

> **See Also:** *Oracle8 Replication*, Appendix B, "Migration and Compatibility", if you are migrating a pre-version 8 database system that has Advanced Replication installed

Migrate the Pre-Version 8 Source Database Using Export/Import

> **Note:** If the source Oracle database is earlier than Version 6, first upgrade the source database to at least version 6 before proceeding with export.

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

1. Export the source database using the Export Utility shipped with the source database. See the source database's server Utilities documents for information about using the Export Utility on the source database. Both version 7 and version 6 database exports can be imported into version 8.

   > **Note:** To ensure a consistent export, make sure the source database is not available for updates during and after the export. If the source database will be available to users for updates after the export, then, prior to making the source database available, put procedures in place to copy the changes made in the source database to the version 8 target database after the import is complete.

2. Install the version 8 software. Installation steps for version 8 are specific to your platform, as detailed in your platform-specific version 8 documentation and in the README file included with version 8.

3. If the new version 8 database will have the same name as the existing source database, shut down the existing database before creating the new version 8 database.
4. Create the version 8 target database.

   See Also: The *Oracle8 Administrator's Guide* for information about creating a version 8 database.

5. Open the new version 8 target database by entering the following commands from Server Manager:

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

6. Pre-create tablespaces, users, and tables in the target database to improve space usage by changing storage parameters. When you pre-create tables using SQL*Plus or Server Manager, either run the database in the original database compatibility mode or make allowances for the specific data definition conversions that occur during import.

   Note: If the new version 8 database will be created on the same machine as the source database, and you do not want to overwrite the source database datafiles, you must pre-create the tablespaces and specify IGNORE=Y and DESTROY=N when you import.

7. Use the version 8 Import Utility to import the objects exported from the source database. Include the LOG parameter to save the informational and error messages from the import session to a file.

   See Also: *Oracle8 Utilities* for a complete description of the Import Utility.

8. After the migration, check the import log file for information about which import(s) of which object(s) completed successfully and which failed.

   See Also: *Oracle8 Utilities* and the version 8 server README file for error handling information.

9. Use further Import scenarios (see *Oracle8 Utilities*) or SQL scripts used to create the source objects to clean up incomplete imports (or possibly to start an entirely new import).
10. If changes are made to the source database after the export (Step 1 on page 4-3), make sure those changes are propagated to the version 8 database prior to making it available to users.
11. Complete the procedures described in Chapter 5, “After Migrating the Database”.

5

After Migrating the Database

This chapter guides you through the procedures to perform after you have completed a migration using either the Migration Utility or Export/Import. This chapter elaborates on “Step 6: Tune and Adjust the New Version 8 Production Database” on page 1-5.

The following post-migration steps are covered in this chapter:

- Back Up the Version 8 Database
- Check for Bad Date Constraints
- Rebuild Invalidated Bitmap Indexes
- Test the Database and Compare Results
- Tune the Target Database
- Add New Features as Appropriate
- Develop New Administrative Procedures as Needed

Back Up the Version 8 Database

Make sure you perform a complete backup of the newly migrated production database. This backup must be complete, including all datafiles, control files, online redo log files, parameter files, and SQL scripts that create objects in the new database. To accomplish a complete backup, a full database export or a cold backup is required, because a hot backup cannot afford full recoverability. This backup can be used as a return point, if necessary, in case subsequent steps adversely affect the newly migrated database.

> **See Also:** *Oracle8 Backup and Recovery Guide* and *Oracle8 Administrator's Guide* for details about backing up the database.

Note: Using the Migration Utility transforms the source database. Therefore, after migration, the source database ceases to exist except for the backup you created under "Preserve the Version 7 Source Database" on page 3-14. This backup also can serve as the first version 8 backup for a version 8 recovery of the newly migrated database.

Check for Bad Date Constraints

A bad date constraint involves invalid date manipulation. An invalid date manipulation is one that implicitly assumes the century in the date, causing problems at the year 2000. The UTLCONST.SQL script runs through all of the check constraints in the database and sets constraints as bad if they include any invalid date manipulation. This script selects all the bad constraints at the end.

To run the UTLCONST.SQL script, enter the following:

```
SVRMGRL> SPOOL UTLRESULT.LOG
SVRMGRL> @UTLCONST.SQL
SVRMGRL> SPOOL OFF
```

After you run the script, the UTLRESULT.LOG log file includes all the constraints that have invalid date constraints.

Note: UTLCONST.SQL does not correct bad constraints, but it does disable them. You should either drop the bad constraints or recreate them after you make the necessary changes.

Rebuild Invalidated Bitmap Indexes

During migration, some bitmap indexes may become invalidated. To find these indexes, use index views, such as ALL_INDEXES and DBA_INDEXES. Once you locate the invalidated bitmap indexes, rebuild them.

> **See Also:** *Oracle8 Tuning* and *Oracle8 Concepts* for more information about using bitmap indexes.

Test the Database and Compare Results

Test the version 8 database using the testing plan you developed in "Develop a Testing Plan" on page 2-12. Compare the results of the test with the results obtained with the original database and make certain the same, or better, results are achieved.

Generally, the performance of the migrated version 8 database should be as good as, or better than, the performance of the source database. If you notice any decline in database performance with version 8, make sure the initialization parameters are set properly because improperly set initialization parameters can hurt performance.

> **See Also:** *Oracle8 Tuning* for database tuning information.

Besides testing the migrated test database, you may run the original source database and the newly migrated version 8 database concurrently for a time, to make sure that the new database is configured correctly and functioning properly. To run both databases concurrently, you must modify the Oracle configuration. Of course, synchronized updates would need to be made to both databases, ideally as part of an automated process.

Tune the Target Database

If you want to improve the performance of the migrated database, tune the database.

> **See Also:** *Oracle8 Tuning* for tuning information.

Most of the actions normally used to tune version 7 databases and related applications either have the same effect on or are unnecessary for version 8 databases. Therefore, actions you used to tune your source database and applications should not impair the performance of the migrated version 8 database.

Add New Features as Appropriate

Getting to Know Oracle8 and the Oracle8 Enterprise Edition describes many of the new features available in version 8. Determine which of these new features can benefit the migrated database and applications; then, develop a plan for using these features.

However, it is not necessary to make any immediate changes to begin using your migrated version 8 database. You may prefer to introduce these enhancements into your database and corresponding applications gradually.

Chapter 6, "Upgrading Version 7 Applications", describes ways to enhance your applications so that you can take advantage of the new version 8 features. However, before you implement new version 8 features, test your applications and successfully run them with the migrated test database (possibly running in special version 7 compatibility mode).

Develop New Administrative Procedures as Needed

After familiarizing yourself with the version 8 features, review your database administration scripts and procedures to determine whether any changes are necessary.

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

6

Upgrading Version 7 Applications

This chapter provides information about using Oracle tools and applications with version 8. You do not need to modify existing (version 7) applications that do not use new version 8 features. Existing applications should achieve the same, or enhanced, functionality on a version 8 database.

The following topics are covered in this chapter:

- Upgrading Oracle Applications to Version 8
- Upgrading Precompiler and OCI Applications
- Upgrading LONGs to LOBs
- Upgrading Version 7 Forms or Developer/2000 Applications
- Data Dictionary Views Update
- Upgrading SQL*Plus Scripts
- PL/SQL V2 Compatibility Mode
- SQL*Net or Net8
- Backup Management: EBU and Recovery Manager
- Dictionary Protection
- Password Management
- Export/Import Usage, Partitioned Objects
- Migration and Compatibility Issues for Thread Safety, OCI
- Upgrade and Compatibility Issues for Standby Database
- Compatibility Issues for Export/Import
- NCHAR and NLS Use

Upgrading Oracle Applications to Version 8

The following version 8 features aid in the process of upgrading applications:

- Both SQL*Net V.2 and Net8 work with various Oracle versions and releases. Thus, version 7 and version 8 databases can communicate by using SQL*Net V2 and Net8.

  Note: SQL*Net V1, however, used a different network addressing scheme and *cannot* be used with version 8.

- For application creation and management, the programming interface (such as the Oracle precompilers and Oracle Call Interface or OCI) remains unchanged between different versions of Oracle. For example, you can use SQL*Net or Net8, or you can relink applications designed for an earlier release database to run them with a version 8 release, or with version 7.
- Backward compatibility generally is built in to accommodate small incompatibilities between different Oracle releases.
- Many new features and enhancements are automatically available after migration. Some of these new features may provide improved performance.

XA Calls: Incompatibility with Release 7.1 XA Calls

The version 8 database does not support version 7, release 7.1.6 XA calls, but it does support release 7.2 and 7.3 calls. Therefore, after migrating a version 7, release 7.1, database to version 8, relink associated Tuxedo applications (and any other associated applications that use XA calls) with the version 8 XA libraries.

Upgrading Precompiler and OCI Applications

Before you migrate your Oracle database from version 7 to version 8, upgrade any OCI and Precompiler applications that you plan to use with your version 8 databases. Then, you can test these applications on a sample version 8 database before migrating your production database.

The effort required to upgrade these applications depends on the degree to which you want to take advantage of the programmatic interfaces and version 8. 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 version 8 database functionality

The following sections, "Upgrading Precompiler Applications" and "Upgrading OCI Applications: Enabling Constraints", include the specific steps required to migrate Precompiler and OCI applications.

Upgrading Precompiler Applications

Complete the following steps to use your existing precompiler applications with a version 8 database:

1. If you want to take advantage of the new features offered by version 8, an existing application may need to be recoded, or new applications written, to reflect the differences between version 7 and version 8. If you do not want to take advantage of version 8 features with an existing application, skip Step 1 and move on to Step 2.
2. Relink the application with the version 8 runtime library, SQLLIB, which is included with the precompiler. This step is the only required step to use your applications with version 8.

Note: To run an existing pre-compiler application against a version 8 database, you do not need to re-precompile, nor recompile, the application.

Simplified Upgrading of Existing Applications

Applications written to work with version 7 precompilers, such as Pro*C 2.2 have a very smooth upgrade path to version 8 precompilers, such as Pro*C 3.0, due to extensive interoperability. Version 7 precompiler clients work with a version 8 server, and version 8 precompiler clients can work with a version 7 server. Specifically, the following list outlines precompiler / server compatibility:

- Applications using version 7 Pro*C 2.2 work unchanged against a version 8 server (all SQLLIB function calls work against version 8 servers).
- Applications using version 8 Pro*C 3.0 can work against a version 7 server if they *do not use any object capabilities* of the precompiler or server.
- Version 7 and version 8 SQLLIB functions can be mixed in the same application and the same transaction.

The following three alternative upgrade paths are available for existing version 7 precompiler applications, without requiring recompilation or re-precompilation:

- Retain version 7 client application without any modification.
  It will continue to work against a new version 8 server.
- Upgrade to version 8 client but do not modify the application. To upgrade a version 7 client to a version 8 client, you only need to relink the new version of OCILIB. Relinked version 7 OCI applications work unchanged against a version 8 server.
- Upgrade to version 8 and modify the application to leverage new version 8 precompiler capabilities for performance and scalability benefits, or to use any object capabilities of the version 8 server. Existing (version 7) applications must be recoded using the new version 8 SQLLIB calls, and must be relinked with the new SQLLIB, to run against a version 8 server. This path is required for any application to use any object capability of the version 8 server.

Migrating Ada Support in Version 8

The Pro*ADA product was officially de-supported by Oracle with version 7, release 7.3.

Migrate Pro*ADA to SQL*Module for Ada 8.0.3, which has a number of new features. However, SQL*Module for ADA 8.0.3 does not provide object support.

PL/SQL Backward Compatibility and Precompilers

PLSQL_V2_COMPATIBILITY backward compatibility behavior (see "PLSQL_V2_COMPATIBILITY Flag" on page 6-8) is available in the precompiler environment by setting the precompiler command line option, DBMS, as follows:

```
... DBMS=V7
```

Precompilers' Potential NCHAR/CHAR Migration or Compatibility Issues

NCHAR columns in version 7 are stored as CHAR. Their migration to version 8 requires special attention.

Upgrading OCI Applications: Enabling Constraints

OCI libraries are shipped with all version 8 releases. You can use existing version 7 OCI applications with a version 8 server, and you can ensure that constraints present in version 7 applications will be properly enabled when they are run on a version 8 database.

To ensure that constraints will be properly enabled requires that you relink the applications with the runtime OCI library for version 8, OCILIB, using one of the following version 8 deferred-mode settings:

- Non-deferred mode, to maintain existing (version 7) performance levels
- Deferred mode linking, to improve performance of applications

---

Note: With deferred linking, bind and define errors may not be reported to the application immediately after bind and define operations. Instead, these errors may be reported later, at the time of a DESCRIBE, EXECUTE, or FETCH call.

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

OCI Application Link Line

For OCI applications, the version 8 link line differs from the version 7 link line. See the $ORACLE_HOME/rdbms/demo/demo_rdbms.mk file for examples about using the version 8 link line as a version 8 OCI application is compiled.

Applications Using Version 6 OCI Libraries

The version 6 OCI library is not supported against the version 8 database. Therefore, applications that use the Oracle6 OCI library cannot be run against a version 8 database.

Upgrading LONGs to LOBs

Version 8 does not provide a facility for a direct migration of LONGs to LOBs. However, you can use the following indirect path for upgrading LONGs to LOBs:

1. Write the data in the LONG RAW to a server side file.
2. Use the version 8 command CREATE DIRECTORY to point to the directory where the file resides.
3. Use the version 8 command OCILobLoadFromFile or DBMS_LOB.LOADFROMFILE to populate the BLOB with the data in the file.

If the LONG RAW is not too big, you also can read the LONG RAW into a buffer and call OCILobWrite or DBMS_LOB.WRITE to write the LONG RAW data to the BLOB.

Upgrading Version 7 Forms or Developer/2000 Applications

Forms applications run the same on version 7 and version 8. However, review the new features described in *Getting to Know Oracle8 and the Oracle8 Enterprise Edition* to determine whether any of the new version 8 features would be beneficial to your applications or might otherwise affect them. Information about the ways in which the version 8 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.*

Data Dictionary Views Update

Certain data dictionary views maintained in version 7 for backward compatibility to Oracle Version 5 and Version 6, created in the files CATALOG5.SQL and CATALOG6.SQL, are obsolete with version 8. Remove all references to these data dictionary views from your database tools and applications.

Upgrading SQL*Plus Scripts

To use SQL*Plus release 4.0, version 8, and PL/SQL Version 3 functionality, complete the following steps:

- Make the following changes to SQL*Plus 3.x scripts to convert them into SQL*Plus 4.0 scripts:
  - If a script contains the line SET COMPATIBILITY V7, change it to SET COMPATIBILITY V8, or remove the line so that the default version 8 setting is used.
  - Check any LOGIN.SQL file(s) and change any SET COMPATIBILITY V7 line found to SET COMPATIBILITY V8.
  - Refer to the *SQL*Plus User's Guide and Reference* to learn about new functionality in SQL*Plus release 4.0.
- To use new version 8 functionality, change existing SQL scripts to conform to version 8 syntax. Existing SQL scripts run unchanged on version 8, and require no modification, if they do not use new version 8 functionality

See Also: *Oracle8 SQL Reference* for more information about upgrading SQL scripts.

Note: No changes to PL/SQL procedures are required.

PL/SQL V2 Compatibility Mode

A PL/SQL V2 compatibility mode is available in PL/SQL V8.0.3. This mode is enabled by the PLSQL_V2_COMPATIBILITY flag.

You can set this flag in any one of the following three ways:

- Add the following line to the INIT.ORA file:

```
PLSQL_V2_COMPATIBILITY=TRUE
```

- Issue the following command:

```
ALTER SYSTEM SET PLSQL_V2_COMPATIBILITY = TRUE;
```

- Issue the following command:

```
ALTER SESSION SET PLSQL_V2_COMPATIBILITY = TRUE;
```

PLSQL_V2_COMPATIBILITY Flag

The PLSQL_V2_COMPATIBILITY flag provides compatibility between PL/SQL V8.03 and PL/SQL V2 in the following situations:

- The PL/SQL V2 compiler allows a record type or index table type to be referenced before its definition in the source. PL/SQL V8.0.3 strictly requires that the type definition precede reference to the type in the source. However, when you enable PL/SQL V2 compatibility mode, PL/SQL V8.0.3 behaves the same as PL/SQL V2 in regard to type definitions.
- The PL/SQL V2 compiler allows the following illegal syntax:

  ```
  return variable-expression
  ```

  This syntax is incorrect and should be changed to the following:

  ```
  return variable-type
  ```

  The PL/SQL V8.0.3 compiler issues an error when it encounters the illegal syntax. However, when you enable PL/SQL V2 compatibility mode, PL/SQL V8.0.3 behaves the same as PL/SQL V2 and does not issues an error.
- In PL/SQL V2 it is possible to modify/delete elements of an index table passed in as an IN parameter, as in the following example:

  ```
  function foo (x IN table_t) is
  begin
  x.delete(2);
  end;
  ```

  This use of an IN parameter is incorrect. PL/SQL V8.0.3 correctly enforces the read-only semantics of IN parameters and does not let index table methods modify index tables passed in as IN parameters. However, when you enable PL/SQL V2 compatibility mode, PL/SQL V8.0.3 behaves the same as PL/SQL V2 and allows the parameter.
- PL/SQL V2 allows the passing (as an OUT parameter) of fields of IN parameters that are records, but PL/SQL V8.0.3 does not allow this type of passing. However, when you enable PL/SQL V2 compatibility mode, PL/SQL V8.0.3 behaves the same as PL/SQL V2 and allows this type of passing.
- The PL/SQL V2 compiler permits fields of OUT parameters that are record variables to be used in expression contexts (for example, in a dot-qualified name on the right-hand side of an assignment statement).

This use of OUT parameters should not be permitted. PL/SQL V8.0.3 does not permit OUT parameters to be used in expression contexts. However, when you enable PL/SQL V2 compatibility mode, PL/SQL V8.0.3 behaves the same as PL/SQL V2 in this regard.

- PL/SQL V2 allows OUT parameters in the FROM clause of a SELECT list. PL/SQL V8.0.3 does not allow this use of OUT parameters. However, when you enable PL/SQL V2 compatibility mode, PL/SQL V8.0.3 behaves the same the PL/SQL V2 in this regard.

Keyword Behavior Differences: Version 7 vs. Version 8

The following keywords or types included in both version 7 and version 8 produce slightly different error message identifiers when used as a function name in a SELECT list:

Keyword	Version 8 Behavior	Version 7 Behavior
CHARACTER, COMMIT, DEC, FALSE, INT, NUMERIC, REAL, SAVEPOINT, TRUE	Generates errors: ORA-06550 and PLS-00222	Generates errors: ORA-06552 and PLS-222

New Keywords or Types Behavior Differences: Version 7 vs. Version 8

The following words are newly defined in version 8 as keywords or types:

BFILE	CLOB	NCLOB	SYS_OP_NTCIMG
BLOB	DEREF	NVARCHAR2	VALUE
CAST	NCHAR	REF	

In version 8, each may be used as a function name in a SELECT list, however, if it is qualified with a schema, as *schema.function*, as in the following example:

```
select scott.true() ...
```

In version 8, if a schema qualification is missing, these words generate an error, while, in version 7, their unqualified use functions without error.

SQL*Net or Net8

Version 7 and version 8 releases can use SQL*Net, Version 2 or Net8; SQL*Net V1, however, used a different network addressing scheme and *cannot* be used with version 8. Therefore, the following requirements apply to migrated applications:

- Both the client and server must run SQL*Net, Version 2 or Net8.
- The Multi-Threaded Server requires SQL*Net, Version 2 or Net8 on the server. Therefore, to connect using the Multi-Threaded Server, you also must use SQL*Net, Version 2 or Net8 on the client.

Upgrading SQL*Net V1 to SQL*Net V2

Make the following changes to upgrade from SQL*Net V1 to V2:

- Install SQL*Net, Version 2 or Net8.
- Re-create each connect string as the next version's connect descriptor. SQL*Net Version 2 uses the syntax outlined in the *SQL*Net V2.0 Administrator's Guide* or *Oracle Net8 Administrator's Guide*.
- Relink any precompiler programs and Oracle executables that you want to use with SQL*Net Version 2, including SQLplus and SQLforms.

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

Version 7 Net2 Clients and Connection Manager

Existing version 7 OCI-Net2 clients can use the connection manager without relinking. The only requirement is that the client's connect string must go through the CMAN.

Thin-client JDBC is the same as a version 7-Net2 client, as are all current clients that do not relink.

Other version 8 benefits also are available to existing clients without relinking, including database link concentration, reduced SQL statement memory, reduced overall shared pool consumption, serially reusable SQL for SQL and PL/SQL, scalable cursor authorization, PL/SQL constant pool paging, more efficient SQL*Plus-PL/SQL interaction, improved parse concurrency, faster array inserts, faster table scans, bulk SQL from PL/SQL, faster character set conversion, and faster multibyte processing.

Net8 Features Available to Relinked Version 7 Clients

Version 7 OCI/UPI clients who relink with Net8 can use connection pooling and OSS authentication. No code changes are necessary to gain this functionality. The same applies to precompilers and application tools that you relink.

Version 8 Net8 Clients

Applications that are recoded to use version 8 OCI can implement all version 8 features, including the following:

- application failover
- transparent data prefetch
- piggybacked commit and cancel, which reduces network roundtrips and server footprint
- full dynamic statement execution in one roundtrip (as in SQL*Plus `select * from emp`)
- faster login, which eliminates 3 roundtrips when NLS_LANG is set
- new object table API, object cache
- security toolkit API (encryption, signatures)
- warnings (including password expiration, failover)
- server scalability due to client-side type conversion
- NCHAR
- LOB
- longer CHAR/VARCHAR
- DML returning values
- separation of server, user, and transaction handles, for simplifying middle-tier implementations

---

Note: The only client program fully converted to version 8 OCI is SQL*Plus.

Version 7 clients also can make selective use of version 8 OCI, mixing version 7 and version 8 calls. The degree of functionality added depends on which calls are used. Utility program such as SQL*Loader and Export/Import convert a subset of calls, primarily for object table access. The encryption API and password reset calls are independently usable as well. To enable failover, prefetch, piggyback commit/cancel, or client-side conversions, use version 8 OCI for all phases of the statements being processed.

Backup Management: EBU and Recovery Manager

EBU and Recovery Manager are client-side utilities for managing Oracle database backups. However, for managing version 8 database backups, you must use Recovery Manager. You cannot use EBU with version 8.

Both EBU and Recovery Manager use the Media Management Language (MML) to talk to third party storage subsystems, such as Legato or EMC.

Investments in tape subsystem management modules for EBU and version 7 should be reusable under Recovery Manager and version 8. However, backup volume formats are not reusable. You need to write new backups to the storage subsystem under version 8 because Recovery Manager produces a different format, and backups from version 7 generally are not useful for version 8 restores.

Note: The scripting language for Recovery Manager is completely different from the scripting language for EBU.

Dictionary Protection

Using tablespace-relative data block addresses (DBAs) would be expected to cause problems in any applications that:

- create user tables in the SYS schema, and
- access them using the *'ANY'* privileges.

Creating and accessing user tables in SYS schema is not secure. Therefore, applications are expected to move the objects to a different schema. Use the O7_DICTIONARY_ACCESSIBILITY switch for temporary compatibility, but this switch is only for interim use.

Applications should not attempt to connect to user SYS without the SYSDBA option. Instead of connecting to the user SYS and sharing the password, grant SYSDBA privilege to the normal user, who will connect to the database as SYSDBA to connect to SYS schema. Refer to *Oracle7 Server Administrator's Guide* for details about the SYSDBA privilege.

In some cases, the user with 'ANY' privilege uses the DML on the dictionary. For example, the user with DELETE ANY TABLE uses the delete statement to purge the audit records in the aud$ table. In such cases, the users can grant the privileges on the objects to the users to do the specific task.

Ideally, a migration script will be available for updating the password column of the user SYS in the dictionary to NULL.

Password Management

Make the following changes to a version 7 (or earlier) application to enable it to work with version 8 password management:

- Use the version 8 OCI call, *OCISessionBegin()*, to connect to the server. If the server returns SUCCESS_WITH_INFO, check to see if this is a password expiration error. If the password has expired but is still within the grace period, signal a warning to the user and use *OCIChangePassword()* call to change the user's password (the password change call is optional but recommended).
- If the password has expired and the error is returned, use the version 8 OCI call, *OCIChangePassword()*, to change the user's password.

If you do not make these changes to version 7 applications, one of the Oracle tools, such as SQL*Plus or SQL Server Manager, will be required to allow the password change after a user's account expires.

However, this version 8 password management feature is off by default. If a version 8 server system does not implement the password-expiration feature, no change is required to version 7 clients for password management. The DEFAULT profile sets all the parameters to UNLIMITED, and sets the password complexity check routine to NULL.

The password verification routine is exported / imported along with its profile definition. The user's history table can also be imported / exported in version 8, release 8.0.2 and higher.

Version 7 or Lower Client with Version 8 Server

Version 7 clients use version 7 OCI calls to connect to the server; therefore, version 8 password expiration cannot be detected. However, other features of version 8 password management, work for version 7 clients. Full version 8 password management, including password expiration handling, can operate in version 7 clients after you make the minor change of replacing their version 7 logon call with the version 8 logon call.

Version 8 Client with Version 7 or Lower Server

A version 8 client can be coded to work with version 7 or lower servers. An example of the code for such version 8 clients follows:

```
OCISessionBegin(...) /* call v8 logon OCI call */
if (SUCCESS_WITH_INFO) then
{ /* Check for password expiration and take appropriate action*/
...
OCIChangePassword(...);
...
}
```

Export/Import Usage, Partitioned Objects

The version 8 export dump file format differs from the version 7 format. Consequently, dump files that have been generated by version 8 Export cannot be imported by version 7 Import into version 7 databases. However, the version 7 version of Export can be used, after running the CATEXP7.SQL script, to export a version 8 database to version 7. Partitioned tables are not exported by the version 7 version of Export. If you need to move a partitioned table to a version 7 database, it first must be reorganized into a non-partitioned table.

Migration and Compatibility Issues for Thread Safety, OCI

The ORLON and OLON calls are not supported in version 8. However, you still should use OLOG, even for single-threaded applications.

Note: The OLOG call is required for multi-threaded applications.

See Also: *Oracle Call Interface Programmer's Guide* for more information about Thread Safety and OCI.

Upgrade and Compatibility Issues for Standby Database

Standby Database operates only on version 7, release 7.3 or version 8. The primary and standby databases must be running the same version and release number of Oracle server, and they must be running on the same release of the operating system platform.

> **See Also:** Your platform-specific Oracle documentation.

To use Standby Database correctly, consider the following:

- The data files, log files, and control files of the primary and standby databases must exist separately on separate physical media. A single control file cannot serve both the primary and standby databases.
- Although not required, it is good administrative procedure to make the following features the same on the primary and standby databases:
  - database identification string
  - data files names
  - log files names
  - control files names

To migrate the Standby Database to version 8, perform the following steps:

1. Apply all redo logs created under version 7.
2. Make sure the primary database has been opened successfully under version 8.
3. Install version 8 software at the standby site.
4. Copy the version 8 control file and Datafile 1 to the standby site.
5. Make a new control file for the standby database from the primary database.

> **See Also:** *Oracle8 SQL Reference*, the *Oracle8 Administrator's Guide*, and *Oracle8 Parallel Server Concepts and Administration* for more information about Standby Database.

Compatibility Issues for Export/Import

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

The version 8 Import Utility can use dump files generated by either Direct Path Export or Conventional Path Export.

Pre-version 8 dump files are upward compatible with the version 8 Import Utility. Dump file data produced by current or prior versions / releases of Direct Path Export or by Conventional Path Export alike have the same upward compatibility with future Oracle server versions and releases.

Downward Compatibility Techniques and Limitations

The version 8 Export Utility makes dump files that are *not* downward compatible with pre-version 8 Import utilities. Their exported data *cannot* be imported by pre-version 8 Import utilities.

To export version 8 data to a version 7 (or earlier) database, the version 7 version of Export must be used, after the CATEXP7.SQL script has been run.

NCHAR and NLS Use

In version 8, the DBA can declare the use of the national character set (NCHAR) for specific columns, attributes, PL / SQL variables, parameters, and return results. Unless such an explicit declaration is made, use of NCHAR and NLS is, for the most part, invisible and has no effect on other version 8 features. An exception is that SELECT statements on the props$ or VALUE$ dictionary view may return the CHARACTER_SET_NAME column or the NLS_NCHAR_CHARACTERSET row.

Migration and NCHAR and NLS

The props$ dictionary table contains two rows that describe the character set(s) specified in the CREATE DATABASE statement. The row holding NAME='NLS_CHARACTERSET' has the database character set's name in VALUE$. The row holding NAME='NLS_NCHAR_CHARACTER SET' has the national character set's name in VALUE$.

NLS_DATABASE_PARAMETERS contains a new row that holds the NCHAR character set name specified in the CREATE DATABASE statement. There is a new parameter, NLS_NCHAR_CHARACTERSET.

Compared to release 7.3, various views contain the new column, CHARACTER_SET_NAME, whose value is:

```
DECODE(x$.CHARSETFORM,
       1, 'CHAR_CS',
       2, 'NCHAR_CS',
```

where x$ represents one of the base tables. The DATA_TYPE or COLTYPE column value of the view will not change to indicate the character set choice.

NCHAR and NLS Compatibility and Interoperability

A release 7.1 and higher clients can interact with a version 8 server that holds data in the national character set. If the output national character data pass through a bind or define handle, the OCI handles the conversion to the client's database character set invisibly. If the data is needed as an input bind value, and is used where only a national character set string is allowed, the SQL or PL/SQL code using the value should surround the use of the bind variable, which will be perceived as having the database character set, with a call to `CSCONVERT()` to convert it to the national character set. The client is restricted in this case to passing the data in the client's database character set, which may not have the complete repertoire of the national character set.

A version 6 or version 7 client can RPC to a subprogram with `ANY_CS` parameters, which will interpret their actual argument as having the server's database character set. A version 6 or version 7 client cannot RPC to a subprogram with a parameter declared as using the national character set, and cannot RPC to a function with a return result using the national character set. These disallowed cases will be caught at run-time, not at compile-time.

A version 6 or version 7 client using SQL code embedded in PL/SQL cannot use the `CSCONVERT()` function, or `CHR()` with a second argument, directly in the SQL statement because these do not exist in a version 6 or version 7 client's PL/SQL package STANDARD. The following two stored procedures can be created on the server to deal with this:

```
create function to_nchar_cs(t varchar2) return nchar varying
   is begin
      return csconvert(t, nchar_cs);
   end;
   create function to_char_cs(t nchar varying) return varchar2
   is begin
      return csconvert(t, char_cs);
   end;
```

7

Migration Issues for the Version 8 ROWIDs

Version 8 ROWIDs embody new internal and external formats that enable you to use two new version 8 features: partitioned tables and tablespace-relative data block addresses (DBAs).

> **See Also:** The *Oracle8 Application Developer's Guide* and *Oracle8 Concepts* for more information.

The following migration issues related to the new version 8 ROWIDs are covered in this chapter:

- Migrating Applications and Data
- DBMS_ROWID Package
- Conversion Procedure Examples
- Snapshot Refresh
- Pre-Version 8 Client Compatibility Issues
- ROWID-Related Migration Questions and Answers

Migrating Applications and Data

ROWIDs can be stored in columns of ROWID type and in columns of character type. However, stored version 7 ROWIDs become invalid after migration of the version 7 database to version 8. Therefore, version 7 ROWIDs must be converted to version 8.

Applications

Applications that do not attempt to assemble and disassemble ROWIDs manually do not need to be changed or recompiled because the new ROWIDs fit the current storage requirements for host variables.

Applications that attempt to manufacture or analyze the contents of ROWIDs have to use the new package, DBMS_ROWID, provided in version 8 to deal with the format and contents of the new version 8 ROWIDs. The package contains functions that extract the information that was available directly from a version 7 ROWID (including file and block address), plus the data object number.

Data

The columns that contain ROWID values (in ROWID format or in character format) must be migrated if they point to tables that were migrated to version 8. Otherwise, it will not be possible to retrieve any rows using their stored values. On the other hand, if the ROWID values stored in the version 8 tables still point to pre-version 8 tables, you do not need to migrate the columns.

Columns are migrated in two stages: definition migration and data migration.

The column definition is adjusted automatically during version 7 to version 8 dictionary migration. The maximum size of ROWID user columns is increased to the size of the extended disk ROWIDs, changing the LENGTH column of COL$ for ROWID columns from 6 to 10 bytes.

The data migration can be performed only *after* the system has been opened in version 8. You can migrate different tables at different times or multiple tables in parallel. Make sure the migration is done *before* the version 7 database file limit is exceeded, thereby guarding against the creation of ambiguous block addresses.

You can use existing ROWID refresh procedures that are available at your installation, or the version 8 DBMS_ROWID functionality, to migrate stored ROWIDs from version 7 format to version 8 format.

Data migration by the Migration Utility applies only to ROWIDs stored in a user-defined column. All system-stored ROWIDs (such as in indexes) remain valid after migration by the Migration Utility and do not require specific actions to be migrated.

Note: Importing a column containing ROWIDs should produce a message warning that special attention might be required to re-establish the validity of the ROWIDs. Special attention is necessary for *all* ROWIDs being imported. Thus, migration to version 8 by Export/Import requires special attention for *every* column containing ROWIDs (not just for user-defined columns).

DBMS_ROWID Package

The DBMS_ROWID PL/SQL package is provided with version 8 and contains the following functionality:

- Creation of ROWIDs in version 7 and version 8 format.
- Interpretation of version 7 and version 8 ROWIDs.
- Conversion between version 7 and version 8 ROWIDs.

Migration of the stored ROWIDs can be accomplished using conversion functions, as described in the following sections.

ROWID Conversion Types

You must specify the type of ROWID being converted, because the ROWID conversion functions perform the conversion differently depending on whether the ROWID is stored in the user column of ROWID type, or in the user column of (VAR)CHAR type.

For a column of ROWID type, the caller of the conversion procedures must pass the following value as a procedure parameter:

```
rowid_convert_internal constant integer := 0;
```

For a column of (VAR)CHAR type, the caller of the conversion procedures must pass the following value as a procedure parameter:

```
rowid_convert_external constant integer := 1;
```

ROWID Conversion Functions

The following functions perform the ROWID conversion:

- ROWID_TO_EXTENDED converts a ROWID from the version 7 (restricted) format to the version 8 (extended) format.
- ROWID_TO_RESTRICTED converts a ROWID from the version 8 (extended) to the version 7 (restricted) format.
- ROWID_VERIFY checks whether a given ROWID can be converted from version 7 format to version 8 format.

The following sections contain detailed information about ROWID_TO_EXTENDED and ROWID_VERIFY procedures.

ROWID_TO_EXTENDED Conversion Procedure

ROWID_TO_EXTENDED uses the following parameters:

- ROWID to be converted (in External Character format).
- Schema Name - schema name of the table which contains a row whose ROWID will be converted to the extended format.
- Table Name - table name of the table which contains a row whose ROWID will be converted to the extended format.
- Conversion Type - the type of ROWID being converted.

> **See Also:** "ROWID Conversion Types" on page 7-3 for more information.

ROWID_TO_EXTENDED returns a version 8 (extended) ROWID in External Character format. Its parameters have the following meaning:

- If the schema name and table name for the target table are not specified (NULL), ROWID_TO_EXTENDED attempts to fetch the page specified by the ROWID to be converted. It will treat the file number stored in this ROWID as the absolute file number, which can cause problems if the file has been dropped and its number has been reused prior to the migration. If the fetched page belongs to a valid table, the ROWID will be converted to an extended format using the Data Object ID of this table, but this conversion is very inefficient, and is only recommended as a last resort, when the target table is not known. The user still must know the correct table name when using the converted value.

- If the schema name and table name is given (a preferred approach), ROWID_TO_EXTENDED will verify SELECT authority on the table and convert the ROWID to an extended format using the Data Object Number of this table. There is no guarantee that the converted ROWID actually references a real row in this table, neither at the time of conversion, nor at the time when the ROWID is used.
- If NULL value is supplied for the ROWID, the procedure ignores the table specification and returns a **NULL** value.
- If a value of '0', or, more generally, '<n>0.<m>0.<p>0' is supplied for ROWID, the table name is ignored and a restricted ROWID of the form '00000000.0000.0000' is returned.
- If a version 8 ROWID is supplied, the data object in the ROWID is verified against the actual data object number (which depends on the table name specification). If these two numbers do not match, the "INVALID ROWID" error appears; otherwise, the original ROWID is returned.

ROWID_VERIFY

A ROWID verification procedure, ROWID_VERIFY, is provided with version 8. This procedure uses the same parameters as ROWID_TO_EXTENDED and returns 0 if the ROWID can be converted successfully to extended format; otherwise, it returns 1.

However, ROWID_VERIFY returns security violation errors, or an "object not found" error, if the user does not have SELECT authority on the underlying table, or if the table does not exist. ROWID_VERIFY can be used to identify bad ROWIDs prior to migration using the ROWID_TO_EXTENDED procedure, because it can identify all bad ROWIDs.

Conversion Procedure Examples

The following sections provide examples of conversion procedures.

Example 1

Assume a table SCOTT.T contains a column C of ROWID format. All these ROWIDs reference a single table, SCOTT.T1.

The values of column C can be converted to extended format using the following statement:

```
UPDATE SCOTT.T SET C = DBMS_ROWID.ROWID_TO_EXTENDED(C,'SCOTT','T1',0);
```

Example 2

In a more general situation, ROWIDs stored in column C may reference different tables, but the table name can be found based on the values of some other columns in the same row. For example, assume that the column TNAME of the table T contains a name of the table which is referenced by a ROWID from column C.

In this case, the values in column C can be converted to extended format using the following statement:

```
UPDATE SCOTT.T SET C = DBMS_ROWID.ROWID_TO_EXTENDED(C,'SCOTT',TNAME,0);
```

Example 3

You can use the ROWID_TO_EXTENDED function in the CREATE AS SELECT statement. This use may be desirable in some cases because conversion can increase the size of the user column of ROWID type (typically from 6 bytes to 10 bytes, although this depends on a specific port) which may create indirect rows.

In this case, CREATE AS SELECT may be a better choice than UPDATE:

```
CREATE TABLE SCOTT.TNEW (A, B, C) AS SELECT A, B,
DBMS_ROWID.ROWID_TO_EXTENDED(C, 'SCOTT','T1',0) FROM SCOTT.T;
```

Example 4

If the target table for ROWIDs stored in column C is not known, conversion can be accomplished using the following statement:

```
UPDATE SCOTT.T SET C = DBMS_ROWID.ROWID_TO_EXTENDED(C, NULL,NULL,0);
```

Example 5

The following SQL statement may be used to find bad ROWIDs prior to conversion:

```
SELECT ROWID,C FROM SCOTT.T WHERE DBMS_ROWID.ROWID_VERIFY(C,NULL,NULL,0)=1;
```

Snapshot Refresh

The version 8 ROWID format forces all ROWID snapshots to perform a complete refresh when both master and snapshot sites are upgraded to version 8.

> **See Also:** *Oracle8 Replication*, Appendix B, "Migration and Compatibility", for more information.

Pre-Version 8 Client Compatibility Issues

Pre-version 8 clients can access a version 8 database, and version 8 clients can access a pre-version 8 database. Binary and character values of the pseudo column ROWID and of columns of type ROWID that are returned by a pre-version 8 database to a version 8 database are always in version 7 (restricted) format because the pre-version 8 system cannot recognize the extended format ROWID.

The DBMS_ROWID package supplied with version 8 can be used for interpreting the contents of the version 7 ROWIDs and for creating the ROWIDs in version 7 format.

A pre-version 8 client accessing a version 8 database receives the ROWID in version 8 extended format. Therefore, the pre-version 8 client cannot interpret the contents of ROWIDs returned by the version 8 server.

Version 8 snapshot compatibility is restricted to version 7, release 7.1.4 and higher. Further, when a master site is upgraded, the version 8 upgrade script invalidates the logs so that snapshots are forced to do a complete refresh before they can do fast refreshes again.

ROWID-Related Migration Questions and Answers

Q: Is there any version 8 restriction on a version 7 import client?

A: A version 7 client cannot import a version 8 table with ROWID user column if a row of this table contains the extended ROWID value.

Q: Do Forms3 (and Forms4) understand the new ROWID format for base table updates?

A: Forms applications which intend to access version 8 databases have to be relinked using the patch #380655. More detailed information about this patch is available in the Oracle release 8.0.4 README document.

Q: How do the version 8 ROWID changes affect PRO* precompiled programs?

A: Programs that use ROWID but do not rely on its format are not affected. Programs that rely on the version 7 ROWID format must be modified to use the new package, DBMS_ROWID.

Q: Do “WHERE CURRENT of CURSOR” operations still work?

A: Yes, even when accessing a version 8 server from a pre-version 8 client or when accessing a pre-version 8 server from a version 8 client.

Q: I currently use dynamic SQL and bind as internal ROWID format. Will I need to malloc() more space?

A: The version 8 ROWID fits into the version 7 storage requirements for host variables; therefore, no changes or additional space allocations are necessary.

Q: Can I still define a column of my table to be of ROWID type?

A: Columns can still be defined of ROWID type. ROWID column requires 10 bytes versus the 6 bytes required in version 7.

Q: I rely on the Version 7 ROWID format at present. Will the conversion algorithm be documented?

A: The new version 8 ROWID format is not documented for such use. However, version 8 provides the DBMS_ROWID (PL/SQL) package to interpret version 8 ROWID contents.

Q: Will I need to rebuild any indexes?

A: Only indexes built on a column that stores the old ROWID format needs to be rebuilt after data migration.

Q: I use ROWID datatype in pre-version 8 PL/SQL, RPC, or from FORMS. Will this continue to work?

A: The format in which ROWIDs are returned into host variables of ROWID type will be the same, and generally no change is needed, except in the following specific known case:

A remote mapped query from a version 7 server to a version 8 database across a dblink (considered a heterogeneous dblink) terminates with error ORA3116 upon a ROWID fetch as a type DTYRID (without CHR conversion) through OCI.

- Using ROWID fetches as type DTYCHR instead invokes an implicit conversion and avoids the problem.
- Using SQLT_RID and a patch (available from Oracle) on the version 7 server avoids the problem without invoking CHR conversion.

8

Upgrading and Downgrading

This chapter contains information about upgrading from your current version 8 release to a new version 8 release. This chapter also contains information about downgrading a version 8 database system to version 7.

The information in this chapter only applies to version 8 installations of Oracle. If your current release is pre-version 8, such as version 7 or version 6, and you want to migrate to version 8, follow the instructions at the beginning of this book, starting with Chapter 1, “Migration Overview”.

The following topics are covered in this chapter:

- Upgrading to a New Version 8 Release
- Product Configurations and Upgrading
- Upgrading the Advanced Queuing Option
- Downgrading

> **See Also:** Some aspects of upgrading and downgrading are platform-specific. See your platform-specific Oracle documentation for additional instructions about these operations on your platform. In addition, this chapter does not discuss using the Installation Utility to upgrade, but that is an option for you. Again, see your platform-specific Oracle documentation for details.

Upgrading to a New Version 8 Release

This section guides you through the process of upgrading to the new version 8 release from release 8.0.3.

WARNING: If you are upgrading from Oracle8 Enterprise Edition to Oracle8 (formerly Workgroup Server), before you upgrade, you must modify any applications that use the advanced features of Oracle8 Enterprise Edition so that they do not use these advanced features. See *Getting to Know Oracle8 and the Oracle8 Enterprise Edition* and "Product Configurations and Upgrading" on page 8-4 for more information about the differences between the editions.

Complete the following steps to upgrade your current version 8 database:

1. Start Server Manager by entering the following:

   ```
   SVRMGRL
   ```

2. Run SHUTDOWN NORMAL on the version 8 database:

   ```
   SVRMGR> SHUTDOWN NORMAL
   ```

3. Perform a full offline backup of the database.
4. Install the new version 8 release.

   Note: Installation is platform-specific. For installation instructions, see your version 8 platform-specific installation documentation and the version 8 README for your platform.

   ---

5. Set the COMPATIBLE parameter in your INIT.ORA file.

   See Also: "COMPATIBLE Parameter" on page C-2 for information.

6. Run CONNECT INTERNAL:

   ```
   SVRMGR> CONNECT INTERNAL
   ```

7. Run STARTUP RESTRICT:

```
SVRMGR> STARTUP RESTRICT
```

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

```
SVRMGR> SPOOL CATOUTU.LOG
```

9. Run CAT8004.SQL:

```
SVRMGR> @CAT8004.SQL
```

   The CAT8004.SQL script creates and alters certain dictionary tables. It also runs the CATALOG.SQL and CATPROC.SQL scripts, which create the system catalog views and all the necessary packages for using PL/SQL.

   See Also: *Oracle8 Reference* for a complete list and descriptions of available scripts, if you want to create additional data dictionary structures.

10. If the Oracle system has Advanced Replication installed, run the following catalog script supplied with your new release:

```
SVRMGR> @CATREP.SQL
```

11. If the Oracle system has Parallel Server installed, run the following catalog script supplied with your new release:

```
SVRMGR> @CATPARR.SQL
```

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

```
SVRMGR> SPOOL OFF
```

   Then, check the spool file and verify that *every* package and procedure compiled successfully. You named the spool file in Step 8; the suggested name was CATOUTU.LOG. Correct any problems you find in this file.

13. Run ALTER SYSTEM DISABLE RESTRICTED SESSION:

```
SVRMGR> ALTER SYSTEM DISABLE RESTRICTED SESSION
```

14. Run UTLCONST.SQL to check for bad date constraints.

> **Note:** If you already ran UTLCONST.SQL after you migrated to a previous version 8 release, you need not run it again. However, running the script many times will not damage your system; therefore, if you are unsure about whether it has been run on your system, you should run it now.

To run UTLCONST.SQL, issue the following commands:

```
SVRMGRL> SPOOL UTLRESULT.LOG
SVRMGRL> @UTLCONST.SQL
SVRMGRL> SPOOL OFF
```

A bad date constraint involves invalid date manipulation. An invalid date manipulation is one that implicitly assumes the century in the date, causing problems at the year 2000. The UTLCONST.SQL script runs through all of the check constraints in the database and sets constraints as bad if they include any invalid date manipulation. This script selects all the bad constraints at the end.

After you run the script, the UTLRESULT.LOG log file includes all the constraints that have invalid date constraints.

> **Note:** UTLCONST.SQL does not correct bad constraints, but it does disable them. You should either drop the bad constraints or recreate them after you make the necessary changes.

Product Configurations and Upgrading

Release 8.0.4 of the Oracle server is offered in two product configurations:

- Oracle8
- Oracle8 Enterprise Edition

> **See Also:** *Getting to Know Oracle8 and the Oracle8 Enterprise Edition* for information about the differences between Oracle8 and the Oracle8 Enterprise Edition and the options and features that are available to you.

If you are upgrading to Oracle8, instead of Oracle8 Enterprise Edition, you may lose certain options and features that were available to you in your previous release. The following table describes the functionality changes in such an upgrade:

Lost Option or Feature	Functionality Change
Partitioning	PARTITION clause in CREATE TABLE / INDEX / CLUSTER is disabled.
	PDML is turned off so that all operations are done serially, not in parallel.
Objects	CREATE TYPE and ALTER TYPE REPLACE are disabled. CREATE VIEW ... OF (i.e. object view) also is disabled.
Parallel Server	Startup SHARED is disabled.
Advanced Replication	Updateable snapshots are disabled. N-way master snapshots are disabled.
Bit-mapped Indexes	CREATE BITMAP INDEX statement is disabled.
	Bit-mapped indexes are not used.
	Run-time subquery optimization is turned off.
Connection Multiplexing	An Oracle error occurs at startup if the INIT.ORA file specifies multiplexing.
Connection Pooling	An Oracle error occurs at startup if the INIT.ORA file specifies pooling.
Database Queuing	Queue and dequeue procedures are disabled.
Instead-of Triggers	INSTEAD OF clause in CREATE TRIGGER statement is disabled.
Incremental Backup and Recovery	Recovery Manager (**rman**) disables incremental backup and recovery.
Parallel Backup and Recovery	Recovery Manager (**rman**) disables parallel backup and recovery.
Parallel Execution	Parallel CREATE INDEX and CREATE TABLE AS SELECT are disabled.
Parallel Load	Parallel load is disabled.
Point-in-time Tablespace Recovery	ALTER DATABASE MOUNT CLONE statement is disabled.

Upgrading the Advanced Queuing Option

The operational interface in Oracle Advanced Queuing (AQ) 8.0.4 is backward compatible with 8.0.3 Oracle AQ interface.

> **Note:** If you have not started the release 8.0.4 database with a setting of to 8.0.4.0.0 for the COMPATIBLE parameter, do so now to use the features described in this section. See "COMPATIBLE Parameter" on page C-2 for more information.

New Fields Enabled for the AQ$_AGENT Data Type

In the latest release, the address field is enabled for the AQ$_AGENT datatype. Consequently, it is now possible for this field to be specified wherever an interface takes an *Agent* as an argument — such as in the recipient list of the message properties, and the DBMS_AQADM.ADD_SUBSCRIBER administrative interface.

The Extended Address Field

The address field in the AQ$_AGENT datatype has been extended to 1024 bytes. To use the extended address field, complete the following steps:

1. Export the contents of the existing queues using the Export Utility.
2. Run CATNOQUEUE.SQL to drop the existing dictionary and queue tables:

   ```
   SVRMGRL> @CATNOQUEUE.SQL
   ```

3. Run CATQUEUE.SQL to redefine the new types and dictionary tables:

   ```
   SVRMGRL> @CATQUEUE.SQL
   ```

4. Import the queues you exported using the Import Utility.

> **Note:** If your application does not require you to extend the address field, you need not complete these steps.

New Dictionary Tables

The upgrade script for release 8.0.4 (CAT8004.SQL) creates the additional dictionary tables:

- SYSTEM.AQ$_SCHEDULES
- SYS.AQ$_MESSAGE_TYPES
- SYS.AQ$_QUEUE_STATISTICS

Downgrading

The term *downgrading* is used to describe transforming an Oracle database into a previous release of the same version, such as transforming a database from release 8.0.4 to release 8.0.3, and downgrading also is used to describe transforming an Oracle database into a previous version, such as transforming a database from version 8 to version 7. The following sections contain instructions for both cases.

WARNING: If you are downgrading from Oracle8 Enterprise Edition to Oracle8 (formerly Workgroup Server), before you downgrade, you must modify any applications that use the advanced features of Oracle8 Enterprise Edition so that they do not use these advanced features. See *Getting to Know Oracle8 and the Oracle8 Enterprise Edition* for more information about the differences between the editions.

Downgrading from Release 8.0.4 to Release 8.0.3

Complete the following steps to downgrade your release 8.0.4 database to release 8.0.3:

1. Shutdown the release 8.0.4 database by issuing the following command:

   ```
   SVRMGR> SHUTDOWN NORMAL
   ```

   Issue the command for all instances if you are running Parallel Server.

2. Perform a full offline backup.

   See Also: *Oracle8 Backup and Recovery Guide* for more information.

3. Open the release 8.0.4 database.

4. Check your COMPATIBLE parameter setting by issuing the following command:

```
SVRMGR> SELECT name, value, description FROM v$parameter
        WHERE name="compatible";
```

   If COMPATIBLE is set to 8.0.3.0.0 or 8.0.0.0.0, skip to Step 10, but, if COMPATIBLE is set to 8.0.4.0.0, continue with Step 5.

5. If you are using propagation in the Advanced Queuing Option, disable propagation; otherwise, move on to Step 6.

   See Also: "Propagation in Advanced Queuing (AQ)" on page C-3 for more information about this feature.

   To disable propagation in the Advanced Queuing Option, complete one of the following procedures:

   - Wait for all messages to be propagated successfully; then, use DBMS_AQDM.UNSCHEDULE_PROPAGATION() to unschedule all propagations.
   - Use DBMS_AQDM.UNSCHEDULE_PROPAGATION() to unschedule all propagations; then, delete all messages that are supposed to be propagated from the queue tables.

     See Also: *Oracle8 Application Developer's Guide* for information about DBMS_AQDM.UNSCHEDULE_PROPAGATION().

6. Run ALTER DATABASE RESET COMPATIBILITY:

```
SVRMGR> ALTER DATABASE RESET COMPATIBILITY
```

7. Run SHUTDOWN NORMAL:

```
SVRMGR> SHUTDOWN NORMAL
```

   WARNING: Do not open the database using COMPATIBLE=8.0.4.0.0 after completing the step.

8. Set the COMPATIBLE parameter in the INIT.ORA file to the following:

```
COMPATIBLE=8.0.0.0.0
```

9. Open the database to ensure that it is compatible with release 8.0.3.

10. Run CAT8004D.SQL:

```
SVRMGR> @CAT8004D.SQL
```

11. Run SHUTDOWN NORMAL:

```
SVRMGR> SHUTDOWN NORMAL
```

12. Install the 8.0.3 release using your release 8.0.3 installation media.

> **Note:** Installation is platform-specific. For installation instructions, see your release 8.0.3 platform-specific installation documentation and the release 8.0.3 README for your platform.

13. Run CONNECT INTERNAL:

```
SVRMGR> CONNECT INTERNAL
```

14. Run STARTUP RESTRICT:

```
SVRMGR> STARTUP RESTRICT
```

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

```
SVRMGR>SPOOL CATOUTD.LOG
```

16. Run CATALOG.SQL:

```
SVRMGR>@CATALOG.SQL
```

17. Run CATPROC.SQL:

```
SVRMGR>@CATPROC.SQL
```

18. If the Oracle system has Advanced Replication installed, run the following catalog script supplied with the 8.0.3 release:

```
SVRMGR>@CATREP.SQL
```

19. If the Oracle system has Parallel Server installed, run the following catalog script supplied with the 8.0.3 release:

```
SVRMGR>@CATPARR.SQL
```

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

```
SVRMGR> SPOOL OFF
```

Then, check the spool file and verify that *every* package and procedure compiled successfully. You named the spool file in Step 15; the suggested name was CATOUTD.LOG. Correct any problems you find in this file.

Downgrading Version 8 to Release 7.*x*

A version 8 database can be downgraded to a version 7 database (such as release 7.3). However, few downgrade paths are available, and the necessary procedures may require a great deal of time and effort.

Oracle does not support downgrading from version 8 to version 7 using the version 8 Migration Utility, and version 8 provides no other facilities specifically for downgrading.

The procedure for downgrading depends on whether the version 8 database contains new data that must be preserved. Use the procedure that applies to your version 8 database:

- Downgrading a Version 8 Database That Contains No New Data
- Downgrading a Version 8 Database That Contains New Data

Downgrading a Version 8 Database That Contains No New Data

If the version 8 database contains *no new data* that needs to be preserved, simply restore the complete backup of the previous release 7.*x* source database and open it again. Make sure the restore includes the initialization parameters that were used in the previous release 7.*x* database.

Downgrading a Version 8 Database That Contains New Data

If the version 8 database contains *new data that needs to be preserved*, complete the following steps:

1. Use version 7 Export Utility with the CATEXP7.SQL-created views installed to export the part of the version 8 database containing the new data.

   See Also: *Oracle8 Utilities* for more information about performing a version 7 export from version 8.

2. Restore the complete backup of the previous version 7 database, making sure the restore includes the previous initialization parameters.

3. Open the restored database.
4. Use the version 7 Import Utility to import the file previously exported from the version 8 database into the restored version 7 database.

Several other methods are available for sending table data from the version 8 database back to the version 7 database. These methods of returning data to version 7 are relatively simple if only a few tables have been updated using version 8. However, copying an entire database of tables can be a long and complicated task; therefore, you should decide whether you need to return to version 7 before you update many tables using version 8.

The following alternate methods are available for downgrading a version 8 database to version 7:

- Use SQL*Plus to create non-Oracle text files from the new version 8 data and then use SQL*Loader to load the data back into the version 7 database.
- Use the SQL*Plus COPY command to copy the new data from the version 8 database tables into the tables in the earlier version 7 database.

  See Also: The *SQL*Plus User's Guide and Reference* for more information about the COPY command.

- If you are still running the earlier version 7 database, re-create the table where data has been added or modified (using CREATE TABLE ... AS SELECT) by selecting the data in the version 8 database table through a distributed query from that version 7 database using a database link.

  See Also: *Oracle8 Server SQL Reference* or more information on the AS clause of the CREATE TABLE command.

A

Migration Utility Messages

The Migration Utility may return error messages and informational messages during migration. This appendix describes errors you may encounter when using the Migration Utility. For each error, a description of its probable cause and instructions for corrective action are provided. Informational messages also are listed, but they require no corrective action.

cannot reduce file number bits in DBA during migration

Cause: The Migration Utility attempted to reduce the number of file-number bits used in a datablock address.

Action: Contact your Oracle Customer Support representative.

cannot create conversion file, records exceed *number* bytes

Cause: An internal error occurred. A valid convert file could not be created from the version 7 control file.

Action: Check the version 7 control file for corruption, fix any problems, and rerun the Migration Utility.

CHECK_ONLY - estimate V8 catalog space requirement ONLY (default=FALSE)

Cause: This is an informational message about the CHECK_ONLY command line argument.

Action: No user action is required.

CHECK_ONLY and NO_SPACE_CHECK are mutually exclusive options

Cause: These 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 (*string*)

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> (*string*)

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 informational message displays the command-line arguments.

Action: No user action is required.

command name not found (*string*)

Cause: An internal error has occurred; the MIGRATE.BSQ script may be corrupted.

Action: Check that the version of the Migration Utility, of MIGRATE.BSQ, and of the target version 8 software are compatible, and that no corruption exists in MIGRATE.BSQ. Fix any problems, and rerun the Migration Utility.

command not of form CMD(ARG1, ARG2, ...)

Cause: An internal error has occurred; the MIGRATE.BSQ script may be corrupted.

Action: Check that the version of the Migration Utility, of MIGRATE.BSQ, and of the target version 8 software are compatible, and that no corruption exists in MIGRATE.BSQ. Fix any problems, and rerun the Migration Utility.

copy long command must be of form COPYLONG(U1,T1,C1,U2,T2,C2,K1<,K2>)

Cause: An internal error has occurred; the MIGRATE.BSQ script may be corrupted.

Action: Check that the version of the Migration Utility, MIGRATE.BSQ, and of the target version 7 software are compatible, and that no corruption exists in MIGRATE.BSQ. Fix any problems, and rerun the Migration Utility.

could not find single contiguous extent of *number* 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 rerun the Migration Utility.

could not find single contiguous extent of *number* 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 rerun the Migration Utility.

could not find single contiguous extent of *number* bytes for i_file#_block#

Cause: You do not have enough contiguous space in your SYSTEM tablespace.

Action: Add free space to your SYSTEM tablespace, and rerun the Migration Utility.

could not find single contiguous extent of *number* bytes for i_ts#

Cause: You do not have enough contiguous space in your SYSTEM tablespace.

Action: Add free space to your SYSTEM tablespace, and rerun the Migration Utility.

could not translate logical name *name*

Cause: An internal error has occurred.

Action: Check that the logical name is defined correctly, and rerun the Migration Utility.

current version: *str* -- Database must be version 7.1 or later

Cause: Current database is an earlier version than version 7, release 7.1.

Action: Migrate or upgrade current database to a release supported by the Migration Utility on your platform. Then, rerun the Migration Utility. See your platform-specific Oracle documentation for information about the releases supported by the Migration Utility on your platform.

data type must be long for column *name*

Cause: An internal error has occurred; the MIGRATE.BSQ script may be corrupted.

Action: Check that the version of the Migration Utility, of MIGRATE.BSQ, and of the target version 7 software are compatible, and that no corruption exists in MIGRATE.BSQ. Fix any problems, and rerun the Migration Utility.

datafiles is found in inconsistent states (internal error) -- *filename*

Cause: An internal error occurred; a datafile was found in inconsistent state.

Action: Contact your Oracle Customer Support representative.

datafile is offline while tablespace is online - apply media recovery and bring datafile online before migration -- *datafile*

Cause: The datafile in a tablespace is offline while the tablespace is online. Migration cannot proceed until the datafile and tablespace are both either online or offline normal.

Action: Apply media recovery and bring the datafile online before rerunning the migration.

DBNAME - current database name (db_name in init.ora)

Cause: This is an informational message about the DBNAME command-line argument.

Action: No user action is required.

dictionary constant not found - *name*

Cause: An internal error has occurred; the MIGRATE.BSQ script may be corrupted.

Action: Check that the version of the Migration Utility, of MIGRATE.BSQ, and of the target version 8 software are compatible, and that no corruption exists in MIGRATE.BSQ. Fix any problems, and rerun the Migration Utility.

entries found in system.def$_call, def$_calldest or def$_error - push all deferred transactions before migration

Cause: Entries exist in system.def$_call, def$_calldest, or def$_error.

Action: If entries are in system.def$_call, push all deferred transactions until system.def$_call is empty. If entries are in system.def$_error, resolve and re-execute any errors in the local queue until it is empty. Rerun the Migration Utility.

error calling slgtd

Cause: Error in getting current time from slgtd, an internal error. The Migration Utility may be corrupted.

Action: Check that the version of the Migration Utility, of MIGRATE.BSQ, and of the target version 8 software are compatible, and that no corruption exists in MIGRATE.BSQ. Fix any problems, and rerun the Migration Utility.

error closing file *name*

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 rerun the Migration Utility.

estimated space requirement for *object* is *number* blocks

Cause: In this informational message, the Migration Utility displays the space required for the object.

Action: No user action is required.

file *filename* is too large for DBA conversion

Cause: An internal error has occurred; *filename* is too large for DBA conversion.

Action: Contact your Oracle Customer Support representative.

file header does not fit in *number* bytes

Cause: An internal error has occurred.

Action: Check the control file for corruption, fix any problems, and rerun the Migration Utility.

fixed portion of control file does not fit in *number* bytes

Cause: An internal error has occurred.

Action: Check the control file for corruption, fix any problems, and rerun the Migration Utility.

found NULL SQL statement

Cause: An internal error has occurred; the MIGRATE.BSQ script may be corrupted.

Action: Check that the version of the Migration Utility, of MIGRATE.BSQ, and of the target version 8 software are compatible, and that no corruption exists in MIGRATE.BSQ. Fix any problems, and rerun the Migration Utility.

free space found in system tablespace is *number* blocks

Cause: This informational message shows the amount of free space in the SYSTEM tablespace.

Action: No user action is needed.

free space found: *number*

Cause: This informational message shows the amount of free space in the SYSTEM tablespace.

Action: No user action is needed.

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 rerun the Migration Utility.

insufficient space for new dictionaries, *number* bytes needed, *number* 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.

invalid NLS_NCHAR value specified

Cause: The NLS_NCHAR value specified in the command line is invalid.

Action: Correct the NLS_NCHAR value specified in the command line, and rerun the Migration Utility.

migration can't proceed - database blocksize size is less than Oracle8's minimum block size 2k

Cause: The existing database blocksize is less than 2KB.

Action: Make sure the block size of the version 7 database is at least 2KB. You may consider rebuilding the version 7 database. Then, rerun the Migration Utility.

migration can't proceed with datafile online while tablespace offline -- *datafile*

Cause: The datafile in a tablespace is online while the tablespace is offline. Migration cannot proceed until the datafile and tablespace are both either online or offline normal.

Action: Make sure the online status of the datafile is the same as the online status of the tablespace, and rerun the Migration Utility.

migration cannot proceed with active transaction or offline tablespaces with outstanding undo

Cause: One or more tablespaces were offline with outstanding save undo when the Migration Utility attempted to migrate the database.

Action: Go to Step 7 on 3-7 in "Prepare the Version 7 Source Database for Migration" and make sure all offline tablespaces have been taken offline cleanly. Then, rerun the version 8 Migration Utility.

mounting database ...

Cause: This is an informational message. The Migration Utility is mounting the version 7 database.

Action: No user action is required.

MULTIPLIER - seg$/uet$ cluster index size increase factor (default=15)

Cause: This is an informational message that the Migration Utility displays about the MULTIPLIER command-line setting.

Action: No user action is required.

MULTIPLIER value must be at least 2

Cause: The MULTIPLIER value, which specifies the initial size of the version 8 i_file#_block# in the command line, is less than 2.

Action: Change the MULTIPLIER value to be equal to or greater than 2, and rerun the Migration Utility.

NEW_DBNAME *name* too long - maximum length is 8 characters

Cause: The new database name specified is more than 8 characters long.

Action: Change the specified name for the new database to 8 or fewer characters, and rerun the Migration Utility.

NEW_DBNAME - new name for the database (max. 8 characters)

Cause: This informational message displays information about the NEW_DBNAME command-line argument.

Action: No user action is required.

NLS_NCHAR - specify the nchar characterset value

Cause: This informational message displays information about the NLS_NCHAR command-line argument.

Action: No user action is required.

NO_SPACE_CHECK - do not execute the space check (default=FALSE)

Cause: This is an informational message about the NO_SPACE_CHECK command-line argument.

Action: No user action is required, but make sure there is adequate space before you run the Migration Utility with this option.

***tablespace/datafile* number being processed is incorrect during creating convert file**

Cause: An internal error occurred while creating the convert file.

Action: Contact your Oracle Customer Support representative.

opening database ...

Cause: This is an informational message. The Migration Utility is opening the version 7 database.

Action: No user action is required.

ORA_NLS33 environment variable is not set or incorrectly set

Cause: The ORA_NLS33 environment variable does not point to the NLS datafiles.

Action: Set the ORA_NLS33 environment variable to point to the correct files, and rerun the Migration Utility.

ORA-*number*:

Cause: The Migration Utility has received an ORA error and cannot retrieve the message text for the error.

Action: Take appropriate action based on the Oracle error *number* (see *Oracle8 Error Messages*).

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 rerun the Migration Utility.

parameter file exceeds *number* bytes

Cause: The parameter file for your version 7 database exceeds the maximum size.

Action: If possible, reduce the size of your parameter file by removing obsolete parameters. Otherwise, contact your Oracle Customer Support representative.

PFILE - use alternate 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 reading file *name*.

Action: Make sure the file and disk are not corrupted. Fix any corruption before you rerun the Migration Utility.

short read, *number* bytes requested, *number* bytes read

Cause: There is a problem reading the control file.

Action: Check the control file for corruption, fix any problems, and rerun the Migration Utility.

shut down database (abort) ...

Cause: An internal error occurred.

Action: Additional error messages should inform you of the cause of the shutdown. Follow the actions suggested for these additional messages.

shutting down database ...

Cause: This is an informational message. The Migration Utility is shutting down the version 7 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 7 instance.

Action: No user action is required.

string argument too long, maximum length *number*

Cause: A string in the command line argument passed to the Migration Utility exceeds the maximum size.

Action: Shorten the string in the command line argument, and rerun the Migration Utility.

tablespace of datafile not taken offline normal. Bring tablespace online, offline normal or drop before migration -- *tablespace*

Cause: Tablespace was taken offline using IMMEDIATE or TEMPORARY.

Action: Bring tablespace online, and then take it offline using NORMAL or drop it. Then, rerun the Migration Utility.

too many args in command (*number* 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 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 Migration Utility, and rerun the Migration Utility.

unable to open file *name*

Cause: An internal error has occurred, or a file was not in the expected location, when you started the version 8 Migration Utility.

Action: Check that the file exists and that its access permissions allow Oracle to open and read the it. If possible, check that the file, and/or the disks on which the file resides, are not corrupt. Fix any problems, and rerun the Migration Utility.

unable to read file *name*

Cause: An internal error occurred or a file was not in the expected location when you started the Migration Utility.

Action: Check that the file exists and that its access permissions allow Oracle to open and read the it. If possible, check that the file, and/or the disks on which the file resides, are not corrupt. Fix any problems, and rerun the Migration Utility.

unable to write file *name*

Cause: An internal error occurred.

Action: Check the access permissions to make sure that Oracle can write to the file. Check that the disks to which the file is being written are not corrupt. Fix any corruption; then, rerun the Migration Utility.

V8 catalog space requirement: *number*

Cause: This is an informational message that shows the amount of additional space required in your SYSTEM tablespace to run the Migration Utility successfully.

Action: Make sure you have the specified amount of additional space before running the Migration Utility.

B

Control File Fixed View Changes

This appendix briefly describes changes from version 7, release 7.3 to the version 8 server Control File Fixed Views.

Date Columns in Control File Views

In version 7, all control file views date columns are VARCHAR2(20) strings in 'MM/DD/YY HH24:MI:SS' format.

In version 8, every new date column is a real DATE column. In contrast to the previous VARCHAR2(20) string, DATE provides users the following benefits:

- Consistency because all date columns are in DATE datatype.
- Easier to perform date arithmetic (including sorting) in SQL and PL/SQL with dates.
- Enables users to set their date format using NLS_DATE_FORMAT.
- Users who want to see dates in the old format can set NLS_DATE_FORMAT to MM/DD/YY HH24:MI:SS.
- Avoids two-digit year numbers to avoid problems at the year 2000 and beyond.

This appendix lists the names of the views and the names of the columns in the views that have been changed or added to version 8.

> **See Also:** *Oracle8 Reference* for descriptions of each column and for a complete listing of the columns in the tables.

Obsolete Views Kept in Version 8

V$ARCHIVE still shows logs that must be archived.

V$LOG was superceded by V$LOG_HISTORY, which is superseded by the new version 8 view, V$ARCHIVED_LOG. In version 8, however, V$LOG is still provided and still lists all archived logs in its ARCHIVED column.

The obsolete V$LOGHIST view is retained for historical compatibility.

V$LOG_HISTORY Retained and Upgraded

In version 8, several V$LOG_HISTORY columns have been renamed, as shown in the following table. Further, the (version 7) ARCHIVE_NAME column is not present in version 8.

Version 8 Column	Version 7 Column
FIRST_TIME	TIME
FIRST_CHANGE#	LOW_CHANGE#
NEXT_CHANGE#	HIGH_CHANGE#
[not present]	ARCHIVE_NAME

V$ARCHIVED_LOG Replaces V$LOG_HISTORY

The new version 8 view, V$ARCHIVED_LOG, shows archived logs, including their names. If a log is never archived, the V$ARCHIVED_LOG view does not return any row for the log. If a log is archived twice, V$ARCHIVED_LOG returns two rows for the log.

Certain columns in this new view correspond to version 7 V$LOG_HISTORY names and use formats that match the column names in other version 8 views. The V$ARCHIVED_LOG view shows archived log information from the control file:

Column Name	Column Type
RECID	NUMBER
STAMP	NUMBER
NAME	VARCHAR2(512)
THREAD#	NUMBER
SEQUENCE#	NUMBER
RESETLOGS_CHANGE#	NUMBER
RESETLOGS_TIME	DATE
FIRST_CHANGE#	NUMBER
FIRST_TIME	DATE
NEXT_CHANGE#	NUMBER
NEXT_TIME	DATE
BLOCKS	NUMBER
BLOCK_SIZE	NUMBER
COMPLETION_TIME	DATE
DELETED	VARCHAR2(3)

An archive log record is inserted each time the online redo log is archived successfully or cleared—the name column is null when the log is cleared. If the log is archived twice, two archive log records are inserted with the same thread#, sequence#, and first_change#, but with different names. An archive log record also is inserted when an archive log is restored from a backup set or a copy.

V$BACKUP_CORRUPTION

The V$BACKUP_CORRUPTION view shows information about corruptions in datafile backups from the control file. Corruptions are not tolerated in control file and archived log backups.

Column Name	Column Type
RECID	NUMBER
STAMP	NUMBER
SET_STAMP	NUMBER
SET_COUNT	NUMBER
PIECE#	NUMBER
FILE#	NUMBER
BLOCK#	NUMBER
BLOCKS	NUMBER
CORRUPTION_CHANGE#	NUMBER
MARKED_CORRUPT	VARCHAR2(3)

V$BACKUP_DATAFILE

The V$bBACKUP_DATAFILE view shows backup datafile and backup control file information from the control file. A datafile backup set can contain one control file and multiple datafiles.

Column Name	Column Type
RECID	NUMBER
STAMP	NUMBER
SET_STAMP	NUMBER
SET_COUNT	NUMBER
FILE#	NUMBER
CREATION_CHANGE#	NUMBER
CREATION_TIME	DATE
RESETLOGS_CHANGE#	NUMBER
RESETLOGS_TIME	DATE
INCREMENTAL_LEVEL	NUMBER
INCREMENTAL_CHANGE#	NUMBER
CHECKPOINT_CHANGE#	NUMBER
CHECKPOINT_TIME	DATE
ABSOLUTE_FUZZY_CHANGE#	NUMBER
MARKED_CORRUPT	NUMBER
MEDIA_CORRUPT	NUMBER
LOGICALLY_CORRUPT	NUMBER
DATAFILE_BLOCKS	NUMBER
BLOCKS	NUMBER
BLOCK_SIZE	NUMBER
OLDEST_OFFLINE_RANGE	NUMBER

V$BACKUP_DEVICE

The new version 8 view, V$BACKUP_DEVICE, shows information about supported backup devices. If a device type does not support named devices, one row with the device type and a null device name is returned for that device type. If a device type supports named devices, one row is returned for each available device of that type. The special device type DISK is not returned by this view because it is always available.

Column Name	Column Type
DEVICE_TYPE	VARCHAR2(17)
DEVICE_NAME	VARCHAR2(512)

V$BACKUP_PIECE

The new version 8 view, V$BACKUP_PIECE, shows information about backup pieces stored in the control file. Each backup set consist of one or more backup pieces.

Column Name	Column Type
RECID	NUMBER
STAMP	NUMBER
SET_STAMP	NUMBER
SET_COUNT	NUMBER
PIECE#	NUMBER
DEVICE_TYPE	VARCHAR2(17)
HANDLE	VARCHAR2(513)
COMMENTS	VARCHAR2(81)
MEDIA	VARCHAR2(65)
CONCUR	VARCHAR2(3)
TAG	VARCHAR2(32)
DELETED	VARCHAR2(3)

V$BACKUP_REDOLOG

The V$BACKUP_REDOLOG view shows information about archived logs in backup sets from the control file. Online redo logs cannot be backed up directly, they first must be archived to disk and then backed up. An archive log backup set can contain one or more archived logs.

V$BACKUP_SET

The V$BACKUP_SET view shows backup set information from the control file. A backup set record is inserted after the backup set is completed successfully.

Column Name	Column Type
RECID	NUMBER
STAMP	NUMBER
SET_STAMP	NUMBER
SET_COUNT	NUMBER
BACKUP_TYPE	VARCHAR2(1)
CONTROLFILE_INCLUDED	VARCHAR2(3)
INCREMENTAL_LEVEL	NUMBER
PIECES	NUMBER
COMPLETION_TIME	DATE
BLOCK_SIZE	NUMBER

V$CONTROLFILE_RECORD_SECTION

The new version 8 view, V$CONTROLFILE_RECORD_SECTION, shows information about the control file record sections.

Column name	Column type
TYPE	VARCHAR2(17)
RECORD_SIZE	NUMBER
RECORDS_TOTAL	NUMBER
RECORDS_USED	NUMBER
FIRST_INDEX	NUMBER
LAST_INDEX	NUMBER
LAST_RECID	NUMBER

V$COPY_CORRUPTION

The new version 8 view, V$COPY_CORRUPTION, shows information about datafile copy corruptions from the control file.

Column name	Column type
RECID	NUMBER
STAMP	NUMBER
COPY_RECID	NUMBER
COPY_STAMP	NUMBER
FILE#	NUMBER
BLOCK#	NUMBER
BLOCKS	NUMBER
CORRUPTION_CHANGE#	NUMBER
MARKED_CORRUPT	VARCHAR2(3)

V$DATABASE New Columns

The V$DATABASE view shows database information from the control file. The following V$DATABASE columns are new to version 8:

Column Name	Value
DBID	NUMBER
RESETLOGS_CHANGE#	NUMBER
RESETLOGS_TIME	DATE
CONTROLFILE_TYPE	VARCHAR2(7)
CONTROLFILE_CREATED	DATE
CONTROLFILE_SEQUENCE#	NUMBER
CONTROLFILE_CHANGE#	NUMBER
nCONTROLFILE_TIME	DATE
OPEN_RESETLOGS	VARCHAR2(11)

V$DATAFILE New Columns

The V$DATAFILE view shows datafile information from the control file. The version 8 V$DATAFILE view shows the following new columns, in addition to the columns of the existing version 7 V$DATAFILE view. See also the V$DATAFILE_HEADER view, which shows information from datafile headers.

Column Name	Column Type
CREATION_CHANGE#	NUMBER
CREATION_TIME	DATE
TS#	NUMBER
RFILE#	NUMBER
CHECKPOINT_TIME	DATE
UNRECOVERABLE_CHANGE#	NUMBER
UNRECOVERABLE_TIME	DATE
LAST_CHANGE#	NUMBER
LAST_TIME	DATE
OFFLINE_CHANGE#	NUMBER
ONLINE_CHANGE#	NUMBER
ONLINE_TIME#	DATE
BLOCKS	NUMBER
BLOCK_SIZE	NUMBER

V$DATAFILE_COPY

The V$DATAFILE_COPY view shows datafile copy information from the control file. A datafile copy record is inserted after the datafile (or datafile copy) is copied successfully or restored from a backup set.

Column Name	Column Type
RECID	NUMBER
STAMP	NUMBER
NAME	VARCHAR2(512)
TAG	VARCHAR2(32)
FILE#	NUMBER
RFILE#	NUMBER
CREATION_CHANGE#	NUMBER
CREATION_TIME	DATE
RESETLOGS_CHANGE#	NUMBER
RESETLOGS_TIME	DATE
CHECKPOINT_CHANGE#	NUMBER
CHECKPOINT_TIME	DATE
ABSOLUTE_FUZZY_CHANGE#	NUMBER
RECOVERY_FUZZY_CHANGE#	NUMBER
RECOVERY_FUZZY_TIME	DATE
ONLINE_FUZZY	VARCHAR2(3)
BACKUP_FUZZY	VARCHAR2(3)
MARKED_CORRUPT	NUMBER
MEDIA_CORRUPT	NUMBER
LOGICALLY_CORRUPT	NUMBER
BLOCKS	NUMBER

Column Name	Column Type
BLOCK_SIZE	NUMBER
OLDEST_OFFLINE_RANGE	NUMBER
COMPLETION_TIME	DATE
DELETED	VARCHAR2(3)

V$DATAFILE_HEADER

The new version 8 view, V$DATAFILE_HEADER, shows datafile information from the datafile head.

Column name	Column type
FILE#	NUMBER
STATUS	VARCHAR2(7)
ERROR	VARCHAR2(18)
RECOVER	VARCHAR2(3)
FUZZY	VARCHAR2(3)
CREATION_CHANGE#	NUMBER
CREATION_TIME	DATE
TABLESPACE_NAME	VARCHAR2(30)
TS#	NUMBER
RFILE#	NUMBER
RESETLOGS_CHANGE#	NUMBER
RESETLOGS_TIME	DATE
CHECKPOINT_CHANGE#	NUMBER
CHECKPOINT_TIME	DATE
CHECKPOINT_COUNT	NUMBER
BYTES	NUMBER

Column name	Column type
BLOCKS	NUMBER
NAME	VARCHAR2(512)

V$DELETED_OBJECT

The new version 8 view, V$DELETED_OBJECT, shows information about deleted archived logs, datafile copies, and backup pieces from the control file. The purpose of this view is to optimize the recovery catalog resync operation. When an archived log, datafile copy, or backup piece is deleted, the corresponding record is marked deleted.

Column Name	Column Type
RECID	NUMBER
STAMP	NUMBER
TYPE	VARCHAR2(13)
OBJECT_RECID	NUMBER
OBJECT_STAMP	NUMBER

V$INSTANCE

The new version 8 view, V$INSTANCE, shows status information for the database instance. Returning only one row, it is incompatible with the version 7 view, V$INSTANCE, which returns one row per value. The STARTUP_TIME, LOGINS, and SHUTDOWN_PENDING columns return the same information as the version 7 version of the V$INSTANCE:

```
SVRMGR> select startup_time, logins, shutdown_pending from v$instance;
```

STARTUP_TIME	LOGINS	SHUTDOWN_PENDING
96.05.29 20:29:38	ALLOWED	NO

```
SQLDBA> select * from v$instance;
```

KEY	VALUE
RESTRICTED MODE	0
SHUTDOWN PENDING	0
STARTUP TIME - JULIAN	2450215
STARTUP TIME - SECONDS	50407
4 rows selected.	

Although the version 8 V$INSTANCE view is not compatible with the version 7 V$INSTANCE view, it provides all of the same information, as well as additional information:

Column Name	**Column Type**
INSTANCE_NUMBER	NUMBER
INSTANCE_NAME	VARCHAR2(16)
HOST_NAME	VARCHAR2(64)
VERSION	VARCHAR2(17)
STARTUP_TIME	DATE
STATUS	VARCHAR2(7)

Column Name	Column Type
PARALLEL	VARCHAR2(3)
THREAD#	NUMBER
ARCHIVER	VARCHAR2(7)
LOG_SWITCH_WAIT	VARCHAR2(11)
LOGINS	VARCHAR2(10)
SHUTDOWN_PENDING	VARCHAR2(3)

If the instance is not started, a query from this view signals ORA-1034.

V$OFFLINE_RANGE

The V$OFFLINE_RANGE view shows offline range information from the control file. The most recent offline range of each datafile is stored in the datafile record. V$OFFLINE_RANGE: Datafile offline range information from control file:

Column Name	Value
RECID	NUMBER
STAMP	NUMBER
FILE#	NUMBER
OFFLINE_CHANGE#	NUMBER
ONLINE_CHANGE#	NUMBER
ONLINE_TIME#	DATE

The last offline range of each datafile is kept in the datafile record; see the V$DATAFILE view.

V$RESOURCE_LIMIT

The new version 8 view, V$RESOURCE_LIMIT, displays information about global resource use for some of the system resources.

Column Name	Column Type
RESOURCE_NAME	VARCHAR2(30)
CURRENT_UTILIZATION	NUMBER
MAX_UTILIZATION	NUMBER
INITIAL_ALLOCATION	VARCHAR2(10)
LIMIT_VALUE	VARCHAR2(10)

V$TABLESPACE

The new version 8 view, V$TABLESPACE, shows tablespace information from the control file:

Column Name	Value	Description
TS#	NUMBER	Tablespace number
NAME	VARCHAR2(30)	Tablespace name

V$THREAD

The V$THREAD view contains thread information from the control file.

Column Name	Value	Description
ENABLE_CHANGE#	NUMBER	SCN at which thread was enabled
ENABLE_TIME	DATE	Time of enable SCN
DISABLE_CHANGE#	NUMBER	SCN at which thread was disabled
DISABLE_TIME	DATE	Time of disable SCN

Changed Column Types

In version 8, the datatype of the following time stamp columns have been changed to DATE from version 7 VARCHAR2(20).

View Name	Column
V$BACKUP	TIME
V$DATABASE	CREATED
V$LOG	FIRST_TIME
V$LOG_HISTORY	FIRST_TIME (used to be TIME)
V$LOGHIST	FIRST_TIME
V$RECOVER_FILE	TIME
V$RECOVERY_STATUS	RECOVERY_CHECKPOINT
V$RECOVERY_STATUS	TIME_NEEDED
V$THREAD	OPEN_TIME
V$THREAD	CHECKPOINT_TIME

Database Scheduling Facilities

The primary goal of the Database Scheduling feature is to provide more control over the following:

- Changed Fixed Views
- New Fixed Views

Changed Fixed Views

- V$SESSION has SCHEDULER_CLASS column added
- V$MYSESSION has SCHEDULER_CLASS column added
- DBA_USERS has DEFAULT_SCHEDULER_CLASS column added
- USER_USERS has DEFAULT_SCHEDULER_CLASS column added

New Fixed Views

- V$SERVER_THRESHOLD: data about current server thresholds
- V$SCHEDULER_CLASS: data about currently active scheduler classes
- V$COMMON_SERVER: data about currently active common servers
- V$COMMON_SERVER_REQUEST: data about types of requests currently being processed
- V$COMMON_SERVER_REQUEST_HISTORY: data about types of requests that have been processed
- V$COMMON_SERVER_QUEUE: data about queues of currently active common servers
- DBA_SCHEDULER_CLASS_PRIVS: new DBA view showing all scheduler classes and users / roles to which they have been granted
- DBA_SCHEDULER_CLASSES: new DBA view showing all scheduler classes that exist in the database
- DBA_SCHEDULER_PLANS: new DBA view showing all scheduler plans that exist in the database
- DBA_SCHEDULER_PLAN_ENTRIES: new DBA view showing all scheduler plan entries that exist in the database
- SCHEDULER_CLASS_POLICIES: new DBA view showing all available scheduling policies for scheduler classes
- SCHEDULER_PLAN_POLICIES: new DBA view showing all available scheduling policies for scheduler plans
- USER_SCHEDULER_CLASS_PRIVS: new PUBLIC view showing all scheduler classes granted to the user

Table (View) Name Changes

Pre-Version 8 View Name	Version 8 Table Name
ALL_HISTOGRAMS	ALL_TAB_HISTOGRAMS
DBA_HISTOGRAMS	DBA_TAB_HISTOGRAMS
USER_HISTOGRAMS	USER_TAB_HISTOGRAMS

C

Version 8 INIT.ORA Changes

Version 8 supports new INIT.ORA initialization parameters, while some release 7.3 parameters have been changed or have become obsolete in version 8.

The following topics are covered in this appendix:

- COMPATIBLE Parameter
- Data Dictionary Protection
- DML_LOCKS
- NCHAR and NLS Parameters and Compatibility
- Pre-Version 8 Parameters Renamed in Version 8
- Release 7.3 Parameters Obsolete in Version 8
- REPLICATION_DEPENDENCY_TRACKING for the Replication Server
- Features No Longer Supported in Version 8

COMPATIBLE Parameter

The COMPATIBLE parameter specifies the release with which the Oracle server must maintain compatibility. The COMPATIBLE parameter setting in your INIT.ORA file determines the functionality of your Oracle server; therefore, it is important to set it correctly.

However, you should set the COMPATIBLE parameter at a specific point in your migration, upgrading, or downgrading process. Therefore, follow the procedure in the appropriate chapter and set the COMPATIBLE parameter only when you are instructed to do so. After the operation is complete, you can set the COMPATIBLE parameter whenever necessary.

To check your current COMPATIBLE parameter setting, issue the following command:

```
SVRMGR> SELECT name, value, description FROM v$parameter
        WHERE name="compatible";
```

WARNING: If you are migrating to version 8, make sure the COMPATIBLE parameter is not set to any version 7 release.

Migrating or Upgrading to Release 8.0.4

If you are migrating or upgrading to release 8.0.4, the COMPATIBLE parameter is set to the following by default:

```
COMPATIBLE=8.0.0.0.0
```

If you do not set the COMPATIBLE parameter at all, it is set to this value. This setting has the same effect as setting COMPATIBLE to 8.0.3.0.0 and allows you to downgrade to release 8.0.3 easily, if you choose to do that in the future.

However, if you use the default setting (8.0.0.0.0), you will not be able to use the following new features of release 8.0.4:

- Propagation in Advanced Queuing (AQ)
- Improved SCN Generation

To use these new release 8.0.4.0.0 features, set the COMPATIBLE parameter to the following:

```
COMPATIBLE=8.0.4.0.0
```

Note: Once the COMPATIBLE parameter is set to 8.0.4.0.0, it may require more work to downgrade to a previous release. However, if you do not plan to downgrade to a previous release, Oracle Corporation recommends a setting of 8.0.4.0.0.

The following sections provide a brief overview of the new features that require a COMPATIBLE parameter setting of 8.0.4.0.0.

Propagation in Advanced Queuing (AQ)

Release 8.0.4 does not support distributed object types; therefore, you cannot implement remote enqueuing or dequeuing using a standard database link. However, in release 8.0.4, you can use AQ's message propagation to enqueue to a remote queue.

For example, you can connect to database X and enqueue the message in a queue, for example “DROPBOX” located in database X. You can configure AQ so that all messages enqueued in queue “DROPBOX” are automatically propagated to another queue in database Y, regardless of whether database Y is local or remote. AQ automatically checks if the type of the remote queue in database Y is structurally equivalent to the type of the local queue in database X, and propagates the messages.

Recipients of propagated messages can be either applications or queues. If the recipient is a queue, the actual recipients are determined by the subscription list associated with the recipient queue. If the queues are remote, messages are propagated using the specified DBLINK. Only AQ to AQ message propagation is supported.

See Also: The *Oracle8 Application Developer’s Guide* for more information about propagation in AQ.

Improved SCN Generation

Release 8.0.4 implements an improved algorithm for System Change Number (SCN) generation. If you run Oracle Parallel Server with the INIT.ORA parameter MAX_COMMIT_PROPAGATION_DELAY set to less than 7 seconds, Oracle Corporation strongly recommends that you set COMPATIBLE to 8.0.4.0.0.

See Also: *Oracle8 Concepts* for more information about SCN generation.

Data Dictionary Protection

O7_DICTIONARY_ACCESSIBILITY is the INIT.ORA parameter switch that continues version 7 data dictionary behavior. It is only a temporary expedient and is not planned for future version 8 releases.

DML_LOCKS

Version 8 systems typically consume more DML locks while performing DDL operations than are required for version 7 systems. Nevertheless, the version 7 DML_LOCKS parameter default settings usually are adequate for version 8 systems, even for DML-intensive applications.

The default value of DML_LOCKS is a multiple of the number of transactions, which is calculated from the number of rollback segments. However, in version 8 fewer transactions are used per rollback segment than are used in version 7. Consequently, DML_LOCKS has a lower default value in version 8. Under some extreme test conditions, a version 8 system exceeded its (version 7) DML lock limit, and the DML_LOCKS parameter value had to be increased.

Also, you may need to adjust the TRANSACTION_PER_ROLLBACK_SEGMENT parameter setting, depending on the platform-specific settings. An informational message about this change may be displayed during database startup operations, for example at Step 12 in “Migration Steps in the Version 8 Environment” on page 3-15.

NCHAR and NLS Parameters and Compatibility

You should set NLS_LANG to your environment, as follows:

- set ORA_NLS32 for 7.3.x environment
- set ORA_NLS33 for 8.0 environment

Verify that the client side has the correct NLS character set environment variables. An error is generated when Oracle 7.3 NLS code tries to load a version 8 character set.

Pre-Version 8 Parameters Renamed in Version 8

The following initialization parameters have been renamed in version 8:

Pre-Version 8 Name	Version 8 Name
ASYNC_READ	DISK_ASYNCH_IO
ASYNC_WRITE	DISK_ASYNCH_IO
DB_FILE_STANDBY_NAME_CONVERT	DB_FILE_NAME_CONVERT
DB_WRITERS	DB_WRITER_PROCESSES
LOG_FILE_STANDBY_NAME_CONVERT	LOG_FILE_NAME_CONVERT
SNAPSHOT_REFRESH_INTERVAL	JOB_QUEUE_INTERVAL
SNAPSHOT_REFRESH_PROCESS	JOB_QUEUE_PROCESSES
USE-ASYNC_IO	DISK_ASYNCH_IO

See Also: *Oracle8 Reference* for more information about initialization parameters.

Note: Some of the parameters listed above are platform-specific. See your platform-specific Oracle documentation for more information about renamed initialization parameters on your platform.

Release 7.3 Parameters Obsolete in Version 8

The following parameters are obsolete in version 8:

CCF_IO_SIZE

CHECKPOINT_PROCESS

GC_DB_LOCKS

GC_FREELIST_GROUPS

GC_ROLLBACK_SEGMENTS

GC_SAVE_ROLLBACK_LOCKS

GC_SEGMENTS

GC_TABLESPACES

IO_TIMEOUT

INIT_SQL_FILES

IPQ_ADDRESS

IPQ_NET

LM_DOMAINS

LM_NON_FAULT_TOLERANT

OPTIMIZER_PARALLEL_PASS

PARALLEL_DEFAUILT_MAX_SCANS

PARALLEL_DEFAULT_SCAN_SIZE

SEQUENCE_CACHE_HASH_BUCKETS

SERIALIZABLE

SESSION_CACHED_CURSORS

UNLIMITED_ROLLBACK_SEGMENTS

USE_IPQ

USE_READV

USE_SIGIO

V733PLANS_ENABLED

See Also: *Oracle8 Reference* for more information about initialization parameters.

Note: Some of the parameters listed in this section are platform-specific. See your platform-specific Oracle documentation for more information about obsolete initialization parameters on your platform.

REPLICATION_DEPENDENCY_TRACKING for the Replication Server

Use the REPLICATION_DEPENDENCY_TRACKING parameter for turning DSCN tracking on or off.

TRUE, the default value, turns dependency tracking ON for read / write operations to the database. Dependency tracking is essential for the Replication Server to propagate changes in parallel.

FALSE turns dependency tracking OFF. This condition allows read / write operations to the database to run faster, but it does not produce dependency information for the Replication Server to perform parallel propagations.

WARNING: If the application performs a read/write operation to any replicated table, do not set REPLICATION_DEPENDENCY_TRACKING to FALSE.

Features No Longer Supported in Version 8

This section identifies features associated with INIT.ORA file parameters from previous Oracle versions / releases that are obsolete in version 8. Oracle support for these items will likely be dropped in future releases; therefore, they should be considered desupported in version 8.

SERIALIZABLE=TRUE or _SERIALIZABLE

The INIT.ORA file parameter SERIALIZABLE=TRUE or _SERIALIZABLE implies automatic table-level locking ON READ in version 8. It should not be used in any version 8 installation and will probably be permanently deleted from the maintenance release after 8.1.

D

New SQL Key and Reserved Words

This appendix lists several new keywords and one new reserved word added to version 8.

Problems can arise if these special words are used as names for database objects (tables, columns, etc.). In addition, you should avoid naming your tables or columns using terms that are reserved by any of the languages or utilities you are likely to use at your installation. Refer to the various language and reference manuals and to this appendix for lists of reserved words.

Consult *Oracle8 SQL Reference* for the words that are reserved by SQL. Tables or columns that have these names also must be specified in double quotation marks. The special characters, reserved words, and keywords new to version 8 are listed below. All new version 8 words are keywords except for one reserved word (VALUE):

ACCOUNT
ARRAY
BFILE
BLOB
CAST
CFILE
CHAR_CS
CHUNK
CLOB
CLONE
DANGLING
NCLOB
NESTED
NOLOGGING
NOPARALLEL
NOREVERSE
NORMAL
NVARCHAR2
OBJECT
OBJNO_REUSE
OID
OIDINDEX

DATAOBJNO
DEFERRABLE
DEFERRED
DEREF
DIRECTORY
ENFORCE
EXCHANGE
EXPIRE
EXTENT
FLOB
GLOBALLY
HASH
HASHKEYS
HEAP
IDGENERATORS
INITIALLY
LIBRARY
LOCKED
LOGGING
LOGICAL_READS_PER_CALL
LOGICAL_READS_PER_SESSION
MASTER
NATIONAL
NCHAR
NCHAR_CS
ORGANIZATION
OVERFLOW
PASSWORD
PCTTHRESHOLD
PCTVERSION
PRESERVE
PURGE
QUEUE
REF
REPLACE
RETURN
RETURNING
REVERSE
SCOPE
SEG_BLOCK
SEG_FILE
SKIP
SYS_OP_NTCIMG
THAN
THE
TOPLEVEL
UNLOCK
USAGE
VALUE (Reserved Word)
VARYING

E

General System Requirements for Migration

This appendix covers the system requirements for successfully migrating, upgrading, and downgrading an Oracle RDBMS. Migrating a database from version 7 (or version 6, by using Export/Import) to version 8 requires some specific configurations of your operating system and hardware.

The following topics are covered in this appendix:

- Memory Requirements
- Using Oracle Parallel Server
- Version 8 New Sizes and Limits

Memory Requirements

Version 8 memory requirements include basic memory requirements, memory needed for executables, and amount of concurrent access.

Basic Memory Requirements

Version 8 requires at least 48 megabytes of RAM to run a database; this is a minimum configuration without much operating system memory swapping to disk. To support connections to multiple users, the system should have additional memory. Concurrent use of the Enterprise Manager requires an additional 20MB.

> **See Also:** Your platform-specific Oracle documentation for more information about memory requirements on your platform.

Version 8 Executables

The version 8 executables are three times larger than version 7 executables, primarily because of new version 8 features. If you have a typical 3-megabyte set of version 7 executables, the space required for these executables upon migration to version 8 is approximately 9 megabytes. This threefold increase can require special attention on large batch systems (which may generate dozens or hundreds of executables). The space required for executables also depends on the options you choose for the version 8 environment, such as:

- Oracle Parallel Server (see *Oracle8 Parallel Server Concepts and Administration*)
- Net8 or SQL*Net use (see *Oracle Net8 Administrator's Guide*)

Make sure you adjust system memory to accommodate the inclusion of options when you are migrating from version 7 to version 8.

Note: 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 new functionality in version 8.

Concurrent Access

The memory size of a version 8 system depends on concurrent access and the way in which concurrent access is accomplished. Version 8 supports the following connect options:

Option 1: Use local connections in dedicated server architecture (also called "two-task Oracle"). This option is the same as version 7.

Option 2: Use remote connections through SQL*Net. This option is the same as version 7.

Option 3: Use multithreaded shared servers for local and remote connections.

Option 4: Use transaction processor (TP) monitors.

Option 1 requires more memory than Option 2 or Option 3. With Option 1, if both 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 version 8 results in 100 additional Oracle server processes on the system, totaling 200 in all.

With **Option 2**, only the Oracle processes reside on the system, and the client processes are connected remotely. Thus, you need to consider only to the size of the Oracle server processes and the size of the available shared memory.

Option 3, using multithreaded server architecture, is the same as version 7. The multithreaded server feature allows the processes of several local or remote client processes to connect to a single dispatcher process, instead of having a dedicated Oracle shadow process. While not designed as a performance enhancement, multithreaded server configuration allows more concurrent connections on a version 8 server, thereby improving throughput. Multiple clients can connect to a single dispatcher, so the memory utilization for concurrent user connections decreases. For further information on the multithreaded server feature of version 8, refer to *Oracle8 Concepts*.

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.

Note: The requirements for using TP monitors vary greatly and are beyond the scope of this manual. Please consult the appropriate TP monitor vendor for system requirements.

In summary, you can estimate system memory requirements, for a single system, by considering the following factors:

- 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

Using Oracle Parallel Server

Migration of Oracle Parallel Server (including DLM) requires that *each node* have its own copy of version 7 and version 8 software. Many platforms, including IBM RS6000, NCR 3XXX series, Pyramid MIS Server, Sequent Symmetry, and Sun SPARCcenter may require additional private disks.

Version 8 New Sizes and Limits

Version 8 increases various capacity limits, as shown in the following lists:

Datatypes:

- CHAR increased to 2000 bytes.
- VARCHAR/VARCHAR2 increased to 4000 bytes.
- NCHAR holds up to 2000 bytes.
- NCHAR VARYING holds up to 4000 bytes.

Columns and indexes:

- Database size is limited only by capacity of the platform.
- Maximum number of datafiles per tablespace is 1022.
- Maximum number of columns in a table increased to 1000.
- Primary key is limited to 16 fields.

CHAR and NCHAR Maximum Size Support

Version 8 supports increased maximum string lengths for several datatypes:

- CHAR maximum length of 2000 bytes (also for RAW datatype)
  (Oracle 7.3 supports CHAR and RAW length maxima of 255 bytes)
- VARCHAR2 maximum length of 4000 bytes
  (Oracle 7.3 supports a VARCHAR2 length maximum of 2000 bytes)

The version 8 datatype string length maxima accord with the NTT/MIA specification, which requires the following string lengths:

- CHAR maximum length of at least 255 bytes
- NCHAR maximum length of 127 characters
- NCHAR VARYING (NVARCHAR) max length of 2000 characters
- VARCHAR2 maximum length of 4000 bytes

Index

A

B

C

D

E

F

G

I

K

L

M

N

O

P

R

S

T

U

V

W

X