Table of Contents

Sometimes getting the exact date format you want is pretty tedious. This is some more tedious stuff that is not too important- until your boss needs a specific date format. When you think about it, Oracle has been putting together tools for business reporting for a long time- and it makes sense that they understand that sometimes you need a specific date format.

The format we discussed for To_Date allow some flexibility in the matching of the string and the date format.

For example: I can use a variety of punctuation marks between the date components and they are all accepted. I can use a 1 or 2 digit month or day value.

Select
```
     To_Date ('1969-5/6', 'YYYY/MM/DD')  as Col1
   , To_Date ('1969.5.6', 'YYYY/MM/DD')  as Col2
   , To_Date ('1969 5 6', 'YYYY/MM/DD')  as Col3
   from dual;
```
```
COL1                      COL2                      COL3
------------------------- ------------------------- -------------------------
06-MAY-69                 06-MAY-69                 06-MAY-69
```

1. FX

Sometimes that much flexibility is not desirable and you can add the FX Format Model Modifier to the format to say that you want the string to exactly match the format.

If I use the format model 'FXYYYY/MM/DD' none of those strings will match
```
SQL Error: ORA-01861: literal does not match format string
*Cause:    Literals in the input must be the same length as literals in
           the format string (with the exception of leading whitespace).  If the
           "FX" modifier has been toggled on, the literal must match exactly,
           with no extra whitespace.
```

The following strings will also cause an error

Here we need a 2 digit month and a 2 digit year
```
   select  To_Date ('1969/5/6', 'FXYYYY/MM/DD')  from dual;
```
```
SQL Error: ORA-01862: the numeric value does not match the length of the format item
*Cause:    When the FX and FM format codes are specified for an input date,
           then the number of digits must be exactly the number specified by the
           format code.  For example, 9 will not match the format specifier DD but
           09 will.
```
Here I have a different literal as the component separator and this fails.
```
   select  To_Date ('1969-05-06', 'FXYYYY/MM/DD')  from dual;
```
This version will work
```
   select  To_Date ('1969/05/06', 'FXYYYY/MM/DD')   from dual;
```

The FX modifier is a toggle switch- so you can turn this specificity on and off during the format. This get a bit dense so don't worry too much about this- but follow along.

A toggle switch is one that turns a setting on and the same toggle turns it off.

So the following says that the string has to match the pattern YYYY/MM/ exactly but the DD part is looser- we could use 1 or 2 digits.
```
   'FXYYYY/MM/FXDD'
```

The following are all acceptable matches

```
   To_Date ('1969/05/6',   'FXYYYY/MM/FXDD')
   To_Date ('1969/05/06',  'FXYYYY/MM/FXDD')
   To_Date ('1969/05/16',  'FXYYYY/MM/FXDD')
   To_Date ('1969/05/ 16', 'FXYYYY/MM/FXDD')
```

These fail

```
   To_Date ('1969-05/6', 'FXYYYY/MM/FXDD')   incorrect separator
   To_Date ('1969/5/06', 'FXYYYY/MM/FXDD')   month must be 2 digits
```

Try to figure this one out first

```
   'YYYYFX/FXMMFX/FXDD'

   To_Date ('69/05/06',   'YYYYFX/FXMMFX/FXDD')   OK
   To_Date ('1969/05/06', 'YYYYFX/FXMMFX/FXDD')   OK
   To_Date ('1969/5/6',   'YYYYFX/FXMMFX/FXDD')   OK
   To_Date ('1969-05-06', 'YYYYFX/FXMMFX/FXDD')   Fails
```

The format starts off being loose about the number of digits for the year

```
      'YYYY
```

Then it gets exact about the separator character

```
   'YYYYFX/
```

Then it gets loose about the number of digits in the month

```
   'YYYYFX/FXMM
```

Then it gets exact about the separator character

```
   'YYYYFX/FXMMFX/
```

Then it gets loose about the number of digits in the day

```
   'YYYYFX/FXMMFX/FXDD'
```

I doubt you would really want to do that- but you might want this model

```
   'FXYYYYFX/FXMMFX/FXDD'
```

When you read this, it says

must be exact about 4 digits for the year	`'FXYYYY`
can be loose about the separator character	`'FXYYYYFX/`
must be exact about 2 digits for the month	`'FXYYYYFX/FXMM`
can be loose about the separator character	`'FXYYYYFX/FXMMFX/FX`
must be exact about 2 digits for the day	`'FXYYYYFX/FXMMFX/FXDD'`

2. FM

The FM modifier is also a toggle switch- it specifies if Oracle should use
trailing blanks after variable length Month and Day values (which makes the date components align in "columns"
and leading zero before numbers.

Assuming the English language and the Gregorian calendar, the longest month name is 9 characters (September) and the longest day name is 9 characters (Wednesday).

If you are displaying dates which of these formats do you want?

```
Version A
Monday July 4, 2011
Wednesday September 7, 2011
Wednesday July 27, 2011
```

```
Version B
Monday    July        4, 2011
Wednesday September  7, 2011
Wednesday July       27, 2011
```

```
Version C
Monday    July      04, 2011
Wednesday September 07, 2011
Wednesday July      27, 2011
```

```
Version D
MONDAY    JULY 04, 2011
WEDNESDAY SEPTEMBER 07, 2011
WEDNESDAY JULY 27, 2011
```

```
   version A: 'FMDay Month DD, YYYY'
   version B: 'Day Month FMDD, YYYY'
   version C: 'Day Month DD, YYYY'
   version D: 'DAY FMMONTH FMDD, YYYY'
```

To read these, think of FM as the fill switch and the default is to fill; so the first time FM is used explicitly it means don't fill (I know, that sounds backwards!); the next time it is used in the format then it means fill.

Version A- don't fill and it stays in that mode

Version B- end fill with blanks until you get to FMDD, so the day numbers are not zero filled

Version C- the default which is to fill all components to a width of 9 for month and day name and zero fill the date

version D- the Day is filled, the month is not filled, and the day number is zero filled. And this is in all caps since the format is in caps.

Version A seems the best format for a date that would be printed on a letter; version C makes it easier to compare the date parts in a column-oriented report. I am not sure who would want version B.

You can use FM with numbers.

```
   Select
      To_char (345,     '99999.00')   as c999
    , To_char (345,     '00000.00')   as c000
   ,  To_char (345,  'FM99999.00')   as cFM999
   ,  To_char (345,  'FM00000.00')   as cFM000
   From dual;
```

```
C999      C000      CFM999    CFM000
--------- --------- --------- ---------
   345.00  00345.00 345.00    00345.00
```

The first column uses '99999.00' and has a single blank followed by 2 blanks for the first 2 digits of the format and then three digits for the value.

The second column uses '00000.00' and has a single blank followed by 5 digits for the value, using leading 0's as necessary.

The third column and fourth columns use the FM modifier which says do not fill with blanks so you do not get that extra blank at the start of the field.