Format Models

A **format model** is a character literal that describes the format of datetime or numeric data stored in a character string. A format model does not change the internal representation of the value in the database. When you convert a character string into a date or number, a format model determines how Oracle Database interprets the string. In SQL statements, you can use a format model as an argument of the `TO_CHAR` and `TO_DATE` functions to specify:

- The format for Oracle to use to return a value from the database
- The format for a value you have specified for Oracle to store in the database

For example:

- The datetime format model for the string '`17:45:29`' is '`HH24:MI:SS`'.
- The datetime format model for the string '`11-Nov-1999`' is '`DD-Mon-YYYY`'.
- The number format model for the string '`$2,304.25`' is '`$9,999.99`'.

For lists of number and datetime format model elements, see Table 2-15 and Table 2-17.

The values of some formats are determined by the value of initialization parameters. For such formats, you can specify the characters returned by these format elements implicitly using the initialization parameter `NLS_TERRITORY`. You can change the default date format for your session with the `ALTER SESSION` statement.

> **See Also:**
>
> - ALTER SESSION for information on changing the values of these parameters and Format Model Examples for examples of using format models
> - TO_CHAR (datetime), TO_CHAR (number), and TO_DATE
> - *Oracle Database Reference* and *Oracle Database Globalization Support Guide* for information on these parameters

This remainder of this section describes how to use the following format models:

- Number Format Models
- Datetime Format Models
- Format Model Modifiers

Number Format Models

h

You can use number format models in the following functions:

- In the `TO_CHAR` function to translate a value of `NUMBER`, `BINARY_FLOAT`, or `BINARY_DOUBLE` data type to `VARCHAR2` data type
- In the `TO_NUMBER` function to translate a value of `CHAR` or `VARCHAR2` data type to `NUMBER` data type
- In the `TO_BINARY_FLOAT` and `TO_BINARY_DOUBLE` functions to translate `CHAR` and `VARCHAR2` expressions to `BINARY_FLOAT` or `BINARY_DOUBLE`values

All number format models cause the number to be rounded to the specified number of significant digits. If a value has more significant digits to the left of the decimal place than are specified in the format, then pound signs (#) replace the value. This event typically occurs when you are using `TO_CHAR` with a restrictive number format string, causing a rounding operation.

- If a positive `NUMBER` value is extremely large and cannot be represented in the specified format, then the infinity sign (~) replaces the value. Likewise, if a negative `NUMBER` value is extremely small and cannot be represented by the specified format, then the negative infinity sign replaces the value (-~).
- If a `BINARY_FLOAT` or `BINARY_DOUBLE` value is converted to `CHAR` or `NCHAR`, and the input is either infinity or `NaN` (not a number), then Oracle always returns the pound signs to replace the value. However, if you omit the format model, then Oracle returns either `Inf` or `Nan` as a string.

Number Format Elements

A number format model is composed of one or more number format elements. The tables that follow list the elements of a number format model and provide some examples.

Negative return values automatically contain a leading negative sign and positive values automatically contain a leading space unless the format model contains the `MI`, `S`, or `PR` format element.

Table 2-15 Number Format Elements

Element	Example	Description

, (comma)	`9,999`	Returns a comma in the specified position. You can specify multiple commas in a number format model. **Restrictions:** • A comma element cannot begin a number format model. • A comma cannot appear to the right of a decimal character or period in a number format model.
. (period)	`99.99`	Returns a decimal point, which is a period (.) in the specified position. **Restriction:** You can specify only one period in a number format model.
$	`$9999`	Returns value with a leading dollar sign.

0	`0999`
	
	`9990`
		Returns leading zeros.
		
		Returns trailing zeros.

9	`9999`	Returns value with the specified number of digits with a leading space if positive or with a leading minus if negative. Leading zeros are blank, except for a zero value, which returns a zero for the integer part of the fixed-point number.
B	`B9999`	Returns blanks for the integer part of a fixed-point number when the integer part is zero (regardless of zeros in the format model).
C	`C999`	Returns in the specified position the ISO currency symbol (the current value of the `NLS_ISO_CURRENCY` parameter).
D	`99D99`	Returns in the specified position the decimal character, which is the current value of the `NLS_NUMERIC_CHARACTER` parameter. The default is a period (.). **Restriction:** You can specify only one decimal character in a number format model.
EEEE	`9.9EEEE`	Returns a value using in scientific notation.

G	`9G999`	Returns in the specified position the group separator (the current value of the `NLS_NUMERIC_CHARACTER` parameter). You can specify multiple group separators in a number format model. **Restriction:** A group separator cannot appear to the right of a decimal character or period in a number format model.
L	`L999`	Returns in the specified position the local currency symbol (the current value of the `NLS_CURRENCY`parameter).
MI	`9999MI`	Returns negative value with a trailing minus sign (-). Returns positive value with a trailing blank. **Restriction:** The MI format element can appear only in the last position of a number format model.

PR	`9999PR`	Returns negative value in <angle brackets>. Returns positive value with a leading and trailing blank. **Restriction:** The PR format element can appear only in the last position of a number format model.

RN
 	
rn
	`RN`
	
	`rn`
		Returns a value as Roman numerals in uppercase.
		
		Returns a value as Roman numerals in lowercase.
		
		Value can be an integer between 1 and 3999.

S	`S9999`
	
	`9999S`
		Returns negative value with a leading minus sign (-).
		
		Returns positive value with a leading plus sign (+).
		
		Returns negative value with a trailing minus sign (-).
		
		Returns positive value with a trailing plus sign (+).
		
		Restriction: The S format element can appear only in the first or last position of a number format model.

TM	`TM`	The text minimum number format model returns (in decimal output) the smallest number of characters possible. This element is case insensitive. The default is TM9, which returns the number in fixed notation unless the output exceeds 64 characters. If the output exceeds 64 characters, then Oracle Database automatically returns the number in scientific notation. **Restrictions:** • You cannot precede this element with any other element. • You can follow this element only with one 9 or one E (or e), but not with any combination of these. The following statement returns an error: `SELECT TO_CHAR(1234, 'TM9e') FROM DUAL;`
U	`U9999`	Returns in the specified position the Euro (or other) dual currency symbol, determined by the current value of the `NLS_DUAL_CURRENCY` parameter.

V	`999V99`	Returns a value multiplied by 10^n (and if necessary, round it up), where *n* is the number of 9's after the `V`.

X	`XXXX`
	
	`xxxx`
		Returns the hexadecimal value of the specified number of digits. If the specified number is not an integer, then Oracle Database rounds it to an integer.
		
		Restrictions:
		
		• This element accepts only positive values or 0. Negative values return an error.
		
		• You can precede this element only with 0 (which returns leading zeroes) or FM. Any other elements return an error. If you specify neither 0 nor FM with X, then the return always has one leading blank. Refer to the format model modifier FM for more information.

Table 2-16 shows the results of the following query for different values of *number* and *'fmt'*:

```
SELECT TO_CHAR(number, 'fmt')
  FROM DUAL;
```

Table 2-16 Results of Number Conversions

number	'fmt'	Result
-1234567890	9999999999S	'1234567890-'
0	99.99	' .00'
+0.1	99.99	' .10'
-0.2	99.99	' -.20'
0	90.99	' 0.00'
+0.1	90.99	' 0.10'
-0.2	90.99	' -0.20'
0	9999	' 0'
1	9999	' 1'
0	B9999	' '
1	B9999	' 1'
0	B90.99	' '
+123.456	999.999	' 123.456'
-123.456	999.999	'-123.456'
+123.456	FM999.009	'123.456'
+123.456	9.9EEEE	' 1.2E+02'

number	'fmt'	Result
+1E+123	9.9EEEE	' 1.0E+123'
+123.456	FM9.9EEEE	'1.2E+02'
+123.45	FM999.009	'123.45'
+123.0	FM999.009	'123.00'
+123.45	L999.99	' $123.45'
+123.45	FML999.99	'$123.45'
+1234567890	9999999999S	'1234567890+'

Datetime Format Models

You can use datetime format models in the following functions:

- In the `TO_*` datetime functions to translate a character value that is in a format other than the default format into a datetime value. (The `TO_*` datetime functions are `TO_DATE`, `TO_TIMESTAMP`, and `TO_TIMESTAMP_TZ`.)
- In the `TO_CHAR` function to translate a datetime value into a character value that is in a format other than the default format (for example, to print the date from an application)

The total length of a datetime format model cannot exceed 22 characters.

The default datetime formats are specified either explicitly with the NLS session parameters `NLS_DATE_FORMAT`, `NLS_TIMESTAMP_FORMAT`, and `NLS_TIMESTAMP_TZ_FORMAT`, or implicitly with the NLS session parameter `NLS_TERRITORY`. You can change the default datetime formats for your session with the `ALTER SESSION` statement.

> **See Also:**
> ALTER SESSION and *Oracle Database Globalization Support Guide* for information on the NLS parameters

Datetime Format Elements

A datetime format model is composed of one or more datetime format elements as listed in Table 2-17.

- For input format models, format items cannot appear twice, and format items that represent similar information cannot be combined. For example, you cannot use 'SYYYY' and 'BC' in the same format string.
- The second column indicates whether the format element can be used in the `TO_*` datetime functions. All format elements can be used in the `TO_CHAR`function.
- The following datetime format elements can be used in timestamp and interval format models, but not in the original `DATE` format model: `FF`, `TZD`, `TZH`, `TZM`, and `TZR`.
- Many datetime format elements are padded with blanks or leading zeroes to a specific length. Refer to the format model modifier FM for more information.

> **Note:**
> Oracle recommends that you use the 4-digit year element (`YYYY`) instead of the shorter year elements for these reasons:
>
> - The 4-digit year element eliminates ambiguity.
> - The shorter year elements may affect query optimization because the year is not known at query compile time and can only be determined at run time.

Uppercase Letters in Date Format Elements

Capitalization in a spelled-out word, abbreviation, or Roman numeral follows capitalization in the corresponding format element. For example, the date format model 'DAY' produces capitalized words like 'MONDAY'; 'Day' produces 'Monday'; and 'day' produces 'monday'.

Punctuation and Character Literals in Datetime Format Models

You can include these characters in a date format model:

- Punctuation such as hyphens, slashes, commas, periods, and colons
- Character literals, enclosed in double quotation marks

These characters appear in the return value in the same location as they appear in the format model.

Table 2-17 Datetime Format Elements

Element	TO_* datetime functions?	Description

`-`
`/`
`,`
`.`
`;`
`:`
`"text"`
	Yes	Punctuation and quoted text is reproduced in the result.

`AD` `A.D.`	Yes	AD indicator with or without periods.
`AM` `A.M.`	Yes	Meridian indicator with or without periods.
`BC` `B.C.`	Yes	BC indicator with or without periods.
`CC` `SCC`		Century. • If the last 2 digits of a 4-digit year are between 01 and 99 (inclusive), then the century is one greater than the first 2 digits of that year. • If the last 2 digits of a 4-digit year are 00, then the century is the same as the first 2 digits of that year. For example, 2002 returns 21; 2000 returns 20.
`D`	Yes	Day of week (1-7). This element depends on the NLS territory of the session.
`DAY`	Yes	Name of day.

`DD`	Yes	Day of month (1-31).
`DDD`	Yes	Day of year (1-366).
`DL`	Yes	Returns a value in the long date format, which is an extension of the Oracle Database `DATE` format, determined by the current value of the `NLS_DATE_FORMAT` parameter. Makes the appearance of the date components (day name, month number, and so forth) depend on the `NLS_TERRITORY` and `NLS_LANGUAGE` parameters. For example, in the `AMERICAN_AMERICA` locale, this is equivalent to specifying the format `'fmDay,Month dd, yyyy'`. In the `GERMAN_GERMANY` locale, it is equivalent to specifying the format `'fmDay, dd. Month yyyy'`. **Restriction:** You can specify this format only with the `TS` element, separated by white space.

`DS`	Yes	Returns a value in the short date format. Makes the appearance of the date components (day name, month number, and so forth) depend on the `NLS_TERRITORY` and `NLS_LANGUAGE` parameters. For example, in the `AMERICAN_AMERICA` locale, this is equivalent to specifying the format '`MM/DD/RRRR`'. In the `ENGLISH_UNITED_KINGDOM` locale, it is equivalent to specifying the format '`DD/MM/RRRR`'. **Restriction:** You can specify this format only with the `TS` element, separated by white space.
`DY`	Yes	Abbreviated name of day.
`E`	Yes	Abbreviated era name (Japanese Imperial, ROC Official, and Thai Buddha calendars).
`EE`	Yes	Full era name (Japanese Imperial, ROC Official, and Thai Buddha calendars).

`FF [1..9]`	Yes	Fractional seconds; no radix character is printed. Use the X format element to add the radix character. Use the numbers 1 to 9 after FF to specify the number of digits in the fractional second portion of the datetime value returned. If you do not specify a digit, then Oracle Database uses the precision specified for the datetime data type or the data type's default precision. Valid in timestamp and interval formats, but not in `DATE` formats. **Examples:** `'HH:MI:SS.FF'` `SELECT TO_CHAR(SYSTIMESTAMP, 'SS.FF3') from DUAL;`
`FM`	Yes	Returns a value with no leading or trailing blanks. **See Also**: FM
`FX`	Yes	Requires exact matching between the character data and the format model. **See Also**: FX
`HH` `HH12`	Yes	Hour of day (1-12).

`HH24`	Yes	Hour of day (0-23).
`IW`		Calendar week of year (1-52 or 1-53), as defined by the ISO 8601 standard. • A calendar week starts on Monday. • The first calendar week of the year includes January 4. • The first calendar week of the year may include December 29, 30 and 31. • The last calendar week of the year may include January 1, 2, and 3.
`IYYY`		4-digit year of the year containing the calendar week, as defined by the ISO 8601 standard.

`IYY`
`IY`
`I`
		Last 3, 2, or 1 digit(s) of the year containing the calendar week, as defined by the ISO 8601 standard.

`J`	Yes	Julian day; the number of days since January 1, 4712 BC. Number specified with J must be integers.
`MI`	Yes	Minute (0-59).

`MM`	Yes	Month (01-12; January = 01).
`MON`	Yes	Abbreviated name of month.
`MONTH`	Yes	Name of month.
`PM` `P.M.`	Yes	Meridian indicator with or without periods.
`Q`		Quarter of year (1, 2, 3, 4; January - March = 1).
`RM`	Yes	Roman numeral month (I-XII; January = I).
`RR`	Yes	Lets you store 20th century dates in the 21st century using only two digits. **See Also:** "The RR Datetime Format Element"
`RRRR`	Yes	Round year. Accepts either 4-digit or 2-digit input. If 2-digit, provides the same return as RR. If you do not want this functionality, then enter the 4-digit year.
`SS`	Yes	Second (0-59).
`SSSSS`	Yes	Seconds past midnight (0-86399).

`TS`	Yes	Returns a value in the short time format. Makes the appearance of the time components (hour, minutes, and so forth) depend on the `NLS_TERRITORY` and `NLS_LANGUAGE` initialization parameters. **Restriction:** You can specify this format only with the `DL` or `DS` element, separated by white space.
`TZD`	Yes	Daylight saving information. The TZD value is an abbreviated time zone string with daylight saving information. It must correspond with the region specified in TZR. Valid in timestamp and interval formats, but not in `DATE` formats. **Example:** `PST` (for US/Pacific standard time); `PDT` (for US/Pacific daylight time).
`TZH`	Yes	Time zone hour. (See `TZM` format element.) Valid in timestamp and interval formats, but not in `DATE` formats. **Example:** `'HH:MI:SS. FFTZH:TZM'`.

`TZM`	Yes	Time zone minute. (See `TZH` format element.) Valid in timestamp and interval formats, but not in `DATE` formats. **Example:** `'HH:MI:SS.FFTZH:TZM'`.
`TZR`	Yes	Time zone region information. The value must be one of the time zone region names supported in the database. Valid in timestamp and interval formats, but not in `DATE` formats. **Example:** US/Pacific
`WW`		Week of year (1-53) where week 1 starts on the first day of the year and continues to the seventh day of the year.
`W`		Week of month (1-5) where week 1 starts on the first day of the month and ends on the seventh.
`X`	Yes	Local radix character. **Example:** `'HH:MI:SSXFF'`.
`Y,YYY`	Yes	Year with comma in this position.

`YEAR` `SYEAR`		Year, spelled out; `S` prefixes BC dates with a minus sign (-).
`YYYY` `SYYYY`	Yes	4-digit year; `S` prefixes BC dates with a minus sign.

`YYY`
`YY`
`Y`
	Yes	Last 3, 2, or 1 digit(s) of year.

Oracle Database converts strings to dates with some flexibility. For example, when the `TO_DATE` function is used, a format model containing punctuation characters matches an input string lacking some or all of these characters, provided each numerical element in the input string contains the maximum allowed number of digits—for example, two digits '05' for 'MM' or four digits '2007' for 'YYYY'. The following statement does not return an error:

```
SELECT TO_CHAR(TO_DATE('0207','MM/YY'), 'MM/YY') FROM DUAL;

TO_CH
-----
02/07
```

However, the following format string does return an error, because the FX (format exact) format modifier requires an exact match of the expression and the format string:

```
SELECT TO_CHAR(TO_DATE('0207', 'fxmm/yy'), 'mm/yy') FROM DUAL;
SELECT TO_CHAR(TO_DATE('0207', 'fxmm/yy'), 'mm/yy') FROM DUAL;
                       *
ERROR at line 1:
ORA-01861: literal does not match format string
```

Any non-alphanumeric character is allowed to match the punctuation characters in the format model. For example, the following statement does not return an error:

```
SELECT TO_CHAR (TO_DATE('02#07','MM/YY'), 'MM/YY') FROM DUAL;

TO_CH
-----
02/07
```

See Also:

> "Format Model Modifiers" and "String-to-Date Conversion Rules" for more information

Datetime Format Elements and Globalization Support

The functionality of some datetime format elements depends on the country and language in which you are using Oracle Database. For example, these datetime format elements return spelled values:

- MONTH
- MON
- DAY
- DY
- BC or AD or B.C. or A.D.
- AM or PM or A.M or P.M.

The language in which these values are returned is specified either explicitly with the initialization parameter `NLS_DATE_LANGUAGE` or implicitly with the initialization parameter `NLS_LANGUAGE`. The values returned by the `YEAR` and `SYEAR` datetime format elements are always in English.

The datetime format element `D` returns the number of the day of the week (1-7). The day of the week that is numbered 1 is specified implicitly by the initialization parameter `NLS_TERRITORY`.

> **See Also:**
>
> *Oracle Database Reference* and *Oracle Database Globalization Support Guide* for information on globalization support initialization parameters

ISO Standard Date Format Elements

Oracle calculates the values returned by the datetime format elements IYYY, IYY, IY, I, and IW according to the ISO standard. For information on the differences between these values and those returned by the datetime format elements YYYY, YYY, YY, Y, and WW, see the discussion of globalization support in *Oracle Database Globalization Support Guide*.

The RR Datetime Format Element

The `RR` datetime format element is similar to the `YY` datetime format element, but it provides additional flexibility for storing date values in other centuries. The `RR` datetime format element lets you store 20th century dates in the 21st century by specifying only the last two digits of the year.

If you use the `TO_DATE` function with the `YY` datetime format element, then the year returned always has the same first 2 digits as the current year. If you use the `RR` datetime format element instead, then the century of the return value varies according to the specified two-digit year and the last two digits of the current year.

That is:

- If the specified two-digit year is 00 to 49, then
  - If the last two digits of the current year are 00 to 49, then the returned year has the same first two digits as the current year.
  - If the last two digits of the current year are 50 to 99, then the first 2 digits of the returned year are 1 greater than the first 2 digits of the current year.
- If the specified two-digit year is 50 to 99, then
  - If the last two digits of the current year are 00 to 49, then the first 2 digits of the returned year are 1 less than the first 2 digits of the current year.
  - If the last two digits of the current year are 50 to 99, then the returned year has the same first two digits as the current year.

The following examples demonstrate the behavior of the RR datetime format element.

RR Datetime Format Examples

Assume these queries are issued between 1950 and 1999:

```
SELECT TO_CHAR(TO_DATE('27-OCT-98', 'DD-MON-RR'), 'YYYY') "Year" FROM DUAL;

Year
----
1998

SELECT TO_CHAR(TO_DATE('27-OCT-17', 'DD-MON-RR'), 'YYYY') "Year" FROM DUAL;

Year
----
2017
```

Now assume these queries are issued between 2000 and 2049:

```
SELECT TO_CHAR(TO_DATE('27-OCT-98', 'DD-MON-RR'), 'YYYY') "Year" FROM DUAL;

Year
----
1998

SELECT TO_CHAR(TO_DATE('27-OCT-17', 'DD-MON-RR'), 'YYYY') "Year" FROM DUAL;

Year
----
2017
```

Note that the queries return the same values regardless of whether they are issued before or after the year 2000. The RR datetime format element lets you write SQL statements that will return the same values from years whose first two digits are different.

Datetime Format Element Suffixes

Table 2-18 lists suffixes that can be added to datetime format elements:

Table 2-18 Date Format Element Suffixes

Suffix	Meaning	Example Element	Example Value
TH	Ordinal Number	`DDTH`	`4TH`
SP	Spelled Number	`DDSP`	`FOUR`
SPTH or THSP	Spelled, ordinal number	`DDSPTH`	`FOURTH`

Notes on date format element suffixes:

- When you add one of these suffixes to a datetime format element, the return value is always in English.
- Datetime suffixes are valid only to format output. You cannot use them to insert a date into the database.

Format Model Modifiers

The `FM` and `FX` modifiers, used in format models in the `TO_CHAR` function, control blank padding and exact format checking.

A modifier can appear in a format model more than once. In such a case, each subsequent occurrence toggles the effects of the modifier. Its effects are enabled for the portion of the model following its first occurrence, and then disabled for the portion following its second, and then reenabled for the portion following its third, and so on.

h

FM

Fill mode. Oracle uses trailing blank characters and leading zeroes to fill format elements to a constant width. The width is equal to the display width of the largest element for the relevant format model:

- Numeric elements are padded with leading zeros to the width of the maximum value allowed for the element. For example, the `YYYY` element is padded to four digits (the length of '9999'), `HH24` to two digits (the length of '23'), and `DDD` to three digits (the length of '366').
- The character elements `MONTH`, `MON`, `DAY`, and `DY` are padded with trailing blanks to the width of the longest full month name, the longest abbreviated month name, the longest full date name, or the longest abbreviated day name, respectively, among valid names determined by the values of `NLS_DATE_LANGUAGE` and `NLS_CALENDAR` parameters. For example, when `NLS_DATE_LANGUAGE` is `AMERICAN` and `NLS_CALENDAR` is `GREGORIAN` (the default), the largest element for `MONTH` is `SEPTEMBER`, so all values of the `MONTH` format element are padded to nine display characters. The values of the `NLS_DATE_LANGUAGE` and `NLS_CALENDAR` parameters are specified in the third argument to `TO_CHAR` and `TO_*` datetime functions or they are retrieved from the NLS environment of the current session.
- The character element RM is padded with trailing blanks to the length of 4, which is the length of 'viii'.
- Other character elements and spelled-out numbers (`SP`, `SPTH`, and `THSP`suffixes) are not padded.

The `FM` modifier suppresses the above padding in the return value of the `TO_CHAR` function.

FX

Format exact. This modifier specifies exact matching for the character argument and datetime format model of a `TO_DATE` function:

- Punctuation and quoted text in the character argument must exactly match (except for case) the corresponding parts of the format model.
- The character argument cannot have extra blanks. Without `FX`, Oracle ignores extra blanks.
- Numeric data in the character argument must have the same number of digits as the corresponding element in the format model. Without `FX`, numbers in the character argument can omit leading zeros.

  When `FX` is enabled, you can disable this check for leading zeros by using the `FM` modifier as well.

If any portion of the character argument violates any of these conditions, then Oracle returns an error message.

Format Model Examples

The following statement uses a date format model to return a character expression:

```
SELECT TO_CHAR(SYSDATE, 'fmDDTH') || ' of ' ||
       TO_CHAR(SYSDATE, 'fmMonth') || ', ' ||
       TO_CHAR(SYSDATE, 'YYYY') "Ides"
```

```
  FROM DUAL;

Ides
------------------
3RD of April, 2008
```

The preceding statement also uses the `FM` modifier. If `FM` is omitted, then the month is blank-padded to nine characters:

```
SELECT TO_CHAR(SYSDATE, 'DDTH') || ' of ' ||
   TO_CHAR(SYSDATE, 'Month') || ', ' ||
   TO_CHAR(SYSDATE, 'YYYY') "Ides"
  FROM DUAL;

Ides
-----------------------
03RD of April    , 2008
```

The following statement places a single quotation mark in the return value by using a date format model that includes two consecutive single quotation marks:

```
SELECT TO_CHAR(SYSDATE, 'fmDay') || '''s Special' "Menu"
  FROM DUAL;

Menu
-----------------
Tuesday's Special
```

Two consecutive single quotation marks can be used for the same purpose within a character literal in a format model.

Table 2-19 shows whether the following statement meets the matching conditions for different values of *`char`* and '*`fmt`*' using `FX` (the table named `table` has a column `date_column` of data type `DATE`):

```
UPDATE table
  SET date_column = TO_DATE(char, 'fmt');
```

Table 2-19 Matching Character Data and Format Models with the FX Format Model Modifier

char	'fmt'	Match or Error?
`'15/ JAN /1998'`	`'DD-MON-YYYY'`	`Match`
`' 15! JAN % /1998'`	`'DD-MON-YYYY'`	`Error`
`'15/JAN/1998'`	`'FXDD-MON-YYYY'`	`Error`

char	'fmt'	Match or Error?
'15-JAN-1998'	'FXDD-MON-YYYY'	Match
'1-JAN-1998'	'FXDD-MON-YYYY'	Error
'01-JAN-1998'	'FXDD-MON-YYYY'	Match
'1-JAN-1998'	'FXFMDD-MON-YYYY'	Match

Format of Return Values: Examples

You can use a format model to specify the format for Oracle to use to return values from the database to you.

The following statement selects the salaries of the employees in Department 80 and uses the `TO_CHAR` function to convert these salaries into character values with the format specified by the number format model '`$99,990.99`':

```
SELECT last_name employee, TO_CHAR(salary, '$99,990.99')
  FROM employees
  WHERE department_id = 80;
```

Because of this format model, Oracle returns salaries with leading dollar signs, commas every three digits, and two decimal places.

The following statement selects the date on which each employee from Department 20 was hired and uses the `TO_CHAR` function to convert these dates to character strings with the format specified by the date format model '`fmMonth DD, YYYY`':

```
SELECT last_name employee, TO_CHAR(hire_date,'fmMonth DD, YYYY') hiredate
  FROM employees
  WHERE department_id = 20;
```

With this format model, Oracle returns the hire dates without blank padding (as specified by `fm`), two digits for the day, and the century included in the year.

> **See Also:**
>
> "Format Model Modifiers" for a description of the `fm` format element

Supplying the Correct Format Model: Examples

When you insert or update a column value, the data type of the value that you specify must correspond to the column data type of the column. You can use format models to specify the format of a value that you are converting from one data type to another data type required for a column.

For example, a value that you insert into a `DATE` column must be a value of the `DATE` data type or a character string in the default date format (Oracle implicitly converts character strings in the default date format to the `DATE` data type). If the value is in another format, then you must use the `TO_DATE` function to convert the value to the `DATE` data type. You must also use a format model to specify the format of the character string.

The following statement updates `Hunold's` hire date using the `TO_DATE`function with the format mask 'YYYY MM DD' to convert the character string '2008 05 20' to a `DATE` value:

```
UPDATE employees
  SET hire_date = TO_DATE('2008 05 20','YYYY MM DD')
  WHERE last_name = 'Hunold';
```

String-to-Date Conversion Rules

The following additional formatting rules apply when converting string values to date values (unless you have used the `FX` or `FXFM` modifiers in the format model to control exact format checking):

- You can omit punctuation included in the format string from the date string if all the digits of the numerical format elements, including leading zeros, are specified. For example, specify 02 and not 2 for two-digit format elements such as MM, DD, and YY.
- You can omit time fields found at the end of a format string from the date string.
- You can use any non-alphanumeric character in the date string to match the punctuation symbol in the format string.
- If a match fails between a datetime format element and the corresponding characters in the date string, then Oracle attempts alternative format elements, as shown in Table 2-20.

Table 2-20 Oracle Format Matching

Original Format Element	Additional Format Elements to Try in Place of the Original
`'MM'`	`'MON'` and `'MONTH'`
`'MON`	`'MONTH'`

Original Format Element	Additional Format Elements to Try in Place of the Original
'MONTH'	'MON'
'YY'	'YYYY'
'RR'	'RRRR'

XML Format Model

The SYS_XMLAgg and SYS_XMLGen (deprecated) functions return an instance of type XMLType containing an XML document. Oracle provides the XMLFormatobject, which lets you format the output of these functions.

Table 2-21 lists and describes the attributes of the XMLFormat object. The function that implements this type follows the table.

See Also:

- SYS_XMLAGG for information on the SYS_XMLAgg function
- SYS_XMLGEN for information on the SYS_XMLGen function
- *Oracle XML Developer's Kit Programmer's Guide* for more information on the implementation of the XMLFormat object and its use

Table 2-21 Attributes of the XMLFormat Object

Attribute	Data Type	Purpose

enclTag	VARCHAR2(4000) or VARCHAR2(32767) Foot 1	The name of the enclosing tag for the result of the SYS_XMLAgg or SYS_XMLGen(deprecated) function. SYS_XMLAgg: The default is ROWSET. SYS_XMLGen: If the input to the function is a column name, then the default is the column name. Otherwise the default is ROW. When schemaType is set to USE_GIVEN_SCHEMA, this attribute also gives the name of the XMLSchema element.
schemaType	VARCHAR2(100)	The type of schema generation for the output document. Valid values are 'NO_SCHEMA' and 'USE_GIVEN_SCHEMA'. The default is 'NO_SCHEMA'.
schemaName	VARCHAR2(4000) or VARCHAR2(32767) Footref	The name of the target schema Oracle uses if the value of the schemaType is 'USE_GIVEN_SCHEMA'. If you specify schemaName, then Oracle uses the enclosing tag as the element name.

`targetNameSpace`	`VARCHAR2(4000)` or `VARCHAR2(32767)` Footref	The target namespace if the schema is specified (that is, `schemaType` is `GEN_SCHEMA_*`, or `USE_GIVEN_SCHEMA`)
`dburlPrefix`	`VARCHAR2(4000)` or `VARCHAR2(32767)` Footref	The URL to the database to use if `WITH_SCHEMA` is specified. If this attribute is not specified, then Oracle declares the URL to the types as a relative URL reference.
`processingIns`	`VARCHAR2(4000)` or `VARCHAR2(32767)` Footref	User-provided processing instructions, which are appended to the top of the function output before the element.

Footnote 1

The data type for this attribute is `VARCHAR2(4000)` if the initialization parameter `MAX_STRING_SIZE = STANDARD`, and `VARCHAR2(32767)` if `MAX_STRING_SIZE =EXTENDED`. See "Extended Data Types" for more information.

The function that implements the `XMLFormat` object follows:

```
STATIC FUNCTION createFormat(
     enclTag IN varchar2 := 'ROWSET',
     schemaType IN varchar2 := 'NO_SCHEMA',
     schemaName IN varchar2 := null,
     targetNameSpace IN varchar2 := null,
     dburlPrefix IN varchar2 := null,
     processingIns IN varchar2 := null) RETURN XMLGenFormatType
       deterministic parallel_enable,
  MEMBER PROCEDURE genSchema (spec IN varchar2),
  MEMBER PROCEDURE setSchemaName(schemaName IN varchar2),
  MEMBER PROCEDURE setTargetNameSpace(targetNameSpace IN varchar2),
  MEMBER PROCEDURE setEnclosingElementName(enclTag IN varchar2),
  MEMBER PROCEDURE setDbUrlPrefix(prefix IN varchar2),
  MEMBER PROCEDURE setProcessingIns(pi IN varchar2),
```

```
  CONSTRUCTOR FUNCTION XMLGenFormatType (
     enclTag IN varchar2 := 'ROWSET',
     schemaType IN varchar2 := 'NO_SCHEMA',
     schemaName IN varchar2 := null,
     targetNameSpace IN varchar2 := null,
     dbUrlPrefix IN varchar2 := null,
     processingIns IN varchar2 := null) RETURN SELF AS RESULT
      deterministic parallel_enable,
  STATIC function createFormat2(
      enclTag in varchar2 := 'ROWSET',
      flags in raw) return sys.xmlgenformattype
      deterministic parallel_enable
);
```