2

Creating an Oracle Database

This chapter discusses the process of creating an Oracle database, and contains the following topics:

- Considerations Before Creating a Database
- Using the Database Configuration Assistant
- Manually Creating an Oracle Database
- Understanding the CREATE DATABASE Statement
- Troubleshooting Database Creation
- Dropping a Database
- Considerations After Creating a Database
- Initialization Parameters and Database Creation
- Managing Initialization Parameters Using a Server Parameter File

See Also:

- Chapter 3, "Using Oracle-Managed Files" for information about creating a database whose underlying operating system files are automatically created and managed by the Oracle database server
- *Oracle9i Real Application Clusters Setup and Configuration* for additional information specific to an Oracle Real Application Clusters environment

Considerations Before Creating a Database

Database creation prepares several operating system files to work together as an Oracle database. You need only create a database once, regardless of how many datafiles it has or how many instances access it. Creating a database can also erase information in an existing database and create a new database with the same name and physical structure.

The following topics can help prepare you for database creation.

- Planning for Database Creation
- Meeting Creation Prerequisites
- Deciding How to Create an Oracle Database

Planning for Database Creation

Prepare to create the database by research and careful planning. The following are some recommended actions:

Action	For more information...
■ Plan the database tables and indexes and estimate the amount of space they will require.	Part II, "Oracle Server Processes and Storage Structure" Part III, "Schema Objects"
■ Plan the layout of the underlying operating system files that are to comprise your database. Proper distribution of files can improve database performance dramatically by distributing the I/O for accessing the files. There are several ways to distribute I/O when you install Oracle and create your database. For example, placing redo log files on separate disks or striping; placing datafiles to reduce contention; and controlling density of data (number of rows to a data block).	*Oracle9i Database Performance Tuning Guide and Reference* Your Oracle operating system specific documentation
■ Consider using the Oracle Managed Files feature to create and manage the operating system files that comprise your database storage. This feature eases their administration.	Chapter 3, "Using Oracle-Managed Files"
■ Select the **global database name**, which is the name and location of the database within the network structure. Create the global database name by setting both the `DB_NAME` and `DB_DOMAIN` initialization parameters.	"Determining the Global Database Name" on page 2-35

Action	For more information...
■ Familiarize yourself with the initialization parameters that comprise the initialization parameter file. Become familiar with the concept and operation of a **server parameter file**. A server parameter file allows you to store and manage your initialization parameters persistently in a server-side disk file.	"Initialization Parameters and Database Creation" on page 2-34 "What is a Server Parameter File?" on page 2-44 *Oracle9i Database Reference*
■ Select the database character set. All character data, including data in the data dictionary, is stored in the database character set. You must specify the database character set when you create the database. If clients using different character sets will access the database, then choose a superset that includes all client character sets. Otherwise, character conversions may be necessary at the cost of increased overhead and potential data loss. You can also specify an alternate character set.	*Oracle9i Database Globalization Support Guide*
■ Consider what time zones your database must support. Oracle uses a time zone file, located in the Oracle home directory, as the source of valid time zones. If you determine that you need to use a time zone that is not in the default time zone file (`timezone.dat`), but that is present in the larger time zone file (`timezlrg.dat`), then you must set the `ORA_TZFILE` environment variable to point to the larger file.	"Specifying the Database Time Zone File" on page 2-28 *Oracle9i Database Globalization Support Guide*
■ Select the standard database block size. This is specified at database creation by the `DB_BLOCK_SIZE` initialization parameter and cannot be changed after the database is created. The `SYSTEM` tablespace and most other tablespaces use the standard block size. Additionally, you can specify up to four non-standard block sizes when creating tablespaces.	"Specifying Database Block Sizes" on page 2-37
■ Use an undo tablespace to manage your undo records, rather than rollback segments.	Chapter 13, "Managing Undo Space"

Action	For more information...
■ Develop a backup and recovery strategy to protect the database from failure. It is important to protect the control file by multiplexing, to choose the appropriate backup mode, and to manage the online and archived redo logs.	Chapter 7, "Managing the Online Redo Log" Chapter 8, "Managing Archived Redo Logs" Chapter 6, "Managing Control Files" *Oracle9i Backup and Recovery Concepts*
■ Familiarize yourself with the principles and options of starting up and shutting down an instance and mounting and opening a database.	Chapter 4, "Starting Up and Shutting Down"

Meeting Creation Prerequisites

To create a new database, the following prerequisites must be met:

- The desired Oracle software is installed. This includes setting up various environment variables unique to your operating system and establishing the directory structure for software and database files.
- You have the operating system privileges associated with a fully operational database administrator. You must be specially authenticated by your operating system or through a password file, allowing you to start up and shut down an instance before the database is created or opened. This authentication is discussed in "Database Administrator Authentication" on page 1-13.
- There is sufficient memory available to start the Oracle instance.
- There is sufficient disk storage space for the planned database on the computer that executes Oracle.

All of these are discussed in the Oracle installation guide specific to your operating system. Additionally, the Oracle Universal Installer will guide you through your installation and provide help in setting up environment variables, directory structure, and authorizations.

Deciding How to Create an Oracle Database

Creating a database includes the following operations:

- Creating information structures, including the data dictionary, that Oracle requires to access and use the database

- Creating and initializing the control files and redo log files for the database
- Creating new datafiles or erasing data that existed in previous datafiles

You use the CREATE DATABASE statement to perform these operations, but other actions are necessary before you have an operational database. A few of these actions are creating users and temporary tablespaces, building views of the data dictionary tables, and installing Oracle built-in packages. This is why the database creation process involves executing prepared scripts. But, you do not necessarily have to prepare this script yourself.

You have the following options for creating your new Oracle database:

- Use the Database Configuration Assistant (DBCA).

  DBCA can be launched by the Oracle Universal Installer, depending upon the type of install that you select, and provides a graphical user interface (GUI) that guides you through the creation of a database. You can chose not to use DBCA, or you can launch it as a standalone tool at any time in the future to create a database. See "Using the Database Configuration Assistant" on page 2-5.

- Create the database manually from a script.

  If you already have existing scripts for creating your database, you can still create your database manually. However, consider editing your existing script to take advantage of new Oracle features. Oracle provides a sample database creation script and a sample initialization parameter file with the database software files it distributes, both of which can be edited to suit your needs. See "Manually Creating an Oracle Database" on page 2-14.

- Upgrade an existing database.

  If you are already using a previous release of Oracle, database creation is required only if you want an entirely new database. You can upgrade your existing Oracle database and use it with the new release of the Oracle software. Database upgrades are not discussed in this book. The *Oracle9i Database Migration* manual contains information about upgrading an existing Oracle database.

Using the Database Configuration Assistant

The Database Configuration Assistant (DBCA) an Oracle supplied tool that enables you to create an Oracle database, configure database options for an existing Oracle database, delete an Oracle database, or manage database templates. DBCA is launched automatically by the Oracle Universal Installer, but it can be invoked

standalone from the Windows operating system start menu (under Configuration Assistants) or by entering the following on the UNIX command line:

```
dbca
```

DBCA can be run in three modes:

Mode	Description
Interactive	This is the default mode if you do not specify any parameters. This mode presents a wizard like GUI interface and provides complete DBCA functionality. Online help is provided.
Progress Only	This mode is typically used by other tools to create a database. For example, it is used by the Oracle Universal Installer, the Enterprise Manager Configuration Assistant, and the Oracle Internet Directory Configuration Assistant. In this mode, only a progress bar is displayed, and it is used when creating databases and templates.
Silent	This mode has only a command-line interface where parameters are specified. There is no other user interaction. Informational, error, and warning message are written to a log file. You specify the template of your choice, for customization or for the creation of a database.

DBCA can be used to create single instance databases, or it can be used to create or add instances in an Oracle Real Application Clusters environment.

This section primarily describes the use of DBCA in interactive mode. It contains the following topics:

- Advantages of Using DBCA
- Creating a Database Using DBCA
- Configuring Database Options
- Deleting a Database Using DBCA
- Managing DBCA Templates
- Using DBCA Silent Mode

Advantages of Using DBCA

These are a few of the advantages of using DBCA:

- You can use its wizards to guide you through a selection of options providing an easy means of creating and tailoring your database. It allows you to provide varying levels of detail. You can provide a minimum of input and allow Oracle to make decisions for you, eliminating the need to spend time deciding how best to set parameters or structure the database. Optionally, it allows you to be very specific about parameter settings and file allocations.
- It builds efficient and effective databases that take advantage of Oracle's new features.
- It uses Optimal Flexible Architecture (OFA), whereby database files and administrative files, including initialization files, follow standard naming and placement practices.

Creating a Database Using DBCA

DBCA enables you to create a database from predefined templates provided by Oracle or from templates that you or others have created. A template is a description of a database. Templates are described in more detail in "Managing DBCA Templates" on page 2-9.

Selecting the Template

DBCA displays the templates that are available, which includes templates that Oracle ships with the DBCA product. These templates are described in "DBCA Templates Provided by Oracle" on page 2-11. If you or others have created templates, those will be displayed also. You select the appropriate template for the database that you want to create. Clicking the "Show Details..." button displays specific information about the database defined by a template.

Including Datafiles

When you select a template, you also specify whether the database definition is to include datafiles. This determines whether you use a seed template (includes datafiles), or a non-seed template (does not include datafiles), to create your database.

Specifying Global Database Name and Database Features

The next page that DBCA displays enables you provide a global database name and SID.

Specifying Database Features

The "Database Features" page is presented only when you select a non-seed template. It enables you to include optional database features.

The following is a representative list of Oracle features that you can install in your database. Some of the listed options might already be included depending upon the database template that you selected. Those options that are already installed are noted as such (grayed out).

- Oracle Spatial
- Oracle Ultra Search
- Oracle Label Security
- Oracle Data Mining
- Oracle OLAP Services
- Sample Schemas

You can also display a list of standard database features. These are features that Oracle recommends you always install, but you have the option of excluding them. These include:

- Oracle JVM
- Oracle Text
- Oracle *inter*Media
- XDB Protocol

Specifying Mode, Initialization Parameters, and Datafiles

The next pages enable you to further define your database. You specify mode (dedicated server of shared server), set initialization parameters, and specify datafile locations. Oracle can determine specific values for you based upon your description of the database you are trying to create. For example, Oracle can choose appropriate settings for SGA memory sizing parameters depending upon whether you select a typical or custom database.

Completing Database Creation

After you have completed the specification of the parameters that define your database you can:

- Create the database now
- Save the description as a database template
- Generate database creation scripts

If you choose to generate scripts, you can use them to create the database later without using DBCA, or you can use them as a checklist

Configuring Database Options

When you elect to configure database options, you can add Oracle options that have not previously been configured for use with your database. This provides you the opportunity to add options and features that you did not include when you created the database. These options are discussed in "Specifying Database Features" on page 2-8.

Deleting a Database Using DBCA

DBCA enables you to delete a database. When you do so, you delete the database instance and its control file(s), redo log files, and datafiles. Any server parameter file (SPFILE) or initialization parameter file used by the database is also deleted.

Managing DBCA Templates

DBCA templates are XML files that contain information required to create a database. Templates are used in DBCA to create new databases and make clones of existing databases. The information in templates includes database options, initialization parameters, and storage attributes (for datafiles, tablespaces, control files and redo logs).

Templates can be used just like scripts, and they can be used in silent mode. But they are more powerful than scripts, because you have the option of cloning a database. This saves time in database creation, because copying an already created seed database's files to the correct locations takes less time than creating them as new.

Templates are stored in the following directory:

```
$ORACLE_HOME/assistants/dbca/templates
```

Advantages of Using Templates

The following are some of the advantages of using templates:

- They save you time. If you use a template you do not have to define the database.
- By creating a template containing your database settings, you can easily create a duplicate database without specifying parameters twice.
- They are easily edited. You can quickly change database options from the template settings.
- Templates are easy to share. They can be copied from one machine to another.

Types of Templates

There are two types of templates:

- Seed templates
- Non-seed templates

The characteristics of each are shown in the following table:

Type	File Extension	Include Datafiles?	Database Structure
Seed	.dbc	Yes	This type of template contains both the structure and the physical datafiles of an existing (seed) database. When you select a seed template, database creation is faster because the physical files and schema of the database have already been created. Your database starts as a copy of the seed database, rather than having to be built. You can change only the following: ■ Name of the database ■ Destination of the datafiles ■ Number control files ■ Number redo log groups ■ Initialization parameters Other changes can be made after database creation using custom scripts that can be invoked by DBCA, command line SQL statements, or the Oracle Enterprise Manager. The datafiles and redo logs for the seed database are stored in zipped format in another file with a `.dfj` extension. Usually the corresponding `.dfj` file of a `.dbc` file has the same file name, but this is not a requirement since the corresponding `.dfj` file's location is stored in the `.dbc` file.
Non-seed	.dbt	No	This type of template is used to create a new database from scratch. It contains the characteristics of the database to be created. Seed templates are more flexible than their seed counterparts because all datafiles and redo logs are created to your specification (not copied), and names, sizes, and other attributes can be changed as required.

DBCA Templates Provided by Oracle

Oracle provides templates for the following environments:

Environment	Description of Environment
DSS (Data Warehousing)	Users perform numerous, complex queries that process large volumes of data. Response time, accuracy, and availability are key issues. These queries (typically read-only) range from a simple fetch of a few records to numerous complex queries that sort thousands of records from many different tables.
OLTP (Online Transaction Processing)	Many concurrent users performing numerous transactions requiring rapid access to data. Availability, speed, concurrence, and recoverability are key issues. Transactions consist of reading (`SELECT` statements), writing (`INSERT` and `UPDATE` statements), and deleting (`DELETE` statements) data in database tables.
General Purpose	This template creates a database designed for general purpose usage. It combines features of both the DSS and OLTP database templates.
New Database	This template allows you maximum flexibility in defining a database.

Creating Templates Using DBCA

The "Template Management" page provides you with three options that enable you to modify existing templates or to create your own custom templates. Your choices are:

- From an existing template

  Using an existing template, you can create a new template based on the pre-defined template settings. You can add or change any template settings such as initialization parameters, storage parameters, or use custom scripts.

- From an existing database (structure only)

  You can create a new template that contains structural information about an existing database, including database options, tablespaces, datafiles, and initialization parameters specified in the source database. User defined schema and their data *will not* be part of the created template. The source database can be either local or remote.

- From an existing database (structure as well as data--a seed database)

  You can create a new template that has both the structural information and physical datafiles of an existing database. Databases created using such a

template are identical to the source database. User defined schema and their data *will* be part of the created template. The source database must be local.

Oracle saves templates as XML files.

While creating templates from existing databases, you can optionally choose to translate file paths into OFA (Optimal Flexible Architecture) or maintain existing file paths. OFA is recommended if the machine on which you plan to create the database using the template has a different directory structure. Non-OFA can be used if the target machine has a similar directory structure.

Deleting DBCA Templates

The "Template Management" page also allows you to delete existing templates.

Using DBCA Silent Mode

Silent mode does not have any user interface (other than what you initially input on the command line) or user interaction. It outputs all messages including information, errors, and warnings into a log file.

From the command line enter the following command to see all of the DBCA options that are available when using silent mode:

```
dbca -help
```

The following sections contain examples that illustrate the use of silent mode.

DBCA Silent Mode Example 1: Creating a Clone Database

To create a clone database, enter the following on the command line:

```
% dbca -silent -createDatabase -templateName Transaction_Processing.dbc
-gdbname ora9i -sid ora9i -datafileJarLocation
/private/oracle9i/ora9i/assistants/dbca/templates -datafileDestination
/private/oracle9i/ora9i/oradata -responseFile NO_VALUE
-characterset WE8ISO8859P1
```

DBCA Silent Mode Example 2: Creating a Seed Template

To create a seed template, enter the following on the command line:

```
% dbca -silent -createCloneTemplate -sourceDB ora9I -sysDBAUserName
sys -sysDBAPassword change_on_install -templateName copy_of_ora9i.dbc
-datafileJarLocation /private/oracle/ora9i/assistants/dbca/templates
```

Manually Creating an Oracle Database

This section presents the steps involved when you create a database manually. These steps should be followed in the order presented. You will previously have created your environment for creating your Oracle database, including most operating system dependent environmental variables, as part of the Oracle software installation process.

The examples shown in these steps are to create the database `mynewdb`.

Note: At this point, you may not be familiar with all of the initialization parameters and database structures discussed in this section. These steps contain many cross references to other parts of this book to allow you to learn about and understand these parameters and structures.

Step 1: Decide on Your Instance Identifier (SID)

Decide on a unique Oracle system identifier (SID) for your instance and set the `ORACLE_SID` environment variable accordingly. This identifier is used to avoid confusion with other Oracle instances that you may create later and run concurrently on your system.

The following example sets the SID for the instance and database we are about to create:

```
% setenv ORACLE_SID mynewdb
```

The value of the DB_NAME initialization parameter should match the SID setting.

Step 2: Establish the Database Administrator Authentication Method

You must be authenticated and granted appropriate system privileges in order to create a database. You can use the password file or operating system authentication method. Database administrator authentication and authorization is discussed in the following sections of this book:

- "Database Administrator Security and Privileges" on page 1-10
- "Database Administrator Authentication" on page 1-13
- "Creating and Maintaining a Password File" on page 1-20

Step 3: Create the Initialization Parameter File

The instance (System Global Area and background processes) for any Oracle database is started using an initialization parameter file. One way of getting started on your initialization parameter file is to edit a copy of the sample initialization parameter file that Oracle provides on the distribution media, or the sample presented in this book.

For ease of operation, store your initialization parameter file in Oracle's default location, using the default name. That way, when you start your database, it is not necessary to specify the PFILE parameter because Oracle automatically looks in the default location for the initialization parameter file.

Default parameter file locations are shown in the following table:

Platform	Default Name	Default Location

UNIX	`init$ORACLE_SID.ora`
	For example, the initialization parameter file for the mynewdb database is named:
	`initmynewdb.ora`
		`$ORACLE_HOME/dbs`
		For example, the initialization parameter file for the `mynewdb` database is stored in the following location:
		`/vobs/oracle/dbs/initmynewdb.ora`

Windows	`init$ORACLE_SID.ora`	`$ORACLE_HOME\database`

The following is the initialization parameter file used to create the `mynewdb` database.

Sample Initialization Parameter File

```
# Cache and I/O
DB_BLOCK_SIZE=4096
DB_CACHE_SIZE=20971520

# Cursors and Library Cache
CURSOR_SHARING=SIMILAR
OPEN_CURSORS=300

# Diagnostics and Statistics
BACKGROUND_DUMP_DEST=/vobs/oracle/admin/mynewdb/bdump
CORE_DUMP_DEST=/vobs/oracle/admin/mynewdb/cdump
TIMED_STATISTICS=TRUE
USER_DUMP_DEST=/vobs/oracle/admin/mynewdb/udump

# Control File Configuration
CONTROL_FILES=("/vobs/oracle/oradata/mynewdb/control01.ctl",
               "/vobs/oracle/oradata/mynewdb/control02.ctl",
               "/vobs/oracle/oradata/mynewdb/control03.ctl")

# Archive
LOG_ARCHIVE_DEST_1='LOCATION=/vobs/oracle/oradata/mynewdb/archive'
LOG_ARCHIVE_FORMAT=%t_%s.dbf
LOG_ARCHIVE_START=TRUE

# Shared Server
# Uncomment and use first DISPATCHES parameter below when your listener is
# configured for SSL
# (listener.ora and sqlnet.ora)
# DISPATCHERS = "(PROTOCOL=TCPS)(SER=MODOSE)",
#               "(PROTOCOL=TCPS)(PRE=oracle.aurora.server.SGiopServer)"
DISPATCHERS="(PROTOCOL=TCP)(SER=MODOSE)",
            "(PROTOCOL=TCP)(PRE=oracle.aurora.server.SGiopServer)",
             (PROTOCOL=TCP)

# Miscellaneous
COMPATIBLE=9.2.0
DB_NAME=mynewdb

# Distributed, Replication and Snapshot
DB_DOMAIN=us.oracle.com
REMOTE_LOGIN_PASSWORDFILE=EXCLUSIVE

# Network Registration
INSTANCE_NAME=mynewdb
```

```
# Pools
JAVA_POOL_SIZE=31457280
LARGE_POOL_SIZE=1048576
SHARED_POOL_SIZE=52428800

# Processes and Sessions
PROCESSES=150

# Redo Log and Recovery
FAST_START_MTTR_TARGET=300

# Resource Manager
RESOURCE_MANAGER_PLAN=SYSTEM_PLAN

# Sort, Hash Joins, Bitmap Indexes
SORT_AREA_SIZE=524288

# Automatic Undo Management
UNDO_MANAGEMENT=AUTO
UNDO_TABLESPACE=undotbs
```

See Also:

- "Initialization Parameters and Database Creation" on page 2-34 for more information on some of these parameters and other initialization parameters that you decide to include

Step 4: Connect to the Instance

Start SQL*Plus and connect to your Oracle instance AS SYSDBA.

```
$ SQLPLUS /nolog
CONNECT SYS/password AS SYSDBA
```

Step 5: Start the Instance.

Start an instance without mounting a database. Typically, you do this only during database creation or while performing maintenance on the database. Use the STARTUP command with the NOMOUNT option. In this example, because the initialization parameter file is stored in the default location, you are not required to specify the PFILE clause:

```
STARTUP NOMOUNT
```

At this point, there is no database. Only the SGA is created and background processes are started in preparation for the creation of a new database.

See Also:

- "Managing Initialization Parameters Using a Server Parameter File" on page 2-44
- Chapter 4, "Starting Up and Shutting Down" to learn how to use the STARTUP command

Step 6: Issue the CREATE DATABASE Statement

To create the new database, use the CREATE DATABASE statement. The following statement creates database mynewdb:

```
CREATE DATABASE mynewdb
   USER SYS IDENTIFIED BY pz6r58
   USER SYSTEM IDENTIFIED BY y1tz5p
   LOGFILE GROUP 1 ('/vobs/oracle/oradata/mynewdb/redo01.log') SIZE 100M,
           GROUP 2 ('/vobs/oracle/oradata/mynewdb/redo02.log') SIZE 100M,
           GROUP 3 ('/vobs/oracle/oradata/mynewdb/redo03.log') SIZE 100M
   MAXLOGFILES 5
   MAXLOGMEMBERS 5
   MAXLOGHISTORY 1
   MAXDATAFILES 100
   MAXINSTANCES 1
   CHARACTER SET US7ASCII
   NATIONAL CHARACTER SET AL16UTF16
   DATAFILE '/vobs/oracle/oradata/mynewdb/system01.dbf' SIZE 325M REUSE
   EXTENT MANAGEMENT LOCAL
   DEFAULT TEMPORARY TABLESPACE tempts1
      DATAFILE '/vobs/oracle/oradata/mynewdb/temp01.dbf'
      SIZE 20M REUSE
   UNDO TABLESPACE undotbs
      DATAFILE '/vobs/oracle/oradata/mynewdb/undotbs01.dbf'
      SIZE 200M REUSE AUTOEXTEND ON NEXT 5120K MAXSIZE UNLIMITED;
```

A database is created with the following characteristics:

- The database is named mynewdb. Its global database name is mynewdb.us.oracle.com. See "DB_NAME Initialization Parameter" and "DB_DOMAIN Initialization Parameter" on page 2-36.
- Three control files are created as specified by the CONTROL_FILES initialization parameter. See "Specifying Control Files" on page 2-36.

- The password for user SYS is `pz6r58` and the password for SYSTEM is `y1tz5p`. These two clauses that specify the passwords for SYS and SYSTEM are not mandatory in this release of Oracle9*i*. However, if you specify either clause, you must specify both clauses. For further information about the use of these clauses, see "Protecting Your Database: Specifying Passwords for Users SYS and SYSTEM" on page 2-23.
- The new database has three online redo log files as specified in the LOGFILE clause. MAXLOGFILES, MAXLOGMEMBERS, and MAXLOGHISTORY define limits for the redo log. See Chapter 7, "Managing the Online Redo Log".
- MAXDATAFILES specifies the maximum number of datafiles that can be open in the database. This number affects the initial sizing of the control file.

  Note: You can set several limits during database creation. Some of these limits are also subject to superseding limits of the operating system and can be affected by them. For example, if you set MAXDATAFILES, Oracle allocates enough space in the control file to store MAXDATAFILES filenames, even if the database has only one datafile initially. However, because the maximum control file size is limited and operating system dependent, you might not be able to set all CREATE DATABASE parameters at their theoretical maximums.

  For more information about setting limits during database creation, see the *Oracle9i SQL Reference* and your operating system specific Oracle documentation.

- MAXINSTANCES specifies that only one instance can have this database mounted and open.
- The US7ASCII character set is used to store data in this database.
- The AL16UTF16 character set is specified as the NATIONAL CHARACRTER SET, used to store data in columns specifically defined as NCHAR, NCLOB, or NVARCHAR2.
- The SYSTEM tablespace, consisting of the operating system file `/vobs/oracle/oradata/mynewdb/system01.dbf`, is created as specified by the DATAFILE clause. If the file already exists, it is overwritten.
- The SYSTEM tablespace is a locally managed tablespace. See "Creating a Locally Managed SYSTEM Tablespace" on page 2-26.

- The DEFAULT_TEMPORARY_TABLESPACE clause creates and names a default temporary tablespace for this database. See "Creating a Default Temporary Tablespace" on page 2-24.
- The UNDO_TABLESPACE clause creates and names an undo tablespace to be used to store undo records for this database if you have specified UNDO_MANAGEMENT=AUTO in the initialization parameter file. See "Using Automatic Undo Management: Creating an Undo Tablespace" on page 2-24.
- Because the ARCHIVELOG clause is not specified in this CREATE DATABASE statement, redo log files will not initially be archived. This is customary during database creation and an ALTER DATABASE statement can be used later to switch to ARCHIVELOG mode. The initialization parameters in the initialization parameter file for mynewdb relating to archiving are LOG_ARCHIVE_DEST_1, LOG_ARCHIVE_FORMAT, and LOG_ARCHIVE_START. See Chapter 8, "Managing Archived Redo Logs".

See Also:

- "Understanding the CREATE DATABASE Statement" on page 2-22
- *Oracle9i SQL Reference* for more information about specifying the clauses and parameter values for the CREATE DATABASE statement

Step 7: Create Additional Tablespaces

To make the database functional, you need to create additional files and tablespaces for users. The following sample script creates some additional tablespaces:

```
CONNECT SYS/password AS SYSDBA
-- create a user tablespace to be assigned as the default tablespace for users
CREATE TABLESPACE users LOGGING
     DATAFILE '/vobs/oracle/oradata/mynewdb/users01.dbf'
     SIZE 25M REUSE AUTOEXTEND ON NEXT  1280K MAXSIZE UNLIMITED
     EXTENT MANAGEMENT LOCAL;
-- create a tablespace for indexes, separate from user tablespace
CREATE TABLESPACE indx LOGGING
     DATAFILE '/vobs/oracle/oradata/mynewdb/indx01.dbf'
     SIZE 25M REUSE AUTOEXTEND ON NEXT  1280K MAXSIZE UNLIMITED
     EXTENT MANAGEMENT LOCAL;
EXIT
```

For information about creating tablespaces, see Chapter 11, "Managing Tablespaces".

Step 8: Run Scripts to Build Data Dictionary Views

Run the scripts necessary to build views, synonyms, and PL/SQL packages:

```
CONNECT SYS/password AS SYSDBA
@/vobs/oracle/rdbms/admin/catalog.sql
@/vobs/oracle/rdbms/admin/catproc.sql
EXIT
```

The following table contains descriptions of the scripts:

Script	Description
CATALOG.SQL	Creates the views of the data dictionary tables, the dynamic performance views, and public synonyms for many of the views. Grants PUBLIC access to the synonyms.
CATPROC.SQL	Runs all scripts required for or used with PL/SQL.

You may want to run other scripts. The scripts that you run are determined by the features and options you choose to use or install. Many of the scripts available to you are described in the *Oracle9i Database Reference*.

See your Oracle installation guide for your operating system for the location of these scripts.

Step 9: Run Scripts to Install Additional Options (Optional)

If you plan to install other Oracle products to work with this database, see the installation instructions for those products. Some products require you to create additional data dictionary tables. Usually, command files are provided to create and load these tables into the database's data dictionary.

See your Oracle documentation for the specific products that you plan to install for installation and administration instructions.

Step 10: Create a Server Parameter File (Recommended)

Oracle recommends you create a server parameter file as a dynamic means of maintaining initialization parameters. The server parameter file is discussed in "Managing Initialization Parameters Using a Server Parameter File" on page 2-44.

The following script creates a server parameter file from the text initialization parameter file and writes it to the default location. The instance is shut down, then restarted using the server parameter file (in the default location).

```
CONNECT SYS/password AS SYSDBA
-- create the server parameter file
CREATE SPFILE='/vobs/oracle/dbs/spfilemynewdb.ora' FROM
       PFILE='/vobs/oracle/admin/mynewdb/scripts/init.ora';
SHUTDOWN
-- this time you will start up using the server parameter file
CONNECT SYS/password AS SYSDBA
STARTUP
EXIT
```

Step 11: Back Up the Database.

You should make a full backup of the database to ensure that you have a complete set of files from which to recover if a media failure occurs. For information on backing up a database, see *Oracle9i Backup and Recovery Concepts.*

Understanding the CREATE DATABASE Statement

When you execute a CREATE DATABASE statement, Oracle performs (at least) the following operations. The actual operations performed are in large part determined by the clauses that you specify in the CREATE DATABASE statement or initialization parameters that you have set.

- Creates the datafiles for the database
- Creates the control files for the database
- Creates the redo log files for the database and establishes the ARCHIVELOG mode.
- Creates the SYSTEM tablespace and the SYSTEM rollback segment
- Creates the data dictionary
- Sets the character set that stores data in the database
- Sets the database time zone
- Mounts and opens the database for use

This section discusses several of the clauses of the CREATE DATABASE statement. It expands upon some of the clauses discussed in "Step 6: Issue the CREATE DATABASE Statement" on page 2-18 and introduces additional ones.

The following topics are contained in this section:

- Protecting Your Database: Specifying Passwords for Users SYS and SYSTEM
- Clauses that Simplify Database Creation and Management
- Creating a Locally Managed SYSTEM Tablespace
- Specifying the Database Time Zone and Time Zone File
- Specifying FORCE LOGGING Mode

Protecting Your Database: Specifying Passwords for Users SYS and SYSTEM

The clauses of the CREATE DATABASE statement used for specifying the passwords for users SYS and SYSTEM are:

- `USER SYS IDENTIFIED BY` *`password`*
- `USER SYSTEM IDENTIFIED BY` *`password`*

If not specified, these users are assigned the default passwords `change_on_install` and `manager`, respectively. A record is written to the alert file indicating that the default passwords were used. To protect your database, you should change these passwords using the ALTER USER statement after database creation.

While these clauses are optional in this Oracle release, Oracle strongly recommends that you specify them. The default passwords are commonly known, and if you neglect to change them later, you leave yourself vulnerable to attack by malicious users.

See Also: "Some Security Considerations" on page 2-32

Clauses that Simplify Database Creation and Management

In addition to using the Database Configuration Assistant for creating your database, Oracle9*i* offers you other options that can simplify the creation, operation, and management of your database. These options, and their associated CREATE DATABASE clauses, are discussed briefly in the following sections, and in more detail in later sections of this book:

- Using Automatic Undo Management: Creating an Undo Tablespace
- Creating a Default Temporary Tablespace
- Using Oracle-Managed Files

Using Automatic Undo Management: Creating an Undo Tablespace

Oracle recommends that instead of using rollback segments in your database, you use an undo tablespace. This requires the use of a different set of initialization parameters, and optionally, the inclusion of the UNDO TABLESPACE clause in your CREATE DATABASE statement.

You must include the following initialization parameter if you want to operate your database in automatic undo management mode:

```
UNDO_MANAGEMENT=AUTO
```

In this mode, rollback information, referred to as **undo**, is stored in an undo tablespace rather than rollback segments and is managed by Oracle. If you want to create and name a specific tablespace for the undo tablespace, you can include the UNDO TABLESPACE clause at database creation time. If you omit this clause, and automatic undo management is specified, Oracle creates a default undo tablespace named SYS_UNDOTBS.

See Also:

- "Specifying the Method of Undo Space Management" on page 2-42
- Chapter 13, "Managing Undo Space" for information about the creation and use of undo tablespaces

Creating a Default Temporary Tablespace

The DEFAULT TEMPORARY TABLESPACE clause of the CREATE DATABASE statement specifies that a temporary tablespace is to be created at database creation time. This tablespace is used as the default temporary tablespace for users who are not otherwise assigned a temporary tablespace.

Users can be explicitly assigned a default temporary tablespace in the CREATE USER statement. But, if no temporary tablespace is specified, they default to using the SYSTEM tablespace. It is not good practice to store temporary data in the SYSTEM tablespace. To avoid this problem, and to avoid the need to assign every user a default temporary tablespace at CREATE USER time, you can use the DEFAULT TEMPORARY TABLESPACE clause of CREATE DATABASE.

If you decide later to change the default temporary tablespace, or to create an initial one after database creation, you can do so. You do this by creating a new temporary tablespace (CREATE TEMPORARY TABLESPACE), then assign it as the temporary tablespace using the ALTER DATABASE DEFAULT TEMPORARY TABLESPACE

statement. Users will automatically be switched (or assigned) to the new temporary default tablespace.

The following statement assigns a new default temporary tablespace:

```
ALTER DATABASE DEFAULT TEMPORARY TABLESPACE tempts2;
```

The new default temporary tablespace must be an existing temporary tablespace. When using a locally managed SYSTEM tablespace, the new default temporary tablespace must also be locally managed.

You cannot drop a default temporary tablespace, but you can assign a new default temporary tablespace, then drop the former one. You are not allowed to change a default temporary tablespace to a permanent tablespace, nor can you take a default temporary tablespace offline.

Users can obtain the name of the current default temporary tablespace using the DATABASE_PROPERTIES view. The PROPERTY_NAME column contains the value "DEFAULT_TEMP_TABLESPACE" and the PROPERTY_VALUE column contains the default temporary tablespace name.

See Also:

- *Oracle9i SQL Reference* for the syntax of the DEFAULT TEMPORARY TABLESPACE clause of CREATE DATABASE and ALTER DATABASE
- "Temporary Tablespaces" on page 11-12 for information about creating and using temporary tablespaces

Using Oracle-Managed Files

You can minimize the number of clauses and parameters that you specify in your CREATE DATABASE statement by using the Oracle Managed Files feature.

If you include the DB_CREATE_FILE_DEST or DB_CREATE_ONLINE_LOG_DEST_*n* initialization parameters in your initialization parameter file, you enable Oracle to create and manage the underlying operating system files of your database. Oracle will automatically create and manage the operating system files for the following database structures, dependent upon the initialization parameters you specify and how you specify clauses in your CREATE DATABASE statement:

- Tablespaces
- Temporary tablespaces
- Control files

- Online redo log files

Briefly, this is how the Oracle Managed Files feature works, using the following CREATE DATABASE statement as an example:

```
CREATE DATABASE rbdb1
     USER SYS IDENTIFIED BY pz6r58
     USER SYSTEM IDENTIFIED BY y1tz5p
     UNDO TABLESPACE undotbs
     DEFAULT TEMPORARY TABLESPACE tempts1;
```

- No DATAFILE clause is specified, therefore Oracle creates an Oracle-managed datafile for the SYSTEM tablespace.
- No LOGFILE clauses are included, therefore Oracle creates two online redo log file groups that are Oracle managed.
- No DATAFILE subclause is specified for the UNDO TABLESPACE clause, therefore Oracle creates an Oracle-managed datafile for the undo tablespace.
- No TEMPFILE subclause is specified for the DEFAULT TEMPORARY TABLESPACE clause, therefore Oracle creates an Oracle-managed tempfile.
- Additionally, if no CONTROL_FILES initialization parameter is specified in the initialization parameter file, Oracle creates an Oracle-managed control file.
- If using a server parameter file (see "Managing Initialization Parameters Using a Server Parameter File" on page 2-44) the initialization parameters are set accordingly and automatically.

> **See Also:** Chapter 3, "Using Oracle-Managed Files" for a complete description of the Oracle Managed Files feature and how to use it

Creating a Locally Managed SYSTEM Tablespace

When you specify the EXTENT MANAGEMENT LOCAL clause in the CREATE DATABASE statement, you cause Oracle to create a locally managed SYSTEM tablespace wherein Oracle determines extent sizes. The COMPATIBLE initialization parameter must be set to 9.2 or higher for this statement to be successful. If you do not specify the EXTENT MANAGEMENT LOCAL clause, the default is to create a dictionary-managed SYSTEM tablespace.

Locally managed tablespaces provide better performance and greater ease of management over dictionary-managed tablespaces. A locally managed SYSTEM tablespace is created AUTOALLOCATE by default, meaning that it is system managed with extent sizes determined and controlled by Oracle. You may notice an increase

in the initial size of objects created in a locally managed SYSTEM tablespace because of the autoallocate policy. It is not possible to create a locally managed SYSTEM tablespace and specify UNIFORM extent size.

When you create your database with a locally managed SYSTEM tablespace, ensure the following conditions are met:

- There must be a default temporary tablespace, and that tablespace cannot be the SYSTEM tablespace.
- You must not create rollback segments in dictionary-managed tablespaces. Attempting to create a rollback segment in a dictionary-managed tablespace will fail if the SYSTEM tablespace is locally managed.

To meet the first condition, you can specify the DEFAULT TEMPORARY TABLESPACE clause in the CREATE DATABASE statement, or you cannot include the clause and allow Oracle to create the tablespace for you using a default name and in a default location.

For fulfilling the second condition, Oracle recommends that instead of using rollback segments to manage the database's undo records, that you use automatic undo management. You can include the UNDO TABLESPACE clause in the CREATE DATABASE statement to create a specific undo tablespace, or if you do not include the clause, Oracle creates a locally managed undo tablespace for you using the default name and in a default location.

Note: When your SYSTEM tablespace is locally managed, there are restrictions on other tablespaces in the database. These restrictions are:

- You cannot create any dictionary-managed tablespaces in the database.
- You cannot migrate a locally managed tablespace to a dictionary-managed tablespace.
- You can transport dictionary-managed tablespaces into the database, but you are not allowed to alter them to read-write.
- Preexisting dictionary-managed tablespaces are allowed to remain in the database, but only in READ ONLY mode. They cannot be altered to READ WRITE.

Oracle also allows you to migrate an existing dictionary-managed SYSTEM tablespace to a locally managed tablespace. using the DBMS_SPACE_ADMIN package. However, there is no procedure for backward migration.

See Also:

- *Oracle9i SQL Reference* for more specific information about the use of the DEFAULT TEMPORARY TABLESPACE and UNDO TABLESPACE clauses when EXTENT MANAGEMENT LOCAL is specified for the SYSTEM tablespace
- "Locally Managed Tablespaces" on page 11-5
- "Migrating the SYSTEM Tablespace to a Locally Managed Tablespace" on page 11-34

Specifying the Database Time Zone and Time Zone File

Oracle allows you to specify the database's default time zone, and provides you with the option of choosing the size of the supporting time zone file.

Specifying the Database Time Zone

You set the database's default time zone by specifying the SET TIME_ZONE clause of the CREATE DATABASE statement. If omitted, the default database time zone is the operating system time zone. The database time zone can be changed for a session with an ALTER SESSION statement.

See Also: *Oracle9i Database Globalization Support Guide* for more information about setting the database time zone

Specifying the Database Time Zone File

Oracle9*i* enables you to specify the default time zone for your database using the SET TIME_ZONE clause of the CREATE DATABASE statement. This section provides information on the time zone files used to support this feature, specifically on Solaris platforms. Names of directories, filenames, and environment variables may differ for each platform but will probably be the same for all UNIX platforms.

The time zone files contain the valid time zone names. The following information is included for each zone (note that abbreviations are only used in conjunction with the zone names):

- Offset from UTC
- Transition times for daylight savings

- Abbreviation for standard time
- Abbreviation for daylight savings time

There are 2 time zone files under the Oracle installation directory:

- `$ORACLE_HOME/oracore/zoneinfo/timezone.dat`

  This is the default. It contains the most commonly used time zones and is smaller, thus enabling better database performance.

- `$ORACLE_HOME/oracore/zoneinfo/timezlrg.dat`

  This file contains the larger set of defined time zones and should be used by users who require zones that are not defined in the default `timezone.dat` file. Note that this larger set of zone information may affect performance.

To enable the use of the larger time zone datafile, do the following:

1. Shut down the database.
2. Set the environment variable `ORA_TZFILE` to the full path name of the location for the `timezlrg.dat` file.
3. Restart the database.

Once the larger `timezlrg.dat` is used, it must continue to be used unless you are sure that none of the nondefault zones are used for data that is stored in the database. Also, all databases that share information must use the same time zone datafile.

To view the time zone names, use the following query:

```
SELECT * FROM V$TIMEZONE_NAMES;
```

Specifying FORCE LOGGING Mode

Through the use of the `NOLOGGING` clause allowed in some DDL statements (for example, `CREATE TABLE`), certain database operations will not generate redo records to the database redo log. Specifying `NOLOGGING` can speed up operations that can be easily recovered outside of the database recovery mechanisms, but it causes problems for media recovery and for a standby database.

Oracle provides a means of forcing the writing of redo records for changes against the database, even where `NOLOGGING` has been specified in DDL statements. Oracle never generates redo records for temporary tablespaces and temporary segments, so forced logging has no affect for these.

See Also:

- *Oracle9i Database Concepts* for additional information about NOLOGGING mode
- *Oracle9i SQL Reference* for information about operations that can be done in NOLOGGING mode

Using the FORCE LOGGING Clause

To put the database into FORCE LOGGING mode, use the FORCE LOGGING clause in the CREATE DATABASE statement. If you do not specify this clause, the database is not placed into FORCE LOGGING mode.

Use the ALTER DATABASE statement to place the database into FORCE LOGGING mode after database creation. This statement can potentially wait a considerable time for completion because it waits for all unlogged direct writes to complete.

You can cancel FORCE LOGGING mode using the following SQL statement:

```
ALTER DATABASE NO FORCE LOGGING
```

Independent of specifying FORCE LOGGING for the database, you can selectively specify FORCE LOGGING or NO FORCE LOGGING at the tablespace level. However, if FORCE LOGGING mode is in effect for the database, it takes precedence over the tablespace mode setting. If it is not in effect for the database, then the individual tablespace settings are enforced. Oracle recommends that either the entire database is placed into FORCE LOGGING mode, or individual tablespaces be placed into FORCE LOGGING mode, but not both.

The FORCE LOGGING mode is a persistent attribute of the database. That is, if the database is shut down and restarted, it remains in the same logging mode state. However, if you re-create the control file, the database not restarted in the FORCE LOGGING mode unless you specify the FORCE LOGGING clause in the CREATE CONTROL FILE statement.

See Also: "Controlling the Writing of Redo Records" on page 11-20 for information about using the FORCE LOGGING clause for tablespace creation.

Performance Considerations of FORCE LOGGING Mode

There is a performance degradation for FORCE LOGGING mode. If there is no standby database active, but the primary reason for specifying FORCE LOGGING is to ensure complete media recovery, then consider the following:

- How many media failures are likely to happen?
- How serious is the damage if unlogged direct writes cannot be recovered?
- Is the performance degradation caused by forced logging tolerable?

If the database is running in NOARCHIVELOG mode, then generally there is no benefit to placing the database in FORCE LOGGING mode. This is because media recovery is not possible in this mode, thus there is performance degradation with little benefit.

Troubleshooting Database Creation

If for any reason database creation fails, shut down the instance and delete any files created by the CREATE DATABASE statement before you attempt to create it once again. After correcting the error that caused the failure of the database creation, try running the script again.

Dropping a Database

To drop a database, you must remove its datafiles, redo log files, and all other associated files (control files, initialization parameter files, archived log files). To view the names of the database's datafiles, redo log files, and control files, query the data dictionary views V$DATAFILE, V$LOGFILE, and V$CONTROLFILE, respectively.

If the database is in archive log mode, locate the archive log destinations by inspecting the initialization parameters LOG_ARCHIVE_DEST_*n*, or LOG_ARCHIVE_DEST and LOG_ARCHIVE_DUPLEX_DEST.

If you used the Database Configuration Assistant to create your database, you can use that tool to delete your database and clean up the files.

> **See Also:** *Oracle9i Database Reference* for more information about these views and initialization parameters

Considerations After Creating a Database

After you create a database, the instance is left running, and the database is open and available for normal database use. You may want to perform other actions, some of which are discussed in this section.

Some Security Considerations

A newly created database has least three users that are useful for administering your database: SYS, SYSTEM and OUTLN (owner of schema where stored outlines are stored).

> **Caution:** To prevent unauthorized access and protect the integrity of your database, the default passwords for SYS and SYSTEM should be changed immediately after the database is created.

Depending on the features and options installed, other users can also be present. Some of these users are:

- MDSYS (*inter*Media Spatial)
- ORDSYS (*inter*Media Audio)
- ORDPLUGINS (*inter*Media Audio)
- CTXSYS (Oracle Text)
- DBSNMP (Enterprise Manager Intelligent Agent)

To change the password for user DBSNMP refer to *Oracle Intelligent Agent User's Guide*.

> **Note Regarding Security Enhancements:** In this release of Oracle and in subsequent releases, several enhancements are being made to ensure the security of default database user accounts.
>
> - During initial installation with the Database Configuration Assistant (DCBA), all default database user accounts except SYS, SYSTEM, SCOTT, DBSNMP, OUTLN, AURORAJISUTILITY$, AURORA$ORB$UNAUTHENTICATED and OSE$HTTP$ADMIN are locked and expired. To activate a locked account, the DBA must manually unlock it and reassign it a new password.
> - In addition, the DBCA prompts for passwords for users SYS and SYSTEM during initial installation of the database rather than assigning default passwords to them. A CREATE DATABASE statement issued manually also lets you supply passwords for these two users.

See Also:

- "A Security Checklist" on page 23-20 for guidance on configuring your Oracle database in a secure manner
- "Database Administrator Usernames" on page 1-11 for more information about the users SYS and SYSTEM
- "Altering Users" on page 24-6 to learn how to add new users and change passwords
- *Oracle9i SQL Reference* for the syntax of the ALTER USER statement used for unlocking user accounts

Installing Oracle's Sample Schemas

The Oracle database server distribution media can include various SQL files that let you experiment with the system, learn SQL, or create additional tables, views, or synonyms.

Starting with Oracle9*i*, Oracle provides sample schemas that enable you to become familiar with Oracle functionality. Some Oracle documents and books use these sample schemas for presenting examples. There is an ongoing effort for most Oracle books to convert to the use of Sample Schemas based examples.

The following table briefly describes the sample schemas:

Schema	Description
Human Resources	The Human Resources (HR) schema is a basic relational database schema. There are six tables in the HR schema: Employees, Departments, Locations, Countries, Jobs, and Job_ History. The Order Entry (OE) schema has links into HR schema
Order Entry	The Order Entry (OE) schema builds on the purely relational Human Relations (HR) schema with some object-relational and object-oriented features. The OE schema contains seven tables: Customers, Product_Descriptions, Product_Information, Order_ Items, Orders, Inventories, and Warehouses. The OE schema has links into the HR schema and PM schema. This schema also has synonyms defined on HR objects to make access transparent to users.

Schema	Description
Product Media	Product Media (PM) schema includes two tables, online_media and print_media, one object type, adheader_typ, and one nested table, textdoc_typ. The PM schema includes *inter*Media and LOB column types. **Note:** To use Oracle Text you must create an Oracle Text index.
Sales History	The Sales History (SH) schema is an example of a relational star schema. It consists of one big range partitioned fact table sales and five dimension tables: times, promotions, channels, products and customers. The additional countries table linked to customers shows a simple snowflake.
Queued Shipping	The Queued Shipping (QS) schema is actually multiple schemas that contain message queues.

Sample Schemas can be installed automatically for you by the Database Configuration Assistant or you can install it manually. The schemas and installation instructions are described in detail in *Oracle9i Sample Schemas*.

Initialization Parameters and Database Creation

Oracle has attempted to provide appropriate values in the starter initialization parameter file provided with your database software, or as created for you by the Database Configuration Assistant. You can edit these Oracle-supplied initialization parameters and add others, depending upon your configuration and options and how you plan to tune the database. For any relevant initialization parameters not specifically included in the initialization parameter file, Oracle supplies defaults.

If you are creating an Oracle database for the first time, it is suggested that you minimize the number of parameter values that you alter. As you become more familiar with your database and environment, you can dynamically tune many initialization parameters using the `ALTER SYSTEM` statement. If you are using a traditional text initialization parameter file, your changes are only for the current instance. To make them permanent, you must update them manually in the initialization parameter file, otherwise they will be lost over the next shutdown and startup of the database.

If you are using a server parameter file, initialization parameter file changes made by the `ALTER SYSTEM` statement can persist across shutdown and startup. This is discussed in "Managing Initialization Parameters Using a Server Parameter File" on page 2-44.

This section introduced you to some of the initialization parameters you may choose to add or edit before you create your new database.

The following topics are contained in this section:

- Determining the Global Database Name
- Specifying Control Files
- Specifying Database Block Sizes
- Setting Initialization Parameters that Affect the Size of the SGA
- Specifying the Maximum Number of Processes
- Specifying the Method of Undo Space Management
- Setting License Parameters

> **See Also:** *Oracle9i Database Reference* for descriptions of all initialization parameters including their default settings

Determining the Global Database Name

A database's global database name consists of the local database name that you assign and its location within a network structure. The DB_NAME initialization parameter determines the local name component of the database's name, while the DB_DOMAIN parameter indicates the domain (logical location) within a network structure. The combination of the settings for these two parameters must form a database name that is unique within a network.

For example, to create a database with a global database name of `test.us.acme.com`, edit the parameters of the new parameter file as follows:

```
DB_NAME = test
DB_DOMAIN = us.acme.com
```

You can rename the GLOBAL_NAME of your database using the ALTER DATABASE RENAME GLOBAL_NAME statement. However, you must also shut down and restart the database after first changing the DB_NAME and DB_DOMAIN initialization parameters and re-creating the control file.

> **See Also:** *Oracle9i Database Utilities* for information about using the DBNEWID utility, which is another means of changing a database name

DB_NAME Initialization Parameter

DB_NAME must be set to a text string of no more than eight characters. During database creation, the name provided for DB_NAME is recorded in the datafiles, redo log files, and control file of the database. If during database instance startup the value of the DB_NAME parameter (in the parameter file) and the database name in the control file are not the same, the database does not start.

DB_DOMAIN Initialization Parameter

DB_DOMAIN is a text string that specifies the network domain where the database is created. This is typically the name of the organization that owns the database. If the database you are about to create will ever be part of a distributed database system, pay special attention to this initialization parameter before database creation.

> **See Also:** Part VI, "Distributed Database Management" for more information about distributed databases

Specifying Control Files

Include the CONTROL_FILES initialization parameter in your new parameter file and set its value to a list of control filenames to use for the new database. When you execute the CREATE DATABASE statement, the control files listed in the CONTROL_FILES parameter are created. If no filenames are listed for the CONTROL_FILES parameter, Oracle uses a default operating system dependent filename.

If you want Oracle to create new operating system files when creating your database's control files, the filenames listed in the CONTROL_FILES parameter must not match any filenames that currently exist on your system. If you want Oracle to reuse or overwrite existing files when creating your database's control files, ensure that the filenames listed in the CONTROL_FILES parameter match the filenames that are to be reused.

> **Caution:** Use extreme caution when setting this option. If you inadvertently specify a file that you did not intend and execute the CREATE DATABASE statement, the previous contents of that file will be overwritten.

Oracle Corporation strongly recommends you use at least two control files stored on separate physical disk drives for each database.

> **See Also:** Chapter 6, "Managing Control Files"

Specifying Database Block Sizes

The DB_BLOCK_SIZE initialization parameter specifies the standard block size for the database. This block size is used for the SYSTEM tablespace and by default in other tablespaces. Oracle can support up to four additional non-standard block sizes.

DB_BLOCK_SIZE Initialization Parameter

The most commonly used block size should be picked as the standard block size. In many cases, this is the only block size that you need to specify. Typically, DB_BLOCK_SIZE is set to either 4K or 8K. If not specified, the default data block size is operating system specific, and is generally adequate.

The block size *cannot* be changed after database creation, except by re-creating the database. If a database's block size is different from the operating system block size, make the database block size a multiple of the operating system's block size.

For example, if your operating system's block size is 2K (2048 bytes), the following setting for the DB_BLOCK_SIZE initialization parameter is valid:

```
DB_BLOCK_SIZE=4096
```

You may want to specify a block size larger than your operating system block size. A larger data block size provides greater efficiency in disk and memory I/O (access and storage of data). Such cases include the following scenarios:

- Oracle is on a large computer system with a large amount of memory and fast disk drives. For example, databases controlled by mainframe computers with vast hardware resources typically use a data block size of 4K or greater.
- The operating system that runs Oracle uses a small operating system block size. For example, if the operating system block size is 1K and the default data block size matches this, Oracle may be performing an excessive amount of disk I/O during normal operation. For best performance in this case, a database block should consist of multiple operating system blocks.

> **See Also:** Your operating system specific Oracle documentation for details about the default block size.

Non-Standard Block Sizes

Tablespaces of non-standard block sizes can be created using the CREATE TABLESPACE statement and specifying the BLOCKSIZE clause. These non-standard block sizes can have any power-of-two value between 2K and 32K: specifically, 2K,

4K, 8K, 16K or 32K. Platform-specific restrictions regarding the maximum block size apply, so some of these sizes may not be allowed on some platforms.

To use non-standard block sizes, you must configure sub-caches within the buffer cache area of the SGA memory for all of the non-standard block sizes that you intend to use. The initialization parameters used for configuring these sub-caches are described in the next section, "Setting Initialization Parameters that Affect the Size of the SGA".

The ability to specify multiple block sizes for your database is especially useful if you are transporting tablespaces between databases. You can, for example, transport a tablespace that uses a 4K block size from an OLTP environment to a datawarehouse environment that uses a standard block size of 8K.

See Also:

- "Creating Tablespaces" on page 11-3
- "Transporting Tablespaces Between Databases" on page 11-34

Setting Initialization Parameters that Affect the Size of the SGA

The initialization parameters discussed in this section affect the amount of memory that is allocated to the System Global Area. Except for the `SGA_MAX_SIZE` initialization parameter, they are dynamic parameters which values can be changed by the `ALTER SYSTEM` statement. The size of the SGA is dynamic, and can grow or shrink by dynamically altering these parameters.

Note: The memory for dynamic components in the SGA is allocated in the unit of granules. Granule size is determined by total SGA size. Generally speaking, on most platforms, if the total SGA size is equal to or less than 128 MB, then granule size is 4 MB. Otherwise, granule size is 16 MB.

However, there may be some platform dependency. For example, on 32-bit Windows NT, the granule size is 8 MB for SGAs larger than 128 MB. Consult your operating system specific documentation for more details.

You can query the V$SGA_DYNAMIC_COMPONENTS view to see the granule size that is being used by an instance. The same granule size is used for all dynamic components in the SGA.

If you specify a size for a component that is not a multiple of granule size, Oracle will round the specified size up to the nearest multiple. For example, if the granule size is 4 MB and you specify DB_CACHE_SIZE as 10 MB, you will actually be allocated 12 MB.

You can see a summary of information about the dynamic components of the SGA in the V$SGA_DYNAMIC_COMPONENTS view. Information about on-going SGA resize operations can be found in the V$SGA_CURRENT_RESIZE_OPS view, and information about the last 100 completed SGA resize operations can be found in the V$SGA_RESIZE_OPS view. To find the amount of SGA memory available for future dynamic SGA resize operations, query the V$SGA_DYNAMIC_FREE_MEMORY view.

See Also:

- *Oracle9i Database Performance Tuning Guide and Reference* for information about the monitoring and tuning of SGA components
- *Oracle9i Database Reference* for descriptions of the dynamic performance views used for monitoring the size of the SGA
- *Oracle9i Database Concepts* for a conceptual discussion of the SGA

Limiting the Size of the SGA

The SGA_MAX_SIZE initialization parameter specifies the maximum size of the System Global Area for the lifetime of the instance. You can dynamically alter the initialization parameters affecting the size of the buffer caches, shared pool, and large pool, but only to the extent that the sum of these sizes and the sizes of the other components of the SGA (fixed SGA, variable SGA, and redo log buffers) does not exceed the value specified by SGA_MAX_SIZE.

If you do not specify SGA_MAX_SIZE, then Oracle selects a default value that is the sum of all components specified or defaulted at initialization time.

Setting the Buffer Cache Initialization Parameters

The buffer cache initialization parameters determine the size of the buffer cache component of the SGA. You use them to specify the sizes of caches for the various block sizes used by the database. These initialization parameters are all dynamic.

If you intend to use multiple block sizes in your database, you must have the DB_CACHE_SIZE and at least one DB_*n*K_CACHE_SIZE parameter set. Oracle assigns an appropriate default value to the DB_CACHE_SIZE parameter, but the DB_*n*K_CACHE_SIZE parameters default to 0, and no additional block size caches are configured.

The size of a buffer cache affects performance. Larger cache sizes generally reduce the number of disk reads and writes. However, a large cache may take up too much memory and induce memory paging or swapping.

DB_CACHE_SIZE Initialization Parameter The DB_CACHE_SIZE initialization parameter replaces the DB_BLOCK_BUFFERS initialization parameter that was used in previous releases. The DB_CACHE_SIZE parameter specifies the size of the cache of standard block size buffers, where the standard block size is specified by DB_BLOCK_SIZE.

For backward compatibility the DB_BLOCK_BUFFERS parameter will still work, but it remains a static parameter and cannot be combined with any of the dynamic sizing parameters.

DB_*n*K_CACHE_SIZE Initialization Parameters The sizes and numbers of non-standard block size buffers are specified by the following initialization parameters:

- DB_2K_CACHE_SIZE
- DB_4K_CACHE_SIZE
- DB_8K_CACHE_SIZE

- DB_16K_CACHE_SIZE
- DB_32K_CACHE_SIZE.

Each parameter specifies the size of the buffer cache for the corresponding block size. For example:

```
DB_BLOCK_SIZE=4096

DB_CACHE_SIZE=12M
DB_2K_CACHE_SIZE=8M
DB_8K_CACHE_SIZE=4M
```

In the above example, the parameters specify that the standard block size of the database will be 4K. The size of the cache of standard block size buffers will be 12M. Additionally, 2K and 8K caches will be configured with sizes of 8M and 4M respectively.

Note: These parameters cannot be used to size the cache for the standard block size. For example, if the value of DB_BLOCK_SIZE is 2K, it is illegal to set DB_2K_CACHE_SIZE. The size of the cache for the standard block size is always determined from the value of DB_CACHE_SIZE.

Adjusting the Size of the Shared Pool

The SHARED_POOL_SIZE initialization parameter is a dynamic parameter that allows you to specify or adjust the size of the shared pool component of the SGA. Oracle selects an appropriate default value.

Adjusting the Size of the Large Pool

The LARGE_POOL_SIZE initialization parameter is a dynamic parameter that allows you to specify or adjust the size of the large pool component of the SGA. Oracle selects an appropriate default value.

Specifying the Maximum Number of Processes

The PROCESSES initialization parameter determines the maximum number of operating system processes that can be connected to Oracle concurrently. The value of this parameter must be 6 or greater (5 for the background processes plus 1 for each user process). For example, if you plan to have 50 concurrent users, set this parameter to at least 55.

Specifying the Method of Undo Space Management

Every Oracle database must have a method of maintaining information that is used to roll back, or undo, changes to the database. Such information consists of records of the actions of transactions, primarily before they are committed. Oracle refers to these records collectively as **undo**. Oracle allows you to store undo in an undo tablespace or in rollback segments.

See Also: Chapter 13, "Managing Undo Space"

UNDO_MANAGEMENT Initialization Parameter

The UNDO_MANAGEMENT initialization parameter determines whether an instance will start up in automatic undo management mode, where undo is stored in an undo tablespace, or manual undo management mode, where undo is stored in rollback segments. A value of AUTO enables automatic undo management mode, MANUAL enables manual undo management mode. For backward compatibility, the default is MANUAL.

UNDO_TABLESPACE Initialization Parameter

When the instance starts up in automatic undo management mode, it selects the first available undo tablespace in the instance for storing undo. A default undo tablespace named SYS_UNDOTBS is automatically created when you execute a CREATE DATABASE statement and the UNDO_MANAGEMENT initialization parameter is set to AUTO. This is the undo tablespace that Oracle normally selects whenever you start up the database.

Optionally, you can specify the UNDO_TABLESPACE initialization parameter. This causes the instance to use the undo tablespace specified by the parameter. The UNDO_TABLESPACE parameter can be used to assign a specific undo tablespace to an instance in an Oracle Real Application Clusters environment.

If there is no undo tablespace available, the instance will start, but uses the SYSTEM rollback segment. This is not recommended in normal circumstances, and an alert message is written to the alert file to warn that the system is running without an undo tablespace.

Oracle recommends using an undo tablespace rather than rollback segments. An undo tablespace is easier to administer and enables you to explicitly set an undo retention time.

ROLLBACK_SEGMENTS Initialization Parameter

The ROLLBACK_SEGMENTS parameter is a list of the non-system rollback segments an Oracle instance acquires at database startup if the database is to operate in manual undo management mode. List your rollback segments as the value of this parameter. If no rollback segments are specified, the system rollback segment is used.

The ROLLBACK_SEGMENTS initialization parameter is supported for backward compatibility. Oracle recommends using an undo tablespace rather than rollback segments.

Setting License Parameters

Note: Oracle no longer offers licensing by the number of concurrent sessions. Therefore the LICENSE_MAX_SESSIONS and LICENSE_SESSIONS_WARNING initialization parameters have been deprecated and are no longer discussed in this book.

If you use named user licensing, Oracle can help you enforce this form of licensing. You can set a limit on the number of users created in the database. Once this limit is reached, you cannot create more users.

Note: This mechanism assumes that each person accessing the database has a unique user name and that no people share a user name. Therefore, so that named user licensing can help you ensure compliance with your Oracle license agreement, do not allow multiple users to log in using the same user name.

To limit the number of users created in a database, set the LICENSE_MAX_USERS initialization parameter in the database's initialization parameter file, as shown in the following example:

```
LICENSE_MAX_USERS = 200
```

Managing Initialization Parameters Using a Server Parameter File

Oracle has traditionally stored initialization parameters in a text initialization parameter file. Starting with Oracle9*i*, you can choose to maintain initialization parameters in a binary server parameter file.

This section introduces the server parameter file, and explains how to manage initialization parameters using either method of storing the parameters. The following topics are contained in this section.

What is a Server Parameter File?

A **server parameter file** (SPFILE) can be thought of as a repository for initialization parameters that is maintained on the machine where the Oracle database server executes. It is, by design, a server-side initialization parameter file. Initialization parameters stored in a server parameter file are persistent, in that any changes made to the parameters while an instance is running can persist across instance shutdown and startup. This eliminates the need to manually update initialization parameters to make changes effected by `ALTER SYSTEM` statements persistent. It also provides a basis for self tuning by the Oracle database server.

A server parameter file is initially built from a traditional text initialization parameter file using the `CREATE SPFILE` statement. It is a binary file that cannot be browsed or edited using a text editor. Oracle provides other interfaces for viewing and modifying parameter settings.

> **Caution:** Although you can open the binary server parameter file with a text editor and view its text, *do not* manually edit it. Doing so will corrupt the file. You will not be able to start you instance, and if the instance is running, it could fail.

At system startup, the default behavior of the STARTUP command is to read a server parameter file to obtain initialization parameter settings. The STARTUP command with no PFILE clause, reads the server parameter file from an operating system specific location. If you choose to use the traditional text initialization parameter file, you must specify the PFILE clause when issuing the STARTUP command. Explicit instructions for starting an instance using a server parameter file are contained in Starting Up a Database on page 4-2.

Migrating to a Server Parameter File

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

1. If the initialization parameter file is located on a client machine, transfer the file (for example, FTP) from the client machine to the server machine.

   > **Note:** If you are using Oracle9*i* Real Application Clusters, you must combine all of your instance specific initialization parameter files into a single initialization parameter file. Instructions for doing this, and other actions unique to using a server parameter file for Oracle Real Application Cluster instances, are discussed in:
   >
   > - *Oracle9i Real Application Clusters Setup and Configuration*
   > - *Oracle9i Real Application Clusters Administration*

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

Creating a Server Parameter File

The server parameter file must initially be created from a traditional text initialization parameter file. It must be created prior to its use in the STARTUP command. The CREATE SPFILE statement is used to create a server parameter file. You must have the SYSDBA or the SYSOPER system privilege to execute this statement.

The following example creates a server parameter file from initialization parameter file /u01/oracle/dbs/init.ora. In this example no SPFILE name is specified, so the file is created in a platform-specific default location and is named spfile$ORACLE_SID.ora.

```
CREATE SPFILE FROM PFILE='/u01/oracle/dbs/init.ora';
```

Another example, below, illustrates creating a server parameter file and supplying a name.

```
CREATE SPFILE='/u01/oracle/dbs/test_spfile.ora'
       FROM PFILE='/u01/oracle/dbs/test_init.ora';
```

The server parameter file is always created on the machine running the database server. If a server parameter file of the same name already exists on the server, it is overwritten with the new information.

Oracle recommends that you allow the database server to default the name and location of the server parameter file. This will ease administration of your database. For example, the STARTUP command assumes this default location to read the parameter file.

When the server parameter file is created from the initialization parameter file, comments specified on the same lines as a parameter setting in the initialization parameter file are maintained in the server parameter file. All other comments are ignored.

The CREATE SPFILE statement can be executed before or after instance startup. However, if the instance has been started using a server parameter file, an error is raised if you attempt to re-create the same server parameter file that is currently being used by the instance.

Note: When you use the Database Configuration Assistant to create a database, it can automatically create a server parameter file for you.

The SPFILE Initialization Parameter

The SPFILE initialization parameter contains the name of the current server parameter file. When the default server parameter file is used by the server (that is, you issue a STARTUP command and do not specify a PFILE), the value of SPFILE is internally set by the server. The SQL*Plus command SHOW PARAMETERS SPFILE (or any other method of querying the value of a parameter) displays the name of the server parameter file that is currently in use.

The SPFILE parameter can also be set in a traditional parameter file to indicate the server parameter file to use. You use the SPFILE parameter to specify a server parameter file located in a nondefault location. *Do not* use an IFILE initialization parameter within a traditional initialization parameter file to point to a server parameter file; instead, use the SPFILE parameter. See "Starting Up a Database" on page 4-2 for details about:

- Starting up a database that uses a server parameter file
- Using the SPFILE parameter to specify the name of a server parameter file to use at instance startup

Using ALTER SYSTEM to Change Initialization Parameter Values

The ALTER SYSTEM statement allows you to set, change, or delete (restore to default value) initialization parameter values. When the ALTER SYSTEM statement is used to alter a parameter setting in a traditional initialization parameter file, the change affects only the current instance, since there is no mechanism for automatically updating initialization parameters on disk. They must be manually updated in order to be passed to a future instance. Using a server parameter file overcomes this limitation.

Setting or Changing Initialization Parameter Values

Use the SET clause of the ALTER SYSTEM statement to set or change initialization parameter values. Additionally, the SCOPE clause specifies the scope of a change as described in the following table:

SCOPE Clause	Description
SCOPE = SPFILE	The change is applied in the server parameter file only. The effect is as follows: ■ For dynamic parameters, the change is effective at the next startup and is persistent. ■ For static parameters, the behavior is the same as for dynamic parameters. This is the only SCOPE specification allowed for static parameters.
SCOPE = MEMORY	The change is applied in memory only. The effect is as follows: ■ For dynamic parameters, the effect is immediate, but it is not persistent because the server parameter file is not updated. ■ For static parameters, this specification is not allowed.
SCOPE = BOTH	The change is applied in both the server parameter file and memory. The effect is as follows: ■ For dynamic parameters, the effect is immediate and persistent. ■ For static parameters, this specification is not allowed.

It is an error to specify SCOPE=SPFILE or SCOPE=BOTH if the server is not using a server parameter file. The default is SCOPE=BOTH if a server parameter file was used to start up the instance, and MEMORY if a traditional initialization parameter file was used to start up the instance.

For dynamic parameters, you can also specify the DEFERRED keyword. When specified, the change is effective only for future sessions.

A COMMENT clause allows a comment string to be associated with the parameter update. When you specify SCOPE as SPFILE or BOTH, the comment is written to the server parameter file.

The following statement changes the maximum number of job queue processes allowed for the instance. It also specifies a comment, and explicitly states that the change is to be made only in memory (that is, it is not persistent across instance shutdown and startup).

```
ALTER SYSTEM SET JOB_QUEUE_PROCESSES=50
                 COMMENT='temporary change on Nov 29'
                 SCOPE=MEMORY;
```

Another example illustrates setting a complex initialization parameter that takes a list of strings. Specifically, the parameter value being set is the LOG_ARCHIVE_DEST_n initialization parameter. The case could be that either the parameter is being changed to a new value or a new archive destination is being added.

```
ALTER SYSTEM
     SET LOG_ARCHIVE_DEST_4='LOCATION=/u02/oracle/rbdb1/',MANDATORY,'REOPEN=2'
         COMMENT='Add new destimation on Nov 29'
         SCOPE=SPFILE;
```

Note that when a value consists of a list of strings, the syntax of the ALTER SYSTEM SET statement does not support editing each element of the list of values by the position or ordinal number. You must specify the complete list of values each time the parameter is updated, and the new list completely replaces the old list.

Deleting Initialization Parameter Values

For initialization parameters whose values are string values you can restore a parameter to its default value (effectively deleting it), by using the following syntax:

```
ALTER SYSTEM SET parameter = '';
```

For numeric and boolean value parameters, you must specifically set the parameter back to its original default value.

Exporting the Server Parameter File

You can export a server parameter file to create a traditional text initialization parameter file. Reasons for doing this include:

- Creating backups of the server parameter file
- For diagnostic purposes, listing all of the parameter values currently used by an instance. This is analogous to the SQL*Plus SHOW PARAMETERS command or selecting from the V$PARAMETER or V$PARAMETER2 views.
- Modifying of the server parameter file by first exporting it, editing the output file, and then re-creating it

The exported file can also be used to start up an instance using the PFILE option.

The CREATE PFILE statement is used to export a server parameter file. You must have the SYSDBA or the SYSOPER system privilege to execute this statement. The exported file is created on the database server machine. It contains any comments associated with the parameter in the same line as the parameter setting.

The following example creates a text initialization parameter file from the server parameter file:

```
CREATE PFILE FROM SPFILE;
```

Because no names were specified for the files, a platform-specific name is used for the initialization parameter file, and it is created from the platform-specific default server parameter file.

The following example creates a text initialization parameter file from a server parameter file where the names of the files are specified:

```
CREATE PFILE='/u01/oracle/dbs/test_init.ora'
       FROM SPFILE='/u01/oracle/dbs/test_spfile.ora';
```

Backing Up the Server Parameter File

You can create a backup of your server parameter file by exporting it, as described in "Exporting the Server Parameter File" on page 2-49. If the backup and recovery strategy for your database is implemented using Recovery Manager (RMAN), then you can use RMAN to create a backup. The server parameter file is backed up automatically by RMAN when you back up your database, but RMAN also allows you to specifically create a backup of the currently active server parameter file.

See Also: Oracle9i Recovery Manager User's Guide

Errors and Recovery for the Server Parameter File

If an error occurs while reading the server parameter file (during startup or an export operation), or while writing the server parameter file during its creation, the operation terminates with an error reported to the user.

If an error occurs while reading or writing the server parameter file during a parameter update, the error is reported in the alert file and all subsequent parameter updates to the server parameter file are ignored. At this point, you have the following options:

- Shut down the instance, recover the server parameter file, then restart the instance
- Continue to run without caring that subsequent parameter updates will not be persistent

Viewing Parameter Settings

You have several options for viewing parameter settings.

Method	Description
SHOW PARAMETERS	This SQL*Plus command displays the currently in use parameter values.
CREATE PFILE	This SQL statement creates a text initialization parameter file from the binary server parameter file.
V$PARAMETER	This view displays the currently in effect parameter values.
V$PARAMETER2	This view displays the currently in effect parameter values. It is easier to distinguish list parameter values in this view because each list parameter value appears as a row.
V$SPPARAMETER	This view displays the current contents of the server parameter file. The view returns NULL values if a server parameter file is not being used by the instance.

See Also: *Oracle9i Database Reference* for a complete description of views

3
Using Oracle-Managed Files

This chapter discusses the use of the Oracle-managed files and contains the following topics: