

Oracle8*i*

Migration

Release 8.1.5

February 1999

Part Number A67774-01

ORACLE[®]

Oracle8i Migration, Release 8.1.5

Part Number A67774-01

Copyright © 1996, 1999, Oracle Corporation. All rights reserved.

Primary Author: Randy Urbano

Graphic Artist: Valarie Moore

Contributors: Nipun Agarwal, Karleen Aghevli, Reema Al-Shaikh, Rick Anderson, Vikas Arora, Neerja Bhatt, Bill Bridge, Thong Bui, Sashi Chandrasekran, Ben Chang, Debashish Chatterjee, Lakshminaray Chidambaran, Maria Chien, Eugene Chong, George Claborn, David Colello, Michael Depledge, Alan Downing, Sandy Dreskin, Sreenivas Gollapudi, Brajesh Goyal, Miranda Gresham, Terry Hart, Michael Hartstein, Jeffrey Hebert, Bhaskar Himatsingka, Thuvan Hoang, Alison Holloway, Chin Hong, Wei Huang, Nancy Ikeda, Pavana Jain, Robert Jenkins, Sanjeev Jhala, Maura Joglekar, Christopher Jones, Sanjay Kaluskar, Dhiraj Kapoor, Vishwanath Karra, Susan Kotsovolos, Viswanathan Krishnamurthy, Muralidhar Krishnaprasad, Janaki Krishnaswamy, Andre Kruglikov, Thomas Kurian, Paul Lane, Gordon Larimer, Lefty Leverenz, Jing Liu, Juan Loaiza, Neil Le, J. Bill Lee, Tracy Lee, Bill Maimone, Ethan Malasky, Raghu Mani, Shailendra Mishra, Ari Mozes, Kannan Muthukkaruppan, Subramanian Muralidhar, Ravi Murthy, Karuna Muthiah, Anil Nori, Peter Ogilvie, Irene Paradisis, Rosanne Park, Joan Pearson, Elizabeth Pitt, Greg Pongracz, Lois Price, Franco Putzolu, Anil Ramdin, N. C. Ramesh, Paul Raveling, Ann Rhee, Mary Rhodes, Anindo Roy, Usha Sangam, Richard Sarwal, Ashok Saxena, Ajay Sethi, Carol Sexton, Franz Spickhoff, James Stamos, Debbie Steiner, Harry Sun, Katia Tarkhanov, Juan Tellez, Alvin To, Alex Tsukerman, Douglas Utzig, Peter Vasterd, Guhan Viswanathan, Rahim Yaseen, Steven Wertheimer, Rick Wessman, Andrew Witkowski, Lik Wong, Aravind Yalamanchi, Qin Yu

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

The Programs (which include both the software and documentation) contain proprietary information of Oracle Corporation; they are provided under a license agreement containing restrictions on use and disclosure and are also protected by copyright, patent, and other intellectual and industrial property laws. Reverse engineering, disassembly, or decompilation of the Programs is prohibited.

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

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

Restricted Rights Notice Programs delivered subject to the DOD FAR Supplement are "commercial computer software" and use, duplication, and disclosure of the Programs including documentation, shall be subject to the licensing restrictions set forth in the applicable Oracle license agreement. Otherwise, Programs delivered subject to the Federal Acquisition Regulations are "restricted computer software" and use, duplication, and disclosure of the Programs shall be subject to the restrictions in FAR 52.227-19, Commercial Computer Software - Restricted Rights (June, 1987). Oracle Corporation, 500 Oracle Parkway, Redwood City, CA 94065.

Oracle is a registered trademark, and Oracle8i, Pro*COBOL, Oracle Parallel Server, SQL*Forms, SQL*Loader, SQL*Module, SQL*Net, SQL*Plus, Advanced Replication Option, Developer/2000, Enterprise Manager, Net8, Oracle7, Oracle7 Server, Oracle8, Oracle Call Interface, Oracle Universal Installer, Oracle Data Migration Assistant, Server Manager, Pro*Ada, Pro*C, Pro*C/C++, Trusted Oracle, and PL/SQL are trademarks or registered trademarks of Oracle Corporation. All other company or product names mentioned are used for identification purposes only and may be trademarks of their respective owners.

Contents

Send Us Your Comments	xiii
Preface.....	xv
1 Overview	
Terminology	1-2
Running Scripts	1-3
Changing Word-Size	1-4
Using Optimal Flexible Architecture (OFA).....	1-5
Rolling Upgrades for Oracle Parallel Server.....	1-5
Deinstalling Options.....	1-6
Overview of Migration Steps	1-6
Step 1: Prepare to Migrate	1-8
Step 2: Test the Migration Process	1-8
Step 3: Test the Migrated Test Database	1-8
Step 4: Prepare and Preserve the Source Database.....	1-9
Step 5: Migrate the Production Database.....	1-9
Step 6: Tune and Adjust the New Production Database	1-9
Role of the Database Administrator During Migration.....	1-10
Role of the Application Developer During Migration.....	1-10

2 Preparing to Migrate

Prepare to Migrate	2-2
Become Familiar with the Features of the New Database	2-2
Choose a Migration Method	2-3
Assess System Requirements vs. Resources Available	2-15
Choose an Oracle Home Directory for the New Release.....	2-18
Avoid Common Migration Problems	2-18
Prepare a Backup Strategy.....	2-18
Develop a Testing Plan	2-19
Test the Migration Process	2-22
Test the Migrated Test Database	2-23

3 Migrating Using the Migration Utility

Documentation Roadmap for Using the Migration Utility	3-2
Overview of Migration Using the Migration Utility	3-3
Outline of the Migration Process.....	3-3
Using the Migration Utility	3-5
System Considerations and Requirements	3-6
Space Requirements	3-6
Block Size Considerations.....	3-6
Considerations for Replication Environments	3-7
Migrating a System with Oracle Parallel Server Installed.....	3-7
Migrating to a Different Operating System	3-7
Character Set Considerations.....	3-8
Prepare the Oracle7 Source Database for Migration	3-9
Install the Release 8.1 Oracle Software	3-12
Review Migration Utility Command-Line Options	3-14
Migrate the Oracle7 Source Database	3-15
Prepare the Oracle7 Environment for Migration	3-15
Migration Steps in the Oracle7 Environment	3-16
Preserve the Oracle7 Source Database	3-20
Migration Steps in the Oracle8i Environment	3-21
Troubleshooting Errors During Migration	3-30
Abandoning the Migration	3-30

4	Migrating Using the Oracle Data Migration Assistant	
	Documentation Roadmap for Using the Oracle Data Migration Assistant	4-2
	Overview of Migration Using the Oracle Data Migration Assistant	4-3
	Start with an Oracle7 Database Supported by the Oracle Data Migration Assistant.....	4-3
	Downgrading.....	4-3
	System Considerations and Requirements	4-4
	Space Requirements	4-4
	Block Size Considerations	4-5
	Considerations for Replication Environments.....	4-5
	Migrating to a Different Operating System.....	4-5
	Character Set Considerations.....	4-6
	Prepare the Oracle7 Source Database for Migration	4-7
	Install the Release 8.1 Oracle Software and Migrate the Database	4-11
	Running the Oracle Data Migration Assistant Independently	4-16
	Finish the Migration	4-17
	Troubleshooting Errors During Migration	4-20
	Abandoning the Migration	4-20
5	Migrating Using Export/Import	
	Basics of Export/Import	5-2
	Export Utility Requirements.....	5-2
	Import Requirements.....	5-2
	Additional Options.....	5-3
	Migrate the Pre-Release 8.0 Source Database Using Export/Import	5-3
6	After Migrating the Database	
	Back Up the Migrated Database	6-2
	Check for Bad Date Constraints.....	6-2
	Rebuild Unusable Bitmap Indexes	6-3
	Avoid Problems with Parallel Execution	6-3
	Migrate Partition Views to Partition Tables	6-4
	Change the Password for the OUTLN User.....	6-4
	Migrate or Upgrade to the New Release of Net8 (Optional).....	6-4
	Modify Your listener.ora File.....	6-5

Test the Database and Compare Results	6-6
Tune the Migrated Database	6-6
Add New Features as Appropriate	6-7
Develop New Administrative Procedures as Needed	6-7

7 Upgrading to the New Oracle8i Release

Upgrade Paths	7-2
Upgrading the Database to the New Oracle8i Release	7-3
Prepare to Upgrade	7-3
Upgrade the Database	7-6
Upgrading Specific Components	7-21
Upgrading Advanced Replication	7-21
Upgrading Oracle Parallel Server	7-24
Upgrading Snapshots	7-25
Upgrading the Advanced Queuing Option	7-26
Upgrading User-Defined Datatypes	7-30
Upgrading the Recovery Catalog	7-30
Recompiling Invalid PL/SQL Modules	7-32
After Upgrading to the New Release	7-33
Using the New TO_LOB Operator	7-33
Checking for Bad Date Constraints	7-33
Avoiding Problems with Parallel Execution	7-34
Adjusting Your INITSID.ORA File for the New Release	7-34
Changing the Password for the OUTLN User	7-35
Modify Your listener.ora File	7-35
Drop Java Objects	7-36
Changing the Word-Size of Your Current Release	7-37

8 Compatibility and Interoperability

What Is Compatibility?	8-2
The COMPATIBLE Parameter	8-2
Features Requiring 8.1.0 or Higher Compatibility Level	8-9
Applications	8-9
Tablespaces	8-9
Schema Objects	8-10

Partitioning	8-11
Built-In Datatypes.....	8-11
User-Defined Datatypes	8-12
Oracle Parallel Server.....	8-12
Data Protection.....	8-13
Distributed Databases.....	8-13
Data Warehousing.....	8-13
Data Access.....	8-14
Spatial and Visual Information.....	8-15
What Is Interoperability?	8-15
Compatibility and Interoperability Issues	8-16
Applications	8-16
Startup and Shutdown.....	8-22
Tablespaces and Datafiles	8-23
Data Dictionary	8-24
Schema Objects	8-25
Datatypes	8-25
User-Defined Datatypes	8-29
SQL and PL/SQL.....	8-30
Advanced Queuing (AQ)	8-32
Procedures and Packages	8-33
Oracle Optimizer	8-34
Oracle Parallel Server.....	8-34
Database Security	8-37
Database Backup and Recovery	8-39
Distributed Databases.....	8-44
SQL*Net or Net8.....	8-46
Export/Import	8-48
Miscellaneous Compatibility and Interoperability Issues.....	8-50

9 Upgrading Your Applications

Overview of Upgrading Applications to Oracle8i	9-2
Upgrading OCI and Precompiler Applications	9-2
Upgrading OCI Applications.....	9-3
Upgrading Precompiler Applications	9-4

Upgrading SQL*Plus Scripts	9-6
Upgrading Oracle7 Forms or Developer/2000 Applications.....	9-6
Copying LONGs to LOBs.....	9-7

10 Migrating from Server Manager to SQL*Plus

Startup Differences	10-2
Starting Server Manager	10-2
Starting SQL*Plus	10-2
Commands	10-3
New SQL*Plus Release 8.1 Commands.....	10-3
Commands Common to Server Manager and SQL*Plus.....	10-5
SQL*Plus Equivalents for Server Manager Commands.....	10-6
Possible Differences in the SET TIMING Command.....	10-7
Server Manager Commands Unavailable in SQL*Plus.....	10-7
Syntax Differences	10-7
Comments	10-7
Blank Lines.....	10-10
The Hyphen Continuation Character	10-10
Ampersands.....	10-12
CREATE TYPE and CREATE LIBRARY Commands.....	10-13
COMMIT Command.....	10-14

11 Migration Issues for Physical Rowids

Migrating Applications and Data	11-2
The DBMS_ROWID Package	11-3
Rowid Conversion Types	11-3
Rowid Conversion Functions.....	11-4
Conversion Procedure Examples	11-5
Example 1	11-5
Example 2.....	11-6
Example 3.....	11-6
Example 4.....	11-6
Example 5.....	11-6

Snapshot Refresh	11-6
Pre-Version 8 Client Compatibility Issues	11-7
Rowid-Related Migration Questions and Answers	11-7

12 Downgrading to an Older Version 8 Release

Perform a Full Offline Backup	12-2
Remove Incompatibilities	12-2
Tablespaces	12-5
Schema Objects	12-6
Partitioning	12-14
Datatypes	12-17
User-Defined Datatypes	12-21
SQL and PL/SQL.....	12-25
Java.....	12-26
Advanced Queuing (AQ)	12-27
Procedures and Packages	12-33
Constraints and Triggers	12-34
Oracle Optimizer	12-35
Security.....	12-36
Database Backup and Recovery	12-37
Distributed Databases.....	12-38
Net8.....	12-41
Reset Database Compatibility	12-42
Downgrade the Database	12-43
Regenerating Advanced Replication Support.....	12-51
Re-Installing the UTL_REF Package on Release 8.0.4	12-51
Re-Installing Recovery Manager Packages on Release 8.0.3	12-52

13 Downgrading to Oracle7

Overview of Downgrading from Oracle8i to Oracle7	13-2
Downgrading a Database That Does Not Contain New or Changed Data	13-2
Downgrading a Database That Contains New or Changed Data	13-3
Alternative Downgrading Methods	13-4

A Troubleshooting Migration Problems

Problems Using the Migration Utility or Oracle Data Migration Assistant	A-2
General Migration Problems	A-2
Migration Utility Errors	A-5
Problems at the ALTER DATABASE CONVERT Command	A-16
Oracle7 Control Files Exist	A-16
Database Started in Mode Other Than NOMOUNT	A-17
Convert File Not Found	A-17
REMOTE_LOGIN_PASSWORDFILE Initialization Parameter Set to EXCLUSIVE	A-18
Database Name Mismatch	A-19
Rerunning the ALTER DATABASE CONVERT Command	A-19
Datafile Version Integrity Problem	A-20

B Changes to Initialization Parameters

Initialization Parameters Added in Version 8	B-2
Initialization Parameters Added in Release 8.0	B-2
Initialization Parameters Added in Release 8.1	B-3
Initialization Parameters Renamed in Version 8	B-4
Initialization Parameters Renamed in Release 8.0	B-4
Initialization Parameters Renamed in Release 8.1.4	B-5
Initialization Parameters Renamed in Release 8.1.5	B-5
Initialization Parameters Obsolete in Version 8	B-6
Initialization Parameters Obsolete in Release 8.0	B-6
Initialization Parameters Obsolete in Release 8.1	B-6
Compatibility Issues with Initialization Parameters	B-8
New Default Value for LOG_CHECKPOINT_TIMEOUT	B-8
Data Dictionary Protection	B-8
The DML_LOCKS Parameter	B-8
The DB_DOMAIN Parameter	B-9
Parallel Execution Allocated from Large Pool	B-9
Archive Log Destination Parameters	B-13

C Changes to Static Data Dictionary Views

Static Data Dictionary Views Added in Version 8	C-2
Static Data Dictionary Views Added in Release 8.0	C-2
Static Data Dictionary Views Added in Release 8.1	C-4
Static Data Dictionary Views with Added Columns in Version 8	C-7
Static Data Dictionary Views with Added Columns in Release 8.0	C-7
Static Data Dictionary Views with Added Columns in Release 8.1	C-9
Static Data Dictionary Views with Dropped Columns in Version 8	C-11
Static Data Dictionary Views with Dropped Columns in Release 8.0	C-11
Static Data Dictionary Views with Dropped Columns in Release 8.1	C-12
Static Data Dictionary Views with Renamed Columns in Version 8	C-13
Static Data Dictionary Views with Renamed Columns in Release 8.0	C-13
Static Data Dictionary Views with Columns That May Return Nulls	C-14
Static Data Dictionary Views Obsolete in Version 8	C-15
Static Data Dictionary Views Obsolete in Release 8.0	C-15
Static Data Dictionary Views Obsolete in Release 8.1	C-15

D Changes to Dynamic Performance Views

Dynamic Performance Views Added in Version 8	D-2
Dynamic Performance Views Added in Release 8.0	D-2
Dynamic Performance Views Added in Release 8.1	D-5
Dynamic Performance Views Renamed in Version 8	D-7
Dynamic Performance Views Renamed in Release 8.1	D-7
Dynamic Performance Views with Added Columns in Version 8	D-8
Dynamic Performance Views with Added Columns in Release 8.0	D-8
Dynamic Performance Views with Added Columns in Release 8.1	D-9
Dynamic Performance Views with Dropped Columns in Release 8.1	D-10
Dynamic Performance Views with Dropped Columns in Release 8.1	D-10
Dynamic Performance Views Obsolete in Version 8	D-11
Dynamic Performance Views Obsolete in Release 8.1	D-11
Date Columns in Dynamic Performance Views	D-11

E New Internal Datatypes and SQL Functions

Internal Datatypes Added in the New Release	E-2
Internal Datatypes Added in Release 8.0	E-2
Internal Datatype Added in Release 8.1	E-2
SQL Functions Added in the New Release	E-3
SQL Functions Added in Release 8.0	E-3
SQL Functions Added in Release 8.1	E-4

Index

Send Us Your Comments

Oracle8i Migration, Release 8.1.5

Part Number A67774-01

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 the Information Development department in the following ways:

- Electronic Mail - infodev@us.oracle.com
- Fax - 650.506.7228. 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 on the Oracle database system. It describes migrating using the following methods: Migration utility, Oracle Data Migration Assistant, export/import, and data copying.

In addition, this manual provides information about compatibility, about upgrading applications to the current release, and about important changes in the current release of Oracle, such as initialization parameter changes and data dictionary changes.

The following topics are covered in this preface:

- [Audience and Assumed Knowledge](#)
- [How Oracle8i Migration is Organized](#)
- [Conventions Used in This Manual](#)
- [Your Comments Are Welcome](#)

Oracle8i Migration contains information that describes the features and functionality of the Oracle8i and the Oracle8i Enterprise Edition products. Oracle8i and the Oracle8i 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 partitioning, you must have the Enterprise Edition and the partitioning option.

See Also: *Getting to Know Oracle8i* for information about the differences between Oracle8i and the Oracle8i Enterprise Edition and the features and options that are available to you.

Audience and Assumed Knowledge

This manual is for database administrators (DBAs), application developers, security administrators, system operators, and anyone who plans or executes migration, upgrade, or downgrade operations on Oracle software. It is assumed that users of this manual are familiar with their current release of the Oracle server and with their operating system environment. It also is assumed that users are familiar with Oracle database management system (DBMS) concepts. The first chapter of *Oracle8i Concepts* provides a comprehensive introduction to the concepts and terminology used in this manual.

How *Oracle8i Migration* is Organized

This manual contains the following chapters and appendices:

Chapter 1: [Overview](#)

Contains definitions for the terms used throughout this manual. This chapter also summarizes migration procedures and the responsibilities of database administrators and application developers.

Chapter 2: [Preparing to Migrate](#)

Describes the steps to complete before migrating the database.

Chapter 3: [Migrating Using the Migration Utility](#)

Provides step-by-step instructions for using the Migration utility to migrate an Oracle7 database to Oracle8i.

Chapter 4: [Migrating Using the Oracle Data Migration Assistant](#)

Provides step-by-step instructions for using the Oracle Data Migration Assistant to migrate an Oracle7 database to Oracle8i.

Chapter 5: [Migrating Using Export/Import](#)

Describes how to migrate an Oracle7 or version 6 database to Oracle8i using the Export and Import utilities.

Chapter 6: [After Migrating the Database](#)

Describes the actions to complete after migrating the database to Oracle8i.

Chapter 7: Upgrading to the New Oracle8i Release

Provides step-by-step instructions for performing the following actions:

- upgrading a database from an 8.0 release or a previous 8.1 release to the new 8.1 release of Oracle
- upgrading specific components of Oracle software to the current release
- changing the word size of your database (switching between 32-bit and 64-bit software)

Chapter 8: Compatibility and Interoperability

Contains information about compatibility and interoperability between different releases of Oracle, including detailed information about the COMPATIBLE initialization parameter. This chapter also lists the release 8.1 features that require an 8.1.0 compatibility level or higher and discusses specific issues relating to compatibility and interoperability.

Chapter 9: Upgrading Your Applications

Provides general information about upgrading Oracle7 applications and tools for use with Oracle8i.

Chapter 10: Migrating from Server Manager to SQL*Plus

Describes modifying your Server Manager line mode scripts for use with SQL*Plus. Server Manager will be obsoleted in a future release of Oracle.

Chapter 11: Migration Issues for Physical Rowids

Covers issues associated with the version 8 ROWIDs, including specific information about migrating columns containing ROWIDs to version 8.

Chapter 12: Downgrading to an Older Version 8 Release

Provides instructions for downgrading a database from the new 8.1 release to a previous 8.1 release or to an 8.0 release. This chapter also includes information about removing incompatibilities with the release to which you are downgrading, and information about resetting the compatibility level of the database.

Chapter 13: Downgrading to Oracle7

Provides instructions for downgrading a database from Oracle8i to Oracle7.

Appendix A: Troubleshooting Migration Problems

Describes common migration problems and the actions required to correct these problems. In addition, this appendix lists the messages displayed by the Migration utility and Oracle Data Migration Assistant, and includes an explanation for each message. If the message is an error message, this appendix discusses probable cause(s) of the error, and suggests corrective action for the error.

Appendix B: Changes to Initialization Parameters

Lists Oracle initialization parameters that are important for migration. Specifically, this appendix describes initialization parameters that have been added, renamed, or obsoleted in version 8. In addition, this appendix describes compatibility issues relating to specific initialization parameters.

Appendix C: Changes to Static Data Dictionary Views

Lists the static data dictionary views that have been added or obsoleted. This appendix also lists the static data dictionary views with added columns, dropped columns, and renamed columns. This appendix also lists columns in static data dictionary views that may return NULLs in release 8.1 but did not return NULLs in past releases.

Appendix D: Changes to Dynamic Performance Views

Lists the dynamic performance views (V\$ views) that have been added or obsoleted. This appendix also lists the dynamic performance views with added columns and dropped columns.

Appendix E: New Internal Datatypes and SQL Functions

Lists the new internal datatypes and SQL functions.

Conventions Used in This Manual

The following conventions are used in this manual:

UPPERCASE Words Uppercase words indicate command keywords, object names, initialization parameters, static data dictionary view names, and dynamic data dictionary view names. For example:

"Oracle enables you to control the compatibility of your database with the COMPATIBLE initialization parameter."

Italicized Words Italicized words indicate the first occurrence of a term and its definition. For example:

"*Migration* is the process of transforming an installed version of an Oracle database into a later version."

Italicized words also indicate variables. For example:

"Run `uold_release.sql` where *old_release* refers to the release you had installed prior to upgrading."

Italicized words also are used for emphasis and for book titles.

Code Examples SQL, Server Manager line mode, and SQL*Plus commands and statements are displayed in a fixed-width courier font, separated from normal text, as in the following example:

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

Example statements may include punctuation, such as commas or quotation marks. All punctuation in example statements is required syntax. Depending on the application, a semicolon or other terminator may or may not be required to end a statement, and some applications require a slash (/) on the next line to end a statement.

Uppercase words in example statements indicate the keywords within 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.

Where a fixed-width courier font appears within paragraphs, it indicates a file name, script name, executable name, or directory path, as in the following example:

To identify the user-defined types at 8.1 compatibility level, run the `utlincmp.sql` script supplied with release 8.1.

Your Comments Are Welcome

We value and appreciate your comments as a user and reader of 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 or any other Oracle manual. If you do not find this form, please write your remarks to the Information Development department 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 - 650.506.7228 Attn: Server Technologies Documentation Manager

Electronic Mail - infodev@us.oracle.com

Overview

This chapter includes an overview of the major steps required to migrate a pre-release 8.0 database (such as Oracle7 or version 6) to Oracle8i. These migration procedures transform an existing pre-release 8.0 database system (including associated applications) into an Oracle8i database system. Oracle8i 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 Oracle8i functionality.

Several preparatory steps are required before you migrate the current production database. After migrating the database, you should perform several additional test steps to test the migration. Other procedures enable you to add new Oracle8i functionality to existing pre-release 8.0 applications.

This chapter also includes definitions for words used throughout this manual and information about changing the word-size of your database.

This chapter covers the following topics:

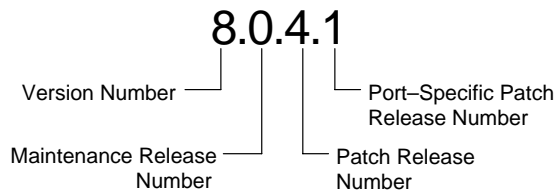
- [Terminology](#)
- [Running Scripts](#)
- [Changing Word-Size](#)
- [Using Optimal Flexible Architecture \(OFA\)](#)
- [Deinstalling Options](#)
- [Overview of Migration Steps](#)
- [Role of the Database Administrator During Migration](#)
- [Role of the Application Developer During Migration](#)

Terminology

The following terms (specified in *italic*) are used throughout this document:

The instructions in this document describe moving between different *versions* and *releases* of Oracle. [Figure 1-1](#) describes what each part of a release number represents.

Figure 1-1 *Example of an Oracle Release Number*



See Also: *Oracle8i Administrator's Guide* for more information about Oracle release numbers.

When a statement is made in this book about a version of Oracle, the statement applies to all releases in that version. References to version 8 include all releases in release 8.0 and release 8.1; references to Oracle7 include all version 7 releases, including release 7.0, 7.1, 7.2, and 7.3.

Similarly, when a statement is made in this book about a maintenance release, the statement applies to all production patch releases and port specific patch releases within that maintenance release. So, a statement about release 8.0 applies to all production releases within release 8.0, including release 8.0.3, 8.0.4, 8.0.5, and 8.0.6, but not necessarily to release 8.0.2, because that release was a beta release.

The same logic applies to patch releases. When a statement is made in this book about a patch release, the statement applies to all port-specific patch releases within that patch release. So, a statement about release 8.0.4 applies to release 8.0.4.0, 8.0.4.1, and all other port-specific patch releases within release 8.0.4.

Migration is the process of transforming an installed version of an Oracle database into a later version. For example, transforming an Oracle7 database into an Oracle8i database is migrating the database.

The *source database* is the database to be migrated to Oracle8i; during migration, the source database uses an older version of Oracle software, such as version 6 or Oracle7. The *target database* is the database into which you are migrating the source database; during migration, the target database uses new Oracle8i software.

Upgrading is the process of transforming an Oracle database from an installed release into a later release of the same version. For example, transforming patch release 8.0.3 into patch release 8.0.4 is upgrading, and transforming an 8.0 maintenance release into an 8.1 maintenance release is upgrading.

See Also: [Chapter 7, "Upgrading to the New Oracle8i Release"](#) for information about 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 8.1.5 back into release 8.0.5 is downgrading, and transforming an Oracle database from Oracle8i back into Oracle7 is downgrading.

See Also: [Chapter 12, "Downgrading to an Older Version 8 Release"](#) for information about downgrading a release 8.1 database to an earlier 8.1 release or to an 8.0 release. See [Chapter 13, "Downgrading to Oracle7"](#) for information about downgrading to Oracle7.

Running Scripts

You need to run various scripts when you perform migration, upgrade, and downgrade operations. When you run a script, the script may report "ORA-" errors. In general, you should look for errors that alert you to insufficient space, and for errors that alert you that a script failed to run. If you see these types of errors, the operation may not be completely successful. However, you typically can ignore errors about the failure to alter or drop an object that does not exist.

Changing Word-Size

You can change the word-size of your Oracle database server during a migration, upgrade, or downgrade operation. A change in word-size includes the following scenarios:

- You have 32-bit Oracle software installed on 64-bit hardware and want to change to 64-bit Oracle software.
- You have 64-bit Oracle software installed on 64-bit hardware and want to change to 32-bit Oracle software.

If you are changing word-size during a migration, upgrade, or downgrade operation, no additional action is required. The word-size is changed automatically during any of these operations. However, if you want to change the word-size within the same release, follow the instructions in ["Changing the Word-Size of Your Current Release"](#) on page 7-37. For example, if you have the 32-bit version of Oracle release 8.1.5 and you want to switch to the 64-bit version of Oracle release 8.1.5, you must complete this procedure.

The following information applies if you are upgrading or downgrading your hardware from 32-bit to 64-bit or from 64-bit to 32-bit:

- If you want to upgrade your hardware, you should be able to switch from 32-bit hardware to 64-bit hardware and still use your existing 32-bit Oracle software without encountering any problems.
- If you want to downgrade your hardware from 64-bit to 32-bit, you must first downgrade your Oracle software to 32-bit software before downgrading your hardware.

The on-disk format for database data, redo, and undo is identical for the 32-bit and 64-bit installations of Oracle. The only internal structural differences between the 32-bit and 64-bit Oracle installations are the following:

- The compiled format of PL/SQL is different. The instructions for how and when to recompile PL/SQL are provided in the appropriate chapters of this book.
- The storage format of objects is based on the release of Oracle that created the database. The existing storage format will be converted to the other format transparently when necessary.

Using Optimal Flexible Architecture (OFA)

Whether you are migrating a version 7 database or upgrading a version 8 database, Oracle Corporation recommends the Optimal Flexible Architecture (OFA) standard for your Oracle8*i* installations. The OFA standard is a set of configuration guidelines for efficient and reliable Oracle databases that require little maintenance.

OFA provides the following benefits:

- organizes large amounts of complicated software and data on disk to avoid device bottlenecks and poor performance
- facilitates routine administrative tasks, such as software and data backup functions, which are often vulnerable to data corruption
- alleviates switching among multiple Oracle databases
- adequately manages and administers database growth
- helps to eliminate fragmentation of free space in the data dictionary, isolate other fragmentation, and minimize resource contention

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

See Also: Your Oracle operating-system specific documentation for more information about OFA, and the *Oracle8i Administrator's Guide* for information about relocating your database files.

Rolling Upgrades for Oracle Parallel Server

Rolling upgrades are not supported. A rolling upgrade is one in which different instances of the same database in Oracle Parallel Server are upgraded to a new version or release of Oracle one at a time. Therefore, a rolling upgrade would result in different releases running concurrently during the upgrade process.

Note: The term rolling upgrade refers both to a migration from Oracle7 to Oracle8*i* and to an upgrade from release 8.0 to release 8.1. Rolling upgrade is not supported in either case.

Deinstalling Options

If you want to deinstall old options when you migrate or upgrade to a new release of Oracle, use the installer to deinstall them. You can deinstall them before or after you upgrade or migrate, but you must use the version of the installer that corresponds with the items you want to remove.

For example, if you are running release 8.0 of Oracle with Oracle Parallel Server installed, and you decide that you do not need this option when you upgrade to release 8.1, you should deinstall Oracle Parallel Server in one of the following ways:

- Before you upgrade to release 8.1, use the installer in your 8.0 release to deinstall Oracle Parallel Server. Then, do not install Oracle Parallel Server when you install the 8.1 release.
- When you upgrade to release 8.1, install and upgrade Oracle Parallel Server. Then, use the release 8.1 installer to deinstall Oracle Parallel Server.

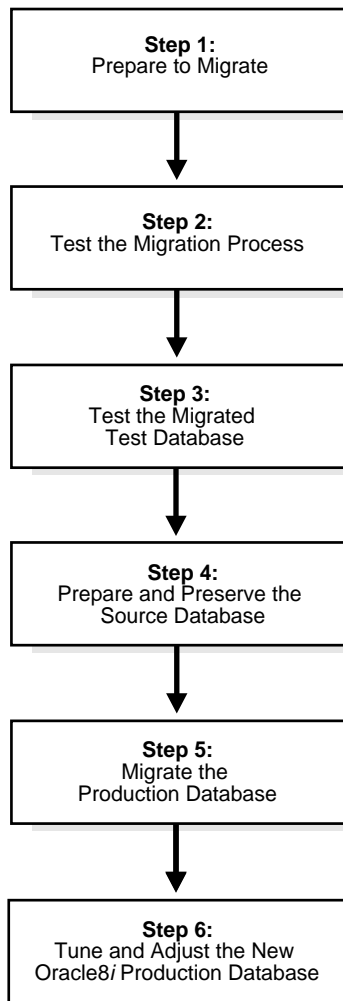
Note: After you deinstall an option, extraneous data dictionary tables may remain.

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

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 operating-system specific details identified in your operating-system specific Oracle documentation.

Note: The rest of this chapter describes migration. If you plan to perform an operation other than migration, such as upgrading or downgrading, you can proceed to the appropriate chapter for the operation.

Figure 1–2 Major Migration Steps

Careful planning and use of Oracle8i tools can ease the process of migrating a database to Oracle8i. The Oracle Data Migration Assistant is the easiest way to migrate an entire database, while the Migration utility is more complicated to use but provides more control over the process of migrating an entire database. Export/Import and SQL copy utilities enable piecemeal migration of parts of a database.

The following sections contain a brief outline of the major steps shown in [Figure 1-2](#). The purpose of these descriptions is to familiarize you with the major steps in the migration process. For detailed instructions, refer to the appropriate chapters and sections later in this book.

Step 1: Prepare to Migrate

- Become familiar with the features of the Oracle8i database. See *Getting to Know Oracle8i* for an overview of these features.
- 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.
- Estimate and secure the system resources required for the migration.
- Develop a plan for testing the migration with an Oracle8i test database and a plan for testing the migrated Oracle8i 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 an Oracle7 test database. The test migration should be conducted in an environment created for migration testing and should not interfere with the actual Oracle7 production database.

Step 3: Test the Migrated Test Database

- Perform the tests you planned in Step 1 on the pre-migration Oracle7 test database and on the Oracle7 test database that was migrated to Oracle8i.
- Compare results, noting anomalies between test results on the pre-migration Oracle7 test database and on the migrated Oracle8i database.
- Investigate ways to correct any anomalies you find and then implement the corrections.
- Repeat Step 1, Step 2, and the first parts of Step 3, as necessary, until the test 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 Oracle8i will be successful.
- Schedule the downtime required for backing up and migrating the pre-release 8.0 production database to Oracle8i.
- Perform a full backup of the current production database. This step is required only if either the Oracle Data Migration Assistant or the Migration utility is used for the migration.

Step 5: Migrate the Production Database

- Migrate the pre-release 8.0 production database to Oracle8i.
- After the migration, perform a full backup of the production database and other post-migration tasks.

[Chapter 3](#) describes Steps 4 and 5 using the Migration utility; [Chapter 4](#) describes Steps 4 and 5 using the Oracle Data Migration Assistant; and [Chapter 5](#) describes Steps 4 and 5 using the Export/Import utilities. [Chapter 6](#) describes the backup procedure after the migration and other post-migration tasks.

See Also: *Oracle8i Replication*, Appendix B, "Migration and Compatibility", if you are migrating a database system that has Advanced Replication installed.

Step 6: Tune and Adjust the New Production Database

- Tune the new Oracle8i production database. The Oracle8i production database should perform as good as, or better than, the pre-migration Oracle database. [Chapter 6](#) describes these tuning adjustments.
- Determine which new features of the Oracle8i 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 Oracle8i database until all applications have been tested and operate properly. [Chapter 9](#) describes considerations for updating applications.

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

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 the following:

- 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 Oracle7 production database
- completing the production database migration
- performing backups of the newly migrated Oracle8i production database

Users should not have access to the migrated Oracle8i 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 Oracle7 database work correctly with the migrated Oracle8i database. Application developers often test applications against the migrated Oracle8i database and decide which new features of Oracle8i should be used.

Before migrating the Oracle7 production database, the DBA or application developer should install an Oracle8i test database. Then, the application developer can test and modify the applications, if necessary, until they work with their original (or enhanced Oracle8i) functionality.

The following references provide information about identifying differences in the migrated Oracle8i database that could affect particular applications. Application developers can use these differences to guide modifications to existing applications.

- [Chapter 8, "Compatibility and Interoperability"](#) describes compatibility and interoperability issues that may result because of differences in releases of Oracle.
- [Chapter 9, "Upgrading Your Applications"](#), describes the changes required to enable existing applications (that access an Oracle7 database) to access an Oracle8i database and provides guidance for upgrading Oracle7 applications to take advantage of Oracle8i functionality.
- *Getting to Know Oracle8i* describes the new features available in Oracle8i.
- [Appendix B, "Changes to Initialization Parameters"](#) lists new, renamed, and obsolete initialization parameters in version 8.
- [Appendix C, "Changes to Static Data Dictionary Views"](#) lists new, changed, and obsolete static data dictionary views in version 8.
- [Appendix D, "Changes to Dynamic Performance Views"](#) lists new, changed, and obsolete dynamic performance views (V\$ views) in version 8.
- [Appendix E, "New Internal Datatypes and SQL Functions"](#) lists new internal datatypes and SQL functions added in version 8.
- *Oracle8i Parallel Server Concepts and Administration* and *Oracle8i SQL Reference* contain descriptions of changes and new Oracle8i functionality.
- For a pre-release 8.0 database system that has Advanced Replication installed, refer to *Oracle8i Replication*, Appendix B, "Migration and Compatibility".
- *Oracle8i Application Developer's Guide - Fundamentals*, *Oracle8i Application Developer's Guide - Large Objects (LOBs)*, and *Oracle8i Application Developer's Guide - Advanced Queuing* provide information about planning and implementing applications.

Oracle8i includes features that aid in upgrading existing applications to Oracle8i, for example:

- 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.

Preparing to Migrate

This chapter covers the steps that must be completed before you migrate a production database. This chapter covers in detail Steps 1 through 3 of the migration process, which were outlined in [Chapter 1](#).

This chapter covers the following topics:

- [Prepare to Migrate](#)
- [Test the Migration Process](#)
- [Test the Migrated Test Database](#)

The information in this chapter is generic and applies generally to Oracle7 and version 6 production databases.

See Also: *Oracle8i Replication*, Appendix B, "Migration and Compatibility", if you are migrating a database system that has Advanced Replication installed.

Prepare to Migrate

Complete the following tasks to prepare to migrate:

- [Become Familiar with the Features of the New Database](#)
- [Choose a Migration Method](#)
- [Assess System Requirements vs. Resources Available](#)
- [Choose an Oracle Home Directory for the New Release](#)
- [Avoid Common Migration Problems](#)
- [Prepare a Backup Strategy](#)
- [Develop a Testing Plan](#)

Become Familiar with the Features of the New Database

Before you plan the migration process, become familiar with the new features of the Oracle8i database. *Getting to Know Oracle8i* is a good starting point for learning the differences between release 8.1, release 8.0, and release 7.3. Also, check specific books in the Oracle documentation library to find information about new features for a certain component; for example, see *Oracle8i Parallel Server Concepts and Administration* for changes in Oracle Parallel Server.

Note: Oracle8i training classes are an excellent way to learn how to take full advantage of the functionality available with Oracle8i. Connect to the following web page for more information:

<http://education.oracle.com>

Choose a Migration Method

Choose one of the following methods to migrate your database to Oracle8i:

- Use the Migration utility to migrate an Oracle7 database to Oracle8i. See your operating-system specific Oracle documentation for information about the earliest release that the Migration utility can migrate on your operating system. For example, on some operating systems, the Migration utility can migrate only release 7.1.4 and higher databases.

The Migration utility is a command-line utility for migration of a complete database from Oracle7 to Oracle8i. It changes datafile headers but leaves actual data unchanged. It does not copy data.

- Use the Oracle Data Migration Assistant to migrate an Oracle7 database to Oracle8i. See your operating-system specific Oracle documentation for information about the earliest release that the Oracle Data Migration Assistant can migrate on your operating system. Typically, the earliest release supported by the Oracle Data Migration Assistant is the same as the earliest release supported by the Migration utility.

The Oracle Data Migration Assistant has a graphical user interface (GUI) for migration or upgrade of a complete database. It changes datafile headers but leaves actual data unchanged. It does not copy data.

- Perform a full or partial export of an Oracle7 (or version 6) source database, followed by a full or partial import into an Oracle8i target database.

Export/Import can migrate parts of the database. Export/Import leaves datafile headers and actual data unchanged, and makes a new copy of the data.

- Copy data from a source database into an Oracle8i database using the COPY command or the AS clause of the CREATE TABLE command.

Data copying can migrate parts of the database. Data copying leaves datafile headers and actual data unchanged, and makes a new copy of the data.

[Table 2-1](#) summarizes the advantages of each of these methods. [Table 2-2](#) summarizes the disadvantages of each of these methods.

Table 2–1 Advantages of Different Migration Methods

Migration Utility	Oracle Data Migration Assistant	Export/Import	Data Copying
<p>Automatic, requires minimal interaction by the DBA.</p> <p>Relatively fast, whatever the size of the database, because the data dictionary objects are the only objects that are changed.</p> <p>Imposes essentially no limit on the size of the database it can migrate.</p> <p>Requires relatively little additional disk space, when compared with other migration methods.</p> <p>Provides more control over the migration process than the Oracle Data Migration Assistant.</p>	<p>Guides you through the migration with an easy-to-use GUI.</p> <p>Automatic, requiring minimal interaction by the DBA.</p> <p>Relatively fast, whatever the size of the database, because the data dictionary objects are the only objects that are changed.</p> <p>Imposes essentially no limit on the size of the database it can migrate.</p> <p>Requires relatively little additional disk space, when compared with other migration methods.</p> <p>Can be used for release-to-release upgrades, for example, upgrading from release 8.0.5 to release 8.1.5.</p>	<p>Can migrate version 6 and Oracle7 databases to Oracle8i.</p> <p>Can migrate specific parts of a database.</p> <p>Can be used to downgrade between versions of Oracle, for example, downgrading from Oracle8i to Oracle7.</p> <p>Can be used for release-to-release upgrade or downgrade operations, for example, upgrading from 8.0.5 to 8.1.5.</p> <p>Datafiles can be defragmented, and migrated data compressed, to improve performance.</p> <p>A database can be restructured with modified or new tablespaces, or by table partitioning.</p> <p>Can be used to migrate to a different operating system and hardware platform.</p>	<p>Datafiles can be defragmented, and migrated data compacted, to improve performance.</p> <p>A database can be restructured with modified or new tablespaces.</p> <p>Can migrate version 6 or Oracle7 databases to Oracle8i.</p> <p>Can migrate specific parts of a database.</p> <p>Can be used for release-to-release upgrade or downgrade operations, for example, upgrading from release 8.0.5 to release 8.1.5.</p> <p>Can be used to migrate to a different operating system and hardware platform.</p>

Table 2–2 Disadvantages of Migration Methods

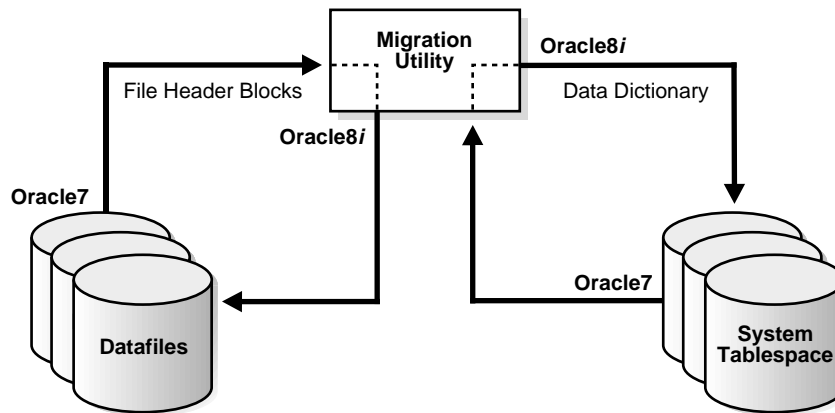
Migration Utility	Oracle Data Migration Assistant	Export/Import	Data Copying
<p>Performs only Oracle7 to Oracle8i migrations, and cannot downgrade back to Oracle7.</p> <p>Cannot perform direct migrations on release 7.0 databases, nor on databases below a specific 7.1 release. The specific 7.1 release requirement is operating-system specific.</p> <p>Cannot perform release-to-release upgrades, for example, cannot upgrade from release 8.0.5 to release 8.1.5.</p> <p>Cannot migrate selected parts of a database; migrates only the entire database.</p> <p>Cannot migrate to a different operating system or hardware platform.</p>	<p>Provides less flexibility than other methods because the migration process is highly automated. The GUI covers only the most essential migration choices.</p> <p>Provides less control over the migration than other methods.</p> <p>Cannot downgrade back to Oracle7.</p> <p>Cannot perform direct migrations on release 7.0 databases, nor on databases below a specific 7.1 release. The specific 7.1 release requirement is operating-system specific.</p> <p>Cannot migrate selected parts of a database; migrates only the entire database.</p> <p>Cannot migrate to a different operating system or hardware platform.</p> <p>Cannot migrate systems with Oracle Parallel Server installed.</p>	<p>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.</p> <p>Requires large amounts of disk space for copying data into export file(s).</p>	<p>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.</p> <p>Requires that both source and target databases be available at once during copying operations.</p>

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

Migration Utility

The Migration utility is a command-line utility that converts files and structures in the Oracle7 source database to Oracle8i 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 or content.

Figure 2–1 Migration Utility



The primary advantages of using the Migration utility are speed and relative 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 Oracle7 features, Export/Import may not be feasible, and the only practical options may be either the Migration utility or the Oracle Data Migration Assistant.

The Migration utility requires only enough temporary space in the SYSTEM tablespace to hold both the Oracle7 (source) and Oracle8i (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 Oracle7 database, you can open and access data with the Oracle7 instance. However, after the Migration utility has migrated the Oracle7 source database to Oracle8i, you can go back to Oracle7 only by restoring a full backup of the Oracle7 source database.

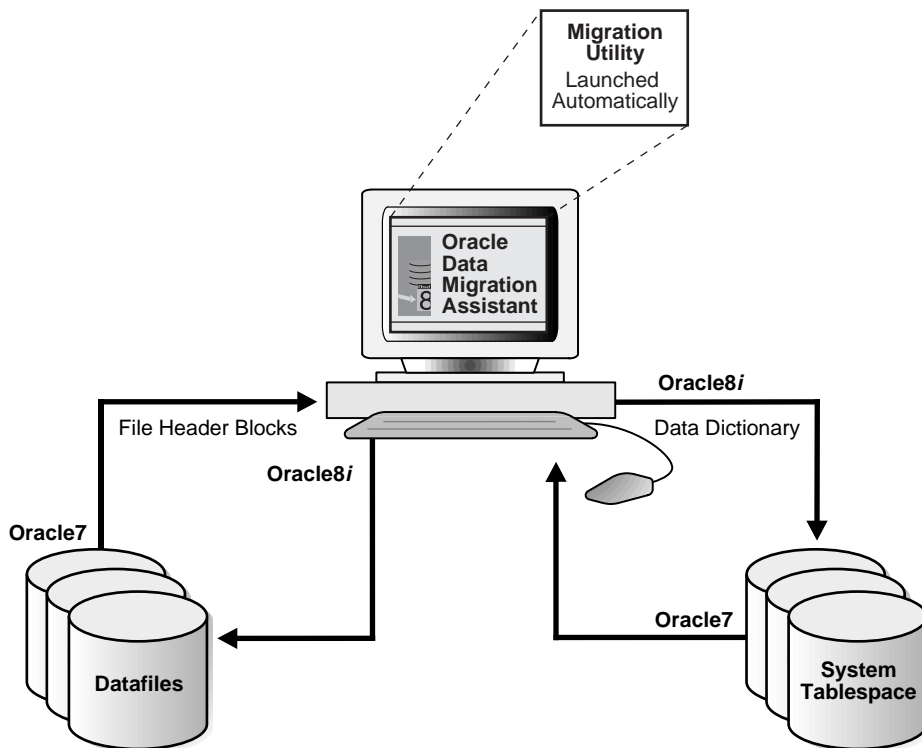
The Migration utility cannot perform direct migrations on release 7.0 databases, nor on databases below a specific 7.1 release. The specific 7.1 release requirement is operating-system specific. For example, on some operating systems, the Migration utility can migrate only release 7.1.4 and higher databases to Oracle8i. If you are using a release below the release supported by the Migration utility on your operating system, you first must migrate or upgrade your database to a supported Oracle7 release before using the Migration utility to migrate to Oracle8i. See your operating-system specific Oracle documentation for information about the earliest release supported by the Migration utility on your operating system.

See Also: [Chapter 3, "Migrating Using the Migration Utility"](#), for detailed information about using the Migration utility.

Oracle Data Migration Assistant

The Oracle Data Migration Assistant provides a user-friendly, graphical user interface (GUI) that guides you through the migration process. The Oracle Data Migration Assistant calls the Migration utility and runs it in the background, which means that you avoid running the Migration utility manually from a command-line.

Figure 2-2 Oracle Data Migration Assistant



The primary advantage of the Oracle Data Migration Assistant is that it is easy to use. Because the Oracle Data Migration Assistant calls the Migration utility, most of the advantages and disadvantages of the Migration utility also apply to the Oracle Data Migration Assistant. The section "[Choosing Between the Oracle Data Migration Assistant and the Migration Utility](#)" on page 2-9 provides information about the differences between the Oracle Data Migration Assistant and the Migration utility.

See Also: [Chapter 4, "Migrating Using the Oracle Data Migration Assistant"](#), for detailed information about using the Oracle Data Migration Assistant.

Choosing Between the Oracle Data Migration Assistant and the Migration Utility

When choosing between the Oracle Data Migration Assistant and the Migration utility, consider these differences:

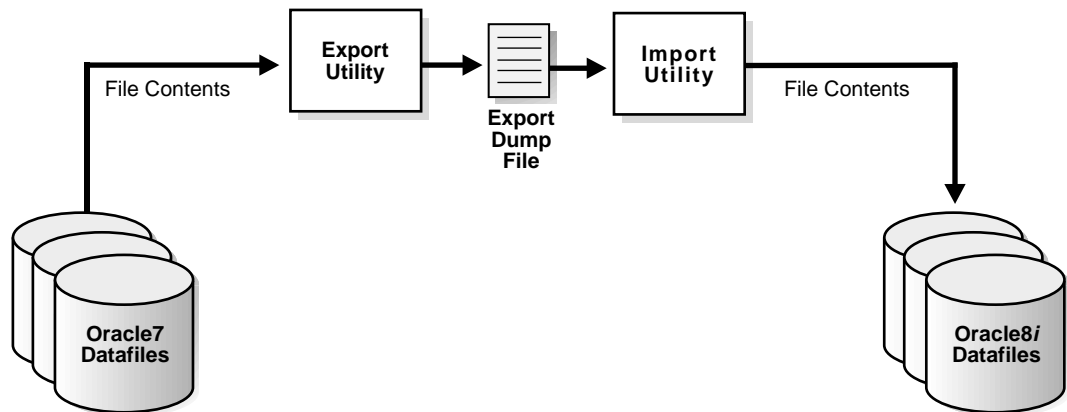
- The Oracle Data Migration Assistant provides a GUI that guides you through the migration process, while the Migration utility is a command-line utility. The Oracle Data Migration Assistant also provides extensive online help. Therefore, the Oracle Data Migration Assistant is easier to use than the Migration utility.
- The Oracle Data Migration Assistant is less flexible than the Migration utility. To avoid complexity, the Oracle Data Migration Assistant automates many of the steps in the migration process. In contrast, if you use the Migration utility, you must perform each step manually, which enables you to make adjustments during the migration process if they are necessary. Of course, the migration process usually takes longer if you use the Migration utility because you are performing more steps manually, and you must make more decisions and complete more steps during the process.
- The Oracle Data Migration Assistant performs all of the steps in a migration each time it is run. Therefore, if you have to exit the Oracle Data Migration Assistant for any reason during a migration, you must restore the backup of your Oracle7 database and start over from the beginning. You may need to exit the Oracle Data Migration Assistant if, for example, you receive an unexpected error. In contrast, if you use the Migration utility, there are several clearly-defined major steps in the process. If you have to abort a particular step, you can start at the end of the previous step, without starting over from the very beginning.
- The Oracle Data Migration Assistant automatically removes obsolete initialization parameters from the `initsid.ora` file. See "[Initialization Parameters Obsolete in Version 8](#)" on page B-6 for lists of the obsolete parameters that are removed by the Oracle Data Migration Assistant. In contrast, the Migration utility does not alter your `initsid.ora` file, and you should remove the obsolete parameters manually if you use the Migration utility.
- The Oracle Data Migration Assistant does not support migration of systems with Oracle Parallel Server installed. However, you can use the Migration utility to migrate systems with Oracle Parallel Server installed.

In general, if you prefer a graphical user interface (GUI) over a command-line interface, and you like highly automated processes with few choices, use the Oracle Data Migration Assistant. If, on the other hand, you prefer a command-line interface over a GUI, and you like to have more control over the migration process, use the Migration utility.

Export/Import

Unlike the Migration utility, the Export/Import operation physically copies 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 Oracle8i Import utility loads the exported data into the new Oracle8i database. However, the new Oracle8i target database already must exist before the export file can be migrated into it.

Figure 2-3 Export/Import



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

See Also: [Chapter 5, "Migrating Using Export/Import"](#), and also *Oracle8i 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, which enables 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, keep an existing Oracle7 production database running while the new Oracle8i 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 Oracle7 database cannot be permitted without the same changes to the data in the Oracle8i database.

The Export/Import method also can be used to upgrade or downgrade a database. For example, the transformation of an Oracle8i database back into an Oracle7 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, and the new Oracle8i software, expect changes in the following areas:

- performance
- data growth patterns
- shared resource usage
- data dictionary size
- 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(s) of the differences.

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

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

Export/Import Limitations Data migration by Export/Import has the 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.
- You may need to partition the export 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 an Oracle8i target database, you must create target data files on the system *before* you import.
- Making multiple changes to the database at the same time, such as migrating to Oracle8i and defragmenting/restructuring the database simultaneously, 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 using Export/Import can take a long time, especially compared to using the Migration utility or the Oracle Data Migration Assistant. 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, Oracle7 database was migrated to Oracle8i using the Migration utility in about an hour. The same Oracle7 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-Release 8.0 Source Database Using Export/Import](#)" on page 5-3 took two days.

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

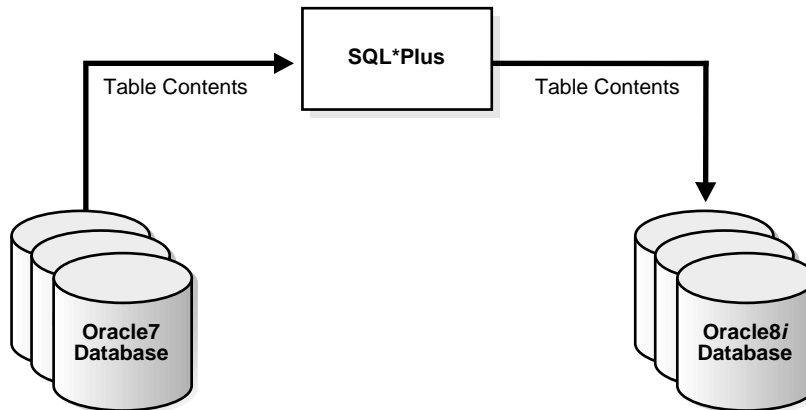
- Migration time easily exceeds the non-peak or off-production hours available in a typical daily schedule. Therefore, making the database unavailable for production tasks for the duration of the migration process might be impractical.
- If you make the source database read-only or do not allow changes 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 Oracle8i Import When importing data from an earlier version, the Oracle8i 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 statement, 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 and the CREATE TABLE ... AS statement.

Figure 2–4 Copying Data



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, enables 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 to migrate data from Oracle7 to Oracle8i, 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 and 20), you can use the following SQL statement:

```
CREATE TABLE new_emp AS
  SELECT empno, ename, job, mgr, hiredate, sal, comm, deptno
  FROM emp@v7db WHERE deptno IN (10, 20);
```

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*Plus 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 *Oracle8i 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 Oracle8i.

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 your database associated with implementing Oracle8i

Oracle8i binaries may require as much as three times the disk space required by Oracle7 binaries. 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 Oracle8i environment, such as the following:

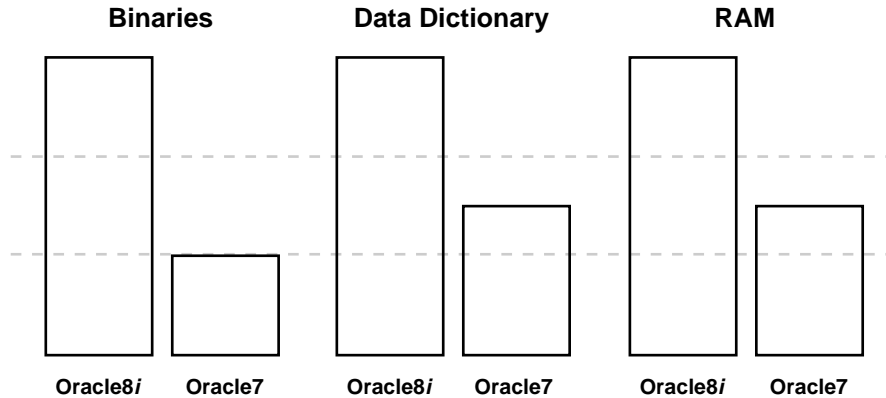
- Oracle Parallel Server (see *Oracle8i Parallel Server Concepts and Administration*)
- Net8 or SQL*Net use (see *Net8 Administrator's Guide*)

In addition, the Oracle8i data dictionary may require as much as double the space of the Oracle7 data dictionary in the SYSTEM tablespace. If you plan to use the Migration utility, you can estimate space requirements for the SYSTEM tablespace by running the Migration utility in CHECK_ONLY mode.

Also, Oracle8i may require up to twice as much RAM as Oracle7. The amount of RAM required also depends on the options you choose for the Oracle8i environment.

Figure 2–5 illustrates the differences in system requirements between Oracle7 and Oracle8i.

Figure 2–5 System Requirements for Migration



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

See Also: Your operating-system specific Oracle documentation for detailed information about system requirements.

Assess Memory Requirements for Concurrent Access

The memory size of a Oracle8i system depends on concurrent access and the way in which concurrent access is accomplished. Oracle8i supports the following connect options:

- Option 1:** Use local connections in dedicated server architecture (also called "two-task common"). Set this option as it was set in Oracle7.
- Option 2:** Use remote connections through SQL*Net. Set this option as it was set in Oracle7.
- Option 3:** Use multithreaded shared servers for local and remote connections. After migrating, set initialization parameters for this option as specified in the *Oracle8i Administrator's Guide*.
- 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 computer, memory is required for both. For example, 100 client application processes connected to Oracle8i 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, enables 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 enables more concurrent connections on an Oracle8i 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 Oracle8i, see *Oracle8i Concepts* and the *Oracle8i Administrator's Guide*.

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

Choose an Oracle Home Directory for the New Release

You must choose an Oracle home directory for the new Oracle8i release that is separate from the Oracle7 Oracle home directory. You cannot install the Oracle8i software into the same Oracle home directory that you used for Oracle7.

Using separate installation directories enables you to keep your Oracle7 software installed along with the Oracle8i software. This method enables you to test the migration process on an Oracle7 test database before replacing your production environment entirely.

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 11, "Migration Issues for Physical Rowids"](#). Because the format for rowids is different in version 8, the old rowids are invalid and must be converted.
- Make sure all Oracle product versions, operating system versions, and third-party software versions are certified against Oracle8i on your hardware platform. See your operating-system specific Oracle documentation for information.

Prepare a Backup Strategy

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

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

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

See Also: The *Oracle7 Server Administrator's Guide* for Oracle7 databases and the *Oracle8i Backup and Recovery Guide* for Oracle8i databases.

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.

CAUTION: 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, the Oracle Data Migration Assistant, 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"](#), [Chapter 4, "Migrating Using the Oracle Data Migration Assistant"](#), and [Chapter 5, "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 Oracle8i 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.

For example, if you are using Oracle Parallel Server, and you want to measure the performance gains realized from using cache fusion when you migrate to the new release, make sure you record your system's statistics before migrating. For cache fusion, record the statistics from the views `V$SYSSTAT`, `V$LOCK_ACTIVITY`, and `V$LOCK_CLASS_PING`. Doing so enables you to compare pre- and post-cache fusion performance statistics.

For best results, run the SQL scripts `utlbstat.sql` and `utlestat.sql` to collect `V$SYSSTAT` statistics for a specific period. Use a collection timeframe that most consistently reflects peak production loads with consistent transaction activity levels. To obtain data from `V$LOCK_ACTIVITY` and `V$LOCK_CLASS_PING`, use a `SELECT *` statement at the beginning and end of the statistics collection period.

Repeat this process after cache fusion is running on the new release and evaluate your system's performance as described in *Oracle8i Parallel Server Concepts and Administration*.

See Also: *Oracle8i 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 and 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 also should 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 clause 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.

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 or the Oracle Data Migration Assistant, 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 production source 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.

CAUTION: 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 Oracle8i database. Then, you can test these applications on a sample Oracle database before migrating your production database. See "[Upgrading OCI and Precompiler Applications](#)" on page 9-2 for more information.

Test the Migrated Test Database

Perform the planned tests on the source database and on the test database that you migrated to Oracle8i. Compare the results, noting anomalies. Repeat the test migration as many times as necessary.

Test the newly migrated Oracle8i test database with existing applications to verify that they operate properly with a migrated Oracle8i database. You also might test enhanced functionality by adding features that use the available Oracle8i functionality. However, first make sure that the applications operate in the same manner as they did in the source database.

See Also: [Chapter 9, "Upgrading Your Applications"](#), for more information on using applications with Oracle8i.

Migrating Using the Migration Utility

This chapter guides you through the process of migrating an Oracle7 database to Oracle8i using the Migration utility. This chapter covers the following topics:

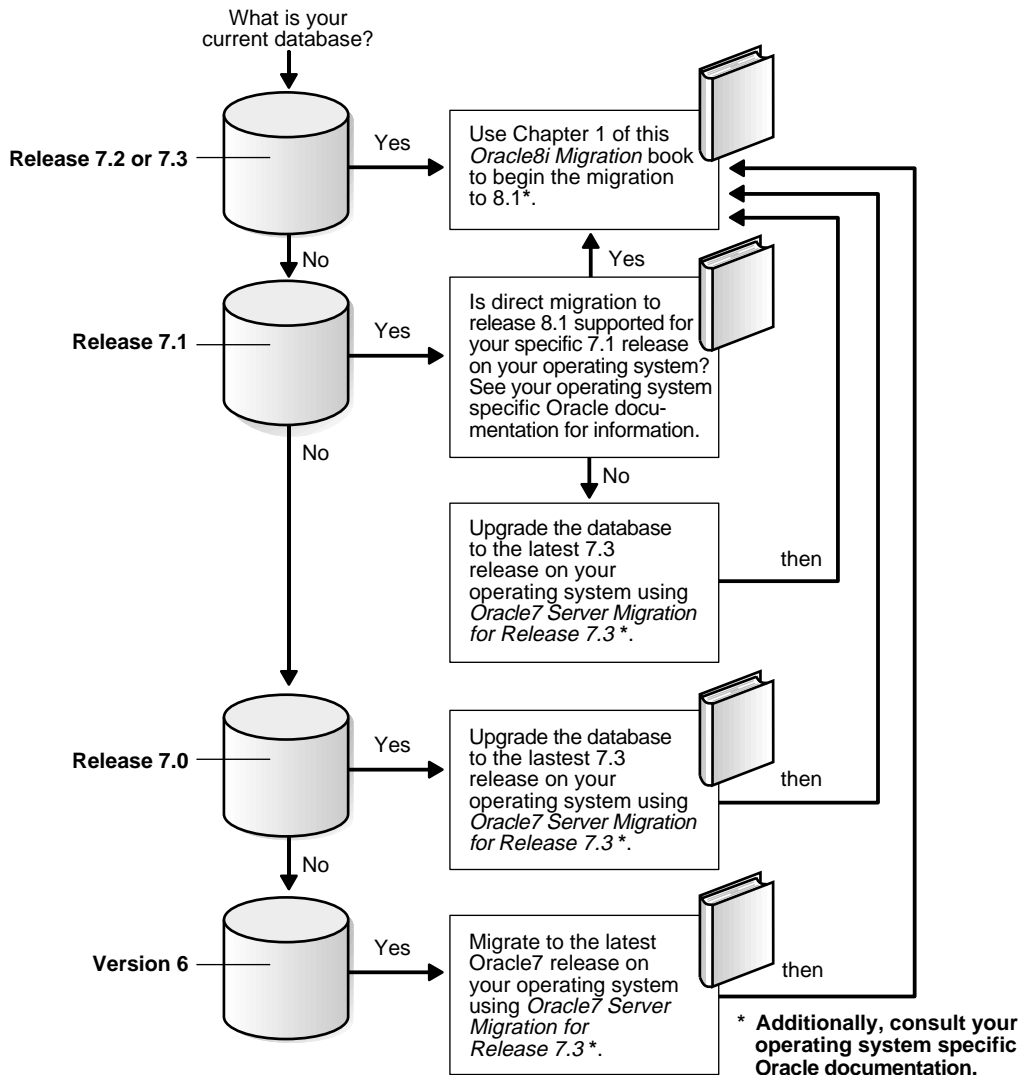
- [Documentation Roadmap for Using the Migration Utility](#)
- [Overview of Migration Using the Migration Utility](#)
- [System Considerations and Requirements](#)
- [Prepare the Oracle7 Source Database for Migration](#)
- [Install the Release 8.1 Oracle Software](#)
- [Review Migration Utility Command-Line Options](#)
- [Migrate the Oracle7 Source Database](#)
- [Troubleshooting Errors During Migration](#)
- [Abandoning the Migration](#)

See Also: Some aspects of migration are operating-system specific. See your operating-system specific Oracle documentation for additional information about migrating.

Documentation Roadmap for Using the Migration Utility

Figure 3-1 is a roadmap that specifies the documentation you should use to migrate your database to release 8.1 based on your current release of Oracle.

Figure 3-1 Documentation Roadmap for Using the Migration Utility



Overview of Migration Using the Migration Utility

Migration converts the data dictionary and structures of an Oracle7 database into Oracle8i format. To migrate the database, you first install the Oracle8i software and run the `migprep` script to copy migration files to the Oracle7 environment. Next, you run the Migration utility on the Oracle7 database. Then, you execute a series of `ALTER DATABASE` commands on the new Oracle8i database and run the `u0703040.sql` conversion script.

The completion of these procedures results in the conversion of the following Oracle7 structures into structures that can be used in Oracle8i:

- data files (file headers 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 Oracle7 Environment

- You run the Oracle8i Migration utility, which creates and populates a new data dictionary based on the data dictionary of the Oracle7 database, and also creates a binary file based on the control file of the Oracle7 database. This binary file is called the convert file.

Note: You can run the Oracle8i Migration utility multiple times (without opening the database in Oracle8i) and still be able to return to the Oracle7 database. However, running the Migration utility automatically eliminates the Oracle7 database catalog views (see "[Abandoning the Migration](#)" on page 3-30).

In the Oracle8i Environment

- You execute an ALTER DATABASE CONVERT command, which creates a new control file based on the convert file generated by the Migration utility, converts all online datafile headers to Oracle8i format, and mounts the Oracle8i 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 if and when they are made read-write sometime after migration; however, they never have to be made read-write.

- You execute an ALTER DATABASE OPEN RESETLOGS statement, which automatically converts all objects and users defined in the new dictionary to Oracle8i specifications, and converts all rollback segments to Oracle8i format.

If a source database rollback segment is in a tablespace that is offline when the Oracle8i database is opened, the rollback segment is not converted immediately to Oracle8i database format. Instead, the rollback segment is converted the first time the tablespace is brought online in Oracle8i.

- You run the database conversion scripts. The primary conversion script is the `u0703040.sql` script. This 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.

Other conversion scripts perform the necessary operations to convert specific components to the current release. For example, the `catrep.sql` script is one of the conversion scripts for Advance Replication.

Using the Migration Utility

This section contains important considerations for using the Migration utility.

Start with an Oracle7 Database Supported by the Migration Utility

A version 6 database must be migrated to at least Oracle7 before it can be migrated to Oracle8i. Also, the Migration utility cannot migrate some Oracle7 releases. See your operating-system specific Oracle documentation for information about the earliest release that is supported by the Migration utility on your operating system.

For example, on some operating systems, the Migration utility can migrate only release 7.1.4 and higher databases, and cannot migrate a release lower than release 7.1.4 (such as release 7.0 or release 7.1.3). If your database release number is lower than the release supported by the Migration utility on your operating system, upgrade or migrate the database to the required release.

See Also: *Oracle7 Server Migration, Release 7.3* for instructions about migrating or upgrading the database to the required release. Then, use this *Oracle8i Migration* manual to migrate to Oracle8i.

Downgrading

Downgrading is the process of transforming an existing Oracle database into a previous version or release. The Migration utility *cannot* transform an Oracle8i database back into Oracle7. In some situations, you can use another facility to downgrade, such as using Export/Import, restoring from backups, and possibly using other functions.

See Also: [Chapter 12](#) and [Chapter 13](#) for information about downgrading.

System Considerations and Requirements

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

Space Requirements

Oracle8i binaries may require as much as three times the disk space required by Oracle7 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 Oracle8i data dictionary simultaneously with the existing Oracle7 data dictionary.

The space required to hold an Oracle data dictionary depends on how many objects are in the database. Typically, a new Oracle8i data dictionary requires double the space that its Oracle7 source data dictionary required. If necessary, add space to the SYSTEM tablespace.

In addition, running the conversion scripts (such as the `u0703040.sql` script) to complete the migration may require more space in the SYSTEM tablespace and in the rollback segments. Insufficient space results in "unable to extend" warning when you run a conversion script. The exact amount of space required to run the conversion scripts varies depending on the number of objects in the database. If you encounter "unable to extend" warnings when you run a conversion script, try increasing the SYSTEM tablespace and the rollback segments; then, rerun the script.

Block Size Considerations

The value of `DB_BLOCK_SIZE` (a parameter in the `init sid .ora` file) in the Oracle7 database and in the migrated Oracle8i database *must* be the same. Oracle8i requires a minimum block size of 2048 bytes (2KB). Above this amount, integer multiples of your operating system'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 Oracle8i block size setting meets the following criteria:

- Matches the Oracle7 setting.
- Is at least 2048 bytes (2KB). The Oracle8i Migration utility displays an error message if the Oracle7 block size is less than 2KB.
- Is an integer-multiple of your operating system's physical block size, preferably a multiple of 2KB.

Considerations for Replication Environments

You can migrate an Oracle7 replication environment to Oracle8i. Oracle7 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. Support for coexistence of different versions of the database is operating-system specific for Oracle Parallel Server.

See Also: *Oracle8i Replication*, Appendix B, "Migration and Compatibility", for detailed instructions about migrating systems using replication features.

Migrating a System with Oracle Parallel Server Installed

If you are migrating a system with Oracle Parallel Server installed, most of the actions described in this chapter should be performed on only one node of the system. So, perform the actions described in this chapter on only one node unless instructed otherwise in a particular step.

Migrating to a Different Operating System

The Oracle8i Migration utility *cannot* migrate a database to a computer system that has a different operating system. For example, it cannot migrate a database from Oracle7 on Solaris to Oracle8i on Windows NT. However, you normally can use Export/Import to migrate a database to a different operating system.

Note: Starting with release 8.1, a change in word-size is supported during the migration process. A change in word size involves switching between 32-bit and 64-bit architecture within the same operating system. See "[Changing Word-Size](#)" on page 1-4 for more information.

Character Set Considerations

It is not possible to change the character set during migration using the Migration utility; that is, the Oracle7 source database and the migrated Oracle8*i* database must have the same character set. All character data in the Oracle8*i* database is assumed to be in the character set specified in the CREATE DATABASE command that created the database.

However, you can change the character set by performing a full Export/Import. Or, you can use the ALTER DATABASE [NATIONAL] CHARACTER SET statement to change the character set, but only if the new character set is a true superset of the existing character set.

See Also: The *Oracle8i National Language Support Guide* for information about National Language Support (NLS), instructions for specifying a character set, and for a full list of character sets that can be used with the ALTER DATABASE [NATIONAL] CHARACTER SET statement.

Prepare the Oracle7 Source Database for Migration

Complete the following steps before you migrate your Oracle7 database to Oracle8i:

1. If your database release number is lower than the release supported by the Migration utility on your operating system, upgrade or migrate the database to a supported release.

See Also: ["Start with an Oracle7 Database Supported by the Migration Utility"](#) on page 3-5 for more information.

2. Make a complete backup of you Oracle7 database.

See Also: *The Oracle7 Server Administrator's Guide* for information about backing up your Oracle7 database.

3. If the Procedural Option is not installed, use your Oracle7 installation media to install it. See your operating-system specific Oracle documentation for instructions.

If you are not sure whether the Procedural Option is installed, you can check by starting Server Manager. The following is an example of the messages you will see when Server Manager starts:

```
Oracle Server Manager Release 2.3.3.0.0 - Production
```

```
Copyright (c) Oracle Corporation 1994, 1995. All rights reserved.
```

```
Oracle7 Server Release 7.3.4.0.0 - Production
```

```
With the distributed, replication, parallel query, Parallel Server  
and Spatial Data options
```

```
PL/SQL Release 2.3.4.0.0 - Production
```

The messages you see may be slightly different, based on the options you have installed and their release numbers. If you see "PL/SQL" in the messages, as in the last line in the preceding example, then the Procedural Option is installed. Otherwise, it is not installed.

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

To determine whether any datafiles require recovery, issue the following SQL statement:

```
SELECT * FROM v$recover_file;
```

You should see a "0 rows selected" message, which indicates that all datafiles are either online or offline normal. If any datafiles are listed, you must restore the datafiles before you migrate the database. You can use the V\$DATAFILE dynamic performance view to find the datafile name based on the datafile number. The Oracle8i 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 Oracle8i after the migration. Typically, tablespaces that are taken offline by using an ALTER TABLESPACE OFFLINE IMMEDIATE or ALTER TABLESPACE OFFLINE TEMPORARY command require media recovery.

After migration, tablespaces that are offline when you open the Oracle8i database remain in Oracle7 database file format. The offline tablespaces can be brought online at any time after migration, and the file headers are converted to Oracle8i 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.

5. Make sure no user or role has the name OUTLN, because this schema is created automatically when you install Oracle8i. If you have a user or role named OUTLN, you must drop the user or role and recreate it with a different name.

To check for a user with the name OUTLN, issue the following SQL statement:

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

If you do not have a user named OUTLN, zero rows are selected.

To check for a role with the name OUTLN, issue the following SQL statement:

```
SELECT role FROM dba_roles WHERE role = 'OUTLN';
```

If you do not have a role named OUTLN, zero rows are selected.

6. Make sure no user or role has the name MIGRATE, because the Oracle8i 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 with the name MIGRATE, issue the following SQL statement:

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

If you do not have a user named MIGRATE, zero rows are selected.

To check for a role with the name MIGRATE, issue the following SQL statement:

```
SELECT role FROM dba_roles WHERE role = 'MIGRATE';
```

If you do not have a role named MIGRATE, zero rows are selected.

7. Make sure the SYSTEM rollback segment does not have an OPTIMAL setting. An OPTIMAL setting may cause errors during migration.

To check the OPTIMAL setting for the SYSTEM rollback segment, issue the following SQL statement:

```
SELECT a.usn, a.name, b.optsize
       FROM v$rollname a, v$rollstat b
       WHERE a.usn = b.usn AND name = 'SYSTEM';
```

Your output should be similar to the following:

```
USN          NAME                OPTSIZE
-----
          0 SYSTEM
1 row selected.
```

If there is a value in the OPTSIZE column, issue the following SQL statement to set optimal to NULL:

```
ALTER ROLLBACK SEGMENT SYSTEM STORAGE (OPTIMAL NULL);
```

You can reset OPTIMAL when migration is complete.

See Also: The troubleshooting information in "[OPTIMAL Setting for the SYSTEM Rollback Segment](#)" on page A-4.

8. Increase the maximum number of extents for your SYSTEM rollback segment by altering the MAXEXTENTS parameter in the STORAGE clause of the ALTER ROLLBACK SEGMENT statement (optional).

The following is an example of the ALTER ROLLBACK SEGMENT statement:

```
ALTER ROLLBACK SEGMENT system
       STORAGE (MAXEXTENTS 121);
```

You may need more space in the SYSTEM rollback segment to complete the migration successfully. If there is not enough space in your SYSTEM rollback segment, you may encounter errors when you run the Migration utility in the Oracle7 environment.

See Also: If you are migrating an Oracle Parallel Server on Windows NT, see the *Oracle Parallel Server Getting Started for Windows NT* for important Operating System Dependent (OSD) layer instructions.

Install the Release 8.1 Oracle Software

Complete the following steps to install the release 8.1 software:

1. Follow the instructions in your operating-system specific Oracle documentation to prepare for installation and start the Oracle Universal Installer.

If you are migrating a system with Oracle Parallel Server installed, see the *Oracle8i Parallel Server Setup and Configuration Guide* for additional installation instructions.

2. At the Welcome screen of the Oracle Universal Installer, click Next.

If you need help at any screen or want to consult more documentation about the Oracle Universal Installer, click the Help button to open the online help.

3. At the File Locations screen, make sure the Destination is the complete path to your release 8.1 Oracle home. Then, click Next.

Note: You must install the new 8.1 release in an Oracle home separate from the Oracle7 release.

4. At the Available Products screen, select the edition of Oracle8i that you want to install, either Oracle8i Enterprise Edition or Oracle8i. Then, click Next.

Note: The Oracle Parallel Server option is available only with Oracle8i Enterprise Edition.

5. At the Installation Types screen, choose either Custom or Minimal. Do not choose Typical unless you want to install a starter database along with your Oracle software. You can avoid installing a starter database if you select Custom or Minimal. Normally, you should not install a starter database if you are migrating an existing database.

Note: Minimal is not supported for Oracle Parallel Server.

After you make your selection, click Next.

If you chose Custom, respond to the screens that enable you to specify your custom installation settings until you reach the "Upgrading or Migrating an Existing Database" screen.

Make sure you install all of the options you installed with the Oracle7 database, assuming you do not want to discontinue use of a particular option. For example, if you installed Advanced Replication in Oracle7, you should install it in Oracle8i.

6. If you are installing Oracle Parallel Server, at the Cluster Node Selection screen, select the nodes onto which you want the software installed. Then, click Next.
7. At the "Upgrading or Migrating an Existing Database" screen, leave the "Upgrade or Migrate an Existing Database" checkbox unselected. Then, click Next.

If you select the "Upgrade or Migrate an Existing Database" checkbox, the Oracle Data Migration Assistant is started automatically after installation. Because you are following the instructions for migrating the database with the Migration utility, you should not start the Oracle Data Migration Assistant.

Note: The Oracle Data Migration Assistant does not support Oracle Parallel Server migrations.

8. At the Create Database screen, select the No radio button, indicating that you do not want to create a database because you are migrating an existing database. Then, click Next.
9. At the Summary screen, make sure all of the settings and choices are correct for your installation. Then, click Install. The Oracle Universal Installer performs the installation, which may take some time.

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

See Also: If you are migrating an Oracle Parallel Server on Windows NT, see the *Oracle Parallel Server Getting Started for Windows NT* for important Operating System Dependent (OSD) layer instructions.

Review Migration Utility Command-Line Options

The next task in the migration process is running the Oracle8i 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 operating-system specific Oracle documentation may contain more information about Migration utility command-line options.

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.

DBNAME Specifies the name of the database to migrate (DB_NAME in *initsid.ora*).

MULTIPLIER Specifies the initial size of the Oracle8i *i_file#_block#* index relative to the Oracle7 *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 Oracle8i that is 1.5 times larger than the Oracle7 *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 PROPSS for the Oracle8i database, for example W52DEC or US7ASCII. If no NLS_NCHAR option is specified, the Migration utility uses the Oracle7 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.

PFILE Specifies the name of the initialization parameter file. If no PFILE command-line option is specified, the Migration utility uses the default *initsid.ora* file.

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

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

Migrate the Oracle7 Source Database

Complete the steps in the following sections to migrate an Oracle7 source database to Oracle8i using the Migration utility.

Prepare the Oracle7 Environment for Migration

The `migprep` utility prepares the Oracle7 environment for migration by copying required migration files from the Oracle8i Oracle home to the Oracle7 Oracle home. With your environment variables pointing to the new release 8.1 Oracle home, run `migprep` in the following way:

```
migprep new_oracle_home old_oracle_home
```

Where *new_oracle_home* is the complete path for the new Oracle8i Oracle home directory and *old_oracle_home* is the complete path for the old Oracle7 Oracle home directory.

For example, on UNIX, if your new Oracle8i Oracle home is `/oracle/product/8.1` and your old Oracle7 Oracle home is `/oracle/product/7.3`, enter the following to run `migprep`:

```
migprep /oracle/product/8.1 /oracle/product/7.3
```

On Windows NT, if your new Oracle8i Oracle home is `\oracle\product\8.1` on the E drive and your old Oracle7 Oracle home is `\oracle\product\7.3` on the D drive, enter the following to run `migprep`:

```
migprep e:\oracle\product\8.1 d:\oracle\product\7.3
```

Migration Steps in the Oracle7 Environment

Complete the following migration steps in the Oracle7 environment:

1. Make sure the following environment variables point to the Oracle7 directories:
 - ORACLE_HOME
 - PATH
 - LD_LIBRARY_PATH
 - ORA_NLS
 - ORACLE_BASE
 - ORACLE_PATH (if set)

Note: Some of these environment variables may not apply to your operating system.

Note: For Oracle Parallel Server, perform this step on all nodes.

2. Refer to your operating-system specific documentation to verify the system settings that control the placement of new Oracle8*i* database objects you will be creating.

The following examples illustrate operating-system specific variable settings:

- On a UNIX system, the TWO_TASK environment variable must be unset.
- On a Windows NT system, TWO_TASK must remain set.
- On VMS, the ORA_DFLT_HOSTSTR logical variable must be unset.

Note: For Oracle Parallel Server, perform this step on all nodes.

3. Set the ORA_NLS33 environment variable to the following directory in your Oracle7 environment:

```
$ORACLE_HOME/migrate/nls/admin/data
```


4. Make sure the NLS_LANG environment variable is set to the character set you are using for your database.

To check your character set, issue the following SQL statement:

```
SELECT * FROM v$nls_parameters
WHERE parameter = 'NLS_LANGUAGE'
OR parameter = 'NLS_TERRITORY'
OR parameter = 'NLS_CHARACTERSET';
```

You use all three values returned by this query to set NLS_LANG. For example, if your NLS_LANGUAGE is AMERICAN, your NLS_TERRITORY is AMERICA, and your NLS_CHARACTERSET is WE8ISO8859P1, set NLS_LANG to the following:

```
AMERICAN_AMERICA.WE8ISO8859P1
```

See Also: *Oracle8i National Language Support Guide* for information about setting NLS_LANG.

5. Make sure you have DBA privileges, which are required to run the Oracle8i Migration utility.

To check if you have DBA privileges, query the DBA_ROLE_PRIVS static data dictionary view. For example, if you are connected as user SYSTEM, enter the following SQL statement:

```
SELECT * FROM dba_role_privs WHERE grantee = 'SYSTEM';
```

You have DBA privileges if 'DBA' is listed in the GRANTED_ROLE column for the user. If you do not have DBA privileges, connect as a user who does.

6. Make sure no other DBA with RESTRICTED SESSION privilege connects to the database while the Migration utility is running. Also, "Normal" users should not connect to the database during migration.
7. Shutdown the Oracle7 database cleanly using the SHUTDOWN NORMAL or SHUTDOWN IMMEDIATE command; *do not use* SHUTDOWN ABORT. The Oracle7 source database must be shut down cleanly; therefore, no redo information or uncommitted transactions can remain.

```
SVRMGR> SHUTDOWN IMMEDIATE
```

If you are using Oracle Parallel Server, shutdown all instances.

Note: If you do not shut down the Oracle7 database before migration starts, the Migration utility will stop and display an error message.

8. Make sure you have enough space in the SYSTEM tablespace (optional).

A common migration problem is running out of space in the SYSTEM tablespace during migration. The Migration utility will not complete the migration unless sufficient space is allocated in the SYSTEM tablespace. To determine disk space requirements for a successful migration, run the Oracle8i Migration utility with the CHECK_ONLY command-line option set to TRUE by entering the following at a system prompt:

```
mi_g CHECK_ONLY=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 Oracle8i data dictionary or perform any other migration processing.

If the CHECK_ONLY command-line option shows that you need to add more space to the SYSTEM tablespace, issue a command similar to the following, substituting the appropriate directory path and name for the new datafile and the amount of space you need to add:

```
ALTER TABLESPACE system
  ADD DATAFILE '/home/user1/mountpoint/oradata/db1/system02.dbf'
  SIZE 50M;
```

If you add space to the SYSTEM tablespace, remember to shut down the database when you are finished.

9. Run the Oracle8i Migration utility by entering the Migration utility command at the system prompt:

```
mi_g
```

The command is `mi_g` unless stated otherwise in your operating-system specific Oracle documentation. Enter `mi_g` alone to run the Migration utility with a default set of options, or enter `mi_g` followed by one or more selected options.

See Also: ["Review Migration Utility Command-Line Options"](#) on page 3-14 for information about command-line options.

10. 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. If the Migration utility exits with an ORA- error, check [Appendix A, "Troubleshooting Migration Problems"](#) for information about the error and the actions to perform to resolve the problem.

The Migration utility creates a convert file that contains the information of the Oracle7 control file. Later in the migration process, the convert file is used by `ALTER DATABASE CONVERT` to create a new control file in Oracle8i.

The name and location of the convert file are operating-system specific. For example, on a UNIX operating system, the default location is `$ORACLE_HOME/dbs` in the Oracle7 environment, and the default filename in this directory is `convsid.dbf`, where *sid* is your Oracle7 instance ID. On Windows NT, the default location is `$ORACLE_HOME\database` in the Oracle7 environment, and the default filename in this directory is `convert.ora`.

CAUTION: Do not open the Oracle7 database, which was shut down by the Oracle8i 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 Oracle8i. If the Oracle7 database is opened after running the Migration utility, the SCN check will fail when the database is converted to Oracle8i, and an ORA-1211 error will be displayed, stating "Oracle7 datafile is not from migration to Oracle8". Therefore, if the Oracle7 database is opened, you must rerun the Migration utility, starting at Step 7.

Preserve the Oracle7 Source Database

After you successfully run the Migration utility, perform a cold backup of the Oracle7 database. This backup serves the following purposes:

- If you wish to return to the Oracle7 database after executing the ALTER DATABASE CONVERT command on Oracle8i, you can restore the backup, start the Oracle7 database, and follow procedure in "[Abandoning the Migration](#)" on page 3-30.
- It can be used as the first Oracle8i backup for an Oracle8i recovery.
- If an error occurs at Oracle8i 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 that was performed before you ran the Migration utility, you must rerun the Migration utility.

See Also: The *Oracle7 Server Administrator's Guide* for information about performing backup and restore operations on your Oracle7 database.

In addition, perform a backup of the entire Oracle7 software distribution, including the Oracle7 home directory. 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 Oracle7 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 Oracle7 database
- scripts that could restore the original database, if necessary

Migration Steps in the Oracle8i Environment

Complete the following migration steps in the Oracle8i environment:

See Also: If you are migrating an Oracle Parallel Server on Windows NT, see the *Oracle Parallel Server Getting Started for Windows NT* for important Operating System Dependent (OSD) layer instructions.

1. Make sure that the following environment variables point to the Oracle8i executables:
 - ORACLE_HOME
 - PATH
 - LD_LIBRARY_PATH
 - ORA_NLS
 - ORACLE_BASE
 - ORACLE_PATH (if set)

If ORACLE_HOME points to the Oracle7 executables, an ORA-223 error is displayed when you run the ALTER DATABASE CONVERT command later in the migration process, stating "conversion data file is invalid or incorrect version".

Note: Some of these environment variables may not apply to your operating system.

Note: For Oracle Parallel Server, perform this step on all nodes.

See Also: Your operating-system specific Oracle8i installation documents for information about setting other important environment variables on your operating system.

2. Either remove or rename the database's control files, or use the CONTROL_FILES initialization parameter to specify new control file names. The CONTROL_FILES initialization parameter typically is set in the `initsid.ora` file, but, if you are using Oracle Parallel Server, it may be set in the `initdb_name.ora` file instead.

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 (derived from the CONVERT file) 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 to specify new control file names, 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.

Control files are considerably larger in Oracle8i than in Oracle7. For example, Oracle7 control files in the hundreds of kilobytes may expand into tens of megabytes in Oracle8i. The larger size in Oracle8i results from the storage of more information in the control file, such as backup and tablespace records. This size increase could be important if a control file is on a raw device or if its available disk space is restricted.

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 *Oracle8i Administrator's Guide* for more information.

Note: For Oracle Parallel Server, perform this step on all nodes.

3. Copy files that are important for migration to a location outside of the Oracle7 Oracle home:
 - a. Move or copy the convert file from the Oracle7 Oracle home directory to the Oracle8i Oracle home directory. On most UNIX operating systems, the convert file, *convsid.dbf* (where *sid* is the Oracle8i database name), should reside in `$ORACLE_HOME/dbs` in both the Oracle7 and the Oracle8i environment. On Windows NT, the convert file, *convert.ora*, should reside in `$ORACLE_HOME\database` in both the Oracle7 and the Oracle8i environment.
 - b. If you have a password file that resides within the Oracle7 Oracle home, move or copy the password file to a location outside of the Oracle7 Oracle home directory.

The name and location of the password file is operating-system specific; for example, on UNIX operating systems, the default password file is `$ORACLE_HOME/dbs/orapwsid`, but on Windows NT, the default password file is `$ORACLE_HOME\database\pwsid.ora`. In both cases, *sid* is your Oracle instance ID.
 - c. If your *initsid.ora* file resides within the Oracle7 Oracle home, move or copy it to a location outside of the Oracle7 Oracle home. In past releases, the default location for the *initsid.ora* file was `$ORACLE_HOME/dbs` on UNIX and `$ORACLE_HOME\database` on Windows NT, but in release 8.1, the default location is `$ORACLE_BASE/admin/sid/pfile`, where *sid* is the Oracle instance ID. The *initsid.ora* file can reside anywhere you wish, but it should not reside in the Oracle7 Oracle home after you migrate to Oracle8i.
 - d. If the *initsid.ora* file contains an IFILE (include file) entry that resides within the Oracle7 Oracle home, copy the file specified in the IFILE entry to a location outside of the Oracle7 Oracle home.
 - e. If you are using Oracle Parallel Server and your *initdb_name.ora* file resides within the Oracle7 Oracle home, move or copy the *initdb_name.ora* file to a location outside of the Oracle7 Oracle home.

Note: For Oracle Parallel Server, perform this step on all nodes.

4. Adjust the `init.ora` file in the Oracle8i environment for use with Oracle8i. Specifically, complete the following steps:
 - a. Set the COMPATIBLE initialization parameter in your `init.ora` file to a valid version 8 setting, such as 8.0.5 or 8.1.0. Make sure the COMPATIBLE parameter is not set to any Oracle7 release, because if it is, you will not be able to start the Oracle8i database and the migration will fail. See ["Setting the COMPATIBLE Parameter"](#) on page 8-6 for information.
 - b. Remove obsolete parameters and adjust changed parameters. Certain Oracle7 initialization parameters are obsolete in version 8. Remove all obsolete initialization parameters from any `init.ora` file that will start an Oracle8i instance; obsolete parameters may cause errors in Oracle8i. Also, alter any parameter whose syntax has changed in version 8; refer to [Appendix B, "Changes to Initialization Parameters"](#) for lists of new, renamed, and obsolete parameters. Also, if you are using Oracle Parallel Server, see *Oracle8i Parallel Server Concepts and Administration* for more information about obsolete Oracle Parallel Server initialization parameters.
 - c. If you are updating snapshots automatically by using the JOB_QUEUE_PROCESSES initialization parameter, comment out the this parameter in the `init.ora` file. After migrating your database, you can remove the comments to use this parameter normally.
 - d. Adjust the LARGE_POOL_SIZE setting, if required. In release 8.1 of Oracle, the LARGE_POOL_SIZE setting may be calculated automatically by Oracle. If the automatic setting is too large, it may increase the time required to perform your migration. See ["Parallel Execution Allocated from Large Pool"](#) on page B-9 for more information.
 - e. If you are using Oracle Parallel Server, set the PARALLEL_SERVER initialization parameter to FALSE. You can change it back to TRUE after migration is complete.
 - f. If you are using a Distributed Lock Manager (DLM) on a UNIX operating system, make sure you set the LM_LOCKS, LM_RESS, and LM_PROCS initialization parameters equal to the lock, resource, and process parameters for the DLM used in Oracle7.
 - g. Make sure your DB_DOMAIN initialization parameter is set properly.

See Also: ["The DB_DOMAIN Parameter"](#) on page B-9 for more information about setting this initialization parameter.

- h. Edit the `initsid.ora` file in the new location by changing any question marks (?) in pathnames to the full path for the old Oracle7 Oracle home. If the full pathname for Oracle home is stated, you do not need to edit the `initsid.ora` file. The `initsid.ora` file should contain only full pathnames, not relative pathnames.
- i. If the `initsid.ora` file contains an `IFILE` entry, change the `IFILE` entry in the `initsid.ora` file to point to the new location you copied it to in Step 3. Then, edit the file specified in the `IFILE` entry in the same way that you edited the `initsid.ora` file in sub-steps a to h.
- j. If you are using Oracle Parallel Server, modify the `initdb_name.ora` file in the same way that you modified the `initsid.ora` file in steps a to h.

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

Note: For Oracle Parallel Server, perform this step on all nodes.

5. If the Oracle8i `DB_NAME` is different from the Oracle7 `DB_NAME`, complete the following steps. Otherwise, skip to Step 6.
 - a. On UNIX operating systems, rename the `convsid.dbf` file to match the Oracle8i `DB_NAME`. For example, if the Oracle7 `DB_NAME` is `DBMS7` and the Oracle8i `DB_NAME` is `DBMS8`, rename the convert file from `convDBMS7.dbs` to `convDBMS8.dbs`. This action is not necessary on Windows NT.
 - b. Set the `DB_NAME` parameter in the `initsid.ora` file to the Oracle8i database name.
6. 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.
7. Change to the `$ORACLE_HOME/rdbms/admin` directory. You should be in the Oracle8i Oracle home.
8. Start Server Manager.
9. Connect to the database instance:

```
SVRMGR> CONNECT INTERNAL
```

10. Start an Oracle8i database instance without mounting the new Oracle8i database:

```
SVRMGR> STARTUP NOMOUNT
```

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

You may need to use the PFILE option to specify the location of your `initsid.ora` file.

You may see error messages listing obsolete parameters. If so, shut down the database, edit the `initsid.ora` file to remove the parameters listed, and then run STARTUP NOMOUNT again.

11. Create a new Oracle8i database control file and convert the file headers of all online tablespaces to Oracle8i format by issuing the following command:

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

Successful execution of this command is the "point of no return" to Oracle7 for this database. However, if necessary, you can restore the Oracle7 database from backups.

If errors occur during this step, correct the conditions that caused the errors and rerun the Migration utility. Restart at Step 1 on page 3-16. Otherwise restore the backup you performed after you ran the Migration utility.

See Also: ["Problems at the ALTER DATABASE CONVERT Command"](#) on page A-16 for information about common errors encountered at this step and the actions required to resolve them.

12. Open the Oracle8i database with the following command:

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

When the Oracle8i database is opened, all rollback segments that are online are converted to the new Oracle8i format.

If you encounter errors when you issue this command, start the Migration process over from the beginning, ensuring the database is not opened in the Oracle7 environment after the Migration utility completes. Start from the beginning of this chapter, [Chapter 3](#), but make sure you completed all of the pre-migration steps described in [Chapter 2](#).

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

```
SVRMGR> SPOOL catoutm.log
```

If you want to see the output on your screen of the scripts you will run, you also can issue a SET ECHO ON statement:

```
SVRMGR> SET ECHO ON
```

14. Run the Oracle8i database conversion script `u0703040.sql`:

```
SVRMGR> @u0703040.sql
```

The `u0703040.sql` script is the database conversion script for all 7.1, 7.2, and 7.3 releases supported by the Migration utility on your operating system. The `u0703040.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.

If you encounter any problems when you run this script, or any of the scripts in the remaining steps, correct the cause(s) of the problems and rerun the script. You can rerun any of the scripts described in this chapter as many times as necessary.

See Also: ["Running Scripts"](#) on page 1-3 for information about the types of errors to look for when you run a script.

Note: If the `u0703040.sql` script runs for an inordinately long time, it may be caused by a setting for `LARGE_POOL_SIZE` that is too large for your installation. Use the `VSPARAMETER` view to check the setting for `LARGE_POOL_SIZE`, and if it is too large, set it to a smaller value in your `init sid .ora` file. See ["Parallel Execution Allocated from Large Pool"](#) on page B-9 for more information.

15. If the Oracle system has Advanced Replication installed, complete the following steps:

- a. Run `catrep.sql`:

```
SVRMGR> @catrep.sql
```

- b. Run `r0703040.sql`:

```
SVRMGR> @r0703040.sql
```

This `r0703040.sql` script performs a post-`catrep.sql` Advanced Replication related upgrade.

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

```
SVRMGR> @catparr.sql
```

17. Run `utlrp.sql` (optional):

```
SVRMGR> @utlrp.sql
```

The `utlrp.sql` script recompiles all existing PL/SQL modules that were previously in an INVALID state, such as packages, procedures, types, etc. These actions are optional; however, they ensure that the cost of recompilation is incurred during installation rather than in the future.

Oracle Corporation highly recommends performing this optional step.

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

```
SVRMGR> SPOOL OFF
```

Then, check the spool file and verify that the packages and procedures compiled successfully. You named the spool file in Step 13; the suggested name was `catoutm.log`.

You should look for errors that alert you to insufficient space, and for errors that alert you that a script failed to run. If you see these types of errors, your migration may not be completely successful. However, you typically can ignore errors about the failure to alter or drop an object that does not exist.

If you specified `SET ECHO ON`, you may want to `SET ECHO OFF` now:

```
SVRMGR> SET ECHO OFF
```

19. Run SHUTDOWN on the Oracle8i database:

```
SVRMGR> SHUTDOWN IMMEDIATE
```

CAUTION: 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 Oracle8i database.

The COMPATIBLE initialization parameter controls the compatibility level of your database. Set the COMPATIBLE initialization parameter in your `initsid.ora` file based on the compatibility level you want for your migrated database.

See Also: "[Setting the COMPATIBLE Parameter](#)" on page 8-6 for information.

20. Complete the procedures described in [Chapter 6, "After Migrating the Database"](#).

CAUTION: If you retain the old Oracle7 software, never start the migrated database with the old Oracle7 software. Only start the database with the executables in the new Oracle8i installation.

Troubleshooting 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, "Troubleshooting Migration Problems"](#) and *Oracle8i Error Messages* for information about errors during migration and about corrective action for each error.

Abandoning the Migration

If you took a backup of your Oracle7 database *before you ran the Migration utility*, the easiest way to abandon a migration is to restore that backup. However, if you do not have a backup, or if you took the backup after running the Migration utility, you must complete the procedure described in this section to abandon the migration.

You can run the Oracle8i Migration utility multiple times and still return to the Oracle7 database. However, running the Migration utility automatically eliminates the Oracle7 database catalog views. Therefore, to return to the Oracle7 database after running the Migration utility, you must run the Oracle7 `catalog.sql` script to restore the Oracle7 database catalog views.

Note: You cannot use the procedure below to abandon the migration if you already executed the ALTER DATABASE CONVERT command. If you executed this command and want to return to Oracle7, complete the procedure in [Chapter 13, "Downgrading to Oracle7"](#).

To abandon the migration, you generally must restore the Oracle7 database by completing the following steps in the Oracle7 environment:

1. Start the Oracle7 database using Server Manager.

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 Oracle8i Migration utility upgrades release 7.1 and release 7.2 databases to release 7.3. If the original Oracle7 production database was release 7.1 or 7.2 and the migration is run but abandoned before the conversion to Oracle8i, the Oracle7 database will be left with a dictionary that is release 7.3. However, in such a case, you do not need to downgrade from release 7.3 to release 7.1 or 7.2; your release 7.1 or 7.2 software should work with the data dictionary without the need for further action.

Migrating Using the Oracle Data Migration Assistant

This chapter guides you through the process of migrating an Oracle7 database to Oracle8i using the Oracle Data Migration Assistant. This chapter covers the following topics:

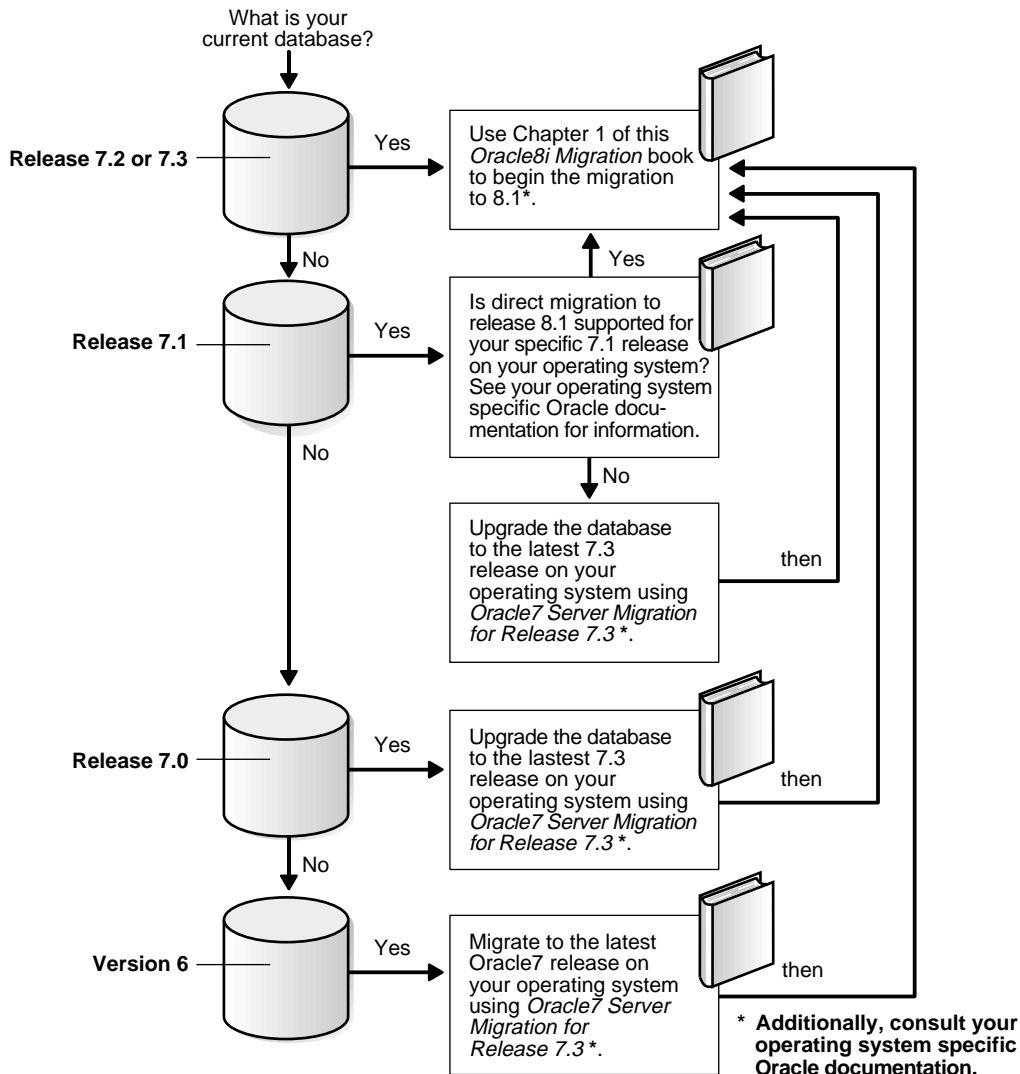
- [Documentation Roadmap for Using the Oracle Data Migration Assistant](#)
- [Overview of Migration Using the Oracle Data Migration Assistant](#)
- [System Considerations and Requirements](#)
- [Prepare the Oracle7 Source Database for Migration](#)
- [Install the Release 8.1 Oracle Software and Migrate the Database](#)
- [Troubleshooting Errors During Migration](#)
- [Abandoning the Migration](#)

See Also: Some aspects of migration are operating-system specific. See your operating-system specific Oracle documentation for additional information about migrating.

Documentation Roadmap for Using the Oracle Data Migration Assistant

Figure 4-1 is a roadmap that specifies the documentation you should use to migrate your database to release 8.1 based on your current release of Oracle.

Figure 4-1 Documentation Roadmap for Using the Oracle Data Migration Assistant



Overview of Migration Using the Oracle Data Migration Assistant

This section contains important considerations for using the Oracle Data Migration Assistant.

CAUTION: The Oracle Data Migration Assistant does not support the migration of systems with Oracle Parallel Server installed. If you have Oracle Parallel Server installed, you must use an another method to migrate your database, such as the Migration utility or Export/Import.

Start with an Oracle7 Database Supported by the Oracle Data Migration Assistant

A version 6 database must be migrated to at least Oracle7 before it can be migrated to Oracle8i. Also, the Oracle Data Migration Assistant cannot migrate some Oracle7 releases. See your operating-system specific Oracle documentation for information about the earliest release that is supported by the Oracle Data Migration Assistant on your operating system.

For example, on some operating systems, the Oracle Data Migration Assistant can migrate only release 7.1.4 and later databases, and cannot migrate a release lower than release 7.1.4 (such as release 7.0 or release 7.1.3). If your database release number is lower than the release supported by the Oracle Data Migration Assistant on your operating system, upgrade or migrate the database to the required release.

See Also: *Oracle7 Server Migration, Release 7.3* for instructions about migrating or upgrading the database to the required release. Then, use this *Oracle8i Migration* manual to migrate to Oracle8i.

Downgrading

Downgrading is the process of transforming an existing Oracle database into a previous version or release. The Oracle Data Migration Assistant *cannot* transform an Oracle8i database back to Oracle7. In some situations, you can use another facility to downgrade, such as using Export/Import, restoring from backups, and possibly using other functions.

See Also: [Chapter 12](#) and [Chapter 13](#) for information about downgrading.

System Considerations and Requirements

The following sections discuss system considerations and requirements for using the Oracle Data Migration Assistant.

Space Requirements

Oracle8i binaries may require as much as three times the disk space required by Oracle7 binaries. This requirement may cause you to run out of disk space during migration. However, the Oracle Data Migration Assistant requires relatively little temporary space. It needs only enough extra room in the SYSTEM tablespace to hold the new Oracle8i data dictionary simultaneously with the existing Oracle7 data dictionary.

The space required to hold an Oracle data dictionary depends on how many objects are in the database. Typically, a new Oracle8i data dictionary requires double the space that its Oracle7 source data dictionary required. If necessary, add space to the SYSTEM tablespace. The Oracle Data Migration Assistant will not complete the migration unless sufficient space is allocated in the SYSTEM tablespace.

If you need to add more space to the SYSTEM tablespace, issue a command similar to the following, substituting the appropriate directory path and name for the new datafile and the amount of space you need to add:

```
ALTER TABLESPACE system
  ADD DATAFILE '/home/user1/mountpoint/oradata/db1/system02.dbf'
  SIZE 50M;
```

Control files are considerably larger in Oracle8i than in Oracle7. For example, Oracle7 control files in the hundreds of kilobytes may expand into tens of megabytes in Oracle8i. The larger size in Oracle8i results from the storage of more information in the control file, such as backup and tablespace records. This size increase could be important if a control file is on a raw device or if its available disk space is restricted.

Block Size Considerations

The value of `DB_BLOCK_SIZE` (a parameter in the `init sid .ora` file) in the Oracle7 database and in the migrated Oracle8i database *must* be the same. Oracle8i requires a minimum block size of 2048 bytes (2KB). Above this amount, integer multiples of your operating system'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 Oracle8i block size setting meets the following criteria:

- Matches the Oracle7 setting.
- Is at least 2048 bytes (2KB). The Oracle Data Migration Assistant displays an error message if the Oracle7 block size is less than 2KB.
- Is an integer-multiple of your operating system's physical block size, preferably a multiple of 2KB.

Considerations for Replication Environments

You can migrate an Oracle7 replication environment to Oracle8i. Oracle7 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: *Oracle8i Replication*, Appendix B, "Migration and Compatibility", for detailed instructions about migrating systems using replication features.

Migrating to a Different Operating System

The Oracle Data Migration Assistant *cannot* migrate a database to a computer system that has a different operating system. For example, it cannot migrate a database from Oracle7 on Solaris to Oracle8i on Windows NT. However, you normally can use Export/Import to migrate a database to a different operating system.

Note: Starting with release 8.1, a change in word-size is supported during the migration process. A change in word size involves switching between 32-bit and 64-bit architecture within the same operating system. See "[Changing Word-Size](#)" on page 1-4 for more information.

Character Set Considerations

It is not possible to change the character set during migration using the Oracle Data Migration Assistant; that is, the Oracle7 source database and the migrated Oracle8i database must have the same character set. All character data in the Oracle8i database is assumed to be in the character set specified in the CREATE DATABASE command that created the database.

However, you can change the character set by performing a full Export/Import. Or, you can use the ALTER DATABASE [NATIONAL] CHARACTER SET statement to change the character set, but only if the new character set is a true superset of the existing character set.

See Also: The *Oracle8i National Language Support Guide* for information about National Language Support (NLS), instructions for specifying a character set, and for a full list of character sets that can be used with the ALTER DATABASE [NATIONAL] CHARACTER SET statement.

Prepare the Oracle7 Source Database for Migration

Complete the following steps before you migrate your Oracle7 database to Oracle8i:

1. If your database release number is lower than the release supported by the Oracle Data Migration Assistant on your operating system, upgrade or migrate the database to a supported release.

See Also: ["Start with an Oracle7 Database Supported by the Oracle Data Migration Assistant"](#) on page 4-3 for more information.

2. Make a complete backup of you Oracle7 database.

See Also: *The Oracle7 Server Administrator's Guide* for information about backing up your Oracle7 database.

3. If the Procedural Option is not installed, use your Oracle7 installation media to install it. See your operating-system specific Oracle documentation for instructions.

If you are not sure whether the Procedural Option is installed, you can check by starting Server Manager. The following is an example of the messages you will see when Server Manager starts:

```
Oracle Server Manager Release 2.3.3.0.0 - Production
```

```
Copyright (c) Oracle Corporation 1994, 1995. All rights reserved.
```

```
Oracle7 Server Release 7.3.4.0.0 - Production
```

```
With the distributed, replication, and Spatial Data options
```

```
PL/SQL Release 2.3.4.0.0 - Production
```

The messages you see may be slightly different, based on the options you have installed and their release numbers. If you see "PL/SQL" in the messages, as in the last line in the preceding example, then the Procedural Option is installed. Otherwise, it is not installed.

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

To determine whether any datafiles require recovery, issue the following SQL statement:

```
SELECT * FROM v$recover_file;
```

You should see a "0 rows selected" message, which indicates that all datafiles are either online or offline normal. If any datafiles are listed, you must restore the datafiles before you migrate the database. You can use the V\$DATAFILE dynamic performance view to find the datafile name based on the datafile number. You will encounter an error during migration 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 Oracle8i after the migration. Typically, tablespaces that are taken offline by using an ALTER TABLESPACE OFFLINE IMMEDIATE or ALTER TABLESPACE OFFLINE TEMPORARY command require media recovery.

After migration, tablespaces that are offline when you open the Oracle8i database remain in Oracle7 database file format. The offline tablespaces can be brought online at any time after migration, and the file headers are converted to Oracle8i 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.

5. Make sure no user or role has the name OUTLN, because this schema is created automatically when you install Oracle8i. If you have a user or role named OUTLN, you must drop the user or role and recreate it with a different name.

To check for a user with the name OUTLN, issue the following SQL statement:

```
SELECT USERNAME FROM dba_users WHERE USERNAME = 'OUTLN';
```

If you do not have a user named OUTLN, zero rows are selected.

To check for a role with the name OUTLN, issue the following SQL statement:

```
SELECT ROLE FROM dba_roles WHERE ROLE = 'OUTLN';
```

If you do not have a role named OUTLN, zero rows are selected.

6. Make sure no user or role has the name MIGRATE, because the Oracle Data Migration Assistant 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, issue the following SQL statement:

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

If you do not have a user named MIGRATE, zero rows are selected.

To check for a role named MIGRATE, issue the following SQL statement:

```
SELECT ROLE FROM dba_roles WHERE ROLE = 'MIGRATE';
```

If you do not have a role named MIGRATE, zero rows are selected.

7. Make sure the SYSTEM rollback segment does not have an OPTIMAL setting. An OPTIMAL setting may cause errors during migration.

To check the OPTIMAL setting for the SYSTEM rollback segment, issue the following SQL statement:

```
SELECT a.usn, a.name, b.optsize
       FROM v$rollname a, v$rollstat b
       WHERE a.usn = b.usn AND name='SYSTEM';
```

Your output should be similar to the following:

```
USN          NAME          OPTSIZE
-----
0 SYSTEM
1 row selected.
```

If there is a value in the OPTSIZE column, issue the following SQL statement to set optimal to NULL:

```
ALTER ROLLBACK SEGMENT SYSTEM STORAGE (OPTIMAL NULL);
```

You can reset OPTIMAL when migration is complete.

See Also: The troubleshooting information in "[OPTIMAL Setting for the SYSTEM Rollback Segment](#)" on page A-4.

8. Increase the maximum number of extents for your SYSTEM rollback segment by altering the MAXEXTENTS parameter in the STORAGE clause of the ALTER ROLLBACK SEGMENT statement (optional).

The following is an example of the ALTER ROLLBACK SEGMENT statement:

```
ALTER ROLLBACK SEGMENT system
       STORAGE (MAXEXTENTS 121);
```

You may need more space in the SYSTEM rollback segment to complete the migration successfully. If there is not enough space in your SYSTEM rollback segment, you may encounter an error when you run the Oracle Data Migration Assistant.

9. Make sure your SYSTEM tablespace has enough free space to hold the Oracle8i data dictionary and the existing Oracle7 data dictionary concurrently.

To check the free space in your SYSTEM tablespace, issue the following SQL statement:

```
SELECT sum(bytes) FROM dba_free_space
WHERE tablespace_name='SYSTEM';
```

This statement displays the number of free bytes in the system tablespace.

See Also: ["Space Requirements"](#) on page 4-4 and ["Assess System Requirements vs. Resources Available"](#) on page 2-15 for more information.

10. 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.
11. Adjust the `initsid.ora` file for use with Oracle8i. The `initsid.ora` file may still reside in the Oracle7 environment; the Oracle Data Migration Assistant will copy it to the Oracle8i environment when it is run.

Specifically, complete the following steps:

- a. If you are updating snapshots automatically by using the `JOB_QUEUE_PROCESSES` initialization parameter, comment out the this parameter in the `initsid.ora` file. After migrating your database, you can remove the comments to use this parameter normally.
- b. Adjust the `LARGE_POOL_SIZE` setting, if required. In release 8.1, the `LARGE_POOL_SIZE` setting may be calculated automatically by Oracle. If the automatic setting is too large, it may increase the time required to perform your migration. See ["Parallel Execution Allocated from Large Pool"](#) on page B-9 for more information.

- c. Make sure your `DB_DOMAIN` initialization parameter is set properly.

See Also: ["The DB_DOMAIN Parameter"](#) on page B-9 for more information about setting this initialization parameter.

- d. If the `initsid.ora` file contains an `IFILE` (include file) entry, edit the file specified in the `IFILE` entry in the same way that you edited the `initsid.ora` file in sub-steps a to d.

Make sure you save the `initsid.ora` file and the file specified in the `IFILE` entry, if one exists, after making these adjustments.

Install the Release 8.1 Oracle Software and Migrate the Database

Complete the following steps to install the release 8.1 software and migrate the database:

1. Follow the instructions in your operating-system specific Oracle documentation to prepare for installation and start the Oracle Universal Installer.

2. At the Welcome screen of the Oracle Universal Installer, click Next.

If you need help at any screen or want to consult more documentation about the Oracle Universal Installer, click the Help button to open the online help.

3. At the File Locations screen, make sure the Destination is the complete path to your release 8.1 Oracle home. Then, click Next.

4. At the Available Products screen, select the edition of Oracle8i that you want to install, either Oracle8i Enterprise Edition or Oracle8i. Then, click Next.

5. At the Installation Types screen, choose an installation type. Then, click Next.

If you chose Custom, respond to the screens that enable you to specify your custom installation settings until you reach the "Upgrading or Migrating an Existing Database" screen.

Make sure you install all of the options you installed with the Oracle7 database, assuming you do not want to discontinue use of a particular option. For example, if you installed Advanced Replication in Oracle7, you should install it in Oracle8i.

6. At the "Upgrading or Migrating an Existing Database" screen, complete the following steps:

- a. Select the "Upgrade or Migrate an Existing Database" check box.
- b. Choose the Oracle7 database to migrate.
- c. Click Next.

7. At the Summary screen, make sure all of the settings and choices are correct for your installation. Then, click Install. The Oracle Universal Installer performs the installation, which may take some time.

When installation is complete, one or more assistants may be started. When the Oracle Data Migration Assistant is started, you are ready to proceed with the migration.

- At the Welcome screen of the Oracle Data Migration Assistant, click Next. The exact screen may be slightly different than the one shown in [Figure 4-2](#), depending on your operating system.

Note: The Oracle Data Migration Assistant uses a SHUTDOWN IMMEDIATE statement to shut down the database. Therefore, no users should be logged into the database when the Oracle Data Migration Assistant starts. Users who are logged in will be disconnected.

Figure 4-2 Welcome Screen of the Oracle Data Migration Assistant



If you need help at any screen or want to consult more documentation about the Oracle Data Migration Assistant, click the Help button to open the online help.

9. At the "Before You Migrate or Upgrade" screen, make sure the Oracle7 database that you are migrating meets the conditions specified. Then, click Next.
10. At the "Select a Database Instance" screen, select the database instance of the Oracle7 database you are migrating. Then, click Next.
11. At the "Database Password and INIT.ORA File" screen, complete the following steps:
 - a. Make sure the specified new Oracle home is correct.
 - b. Make sure the location of the `init sid .ora` file specified is the complete path to the `init sid .ora` file of the Oracle7 database that you are migrating.
 - c. Make sure the old Oracle home specified is the complete path to the Oracle home of the Oracle7 database you are migrating.
 - d. Click Next.

12. At the "Choose Migration Type" screen, choose a migration type. Then, click Next.

If you chose Custom, respond to the screens that enable you to specify your custom migration settings until you reach the "Confirm Backup" screen.

13. At the "Confirm Backup" screen, click Next if you have backed up your Oracle7 database. If you have not, you should do so now and then repeat the migration.
14. At the "Start the Migration or Upgrade" screen, make sure all of the specifications are correct. If anything is incorrect, click Back until you can correct the specification. If everything is correct, click Next.

The Oracle Data Migration Assistant begins to perform the migration, and a status bar shows its progress.

15. At the "Listener.ora Migration Confirmation" screen, click the Yes button if you want the assistant to modify your `listener.ora` file automatically, or click the No button if you do not want the assistant to modify the `listener.ora` file.

Certain modifications are required to the `listener.ora` file for your database to work properly with Oracle Enterprise Manager. If you plan to use Oracle Enterprise Manager, you should click the Yes button to automatically modify the `listener.ora` file. However, if you do not plan to use Oracle Enterprise Manager, click the No button.

If you click the Yes button, the Oracle Data Migration Assistant modifies the `listener.ora` file in the following way:

- a. The assistant shuts down the old listener and the Oracle8i listener.
- b. The assistant modifies the `SID_DESC` entry for the migrated database in the Oracle8i `listener.ora` in one of the following ways:

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

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

If the database name is SAL, the domain name is COM, and the Oracle home is `/oracle/product/8.1`, the assistant adds the following entry:

```
...
(SID_DESC =
  (GLOBAL_DENAME = sal.com)
  (ORACLE_HOME = /oracle/product/8.1)
  (SID_NAME = SAL)
)
...
```

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

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

If *an_entry* does not match the GLOBAL_DBNAME of the migrated database, and if the database name is SAL, the domain name is COM, and the Oracle home is /oracle/product/8.1, the assistant adds the following entry:

```
...
  (SID_DESC =
    (GLOBAL_DBNAME = sal.com)
    (ORACLE_HOME = /oracle/product/8.1)
    (SID_NAME = SAL)
  )
...
```

This entry is the same as the entry in the simple case, but the assistant also adds the entry *an_entry* to the SERVICE_NAMES parameter. Therefore, the assistant changes the SERVICE_NAMES parameter to the following:

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

- c. On Windows NT, the assistant removes the entry of the migrated database from the old listener.ora file. The assistant does not perform this action on UNIX operating systems.
- d. The assistant starts up the Oracle8i listener.

Running the Oracle Data Migration Assistant Independently

If you installed Oracle8i without specifying that you are migrating an existing database, you can run the Oracle Data Migration Assistant independently after the Oracle8i installation is complete.

Complete the following steps to run the Oracle Data Migration Assistant independently:

1. Start the Oracle Data Migration Assistant.

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

```
odma
```

On Windows NT, select:

```
Start > Programs > Oracle - ORACLE_HOME_NAME > Migration Utilities > Oracle  
Data Migration Assistant
```

When you start the Oracle Data Migration Assistant, its welcome screen appears (see [Figure 4-2](#) on page 4-12).

2. Respond to questions in each Oracle Data Migration Assistant window, and click Next when you are ready to continue to the next window.

See Also: Step 8 to Step 15 in "[Install the Release 8.1 Oracle Software and Migrate the Database](#)" on page 4-11 for more information.

Finish the Migration

Complete the following steps after you have successfully run the Oracle Data Migration Assistant:

1. Start Server Manager.
2. Connect to the database instance:

```
SVRMGR> CONNECT INTERNAL;
```

3. Run STARTUP RESTRICT:

```
SVRMGR> STARTUP RESTRICT
```

You may need to use the PFILE option to specify the location of your `initsid.ora` file.

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

```
SVRMGR> SPOOL catoutma.log
```

If you want to see the output on your screen of the scripts you will run, you also can issue a SET ECHO ON statement:

```
SVRMGR> SET ECHO ON
```

5. If the Oracle system has Advanced Replication installed, complete the following steps:

- a. Run `catrep.sql`:

```
SVRMGR> @catrep.sql
```

- b. Run `r0703040.sql`:

```
SVRMGR> @r0703040.sql
```

This `r0703040.sql` script performs a post-`catrep.sql` Advanced Replication related upgrade.

If you encounter any problems when you run these scripts, or any of the scripts in the remaining steps, correct the causes of the problems and rerun the scripts. You can rerun any of the scripts described in this chapter as many times as necessary.

6. Run `utlrp.sql` (optional):

```
SVRMGR> @utlrp.sql
```

The `utlrp.sql` script recompiles all existing PL/SQL modules that were previously in an INVALID state, such as packages, procedures, types, etc. These actions are optional; however, they ensure that the cost of recompilation is incurred during installation rather than in the future.

Oracle Corporation highly recommends performing this optional step.

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

```
SVRMGR> SPOOL OFF
```

Then, check the spool file and verify that the packages and procedures compiled successfully. You named the spool file in Step 4; the suggested name was `catoutma.log`.

You should look for errors that alert you to insufficient space, and for errors that alert you that a script failed to run. If you see these types of errors, your migration may not be completely successful. However, you typically can ignore errors about the failure to alter or drop an object that does not exist.

If you specified `SET ECHO ON`, you may want to `SET ECHO OFF` now:

```
SVRMGR> SET ECHO OFF
```

8. Run `SHUTDOWN` on the Oracle8i database:

```
SVRMGR> SHUTDOWN IMMEDIATE
```

CAUTION: 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 Oracle8i database.

9. Adjust the `initsid.ora` file for Oracle8i.

Alter any parameter whose syntax has changed in version 8; refer to [Appendix B, "Changes to Initialization Parameters"](#) for lists of new, renamed, and obsolete parameters.

Also, learn about the new parameters listed in [Appendix B, "Changes to Initialization Parameters"](#) and decide which ones you want to use for your migrated database.

In addition, the Oracle Data Migration Assistant sets the COMPATIBLE initialization parameter to 8.0.5. See [Chapter 8, "Compatibility and Interoperability"](#) for information about resetting the COMPATIBLE initialization parameter.

See Also: *Oracle8i Reference* for detailed information about initialization parameters.

10. Complete the procedures described in [Chapter 6, "After Migrating the Database"](#).

CAUTION: If you retain the old Oracle7 software, never start the migrated database with the old Oracle7 software. Only start the database with the executables in the new Oracle8i installation directory.

Troubleshooting 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, "Troubleshooting Migration Problems"](#) and *Oracle8i Error Messages* for information about errors during migration and about corrective action for each error.

Abandoning the Migration

If you took a backup of your Oracle7 database *before you ran the Oracle Data Migration Assistant*, the easiest way to abandon a migration is to restore that backup. However, if you do not have a backup, or if you took the backup after running the assistant, you must complete the procedure described in this section to abandon the migration.

You can run the Oracle Data Migration Assistant multiple times and still return to the Oracle7 database. However, running the Oracle Data Migration Assistant automatically eliminates the Oracle7 database catalog views. Therefore, to return to the Oracle7 database after running the Oracle Data Migration Assistant, you must run the Oracle7 `catalog.sql` script to restore the Oracle7 database catalog views.

To abandon the migration, you generally must restore the Oracle7 database by completing the following steps in the Oracle7 environment:

1. Start the Oracle7 database using Server Manager.

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 Advanced Replication is installed, run `catrep.sql`:

```
SVRMGR> @catrep.sql
```

Note: The Oracle Data Migration Assistant upgrades release 7.1 and release 7.2 databases to release 7.3. If the original Oracle7 production database was release 7.1 or 7.2 and the migration is run but abandoned before the conversion to Oracle8i, the Oracle7 database will be left with a dictionary that is release 7.3. However, in such a case, you do not need to downgrade from release 7.3 to release 7.1 or 7.2; your release 7.1 or 7.2 software should work with the data dictionary without the need for further action.

Migrating Using Export/Import

This chapter provides information about using Export/Import to migrate a database from Oracle7 to Oracle8i.

This chapter covers the following topics:

- [Basics of Export/Import](#)
- [Migrate the Pre-Release 8.0 Source Database Using Export/Import](#)

See Also: *Oracle8i Utilities* for detailed information about using the Export and Import utilities.

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 Oracle8i database into which you will import the exported data (the target database).
3. Import the exported data into the new Oracle8i target database.

See Also: "[Choose a Migration Method](#)" on page 2-3 and *Oracle8i Utilities* for information that can help you to evaluate the choice of Export/Import for migration.

Export Utility Requirements

To migrate or upgrade a database, use the Export utility shipped with the 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.

For example, if you are migrating to release 8.1 from release 7.3, use the Export utility for release 7.3.

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

Import Requirements

To migrate, upgrade, or downgrade a database, use the Import utility shipped with the release of the *target* database. For example, if you are migrating to release 8.1 from release 7.3, use the Import utility for release 8.1.

Additional Options

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

- The Trusted Oracle documentation for information about migrating the features of the Trusted Oracle database if you are exporting from, or importing to, a Trusted Oracle database.
- *Oracle8i Replication*, Appendix B, "Migration and Compatibility", if you are migrating a database system that has Advanced Replication installed.

Migrate the Pre-Release 8.0 Source Database Using Export/Import

To migrate a version 6 or Oracle7 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 Oracle7 and version 6 database exports can be imported into Oracle8i.

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 Oracle8i target database after the import is complete.

2. Install the Oracle8i software. Installation is operating-system specific. Installation steps for Oracle8i are covered in your operating-system specific Oracle8i documentation.
3. If the new Oracle8i database will have the same name as the existing source database, shut down the existing database before creating the new Oracle8i database.
4. Create the Oracle8i target database.

See Also: The *Oracle8i Administrator's Guide* for information about creating an Oracle8i database.

5. Start Server Manager in the Oracle8i environment.
6. Connect to the database instance:

```
SVRMGR> CONNECT INTERNAL
```

7. Start an Oracle8i database instance using STARTUP.
8. 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 Oracle8i database will be created on the same computer 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.

9. Use the Oracle8i 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: *Oracle8i Utilities* for a complete description of the Import utility.

10. After the migration, check the import log file for information about which imports of which objects completed successfully and, if there were failures, which failed.

See Also: *Oracle8i Utilities* and the Oracle8i server README.doc file for error handling information.

11. Use further Import scenarios (see *Oracle8i Utilities*) or SQL scripts that create the source objects to clean up incomplete imports (or possibly to start an entirely new import).
12. If changes are made to the source database after the export, make sure those changes are propagated to the Oracle8i database prior to making it available to users. See Step 1 on page 5-3 for more information.
13. Complete the procedures described in [Chapter 6, "After Migrating the Database"](#).

After Migrating the Database

This chapter guides you through the procedures to perform after you have completed a migration using the Migration utility, the Oracle Data Migration Assistant, Export/Import, or data copy. This chapter elaborates on "[Step 6: Tune and Adjust the New Production Database](#)" on page 1-9.

This chapter covers the following post-migration steps:

- [Back Up the Migrated Database](#)
- [Check for Bad Date Constraints](#)
- [Rebuild Unusable Bitmap Indexes](#)
- [Avoid Problems with Parallel Execution](#)
- [Migrate Partition Views to Partition Tables](#)
- [Change the Password for the OUTLN User](#)
- [Migrate or Upgrade to the New Release of Net8 \(Optional\)](#)
- [Modify Your listener.ora File](#)
- [Test the Database and Compare Results](#)
- [Tune the Migrated Database](#)
- [Add New Features as Appropriate](#)
- [Develop New Administrative Procedures as Needed](#)

Back Up the Migrated 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: *Oracle8i Backup and Recovery 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 Oracle7 Source Database](#)" on page 3-20. This backup also can serve as the first Oracle8i backup for a recovery of the newly migrated database.

Check for Bad Date Constraints

A bad date constraint involves invalid date manipulation, which is a date manipulation 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, complete the following steps:

1. At a system prompt, change to the `$ORACLE_HOME/rdbms/admin` directory.
2. Start Server Manager.
3. Connect to the database instance:

```
SVRMGR> CONNECT INTERNAL
```

4. Enter the following:

```
SVRMGR> SPOOL utlresult.log
SVRMGR> @utlconst.sql
SVRMGR> SPOOL OFF
```

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

Note: The `utlconst.sql` script does not correct bad constraints, but instead disables them. You should either drop the bad constraints or recreate them after you make the necessary changes.

Rebuild Unusable Bitmap Indexes

During migration, some bitmap indexes may become unusable. To find these indexes, issue the following SQL statement:

```
SELECT index_name, index_type, table_owner, status
FROM dba_indexes
WHERE index_type = 'BITMAP'
AND status = 'UNUSABLE';
```

Rebuild the unusable bitmap indexes listed.

See Also: *Oracle8i Tuning* and *Oracle8i Concepts* for more information about using bitmap indexes.

Avoid Problems with Parallel Execution

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

- SHARED_POOL_SIZE
- LARGE_POOL_SIZE

See Also: ["Parallel Execution Allocated from Large Pool"](#) on page B-9 for information about adjusting these parameters.

Migrate Partition Views to Partition Tables

Partition views are not recommended for new applications in Oracle8i, and existing partition views should be converted to partitioned tables. You can convert partition views created for Oracle7 databases to partitioned tables by using the EXCHANGE PARTITION option of the ALTER TABLE command.

See Also: *Oracle8i Administrator's Guide* for information about converting partitioned views to partitioned tables and *Oracle8i Concepts* for background information about partition views and partitioned tables.

Note: Partition views will be de-supported in a future release of Oracle.

Change the Password for the OUTLN User

The OUTLN user is created automatically during installation of Oracle8i. This user has DBA privileges. Use the ALTER USER command to change the password for this user. Oracle8i adds the OUTLN user schema to support Plan Stability. The OUTLN user acts as a place to centrally manage metadata associated with stored outlines.

Migrate or Upgrade to the New Release of Net8 (Optional)

Migrating or upgrading to the new release of Net8 is not required. However, Net8 provides significant advantages over SQL*Net V2, including simplified configuration and expanded functionality. The new release of Net8 also provides the following advantages over past releases of Net8 and SQL*Net:

- **Service naming** enables clients to access a service as a whole, using the service name, rather than a specific database instance. Service naming logically separates the service name from any particular instance name and replaces the SID parameter, enabling one instance to serve multiple services. Individual instances also can be part of multiple services.
- **Instance registration** is automatic. Instances register themselves with the listener when they are started. In past releases, information about the instance was configured manually in the `listener.ora` file.

- **Failover** is automatic. If an instance is down, a client connect request is sent to a different listener automatically.
- **Load balancing** distributes connections over the available listeners.

See Also: *Net8 Administrator's Guide* for more information about the advantages of Net8 and for detailed instructions about migrating or upgrading to the new release of Net8.

Modify Your listener.ora File

If you are using Oracle Enterprise Manager to manage database objects, your `listener.ora` file must be configured with information about the database in the following manner:

```
sid_list_listener_name =
  (sid_list =
    (sid_desc =
      (global_dbname = global_database_name)
      (oracle_home = oracle_home)
      (sid_name = sid)
    )
  )
```

GLOBAL_DBNAME identifies the global database name, which is a name comprised of the database name and domain name of the database. ORACLE_HOME identifies the directory where your Oracle software is installed, and SID_NAME identifies the Oracle System Identifier of the database instance.

For example, the following information is for a database with a listener name of LISTENER. The global database name of SAL.COM is referenced by an instance with a SID of SAL, using the software installed in the `/oracle/product/8.1` directory:

```
sid_list_listener =
  (sid_list =
    (sid_desc =
      (global_dbname = sal.com)
      (oracle_home = d:/oracle/product/8.1)
      (sid_name = sal)
    )
  )
```

If you are not using Oracle Enterprise Manager, you do not need to configure information about the database, because an Oracle8i database instance registers itself with the listener automatically.

See Also: *Net8 Administrator's Guide* for more information about configuring the `listener.ora` file and the `SID_LIST` parameter.

Test the Database and Compare Results

Test the Oracle8i database using the testing plan you developed in "[Develop a Testing Plan](#)" on page 2-19. 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 Oracle8i database should be as good as, or better than, the performance of the source database. If you notice any decline in database performance with Oracle8i, make sure the initialization parameters are set properly, because improperly set initialization parameters can hurt performance.

Besides testing the migrated test database, you may run the original source database and the newly migrated Oracle8i 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 Migrated Database

If you want to improve the performance of the migrated database, tune the database. Most of the actions normally used to tune Oracle7 databases and related applications either have the same effect on or are unnecessary for Oracle8i databases. Therefore, actions you used to tune your source database and applications should not impair the performance of the migrated Oracle8i database.

See Also: *Oracle8i Tuning* for tuning information.

Add New Features as Appropriate

Getting to Know Oracle8i describes many of the new features available in Oracle8i. Determine which of these new features can benefit the migrated database and applications; then, develop a plan for using these features.

It is not necessary to make any immediate changes to begin using your migrated Oracle8i database. You may prefer to introduce these enhancements into your database and corresponding applications gradually.

[Chapter 9, "Upgrading Your Applications"](#), describes ways to enhance your applications so that you can take advantage of the new Oracle8i features. However, before you implement new Oracle8i features, test your applications and successfully run them with the migrated database.

Develop New Administrative Procedures as Needed

After familiarizing yourself with the Oracle8i features, review your database administration scripts and procedures to determine whether any changes are necessary.

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

Upgrading to the New Oracle8i Release

This chapter contains information about upgrading your current release of Oracle to the new Oracle8i release. The information in this chapter only applies to release 8.0 and higher installations of Oracle. If your current release is pre-release 8.0, such as Oracle7 or version 6, and you want to migrate to Oracle8i, follow the instructions at the beginning of this book, starting with [Chapter 1, "Overview"](#).

This chapter covers the following topics:

- [Upgrade Paths](#)
- [Upgrading the Database to the New Oracle8i Release](#)
- [Upgrading Specific Components](#)
- [After Upgrading to the New Release](#)
- [Changing the Word-Size of Your Current Release](#)

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

Upgrading Oracle Parallel Server: If you are upgrading a system with Oracle Parallel Server installed, most of the actions described in this chapter should be performed on only one node of the system. So, perform the actions described in this chapter on only one node unless instructed otherwise in a particular step.

Upgrade Paths

The path that you must take to upgrade your database to the new release depends on the release you are currently using. [Table 7-1](#) contains the upgrade path required for each old release of Oracle. Use the upgrade path and the documentation specified for the release you are running currently.

Table 7-1 Upgrade Paths

Old Release	Upgrade Path
8.0.1	<p>Direct upgrade <i>is not supported</i>. Complete the following steps to upgrade to the new release:</p> <ol style="list-style-type: none"> 1. Upgrade to release 8.0.2 using the instructions in the release 8.0.2 README .doc file. 2. Upgrade to release 8.0.5 using the instructions in the release 8.0.5 README .doc file. 3. Upgrade to the new release using the instructions in "Upgrading the Database to the New Oracle8i Release" on page 7-3.
8.0.2 8.0.4S	<p>Direct upgrade <i>is not supported</i>. Complete the following steps to upgrade to the new release:</p> <ol style="list-style-type: none"> 1. Upgrade to release 8.0.5 using the instructions in the release 8.0.5 README .doc file. 2. Upgrade to the new release using the instructions in "Upgrading the Database to the New Oracle8i Release" on page 7-3.
8.0.3 8.0.4 8.0.5 8.1.3 8.1.4	<p>Direct upgrade <i>is supported</i>. Upgrade to the new release using the instructions in "Upgrading the Database to the New Oracle8i Release" on page 7-3.</p>
8.1.1 8.1.2	<p>Upgrading to the new release <i>is not supported</i>.</p>

Upgrading the Database to the New Oracle8i Release

This section guides you through the process of upgrading your database to the new Oracle8i release.

Note: If you are upgrading from Oracle8i Enterprise Edition to Oracle8i (formerly Workgroup Server), before you upgrade, modify any applications that use the advanced features of Oracle8i Enterprise Edition so that they do not use these advanced features. See *Getting to Know Oracle8i* for more information about the differences between the editions.

Prepare to Upgrade

Complete the following steps to begin the upgrade process:

1. If you are upgrading from release 8.1.4 and you have Java source, class, or resource objects that you want to preserve, export these Java objects before you upgrade, and then import them after you upgrade.

Note: Exporting and importing Java objects between pre-release 8.1.5 databases and the current release is not guaranteed to be successful.

See Also: *Oracle8i Utilities* for information about exporting and importing Java source, class, and resource objects.

2. If you are upgrading from release 8.1.3 or 8.1.4 and you created any release 8.1 compatible queue tables using release 8.1.3 or release 8.1.4, drop all 8.1 compatible queue tables before upgrading the database. If you are upgrading from a release other than release 8.1.3 or 8.1.4, skip to Step 4.

To determine if there are any existing release 8.1 compatible queue tables in the database, issue the following SQL statement:

```
SELECT owner, queue_table FROM dba_queue_tables where
compatible like '8.1%';
```

For example, you might see the following table as a result of issuing the above query:

```
OWNER                                QUEUE_TABLE
-----
AQUSER1                              RAW_MSG_TABLE
1 row selected.
```

You can drop the queue tables listed by using DBMS_AQADM.DROP_QUEUE_TABLE, as in the following example:

```
EXECUTE dbms_aqadm.drop_queue_table(queue_table=>'AQUSER1.RAW_MSG_TABLE',
                                     force => TRUE);
```

3. Make sure your DB_DOMAIN initialization parameter is set properly.

See Also: ["The DB_DOMAIN Parameter"](#) on page B-9 for more information about setting this initialization parameter.

4. Start Server Manager.
5. Connect to the database instance:

```
SVRMGR> CONNECT INTERNAL
```

6. If you are upgrading from an 8.0 release, make sure no user or role has the name OUTLN, because this schema is created automatically when you install Oracle8i. If you have a user or role named OUTLN, you must drop the user or role and recreate it with a different name.

Note: If you are upgrading from an 8.1 release, you do not need to perform this check because the OUTLN user should have been created when you installed the previous 8.1 release. Therefore, if you are upgrading from an 8.1 release, skip to Step 7. *Do not* drop the OUTLN user if you are upgrading from a previous 8.1 release.

To check for a user with the name OUTLN, enter the following SQL statement:

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

If you do not have a user named OUTLN, zero rows are selected.

To check for a role with the name OUTLN, enter the following SQL command:

```
SELECT role FROM dba_roles WHERE role = 'OUTLN';
```

If you do not have a role named OUTLN, zero rows are selected.

7. Add space to your SYSTEM tablespace and to the tablespaces where you store rollback segments, if necessary.

Upgrading to a new release requires more space in your SYSTEM tablespace and in the tablespaces where you store rollback segments. If you have enough space on your system, consider adding more space to these tablespaces. If you run out of space during the upgrade, you will need to perform the upgrade again.

The following SQL statement illustrates how to add more space to a tablespace:

```
ALTER TABLESPACE system
  ADD DATAFILE '/home/user1/mountpoint/oradata/db1/system02.dbf'
  SIZE 50M;
```

8. Run SHUTDOWN IMMEDIATE on the database:

```
SVRMGR> SHUTDOWN IMMEDIATE
```

If you are using Oracle Parallel Server, shutdown all instances.

9. Perform a full offline backup of the database.

See Also: *Oracle8i Backup and Recovery Guide* for more information.

Upgrade the Database

Choose an upgrade method and then follow the instructions for upgrading using the method you have chosen.

Choose an Upgrade Method

There are two ways to upgrade your database to release 8.1. You can either use the Oracle Data Migration Assistant to complete the upgrade, or you can perform the upgrade manually.

The Oracle Data Migration Assistant provides a completely automated upgrade of your database. You use a graphical user interface (GUI), which guides you through each step of the process. In addition, the Oracle Data Migration Assistant includes extensive online help. The Oracle Data Migration Assistant runs the appropriate upgrade script for your current release, deletes any obsolete initialization parameters from your `initsid.ora` file, and optionally configures your `listener.ora` file. See [Appendix B, "Changes to Initialization Parameters"](#) for lists of obsolete initialization parameters.

On the other hand, you lose some flexibility and control by using the Oracle Data Migration Assistant. If you want complete control over the upgrade process, especially with regard to setting initialization parameters, you may want to perform the upgrade manually.

CAUTION: The Oracle Data Migration Assistant cannot upgrade systems with Oracle Parallel Server installed. If you have Oracle Parallel Server installed, you must upgrade the database manually.

Decide which method you want to use to upgrade your database, and then complete the steps in one of the following sections accordingly:

- [Upgrade the Database Using the Oracle Data Migration Assistant](#) on page 7-7.
- [Upgrade the Database Manually](#) on page 7-13.

Upgrade the Database Using the Oracle Data Migration Assistant

Complete the following steps to upgrade the database using the Oracle Data Migration Assistant:

1. Follow the instructions in your operating-system specific Oracle documentation to prepare for installation and start the Oracle Universal Installer.
2. At the Welcome screen of the Oracle Universal Installer, click Next.

If you need help at any screen or want to consult more documentation about the Oracle Universal Installer, click the Help button to open the online help.

3. At the File Locations screen, make sure the Destination is the complete path to the directory you want to use for your release 8.1 Oracle home. Then, click Next.

Note: If you are upgrading from an 8.0 release, you must install the new 8.1 release in an Oracle home separate from the 8.0 release. However, if you are upgrading from a previous 8.1 release, this restriction does not apply, and you can install the new release into the same Oracle home as the previous release if you wish.

4. At the Available Products screen, select the edition of Oracle8i that you want to install, either Oracle8i Enterprise Edition or Oracle8i. Then, click Next.
5. At the Installation Types screen, choose an installation type. Then, click Next.

If you chose Custom, respond to the screens that enable you to specify your custom installation settings until you reach the "Upgrading or Migrating an Existing Database" screen.

Make sure you install all of the options you installed with the Oracle7 database, assuming you do not want to discontinue use of a particular option. For example, if you installed Advanced Replication in Oracle7, you should install it in Oracle8i.

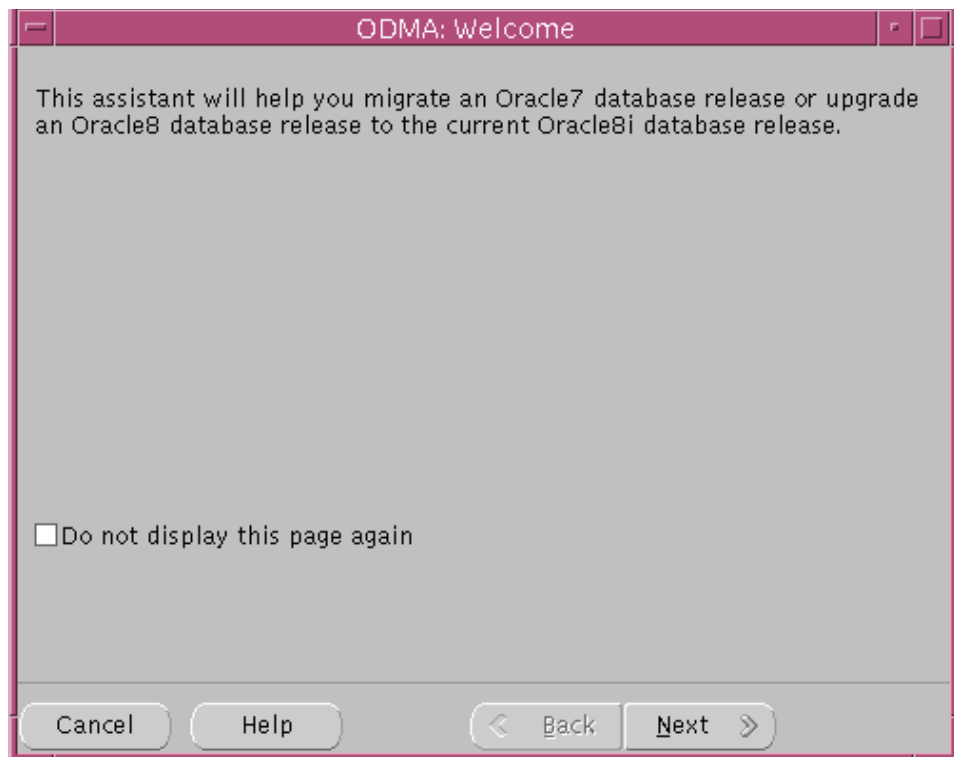
6. At the "Upgrading or Migrating an Existing Database" screen, complete the following steps:
 - a. Select the "Upgrade or Migrate an Existing Database" check box.
 - b. Choose the database to upgrade.
 - c. Click Next.

7. At the Summary screen, make sure all of the settings and choices are correct for your installation. Then, click Install. The Oracle Universal Installer performs the installation, which may take some time.

When installation is complete, one or more assistants may be started. When the Oracle Data Migration Assistant is started, you are ready to proceed with the upgrade.

8. At the Welcome screen of the Oracle Data Migration Assistant, click Next. The exact screen may be slightly different than the one shown in [Figure 7-1](#), depending on your operating system.

Figure 7-1 Welcome Screen of the Oracle Data Migration Assistant



If you need help at any screen or want to consult more documentation about the Oracle Data Migration Assistant, click the Help button to open the online help.

9. At the "Before You Migrate or Upgrade" screen, make sure the database that you are upgrading meets the conditions specified. Then, click Next.
10. At the "Select a Database Instance" screen, select the database instance of the database you are upgrading. Then, click Next.

Note: The database you choose must be release 8.0 or higher. If the database is an Oracle7 or lower database, you must complete a migration of the database, not an upgrade. If the database is an Oracle7 database, exit the Oracle Data Migration Assistant, and see [Chapter 1](#) to start the migration process.

11. At the "Database Password and INIT.ORA File" screen, complete the following steps:
 - a. Make sure the specified new Oracle home is correct.
 - b. Make sure the location of the `init sid .ora` file specified is the complete path to the `init sid .ora` file of the database that you are upgrading.
 - c. Make sure the old Oracle home specified is the complete path to the Oracle home of the database you are upgrading.
 - d. Click Next.

12. At the "Choose Migration or Upgrade Type" screen, choose a migration type. Then, click Next.

If you chose Custom, respond to the screens that enable you to specify your custom migration settings until you reach the "Confirm Backup" screen.

13. At the "Confirm Backup" screen, click Next if you have backed up the database. If you have not, you should do so now and then repeat the upgrade.
14. At the "Start the Migration or Upgrade" screen, make sure all of the specifications are correct. If anything is incorrect, click Back until you can correct the specification. If everything is correct, click Next.

The Oracle Data Migration Assistant performs the upgrade. If the COMPATIBLE initialization parameter was not set prior to upgrading, the assistant sets COMPATIBLE to 8.0.5. However, if COMPATIBLE was set prior to upgrading, the original setting is retained after the migration. See [Chapter 8, "Compatibility and Interoperability"](#) for information about resetting the COMPATIBLE initialization parameter.

You may encounter a series of messages similar to the following during the upgrade:

```
ORA-00604: error occurred at recursive SQL level 1
ORA-00001: unique constraint (SYSTEM.AQ$_QUEUES_CHECK) violated
ORA-06512: at "SYS.DEMS_AQADM", line 2023
ORA-06512: at line 2
```

You can ignore these messages.

CAUTION: If you retain the old Oracle software, never start the upgraded database with the old software. Only start the database with the executables in the new release 8.1 installation directory.

15. At the "Listener.ora Migration Confirmation" screen, click the Yes button if you want the assistant to modify your `listener.ora` file automatically, or click the No button if you do not want the assistant to modify the `listener.ora` file.

Certain modifications are required to the `listener.ora` file for your database to work properly with Oracle Enterprise Manager. If you plan to use Oracle Enterprise Manager, you should click the Yes button to automatically modify the `listener.ora` file. However, if you do not plan to use Oracle Enterprise Manager, click the No button.

If you click the Yes button, the Oracle Data Migration Assistant modifies the `listener.ora` file in the following way:

- a. The assistant shuts down the old listener and the new Oracle8i listener.
- b. The assistant modifies the `SID_DESC` entry for the migrated database in the Oracle8i `listener.ora` in one of the following ways:

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

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

If the database name is SAL, the domain name is COM, and the Oracle home is `/oracle/product/8.1`, the assistant adds the following entry:

```
...
  (SID_DESC =
    (GLOBAL_DBNAME = sal.com)
    (ORACLE_HOME = /oracle/product/8.1)
    (SID_NAME = SAL)
  )
...
```

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

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

If `an_entry` does not match the `GLOBAL_DBNAME` of the migrated database, and if the database name is SAL, the domain name is COM, and the Oracle home is `/oracle/product/8.1`, the assistant adds the following entry:

```
...
  (SID_DESC =
    (GLOBAL_DBNAME = sal.com)
    (ORACLE_HOME = /oracle/product/8.1)
    (SID_NAME = SAL)
  )
...
```

This entry is the same as the entry in the simple case, but the assistant also adds the entry `an_entry` to the `SERVICE_NAMES` parameter. Therefore, the assistant changes the `SERVICE_NAMES` parameter to the following:

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

- c. On Windows NT, the assistant removes the entry of the migrated database from the old `listener.ora` file. The assistant does not perform this action on UNIX operating systems.
- d. The assistant starts up the Oracle8i listener.

Running the Oracle Data Migration Assistant Independently If you installed Oracle8i without specifying that you are migrating or upgrading an existing database, you can run the Oracle Data Migration Assistant independently after the Oracle8i installation is complete.

Complete the following steps to run the Oracle Data Migration Assistant independently:

1. Start the Oracle Data Migration Assistant.

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

```
odma
```

On Windows NT, select:

```
Start > Programs > Oracle - ORACLE_HOME_NAME > Migration Utilities > Oracle  
Data Migration Assistant
```

When you start the Oracle Data Migration Assistant, its welcome screen appears (see [Figure 7-1](#) on page 7-8).

2. Respond to questions in each Oracle Data Migration Assistant window, and click Next when you are ready to continue to the next window.

See Also: Step 8 to Step 15 in "[Upgrade the Database Using the Oracle Data Migration Assistant](#)" on page 7-7 for more information.

Upgrade the Database Manually

Complete the following steps to upgrade the database manually:

See Also: If you are upgrading an Oracle Parallel Server on Windows NT, see the *Oracle Parallel Server Getting Started for Windows NT* for important Operating System Dependent (OSD) layer instructions.

1. Install Oracle8i using the Oracle Universal Installer.
 - a. Follow the instructions in your operating-system specific Oracle documentation to prepare for installation and start the Oracle Universal Installer.

If you are upgrading a system with Oracle Parallel Server installed, see the *Oracle8i Parallel Server Setup and Configuration Guide* for additional installation instructions.

- b. At the Welcome screen of the Oracle Universal Installer, click Next.

If you need help at any screen or want to consult more documentation about the Oracle Universal Installer, click the Help button to open the online help.
 - c. At the File Locations screen, make sure the Destination is the complete path to the directory you want to use for your release 8.1 Oracle home. Then, click Next.

Note: If you are upgrading from an 8.0 release, you must install the new 8.1 release in an Oracle home separate from the 8.0 release. However, if you are upgrading from a previous 8.1 release, this restriction does not apply, and you can install the new release into the same Oracle home as the previous release if you wish.

- d. At the Available Products screen, select the edition of Oracle8i that you want to install, either Oracle8i Enterprise Edition or Oracle8i. Then, click Next.

Note: The Oracle Parallel Server option is available only with Oracle8i Enterprise Edition.

- e. At the Installation Types screen, choose either Custom or Minimal. Do not choose Typical unless you want to install a starter database along with your Oracle software. You can avoid installing a starter database if you select Custom or Minimal. Normally, you should not install a starter database if you are upgrading an existing database.

Note: Minimal is not supported for Oracle Parallel Server installations.

After you make your selection, click Next.

If you chose Custom, respond to the screens that enable you to specify your custom installation settings until you reach the "Upgrading or Migrating an Existing Database" screen.

Make sure you install all of the options you installed with the Oracle7 database, assuming you do not want to discontinue use of a particular option. For example, if you installed Advanced Replication in Oracle7, you should install it in Oracle8i.

- f. If you are installing Oracle Parallel Server, at the Cluster Node Selection screen, select the nodes onto which you want the software installed. Then, click Next.
- g. At the "Upgrading or Migrating an Existing Database" screen, leave the "Upgrade or Migrate an Existing Database" checkbox unselected. Then, click Next.

If you select the "Upgrade or Migrate an Existing Database" checkbox, the Oracle Data Migration Assistant is started automatically after installation. Because you are following the instructions for upgrading the database manually, you should not start the Oracle Data Migration Assistant.

Note: The Oracle Data Migration Assistant does not support Oracle Parallel Server migrations.

- h. At the Summary screen, make sure all of the settings and choices are correct for your installation. Then, click Install. The Oracle Universal Installer performs the installation, which may take some time.

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

2. If your `init sid .ora` file resides within the old environment's Oracle home, copy it to a location outside of the old environment's Oracle home. In past releases, the default location for the `init sid .ora` file was `$ORACLE_HOME/dbs` on UNIX and `$ORACLE_HOME\database` on Windows NT, but in release 8.1, the default location is `$ORACLE_BASE/admin/ sid /pfile`, where sid is the Oracle instance ID. The `init sid .ora` file can reside anywhere you wish, but it should not reside in the old environment's Oracle home after you upgrade to the new release.

If your `init sid .ora` file has an IFILE (include file) entry and the file specified in the IFILE entry resides within the old environment's Oracle home, then copy the file specified by the IFILE entry to a location outside of the old environment's Oracle home. The file specified in the IFILE entry has additional initialization parameters. After you copy this file, make the same edits to it as you do to the `init sid .ora` file, as specified in the next step.

Note: For Oracle Parallel Server, perform this step on all nodes. Also, if your `init db_name .ora` file resides within the old environment's Oracle home, move or copy the `init db_name .ora` file to a location outside of the old environment's Oracle home.

3. Adjust the `init sid .ora` file for use with the new release.

Certain initialization parameters are obsolete in the new 8.1 release. Remove all obsolete parameters from any initialization parameter file that will start a new release 8.1 instance. Obsolete parameters may cause errors. Also, alter any parameter whose syntax has changed in the new 8.1 release; refer to [Appendix B, "Changes to Initialization Parameters"](#) for lists of new, renamed, and obsolete parameters. Also, if you are using Oracle Parallel Server, see *Oracle8i Parallel Server Concepts and Administration* for more information about obsolete Oracle Parallel Server initialization parameters.

If you are updating snapshots automatically by using the `JOB_QUEUE_PROCESSES` initialization parameter, comment out this parameter in the `init sid .ora` file. Also, if you are using Advanced Queuing and have propagation schedules, comment out both the `JOB_QUEUE_PROCESSES` and `AQ_TM_PROCESSES` initialization parameters. After upgrading your database, you can remove the comments to use these parameters normally.

In release 8.1, the `LARGE_POOL_SIZE` setting may be calculated automatically by Oracle. If the automatic setting is too large, it may increase the time required to perform your upgrade. See "[Parallel Execution Allocated from Large Pool](#)" on page B-9 for more information.

If the `init sid .ora` file contains an `IFILE` entry, change the `IFILE` entry in the `init sid .ora` file to point to the new location you copied it to in Step 2.

If you are using Oracle Parallel Server, modify the `init db_name .ora` file in the same way that you modified the `init sid .ora` file.

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

Note: For Oracle Parallel Server, perform this step on all nodes. Also, set the `PARALLEL_SERVER` initialization parameter to `FALSE`. You can change it back to `TRUE` after the upgrade operation is complete.

4. Make sure that the following environment variables point to the new release 8.1 directories:
 - `ORACLE_HOME`
 - `PATH`
 - `ORA_NLS`
 - `ORACLE_BASE`
 - `LD_LIBRARY_PATH` (on UNIX)
 - `ORACLE_PATH` (if set)

Note: Some of these environment variables may not apply to your operating system.

Note: For Oracle Parallel Server, perform this step on all nodes.

See Also: Your operating-system specific Oracle8i installation documents for information about setting other important environment variables on your operating system.

5. At a system prompt, change to the `$ORACLE_HOME/rdbms/admin` directory.
6. Start Server Manager.
7. Connect to the database instance:

```
SVRMGR> CONNECT INTERNAL
```

8. Run `STARTUP RESTRICT`:

```
SVRMGR> STARTUP RESTRICT
```

You may need to use the `PFILE` option to specify the location of your `init sid .ora` file.

You may see error messages listing obsolete parameters. If so, shut down the database, edit the `init sid .ora` file to remove the parameters listed, and then run `STARTUP RESTRICT` again.

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

```
SVRMGR> SPOOL catoutu.log
```

If you want to see the output of the script you will run on your screen, you also can issue a `SET ECHO ON` statement:

```
SVRMGR> SET ECHO ON
```

10. Run `uold_release.sql` where *old_release* refers to the release you had installed prior to upgrading. See [Table 7-2](#) to choose the correct script. Each script provides a direct upgrade from the release specified in the "Old Release" column. The "Old Release" is the release from which you are upgrading.

To run a script, enter the following:

```
SVRMGR> @uold_release.sql
```

Table 7-2 Upgrade Scripts

Old Release	Run Script
8.0.1	Direct upgrade <i>is not supported</i> . See " Upgrade Paths " on page 7-2 for more information.
8.0.2	Direct upgrade <i>is not supported</i> . See " Upgrade Paths " on page 7-2 for more information.
8.0.3	<code>u0800030.sql</code>
8.0.4	<code>u0800040.sql</code>
8.0.4S	Direct upgrade <i>is not supported</i> . See " Upgrade Paths " on page 7-2 for more information.
8.0.5	<code>u0800050.sql</code>
8.1.1	Upgrading to the new release <i>is not supported</i> .
8.1.2	Upgrading to the new release <i>is not supported</i> .
8.1.3	<code>u0801030.sql</code>
8.1.4	<code>u0801040.sql</code>

Note: If the old release you had installed prior to upgrading is not listed in [Table 7-2](#), see the readme files in the new installation for the correct upgrade script to run.

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

- You must use the version of the script supplied with new release 8.1 installation.
- You must run the script in the new release 8.1 environment.
- You only need to run ONE script, even if your upgrade spans several releases. For example, if your old release was 8.0.4, then you need to run only `u0800040.sql`.

The script you run creates and alters certain dictionary tables. It also runs the `catalog.sql` and `catproc.sql` scripts that come with the release to which you are upgrading, which create the system catalog views and all the necessary packages for using PL/SQL.

If you encounter any problems when you run the script, or any of the scripts in the remaining steps, correct the causes of the problems and rerun the script. You can rerun any of the scripts described in this chapter as many times as necessary.

See Also: ["Running Scripts"](#) on page 1-3 for information about the types of errors to look for when you run a script.

You may encounter a series of messages similar to the following during the upgrade:

```
ORA-00604: error occurred at recursive SQL level 1
ORA-00001: unique constraint (SYSTEM.AQ$_QUEUES_CHECK) violated
ORA-06512: at "SYS.DBMS_AQADM", line 2023
ORA-06512: at line 2
```

You can ignore these messages.

Note: If the upgrade script runs for an inordinately long time, it may be caused by a setting for `LARGE_POOL_SIZE` that is too large for your installation. Use the `V$PARAMETER` view to check the setting for `LARGE_POOL_SIZE`, and if it is too large, set it to a smaller value in your `initsid.ora` file. See ["Parallel Execution Allocated from Large Pool"](#) on page B-9 for more information.

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

```
SVRMGR> SPOOL OFF
```

Then, check the spool file and verify that the packages and procedures compiled successfully. You named the spool file in Step 9; the suggested name was `catoutu.log`. Correct any problems you find in this file.

If you specified `SET ECHO ON`, you may want to `SET ECHO OFF` now:

```
SVRMGR> SET ECHO OFF
```

12. Run `ALTER SYSTEM DISABLE RESTRICTED SESSION`:

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

13. Run `SHUTDOWN` on the Oracle8i database:

```
SVRMGR> SHUTDOWN IMMEDIATE
```

CAUTION: 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 upgraded Oracle8i database.

Your database is now upgraded to the new 8.1 release.

CAUTION: If you retain the old Oracle software, never start the upgraded database with the old software. Only start the database with the executables in the new release 8.1 installation directory.

Upgrading Specific Components

Some components of the Oracle database server require an upgrade operation separate from the general database upgrade operation. This section contains information about upgrading specific components. You should perform the actions described in these sections only after you have upgraded the database by following the instructions in "[Upgrading the Database to the New Oracle8i Release](#)" on page 7-3.

Some of the upgrading procedures below involve Export/Import. See *Oracle8i Utilities* for Export/Import instructions.

Upgrading Advanced Replication

If the Oracle system has Advanced Replication installed, complete the following steps:

1. Shut down the database if it is running:

```
SVRMGR> SHUTDOWN IMMEDIATE
```

Note: For Oracle Parallel Server, set the `PARALLEL_SERVER` initialization parameter to `FALSE`. You can change it back to `TRUE` after the upgrade operation is complete.

2. At a system prompt, change to the `$ORACLE_HOME/rdbms/admin` directory.
3. Start Server Manager.
4. Connect to the database instance:

```
SVRMGR> CONNECT INTERNAL
```

5. Run `STARTUP RESTRICT`:

```
SVRMGR> STARTUP RESTRICT
```

You may need to use the `PFILE` option to specify the location of your `initsid.ora` file.

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

```
SVRMGR> SPOOL catoutrep.log
```

If you want to see the output of the script you will run on your screen, you also can issue a SET ECHO ON statement:

```
SVRMGR> SET ECHO ON
```

7. Run `catrep.sql`:

```
SVRMGR> @catrep.sql
```

8. Run the `rold_release.sql` script where `old_release` refers to the release you had installed prior to upgrading. See [Table 7-3](#) to choose the correct script. Each script provides a direct upgrade for Advanced Replication from the release specified in the "Old Release" column. The "Old Release" is the release from which you are upgrading

To run a script enter the following:

```
SVRMGR> @rold_release.sql
```

Table 7-3 Upgrade Scripts for Advanced Replication

Old Release	Run Script
8.0.1	Direct upgrade <i>is not supported</i> . See " Upgrade Paths " on page 7-2 for more information.
8.0.2	Direct upgrade <i>is not supported</i> . See " Upgrade Paths " on page 7-2 for more information.
8.0.3	<code>r0800030.sql</code>
8.0.4	<code>r0800040.sql</code>
8.0.4S	Direct upgrade <i>is not supported</i> . See " Upgrade Paths " on page 7-2 for more information.
8.0.5	<code>r0800050.sql</code>
8.1.1	Upgrading to the new release <i>is not supported</i> .
8.1.2	Upgrading to the new release <i>is not supported</i> .
8.1.3	No upgrade script is required. Advanced Replication will work with the new release 8.1 database.
8.1.4	No upgrade script required. Advanced Replication will work with the new release 8.1 database.

Note: If the old release you had installed prior to upgrading is not listed in [Table 7-3](#), see the readme files in the new installation for the correct upgrade script to run for Advanced Replication.

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

- You must use the version of the script supplied with new release 8.1 installation.
- You must run the script in the new release 8.1 environment.
- You only need to run ONE script, even if your upgrade spans several releases. For example, if your old release was 8.0.4, then you need to run only `r0800040.sql`.

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

```
SVRMGR> SPOOL OFF
```

Then, check the spool file and verify that the packages and procedures compiled successfully. You named the spool file in Step 6; the suggested name was `catoutrep.log`. Correct any problems you find in this file.

If you specified `SET ECHO ON`, you may want to `SET ECHO OFF` now:

```
SVRMGR> SET ECHO OFF
```

10. Run ALTER SYSTEM DISABLE RESTRICTED SESSION:

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

Advanced Replication is upgraded to the new release.

Upgrading Oracle Parallel Server

If the Oracle system has Oracle Parallel Server installed, complete the following steps:

1. Shut down all instances using SHUTDOWN IMMEDIATE:

```
SVRMGR> SHUTDOWN IMMEDIATE
```

Note: For Oracle Parallel Server, set the PARALLEL_SERVER initialization parameter to FALSE. You can change it back to TRUE after the upgrade operation is complete.

2. At a system prompt, change to the \$ORACLE_HOME/rdbms/admin directory.
3. Start Server Manager.
4. Connect to the database instance:

```
SVRMGR> CONNECT INTERNAL;
```

5. Run STARTUP RESTRICT:

```
SVRMGR> STARTUP RESTRICT
```

You may need to use the PFILE option to specify the location of your *initsid.ora* file.

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

```
SVRMGR> SPOOL catoutpar.log
```

If you want to see the output of the script you will run on your screen, you also can issue a SET ECHO ON statement:

```
SVRMGR> SET ECHO ON
```

7. Run `catparr.sql`:

```
SVRMGR> @catparr.sql
```

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

```
SVRMGR> SPOOL OFF
```

Then, check the spool file and verify that the packages and procedures compiled successfully. You named the spool file in Step 6; the suggested name was `catoutpar.log`. Correct any problems you find in this file.

If you specified `SET ECHO ON`, you may want to `SET ECHO OFF` now:

```
SVRMGR> SET ECHO OFF
```

9. Run ALTER SYSTEM DISABLE RESTRICTED SESSION:

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

Oracle Parallel Server is upgraded to the new release.

Upgrading Snapshots

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

1. Grant query rewrite privileges to the owner of the snapshot. Only local snapshots are available for query rewrite.
2. Issue the `ALTER SNAPSHOT ... ENABLE QUERY REWRITE` command on the snapshot.

For example, on a snapshot named `SSORDERS`, issue the following command:

```
ALTER SNAPSHOT ssorders ENABLE QUERY REWRITE;
```

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

Upgrading the Advanced Queuing Option

The following sections describe the actions required to upgrade the Advanced Queuing (AQ) option.

Use the Extended Address Field

Note: This section only applies to systems that were upgraded from release 8.0.3 of Oracle. If you never ran release 8.0.3 of Oracle on your current system, you *do not* need to perform the procedure in this section, and you can move on to "[Upgrade Your Queue Tables](#)" on page 7-29.

Release 8.0.4 introduced an extended address field in the AQ\$_AGENT datatype; the address field was extended to 1024 bytes. If you installed release 8.0.3, you must perform the procedure in this section to use the extended address field.

Also, if you installed release 8.0.3 but performed the steps described in this section to extend the address field when you upgraded to a prior release, such as release 8.0.4, you need not perform the steps below. Or, if you do not plan to use the AQ, you need not perform these steps.

However, if you installed release 8.0.3, you have not performed these steps in a prior release, and you want to use the extended address field, you should perform this procedure now. Oracle Corporation recommends using the extended address field.

To use the extended address field, complete the following steps:

1. Shut down the database:

```
SVRMGR> SHUTDOWN IMMEDIATE
```

Note: For Oracle Parallel Server, set the PARALLEL_SERVER initialization parameter to FALSE. You can change it back to TRUE after the upgrade operation is complete.

2. In the `initsid.ora` file, comment out the `JOB_QUEUE_PROCESSES` and `AQ_TM_PROCESSES` initialization parameters. You can remove the comments on these parameters after this procedure is complete.
3. At a system prompt, change to the `$ORACLE_HOME/rdbms/admin` directory.

4. Start Server Manager.
5. Connect to the database instance:

```
SVRMGR> CONNECT INTERNAL;
```

6. Run STARTUP RESTRICT:

```
SVRMGR> STARTUP RESTRICT
```

You may need to use the PFILE option to specify the location of your `initsid.ora` file.

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

```
SVRMGR> SPOOL catoutaq.log
```

If you want to see the output of the script you will run on your screen, you also can issue a SET ECHO ON statement:

```
SVRMGR> SET ECHO ON
```

8. Export the contents of all existing queue tables using the Export utility.

To determine the existing queue tables in the database, issue the following SQL statement:

```
SELECT owner, queue_table FROM dba_queue_tables;
```

You also must export the SYSTEM.DEF\$_AQCALL and SYSTEM.DEF\$_AQERROR queue tables and then import them in Step 12. These default queue tables are used by Advanced Replication.

See Also: *Oracle8i Application Developer's Guide - Advanced Queuing* for more information about the required procedure for exporting queue tables.

9. Use the DBMS_AQADM.DROP_QUEUE_TABLE procedure to drop all of your queue tables. Make sure you drop the SYSTEM.DEF\$_AQCALL and SYSTEM.DEF\$_AQERROR queue tables.

See Also: *Oracle8i Application Developer's Guide - Advanced Queuing* for information about the DBMS_AQADM.DROP_QUEUE_TABLE procedure.

- 10.** Run `catnoqueue.sql` to drop the existing AQ dictionary tables:

```
@catnoqueue.sql
```

- 11.** Run `catqueue.sql` to redefine the new types and dictionary tables:

```
@catqueue.sql
```

- 12.** Import the queue tables you exported in Step 8 using the Import utility.

- 13.** Check that all queue tables exported in Step 8, including the `SYSTEM.DEF$_AQCALL` and `SYSTEM.DEF$_AQERROR` queue tables, were properly imported in Step 12 by issuing the following SQL statement:

```
SELECT owner, queue_table FROM dba_queue_tables;
```

The following is an example of the output you should see when you issue this SQL statement:

OWNER	QUEUE_TABLE
SYSTEM	DEF\$_AQCALL
SYSTEM	DEF\$_AQERROR
AQUSER1	QUEUE_TABLE1
AQUSER2	QUEUE_TABLE2

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

```
SVRMGR> SPOOL OFF
```

Then, check the spool file and verify that the packages and procedures compiled successfully. You named the spool file in Step 7; the suggested name was `catoutaq.log`. Correct any problems you find in this file.

If you specified `SET ECHO ON`, you may want to `SET ECHO OFF` now:

```
SVRMGR> SET ECHO OFF
```

- 15.** Run `ALTER SYSTEM DISABLE RESTRICTED SESSION`:

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

You can now use the extended address field.

Upgrade Your Queue Tables

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

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

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

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

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

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

Note: The `COMPATIBLE` initialization parameter must be set to 8.1.0 or higher to upgrade your queue tables and to create new release 8.1 compatible queue tables.

Upgrading User-Defined Datatypes

When you upgrade your database from release 8.0 to release 8.1, the existing user-defined datatypes (such as object types, nested tables, and varrays) retain the release 8.0 representation format. The representation format is changed in release 8.1 to optimize disk space utilization for better performance.

You can continue to use the release 8.0 format in release 8.1, but then you will not benefit from the improved representation format. Therefore, upgrading your existing user-defined datatypes is not required, but it is recommended.

To use the new format for existing user-defined datatypes, complete the following steps:

1. Export all of the tables (including queue tables) containing release 8.0 user-defined object types, nested tables, and varrays using the Export utility.
2. Drop the tables you exported.
3. Make sure the compatibility level of your database is at 8.1.0 or higher by setting the COMPATIBLE initialization parameter.
4. Import the tables you exported in Step 1 using the Import utility.

Upgrading the Recovery Catalog

Your recovery catalog schema for the upgraded database resides in a database that is separate from the database you upgraded. If you upgraded the Recovery Manager executable to release 8.1, you must upgrade the recovery catalog to release 8.1 as well.

Also, if you have multiple databases of different releases managed by a single recovery catalog, you need to consider compatibility issues between a particular Recovery Manager release and the recovery catalog release. For example, release 8.1.3 and 8.1.4 of Recovery Manager cannot access a release 8.1.5 or higher recovery catalog. Therefore, in this case, you must upgrade all of the databases managed by the recovery catalog to release 8.1.5 or higher. For more information about recovery catalog compatibility with Recovery Manager, see "[Recovery Manager](#)" on page 8-39.

Complete the following steps to upgrade the recovery catalog:

1. Log in to Recovery Manager and connect to the recovery catalog.

For example, if RCAT/RCAT is the user name and password for the recovery catalog owner, and RECDB is the network service name, enter the following:

```
rman rcvcat rcat/rcat@recdb
```

The first time you connect to an older recovery catalog with the 8.1 release of Recovery Manager, you will see message RMAN-06186, indicating that the recovery catalog must be upgraded.

2. Use the UPGRADE CATALOG command to upgrade the recovery catalog to the most current release. Recovery Manager prompts you to enter the command twice to confirm the catalog upgrade. If any errors are encountered while upgrading, they are displayed in the Recovery Manager log.

Here is the log from a session that upgrades the recovery catalog from release 8.0.4:

```
Recovery Manager: Release 8.1.5.0.0
```

```
RMAN-06008: connected to recovery catalog database
```

```
RMAN-06186: PL/SQL package rcat.DBMS_RCVCAT version 08.00.04 in RCVCAT  
database is too old
```

```
RMAN> upgrade catalog
```

```
RMAN-06435: recovery catalog owner is rcat
```

```
RMAN-06442: enter UPGRADE CATALOG command again to confirm catalog upgrade
```

```
RMAN> upgrade catalog
```

```
RMAN-06408: recovery catalog upgraded to version 08.01.05
```

Recompiling Invalid PL/SQL Modules

The `utlrbp.sql` script recompiles all existing PL/SQL modules that were previously in an INVALID state, such as packages, procedures, types, etc. These actions are optional; however, they ensure that the cost of recompilation is incurred during installation rather than in the future.

To run the `utlrbp.sql` script, complete the following steps:

1. At a system prompt, change to the `$ORACLE_HOME/rdbms/admin` directory.
2. Start Server Manager.
3. Connect to the database instance:

```
SVRMGR> CONNECT INTERNAL;
```

4. Run `utlrbp.sql`:

```
SVRMGR> @utlrbp.sql
```

Oracle Corporation highly recommends running `utlrbp.sql`.

After Upgrading to the New Release

The following sections provide information about actions you may need to perform after you upgrade to the new 8.1 release.

Using the New TO_LOB Operator

In release 8.1, a new SQL operator, TO_LOB, copies data from a LONG column in one table to a LOB in another table.

See Also: ["Copying LONGs to LOBs"](#) on page 9-7 for detailed instructions.

Checking for Bad Date Constraints

A bad date constraint involves invalid date manipulation, which is a date manipulation 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.

Note: If you already ran `utlconst.sql` after you migrated or upgraded to a previous 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 the `utlconst.sql` script, complete the following steps:

1. At a system prompt, change to the `$ORACLE_HOME/rdbms/admin` directory.
2. Start Server Manager.
3. Connect to the database instance:

```
SVRMGR> CONNECT INTERNAL;
```

4. Enter the following:

```
SVRMGR> SPOOL utlresult.log
SVRMGR> @utlconst.sql
SVRMGR> SPOOL OFF
```

After you run the script, the `utlresult.log` file includes all the constraints that have invalid date constraints. The `utlconst.sql` script does not correct bad constraints, but instead disables them. You should either drop the bad constraints or recreate them after you make the necessary changes.

Avoiding Problems with Parallel Execution

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

- `SHARED_POOL_SIZE`
- `LARGE_POOL_SIZE`

See Also: ["Parallel Execution Allocated from Large Pool"](#) on page B-9 for information about adjusting these parameters.

Adjusting Your `INITSID.ORA` File for the New Release

Each new release of Oracle introduces new initialization parameters, changes some parameters, and obsoletes some parameters. You should adjust your `initSID.ora` file to account for these changes and to take advantage of new initialization parameters that may be beneficial to your system.

See Also: [Appendix B, "Changes to Initialization Parameters"](#) for lists of the new, changed, and obsoleted initialization parameters in release 8.1, and see *Oracle8i Reference* for detailed information about each parameter.

The `COMPATIBLE` initialization parameter controls the compatibility level of your database. Set the `COMPATIBLE` initialization parameter in your `initSID.ora` file based on the compatibility level you want for your upgraded database.

See Also: ["Setting the COMPATIBLE Parameter"](#) on page 8-6 for information.

Changing the Password for the OUTLN User

The OUTLN user is created automatically during installation of Oracle8i. This user has DBA privileges. Use the ALTER USER command to change the password for this user. Oracle8i adds the OUTLN user schema to support Plan Stability. The OUTLN user acts as a place to centrally manage metadata associated with stored outlines.

Modify Your listener.ora File

If you are using Oracle Enterprise Manager to manage database objects, your `listener.ora` file must be configured with information about the database in the following manner:

```
sid_list_listener_name =
  (sid_list =
    (sid_desc =
      (global_dbname = global_database_name)
      (oracle_home = oracle_home)
      (sid_name = sid)
    )
  )
```

GLOBAL_DBNAME identifies the global database name, which is a name comprised of the database name and domain name of the database. ORACLE_HOME identifies the directory where your Oracle software is installed, and SID_NAME identifies the Oracle System Identifier of the database instance.

For example, the following information is for a database with a listener name of LISTENER. The global database name of SAL.COM is referenced by an instance with a SID of SAL, using the software installed in the `/oracle/product/8.1` directory:

```
sid_list_listener =
  (sid_list =
    (sid_desc =
      (global_dbname = sal.com)
      (oracle_home = /oracle/product/8.1)
      (sid_name = sal)
    )
  )
```

If you are not using Oracle Enterprise Manager, you do not need to configure information about the database, because an Oracle8i database instance registers itself with the listener automatically.

See Also: *Net8 Administrator's Guide* for more information about configuring the `listener.ora` file and the `SID_LIST` parameter.

Drop Java Objects

If you upgraded from release 8.1.4, and you created Java objects before upgrading, you must drop these objects. The Java format changed in release 8.1.5, causing incompatibilities between release 8.1.4 Java objects and release 8.1.5 and higher Java objects. If you upgraded from an 8.0 release, you do not need to complete this procedure.

To drop the Java objects created in release 8.1.4, complete the following steps:

1. At a system prompt, change to the `$ORACLE_HOME/rdbms/admin` directory.
2. Start SQL*Plus and connect as a user with SYS privileges.
3. Enter the following:

```
SQL> SPOOL utljavaout.log
SQL> @utljavarm.sql
SQL> SPOOL OFF
```

Check the spool file and verify that the statements executed successfully.

4. At a system prompt, change to the `$ORACLE_HOME/javavm/install` directory.
5. Start SQL*Plus and connect as a user with SYS privileges.
6. Enter the following:

```
SQL> SPOOL initjvmout.log
SQL> @initjvm.sql
SQL> SPOOL OF
```

Check the spool file and verify that the statements executed successfully.

If you exported your Java source, class, and resource objects before you upgraded, you can import them using the Import utility after you have completed all of the steps in the above procedure.

Note: Exporting and importing Java objects between pre-release 8.1.5 databases and the current release is not guaranteed to be successful.

See Also: *Oracle8i Utilities* for information about exporting and importing Java source, class, and resource objects.

Changing the Word-Size of Your Current Release

The instructions in this section guide you through changing the word-size of your current release (switching from 32-bit software to 64-bit software or vice versa).

See Also: ["Changing Word-Size"](#) on page 1-4 for more information about changing word-size.

Complete the following steps to change the word-size of your current release:

1. Start Server Manager.
2. Connect to the database instance:

```
SVRMGR> CONNECT INTERNAL;
```

3. Run SHUTDOWN IMMEDIATE on the database:

```
SVRMGR> SHUTDOWN IMMEDIATE
```

Note: For Oracle Parallel Server, issue this command for all instances. Also, set the `PARALLEL_SERVER` initialization parameter to `FALSE`. You can change it back to `TRUE` after the upgrade operation is complete.

4. Perform a full offline backup of the database.

See Also: *Oracle8i Backup and Recovery Guide* for more information.

5. If you currently have a 32-bit installation, install the 64-bit version of the same release. Or, if you currently have a 64-bit installation, install the 32-bit version of the same release.

Note: Installation is operating-system specific. For installation instructions, see your Oracle8i operating-system specific installation documentation and the Oracle8i README for your platform.

6. At a system prompt, change to the `$ORACLE_HOME/rdbms/admin` directory.
7. Start Server Manager.
8. Connect to the database instance:

```
SVRMGR> CONNECT INTERNAL;
```

9. Run `STARTUP RESTRICT`:

```
SVRMGR> STARTUP RESTRICT
```

You may need to use the `PFILE` option to specify the location of your `initsid.ora` file.

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

```
SVRMGR> SPOOL catoutw.log
```

If you want to see the output of the script you will run on your screen, you also can issue a `SET ECHO ON` statement:

```
SVRMGR> SET ECHO ON
```


11. Run utlirp.sql:

```
SVRMGR> @utlirp.sql
```

The `utlirp.sql` script recompiles existing PL/SQL modules in the format required by the new database. This script first alters certain dictionary tables. Then, it reloads package STANDARD and DBMS_STANDARD, which are necessary for using PL/SQL. Finally, it triggers a recompile of all PL/SQL modules, such as packages, procedures, types, etc.

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

```
SVRMGR> SPOOL OFF
```

Then, check the spool file and verify that the packages and procedures compiled successfully. You named the spool file in Step 10; the suggested name was `catoutw.log`. Correct any problems you find in this file.

If you specified SET ECHO ON, you may want to SET ECHO OFF now:

```
SVRMGR> SET ECHO OFF
```

13. Run ALTER SYSTEM DISABLE RESTRICTED SESSION:

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

The word-size of your database is changed. You can open the database for normal use.

Compatibility and Interoperability

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

This chapter covers the following topics:

- [What Is Compatibility?](#)
- [Features Requiring 8.1.0 or Higher Compatibility Level](#)
- [What Is Interoperability?](#)
- [Compatibility and Interoperability Issues](#)

See Also: *Getting to Know Oracle8i* for information about desupported features.

What Is Compatibility?

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

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

The COMPATIBLE Parameter

Oracle enables you to control the compatibility of your database with the COMPATIBLE initialization parameter. By default, when the COMPATIBLE parameter is not set in your `init sid .ora` file, it defaults to the lowest possible setting for the release, which is 8.0.0 for all 8.0 and 8.1 releases. You cannot use new features that would make your database incompatible with release 8.0.0 until you reset the COMPATIBLE parameter to a higher value.

This default behavior has the following advantages:

- Because compatibility with your previous release is maintained by default, it is easier to downgrade.
- If you are operating in an environment with more than one database, your upgraded database remains compatible with databases that have not yet been upgraded.



Of course, the major disadvantage of the default setting is that many of the features of the new release are not available to you if you leave the COMPATIBLE parameter unset.

See Also: ["Features Requiring 8.1.0 or Higher Compatibility Level"](#) on page 8-9 for a list of these features in the new release.

Depending on the products you chose to install during your release 8.1 installation of Oracle, the Oracle Universal Installer may set the COMPATIBLE parameter to a higher value, such as 8.1.0. Check your `init sid .ora` file if you are unsure of the current setting for the COMPATIBLE parameter.

Figure 8–1 illustrates the default settings and the possible settings for release 8.0 and release 8.1 of Oracle.

Figure 8–1 The COMPATIBLE Parameter

Default 8.0.0  Release 8.0 Can be set to 8.0.x only	Default 8.0.0  Release 8.1 Can be set to 8.1.y or 8.0.x
Lowest Possible Setting: 8.0.0	Lowest Possible Setting: 8.0.0
Highest Possible Setting: Your current release	Highest Possible Setting: Your current release
Cannot be set to: <ul style="list-style-type: none"> • Any Oracle7 or lower • Any release higher than current, including 8.1.0 or higher 	Cannot be set to: <ul style="list-style-type: none"> • Any Oracle7 or lower • Any release higher than current, including 8.2.0 or higher

How the COMPATIBLE Parameter Operates

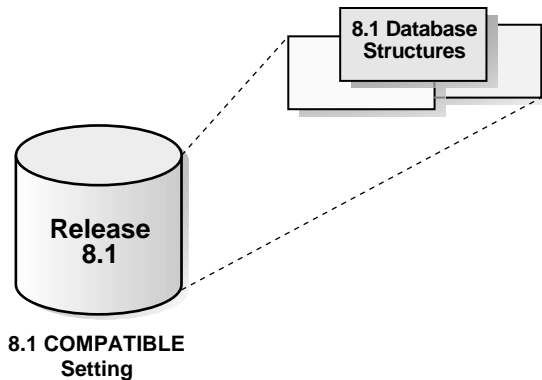
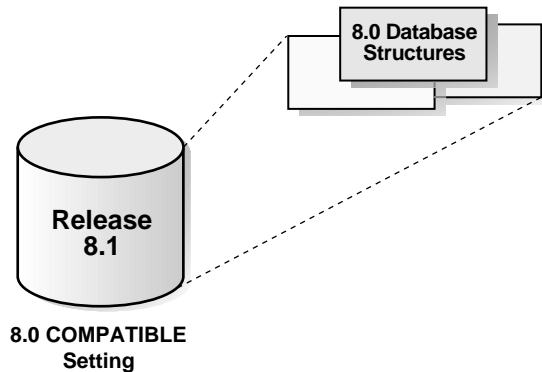
The COMPATIBLE parameter operates in the following way:

- It controls the behavior of your Oracle database. For example, if you run a release 8.1 database with the COMPATIBLE parameter set to 8.0.0, then the release 8.1 database generates release 8.0 compatible database structures on disk. Therefore, the COMPATIBLE setting enables or disables the use of features. If you try to use any of the new features that make the database incompatible with the COMPATIBLE parameter setting, an error is returned.

However, any new features that do not make incompatible changes on disk are enabled.

- It makes sure that the database is compatible with its setting. If the database becomes incompatible with its setting, the database does not start and terminates with an error. If this happens, you must set the COMPATIBLE parameter in your `init sid .ora` file to an appropriate value for the database.

Figure 8–2 Database Structures Depend on the COMPATIBLE Parameter Setting



See Also: *Oracle8i Concepts* for more information about database structures.

Downgrading and Compatibility

Once you upgrade or migrate to a new release, you can set the COMPATIBLE parameter to match the new release. Doing so enables you to use all of the features of the new release, but may make it more difficult for you to downgrade to your previous release. If you want to downgrade, you must remove all of the incompatibilities with the release to which you are downgrading, which is a process that may require a great deal of time and effort.

See Also: [Chapter 12, "Downgrading to an Older Version 8 Release"](#) for more information about downgrading.

Compatibility Level

The compatibility level of your database corresponds to your COMPATIBLE parameter setting. For example, if you set the COMPATIBLE parameter to 8.1.5, the database runs at 8.1.5 compatibility level.

Checking Your Current COMPATIBLE Parameter Setting

To check your current COMPATIBLE parameter setting, issue the following command:

```
SELECT name, value, description FROM v$parameter
       WHERE name='compatible';
```

When to Set the COMPATIBLE Parameter

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

Note: After the migration, upgrade, or downgrade procedure is complete, you can set the COMPATIBLE parameter whenever necessary.

Setting the COMPATIBLE Parameter

Complete the following steps to set the COMPATIBLE parameter:

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

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

See Also: *Oracle8i Backup and Recovery Guide* for more information about performing a backup.

2. If you are changing the COMPATIBLE parameter to a lower setting, make sure your database does not have any incompatibilities with the intended lower setting. Alternately, if you are changing the COMPATIBLE parameter to a higher setting, skip to Step 4.

If you plan to lower the COMPATIBLE parameter to an 8.0.x setting, see "[Remove Incompatibilities](#)" on page 12-2 and follow the instructions in all of the sub-sections for removing incompatibilities with 8.0 releases.

Also, if you created your database at 8.1.0 compatibility level or higher, Oracle created certain system-defined types that are incompatible with 8.0 releases. To remove these incompatibilities, run the `utldst.sql` script supplied with release 8.1:

```
@utldst.sql
```

3. If you are changing the COMPATIBLE parameter to a lower setting, run ALTER DATABASE RESET COMPATIBILITY:

```
ALTER DATABASE RESET COMPATIBILITY;
```

Alternately, if you are changing the COMPATIBLE parameter to a higher setting, skip to Step 4.

See Also: "[About ALTER DATABASE RESET COMPATIBILITY](#)" on page 8-7 for more information.

4. Shutdown the database:

```
SHUTDOWN IMMEDIATE
```


5. Edit the `init sid .ora` file to enter or change the COMPATIBLE setting.

For example, to set the COMPATIBLE parameter to 8.1.0, enter the following in the `init sid .ora` file:

```
COMPATIBLE=8.1.0
```

CAUTION: If you have migrated to release 8.0 or higher, you cannot set the COMPATIBLE parameter to any Oracle7 release. Also, you cannot set the COMPATIBLE parameter to a release higher than your current release.

6. Start the database using STARTUP.

About ALTER DATABASE RESET COMPATIBILITY

You use the ALTER DATABASE RESET COMPATIBILITY statement to instruct Oracle that you want to change the compatibility level to a lower release. Some Oracle features, such as temporary tables for example, require a compatibility level of 8.1.0 or higher. If you set the COMPATIBLE parameter to 8.1.0 or higher and then create a temporary table, the temporary table is an 8.1.0 compatible object in the database.

ALTER DATABASE RESET COMPATIBILITY checks for each feature that may have created an object that is incompatible with the lowest possible compatibility level, which is 8.0.0. If the check indicates that no incompatible objects exist for a certain feature, the compatibility level of the feature is set to 0.0.0, which means that the feature is not in use. If, however, the check indicates that incompatible objects created by a certain feature exist, the compatibility level for that feature is set to the required compatibility level.

For example, if one or more temporary tablespaces exist, the compatibility level for the temporary tablespaces feature is set to 8.1.0, because 8.1.0 is the required compatibility level for that feature. It is important to understand, however, that ALTER DATABASE RESET COMPATIBILITY cannot raise the compatibility level of your database. You must first set the COMPATIBLE parameter to a higher value, such as 8.1.0, before you can create database objects that require 8.1.0 compatibility level.

If you close the database, reset the `COMPATIBLE` parameter to a lower setting, and then open the database, Oracle checks the compatibility level of each feature. If a feature has a compatibility level higher than the compatibility level specified by the `COMPATIBLE` parameter in the `init sid .ora` file, the database fails to open and displays an error message indicating the incompatible feature or features.

Most importantly, if you remove all of the incompatibilities that exist in your database, but fail to run the `ALTER DATABASE RESET COMPATIBLE` statement before shutting down the database, the database will still fail to open, even if no incompatibilities exist. The database will fail to open because it was not instructed to check the compatibility level of each feature against the objects that exist in the database. Because it did not reset the compatibility level for these features, Oracle simply remembers that incompatible objects were created at some time in the past. Running `ALTER DATABASE RESET COMPATIBLE` statement instructs Oracle to check for incompatible objects explicitly, and resets the compatibility level if no incompatible objects exist.

Features Requiring 8.1.0 or Higher Compatibility Level

To use the features listed in the following tables, the `COMPATIBLE` parameter must be set to 8.1.0 or higher.

The features listed *do not* represent a complete list of the new features introduced in release 8.1. Instead, the features listed are only those new release 8.1 features that require an 8.1.0 or higher compatibility level; some new features do not require this compatibility level.

See Also: *Getting to Know Oracle8i* for more information about the features listed below and for information about other new release 8.1 features. You also can check the *Oracle8i Server Documentation Master Index* for entries relating to the new features listed below.

Applications

Table 8–1 Applications: Features Requiring 8.1.0 or Higher Compatibility Level

Functional Area	Features Requiring 8.1.0 or Higher Compatibility Level
Java	Enterprise JavaBeans Java code in stored procedures SQLJ Translator
Oracle Call Interface (OCI)	Support for Client Notification

Tablespaces

Table 8–2 Tablespaces: Features Requiring 8.1.0 or Higher Compatibility Level

Functional Area	Features Requiring 8.1.0 or Higher Compatibility Level
Tablespaces	Locally Managed Tablespaces Online Read-Only Tablespaces Transportable Tablespaces

Schema Objects

Table 8–3 Schema Objects: Features Requiring 8.1.0 or Higher Compatibility Level

Functional Area	Features Requiring 8.1.0 or Higher Compatibility Level
Tables	Drop Column Support Temporary Tables
Index-Organized Tables	Key Compression Support for Index-Organized Tables Secondary Indexes on Index-Organized Tables LOBs in Index-Organized Tables Partitioning of Index-Organized Tables
Indexes	Bitmap Index Protection (the ALTER TABLE command no longer invalidates bitmap indexes) Extensible Indexing Function-Based Indexes Index Segment Coalesce Key Compression Support for Indexes Online Index (Re)build
Hash Clusters	Single-Table Hash Clusters

Partitioning

Table 8–4 Partitioning: Features Requiring 8.1.0 or Higher Compatibility Level

Functional Area	Features Requiring 8.1.0 or Higher Compatibility Level
Partitioning of Tables	Partitioning of Index-Organized Tables Partitioning of Tables with LOBs Partitioning of Object Tables Partitioning of Tables with User-Defined Types, including Columns with the following Types: object, REF, VARRAY, and nested table Partitioning of Tables Using Composite Methods and Hash Methods
Partitioning of Hash Clusters	Support for Hash Partitioning

Built-In Datatypes

Table 8–5 Built-In Datatypes: Features Requiring 8.1.0 or Higher Compatibility Level

Functional Area	Features Requiring 8.1.0 or Higher Compatibility Level
LOBs (large objects))	Partitioning of Tables with LOBs Temporary LOBs Varying-Width Character Sets for CLOBs and NCLOBs
ROWIDs	UROWIDs (universal rowids) Note: An 8.1.0 compatibility level is required to use the new UROWID datatype as part of a database object, such as a table. However, UROWID variables can be used in PL/SQL code on a release 8.1 database with any 8.0 compatibility level.

User-Defined Datatypes

Table 8–6 User-Defined Datatypes: Features Requiring 8.1.0 or Higher Compatibility Level

Functional Area	Features Requiring 8.1.0 or Higher Compatibility Level
Nested Tables	Collection Locators Nested Table Data in Index-Organized Tables Triggers on Nested Table View Columns User-Specified Storage Clauses for Nested Tables
Object Identifiers	User-Defined Object Identifiers
REFs	Referential Integrity Constraint on a REF Column
Varrays	Storage Parameters for Storing Varrays as LOBs Varray Data in Index-Organized Tables

Oracle Parallel Server

Table 8–7 Oracle Parallel Server: Features Requiring 8.1.0 or Higher Compatibility Level

Functional Area	Features Requiring 8.1.0 or Higher Compatibility Level
Oracle Parallel Server	Instance Affinity for Jobs

Data Protection

Table 8–8 Data Protection: Features Requiring 8.1.0 or Higher Compatibility Level

Functional Area	Features Requiring 8.1.0 or Higher Compatibility Level
Security	Application Context Fine-Grained Access Control N-Tier Authentication and Authorization
Database Backup and Recovery	Fast-Start On-Demand Rollback Proxy Copy Support for the Oracle Media Management API

Distributed Databases

Table 8–9 Distributed Databases: Features Requiring 8.1.0 or Higher Compatibility Level

Functional Area	Features Requiring 8.1.0 or Higher Compatibility Level
Advanced Replication	Column Level Snapshot Subsetting for Updatable Snapshots
Heterogeneous Services	Agent Self-Registration

Data Warehousing

Table 8–10 Data Warehousing: Features Requiring 8.1.0 or Higher Compatibility Level

Functional Area	Features Requiring 8.1.0 or Higher Compatibility Level
Summary Management Using Materialized Views	Dimensions Schema Object Query Rewrite Rewrite Privileges for Query Rewrite

Data Access

Table 8–11 Data Access: Features Requiring 8.1.0 or Higher Compatibility Level

Functional Area	Features Requiring 8.1.0 or Higher Compatibility Level
SQL and PL/SQL	<ul style="list-style-type: none"> Autonomous Transactions Bulk Binds C Call Specifications CALL Statement Native Dynamic SQL NOCOPY Parameter Passing Mode The <code>utlchain1.sql</code> and <code>utlexcpt1.sql</code> Scripts
Advanced Queuing	<ul style="list-style-type: none"> Addition of the Original Message ID Column for Propagated Messages Addition of a Sender's ID Column Database Event Publication Instance Affinity for Queue Tables Non-Persistent Queues Queue Level and System Level Privileges Rules Based Subscriptions Separate Storage of History Management Information Note: Propagation requires a compatibility level of 8.0.4 or higher.
Packages	DBMS_REPAIR Package
Constraints	DISABLE VALIDATE Constraint State
Triggers	<ul style="list-style-type: none"> Object OutBinds in Triggers Triggers on Nested Table View Columns Triggers on DATABASE and SCHEMA Triggers with a CALL to a Procedure as the Trigger Body
The Optimizer	<ul style="list-style-type: none"> Extensible Optimizer Optimizer Plan Stability
Database Resources	Database Resource Manager

Spatial and Visual Information

Table 8–12 Spatial and Visual Information: Features Requiring 8.1.0 or Higher Compatibility Level

Functional Area	Features Requiring 8.1.0 or Higher Compatibility Level
Spatial	Spatial Indextype Spatial Operators
Visual Information Retrieval (VIR)	VIR Indextype VIR Operators

Note: Spatial and VIR require that the COMPATIBLE parameter be set to 8.1.0 or higher *before* you install them.

What Is Interoperability?

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

For example, a distributed database application cannot expect an Oracle7 database to understand the object SQL extensions that are available only with release 8.0 and higher.

Note: This definition of interoperability is appropriate for this *Oracle8i Migration* book because this book documents migrating, upgrading, and downgrading between different versions and releases of Oracle. However, other Oracle documentation may use a broader definition of the term *interoperability*; for example, in some cases, interoperability may describe communication between different hardware platforms and operating systems.

Compatibility and Interoperability Issues

The following sections describe compatibility and interoperability issues and the actions you can take to prevent problems resulting from these issues. The issues discussed in these sections occur because of differences between Oracle releases.

Applications

You do not need to modify existing (Oracle7 and release 8.0) applications that do not use new release 8.1 features. Existing applications should achieve the same, or enhanced, functionality on release 8.1. To increase the likelihood that applications running against your release 8.1 database will continue to work if you downgrade to release 8.0, you can set the COMPATIBLE parameter to 8.0.5 or lower.

However, the COMPATIBLE parameter only restricts the use of release 8.1 features that change the formatting on disk, not the use of other release 8.1 features. Therefore, a setting of 8.0.5 or lower does not guarantee that applications developed in release 8.1 will run correctly if the database is downgraded to release 8.0.

See Also: [Chapter 9, "Upgrading Your Applications"](#) for more information about upgrading applications.

General Compatibility and Interoperability Issues for Applications

This section describes general compatibility and interoperability issues for applications.

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

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

- If the application uses release 8.0 or higher OCI rowid descriptors (OCIROWID * for Pro*C and SQL-ROWID for Pro*COBOL), the application should continue to function properly without any changes.
- If the application always performs DESCRIBE on the host variables, the application should continue to function properly without any changes. Make sure the application can accommodate the new SQLT_RDD datatype.

- If the application uses `SQLT_RID` host variables, you must rewrite the application to use `VARCHAR` host variables or rowid descriptors. Rowid descriptors are preferred.
- If the application uses `CHARACTER` host variables, the behavior also depends on the size of the host variables. If the size can accommodate the primary key and if the variable is a variable length string, the application should continue to function properly without any changes. However, if the application uses a fixed size 18 character string, you must change the application to use longer variable strings or OCI descriptors.

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

Incompatibility with Release 7.1 XA Calls The Oracle8i database does not support release 7.1 XA calls, but it does support release 7.2 and 7.3 calls. Therefore, after migrating a release 7.1 database to Oracle8i, relink associated Tuxedo applications (and any other associated applications that use XA calls) with the Oracle8i XA libraries. This relinking is not required if you migrate a release 7.2 or 7.3 database.

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

OCI Applications

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

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

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

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

Thread Safety 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 multithreaded applications.

OCI Application Link Line For OCI applications, the Oracle8*i* link line differs from the Oracle7 link line. See the `$ORACLE_HOME/rdbms/demo/demo_rdbms.mk` file for examples of using the Oracle8*i* link line as an Oracle8*i* OCI application is compiled.

Oracle7 Clients Oracle7 clients can make selective use of Oracle8*i* OCI, combining Oracle7 and Oracle8*i* calls. The degree of functionality added depends on which calls are used. The encryption API and password reset calls are independently usable as well. Use Oracle8*i* OCI for all phases of the statements being processed to enable the following functionality:

- failover
- prefetch
- piggybacked commit or cancel
- client-side conversions

Oracle7 clients must log in using Oracle8*i* calls if they want to combine Oracle7 code with Oracle8*i* code.

Precompiler Applications

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

See Also: The *Pro*C/C++ Precompiler Programmer's Guide* and the *Pro*COBOL Precompiler Programmer's Guide* for more information.

Connecting With SYSDBA Privileges in Pro*C/C++ SYSDBA privileges are no longer available by default when you issue the CONNECT statement in Pro*C/C++. In release 8.0, the following CONNECT statement connected with SYSDBA privileges in Pro*C/C++:

```
EXEC SQL CONNECT :sys IDENTIFIED BY :sys_passwd;
```

In release 8.1, issue the following CONNECT statement to connect with SYSDBA privileges in Pro*C/C++:

```
EXEC SQL CONNECT :sys IDENTIFIED BY :sys_passwd IN SYSDBA MODE;
```

Connecting With SYSDBA Privileges in Pro*COBOL SYSDBA privileges are no longer available by default when you issue the CONNECT statement in Pro*COBOL. In release 8.0, the following CONNECT statement connected with SYSDBA privileges:

```
EXEC SQL
    CONNECT :sys IDENTIFIED BY :SYS-PASSWD
END-EXEC.
```

In release 8.1, issue the following CONNECT statement to connect with SYSDBA privileges:

```
EXEC SQL
    CONNECT :sys IDENTIFIED BY :SYS-PASSWD IN SYSDBA MODE
END-EXEC.
```

Ada Support in Version 8 The Pro*ADA product was officially desupported by Oracle in release 7.3. You can upgrade Pro*ADA to the latest release of SQL*Module for Ada 8.1, which has a number of new features. However, SQL*Module for ADA 8.1 does not provide object support.

PL/SQL Backward Compatibility and Precompilers PLSQL_V2_COMPATIBILITY backward compatibility behavior (see "[PL/SQL Applications](#)" on page 8-20) is available in the precompiler environment by setting the DBMS precompiler command line option as follows:

```
... DBMS=Oracle7
```

PL/SQL Applications

This section includes compatibility and interoperability issues for PL/SQL applications.

See Also: The *PL/SQL User's Guide and Reference* for more information.

PL/SQL V2 Compatibility Mode The PL/SQL V2 compatibility mode is available in PL/SQL release 8.0 and higher. This mode is enabled by the `PLSQL_V2_COMPATIBILITY` initialization parameter.

You can set PL/SQL V2 compatibility mode in any one of the following three ways:

- Add the following line to your `init sid .ora`:

```
PLSQL_V2_COMPATIBILITY = TRUE
```

- Issue the following SQL statement:

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

- Issue the following SQL statement:

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

The `PLSQL_V2_COMPATIBILITY` initialization parameter provides compatibility between PL/SQL release 8.0 and higher and PL/SQL V2 in the following situations:

- The PL/SQL V2 compiler allows a record type or index table type to be referenced before its definition in the source. PL/SQL release 8.0 and higher strictly requires that the type definition precede reference to the type in the source. However, when you enable PL/SQL V2 compatibility mode, PL/SQL release 8.0 and higher behaves the same as PL/SQL V2 regarding type definitions.
- The PL/SQL V2 compiler allows the following illegal syntax:

```
return variable-expression
```

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

```
return variable-type
```

The PL/SQL release 8.0 and higher compiler issues an error when it encounters the illegal syntax. However, when you enable PL/SQL V2 compatibility mode,

PL/SQL release 8.0 and higher behaves the same as PL/SQL V2 and does not issue an error.

- In PL/SQL V2 it is possible to modify or delete elements of an index table passed in as an IN parameter, as in the following example:

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

This use of an IN parameter is incorrect. PL/SQL release 8.0 and higher correctly enforces the read-only semantics of IN parameters and does not let index table methods modify index tables passed in as IN parameters. However, when you enable PL/SQL V2 compatibility mode, PL/SQL release 8.0 and higher behaves the same as PL/SQL V2 and allows the parameter.

- PL/SQL V2 allows the passing (as an OUT parameter) of fields of IN parameters that are records, but PL/SQL release 8.0 and higher does not allow this type of passing. However, when you enable PL/SQL V2 compatibility mode, PL/SQL release 8.0 and higher behaves the same as PL/SQL V2 and allows this type of passing.
- The PL/SQL V2 compiler permits fields of OUT parameters that are record variables to be used in expression contexts (for example, in a dot-qualified name on the right-hand side of an assignment statement).

This use of OUT parameters should not be permitted. PL/SQL release 8.0 and higher does not permit OUT parameters to be used in expression contexts. However, when you enable PL/SQL V2 compatibility mode, PL/SQL release 8.0 and higher behaves the same as PL/SQL V2 in this regard.

- PL/SQL V2 allows OUT parameters in the FROM clause of a SELECT list. PL/SQL release 8.0 and higher does not allow this use of OUT parameters. However, when you enable PL/SQL V2 compatibility mode, PL/SQL release 8.0 and higher behaves the same as PL/SQL V2 in this regard.

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:

Table 8–13 Keyword Behavior Differences

Keywords	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

Startup and Shutdown

This section describes compatibility and interoperability issues related to starting up, connecting to, and shutting down an Oracle database.

Shutdown Changes

In release 8.0.4 and higher, idle server processes are killed during SHUTDOWN IMMEDIATE; consequently, the error(s) returned to users are different than in past releases.

Initialization Parameters

See [Appendix B, "Changes to Initialization Parameters"](#) for lists of new, changed, and obsoleted parameters. Also, see "[Compatibility Issues with Initialization Parameters](#)" on page B-8 for information about compatibility issues related to specific initialization parameters.

Tablespaces and Datafiles

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

Transportable Tablespace

There are compatibility issues when you transport a tablespace between two databases.

See Also: *Oracle8i Administrator's Guide* for information about these compatibility issues.

Tempfiles

Release 8.1 introduces tempfiles. The information about tempfiles is in different static data dictionary views and dynamic performance views than the information about datafiles. To view information about tempfiles, consult the `DBA_TEMP_FILES` static data dictionary view and the following dynamic performance views:

- `V$TEMPFILE`
- `V$TEMP_EXTENT_MAP`
- `V$TEMP_EXTENT_POOL`
- `V$TEMP_SPACE_HEADER`
- `V$TEMPSTAT`
- `V$TEMP_PING`

Oracle automatically assigns numbers to both datafiles and tempfiles. Two datafiles cannot share the same number; similarly, two tempfiles cannot share the same number. However, a tempfile and a datafile can share the same number.

Active Transactions Restriction for Read-Only Tablespaces

In releases prior to release 8.1, there could not be any active transactions in your database before you made a tablespace read-only. In release 8.1, this restriction is lifted if the `COMPATIBLE` parameter is set to 8.1.0 or higher. With the database at 8.1.0 or higher compatibility level, the `ALTER TABLESPACE ... READ ONLY` command waits for active transactions to complete, and then makes the tablespace read-only. If, however, the `COMPATIBLE` parameter is set below 8.1.0, the restriction still applies.

Data Dictionary

This section describes possible compatibility and interoperability issues resulting from data dictionary changes.

See Also: [Appendix C, "Changes to Static Data Dictionary Views"](#) and [Appendix D, "Changes to Dynamic Performance Views"](#) for more information, including lists of obsolete views.

Data Dictionary Protection

The new Oracle8i data dictionary protection mechanism may cause problems in any applications that create user tables in the SYS schema and access them using the 'ANY' privileges. For example, the user must have DELETE CATALOG ROLE to use the DELETE statement to purge the audit records in the AUD\$ table.

Creating and accessing user tables in SYS schema is not secure. Therefore, applications are expected to move the objects to a different schema. Use the O7_DICTIONARY_ACCESSIBILITY initialization parameter for temporary compatibility, 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 DBA privilege to a normal user, who will connect to the database as SYSDBA to connect to SYS schema.

Obsolete Data Dictionary Views

Certain data dictionary views maintained in Oracle7 for backward compatibility to Oracle version 5 and version 6, created in the files `catalog5.sql` and `catalog6.sql`, are obsolete in version 8. Remove all references to these data dictionary views from your database tools and applications.

Schema Objects

This section describes compatibility and interoperability issues relating to schema objects.

Bitmap Index Protection

In releases prior to release 8.1, it was possible to unintentionally invalidate bitmap indexes by issuing certain SQL statements. The most common causes of bitmap index invalidation were the following types of statements:

- ALTER TABLE statements that define a primary key on an existing table that did not have a primary key previously.
- ALTER TABLE statements that define a NOT NULL constraint on a column that did not have this constraint previously.

Release 8.1 eliminates these unintentional invalidations if the COMPATIBLE parameter is set to 8.1.0 or higher. However, if the COMPATIBLE parameter is set lower than 8.1.0, bitmap index protection is not enabled, and certain SQL statements, such as the ones described earlier, may invalidate a bitmap index.

Datatypes

This section describes compatibility and interoperability issues relating to datatypes.

Large Objects (LOBs)

This section describes compatibility and interoperability issues relating to LOBs.

Varying-Width Character Sets for CLOBs and NCLOBs Release 8.0 did not allow users other than SYSTEM to create tables with the CLOB or NCLOB datatype if the database character set was varying-width. Release 8.1 supports CLOB and NCLOB datatypes in tables with a varying-width character set, and the data is stored as UCS2 (2-byte fixed-width unicode).

This functionality is restricted in the following ways if the COMPATIBLE parameter is set below 8.1.0:

- Non-SYSTEM users cannot create a table that contains the CLOB or NCLOB datatype if the database character set is varying-width.
- SYSTEM users can create a table that contains the CLOB or NCLOB datatype, but cannot store data in the CLOB or NCLOB columns.

If COMPATIBLE is set to 8.1.0 or higher, these restrictions do not apply.

LOB Index Clause The LOB index clause will be de-supported in a future release of Oracle. If you used the LOB index clause to store LOB index data in a tablespace separate from the tablespace used to store the LOB, the index data will be relocated to reside in the same tablespace as the LOB if you perform either of the following actions in release 8.1:

- perform an Export/Import on the LOB
- exchange the LOB into a partitioned table

If you used Export/Import to migrate from Oracle7 to Oracle8i, then the index data was relocated automatically during migration. However, the index data was not relocated if you used the Migration utility.

Also, if you create a new table in release 8.1 and specify a tablespace for the LOB index for a non-partitioned table, the tablespace specification will be ignored and the LOB index will be located in the same tablespace as the LOB.

To check the storage of LOB indexes, issue the following SQL statement connected as SYSDBA:

```
SELECT index_name, index_type, tablespace_name
       FROM dba_indexes
       WHERE index_type = 'LOB';
```

Oracle ROWIDs

This section describes compatibility and interoperability issues related to rowids.

New Physical ROWID Datatype Format The format for physical rowids has changed in version 8. If you use physical rowids stored in columns or in application code, the old physical rowids are invalid and must be converted.

See Also: [Chapter 11, "Migration Issues for Physical Rowids"](#) for more information about the new physical rowid format.

UROWID Datatype The UROWID (universal rowid) datatype is a new feature introduced in release 8.1. Pre-release 8.1 clients can access columns of UROWID datatype using character host variables only; other types of variables are not supported.

NCHAR and NLS Use

In version 8, you 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 either the PROPS\$ or the VALUE\$ dictionary view may return the CHARACTER_SET_NAME column or the NLS_NCHAR_CHARACTERSET row.

Migration Issues with NCHAR and NLS The PROPS\$ dictionary table contains two rows that describe the character sets specified in the CREATE DATABASE statement. The row holding NAME='NLS_CHARACTERSET' has the database character set's name in the VALUE\$ column. The row holding NAME='NLS_NCHAR_CHARACTERSET' has the national character set's name in the VALUE\$ column.

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 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. The bind variable 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 Oracle7 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 Oracle7 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 Oracle7 client using SQL code embedded in PL/SQL cannot use the CSCONVERT() function, nor CHR() with a second argument, directly in the SQL statement because these do not exist in a version 6 or Oracle7 client's PL/SQL package STANDARD. The following two stored procedures can be created on the server to correct this incompatibility:

```
CREATE OR REPLACE FUNCTION to_nchar_cs(t VARCHAR2) RETURN NCHAR VARYING
IS BEGIN
    RETURN csconvert(t, nchar_cs);
END;
/
```

```
CREATE OR REPLACE FUNCTION to_char_cs(t nchar varying) RETURN VARCHAR2
IS BEGIN
    RETURN csconvert(t, char_cs);
END;
/
```

NCHAR and NLS Environment Variables and Compatibility You should set NLS_LANG to your environment as follows:

- set the ORA_NLS32 environment variable for the release 7.3.x environment
- set the ORA_NLS33 environment variable for the version 8 environment

Verify that the client has the correct NLS character set environment variables. An error is generated when release 7.3 NLS code tries to load a version 8 character set.

User-Defined Datatypes

Release 8.1 introduces a new format for user-defined datatypes. The new format can result in significant performance improvements over the format used in release 8.0. To use the new user-defined datatypes format, the COMPATIBLE parameter must be set to 8.1.0 or higher.

You can use release 8.0 user-defined datatypes in a release 8.1 database without causing compatibility problems. However, the database will not realize the performance gains possible with the new format.

Release 8.1 Clients Accessing Release 8.0 User-Defined Datatypes

The user-defined datatypes format is negotiated as part of the compatibility exchange between the client and server. If you are using a release 8.0 database server, or a release 8.1 database server with the COMPATIBLE parameter set to 8.0.x, release 8.1 clients can access the database, but they are set to release 8.0.

Release 8.0 Clients Accessing Release 8.1 User-Defined Datatypes

When a release 8.0 client accesses a server with release 8.1 user-defined datatypes, the database converts the user-defined datatypes to release 8.0 format. Consequently, the release 8.0 client can access the data, but performance may suffer.

Nested Tables

Release 8.0 clients do not support the following release 8.1 nested table features:

- collection locators
- user-specified storage for collection columns, including storage of nested table data in an index-organized table

Therefore, access fails with an incompatibility error when a release 8.0 client attempts to access a release 8.1 server and a nested table is specified to be returned as a locator, or the storage for the nested table is user-specified.

Varrays Stored as LOBs

Release 8.0 clients do not support specifications of storage parameters for storing varrays as LOBs. Therefore, access fails with an incompatibility error when a release 8.0 client attempts to access a release 8.1 server where there is a specification of storage parameters for storing a varray as a LOB.

SQL and PL/SQL

This section describes compatibility and interoperability issues relating to SQL and PL/SQL.

See Also: *Oracle8i SQL Reference* and *PL/SQL User's Guide and Reference* for more information about SQL and PL/SQL.

Functions GREATEST_LB, LEAST_UB, and TO_LABEL Desupported

With release 8.1.5 and higher of Oracle8i, the built-in PL/SQL functions GREATEST_LB, LEAST_UB, and TO_LABEL, which operate on Trusted Oracle labels, are no longer supported.

Native Dynamic SQL in PL/SQL

The following sections describe interoperability issues related to native dynamic SQL in PL/SQL:

Server-Side PL/SQL An Oracle database server at release 8.1.0 or higher compatibility level can execute native dynamic SQL statements that contain references to objects on a remote server at any compatibility level.

For example, the following procedure contains a native dynamic SQL statement and links to a remote Oracle database server:

```
PROCEDURE dyn1 is
BEGIN
    EXECUTE IMMEDIATE 'insert into tab@remote_link
        values ('a', 10)';
END;
```

In the example, *remote_link* can be a link to any version of Oracle, such as release 7.3, 8.0, or 8.1.

Native Dynamic SQL and RPC Calls PL/SQL programs that are targets of RPC calls can use native dynamic SQL, regardless of the release of the clients making the RPC calls. For example, release 7.3 or 8.0 clients can issue RPC calls to an Oracle database server at release 8.1.0 or higher compatibility level.

SQL Scripts `utlchain.sql` and `utlchain1.sql`

The release 8.1 installation includes the following two scripts for creating a table that stores migrated and chained rows: `utlchain.sql` and `utlchain1.sql`. The `utlchain1.sql` script can be run on index-organized tables as well as regular tables, while the `utlchain.sql` script can be run only on regular tables, but not on index-organized tables.

The correct script to run depends on the compatibility level of your database:

- Run `utlchain.sql` if the compatibility level is lower than 8.1.0.
- Run `utlchain1.sql` if the compatibility level is 8.1.0 or higher.

SQL Scripts `utlexcpt.sql` and `utlexcpt1.sql`

The release 8.1 installation includes the following two scripts for creating a table that stores exceptions from enabling constraints: `utlexcpt.sql` and `utlexcpt1.sql`. The `utlexcpt1.sql` script can be run on index-organized tables as well as regular tables, while the `utlexcpt.sql` script can be run only on regular tables, but not on index-organized tables.

The correct script to run depends on the compatibility level of your database:

- Run `utlexcpt.sql` if the compatibility level is lower than 8.1.0.
- Run `utlexcpt1.sql` if the compatibility level is 8.1.0 or higher.

Behavior Change in Parallel CREATE TABLE Statements with the AS Subquery

In release 8.0 or higher, if you use the `PARALLEL` clause in a `CREATE TABLE` statement with the `AS` subquery, Oracle ignores the `INITIAL` storage parameter and instead uses the `NEXT` storage parameter. Oracle7 did not ignore the `INITIAL` storage parameter.

For example, consider the following SQL statement:

```
CREATE TABLE tb_2 STORAGE (INITIAL 1M NEXT 500K)
  PARALLEL (DEGREE 2)
  AS SELECT * FROM tb_1;
```

In release 8.0 or higher, the value of `INITIAL` is 500K, while, in Oracle7, the value of `INITIAL` is 1M.

Advanced Queuing (AQ)

This section includes compatibility and interoperability issues for AQ.

See Also: The *Oracle8i Application Developer's Guide - Advanced Queuing* for more information about AQ. The sections below only provide compatibility and interoperability information about new AQ features, while the *Oracle8i Application Developer's Guide - Advanced Queuing* provides detailed information about using them.

Queue Level and System Level Privileges

To use queue level and system level privileges, the queue table must be at 8.1.0 compatibility level or higher. Specifically, to grant queue level privileges using the following procedures in the DBMS_AQADM package requires an 8.1.0 or higher queue table compatibility level:

- GRANT_QUEUE_PRIVILEGE
- REVOKE_QUEUE_PRIVILEGE

Interoperability and the Sender's ID Column

In release 8.1, the sender's ID is mapped as an additional attribute in the message properties. This new attribute is ignored when there is communication between release 8.0 and release 8.1 databases.

For OCI applications, the sender's ID attribute is available as a new attribute in the message properties descriptor. Release 8.1 OCI clients use a new RPC code to send and receive the message properties to and from the server.

Rule Based Subscriptions

When you migrate a queue table from release 8.0 to release 8.1 using the DBMS_AQADM.MIGRATE_QUEUE_TABLE procedure, any existing subscribers are upgraded automatically to subscribers with null rules.

Message Streaming

Message streaming is supported only if the source and destination databases both are release 8.1 or higher. A COMPATIBLE parameter setting of 8.1.0 is not required for message streaming; it is supported even if COMPATIBLE is set to 8.0.5 or lower on a release 8.1 database.

Procedures and Packages

This section describes compatibility and interoperability issues related to procedures and packages.

The DBMS_LOB Package and NOCOPY

If the COMPATIBLE parameter is set to 8.1.0 or higher, the DBMS_LOB package uses the new NOCOPY syntax for the LOB parameters, and LOB locators that are passed to the DBMS_LOB package follow the new NOCOPY semantics.

If the COMPATIBLE parameter is set lower than 8.1.0, the NOCOPY syntax is not supported. Therefore, if you are at an 8.0.x compatibility level, you should not:

- use the new NOCOPY syntax in the DBMS_LOB package
- run the new `dbmslob.sql` script

The DBMS_REPAIR Package

The COMPATIBLE parameter must be set to 8.1.0 or higher to use the DBMS_REPAIR package. The DBMS_REPAIR package will fail if the compatibility level is below 8.1.0.

Syntax Change for the SET_SESSION_LONGOPS Procedure

Version 8 introduces changes to the DBMS_APPLICATION_INFO.SET_SESSION_LONGOPS procedure. For information about the new syntax, refer to the `dbmsapin.sql` file. If any of your applications use this procedure, change the applications accordingly.

Oracle Optimizer

Setting the COMPATIBLE parameter to 8.1.0 or higher may enable the optimizer to improve its choice of execution plan. An 8.1.0 compatibility level enables the optimizer to use a new column, AVGCLN, in the HIST_HEAD\$ data dictionary table to determine its choice of execution plan.

Oracle Parallel Server

Support for different releases of Oracle within one Oracle Parallel Server environment is operating-system specific. See your operating-system specific Oracle documentation for information about whether or not the co-existence of different releases within one Oracle Parallel Server environment is supported on your operating system.

In release 8.0 and later, all Oracle instances that belong to a database and are linked in Parallel Server mode to be run on a hardware cluster must match the word-size of the GMS executable. Therefore, they must all run a 32-bit executable, or they must all run a 64-bit executable.

Also, mixing word-sizes of Oracle Parallel Server executables across different databases is not supported in release 8.0, but this restriction does not apply to Oracle executables that are not linked in Oracle Parallel Server mode. In release 8.1, the GMS process is eliminated, and therefore this restriction is relaxed. Oracle instances that belong to different databases may run with different word-sizes in release 8.1.

INSTANCES Keyword Will Be Removed from PARALLEL Clause

In a future release of Oracle, the INSTANCES keyword of the PARALLEL clause will be obsoleted. The INSTANCES keyword still can be used in release 8.1, but it will be interpreted differently than in past releases. In Oracle7 and release 8.0, the INSTANCES keyword could be used in the PARALLEL clause of commands such as the following:

- ALTER CLUSTER
- ALTER DATABASE ... RECOVER
- ALTER INDEX ... REBUILD
- ALTER TABLE
- CREATE CLUSTER
- CREATE INDEX
- CREATE TABLE

It also could be used in hints. The INSTANCES keyword was used to specify the number of Oracle Parallel Server instances to use in a parallel operation.

Also beginning with release 8.1, the syntax for specifying degree in a PARALLEL clause has changed. You can specify degree simply by placing a number after PARALLEL, as in the following example:

```
ALTER TABLE emp PARALLEL 5;
```

However, the keyword DEGREE remains valid if you choose to use it. The preceding syntax is equivalent to the following statement:

```
ALTER TABLE emp PARALLEL (DEGREE 5 INSTANCES 1);
```

Regardless of the syntax, the value you specify is the number of query threads used in a parallel operation. Neither syntax will affect how many instances are used to execute a query. The system will determine how many instances to use based on the instances available and the load on each of the instances. So, either syntax will produce the same result.

Continuing to Use the INSTANCES Keyword in Release 8.1 You still can use the old syntax to specify both INSTANCES and DEGREE in release 8.1, but Oracle interprets it as single keyword that specifies the degree. Therefore, the obsolete command syntax still is accepted in release 8.1, but its interpretation may be different than in past releases. [Table 8-14](#) illustrates the way in which Oracle interprets the possible settings of INSTANCES and DEGREE if you continue to use the obsolete syntax. The columns in [Table 8-14](#) represent the following:

- The **Degree** column indicates the setting for the DEGREE keyword in the PARALLEL clause.
- The **Instances** column indicates the setting for the INSTANCES keyword in the PARALLEL clause.
- The **8.1 Degree** column indicates Oracle's interpretation of the degree in release 8.1 based on the DEGREE and INSTANCES settings.

Table 8-14 Conversion of INSTANCES Keyword in Release 8.1

Degree	Instances	8.1 Degree
Unset or 1	Unset or 1	1
x	DEFAULT	x
x	Unset or 1	x
Unset or 1	DEFAULT	DEFAULT
DEFAULT	y	y
Unset or 1	y	y
DEFAULT	Unset or 1	DEFAULT
x	y	x * y

In the table, x and y are variables representing an integer value.

If you leave a keyword unset, Oracle uses 1 as its value.

The following scenarios illustrate the way Oracle may behave differently in release 8.1 because of these interpretations:

- Setting DEGREE to x and INSTANCES to 1 does not guarantee that parallel operations use only one instance.
- Setting DEGREE to 1 and INSTANCES to y does not guarantee that parallel operations use only one query thread per instance.
- Setting DEGREE to x and INSTANCES to y does not guarantee either setting. Instead, Oracle attempts to set the degree to x multiplied by y.

Oracle Corporation recommends that you discontinue use of the INSTANCES keyword to avoid unexpected behavior. Also, consider using the PARALLEL_INSTANCE_GROUP initialization parameter.

See Also: *Oracle8i SQL Reference* for more information about the PARALLEL clause and about the PARALLEL_INSTANCE_GROUP initialization parameter.

Database Security

This section describes compatibility and interoperability issues relating to database security.

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 the password has expired and is still within the grace period. If the password has expired but is still within the grace period, signal a warning to the user and use the *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 *OCIChangePassword()* is not used to change the password, then the password verification routine will not check if the new password is equal to the old password and will not reject the change if they are same.

If you do not make these changes to Oracle7 applications, one of the Oracle tools, such as SQL*Plus, will be required to allow the password change after a user's account expires.

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 Oracle7 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 also can be imported/exported in version 8.

Oracle7 or Lower Client with Release 8.0 or Higher Server Oracle7 clients use Oracle7 OCI calls to connect to the server; therefore, release 8.0 and higher password expiration cannot be detected. However, other features of release 8.0 and higher password management work for Oracle7 clients. Full release 8.0 and higher password management, including password expiration handling, can operate in Oracle7 clients after you make the minor change of replacing their Oracle7 log in call with the release 8.0 and higher log in call.

Release 8.0 or Higher Client with Oracle7 or Lower Server A release 8.0 or higher client can be coded to work with Oracle7 or lower servers. An example of the code for such clients follows:

```
OCISessionBegin(...) /* call release 8.0 and higher logon OCI call */
if (SUCCESS_WITH_INFO) then
{ /* Check for password expiration and take appropriate action*/
...
OCIChangePassword(...);
...
}
```

Enterprise User Management

This section includes compatibility and interoperability issues related to enterprise user management.

Note: The global user functionality that was available in release 8.0 is being modified, and is currently available to beta customers only. This functionality will be part of Oracle8i in a future release.

Interoperability with Release 8.0 Release 8.0 servers cannot share global users and roles with release 8.1 servers. In addition, current user database links between release 8.0 and 8.1 are not supported.

Interoperability with Oracle7 and Version 6 Releases Because global users cannot be created or authorized on version 7 or version 6 servers, those servers cannot share global users or roles with version 8. Also, current user database links from version 8 to version 6 or version 7 are not supported.

Database Backup and Recovery

This section describes compatibility and interoperability issues related to database backup and recovery.

Recovery Manager

The Recovery Manager executable must be the same release as the database on which it operates. Therefore, a release 8.0 Recovery Manager executable cannot be used on a release 8.1 database, and a release 8.1 Recovery Manager executable cannot be used on a release 8.0 database.

The Recovery Catalog In general, Recovery Manager is compatible with the same or higher release of the recovery catalog and the target database. However, there are some exceptions to that general rule, especially when a beta release is involved. The following table illustrates Recovery Manager compatibility with the recovery catalog and the target database.

Table 8–15 Recovery Manager Compatibility

Recovery Manager Release	Compatible Recovery Catalog Release(s)	Compatible Target Database Release(s)
8.1.5	8.1.5 or higher	8.1.5 or higher
8.1.4 beta	8.1.4 beta only	8.1.4 beta only
8.1.3 beta	beta releases 8.1.3 or 8.1.4	beta releases 8.1.3 or 8.1.4
8.0.5	8.0.5 or higher <i>except for</i> beta releases 8.1.3 and 8.1.4	8.0.5 or higher <i>except for</i> beta releases 8.1.3 and 8.1.4
8.0.4	8.0.4 or higher <i>except for</i> beta releases 8.1.3 and 8.1.4	8.0.4 or higher <i>except for</i> beta releases 8.1.3 and 8.1.4
8.0.3	8.0.3 only	8.0.3 only

See Also: ["Upgrading the Recovery Catalog"](#) on page 7-30 for the information about upgrading the recovery catalog.

Recovery Manager Commands Release 8.1 of Recovery Manager introduces changes to some Recovery Manager commands. However, all commands used in prior releases will continue to work with release 8.1 of Recovery Manager.

For example, the **clone** command is changed to the **duplicate** command, but the **clone** command will continue to work. Also, the **clone** option of the **allocate** and **connect** commands is now the **auxiliary** option, but the **clone** option will continue to work. Similarly, the **clonename** keyword in the **copy** and **set** commands is now **auxname**, but the **clonename** keyword will continue to work.

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 communicate with third party storage subsystems, such as Legato or EMC. Investments in tape subsystem management modules for EBU and Oracle7 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 Oracle7 generally are not useful for version 8 restores.

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

Datafile Backups

A datafile backup taken with Oracle7 cannot be restored with any release of version 8, with the following exception: a backup of an Oracle7 database taken after running the Migration utility can be restored and recovered with any release of version 8. If EBU is used to backup the Oracle7 database, and the database must later be restored for recovery with version 8, you must use EBU to restore the datafiles prior to recovering them with version 8. If the Oracle7 database is backed up with operating system commands to disk files, then those disk files can be registered with Recovery Manager by using the **catalog datafilecopy** command.

A datafile backup taken with version 8 can be restored and recovered with any later release of version 8, if a direct upgrade path between the release that backed-up the file and the release that recovers the file is supported in [Table 7-1, "Upgrade Paths"](#) on page 7-2. You can also restore and recover version 8 backups with an earlier release of version 8 if the datafile contents are compatible with the earlier release.

Standby Database

Standby database operates only on release 7.3 and higher of Oracle. The following compatibility restrictions apply to standby databases:

- The primary and standby databases should run on the same operating system. In addition, Oracle Corporation recommends that the primary and standby databases run on the same release of the operating system.
- The primary and standby databases must run the same maintenance release of Oracle. For example, if your primary database is running release 8.0.5, the standby database can run any production 8.0 release, such as release 8.0.3, 8.0.4, or 8.0.5. However, in this case, the standby database cannot run Oracle7 or release 8.1.

See Also: Your operating-system specific Oracle documentation for more information about operating system requirements for standby database.

Migrating a Standby Database to Oracle8i To migrate the Standby Database from Oracle7 to Oracle8i, perform the following steps:

1. Apply all redo logs created under Oracle7.
2. Make sure the primary database has been opened successfully under Oracle8i.
3. Migrate the standby database to Oracle8i.
4. Create a control file for your standby database by issuing the ALTER DATABASE CREATE STANDBY CONTROLFILE AS *file_name* command, which creates a modified copy of the primary database's control file.
5. Copy the control file you created in Step 4 to the standby database site.

6. Copy the initial datafile from the primary database to the standby database site. If you are unsure about which datafile is the initial datafile, the following SQL statement provides that information:

```
SELECT file_name, file_id FROM dba_data_files WHERE file_id = 1;
```

See Also: *Oracle8i Backup and Recovery Guide* for more information about Standby Database.

Fast-Start On-Demand Rollback and Fast-Start Parallel Rollback

As part of the recovery process, after a session or instance is abnormally terminated, Oracle rolls back uncommitted transactions. Oracle8i has two new features to improve rollback performance: fast-start on-demand rollback and fast-start parallel rollback.

When a dead transaction holds a row lock on a row that another transaction needs, fast-start on-demand rollback automatically recovers the data block required by the new transaction. Other data blocks and transactions that do not block any new transaction's progress are rolled back in the background. Fast-start on-demand rollback is enabled only when you set the COMPATIBLE initialization parameter to 8.1.0 or higher.

Fast-start parallel rollback improves background rollback performance by recovering each dead transaction using multiple server processes. You can use fast-start parallel rollback when the COMPATIBLE initialization parameter is set to any 8.0 or 8.1 release. Fast-start parallel rollback recovers each dead transaction using multiple server processes only if the following conditions are met:

- There are enough server processes to allocate one or more processes to each dead transaction.
- If COMPATIBLE is set to an 8.0 release, the transaction was created with multiple server processes. If a transaction was created with only one server process, only one server process is used in the rollback operation. This restriction does not apply if COMPATIBLE is set to 8.1.0 or higher.

See Also: *Oracle8i Concepts* for more information about fast-start on-demand rollback.

Archiving of Redo Logs

Release 8.1 enables you to archive online redo log files to multiple destinations, including to a local disk-based file or to a specified standby database. The compatibility and interoperability issues described in this section may arise because of this new functionality.

Re-Archiving Previously Archived Online Redo Logs Prior to release 8.1, it was possible to re-archive an online redo log that already had been successfully and fully archived. In addition, it was possible to re-archive redo log files to successfully archived destinations.

Starting with release 8.1, the following restrictions apply:

- Successfully archived online redo logs cannot be re-archived.
- Successfully archived destinations cannot be re-archived.

Archive Operation Error Detection Behavior Prior to release 8.1, when any error was detected, an archive operation stopped immediately, reported the error to the alert log, and signaled the error to the user.

Starting with release 8.1, an archive operation does not stop processing unless all of the archive destinations cannot be processed. An error at one or more destinations does not stop the archive operation; the archive operation only stops if all archive destinations cannot be processed.

LogMiner

LogMiner runs in a release 8.1 or higher instance and can analyze redo log files from any database that meets the following criteria:

- has the same DBCS (Database Character Set) as the analyzing Oracle instance
- is running on the same hardware platform as the analyzing Oracle instance
- is a release 8.0 or higher database

LogMiner does not require a mounted database to analyze redo log files. However, to fully translate the contents of the redo log files, LogMiner requires access to a LogMiner dictionary (catalog). LogMiner uses the dictionary to translate internal object identifiers and data types to object names and external data formats. You can use the PL/SQL package `DBMS_LOGMNR_D` to extract a database dictionary into an external file for later use in analyzing redo log files. Without a dictionary, LogMiner returns the internal object identifiers and presents data as hex bytes.

Analyzing Archived Redo Log Files from Other Databases You can run LogMiner on an instance of a database while analyzing redo log files from a different database. To analyze archived redo log files from other databases, LogMiner must:

- access a dictionary file that is both created from the same database as the redo log files and created with the same database character set
- run on the same hardware platform that generated the log files, although it does not need to be on the same system
- use redo log files that can be applied for recovery from Oracle release 8.0 and higher

Oracle Media Management API and Proxy Copy

Starting with Oracle Media Management API version 2, proxy copy functionality is supported. If a Recovery Manager proxy backup is attempted, and Oracle is linked with Oracle Media Management API release 1.1, or a version 2 that does not support proxy copy functionality, then Recovery Manager will return an error and the backup will fail.

Distributed Databases

This section describes compatibility and interoperability issues related to distributed databases.

Basic Replication

Prior to release 8.1, an Oracle snapshot always consisted of a snapshot base table and a view on the base table. For example, creating a snapshot SNAP_EMP creates a view SNAP_EMP and a base table normally called SNAP\$SNAP_EMP. In release 8.1, most snapshots will have only a base table with the same name as the snapshot. The view will not be created.

A view will be added to the snapshot under the following conditions:

- The snapshot was imported from a database prior to release 8.1, such as release 8.0.
- The COMPATIBLE parameter is set lower than 8.1.0.
- The snapshot requires hidden columns (that is, rowid snapshots and fast-refreshable snapshots that contain subqueries).

Advanced Replication

The following compatibility restrictions apply to a replicated environment:

- If you have a replicated environment with different releases of Oracle, you cannot replicate data that is incompatible on the lower releases. For example, in a replicated environment with a database at 8.1.0 compatibility level and another database at 8.0.0 compatibility level, you cannot replicate data between them if the data is incompatible with release 8.0.
- To improve performance and protect data integrity, a number of Advanced Replication packages that were external in pre-release 8.1 environments have been internalized in release 8.1. *Oracle8i Replication* contains a list of these internalized packages.

If one or more of your n-way master sites is a release prior to release 8.1, the `GENERATE_80_COMPATIBLE` flag must be unset or set to `TRUE` in the following procedures:

- `GENERATE_REPLICATION_SUPPORT`
- `CREATE_SNAPSHOT_REOBJECT`
- `GENERATE_SNAPSHOT_SUPPORT`

However, if your replication environment includes only masters that are running release 8.1 or higher, setting `GENERATE_80_COMPATIBLE` to `FALSE` enables you to replicate data from sites instantiated at the top flavor using the release 8.1 protocol for replication-generated RPCs.

Heterogeneous Services Agents

This section describes compatibility and interoperability issues related to Heterogeneous Services agents.

Interoperability Between Servers of Different Releases Servers at release 8.0.3 and higher can connect to and use Heterogeneous Services agents of any other server at release 8.0.3 and higher. In a connection between servers of different releases, the functionality is limited to that of the lower release.

Multithreaded Service Agents Beginning with release 8.1, multithreaded Heterogeneous Services agents are supported. If you have existing agents and you want to take advantage of the multithreaded features, create the agent initialization file and explicitly start the agents using the Agent Control Utility.

See Also: *Oracle8i Distributed Database Systems* for general information about Heterogeneous Services, and for information about creating the agent initialization file and starting the agents using the Agent Control utility.

SQL*Net or Net8

Version 7 and version 8 releases can use SQL*Net V2 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 upgraded applications:

- Both the client and server must run SQL*Net V2 or Net8.
- The multithreaded server requires SQL*Net V2 or Net8 on the server. Therefore, to connect using the multithreaded server, you also must use SQL*Net V2 or Net8 on the client.

Upgrading SQL*Net V1 to SQL*Net V2 or Net8

Make the following changes to upgrade from SQL*Net V1 to SQL*Net V2 or Net8:

- Install SQL*Net V2 or Net8.
- Re-create each connect string as the next version's connect descriptor. SQL*Net V2 uses the syntax outlined in the *SQL*Net Version 2.0 Administrator's Guide* or and Net8 uses the syntax outlined in the *Net8 Administrator's Guide*.
- Relink any precompiler programs and Oracle executables that you want to use with SQL*Net V2 or Net8, including SQL*Plus and SQL*Forms.

See Also: *SQL*Net Version 2.0 Administrator's Guide* and *SQL*Net V2 Migration Guide* for complete instructions about upgrading SQL*Net from V1 to V2. See *Net8 Administrator's Guide* for complete instructions about upgrading SQL*Net V1 to Net8.

Service Naming and Connection Load Balancing

Release 8.1 and higher supports service naming and connection load balancing for services that include more than one database instance. Each service can include multiple instances, and each instance can include multiple handlers. This support enables clients to access a service rather than a specific database instance, and logically separates the service name from any particular instance name.

To support services that include multiple instances, use the following new parameters in connect descriptors:

- SERVICE_NAME
- INSTANCE_NAME

The new parameters enable connection load balancing by taking requests through the following process:

1. A client program specifies the name of the service to which it wants to connect.
2. The TNS Listener finds the least loaded instance in the service.
3. The TNS Listener finds the least loaded handler in the instance.
4. The TNS Listener redirects the client to the optimal handler, or passes the client connection to the handler, if necessary.

To use connection load balancing, perform the following actions:

- Discontinue the use of the SID parameter in connect descriptors.
- Use the SERVICE_NAMES and INSTANCE_NAME initialization parameter in your `initsid.ora` file.
- Use the SERVICE_NAME parameter in the `tnsnames.ora` file.

Note: Before configuring the TNS Listener to handle two or more instances with the same instance name, make sure no client programs use connections based on the SID parameter.

See Also: *Net8 Administrator's Guide* for more information about using connection load balancing and the SERVICE_NAME parameter.

Export/Import

The version 8 Export utility makes dump files that are *not* downward compatible with pre-release 8.0 Import utilities. Their exported data *cannot* be imported by pre-release 8.0 Import utilities.

Starting with version 5, export dump files must be importable into all future major, patch, and maintenance releases of Oracle. [Table 8–16](#) details this support.

Table 8–16 *Export Dump File Forward Compatibility*

Dump File	Can Be Imported Into Future Releases
Version 5 and Version 6 Note: For version 5, only release 5.1.22 and higher export dump files are supported.	Oracle7, Release 8.0, Release 8.1, and all future releases
Oracle7, Release 8.0, and Release 8.1	Release 8.0, Release 8.1, and all future releases

An export dump file can be imported into the previous production release using the Export and Import utilities of the previous release. [Table 8–17](#) details this support.

Table 8–17 *Export Dump File Backward Compatibility*

Dump File Release	Can Be Imported Into Previous Release	Use Export/Import Utility
Release 8.0	Release 7.3	Release 7.3
Release 8.1	Release 8.0	Release 8.0

To export release 8.0 or higher data to an Oracle7 (or earlier) database, the Oracle7 Export utility must be used, after the `catexp7.sql` script has been run on your release 8.0 or higher database. This script resides in the `$ORACLE_HOME/rdbms/admin` directory.

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 release 8.0 or higher Import utility can use dump files generated by either Direct Path Export or Conventional Path Export. Pre-release 8.0 dump files are upward compatible with the release 8.0 or higher Import utility. Dump file data produced by current or prior versions/releases of Direct Path Export or by Conventional Path Export both have the same upward compatibility with future Oracle server versions and releases.

Export/Import Usage on Data Incompatible with Oracle7

When you export data from a release 8.0 or higher database using an Oracle7 Export utility, data that is incompatible with Oracle7 is not exported. For example, partitioned tables are not exported by the Oracle7 Export utility. If you need to move a release 8.0 or higher partitioned table to an Oracle7 database, first re-organize the table into a non-partitioned table. Similarly, if you need to move another type of incompatible data from release 8.0 or higher to Oracle7, first re-organize the data to make it compatible with Oracle7.

Miscellaneous Compatibility and Interoperability Issues

This section describes miscellaneous compatibility and interoperability issues related to your Oracle installation.

Multi-Versioning

You can run different versions of Oracle on the same computer system at the same time. However, each version can only access a database that is consistent with its version. For example, if you have version 7 and version 8 installed on the same computer system, the version 7 server can access version 7 databases but not version 8 databases, and the version 8 server can access version 8 databases but not version 7 databases.

See Also: Your operating-system specific Oracle documentation for more information about multi-versioning on your operating system. Restrictions may apply on some operating systems.

2 GB File Size Dependencies

Oracle release 8.0.4 and higher can access files that are larger than 2 GB. However, this access is subject to the following operating system dependencies:

- **File Mode:** Is the file a file system file or a raw device file? Many UNIX platforms support greater than 2 GB file sizes only on raw devices.
- **Asynchronous IO:** Does the operating system support asynchronous IO on files, for both raw and file system files? Is asynchronous IO supported for files that are greater than 2 GB?
- **Operating System Revision:** Does your operating system release number support file size greater than 2 GB? For example, in Solaris 2.5.1, a file size of greater than 2 GB is supported only on raw devices. However, in Solaris 2.6, both raw and file system files can be greater than 2 GB.
- **Operating System IO Subsystem Issues:** Does your operating system require a firmware upgrade to support file size greater than 2 GB? Because support for file size greater than 2 GB is fairly recent, many disk arrays or IO subsystems need firmware upgrades to support large files. It is important to determine from the operating system vendor which firmware patches are required for large file support.

It is very important to check these operating system dependencies before using files that are greater than 2 GB in size.

Upgrading Your Applications

This chapter describes upgrading your current applications to take advantage of new release 8.1 features. You do not need to modify existing Oracle7 and release 8.0 applications that do not use new release 8.1 features. Existing applications should achieve the same, or enhanced, functionality on a release 8.1 database.

The following topics are covered in this chapter:

- [Overview of Upgrading Applications to Oracle8i](#)
- [Upgrading OCI and Precompiler Applications](#)
- [Upgrading SQL*Plus Scripts](#)
- [Upgrading Oracle7 Forms or Developer/2000 Applications](#)
- [Copying LONGs to LOBs](#)

Overview of Upgrading Applications to Oracle8i

The following version 8 features aid in the process of upgrading applications to release 8.1:

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

Note: SQL*Net version 1, 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) 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 an Oracle7 release.
- Backward compatibility generally is supported to accommodate small incompatibilities between different Oracle releases.
- Many new features and enhancements are available automatically after migration. Some of these new features may provide improved performance.

Upgrading OCI and Precompiler Applications

If you are migrating your Oracle database from Oracle7 to Oracle8i, before you migrate the database, upgrade any precompiler and OCI applications that you plan to use with your Oracle8i databases. Then, you can test these applications on a sample Oracle8i 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 Oracle8i. 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 Oracle8i database functionality

The following sections provide more information about upgrading OCI and precompiler applications.

Upgrading OCI Applications

Application developers have the following options if they want to incorporate release 8.1 functionality into existing OCI applications:

- If your application uses Oracle7 OCI calls, completely rewrite the applications to use only new release 8.1 OCI calls.
- If your application uses Oracle7 OCI calls, incorporate release 8.1 OCI calls into the existing application, while still using Oracle7 calls for some operations.
- If your application uses only version 8 calls, incorporate the new release 8.1 OCI calls into existing applications.

See Also: See *Oracle Call Interface Programmer's Guide* for more information about writing and upgrading OCI applications.

Using Batch Error Mode for Statement Execution

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

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

Support for Client Notification

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

Support for the LISTEN Call with the Advanced Queuing Option

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

Upgrading Precompiler Applications

Complete the following steps to use your existing precompiler applications with a release 8.1 database:

1. If you want to take advantage of the new release 8.1 features in your existing application, complete one of the following actions based on whether the existing application is an Oracle7 application or a version 8 application:
 - If you have a Oracle7 application, the existing Oracle7 application may need to be recoded, or new applications written, to reflect the differences between Oracle7 and Oracle8i.
 - If you have a version 8 application and you want to take advantage of the new release 8.1 features, incorporate code for the new release 8.1 functionality into the existing version 8 application.

If you decide not to take advantage of release 8.1 features with an existing application, skip this step and move on to Step 2.

2. Relink the application with the release 8.1 runtime library, `SQLLIB`, which is included with the precompiler. This step is the only required step to use your applications with release 8.1.

Note: To run an existing Oracle7 pre-compiler application against an Oracle8i database, you do not need to re-compile, nor recompile, the application.

Simplified Upgrading of Existing Oracle7 Precompiler Applications

Applications written to work with Oracle7 precompilers, such as Pro*C 2.2, have a very smooth upgrade path to Oracle8i precompilers, such as Pro*C 8.1. Oracle7 precompiler clients work with an Oracle8i server, and Oracle8i precompiler clients can work with an Oracle7 server.

Specifically, the following list outlines precompiler/server compatibility. The list uses Pro*C as an example, but the information also applies to Pro*COBOL:

- Applications using Oracle7 Pro*C 2.2 work unchanged against an Oracle8i server (all SQLLIB function calls work against Oracle8i servers).
- Applications using Pro*C 8.1 can work against an Oracle7 server if they *do not use any new version 8 features, including the object capabilities*, of the precompiler or server.
- Oracle7 and Oracle8i SQLLIB function calls can be mixed in the same application and the same transaction.

The following three alternative upgrade paths are available for an existing Oracle7 application, without requiring recompilation or re-precompilation. Again, the list uses Pro*C as an example, but the information also applies to Pro*COBOL:

- Retain the Oracle7 client application without any modification. It will continue to work against a new Oracle8i server.
- Upgrade to Oracle8i client but do not modify the application. To upgrade an Oracle7 client to an Oracle8i client, you only need to relink the new version of SQLLIB and *do not* need to recompile the application. Relinked Oracle7 applications work unchanged against an Oracle8i server.
- Upgrade to Oracle8i and modify the application to leverage new Oracle8i precompiler capabilities for performance and scalability benefits, or to use any object capabilities of Oracle8i. Existing Oracle7 applications must be recoded using the new Oracle8i SQLLIB calls, and must be relinked with the new SQLLIB, to run against an Oracle8i server. This upgrade path is required for any application to use any object capability of the Oracle8i server. Further, if application developers need to use any of the object capabilities of the Oracle8i server, they need to upgrade their client to use Pro*C 8.0 or higher.

See Also: *Pro*C/C++ Precompiler Programmer's Guide* and *Pro*COBOL Precompiler Programmer's Guide* for more information about precompiler applications.

Upgrading SQL*Plus Scripts

To use SQL*Plus release 8.0 or higher, a release 8.0 or higher database, and PL/SQL release 8.0 and higher functionality, complete the following steps:

- Make the following changes to SQL*Plus 3.x scripts to convert them into SQL*Plus 8.0 or higher scripts:
 - If a script contains the line `SET COMPATIBILITY VERSION 7`, change it to `SET COMPATIBILITY VERSION 8`, or remove the line so that the default setting is used (`VERSION 8`).
 - Check any `login.sql` file(s) and change any `SET COMPATIBILITY VERSION 7` line found to `SET COMPATIBILITY VERSION 8`.
 - Refer to the *SQL*Plus User's Guide and Reference* to learn about new functionality in SQL*Plus.
- To use new Oracle8i functionality, change existing SQL scripts to conform to Oracle8i syntax. Existing SQL scripts run unchanged on Oracle8i, and require no modification, if they do not use new Oracle8i functionality

See Also: *Oracle8i SQL Reference* for more information about upgrading SQL scripts.

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

Upgrading Oracle7 Forms or Developer/2000 Applications

Forms applications run the same on Oracle7 and Oracle8i. However, review the new features described in *Getting to Know Oracle8i* to determine whether any of the new Oracle8i features would be beneficial to your applications or might otherwise affect them. Information about the ways in which the Oracle8i 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*.

Copying LONGs to LOBs

LOB datatypes (BFILE, BLOB, CLOB, and NCLOB) can provide many advantages over LONG datatypes. See *Oracle8i Concepts* for information about the differences between LOB and LONG datatypes.

In release 8.1, a new SQL operator, TO_LOB, copies data from a LONG column in a table to a LOB column. The datatype of the LONG and LOB must be the same for a successful copy. For example, LONG RAW data must be copied to BLOB data, and LONG data must be copied to CLOB data.

In the examples in the following procedure, the LONG column named LONG_COL in table LONG_TAB is copied to a LOB column named LOB_COL in table LOB_TAB. These tables include an ID column that contains identification numbers for each row in the table.

Complete the following steps to copy data from a LONG column to a LOB column:

1. Create a new table with the same definition as the table that contains the LONG column, but use a LOB datatype in place of the LONG datatype.

For example, suppose you have a table with the following definition:

```
CREATE TABLE long_tab (  
    id NUMBER,  
    long_col LONG);
```

Create a new table using the following SQL statement:

```
CREATE TABLE lob_tab (  
    id NUMBER,  
    clob_col CLOB);
```

Note: When you create the new table, make sure you preserve the table's schema, including integrity constraints, triggers, grants, and indexes. The TO_LOB operator only copies data; it does not preserve the table's schema.

2. Issue an INSERT command using the TO_LOB operator to insert the data from the table with the LONG datatype into the table with the LOB datatype.

For example, issue the following SQL statement:

```
INSERT INTO lob_tab
  SELECT id,
  TO_LOB(long_col)
  FROM long_tab;
```

3. When you are certain that the copy was successful, drop the table with the LONG column.

For example, issue the following SQL command to drop the LONG_TAB table:

```
DROP TABLE long_tab;
```

4. Create a synonym for the new table using the name of the table with LONG data. The synonym ensures that your database and applications continue to function properly.

For example, issue the following SQL statement:

```
CREATE SYNONYM long_tab FOR lob_tab;
```

Once the copy is complete, any applications that use the table must be modified to use the LOB data.

See Also: *Oracle8i Application Developer's Guide - Large Objects (LOBs)* for information about modifying applications to use LOB data.

Migrating from Server Manager to SQL*Plus

Server Manager line mode will be obsoleted in a future release of Oracle. If you run SQL scripts using Server Manager line mode, you should change these scripts so that they are compatible with SQL*Plus, and then run them using SQL*Plus. This chapter guides you through changing your Server Manager line mode scripts to work with SQL*Plus.

This chapter covers the following topics:

- [Startup Differences](#)
- [Commands](#)
- [Syntax Differences](#)

See Also: *SQL*Plus User's Guide and Reference* for detailed information about using SQL*Plus release 8.1.

Note: For brevity, Server Manager line mode is referred to as Server Manager in the rest of this chapter.

Startup Differences

The methods for starting Server Manager and SQL*Plus are different, and your SQL scripts must be modified to start SQL*Plus properly. The following sections explain the startup differences and provide options for starting SQL*Plus.

Starting Server Manager

To start Server Manager, enter the name of the Server Manager program at a system prompt; the name of this program is operating-system specific. After you start up Server Manager, connect using the CONNECT command, as in the following example:

```
CONNECT scott/tiger
```

Starting SQL*Plus

The following sections describe various ways to start SQL*Plus.

Starting SQL*Plus with the NOLOG Option

If you want SQL*Plus to behave in the same way as Server Manager, use the NOLOG option when you start SQL*Plus, as in the following example:

```
sqlplus /nolog
```

SQL*Plus starts and you can use the CONNECT command to connect as a user.

Starting SQL*Plus with Connect Information

Another option for starting SQL*Plus is to enter the connect information when you start the program. For example, to start SQL*Plus and connect as SCOTT/TIGER, enter the following:

```
sqlplus scott/tiger
```

SQL*Plus starts and connects as user SCOTT.

Starting SQL*Plus without Options or Connect Information

To start SQL*Plus without options or connect information, enter the following:

```
sqlplus
```

SQL*Plus prompts you for a user name and password. When you enter a valid user name and password, SQL*Plus starts and connects as the user you specified at the prompts. In your SQL scripts, however, you may not want to prompt the user to enter a user name and password.

Commands

Server Manager and SQL*Plus share certain commands that behave the same in both programs. Other commands, however, behave differently in SQL*Plus than they do in Server Manager. To successfully migrate from Server Manager to SQL*Plus, you need to understand these differences and similarities. The following sections include information about modifying your SQL scripts to use commands that are interpreted correctly by SQL*Plus.

New SQL*Plus Release 8.1 Commands

[Table 10-1](#) lists Server Manager commands that are now available in SQL*Plus. You can use these commands in SQL scripts that you run with SQL*Plus release 8.1.

Note: On pre-release 8.1 databases (such as Oracle7 and release 8.0), use Server Manager to run scripts containing these commands. Versions of SQL*Plus before SQL*Plus release 8.1 will not run scripts containing these new commands.

Table 10–1 New SQL*Plus Commands in Release 8.1

Command	Description
ARCHIVE LOG	Starts or stops automatic archiving of online redo log files, manually (explicitly) archives specified redo log files, or displays information about archives.
RECOVER	Performs media recovery on one or more tablespaces, one or more datafiles, or the entire database.
SET AUTORECOVERY	ON causes the RECOVER command to automatically apply the default filenames of archived redo log files needed during recovery. No interaction is needed when AUTORECOVERY is set to ON, provided the necessary files are in the expected locations with the expected names.
SET INSTANCE	Changes the default instance for your session to the specified instance path. Does not connect to a database. The default instance is used for commands when no instance is specified.
SET LOGSOURCE	Specifies the location from which archive logs are retrieved during recovery. The default value is set by the LOG_ARCHIVE_DEST initialization parameter. Issuing the SET LOGSOURCE command without a pathname restores the default location.
SHOW AUTORECOVERY	Shows whether autorecovery is enabled.
SHOW INSTANCE	Shows the connect string for the default instance. SHOW INSTANCE returns the value LOCAL if you have not used SET INSTANCE or if you have used the LOCAL option of the SET INSTANCE command.
SHOW LOGSOURCE	Shows the current setting for the archive log location. Displays DEFAULT if the default setting is in effect, as specified by the LOG_ARCHIVE_DEST initialization parameter.
SHOW PARAMETERS	Displays the current values for one or more initialization parameters. The SHOW PARAMETERS command, without any string following the command, displays all initialization parameters.
SHOW SGA	Displays information about the current instance's System Global Area.
SHUTDOWN	Shuts down a currently running Oracle instance, optionally closing and dismounting a database.
STARTUP	Starts an Oracle instance with several options, including mounting and opening a database.

Commands Common to Server Manager and SQL*Plus

The commands listed in [Table 10–2](#) are available in both Server Manager and SQL*Plus, and have been available in both programs in past releases of Oracle. You do not need to alter these commands in your SQL scripts to use SQL*Plus.

Note: There may be minor formatting differences in the output for these commands in the two programs.

Table 10–2 *Server Manager Commands Corresponding to Existing SQL*Plus Commands*

Command	Description
CONNECT	Connects to a database using the specified user name.
DESCRIBE	Describes a function, package, package body, procedure, table, or view. For example, for a table, displays the definitions of each column in the table.
REMARK	Enters a comment, typically in SQL script files.
SET COMPATIBILITY	Sets compatibility mode to V7, V8, or NATIVE. The compatibility mode setting affects the specification of character columns, integrity constraints, and rollback segment storage parameters. NATIVE matches the version of the database.
SET ECHO	Controls whether the START command lists each command in a command file as the command is executed. ON lists the commands; OFF suppresses the listing.
SET NUMWIDTH	Sets the default width for displaying numbers.
SET SERVEROUTPUT	Controls whether to display the output (that is, DBMS_OUTPUT.PUT_LINE) of stored procedures or PL/SQL blocks in SQL*Plus. OFF suppresses the output of DBMS_OUTPUT.PUT_LINE; ON displays the output.
SET TERMOUT	Controls the display of output generated by commands executed from a command file. OFF suppresses the display so that you can spool output from a command file without seeing the output on the screen. ON displays the output.
SHOW ALL	Lists all of the system variables set by the SET command in alphabetical order, except ERRORS, PARAMETERS, and SGA.
SHOW ERRORS	Shows the errors generated from the last compilation of a procedure, package, or function, if any.
SPOOL	Stores query results in an operating system file and, optionally in SQL*Plus, sends the file to a printer. Note: The extension of spool files may differ between SQL*Plus and Server Manager. To ensure an extension, specify it when you issue the SPOOL command.

SQL*Plus Equivalents for Server Manager Commands

Table 10-3 lists the SQL*Plus commands that correspond to Server Manager commands with different names. If you are using any of these Server Manager commands in SQL scripts, modify the scripts to use the SQL*Plus commands instead.

Table 10-3 SQL*Plus Equivalents for Server Manager Commands

Server Manager Commands	SQL*Plus Commands	Description
SET CHARWIDTH SET DATEWIDTH SET LONGWIDTH	COLUMN FORMAT	<p>You can use the COLUMN FORMAT command in SQL*Plus to set the column width of character columns, date columns, and number columns. In your SQL scripts, replace the SET CHARWIDTH, SET DATEWIDTH, and SET LONGWIDTH Server Manager commands with the SQL*Plus command COLUMN FORMAT.</p> <p>For example, suppose you have the following entry in a SQL script:</p> <pre>SET CHARWIDTH 5</pre> <p>This command sets the width for all character columns to 5 in Server Manager.</p> <p>To specify that a particular column, such as ENAME, display with a width of 5 characters, enter the following SQL*Plus command:</p> <pre>COLUMN ENAME FORMAT A5</pre>
SET STOPONERROR	WHENEVER SQLERROR WHENEVER OSERROR	<p>Use the WHENEVER SQLERROR and WHENEVER OSERROR commands to direct SQL*Plus to either exit or continue whenever a SQL error or operating system error occurs. Use these commands in your SQL scripts instead of the Server Manager command SET STOPONERROR.</p> <p>For both WHENEVER SQLERROR and WHENEVER OSERROR, the EXIT clause directs SQL*Plus to exit, while the CONTINUE clause directs SQL*Plus to continue. Other terms and clauses are also available for these commands.</p>

Possible Differences in the SET TIMING Command

The SET TIMING command is available in both Server Manager and SQL*Plus, but this command may function differently in the two programs on some operating systems. Check your operating-system specific Oracle documentation for more information. If the SET TIMING command functions differently in these two programs on your operating system, modify your SQL scripts so that this command functions properly with SQL*Plus.

Server Manager Commands Unavailable in SQL*Plus

The following Server Manager commands are unavailable in SQL*Plus release 8.1:

- SET MAXDATA
- SET RETRIES

Remove these commands from your SQL scripts.

Syntax Differences

The following sections explain the syntax differences between Server Manager and SQL*Plus. Modify your SQL scripts to conform with SQL*Plus syntax conventions before you attempt to run your scripts using SQL*Plus.

Comments

SQL*Plus recognizes the following types of comments:

- the SQL*Plus REMARK command (or REM)
- the SQL comment delimiters, /*...*/
- the ANSI/ISO comments, --

The *SQL*Plus User's Guide and Reference* provides detailed information about using these types of comments in SQL*Plus code.

Server Manager supports these types of comments, but the behavior is different for some of them. Also, certain types of comments are available in Server Manager, but not in SQL*Plus. The sections below discuss each type of comment and the syntax differences between Server Manager and SQL*Plus.

REMARK Command (or REM)

In general, the REMARK command works the same in Server Manager and SQL*Plus, and you do not need to change the occurrences of the REMARK command in your SQL scripts. There is, however, one difference: SQL*Plus interprets a hyphen that terminates a REMARK command differently than Server Manager. See "[Hyphens Used as Dividing Lines](#)" on page 10-11 for information about this difference.

SQL Comment Delimiters, /*...*/

In Server Manager, the SQL comment delimiters can be placed after a semi-colon, but in SQL*Plus, placing a SQL comment delimiter after a semi-colon is not allowed. Except for this one difference, SQL comment delimiters work the same in Server Manager and SQL*Plus.

If your SQL scripts contain any SQL comment delimiters placed after a semi-colon, either move the comment to its own line, or remove the semi-colon and place a slash (/) on the next line to end the SQL statement.

For example, suppose you have the following Server Manager code in one of your SQL scripts:

```
SELECT * FROM scott.emp
      WHERE job = 'CLERK'; /* Includes only clerks. */
```

In SQL*Plus, replace this code with either of the following entries:

```
SELECT * FROM scott.emp
      WHERE job = 'CLERK';
/* Includes only clerks. */
```

```
SELECT * FROM scott.emp
      WHERE job = 'CLERK' /* Includes only clerks. */
/
```

ANSI/ISO Comments, --

In Server Manager, the ANSI/ISO comments can be placed after a semi-colon, but in SQL*Plus, placing an ANSI/ISO comment after a semi-colon is not allowed. Except for this one difference, ANSI/ISO comments work the same in Server Manager and SQL*Plus.

If your SQL scripts contain any ANSI/ISO comments that are placed after a semi-colon, either move the comment to its own line, or remove the semi-colon and place a slash (/) on the next line to end the SQL statement.

For example, suppose you have the following Server Manager code in one of your SQL scripts:

```
SELECT * FROM scott.emp
      WHERE job = 'CLERK'; -- Includes only clerks.
```

In SQL*Plus, replace this code with either of the following entries:

```
SELECT * FROM scott.emp
      WHERE job = 'CLERK';
-- Includes only clerks.
```

```
SELECT * FROM scott.emp
      WHERE job = 'CLERK' -- Includes only clerks.
/
```

Server Manager Pound (#) Comments

Server Manager supports the use of the pound sign (#) to indicate a comment line. If your scripts contain these comments, change the '#' to '--' to run the scripts using SQL*Plus.

For example, suppose you have the following Server Manager code in one of your SQL scripts:

```
# This statement returns only clerks.
SELECT * FROM scott.emp
      WHERE job = 'CLERK';
```

In SQL*Plus, replace this code with the following entry:

```
-- This statement returns only clerks.
SELECT * FROM scott.emp
      WHERE job = 'CLERK';
```

Blank Lines

Server Manager allows blank lines within SQL statements, but SQL*Plus does not. SQL*Plus buffers SQL statements at the first blank line, which enables you to easily rerun a particular SQL statement.

If any of your SQL scripts contain blank lines within SQL statements, remove the blank lines before you run these scripts using SQL*Plus. For example, suppose you have the following SQL statement in one of your SQL scripts:

```
SELECT empno, ename, sal, comm

      FROM scott.emp

      WHERE job = 'MANAGER';
```

In SQL*Plus, replace this code with the following SQL statement by deleting the blank lines:

```
SELECT empno, ename, sal, comm
      FROM scott.emp
      WHERE job = 'MANAGER';
```

Note: You still can place blank lines *in between* SQL statements. This restriction only applies to blank lines *within* SQL statements.

The Hyphen Continuation Character

SQL*Plus supports the use of a hyphen as a continuation character for long SQL statements or SQL*Plus commands. For example, you can use the continuation character in the following way:

```
SELECT empno, ename, sal, comm FROM scott.emp -
WHERE job = 'MANAGER';
```

Server Manager does not support the use of a hyphen as a continuation character, but you may use hyphens for other purposes in your SQL scripts. If you do, SQL*Plus may interpret a hyphen as a continuation character, which can cause unexpected output.

The following sections provide scenarios in which SQL*Plus interprets the use of hyphens in SQL scripts as continuation characters, when the hyphens were meant for another purpose. Check your SQL scripts for the use of hyphens and modify them to avoid scenarios similar to those described below.

Hyphens Used as Dividing Lines

Your SQL scripts may use a long row of hyphens following a REMARK command as a dividing line in the code. Consider the following sample lines from a SQL script:

```
Rem -----
SELECT empno, ename, job
       FROM scott.emp;
```

In this statement, SQL*Plus interprets the first line of the SELECT statement as a continuation of the previous line, which is a REMARK comment. Therefore, the FROM line is interpreted as the first line of a SQL statement, and SQL*Plus returns the following error:

```
unknown command beginning "FROM scott..." - rest of line ignored.
```

If you use hyphens as dividing lines in your SQL scripts, remove the REM command preceding the hyphens before you run the scripts using SQL*Plus.

Hyphens Used as Minus Signs

Because the hyphen is the same keyboard character as the minus sign, you may have a hyphen at the end of a line. Consider the following sample lines from a SQL script:

```
CREATE TABLE xx (
  a int,
  b int,
  c int);

INSERT INTO xx VALUES (10, 20, 30);

SELECT a + b -
       c FROM xx;
```

SQL*Plus interprets the 'c' as an alias because the minus symbol is interpreted as a continuation character:

```
SELECT a + b c FROM xx;
```

Therefore, SQL*Plus returns the following unexpected output:

```
      c
-----
      30
```

Server Manager, however, interprets this code as the following:

```
SELECT a + b - c FROM xx;
```

Therefore, Server Manager returns the following expected output:

```
A+B-C  
-----  
0
```

Make sure you do not have a minus sign at the end of a line in your SQL scripts.

Ampersands

SQL*Plus interprets an ampersand (&) as a substitution variable, whereas Server Manager interprets an ampersand as a normal string. If the text following the ampersand does not have a defined value, SQL*Plus interprets it as an undefined value and prompts the user for input, even if the ampersand is enclosed in a comment. Therefore, ampersands can cause unexpected output in SQL*Plus.

If you have SQL scripts that use ampersands as normal text strings, you have two options:

- Use the SET ESCAPE command to place an escape character before each ampersand.
- Use the SET DEFINE OFF command to disable the recognition of substitution variables.

Note: Do not use the SET DEFINE OFF command if you have other, valid substitution variables; if you do, the other variables will not be recognized.

For example, the following SQL statement prompts the user for input in SQL*Plus:

```
CREATE TABLE "Employees & Managers" (  
    Employees varchar(16),  
    Managers varchar(16));
```

Enter value for managers:

Using the SET ESCAPE Command

To avoid the user prompt, you can use the SET ESCAPE command to set an escape character. Then, place the escape character before the ampersand. A backslash (\) is often used as an escape character.

To avoid the prompt in the example preceding example by using the SET ESCAPE command, change the entry to the following:

```
SET ESCAPE \

CREATE TABLE "Employees \& Managers" (
    Employees varchar(16),
    Managers varchar(16));
```

Using the SET DEFINE OFF Command

To avoid the prompt in the preceding example by using the SET DEFINE OFF command, change the entry to the following:

```
SET DEFINE OFF

CREATE TABLE "Employees & Managers" (
    Employees varchar(16),
    Managers varchar(16));
```

CREATE TYPE and CREATE LIBRARY Commands

SQL*Plus treats the CREATE TYPE and CREATE LIBRARY commands as PL/SQL blocks. Therefore, in SQL*Plus, you must use a slash (/) on a separate line to end these commands, while Server Manager allows you to end these commands with a semi-colon.

If you end any CREATE TYPE or CREATE LIBRARY command with a semi-colon in your SQL scripts, remove the semi-colon and place a slash on the next line. For example, the following SQL statements are not recognized by SQL*Plus:

```
CREATE OR REPLACE TYPE sys.dummy AS OBJECT (data CHAR(1));
CREATE OR REPLACE LIBRARY DBMS_SPACE_ADMIN_LIB TRUSTED AS STATIC;
```

Edit these statements in the following way before you run them with SQL*Plus:

```
CREATE OR REPLACE TYPE sys.aq$_dummy_t AS OBJECT (data CHAR(1))
/
CREATE OR REPLACE LIBRARY DBMS_SPACE_ADMIN_LIB TRUSTED AS STATIC
/
```

COMMIT Command

SQL*Plus requires that the COMMIT command be terminated either with a semi-colon (;) or a slash (/), but Server Manager allows the COMMIT command with no terminator. Therefore, if you use the COMMIT command in your SQL scripts without a terminator, edit these scripts to include a terminator.

For example, suppose you have the following COMMIT command in a SQL script:

```
commit
```

Include a terminator for the command, as shown in either of the following examples:

```
commit;
```

```
commit  
/
```

Migration Issues for Physical Rowids

Version 8 physical rowids embody new internal and external formats that enable you to use some new version 8 features, including partitioning and global indexes.

See Also: The *Oracle8i Application Developer's Guide - Fundamentals* and *Oracle8i Concepts* for more information.

This chapter covers the following migration issues related to the new version 8 physical rowids:

- [Migrating Applications and Data](#)
- [The DBMS_ROWID Package](#)
- [Conversion Procedure Examples](#)
- [Snapshot Refresh](#)
- [Pre-Version 8 Client Compatibility Issues](#)
- [Rowid-Related Migration Questions and Answers](#)

Note: In the rest of this chapter, the word "rowid" means "physical rowid". This chapter does not discuss the UROWID (universal rowid) datatype. See [Chapter 8, "Compatibility and Interoperability"](#) for compatibility issues relating to the UROWID datatype.

Migrating Applications and Data

Rowids can be stored in columns of ROWID datatype and in columns of character type. Stored version 7 rowids become invalid after migration of the version 7 database to version 8. Therefore, stored version 7 rowids must be converted to version 8 format.

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 must use the new package, `DBMS_ROWID`, provided in version 8 to deal with the format and contents of the new version 8 rowids. This 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 datatype 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 version 7 or version 6 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 `COLS` for rowid columns from six to ten 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 or Oracle Data Migration Assistant applies only to rowids stored in a user-defined column. All system-stored rowids (such as in local indexes) remain valid after migration by the Migration utility or Oracle

Data Migration Assistant, and do not require specific actions to be migrated. Also, indexes are not invalidated because, during migration to version 8 by Migration utility or Oracle Data Migration Assistant, indexes on non-partitioned tables can continue to use the restricted ROWID datatype format.

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).

The 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 datatype, or in the user column of CHAR or VARCHAR datatype.

For a column of ROWID datatype, 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 CHAR or VARCHAR datatype, 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 the **ROWID_TO_EXTENDED** and **ROWID_VERIFY** procedures.

The **ROWID_TO_EXTENDED** Conversion Procedure

ROWID_TO_EXTENDED uses the following parameters:

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

See Also: ["Rowid Conversion Types"](#) on page 11-3 for more information.

ROWID_TO_EXTENDED returns a version 8 (extended) rowid in External Character format, and its parameters are interpreted in the following way:

- 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. You 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 a 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.

Conversion Procedure Examples

The following are examples of conversion procedures for rowids:

Example 1

Assume a table SCOTT.T contains a column C of ROWID datatype 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 datatype (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 datatype format forces all rowid snapshots to perform a complete refresh when both master and snapshot sites are upgraded to version 8.

See Also: *Oracle8i Replication*, Appendix B, "Migration and Compatibility", for more information.

Pre-Version 8 Client Compatibility Issues

Pre-release 8.0 clients can access a version 8 database, and version 8 clients can access a pre-release 8.0 database. Binary and character values of the pseudo column ROWID and of columns of datatype ROWID that are returned by a pre-release 8.0 database to a version 8 database are always in pre-release 8.0 (restricted) format, because the pre-release 8.0 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 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 a ROWID user column if a row of this table contains the extended rowid value.

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

A: Forms applications which intend to access version 8 databases have to be relinked using the patch #380655.

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

A: Programs that use rowids but do not rely on their format are not affected. Programs that rely on the version 7 ROWID datatype 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-release 8.0 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 datatype format. Will I need to malloc() more space?

A: Version 8 rowids fit 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 datatype?

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

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

A: The new version 8 ROWID datatype 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 datatype format needs to be rebuilt after data migration.

Q: I use ROWID datatype in pre-release 8.0 PL/SQL, RPC, or from FORMS. Will this continue to work?

A: The format in which rowids are returned into host variables of ROWID datatype 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 an ORA-3116 error upon a rowid fetch as a type DTYRID (without CHR conversion) through OCI. The following are ways to avoid this problem:

- 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.

Downgrading to an Older Version 8 Release

The information in this chapter only applies to release 8.1 installations of Oracle. The term *downgrading* describes transforming an Oracle database into a previous release of the same version, such as transforming a database from release 8.1.5 to release 8.0.5. The term *downgrading* also describes transforming an Oracle database into a previous version, such as transforming a database from Oracle8i to Oracle7. This chapter describes downgrading to an older 8.1 release of Oracle or to an 8.0 release of Oracle. If you want to downgrade to Oracle7, see [Chapter 13, "Downgrading to Oracle7"](#).

Perform the procedures in the following sections, in the order shown, to downgrade your database:

- [Perform a Full Offline Backup](#)
- [Remove Incompatibilities](#)
- [Reset Database Compatibility](#)
- [Downgrade the Database](#)

See Also: Some aspects of downgrading are operating-system specific. See your operating-system specific Oracle documentation for additional operating-system specific instructions about downgrading.

Perform a Full Offline Backup

Perform a full offline backup of your release 8.1 database before you downgrade.

See Also: *Oracle8i Backup and Recovery Guide* for more information.

Remove Incompatibilities

The process for removing incompatibilities depends on whether you are downgrading to a previous 8.1 release or to an 8.0 release. Follow the instructions in the appropriate section based on the release to which you are downgrading.

Removing Incompatibilities If You Are Downgrading to an 8.1 Release

If you are downgrading to either release 8.1.4 or 8.1.3, complete the following actions to remove incompatibilities:

- If you used single-table hash clusters, follow the instructions in "[Drop All Single-Table Hash Clusters](#)" on page 12-9.
- If you used temporary tables, follow the instructions in "[Drop Temporary Tables](#)" on page 12-6.
- If you created Java objects, follow the instructions in "[Java](#)" on page 12-26. Although Java is supported in release 8.1.4, the format changed in release 8.1.5. Therefore, the Java objects used in release 8.1.5 cannot be used in previous 8.1 releases. If you are downgrading to release 8.1.4 and want to preserve your Java objects, you can export them before you drop them and then import them after you downgrade to release 8.1.4.

After completing these actions, proceed to the "[Reset Database Compatibility](#)" section on page 12-42 if you used any of the following features:

- single-table hash clusters
- temporary tables
- Java
- online indexes

However, if you did not use these features, you do not need to reset database compatibility, and you can proceed to the "[Downgrade the Database](#)" section on page 12-43.

Removing Incompatibilities If You Are Downgrading to an 8.0 Release

If the compatibility level of your database is higher than the release to which you are downgrading, your database may have incompatibilities with the previous release that must be removed before you downgrade. Check your COMPATIBLE parameter setting by issuing the following SQL statement:

```
SELECT name, value, description FROM v$parameter
       WHERE name='compatible';
```

You do not need to remove incompatibilities if the COMPATIBLE parameter is set to the release to which you are downgrading or lower. For example, if you are downgrading to release 8.0.5 and the COMPATIBLE parameter is set to 8.0.5 or lower, you do not need to remove incompatibilities. In this case, no incompatibilities exist in your database with the release to which you are downgrading, and you can skip the rest of this section and move on to the ["Downgrade the Database"](#) section on page 12-43.

However, if you are downgrading to an 8.0 release and the COMPATIBLE parameter is set higher than the release to which you are downgrading, some incompatibilities may exist. For example, if you are downgrading to release 8.0.5, and COMPATIBLE is set to 8.1.0 or higher, incompatibilities may exist. Similarly, if you are downgrading to release 8.0.3, and COMPATIBLE is set to 8.0.4 or higher, incompatibilities may exist.

If incompatibilities may exist, use the following general procedure to remove incompatibilities with the release to which you are downgrading:

1. At a system prompt, change to the \$ORACLE_HOME/rdbms/admin directory.
2. Start Server Manager.
3. Connect to the database instance:

```
SVRMGR> CONNECT INTERNAL
```

4. Identify incompatibilities by completing the following steps:
 - a. Query the V\$COMPATIBILITY dynamic performance view to identify the incompatibilities:

```
SVRMGR> SELECT * FROM v$compatibility WHERE release != '0.0.0.0.0';
```

An incompatibility exists wherever the value in the RELEASE column is higher than the release to which you are downgrading.

Note: This query does not show features with a compatibility level of 0.0.0.0. These features currently are not in use, and no action is required for them.

b. Run `utlimcmt.sql`:

```
SVRMGR> SPOOL utlimcmt.out
SVRMGR> @utlimcmt.sql
SVRMGR> SPOOL OFF
```

The `utlimcmt.sql` script runs all of the queries described in the rest of this chapter to identify incompatibilities. Therefore, you can perform all of the `SELECT` statements described in the rest of this chapter simply by running the `utlimcmt.sql` script.

After the `utlimcmt.sql` script runs, view the `utlimcmt.out` file and look for instances where a `SELECT` statement returned values. The values returned are incompatibilities with release 8.0.

5. Drop or change all incompatibilities to make your database compatible with the release to which you are downgrading.

The following sections provide detailed information about removing incompatibilities with release 8.0. To remove incompatibilities, you may need to complete actions that require the privileges of `SYS` user. Therefore, you should log in as `SYS` user and connect as `SYSDBA` to perform the actions described in the following sections, unless instructed otherwise.

Also, if you created your database at 8.1.0 compatibility level or higher, Oracle created certain system-defined types that are incompatible with 8.0 releases. To remove these incompatibilities, run the `utldst.sql` script supplied with release 8.1:

```
@utldst.sql
```

Note: If you are downgrading from Oracle8i Enterprise Edition to Oracle8i (formerly Workgroup Server), before you downgrade, modify any applications that use the advanced features of Oracle8i Enterprise Edition so that they do not use these advanced features. See *Getting to Know Oracle8i* for more information about the differences between the editions.

Tablespaces

This section describes removing incompatibilities relating to tablespaces.

Remove Transported Tablespaces

If you used the transportable tablespace feature to either move a tablespace into the database you are downgrading, or to transport a tablespace from this database to another database, perform the following steps before downgrading:

1. Identify the transported tablespaces that were plugged into the database by issuing the following SQL statement:

```
SELECT tablespace_name, plugged_in
       FROM dba_tablespaces
       WHERE plugged_in = 'YES';
```

2. Either drop or move each transported tablespace listed by the SQL statement.

If you do not need to preserve the data in a transported tablespace, drop the tablespace. If you need to preserve the data, either export the data from your current database and import the data after you downgrade, or transport the tablespace to another database before you downgrade.

3. Execute DBMS_TTS.DOWNGRADE:

```
EXECUTE dbms_tts.downgrade;
```

The DBMS_TTS.DOWNGRADE procedure drops the temporary tables in the system tablespace used by the transportable tablespace feature.

Discontinue Use of Locally Managed Tablespaces

Release 8.1 supports locally managed tablespaces. Before you downgrade, you must convert all locally managed tablespaces to dictionary tablespaces.

To identify locally managed tablespaces, enter the following SQL statement:

```
SELECT tablespace_name, extent_management
       FROM dba_tablespaces
       WHERE extent_management = 'LOCAL';
```

Run the DBMS_ADMIN.TABLESPACE_MIGRATE_FROM_LOCAL procedure on all tablespaces listed. For example, if a tablespace named TS_1 is listed, enter the following SQL statement to convert TS_1 to a dictionary tablespace:

```
EXECUTE dbms_admin.tablespace_migrate_from_local('ts_1');
```

Schema Objects

This section describes removing incompatibilities relating to schema objects.

Drop Temporary Tables

Before you downgrade, drop all temporary tables. To identify existing temporary tables, issue the following SQL statement:

```
SELECT owner, table_name FROM dba_tables
       WHERE temporary = 'Y' AND
              table_name NOT LIKE 'RUPD$%' AND
              table_name NOT LIKE 'ATEMPTAB$';
```

Drop all tables listed.

Discontinue Use of Key Compression on Indexes and Index-Organized Tables

Before you downgrade, discontinue use of all indexes and index-organized tables with key compression in your database. To identify existing indexes and index-organized tables with key compression, issue the following SQL statement:

```
SELECT index_name, index_type, table_owner, table_name
       FROM dba_indexes WHERE compression = 'ENABLED';
```

For each index listed, issue an ALTER INDEX ... REBUILD NOCOMPRESS statement. For example, if you have an index with key compression named I_JOB, enter the following SQL statement:

```
ALTER INDEX i_job REBUILD NOCOMPRESS;
```

For all of the index-organized tables listed, issue an ALTER TABLE ... MOVE NOCOMPRESS statement. For example, if you have an index-organized table with key compression named IOT_ITEM, issue the following SQL statement:

```
ALTER TABLE iot_item MOVE NOCOMPRESS;
```

Note: The ALTER TABLE ... MOVE NOCOMPRESS statement is not allowed on nested table storage tables that are stored as index-organized tables. For each such nested table column, either drop the column, or recreate the parent table with a CREATE TABLE ... AS SELECT statement, specifying a heap storage table for the nested table column. The parent table is the table containing the nested table column.

Discontinue Use of LOBs and Varrays in Index-Organized Tables

Before you downgrade, drop all index-organized tables with LOBs or varrays in your database. To identify existing index-organized tables with LOBs, issue the following SQL statement:

```
SELECT column_name, t.owner, t.table_name
FROM dba_lobs l, dba_tables t
WHERE l.table_name = t.table_name
AND l.owner = t.owner
AND t.iot_type = 'IOT';
```

To identify existing index-organized tables with varrays, issue the following SQL statement:

```
SELECT v.parent_table_column, t.owner, t.table_name
FROM dba_varrays v, dba_tables t
WHERE v.parent_table_name = t.table_name
AND v.owner = t.owner
AND t.iot_type = 'IOT';
```

If you do not need to preserve the data in the tables listed by these SQL statements, drop the tables. However, if you need to preserve the data in any of these tables, complete the following steps for each table:

1. Create a new table that is not index-organized by selecting all rows from the index-organized table with LOBs or varrays.

For example, assume you have an index-organized table with LOBs named LOBIOT with the following definition:

```
CREATE TABLE lobiot (a INT, b CLOB, c INT PRIMARY KEY) ORGANIZATION INDEX;
```

Issue the following SQL statement to create a table that is not index-organized named NIOTD2 using the data in LOBIOT:

```
CREATE TABLE niotd2 (a,b,c PRIMARY KEY) AS SELECT * FROM lobiot;
```

2. When you are sure the new table is functioning properly, drop the original index-organized table with LOBs or varrays.
3. Rename the new table to its original name.

Drop All Secondary Indexes on Index-Organized Tables

Before you downgrade, drop all secondary indexes on index-organized tables in your database. To identify existing secondary indexes on index-organized tables, issue the following SQL statement:

```
SELECT index_name, i.owner, t.table_name
   FROM dba_indexes i, dba_tables t
  WHERE i.index_type = 'NORMAL'
        AND i.table_name = t.table_name
        AND t.owner = i.table_owner
        AND t.iot_type = 'IOT';
```

Drop the indexes listed.

Drop Unused and Partially Dropped Columns

Before you downgrade, drop all unused and partially dropped columns.

Dropping Unused Columns You will not be able to downgrade if any tables in your database have unused columns. To identify tables that have unused columns, issue the following SQL statement:

```
SELECT * FROM dba_unused_col_tabs;
```

To drop all of the unused columns in a table, use the ALTER TABLE ... DROP UNUSED COLUMNS command. Run this command for each table in the list. For example, to drop all of the unused columns in a table named CUSTOMERS, enter the following command:

```
ALTER TABLE customers DROP UNUSED COLUMNS;
```

Dropping Partially Dropped Columns You will not be able to downgrade if any tables in your database have partially dropped columns. To identify tables that have partially dropped columns, issue the following SQL statement:

```
SELECT * FROM dba_partial_drop_tabs;
```

To drop all of the partially dropped columns in a table, use the ALTER TABLE ... DROP COLUMNS CONTINUE command. Run this command for each table in the list. For example, to drop all partially dropped columns in a table named CUSTOMERS, enter the following command:

```
ALTER TABLE customers DROP COLUMNS CONTINUE;
```

Drop All Single-Table Hash Clusters

You must drop all single-table hash clusters before you downgrade. To check for single table-only hash clusters, issue the following SQL statement:

```
SELECT cluster_name, single_table FROM dba_clusters
       WHERE single_table='Y';
```

Drop all of the clusters listed.

Drop Incompatible Materialized Views

Identify materialized views that are incompatible with release 8.0 by issuing the following SQL statement:

```
SELECT mv.owner, mv.name
       FROM dba_snapshots mv, dba_mview_analysis mva
       WHERE mva.owner = mv.owner
       AND mva.mview_name = mv.name;
```

Drop all of the materialized views listed. For example, if a materialized view owned by SCOTT and named MV_1 is listed, issue the following SQL statement to drop the materialized view:

```
DROP MATERIALIZED VIEW scott.mv_1;
```

Identify Materialized Views That Will Be Changed or Dropped During Downgrade

The following sections provide instructions for identifying materialized views that will be changed or dropped during the downgrade process described in ["Downgrade the Database"](#) on page 12-43.

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

REFRESH ON COMMIT Mode Changed to REFRESH ON DEMAND Mode Release 8.1 enables you to use the REFRESH ON COMMIT mode for materialized views, but this mode is not available in release 8.0. To identify the materialized views in REFRESH ON COMMIT mode, issue the following SQL statement:

```
SELECT owner, name, refresh_mode
       FROM dba_snapshots
       WHERE refresh_mode = 'COMMIT';
```

All of the materialized views listed are in REFRESH ON COMMIT mode. When you downgrade, these materialized views will be changed to REFRESH ON DEMAND mode automatically.

FAST REFRESH Mode Unavailable After Downgrade Materialized views that use joins or the GROUP BY clause (aggregate queries) can no longer use the FAST REFRESH mode after you downgrade.

NEVER REFRESH Mode Materialized Views Dropped Release 8.1 enables you to use the NEVER REFRESH mode for materialized views, but this mode is not available in release 8.0. To identify the materialized views in NEVER REFRESH mode, issue the following SQL statement:

```
SELECT owner, name, type
       FROM dba_snapshots
       WHERE type = 'NEVER';
```

All of the materialized views listed are in NEVER REFRESH mode. When you downgrade, these materialized views will be dropped automatically.

Materialized Views Created with the PREBUILT TABLE Clause Dropped Release 8.1 enables you to use the PREBUILT TABLE clause to create materialized views, but these views are not supported in release 8.0. Any views created with the PREBUILT TABLE clause will be dropped automatically when you downgrade.

Materialized Views Created without a View Any materialized views created without a view will be dropped automatically when you downgrade.

Materialized Views Created with the BUILD DEFERRED Clause Refreshed When you downgrade, complete refresh will be performed automatically on any views created with the BUILD DEFERRED clause.

Mutually Referencing Views and Downgrading to Release 8.0.4 or Lower

If you have mutually referencing views, and you are downgrading to release 8.0.4 or lower, you must drop these views. If you do not have mutually referencing views, or if you are downgrading to release 8.0.5 or higher, skip this section.

Note: You can either drop the mutually referencing views before you downgrade or after you downgrade. They will not affect the downgrade operation

Mutually referencing views are not supported in release 8.0.3. If you are downgrading to release 8.0.3, drop all mutually referencing views.

Mutually referencing views are supported in release 8.0.4 and higher. However, you still must drop these views if you are downgrading to release 8.0.4. After you downgrade, you can recreate the previously dropped mutually referencing views in your 8.0.4 release. This action is required because of bug #662863, which is present in release 8.0.4, but is corrected in release 8.0.5 and higher.

Mutually referencing views are views in which the object views refer to each other through the MAKE_REF operator. In the following example of mutually referencing views, HUSBAND and WIFE types have references to each other, and object views were created with MAKE_REF operators:

```
CREATE TYPE husband
/

CREATE TYPE wife AS object
  (id2 NUMBER,
   name2 CHAR(10),
   salary number,
   buddy2 REF husband)
/

CREATE OR replace TYPE husband AS object
  (id NUMBER,
   name CHAR(10),
   buddy REF wife)
/

CREATE TABLE husbandtab
  (id NUMBER,
   name CHAR(10),
   buddy NUMBER);

CREATE TABLE wifetab
  (id2 NUMBER,
   name2 CHAR(10),
   salary NUMBER,
   buddy2 NUMBER);
```

```
CREATE VIEW husbandview OF husband
  WITH object OID(id) AS
  SELECT id, name, NULL FROM husbandtab;

CREATE VIEW wifeview OF wife WITH object OID(id2) AS
  SELECT id2, name2, salary,
  MAKE_REF(husbandview, buddy2)
  FROM wifetab;

CREATE OR replace VIEW husbandview
  OF husband WITH object OID(id) AS
  SELECT id, name, MAKE_REF(wifeview, buddy)
  FROM husbandtab;
```

Check for Bitmap Indexes That Will Be Invalidated

Release 8.1 provides protections for bitmap indexes. These protections prevent bitmap indexes from being unintentionally invalidated.

See Also: ["Bitmap Index Protection"](#) on page 8-25 for information.

When you downgrade to release 8.0, any bitmap indexes that were protected by this new feature will be invalidated automatically during the downgrade process described in ["Downgrade the Database"](#) on page 12-43. To list the indexes that will be invalidated during the downgrade process, issue the following SQL statement:

```
SELECT o.name INDEX_NAME, u.name INDEX_OWNER
  FROM sys.user$ u, sys.obj$ o, sys.ind$ i, sys.tab$ t
  WHERE t.obj# = o.obj#
  AND i.bo# = t.obj# AND t.spare1 > 32767
  AND i.type# = 2 AND o.owner# = u.user#;
```

Drop Function-Based Indexes

You will not be able to downgrade if your database has any function-based indexes. To identify function-based indexes, issue the following SQL statement:

```
SELECT DISTINCT index_owner, index_name FROM dba_ind_columns
      WHERE column_name IS NULL;
```

To drop all of the function-based indexes, use a DROP INDEX statement. For example, to drop a function-based index named FUNCIN1, issue the following SQL statement:

```
DROP INDEX funcin1;
```

Issue this statement for each function-based index listed.

Discontinue Use of Extensible Indexing

Release 8.1 supports extensible indexing. This feature enables the creation of domain indexes, indextypes, and operators. Before you downgrade, you must drop these objects.

Identifying and Dropping Domain Indexes To identify domain indexes, issue the following SQL statement:

```
SELECT owner, index_name, index_type
      FROM dba_indexes
      WHERE index_type = 'DOMAIN';
```

Drop the domain indexes listed.

Identifying and Dropping Indextypes To identify indextypes, issue the following SQL statement:

```
SELECT owner, indextype_name FROM dba_indextypes;
```

To drop the indextypes listed, use a DROP INDEXTYPE statement. For example, if an indextype named IX_TYPE owned by USER2 is listed, issue the following SQL statement to drop the indextype:

```
DROP INDEXTYPE user2.ix_type;
```

Identifying and Dropping Operators To identify operators, issue the following SQL statement:

```
SELECT owner, operator_name FROM dba_operators;
```

To drop the operators listed, use a DROP OPERATOR statement. For example, if an operator named OP1 owned by USER3 is listed, issue the following SQL statement to drop the operator:

```
DROP OPERATOR user3.op1;
```

Drop All Dimensions

Before you downgrade, you must drop all dimensions. Dimensions are not supported in release 8.0.

To identify the dimensions that must be dropped, issue the following SQL statement:

```
SELECT * FROM dba_dimensions;
```

Drop the dimensions listed by this SQL statement.

Partitioning

This section describes disabling release 8.1 partitioning features.

Discontinue Use of Partitioned Index-Organized Tables

Before you downgrade, drop all partitioned index-organized tables in your database. To identify existing partitioned index-organized tables, issue the following SQL statement:

```
SELECT table_name, tablespace_name, iot_type, partitioned
       FROM dba_tables WHERE partitioned = 'YES' AND iot_type = 'IOT';
```

If you do not need to preserve the data in the tables listed, drop the tables. However, if you need to preserve the data in a table, complete the following steps for the table:

1. Drop all partitioned secondary indexes on the table.
2. Create a new table that is either not index-organized or not partitioned by selecting all rows from the partitioned index-organized table.

For example, assume you have a partitioned index-organized table named PIOT with the following definition:

```
CREATE TABLE plot (a int, b int, c int, d int, e int,  
    PRIMARY KEY (d,e)) ORGANIZATION INDEX  
    PARTITION BY RANGE (d)  
    (  
        PARTITION itp1 VALUES LESS THAN (15),  
        PARTITION itp2 VALUES LESS THAN (30),  
        PARTITION itp3 VALUES LESS THAN (MAXVALUE)  
    );
```

Create a non-partitioned index-organized table named IOT using the data in PIOT by issuing the following SQL statement:

```
CREATE TABLE iot (a, b, c, d, e,  
    PRIMARY KEY (d,e)) ORGANIZATION INDEX  
    AS SELECT * FROM plot;
```

Or, if you want to keep the partitions but not the index organization, create a partitioned table that is not index-organized named PAR using the data in PIOT by issuing the following SQL statement:

```
CREATE TABLE par (a, b, c, d, e,  
    PRIMARY KEY (d,e)) PARTITION BY RANGE (d)  
    (  
        PARTITION itp1 VALUES LESS THAN (15),  
        PARTITION itp2 VALUES LESS THAN (30),  
        PARTITION itp3 VALUES LESS THAN (MAXVALUE)  
    ) AS SELECT * FROM plot;
```

3. When you are sure the new table is functioning properly, drop the partitioned index-organized table.
4. Rename the new table to its original name.

Discontinue Use of Partitioned Object Tables

Release 8.1 supports the partitioning of object tables and tables with the following user-defined types:

- nested table
- object
- REF
- VARRAY

Before you downgrade, drop all partitioned object tables. To identify all partitioned object tables, issue the following SQL statement:

```
SELECT UNIQUE t.table_name, t.owner
   FROM dba_part_tables t, dba_tab_columns c
  WHERE t.table_name = c.table_name
     AND c.data_type IN
     (SELECT type_name
      FROM dba_types
     WHERE predefined = 'NO');
```

If you do not need to preserve the data in the tables listed, drop the tables. However, if you need to preserve the data in one or more of the tables listed, use the CREATE TABLE ... AS SELECT statement to copy the data in a table to a non-partitioned table.

For example, if a table named OBP1 is listed by the SQL statement, and you want to save the data in this table, complete the following steps:

1. Create a table named TEMP_OBP1 by issuing the following SQL statement:

```
CREATE TABLE temp_obp1 AS SELECT * FROM obp1;
```

2. When you are sure that table TEMP_OBP1 has the data and is functioning properly, drop OBP1:

```
DROP TABLE obp1;
```

3. Rename TEMP_OBP1 to OBP1:

```
ALTER TABLE temp_obp1 RENAME TO obp1;
```

Discontinue Use of Partitioned Tables That Use Composite Methods

Release 8.1 supports the creation of partitioned tables using composite methods and non-composite methods other than RANGE. If you have any such tables in your database, you must perform one of the following actions:

- If you do not need to preserve the data in these tables, drop the tables.
- If you need to preserve the data, either copy the data into non-partitioned tables, or copy the data into tables partitioned by RANGE. Another option is to exchange partitions (of tables partitioned using the hash method) or subpartitions (of tables partitioned using the composite method) into non-partitioned tables using EXCHANGE.

To list the tables partitioned with composite methods and non-composite methods other than RANGE, issue the following SQL statement:

```
SELECT owner, table_name FROM dba_part_tables
       WHERE partitioning_type != 'RANGE' or SUBPARTITIONING_TYPE != 'NONE';
```

Datatypes

This section describes disabling datatypes that are available only in release 8.1 and higher.

Drop All Uses of the Universal ROWID Datatype

Complete the procedures in the following sections to remove all uses of the UROWID (universal rowid) datatype.

Drop All UROWID Columns To list all of the tables with UROWID datatype columns, issue the following SQL statement:

```
SELECT owner, table_name, column_name FROM dba_tab_columns
       WHERE data_type = 'UROWID' ORDER BY owner, table_name;
```

For each table listed as a result of this command, drop its UROWID datatype columns, or drop the whole table.

Drop All Stored Procedures with UROWID Arguments To list all stored procedures with arguments of UROWID datatype, issue the following SQL statement:

```
SELECT owner, object_name, package_name, argument_name
       FROM all_arguments
       WHERE data_type = 'UROWID' AND package_name != 'STANDARD'
       ORDER BY owner, object_name, package_name;
```

Drop each of the procedures listed, or change the argument to ROWID datatype.

Drop Existing Chained Row and Exception Tables In release 8.1, the UROWID datatype enables you to use a single table for chained rows and a single table for exceptions, but this functionality is not supported in release 8.0 databases. Therefore, you must prepare multiple tables for both chained rows and exceptions because you need at least one table for all regular tables and at least one table for each index-organized table.

Complete the following steps to downgrade a chained rows table called CH_ROWS:

1. Drop the existing CH_ROWS table.
2. Recreate the CH_ROWS table using the `utlchain.sql` script to store chained rows for the regular tables.
3. Create an individual chained table for each index-organized table using the `DBMS_IOT.BUILD_CHAIN_ROWS_TABLE` procedure.

Complete the following steps to downgrade an exception table called EXC_TB:

1. Drop the existing EXC_TB table.
2. Recreate the EXC_TB table using the `utlexcp.sql` script to store exceptions for the regular tables.
3. Create an individual exception table for each index-organized table using the `DBMS_IOT.BUILD_EXCEPTIONS_TABLE` procedure.

Discontinue Use of Release 8.1 LOB Features

Release 8.1 supports several new LOB features. Before you downgrade, discontinue the use of these new features by performing the actions described in the following sections.

Remove CLOBs and NCLOBs from Tables in a Database with a Varying-Width Character Set If your database is using a varying-width character set, remove all CLOB and NCLOB columns by completing the following steps. You do not need to complete this procedure if your database has a fixed-width character set.

1. List all of the tables that contain LOB columns by issuing the following SQL statement:

```
SELECT owner, table_name, column_name FROM dba_lobs
       WHERE dba_lobs.owner != 'SYSTEM'
       AND table_name NOT IN ('KOTAD$', 'KOTMD$', 'KOTTB$', 'KOTTD$');
```

2. Determine whether your database contains CLOB or NCLOB columns by running DESC on each of the tables listed; check the TYPE column when you run DESC.
3. Either drop the CLOB and NCLOB columns, or drop each table that contains the columns.

CAUTION: Check the tables created by SYS carefully, and make sure you do not drop tables that are required for version 8; that is, do not drop the tables that include a \$ symbol in the name. Examples of tables that you should not drop include: kotad\$, kotmd\$, kottb\$, and kottd\$.

4. If you use Advanced Replication, complete the following steps. If you do not use Advanced Replication, these steps are not required.
 - a. Push the replication deferred transaction queue using SQL*Plus:

```
DECLARE
    rc number;
BEGIN
    rc := dbms_defer_sys.push();
END;
```

- b. Drop the deferred LOB view by issuing the following SQL statement:

```
DROP VIEW deflob;
```

See Also: *Oracle8i Replication* for more information about completing these Advanced Replication steps.

Remove LOB Columns from Partitioned Tables Before you downgrade, remove all LOB columns from partitioned tables. To determine if your database contains LOB columns in partitioned tables, issue the following SQL statement:

```
SELECT table_name, lob_name FROM dba_part_lobs;
```

If you do not need to preserve your LOB data in partitioned tables, drop the LOB columns. However, if you need to preserve your LOB data in partitioned tables, use the `ALTER TABLE ... EXCHANGE PARTITION` command to move the data into non-partitioned tables, as illustrated in the following example:

Assume you have an existing partitioned table with a LOB column, and the LOB column already contains data that you want to save before downgrading from release 8.1 to release 8.0. The partitioned table has the following definition:

```
CREATE TABLE part_lob_table (part_id NUMBER, part_blob_col BLOB)
  PARTITION BY RANGE (part_id) (
    PARTITION p1 VALUES LESS THAN (10) TABLESPACE ts1,
    PARTITION p2 VALUES LESS THAN (20) TABLESPACE ts2)
  TABLESPACE tsx;
```

Complete the following steps to move the LOB data into non-partitioned tables:

1. Create non-partitioned tables with a LOB column by issuing commands similar to the following:

```
CREATE TABLE lob_table_p1 (id NUMBER, blob_col BLOB);
```

```
CREATE TABLE lob_table_p2 (id NUMBER, blob_col BLOB);
```

Create one table for each partition that is in the partitioned LOB table, but do not insert any data into these new non-partitioned tables.

2. Use EXCHANGE to swap the partitioned table's LOB data from the PART_BLOB_COL column with the non-partitioned tables' LOB data in the BLOB_COL column:

```
ALTER TABLE part_lob_table  
    EXCHANGE PARTITION p1 WITH TABLE lob_table_p1;
```

```
ALTER TABLE part_lob_table  
    EXCHANGE PARTITION p2 WITH TABLE lob_table_p2;
```

These commands move the data from the LOB column PART_BLOB_COL in the partitioned table to the LOB column BLOB_COL in each non-partitioned table.

After you have moved all of the LOB data in partitioned tables to non-partitioned tables, you can downgrade your database and preserve the data.

User-Defined Datatypes

This section describes disabling release 8.1 features related to user-defined datatypes.

Convert User-Defined Datatypes to Release 8.0 Format

Release 8.1 supports a new format for user-defined datatypes. The new format can result in significant performance improvements over the format used in release 8.0.

When you downgrade your database to release 8.0, you must convert your user-defined datatypes to the release 8.0 format. However, if your release 8.1 database has no user-defined datatypes in the new format, you do not need to perform the conversion procedure below.

To identify the user-defined types at 8.1 compatibility level, issue the following SQL statement:

```
SELECT u.name AS USER_NAME, o.name AS TABLE_NAME, c.name AS COLUMN_NAME
FROM sys.user$ u, sys.obj$ o, sys.tab$ t, sys.col$ c, sys.coltype$ ct
WHERE bitand(ct.flags, 128) != 128 AND
       o.obj# = c.obj# and o.obj# = ct.obj# and t.obj# = o.obj# and
       c.intcol# = ct.intcol# AND
       bitand(t.property, 8192) = 0 AND
       u.user# = o.owner# AND
       o.type# = 2 AND
       bitand(c.property, 32) = 0 AND
       (c.type# = 123 OR
        (c.type# = 121 and bitand(c.property, 8) = 0) OR
        (c.type# = 122 and exists
         (SELECT * FROM sys.ntab$ n1, sys.col$ c1, sys.coltype$ ct1
          WHERE n1.obj# = c.obj# AND n1.intcol# = c.intcol# AND
                n1.ntab# = ct1.obj# and bitand(ct1.flags, 128) = 0 AND
                ct1.obj# = c1.obj# and ct1.intcol# = c1.intcol# AND
                bitand(c1.property, 8) = 0)));
```

To downgrade the user-defined datatypes listed, complete the following steps:

1. Export the parts of your release 8.1 database that contain user-defined types at 8.1 compatibility level using the release 8.1 Export utility.
2. Drop the parts of your release 8.1 database that contain user-defined datatypes at 8.1 compatibility level.
3. Downgrade your database using the instructions in ["Downgrade the Database"](#) on page 12-43.
4. Import the exported file into the downgraded database using the release 8.1 Import utility.

See Also: *Oracle8i Utilities* for Export/Import instructions.

Drop Tables With User-Defined Object Identifiers

Release 8.1 supports user-defined object identifiers (OIDs). This functionality enables you to specify your own object identifiers instead of using Oracle's default mechanism for specifying these identifiers. Before you downgrade, drop all tables that have user-defined object identifiers and all tables with REF columns that are based on user-defined object identifiers.

To identify tables with user-defined object identifiers, issue the following SQL statement:

```
SELECT owner, table_name FROM dba_object_tables
       WHERE object_id_type = 'USER-DEFINED';
```

Drop all tables listed.

To identify tables with REF columns that are based on user-defined object identifiers, issue the following SQL statement:

```
SELECT owner, table_name, column_name FROM dba_refs
       WHERE object_id_type = 'USER-DEFINED';
```

Drop all tables listed.

Discontinue Use of Release 8.1 Nested Table Features

Before your downgrade, discontinue use of the following release 8.1 nested table features:

- collection locators
- nested table data in index-organized tables
- user-specified storage clauses for nested tables

To identify tables that use one or more of these features, issue the following SQL statement:

```
SELECT owner, parent_table_name
       FROM dba_nested_tables
       WHERE storage_spec LIKE '%USER_SPECIFIED%'
          OR return_type LIKE '%LOCATOR%';
```

Drop all of the tables listed.

Discontinue Use of Release 8.1 Varray Features

Before your downgrade, discontinue use of specifications of storage parameters for storing varrays as LOBs. To identify tables that use storage parameters for storing a varray as a LOB, issue the following SQL statement:

```
SELECT owner, parent_table_name
       FROM dba_varrays
       WHERE storage_spec LIKE '%USER_SPECIFIED%';
```

Drop all of the tables listed.

Mutually Referencing Types and Downgrading to Release 8.0.4.0 or Lower

If you are using mutually referencing types, then downgrading to release 8.0.3.0 or 8.0.4.0 is not supported. You have two options for downgrading if you are using mutually referencing types:

- Drop the mutually referencing types. Then, downgrading to release 8.0.3.0 or 8.0.4.0 is supported using the procedure in "[Downgrade the Database](#)" on page 12-43.
- Downgrade to release 8.0.4.1 or higher instead of release 8.0.3.0 or 8.0.4.0. If you choose this option, contact Oracle Corporation to get the 8.0.4.1 or higher release.

The following SQL statements provide an example of mutually referencing types:

```
CREATE TYPE manager
/

CREATE TYPE employee AS OBJECT
    (empno NUMBER, ename VARCHAR2(20), mgr REF manager)
/

CREATE OR REPLACE TYPE manager AS OBJECT
    (dept NUMBER, empno REF employee)
/
```

Note: Due to bug #629468, which exists in release 8.0.3.0 and 8.0.4.0, but is fixed in release 8.0.4.1 and higher, the existence of mutually referencing types causes the compilation of package STANDARD to enter a loop and exit with error ORA-01000: "maximum open cursors exceeded". Package STANDARD is required for compilation of PL/SQL code, and is run during a downgrade operation. This bug causes the downgrade to fail.

SQL and PL/SQL

Release 8.1 introduces many changes and additions to SQL and PL/SQL. If you currently use any SQL or PL/SQL code in a script or stored procedure that is available only in release 8.1 and higher, remove this code before you downgrade. You will encounter errors if you try to compile or run the code on a release 8.0 database.

See Also: *Getting to Know Oracle8i*, *Oracle8i SQL Reference*, and *PL/SQL User's Guide and Reference* for information about new SQL and PL/SQL functionality. Also see [Appendix E, "New Internal Datatypes and SQL Functions"](#) in this book.

The following sections describe specific SQL and PL/SQL downgrading issues. The actions described in these sections help you to avoid compile and runtime errors in SQL scripts and stored procedures. Although these actions are not strictly required, Oracle Corporation recommends that you perform them before you downgrade.

Remove C Call Specifications

Before you downgrade, remove stored procedures defined as C call specifications.

Remove Invoker-Rights Clauses

If you use invoker-rights clauses in your SQL code, remove them before you downgrade. Invoker-rights clauses include the AUTHID clause and the SQL_NAME_RESOLVE clause.

Remove Native Dynamic SQL in PL/SQL

PL/SQL programs using native dynamic SQL will cause compile-time errors in releases prior to PL/SQL release 8.1. Before you downgrade, delete all native dynamic SQL syntax in order to compile your programs successfully in release 8.0.

Remove Bulk Binds in PL/SQL

PL/SQL programs using the bulk binds feature will cause compile-time errors in releases prior to PL/SQL 8.1. The bulk binds feature defines new syntax and semantics; thus, the programs containing this feature must be deleted, or, whenever possible, modified to use the scalar binds. PL/SQL statements that use the bulk binds feature contain one or more of the following keywords:

- `FORALL`
- `BULK COLLECT INTO`
- `BULK_ROWCOUNT`

Remove the UROWID Datatype in PL/SQL

If you are using the UROWID datatype as a variable in PL/SQL code, remove this variable before you downgrade.

Delete References to NOCOPY Parameter Passing Mode in PL/SQL

PL/SQL programs using NOCOPY mode will cause compile-time errors in releases prior to PL/SQL 8.1. Before you downgrade, delete references to NOCOPY in order to compile your programs successfully in release 8.0. When you delete references to NOCOPY, make sure the changed aliasing and exception semantics are acceptable.

Java

Java support is not available in release 8.0. Before you downgrade, you must drop all Java objects in your database. The `utljavarm.sql` script drops all Java objects. To identify the Java objects dropped by the `utljavarm.sql` script, issue the following SQL statement:

```
SELECT object_name, owner FROM all_objects WHERE object_type LIKE 'JAVA%';
```

To run the `utljavarm.sql` script, complete the following steps:

1. At a system prompt, change to the `$ORACLE_HOME/rdbms/admin` directory.
2. Start SQL*Plus and connect as a user with SYS privileges.

3. Enter the following:

```
SQL> SPOOL utljavaout.log
SQL> @utljavarm.sql
SQL> SPOOL OFF
```

Check the spool file and verify that the statements executed successfully.

Advanced Queuing (AQ)

Complete the following tasks to disable release 8.1 AQ features in your queue tables:

1. Drop all non-persistent queues and queue tables.
2. Identify the release 8.1 compatible queue tables.
3. Remove the incompatibilities from the release 8.1 compatible queue tables.
4. Downgrade the queue tables.
5. Export the queue tables and import them after downgrading.
6. If you are downgrading to release 8.0.3.0, remove propagation.

These steps are described in detail in the following sections.

See Also: *Oracle8i Application Developer's Guide - Advanced Queuing* for more information about completing the actions described in these sections.

Task 1: Drop All Non-persistent Queues and Queue Tables.

If you are using any non-persistent queues, you must drop these queues and the queue tables that contain them. For every schema (user) that has non-persistent queues, there may be one or two queue tables that contain all the non-persistent queues for that schema. To check for the existence of queue tables that contain non-persistent queues, enter the following SQL statement:

```
SELECT owner, queue_table FROM dba_queue_tables
WHERE queue_table = 'AQ$_MEM_MC' OR queue_table = 'AQ$_MEM_SC';
```

For every queue table returned by the SQL statement, use the DBMS_AQADM.DROP_QUEUE_TABLE procedure (with the force parameter set to TRUE) to drop all of the non-persistent queues and the corresponding queue table. The following is an example of the command:

```
EXECUTE dbms_aqadm.drop_queue_table (  
    queue_table => 'SCOTT.AQ$MEM_MC',  
    force => TRUE);
```

Task 2: Identify the Release 8.1 Compatible Queue Tables

If any of your queue tables are release 8.1 compatible, you must downgrade them. To check the compatibility of your queue tables, enter the following SQL statement:

```
SELECT owner, queue_table, compatible FROM dba_queue_tables  
WHERE compatible LIKE '8.1%';
```

The listed queue tables are release 8.1 compatible and have incompatibilities with release 8.0 that must be removed before you downgrade. Print a list of the queue tables that are release 8.1 compatible. You will need to downgrade these queue tables when you reach "[Task 4: Downgrade the Queue Tables](#)" on page 12-31.

Note: This query does not list queue tables that were at release 8.1 compatibility and then downgraded back to release 8.0 compatibility. However, if you have any such queue tables, you must drop them before you downgrade to release 8.0. Follow the instructions in "[Task 5: Export the Queue Tables and Import Them After Downgrading](#)" on page 12-31 for these queue tables.

Task 3: Remove Incompatibilities from the Release 8.1 Compatible Queue Tables

Your queue tables may have many incompatibilities. These incompatibilities are caused by the use of certain features that are available on release 8.1 but not on release 8.0.

The following sections provide instructions for removing these incompatibilities based on the release 8.1 features in use.

Rule Based Subscriptions Use the `AQ$queue_table_name_r` view to identify queues that use release 8.1 rule based subscription functionality. Perform the check for all of the release 8.1 compatible queue tables listed in Task 2. For example, if a queue is named QTABLE3, issue the following SQL statement to check for rule based subscribers:

```
SELECT * FROM aq$htable3_r;
```

Note: If you receive the error "ORA-04063: view 'aq\$queue_table_name_r' has errors" when you issue this SQL statement, the queue table does not contain any queues with rule-subscribers.

Either drop the rule based subscribers, or change the rule for each rule based subscriber to null using the `DBMS_AQADM.ALTER_SUBSCRIBER` procedure. For example, suppose you have a subscriber for a queue named AQ.MSG_QUEUE with the values shown in [Table 12-1](#):

Table 12-1 Sample Subscriber Values

Parameter	Value
name	SUBSCRIBER1
address	AQ2.MSG_QUEUE2@LONDON
protocol	NULL
rule	'PRIORITY = 1'

You can change the rule to NULL for this subscriber in two different ways: using a PL/SQL block or using a SQL statement. [Example 12-1](#) shows the PL/SQL block, and [Example 12-2](#) shows the SQL statement.

Example 12-1 PL/SQL Block for Changing a Rule to NULL

```
DECLARE
    subscriber sys.aq$_agent;
BEGIN
    subscriber := sys.aq$_agent ('SUBSCRIBER1', 'aq2.msg_queue2@london', null);
    dbms_aqadm.alter_subscriber
        (queue_name => 'aq.msg_queue',
         subscriber => subscriber,
         rule => NULL);
END;
```

Example 12–2 SQL Statement for Changing a Rule to NULL

```
EXECUTE dbms_aqadm.alter_subscriber (  
    'aq.msg_queue',  
    sys.aq$_agent ('SUBSCRIBER1', 'aq2.msg_queue2@london', NULL),  
    NULL);
```

See Also: *Oracle8i Application Developer's Guide - Advanced Queuing* for more information about the DBMS_AQADM.ALTER_SUBSCRIBER procedure.

Object Level and System Level Privileges You are using object level and system level privileges if you used any of the following procedures in the DBMS_AQADM package:

- GRANT_ACCESS_PRIVILEGES
- REVOKE_ACCESS_PRIVILEGES
- GRANT_SYSTEM_PRIVILEGES
- REVOKE_SYSTEM_PRIVILEGES

If you used any of these procedures, all object level and system level privileges must be revoked before you downgrade.

To identify the object level privileges, issue the following SQL statement:

```
SELECT owner, table_name, privilege  
    FROM dba_tab_privs WHERE privilege LIKE '%QUEUE%';
```

Use the DBMS_AQADM.REVOKE_ACCESS_PRIVILEGES procedure to revoke each privilege with ENQUEUE or DEQUEUE in the PRIVILEGE column.

To identify the system level privileges, issue the following SQL statement:

```
SELECT * FROM dba_sys_privs WHERE privilege LIKE '%QUEUE%';
```

Use the DBMS_AQADM.REVOKE_SYSTEM_PRIVILEGE procedure to revoke each privilege with any of the following types of privileges listed in the PRIVILEGE column:

- MANAGE ANY QUEUE
- ENQUEUE ANY QUEUE
- DEQUEUE ANY QUEUE

Task 4: Downgrade the Queue Tables

Complete the following steps to downgrade each queue table that was incompatible with release 8.0. You listed all of the incompatible queue tables in "[Task 2: Identify the Release 8.1 Compatible Queue Tables](#)" on page 12-28.

1. Before you downgrade, disable all propagation schedules for all queues in the queue table using DBMS_AQADM.DISABLE_PROPAGATION_SCHEDULE.
2. Downgrade the incompatible queue tables back to release 8.0 compatibility.

To downgrade a queue table, run the DBMS_AQADM.MIGRATE_QUEUE_TABLE procedure and specify 8.0 for the COMPATIBLE setting. The following example illustrates running this procedure:

```
EXECUTE dbms_aqadm.migrate_queue_table (  
  queue_table => 'sys.tkaqqtdef',  
  compatible => '8.0');
```

Note: Only the owner of a queue table can run the DBMS_AQADM.MIGRATE_QUEUE_TABLE procedure on the queue table.

Task 5: Export the Queue Tables and Import Them After Downgrading

Complete the following steps to export the incompatible queue tables and import them after downgrading. You listed all of the incompatible queue tables in "[Task 2: Identify the Release 8.1 Compatible Queue Tables](#)" on page 12-28.

1. Using the release 8.1 Export utility, export all incompatible queue tables.
2. Drop all the queue tables that have been exported.
3. Follow the instructions in the rest of this chapter to downgrade the database.
4. After the database is downgraded to release 8.0, import the exported file into the database using the release 8.1 Import utility.

Repeat these steps for every incompatible queue table.

See Also: *Oracle8i Application Developer's Guide - Advanced Queuing* for information about exporting and importing queue tables.

Task 6: If You Are Downgrading to Release 8.0.3, Remove Propagation

If you are using message propagation in the Advanced Queuing Option, and you are downgrading to release 8.0.3, remove propagation. If you are not using message propagation, or if you are downgrading to release 8.0.4 or higher, skip this task.

Complete the following steps to remove propagation:

1. Identify your multi-consumer queues by issuing the following SQL statement:

```
SELECT owner, queue_table FROM dba_queue_tables
       WHERE recipients = 'MULTIPLE';
```

Save the results of this query. Propagation is supported only from multi-consumer queues. If this query does not return any rows, propagation is not in use and you can skip the remaining steps in this procedure.

2. Determine if any of the multi-consumer queues are utilizing the propagation feature by running the following SQL statement for each queue table listed by the query in Step 1.

The following SQL statement uses queue table SCOTT.QTABLE1 as an example:

```
SELECT unique(q_name) FROM scott.qtable1 a
       WHERE EXISTS (SELECT consumer
                    FROM the (SELECT cast(history as sys.aq$dequeue_history_t)
                             FROM scott.qtable1 b
                             WHERE a.msgid = b.msgid)
                    WHERE consumer like 'AQ$_%');
```

For each queue table, you must drop all queues returned by the SQL statement.

Note: To list all of your queue tables, query the DBA_QUEUE_TABLES static data dictionary view.

3. Check the DBA_QUEUE_SCHEDULES static data dictionary view to determine if there are propagation schedules for any queues. You must unschedule propagation for each queue that is selected in the view using the DBMS_AQADM.UNSCHEDULE_PROPAGATION procedure.
4. Eliminate all remote subscribers. Remote subscribers have the address field or protocol field specified. Run the DBMS_AQADM.QUEUE_SUBSCRIBERS procedure and check for subscribers with a non-null address field or a non-null protocol field. Drop all of these subscribers.

Procedures and Packages

This section describes disabling release 8.1 features related to procedures and packages.

Syntax Change for the SET_SESSION_LONG_LONGOPS Procedure

Release 8.1 introduces changes to the DBMS_APPLICATION_INFO.SET_SESSION_LONGOPS procedure. If any of your applications use this procedure and you changed them to conform to the release 8.1 syntax, change the applications accordingly so that they conform to the release to which you are downgrading. For information about the syntax, refer to the `dbmsapin.sql` file in the release to which you are downgrading.

The UTL_REF Package

If you are downgrading to release 8.0.3, discontinue use of the UTL_REF package. This package is not available in release 8.0.3.

If you are downgrading to release 8.0.4, the UTL_REF package will be dropped automatically during the downgrading process. The package is dropped because the UTL_REF package is not part of the standard release 8.0.4 installation. To continue using this package, you must re-install it manually after downgrading to release 8.0.4.

Note: If you are downgrading to release 8.0.5 or higher, no action is required for the UTL_REF package. This package is part of the standard installation for release 8.0.5 and higher and is preserved automatically during the downgrade process.

The DBMS_REPAIR Package

Release 8.1 supports the DBMS_REPAIR package. Before you downgrade, make sure all objects have skip corrupt disabled.

To identify objects that have skip corrupt enabled, issue the following SQL statement:

```
SELECT owner, table_name from dba_tables where skip_corrupt = 'ENABLED';
```

For each such table selected, clear the skip corrupt attribute. For example, for a table named TB_5 owned by SCOTT, enter the following:

```
EXECUTE DBMS_REPAIR.SKIP_CORRUPT_BLOCKS (schema_name => 'SCOTT',  
    object_name => 'TB_5', flags => DBMS_REPAIR.NOSKIP_FLAG);
```

Constraints and Triggers

This section describes removing incompatibilities relating to constraints and triggers.

Discontinue Use of DISABLE VALIDATE Constraints

Release 8.1 supports the DISABLE VALIDATE constraint state. Before you downgrade, you must drop or invalidate all DISABLE VALIDATE constraints.

To identify DISABLE VALIDATE constraints, issue the following SQL statement:

```
SELECT constraint_name, status, validated  
    FROM dba_constraints  
    WHERE status = 'DISABLED'  
    AND validated = 'VALIDATED';
```

Use the DROP clause in the ALTER TABLE command to drop all of the constraints listed. Or, use the DISABLE clause in the ALTER TABLE command to invalidate all of the constraints listed.

Drop Triggers on Nested Table View Columns

Release 8.1 supports creating triggers on nested table view columns. Before you downgrade, you must drop all of these triggers.

To identify nested table triggers on view columns, enter the following SQL statement:

```
SELECT trigger_name, table_name, column_name  
    FROM dba_triggers  
    WHERE column_name IS NOT NULL;
```

Drop all of the triggers listed using the DROP TRIGGER command.

Drop Incompatible Triggers

Triggers are enhanced in release 8.1 to support database event publication. Before you downgrade, all triggers that are incompatible with release 8.0 must be dropped.

To identify the triggers that must be dropped during the downgrade process, issue the following SQL statement:

```
SELECT trigger_name, base_object_type, action_type
       FROM dba_triggers
       WHERE base_object_type
             IN ('DATABASE', 'SCHEMA')
             OR action_type = 'CALL';
```

Triggers on SCHEMA and DATABASE cannot be made compatible with release 8.0; you must drop these triggers. However, CALL triggers can be preserved during the downgrade process. To make CALL triggers compatible with release 8.0, wrap a BEGIN ... END block around the CALL statement.

Oracle Optimizer

This section describes removing incompatibilities relating to the Oracle optimizer.

Extensible Optimizer

Release 8.1 supports the extensible optimizer. Before you downgrade, you must discontinue use of the extensible optimizer by dropping all associations. To identify associations, enter the following SQL statement:

```
SELECT object_owner, object_name, column_name, object_type
       FROM dba_associations;
```

For each association listed, run the DISASSOCIATE STATISTICS command with the FORCE option. For example, assume you receive the following output when you issue the preceding SQL statement:

OBJECT_OWNER	OBJECT_NAME	COLUMN_NAME	OBJECT_TYPE
-----	-----	-----	-----
SYS	TYPE1		TYPE
TKOQEX	TKOQ_TAB1	A	COLUMN

2 rows selected.

Issue the appropriate DISASSOCIATE STATISTICS statement corresponding to the object type listed. To drop the associations listed, where the object types are TYPE and COLUMN, issue the following SQL statements:

```
DISASSOCIATE STATISTICS FROM types sys.type1 FORCE;
```

```
DISASSOCIATE STATISTICS FROM columns tkogex.tkoq_tab1.a FORCE;
```

See Also: *Oracle8i SQL Reference* for more information about the DISASSOCIATE STATISTICS command.

Optimizer Plan Stability

Release 8.1 supports optimizer plan stability. This feature enables you to create stored outlines with the CREATE OUTLINE statement. Stored outlines are not supported in release 8.0.

To identify stored outlines, issue the following SQL statement:

```
SELECT owner, name FROM dba_outlines;
```

Drop any outlines listed by this SQL statement.

Security

This section describes removing incompatibilities relating to database security.

Drop All Application Contexts

The ability to specify an application context is a new feature in release 8.1. Before you downgrade, drop all application contexts. To identify the application contexts, issue the following SQL statement:

```
SELECT * FROM dba_context;
```

Drop all of the application contexts listed by this SQL statement using a DROP CONTEXT statement.

Drop All User-Defined Security Policies

Fine-grained access control is a new feature in release 8.1 that enables the creation of user-defined security policies. Before you downgrade, drop all user-defined security policies.

To identify user-defined security policies, issue the following SQL statement:

```
SELECT object_owner, object_name, policy_name
       FROM dba_policies;
```

Drop all of the policies listed by this SQL statement using the DBMS_RLS.DROP_POLICY procedure.

Revoke CONNECT THROUGH Privileges for Proxy Users

The release 8.1 n-tier authentication and authorization feature is not available in release 8.0. Therefore, if any proxy users have CONNECT THROUGH privileges, you must revoke these privileges.

To list the proxy users, issue the following SQL statement:

```
SELECT * FROM proxy_users;
```

To revoke CONNECT THROUGH privileges, issue an ALTER USER ... REVOKE CONNECT THROUGH statement. For example, the following statement revokes the right of proxy user APPSERVER1 to connect as the user JANE:

```
ALTER USER jane REVOKE CONNECT THROUGH appserver1;
```

Database Backup and Recovery

This section provides information about ensuring that your backups can be recovered by your downgraded database.

Oracle Media Management API and Proxy Copy

Oracle Media Management API version 2 supports proxy copy functionality, but this functionality will not be supported after you downgrade your database to release 8.0. Therefore, any release 8.1 proxy backups created using a version 2 software backup to tape (SBT) layer that supports proxy copy cannot be restored using release 8.0.

If you may need to restore backups of your release 8.1 database with your downgraded release 8.0 database, before you downgrade, create these backups with proxy copy turned off, because turning proxy copy off enables release 8.0 to restore the backups. Also, if your media manager provides only Oracle Media Management API version 2 support, you should obtain a version 1.1 SBT layer to use with release 8.0.

Change Back to the Old Archive Log Destination Parameters

If you used the new archive log destination parameters in release 8.1 (LOG_ARCHIVE_DEST_1 and LOG_ARCHIVE_DEST_STATE_1), switch back to the old archive log destination parameters before you downgrade (LOG_ARCHIVE_DEST and LOG_ARCHIVE_DUPLEX_DEST).

See Also: ["Changing Back to the Old Archive Log Destination Parameters"](#) on page B-14 for instructions.

Distributed Databases

This section describes removing incompatibilities relating to distributed databases.

Prepare Your Advanced Replication Environment for Downgrading

If you are using Advanced Replication, perform the actions described in the following sections to prepare the Advanced Replication environment for downgrading.

See Also: *Oracle8i Replication* for more information about completing these actions.

Remove Incompatibilities In Your Advanced Replication Environment If any database in the replication environment sends RPCs that use the release 8.1 protocol to the database you are downgrading, you must either apply or delete all deferred RPCs before you downgrade. A database's RPCs use the release 8.1 protocol if GENERATE_80_COMPATIBLE has ever been set to FALSE in any of the following calls:

- GENERATE_REPLICATION_SUPPORT (affects all master sites)
- CREATE_SNAPSHOT_REOBJECT (affects the local snapshot site)
- GENERATE_SNAPSHOT_SUPPORT (affects the local snapshot site)

Complete the following steps to apply or delete all deferred RPCs:

1. If the database to be downgraded is a master site for one or more object groups and the masterdef site is at release 8.1 or higher, complete the following steps:
 - a. Quiesce each of the object groups.
 - b. Convert each master to the top flavor of each object group.
 - c. Regenerate replication support for each replicated table by setting `GENERATE_80_COMPATIBLE` to `TRUE` in the `GENERATE_REPLICATION_SUPPORT` calls.
 - d. Resume master activity for the quiesced replication object groups.
 - e. Convert each snapshot site of each master to the top flavor of each object group.
2. If the database to be downgraded is a master site for one or more snapshot sites that are at release 8.1 or higher, complete the following steps:
 - a. Regenerate support for all snapshots at these sites by setting `GENERATE_80_COMPATIBLE` to `TRUE` in the `GENERATE_SNAPSHOT_SUPPORT` calls.
 - b. Push the queue from each release 8.1 or higher snapshot site.
3. Use either the `DBMS_DEFER_SYS.EXECUTE_ERROR` or the `DBMS_DEFER_SYS.DELETE_ERROR` procedure to apply or delete each error in the error queue for the database to be downgraded.

Make Sure the Database's Object Groups Are At the Top Flavor Each object group of the database to be downgraded must be at the top flavor before you downgrade. To identify the object groups that are not at the top flavor, issue the following SQL statement:

```
SELECT gname, fname FROM dba_repgroup
       WHERE fname IS NOT NULL;
```

If any of the database's object groups are listed, you must change the flavor of the listed object groups to the top flavor before downgrading. If an object group is not listed because `FNAME` is null for the object group, then the object group already is at the top flavor, and no action is required for it.

Remove Temporary Updatable Snapshot Logs Determine if you have temporary updatable snapshot logs by issuing the following SQL statement:

```
SELECT owner, table_name FROM dba_tables
       WHERE temporary='Y' AND
       table_name LIKE 'RUPD$%';
```

If any rows are returned, temporary updatable snapshot logs exist in your database. Run the following PL/SQL block to remove them:

```
DECLARE
  sql_cur  BINARY_INTEGER;
  dummy    BINARY_INTEGER;
  new_flag BINARY_INTEGER;

  CURSOR mv_logs IS
    SELECT '''||mowner||'."'||temp_log||''' temp_log,
           flag, mowner, master
    FROM mlog$ m
    WHERE temp_log IS NOT NULL
    FOR UPDATE;
BEGIN
  sql_cur := dbms_sql.open_cursor;
  FOR alog IN mv_logs LOOP
    new_flag := alog.flag;
    IF dbms_ijob.bit(new_flag, 64) THEN ---KKZLOGTUPS
      new_flag := new_flag - 64;
    END IF;

    BEGIN
      dbms_sql.parse(sql_cur, 'DROP TABLE ' || alog.temp_log, dbms_sql.v7);
      dummy := dbms_sql.execute(sql_cur);

      UPDATE mlog$ m
        SET flag = new_flag, temp_log = NULL
        WHERE m.mowner = alog.mowner AND m.master = alog.master;
    EXCEPTION WHEN others THEN
      NULL; --- Ignore the error
    END;
  END LOOP;
  dbms_sql.close_cursor(sql_cur);
  COMMIT;
```

```
EXCEPTION WHEN others THEN
    IF dbms_sql.is_open(sql_cur) THEN
        dbms_sql.close_cursor(sql_cur);
    END IF;
    RAISE;
END;
/
```

Identify Incompatibilities in Snapshots The word "snapshot" is synonymous with the word "materialized view".

See Also: ["Identify Materialized Views That Will Be Changed or Dropped During Downgrade"](#) on page 12-9 for information about identifying incompatibilities in materialized views.

Net8

This section describes removing incompatibilities relating to Net8.

Discontinue Use of Service Naming

Release 8.1 supports service naming in Net8, but service naming is not supported in release 8.0. To discontinue use of service naming, perform the following actions:

- Remove the SERVICE_NAMES and INSTANCE_NAME parameters in your `initsid.ora` file.
- Remove the SERVICE_NAME parameter in the `tnsnames.ora` file.
- Use the SID parameter in connect descriptors.
- Use the SID_LIST parameter in the `listener.ora` file to manually configure information about the instances served by the listener.

See Also: See the appropriate Net8 or SQL*Net documentation for the release to which you are downgrading for information about using the SID parameter in connect descriptors.

Reset Database Compatibility

After you have removed all of the incompatibilities with the release to which you are downgrading, reset the compatibility level of the database to the prior release by completing the following steps:

1. If you are using any initialization parameters that were added in a release higher than the release to which you are downgrading, remove them from your `init sid .ora` file.

See Also: [Appendix B, "Changes to Initialization Parameters"](#) for lists of parameters added in each release.

2. Start SQL*Plus and connect as a user with SYSDBA privileges.
3. Start the database using STARTUP.
4. Run ALTER DATABASE RESET COMPATIBILITY:

```
SQL> ALTER DATABASE RESET COMPATIBILITY;
```

See Also: ["About ALTER DATABASE RESET COMPATIBILITY"](#) on page 8-7 for more information.

5. Run SHUTDOWN IMMEDIATE:

```
SQL> SHUTDOWN IMMEDIATE
```

CAUTION: Do not open the database with COMPATIBLE set higher than the release to which you are downgrading after completing the step.

6. Set the COMPATIBLE parameter in the `init sid .ora` file to match the release to which you are downgrading.

For example, if you are downgrading to release 8.0.5, set the COMPATIBLE parameter to the following:

```
COMPATIBLE=8.0.5
```

7. Open the database to ensure that it is compatible with release you specified with the COMPATIBLE parameter.

If your database fails to open, some incompatibilities still exist. If so, reset the COMPATIBLE parameter to a higher setting, such as 8.1.0. Then, remove the incompatibilities and attempt to reset database compatibility again. All incompatibilities with the database to which you are downgrading must be removed before you proceed with the downgrading process described in ["Downgrade the Database"](#) on page 12-43.

See Also: ["Remove Incompatibilities"](#) on page 12-2 for information about removing incompatibilities.

Downgrade the Database

Make sure your database is compatible with the release to which you are downgrading before you perform the downgrade steps in this section. See ["Remove Incompatibilities"](#) on page 12-2 if you have not removed incompatibilities yet.

Complete the following steps to downgrade your release 8.1 database to an older release:

1. Copy the following files from the `$ORACLE_HOME/rdbms/admin` directory to a directory outside of Oracle home, such as the temporary directory on your system:
 - `catlg803.sql`
 - `utlip.sql`
 - `utlrp.sql`

Make a note of the new location of these files. You may need them later in the downgrade process.

2. At a system prompt, change to the `$ORACLE_HOME/rdbms/admin` directory.

3. Start SQL*Plus with the NOLOG option:

```
sqlplus /nolog
```

Note: You must use SQL*Plus for this part of the downgrade procedure. Do not use Server Manager.

4. Connect to the database instance:

```
SQL> CONNECT INTERNAL
```

5. If the database is shut down, start the database using STARTUP:

```
SQL> STARTUP
```

You may need to use the PFILE option to specify the location of your *initsid.ora* file.

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

```
SQL> SPOOL catoutd.log
```

If you want to see the output of the scripts you will run on your screen, you also can issue a SET ECHO ON statement:

```
SQL> SET ECHO ON
```

7. Run `dold_release.sql` where *old_release* refers to the release to which you are downgrading. See [Table 12-2](#) to choose the correct script. Each script provides a direct downgrade to the release specified in the "Downgrading To" column.

To run a script, enter the following:

```
SQL> @dold_release.sql
```

Table 12-2 Downgrade Scripts

Downgrading To	Run Script
8.1.4	d0801040.sql
8.1.3	d0801030.sql
8.1.2	Not supported.
8.1.1	Not supported.
8.0.5	d0800050.sql
8.0.4S	Not supported.
8.0.4	d0800040.sql
8.0.3	d0800030.sql
8.0.2	Not supported.
8.0.1	Not supported.

Note: If the release to which you are downgrading is not included in [Table 12-2](#), see the readme files in the new installation for the correct downgrade script to run.

The following are notes about running the script:

- You must use the version of the script included with the release from which you are downgrading.
- You must run the script in the environment of the release from which you are downgrading.

- You only need to run one script, even if your downgrade spans several releases. For example, if you are downgrading to release 8.0.3, then you need to run only `d0800030.sql`.
- If you are using mutually referencing types, then downgrading to release 8.0.3 or 8.0.4.0 is not supported. See "[Mutually Referencing Types and Downgrading to Release 8.0.4.0 or Lower](#)" on page 12-24 for more information.

If you encounter any problems when you run the script, or any of the scripts in the remaining steps, correct the causes of the problems and rerun the script. You can rerun any of the scripts described in this chapter as many times as necessary.

See Also: "[Running Scripts](#)" on page 1-3 for information about the types of errors to look for when you run a script.

8. Run SHUTDOWN IMMEDIATE and exit SQL*Plus:

```
SQL> SHUTDOWN IMMEDIATE
SQL> EXIT
```

If you are using Oracle Parallel Server, shutdown all instances.

9. Change the following environment variables point to the directories of the release to which you are downgrading:

- ORACLE_HOME
- PATH
- ORA_NLS
- ORACLE_BASE
- LD_LIBRARY_PATH (on UNIX)
- ORACLE_PATH (if set)

Note: Some of these environment variables may not apply to your operating system.

See Also: Your operating-system specific Oracle8i installation documents for information about setting other important environment variables on your operating system.

10. Install the release to which you are downgrading using the installation media for that release.

For example, if you are downgrading to release 8.0.5, use the release 8.0.5 installation media to install the release 8.0.5 distribution of Oracle.

Also, if you are downgrading to an 8.0 release, you must install the release 8.0 software in an Oracle home separate from the 8.1 release. However, if you are downgrading to a previous 8.1 release, this restriction does not apply, and you can install the new release into the same Oracle home if you wish.

Note: Installation is operating-system specific. For installation instructions, see your operating-system specific installation documentation and the README for your operating system.

11. If you are using separate Oracle homes and your `init sid .ora` file resides within the Oracle home of the database from which you are downgrading, copy the `init sid .ora` file to a location outside of the Oracle home. In release 8.1, the default location for the `init sid .ora` file is `$ORACLE_BASE/admin/ sid /pfile`, where `sid` is the Oracle instance ID, but in past releases, the default was `$ORACLE_HOME/dbs` on UNIX and `$ORACLE_HOME\database` on Windows NT. The `init sid .ora` file can reside anywhere you wish, but it should not reside in the Oracle home of the release from which you are downgrading.

If your `init sid .ora` file has an IFILE (include file) entry and the file specified in the IFILE entry resides within the Oracle home of the database from which you are downgrading, then copy the file specified by the IFILE entry to a location outside of the Oracle home. The file specified in the IFILE entry has additional initialization parameters. After you copy this file, edit the `init sid .ora` file to point to its new location.

Note: For Oracle Parallel Server, perform this step on all nodes. Also, set the `PARALLEL_SERVER` initialization parameter to `FALSE`. You can change it back to `TRUE` after the downgrade operation is complete.

12. Copy the following files into the `$ORACLE_HOME/rdbms/admin` directory:

- `catlg803.sql`
- `utlip.sql`
- `utlrp.sql`

You copied these files to a directory outside of Oracle home in Step 1.

13. At a system prompt, change to the `$ORACLE_HOME/rdbms/admin` directory.

14. Start Server Manager and run `CONNECT INTERNAL`:

```
SVRMGR> CONNECT INTERNAL
```

Note: You must use Server Manager for the rest of the downgrade procedure. Do not use SQL*Plus.

15. Run `STARTUP`:

```
SVRMGR> STARTUP
```

You may need to use the `PFILE` option to specify the location of your `init sid .ora` file.

16. Perform this step if *either one* of the following conditions is true:

- Your installation involves a change in word-size (switching between 32-bit and 64-bit software).
- You are downgrading to release 8.0.

Otherwise, this step is optional (performing this step if it is not necessary could cause invalidations).

Run `utlip.sql`:

```
SVRMGR> @utlip.sql
```

The `UTLIP.SQL` script invalidates all existing PL/SQL modules by altering certain dictionary tables so that subsequent recompilations will happen in the format required by the database. It also reloads package `STANDARD` and `DBMS_STANDARD`, which are necessary for any PL/SQL compilations.

17. Run either the `catalog.sql` script or the `catlg803.sql` script, depending on the release to which you are downgrading. *Do not* run both of these scripts.

If you are downgrading to release 8.0.4 or higher, run `catalog.sql`:

```
SVRMGR> @catalog.sql
```

If you are downgrading to release 8.0.3, run `catlg803.sql`:

```
SVRMGR> @catlg803.sql
```

Note: Due to bug #571546, which exists in release 8.0.3 but is fixed in release 8.0.4 and higher, you *should not* run `catalog.sql` after downgrading to release 8.0.3. The recreation of package STANDARD triggers this bug. Because `catalog.sql` recreates package STANDARD, Oracle has provided a new script (`catlg803.sql`) that effectively does everything the release 8.0.3 `catalog.sql` script does, except for performing a few additional steps to work around the problem described in bug #571546.

18. Run `catproc.sql`:

```
SVRMGR> @catproc.sql
```

19. If the Oracle system has Advanced Replication installed, run the following catalog script supplied with the release to which you downgraded:

```
SVRMGR> @catrep.sql
```

20. If the Oracle system has Parallel Server installed, run the following catalog script supplied with the release to which you downgraded:

```
SVRMGR> @catparr.sql
```

21. Run `utlrp.sql`. This step is optional and can be done regardless of whether there was a change in word-size.

```
SVRMGR> @utlrp.sql
```

The `utlrp.sql` script recompiles all existing PL/SQL modules that were previously in an INVALID state, such as packages, procedures, types, etc. These actions are optional; however, they ensure that the cost of recompilation is incurred during installation rather than in the future.

Oracle Corporation highly recommends running `utlrp.sql`.

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

```
SVRMGR> SPOOL OFF;
```

Then, check the spool file and verify that the packages and procedures compiled successfully. You named the spool file in Step 6; the suggested name was `catoutd.log`. Correct any problems you find in this file.

If you specified `SET ECHO ON`, you may want to `SET ECHO OFF` now:

```
SVRMGR> SET ECHO OFF;
```

23. If you removed mutually referencing views while following the instructions in ["Mutually Referencing Views and Downgrading to Release 8.0.4 or Lower"](#) on page 12-10, and you downgraded to release 8.0.4, recreate these views now.

Note: You cannot recreate these views if you downgraded to release 8.0.3, because mutually referencing views are not supported in release 8.0.3.

Your database is now downgraded. Complete the procedures described in the following sections to finish downgrading specific components.

Regenerating Advanced Replication Support

If you are using Advanced Replication, before you regenerate Advanced Replication support, make sure you completed the procedures described in "[Prepare Your Advanced Replication Environment for Downgrading](#)" on page 12-38. Then, complete the actions described below based on whether the downgraded database is a master site or a snapshot site.

Master Site

If the downgraded database is a master site for one or more object groups, complete the following steps to regenerate Advanced Replication support:

1. Quiesce each object group.
2. Generate replication support for each replicated table in the group.
3. Resume master activity for the object group. If the masterdef site is at release 8.1, make sure you specify `GENERATE_80_COMPATIBLE=>TRUE` in the `GENERATE_REPLICATION_SUPPORT` calls.

Snapshot Site

If the downgraded database is a snapshot site, generate replication support for each updatable snapshot.

See Also: *Oracle8i Replication* for more information about generating replication support.

Re-Installing the UTL_REF Package on Release 8.0.4

If you downgraded to release 8.0.4, and you were using the UTL_REF package before you downgraded, re-install the UTL_REF package. This package was automatically dropped during the downgrading process because the package is not part of the standard installation for release 8.0.4.

Note: If you downgraded to release 8.0.5 or higher, no action is required. The UTL_REF package was preserved during the downgrading process. If you downgraded to release 8.0.3, you cannot use the UTL_REF package because it is not available in release 8.0.3.

Re-Installing Recovery Manager Packages on Release 8.0.3

If you downgraded to release 8.0.3, and if you used Recovery Manager (RMAN) release 8.0.4 or higher before you downgraded, re-install the following release 8.0.3 packages on the recovery catalog database:

- `dbmsrman.sql` and `prvtrmnu.plb`
- `dbmsrvct.sql` and `prvtrvct.plb`

Note: If you downgraded to release 8.0.4 or higher, or if you did not use Recovery Manager release 8.0.4 or higher before downgrading to release 8.0.3, you do not need to re-install these packages.

Downgrading to Oracle7

The information in this chapter only applies to release 8.0 and higher installations of Oracle. The term *downgrading* describes transforming an Oracle database into a previous release of the same version, such as transforming a database from release 8.1 to release 8.0. The term *downgrading* also describes transforming an Oracle database into a previous version, such as transforming a database from Oracle8i to Oracle7.

This chapter describes downgrading to Oracle7. If you want to downgrade to a previous 8.0 or 8.1 release, see [Chapter 12](#).

This chapter covers the following topics:

- [Overview of Downgrading from Oracle8i to Oracle7](#)
- [Downgrading a Database That Does Not Contain New or Changed Data](#)
- [Downgrading a Database That Contains New or Changed Data](#)

See Also: Some aspects of downgrading are operating-system specific. See your operating-system specific Oracle documentation for additional instructions about downgrading.

Overview of Downgrading from Oracle8i to Oracle7

An Oracle8i database can be downgraded to an Oracle7 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. Also, it may not be possible to preserve data that uses new version 8 features that are not available with Oracle7.

Oracle does not support downgrading from Oracle8i to Oracle7 using the Oracle8i Migration utility; in other words, the Oracle8i Migration utility does not support backward migration. Oracle8i provides no other facilities specifically for downgrading.

The procedure for downgrading depends on whether the Oracle8i database contains new or changed data that must be preserved. Use the procedure that applies to your Oracle8i database:

- [Downgrading a Database That Does Not Contain New or Changed Data](#)
- [Downgrading a Database That Contains New or Changed Data](#)

Downgrading a Database That Does Not Contain New or Changed Data

If the Oracle8i database contains *no new or changed data* that must be preserved, simply restore the complete backup of the previous Oracle7 source database and open it again. Make sure the restore includes the initialization parameters that were used in the previous Oracle7 database.

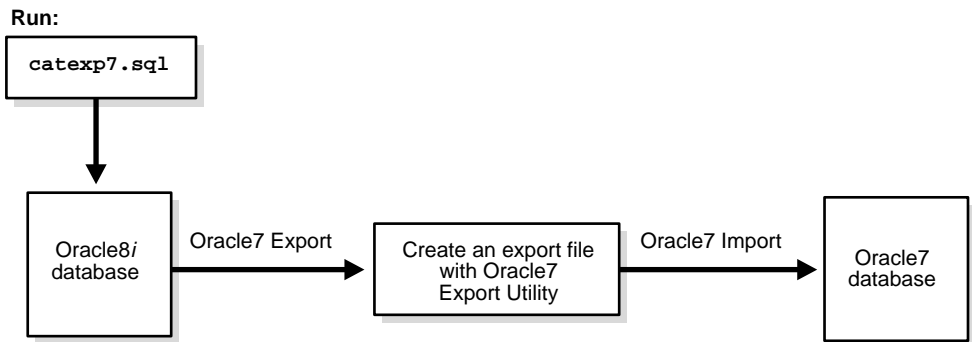
See Also: *Oracle7 Server Administrator's Guide* for more information about restoring a backed up Oracle7 database.

Any new or changed data in the release 8.1 database is lost when you use the method described in the previous paragraph. If your release 8.1 database has new or changed data that must be preserved, use the procedure described in "[Downgrading a Database That Contains New or Changed Data](#)" on page 13-3.

Downgrading a Database That Contains New or Changed Data

If the Oracle8i database contains *new or changed data that must be preserved*, complete the procedure illustrated in [Figure 13-1](#).

Figure 13-1 Downgrading to Oracle7



The following steps describe the procedure in more detail:

1. Run the `catexp7.sql` script on the Oracle8i database. This script is in the `$ORACLE_HOME/rdbms/admin` directory.
2. Use the Oracle7 Export utility to export the parts of the Oracle8i database containing the new or changed data.

See Also: *Oracle8i Utilities* for more information about performing an Oracle7 export from an Oracle8i database.
3. Restore the complete backup of the previous Oracle7 database, and make sure the restore includes the previous initialization parameters.

See Also: *Oracle7 Server Administrator's Guide* for more information about restoring a backed up Oracle7 database.
4. Open the restored Oracle7 database.
5. Use the Oracle7 Import utility to import the file previously exported from the Oracle8i database into the restored Oracle7 database.

Alternative Downgrading Methods

Several other methods are available for sending table data from the Oracle8i database back to the Oracle7 database. These methods of returning data to Oracle7 are relatively simple if only a few tables have been updated using Oracle8i. However, copying an entire database of tables can be a long and complicated task; therefore, you should decide whether you need to return to Oracle7 before you update many tables using Oracle8i.

The following alternate methods are available for downgrading an Oracle8i database to Oracle7:

- Use SQL*Plus to create non-Oracle text files from the new Oracle8i data and then use SQL*Loader to load the data back into the Oracle7 database.
- Use the SQL*Plus COPY command to copy the new data from the Oracle8i database tables into the tables in the earlier Oracle7 database.

See Also: The *SQL*Plus User's Guide and Reference* for more information about the COPY command.
- If you still are running the earlier Oracle7 database, re-create the table where data has been added or changed (using CREATE TABLE ... AS SELECT). Select the data in the Oracle8i database table through a distributed query from the Oracle7 database using a database link.

See Also: *Oracle8i SQL Reference* for more information on the AS clause of the CREATE TABLE command.

Troubleshooting Migration Problems

This appendix describes corrective action for common problems that you may encounter during the migration process when you are using either the Migration utility or the Oracle Data Migration Assistant. If you are using the Migration utility, the migration process is described in [Chapter 3, "Migrating Using the Migration Utility"](#); if you are using the Oracle Data Migration Assistant, the migration process is described in [Chapter 4, "Migrating Using the Oracle Data Migration Assistant"](#).

This appendix covers the following topics:

- [Problems Using the Migration Utility or Oracle Data Migration Assistant](#)
- [Problems at the ALTER DATABASE CONVERT Command](#)

Problems Using the Migration Utility or Oracle Data Migration Assistant

When you run either the Migration utility or the Oracle Data Migration Assistant, you may encounter the following types of problems:

- [General Migration Problems](#)
- [Migration Utility Errors](#)

General Migration Problems

General migration problems may occur when you run either the Migration utility or the Oracle Data Migration Assistant, but they are caused by your database system's configuration. While either the utility or the assistant is performing the necessary actions to migrate the database, an error is generated by your Oracle software. Typically, when such an error occurs, the utility or assistant stops and displays one or more error messages.

If you encounter one of the following problems when you run either the utility or the assistant, perform the suggested actions, and then re-run the utility or assistant.

Note: If you are using the Oracle Data Migration Assistant, you must restore the backup of your Oracle7 database before you rerun the assistant.

Insufficient Space in the SYSTEM Tablespace

This problem may return an error message similar to the following:

```
ORA-01652: unable to extend temp segment by 473 in tablespace SYSTEM
```

You need to add a new datafile to the SYSTEM tablespace and allocate enough space to the new datafile to successfully complete the migration.

It also is possible to run out of space in the temporary tablespace during migration. If you do, add a new datafile to the temporary tablespace and allocate enough space to the new datafile to successfully complete the migration.

See Also: If you are using the Migration utility, see "[Space Requirements](#)" on page 3-6 and Step 8 on page 3-18. If you are using the Oracle Data Migration Assistant, see "[Space Requirements](#)" on page 4-4 and Step 9 on page 4-10.

These sections provide more information about the space requirements for the SYSTEM tablespace, and information about adding a new datafile to increase its available space.

Incorrect AUDIT_TRAIL Parameter Setting

This problem may return error messages similar to the following:

```
ORA-00604: error occurred at recursive SQL level num
ORA-01552: cannot use system rollback segment for non-system tablespace 'name'
ORA-02002: error while writing to audit trail
```

You will encounter these errors only under the following conditions:

- either the initialization parameter AUDIT_TRAIL is set to DB or to TRUE
- the SYS.AUD\$ table is located in a tablespace other than SYSTEM

To correct this problem, complete the following steps:

1. Shutdown the database if it is open.
2. Set the AUDIT_TRAIL parameter in the *init*sid*.ora* file in the following way:

```
AUDIT_TRAIL = NONE
```

3. Re-run the Migration utility, or restore your Oracle7 backup and rerun the Oracle Data Migration Assistant.

OPTIMAL Setting for the SYSTEM Rollback Segment

This problem may return an error message similar to the following:

```
ORA-01562: failed to extend rollback segment (id = 0)
```

Both the Migration utility and the Oracle Data Migration Assistant take all non-SYSTEM rollback segments offline and then freeze the size of the SYSTEM rollback segment by altering MAXEXTENTS to the number of extents currently allocated. This action prevents any space operations, such as an extent allocation, while the utility or assistant handles the space management tables.

If the SYSTEM rollback segment has an OPTIMAL setting, extents are de-allocated dynamically when their data is no longer needed for active transactions. The dynamic de-allocation may cause the number of currently allocated extents to be small when the SYSTEM rollback segment is frozen. Therefore, the SYSTEM rollback segment may not be large enough to handle the transactions involving the space management tables during migration.

See Also: If you are using the Migration utility, see Step 7 on page 3-11. If you are using the Oracle Data Migration Assistant, see Step 7 on page 4-9. These sections provide instructions for checking your OPTIMAL setting and resetting it if necessary.

Small MULTIPLIER Option Setting

This problem may return an error message similar to the following:

```
ORA-01632: max # extents (%s) reached in index %s.%s
```

The Migration utility is using the default value of 15 for the MULTIPLIER option, and this value is too low. To correct the problem, increase the value of the MULTIPLIER option.

If you are using the Migration utility, when you run it from the command line, enter the following to raise the MULTIPLIER option to 30:

```
mig MULTIPLIER=30
```

If, however, you are running the Migration utility in the background by using the Oracle Data Migration Assistant, restore your Oracle7 backup and then re-run the assistant. Choose the Custom migration option in the assistant. When you are prompted for the MULTIPLIER value, enter a value greater than the default of 15.

See Also: ["Review Migration Utility Command-Line Options"](#) on page 3-14 for more information about the MULTIPLIER option.

Migration Utility Errors

The Migration utility may return error messages and informational messages during migration. This section 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.

If you are using the Oracle Data Migration Assistant, the Migration utility messages are recorded in a log file. See the online help for the Oracle Data Migration Assistant for information about accessing its log files. Also, if you are using the Oracle Data Migration Assistant and the recommended action for a message includes rerunning the Migration utility, you should rerun the assistant.

The following messages are listed in alphabetical order:

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 Oracle7 control file.

Action: Check the Oracle7 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.

client nls_character set does not match server nls_character set - check that NLS_LANG environment variable is set

Cause: The NLS_LANG character set does not match the character set in PROPS\$.

Action: Check the database character set in PROPS\$ and set the NLS_LANG environment variable to match it.

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 Oracle8i 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 Oracle8i 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, of `migrate.bsq`, and of the target Oracle7 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 Oracle7.1 or later

Cause: Current database is an earlier version than Oracle7, release 7.1.

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

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 Oracle7 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 Oracle8i software are compatible, and that no corruption exists in `migrate.bsq`. Fix any problems, and rerun the Migration utility.

entries found in `system.defs_call`, `defs_calldest` or `defs_error` - push all deferred transactions before migration

Cause: Entries exist in `SYSTEM.DEFS_CALL`, `DEF$_CALLDEST`, or `DEF$_ERROR`.

Action: If entries are in `SYSTEM.DEFS_CALL`, push all deferred transactions until `SYSTEM.DEFS_CALL` is empty. If entries are in `SYSTEM.DEFS_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 Oracle8i 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 Oracle8i 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 Oracle8i's minimum block size 2k

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

Action: Make sure the block size of the Oracle7 database is at least 2KB. You may consider rebuilding the Oracle7 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: If you are using the Migration utility, go to Step 4 on page 3-9 and make sure all offline tablespaces have been taken offline cleanly. If you are using the Oracle Data Migration Assistant, go to Step 4 on page 4-7 and make sure all offline tablespaces have been taken offline cleanly. Then, rerun the Oracle8i Migration utility.

mounting database ...

Cause: This is an informational message. The Migration utility is mounting the Oracle7 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 Oracle8i `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 charset 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 Oracle7 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 *Oracle8i 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 Oracle7 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 Oracle7 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 an Oracle7 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 computer 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 Oracle8i Migration utility.

Action: Check that the file exists and that its access permissions allow Oracle to open and read it. If possible, check that the file, and 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 it. If possible, check that the file, and 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.

Problems at the ALTER DATABASE CONVERT Command

You may encounter one of the problems described in this section when you issue the ALTER DATABASE CONVERT command during the migration process after you run the Migration utility. Typically, the conversion will stop and one or more error messages will be displayed. If you encounter one of the following problems when you run the ALTER DATABASE CONVERT command, perform the suggested actions to correct the problem.

Note: These problems should not occur if you are using the Oracle Data Migration Assistant, because the assistant runs the ALTER DATABASE CONVERT command automatically in the background and avoids the conditions that cause these problems. Therefore, these problems only apply if you are using the Migration utility.

Oracle7 Control Files Exist

This problem may return the following error messages:

```
ORA-00200: cannot create control file name
ORA-00202: controlfile: name
ORA-27038: skgfrcre: file exists
```

The old Oracle7 control files must be renamed or removed before you issue the ALTER DATABASE CONVERT command.

See Also: Step 2 on page 3-22 in the "[Migration Steps in the Oracle8i Environment](#)" section.

Database Started in Mode Other Than NOMOUNT

This problem may return the following error messages:

```
ORA-00227: corrupt block detected in controlfile: (block num, # blocks num)
ORA-00202: control file: '%s'
```

The old Oracle7 control files must be renamed or removed before you issue the ALTER DATABASE CONVERT command. Also, the database must be started in NOMOUNT mode when you issue the ALTER DATABASE CONVERT command, but the database was started in a different mode.

See Also: Step 2 on page 3-22 and Step 10 on page 3-26 in the ["Migration Steps in the Oracle8i Environment"](#) section.

Convert File Not Found

This problem may return the following error messages:

```
ORA-00404: convert file not found: name
ORA-27037: unable to obtain file status
```

The convert file (*convsid.dbf* on UNIX and *convert.ora* on Windows NT) generated by the Migration utility was not found in the expected location. On UNIX, the expected location is the `$ORACLE_HOME/dbs` directory in the Oracle8i environment; on Windows NT, the expected location is the `$ORACLE_HOME\database` directory in the Oracle8i environment. The convert file must be moved to this location before you issue the ALTER DATABASE CONVERT command.

See Also: Step 3 on page 3-23 in the ["Migration Steps in the Oracle8i Environment"](#) section.

REMOTE_LOGIN_PASSWORDFILE Initialization Parameter Set to EXCLUSIVE

This problem may return the following error message:

```
ORA-00600: internal error code, arguments: [kzsrsdn: 1], [32]
```

You will encounter this error under the following conditions:

- Your database is using a password file.
- The password file was not moved to the correct directory. On UNIX, the correct directory is `$ORACLE_HOME/dbs` in the Oracle8i environment; on Windows NT, the correct directory is `$ORACLE_HOME\database` in the Oracle8i environment
- The `REMOTE_LOGIN_PASSWORDFILE` initialization parameter is set to `EXCLUSIVE` in the `init sid .ora` file.

To continue with the migration, complete the following steps:

1. Shutdown the database.
2. Set `REMOTE_LOGIN_PASSWORDFILE` to `NONE` in the `init sid .ora` file:

```
REMOTE_LOGIN_PASSWORDFILE = NONE
```

3. Startup mount the database by entering the following SQL command:

```
SVRMGR> STARTUP MOUNT
```

You may need to use the `PFILE` option to specify the location of your `init sid .ora` file.

4. Issue the `ALTER DATABASE OPEN RESETLOGS` command:

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

5. Continue with the migration process starting with Step 13 on page 3-27.

You cannot use the existing password file because it is no longer valid. If you want to use a password file with Oracle8i, recreate the password file and repopulate it with users. Remember to set `REMOTE_LOGIN_PASSWORDFILE` correctly.

Database Name Mismatch

This problem may return the following error message:

```
ORA-01103: database name name in control file is not name
```

There is a mismatch in the database name. This mismatch is in one or more of the following places:

- The database name specified by the DB_NAME parameter in the *init*sid*.ora* file does not match the database name in the *conv*sid*.dbf* filename.
- The Oracle8i instance ID set by the ORACLE_SID environment variable does not match the database name in the *conv*sid*.dbf* filename.

Note: This problem only occurs on UNIX operating systems. It does not apply to Windows NT.

To correct the problem, make sure the correct database name is specified in each of the following places:

- ORACLE_SID environment variable
- DB_NAME parameter in the *init*sid*.ora* file
- the *sid* part of the *conv*sid*.dbf* filename

For example, if your ORACLE_SID environment variable and the DB_NAME parameter in the *init*sid*.ora* file are both set to DB1, the *conv*sid*.dbf* filename should be the following:

```
convDB1.dbf
```

Rerunning the ALTER DATABASE CONVERT Command

This problem may return the following error messages:

```
ORA-01122: datafile name - failed verification check
```

```
ORA-01110: data file name: str
```

```
ORA-01202: wrong incarnation of this file - wrong creation time
```

These errors usually indicate that the ALTER DATABASE CONVERT command was issued previously but failed. If you encounter these errors, you can attempt to move on to the next step in the migration process by issuing the ALTER DATABASE OPEN RESETLOGS command. However, if you encounter problems, restore the backup you created before you started the migration process, and use it to start the migration again from the beginning. Start at the beginning of [Chapter 3](#), but make sure you performed the pre-migration actions described in [Chapter 2](#).

Datafile Version Integrity Problem

This problem may return the following error messages:

```
ORA-01122: datafile name - failed verification check
ORA-01110: data file name: str
ORA-01211: Oracle7 data file is not from migration to Oracle8i
```

The Migration utility must be the last utility to access the database in the Oracle7 environment. The datafile specified in the error messages is either a backup taken before you ran the Migration utility, or the database was opened by Oracle7 after you ran the Migration utility. Only the datafiles that were current when the Migration utility ran can be accessed by Oracle8i.

To ensure datafile version integrity, the system change numbers (SCNs) in the data dictionary, the convert file, and the file headers must all be consistent when the database is converted to Oracle8i. If the database is opened under Oracle7 after the Migration utility has run, the SCN checking fails when you issue the ALTER DATABASE CONVERT command.

To correct the problem, complete the following steps:

1. Shutdown the database.
2. Rename the control files created by ALTER DATABASE CONVERT to different file names.
3. Restore the saved copy of Oracle7 control files from immediately before the issuing of the STARTUP NOMOUNT command.

If you do not have the Oracle7 control files saved, restore the backup you made prior to starting the migration process.

4. Start the Migration process over from the beginning, ensuring the database is not opened in the Oracle7 environment after the Migration utility completes. Start from the beginning of [Chapter 3](#).

Changes to Initialization Parameters

Version 8 supports new initialization parameters for use in the `initsid.ora` file, and some initialization parameters have been renamed or have become obsolete in version 8. This appendix lists the new, renamed, and obsolete parameters in version 8, and this appendix discusses compatibility issues with certain initialization parameters.

This appendix covers the following topics:

- [Initialization Parameters Added in Version 8](#)
- [Initialization Parameters Renamed in Version 8](#)
- [Initialization Parameters Obsolete in Version 8](#)
- [Compatibility Issues with Initialization Parameters](#)

See Also: *Oracle8i Reference* for detailed information about the new and changed initialization parameters listed in this appendix.

Note: Some of the parameters listed in this appendix are operating-system specific. See your operating-system specific Oracle documentation for more information about these initialization parameters.

Initialization Parameters Added in Version 8

The initialization parameters listed in this section are new in version 8.

Initialization Parameters Added in Release 8.0

The following parameters were added in release 8.0:

ALLOW_PARTIAL_SN_RESULTS	ALWAYS_SEMI_JOIN
AQ_TM_PROCESSES	ARCH_IO_SLAVES
BACKUP_DISK_IO_SLAVES	BACKUP_TAPE_IO_SLAVES
BUFFER_POOL_KEEP	BUFFER_POOL_RECYCLE
COMPLEX_VIEW_MERGING	CONTROL_FILE_RECORD_KEEP_TIME
DB_BLOCK_MAX_DIRTY_TARGET	DB_FILE_DIRECT_IO_COUNT
DB_FILE_NAME_CONVERT	DB_WRITER_PROCESSES
DBWR_IO_SLAVES	DISK_ASYNC_IO
FAST_FULL_SCAN_ENABLED	FREEZE_DB_FOR_FAST_INSTANCE_RECOVERY
GC_DEFER_TIME	GC_LATCHES
HI_SHARED_MEMORY_ADDRESS	INSTANCE_GROUPS
LARGE_POOL_MIN_ALLOC	LARGE_POOL_SIZE
LGWR_IO_SLAVES	LM_LOCKS
LM_PROCS	LM_RESS
LOCAL_LISTENER	LOCK_NAME_SPACE
LOCK_SGA	LOCK_SGA_AREAS
LOG_ARCHIVE_DUPLEX_DEST	LOG_ARCHIVE_MIN_SUCCEED_DEST
LOG_FILE_NAME_CONVERT	MTS_RATE_LOG_SIZE
MTS_RATE_SCALE	NLS_CALENDAR
O7_DICTIONARY_ACCESSIBILITY	OBJECT_CACHE_MAX_SIZE_PERCENT
OBJECT_CACHE_OPTIMAL_SIZE	OGMS_HOME
OPEN_LINKS_PER_INSTANCE	OPS_ADMIN_GROUP
OPTIMIZER_FEATURES_ENABLE	OPTIMIZER_INDEX_CACHING
OPTIMIZER_INDEX_COST_ADJ	OPTIMIZER_MAX_PERMUTATIONS
PARALLEL_ADAPTIVE_MULTI_USER	PARALLEL_BROADCAST_ENABLED

PARALLEL_EXECUTION_MESSAGE_SIZE	PARALLEL_INSTANCE_GROUP
PARALLEL_MIN_MESSAGE_POOL	PARALLEL_SERVER
PARALLEL_TRANSACTION_RESOURCE_TIMEOUT	PLSQL_V2_COMPATIBILITY
PUSH_JOIN_PREDICATE	READ_ONLY_OPEN_DELAYED
REPLICATION_DEPENDENCY_TRACKING	SERIAL_REUSE
SESSION_MAX_OPEN_FILES	SHARED_MEMORY_ADDRESS
STAR_TRANSFORMATION_ENABLED	TAPE_ASYNCH_IO
TIMED_OS_STATISTICS	TRANSACTION_AUDITING
USE_INDIRECT_DATA_BUFFERS	

Initialization Parameters Added in Release 8.1

The following initialization parameters were added in release 8.1:

DB_BLOCK_CHECKING	ENT_DOMAIN_NAME
FAST_START_IO_TARGET	FAST_START_PARALLEL_ROLLBACK
HS_AUTOREGISTER	INSTANCE_NAME
JAVA_POOL_SIZE	LOG_ARCHIVE_DEST_1
LOG_ARCHIVE_DEST_STATE_1	LOG_ARCHIVE_MAX_PROCESSES
NLS_COMP	NLS_DUAL_CURRENCY
PARALLEL_AUTOMATIC_TUNING	PARALLEL_SERVER_INSTANCES
PARALLEL_THREADS_PER_CPU	PLSQL_LOAD_WITHOUT_RECOMPILE
QUERY_REWRITE_ENABLED	QUERY_REWRITE_INTEGRITY
RESOURCE_MANAGER_PLAN	SERVICE_NAMES
SORT_MULTIBLOCK_READ_COUNT	STANDBY_ARCHIVE_DEST

Initialization Parameters Renamed in Version 8

The initialization parameters in this section have been renamed in version 8.

Initialization Parameters Renamed in Release 8.0

The following initialization parameters were renamed in release 8.0:

Table B-1 *Initialization Parameters Renamed in Release 8.0*

Pre-Release 8.0 Name	Release 8.0 Name
ASYNCH_READ	DISK_ASYNCH_IO
ASYNCH_WRITE	DISK_ASYNCH_IO
CCF_IO_SIZE *	DB_FILE_DIRECT_IO_COUNT *
DB_FILE_STANDBY_NAME_CONVERT	DB_FILE_NAME_CONVERT
DB_WRITERS	DBWR_IO_SLAVES
LOG_FILE_STANDBY_NAME_CONVERT	LOG_FILE_NAME_CONVERT
SNAPSHOT_REFRESH_INTERVAL	JOB_QUEUE_INTERVAL

* The units are different for CCF_IO_SIZE (bytes) and DB_FILE_DIRECT_IO_COUNT (database blocks).

Initialization Parameters Renamed in Release 8.1.4

The following initialization parameters were renamed in release 8.1.4:

Table B–2 *Initialization Parameters Renamed in Release 8.1.4*

Release 8.1.3 Name	Release 8.1.4 and Higher Name
MVIEW_REWRITE_ENABLED	QUERY_REWRITE_ENABLED
REWRITE_INTEGRITY	QUERY_REWRITE_INTEGRITY

Initialization Parameters Renamed in Release 8.1.5

The following initialization parameters were renamed in release 8.1.5:

Table B–3 *Initialization Parameters Renamed in Release 8.1.5*

Release 8.1.3 and 8.1.4 Name	Release 8.1.5 and Higher Name
NLS_UNION_CURRENCY	NLS_DUAL_CURRENCY
PARALLEL_TRANSACTION_RECOVERY	FAST_START_PARALLEL_ROLLBACK

Initialization Parameters Obsolete in Version 8

The initialization parameters in this section are obsolete in version 8.

Initialization Parameters Obsolete in Release 8.0

The following initialization parameters became obsolete in release 8.0 and cannot be used in release 8.0 and higher:

CHECKPOINT_PROCESS	FAST_CACHE_FLUSH
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
MLS_LABEL_FORMAT	OPTIMIZER_PARALLEL_PASS
PARALLEL_DEFAULT_MAX_SCANS	PARALLEL_DEFAULT_SCAN_SIZE
POST_WAIT_DEVICE	SEQUENCE_CACHE_HASH_BUCKETS
UNLIMITED_ROLLBACK_SEGMENTS	USE_IPQ
USE_POST_WAIT_DRIVER	USE_READV
USE_SIGIO	V733_PLANS_ENABLED

Note: An attempt to start any release 8.0 or higher database using one or more of the these obsolete initialization parameters will result in an error, and the database will not start.

Initialization Parameters Obsolete in Release 8.1

A relatively large number of initialization parameters were obsoleted in release 8.1 to simplify database administration. The following initialization parameters became obsolete in release 8.1 and cannot be used in release 8.1 and higher:

ALLOW_PARTIAL_SN_RESULTS	ARCH_IO_SLAVES
B_TREE_BITMAP_PLANS	BACKUP_DISK_IO_SLAVES
CACHE_SIZE_THRESHOLD	CLEANUP_ROLLBACK_ENTRIES
CLOSE_CACHED_OPEN_CURSORS	COMPATIBLE_NO_RECOVERY
COMPLEX_VIEW_MERGING	DB_BLOCK_CHECKPOINT_BATCH
DB_BLOCK_LRU_EXTENDED_STATISTICS	DB_BLOCK_LRU_STATISTICS
DB_FILE_SIMULTANEOUS_WRITES	DELAYED_LOGGING_BLOCK_CLEANOUTS
DISCRETE_TRANSACTIONS_ENABLED	DISTRIBUTED_LOCK_TIMEOUT
DISTRIBUTED_RECOVERY_CONNECTION_HOLD_TIME	FAST_FULL_SCAN_ENABLED
FREEZE_DB_FOR_FAST_INSTANCE_RECOVERY	GC_LATCHES
GC_LCK_PROCS	JOB_QUEUE_KEEP_CONNECTIONS
LARGE_POOL_MIN_ALLOC	LGWR_IO_SLAVES
LOCK_SGA_AREAS	LOG_ARCHIVE_BUFFER_SIZE
LOG_ARCHIVE_BUFFERS	LOG_BLOCK_CHECKSUM
LOG_FILES	LOG_SIMULTANEOUS_COPIES
LOG_SMALL_ENTRY_MAX_SIZE	MAX_TRANSACTION_BRANCHES
MTS_LISTENER_ADDRESS	MTS_MULTIPLE_LISTENERS
MTS_RATE_LOG_SIZE	MTS_RATE_SCALE
MTS_SERVICE	OGMS_HOME
OPS_ADMIN_GROUP	PARALLEL_DEFAULT_MAX_INSTANCES
PARALLEL_MIN_MESSAGE_POOL	PARALLEL_SERVER_IDLE_TIME
PARALLEL_TRANSACTION_RESOURCE_TIMEOUT	PUSH_JOIN_PREDICATE
REDUCE_ALARM	ROW_CACHE_CURSORS
SEQUENCE_CACHE_ENTRIES	SEQUENCE_CACHE_HASH_BUCKETS
SHARED_POOL_RESERVED_MIN_ALLOC	SNAPSHOT_REFRESH_KEEP_CONNECTIONS
SNAPSHOT_REFRESH_PROCESSES	SORT_DIRECT_WRITES
SORT_READ_FAC	SORT_SPACEMAP_SIZE
SORT_WRITE_BUFFER_SIZE	SORT_WRITE_BUFFERS
SPIN_COUNT	TEMPORARY_TABLE_LOCKS
TEXT_ENABLE	USE_ISM

Note: An attempt to start a release 8.1 database using one or more of these obsolete initialization parameters will succeed, but a warning will be returned and recorded in the alert log.

Compatibility Issues with Initialization Parameters

The lists of new, changed, and obsolete initialization parameters earlier in this chapter show differences in initialization parameters across different releases of Oracle. However, certain initialization parameter changes require special attention because they may raise compatibility issues for your database. These parameter changes are described in this section.

New Default Value for LOG_CHECKPOINT_TIMEOUT

Starting in release 8.1.5, the LOG_CHECKPOINT_TIMEOUT initialization parameter has a new default value. In previous releases, the default value was zero seconds, but in release 8.1.5 and higher, the default value is 1800 seconds. See *Oracle8i Reference* for more information.

Data Dictionary Protection

O7_DICTIONARY_ACCESSIBILITY is the initialization parameter switch that continues Oracle7 data dictionary behavior. It is only a temporary expedient and is not planned for future Oracle8i releases.

The DML_LOCKS Parameter

Oracle8i systems typically consume more DML locks while performing DDL operations than are required for Oracle7 systems. Nevertheless, the Oracle7 DML_LOCKS parameter default settings usually are adequate for Oracle8i 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 Oracle8i fewer transactions are used per rollback segment than are used in Oracle7. Consequently, DML_LOCKS has a lower default value in Oracle8i. Under some extreme test conditions, an Oracle8i system exceeded its (Oracle7) 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 operating-system specific settings. An informational message about this change may be displayed during database startup operations.

The `DB_DOMAIN` Parameter

Beginning with release 8.1, if the `DB_DOMAIN` initialization parameter is unset, it is set to `NULL` by default. In prior releases of Oracle, the default setting was the following:

```
.WORLD
```

A `NULL` setting for `DB_DOMAIN` may cause database connection problems in some environments. Before you migrate or upgrade to release 8.1, make sure the `DB_DOMAIN` parameter in your `init sid .ora` file is set to one of the following:

- `.WORLD`
- a valid domain setting for your environment

If `DB_DOMAIN` is not set in your current database, set it to `.WORLD` before you migrate or upgrade to release 8.1.

If `DB_DOMAIN` is set to a valid domain for your environment in your current database, retain the setting in your `init sid .ora` file when you migrate or upgrade to release 8.1.

Parallel Execution Allocated from Large Pool

Starting with release 8.1, parallel execution message buffers are allocated from the large pool whenever `PARALLEL_AUTOMATIC_TUNING`, a new initialization parameter, is set to `TRUE`. In past releases, this allocation was from the shared pool. If you are migrating or upgrading to release 8.1 and you choose to set `PARALLEL_AUTOMATIC_TUNING` to `TRUE`, you can avoid problems by modifying the settings for the following initialization parameters:

- `SHARED_POOL_SIZE`
- `LARGE_POOL_SIZE`

Typically, you should reduce the setting for `SHARED_POOL_SIZE` and raise the setting for `LARGE_POOL_SIZE` to avoid problems. Alternatively, you can reduce the setting for `SHARED_POOL_SIZE` and let Oracle calculate the setting for `LARGE_POOL_SIZE`. Oracle calculates a default `LARGE_POOL_SIZE` only if `PARALLEL_AUTOMATIC_TUNING` is set to `TRUE` and `LARGE_POOL_SIZE` is unset.

The calculation is based on the settings for the following initialization parameters:

- `PARALLEL_MAX_SERVERS`
- `PARALLEL_THREADS_PER_CPU`
- `PARALLEL_SERVER_INSTANCES`
- `MTS_DISPATCHERS`
- `DBWR_IO_SLAVES`

If `PARALLEL_AUTOMATIC_TUNING` is unset or set to `FALSE`, and if `LARGE_POOL_SIZE` is unset, the value for `LARGE_POOL_SIZE` defaults to zero.

Note: When `PARALLEL_AUTOMATIC_TUNING` is set to `TRUE`, the new behavior applies even if your `COMPATIBLE` parameter is set below 8.1.0.

See Also: *Oracle8i Reference* and *Oracle8i Tuning* for more information about other effects of the `PARALLEL_AUTOMATIC_TUNING` initialization parameter.

The following scenarios illustrate the behavior that results from various initialization parameter settings when you migrate or upgrade to release 8.1.

Retaining Parameter Settings without Modifications

You do not alter the parameters from their previous settings:

Table B-4 Retaining Parameter Settings without Modifications

Parameter	Setting
PARALLEL_AUTOMATIC_TUNING	Unset (defaults to FALSE).
SHARED_POOL_SIZE	Set to a large value, including the space required for parallel execution.
LARGE_POOL_SIZE	Unset (defaults to zero).

These settings are the most common scenario. In this case, you already have accounted for the space required for parallel execution in the shared pool.

Using PARALLEL_AUTOMATIC_TUNING

You alter the parameters from their previous settings to the following settings:

Table B-5 Using PARALLEL_AUTOMATIC_TUNING

Parameter	Setting
PARALLEL_AUTOMATIC_TUNING	Set to TRUE.
SHARED_POOL_SIZE	Set to a small value that accounts for all clients except parallel execution.
LARGE_POOL_SIZE	Unset (defaults to a large value that includes the space required for parallel execution).

In this case, parallel execution allocates buffers from the large pool based on Oracle's automatic calculation. Buffer allocation is more efficient, and failures to allocate are isolated from the clients of the shared pool.

Using PARALLEL_AUTOMATIC_TUNING and Setting LARGE_POOL_SIZE

You alter the parameters from their previous settings to the following settings:

Table B-6 *Using PARALLEL_AUTOMATIC_TUNING and Setting LARGE_POOL*

Parameter	Setting
PARALLEL_AUTOMATIC_TUNING	Set to TRUE.
SHARED_POOL_SIZE	Set to a small value that accounts for all clients except parallel execution.
LARGE_POOL_SIZE	Set to a value that includes the space required for parallel execution.

In this case, parallel execution allocates buffers from the large pool. After initial testing with LARGE_POOL_SIZE unset, you determined that the default calculation for LARGE_POOL_SIZE did not reflect your requirements for the large pool. Therefore, you decided to set LARGE_POOL_SIZE manually. After you set LARGE_POOL_SIZE properly, buffer allocation is more efficient, and failures to allocate are isolated from the clients of the shared pool.

Using PARALLEL_AUTOMATIC_TUNING without Modifying SHARED_POOL_SIZE

You alter the parameters from their previous settings to the following settings:

Table B-7 *Using PARALLEL_AUTOMATIC_TUNING without Modifying SHARED_POOL_SIZE*

Parameter	Setting
PARALLEL_AUTOMATIC_TUNING	Set to TRUE.
SHARED_POOL_SIZE	Set to a large value, including the space required for parallel execution.
LARGE_POOL_SIZE	Unset (defaults to a large value that includes the space required for parallel execution).

In this case, parallel execution allocates buffers from the large pool, but because you did not modify SHARED_POOL_SIZE, it is likely that the SGA will be unnecessarily large, causing performance problems. Therefore, avoid setting PARALLEL_AUTOMATIC_TUNING to TRUE without modifying the settings of SHARED_POOL_SIZE and LARGE_POOL_SIZE appropriately.

Archive Log Destination Parameters

Release 8.1 supports new archive log destination parameters. After you migrate or upgrade to release 8.1, you can dynamically convert from the old pre-release 8.1 parameters (LOG_ARCHIVE_DEST and LOG_ARCHIVE_DUPLEX_DEST) to the new release 8.1 parameters (LOG_ARCHIVE_DEST_1 and LOG_ARCHIVE_DEST_STATE_1). You also can dynamically revert to the old parameters.

Changing to the New Archive Log Destination Parameters

After you determine the new archive destinations, associated states, and options, complete the following steps to change from the old archive log destination parameters to the new ones:

1. Use ALTER SYSTEM to set LOG_ARCHIVE_MIN_SUCCEED_DEST to 1.
2. Use ALTER SYSTEM to set LOG_ARCHIVE_DUPLEX_DEST to NULL.
3. Use ALTER SYSTEM to set LOG_ARCHIVE_DEST to NULL.
4. Use ALTER SYSTEM to set any LOG_ARCHIVE_DEST_STATE_n parameters to "defer" or "enable" as required. Although enable is the default, Oracle Corporation recommends that you set a state for each destination explicitly.
5. Use ALTER SYSTEM to set at least one LOG_ARCHIVE_DEST_n parameter to a value specifying a local destination.
6. Use ALTER SYSTEM to set other LOG_ARCHIVE_DEST_n parameters as required.
7. Use ALTER SYSTEM to set LOG_ARCHIVE_MIN_SUCCEED_DEST to the required value.

For example, assume there are the following two destinations:

- /oracle/dbs/arclog
- /backup/dbs/arclog

Both destinations are mandatory (minimum succeed destination count is 2). The new destinations are the following:

- /oracle/dbs/arclog (local)
- stndby1 (a standby database)
- /backup/dbs/arclog
- /backup2/dbs/arclog

The first destination, the standby destination, and either of the backup destinations are mandatory (minimum succeed destination count is 3).

With these assumptions, issue the following SQL statements to change your old archive log destination parameters to the new ones:

```
ALTER SYSTEM SET LOG_ARCHIVE_MIN_SUCCEED_DEST = 1;
```

```
ALTER SYSTEM SET LOG_ARCHIVE_DUPLEX_DEST = ' ';
```

```
ALTER SYSTEM SET LOG_ARCHIVE_DEST = ' ';
```

```
ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_1 = 'enable';
```

```
ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_2 = 'enable';
```

```
ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_3 = 'enable';
```

```
ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_4 = 'enable';
```

```
ALTER SYSTEM SET LOG_ARCHIVE_DEST_1 = 'LOCATION=/oracle/dbs/arclog MANDATORY';
```

```
ALTER SYSTEM SET LOG_ARCHIVE_DEST_2 = 'SERVICE=standby1 MANDATORY';
```

```
ALTER SYSTEM SET LOG_ARCHIVE_DEST_3 = 'LOCATION=/backup/dbs/arclog OPTIONAL';
```

```
ALTER SYSTEM SET LOG_ARCHIVE_DEST_4 = 'LOCATION=/backup2/dbs/arclog OPTIONAL';
```

```
ALTER SYSTEM SET LOG_ARCHIVE_MIN_SUCCEED_DEST = 3;
```

Changing Back to the Old Archive Log Destination Parameters

Complete the following steps to change back to the old archive log destination parameters:

1. Use ALTER SYSTEM to set LOG_ARCHIVE_MIN_SUCCEED_DEST to 1.
2. Use ALTER SYSTEM to set all LOG_ARCHIVE_DEST_n parameters to NULL.
3. Use ALTER SYSTEM to set the LOG_ARCHIVE_DEST parameter to a value specifying a local destination.
4. Use ALTER SYSTEM to set the LOG_ARCHIVE_DUPLEX_DEST parameter as required.
5. Use ALTER SYSTEM to set LOG_ARCHIVE_MIN_SUCCEED_DEST to the required value.

For example, assume there are the following two destinations:

- /oracle/dbs/arclog (LOG_ARCHIVE_DEST_1)
- /backup/dbs/arclog (LOG_ARCHIVE_DEST_4)

Both destinations are mandatory. The new destinations and minimum succeed count are the same.

With these assumptions, issue the following SQL statements to change your new archive log destination parameters to the old ones:

```
ALTER SYSTEM SET LOG_ARCHIVE_MIN_SUCCEED_DEST = 1;
```

```
ALTER SYSTEM SET LOG_ARCHIVE_DEST_4 = ' ';
```

```
ALTER SYSTEM SET LOG_ARCHIVE_DEST_1 = ' ';
```

```
ALTER SYSTEM SET LOG_ARCHIVE_DEST = '/oracle/dbs/arclog';
```

```
ALTER SYSTEM SET LOG_ARCHIVE_DUPLEX_DEST = '/backup/dbs/arclog';
```

```
ALTER SYSTEM SET LOG_ARCHIVE_MIN_SUCCEED_DEST = 2;
```

Note: If you plan to downgrade from release 8.1 to release 8.0, you should change back to the old archive log destination parameters by replacing the new parameters with the old ones in your `init sid .ora` file. Do not use the dynamic method if you are downgrading.

Possible Errors During the Transition in Parameters

When you follow the procedures described previously in this section for changing your archive destination parameters, you may encounter the following error messages in your log files if archiving is enabled:

- In the Alert log - "Archiving not possible: No available destinations"
- In the Trace log - "ARCH: INCOMPLETE, no available destinations"

You will not encounter these errors if archiving is disabled. The errors may occur during the procedure when there are no valid archive destinations. However, when the transition in parameters is complete, the errors should cease. You *should not* disable archiving during the transition to avoid these errors.

Changes to Static Data Dictionary Views

Version 8 supports new static data dictionary views, and some static data dictionary views have been changed or have become obsolete in version 8. This appendix lists the new, changed, and obsolete static data dictionary views in version 8.

This appendix covers the following topics:

- [Static Data Dictionary Views Added in Version 8](#)
- [Static Data Dictionary Views with Added Columns in Version 8](#)
- [Static Data Dictionary Views with Dropped Columns in Version 8](#)
- [Static Data Dictionary Views with Renamed Columns in Version 8](#)
- [Static Data Dictionary Views with Columns That May Return Nulls](#)
- [Static Data Dictionary Views Obsolete in Version 8](#)

See Also: *Oracle8i Reference* for descriptions of the new and changed views listed in this appendix, and for descriptions of the columns in these views.

Static Data Dictionary Views Added in Version 8

The static data dictionary views listed in this section are new in version 8.

Static Data Dictionary Views Added in Release 8.0

The following static data dictionary views were added in release 8.0:

ALL_ALL_TABLES	ALL_COLL_TYPES
ALL_DIRECTORIES	ALL_IND_PARTITIONS
ALL_LIBRARIES	ALL_LOBS
ALL_METHOD_PARAMS	ALL_METHOD_RESULTS
ALL_NESTED_TABLES	ALL_OBJECT_TABLES
ALL_OBJECTS	ALL_PART_COL_STATISTICS
ALL_PART_HISTOGRAMS	ALL_PART_INDEXES
ALL_PART_KEY_COLUMNS	ALL_PART_TABLES
ALL_PROBE_OBJECTS	ALL_REFS
ALL_REGISTERED_SNAPSHOTS	ALL_REPCOLUMN
ALL_REPGENOBJECTS	ALL_SNAPSHOT_LOGS
ALL_SNAPSHOT_REFRESH_TIMES	ALL_TAB_COL_STATISTICS
ALL_TAB_HISTOGRAMS	ALL_TAB_PARTITIONS
ALL_TYPE_ATTRS	ALL_TYPE_METHODS
ALL_TYPES	DBA_ALL_TABLES
DBA_COLL_TYPES	DBA_DIRECTORIES
DBA_IND_PARTITIONS	DBA_LIBRARIES
DBA_LOBS	DBA_METHOD_PARAMS
DBA_METHOD_RESULTS	DBA_NESTED_TABLES
DBA_OBJECT_TABLES	DBA_PART_COL_STATISTICS
DBA_PART_HISTOGRAMS	DBA_PART_INDEXES
DBA_PART_KEY_COLUMNS	DBA_PART_TABLES
DBA_PENDING_TRANSACTIONS	DBA_QUEUE_SCHEDULES
DBA_QUEUE_TABLES	DBA_QUEUEES
DBA_REFS	DBA_REGISTERED_SNAPSHOT_GROUPS

DBA_REGISTERED_SNAPSHOTS	DBA_REPCOLUMN
DBA_REPGENOBJECTS	DBA_SNAPSHOT_LOG_FILTER_COLS
DBA_SNAPSHOT_REFRESH_TIMES	DBA_TAB_COL_STATISTICS
DBA_TAB_HISTOGRAMS	DBA_TAB_PARTITIONS
DBA_TYPE_ATTRS	DBA_TYPE_METHODS
DBA_TYPES	DEFLOB
DEFPROPAGATOR	FILEXT\$
HS_ALL_CAPS	HS_ALL_DD
HS_ALL_INITS	HS_BASE_CAPS
HS_BASE_DD	HS_CLASS_CAPS
HS_CLASS_DD	HS_CLASS_INIT
HS_EXTERNAL_OBJECT_PRIVILEGES	HS_EXTERNAL_OBJECTS
HS_EXTERNAL_USER_PRIVILEGES	HS_FDS_CLASS
HS_FDS_INST	HS_INST_CAPS
HS_INST_DD	HS_INST_INIT
TRUSTED_SERVERS	TS_PITR_CHECK
TS_PITR_OBJECTS_TO_BE_DROPPED	USER_ALL_TABLES
USER_COLL_TYPES	USER_IND_PARTITIONS
USER_LIBRARIES	USER_LOBS
USER_METHOD_PARAMS	USER_METHOD_RESULTS
USER_NESTED_TABLES	USER_OBJECT_TABLES
USER_PART_COL_STATISTICS	USER_PART_HISTOGRAMS
USER_PART_INDEXES	USER_PART_KEY_COLUMNS
USER_PART_TABLES	USER_PASSWORD_LIMITS
USER_QUEUE_TABLES	USER_QUEUES
USER_REFS	USER_REGISTERED_SNAPSHOTS
USER_REPCOLUMN	USER_REPGENOBJECTS
USER_SNAPSHOT_REFRESH_TIMES	USER_TAB_COL_STATISTICS
USER_TAB_HISTOGRAMS	USER_TAB_PARTITIONS
USER_TYPE_ATTRS	USER_TYPE_METHODS
USER_TYPES	

Static Data Dictionary Views Added in Release 8.1

The following static data dictionary views were added in release 8.1:

ALL_ASSOCIATIONS	ALL_CONTEXT
ALL_DIM_ATTRIBUTES	ALL_DIM_CHILD_OF
ALL_DIM_HIERARCHIES	ALL_DIM_JOIN_KEY
ALL_DIM_LEVEL_KEY	ALL_DIM_LEVELS
ALL_DIMENSIONS	ALL_IND_EXPRESSIONS
ALL_IND_SUBPARTITIONS	ALL_INDEXTYPE_OPERATORS
ALL_INDEXTYPES	ALL_INTERNAL_TRIGGERS
ALL_LOB_PARTITIONS	ALL_LOB_SUBPARTITIONS
ALL_MVIEW_AGGREGATES	ALL_MVIEW_ANALYSIS
ALL_MVIEW_DETAIL_RELATIONS	ALL_MVIEW_JOINS
ALL_MVIEW_KEYS	ALL_OPANCILLARY
ALL_OPARGUMENTS	ALL_OPBINDINGS
ALL_OPERATORS	ALL_PART_LOBS
ALL_PARTIAL_DROP_TABS	ALL_POLICIES
ALL_QUEUE_TABLES	ALL_QUEUES
ALL_REFRESH_DEPENDENCIES	ALL_REPCAT_REFRESH_TEMPLATES
ALL_REPCAT_TEMPLATE_OBJECTS	ALL_REPCAT_TEMPLATE_PARMS
ALL_REPCAT_TEMPLATE_SITES	ALL_REPCAT_USER_AUTHORIZATIONS
ALL_REPCAT_USER_PARM_VALUES	ALL_SUBPART_COL_STATISTICS
ALL_SUBPART_HISTOGRAMS	ALL_SUBPART_KEY_COLUMNS
ALL_SUMDELTA	ALL_SUMMARIES
ALL_SUMMARY_AGGREGATES	ALL_SUMMARY_DETAIL_TABLES
ALL_SUMMARY_JOINS	ALL_SUMMARY_KEYS
ALL_TAB_MODIFICATIONS	ALL_TAB_SUBPARTITIONS
ALL_UNUSED_COL_TABS	ALL_USTATS
ALL_VARRAYS	DATABASE_COMPATIBLE_LEVEL
DBA_ASSOCIATIONS	DBA_CONTEXT
DBA_DIM_ATTRIBUTES	DBA_DIM_CHILD_OF

DBA_DIM_HIERARCHIES	DBA_DIM_JOIN_KEY
DBA_DIM_LEVEL_KEY	DBA_DIM_LEVELS
DBA_DIMENSIONS	DBA_IND_EXPRESSIONS
DBA_IND_SUBPARTITIONS	DBA_INDEXTYPE_OPERATORS
DBA_INDEXTYPES	DBA_INTERNAL_TRIGGERS
DBA_LOB_PARTITIONS	DBA_LOB_SUBPARTITIONS
DBA_MVIEW_AGGREGATES	DBA_MVIEW_ANALYSIS
DBA_MVIEW_DETAIL_RELATIONS	DBA_MVIEW_JOINS
DBA_MVIEW_KEYS	DBA_OPANCILLARY
DBA_OPARGUMENTS	DBA_OPBINDINGS
DBA_OPERATORS	DBA_OUTLINE_HINTS
DBA_OUTLINES	DBA_PART_LOBS
DBA_PARTIAL_DROP_TABS	DBA_POLICIES
DBA_REPCAT_REFRESH_TEMPLATES	DBA_REPCAT_TEMPLATE_OBJECTS
DBA_REPCAT_TEMPLATE_PARS	DBA_REPCAT_TEMPLATE_SITES
DBA_REPCAT_USER_AUTHORIZATIONS	DBA_REPCAT_USER_PARM_VALUES
DBA_RSRC_CONSUMER_GROUP_PRIVS	DBA_RSRC_CONSUMER_GROUPS
DBA_RSRC_MANAGER_SYSTEM_PRIVS	DBA_RSRC_PLAN_DIRECTIVES
DBA_RSRC_PLANS	DBA_RULESETS
DBA_SUBPART_COL_STATISTICS	DBA_SUBPART_HISTOGRAMS
DBA_SUBPART_KEY_COLUMNS	DBA_SUMMARIES
DBA_SUMMARY_DETAIL_TABLES	DBA_TAB_MODIFICATIONS
DBA_TAB_SUBPARTITIONS	DBA_TEMP_FILES
DBA_UNUSED_COL_TABS	DBA_USTATS
DBA_VARRAYS	PLUGGABLE_SET_CHECK
PROXY_USERS	QUEUE_PRIVILEGES
SESSION_CONTEXT	STRADDLING_TS_OBJECTS
UNI_PLUGGABLE_SET_CHECK	USER_ASSOCIATIONS
USER_DIM_ATTRIBUTES	USER_DIM_CHILD_OF
USER_DIM_HIERARCHIES	USER_DIM_JOIN_KEY
USER_DIM_LEVEL_KEY	USER_DIM_LEVELS

USER_DIMENSIONS	USER_IND_EXPRESSIONS
USER_IND_SUBPARTITIONS	USER_INDEXTYPE_OPERATORS
USER_INDEXTYPES	USER_INTERNAL_TRIGGERS
USER_LOB_PARTITIONS	USER_LOB_SUBPARTITIONS
USER_MVIEW_AGGREGATES	USER_MVIEW_ANALYSIS
USER_MVIEW_DETAIL_RELATIONS	USER_MVIEW_JOINS
USER_MVIEW_KEYS	USER_OPANCILLARY
USER_OPARGUMENTS	USER_OPBINDINGS
USER_OPERATORS	USER_OUTLINE_HINTS
USER_OUTLINES	USER_PART_LOBS
USER_PARTIAL_DROP_TABS	USER_POLICIES
USER_QUEUE_SCHEDULES	USER_REPCAT_REFRESH_TEMPLATES
USER_REPCAT_TEMPLATE_OBJECTS	USER_REPCAT_TEMPLATE_PARMs
USER_REPCAT_TEMPLATE_SITES	USER_REPCAT_USER_AUTHORIZATION
USER_REPCAT_USER_PARM_VALUES	USER_RSRC_CONSUMER_GROUP_PRIVS
USER_RSRC_MANAGER_SYSTEM_PRIVS	USER_RULESETS
USER_SUBPART_COL_STATISTICS	USER_SUBPART_HISTOGRAMS
USER_SUBPART_KEY_COLUMNS	USER_SUMMARIES
USER_SUMMARY_DETAIL_TABLES	USER_TAB_MODIFICATIONS
USER_TAB_SUBPARTITIONS	USER_UNUSED_COL_TABS
USER_USTATS	USER_VARRAYS

Static Data Dictionary Views with Added Columns in Version 8

New columns were added to the static data dictionary views listed in the following sections.

See Also: *Oracle8i Reference* for detailed information about these views.

Static Data Dictionary Views with Added Columns in Release 8.0

New columns were added to the following static data dictionary views in release 8.0:

ALL_ARGUMENTS	ALL_CLUSTERS
ALL_CONSTRAINTS	ALL_DEPENDENCIES
ALL_INDEXES	ALL_OBJECTS
ALL_REFRESH	ALL_REFRESH_CHILDREN
ALL_REPOBJECT	ALL_REPPRIORITY
ALL_REPSHEMA	ALL_REPSITES
ALL_SNAPSHOTS	ALL_TABLES
ALL_TAB_COLUMNS	ALL_UPDATABLE_COLUMNS
ALL_VIEWS	COL
DBA_CLUSTERS	DBA_CONSTRAINTS
DBA_DATA_FILES	DBA_DEPENDENCIES
DBA_EXTENTS	DBA_FREE_SPACE
DBA_INDEXES	DBA_OBJECTS
DBA_OBJ_AUDIT_OPTS	DBA_PROFILES
DBA_REFRESH	DBA_REFRESH_CHILDREN
DBA_REPOBJECT	DBA_REPPRIORITY
DBA_RGROUP	DBA_ROLLBACK_SEGS
DBA_SEGMENTS	DBA_SNAPSHOTS
DBA_SNAPSHOT_LOGS	DBA_TABLES
DBA_TABLESPACES	DBA_TAB_COLUMNS
DBA_UPDATABLE_COLUMNS	DBA_USERS
DBA_VIEWS	DEFERROR

DEFTRANDEST	FILE_PING
INDEX_STATS	SYS_OBJECTS
USER_ARGUMENTS	USER_CLUSTERS
USER_CONSTRAINTS	USER_DEPENDENCIES
USER_EXTENTS	USER_FREE_SPACE
USER_INDEXES	USER_OBJECTS
USER_OBJ_AUDIT_OPTS	USER_REFRESH
USER_REFRESH_CHILDREN	USER_REPOBJECT
USER_REPPRIORITY	USER_REPSHEMA
USER_REPSITES	USER_SEGMENTS
USER_SNAPSHOTS	USER_SNAPSHOT_LOGS
USER_TAB_COLUMNS	USER_UPDATABLE_COLUMNS
USER_TABLES	USER_TABLESPACES
USER_USERS	USER_VIEWS

Static Data Dictionary Views with Added Columns in Release 8.1

New columns were added to the following static data dictionary views in release 8.1:

ALL_ALL_TABLES	ALL_CLUSTERS
ALL_COLL_TYPES	ALL_CONSTRAINTS
ALL_IND_PARTITIONS	ALL_NESTED_TABLES
ALL_INDEXES	ALL_IND_COLUMNS
ALL_IND_PARTITIONS	ALL_IND_SUBPARTITIONS
ALL_MVIEW_ANALYSIS	ALL_OBJECTS
ALL_OBJECT_TABLES	ALL_PART_COL_STATISTICS
ALL_PART_INDEXES	ALL_PART_TABLES
ALL_PROBE_OBJECTS	ALL_REFS
ALL_REPCAT	ALL_REPCAT_TEMPLATE_SITES
ALL_REPCOLUMN	ALL_REPGROUP
ALL_REPOBJECT	ALL_SNAPSHOTS
ALL_TAB_COL_STATISTICS	ALL_TAB_PARTITIONS
ALL_TABLES	ALL_TAB_COLUMNS
ALL_TRIGGERS	DBA_ALL_TABLES
DBA_CLUSTERS	DBA_COLL_TYPES
DBA_CONSTRAINTS	DBA_CONTEXT
DBA_DATA_FILES	DBA_IND_PARTITIONS
DBA_JOBS	DBA_INDEXES
DBA_IND_COLUMNS	DBA_IND_PARTITIONS
DBA_IND_SUBPARTITIONS	DBA_JOBS_RUNNING
DBA_MVIEW_ANALYSIS	DBA_NESTED_TABLES
DBA_OBJECTS	DBA_OBJECT_TABLES
DBA_OUTLINE_HINTS	DBA_PART_COL_STATISTICS
DBA_PART_INDEXES	DBA_PART_TABLES
DBA_PRIV_AUDIT_OPTS	DBA_QUEUE_SCHEDULES
DBA_QUEUE_TABLES	DBA_REFS

DBA_REGISTERED_SNAPSHOT_GROUPS	DBA_REPCAT
DBA_REPCAT_TEMPLATE_SITES	DBA_REPCOLUMN
DBA_REPGROUP	DBA_REPOBJECT
DBA_SNAPSHOTS	DBA_STMT_AUDIT_OPTS
DBA_TAB_COLUMNS	DBA_TAB_COL_STATISTICS
DBA_TAB_PARTITIONS	DBA_TRIGGERS
DBA_TABLES	DBA_TABLESPACES
DBA_USERS	INDEX_STATS
REPCAT_REPCAT	REPCAT_REPOBJECTS
USER_ALL_TABLES	USER_CLUSTERS
USER_COLL_TYPES	USER_CONSTRAINTS
USER_IND_PARTITIONS	USER_IND_SUBPARTITIONS
USER_JOBS	USER_INDEXES
USER_IND_COLUMNS	USER_MVIEW_ANALYSIS
USER_NESTED_TABLES	USER_OBJECTS
USER_OBJECT_TABLES	USER_OUTLINE_HINTS
USER_PART_COL_STATISTICS	USER_PART_INDEXES
USER_PART_TABLES	USER_QUEUE_TABLES
USER_REFS	USER_REPCAT
USER_REPCAT_TEMPLATE_SITES	USER_REPCOLUMN
USER_REPGROUP	USER_REPOBJECT
USER_SNAPSHOTS	USER_TABLES
USER_TAB_COL_STATISTICS	USER_TAB_PARTITIONS
USER_TABLESPACES	USER_TAB_COLUMNS
USER_TRIGGERS	USER_USERS

Static Data Dictionary Views with Dropped Columns in Version 8

The columns listed in the following sections were dropped in version 8. If an application requires one or more of the columns listed below, modify the application accordingly.

Static Data Dictionary Views with Dropped Columns in Release 8.0

The columns listed [Table C-1](#) were dropped in release 8.0.

Table C-1 *Static Data Dictionary View with Dropped Columns in Release 8.0*

Static Data Dictionary Views	Dropped Columns
DEFCALLDEST	DEFERRED_TRAN_DB
DEFERROR	DEFERRED_TRAN_DB ERROR_TIME
DEFTRAN	COMMIT_COMMENT DEFERRED_TRAN_DB DESTINATION_LIST ORIGIN_TRAN_DB ORIGIN_TRAN_ID ORIGIN_USER
DEFTRANDEST	DEFERRED_TRAN_DB

Static Data Dictionary Views with Dropped Columns in Release 8.1

The columns listed [Table C-2](#) were dropped in release 8.1.

Table C-2 *Static Data Dictionary Views with Dropped Columns in Release 8.1*

Static Data Dictionary Views	Dropped Columns
DBA_AUDIT_OBJECT USER_AUDIT_OBJECT	OBJECT_LABEL SESSION_LABEL
DBA_AUDIT_SESSION USER_AUDIT_SESSION	SESSION_LABEL
DBA_AUDIT_STATEMENT USER_AUDIT_STATEMENT	SESSION_LABEL
DBA_AUDIT_TRAIL USER_AUDIT_TRAIL	OBJECT_LABEL SESSION_LABEL
DBA_CONTEXT	ATTRIBUTE
ALL_IND_COLUMNS DBA_IND_COLUMNS USER_IND_COLUMNS	COLUMN_EXPRESSION
ALL_JOBS DBA_JOBS USER_JOBS	CLEARANCE_HI CLEARANCE_LO CURRENT_SESSION_LABEL
ALL_REFS DBA_REFS USER_REFS	HAS_REFERENTIAL_CONS REFERENTIAL_CONS_NAME

Static Data Dictionary Views with Renamed Columns in Version 8

The columns listed in the following sections were renamed in version 8. If an application requires one or more of the columns listed below, modify the application accordingly.

Static Data Dictionary Views with Renamed Columns in Release 8.0

The columns listed [Table C-3](#) were renamed in release 8.0.

Table C-3 *Static Data Dictionary Views with Renamed Columns in Release 8.0*

Static Data Dictionary View	Oracle7 Column Name	Release 8.0 Column Name
DBA_RCHILD	TYPE	TYPE#
DEFSCHEDULE	LAST_ERROR	LAST_ERROR_NUMBER
	LAST_MSG	LAST_ERROR_MESSAGE

Note: There are no static data dictionary views with renamed columns in release 8.1.

Static Data Dictionary Views with Columns That May Return Nulls

Starting with release 8.1, the columns in the static data dictionary views listed in [Table C-4](#) may return nulls; in previous releases, these columns could not return nulls. If an application requires non-null values for one or more of the columns listed below, modify the application accordingly.

Table C-4 Columns That May Return Nulls in Release 8.1

Static Data Dictionary Views	Columns	Explanation
DBA_DATA_FILES	AUTOEXTENSIBLE BLOCKS BYTES INCREMENT_BY MAXBLOCKS MAXBYTES	These columns return a null if the data file is offline and therefore not readable.
ALL_IND_COLUMNS DBA_IND_COLUMNS USER_IND_COLUMNS	COLUMN_NAME	This column returns a null if an index is on a function instead of a column. In this case, there is no column to list.
ALL_IND_PARTITIONS DBA_IND_PARTITIONS USER_IND_PARTITIONS	INITIAL_EXTENT MAX_EXTENT MIN_EXTENT NEXT_EXTENT PCT_INCREASE	These columns return a null if the index is partitioned using a composite method and no default value was specified for the partition.
ALL_OBJECT_TABLES DBA_OBJECT_TABLES USER_OBJECT_TABLES	TABLESPACE_NAME	This column returns a null in if an object table is partitioned or if it is a temporary table.
ALL_SEGMENTS DBA_SEGMENTS USER_SEGMENTS	BLOCKS BYTES EXTENTS NEXT_EXTENT PCT_INCREASE	The BLOCKS, BYTES, and EXTENTS columns return a null if the segment header cannot be read because the file is offline or if there is some other corruption. The of NEXT_EXTENT and PCT_INCREASE columns return a null if the tablespace storing the segment is locally managed and uses the AUTOALLOCATE option, because the system chooses the extent sizes, and the algorithm cannot be explained in terms of NEXT_EXTENT and PCT_INCREASE.

Table C-4 Columns That May Return Nulls in Release 8.1

Static Data Dictionary Views	Columns	Explanation
ALL_TAB_PARTITIONS DBA_TAB_PARTITIONS USER_TAB_PARTITIONS	INITIAL_EXTENT MAX_EXTENT MIN_EXTENT NEXT_EXTENT PCT_INCREASE	These columns return a null if the table is partitioned using a composite method and no default value was specified for the partition.
ALL_TABLESPACES DBA_TABLESPACES USER_TABLESPACES	NEXT_EXTENT PCT_INCREASE	These columns return a null if the tablespace is locally managed and uses the AUTOALLOCATE option, because the system chooses the extent sizes, and the algorithm cannot be explained in terms of NEXT_EXTENT and PCT_INCREASE.
ALL_TRIGGERS DBA_TRIGGERS USER_TRIGGERS	TABLE_NAME	This column returns a null if the trigger is a system trigger. In this case, the base object type of the trigger will be SCHEMA or DATABASE, instead of TABLE or VIEW.

Static Data Dictionary Views Obsolete in Version 8

The static data dictionary views in this section are obsolete in version 8.

Static Data Dictionary Views Obsolete in Release 8.0

The following static data dictionary views became obsolete in release 8.0 and are not available in release 8.0 and higher:

ALL_HISTOGRAMS

DBA_HISTOGRAMS

DEFCALL

USER_HISTOGRAMS

Static Data Dictionary Views Obsolete in Release 8.1

The following static data dictionary view became obsolete in release 8.1 and is not available in release 8.1 and higher:

ALL_LABELS

Changes to Dynamic Performance Views

Version 8 supports new dynamic performance views (V\$ views), and some dynamic performance views have been changed or have become obsolete in version 8. This appendix lists the new, changed, and obsolete dynamic performance views in version 8.

This appendix covers the following topics:

- [Dynamic Performance Views Added in Version 8](#)
- [Dynamic Performance Views with Added Columns in Version 8](#)
- [Dynamic Performance Views with Dropped Columns in Release 8.1](#)
- [Dynamic Performance Views Obsolete in Version 8](#)
- [Date Columns in Dynamic Performance Views](#)

See Also: *Oracle8i Reference* for descriptions of the new and changed views listed in this appendix, and for descriptions of the columns in these views.

Dynamic Performance Views Added in Version 8

The dynamic performance views listed in this section are new in version 8.

Dynamic Performance Views Added in Release 8.0

The following dynamic performance views were added in release 8.0:

GV\$ACCESS	GV\$ACTIVE_INSTANCES
GV\$AQ	GV\$ARCHIVE
GV\$ARCHIVE_DEST	GV\$ARCHIVED_LOG
GV\$BACKUP	GV\$BACKUP_CORRUPTION
GV\$BACKUP_DATAFILE	GV\$BACKUP_DEVICE
GV\$BACKUP_PIECE	GV\$BACKUP_REDOLOG
GV\$BACKUP_SET	GV\$BGPROCESS
GV\$BH	GV\$BUFFER_POOL
GV\$CACHE	GV\$CIRCUIT
GV\$CLASS_PING	GV\$COMPATIBILITY
GV\$COMPATSEG	GV\$CONTROLFILE
GV\$CONTROLFILE_RECORD_SECTION	GV\$COPY_CORRUPTION
GV\$CURRENT_BUCKET	GV\$DATABASE
GV\$DATAFILE	GV\$DATAFILE_COPY
GV\$DATAFILE_HEADER	GV\$DB_OBJECT_CACHE
GV\$DB_PIPES	GV\$DBFILE
GV\$DBLINK	GV\$DELETED_OBJECT
GV\$DISPATCHER	GV\$DISPATCHER_RATE
GV\$DLM_CONVERT_LOCAL	GV\$DLM_CONVERT_REMOTE
GV\$DLM_LATCH	GV\$DLM_LOCKS
GV\$DLM_MISC	GV\$ENABLEDPRIVS
GV\$ENQUEUE_LOCK	GV\$EVENT_NAME
GV\$EXECUTION	GV\$FALSE_PING
GV\$FILE_PING	GV\$FILESTAT
GV\$FIXED_TABLE	GV\$FIXED_VIEW_DEFINITION

GV\$GLOBAL_TRANSACTION	GV\$INDEXED_FIXED_COLUMN
GV\$INSTANCE	GV\$LATCH
GV\$LATCH_CHILDREN	GV\$LATCH_MISSES
GV\$LATCH_PARENT	GV\$LATCHHOLDER
GV\$LATCHNAME	GV\$LIBRARYCACHE
GV\$LICENSE	GV\$LOADCSTAT
GV\$LOADPSTAT	GV\$LOADTSTAT
GV\$LOCK	GV\$LOCK_ACTIVITY
GV\$LOCK_ELEMENT	GV\$LOCKED_OBJECT
GV\$LOCKS_WITH_COLLISIONS	GV\$LOG
GV\$LOG_HISTORY	GV\$LOGFILE
GV\$LOGHIST	GV\$MLS_PARAMETERS
GV\$MTS	GV\$MYSTAT
GV\$NLS_PARAMETERS	GV\$NLS_VALID_VALUES
GV\$OBJECT_DEPENDENCY	GV\$OFFLINE_RANGE
GV\$OPEN_CURSOR	GV\$OPTION
GV\$PARAMETER	GV\$PING
GV\$PQ_SESSTAT	GV\$PQ_SLAVE
GV\$PQ_SYSSTAT	GV\$PQ_TQSTAT
GV\$PROCESS	GV\$PWFILE_USERS
GV\$QUEUE	GV\$RECENT_BUCKET
GV\$RECOVER_FILE	GV\$RECOVERY_FILE_STATUS
GV\$RECOVERY_LOG	GV\$RECOVERY_PROGRESS
GV\$RECOVERY_STATUS	GV\$REQDIST
GV\$RESOURCE	GV\$RESOURCE_LIMIT
GV\$ROLLSTAT	GV\$ROWCACHE
GV\$ROWCACHE_PARENT	GV\$ROWCACHE_SUBORDINATE
GV\$SESS_IO	GV\$SESSION
GV\$SESSION_CONNECT_INFO	GV\$SESSION_CURSOR_CACHE
GV\$SESSION_EVENT	GV\$SESSION_LONGOPS
GV\$SESSION_OBJECT_CACHE	GV\$SESSION_WAIT

GV\$SESSTAT	GV\$SGA
GV\$SGASTAT	GV\$SHARED_POOL_RESERVED
GV\$SHARED_SERVER	GV\$SORT_SEGMENT
GV\$SORT_USAGE	GV\$SQL
GV\$SQL_BIND_DATA	GV\$SQL_BIND_METADATA
GV\$SQL_CURSOR	GV\$SQL_SHARED_MEMORY
GV\$SQLAREA	GV\$SQLTEXT
GV\$SQLTEXT_WITH_NEWLINES	GV\$STATNAME
GV\$SUBCACHE	GV\$SYSSTAT
GV\$SYSTEM_CURSOR_CACHE	GV\$SYSTEM_EVENT
GV\$SYSTEM_PARAMETER	GV\$TABLESPACE
GV\$THREAD	GV\$TIMER
GV\$TRANSACTION	GV\$TRANSACTION_ENQUEUE
GV\$TYPE_SIZE	GV\$VERSION
GV\$WAITSTAT	V\$AQ
V\$ARCHIVE_DEST	V\$ARCHIVED_LOG
V\$BACKUP_CORRUPTION	V\$BACKUP_DATAFILE
V\$BACKUP_DEVICE	V\$BACKUP_PIECE
V\$BACKUP_REDOLOG	V\$BACKUP_SET
V\$BUFFER_POOL	V\$CLASS_PING
V\$CONTROLFILE_RECORD_SECTION	V\$COPY_CORRUPTION
V\$CURRENT_BUCKET	V\$DATAFILE_COPY
V\$DATAFILE_HEADER	V\$DELETED_OBJECT
V\$DISPATCHER_RATE	V\$DLM_CONVERT_LOCAL
V\$DLM_CONVERT_REMOTE	V\$DLM_LATCH
V\$DLM_LOCKS	V\$DLM_MISC
V\$ENQUEUE_LOCK	V\$FILE_PING
V\$GLOBAL_TRANSACTION	V\$LOADPSTAT
V\$OFFLINE_RANGE	V\$RECENT_BUCKET
V\$RECOVERY_PROGRESS	V\$RESOURCE_LIMIT
V\$ROWCACHE_PARENT	V\$ROWCACHE_SUBORDINATE

V\$SESSION_LONGOPS	V\$SESSION_OBJECT_CACHE
V\$SORT_USAGE	V\$SUBCACHE
V\$TABLESPACE	V\$TRANSACTION_ENQUEUE

Dynamic Performance Views Added in Release 8.1

The following dynamic performance views were added in release 8.1:

GV\$ARCHIVE_PROCESSES	GV\$BACKUP_ASYNC_IO
GV\$BACKUP_SYNC_IO	GV\$CONTEXT
GV\$DLM_ALL_LOCKS	GV\$DLM_RESS
GV\$FAST_START_SERVERS	GV\$FAST_START_TRANSACTIONS
GV\$GLOBAL_BLOCKED_LOCKS	GV\$HS_AGENT
GV\$HS_SESSION	GV\$INSTANCE_RECOVERY
GV\$LOGMNR_CONTENTS	GV\$LOGMNR_DICTIONARY
GV\$LOGMNR_LOGS	GV\$LOGMNR_PARAMETERS
GV\$PARALLEL_DEGREE_LIMIT_MTH	GV\$PROXY_ARCHIVEDLOG
GV\$PROXY_DATAFILE	GV\$PX_PROCESS
GV\$PX_PROCESS_SYSSTAT	GV\$PX_SESSION
GV\$PX_SESSTAT	GV\$RESERVED_WORDS
GV\$RSRC_CONSUMER_GROUP	GV\$RSRC_CONSUMER_GROUP_CPU_MTH
GV\$RSRC_PLAN	GV\$RSRC_PLAN_CPU_MTH
GV\$TEMP_EXTENT_MAP	GV\$TEMP_EXTENT_POOL
GV\$TEMP_PING	GV\$TEMP_SPACE_HEADER
GV\$TEMPFILE	GV\$TEMPORARY_LOBS
GV\$TEMPSTAT	V\$ARCHIVE_PROCESSES
V\$BACKUP_ASYNC_IO	V\$BACKUP_SYNC_IO
V\$CONTEXT	V\$DLM_ALL_LOCKS
V\$DLM_RESS	V\$FAST_START_SERVERS
V\$FAST_START_TRANSACTIONS	V\$GLOBAL_BLOCKED_LOCKS
V\$HS_AGENT	V\$HS_SESSION
V\$INSTANCE_RECOVERY	V\$LOGMNR_CONTENTS

V\$LOGMNR_DICTIONARY	V\$LOGMNR_LOGS
V\$LOGMNR_PARAMETERS	V\$OBSOLETE_PARAMETER
V\$PARALLEL_DEGREE_LIMIT_MTH	V\$PROXY_ARCHIVEDLOG
V\$PROXY_DATAFILE	V\$PX_PROCESS
V\$PX_PROCESS_SYSSTAT	V\$PX_SESSION
V\$PX_SESSTAT	V\$RESERVED_WORDS
V\$RSRC_CONSUMER_GROUP	V\$RSRC_CONSUMER_GROUP_CPU_MTH
V\$RSRC_PLAN	V\$RSRC_PLAN_CPU_MTH
V\$TEMP_EXTENT_MAP	V\$TEMP_EXTENT_POOL
V\$TEMP_PING	V\$TEMP_SPACE_HEADER
V\$TEMPFILE	V\$TEMPORARY_LOBS
V\$TEMPSTAT	

Dynamic Performance Views Renamed in Version 8

The dynamic performance views in this section have been renamed in version 8. If an application requires one or more of the views listed below, modify the application accordingly.

Dynamic Performance Views Renamed in Release 8.1

The following dynamic performance views were renamed in release 8.1:

Table D-1 Dynamic Performance Views Renamed in Release 8.1

Release 8.1.3 and 8.1.4 Name	Release 8.1.5 and Higher Name
GV\$RECOVERY_SERVERS	GV\$FAST_START_SERVERS
GV\$RECOVERY_TRANSACTIONS	GV\$FAST_START_TRANSACTIONS
GV\$TARGETRBA	GV\$INSTANCE_RECOVERY
V\$RECOVERY_SERVERS	V\$FAST_START_SERVERS
V\$RECOVERY_TRANSACTIONS	V\$FAST_START_TRANSACTIONS
V\$TARGETRBA	V\$INSTANCE_RECOVERY

Note: No dynamic performance views were renamed in release 8.0.

Dynamic Performance Views with Added Columns in Version 8

New columns were added to the dynamic performance views listed in the following sections.

Dynamic Performance Views with Added Columns in Release 8.0

New columns were added to the following dynamic performance views in release 8.0:

V\$BH	V\$CACHE
V\$CACHE_LOCK	V\$DATABASE
V\$DATAFILE	V\$FALSE_PING
V\$FILESTAT	V\$INSTANCE
V\$LATCH_MISSES	V\$LOADCSTAT
V\$LOADTSTAT	V\$LOG_HISTORY
V\$PING	V\$SESSION
V\$SESSION_EVENT	V\$SGASTAT
V\$SORT_SEGMENT	V\$SQL
V\$THREAD	V\$TRANSACTION

Dynamic Performance Views with Added Columns in Release 8.1

New columns were added to the following dynamic performance views in release 8.1:

GV\$ARCHIVE_DEST	GV\$AQ
GV\$BACKUP_PIECE	GV\$BH
GV\$CIRCUIT	GV\$DATABASE
GV\$DATAFILE	GV\$DISPATCHER
GV\$DLM_LATCH	GV\$DLM_LOCKS
GV\$INSTANCE	GV\$INSTANCE_RECOVERY
GV\$LOGMNR_CONTENTS	GV\$SESSION
GV\$SESSION_LONGOPS	GV\$SQL
V\$ARCHIVE_DEST	V\$AQ
V\$BACKUP_PIECE	V\$BH
V\$CIRCUIT	V\$DATABASE
V\$DATAFILE	V\$DISPATCHER
V\$DLM_LATCH	V\$DLM_LOCKS
V\$INSTANCE	V\$INSTANCE_RECOVERY
V\$LOGMNR_CONTENTS	V\$SESSION
V\$SESSION_LONGOPS	V\$SQL

Note: The GV\$TARGETRBA and V\$TARGETRBA dynamic performance views were added in release 8.1.3. The names of these views are changed to GV\$INSTANCE_RECOVERY and V\$INSTANCE_RECOVERY, respectively, in release 8.1.5 and higher. As the preceding list shows, new columns were added to these dynamic performance views under the new names in release 8.1.5.

Dynamic Performance Views with Dropped Columns in Release 8.1

The columns listed in the following sections were dropped in release 8.1. If an application requires one or more of the columns listed below, modify the application accordingly.

Dynamic Performance Views with Dropped Columns in Release 8.1

The columns listed in [Table D-2](#) were dropped in release 8.1.

Table D-2 *Dynamic Performance Views with Dropped Columns in Release 8.1*

Dynamic Performance View	Dropped Columns
V\$ARCHIVE_DEST	ARCMODE
V\$DLM_LATCH	IMM_GETS LATCH_TYPE TTL_GETS
V\$DLM_LOCKS	RESOURCE_NAME
V\$SESSION_LONGOPS	APPLICATION_DATA_1 APPLICATION_DATA_2 APPLICATION_DATA_3 COMPNAM CURRENT_TIME MSG OBJID OPID STEPID STEPSOFAR STEPTOTAL UPDATE_COUNT

Dynamic Performance Views Obsolete in Version 8

The dynamic performance views in this section are obsolete in version 8.

Dynamic Performance Views Obsolete in Release 8.1

The following dynamic performance views became obsolete in release 8.1 and are not available in release 8.1 and higher:

GV\$CURRENT_BUCKET

GV\$RECENT_BUCKET

V\$CURRENT_BUCKET

V\$RECENT_BUCKET

Note: No dynamic performance views became obsolete in release 8.0.

Date Columns in Dynamic Performance Views

In Oracle7, all date columns in dynamic performance views were VARCHAR2(20) strings in MM/DD/YY HH24:MI:SS format. In version 8, every new date column is a real DATE column that uses the DATE datatype. In contrast to the previous VARCHAR2(20) string, the DATE datatype provides the following benefits:

- Establishes consistency, because all date columns are in the DATE datatype.
- Makes it easier to perform date arithmetic (including sorting) in SQL and PL/SQL.
- Enables you to set your date format using NLS_DATE_FORMAT.
- Lets you to see dates in the old format by setting NLS_DATE_FORMAT to MM/DD/YY HH24:MI:SS.
- Avoids two-digit year numbers, thereby avoiding problems at the year 2000 and beyond.

Note: Although Oracle7 displays dates using the VARCHAR(20) datatype in dynamic performance views, Oracle7 still is fully year-2000 compliant. Oracle7 stores time to the nearest second in the redo log files and control files.

New Internal Datatypes and SQL Functions

This appendix lists the new internal datatypes and SQL functions added in version 8. This appendix covers the following topics:

- [Internal Datatypes Added in the New Release](#)
- [SQL Functions Added in the New Release](#)

See Also: *Oracle8i SQL Reference* for a complete list and descriptions of Oracle internal datatypes and SQL functions.

Internal Datatypes Added in the New Release

The internal datatypes listed in this section are new in version 8. Each may be used as a function name in a SELECT list, but only if it is qualified with a schema (*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 did not generate an error.

Internal Datatypes Added in Release 8.0

The following internal datatypes were added in release 8.0:

- NCHAR
- NVARCHAR2
- CLOB
- NCLOB
- BLOB
- BFILE

Internal Datatype Added in Release 8.1

The following internal datatype was added in release 8.1:

- UROWID (universal rowid)

SQL Functions Added in the New Release

The SQL functions listed in this section are new in version 8.

SQL Functions Added in Release 8.0

The following SQL functions were added in release 8.0:

Single-Row Functions

The following single-row functions were added in release 8.0:

- EMPTY_[B | C]LOB
- BFILENAME
- NLS_CHARSET_DECL_LEN
- NLS_CHARSET_ID
- NLS_CHARSET_NAME

Object Reference Functions

The following object reference functions were added in release 8.0:

- DEFREF
- MAKE_REF
- REF
- REFTOHEX
- VALUE

SQL Functions Added in Release 8.1

The following functions were added in release 8.1:

Character Functions

The following character function was added in release 8.1:

- TRIM

Conversion Functions

The following conversion function was added in release 8.1:

- TO_LOB

Aggregate Functions

The following aggregate function was added in release 8.1:

- GROUPING

Single-Row Functions

The following single-row functions were added in release 8.1:

- SYS_CONTEXT
- SYS_GUID

Note: Function names beginning with SYS are reserved by Oracle. You should not begin any function names with SYS; if you do, you may encounter compatibility problems.

Numerics

- 2 GB and larger files
 - operating system dependencies, 8-50
- 32-bit to 64-bit conversion. See word-size

A

- active transactions
 - read-only tablespaces, 8-23
- Ada. See SQL*Module for Ada
- administrative procedures
 - new after migration, 6-7
- Advanced Queuing
 - compatibility, 8-14, 8-32
 - message streaming, 8-32
 - privileges, 8-32
 - rule based subscriptions, 8-32
 - downgrading queue tables, 12-31
 - extended address field, 7-26
 - interoperability, 8-32
 - sender's ID column, 8-32
 - removing propagation, 12-32
 - revoking object and system level privileges, 12-30
 - scripts
 - CATNOQUEUE.SQL, 7-28
 - CATQUEUE.SQL, 7-28
 - upgrading, 7-26
- advanced replication
 - compatibility, 8-13, 8-45
 - downgrading, 12-49
 - preparing environment for, 12-38
 - regenerating after, 12-51
 - migration, 3-7
 - upgrading, 7-21
- agent self-registration
 - compatibility, 8-13
- ALTER DATABASE CONVERT command, 3-4, 3-26
 - re-running, A-19
- ALTER DATABASE OPEN RESETLOGS
 - command, 3-4, 3-26
- ALTER DATABASE RESET COMPATIBILITY, 8-7
- ALTER TABLE command
 - bitmap index invalidation, 8-25
- ANALYZE TABLE VALIDATE STRUCTURE
 - command
 - change in release 8.1, 8-17
- application context
 - compatibility, 8-13
 - downgrading, 12-36
- application developer
 - role during migration, 1-10
- applications
 - compatibility, 8-16
 - index-organized tables
 - compatibility, 8-16
 - interoperability, 8-16
 - OCI
 - compatibility, 8-17
 - interoperability, 8-17
 - physical ROWIDs and UROWIDs, 8-16
 - physical ROWIDs in Oracle8i, 11-2
- PL/SQL
 - compatibility, 8-20
 - interoperability, 8-20
- precompiler

- compatibility, 8-18
- interoperability, 8-18
- Tuxedo applications
 - compatibility, 8-17
 - upgrading, 9-1
- AQ. See Advanced Queuing
- archive log destination parameters
 - new in release 8.1, B-13
- archiving
 - error detection behavior, 8-43
- associations
 - downgrading, 12-35
- AUDIT_TRAIL initialization parameter
 - migration, A-3
- autonomous transactions
 - compatibility, 8-14

B

- backup
 - after migration, 6-2
 - before migration, 3-20
 - compatibility, 8-13, 8-39
 - EBU, 8-40
 - preparing a strategy for migration, 2-18
 - Recovery Manager and EBU, 8-40
- bitmap indexes
 - invalidations, 8-25
 - during migration, 6-3
 - when downgrading, 12-12
- block size
 - DB_BLOCK_SIZE initialization parameter, 3-6, 4-5
 - minimums for migration, 3-6, 4-5
- BUILD DEFERRED clause
 - materialized views and downgrading, 12-10
- bulk binds
 - compatibility, 8-14

C

- C call specifications
 - compatibility, 8-14
- CALL statement
 - compatibility, 8-14

- CATALOG5.SQL, obsolete with Oracle8i, 8-24
- CATALOG6.SQL, obsolete with Oracle8i, 8-24
- CATALOG.SQL script, 3-4, 3-27, 7-19, 12-49
- CATEXP7.SQL script, 8-48
 - downgrading to Oracle7, 13-3
- CATNOQUEUE.SQL script, 7-28
- CATPARR.SQL script, 3-28, 7-24, 12-49
- CATPROC.SQL script, 3-4, 3-27, 7-19, 12-49
- CATQUEUE.SQL script, 7-28
- CATREP.SQL script, 3-28, 4-17, 7-21, 12-49
- chained row tables
 - dropping, 12-18
- CHARACTER keyword
 - behavior in Oracle7 and Oracle8i, 8-22
- character set
 - migrating the database, 3-8, 4-6
- character sets
 - Migration utility, 3-8
 - Oracle Data Migration Assistant, 4-6
 - varying-width
 - CLOBs and NCLOBs, 8-25
- CHECK_ONLY
 - Migration utility option, 3-14, 3-18
- CLOBs
 - compatibility, 8-25
- cluster tables
 - copying data, 2-15
- collection columns
 - user-specified storage
 - compatibility, 8-29
- collection locators
 - compatibility, 8-12, 8-29
- columns
 - partially dropped
 - dropping, 12-8
 - unused
 - dropping, 12-8
- command-line options
 - Migration utility, 3-14
- comments
 - differences between Server Manager and SQL*Plus, 10-7
- COMMIT command
 - differences between Server Manager and SQL*Plus, 10-14

- COMMIT keyword
 - behavior in Oracle7 and Oracle8i, 8-22
- compatibility, 8-1
 - 8.1.0 compatibility level
 - features requiring, 8-9
 - Advanced Queuing, 8-14, 8-32
 - Advanced Replication, 8-13, 8-45
 - agent self-registration, 8-13
 - ALTER DATABASE RESET COMPATIBILITY, 8-7
 - ANALYZE VALIDATE STRUCTURE
 - command, 8-17
 - application context, 8-13
 - applications, 8-16
 - index-organized tables, 8-16
 - physical ROWIDs and UROWIDs, 8-16
 - Tuxedo applications, 8-17
 - XA calls, 8-17
 - backup, 8-13, 8-39
 - compatibility level, 8-5
 - COMPATIBLE initialization parameter, 8-2
 - conventional path export, 8-48
 - CREATE TABLE behavior change, 8-31
 - database resource manager, 8-14
 - date columns in dynamic performance views, D-11
 - DBMS_REPAIR package, 8-14
 - direct path export, 8-48
 - DISABLE VALIDATE constraints, 8-14
 - Export/Import, 8-48
 - extensible optimizer, 8-14
 - fine-grained access control, 8-13
 - Heterogeneous Services agents, 8-45
 - indexes
 - key compression, 8-10
 - initialization parameters, B-8
 - LOBs, 8-11, 8-25
 - CLOBs and NCLOBs, 8-25
 - materialized views, 8-13
 - NCHAR and NLS, 8-27
 - nested tables, 8-29
 - NLS and NCHAR environment variables, 8-28
 - n-tier authentication and authorization, 8-13
 - OCI, 8-17
 - link line, 8-18
 - thread safety, 8-18
 - optimization, 8-34
 - optimizer plan stability, 8-14
 - Oracle Media Management API, 8-13
 - Oracle Parallel Server
 - instance affinity for jobs, 8-12
 - packages
 - DBMS_LOB, 8-33
 - DBMS_REPAIR, 8-33
 - partitioning, 8-11
 - PL/SQL, 8-20
 - PLSQL_V2_COMPATIBILITY initialization parameter, 8-20
 - precompilers, 8-18
 - recovery, 8-13, 8-39
 - fast-start rollback, 8-13
 - removing incompatibilities, 12-2
 - replication, 8-44
 - resetting for database, 12-42
 - schema objects, 8-10
 - collection locators, 8-12
 - drop column, 8-10
 - extensible indexing, 8-10
 - function-based indexes, 8-10
 - indexes, 8-10
 - index-organized tables, 8-10
 - nested tables, 8-12
 - online index (re)build, 8-10
 - single-table hash clusters, 8-10
 - temporary tables, 8-10
 - scripts
 - UTLCHAIN1.SQL, 8-14, 8-31
 - UTLEXCPT1.SQL, 8-14, 8-31
 - Spatial, 8-15
 - SQL and PL/SQL, 8-14
 - autonomous transactions, 8-14
 - bulk binds, 8-14
 - C call specifications, 8-14
 - CALL statement, 8-14
 - native dynamic SQL, 8-14
 - NOCOPY parameter passing mode, 8-14
 - standby database, 8-41
 - summary management, 8-13
 - tablespaces, 8-9
 - locally managed tablespaces, 8-9

- online read-only tablespaces, 8-9
 - transportable tablespaces, 8-9
- triggers, 8-14
- UROWIDs, 8-11
- user-defined object identifiers, 8-12
- VARRAYs, 8-12
- Visual Image Retrieval, 8-15
- COMPATIBLE initialization parameter, 8-2
 - 8.1.0 setting
 - features requiring, 8-9
 - checking the setting, 8-5
 - database structures, 8-3
 - setting the, 8-6
- composite methods
 - partitioning and downgrading, 12-17
- concurrent access, 2-16
- concurrent users, 2-17
- connections
 - load balancing in Net8, 8-47
 - local and remote, 2-16
 - with multithreaded shared server, 2-16
- control files
 - renaming or removing for migration, 3-22
- copying data
 - large cluster tables, 2-15
 - migration method, 2-14
 - space requirements, 2-15
- CREATE LIBRARY command
 - differences between Server Manager and SQL*Plus, 10-13
- CREATE TABLE
 - behavior change
 - Oracle7 to Oracle8i, 8-31
- CREATE TYPE command
 - differences between Server Manager and SQL*Plus, 10-13
- cursors
 - number of open, 2-17

D

- data definition conversion
 - Import utility, 2-13
- data dictionary
 - protection, 8-24

- database administrator
 - role during migration, 1-10
- database resource manager
 - compatibility, 8-14
- databases
 - backing up for migration, 3-20
 - downgrading, 12-43
 - test migration results, 6-6
 - tuning after migration, 6-6
- datafiles
 - offline during migration, 3-4
- datatypes
 - internal
 - new in Oracle8i, E-2
- date constraints
 - checking for bad, 6-2, 7-33
- DB_BLOCK_SIZE initialization parameter
 - migration, 3-6, 4-5
- DB_DOMAIN initialization parameter
 - compatibility, B-9
- DBMS
 - precompiler command line option, 8-19
- DBMS_LOB package
 - NOCOPY syntax and compatibility, 8-33
- DBMS_REPAIR package
 - compatibility, 8-14, 8-33
- DBMS_ROWID package, 11-3
- DBMSLOB.SQL script
 - compatibility, 8-33
- DBNAME
 - Migration utility option, 3-14
- DEC keyword
 - behavior in Oracle7 and Oracle8i, 8-22
- definitions. See terminology
- DEGREE keyword
 - in PARALLEL clause, 8-35
- deinstalling, 1-6
- Developer/2000 Applications
 - upgrading, 9-6
- dimensions
 - downgrading, 12-14
- DISABLE VALIDATE constraints
 - compatibility, 8-14
 - downgrading, 12-34
- DML_LOCKS initialization parameter

- compatibility, B-8
- documentation
 - roadmap for Migration utility, 3-2
 - roadmap for Oracle Data Migration Assistant, 4-2
- domain indexes
 - dropping, 12-13
- downgrading
 - Advanced Replication
 - regenerating, 12-51
 - advanced replication, 12-49
 - CATALOG.SQL, 12-49
 - CATPROC.SQL, 12-49
 - definition, 1-3
 - Oracle Parallel Server, 12-49
 - procedure for, 12-43
 - queue tables, 12-31
 - removing incompatibilities, 12-2
 - Advanced Replication, 12-38
 - application context, 12-36
 - associations, 12-35
 - bitmap index invalidation, 12-12
 - chained row tables, 12-18
 - dimensions, 12-14
 - DISABLE VALIDATE constraints, 12-34
 - domain indexes, 12-13
 - exception tables, 12-18
 - extensible indexing, 12-13
 - fine-grained access control, 12-37
 - function-based indexes, 12-13
 - indextypes, 12-13
 - Java, 12-26
 - key compression on indexes and index-organized tables, 12-6
 - LOBs, 12-19
 - LOBs in index-organized tables, 12-7
 - locally managed tablespaces, 12-5
 - materialized views, 12-9
 - mutually referencing types, 12-24
 - mutually referencing views, 12-10
 - nested tables, 12-23
 - Net8 service naming, 12-41
 - operators, 12-14
 - optimizer, 12-35
 - partially dropped columns, 12-8
 - partitioning, 12-14
 - plan stability, 12-36
 - secondary indexes on index-organized tables, 12-8
 - security policies, 12-37
 - SET_SESSION_LONGOPS syntax
 - change, 12-33
 - single-table hash clusters, 12-9
 - SQL and PL/SQL, 12-25
 - stored outlines, 12-36
 - temporary tables, 12-6
 - transported tablespaces, 12-5
 - triggers, 12-35
 - triggers on nested table view columns, 12-34
 - unused columns, 12-8
 - UROWIDs, 12-17
 - user-defined datatypes, 12-21
 - user-defined object identifiers, 12-22
 - UTL_REF package, 12-33
 - VARRAYs, 12-24
 - resetting database compatibility, 12-42
 - scripts, 12-45
 - errors while running, 1-3
 - rerunning, 12-46
 - to an older Oracle8i release, 12-1
 - to Oracle7, 13-1
 - alternative methods, 13-4
 - CATEXP7.SQL script, 13-3
 - drop column
 - compatibility, 8-10
 - DTYCHR type, 11-8
 - dynamic performance views
 - added in Oracle8i, D-2
 - changed in Oracle8i
 - added columns, D-8
 - date columns
 - compatibility, D-11
 - obsolete in Oracle8i, D-11
 - renamed in Oracle8i, D-7

E

 - EBU
 - backup management, 8-40
 - enterprise directory service

- interoperability, 8-38
- environment variables
 - compatibility
 - NCHAR and NLS, 8-28
 - ORA_NLS32, 8-28
 - ORA_NLS33, 3-16, 8-28
 - required for migration, 3-16, 3-21
 - required for upgrading, 7-16
 - TWO_TASK, 3-16
- errors during migration. See troubleshooting
- exception tables
 - dropping, 12-18
- Export utility
 - migration, 2-10, 5-1
 - Oracle8i using CATEXP7.SQL, 13-3
 - requirements for migration, 5-2
- Export/Import
 - advantages and disadvantages, 2-10
 - basic steps for migration, 5-2
 - benefits for migration, 2-12
 - compatibility, 8-48
 - data definition conversion, 2-13
 - direct and conventional path, 8-48
 - effects on migrated databases, 2-11
 - incompatible data, 8-49
 - limitations for migration, 2-12
 - migration steps using, 5-3
 - scripts
 - CATEXP7.SQL, 8-48
 - time requirements for migration, 2-13
 - Trusted Oracle, 5-3
- extended address field
 - Advanced Queuing, 7-26
- extensible indexing
 - compatibility, 8-10
 - downgrading, 12-13

F

- FALSE keyword
 - behavior in Oracle7 and Oracle8i, 8-22
- FAST REFRESH mode
 - materialized views and downgrading, 12-10
- fast-start parallel recovery
 - compatibility, 8-42

- fast-start rollback
 - compatibility, 8-13, 8-42
- features
 - new features, 8-9
- fine-grained access control
 - compatibility, 8-13
- Forms
 - upgrading Oracle Forms applications, 9-6
- function-based indexes
 - compatibility, 8-10
- functions
 - SQL
 - new in Oracle8i, E-3

G

glossary. See terminology

H

- Heterogeneous Services
 - agents
 - compatibility, 8-45
 - interoperability, 8-45
 - multithreaded, 8-45

I

- Import utility
 - data definition conversion, 2-13
 - migration, 2-10, 5-1
 - requirements for migration, 5-2
- incompatibilities
 - removing, 12-2
 - system-defined, 12-4
- incompatibilities, system-defined, 8-6
- index
 - key compression, 8-10
 - index on physical ROWID, 11-8
 - index rebuilding
 - physical ROWIDs, 11-8
- indexes
 - bitmap, 6-3
 - compatibility, 8-10
 - domain

- dropping, 12-13
- function-based
 - dropping, 12-13
- index-organized tables
 - compatibility, 8-10
 - removing LOBs from, 12-7
 - removing partitions from, 12-14
- indextypes
 - dropping, 12-13
- initialization parameters
 - added in Oracle8i, B-2
 - adjusting for release 8.1, 3-24, 4-10, 7-15
 - archive log destination
 - switching to new, B-13
 - changes in Oracle8i, B-1
 - compatibility, B-8
 - DB_DOMAIN, B-9
 - DML_LOCKS, B-8
 - O7_DICTIONARY_ACCESSIBILITY, B-8
 - COMPATIBLE, 8-2
 - LARGE_POOL_SIZE
 - parallel execution allocation, B-9
 - obsolete in Oracle8i, B-6
 - renamed in Oracle8i, B-4
 - SHARED_POOL_SIZE
 - parallel execution allocation, B-9
- INIT.ORA parameters. See initialization parameters
- installation
 - release 8.1 Oracle software, 3-12, 4-11, 5-3
- INSTANCES keyword
 - removed from PARALLEL clause, 8-35
- INT keyword
 - behavior in Oracle7 and Oracle8i, 8-22
- internal datatypes
 - new in Oracle8i, E-2
- interoperability, 8-1, 8-15
 - Advanced Queuing, 8-32
 - applications, 8-16
 - Heterogeneous Services agents, 8-45
 - native dynamic SQL, 8-30
 - NCHAR and NLS, 8-27
 - OCI, 8-17
 - Oracle7 clients, 8-18
 - shared structures, 8-17
 - PL/SQL, 8-20

- precompilers, 8-18
- UROWIDs, 8-26
- user-defined datatypes, 8-29

J

- Java
 - compatibility, 8-9
 - removing incompatibilities for
 - downgrading, 12-26
 - upgrading
 - drop Java objects, 7-36
 - export Java objects, 7-3
 - import Java objects, 7-36

K

- key compression
 - discontinuing use of, 12-6
- keywords
 - behavior differences
 - Oracle7 and Oracle8i, 8-22

L

- large files
 - operating system dependencies, 8-50
- Large Objects. See LOBs
- LARGE_POOL_SIZE initialization parameter
 - changes in release 8.1, 6-3, 7-34
 - parallel execution allocation, B-9
- load balancing
 - Net8, 8-47
- LOBs
 - compatibility, 8-11, 8-25
 - downgrading, 12-19
 - removing from index-organized tables, 12-7
 - removing from partitioned tables, 12-20
- local connections
 - multithreaded shared servers, 2-16
- locally managed tablespaces
 - compatibility, 8-9
 - converting to dictionary tablespaces, 12-5
- locks
 - DML lock limit, DML_LOCKS, B-8

M

materialized views

- compatibility, 8-13
- downgrading, 12-9
 - BUILD DEFERRED clause, 12-10
 - FAST REFRESH mode, 12-10
 - NEVER REFRESH mode, 12-10
 - PREBUILT TABLE clause, 12-10
 - REFRESH ON COMMIT mode, 12-9
- removing incompatibilities, 12-9

memory requirements

- concurrent access, 2-16
- migration, 2-15, 3-6, 4-4

MIGRATE user

- avoid, 3-10, 4-8

MIGRATE.BSQ script, 3-19

migrating

- from Server Manager to SQL*Plus, 10-1

migration

- abandoning, 3-30, 4-20
- advanced replication, 3-7
- ALTER DATABASE CONVERT command, 3-4, 3-26
 - re-running, A-19
- ALTER DATABASE OPEN RESETLOGS command, 3-4, 3-26
- AUDIT_TRAIL initialization parameter, A-3
- avoiding common problems, 2-18
- backup strategy, 2-18
- block size minimums, 3-6, 4-5
- character set, 3-8, 4-6
- choosing a method, 2-3
- control files, 3-22
- copying data, 2-14
- exclusive password file, A-18
- Export/Import, 2-10
 - steps, 5-2
- initialization parameters, 3-24, 4-10, B-1
- memory requirements, 2-15, 3-6, 4-4
- MIGRATE user, avoid, 3-10, 4-8
- MIGRATE.BSQ script, 3-19
- Migration utility, 2-6
- NCHAR and NLS, 8-27
- new administrative procedures, 6-7

offline datafiles, 3-4

offline tablespaces, 3-10, 4-8

OPTIMAL setting for SYSTEM rollback segment, A-4

Oracle Data Migration Assistant, 2-8

Oracle Parallel Server, 3-7

rolling upgrade, 1-5

OUTLN user, avoid, 3-10, 4-8

overview of steps, 1-6

parallel execution, 6-3

post-migration actions, 6-1, 10-1

prepare the Oracle7 source database, 3-9, 4-7

preparing to migrate, 2-2

read-only tablespaces, 3-4

role of application developer, 1-10

role of database administrator, 1-10

rollback segments, 3-4

scripts

CATALOG.SQL, 3-4, 3-27

CATPARR.SQL, 3-28

CATPROC.SQL, 3-4, 3-27

CATREP.SQL, 3-28, 4-17

errors while running, 1-3

R0703040.SQL, 3-28, 4-17

rerunning, 3-27, 4-17

U0703040.SQL, 3-27

UTLRP.SQL, 3-28, 4-18

space requirements, 2-15, 3-6, 4-4

system requirements, 2-15

SYSTEM tablespace, A-2

temporary tablespace, A-2

terminology, 1-2

testing, 2-19

testing results, 6-6

troubleshooting, A-1

ALTER DATABASE CONVERT

command, A-16

AUDIT_TRAIL initialization parameter, A-3

database name mismatch, A-19

datafile version integrity, A-20

Migration utility error messages, A-5

missing convert file, A-17

MULTIPLIER option, A-4

NOMOUNT database start mode, A-17

OPTIMAL setting, A-4

- Oracle7 control file, A-16
- password file, A-18
- running the Migration utility, A-2
- running the Oracle Data Migration Assistant, A-2
- SYSTEM tablespace, A-2
- temporary tablespace, A-2
- tuning after, 6-6
- Migration Assistant. See Oracle Data Migration Assistant
- Migration utility
 - advantages and disadvantages, 2-6
 - character set used, 3-8
 - command-line options, 3-14
 - differences from the Oracle Data Migration Assistant, 2-9
 - documentation roadmap, 3-2
 - errors and messages, A-5
 - migrating to a different operating system, 3-7
 - MULTIPLIER option, A-4
 - options
 - CHECK_ONLY, 3-14, 3-18
 - DBNAME, 3-14
 - MULTIPLIER, 3-14
 - NEW_DBNAME, 3-14
 - NLS_NCHAR, 3-14
 - NO_SPACE_CHECK, 3-14
 - PFILE, 3-14
 - SPOOL, 3-14
 - overview, 3-3
 - privileges required, 3-16
 - space required for SYSTEM tablespace, 3-6
 - SYSTEM tablespace, 2-6
 - using, 3-16
- MULTIPLIER
 - Migration utility option, 3-14, A-4
- multithreaded server
 - requirements for running, 8-46
 - shared, 2-17
 - shared and local/remote connections, 2-16
- multi-versioning, 8-50

N

- national character set
 - in Oracle8i, 8-27
- native dynamic SQL
 - compatibility, 8-14
 - interoperability, 8-30
- NCHAR
 - compatibility, 8-27
 - interoperability, 8-27
 - migration, 8-27
 - use in Oracle8i, 8-27
- NCHAR and NLS environment variables
 - compatibility, 8-28
- NCLOBs
 - compatibility, 8-25
- nested tables
 - compatibility, 8-12, 8-29
- Net8
 - connection load balancing, 8-47
 - migrating or upgrading to, 6-4, 8-46
 - service naming, 8-47
 - downgrading, 12-41
- NEVER REFRESH mode
 - materialized views and downgrading, 12-10
- new features of Oracle8i
 - adding after migration, 6-7
- NEW_DBNAME
 - Migration utility option, 3-14
- NLS
 - compatibility, 8-27
 - interoperability, 8-27
 - migration, 8-27
- NLS and NCHAR environment variables
 - compatibility, 8-28
- NLS_LANG environment variable
 - compatibility, 8-28
- NLS_NCHAR
 - Migration utility option, 3-14
- NO_SPACE_CHECK
 - Migration utility option, 3-14
- NOCOPY parameter passing mode
 - compatibility, 8-14
- n-tier authentication and authorization
 - compatibility, 8-13

NUMERIC keyword
behavior in Oracle7 and Oracle8i, 8-22

O

O7_DICTIONARY_ACCESSIBILITY parameter

compatibility, 8-24, B-8

object tables

partitioned

downgrading, 12-16

object-identifiers

user-defined

downgrading, 12-22

OCI

compatibility, 8-17

client notification, 8-9

link line, 8-18

thread safety, 8-18

interoperability, 8-17

Oracle7 clients, 8-18

shared structures, 8-17

OCISessionBegin call, 8-37

OCISessionBegin call, 8-37

upgrading applications to Oracle8i, 9-2

OFA, 1-5

offline datafiles

migration, 3-4

OLON calls

obsolete, 8-18

OLQP, 2-17

OLTP, 2-17

online index (re)build

compatibility, 8-10

online read-only tablespaces

compatibility, 8-9

operating system

migrating to a different, 3-7, 4-5

operators

dropping, 12-14

Optimal Flexible Architecture. See OFA

OPTIMAL setting for SYSTEM rollback segment

migration, A-4

optimization

compatibility, 8-34

extensible optimizer, 8-14

optimizer plan stability, 8-14

optimizer

downgrading, 12-35

plan stability

compatibility, 8-14

options

deinstalling, 1-6

for Migration utility, 3-14

ORA_NLS32 environment variable

compatibility, 8-28

ORA_NLS33 environment variable, 3-16

compatibility, 8-28

Oracle Call Interface. See OCI

Oracle Data Migration Assistant

advantages and disadvantages, 2-8

character set used, 4-6

differences from the Migration utility, 2-9

documentation roadmap, 4-2

migrating to a different operating system, 4-5

Oracle Parallel Server

does not support migration, 2-9, 4-3

does not support upgrading, 7-6

overview, 4-3

running, 4-16, 7-7, 7-12

space required for SYSTEM tablespace, 4-4

Oracle Media Management API

compatibility, 8-13

proxy copy requirement, 8-44

proxy copy and downgrading, 12-37

Oracle Parallel Server

compatibility requirements, 8-34

downgrading, 12-49

instance affinity for jobs

compatibility, 8-12

migration, 3-7

not supported by Oracle Data Migration Assistant, 2-9, 4-3

rolling upgrade, 1-5

upgrading, 7-1, 7-24

not supported by Oracle Data Migration Assistant, 7-6

Oracle7

downgrading to, 13-1

prepare for migration, 3-9, 4-7

upgrading applications to Oracle8i, 9-1

Oracle8i
 changes to initialization parameters, B-1
 new features
 adding after migration, 6-7
 new internal types, E-1
 new SQL functions, E-1
ORLON calls
 obsolete, 8-18
OUTLN user
 avoid, 3-10, 4-8, 7-4
 change password, 6-4, 7-35

P

PARALLEL clause
 DEGREE keyword, 8-35
 INSTANCES keyword removed, 8-35
parallel execution
 allocated from large pool, B-9
 avoiding problems with, 6-3, 7-34
Parallel Server. See Oracle Parallel Server
parameters for Migration utility. See command-line
 options
partially dropped columns
 dropping, 12-8
partitioning
 compatibility, 8-11
 downgrading, 12-14
 removing LOBs, 12-20
password file
 migration
 exclusive setting, A-18
password management
 application changes required for Oracle8i, 8-37
 interoperability, 8-38
 password expiration, 8-38
PFILE
 Migration utility option, 3-14
plan stability
 compatibility, 8-14
 downgrading, 12-36
PL/SQL
 backward compatibility, 8-19
 compatibility, 8-14, 8-20
 functions

 desupported in Oracle8i, 8-30
 interoperability, 8-20
 modules
 recompiling, 7-32
 PLSQL_V2_COMPATIBILITY initialization
 parameter, 8-20
 removing incompatibilities for
 downgrading, 12-25
 variables
 NCHAR and NLS, 8-27
PREBUILT TABLE clause
 materialized views and downgrading, 12-10
precompilers
 compatibility, 8-18
 interoperability, 8-18
 PL/SQL backward compatibility, 8-19
 SQLLIB calls
 relinking, 9-5
 upgrading applications to Oracle8i, 9-2
Pro*Ada
 upgrading to SQL*Module for Ada, 8-19
Pro*C/C++
 connecting with SYSDBA privileges, 8-18
 interoperability with Oracle7, 9-5
Pro*COBOL
 connecting with SYSDBA privileges, 8-19
Procedural Option
 required for migration, 3-9, 4-7
propagation
 removing, 12-32
PROPSS view
 NCHAR and NLS, 8-27
 NCHAR character set, 3-14
proxy copy
 requirement, 8-44

Q

queue tables
 downgrading, 12-31
 upgrading, 7-29

R

R0703040.SQL script
 replication, 3-28, 4-17
read-only tablespaces
 active transactions, 8-23
 compatibility, 8-23
 migration, 3-4
REAL keyword
 behavior in Oracle7 and Oracle8i, 8-22
rebuilding indexes
 physical ROWIDs, 11-8
recovery
 compatibility, 8-13, 8-39
recovery catalog
 PL/SQL packages requirement, 8-39
 upgrading, 7-30
Recovery Manager
 backup management, 8-40
 compatibility, 8-39
 downgrading to 8.0.3
 re-installing packages, 12-52
redo log files
 re-archiving, 8-43
REFRESH ON COMMIT mode
 materialized views and downgrading, 12-9
relinking with SQL*Net, 9-2
remote connections
 multithreaded shared servers, 2-16
REMOTE_LOGIN_PASSWORDFILE initialization
 parameter
 migration, A-18
removing incompatibilities for
 downgrading, 12-23
replication
 compatibility, 8-44
requirements
 export, 5-2
resource manager. See database resource manager
reverse migration
 not supported, 13-2
rollback segments
 migration, 3-4
ROWIDs
 compatibility, client access, 11-7

 conversion to Oracle8i format, 11-3
 examples, 11-5
DBMS_ROWID compatibility package, 11-3
indexes, 11-8
migration
 questions and answers, 11-7
snapshot refresh, 11-6

S

SAVEPOINT keyword
 behavior in Oracle7 and Oracle8i, 8-22
schema objects
 compatibility, 8-10
scripts
 downgrading, 12-45
 errors while running, 1-3
 migrating from Server Manager to
 SQL*Plus, 10-1
 rerunning, 3-27, 4-17, 7-19, 12-46
 upgrading, 7-18
secondary indexes
 dropping from index-organized tables, 12-8
security policies
 downgrading, 12-37
 fine-grained access control, 12-37
Server Manager
 differences with SQL*Plus
 ampersands, 10-12
 blank lines, 10-10
 commands, 10-3
 comments, 10-7
 COMMIT command, 10-14
 CREATE LIBRARY command, 10-13
 CREATE TYPE command, 10-13
 hyphen continuation character, 10-10
 startup, 10-2
 syntax, 10-7
 migrating to SQL*Plus, 10-1
service naming
 Net8, 8-47
SET COMPATIBILITY command
 SQL*Plus scripts, 9-6
SET_SESSION_LONGOPS procedure
 syntax change, 8-33

- shadow processes
 - open cursors and, 2-17
- shared structures
 - interoperability, 8-17
- SHARED_POOL_SIZE initialization parameter
 - changes in release 8.1, 6-3, 7-34
 - parallel execution allocation, B-9
- single-table hash clusters
 - compatibility, 8-10
 - dropping, 12-9
- snapshot refresh
 - physical ROWIDs, 11-6
- snapshots
 - upgrading, 7-25
- snapshots. See Also materialized views
- source database
 - definition, 1-3
- space requirements
 - copying data, 2-15
 - migration, 2-15, 3-6, 4-4
- Spatial
 - compatibility, 8-15
- SPOOL
 - Migration utility option, 3-14
- SQL
 - compatibility, 8-14
 - removing incompatibilities for
 - downgrading, 12-25
- SQL functions
 - new in Oracle8i, E-3
- SQL scripts
 - migrating from Server Manager to
 - SQL*Plus, 10-1
- SQL*Module
 - for Ada, 8-19
- SQL*Net
 - migrating to Net8, 6-4, 8-46
 - relinking, 9-2
 - SQL*Net V1
 - cannot use with Oracle8i, 9-2
 - upgrading from V1 to V2, 8-46
 - use with Oracle8i, 9-2
- SQL*Plus
 - differences with Server Manager
 - ampersands, 10-12
 - blank lines, 10-10
 - commands, 10-3
 - comments, 10-7
 - COMMIT command, 10-14
 - CREATE LIBRARY command, 10-13
 - CREATE TYPE command, 10-13
 - hyphen continuation character, 10-10
 - startup, 10-2
 - syntax, 10-7
 - migrating from Server Manager to, 10-1
 - SQL*Plus scripts
 - upgrading, 9-6
 - SQLLIB calls
 - relinking, 9-5
 - work against Oracle8i, 9-5
 - standby database
 - compatibility, 8-41
 - migrating to Oracle8i, 8-41
 - static data dictionary views
 - added in Oracle8i, C-2
 - changed in Oracle8i
 - added columns, C-7
 - columns that may return NULLs, C-14
 - dropped columns, C-11
 - renamed columns, C-13
 - obsolete in Oracle8i, 8-24, C-15
 - stored outlines
 - downgrading, 12-36
 - summary management
 - compatibility, 8-13
 - SYS schema
 - user-created objects in, 8-24
 - SYSDBA
 - connecting in Pro*C/C++, 8-18
 - connecting in Pro*COBOL, 8-19
 - system requirements
 - for migration, 2-15
 - SYSTEM tablespace
 - Migration utility, 2-6, 3-6
 - Oracle Data Migration Assistant, 4-4
 - space
 - insufficient for migration, A-2

T

- tablespaces
 - compatibility, 8-9
 - migrating offline tablespaces, 3-10, 4-8
- target database
 - definition, 1-3
- tempfiles
 - data dictionary information, 8-23
- temporary tables
 - compatibility, 8-10
 - dropping for downgrading, 12-6
- temporary tablespace
 - space
 - insufficient for migration, A-2
- terminology
 - migration, 1-2
- testing
 - applications for migration, 2-23
 - developing a plan for migration, 2-19
 - EXPLAIN PLAN, 2-22
 - functional for migration, 2-20
 - integration for migration, 2-20
 - INTO clause, 2-22
 - migration results, 6-6
 - minimal for migration, 2-19
 - performance for migration, 2-20
 - pre- and post-migration, 2-21
 - the migrated test database, 2-23
 - the migration process, 2-22
 - volume/load stress for migration, 2-21
- thread safety
 - compatibility, 8-18
- TP monitors, 2-17
- transportable tablespaces
 - compatibility, 8-9
 - removing transported tablespaces, 12-5
- triggers
 - compatibility, 8-14
 - dropped automatically when
 - downgrading, 12-35
 - on nested table view columns
 - downgrading, 12-34
- troubleshooting
 - migration, A-1

- ALTER DATABASE CONVERT
 - command, A-16
- AUDIT_TRAIL initialization parameter, A-3
- database name mismatch, A-19
- datafile version integrity, A-20
- Migration utility error messages, A-5
- missing convert file, A-17
- MULTIPLIER option, A-4
- NOMOUNT database start mode, A-17
- OPTIMAL setting, A-4
- Oracle7 control file, A-16
- password file, A-18
- running the Migration utility, A-2
- running the Oracle Data Migration Assistant, A-2
- SYSTEM tablespace, A-2
- temporary tablespace tablespace, A-2

TRUE keyword

- behavior in Oracle7 and Oracle8i, 8-22

Trusted Oracle

- Export/Import, 5-3

tuning

- after migration, 6-6

Tuxedo applications

- compatibility with Oracle8i XA libraries, 8-17

TWO_TASK environment variable, 3-16

type DTYCHR, 11-8

types

- mutually referencing
 - downgrading, 12-24

U

- U0703040.SQL script, 3-27
- Universal ROWIDs. See UROWIDs
- unused columns
 - dropping, 12-8
- upgrading
 - Advanced Queuing, 7-26
 - Advanced Replication, 7-21
 - applications, 9-1
 - definition, 1-3
 - drop Java objects, 7-36
 - export Java objects, 7-3
 - import Java objects, 7-36

- initialization parameters, 7-15
- manually, 7-13
- Oracle Forms applications, 9-6
- Oracle Parallel Server, 7-1, 7-24
 - rolling upgrade, 1-5
- OUTLN user, avoid, 7-4
- parallel execution, 7-34
- paths, 7-2
- post-upgrading actions, 7-33
- precompiler applications, 9-4
- queue tables, 7-29
- recovery catalog, 7-30
- scripts, 7-18
 - CATALOG.SQL, 7-19
 - CATPARR.SQL, 7-24
 - CATPROC.SQL, 7-19
 - CATREP.SQL, 7-21
 - errors while running, 1-3
 - rerunning, 7-19
 - UTLCONST.SQL script, 7-33
 - UTLRP.SQL, 7-32
- snapshots, 7-25
- specific components, 7-21
- SQL*Plus scripts, 9-6
- steps, 7-3
- supported releases, 7-2
- user-defined datatypes, 7-30
- using the Oracle Data Migration Assistant, 7-7

UROWIDs

- compatibility, 8-11
- downgrading, 12-17
- interoperability, 8-26

user-created objects

- in SYS schema, 8-24

user-defined datatypes

- downgrading, 12-21
- interoperability, 8-29
- new format, 8-29
- upgrading, 7-30

user-defined object identifiers

- compatibility, 8-12
- downgrading, 12-22

UTL_REF package

- downgrading, 12-33
- re-installing after downgrading, 12-51

- UTLCHAIN1.SQL script, 8-31
 - compatibility, 8-14
- UTLCONST.SQL script, 6-2, 7-33
- UTLDST.SQL, 8-6, 12-4
- UTLEXCPT1.SQL script, 8-31
 - compatibility, 8-14
- utljavarm.sql script, 7-36
- UTLRP.SQL script, 3-28, 4-18, 7-32

V

VALUES view

- NCHAR and NLS, 8-27

VARRAYs

- compatibility, 8-12
- removing incompatibilities, 12-24

varying-width character set

- LOBs
 - downgrading, 12-19

version 6

- Export/Import for migration, 5-2

views

- mutually referencing
 - downgrading, 12-10
 - example of, 12-11

Visual Image Retrieval

- compatibility, 8-15

W

word-size

- changing, 1-4, 7-37

X

XA libraries

- compatibility, 8-17

