7

Working in a Global Environment

Globalization support features enable multilingual applications to run simultaneously from anywhere in the world. Applications can render the content of the user interface, and process data, using the native language and locale preferences of the user.

About Globalization Support Features

Globalization support features enable you to develop multilingual applications that can be run simultaneously from anywhere in the world. An application can render the content of the user interface, and process data, using the native language and locale preferences of the user.

Note:

In the past, Oracle called globalization support **National Language Support (NLS)**, but NLS is actually a subset of globalization support. NLS is the ability to choose a national language and store data using a specific character set. NLS is implemented with NLS parameters.

See Also:

Oracle Database Globalization Support Guide for more information about globalization support features

About Language Support

Oracle Database enables you to store, process, and retrieve data in native languages. The languages that can be stored in a database are all languages written in scripts that are encoded by Oracle-supported character sets. Through the use of Unicode databases and data types, Oracle Database supports most contemporary languages.

Additional support is available for a subset of the languages. The database can, for example, display dates using translated month names, and can sort text data according to cultural conventions.

In this document, the term **language support** refers to the additional language-dependent functionality, and not to the ability to store text of a specific language. For example, language support includes displaying dates or sorting text according to specific locales and cultural conventions. Additionally, for some supported languages, Oracle Database provides translated server messages and a translated user interface for the database utilities.

See Also:

- "About the NLS_LANGUAGE Parameter"
- *Oracle Database Globalization Support Guide* for a complete list of languages that Oracle Database supports
- *Oracle Database Globalization Support Guide* for a list of languages into which Oracle Database messages are translated

About Territory Support

The default local time format, date format, and numeric and monetary conventions depend on the local territory setting.

Oracle Database supports cultural conventions that are specific to geographical locations. The default local time format, date format, and numeric and monetary conventions depend on the local territory setting. Setting different NLS parameters enables the database session to use different cultural settings. For example, you can set the euro (`EUR`) as the primary currency and the Japanese yen (`JPY`) as the secondary currency for a given database session, even when the territory is `AMERICA`.

See Also:

- "About the NLS_TERRITORY Parameter"
- *Oracle Database Globalization Support Guide* for a complete list of territories that Oracle Database supports

About Date and Time Formats

Different countries have different conventions for displaying the hour, day, month, and year.

For example, this table shows the local date and time format for five countries and gives an example of each format:

Country	Date Format	Example	Time Format	Example
China	`yyyy-mm-dd`	`2005-02-28`	`hh24:mi:ss`	`13:50:23`
Estonia	`dd.mm.yyyy`	`28.02.2005`	`hh24:mi:ss`	`13:50:23`
Germany	`dd.mm.rr`	`28.02.05`	`hh24:mi:ss`	`13:50:23`
UK	`dd/mm/yyyy`	`28/02/2005`	`hh24:mi:ss`	`13:50:23`
US	`mm/dd/yyyy`	`02/28/2005`	`hh:mi:ssxff am`	`1:50:23.555 PM`

See Also:

- "About the NLS_DATE_FORMAT Parameter"
- "About the NLS_DATE_LANGUAGE Parameter"
- "About NLS_TIMESTAMP_FORMAT and NLS_TIMESTAMP_TZ_FORMAT Parameters"
- *Oracle Database Globalization Support Guide* for information about date/time data types and time zone support
- *Oracle Database SQL Language Reference* for information about date and time formats

About Calendar Formats

Different countries use different calendars.

Oracle Database stores this calendar information for each territory:

- **First day of the week**

  Sunday in some cultures, Monday in others. Set by the NLS_TERRITORY parameter.
- **First week of the calendar year**

  Some countries use week numbers for scheduling, planning, and bookkeeping. In the ISO standard, this week number can differ from the week number of the calendar year. For example, 1st Jan 2005 is in ISO week number 53 of 2004. An ISO week starts on Monday and ends on Sunday. To support the ISO standard, Oracle Database provides the IW date format element, which returns the ISO week number. The first calendar week of the year is set by the NLS_TERRITORY parameter.
- **Number of days and months in a year**

  Oracle Database supports six calendar systems in addition to the Gregorian calendar, which is the default. These additional calendar systems are:

  - Japanese Imperial

    Has the same number of months and days as the Gregorian calendar, but the year starts with the beginning of each Imperial Era.
  - ROC Official

    Has the same number of months and days as the Gregorian calendar, but the year starts with the founding of the Republic of China.
  - Persian

    The first six months have 31 days each, the next five months have 30 days each, and the last month has either 29 days or (in leap year) 30 days.
  - Thai Buddha uses a Buddhist calendar.
  - Arabic Hijrah has 12 months and 354 or 355 days.
  - English Hijrah has 12 months and 354 or 355 days.

  The calendar system is specified by the NLS_CALENDAR parameter.

- **First year of era**

  The Islamic calendar starts from the year of the Hegira. The Japanese Imperial calendar starts from the beginning of an Emperor's reign (for example, 1998 is the tenth year of the Heisei era).

See Also:

- "About the NLS_TERRITORY Parameter"
- "About the NLS_CALENDAR Parameter"
- *Oracle Database Globalization Support Guide* for information about calendar formats

About Numeric and Monetary Formats

Different countries have different numeric and monetary conventions.

This table shows the local numeric and monetary format for five countries and gives an example of each format:

Country	Numeric Format	Monetary Format
China	1,234,567.89	©1,234.56
Estonia	1 234 567,89	1 234,56 kr
Germany	1.234.567,89	1.234,56€
UK	1,234,567.89	£1,234.56
US	1,234,567.89	$1,234.56

See Also:

- "About the NLS_NUMERIC_CHARACTERS Parameter"
- "About the NLS_CURRENCY Parameter"
- "About the NLS_ISO_CURRENCY Parameter"
- "About the NLS_DUAL_CURRENCY Parameter"
- *Oracle Database Globalization Support Guide* for information about numeric and list parameters
- *Oracle Database Globalization Support Guide* for information about monetary parameters
- *Oracle Database SQL Language Reference* for information about number format models

About Linguistic Sorting and String Searching

Different languages have different sort orders (collating sequences). Also, different countries or cultures that use the same alphabets sort words differently. For example, in Danish, Æ is after Z, and Y and Ü are considered to be variants of the same letter.

See Also:

- "About the NLS_SORT Parameter"
- "About the NLS_COMP Parameter"
- *Oracle Database Globalization Support Guide* for more information about linguistic sorting and string searching

About Length Semantics

To calculate the number of characters in a string, using byte length, you must know the number of bytes in each character in the character set.

In single-byte character sets, the number of bytes and the number of characters in a string are the same. In multibyte character sets, a character or code point consists of one or more bytes. Calculating the number of characters based on byte length can be difficult in a variable-width character set. Calculating column length in bytes is called **byte semantics**, while measuring column length in characters is called **character semantics**.

Character semantics is useful for specifying the storage requirements for multibyte strings of varying widths. For example, in a Unicode database (AL32UTF8), suppose that you must have a VARCHAR2 column that can store up to five Chinese characters with five English characters. Using byte semantics, this column requires 15 bytes for the Chinese characters, which are 3 bytes long, and 5 bytes for the English characters, which are 1 byte long, for a total of 20 bytes. Using character semantics, the column requires 10 characters.

See Also:

- "About the NLS_LENGTH_SEMANTICS Parameter"
- *Oracle Database Globalization Support Guide* for information about character sets and length semantics

About Unicode and SQL National Character Data Types

Unicode is a character encoding system that defines every character in most of the spoken languages in the world. In Unicode, every character has a unique code, regardless of the platform, program, or language.

You can store Unicode characters in an Oracle Database in two ways:

- You can create a Unicode database that enables you to store UTF-8 encoded characters as SQL character data types (CHAR, VARCHAR2, CLOB, and LONG).

- You can declare columns and variables that have SQL national character data types.

The **SQL national character data types** are NCHAR, NVARCHAR2, and NCLOB. They are also called **Unicode data types**, because they are used only for storing Unicode data.

The national character set, which is used for all SQL national character data types, is specified when the database is created. The national character set can be either UTF8 or AL16UTF16 (default).

When you declare a column or variable of the type NCHAR or NVARCHAR2, the length that you specify is the number of characters, not the number of bytes.

See Also:

- *Oracle Database Globalization Support Guide* for more information about Unicode
- *Oracle Database Globalization Support Guide* for more information about storing Unicode characters in an Oracle Database
- *Oracle Database Globalization Support Guide* for more information about SQL national character data types

About Initial NLS Parameter Values

Except in SQL Developer, the initial values of NLS parameters are set by database initialization parameters.

The DBA can set the values of initialization parameters in the initialization parameter file, and they take effect the next time the database is started.

In SQL Developer, the initial values of NLS parameters are as shown in Table 7-1.

Table 7-1 Initial Values of NLS Parameters in SQL Developer

Parameter	Initial Value
NLS_CALENDAR	`GREGORIAN`
NLS_CHARACTERSET	`AL32UTF8`
NLS_COMP	`BINARY`
NLS_CURRENCY	`$`
NLS_DATE_FORMAT	`DD-MON-RR`
NLS_DATE_LANGUAGE	`AMERICAN`
NLS_DUAL_CURRENCY	`$`
NLS_ISO_CURRENCY	`AMERICA`
NLS_LANGUAGE	`AMERICAN`
NLS_LENGTH_SEMANTICS	`BYTE`

Table 7-1 (Cont.) Initial Values of NLS Parameters in SQL Developer

Parameter	Initial Value
NLS_NCHAR_CHARACTER SET	`AL16UTF16`
NLS_NCHAR_CONV_EXCP	`FALSE`
NLS_NUMERIC_CHARACT ERS	`.,`
NLS_SORT	`BINARY`
NLS_TERRITORY	`AMERICA`
NLS_TIMESTAMP_FORMAT	`DD-MON-RR HH.MI.SSXFF AM`
NLS_TIMESTAMP_TZ_FOR MAT	`DD-MON-RR HH.MI.SSXFF AM TZR`
NLS_TIME_FORMAT	`HH.MI.SSXFF AM`
NLS_TIME_TZ_FORMAT	`HH.MI.SSXFF AM TZR`

See Also:

Oracle Database Administrator's Guide for information about initialization parameters and initialization parameter files

Viewing NLS Parameter Values

To view the current values of NLS parameters, use the SQL Developer report National Language Support Parameters.

To view the National Language Support Parameters report:

1. From the SQL Developer menu View, select **Reports**.
2. In the Reports pane, expand **Data Dictionary Reports**.
3. In the list of reports, expand **About Your Database**.
4. In the list of reports, select **National Language Support Parameters**.
5. In the Select Connection window, select **hr_conn**.
6. Click **OK**.

   The Select Connection window closes and the National Language Support Parameters pane appears, showing the names of the NLS parameters and their current values.

See Also:

Oracle SQL Developer User's Guide for more information about SQL Developer reports

Changing NLS Parameter Values

You can change the value of one or more NLS parameters in any of these ways.

- Change the values for all SQL Developer connections, current and future.
- On the client, change the settings of the corresponding NLS environment variables.

  Only on the client, the new values of the NLS environment variables override the values of the corresponding NLS parameters.

  You can use environment variables to specify locale-dependent behavior for the client. For example, on a Linux system, this statement sets the value of the NLS_SORT environment variable to `FRENCH`, overriding the value of the NLS_SORT parameter:

  ```
  % setenv NLS_SORT FRENCH
  ```

  > **Note:**
  >
  > Environment variables might be platform-dependent.

- Change the values only for the current session, using an ALTER SESSION statement with this syntax:

  ```
  ALTER SESSION SET parameter_name=parameter_value
    [ parameter_name=parameter_value ]... ;
  ```

  Only in the current session, the new values override those set in all of the preceding ways.

  You can use the ALTER SESSION to test your application with the settings for different locales.

- Change the values only for the current SQL function invocation.

  Only for the current SQL function invocation, the new values override those set in all of the preceding ways.

> **See Also:**
>
> - *Oracle Database SQL Language Reference* for more information about the ALTER SESSION statement
> - *Oracle Database Globalization Support Guide* for more information about setting NLS parameters

Changing NLS Parameter Values for All SQL Developer Connections

You can change the values of NLS parameters for all SQL Developer connections, current and future.

To change National Language Support Parameter values:

1. From the SQL Developer menu Tools, select **Preferences**.
2. In the Preferences window, in the left frame, expand **Database**.
3. In the list of database preferences, click **NLS**.

   A list of NLS parameters and their current values appears. The value fields are menus.
4. From the menu to the right of each parameter whose value you want to change, select the desired value.
5. Click **OK**.

   The NLS parameters now have the values that you specified. To verify these values, see "Viewing NLS Parameter Values".

> **Note:**
>
> If the NLS parameter values do not reflect your changes, click the icon **Run Report**.

> **See Also:**
>
> *Oracle SQL Developer User's Guide* for more information about SQL Developer preferences

Changing NLS Parameter Values for the Current SQL Function Invocation

SQL functions whose behavior depends on the values of NLS parameters are called **locale-dependent**. Some locale-dependent SQL functions have optional NLS parameters.

The local-dependent functions that have optional NLS parameters are:

- `TO_CHAR`
- `TO_DATE`
- `TO_NUMBER`
- `NLS_UPPER`
- `NLS_LOWER`
- `NLS_INITCAP`
- `NLSSORT`

In all of the preceding functions, you can specify these NLS parameters:

- `NLS_DATE_LANGUAGE`
- `NLS_DATE_LANGUAGE`
- `NLS_NUMERIC_CHARACTERS`
- `NLS_CURRENCY`

- `NLS_ISO_CURRENCY`
- `NLS_DUAL_CURRENCY`
- `NLS_CALENDAR`
- `NLS_SORT`

In the `NLSSORT` function, you can also specify these NLS parameters:

- `NLS_LANGUAGE`
- `NLS_TERRITORY`
- `NLS_DATE_FORMAT`

To specify NLS parameters in a function, use this syntax:

```
'parameter=value' ['parameter=value']...
```

Suppose that you want NLS_DATE_LANGUAGE to be `AMERICAN` when this query is evaluated:

```
SELECT last_name FROM employees WHERE hire_date > '01-JAN-1999';
```

You can set NLS_DATE_LANGUAGE to `AMERICAN` before running the query:

```
ALTER SESSION SET NLS_DATE_LANGUAGE=American;
SELECT last_name FROM employees WHERE hire_date > '01-JAN-1999';
```

Alternatively, you can set NLS_DATE_LANGUAGE to `AMERICAN` inside the query, using the locale-dependent SQL function TO_DATE with its optional NLS_DATE_LANGUAGE parameter:

```
SELECT last_name FROM employees
WHERE hire_date > TO_DATE('01-JAN-1999', 'DD-MON-YYYY',
                          'NLS_DATE_LANGUAGE=AMERICAN');
```

Tip:

Using session default values for NLS parameters in SQL functions usually results in better performance. Therefore, specify optional NLS parameters in locale-dependent SQL functions only in SQL statements that must not use the default NLS parameter values.

See Also:

Oracle Database Globalization Support Guide for more information about locale-dependent SQL functions with optional NLS parameters

About Individual NLS Parameters

Many individual NLS parameters are available.

See Also:

- *Oracle Database Globalization Support Guide* for more information about setting up a globalization support environment
- "Changing NLS Parameter Values"

About Locale and the NLS_LANG Parameter

A **locale** is a linguistic and cultural environment in which a system or application runs. The simplest way to specify a locale for Oracle Database software is to set the NLS_LANG parameter.

The NLS_LANG parameter sets the default values of the parameters NLS_LANGUAGE and NLS_TERRITORY for both the server session (for example, SQL statement processing) and the client application (for example, display formatting in Oracle Database tools). The NLS_LANG parameter also sets the character set that the client application uses for data entered or displayed.

The default value of NLS_LANG is set during database installation. You can use the ALTER SESSION statement to change the values of NLS parameters, including those set by NLS_LANG, for your session. However, only the client can change the NLS settings in the client environment.

See Also:

- *Oracle Database Globalization Support Guide* for more information about specifying a locale with the `NLS_LANG` parameter
- *Oracle Database Globalization Support Guide* for information about languages, territories, character sets, and other locale data supported by Oracle Database
- "About the NLS_LANGUAGE Parameter"
- "About the NLS_TERRITORY Parameter"
- "Changing NLS Parameter Values"

About the NLS_LANGUAGE Parameter

This parameter specifies the default language of the database.

Specifies: Default language of the database. Default conventions for:

- Language for server messages
- Language for names and abbreviations of days and months that are specified in the SQL functions TO_CHAR and TO_DATE
- Symbols for default-language equivalents of AM, PM, AD, and BC
- Default sorting order for character data when the ORDER BY clause is specified
- Writing direction

- Affirmative and negative response strings (for example, YES and NO)

Acceptable Values: Any language name that Oracle supports. For a list, see *Oracle Database Globalization Support Guide*.

Default Value: Set by NLS_LANG, described in "About Locale and the NLS_LANG Parameter".

Sets default values of:

- NLS_DATE_LANGUAGE, described in "About the NLS_DATE_LANGUAGE Parameter".
- NLS_SORT, described in "About the NLS_SORT Parameter".

Example 7-1 shows how setting NLS_LANGUAGE to `ITALIAN` and `GERMAN` affects server messages and month abbreviations.

To try this example in SQL Developer, enter the statements and queries in the Worksheet. For information about the Worksheet, see "Running Queries in SQL Developer". The results shown here are from SQL*Plus; their format is slightly different in SQL Developer.

Example 7-1 NLS_LANGUAGE Affects Server Message and Month Abbreviations

1. Note the current value of NLS_LANGUAGE.
2. If the value in step 1 is not `ITALIAN`, change it:

   ```
   ALTER SESSION SET NLS_LANGUAGE=ITALIAN;
   ```

3. Query a nonexistent table:

   ```
   SELECT * FROM nonexistent_table;
   ```

   Result:

   ```
   SELECT * FROM nonexistent_table
                 *
   ERROR at line 1:
   ORA-00942: tabella o vista inesistente
   ```

4. Run this query:

   ```
   SELECT LAST_NAME, HIRE_DATE
   FROM EMPLOYEES
   WHERE EMPLOYEE_ID IN (111, 112, 113);
   ```

   Result:

   ```
   LAST_NAME                 HIRE_DATE
   ------------------------- ---------
   Sciarra                   30-SET-97
   Urman                     07-MAR-98
   Popp                      07-DIC-99

   3 rows selected.
   ```

5. Change the value of NLS_LANGUAGE to `GERMAN`:

   ```
   ALTER SESSION SET NLS_LANGUAGE=GERMAN;
   ```

6. Repeat the query from step 3.

   Result:

```
SELECT * FROM nonexistent_table
              *
ERROR at line 1:
ORA-00942: Tabelle oder View nicht vorhanden
```

7. Repeat the query from step 4.

   Result:

```
LAST_NAME                 HIRE_DATE
------------------------- ---------
Sciarra                   30-SEP-97
Urman                     07-MRZ-98
Popp                      07-DEZ-99

3 rows selected.
```

8. Set NLS_LANGUAGE to the value that it had at step 1.

See Also:

- *Oracle Database Globalization Support Guide* for more information about the NLS_LANGUAGE parameter
- "About Language Support"
- "Changing NLS Parameter Values"

About the NLS_TERRITORY Parameter

This parameter specifies default conventions for date format, time stamp format, decimal and group separator, local currency symbol, ISO currency symbol, and dual currency symbol.

Specifies: Default conventions for:

- Date format
- Time stamp format
- Decimal character and group separator
- Local currency symbol
- ISO currency symbol
- Dual currency symbol

Acceptable Values: Any territory name that Oracle supports. For a list, see *Oracle Database Globalization Support Guide*.

Default Value: Set by NLS_LANG, described in "About Locale and the NLS_LANG Parameter".

Sets default values of:

- NLS_DATE_FORMAT, described in "About the NLS_DATE_FORMAT Parameter".
- NLS_TIMESTAMP_FORMAT and NLS_TIMESTAMP_TZ_FORMAT, described in "About NLS_TIMESTAMP_FORMAT and NLS_TIMESTAMP_TZ_FORMAT Parameters".

- NLS_NUMERIC_CHARACTERS, described in "About the NLS_NUMERIC_CHARACTERS Parameter".
- NLS_CURRENCY, described in "About the NLS_CURRENCY Parameter".
- NLS_ISO_CURRENCY, described in "About the NLS_ISO_CURRENCY Parameter".
- NLS_DUAL_CURRENCY, described in "About the NLS_DUAL_CURRENCY Parameter".

Example 7-2 shows how setting NLS_TERRITORY to JAPAN and AMERICA affects the currency symbol.

To try this example in SQL Developer, enter the statements and queries in the Worksheet. For information about the Worksheet, see "Running Queries in SQL Developer". The results shown here are from SQL*Plus; their format is slightly different in SQL Developer.

Example 7-2 NLS_TERRITORY Affects Currency Symbol

1. Note the current value of NLS_TERRITORY.
2. If the value in step 1 is not JAPAN, change it:

   ```
   ALTER SESSION SET NLS_TERRITORY=JAPAN;
   ```
3. Run this query:

   ```
   SELECT TO_CHAR(SALARY,'L99G999D99') SALARY
   FROM EMPLOYEES
   WHERE EMPLOYEE_ID IN (100, 101, 102);
   ```

   Result:

   ```
   SALARY
   --------------------
             ©24,000.00
             ©17,000.00
             ©17,000.00

   3 rows selected.
   ```
4. Change the value of NLS_TERRITORY to AMERICA:

   ```
   ALTER SESSION SET NLS_TERRITORY=AMERICA;
   ```
5. Repeat the query from step 3.

   Result:

   ```
   SALARY
   --------------------
             $24,000.00
             $17,000.00
             $17,000.00

   3 rows selected.
   ```
6. Set NLS_TERRITORY to the value that it had at step 1.

See Also:

- *Oracle Database Globalization Support Guide* for more information about the NLS_TERRITORY parameter
- "About Territory Support"
- "Changing NLS Parameter Values"

About the NLS_DATE_FORMAT Parameter

This parameter specifies the default date format to use with the TO_CHAR and TO_DATE functions.

Specifies: Default date format to use with the TO_CHAR and TO_DATE functions (which are introduced in "Using Conversion Functions in Queries").

Acceptable Values: Any any valid datetime format model. For example:

```
NLS_DATE_FORMAT='MM/DD/YYYY'
```

For information about datetime format models, see *Oracle Database SQL Language Reference*.

Default Value: Set by NLS_TERRITORY, described in "About the NLS_TERRITORY Parameter".

The default date format might not correspond to the convention used in a given territory. To get dates in localized formats, you can use the 'DS' (short date) and 'DL' (long date) formats.

Example 7-3 shows how setting NLS_TERRITORY to `AMERICA` and `FRANCE` affects the default, short, and long date formats.

Example 7-4 changes the value of NLS_DATE_FORMAT, overriding the default value set by NLS_TERRITORY.

To try the examples in SQL Developer, enter the statements and queries in the Worksheet. For information about the Worksheet, see "Running Queries in SQL Developer". The results shown here are from SQL*Plus; their format is slightly different in SQL Developer.

Example 7-3   NLS_TERRITORY Affects Date Formats

1. Note the current value of NLS_TERRITORY.
2. If the value in step 1 is not `AMERICA`, change it:

   ```
   ALTER SESSION SET NLS_TERRITORY=AMERICA;
   ```

3. Run this query:

   ```
   SELECT hire_date "Default",
          TO_CHAR(hire_date,'DS') "Short",
          TO_CHAR(hire_date,'DL') "Long"
   FROM employees
   WHERE employee_id IN (111, 112, 113);
   ```

   Result:

```
Default    Short      Long
--------- ---------- -----------------------------
30-SEP-05 9/30/2005  Friday, September 30, 2005
07-MAR-98 3/7/2006   Tuesday, March 07, 2006
07-DEC-99 12/7/2007  Friday, December 07, 2007

3 rows selected.
```

4. Change the value of NLS_TERRITORY to `FRANCE`:

   ```
   ALTER SESSION SET NLS_TERRITORY=FRANCE;
   ```

5. Repeat the query from step 3.

   Result:

   ```
   Default  Short      Long
   -------- ---------- ---------------------------
   30/09/05 30/09/2005 friday 30 september 2005
   07/03/06 07/03/2006 tuesday 7 march 2006
   07/12/07 07/12/2007 friday 7 december 2007

   3 rows selected.
   ```

   (To get the names of the days and months in French, you must set either NLS_LANGUAGE or NLS_DATE_LANGUAGE to `FRENCH` before running the query.)

6. Set NLS_TERRITORY to the value that it had at step 1.

Example 7-4 NLS_DATE_FORMAT Overrides NLS_TERRITORY

1. Note the current values of NLS_TERRITORY and NLS_DATE_FORMAT.
2. If the value of NLS_TERRITORY in step 1 is not `AMERICA`, change it:

   ```
   ALTER SESSION SET NLS_TERRITORY=AMERICA;
   ```

3. If the value of NLS_DATE_FORMAT in step 1 is not `'Day Month ddth'`, change it:

   ```
   ALTER SESSION SET NLS_DATE_FORMAT='Day Month ddth';
   ```

4. Run this query (from previous example, step 3):

   ```
   SELECT hire_date "Default",
          TO_CHAR(hire_date,'DS') "Short",
          TO_CHAR(hire_date,'DL') "Long"
   FROM employees
   WHERE employee_id IN (111, 112, 113);
   ```

   Result:

   ```
   Default                  Short      Long
   ------------------------ ---------- -----------------------------
   Friday    September 30th 9/30/2005  Tuesday, September 30, 2005
   Tuesday   March     07th 3/7/2006   Saturday, March 07, 2006
   Friday    December  07th 12/7/2007  Tuesday, December 07, 2007

   3 rows selected.
   ```

5. Set NLS_TERRITORY and NLS_DATE_FORMAT to the values that they had at step 1.

See Also:

- *Oracle Database Globalization Support Guide* for more information about the NLS_DATE_FORMAT parameter
- *Oracle Database SQL Language Reference* for more information about the TO_CHAR function
- *Oracle Database SQL Language Reference* for more information about the TO_DATE function
- "About Date and Time Formats"
- "Changing NLS Parameter Values"

About the NLS_DATE_LANGUAGE Parameter

This parameter specifies the language for names and abbreviations of days and months that are produced by: SQL functions TO_CHAR and TO_DATE, the default date format (set by NLS_DATE_FORMAT), and symbols for the default-language equivalents of AM, PM, AD, and BC.

Specifies: Language for names and abbreviations of days and months that are produced by:

- SQL functions TO_CHAR and TO_DATE (which are introduced in "Using Conversion Functions in Queries")
- Default date format (set by NLS_DATE_FORMAT, described in "About the NLS_DATE_FORMAT Parameter")
- Symbols for default-language equivalents of AM, PM, AD, and BC

Acceptable Values: Any language name that Oracle supports. For a list, see *Oracle Database Globalization Support Guide*.

Default Value: Set by NLS_LANGUAGE, described in "About the NLS_LANGUAGE Parameter".

Example 7-5 shows how setting NLS_DATE_LANGUAGE to `FRENCH` and `SWEDISH` affects the displayed system date.

To try this example in SQL Developer, enter the statements and queries in the Worksheet. For information about the Worksheet, see "Running Queries in SQL Developer". The results shown here are from SQL*Plus; their format is slightly different in SQL Developer.

Example 7-5 NLS_DATE_LANGUAGE Affects Displayed SYSDATE

1. Note the current value of NLS_DATE_LANGUAGE.
2. If the value of NLS_DATE_LANGUAGE in step 1 is not `FRENCH`, change it:

   ```
   ALTER SESSION SET NLS_DATE_LANGUAGE=FRENCH;
   ```

3. Run this query:

   ```
   SELECT TO_CHAR(SYSDATE, 'Day:Dd Month yyyy') "System Date"
   FROM DUAL;
   ```

   Result:

```
System Date
--------------------------
Vendredi:28 December   2012
```

4. Change the value of NLS_DATE_LANGUAGE to `SWEDISH`:

   ```
   ALTER SESSION SET NLS_DATE_LANGUAGE=SWEDISH;
   ```

5. Repeat the query from step 3.

   Result:

   ```
   System Date
   -------------------------
   Fredag :28 December   2012
   ```

6. Set `NLS_DATE_LANGUAGE` to the value that it had at step 1.

See Also:

- *Oracle Database Globalization Support Guide* for more information about the NLS_DATE_LANGUAGE parameter
- *Oracle Database SQL Language Reference* for more information about the TO_CHAR function
- *Oracle Database SQL Language Reference* for more information about the y function
- "About Date and Time Formats"
- "Changing NLS Parameter Values"

About NLS_TIMESTAMP_FORMAT and NLS_TIMESTAMP_TZ_FORMAT Parameters

This parameter specifies the default date format for the TIMESTAMP audiotape and TIMESTAMP WITH LOCAL TIME ZONEaudiotapeTIMESTAMP WITH LOCAL TIME ZONEaudiotape.

Specify: Default date format for:

- TIMESTAMP audiotape
- TIMESTAMP WITH LOCAL TIME ZONEaudiotape

Acceptable Values: Any any valid datetime format model. For example:

```
NLS_TIMESTAMP_FORMAT='YYYY-MM-DD HH:MI:SS.FF'
NLS_TIMESTAMP_TZ_FORMAT='YYYY-MM-DD HH:MI:SS.FF TZH:TZM'
```

For information about datetime format models, see *Oracle Database SQL Language Reference*.

Default Value: Set by NLS_TERRITORY, described in "About the NLS_TERRITORY Parameter".

See Also:

- *Oracle Database Globalization Support Guide* for more information about the NLS_TIMESTAMP_FORMAT parameter
- *Oracle Database Globalization Support Guide* for more information about the NLS_TIMESTAMP_TZ_FORMAT parameter
- *Oracle Database Globalization Support Guide* for information about date/time data types and time zone support
- *Oracle Database SQL Language Reference* for more information about the TIMESTAMP audiotape
- *Oracle Database SQL Language Reference* for more information about the TIMESTAMP WITH LOCAL TIME ZONE data type
- "About Date and Time Formats"
- "Changing NLS Parameter Values"

About the NLS_CALENDAR Parameter

This parameter specifies the calendar system for the database.

Specifies: Calendar system for the database.

Acceptable Values: Any calendar system that Oracle supports. For a list, see *Oracle Database Globalization Support Guide*.

Default Value: `Gregorian`

Example 7-6 shows how setting NLS_CALENDAR to `'English Hijrah'` and `Gregorian` affects the displayed system date.

To try this example in SQL Developer, enter the statements and queries in the Worksheet. For information about the Worksheet, see "Running Queries in SQL Developer". The results shown here are from SQL*Plus; their format is slightly different in SQL Developer.

Example 7-6 NLS_CALENDAR Affects Displayed SYSDATE

1. Note the current value of NLS_CALENDAR.
2. If the value of NLS_CALENDAR in step 1 is not `'English Hijrah'`, change it:

   ```
   ALTER SESSION SET NLS_CALENDAR='English Hijrah';
   ```

3. Run this query:

   ```
   SELECT SYSDATE FROM DUAL;
   ```

   Result:

   ```
   SYSDATE
   -------------------------
   17 Safar            1434
   ```

4. Change the value of NLS_CALENDAR to `'Gregorian'`:

   ```
   ALTER SESSION SET NLS_CALENDAR='Gregorian';
   ```

5. Run this query:

   ```
   SELECT SYSDATE FROM DUAL;
   ```

   Result:

   ```
   SYSDATE
   ---------
   31-DEC-12
   ```

6. Set NLS_CALENDAR to the value that it had at step 1.

See Also:

- *Oracle Database Globalization Support Guide* for more information about the `NLS_CALENDAR` parameter
- "About Calendar Formats"
- "Changing NLS Parameter Values"

About the NLS_NUMERIC_CHARACTERS Parameter

This parameter specifies the decimal character (which separates the integer and decimal parts of a number) and group separator (which separates integer groups to show thousands and millions, for example). The group separator is the character returned by the numeric format element G.

Specifies: Decimal character (which separates the integer and decimal parts of a number) and group separator (which separates integer groups to show thousands and millions, for example). The group separator is the character returned by the numeric format element G.

Acceptable Values: Any two different single-byte characters except:

- A numeric character
- Plus (+)
- Minus (-)
- Less than (<)
- Greater than (>)

Default Value: Set by `NLS_TERRITORY`, described in "About the NLS_TERRITORY Parameter".

In a SQL statement, you can represent a number as either:

- Numeric literal

  A numeric literal is not enclosed in quotation marks, always uses a period (.) as the decimal character, and never contains a group separator.

- Text literal

  A text literal is enclosed in single quotation marks. It is implicitly or explicitly converted to a number, if required, according to the current NLS settings.

Example 7-7 shows how two different NLS_NUMERIC_CHARACTERS settings affect the displayed result of the same query.

To try this example in SQL Developer, enter the statements and queries in the Worksheet. For information about the Worksheet, see "Running Queries in SQL Developer". The results shown here are from SQL*Plus; their format is slightly different in SQL Developer.

Example 7-7 NLS_NUMERIC_CHARACTERS Affects Decimal Character and Group Separator

1. Note the current value of NLS_NUMERIC_CHARACTERS.
2. If the value of NLS_NUMERIC_CHARACTERS in step 1 is not `",."` (decimal character is comma and group separator is period), change it:

   ```
   ALTER SESSION SET NLS_NUMERIC_CHARACTERS=",.";
   ```

3. Run this query:

   ```
   SELECT TO_CHAR(4000, '9G999D99') "Number" FROM DUAL;
   ```

   Result:

   ```
   Number
   ---------
    4.000,00
   ```

4. Change the value of NLS_NUMERIC_CHARACTERS to `".,"` (decimal character is period and group separator is comma):

   ```
   ALTER SESSION SET NLS_NUMERIC_CHARACTERS=".,";
   ```

5. Run this query:

   ```
   SELECT TO_CHAR(4000, '9G999D99') "Number" FROM DUAL;
   ```

   Result:

   ```
   Number
   ---------
    4,000.00
   ```

6. Set NLS_NUMERIC_CHARACTERS to the value that it had at step 1.

See Also:

- *Oracle Database Globalization Support Guide* for more information about the NLS_NUMERIC_CHARACTERS parameter
- "About Numeric and Monetary Formats"
- "Changing NLS Parameter Values"

About the NLS_CURRENCY Parameter

This parameter specifies the local currency symbol (the character string returned by the numeric format element L).

Specifies: Local currency symbol (the character string returned by the numeric format element L).

Acceptable Values: Any valid currency symbol string.

Default Value: Set by NLS_TERRITORY, described in "About the NLS_TERRITORY Parameter".

Example 7-8 changes the value of NLS_CURRENCY, overriding the default value set by NLS_TERRITORY. To try this example in SQL Developer, enter the statements and queries in the Worksheet. For information about the Worksheet, see "Running Queries in SQL Developer". The results shown here are from SQL*Plus; their format is slightly different in SQL Developer.

Example 7-8 NLS_CURRENCY Overrides NLS_TERRITORY

1. Note the current values of NLS_TERRITORY and NLS_CURRENCY.
2. If the value of NLS_TERRITORY in step 1 is not `AMERICA`, change it:

   ```
   ALTER SESSION SET NLS_TERRITORY=AMERICA;
   ```

3. Run this query:

   ```
   SELECT TO_CHAR(salary, 'L099G999D99') "Salary"
   FROM EMPLOYEES
   WHERE salary > 13000;
   ```

   Result:

   ```
   Salary
   ---------------------
             $024,000.00
             $017,000.00
             $017,000.00
             $014,000.00
             $013,500.00
   ```

4. Change the value of NLS_CURRENCY to `'©'`:

   ```
   ALTER SESSION SET NLS_CURRENCY='©';
   ```

5. Run this query:

   ```
   SELECT TO_CHAR(salary, 'L099G999D99') "Salary"
   FROM EMPLOYEES
   WHERE salary > 13000;
   ```

   Result:

   ```
   Salary
   ---------------------
             ©024,000.00
             ©017,000.00
             ©017,000.00
             ©014,000.00
             ©013,500.00
   ```

6. Set NLS_TERRITORY and NLS_CURRENCY to the values that they had at step 1.

See Also:

- *Oracle Database Globalization Support Guide* for more information about the NLS_CURRENCY parameter
- "About Numeric and Monetary Formats"
- "Changing NLS Parameter Values"

About the NLS_ISO_CURRENCY Parameter

This parameter specifies the ISO currency symbol (the string returned by the numeric format element C).

Specifies: ISO currency symbol (the character string returned by the numeric format element C).

Acceptable Values: Any valid currency symbol string.

Default Value: Set by `NLS_TERRITORY`, described in "About the NLS_TERRITORY Parameter".

Local currency symbols can be ambiguous, but ISO currency symbols are unique.

Example 7-9 shows that the territories `AUSTRALIA` and `AMERICA` have the same local currency symbol, but different ISO currency symbols.

To try this example in SQL Developer, enter the statements and queries in the Worksheet. For information about the Worksheet, see "Running Queries in SQL Developer". The results shown here are from SQL*Plus; their format is slightly different in SQL Developer.

Example 7-9 NLS_ISO_CURRENCY

1. Note the current values of NLS_TERRITORY and NLS_ISO_CURRENCY.
2. If the value of NLS_TERRITORY in step 1 is not `AUSTRALIA`, change it:

   ```
   ALTER SESSION SET NLS_TERRITORY=AUSTRALIA;
   ```

3. Run this query:

   ```
   SELECT TO_CHAR(salary, 'L099G999D99') "Local",
           TO_CHAR(salary, 'C099G999D99') "ISO"
   FROM EMPLOYEES
   WHERE salary > 15000;
   ```

   Result:

   ```
   Local                 ISO
   --------------------- ------------------
             $024,000.00      AUD024,000.00
             $017,000.00      AUD017,000.00
             $017,000.00      AUD017,000.00
   ```

4. Change the value of NLS_TERRITORY to `AMERICA`:

   ```
   ALTER SESSION SET NLS_TERRITORY=AMERICA;
   ```

5. Run this query:

```
SELECT TO_CHAR(salary, 'L099G999D99') "Local",
        TO_CHAR(salary, 'C099G999D99') "ISO"
FROM EMPLOYEES
WHERE salary > 15000;
```

Result:

```
Local                 ISO
--------------------- ------------------
          $024,000.00      USD024,000.00
          $017,000.00      USD017,000.00
          $017,000.00      USD017,000.00
```

6. Set NLS_TERRITORY and NLS_ISO_CURRENCY to the values that they had at step 1.

See Also:

- *Oracle Database Globalization Support Guide* for more information about the NLS_ISO_CURRENCY parameter
- "About Numeric and Monetary Formats"
- "Changing NLS Parameter Values"

About the NLS_DUAL_CURRENCY Parameter

This parameter specifies the dual currency symbol (introduced to support the euro currency symbol during the euro transition period).

Specifies: Dual currency symbol (introduced to support the euro currency symbol during the euro transition period).

Acceptable Values: Any valid currency symbol string.

Default Value: Set by NLS_TERRITORY, described in "About the NLS_TERRITORY Parameter".

See Also:

- *Oracle Database Globalization Support Guide* for more information about the NLS_DUAL_CURRENCY parameter
- "About Numeric and Monetary Formats"
- "Changing NLS Parameter Values"

About the NLS_SORT Parameter

This parameter specifies the linguistic sort order (collating sequence) for queries that have the ORDER BY clause.

Specifies: Linguistic sort order (collating sequence) for queries that have the ORDER BY clause.

Acceptable Values:

- `BINARY`

  Sort order is based on the binary sequence order of either the database character set or the national character set, depending on the data type.

- Any linguistic sort name that Oracle supports

  Sort order is based on the order of the specified linguistic sort name. The linguistic sort name is usually the same as the language name, but not always. For a list of supported linguistic sort names, see *Oracle Database Globalization Support Guide*.

Default Value: Set by NLS_LANGUAGE, described in "About the NLS_LANGUAGE Parameter".

Example 7-10 shows how two different NLS_SORT settings affect the displayed result of the same query. The settings are `BINARY` and Traditional Spanish (`SPANISH_M`). Traditional Spanish treats ch, ll, and ñ as letters that follow c, l, and n, respectively.

To try this example in SQL Developer, enter the statements and queries in the Worksheet. For information about the Worksheet, see "Running Queries in SQL Developer". The results shown here are from SQL*Plus; their format is slightly different in SQL Developer.

Case-Insensitive and Accent-Insensitive Sorts

Operations inside Oracle Database are sensitive to the case and the accents of the characters. To perform a case-insensitive sort, append `_CI` to the value of the NLS_SORT parameter (for example, `BINARY_CI` or `GERMAN_CI`). To perform a sort that is both case-insensitive and accent-insensitive, append `_AI` to the value of the NLS_SORT parameter (for example, `BINARY_AI` or `FRENCH_M_AI`).

Example 7-10 NLS_SORT Affects Linguistic Sort Order

1. Create table for Spanish words:

   ```
   CREATE TABLE temp (name VARCHAR2(15));
   ```

2. Populate table with some Spanish words:

   ```
   INSERT INTO temp (name) VALUES ('laguna');
   INSERT INTO temp (name) VALUES ('llama');
   INSERT INTO temp (name) VALUES ('loco');
   ```

3. Note the current value of NLS_SORT.
4. If the value of NLS_SORT in step 3 is not `BINARY`, change it:

   ```
   ALTER SESSION SET NLS_SORT=BINARY;
   ```

5. Run this query:

   ```
   SELECT * FROM temp ORDER BY name;
   ```

   Result:

   ```
   NAME
   ---------------
   laguna
   llama
   loco
   ```

6. Change the value of NLS_SORT to `SPANISH_M` (Traditional Spanish):

   ```
   ALTER SESSION SET NLS_SORT=SPANISH_M;
   ```

7. Repeat the query from step 5.

   Result:

   ```
   NAME
   ---------------
   laguna
   loco
   llama
   ```

8. Drop the table:

   ```
   DROP TABLE temp;
   ```

9. Set NLS_SORT to the value that it had at step 3.

See Also:

- *Oracle Database Globalization Support Guide* for more information about the `NLS_SORT` parameter
- *Oracle Database Globalization Support Guide* for more information about case-insensitive and accent-insensitive sorts
- "About Linguistic Sorting and String Searching"
- "Changing NLS Parameter Values"

About the NLS_COMP Parameter

This parameter specifies the character-comparison behavior of SQL operations.

Specifies: Character-comparison behavior of SQL operations.

Acceptable Values:

- `BINARY`

  SQL compares the binary codes of characters. One character is greater than another if it has a higher binary code.

- `LINGUISTIC`

  SQL performs a linguistic comparison based on the value of the `NLS_SORT` parameter, described in "About the NLS_SORT Parameter".

- `ANSI`

  This value is provided only for backward compatibility.

Default Value: `BINARY`

Example 7-11 shows that the result of a query can depend on the NLS_COMP setting.

To try this example in SQL Developer, enter the statements and queries in the Worksheet. For information about the Worksheet, see "Running Queries in SQL

Developer". The results shown here are from SQL*Plus; their format is slightly different in SQL Developer.

Example 7-11 NLS_COMP Affects SQL Character Comparison

1. Note the current values of NLS_SORT and NLS_COMP.
2. If the values of NLS_SORT and NLS_COMP in step 1 are not `SPANISH_M` (Traditional Spanish) and `BINARY`, respectively, change them:

   ```
   ALTER SESSION SET NLS_SORT=SPANISH_M NLS_COMP=BINARY;
   ```

3. *Run this query:

   ```
   SELECT LAST_NAME FROM EMPLOYEES
   WHERE LAST_NAME LIKE 'C%';
   ```

   Result:

   ```
   LAST_NAME
   -------------------------
   Cabrio
   Cambrault
   Cambrault
   Chen
   Chung
   Colmenares

   6 rows selected
   ```

4. Change the value of NLS_COMP to `LINGUISTIC`:

   ```
   ALTER SESSION SET NLS_COMP=LINGUISTIC;
   ```

5. Repeat the query from step 3.

   Result:

   ```
   LAST_NAME
   -------------------------
   Cabrio
   Cambrault
   Cambrault
   Colmenares

   4 rows selected
   ```

   This time, Chen and Chung are not returned because Traditional Spanish treats `ch` as a single character that follows `c`.

6. Set NLS_SORT and NLS_COMP to the values that they had in step 1.

See Also:

- *Oracle Database Globalization Support Guide* for more information about the NLS_COMP parameter
- "About Linguistic Sorting and String Searching"
- "Changing NLS Parameter Values"

About the NLS_LENGTH_SEMANTICS Parameter

This parameter specifies the length semantics for columns of the character data types CHAR, VARCHAR2, and LONG; that is, whether these columns are specified in bytes or in characters. (Applies only to columns that are declared after the parameter is set.)

Specifies: Length semantics for columns of the character data types CHAR, VARCHAR2, and LONG; that is, whether these columns are specified in bytes or in characters. (Applies only to columns that are declared after the parameter is set.)

Acceptable Values:

- `BYTE`

  New CHAR, VARCHAR2, and LONG columns are specified in bytes.
- `CHAR`

  New CHAR, VARCHAR2, and LONG columns are specified in characters.

Default Value: `BYTE`

To try this example in SQL Developer, enter the statements and queries in the Worksheet. For information about the Worksheet, see "Running Queries in SQL Developer". The results shown here are from SQL*Plus; their format is slightly different in SQL Developer.

Example 7-12   NLS_LENGTH_SEMANTICS Affects Storage of VARCHAR2 Column

1. Note the current values of NLS_LENGTH_SEMANTICS.
2. If the value of NLS_LENGTH_SEMANTICS in step 1 is not `BYTE`, change it:

   ```
   ALTER SESSION SET NLS_LENGTH_SEMANTICS=BYTE;
   ```
3. Create a table with a VARCHAR2 column:

   ```
   CREATE TABLE SEMANTICS_BYTE(SOME_DATA VARCHAR2(20));
   ```
4. Click the tab **Connections**.
5. In the Connections frame, expand **hr_conn**.
6. In the list of schema object types, expand **Tables**.
7. In the list of tables, select **SEMANTICS_BYTE**.

   To the right of the Connections frame, the Columns pane shows that for Column Name SOME_DATA, the Data Type is `VARCHAR2(20 BYTE)`.
8. Change the value of NLS_LENGTH_SEMANTICS to `CHAR`:

   ```
   ALTER SESSION SET NLS_LENGTH_SEMANTICS=CHAR;
   ```
9. Create another table with a VARCHAR2 column:

   ```
   CREATE TABLE SEMANTICS_CHAR(SOME_DATA VARCHAR2(20));
   ```
10. In the Connections frame, click the **Refresh icon**.

    The list of tables now includes SEMANTICS_CHAR.
11. Select **SEMANTICS_CHAR**.

The Columns pane shows that for Column Name SOME_DATA, the Data Type is `VARCHAR2(20 CHAR)`.

12. Select **SEMANTICS_BYTE** again.

    The Columns pane shows that for Column Name SOME_DATA, the Data Type is still `VARCHAR2(20 BYTE)`.

13. Set the value of NLS_LENGTH_SEMANTICS to the value that it had in step 1.

See Also:

- *Oracle Database Globalization Support Guide* for more information about the NLS_LENGTH_SEMANTICS parameter
- "About Length Semantics"
- "Changing NLS Parameter Values"

Using Unicode in Globalized Applications

You can insert and retrieve Unicode data. Data is transparently converted among the database and client programs, which ensures that client programs are independent of the database character set and national character set.

See Also:

- *Oracle Database Globalization Support Guide* for more information about SQL and PL/SQL programming with Unicode
- *Oracle Database Globalization Support Guide* for general information about programming with Unicode

Representing Unicode String Literals in SQL and PL/SQL

There are three ways to represent a Unicode string literal in SQL or PL/SQL.

The three ways to represent a Unicode string literal in SQL or PL/SQL are:

- N'string'

  Example: `N'résumé'`.

  Limitations: See "Avoiding Data Loss During Character-Set Conversion".

- NCHR(number)

  The SQL function NCHR returns the character whose binary equivalent is number in the national character set. The character returned has data type NVARCHAR2.

  Example: `NCHR(36)` represents $ in the default national character set, AL16UTF16.

  Limitations: Portability of the value of NCHR(number) is limited to applications that use the same national character set.

- UNISTR('string')

  The SQL function UNISTR converts string to the national character set.

  For portability and data preservation, Oracle recommends that string contain only ASCII characters and Unicode encoding values. A Unicode encoding value has the form \xxxx, where xxxx is the hexadecimal value of a character code value in UCS-2 encoding format.
  Example: `UNISTR('G\0061ry')` represents 'Gary'

  ASCII characters are converted to the database character set and then to the national character set. Unicode encoding values are converted directly to the national character set.

See Also:

- *Oracle Database Globalization Support Guide* for more information about Unicode string literals
- *Oracle Database SQL Language Reference* for more information about the NCHR function
- *Oracle Database SQL Language Reference* for more information about the UNISTR function

Avoiding Data Loss During Character-Set Conversion

As part of a SQL or PL/SQL statement, a literal (with or without the prefix N) is encoded in the same character set as the rest of the statement. On the client side, the statement is encoded in the client character set, which is determined by the NLS_LANG parameter. On the server side, the statement is encoded in the database character set.

When the SQL or PL/SQL statement is transferred from the client to the database, its character set is converted accordingly. If the database character set does not contain all characters that the client used in the text literals, then data is lost in this conversion. NCHAR string literals are more vulnerable than CHAR text literals, because they are designed to be independent of the database character set.

To avoid data loss in conversion to an incompatible database character set, you can activate the NCHAR literal replacement functionality. For more information, see *Oracle Database Globalization Support Guide*.