Oracle® Database Express Edition

2 Day DBA

11*g* Release 2 (11.2)

E18804-05

May 2014

ORACLE®

Oracle Database Express Edition 2 Day DBA, 11*g* Release 2 (11.2)

E18804-05

Primary Author: Chuck Murray

Contributing Author: Steve Fogel, Kathy Rich, Antonio Romero, Marcie Young, Colin McGregor, Terri Jennings

Contributor: Mark Townsend, Bjørn Engsig, Santanu Datta, Michael Hichwa, Christina Cho, Matt McKerley, Graham Wood, Kant Patel, Ramanujam Srinivasan, Tammy Bednar, Beldalker Anand, Ed Miner, Paul Lo, Roy Swonger, Rae Burns,Valarie Moore, Tirthankar Lahiri, Satish Panchumarthy, Srinivas Poovala, Aneesh Khandelwal, Kevin Reardon, Thomas Baby, Roy Crow, Randy Urbano

Contents

List of Figures

List of Tables

List of Examples

Preface

Welcome to *Oracle Database Express Edition 2 Day DBA*. This documentation provides background and how-to information for administering Oracle Database Express Edition. The preface contains the following sections:

Audience

Oracle Database Express Edition 2 Day DBA is for anyone who wants to perform common, day-to-day administrative tasks with Oracle Database Express Edition. Prior knowledge or experience with managing databases is not required.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc`.

Access to Oracle Support

Oracle customers have access to electronic support through My Oracle Support. For information, visit `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info` or visit `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs` if you are hearing impaired.

Related Documentation

For more information, see these Oracle resources:

- *Oracle Database Express Edition Getting Started Guide*
- *Oracle Database Express Edition 2 Day Developer Guide*
- *Oracle Database Express Edition Installation Guide* for your platform
- *Oracle Database Administrator's Guide*
- *Oracle Database SQL Language Reference*
- *Oracle Database Reference*

For the most recent version of the Oracle Database Express Edition documentation, see the Oracle Database XE online library:

http://www.oracle.com/pls/xe112/homepage

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in the text or a glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

1

Introducing Oracle Database XE

Oracle Database Express Edition (Oracle Database XE) is a free, smaller-footprint edition of Oracle Database. Oracle Database XE is easy to install and easy to manage.

With Oracle Database XE and related tools you can:

- Administer the database
- Create tables, views, and other database objects
- Import, export, and view table data
- Run queries and SQL scripts

This guide assumes that you are familiar with the following concepts and operations described in *Oracle Database Express Edition Getting Started Guide*:

- The Oracle Database XE user interface, including the system menu commands and the database administration interface
- Creating a database user
- Installing and using SQL Developer, including creating database connections

If you are not familiar with these basics, see the *Oracle Database Express Edition Getting Started Guide* for a quick overview. (However, you do not need to do the tutorial for creating an Oracle Application Express application in that guide, because that is nor directly relevant to DBA activities.)

The remaining chapters cover the following topics:

- Starting Up and Shutting Down
- Connecting to the Database
- Managing Network Connections
- Managing Database Memory
- Managing Database Storage
- Managing Users and Security
- Monitoring the Database
- Viewing Database Version and Globalization Information
- Exporting and Importing Metadata and Data
- Backing Up and Recovering

2

Starting Up and Shutting Down

This section describes how to start up and shut down Oracle Database Express Edition. It contains the following topics:

Starting Up the Database

Oracle Database Express Edition (Oracle Database XE) starts up automatically immediately after installation and after each system restart. Thus, there is no need to start up the database unless you previously shut it down.

You can start up the database from the desktop or with the SQL Command Line (SQL*Plus). Each of these methods is described in the following sections:

Starting Up the Database from the Desktop

This section explains how to start up the database from the desktop in Windows and in the following two Linux windowing managers: KDE and Gnome. If your Linux computer is not running a windowing manager, or is running a windowing manager other than KDE or Gnome, you must start the database with the SQL Command Line. See "Starting Up the Database Using the SQL Command Line" on page 2-2 for instructions.

To start up the database using the desktop:

1. Do one of the following:
   - On Windows: Log in to the Oracle Database XE host computer as the user that installed Oracle Database XE or as a Windows administrator—that is, as a user who is a member of the Administrator group.
   - On Linux: Log in to the Oracle Database XE host computer as the user that installed Oracle Database XE or as a user who is a member of the `dba` user group. This is typically the user `oracle`. See "Operating System Authentication" on page 7-4 for more information.
2. Do one of the following:
   - On Windows: Click **Start**, point to **Programs (or All Programs)**, point to **Oracle Database 11g Express Edition**, and then select **Start Database**.

- On Linux with Gnome: In the Applications menu, point to **Oracle Database 11g Express Edition**, and then select **Start Database**.
- On Linux with KDE: Click the icon for the K Menu, point to **Oracle Database 11g Express Edition**, and then select **Start Database**.

Starting Up the Database Using the SQL Command Line

When you start up the database with the SQL Command Line, you must run the SQL Command Line on the same computer where you installed Oracle Database XE (the "Oracle Database XE host computer").

To start up the database using the SQL Command Line:

1. Do one of the following:
   - On Windows: Log in to the Oracle Database XE host computer as the user that installed Oracle Database XE or as a Windows administrator—that is, as a user who is a member of the Administrator group.
   - On Linux: Log in to the Oracle Database XE host computer as the user that installed Oracle Database XE or as a user who is a member of the `dba` user group. This is typically the user `oracle`. See "Operating System Authentication" on page 7-4 for more information.
2. If not already opened, open a terminal session or command window.
3. Linux platform only: Ensure that environment variables are set properly.

   See "Setting Environment Variables on the Linux Platform" on page 3-5 for details.
4. At the operating system prompt, enter the following command to start the SQL Command Line and connect to the database:

   ```
   sqlplus / as sysdba
   ```

   The slash (/) indicates that the database should authenticate you with operating system authentication.
5. At the SQL Command Line prompt, enter the following command:

   ```
   SQL> STARTUP
   ```

   If the command is successful, it displays output similar to the following. (System global area sizes will vary depending on the amount of physical memory in your Oracle Database XE host computer.)

   ```
   ORACLE instance started.

   Total System Global Area  599785472 bytes
   Fixed Size                  1220804 bytes
   Variable Size             180358972 bytes
   Database Buffers          415236096 bytes
   Redo Buffers                2969600 bytes
   Database mounted.
   Database opened.
   ```
6. (Optional) Enter the following SQL query to verify that the database started up properly:

   ```
   SQL> select count(*) from hr.employees;
   ```

   The query results should look similar to the following:

```
  COUNT(*)
----------
       107
```

7. To exit the SQL Command Line. enter the following command:

```
SQL> EXIT
```

Shutting Down the Database

Oracle Database Express Edition (Oracle Database XE) shuts down automatically when you shut down the computer that hosts it. However, you can also shut Oracle Database XE down manually whenever you want, such as to reduce the overall system processing overhead when you do not need to use the database.

Before shutting down Oracle Database XE, it is best to ensure that all users and applications have completed their work and logged out. See "Monitoring Sessions" on page 8-1 for information on how to view current database sessions.

If users or applications are still logged in when you begin a shutdown operation, the shutdown proceeds under the following conditions:

- No new connections are permitted, and no new transactions are allowed to be started.
- Any uncommitted transactions are rolled back.
- All users and applications are immediately disconnected.

You can shut down the database with the desktop or with the SQL Command Line (SQL*Plus). Each of these methods is described in the following sections:

- Shutting Down the Database from the Desktop on page 2-3
- Shutting Down the Database Using the SQL Command Line on page 2-4

Shutting Down the Database from the Desktop

This section explains how to shut down the database from the desktop in Windows and in the following two Linux windowing managers: KDE and Gnome. If your Linux computer is not running a windowing manager, or is running a windowing manager other than KDE or Gnome, you must shut down the database with the SQL Command Line.

To shut down the database using the desktop:

1. Do one of the following:
   - On Windows: Log in to the Oracle Database XE host computer as the user that installed Oracle Database XE or as a Windows administrator—that is, as a user who is a member of the Administrator group.
   - On Linux: Log in to the Oracle Database XE host computer as the user that installed Oracle Database XE or as a user who is a member of the `dba` user group. This is typically the user `oracle`. See "Operating System Authentication" on page 7-4 for more information.
2. Do one of the following:
   - On Windows: Click **Start**, point to **Programs (or All Programs)**, point to **Oracle Database 11g Express Edition**, and then select **Stop Database**.

- On Linux with Gnome: In the Applications menu, point to **Oracle Database 11g Express Edition**, and then select **Stop Database**.
- On Linux with KDE: Click the icon for the K Menu, point to **Oracle Database 11g Express Edition**, and then select **Stop Database**.

Shutting Down the Database Using the SQL Command Line

When you shut down the database with the SQL Command Line, you must run the SQL Command Line on the same computer where you installed Oracle Database XE (the "Oracle Database XE host computer").

To shut down the database using the SQL Command Line:

1. Do one of the following:
   - On Windows: Log in to the Oracle Database XE host computer as the user that installed Oracle Database XE or as a Windows administrator—that is, as a user who is a member of the Administrator group.
   - On Linux: Log in to the Oracle Database XE host computer as the user that installed Oracle Database XE or as a user who is a member of the `dba` user group. This is typically the user `oracle`. See "Operating System Authentication" on page 7-4 for more information.
2. If not already opened, open a terminal session or command window.
3. Linux platform only: Ensure that environment variables are set properly.

   See "Setting Environment Variables on the Linux Platform" on page 3-5 for details.
4. At the operating system prompt, enter the following command to start the SQL Command Line and connect to the database:

   ```
   sqlplus / as sysdba
   ```

   The slash (/) indicates that the database should authenticate you with operating system authentication.
5. At the SQL Command Line prompt, enter the following command:

   ```
   SQL> SHUTDOWN IMMEDIATE
   ```

   Note that this command may take a short while to complete. If the command is successful, it displays the following output:

   ```
   Database closed.
   Database dismounted.
   ORACLE instance shut down.
   ```

   If the command displays no output after a number of minutes, indicating that the shutdown operation is not proceeding, you can press `CTRL-C` to interrupt the command, and then enter the following command:

   ```
   SQL> SHUTDOWN ABORT
   ```

   The database must go through a recovery process when it starts up after a `SHUTDOWN ABORT` command. It is recommended that you enable the recovery process to take place immediately, after which you can shut down the database normally. To do this, enter the following commands when the `SHUTDOWN ABORT` completes:

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

See *Oracle Database Administrator's Guide* for information on the `SHUTDOWN ABORT` command.

6. To exit the SQL Command Line. enter the following command:

```
SQL> EXIT
```

3

Connecting to the Database

To interact with Oracle Database XE, you must typically connect to the database as a database user. The interaction might be through the SQL Command Line, through SQL Developer, or through utilities invoked from the system command line.

This chapter focuses on the SQL Command Line (SQL*Plus). For information about using database connections in SQL Developer, see the section about creating database connections in *Oracle Database Express Edition Getting Started Guide*.

This chapter covers the following topics:

See Also: The following documents, for information on how to connect to the database from your application:

- *Oracle Database Express Edition Java 2 Day Developer Guide*
- *Oracle Database Express Edition 2 Day Plus .NET Developer Guide*
- *Oracle Database Express Edition 2 Day + PHP Developer's Guide*

About Local and Remote Connections

Oracle Database XE supports connections between the SQL Command Line (SQL*Plus) and the database either *locally*, or *remotely* over a TCP/IP network. The method that you use to connect to Oracle Database XE with the SQL Command Line depends on whether you are initiating a local connection or a remote connection. Local and remote connections are explained in the following sections:

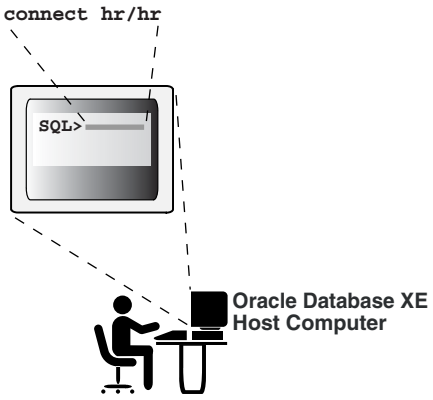
See Also:

- "Connecting Locally with the SQL Command Line" on page 3-5
- "Connecting Remotely with the SQL Command Line" on page 3-6
- "Managing Network Connections" on page 4-1 for information on how Oracle Database XE accepts connection requests over the network.

About Local Connections

Connecting **locally** means running the SQL Command Line (or any other Oracle command-line utility) on the same host computer where Oracle Database XE is installed (the "Oracle Database XE host computer") and then initiating a database connection from the SQL Command Line (or other utility), or using SQL Developer to connect as a local database user. To connect locally, you must supply only a database user name and password. For example, Figure 3–1 shows a user connecting locally with the SQL Command Line and entering a `connect` command that supplies the user name `hr` and password `hr`.

Figure 3–1 Local Connection



Environment Variables

Before making a local connection on Linux, you must set environment variables. See "Setting Environment Variables" on page 3-4 for more information.

Note: Logging in to the Oracle Database XE host computer with an `ssh` (or `telnet`) session and then starting the SQL Command Line (or other Oracle command-line utility) is considered a local connection, even though you start the `ssh` (or `telnet`) application on a remote computer.

About Remote Connections

Connecting **remotely** means running the SQL Command Line (or any other Oracle command-line utility) on a computer other than the Oracle Database XE host computer, and then initiating a database connection from the SQL Command Line (or other utility) over the network.

Oracle Client Software

The remote computer must have Oracle client software installed. It is through Oracle client software that Oracle command-line utilities (and your applications) connect to the database. Oracle Database XE accepts connections from all of the following types of Oracle client software:

- Oracle Database Express Edition Client (Oracle Database XE)

  When you install Oracle Database XE, Oracle Database Express Edition Client is also installed on the same computer. You can install Oracle Database XE separately on remote computers. It is available at

  http://www.oracle.com/technetwork/database/express-edition/

- Instant Client

  Instant Client is available at

http://www.oracle.com/technetwork/database/features/instant-client/index-100365.html

- Oracle client software for Oracle Database Enterprise Edition or Standard Edition (all supported releases of Oracle Database)

All Oracle client software includes Oracle Net, which is the Oracle network software that enables client applications on one computer to connect to databases on another computer over a network.

Connect Strings

To connect remotely, you must supply not just a user name and password, but a complete Oracle Net connect string. In addition to the database user name and password, a **connect string** includes a host name or host IP address, an optional TCP port number, and an optional database service name. These additional parameters are required to help Oracle Net find the right host computer and connect to Oracle Database XE. An Oracle Net connect string has the following format:

```
username/password@[//]host[:port][/service_name]
```

where:

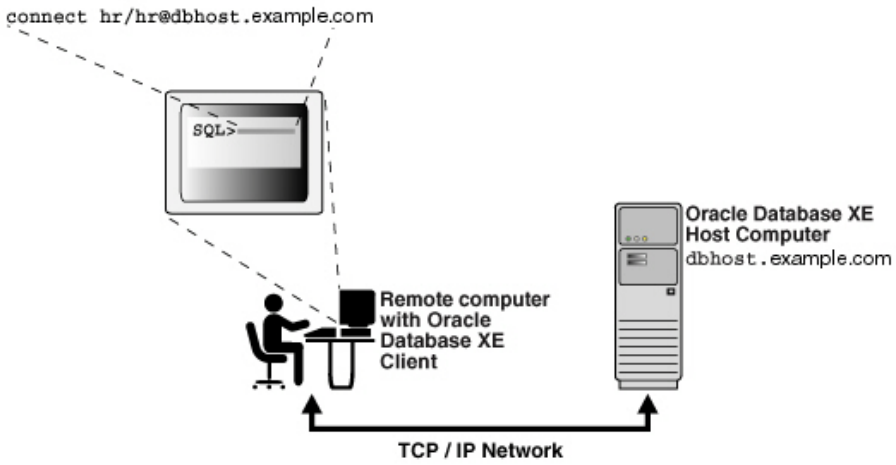
- `//` is optional
- *`host`* is the host name or IP address of the computer that is running Oracle Database XE
- *`port`* (optional) is the TCP port number on which the Oracle Net listener is listening. If not specified, the default port number 1521 is assumed.
- *`service_name`* (optional) is the name of the database service to which to connect. For Oracle Database XE, the service name is `XE`. If *`service_name`* is omitted, Oracle Database XE Client appends a request for the *default database service*, which is configured during installation as `XE`.

> **Note:** Only Oracle Database XE supports the notion of a default database service. If you connect remotely from any Oracle client software other than Oracle Database XE, you must include the `XE` service name.

For example, Figure 3–2 shows a user connecting remotely with the SQL Command Line and entering a `connect` command that includes a connect string that supplies: the

user name `hr`, the password `hr`, and the host name `dbhost.example.com`. This connect string connects to the default database service (`XE`) on the default port (`1521`).

Figure 3–2 Remote Connection



Environment Variables

Before making a remote connection from Linux, you must set environment variables. See "Setting Environment Variables" on page 3-4 for more information.

Remote Connection Examples

In the following examples of the SQL Command Line `connect` commands, Oracle Database XE is running on the host computer `mydbserver.example.com`.

Example 1 This example initiates a remote connection from Oracle Database XE, using the default port number.

```
CONNECT system/mypassword@mydbserver.example.com
```

Example 2 This example initiates a remote connection from Oracle Database XE, using a nondefault port number (1522):

```
CONNECT system/mypassword@mydbserver.example.com:1522
```

Example 3 This example initiates a remote connection using the default port number, and includes the optional service name.

```
CONNECT system/mypassword@mydbserver.example.com/XE
```

Setting Environment Variables

the SQL Command Line (SQL*Plus) and other Oracle utilities retrieve configuration information from operating system environment variables. This section explains how to set these environment variables, and contains the following topics:

- Setting Environment Variables on the Windows Platform on page 3-5
- Setting Environment Variables on the Linux Platform on page 3-5

Setting Environment Variables on the Windows Platform

On the Windows platform, environment variables are stored in the Windows registry, and are automatically set for you. You need not take any action involving environment variables before initiating a database connection.

Setting Environment Variables on the Linux Platform

On the Linux platform, before running the SQL Command Line or other Oracle utilities from a terminal session, you must set some environment variables for that session. The procedure for setting environment variables depends on whether you are connecting locally, or connecting remotely from Oracle Database XE.

Setting Environment Variables for a Local Connection

To set environment variables when connecting locally, enter one of the following commands in a terminal session.

For Bourne, Korn, or Bash shell:

```
source /u01/app/oracle/product/11.2.0/xe/bin/oracle_env.sh
```

For C shell:

```
source /u01/app/oracle/product/11.2.0/xe/bin/oracle_env.csh
```

See "Environment Variables Reference for Linux" on page 3-7 for more information.

Note: There is no need to set environment variables before running the SQL Command Line from the K menu (KDE) or Applications menu (Gnome).

Setting Environment Variables for a Remote Connection

To set environment variables when connecting remotely from Oracle Database XE, enter one of the following commands in a terminal session at the remote computer:

For Bourne, Korn, or Bash shell:

```
source /usr/lib/oracle/xe/app/oracle/product/11.2.0/client/bin/oracle_env.sh
```

For C shell:

```
source /usr/lib/oracle/xe/app/oracle/product/11.2.0/client/bin/oracle_env.csh
```

See "Environment Variables Reference for Linux" on page 3-7 for more information.

See Also:

- "About Local and Remote Connections" on page 3-1
- "Connecting Locally with the SQL Command Line"
- "Connecting Remotely with the SQL Command Line"

Connecting Locally with the SQL Command Line

Connecting locally means running the SQL Command Line (SQL*Plus) and Oracle Database XE on the same computer. There are two ways to start a local connection with the SQL Command Line:

- From the desktop

- From a terminal session (Linux) or command window (Windows)

Starting the SQL Command Line from the Desktop

To start the SQL Command Line from the desktop and connect locally:

1. Do one of the following:
   - On Windows: Click **Start**, point to **Programs (or All Programs)**, point to **Oracle Database 11g Express Edition**, and then select **Run SQL Command Line**.
   - On Linux with Gnome: In the Applications menu, point to **Oracle Database 11g Express Edition**, and then select **Run SQL Command Line**.
   - On Linux with KDE: Click the icon for the K Menu, point to **Oracle Database 11g Express Edition**, and then select **Run SQL Command Line**.

   A the SQL Command Line command window opens.
2. At the SQL Command Line prompt, enter the following command:

   ```
   CONNECT username/password
   ```

   For example, to connect as user `HR` with the password `PEOPLE`, enter the following command:

   ```
   CONNECT HR/PEOPLE
   ```

Starting the SQL Command Line from a Terminal Session or Command Window

To start the SQL Command Line from a terminal session or command window and connect locally:

1. If not already open, open a terminal session (Linux) or a command window (Windows).
2. (Linux only) If the required environment variables are not already set for your session, set them as described in "Setting Environment Variables on the Linux Platform" on page 3-5.
3. Enter the following command at the operating system prompt:

   ```
   sqlplus /nolog
   ```

4. At the SQL Command Line prompt, enter the following command:

   ```
   CONNECT username/password
   ```

   For example, to connect as user `HR` with the password `PEOPLE`, enter the following command:

   ```
   CONNECT HR/PEOPLE
   ```

See Also:

- "About Local and Remote Connections" on page 3-1

Connecting Remotely with the SQL Command Line

Connecting remotely means running the SQL Command Line (SQL*Plus) on one computer (the remote computer), and then initiating a connection to Oracle Database XE on a different computer.

To initiate a remote connection from the SQL Command Line using the Oracle Database XE:

1. On the remote computer, start a terminal session (Linux) or open a command window (Windows.)

   If prompted for host credentials, log in to the remote computer.

2. (Linux only) If the required environment variables are not already set for your session, set them as described in "Setting Environment Variables on the Linux Platform" on page 3-5.

3. Enter the following command at the operating system prompt:

   ```
   sqlplus /nolog
   ```

4. Enter a `CONNECT` command at the SQL Command Line prompt, supplying a connect string.

   ```
   CONNECT username/password@[//]host[:port][/service_name]
   ```

   See "About Remote Connections" on page 3-2 for a description and examples of connect strings.

See Also: "About Local and Remote Connections" on page 3-1

Environment Variables Reference for Linux

This section provides reference information for setting environment variables on Linux for the following two scenarios:

- Connecting locally
- Connecting remotely from Oracle Database XE.

Table 3–1 on page 3-7 lists the environment variables that you must set for each of these scenarios. Table 3–2 on page 3-8 provides environment variable descriptions and required values.

Table 3–1 Required Linux Environment Variables for Connecting with Oracle Utilities

Connection Type	Required Environment Variables
Local	ORACLE_SID ORACLE_HOME PATH NLS_LANG LD_LIBRARY_PATH
Remote, using Oracle Database XE	ORACLE_HOME PATH NLS_LANG LD_LIBRARY_PATH SQLPATH

Table 3–2 Environment Variable Descriptions and Values for Linux

Variable Name	Description	Required Value
ORACLE_SID	Oracle Instance ID	XE
ORACLE_HOME	Oracle home directory	For local connection: /usr/lib/oracle/xe/app/oracle/product/11.2.0/server For remote connection with Oracle Database XE: /usr/lib/oracle/xe/app/oracle/product/11.2.0/client
PATH	Search path for executables. (Must add $ORACLE_HOME/bin to the path.)	For Bourne, Korn, or Bash shell: $ORACLE_HOME/bin:$PATH For C shell: $ORACLE_HOME/bin:${PATH}
NLS_LANG	Locale (language and territory used by client applications and the database; character set used by client applications)	(The desired language, territory, and character set. See *Oracle Database Express Edition Installation Guide for Linux x86-64* for details.) Defaults to AMERICAN_AMERICA.US7ASCII
LD_LIBRARY_PATH	Search path for shared libraries. (Must add $ORACLE_HOME/lib to the path.	$ORACLE_HOME/lib:$LD_LIBRARY_PATH
SQLPATH	Search path used by the SQL Command Line (SQL*Plus) for *.sql scripts. Contains a colon-separated list of paths. Must include the location of the site profile script, glogin.sql.	$ORACLE_HOME/sqlplus/admin

Example

The following are the Bash shell commands that set the required environment variables for a local connection on a Linux installation in the United States:

```
ORACLE_SID=XE;export ORACLE_SID
ORACLE_HOME=/usr/lib/oracle/xe/app/oracle/product/11.2.0/server;export ORACLE_HOME
PATH=$ORACLE_HOME/bin:$PATH;export PATH
NLS_LANG=AMERICAN_AMERICA.AL32UTF8;export NLS_LANG
LD_LIBRARY_PATH=$ORACLE_HOME/lib:$LD_LIBRARY_PATH;export LD_LIBRARY_PATH
```

Environment Variable Scripts

Oracle Database XE and Oracle Database XE ship with two shell scripts that you can use to easily set environment variables. The scripts are located in $ORACLE_HOME/bin and are named as follows:

oracle_env.sh (for Bourne, Korn, or Bash shell)
oracle_env.csh (for C shell)

You can invoke these scripts from within dot files so that environment variables are set automatically each time that you start a new terminal session (start a new shell). The following is an example of a command that you can add to the .cshrc file in your home directory:

```
source /usr/lib/oracle/xe/app/oracle/product/11.2.0/server/bin/oracle_env.csh
```

4

Managing Network Connections

This section explains how to manage network connections to the database. It includes the following topics:

See Also: "Connecting to the Database" on page 3-1

About Network Connections and the Oracle Net Listener

Oracle Database Express Edition (Oracle Database XE) supports connections from client applications both remotely over the network and locally. Remote client applications and the database communicate through Oracle Net, which is a software layer that resides both on the remote computer and on the Oracle Database XE host computer. Oracle Net establishes the connection between the client application and the database, and exchanges messages between them using TCP/IP. Oracle Net is automatically installed when you install Oracle Database XE and Oracle Database Express Edition Client.

Included with Oracle Net in an Oracle Database XE installation is the **Oracle Net listener**, commonly known as the listener. It is the host process that listens on specific TCP/IP ports for connection requests. When the listener receives a valid connection request from a client application, it routes the connection request to the database. The client application and the database then communicate directly.

Table 4–1 lists the types of connection requests that the listener handles.

Table 4–1 Types of Connection Requests Handled by the Listener

Connection Request Type	Default TCP Port Number	Used For
Database	1521	Database connections using Oracle Net over TCP/IP. Examples include: ■ Remote connection from the SQL Command Line. ■ Remote connection from a Java application that connects with JDBC.

Table 4–1 (Cont.) Types of Connection Requests Handled by the Listener

Connection Request Type	Default TCP Port Number	Used For
HTTP	8080	Database connections using the HTTP protocol. Examples include: ■ Accessing Oracle Application Express applications that you create on the local system. ■ Accessing the Oracle XML DB repository. Oracle XML DB is the Oracle Database XE feature that provides high-performance, native XML storage and retrieval. Through the Oracle XML DB repository, you can access XML data with the HTTP and WebDAV (Web folder) protocols. See *Oracle XML DB Developer's Guide* for more information.

Note: The listener can also handle FTP connection requests for the Oracle XML DB repository. For security reasons, FTP requests are disabled when you install Oracle Database XE. See *Oracle XML DB Developer's Guide* for more information.

You can disable certain types of connection requests by manually stopping the listener, and reenable them by restarting the listener. (The listener is automatically started when you install Oracle Database XE and when you restart the Oracle Database XE host computer.)

Table 4–2 indicates the types of connections that require the listener to be started.

Table 4–2 Connections That Require the Listener

Connection Type	Local Connection	Remote Connection
Database	Not required	Required
HTTP	Required	Required

As the table shows, stopping the listener disables all connection requests except local database connection requests.

Configuring the Listener

You can change the ports that the listener listens on, both for database and HTTP connection requests, either during the Oracle Database XE installation process, or at a later time after installation. See "Changing Listener Port Numbers" on page 4-5 for details on changing port numbers after installation.

Note: The Windows installation process prompts for the port number for HTTP requests only if the default port number, 8080, is already in use. The Linux configuration script always prompts for HTTP port number.

Remote HTTP Connections Initially Disabled

As a security measure, remote HTTP connection requests are initially disabled. This means that remote users cannot use the Oracle Database XE graphical user interface

until you enable remote HTTP connections. See "Enabling Remote HTTP Connection to the Database" on page 4-8 for instructions.

See Also:

- "Stopping and Starting the Listener" on page 4-4
- "Changing Listener Port Numbers" on page 4-5
- "Connecting to the Database" on page 3-1
- "Viewing Listener Status"

Viewing Listener Status

You view listener status to determine if the listener is started and to check listener properties (such as the TCP/IP port numbers that the listener is listening on). You do so with the Listener Control (`lsnrctl`) utility.

To view listener status:

1. Do one of the following:
   - On Linux: Start a terminal session and log in to the Oracle Database XE host computer with the `oracle` user account.
   - On Windows: Log in to the Oracle Database XE host computer as the user who installed Oracle Database XE, and then open a command window.
2. On Linux, ensure that environment variables are set according to the instructions in "Setting Environment Variables on the Linux Platform" on page 3-5.
3. Enter the following command:

   ```
   LSNRCTL STATUS
   ```

If the listener is not started, the command displays the following error messages:

```
TNS-12541: TNS:no listener
 TNS-12560: TNS:protocol adapter error
  TNS-00511: No listener
```

If the listener is started, the command displays a report that looks something like this:

```
Connecting to (DESCRIPTION=(ADDRESS=(PROTOCOL=IPC)(KEY=EXTPROC1)))
STATUS of the LISTENER
------------------------
Alias                     LISTENER
Version                   TNSLSNR for 32-bit Windows: Version 11.2.0.2.0 - Production
Start Date                08-MAR-2011 16:12:26
Uptime                    0 days 1 hr. 57 min. 59 sec
Trace Level               off
Security                  ON: Local OS Authentication
SNMP                      OFF
Default Service           XE
Listener Parameter File
C:\oraclexe\app\oracle\product\11.2.0\server\network\admin\listener.ora
Listener Log File
C:\oraclexe\app\oracle\diag\tnslsnr\user1-pc\listener\alert\log.xml
Listening Endpoints Summary...
  (DESCRIPTION=(ADDRESS=(PROTOCOL=ipc)(PIPENAME=\\.\pipe\EXTPROC1ipc)))
  (DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=user1-pc.example.com)(PORT=1521)))
```

```
(DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=user1-pc.example.com)(PORT=8080))(Presentation=HTTP)(Sess
ion=RAW))
Services Summary...
Service "CLRExtProc" has 1 instance(s).
  Instance "CLRExtProc", status UNKNOWN, has 1 handler(s) for this service...
Service "PLSExtProc" has 1 instance(s).
  Instance "PLSExtProc", status UNKNOWN, has 1 handler(s) for this service...
Service "XEXDB" has 1 instance(s).
  Instance "xe", status READY, has 1 handler(s) for this service...
Service "xe" has 1 instance(s).
  Instance "xe", status READY, has 1 handler(s) for this service...
The command completed successfully
```

In the preceding report, the bold text indicates particulars to look for. Their meanings are as follows:

- `(PORT=1521)`

  The listener is listening for database connections through Oracle Net on port 1521.

- `(PORT=8080))(Presentation=HTTP)`

  The listener is listening for database connections through HTTP on port 8080.

- `Service "XE" has 1 instance(s)` and `Instance "XE", status READY`

  Oracle Database XE is properly registered with the listener and is ready to accept connections.

Stopping and Starting the Listener

The listener is configured to start automatically when you install Oracle Database Express Edition (Oracle Database XE), and whenever the computer running Oracle Database XE is restarted. The following are reasons why you may want to stop and restart the listener:

- To recover from system errors
- To temporarily block remote connection requests

  You stop the listener to disable remote connection requests, and restart the listener to enable them.

- To change the TCP port number that the listener listens on

  See "Changing Listener Port Numbers" on page 4-5 for more information.

Stopping the Listener

To stop the listener:

1. Do one of the following:
   - On Linux: Start a terminal session and log in to the Oracle Database XE host computer with the `oracle` user account.
   - On Windows: Log in to the Oracle Database XE host computer as the user who installed Oracle Database XE, and then open a command window.
2. On Linux, ensure that environment variables are set according to the instructions in "Setting Environment Variables on the Linux Platform" on page 3-5.
3. Enter the following command:

```
LSNRCTL STOP
```

The command displays the following output if successful.

On Linux:

```
Connecting to (DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=myhost)(PORT=1521)))
The command completed successfully
```

On Windows:

```
Connecting to (DESCRIPTION=(ADDRESS=(PROTOCOL=IPC)(KEY=EXTPROC1)))
The command completed successfully
```

If the listener was already stopped, the command displays one or more `TNS: no listener` messages.

Starting the Listener

To start the listener:

1. Do one of the following:
   - On Linux: Start a terminal session and log in to the Oracle Database XE host computer with the `oracle` user account.
   - On Windows: Log in to the Oracle Database XE host computer as the user who installed Oracle Database XE, and then open a command window.
2. On Linux, ensure that environment variables are set according to the instructions in "Setting Environment Variables on the Linux Platform" on page 3-5.
3. Enter the following command:

   ```
   LSNRCTL START
   ```

   If successful, the command displays the report shown in "Viewing Listener Status" on page 4-3.

Note: If you stop and then start the listener while the database is running, it may take a minute or so for the database to reregister with the listener and to begin accepting connections. To determine if the database is ready to accept connections, run the `lsnrctl status` command repeatedly until you see the following lines in the report:

```
Service "XE" has 1 instance(s).
  Instance "XE", status READY, has 1 handler(s) for this service...
```

See Also: "Viewing Listener Status" on page 4-3

Changing Listener Port Numbers

You would need to change a default listener port number only if there were a port number conflict with another TCP/IP service. You are given the opportunity to change listener port numbers during installation (Windows) or configuration (Linux). This section explains how to change port numbers after installation or configuration. It contains the following topics:

- "Changing the Listener Port Number for Database Connection Requests" on page 4-6

- "Changing the Listener Port Number for HTTP Connection Requests" on page 4-7

See Also: "About Network Connections and the Oracle Net Listener" on page 4-1

Changing the Listener Port Number for Database Connection Requests

If you change the listener port number for database connection requests, you must ensure that all future database connection requests use the new port number. This means that connection requests such as those discussed in "Connecting Remotely with the SQL Command Line" on page 3-6 must explicitly include the port number.

For example, if you change the port number for database connection requests to 1522, subsequent the SQL Command Line (SQL*Plus) `connect` statements must be similar to the following (assuming a connection from Oracle Database Express Edition Client):

```
connect system/mypassword@myhost.example.com:1522
```

Example: Changing Listener Port Number for Database Connection Requests

Assume that your Oracle Database XE host computer is named `myhost.example.com` and that you want to install a new software package on this computer that requires TCP port number 1521. Assume also that the port number for that software package cannot be configured, and that you must therefore resolve the port number conflict by reconfiguring Oracle Database XE. You decide to change the listener port number for database connection requests to 1522.

To change the listener port number for database connection requests to 1522:

1. Stop the listener.

   See "Stopping and Starting the Listener" on page 4-4 for instructions.

2. Open the file `listener.ora` with a text editor.

   Table 4–3 shows the location of this file on each platform.

Table 4–3 Location of the listener.ora File

Platform	Location
Linux	/usr/lib/oracle/xe/app/oracle/product/11.2.0/server/network/admin/
Windows	c:\oraclexe\app\oracle\product\11.2.0\server\NETWORK\ADMIN\

3. Locate the following section of the file:

```
LISTENER =
  (DESCRIPTION_LIST =
    (DESCRIPTION =
      (ADDRESS = (PROTOCOL = IPC)(KEY = EXTPROC1))
      (ADDRESS = (PROTOCOL = TCP)(HOST = myhost)(PORT = 1521))
    )
  )
```

   Note that the line indicated in bold may or may not be present in the file.

4. Change the text `(PORT = 1521)` to `(PORT = 1522)`.
5. Save the modified `listener.ora` file.
6. Start the listener.

   See "Stopping and Starting the Listener" on page 4-4 for instructions.

7. Start the SQL Command Line and connect to the database as user `SYSTEM`.

   See "Connecting Locally with the SQL Command Line" on page 3-5 for instructions. You must supply the `SYSTEM` password. You set this password upon installation (Windows) or configuration (Linux) of Oracle Database XE.

8. Enter the following two commands:

```
ALTER SYSTEM SET LOCAL_LISTENER =
"(ADDRESS=(PROTOCOL=TCP)(HOST=myhost.example.com)(PORT=1522))";

ALTER SYSTEM REGISTER;
```

9. Exit the SQL Command Line and run the `lsnrctl status` command to verify the port number change.

   The new port number should be displayed in the Listening Endpoints Summary section of the status report, and the report should include the following lines:

```
Service "XE" has 1 instance(s).
  Instance "XE", status READY, has 1 handler(s) for this service...
```

Changing the Listener Port Number for HTTP Connection Requests

If you change the listener port number for HTTP connection requests, you must ensure that all future HTTP connection requests use the new port number.

For example, if you change the listener port number for HTTP requests to 8087, you must use the following URL to access the Oracle Application Express login page locally:

```
http://127.0.0.1:8087/apex
```

> **Note:** When you change the listener port number for HTTP, the Get Started menu item on the desktop can no longer open the Database Home Page. The following procedure contains an optional step that explains how to modify this menu item to function with the new port number.

To change the listener port number for HTTP connection requests:

1. Do one of the following:
   - On Linux: Start a terminal session and log in to the Oracle Database XE host computer with the `oracle` user account.
   - On Windows: Log in to the Oracle Database XE host computer as the user who installed Oracle Database XE, and then open a command window.
2. On Linux, ensure that environment variables are set according to the instructions in "Setting Environment Variables on the Linux Platform" on page 3-5.
3. Ensure that the listener is started.

   See "Viewing Listener Status" on page 4-3 and "Stopping and Starting the Listener" on page 4-4 for instructions.
4. Enter the following command at the operating system prompt to start the SQL Command Line:

```
sqlplus /nolog
```

5. At the SQL Command Line prompt, enter the following command:

   ```
   CONNECT SYSTEM/password
   ```

   where *password* is the `SYSTEM` password that you set upon installation (Windows) or configuration (Linux) of Oracle Database XE.

6. At the SQL Command Line prompt, enter the following command:

   ```
   EXEC DBMS_XDB.SETHTTPPORT(nnnn);
   ```

   where *nnnn* represents the new port number to use for HTTP connection requests. Be certain that you select a port number that is not already in use.

   For example, to use port number 8087 for HTTP connection requests, enter this command:

   ```
   EXEC DBMS_XDB.SETHTTPPORT(8087);
   ```

   If the command is successful, the following message is displayed:

   ```
   PL/SQL procedure successfully completed.
   ```

7. Exit the SQL Command Line (by entering the `exit` command) and view listener status to verify the port number change.

   See "Viewing Listener Status" on page 4-3 for instructions. The new port number is displayed in the Listening Endpoints Summary section of the status report.

8. (Optional) To enable the Get Started command on the desktop to work with the new port number, change the port number in the script or shortcut that this command uses. The following table shows the script or shortcut that you must change on each platform.

Platform	Script or Shortcut to Change for Changing the Listener Port for HTTP Connection Requests
Linux	/usr/lib/oracle/xe/app/oracle/product/11.2.0/server/config/scripts/DatabaseHomePage.sh
Windows	C:\oraclexe\app\oracle\product\11.2.0\server\Database_homepage

Enabling Remote HTTP Connection to the Database

After installation, database connection requests with the HTTP protocol are enabled only on the computer on which you installed Oracle Database XE. This means that remote users cannot access applications that you create using Oracle Application Express on the local system. As an administrator, you can enable HTTP access for remote users, thereby enabling them to access the Oracle Database XE graphical user interface.

> **Security Note:** With remote HTTP access to Oracle Database XE, all information exchanged between the browser and the database is in clear text—that is, unencrypted—including database user names and passwords. If this is cause for concern, do not enable remote HTTP connection to the database.

Enabling Remote HTTP Connection with the SQL Command Line

To enable remote HTTP connection requests using the SQL Command Line:

1. Start the SQL Command Line and connect to the database as user `SYSTEM`. Provide the `SYSTEM` password that you assigned upon installation (Windows) or configuration (Linux) of Oracle Database XE.

   See "Connecting Locally with the SQL Command Line" on page 3-5 or "Connecting Remotely with the SQL Command Line" on page 3-6 for instructions.

2. At the SQL Command Line prompt, enter the following command:

   ```
   EXEC DBMS_XDB.SETLISTENERLOCALACCESS(FALSE);
   ```

5

Managing Database Memory

This section provides background information on memory management in Oracle Database Express Edition and describes how to adjust memory allocation.

The following topics are covered:

Automatic Memory Management

Oracle Database XE uses **automatic memory management**, which you cannot disable. With automatic memory management in Oracle Database XE, the database dynamically exchanges memory between the System Global Area (SGA) and the instance Program Global Area (PGA) as needed to meet processing demands. The database also dynamically tunes the sizes of the individual SGA components and the sizes of the individual PGAs.

For an overview of Oracle Database memory management, including the SGA and PGA, see "About Memory Management".

About Memory Management

This section provides background information on memory management in Oracle Database Express Edition (Oracle Database XE). It includes the following topics:

Memory Allocation Overview

To support database operation, Oracle Database XE needs to start a set of processes, called background processes, and needs to allocate some memory in the host computer. The background processes and allocated memory together make up an **Oracle instance**.

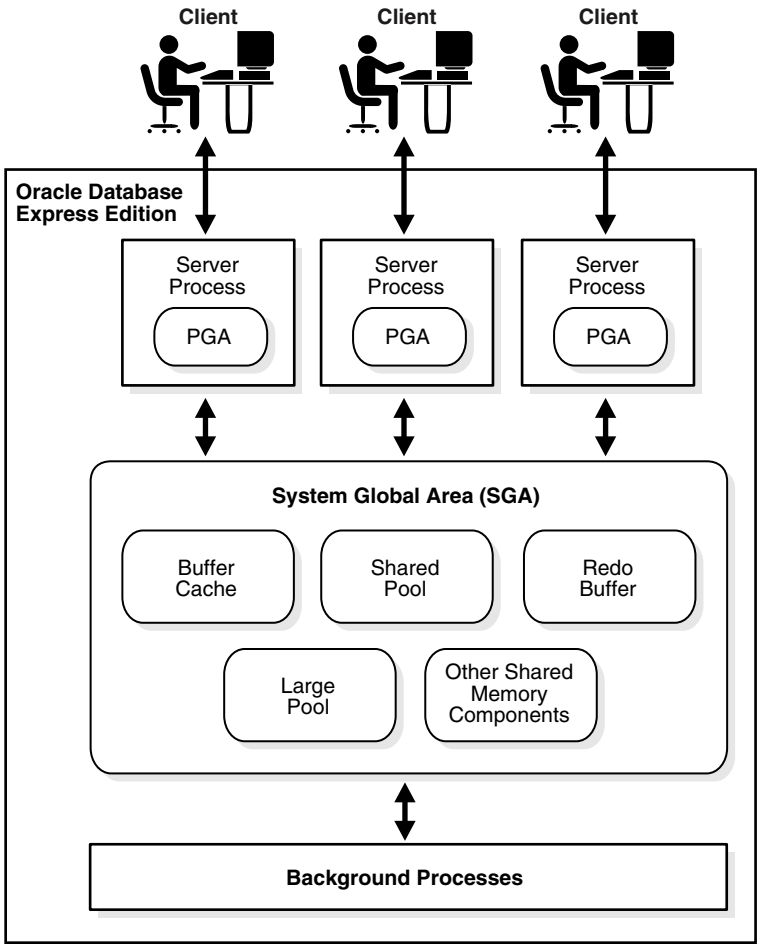
There are two types of memory that the Oracle instance allocates:

- System global area (SGA)—A shared memory area that contains data buffers and control information for the instance. The SGA is divided into separate buffer areas and data pools. These are described in "SGA Components" on page 5-2.

- Program global area (PGA)—A memory area used by a single Oracle server process. An Oracle **server process** is a process that services a client's requests. Oracle Database XE creates a new server process whenever it receives a new database connection request. Each new server process then allocates its own private PGA area. The PGA is used to process SQL statements and to hold logon and other session information.

Figure 5–1 illustrates memory allocation in Oracle Database XE.

Figure 5–1 Memory Allocation in Oracle Database XE



The amount of memory allocated to the SGA and PGA directly affects the performance of your database. The SGA and PGA sizes are configured automatically when you install Oracle Database XE. See "SGA and PGA Sizes" on page 5-3 for a discussion of when you might change them.

See Also:

- "Example: Changing SGA and PGA Aggregate Sizes (Advanced Users)" on page 5-4

SGA Components

The SGA has several components, as listed in the following table. Oracle Database XE automatically tunes the individual sizes of these components for optimal performance.

Component	Description
Buffer cache	The buffer cache is the component of the SGA that acts as the buffer to store any data being queried or modified. All clients connected to the database share access to the buffer cache. The buffer cache helps avoid repeated access from the physical disk, a time-consuming operation.
Shared pool	The shared pool caches operational information and code that can be shared among users. For example: ■ SQL statements are cached so that they can be reused. ■ Information from the data dictionary, such as user account data, table and index descriptions, and privileges, is cached for quick access and reusability. ■ Stored procedures are cached for faster access.
Redo log buffer	The redo log buffer improves performance by caching redo information (used for instance recovery) until it can be written at once and at a more opportune time to the physical redo log files that are stored on disk. Redo information and redo log files are discussed in "Online Redo Log Files" on page 6-6.
Large pool	The large pool is an optional area that is used for buffering large I/O requests for various server processes.

SGA and PGA Sizes

Caution: Oracle Database XE uses Automatic Memory Management, which usually provides the best memory configuration based on your resources and workload. Do not set any memory-related parameters unless you fully understand the consequences.

If you change any memory-related parameter values, test them fully before implementing the changes in a production environment.

The default sizes for the SGA and PGA are set upon installation, based on the total amount of physical memory in your system. Rather than changing the sizes of individual SGA components, you can change the overall size of the SGA by setting a parameter called *SGA Target*, and Oracle Database XE automatically adjusts the sizes of the individual SGA components, continuously tuning these sizes to optimize performance. Similarly, rather than changing the size of individual PGAs, you can change the total amount of memory allocated for the collection of PGAs, and Oracle Database XE adjusts individual PGA sizes as needed. The collection of PGAs is known as the **PGA Aggregate**. You change the PGA Aggregate maximum size by setting a parameter called *PGA Aggregate Target*.

Note: Oracle Database XE always allocates the full amount of memory specified by the SGA Target parameter. That is, the current SGA size is always equal to SGA Target. In contrast, the current size of the PGA Aggregate may be less than the amount specified by the PGA Aggregate Target parameter. The database allocates more memory for the PGA Aggregate as needed, up to the maximum indicated by PGA Aggregate Target.

The maximum amount of memory that Oracle Database XE allows for the SGA and PGA Aggregate is **1 gigabyte (GB)**. If you attempt to change memory allocation so

that the sum of the SGA size and PGA Aggregate size exceeds 1 GB, Oracle Database XE issues an error message. (For SGA changes, the error message does not appear until you restart the database.)

The only circumstances under which you should need to change SGA and PGA Aggregate sizes are the following:

- You add physical memory to the computer running Oracle Database XE and want to allocate more to the database.

  In this case, increase both the SGA and PGA Aggregate sizes, maintaining roughly the original ratio of SGA size to PGA Aggregate size.

- You receive an error due to insufficient memory.

  If the error message indicates insufficient memory for an SGA component, increase the SGA size. Examples of such errors include the following:

  ```
  ORA-04031: unable to allocate n bytes of shared memory
  ORA-00379: no free buffers available in buffer pool...
  ```

  If the error message indicates insufficient memory for a process, increase the PGA Aggregate size. An example of such an error is the following:

  ```
  ORA-04030: out of process memory when trying to allocate n bytes
  ```

  If you are not sure whether the insufficient memory error involves the SGA or PGA, increase both SGA and PGA Aggregate sizes, maintaining roughly the original ratio of SGA size to PGA Aggregate size.

For SGA size changes, you must shut down and restart the database for the changes to take effect. For PGA Aggregate size changes, there is no need to restart the database.

Example: Changing SGA and PGA Aggregate Sizes (Advanced Users)

This example is for advanced users. It assumes that you have sufficient knowledge to make a sound decision to override the Oracle Database XE default memory management, and that you will test the changes thoroughly before implementing them in a production environment.

Suppose you just upgraded the computer running Oracle Database Express Edition (Oracle Database XE) to add 1 gigabyte (GB) or more of system memory, and you want to increase the memory for the database by 250 MB. Of this 250 MB, you want to add 200 MB to the SGA and 50 MB to the PGA Aggregate.

To accomplish this, you can the DBA navigator in SQL Developer to check the current values of the SGA_TARGET and PGA_AGGREGATE TARGET initialization parameters, and then use the ALTER SYSTEM command to reset the values of these parameters, as follows:

1. In SQL Developer, click **View**, then **DBA** to display the DBA navigator.
2. If a connection to the `SYSTEM` user does not already exist in that navigator, add one by clicking the Connections node, selecting **Add Connection**, and completing the actions for adding the `SYSTEM` account connection.
3. In the DBA navigator, expand the `SYSTEM` connection, expand **Database Configuration**, and click **Initialization Parameters**.

   Note the current values for `pga_aggregate_target` and `sga_target`, and calculate the desired new values. Assume for this example that the desired new values are 140 megabytes for `pga_aggregate_target` and 472 megabytes for `sga_target`.

4. In the Connections navigator, select the `SYSTEM` connection; and if a SQL Worksheet is not already open for that connection, right-click `SYSTEM` and select **Open SQL Worksheet**.
5. In the SQL Worksheet for the `SYSTEM` connection, enter the following command lines:

```
ALTER SYSTEM SET pga_aggregate_target = 140 M;
ALTER SYSTEM SET sga_target = 472 M;
```

6. In the SQL Worksheet for the `SYSTEM` connection, click the Run Script icon to execute these statements.
7. At the next convenient time, shut down and restart the database to enable the SGA size changes to take effect.

See Also: "About Memory Management" on page 5-1

6

Managing Database Storage

This section describes the storage structures of your database, and explains how to monitor and manage the amount of storage that is in use and available for the database and its backups. It contains the following topics:

About the Database Storage Structures

Oracle Database Express Edition (Oracle Database XE) is composed of the following storage structures:

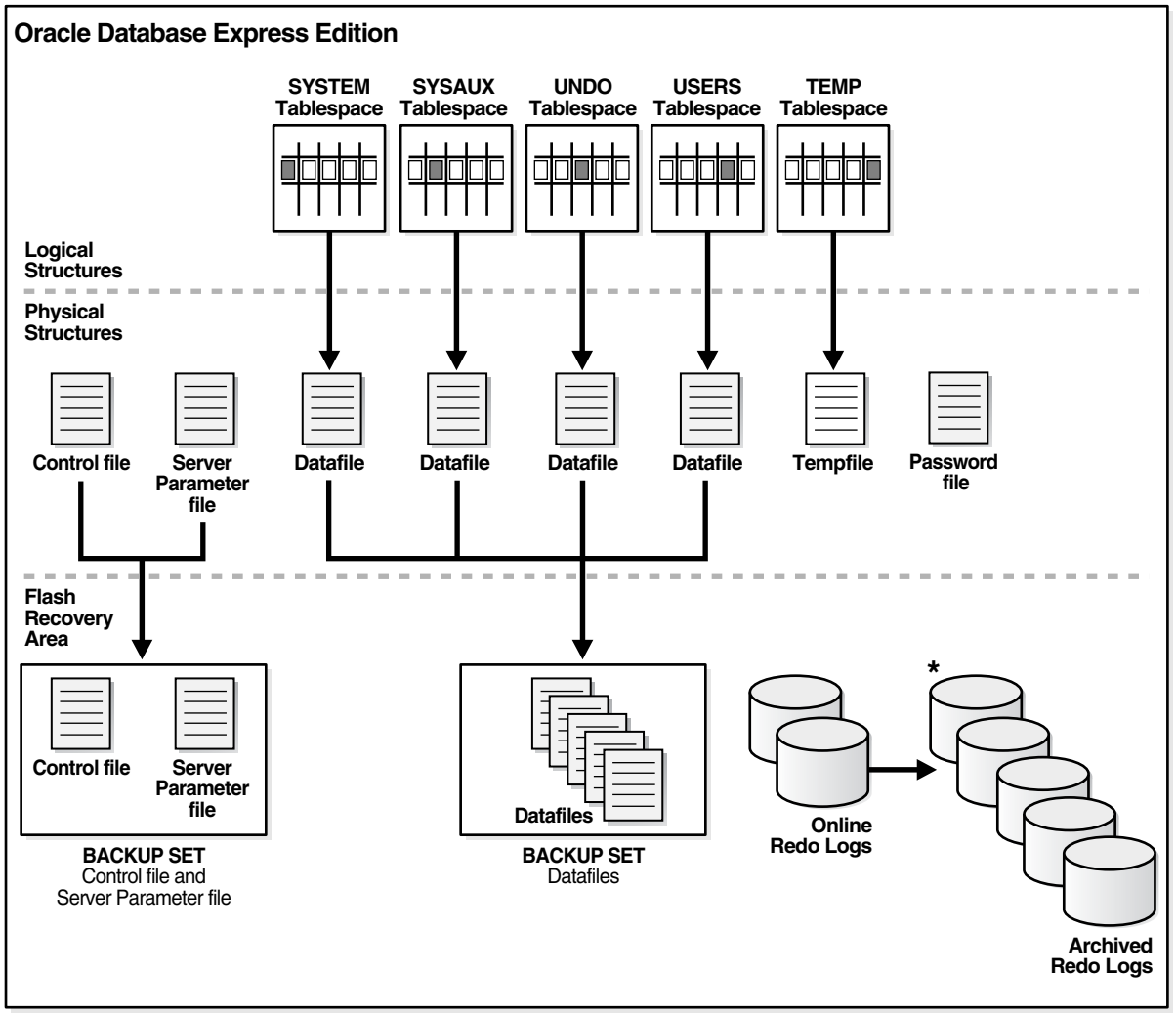
- **Logical structures** such as tablespaces are created and recognized by the database only, and are not known to the operating system.
- **Physical structures** are those that can be seen and operated on from the operating system, such as the physical files that store data on disk.
- **Recovery-related structures** such as redo logs and database backups are used to recover the database after an operating system failure, Oracle instance failure, or media (disk) failure. Recovery-related structures are stored in an automatically managed disk storage area called the *flash recovery area*.

Oracle Database XE completely automates the management of its logical and physical structures and flash recovery area storage. You use the Oracle Database XE graphical user interface to monitor these structures, mostly to understand how much storage your applications have used so far, how much free storage remains, and whether more space is needed for backups.

The following sections provide a closer look at the database and its storage structures. Refer to Figure 6–1 as you review these sections.

- Server Parameter File on page 6-4
- Password File on page 6-5
- Recovery-Related Structures in the Flash Recovery Area on page 6-5

Figure 6–1 Database Storage Structure



* Archived Redo Logs present only after turning on log archiving (ARCHIVELOG mode)

Database

The **database** is the collection of logical and physical structures that together contain all the data and metadata for your applications. The database also contains control structures (such as *control files*) that it needs for startup and operation. All of these structures are described in subsequent sections, and are summarized in Figure 6–1.

The Oracle Database XE instance (which consists of the Oracle Database XE background processes and allocated memory) works with a single database only. Rather than enabling you to create multiple databases to accommodate different applications, Oracle Database XE uses a single database, and accommodates multiple applications by enabling you to separate data into different *schemas*. See "About User Accounts" on page 7-1 for more information about schemas.

The maximum database size in Oracle Database XE is 5 gigabytes (GB). This includes between 0.5 and 0.9 GB for the data dictionary, internal schemas, and temporary space, which leaves just over 4.0 GB for user data.

> **See Also:** See "Internal User Accounts" on page 7-2 for information about internal schemas, and "Tablespaces" on page 6-3 for information about temporary space.

Tablespaces

A database consists of one or more tablespaces. A **tablespace** is a logical grouping of one or more physical datafiles or tempfiles, and is the primary structure by which the database manages storage.

There are various types of tablespaces, including the following:

- Permanent tablespaces

  These tablespaces are used to store system and user data. Permanent tablespaces consist of one or more datafiles. In Oracle Database XE, all your application data is by default stored in the tablespace named `USERS`. This tablespace consists of a single datafile that automatically grows (*autoextends*) as your applications store more data.

- Temporary tablespaces

  Temporary tablespaces improve the concurrency of multiple sort operations, and reduce their overhead. Temporary tablespaces are the most efficient tablespaces for disk sorts. Temporary tablespaces consist of one or more tempfiles. Oracle Database XE automatically manages storage for temporary tablespaces.

- Undo tablespace

  Oracle Database XE transparently creates and automatically manages *undo data* in this tablespace.

  When a transaction modifies the database, Oracle Database XE makes a copy of the original data before modifying it. The original copy of the modified data is called **undo data**. This information is necessary for the following reasons:

  – To undo any uncommitted changes made to the database in the event that a rollback operation is necessary. A rollback operation can be the result of a user specifically issuing a `ROLLBACK` statement to undo the changes of a misguided or unintentional transaction, or it can be part of a recovery operation.

  – To provide read consistency, which means that each user can get a consistent view of data, even while other uncommitted changes may be occurring against the data. For example, if a user issues a query at 10:00 a.m. and the query runs for 15 minutes, then the query results should reflect the entire state of the data at 10:00 a.m., regardless of updates or inserts by other users during the query.

    See *Oracle Database Concepts* for a discussion of read consistency.

  – To support the Flashback Query feature, which enables you to view or recover older versions of data. See "Viewing and Restoring Historical Data with Flashback Query" on page 11-7 for more information.

Table 6–1 describes the tablespaces included in Oracle Database XE.

Table 6–1 Tablespaces and Descriptions

Tablespace	Description
SYSTEM	This tablespace is automatically created when Oracle Database XE is installed. It contains the data dictionary, which is the central set of tables and views used as a read-only reference for the database. It also contains various tables and views that contain administrative information about the database. These are all contained in the `SYS` schema, and can be accessed only by user `SYS` or other administrative users with the required privilege.
SYSAUX	This is an auxiliary tablespace to the `SYSTEM` tablespace, and is also automatically created upon installation. Some database components and products use this tablespace. The `HR` sample schema is also stored in the `SYSAUX` tablespace.
TEMP	This tablespace stores temporary data generated when processing SQL statements. For example, this tablespace is used for sort work space. The `TEMP` tablespace is specified as the default temporary tablespace for every user.
UNDO	This is the tablespace used by the database to store undo information.
USERS	This tablespace is used to store permanent user objects and data. In Oracle Database XE, `USERS` is the assigned default tablespace for all users except the `SYS` user, which has the default permanent tablespace of `SYSTEM`.

Note: You can create additional permanent tablespaces in Oracle Database XE, although typically there is no need to do so. One situation where you may have to create new permanent tablespaces is if you are importing objects from another Oracle database and the import file specifies tablespace names. See the `CREATE TABLESPACE` command in *Oracle Database SQL Language Reference*, and "Exporting and Importing Data" on page 10-13 for more information.

Datafiles and Tempfiles

Datafiles are the operating system files that hold database data. The data is written to these files in an Oracle-proprietary format that cannot be read by programs other than an Oracle database. **Tempfiles** are a special class of datafiles that are associated only with temporary tablespaces. Temporary tablespaces provide workspaces to help process queries.

Control File

The **control file** is a binary file that tracks the names and locations of the physical components of the database, and that maintains other control information, including records of all database backup-related files. It is essential to the functioning of the database.

Server Parameter File

The server parameter file (`SPFILE`) contains initialization parameters that Oracle Database XE uses at startup to determine the settings and run-time resources for the database. Do not attempt to edit this file with a text editor, as it is a binary file. You can change initialization parameter values by submitting `ALTER SYSTEM` commands with the SQL Command Line. See *Oracle Database SQL Language Reference* for more information.

You can view current initialization parameter settings with SQL Developer: in the Reports navigator, expand **Data Dictionary Reports** and then **Database Administration**, and see the reports under **Database Parameters**. Reports are available for All Parameters and Non-Default Parameters (the latter identified as "Modified" in the **Parameters** display available from the XE Database Home Page), and each report indicates whether each parameter is Session Modifiable and System Modifiable.

Password File

Oracle Database XE uses a password file to authenticate a user who is logging in remotely as user `SYS`. The `SYS` user can then perform administrative functions from a remote workstation. The password file contains the `SYS` password (encrypted). Whenever you change the password for `SYS`, the password file is automatically updated.

The password file is automatically created when you install Oracle Database XE. Unlike the other physical structures of the database, the password file is not backed up to the flash recovery area.

Note: Under typical circumstances, you should never log in to Oracle Database XE as user `SYS`.

Recovery-Related Structures in the Flash Recovery Area

The **flash recovery area** is essential for data protection in Oracle Database XE. It is a directory, separate from the database itself, where recovery-related structures are stored. These recovery-related structures include:

- Backups of the physical files that make up the database (datafiles, the control file, and the server parameter file (`SPFILE`))

  Note: Oracle database backup and recovery is based on protecting the physical files of the database, rather than individual database objects such as tables.

  Backups are stored in collections called **backup sets**. A backup set consists of one or more **backup pieces**, which are files in a proprietary archival format that can be accessed only by an Oracle database. See "About Backing Up and Restoring the Database" on page 11-1 for more information.

- Online redo logs

  The online redo log files record all changes made to the database. They can be used to reconstruct data in the event of a failure. See "Online Redo Log Files" on page 6-6 for more information.

- Archived redo logs

  If you enable log archiving, filled redo log files are *archived* (copied) in the flash recovery area before being reused. The online and archived logs together constitute a record of all changes committed to the database since the last backup was taken. See "Archived Redo Log Files" on page 6-7 for more information.

The database automatically manages all contents of the flash recovery area. You must not directly manipulate files within the flash recovery area using operating system commands.

Table 6–2 lists the default location for the flash recovery area on each platform.

Table 6–2 Flash Recovery Area Default Locations

Platform	Location
Linux	`/usr/lib/oracle/xe/app/oracle/flash_recovery_area/`
Windows	`c:\oraclexe\app\oracle\flash_recovery_area\`

Caution: The default configuration of Oracle Database XE stores the flash recovery area on the same disk as your database files. In this configuration, if a media (disk) failure occurs, you can lose both your database and your backups. For any database where data protection is essential, change the location of the flash recovery area so that it is stored on a different disk. See "Setting Flash Recovery Area Location and Size" on page 6-14 for instructions.

Online Redo Log Files

The most crucial structure for database recovery is a set of redo log files. This set of files is collectively known as the **redo log** for the database. A redo log is made up of redo entries, which are also called redo records.

The primary function of the redo log is to record all changes made to data in the database. If an Oracle instance failure or operating system failure prevents modified data from being permanently written to the datafiles, the changes can be recovered from the redo log so that committed data updates are not lost.

The database writes to the redo log files in a circular fashion. When the current redo log file fills, the database begins writing to the next available redo log file. (The redo log files that are not current are called **inactive**.) When the last available redo log file is filled, the database returns to the first redo log file and writes to it (overwriting previous redo entries), starting the cycle again.

Multiplexed Redo Log

To protect against a failure involving the redo log itself, Oracle Database XE allows a **multiplexed** redo log, meaning that two or more identical copies of the redo log can be automatically maintained in separate locations. For the most benefit, these locations should be on separate disks. Even if all copies of the redo log are on the same disk, however, the redundancy can help protect against I/O errors, file corruption, and so on.

Multiplexing is implemented by creating *groups* of redo log files. A **group** consists of a redo log file and its multiplexed copies. Each identical copy is said to be a **member** of the group. When the database writes to the current log file, all members in that log file's group are updated so that they remain identical. Each redo log group is defined by a number, such as group 1, group 2, and so on.

The current and inactive redo log files—that is, the current and inactive log groups and all their members—taken together, are called the **online redo log files**, to distinguish them from *archived* redo log files, which are described later in this section.

The default installation of Oracle Database XE configures two redo log groups of one member each. Thus, the default configuration for the redo logs does not use multiplexing. As shown in Figure 6–1 on page 6-2, both single-member redo log groups are stored in the flash recovery area.

You may want to multiplex the redo logs to protect against failures. Again, the ideal configuration is to separate members of the same log group onto different disks to protect against disk failure. Assuming that you decided to configure two members per group, the best practice for Oracle Database XE would be the following:

1. Move the flash recovery area to a different disk.

   See "Setting Flash Recovery Area Location and Size" on page 6-14 for instructions.

2. Create the second member of each redo log group in the same location as the datafiles.

   See "Creating Redo Log Members" in *Oracle Database Administrator's Guide* for instructions. Table 6–3 shows the location of the datafiles on each platform.

Table 6–3 Datafile Locations

Platform	Datafile Location
Linux	`/usr/lib/oracle/xe/oradata/XE/`
Windows	`C:\oraclexe\oradata\XE\`

Moving the flash recovery area to a different disk is preferred over leaving the flash recovery where it is and creating the second group member on a different disk. This is because the flash recovery area also contains database backups, and backups are best placed on a disk other than the disk that contains the datafiles.

Note: When you multiplex the redo log, the database must increase the amount of I/O that it performs. Depending on your configuration, this may affect overall database performance.

See Also:

- "Viewing Redo Log Files" on page 6-11 for instructions for viewing information on Oracle Database XE redo log groups, including the location of each log group member.
- *Oracle Database Administrator's Guide* for more details about the redo log.

Archived Redo Log Files

Oracle Database XE can be configured so that a background *archiving process* makes copies of filled, inactive redo log files in the flash recovery area before they are reused. Redo log files copied in this way are called **archived redo log files**.

Note: When the redo log is multiplexed, the database selects one member of that log file's group to archive. If a member is damaged or unavailable, the database attempts to archive another member.

A database configured to archive redo logs is said to be in **`ARCHIVELOG`** mode. (A database not configured to archive redo logs is said to be in **`NOARCHIVELOG`** mode.)

The advantages of running in `ARCHIVELOG` mode are the following:

- After a media failure causing the loss of some or all database files, the database can be reconstructed with all committed transactions intact if you have backups of

the control file and datafiles, and a complete set of all archived and online redo log files created since the last backup.

The online and archived redo log files contain a complete record of all database changes since the last backup. This reconstruction process is called **media recovery**.

- A database in ARCHIVELOG mode can be backed up while it is online.

  A NOARCHIVELOG mode database can be backed up only while it is in the mounted (but not open) state after a successful SHUTDOWN or SHUTDOWN IMMEDIATE operation. Your applications are unavailable during the backup of a NOARCHIVELOG database.

In ARCHIVELOG mode, the archived redo log files require disk space in the flash recovery area, and the flash recovery area requires monitoring to ensure that it does not fill completely.

Log archiving is disabled by default, to simplify the management of your database. Thus, the default configuration of Oracle Database XE protects your database from instance failure or operating system failure, but does not protect your database from media failure. Oracle therefore recommends that you do the following for complete data protection:

- Enable ARCHIVELOG mode.
- Back up the database frequently.

Note: If you enable ARCHIVELOG mode, you must perform regular backups of the database to avoid completely filling the flash recovery area. A completely filled flash recovery area can lead to database failure.

See Also:

- "Enabling ARCHIVELOG Mode for Media Failure Protection" on page 11-2
- "Monitoring Space in the Flash Recovery Area" on page 6-13
- "Viewing Redo Log Files" on page 6-11 for instructions for viewing information on Oracle Database XE redo log groups, including the location of each log group member.
- "Backing Up and Restoring the Database" on page 11-1 for details on database backup and recovery
- *Oracle Database Administrator's Guide* for more details about redo logs and log archiving

Monitoring Storage Space Usage

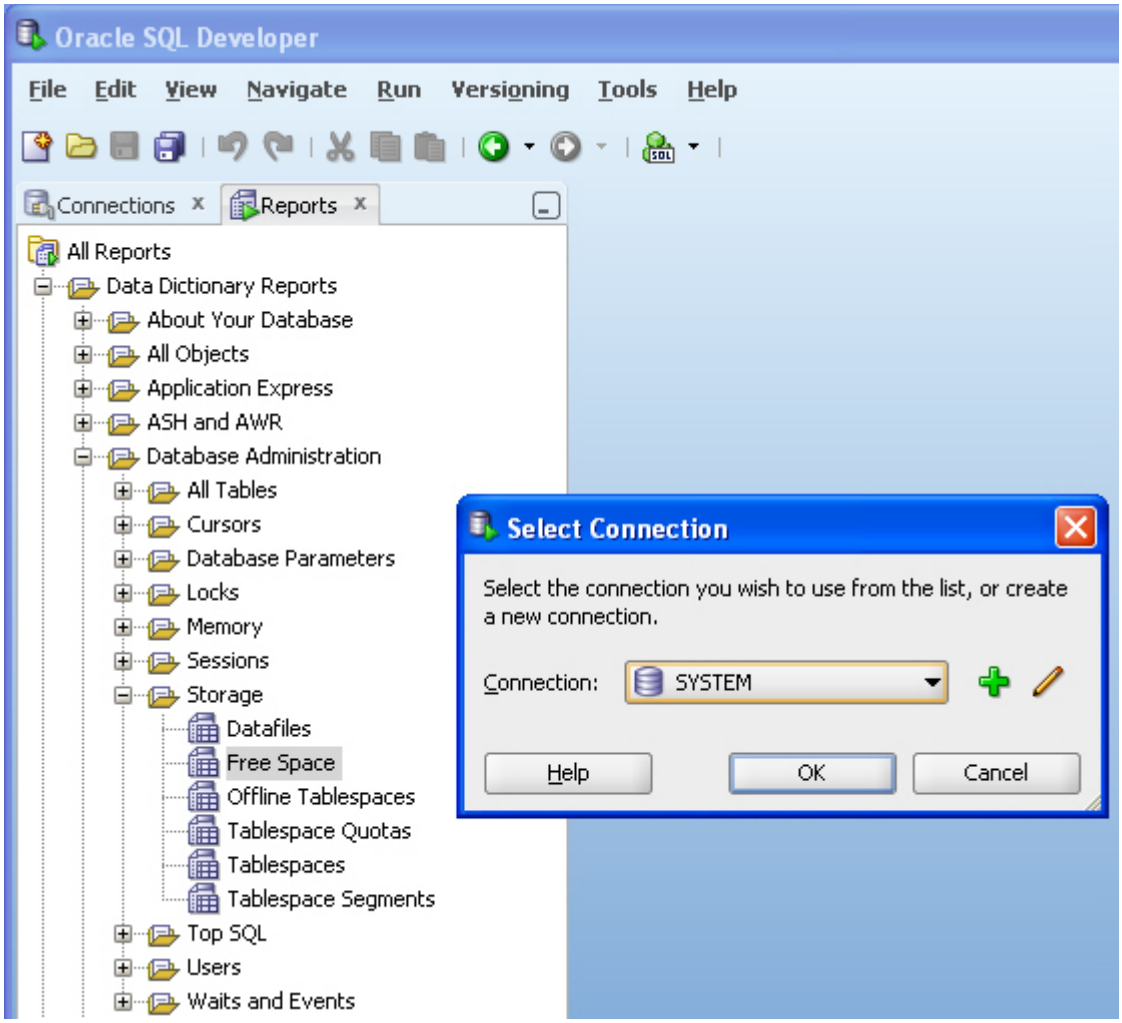
Because Oracle Database Express Edition (Oracle Database XE) is limited to just over four gigabytes (GB) of user data, your most important storage management task is monitoring the amount of free storage space and storage space used for any tablespaces used for storing user data.

To check this usage information:

1. In SQL Developer, click the Reports navigator tab, and expand the hierarchy as follows: **All Reports**, then **Data Dictionary Reports**, then **Database Administration**, then **Storage**.

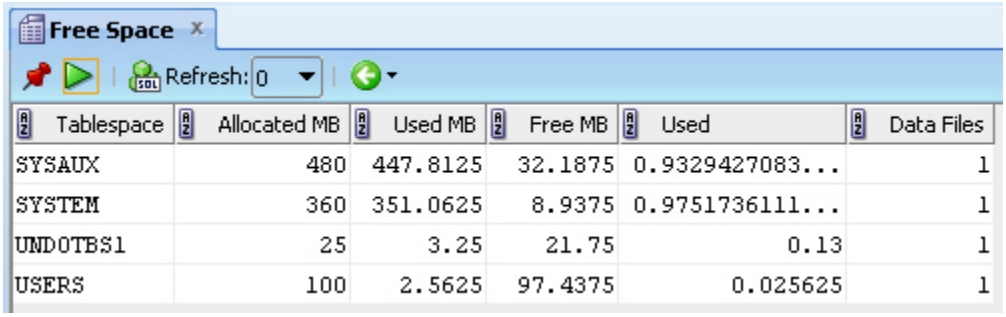
2. Under Storage, click **Free Space**, then select a database connection for a privileged user such as `SYSTEM`, as shown in Figure 6–2.

Figure 6–2 Selecting a Connection for the Free Space Report



3. Click **OK** in the Select Connection dialog box to display the Free Space report, which is shown in Figure 6–3.

Figure 6–3 Free Space Report



Tablespace	Allocated MB	Used MB	Free MB	Used	Data Files
SYSAUX	480	447.8125	32.1875	0.9329427083...	1
SYSTEM	360	351.0625	8.9375	0.9751736111...	1
UNDOTBS1	25	3.25	21.75	0.13	1
USERS	100	2.5625	97.4375	0.025625	1

You are interested in the USERS tablespace because it is typically used only for user data, not for Oracle-maintained system data. In Figure 6–3, only about 2.6 percent (0.025625) of the available allocated space for the USERS tablespace is actually being used.

If you notice that space remaining is becoming low, you can attempt to free some space by doing the following:

1. For each schema:
   a. Log in as the schema owner.
   b. Drop (delete) unused database objects.
   c. Purge the recycle bin.

      See "Purging the Recycle Bin" on page 11-10 for instructions.
2. Compact storage.

   See "Compacting Storage" on page 6-10 for instructions.

Note: If you log in to the database and connect as SYSDBA, you can purge the entire recycle bin (all schemas simultaneously). See *Oracle Database Administrator's Guide* for details.

If this procedure does not free a significant amount of space and you expect space requirements to continue to grow, you must consider upgrading to Oracle Database Standard Edition or Enterprise Edition.

See Also:

- "About the Database Storage Structures" on page 6-1
- "Logging In and Connecting to the Database as SYSDBA" on page 7-6

Compacting Storage

If you notice that space remaining in the database is becoming low, you can compact storage. Compacting storage attempts to recover unused fragmented free space in the database. Depending on the state of the database, compacting storage may or may not recover unused space.

Compacting and other options for reclaiming space are explained in the chapter about reclaiming waster space in *Oracle Database Administrator's Guide*.

See Also:

- "Monitoring Storage Space Usage" on page 6-8
- "About the Database Storage Structures" on page 6-1
- "About Administrative Accounts and Privileges" on page 7-3

Viewing Tablespaces

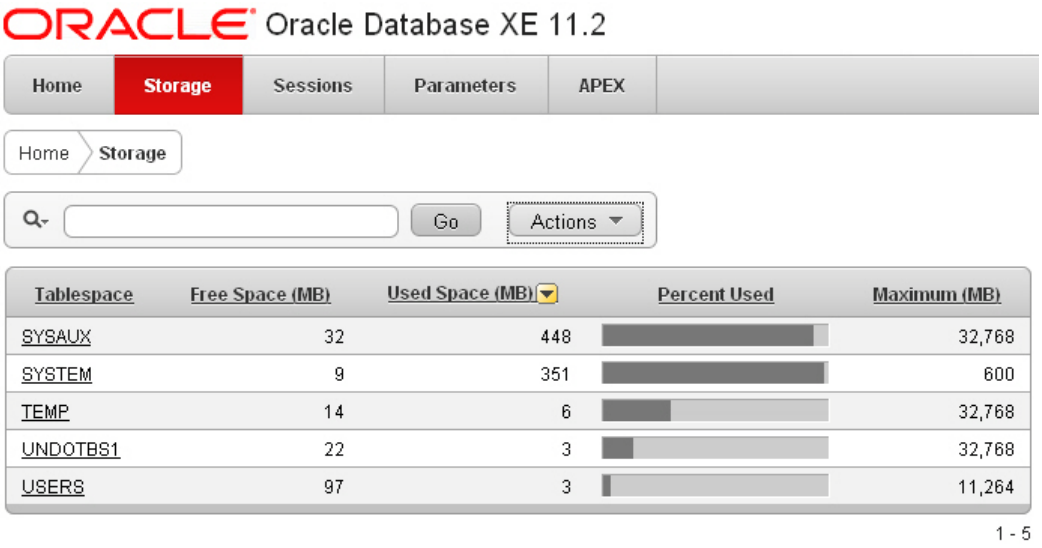
You can use the Oracle Database XE graphical user interface to view a list of tablespaces in the database, view tablespace properties, and view datafile properties.

To view Oracle Database XE tablespaces:

1. From the **Oracle Database 11g Express Edition** menu, select **Get Started**.
2. Click Storage.

   If prompted for administrator credentials, enter the `SYSTEM` user name and password or another administrator user name and password, and then click **Login**.

   The Storage page appears.



   Depending on datafile settings, a tablespace can grow beyond its currently allocated size. For example, the `USERS` tablespace may start with one datafile with an initially allocated size of 100 megabytes (MB), but the datafile can autoextend as needed, 10 MB at a time.

3. (Optional) Click a tablespace name to view information on that tablespace's segments.

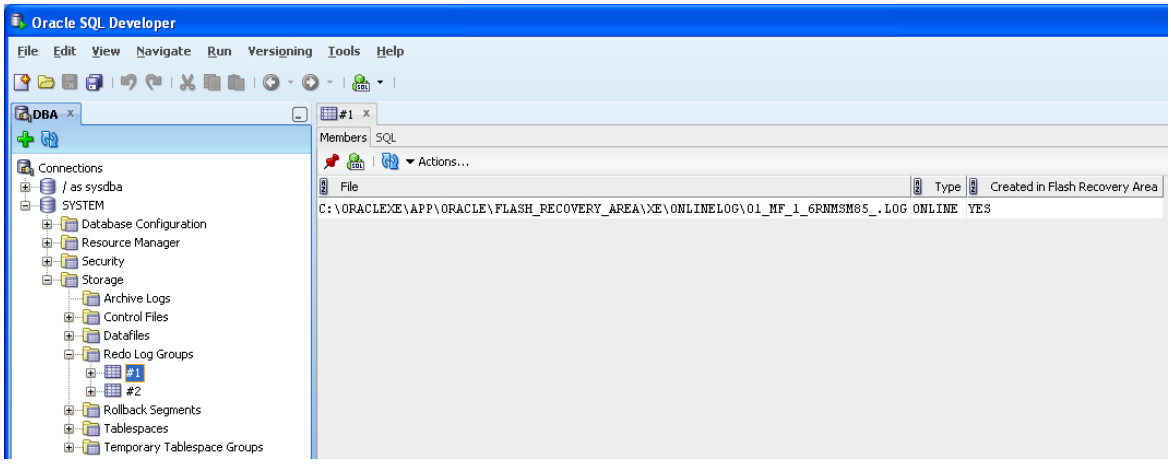
See Also:

- "About the Database Storage Structures" on page 6-1
- "About Administrative Accounts and Privileges" on page 7-3

Viewing Redo Log Files

You can use the DBA navigator in SQL Developer to view location and status information for the online redo log files.

1. In SQL Developer, if the DBA navigator is not visible, click **View**, then **DBA**.
2. If the DBA navigator does not already include a connection to the `SYSTEM` user, right-click Connections in the DBA navigator, select **Add Connection**, and select the connection for `SYSTEM`. (This does not create a new database connection; it just makes the `SYSTEM` connection visible in the DBA navigator.)
3. Expand the `SYSTEM` connection in the DBA navigator; then expand **Storage** and select **Redo Log Groups**.



4. You can select any redo log group to see, for each redo log file in the group, its location, type, and whether it was created in the flash recovery area.

See "Online Redo Log Files" on page 6-6 for more information on redo log groups and log group members.

Note: The display does not include information on archived redo log files. In addition, the default configuration of Oracle Database XE does not include redo log multiplexing.

See Also: "Recovery-Related Structures in the Flash Recovery Area" on page 6-5

Managing the Flash Recovery Area

Oracle Database Express Edition (Oracle Database XE) stores database backups, online redo log files, and archived redo log files in the flash recovery area. The primary management tasks related to the flash recovery area are the following:

- Monitoring flash recovery area available space
- Changing the flash recovery area location and size

The flash recovery area is a fixed-size storage area. The default size is 10 gigabytes (GB). Because Oracle Database XE storage is limited to 5 GB, two backups of the database are expected to fit in the flash recovery area. However, when running in `ARCHIVELOG` mode, you may need to allocate extra space for archived redo log files, and you must monitor flash recovery area available space more closely. (See "Archived Redo Log Files" on page 6-7 for information about `ARCHIVELOG` mode.)

This section contains the following topics:

See Also:

- "Recovery-Related Structures in the Flash Recovery Area" on page 6-5
- "About Backing Up and Restoring the Database" on page 11-1

Monitoring Space in the Flash Recovery Area

You must run a SQL query to determine the current available space in the flash recovery area.

To view current available space in the flash recovery area:

1. Using the SQL Command Line, log in and connect to the database as `SYSTEM` or `SYSDBA`.
2. Enter the following query:

```
SELECT
    NAME,
    TO_CHAR(SPACE_LIMIT, '999,999,999,999') AS SPACE_LIMIT,
    TO_CHAR(SPACE_LIMIT - SPACE_USED + SPACE_RECLAIMABLE, '999,999,999,999')
       AS SPACE_AVAILABLE,
    ROUND((SPACE_USED - SPACE_RECLAIMABLE)/SPACE_LIMIT * 100, 1)
       AS PERCENT_FULL
    FROM V$RECOVERY_FILE_DEST;
```

The query results should look something like this:

```
------------------------------------------------------------------------------------------
NAME                                    SPACE_LIMIT      SPACE_AVAILABLE   PERCENT_FULL
C:\oraclexe\app\oracle\flash_recovery_area  21,474,836,480   20,050,844,672    6.6
------------------------------------------------------------------------------------------
```

Interpret the results as follows:

- `NAME` indicates the current flash recovery area location.
- `SPACE_LIMIT` indicates the current flash recovery area maximum size.
- `SPACE_AVAILABLE` indicates the space available for storing new backups and archived redo logs, including space that can be reclaimed by deleting files that are no longer needed to meet the retention policy.
- `PERCENT_FULL` indicates the current percentage of flash recovery area space used for backups and archived redo logs that are within the backup retention policy.

If the `PERCENT_FULL` value is approaching 100% (for example, is 85% or more), and log archiving is enabled (the database is in `ARCHIVELOG` mode), it may be time to back up the database. Backing up the database deletes archived log files and frees space in the flash recovery area.

If the `PERCENT_FULL` value is frequently close to 100% after several recent backups, consider allocating more space for your flash recovery area as described in "Setting Flash Recovery Area Location and Size" on page 6-14, or, if in `ARCHIVELOG` mode, taking backups more frequently to reduce the size of the retained archived log files.

See Also:

- *Oracle Database Reference* for details on the `V$RECOVERY_FILE_DEST` view

Setting Flash Recovery Area Location and Size

This section explains the procedures for setting the flash recovery area location and for changing the flash recovery area size. The flash recovery area location and size are specified by the initialization parameters DB_RECOVERY_FILE_DEST and DB_RECOVERY_FILE_DEST_SIZE.

Setting the Flash Recovery Area Location

Table 6–2 on page 6-6 shows the default flash recovery area locations on both platforms.

To change the flash recovery area location:

1. Using the SQL Command Line, log in and connect to the database as SYSDBA.

   See "Logging In and Connecting to the Database as SYSDBA" on page 7-6 for instructions.

2. At the SQL Command Line prompt, enter the following command:

   ```
   ALTER SYSTEM SET DB_RECOVERY_FILE_DEST = 'new_path';
   ```

   where *new_path* is an absolute path to the new directory for the flash recovery area. The path must exist. (The ALTER SYSTEM command cannot create directories.)

   For example, in Windows, to set the flash recovery location to the FRA directory on the E: drive, enter the following command:

   ```
   ALTER SYSTEM SET DB_RECOVERY_FILE_DEST = 'E:\FRA';
   ```

3. Enter the following command, which runs a PL/SQL script that moves the online redo log files to the new flash recovery area location and drops the log files from the old location:

   ```
   @?/sqlplus/admin/movelogs
   ```

   Note that the command must be entered in lower case. The '@' symbol is an abbreviation for the START command, which runs the named SQL script. The '?' symbol, when used in a SQL Command Line command, is an abbreviation for the Oracle home directory. This command therefore runs the script named movelogs.sql, which is located in the path *Oracle_home*/sqlplus/admin.

   If the script is successful, SQL Command Line displays the following message:

   ```
   PL/SQL procedure successfully completed.
   ```

   A listing of the movelogs.sql script appears later in this section.

4. Enter the following command to exit the SQL Command Line:

   ```
   EXIT
   ```

Note: After you change the location of the flash recovery area, Recovery Manager (RMAN) can still use the backups and archived logs in the old location until they become obsolete. The old backups cannot be moved to the new flash recovery area location.

Do not manually delete the contents of the old flash recovery area using operating system utilities. Instead, make the backups in the old flash recovery area location obsolete by backing up your database twice after you change the location of the flash recovery area.

Each time that you back up the database as described in "Backing Up the Database" on page 11-4, obsolete backups and archived logs are deleted so that only the two most recent backups and accompanying archived redo logs are retained. Thus, after the new flash recovery area contains two recent backups, all files are deleted from the old location.

See Also:

- *Oracle Database Administrator's Guide* for details on setting and changing database initialization parameters
- "Viewing Redo Log Files" on page 6-11 for information on how to view the online redo log files in their new location.
- "Recovery-Related Structures in the Flash Recovery Area" on page 6-5

Changing the Flash Recovery Area Size

To change the flash recovery area size:

1. Using the SQL Command Line, log in and connect to the database as `SYSDBA`.

   See "Logging In and Connecting to the Database as SYSDBA" on page 7-6 for instructions.

2. Enter the following command at the SQL Command Line prompt:

   ```
   ALTER SYSTEM SET DB_RECOVERY_FILE_DEST_SIZE = new_size;
   ```

   where *new_size* can be of the format *n*`K` (kilobytes), *n*`M` (megabytes) or *n*`G` (gigabytes).

   For example, to set the flash recovery area size to 20 gigabytes, enter the following command:

   ```
   ALTER SYSTEM SET DB_RECOVERY_FILE_DEST_SIZE = 20G;
   ```

3. Enter the following command to exit the SQL Command Line:

   ```
   EXIT
   ```

See Also:

- *Oracle Database Administrator's Guide* for details on setting and changing database initialization parameters
- "Recovery-Related Structures in the Flash Recovery Area" on page 6-5

movelogs.sql Script

The following is a listing of the movelogs.sql script, which you must run after changing the location of the flash recovery area. The script moves the online redo log files to the new flash recovery area location and drops the log files from the old location.

```
declare
   cursor rlc is
      select group# grp, thread# thr, bytes/1024 bytes_k
        from v$log
      order by 1;
   stmt     varchar2(2048);
   swtstmt  varchar2(1024) := 'alter system switch logfile';
   ckpstmt  varchar2(1024) := 'alter system checkpoint global';
begin
   for rlcRec in rlc loop
 stmt := 'alter database add logfile thread ' ||
               rlcRec.thr || ' size ' ||
               rlcRec.bytes_k || 'K';
      execute immediate stmt;
      begin
         stmt := 'alter database drop logfile group ' || rlcRec.grp;
         execute immediate stmt;
      exception
         when others then
            execute immediate swtstmt;
            execute immediate ckpstmt;
            execute immediate stmt;
      end;
      execute immediate swtstmt;
   end loop;
end;
/
```

7

Managing Users and Security

Users access Oracle Database Express Edition through database user accounts. Some of these accounts are automatically created administrative accounts—accounts with database administration privileges. You log in to these administrative accounts to create and manage other user accounts, maintain database security, and perform other database administration tasks.

This section contains the following topics:

About User Accounts

A user account is identified by a user name and defines the user's attributes, including the following:

- Password for database authentication
- Privileges and roles
- Default tablespace for database objects
- Default temporary tablespace for query processing work space

When you create a user, you are also implicitly creating a schema for that user. A **schema** is a logical container for the database objects (such as tables, views, triggers, and so on) that the user creates. The schema name is the same as the user name, and can be used to unambiguously refer to objects owned by the user. For example, `HR.EMPLOYEES` refers to the table named `EMPLOYEES` in the `HR` schema. (The `EMPLOYEES` table is owned by `HR`.) The terms *database object* and *schema object* are used interchangeably.

When you drop (delete) a user, you must either first drop all the user's schema objects, or use the **cascade** feature of the drop operation, which simultaneously drops a user and all of that user's schema objects.

This section contains these topics:

- Internal User Accounts on page 7-2

See Also: "Predefined User Accounts" on page 7-12

User Privileges and Roles

When creating a user, you grant privileges to enable the user to connect to the database, to run queries and make updates, and to create schema objects. There are two main types of user privileges:

- **System privileges**—A system privilege is the right to perform a particular action, or to perform an action on any schema objects of a particular type. For example, the privileges to create tables and to delete the rows of any table in a database are system privileges.
- **Object privileges**—An object privilege is a right to perform a particular action on a specific schema object. Different object privileges are available for different types of schema objects. The privilege to delete rows from the `DEPARTMENTS` table is an example of an object privilege.

Managing and controlling privileges is made easier by using **roles**, which are named groups of related privileges. You create roles, grant system and object privileges to the roles, and then grant roles to users. Unlike schema objects, roles are not contained in any schema.

Oracle Database Express Edition comes with some predefined roles:

- The `DBA` role enables a user to perform most administrative functions, including creating users and granting privileges; creating and granting roles; creating and dropping schema objects in other users' schemas; and more. It grants all system privileges, but does not include the privileges to start up or shut down the database. It is by default granted to user SYSTEM. You should be very cautious about assigning the DBA role to any other database users.
- Use of the `CONNECT` and `RESOURCE` roles is discouraged. Instead, grant only those privileges that the specific user will need. For example:

```
grant CREATE SESSION, ALTER SESSION, CREATE DATABASE LINK, -
  CREATE MATERIALIZED VIEW, CREATE PROCEDURE, CREATE PUBLIC SYNONYM, -
  CREATE ROLE, CREATE SEQUENCE, CREATE SYNONYM, CREATE TABLE, -
  CREATE TRIGGER, CREATE TYPE, CREATE VIEW, UNLIMITED TABLESPACE -
  to chris;
```

See Also:

- *Oracle Database Security Guide* for more information on privileges and roles
- *Oracle Database SQL Language Reference* for tables of system privileges, object privileges, and predefined roles.
- "Creating Users" on page 7-7
- "Predefined User Accounts" on page 7-12
- "About Administrative Accounts and Privileges" on page 7-3

Internal User Accounts

Certain user accounts are created automatically for database administration. Examples are `SYS` and `SYSTEM`. Other accounts are automatically created just so that individual Oracle Database XE features or products can have their own schemas. An example is

the CTXSYS account, which is used by the Oracle Text product. Oracle Text is used to index the Oracle Database XE online Help. The Help index is stored in the CTXSYS schema in the database.

These automatically created accounts are called **internal user accounts**, and their schemas are called **internal schemas**.

The only internal accounts that you may log in with are the SYS and SYSTEM accounts, although it is recommended that you avoid logging in with the SYS account. Do not attempt to log in with other internal accounts. See "The SYS and SYSTEM Users" on page 7-3 for more information.

About Administrative Accounts and Privileges

Administrative accounts and privileges enable you to perform administrative functions like managing users, managing database memory, and starting up and shutting down the database.

This section contains the following topics:

See Also:

- About User Accounts on page 7-1
- Logging In as an Administrator on page 7-5

The SYS and SYSTEM Users

The following administrative user accounts are automatically created when you install Oracle Database Express Edition (Oracle Database XE). They are both created with the password that you supplied upon installation (Windows operating systems) or configuration (Linux operating systems).

- SYSTEM

  This is the user account that you log in with to perform all administrative functions other than starting up and shutting down the database.

- SYS

  All base tables and views for the database data dictionary are stored in the SYS schema. These base tables and views are critical for the operation of Oracle Database XE. To maintain the integrity of the data dictionary, tables in the SYS schema are manipulated only by the database. They should never be modified by any user or database administrator. You must not create any tables in the SYS schema.

  There is typically no reason to log in as user SYS. User SYSTEM is preferred for all administrative tasks except starting up and shutting down. See "Starting Up and Shutting Down" on page 2-1 for more information.

See Also:

- "Logging In as an Administrator" on page 7-5
- "Changing Administrative User Passwords" on page 7-6

The SYSDBA System Privilege

SYSDBA is a system privilege that is assigned only to user SYS. It enables SYS to perform high-level administrative tasks such as starting up and shutting down the database.

Although under typical circumstances it is not necessary to log in to the database as user SYS, if you want to log in as SYS with the SQL Command Line (SQL*Plus), you must connect to the database "AS SYSDBA." Connecting AS SYSDBA invokes the SYSDBA privilege. If you omit the AS SYSDBA clause when logging in as user SYS, the SQL Command Line rejects the login attempt.

The following example illustrates how to connect to the database with the SYSDBA privilege from the SQL Command Line:

```
SQL > connect sys/password as sysdba
```

password is the password for the SYS user account.

Caution: When you connect as user SYS, you have unlimited privileges on data dictionary tables. Be certain that you do not modify any data dictionary tables.

See Also:

- "Changing Administrative User Passwords" on page 7-6
- Chapter 2, "Starting Up and Shutting Down" on page 2-1

Operating System Authentication

Operating system authentication (OS authentication) is a way of using operating system login credentials to authenticate database users. One aspect of OS authentication can be used to authenticate database administrators. If you log in to the Oracle Database XE host computer with a user name that is in a special operating system user group, you are then permitted to connect to the database with the SYSDBA privilege. An administrator who is authenticated through OS authentication does not need to know the SYS or SYSTEM account password.

OS authentication is needed because there must be a way to identify administrative users even if the database is shut down. A user authenticated in this way can then start up the database. (See "Starting Up and Shutting Down" on page 2-1 for more information.)

Table 7–1 lists the operating system user groups whose member users can connect to the database with the SYSDBA privilege.

Table 7–1 Operating System User Groups for OS Authentication

Platform	Operating System User Group Name
Linux	dba
Windows	ORA_DBA

On each platform, if the OS authentication user group does not already exist, it is automatically created when you install Oracle Database XE. In addition, upon installation on the Linux platform, the user account oracle is automatically created and placed in the dba group. Upon installation on the Windows platform, the user performing the installation is automatically added to the ORA_DBA group. On both

platforms, you can add other host users to the OS authentication user group to enable them to connect to the database with the SYSDBA privilege.

Caution: Adding other users to the OS authentication user group has security implications, because these users can modify any database object.

See Also:

- "The SYSDBA System Privilege" on page 7-4
- "Logging In and Connecting to the Database as SYSDBA" on page 7-6

Logging In as an Administrator

There are three ways to log in to Oracle Database Express Edition (Oracle Database XE) to perform administrative tasks:

- Log in as user SYSTEM
- Log in as a user who has been granted the DBA role
- Log in and connect to the database as SYSDBA

Table 7–2 provides information about each of these login methods.

Table 7–2 Database Administrator Login Methods

Login Method	Permitted In	Notes	See
Log in to the database as user SYSTEM	The Oracle Database XE graphical user interface and the SQL Command Line	For routine administrative tasks like managing memory and managing users. You must supply the password for the SYSTEM user.	"Logging In as User SYSTEM" on page 7-5
Log in to the database as a user who has been granted the DBA role	The Oracle Database XE graphical user interface and the SQL Command Line	For routine administrative tasks like managing users. An administrator must first grant the DBA role to the user.	"Logging In as a User with the DBA Role" on page 7-6
Log in and connect to the database as SYSDBA	the SQL Command Line	For high-level administrative tasks like starting up and shutting down the database, and changing the SYS password. You can connect as SYSDBA using the SYS user name and password, or using operating system authentication.	"Logging In and Connecting to the Database as SYSDBA" on page 7-6

See Also:

- "About Administrative Accounts and Privileges" on page 7-3
- "Changing Administrative User Passwords" on page 7-6

Logging In as User SYSTEM

You can log in as user SYSTEM using either of the following methods:

- Using SQL Developer, open a database connection to the SYSTEM user.
- Using the SQL Command Line, enter the following statement:

```
SQL> CONNECT SYSTEM/<password>;
```

Logging In as a User with the DBA Role

The procedures for logging in as a user who has been granted the DBA role are the same as those for logging in as user SYSTEM, with the following exceptions:

- When logging in, you must supply the user name and password for this user account.
- An administrator must have previously logged in and granted the DBA role to this user.

  See "User Privileges and Roles" on page 7-2 for more information.

Logging In and Connecting to the Database as SYSDBA

You can log in and connect as SYSDBA using either of the following methods:

- Using SQL Developer, open a database connection to the SYS user AS SYSDBA.
- Using the SQL Command Line, enter one the following statements.

  To use database authentication:

  ```
  SQL> CONNECT SYS/<password> AS SYSDBA;
  ```

  To use operating system (OS) authentication:

  ```
  SQL> CONNECT / AS SYSDBA;
  ```

  The slash (/) indicates that the database should authenticate you with operating system (OS) authentication. Remember that when you connect with OS authentication, you are effectively logging in to the database as user SYS.

  See Also:

  - "Logging In as an Administrator" on page 7-5
  - "The SYSDBA System Privilege" on page 7-4
  - "Operating System Authentication" on page 7-4

Changing Administrative User Passwords

To change the password for user SYS or SYSTEM:

1. Using the SQL Command Line, connect to the database as SYSDBA.

   See "Logging In and Connecting to the Database as SYSDBA" on page 7-6 for instructions.

2. Enter one of the following commands:

   ```
   ALTER USER SYS IDENTIFIED BY newpassword;
   ALTER USER SYSTEM IDENTIFIED BY newpassword;
   ```

   where *newpassword* is the desired new password.

   See Also: "About Administrative Accounts and Privileges" on page 7-3

Managing Database Users

You can use SQL Developer or the SQL Command Line (SQL*Plus) to manage database users. This section discusses using SQL Developer, and contains the following topics:

To perform these operations, in the SQL Developer Connections navigator, open a connection

1. In the SQL Developer Connections navigator, open a connection to the `SYSTEM` user.
2. In the nodes under this `SYSTEM` connection, expand **Other Users**.

   This displays nodes for all database users, including several Oracle-supplied internal users. The Connections navigator hierarchy may look like this:

```
Connections
  . . .
  SYSTEM
    Views
    Editioning Views
    . . .
    Other Users
      ANONYMOUS
      APEX_040000
      APEX_PUBLIC_USER
      APQQOSSYS
      CHRIS
      CTXSYS
      . . .
      HR
      MDSYS
```

3. To create a new database user, right-click the **Other Users** node in the Connections navigator and select **Create User**.
4. To perform an action on a database user, right-click that user in the hierarchy and select the appropriate command (**Edit User** or **Drop User**).

> **See Also:** *Oracle Database SQL Language Reference* and *Oracle Database Security Guide* for information on managing users with the SQL Command Line (SQL*Plus).

Creating Users

To create a new database user, right-click the **Other Users** node in the SQL Developer Connections navigator and select **Create User**. Before creating a user, determine the following:

- Whether or not you want to permit the user to create database objects in that user's own schema.

If so, on the Create Database User page, grant individual session-related and *create object* system privileges. See the following topics for more information:

- "User Privileges and Roles" on page 7-2 for details on privileges and roles
- "Configuring Privilege and Role Authorization" in *Oracle Database Security Guide* for more information on system privileges
- "Creating and Managing Schema Objects" in *Oracle Database Express Edition 2 Day Developer's Guide* for more information on database objects

- Whether or not you want to grant the user DBA privileges.

  If so, on the Create Database User page, grant the **DBA** role. See "User Privileges and Roles" on page 7-2 for details on the `DBA` role.

  Because DBA privileges include the ability to create database objects in any schema, if you grant the `DBA` role, you do not need to grant individual *create object* system privileges.

  Caution: Granting the `DBA` role to a user has security implications, because the user can modify objects in other users' schemas.

- Whether or not to create the user with an expired password.

  When you do this, the password that you assign the user is used only for the user's first login. Upon first login, the user is prompted to select a new password.

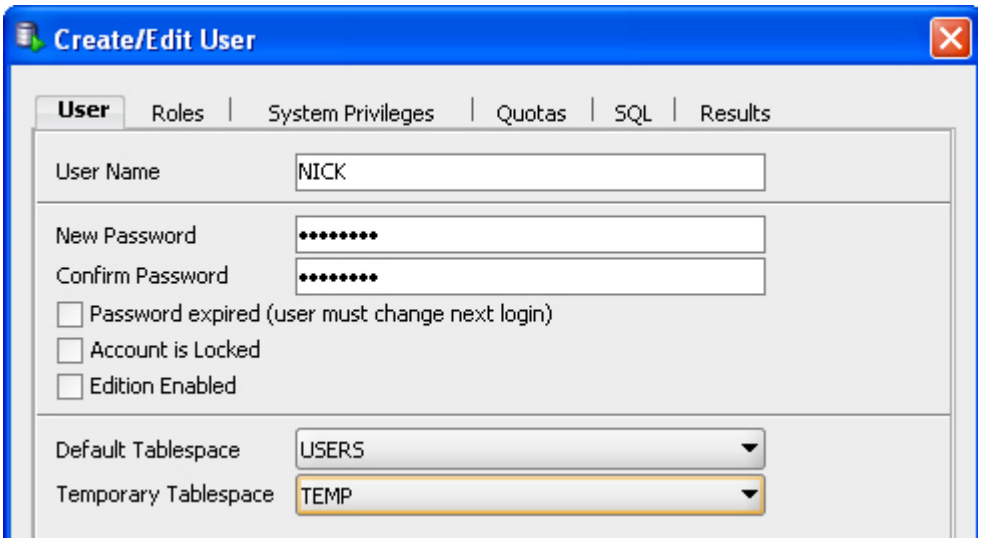
See Also: "About User Accounts" on page 7-1

Example: Creating a User

Suppose you want to create a user account for a database application developer named Nick. Because Nick is a developer, you want to grant him all `CREATE` system privileges so that he can create the schema objects that his applications require. In addition, you want to create his account with the password *firesign*.

To create the user `Nick`:

1. Right-click the **Other Users** node in the SQL Developer Connections navigator and select **Create User**.
2. In the Create/Edit User dialog box, for the **User** tab, enter the information shown in the following figure:



User Name: `NICK`

New Password and **Confirm Password**: Desired password for the user.

Password expired (user must change): Select or not, as desired. (It is not selected in the figure.)

Account is Locked: Select or not, as desired. (It is not selected in the figure.)

Edition Enabled: Select or not, as desired. (It is not selected in the figure.)

Default Tablespace: `USERS`

Temporary Tablespace: `TEMP`

3. In the Create/Edit User dialog box, click the **System Privileges** tab, and under Granted select the following privileges because you want to be sure that `NICK` will have them):
   - `ALTER SESSION`
   - `CREATE SESSION`
   - `CREATE DATABASE LINK`
   - `CREATE MATERIALIZED VIEW`
   - `CREATE PROCEDURE`
   - `CREATE PUBLIC SYNONYM`
   - `CREATE ROLE`
   - `CREATE SEQUENCE`
   - `CREATE SYNONYM`
   - `CREATE TABLE`
   - `CREATE TRIGGER`
   - `CREATE TYPE`
   - `CREATE VIEW`
   - `UNLIMITED TABLESPACE`
4. In the Create/Edit User dialog box, click **Apply**, then click **Close**.

Altering Users

You can use the Manage Database Users page to alter a user. Altering a user means changing some of that user's attributes. You can change all user attributes except the user name, default tablespace, and temporary tablespace. If you want to change the user name, you must drop the user and re-create that user with a different name. (Before you drop the user, ensure that the user's schema objects are either no longer needed or are backed up (for example, by exporting them). See "Dropping Users" on page 7-11 for more information.)

One of the attributes that you can alter is the user password. If you do this, you must either communicate the new password to the user, or request the new password from the user and then enter it. An easier and more secure way to cause a password change is to expire the password. When you **expire** a password, the user is prompted to change the password at the next login. See "Expiring a User Password" on page 7-11 for more information.

See Also:

- "Locking and Unlocking User Accounts" on page 7-10
- "Exporting and Importing Metadata and Data" on page 10-1 for information on how to export and import a schema.

Example: Altering a User

Suppose Nick is promoted to senior developer, and he has shown an interest in helping with routine database administration tasks. You decide to grant the `DBA` role to Nick.

To alter Nick's user account:

1. In the SQL Developer Connections navigator, expand the `SYSTEM` connection and right-click the **Other Users** node.
2. Right-click **NICK** and select **Edit User**.
3. In the Create/Edit User dialog box, click the **Roles** tab.
4. Under Granted, select **DBA**.
5. In the Create/Edit User dialog box, click **Apply**, then click **Close**.

Locking and Unlocking User Accounts

To temporarily deny access to the database for a particular user, you can lock the user account. If the user then attempts to connect, the database displays an error message and disallows the connection. You can unlock the user account when you want to allow database access again for that user.

Note: Many internal user accounts are locked (or both expired and locked). You should not attempt to log in with these locked user accounts. See "Internal User Accounts" on page 7-2 for more information.

The `HR` user account, which contains a sample schema, is initially expired and locked. You must log in as `SYSTEM`, unlock the account, and assign a password before you can log in as `HR`.

To lock or unlock a user account:

1. In the SQL Developer Connections navigator, expand the SYSTEM connection and right-click the **Other Users** node.
2. Right-click the desired user and select **Edit User**.
3. In the Create/Edit User dialog box, in the User tab, select or deselect **Account is Locked**: selecting (checking) causes the account to be locked; deselecting (unchecking) causes the account to be unlocked.
4. In the Create/Edit User dialog box, click **Apply**, then click **Close**.

Expiring a User Password

When you expire a user password, the user is prompted to change the password at the next login. Reasons to expire a password include the following:

- A user password becomes compromised.
- You have a security policy in place that requires users to change their passwords on a regular basis.
- A user has forgotten his or her password.

  In this case, you alter the user account, assign a new temporary password, and expire the password. The user then logs in with the temporary password and is prompted to choose a new password.

  See "Altering Users" on page 7-10 for more information.

Example: Expiring a Password

Suppose Nick's password becomes compromised, and you want to assign him a new one. The easiest way to do this is to expire his current password. The next time that Nick logs in with the compromised password, he is prompted to choose a new password.

To expire Nick's password:

1. In the SQL Developer Connections navigator, expand the SYSTEM connection and right-click the **Other Users** node.
2. Right-click **NICK** and select **Edit User**.
3. In the Create/Edit User dialog box, in the User tab, select (check) **Password expired (user must change next login)**.
4. In the Create/Edit User dialog box, click **Apply**, then click **Close**.

Dropping Users

Dropping a user removes the user from the database. Before you can drop a user, you must first drop all the user's schema objects. Or, you can use the **cascade** feature of the drop operation, which simultaneously drops a user and also that user's schema objects. The following are two alternatives to dropping a user and losing all the user's schema objects:

- To temporarily deny access to the database for a particular user while preserving the user's schema objects, you can lock the user account. See "Locking and Unlocking User Accounts" on page 7-10 for more information.
- To drop a user but retain the data from the user's tables, export the tables first. See "Exporting and Importing Metadata and Data" on page 10-1 for instructions.

Caution: Under no circumstances should you attempt to drop the SYS or SYSTEM users, or any other internal user accounts. Doing so could cause Oracle Database XE to malfunction.

Example: Dropping a User

Suppose Nick's project is canceled and Nick takes a position in another department. You want to drop the user NICK and all associated schema objects.

To drop user NICK and all his owned schema objects:

1. In the SQL Developer Connections navigator, expand the SYSTEM connection and right-click the **Other Users** node.
2. Right-click **NICK** and select **Drop User**.
3. In the Drop User dialog box, (check) **Cascade**

   This indicates that you want to drop the user's schema objects also. If the user has schema objects and you do not select this option, you receive an error message if you attempt to complete the drop operation.
4. In the Drop User dialog box, click **Apply**.

User Accounts Reference

This section provides reference information for managing user accounts. It covers the following topics:

Predefined User Accounts

Table 7–3 lists the Oracle Database XE predefined user accounts. Many of these accounts are internal accounts. You must not drop internal accounts, and with the exception of the accounts SYS and SYSTEM, you must not attempt to log in with an internal account.

Table 7–3 Oracle Database Express Edition Predefined User Accounts

User Account Name	Purpose
ANONYMOUS	Internal. Used for anonymous HTTP access to the database. Required by the Oracle Database XE graphical user interface. This account must remain unlocked. The account password is set upon installation (Windows) or configuration (Linux). For optimal security, avoid changing the password for this account.
CTXSYS	Internal.
DBSNMP	Internal.
DIP	Internal.
FLOWS_*version*	Internal.
FLOWS_FILES	Internal.
HR	For the HR sample schema. This account is initially expired and locked.
MDSYS	Internal.

Table 7–3 (Cont.) Oracle Database Express Edition Predefined User Accounts

User Account Name	Purpose
OUTLN	Internal.
SYS	Owns the data dictionary base tables and views. The account password is set upon installation (Windows) or configuration (Linux).
SYSTEM	Log in with this account to perform routine database administration. The account password is set upon installation (Windows) or configuration (Linux).
TSMSYS	Internal.
XDB	Internal.

See Also:

- "About User Accounts" on page 7-1
- "About Administrative Accounts and Privileges" on page 7-3

8

Monitoring the Database

As an administrator, you can monitor the activities of the database and its users. You can use this information for tuning, troubleshooting, and more.

This section contains the following topics:

Monitoring Sessions

You can use SQL Developer to monitor the current database sessions. This enables you to determine the users who are currently logged in to the database and what applications they are running.

You can also **kill** a session—to cause it to be disconnected and its resources to be relinquished.

This section contains the following topics:

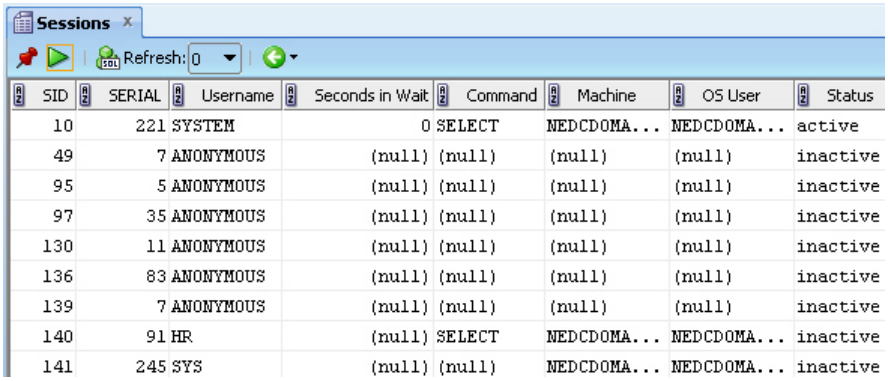
Viewing Sessions

To view sessions:

1. In SQL Developer, click **Tools**, then **Monitor Sessions**.
2. In the Select Connection dialog box, select a connection to `SYSTEM` (or another account with full DBA privileges)

   A Sessions tab is displayed. Figure 8–1 shows part of the display.

Figure 8–1 Monitor Sessions Page



SID	SERIAL	Username	Seconds in Wait	Command	Machine	OS User	Status
10	221	SYSTEM	0	SELECT	NEDCDOMA...	NEDCDOMA...	active
49	7	ANONYMOUS	(null)	(null)	(null)	(null)	inactive
95	5	ANONYMOUS	(null)	(null)	(null)	(null)	inactive
97	35	ANONYMOUS	(null)	(null)	(null)	(null)	inactive
130	11	ANONYMOUS	(null)	(null)	(null)	(null)	inactive
136	83	ANONYMOUS	(null)	(null)	(null)	(null)	inactive
139	7	ANONYMOUS	(null)	(null)	(null)	(null)	inactive
140	91	HR	(null)	SELECT	NEDCDOMA...	NEDCDOMA...	inactive
141	245	SYS	(null)	(null)	NEDCDOMA...	NEDCDOMA...	inactive

See the chapter about monitoring database operations in *Oracle Database Administrator's Guide* for more information.

3. (Optional) Right-click in any row in the display, and explore the options available as shown in the context menu commands, which include **Trace Session**, **Kill Session**, and **Find/Highlight** (to search for rows in the grid that contain a specified text string).

Killing (Terminating) a Session

You can use SQL Developer to kill (terminate) a database session. This logs off and disconnects the user running the session. If the user is processing a transaction when you kill the session, the transaction is rolled back.

Reasons to kill a session include the following:

- The session is not responding.
- You want to perform an administrative function that requires all users to log off first, but the user is not available to end his or her session.

To kill a session:

1. In SQL Developer, click **Tools**, then **Monitor Sessions**.
2. In the Select Connection dialog box, select a connection to `SYSTEM` (or another account with full DBA privileges)
3. Right-click in the row for the session to be terminated, and select **Kill Session**.

Monitoring the Top SQL Statements

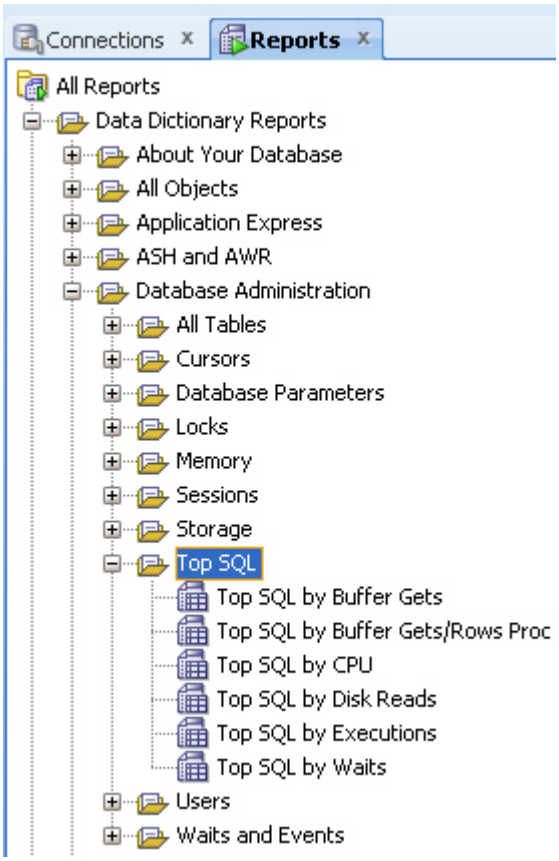
The "top" SQL statements represent the SQL statements that are executed most often, that use more system resources than other SQL statements, or that use system resources more frequently than other SQL statements. Viewing the top SQL statements reports that are available in SQL Developer enables you to focus your SQL tuning efforts on the statements that can have the most impact on database performance.

See *Oracle Database Performance Tuning Guide* for a discussion of tuning SQL statements.

Note: Some of the statements that appear in the top SQL statements report may be from Oracle Database XE internal operations, including automatically scheduled internal database jobs (such as statistics gathering jobs).

To monitor the top SQL statements:

1. In SQL Developer, click the Reports navigator tab, and expand the hierarchy as follows: **All Reports**, then **Data Dictionary Reports**, then **Database Administration**, then **Top SQL**.



2. Under Top SQL, select any of the listed "**Top SQL by**" reports: **Buffer Gets**, B**uffer Gets/Rows Proc**, **CPU**, **Disk Reads**, **Executions**, or **Waits**.

   If you are asked to select a connection, select one for `SYSTEM` for `SYS AS SYSDBA`

   Each available report lists the top SQL statements in that category, with the following information for each statement: SQL (the SQL statement), CPU_Seconds, Disk_Reads, Executions, Buffer_gets/rows_proc, Buffer_gets/executions, Elapsed_Seconds, Module.

Monitoring Long Operations

The Active Sessions report in SQL Developer lists active sessions and include the current "up time" for each, which you can check to see if any are running much longer than you would expect.

To monitor long operations:

1. In SQL Developer, click the Reports navigator tab, and expand the hierarchy as follows: **All Reports**, then **Data Dictionary Reports**, then **Database Administration**, then **Sessions**.
2. Under Sessions, select **Active Sessions**.

   If you are asked to select a connection, select one for `SYS AS SYSDBA`.
3. Check the **UP_TIME** value for each listed session, and note any that you consider to be longer than desired or expected.

9

Viewing Database Version and Globalization Information

You can use SQL Developer to view the database version information and national language support (globalization) settings.

This section contains the following topics:

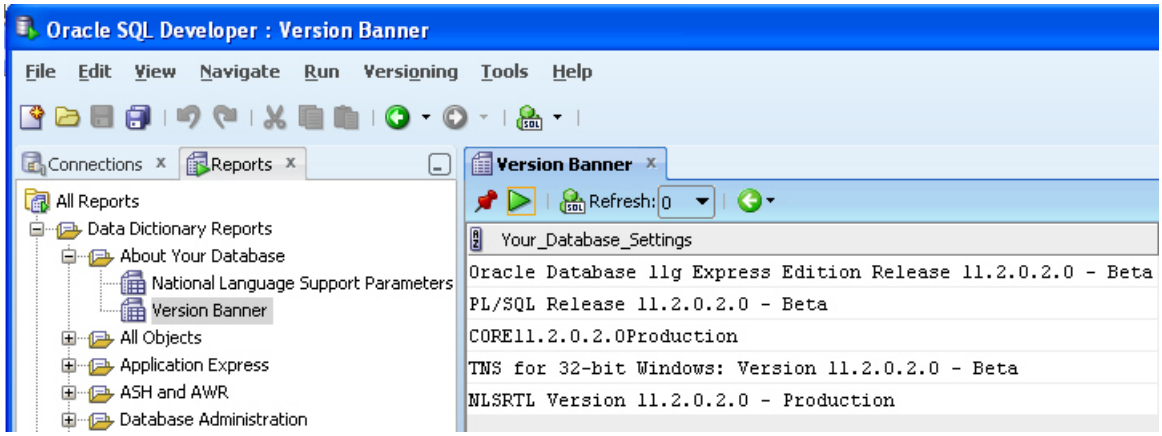
Viewing Database Version Information

To view database version information:

1. In SQL Developer, click the **Reports** tab on the left, near the Connections navigator. (If this tab is not visible, click **View**, then **Reports**.)
2. In the Reports navigator, expand **Data Dictionary Reports**.
3. Under Data Dictionary Reports, expand **About Your Database**.
4. Under About Your Database, click **Version Banner**.

   The Version Banner report is displayed, as shown in Figure 9–1.

Figure 9–1 Version Banner Report



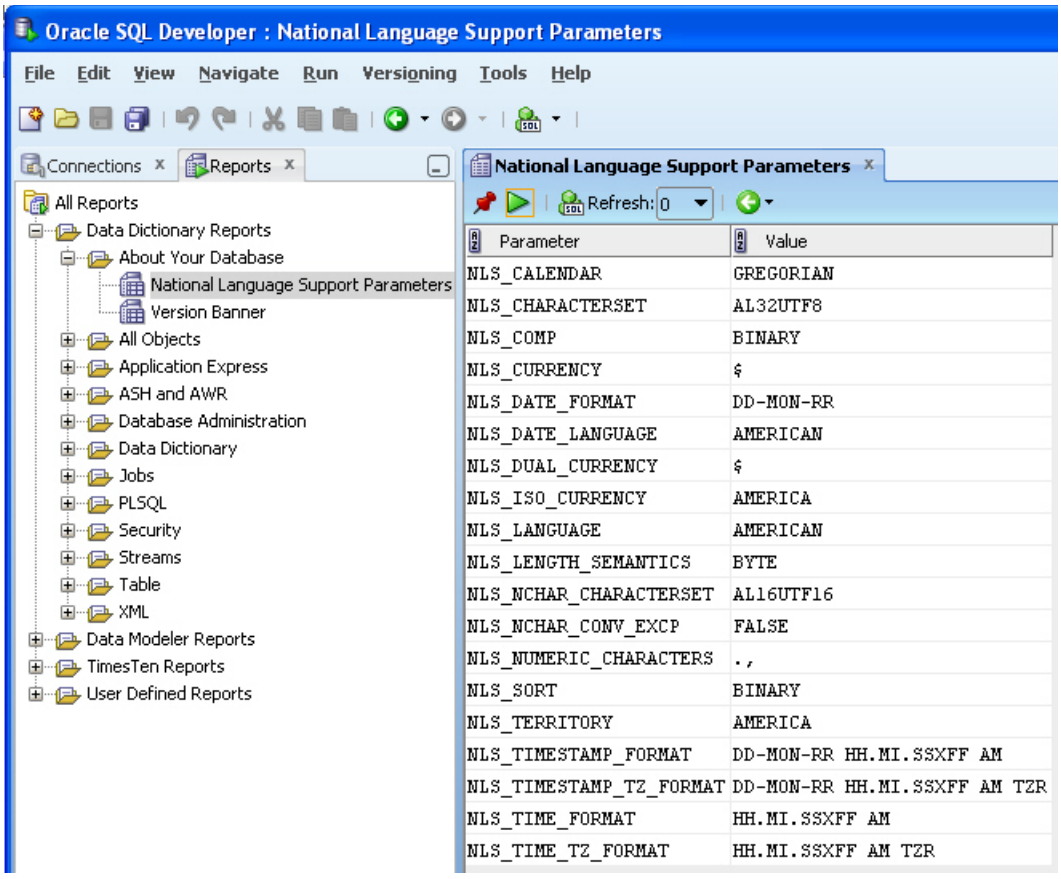
Viewing Database Globalization Information

To view database globalization (national language support, or NLS) parameter information:

1. In SQL Developer, click the **Reports** tab on the left, near the Connections navigator. (If this tab is not visible, click **View**, then **Reports**.)
2. In the Reports navigator, expand **Data Dictionary Reports**.
3. Under Data Dictionary Reports, expand **About Your Database**.
4. Under About Your Database, click **National Language Support Parameters**.

   The National Language Support Parameters report is displayed, as shown in Figure 9–1.

Figure 9–2 National Language Support Parameters Report



For information about globalization support, see the "Working in a Global Environment" chapter in *Oracle Database Express Edition 2 Day Developer's Guide*.

10

Exporting and Importing Metadata and Data

This chapter describes how to export (unload) from and import (load) into Oracle Database XE. You can export and import metadata (database object definitions), data, or both metadata and data. It contains the following topics:

Data can be exported for later importing (loading) into another Oracle database or into a non-Oracle database. Data that has been unloaded from a non-Oracle database can be loaded into an Oracle database, if the data is in a suitable format for loading.

This chapter includes the following topics:

For convenience and the range of features available, you may want to use SQL Developer for export and import operations unless you need to use another tool (command-line utility).

Using SQL Developer for Exporting and Importing

SQL Developer provides convenient wizards for exporting and importing metadata and data:

- To export metadata or data, or both, use the Export Wizard: click **Tools**, then **Database Export**.
- To import metadata or data, or both, use an appropriate method depending on how the material to be imported was created, or the format of the data to be imported. This method might be running a script file, or using the Data Import Wizard to import from a data file (such as a .csv file or a Microsoft Excel .xls file).

See the following examples of using SQL Developer for performing export and import operations:

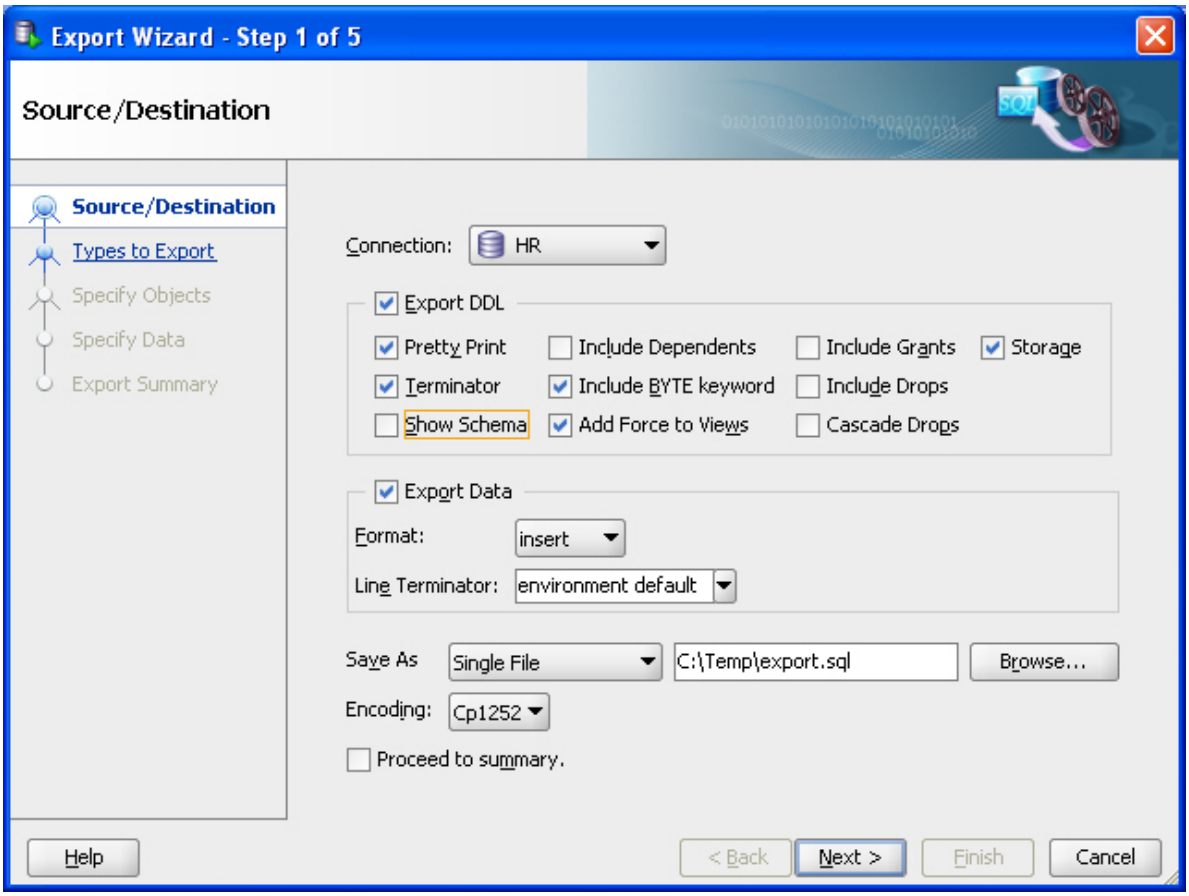
Example: Exporting Metadata and Data for a Table

Assume that you want to export the `REGIONS` table, which is part of the `HR` sample schema, so that it can be created, along with its data, in another schema (either in the same Oracle database or another Oracle database).

To unload the REGIONS table:

1. In SQL Developer, click Tools, then Database Export. Figure 10–1 shows the first page of the Export Wizard, but with entries reflecting selections that you will make.

Figure 10–1 Export Wizard: Source/Destination



2. Accept the default values for the Source/Destination page options, except as follows:

   Connection: Select HR.

   Show Schema: ***Deselect*** (uncheck) this option, so that the HR schema name is not included in CREATE and INSERT statements in the .sql script file that will be created. (This enables you to re-create the table in a schema with any name, such as one not named HR.)

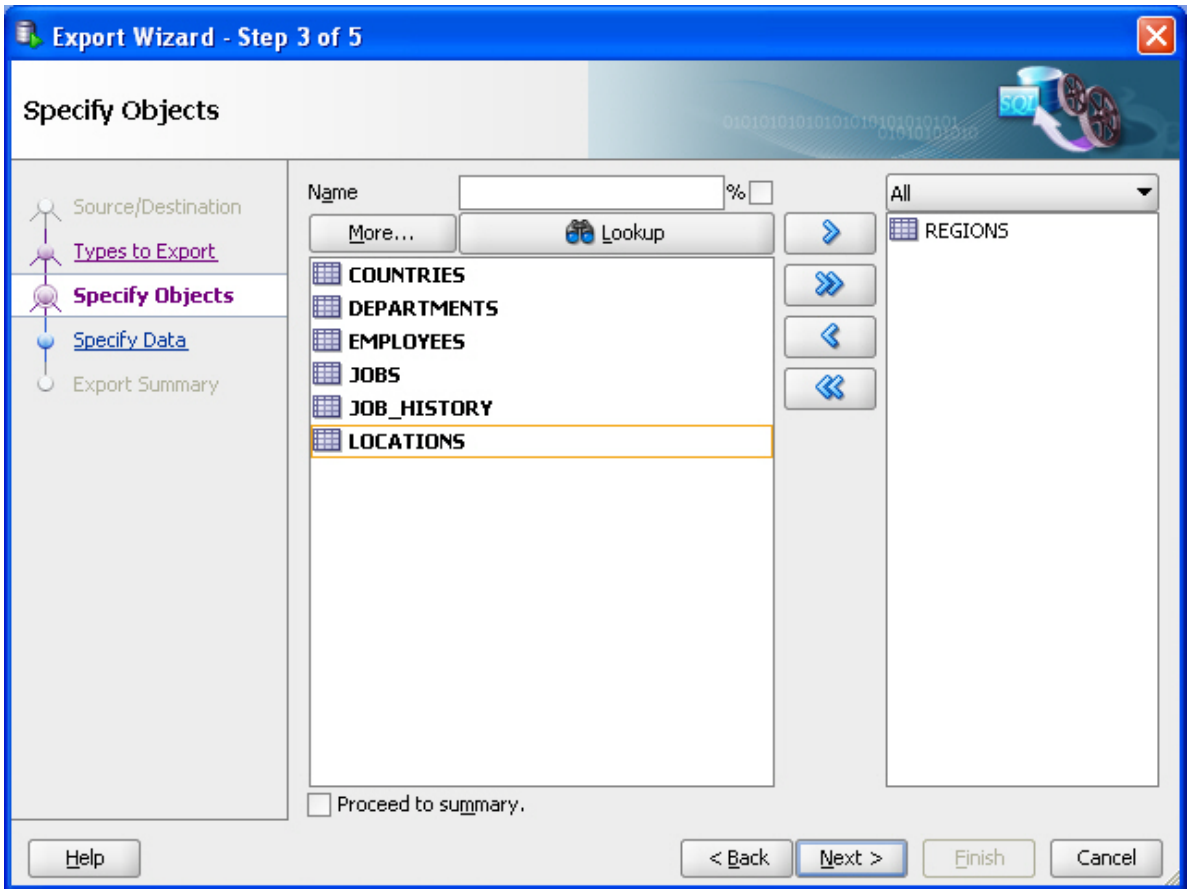
   Save As location: Enter or browse to a desired folder on your local hard drive, and specify the file name for the script file. (In the figure, this file is C:\temp\export.sql.) The script file containing CREATE and INSERT statements will be created in this location.

Note: For explanations of the options on this or any other wizard page, click the **Help** button.

For example, **Format** has other possible values besides the default `insert` , which causes SQL `INSERT` statements to be included to insert the data. Other values include `loader` to cause SQL*Loader files to be created, and `xls` to cause a Microsoft Excel .xls file to be created.

3. Click **Next**.
4. On the Types to Export page, deselect Toggle All, then select *only* **Tables** (because you only want to export a table).
5. Click Next.
6. On the Specify Objects page, click Lookup, then double-click the `REGIONS` table on the left to move it to the right-hand column. Figure 10–2 shows the result of these actions.

Figure 10–2 Export Wizard: Specify Objects



7. Click **Next**.
8. On the Specify Data page, accept the defaults and click **Next**.

   By default, all data from the specified table or tables is exported; however, if you want to limit the data to be exported, you can specify one or more "`WHERE` clauses" in the bottom part of this page.

9. On the Summary page, review the information; and if it is what you want, click **Finish**. (Given what you specified, this causes the export script to be created as `C:\temp\export.sql`.)

   If you need to make any changes, go back to the appropriate page or pages and make them, and then move forward to the Summary page again.

Example: Importing Metadata and Data Using a Script File

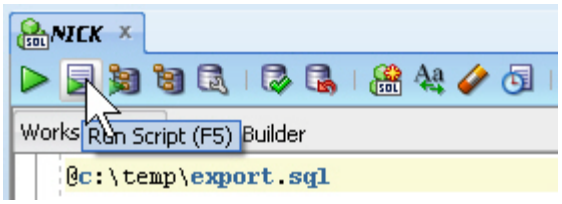
Assume that you wanted to re-create the `REGIONS` table that you exported in "Example: Exporting Metadata and Data for a Table" on page 10-1, but in a different schema. This other schema can be an existing one or one that you create.

For example, assume that you created a user named `NICK` following the instructions in "Example: Creating a User" on page 7-8. To re-create the `REGIONS` table in the schema of user `NICK` by invoking the script in `C:\temp\export.sql` follow these steps using SQL Developer:

1. If you have not already created a database connection for `NICK`, create the connection.
2. Open the `NICK` connection.
3. In the SQL Worksheet for the `NICK` connection, type the following:

   ```
   @c:\temp\export.sql
   ```

4. Click the Run Script icon.



   The Script Output pane shows that the `REGIONS` table has been created and four rows have been inserted.

5. In the Connections navigator, expand the Tables node under the `NICK` connection. You now see the `REGIONS` table.
6. Optionally, click the `REGIONS` table in the Connections navigator, and examine the information under the **Columns** and **Data** tabs in the main display area.

Example: Exporting Data to a Microsoft Excel File

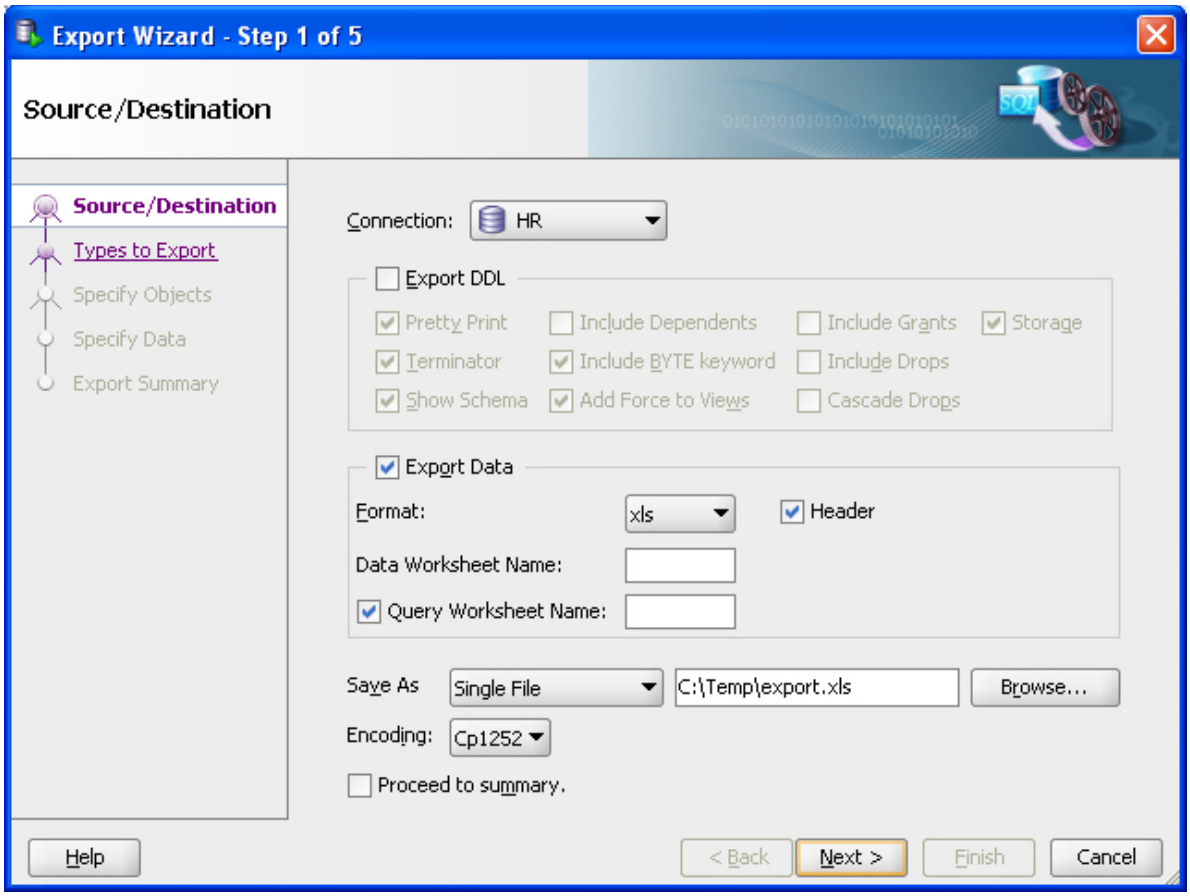
Assume that you want to export only the data from the `REGIONS` table, which is part of the `HR` sample schema, so that the data can be imported into a table with the same column definitions. This might be a `REGIONS` table in another schema (either in the same Oracle database or another Oracle database).

You use the same Database Export wizard, but export only the data, and not the DDL (Data Definition Language statements for creating database objects).

To export the data the `REGIONS` table:

1. In SQL Developer, click Tools, then Database Export. Figure 10–3 shows the first page of the Export Wizard, but with entries reflecting selections that you will make.

Figure 10–3 Export Wizard: Source/Destination Specifying Data Export Only



2. Accept the default values for the Source/Destination page options, except as follows:

   Connection: Select `HR`.

   Export DDL: ***Deselect*** (uncheck) this option. If a .sql script file is generated (which will not happen in this example), it will not contain any `CREATE` statements, but only `INSERT` statements.

   Format: Select xls to have the data saved to a Microsoft Excel .xls file.

   Save As location: Enter or browse to a desired folder on your local hard drive, and specify the file name for the .xls file. (In the figure, this file is `C:\temp\export.xls`.)

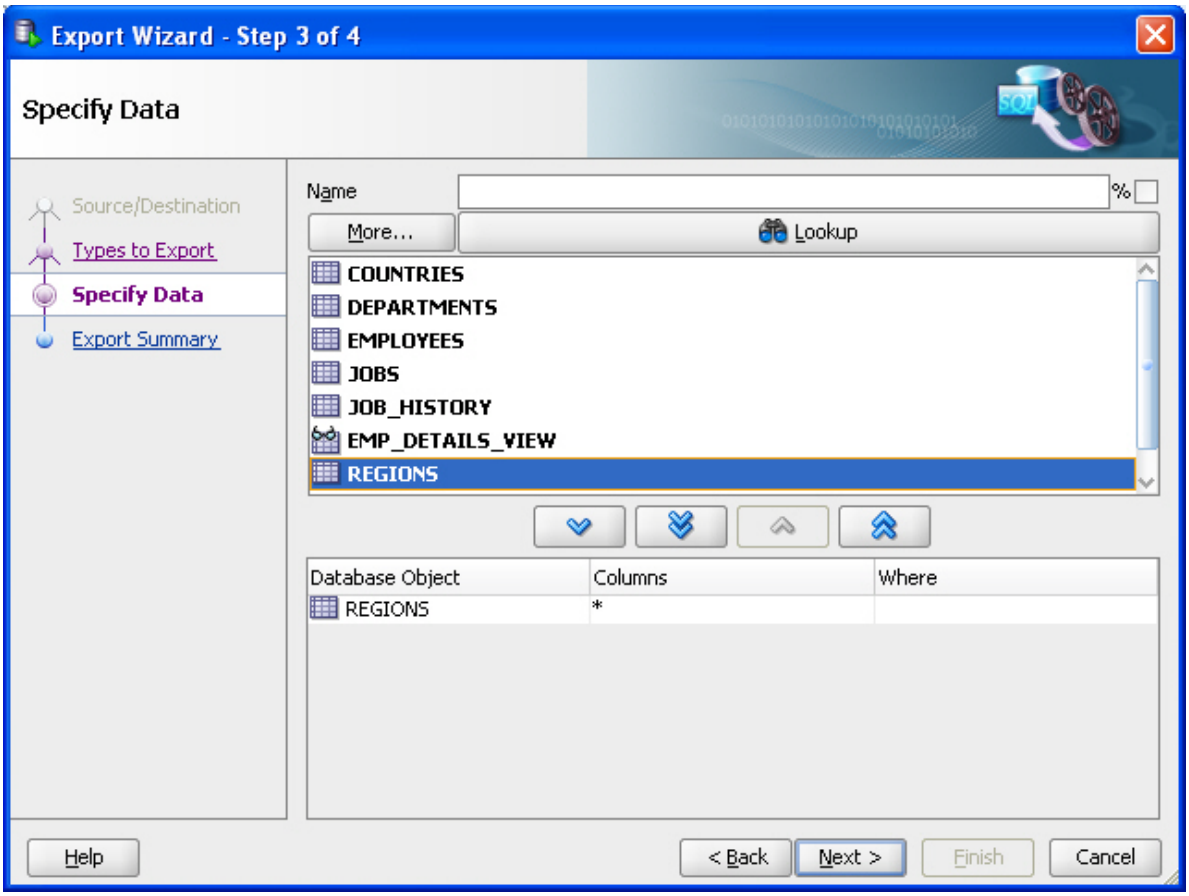
3. Click **Next**.

4. On the Types to Export page, deselect Toggle All, then select *only* **Tables** (because you only want to export data for a table).

5. Click Next.

6. On the Specify Objects page, click **Lookup**, then double-click the `REGIONS` table on the left to have it appear in a row in the bottom part of the page. Figure 10–2 shows the result of these actions.

Figure 10–4 Export Wizard: Specify Objects for Exporting Data



By default, all data from the specified table or tables is exported; however, if you want to limit the data to be exported, you can specify one or more "`WHERE` clauses" in the bottom part of this page.

7. Click **Next**.
8. On the Summary page, review the information; and if it is what you want, click **Finish**. (Given what you specified, this causes the data in the `REGIONS` table to be exported to the file `C:\temp\export.xls`.)

   If you need to make any changes, go back to the appropriate page or pages and make them, and then move forward to the Summary page again.

Example: Importing Data from a Microsoft Excel File

Assume that you wanted to import the data that was exported in "Example: Exporting Data to a Microsoft Excel File" on page 10-4, into a new table that has the same column definitions as the original (`REGIONS`) table.

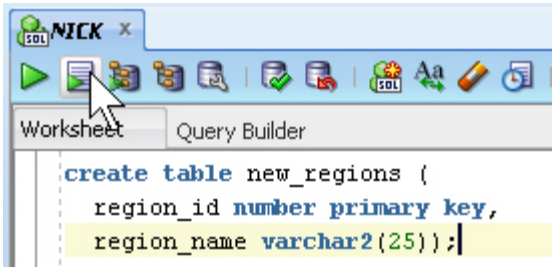
For example, assume that you created a user named `NICK` following the instructions in "Example: Creating a User" on page 7-8. This user wants to take the exported data, add one row in the Excel file, and import it into a new table that has the same column definitions as the `REGIONS` table. (This example is trivial, and adding a row to the Excel file may not be typical, but it is presented merely to illustrate some capabilities.)

To accomplish these goals, follow these steps:

1. In SQL Developer, if you have not already created a database connection for NICK, create the connection.
2. Open the NICK connection.
3. In the SQL Worksheet for the NICK connection, type the following:

```
create table new_regions (
  region_id number primary key,
  region_name varchar2(25));
```

4. Click the Run Script icon.



The Script Output pane shows that the NEW_REGIONS table has been created.

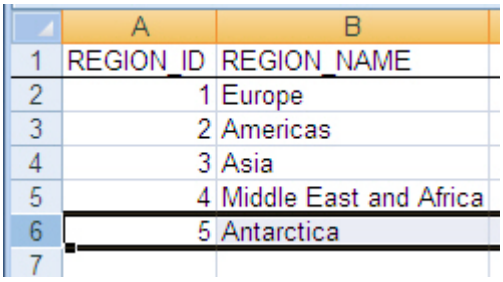
5. In the Connections navigator, expand the Tables node under the NICK connection. You now see the NEW_REGIONS table.

   If you do not see the NEW_REGIONS table, disconnect from NICK (right-click NICK in the Connections navigator and select Disconnect) and connect again, and expand the Tables node.

6. Using Microsoft Excel, open the file containing the exported data (for example, c:\temp\export.xls), and optionally add one or more rows.

   Figure 10–5 shows the original file with one row added for the Antarctica region.

Figure 10–5 Microsoft Excel File with Exported Data (Modified)



	A	B
1	REGION_ID	REGION_NAME
2	1	Europe
3	2	Americas
4	3	Asia
5	4	Middle East and Africa
6	5	Antarctica
7		

7. Save and close the Microsoft Excel .xls file.
8. In SQL Developer, in the Connections navigator display for NICK, right-lick the **NEW_REGIONS** table and select **Import Data**.
9. In the dialog box that is displayed, navigate to the c:\temp folder, select export.xls, and click **Open**.
10. In the Data Import Wizard, accept all the defaults; click **Next** on each page until Summary, and click **Finish** there. (For information about the options on any wizard page, click the **Help** button.)

    The data from the .xls file is loaded into the NEW_REGIONS table and is committed.

Using Other Tools for Exporting and Importing Data

If the SQL Developer export and import wizards are not satisfactory for your needs, you can use one of the command-line utilities available with Oracle Database XE. These other tools are described in the following sections:

Choosing the Right Export/Import Utility

Oracle Database XE provides a number of powerful utilities for exporting and importing data. Table 10–1 provides a summary of these options.

Table 10–1 Summary of Other Export/Import Data Options

Feature or Utility	Description
SQL*Loader utility	■ Command-line interface, invoked with `sqlldr` command ■ Bulk-loads data into the database from external files ■ Supports numerous input formats, including delimited, fixed record, variable record, and stream ■ Loads multiple tables simultaneously ■ Powerful data filtering capabilities
Data Pump Export and Data Pump Import utilities	■ Command-line interface, invoked with `expdp` and `impdp` commands ■ Exports and imports from one Oracle database to another (proprietary binary format) ■ Imports/exports all schema object types ■ Imports/exports entire database, entire schema, multiple schemas, multiple tablespaces, or multiple tables ■ Powerful data filtering capabilities ■ High speed ■ Does not support XMLType data
Export and Import utilities	■ Command-line interface, invoked with `exp` and `imp` commands ■ Exports and imports from one Oracle database to another (proprietary binary format) ■ Supports XMLType data ■ Does not support the `FLOAT` and `DOUBLE` data types ■ Capabilities similar to Data Pump; Data Pump is preferred unless you must import or export XMLType data

Table 10–2 provides a number of load/unload/import/export scenarios and suggests the appropriate option to use for each.

Table 10–2 Import/Export Scenarios and Recommended Options

Import/Export Scenario	Recommended Option
You have to load data that is not delimited. The records are fixed length, and field definitions depend on column positions.	SQL*Loader
You have tab-delimited text data to load, and there are more than 10 tables.	SQL*Loader
You have text data to load, and you want to load only records that meet certain selection criteria (for example, only records for employees in department number 3001).	SQL*Loader
You want to import or export an entire schema from or to another Oracle database. There is no XMLType data in any of the data.	Data Pump Export and Data Pump Import
You want to import or export data from or to another Oracle database. The data contains XMLType data and contains no `FLOAT` or `DOUBLE` data types.	Import (`imp`) and Export (`exp`)

See Also: *Oracle Database Utilities* for more information on Data Pump, the Import and Export utilities, and SQL*Loader

Loading Data with SQL*Loader

SQL*Loader loads data from external datafiles into tables of an Oracle database. A particular datafile can be in fixed record format, variable record format, or stream record format (the default).

The input for a typical SQL*Loader session is a control file, which controls the behavior of SQL*Loader, and some data, located either at the end of the control file itself, or in a separate datafile.

The output of a SQL*Loader session is an Oracle database (where the data is loaded), a log file, a "bad" file, and potentially, a discard file. The log file contains a detailed summary of the load, including a description of any errors that occurred during the load. The bad file contains records that were rejected, either by SQL*Loader or by the Oracle database. The discard file contains records that were filtered out of the load because they did not match any record-selection criteria specified in the control file.

Methods SQL*Loader Uses to Load Data

SQL*Loader uses three different methods to load data, depending on the situation: conventional path, direct path, and external tables.

Conventional Path

A conventional path load is the default loading method. It executes SQL `INSERT` statements to populate tables in an Oracle database. This method can sometimes be slower than other methods because extra overhead is added as SQL statements are generated, passed to Oracle, and executed. It can also be slower because when SQL*Loader performs a conventional path load, it competes equally with all other processes for buffer resources.

Direct Path

A direct path load does not compete with other users for database resources. It eliminates much of the Oracle database overhead by formatting Oracle data blocks and writing them directly to the database files, bypassing much of the data processing

that normally takes place. Therefore, a direct path load can usually load data faster than conventional path.

However, there are several restrictions on direct path loads that may require you to use a conventional path load. For example, direct path load cannot be used on clustered tables or on tables for which there are transactions pending.

See *Oracle Database Utilities* for a complete discussion of situations in which direct path load should and should not be used.

External Tables

An external table load creates an external table for data that is contained in a datafile. The load executes `INSERT` statements to insert the data from the datafile into the target table. An external table load allows modification of the data being loaded by using SQL functions and PL/SQL functions as part of the `INSERT` statement that is used to create the external table.

See *Oracle Database Administrator's Guide* for more information on external tables.

SQL*Loader Features

You can use SQL*Loader to do the following:

- Load data across a network. This means that you can run the SQL*Loader client on a different system from the one that is running the SQL*Loader server.
- Load data from multiple data files during the same load session.
- Load data into multiple tables during the same load session.
- Specify the character set of the data.
- Selectively load data (you can load records based on the records' values).
- Manipulate the data before loading it, using SQL functions.
- Generate unique sequential key values in specified columns.
- Use the operating system's file system to access the datafiles.
- Load data from disk, tape, or named pipe.
- Generate sophisticated error reports, which greatly aid troubleshooting.
- Load arbitrarily complex object-relational data.
- Use secondary datafiles for loading LOBs and collections.

Example: Using SQL*Loader

In the following example, a new table named `dependents` will be created in the `HR` sample schema. It will contain information about dependents of employees listed in the `employees` table of the `HR` schema. After the table is created, SQL*Loader will be used to load data about the dependents from a flat data file into the `dependents` table.

This example requires a data file and a SQL*Loader control file, which you will create in the first two steps.

1. Create the data file, `dependents.dat`, in your current working directory. You can create this file using a variety of methods, such as a spreadsheet application or by simply typing it into a text editor. It should have the following content:

```
100,"Susan, Susie",Kochhar,17-JUN-1997,daughter,101,NULL,
102,David,Kochhar,02-APR-1999,son,101,NULL,
104,Jill,Colmenares,10-FEB-1992,daughter,119,NULL,
```

```
106,"Victoria, Vicki",Chen,17-JUN-1997,daughter,110,NULL,
108,"Donald, Donnie",Weiss,24-OCT-1989,son,120,NULL,
```

This file is a CSV (comma-separated values) file in which the commas act as delimiters between the fields. The field containing the first name is enclosed in double quotation marks in cases where a variant of the official name is also provided—that is, where the first name field contains a comma.

2. Create the SQL*Loader control file, `dependents.ctl`, in your current working directory. You can create this file with any text editor. It should have the following content:

```
LOAD DATA
INFILE dependents.dat
INTO TABLE dependents
REPLACE
FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"'
(
dep_id,
first_name,
last_name,
birthdate,
relation,
relative_id,
benefits
)
```

3. Do one of the following:
   - On Linux: Start a terminal session and log in to the Oracle Database XE host computer with the `oracle` user account.
   - On Windows: Log in to the Oracle Database XE host computer as the user who installed Oracle Database XE, and then open a command window.
4. On Linux, ensure that environment variables are set according to the instructions in "Setting Environment Variables on the Linux Platform" on page 3-5.
5. Start the SQL Command Line (SQL*Plus) and connect as user `hr` by entering the following at the command prompt:

```
sqlplus hr/hr
```

6. At the SQL prompt, create the `dependents` table, as follows:

```
CREATE TABLE dependents (
  dep_id      NUMBER(6),
  first_name  VARCHAR2(20),
  last_name   VARCHAR2(25) CONSTRAINT dep_last_name_nn NOT NULL,
  birthdate   DATE,
  relation    VARCHAR2(25),
  relative_id  NUMBER(6) CONSTRAINT emp_dep_rel_id_fk REFERENCES employees
    (employee_id),
  benefits    CLOB
)
/
```

The constraint on the `last_name` column indicates that a value must be provided. The constraint on the `relative_id` column indicates that it must match a value in the `employee_id` column of the `employees` table. The `benefits` column has a datatype of `CLOB` so that it can hold large blocks of character data. (In this example,

there is not yet any benefits information available so the column is shown as NULL in the data file, dependents.dat.)

After you receive the Table created message, enter exit to exit the SQL Command Line.

7. From within your current working directory (where you created the control and data files), issue the following SQL*Loader command at the system prompt:

```
sqlldr hr/hr DATA=dependents.dat CONTROL=dependents.ctl LOG=dependents.log
```

The data in the dependents.dat file is loaded into the dependents table and the following message is displayed:

```
Commit point reached - logical record count 5
```

Information about the load is written to the log file, dependents.log. The content of the log file looks similar to the following:

```
Control File:   dependents.ctl
Data File:      dependents.dat
  Bad File:     dependents.bad
  Discard File:  none specified

 (Allow all discards)

Number to load: ALL
Number to skip: 0
Errors allowed: 50
Bind array:     64 rows, maximum of 256000 bytes
Continuation:    none specified
Path used:      Conventional

Table DEPENDENTS, loaded from every logical record.
Insert option in effect for this table: REPLACE

   Column Name                  Position   Len  Term Encl Datatype
------------------------------ ---------- ----- ---- ---- ---------------------
DEP_ID                             FIRST     *   ,  O(") CHARACTER
FIRST_NAME                          NEXT     *   ,  O(") CHARACTER
LAST_NAME                           NEXT     *   ,  O(") CHARACTER
BIRTHDATE                           NEXT     *   ,  O(") CHARACTER
RELATION                            NEXT     *   ,  O(") CHARACTER
RELATIVE_ID                         NEXT     *   ,  O(") CHARACTER
BENEFITS                            NEXT     *   ,  O(") CHARACTER


Table DEPENDENTS:
  5 Rows successfully loaded.
  0 Rows not loaded due to data errors.
  0 Rows not loaded because all WHEN clauses were failed.
  0 Rows not loaded because all fields were null.


Space allocated for bind array:                 115584 bytes(64 rows)
Read   buffer bytes: 1048576

Total logical records skipped:          0
Total logical records read:             5
Total logical records rejected:         0
Total logical records discarded:        0
```

```
Run began on Mon Dec 05 16:16:29 2005
Run ended on Mon Dec 05 16:16:42 2005

Elapsed time was:     00:00:12.22
CPU time was:         00:00:00.09
```

You can now work with the `dependents` table, as you would any other table.

Exporting and Importing Data

Oracle Database XE provides the following command-line utilities for exporting and importing data:

- Data Pump Export and Data Pump Import
- Export and Import

The following sections provide an overview of each utility. For a summary of when you might want to use each utility, see Table 10–2 on page 10-9.

- Exporting and Importing with Data Pump Export and Data Pump Import on page 10-13
- Exporting and Importing Data with the Export and Import Utilities on page 10-16

> **See Also:** *Oracle Database Utilities* for detailed information on these utilities, including command line parameter descriptions and additional examples.

Exporting and Importing with Data Pump Export and Data Pump Import

The Data Pump Export utility exports data and metadata into a set of operating system files called a **dump file set**. The Data Pump Import utility imports an export dump file set into a target Oracle database.

A dump file set is made up of one or more disk files that contain table data, database object metadata, and control information. The files are written in a proprietary, binary format, which means that the dump file set can be imported only by the Data Pump Import utility. The dump file set can be imported to the same database or it can be moved to another system and loaded into the Oracle database there.

Because the dump files are written by the database, rather than by the Data Pump client application, you must create *directory objects* for the directories to which files will be written. A **directory object** is a database object that is an alias for a directory in the host operating system's file system.

Data Pump Export and Import enable you to move a subset of the data and metadata. This is done by using Data Pump parameters to specify export and import modes, as well as various filtering criteria.

You can also perform exports and imports over a network. In a network export, the data from the source database instance is written to a dump file set on the connected database instance. In a network import, a target database is loaded directly from a source database with no intervening dump files. This allows export and import operations to run concurrently, minimizing total elapsed time.

Data Pump Export and Import also provide a set of interactive commands so that you can monitor and modify ongoing export and import jobs.

> **Note:** Data Pump Export and Data Pump Import do not support XMLType data. If you need to export and import XMLType data, use the Export and Import options described in "Exporting and Importing Data with the Export and Import Utilities" on page 10-16.

Example: Using Data Pump Export and Data Pump Import In this example, suppose that you want to make some changes to the HR sample schema and then test those changes without affecting the current HR schema. You could export the HR schema and then import it into a new HRDEV schema, where you could perform development work and conduct testing. To do this, take the following steps:

1. Do one of the following:
   - On Windows: Log in to the Oracle Database XE host computer as the user who installed Oracle Database XE, and then open a command window.
   - On Linux: Start a terminal session and log in to the Oracle Database XE host computer with the oracle user account.
2. On Linux, ensure that environment variables are set according to the instructions in "Setting Environment Variables on the Linux Platform" on page 3-5.
3. At the command prompt, issue the command appropriate to your operating system, to create the directory where the exported files will be placed:

   On Windows:

   ```
   MKDIR c:\oraclexe\app\tmp
   ```

   On Linux:

   ```
   mkdir /usr/lib/oracle/xe/tmp
   ```

4. Start the SQL Command Line (SQL*Plus) and connect as user SYSTEM by entering the following at the command prompt:

   ```
   sqlplus SYSTEM/password
   ```

   where *password* is the password that you specified for the SYS and SYSTEM user accounts upon installation (Windows) or configuration (Linux) of Oracle Database XE.
5. At the SQL prompt, enter the following commands to create a directory object named dmpdir for the tmp directory that you just created, and to grant read and write access to it for user HR.

   On Windows:

   ```
   CREATE OR REPLACE DIRECTORY dmpdir AS 'c:\oraclexe\app\tmp';
   GRANT READ,WRITE ON DIRECTORY dmpdir TO hr;
   ```

   On Linux:

   ```
   CREATE OR REPLACE DIRECTORY dmpdir AS '/usr/lib/oracle/xe/tmp';
   GRANT READ,WRITE ON DIRECTORY dmpdir TO hr;
   ```

6. Export the HR schema to a dump file named schema.dmp by issuing the following command at the system command prompt:

   ```
   expdp SYSTEM/password SCHEMAS=hr DIRECTORY=dmpdir DUMPFILE=schema.dmp
   LOGFILE=expschema.log
   ```

where *password* is the password for the SYSTEM user.

As the export operation takes place, messages similar to the following are displayed:

```
Connected to: Oracle Database 11g Express Edition Release 11.2.0.1.0 -
Production
Starting "SYSTEM"."SYS_EXPORT_SCHEMA_01":  SYSTEM/******** SCHEMAS=hr
  DIRECTORY=dmpdir DUMPFILE=schema.dmp LOGFILE=expschema.log
Estimate in progress using BLOCKS method...
Processing object type SCHEMA_EXPORT/TABLE/TABLE_DATA
Total estimation using BLOCKS method: 448 KB
Processing object type SCHEMA_EXPORT/USER
Processing object type SCHEMA_EXPORT/SYSTEM_GRANT
Processing object type SCHEMA_EXPORT/ROLE_GRANT
Processing object type SCHEMA_EXPORT/DEFAULT_ROLE
Processing object type SCHEMA_EXPORT/TABLESPACE_QUOTA
Processing object type SCHEMA_EXPORT/PRE_SCHEMA/PROCACT_SCHEMA
Processing object type SCHEMA_EXPORT/SEQUENCE/SEQUENCE
Processing object type SCHEMA_EXPORT/TABLE/TABLE
Processing object type SCHEMA_EXPORT/TABLE/INDEX/INDEX
Processing object type SCHEMA_EXPORT/TABLE/CONSTRAINT/CONSTRAINT
Processing object type SCHEMA_EXPORT/TABLE/INDEX/STATISTICS/INDEX_STATISTICS
Processing object type SCHEMA_EXPORT/TABLE/COMMENT
Processing object type SCHEMA_EXPORT/PROCEDURE/PROCEDURE
Processing object type SCHEMA_EXPORT/PROCEDURE/ALTER_PROCEDURE
Processing object type SCHEMA_EXPORT/VIEW/VIEW
Processing object type SCHEMA_EXPORT/TABLE/CONSTRAINT/REF_CONSTRAINT
Processing object type SCHEMA_EXPORT/TABLE/TRIGGER
Processing object type SCHEMA_EXPORT/TABLE/STATISTICS/TABLE_STATISTICS
. . exported "HR"."COUNTRIES"                            6.093 KB      25 rows
. . exported "HR"."DEPARTMENTS"                          6.640 KB      27 rows
. . exported "HR"."EMPLOYEES"                            15.77 KB     107 rows
. . exported "HR"."JOBS"                                 6.609 KB      19 rows
. . exported "HR"."JOB_HISTORY"                          6.585 KB      10 rows
. . exported "HR"."LOCATIONS"                            7.710 KB      23 rows
. . exported "HR"."REGIONS"                              5.296 KB       4 rows
Master table "SYSTEM"."SYS_EXPORT_SCHEMA_01" successfully loaded/unloaded
******************************************************************************
Dump file set for SYSTEM.SYS_EXPORT_SCHEMA_01 is:
  C:\ORACLEXE\APP\TMP\SCHEMA.DMP
Job "SYSTEM"."SYS_EXPORT_SCHEMA_01" successfully completed at 11:48:46
```

The schema.dmp file and the expschema.log file are written to the dmpdir directory.

7. Import the dump file, schema.dmp, into another schema, in this case, HRDEV. You use the REMAP_SCHEMA command parameter to indicate that objects are to be imported into a schema other than their original schema. Because the HRDEV user account does not already exist, the import process automatically creates it. In this example, you will import everything except constraints, ref_constraints, and indexes. If a table already exists, it is replaced with the table in the export file.

   At the operating system command prompt, issue the following command:

   ```
   impdp SYSTEM/password SCHEMAS=hr DIRECTORY=dmpdir DUMPFILE=schema.dmp
     REMAP_SCHEMA=hr:hrdev EXCLUDE=constraint, ref_constraint, index
     TABLE_EXISTS_ACTION=replace LOGFILE=impschema.log
   ```

   where *password* is the password for the SYSTEM user.

As the import operation takes place, messages similar to the following are displayed (this output is also written to the `impschema.log` file in the `dmpdir` directory):

```
Connected to: Oracle Database 11g Express Edition Release 11.2.0.1.0 -
Production
Master table "SYSTEM"."SYS_IMPORT_SCHEMA_01" successfully loaded/unloaded
Starting "SYSTEM"."SYS_IMPORT_SCHEMA_01":  SYSTEM/******** SCHEMAS=hr
  DIRECTORY=dmpdir DUMPFILE=schema.dmp REMAP_SCHEMA=hr:hrdev
  EXCLUDE=constraint, ref_constraint, index TABLE_EXISTS_ACTION=replace
LOGFILE=impschema.log
Processing object type SCHEMA_EXPORT/USER
Processing object type SCHEMA_EXPORT/SYSTEM_GRANT
Processing object type SCHEMA_EXPORT/ROLE_GRANT
Processing object type SCHEMA_EXPORT/DEFAULT_ROLE
Processing object type SCHEMA_EXPORT/TABLESPACE_QUOTA
Processing object type SCHEMA_EXPORT/PRE_SCHEMA/PROCACT_SCHEMA
Processing object type SCHEMA_EXPORT/SEQUENCE/SEQUENCE
Processing object type SCHEMA_EXPORT/TABLE/TABLE
Processing object type SCHEMA_EXPORT/TABLE/TABLE_DATA
. . imported "HRDEV"."COUNTRIES"                          6.093 KB      25 rows
. . imported "HRDEV"."DEPARTMENTS"                        6.640 KB      27 rows
. . imported "HRDEV"."EMPLOYEES"                          15.77 KB     107 rows
. . imported "HRDEV"."JOBS"                               6.609 KB      19 rows
. . imported "HRDEV"."JOB_HISTORY"                        6.585 KB      10 rows
. . imported "HRDEV"."LOCATIONS"                          7.710 KB      23 rows
. . imported "HRDEV"."REGIONS"                            5.296 KB       4 rows
Processing object type SCHEMA_EXPORT/TABLE/COMMENT
Processing object type SCHEMA_EXPORT/PROCEDURE/PROCEDURE
Processing object type SCHEMA_EXPORT/PROCEDURE/ALTER_PROCEDURE
Processing object type SCHEMA_EXPORT/VIEW/VIEW
Processing object type SCHEMA_EXPORT/TABLE/TRIGGER
Processing object type SCHEMA_EXPORT/TABLE/STATISTICS/TABLE_STATISTICS
Job "SYSTEM"."SYS_IMPORT_SCHEMA_01" successfully completed at 11:49:49
```

The `HRDEV` schema is now populated with data from the `HR` schema.

8. Assign a password to the newly created `HRDEV` user account. To do so, start the SQL Command Line and connect as user `SYSTEM` (as you did in step 4), and then at the SQL prompt, enter the following `ALTER USER` statement:

   ```
   ALTER USER hrdev IDENTIFIED BY hrdev;
   ```

   This statement assigns the password `hrdev`.

   You can now work in the `HRDEV` schema without affecting your production data in the `HR` schema.

Exporting and Importing Data with the Export and Import Utilities

The Export and Import utilities provide a simple way for you to transfer data objects between Oracle databases. They are invoked with the `exp` and `imp` commands, respectively. These utilities provide support for XMLType data, whereas the Data Pump Export and Import utilities do not.

> **Note:** The Export and Import utilities do not support the `FLOAT` and `DOUBLE` data types. If your data contains these types and does not contain XMLType data, you must use Data Pump Export and Import, described in "Exporting and Importing with Data Pump Export and Data Pump Import" on page 10-13.

When you run the Export utility against an Oracle database, objects (such as tables) are extracted, followed by their related objects (such as indexes, comments, and grants), if any. The extracted data is written to an export dump file. The dump file is an Oracle binary-format dump file that can be read only by the Import utility. The version of the Import utility cannot be earlier than the version of the Export utility used to create the dump file.

> **Note:** Dump files generated by the Export (`exp`) utility can only be imported by the Import (`imp`) utility; they cannot be imported with the Data Pump Import (`impdp`) utility.

Like Data Pump Import and Export, data exported with the Export utility can be imported with the Import utility into the same or a different Oracle database.

See *Oracle Database Utilities* for further information about the Export and Import utilities and for examples of how to use them.

11

Backing Up and Recovering

This section discusses backing up and restoring the entire database and recovering data from individual schema objects. It includes the following topics:

Backing Up and Restoring the Database

This section includes the following topics:

> **See Also:** "Recovery-Related Structures in the Flash Recovery Area" on page 6-5

About Backing Up and Restoring the Database

Backing up and restoring Oracle Database XE is based on protecting the physical files that make up the database: the datafiles, the control file, the server parameter file (`SPFILE`), and, if in `ARCHIVELOG` mode, the redo log files.

In Oracle Database XE, the database backup and recovery facility is based upon the Recovery Manager (`RMAN`) utility that is integrated into the database. Although there is an `RMAN` command line client similar to the SQL Command Line, you do not need to interact with it directly to back up or restore your database. Oracle Database XE includes backup and restore scripts that you access using menu choices on your desktop. These scripts perform a full backup and restore of the entire database, and store backup files in the flash recovery area.

Automatic Management of Backup Storage

Oracle Database XE implements a *backup retention policy* that dictates that two complete backups of the database must be retained, to provide a level of redundant protection for the database. In `ARCHIVELOG` mode, all archived logs required for media recovery from either backup are also retained. The database automatically manages backups and archived logs in the flash recovery area, deleting any that are *obsolete* (no

longer needed to satisfy the retention policy) as space is needed for new files. The backup script provided with Oracle Database XE also deletes obsolete backups and archived logs at the end of each backup job.

Backup Script

The provided backup script performs *online* backups of a database that is in ARCHIVELOG mode and *offline* backups of a database that is in NOARCHIVELOG mode. **Online backups** are backups that can run while the database is running. **Offline backups** are backups that run when the database is in the mounted (but not open) state. For offline backups, the backup script automatically puts the database in the proper state. During offline backups, the database is unavailable to your applications.

You run the backup script by running the Backup Database command from the desktop.

Restore Script

The provided restore script restores the database differently depending on whether log archiving is on or off:

- Log archiving on (ARCHIVELOG mode)—The restore script restores the backed up database files, and then uses the online and archived redo log files to *recover* the database to the state it was in before the software or media failure occurred. All committed transactions that took place after the last backup are recovered, and any uncommitted transactions that were under way when the failure took place are rolled back (using undo data from the restored undo tablespace).
- Log archiving off (NOARCHIVELOG mode)—The restore script restores the database to its state at the last backup. Any transactions that took place after the last backup are lost.

You run the restore script by running the Restore Database command from the desktop.

See Also:

- *Oracle Database Backup and Recovery User's Guide* for more information on Oracle database backup and recovery with RMAN
- "Archived Redo Log Files" on page 6-7

Enabling ARCHIVELOG Mode for Media Failure Protection

This section describes how to turn on ARCHIVELOG mode so that your database is fully protected not only against operating system and Oracle instance failure, but also against media (disk) failure. The following topics are covered:

See Also: "Archived Redo Log Files" on page 6-7

Viewing the Current ARCHIVELOG Mode Setting

To view the current ARCHIVELOG mode setting:

1. Using the SQL Command Line, log in to the database and connect as SYSDBA, as described in "Logging In and Connecting to the Database as SYSDBA" on page 7-6.
2. Enter the following command:

```
SELECT log_mode FROM v$database;
```

The LOG_MODE value will be either ARCHIVELOG (that is, on) or NOARCHIVELOG (that is, off).

Turning on ARCHIVELOG Mode

Turning on ARCHIVELOG mode is a one-time operation. After it is turned on, it remains on until you turn it off. Restarting the database does not change the ARCHIVELOG mode setting.

> **Note:** If you turn on ARCHIVELOG mode, you must perform regular backups of the database to avoid completely filling the flash recovery area. A completely filled flash recovery area can lead to database failure.

To turn on ARCHIVELOG mode:

1. Using the SQL Command Line, log in to the database and connect as SYSDBA, as described in "Logging In and Connecting to the Database as SYSDBA" on page 7-6.
2. At the SQL Command Line prompt, enter the following command:

   ```
   SHUTDOWN IMMEDIATE
   ```

   If the command is successful, it displays the following output.

   ```
   Database closed.
   Database dismounted.
   ORACLE instance shut down.
   ```

3. At the SQL Command Line prompt, enter the following command:

   ```
   STARTUP MOUNT
   ```

   If the command is successful, it displays the following output. (System global area sizes will vary depending on the amount of physical memory in your Oracle Database XE host computer.)

   ```
   ORACLE instance started.

   Total System Global Area  803500032 bytes
   Fixed Size                  1386556 bytes
   Variable Size             222300100 bytes
   Database Buffers          574619648 bytes
   Redo Buffers                5193728 bytes
   Database mounted.
   ```

4. Enter the following command:

   ```
   ALTER DATABASE ARCHIVELOG;
   ```

   If the command is successful, it displays the following output:

   ```
   Database altered.
   ```

5. Enter the following command:

   ```
   ALTER DATABASE OPEN;
   ```

   If the command is successful, it displays the following output:

```
Database altered.
```

The database is now running with the new ARCHIVELOG mode setting.

6. Change the size of the flash recovery area to at least 15 gigabytes to allow for the extra space required for archived log files.

   For example, to set the flash recovery area size to 20 gigabytes, enter the following command:

   ```
   ALTER SYSTEM SET DB_RECOVERY_FILE_DEST_SIZE = 20G;
   ```

   See "Changing the Flash Recovery Area Size" on page 6-15 for more information.

Note: To turn off ARCHIVELOG mode (that is, to set NOARCHIVELOG mode), follow the previous steps 1 through 5, but enter the following command in Step 4:

```
ALTER DATABASE NOARCHIVELOG;
```

Caution: When you change the ARCHIVELOG mode setting for your database, all of your existing backups become unusable. You must immediately perform a backup after changing the ARCHIVELOG mode, as described in "Backing Up the Database" on page 11-4.

See Also: "Shutting Down the Database Using the SQL Command Line" on page 2-4 for information on how to handle a failed SHUTDOWN IMMEDIATE command.

Backing Up the Database

Note: If ARCHIVELOG mode is on, the script performs an online backup. The database is available during the backup.

If ARCHIVELOG mode is off, the script performs an offline backup. The database is shut down during the backup and restarted afterwards. Your applications are unavailable during the backup.

To back up the database:

1. Do one of the following:
   - On Windows: Log in to the Oracle Database XE host computer as a user who is a member of the ORA_DBA user group. This is typically the user that installed Oracle Database XE.
   - On Linux: Log in to the Oracle Database XE host computer as a user who is a member of the dba user group. This is typically the oracle user.

   See "Operating System Authentication" on page 7-4 for more information.
2. Do one of the following:
   - On Windows: Click **Start**, point to **Programs (or All Programs)**, point to **Oracle Database 11g Express Edition**, and then select **Backup Database**.

- On Linux with Gnome: In the Applications menu, point to **Oracle Database 11g Express Edition**, and then select **Backup Database**.
- On Linux with KDE: Click the icon for the K Menu, point to **Oracle Database 11g Express Edition**, and then select **Backup Database**.

A console window opens so that you can interact with the backup script.

If running in `ARCHIVELOG` mode, the script displays the following output:

```
Doing online backup of the database.
```

If running in `NOARCHIVELOG` mode, the script displays the following output:

```
Warning: Log archiving (ARCHIVELOG mode) is currently disabled. If
you restore the database from this backup, any transactions that take
place between this backup and the next backup will be lost. It is
recommended that you enable ARCHIVELOG mode before proceeding so
that all transactions can be recovered upon restore. See the section
'Enabling ARCHIVELOG Mode...' in the online help for instructions.
Backup with log archiving disabled will shut down and restart the
database. Are you sure [Y/N]?
```

3. If prompted, enter **`y`** and press **Enter** to confirm the database shutdown and begin the backup.

   After the backup is complete, the script displays the following output:

   ```
   Backup of the database succeeded.
   Log file is at location
   Press ENTER key to exit
   ```

   where *`location`* is the location of the log file.

4. Press **Enter** to close the Backup Database window.

Logs containing the output from the last two backups are stored in the locations listed in Table 11–1.

Table 11–1 Backup Script Output Log Locations

Platform	Location
Linux	`$HOME/oxe_backup_current.log`
	`$HOME/oxe_backup_previous.log`
Windows	`C:\oraclexe\app\oracle\product\11.2.0\server\database\OXE_BACKUP_CURRENT.LOG.`
	`C:\oraclexe\app\oracle\product\11.2.0\server\database\OXE_BACKUP_PREVIOUS.LOG.`

You can review the output of the two most recent backup attempts in the files `OXE_BACKUP_CURRENT.LOG` and `OXE_BACKUP_PREVIOUS.LOG`.

See Also:

- "About Backing Up and Restoring the Database" on page 11-1

Scheduling Automatic Backups

To schedule automatic backups, use any operating system or third party task scheduling software to run the supplied backup script for your platform. Table 11–2 shows the name and path of this script for each platform.

Table 11–2 Name and Path of the Backup Script for Each Platform

Platform	Backup Script Name and Path
Linux	`/usr/lib/oracle/xe/app/oracle/product/11.2.0/server/config/scripts/backup.sh`
Windows	`C:\oraclexe\app\oracle\product\11.2.0\server\bin\Backup.bat`

Restoring and Recovering the Database

You restore and recover the database with the supplied restore script. The instructions in this section are based on the following assumptions:

- A software failure, media (disk) failure, or operator error caused the loss or corruption of one or more database files, rendering the database unusable.
- In the flash recovery area, the backup sets and, if in `ARCHIVELOG` mode, archived logs, are intact and available.
- If in `ARCHIVELOG` mode, the online redo logs are intact and available.

> **Note:** If they are not available, the database is restored to the point of the last transaction included in an archived log. See *Oracle Database Backup and Recovery User's Guide* for more information.

- The Oracle Database XE host computer and operating system are operational.
- The Oracle Database XE installed software (binaries) are intact and operational.

In situations where not all of these assumptions are true, before proceeding with the following steps to restore and recover the database, you may first have to complete one or more of the following tasks: repair or replace computer hardware, reinstall operating system software, or reinstall Oracle Database XE. After reinstalling Oracle Database XE, if your flash recovery area was previously on a separate disk from your Oracle Database XE installation and is still available, you must specify the location of the flash recovery area so that the restore script can find the required files. See "Setting Flash Recovery Area Location and Size" on page 6-14 for instructions.

To restore the database:

1. Do one of the following:
   - On Windows: Log in to the Oracle Database XE host computer as a user who is a member of the `ORA_DBA` user group. This is typically the user that installed Oracle Database XE.
   - On Linux: Log in to the Oracle Database XE host computer as a user who is a member of the `dba` user group. This is typically the `oracle` user.

   See "Operating System Authentication" on page 7-4 for more information.
2. Do one of the following:
   - On Windows: Click **Start**, point to **Programs (or All Programs)**, point to **Oracle Database 11g Express Edition**, and then select **Restore Database**.
   - On Linux with Gnome: In the Applications menu, point to **Oracle Database 11g Express Edition**, and then select **Restore Database**.
   - On Linux with KDE: Click the icon for the K Menu, point to **Oracle Database 11g Express Edition**, and then select **Restore Database**.

A console window opens so that you can interact with the restore script. The script displays the following output:

```
This operation will shut down and restore the database. Are you sure [Y/N]?
```

3. Enter **y** and press **Enter** to confirm the database restore.

   The database is shut down, and the script runs RMAN to restore the database and, if running in ARCHIVELOG mode, recover all changes since the last backup.

   Note: In some situations, such as when you run the Restore Database command in a new Oracle Database XE installation before having backed up, the restore script may prompt you for the location of the flash recovery area:

   ```
   Enter the flash recovery area location:
   ```

   If so prompted, enter the complete path to the location of the flash recovery area. (The default location for each platform is listed in Table 6–2 on page 6-6.) The restore script then restores the database from the backup files in this location.

   If the restore and recovery process is completed successfully, the database is opened again. The script then displays the following output:

   ```
   Restore of the database succeeded.
   Log file is at location
   Press any key to continue . . .
   ```

   where *location* is the location of the log file.

   If the restore and recovery process fails, messages describing the error are displayed.

   ```
   ====================   ERROR ============================
            Restore of the database Failed.
            RMAN Error - See log for details
            Log file is at location
    ====================   ERROR ============================
   ```

   Refer to the log file for details on the cause of the error.

4. Press **Enter** to close the Restore Database window.

See Also:

- "About Backing Up and Restoring the Database" on page 11-1
- "Online Redo Log Files" on page 6-6
- "Recovery-Related Structures in the Flash Recovery Area" on page 6-5

Viewing and Restoring Historical Data with Flashback Query

The Flashback Query feature of Oracle Database Express Edition (Oracle Database XE) enables you to view data at a point in time in the past. You can then reconstruct lost data that was deleted or changed by accident.

This section contains the following topics:

About Flashback Query

When you write a Flashback Query, you add a clause to the `SELECT` statement that specifies either a time or a system change number (SCN). The query then uses the committed data from the corresponding time. The Flashback Query feature does not change any data; it queries only. It is up to you to analyze the historical data and then construct and issue data manipulation language (DML) statements to restore data.

The Flashback Query feature retrieves historical data by applying undo data as needed. The length of time that you can flash back therefore depends on the amount of undo data that is available. For more information on the Flashback Query feature, see the discussion of the *flashback_query_clause*, including the `AS OF` clause, for the `SELECT` statement in *Oracle Database SQL Language Reference*.

Example: Recovering Data with Flashback Query

This example uses a Flashback Query to examine the state of a table at a previous time. Suppose that you discover at 12:30 p.m. that the row for employee Chung was deleted from the `employees` table. You also know that at 9:30 a.m. the data for Chung was correctly stored in the database. You can use a Flashback Query to examine the contents of the table at 9:30 a.m. to find out what data was lost. If appropriate, you can then reinsert the lost data.

Example 11–1 retrieves the state of the record for `Chung` at 9:30 a.m., April 4, 2005.

Example 11–1 Retrieving a Row with Flashback Query

```
SELECT * FROM employees AS OF TIMESTAMP
   TO_TIMESTAMP('2005-04-04 09:30:00', 'YYYY-MM-DD HH:MI:SS')
   WHERE last_name = 'Chung';
```

The update in Example 11–2 restores Chung's information to the `employees` table.

Example 11–2 Reinserting a Row After a Flashback Query

```
INSERT INTO employees
    (SELECT * FROM employees AS OF TIMESTAMP
     TO_TIMESTAMP('2005-04-04 09:30:00', 'YYYY-MM-DD HH:MI:SS')
     WHERE last_name = 'Chung');
```

See Also:

- "About Flashback Query" on page 11-8

Tips for Using Flashback Query

Keep the following in mind when using a Flashback Query (`SELECT ... AS OF`):

- You can specify or omit the `AS OF` clause for each table in the query and specify different times for different tables. Use an `AS OF` clause in a query to perform data definition language (DDL) operations (such as creating and truncating tables) or DML operations (such as inserting and deleting) in the same session as the query.

- To use the results of a Flashback Query in a DDL or DML statement that affects the current state of the database, use an `AS OF` clause inside an `INSERT` or `CREATE TABLE AS SELECT` statement.

Recovering Dropped Tables

When you drop (delete) a table, the database does not immediately remove the space associated with the table. Instead, the database renames the table and places it and any dependent objects in a *recycle bin*, where, in case the table was dropped in error, it can be recovered at a later time.

This section contains the following topics:

About the Recycle Bin

The **recycle bin** is a data dictionary table containing information about dropped objects. Dropped objects and any dependent objects (such as indexes, constraints, nested tables, and so on) are not removed and still occupy space until you purge them from the recycle bin or until they are automatically purged by the database when available space becomes low.

You can **restore** objects from the recycle bin, which is equivalent to "undropping" them. When you restore an object, it is returned to the state that it was in before the drop operation. When you restore a table, all of the table's dependent objects are also automatically restored.

You can use SQL Developer to view the contents of the recycle bin, restore dropped objects, and purge objects from the recycle bin. To view, purge, or restore objects owned by a particular user, you must log in as that user.

Note: If you log in and connect as `SYSDBA` with the SQL Command Line (SQL*Plus), you can view, restore, and purge objects owned by other users. See *Oracle Database Administrator's Guide* for details.

See Also:

- "Logging In and Connecting to the Database as SYSDBA" on page 7-6

Viewing Recycle Bin Contents

To view recycle bin contents:

1. In SQL Developer, open a database connection to the schema of interest.
2. In the Connections navigator, expand the connection and select the **Recycle Bin** node.
3. (Optional) Click a table name to see its properties.

See Also:

- "About the Recycle Bin" on page 11-9

Example: Restoring a Table from the Recycle Bin

Suppose you drop the JOB_HISTORY table in the HR schema, and then decide that you want to recover it. You can recover (undrop) the table by restoring it from the recycle bin.

To drop the JOB_HISTORY table and then restore it from the recycle bin:

1. In SQL Developer, connect to the HR user.

   Note: You must have previously unlocked the HR account and assigned it a password. See "Altering Users" on page 7-10 and "Locking and Unlocking User Accounts" on page 7-10 for more information.

2. In the Connections navigator under HR, expand Tables.
3. Right-click JOB_HISTORY and select **Drop**.
4. In the Drop dialog box, click **Apply**.
5. In the Connections navigator, right-click **Recycle Bin** and select **Refresh**.

   The JOB_HISTORY table is listed in the Recycle Bin. You can now proceed to restore it ("undrop" it) or to purge it (delete it completely).
6. To restore the table, in the Recycle Bin, right-click JOB_HISTORY (and no other objects), and select **Flashback to Before Drop**.
7. In the Flashback to Before Drop dialog box, for **Flashback to Object Name** specify the desired name for the restored table: **JOB_HISTORY**
8. Click **Apply**.
9. To see JOB_HISTORY now listed among the tables, right-click the Tables node and select **Refresh**.

See Also:

- "About the Recycle Bin" on page 11-9
- "Viewing Recycle Bin Contents" on page 11-9

Purging the Recycle Bin

When you drop objects, space is not freed until you purge the recycle bin, or until the database automatically purges the recycle bin when it detects a low space condition. You can purge objects from the recycle bin.

Purging an Object from the Recycle Bin

To purge an object from the recycle bin:

1. In SQL Developer, connect to the database user that owns the object.
2. In the Connections navigator under that connection, expand **Recycle Bin**.
3. Right-click object that you want to purge and select **Purge**.
4. In the Purge confirmation box, click **Apply**.

See Also:

- "About the Recycle Bin" on page 11-9
- "Viewing Recycle Bin Contents" on page 11-9

Index

E

F

H

I

L

M

N

O

P

R

S

T

U

V