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.

# 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:

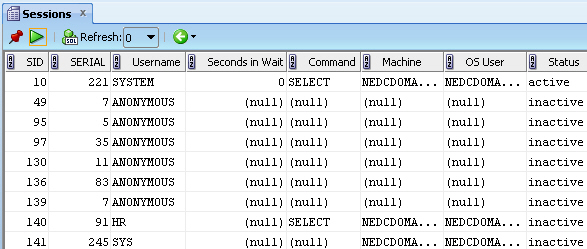
Viewing Sessions

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](https://docs.oracle.com/cd/E17781_01/server.112/e18804/monitoring.htm#BABIIJHH) shows part of the display.

***Figure 8-1 Monitor Sessions Page***

  
[Description of "Figure 8-1 Monitor Sessions Page"](https://docs.oracle.com/cd/E17781_01/server.112/e18804/img_text/monitor_sessions.htm)

See the chapter about monitoring database operations in [*Oracle Database Administrator's Guide*](http://www.oracle.com/pls/topic/lookup?ctx=xe112&id=ADMIN005) for more information.

# KILLING A SESSION

1. (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

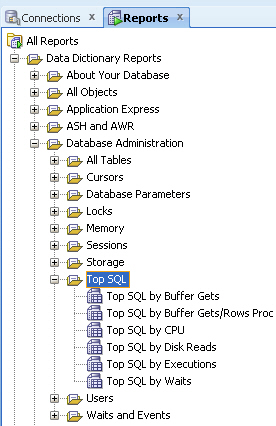
The "top" SQL statements represent the SQL statements that are executed most often, that use more system resources than other SQL statements, or that use system resources more frequently than other SQL statements. Viewing the top SQL statements reports that are available in SQL Developer enables you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

**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**.



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**

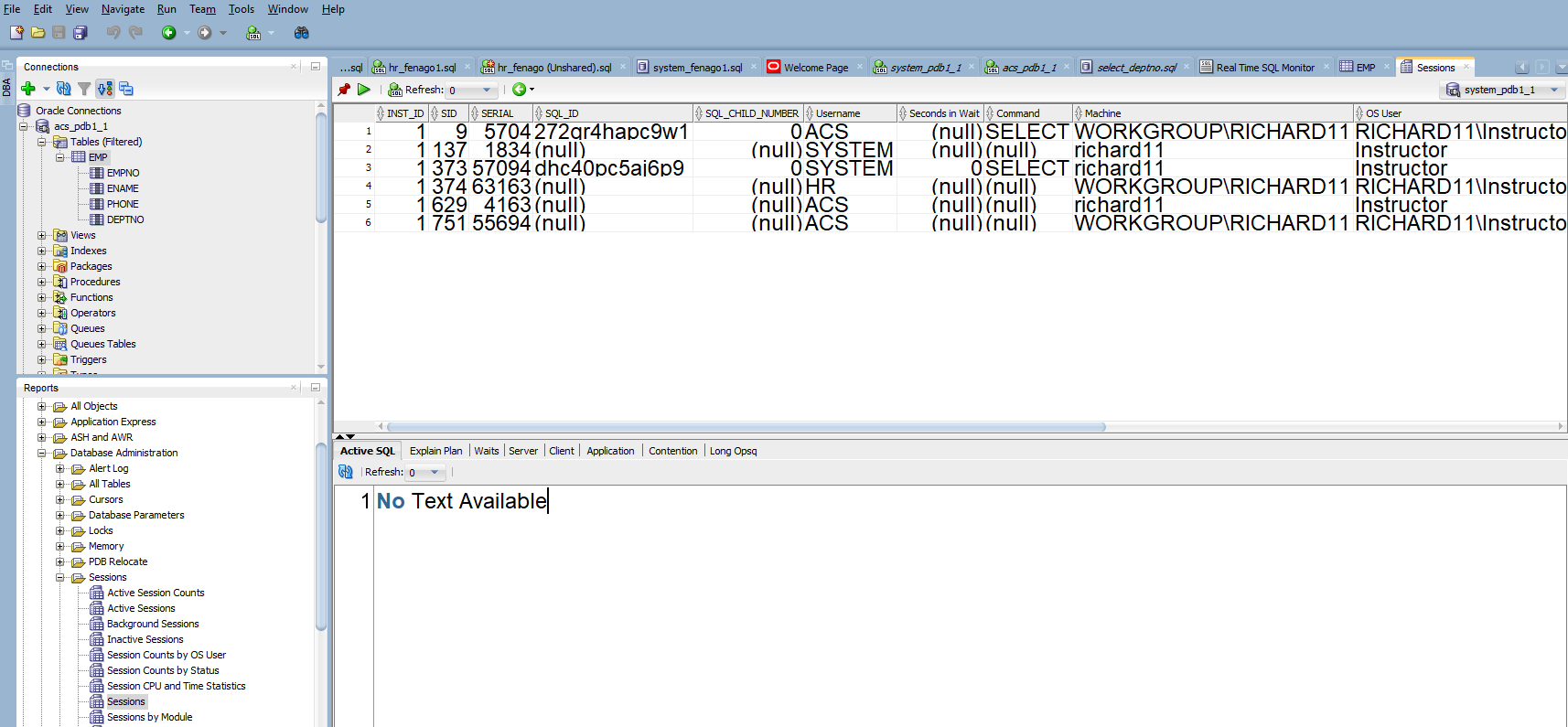
Each available report lists the top SQL statements in that category, with the following information for each statement: SQL (the SQL statement), CPU\_Seconds, Disk\_Reads, Executions, Buffer\_gets/rows\_proc, Buffer\_gets/executions, Elapsed\_Seconds, Module.

# 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.

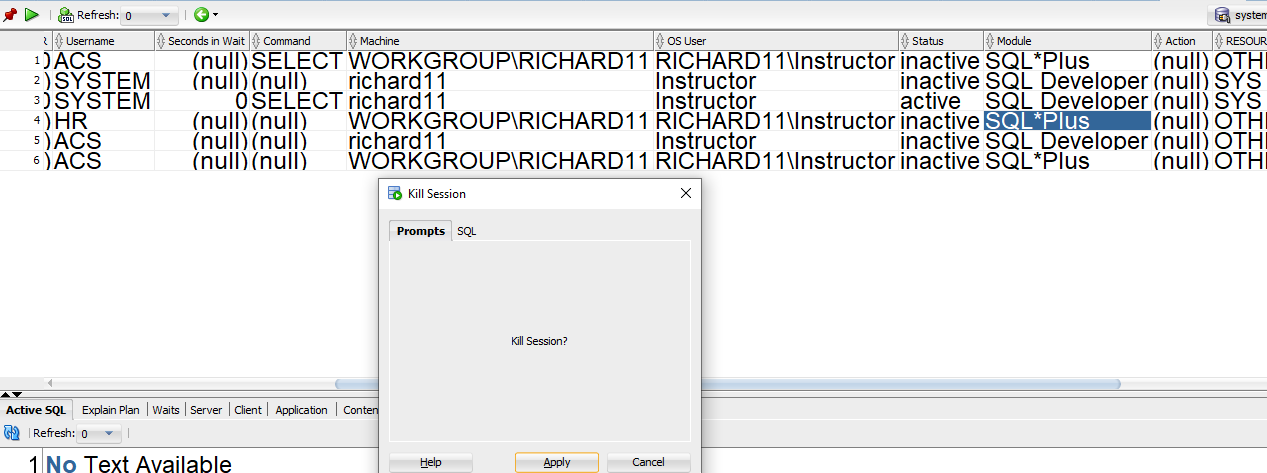
To monitor long operations:

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 **Sessions**.



If you are asked to select a connection, select one for **SYS AS SYSDBA** or system or any DBA account.

1. You can select a session and right click on it to bring up the KILL Session. You can do this for each and every session.



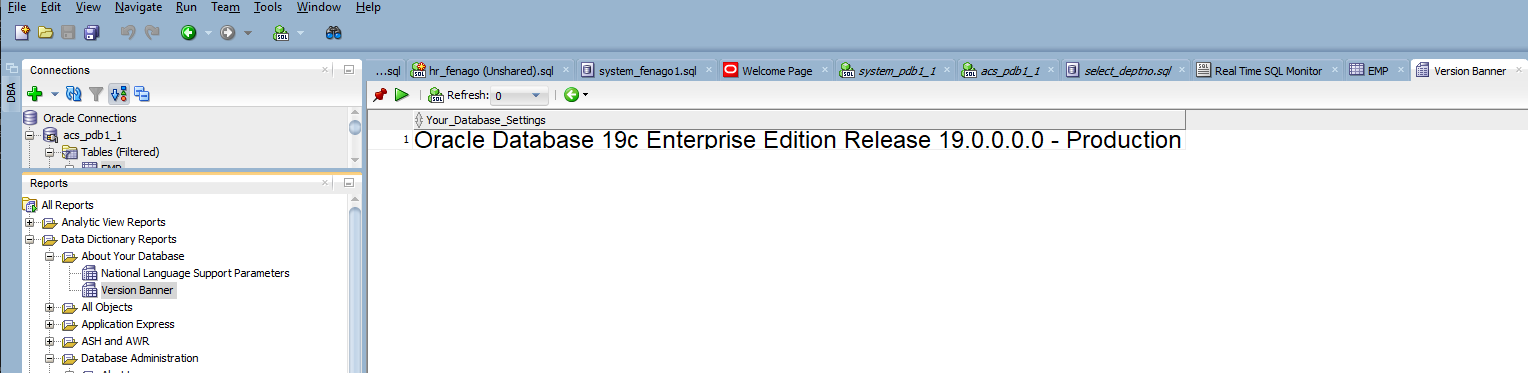
# Viewing Database Version Information

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](https://docs.oracle.com/cd/E17781_01/server.112/e18804/dbconfig.htm#CEGDFIHH).

***Figure 9-1 Version Banner Report***

  
[Description of "Figure 9-1 Version Banner Report"](https://docs.oracle.com/cd/E17781_01/server.112/e18804/img_text/version_banner.htm)

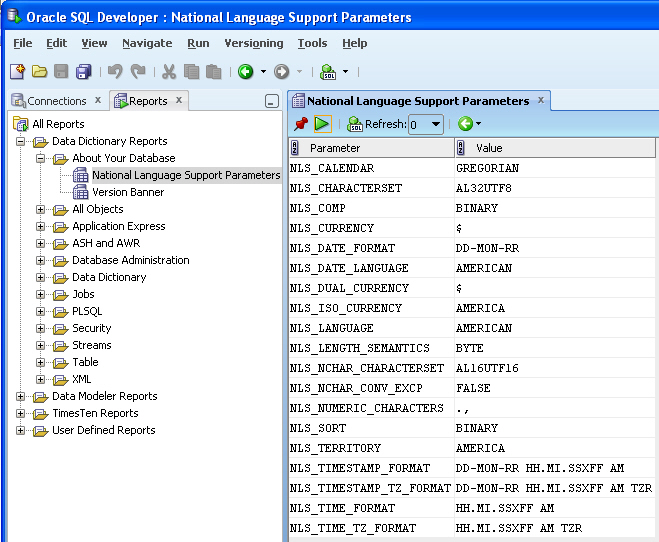
# Viewing Database Globalization Information

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](https://docs.oracle.com/cd/E17781_01/server.112/e18804/dbconfig.htm#CEGDFIHH).

***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*](https://docs.oracle.com/cd/E17781_01/appdev.112/e18147/tdddg_globalization.htm#TDDDG60000).

## 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:

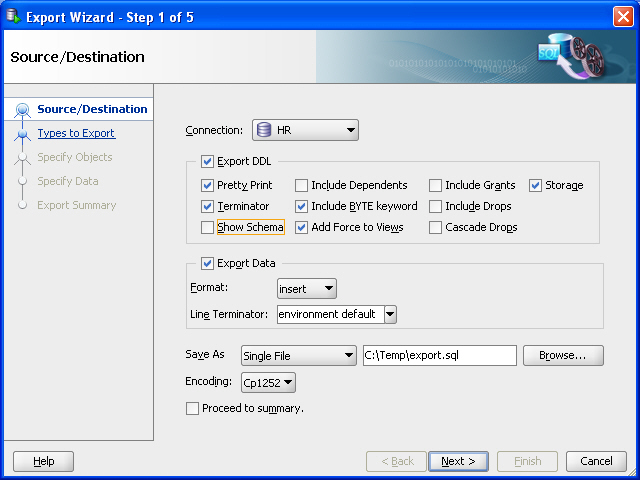
### Example: Exporting Metadata and Data for a Table

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](https://docs.oracle.com/cd/E17781_01/server.112/e18804/impexp.htm#BABBAHAJ) shows the first page of the Export Wizard, but with entries reflecting selections that you will make.

***Figure 10-1 Export Wizard: Source/Destination***

  
[Description of "Figure 10-1 Export Wizard: Source/Destination"](https://docs.oracle.com/cd/E17781_01/server.112/e18804/img_text/export_wizard_1.htm)

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

# EXPORTING DATA CONTINUED

**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.

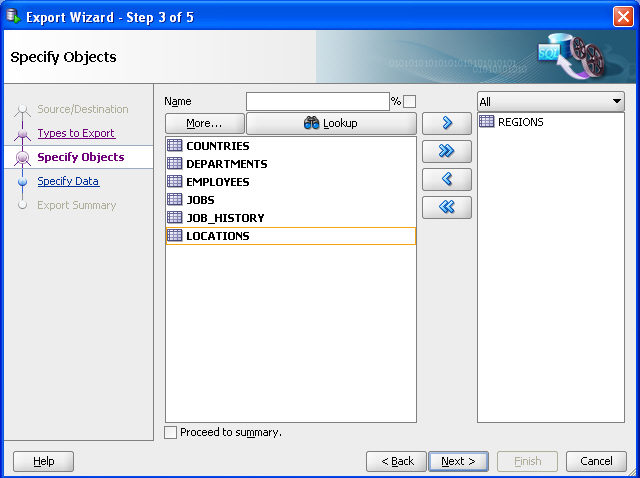
**Note:**

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.

1. Click **Next**.
2. On the Types to Export page, deselect Toggle All, then select *only* **Tables** (because you only want to export a table).
3. Click Next.
4. 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](https://docs.oracle.com/cd/E17781_01/server.112/e18804/impexp.htm#BABGBFAG) shows the result of these actions.

***Figure 10-2 Export Wizard: Specify Objects***

  
[Description of "Figure 10-2 Export Wizard: Specify Objects"](https://docs.oracle.com/cd/E17781_01/server.112/e18804/img_text/export_wizard_3.htm)

# EXPORTING DATA CONTINUED

1. Click **Next**.
2. 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.

1. 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**.)

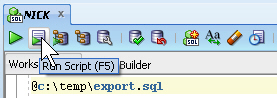
If you need to make any changes, go back to the appropriate page or pages and make them, and then move forward to the Summary page again.

### 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"](https://docs.oracle.com/cd/E17781_01/server.112/e18804/impexp.htm#BABJIFDA), 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"](https://docs.oracle.com/cd/E17781_01/server.112/e18804/users_secure.htm#CHDHIHCE). 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:
4. @c:\temp\export.sql
5. Click the Run Script icon.



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

1. In the Connections navigator, expand the Tables node under the **NICK** connection. You now see the **REGIONS** table.
2. 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

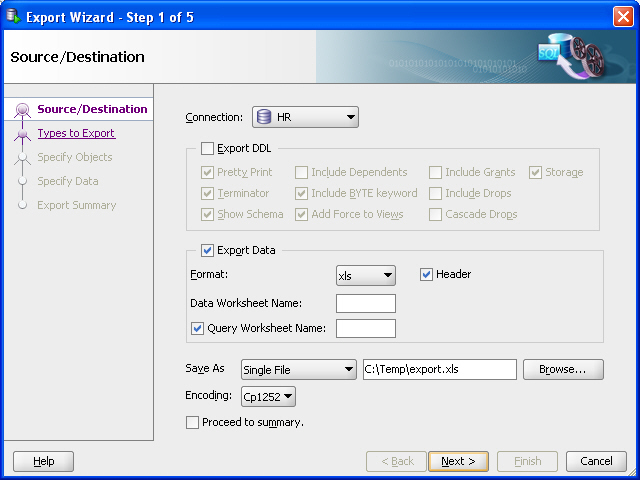
Assume that you want to export only the data from the **REGIONS** table, which is part of the **HR** sample schema, so that the data can be imported into a table with the same column definitions. This might be a **REGIONS** table in another schema (either in the same Oracle database or another Oracle 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](https://docs.oracle.com/cd/E17781_01/server.112/e18804/impexp.htm#BABJJGCI) 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***

  
[Description of "Figure 10-3 Export Wizard: Source/Destination Specifying Data Export Only"](https://docs.oracle.com/cd/E17781_01/server.112/e18804/img_text/export_wizard_1_data.htm)

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

# EXPORTING TO AN EXCEL SPREADSHEET CONTINUED

**Connection**: Select **HR**.

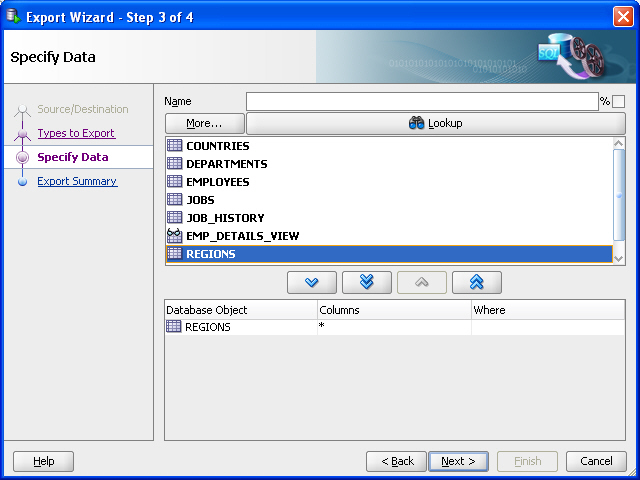
Export DDL: ***Deselect*** (uncheck) this option. If a .sql script file is generated (which will not happen in this example), it will not contain any **CREATE** statements, but only **INSERT** statements.

**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**.)

1. Click **Next**.
2. On the Types to Export page, deselect Toggle All, then select *only* **Tables** (because you only want to export data for a table).
3. Click Next.
4. 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](https://docs.oracle.com/cd/E17781_01/server.112/e18804/impexp.htm#BABGBFAG) shows the result of these actions.

***Figure 10-4 Export Wizard: Specify Objects for Exporting Data***

  
[Description of "Figure 10-4 Export Wizard: Specify Objects for Exporting Data"](https://docs.oracle.com/cd/E17781_01/server.112/e18804/img_text/export_wizard_3_data.htm)

# CONTINUED

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.

1. Click **Next**.
2. 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.

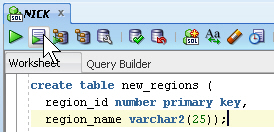
### Example: Importing Data from a Microsoft Excel File

Assume that you wanted to import the data that was exported in ["Example: Exporting Data to a Microsoft Excel File"](https://docs.oracle.com/cd/E17781_01/server.112/e18804/impexp.htm#BABHFHGH), 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"](https://docs.oracle.com/cd/E17781_01/server.112/e18804/users_secure.htm#CHDHIHCE). 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:
4. create table new\_regions ( region\_id number primary key, region\_name varchar2(25));
5. Click the Run Script icon.



The Script Output pane shows that the **NEW\_REGIONS** table has been created.

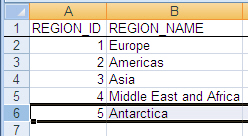
1. In the Connections navigator, expand the Tables node under the **NICK** connection. You now see the **NEW\_REGIONS** table.

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.

1. 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](https://docs.oracle.com/cd/E17781_01/server.112/e18804/impexp.htm#BABCCICH) shows the original file with one row added for the Antarctica region.

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

  
[Description of "Figure 10-5 Microsoft Excel File with Exported Data (Modified)"](https://docs.oracle.com/cd/E17781_01/server.112/e18804/img_text/exported_xls_modified.htm)

1. Save and close the Microsoft Excel .xls file.
2. In SQL Developer, in the Connections navigator display for **NICK**, right-lick the **NEW\_REGIONS** table and select **Import Data**.
3. In the dialog box that is displayed, navigate to the **c:\temp** folder, select **export.xls**, and click **Open**.
4. 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](https://docs.oracle.com/cd/E17781_01/server.112/e18804/impexp.htm#BCECBCEC)
* [Loading Data with SQL\*Loader](https://docs.oracle.com/cd/E17781_01/server.112/e18804/impexp.htm#BABCJCBD)
* [Exporting and Importing Data](https://docs.oracle.com/cd/E17781_01/server.112/e18804/impexp.htm#BCEIHIHE)

### Choosing the Right Export/Import Utility

Oracle Database XE provides a number of powerful utilities for exporting and importing data. [Table 10-1](https://docs.oracle.com/cd/E17781_01/server.112/e18804/impexp.htm#BCEBGBEH) 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](https://docs.oracle.com/cd/E17781_01/server.112/e18804/impexp.htm#BCEIBADF) 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**) |

### 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*](http://www.oracle.com/pls/topic/lookup?ctx=xe112&id=SUTIL009) 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*](http://www.oracle.com/pls/topic/lookup?ctx=xe112&id=ADMIN01507) 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:
2. 100,"Susan, Susie",Kochhar,17-JUN-1997,daughter,101,NULL,
3. 102,David,Kochhar,02-APR-1999,son,101,NULL,
4. 104,Jill,Colmenares,10-FEB-1992,daughter,119,NULL,
5. 106,"Victoria, Vicki",Chen,17-JUN-1997,daughter,110,NULL,
6. 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.

1. 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:
2. LOAD DATA
3. INFILE dependents.dat
4. INTO TABLE dependents
5. REPLACE
6. FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"'
7. (
8. dep\_id,
9. first\_name,
10. last\_name,
11. birthdate,
12. relation,
13. relative\_id,
14. benefits
15. )
16. 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.
17. On Linux, ensure that environment variables are set according to the instructions in ["Setting Environment Variables on the Linux Platform"](https://docs.oracle.com/cd/E17781_01/server.112/e18804/connecting.htm#CHDJBIAH).
18. Start the SQL Command Line (SQL\*Plus) and connect as user **hr** by entering the following at the command prompt:
19. sqlplus hr/hr
20. At the SQL prompt, create the **dependents** table, as follows:
21. CREATE TABLE dependents (
22. dep\_id NUMBER(6),
23. first\_name VARCHAR2(20),
24. last\_name VARCHAR2(25) CONSTRAINT dep\_last\_name\_nn NOT NULL,
25. birthdate DATE,
26. relation VARCHAR2(25),
27. relative\_id NUMBER(6) CONSTRAINT emp\_dep\_rel\_id\_fk REFERENCES employees
28. (employee\_id),
29. benefits CLOB
30. )
31. /

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.

1. From within your current working directory (where you created the control and data files), issue the following SQL\*Loader command at the system prompt:
2. 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.