文章
姚 鑫 · 五月 15, 2022 阅读大约需 7 分钟

第143章 SQL函数 TODATE（一）

第143章 SQL函数 TODATE（一）

将格式化字符串转换为日期的日期函数。

大纲

```
TO_DATE(date_string[,format])

TODATE(date_string[,format])
```

参数

- datestring - 要转换为日期的字符串。基础数据类型为 CHAR 或 VARCHAR2 的字符串日期表达式。
- format - 可选 — 对应于 datestring 的日期格式字符串。如果省略格式，DD MON YYYY& 是默认值；此默认值是可配置的。

描述

名称 TODATE 和 TODATE 是可互换的，并且支持 Oracle 兼容性。

TODATE 函数将各种格式的日期字符串转换为日期整数值，数据类型为 DATE。它用于输入各种字符串格式的日期，并将它们存储在标准的内部表示中。 TODATE 返回具有以下格式的日期：

nnnnn

其中 nnnnn 是介于 0（1840 年 12 月 31 日）和 2980013（9999 年 12 月 31 日）之间的正整数，包括端点。这表示天数。时间值被忽略。 1840 年 12 月 31 日之前的日期可以使用儒略日期来表示，如下所述。

也可以使用 TODATE() 方法调用从 ObjectScript 调用此函数：

```
$SYSTEM.SQL.Functions.TODATE(date_string,format)
```

在为字段提供默认值时，可以在数据定义中使用 TODATE 函数。例如：

```
CREATE TABLE mytest
(ID NUMBER(12,0) NOT NULL,
End_Year DATE DEFAULT TO_DATE('12-31-2018','MM-DD-YYYY') NOT NULL)
```

相关 SQL 函数

- TODATE 将格式化的日期字符串转换为日期整数。
- TOCHAR 执行相反的操作；它将日期整数转换为格式化的日期字符串。
- TOTIMESTAMP 将格式化的日期和时间字符串转换为标准时间戳。
- CAST 和 CONVERT执行DATE 数据类型转换。

日期字符串

第一个参数指定一个日期字符串文字。可以为输入 datestring
提供任何类型的日期字符串。每个字符都必须对应于格式字符串，但以下情况除外：
- 可以包含或省略前导零（不带分隔符的 datestring 除外）。
- 年份可以用两位数或四位数字指定。
-
月份名称可以完整指定，也可以指定为名称的前三个字母。只有前三个字母必须是正确的。月份名称不区分大小写。
- 附加到日期的时间值将被忽略。

格式化

第二个参数将日期格式指定为一串代码字符。

默认日期格式

如果未指定格式，TODATE 将使用默认格式解析日期字符串。默认格式为 DD MON YYYY。例如，'22 Feb 2018'。

此默认格式可在系统范围内配置，可使用以下任一方式：

```
$SYSTEM.SQL.Util.SetOption("ToDateDefaultFormat")
```

Management Portal TODATE 默认格式配置选项。从系统管理中，选择配置，然后选择 SQL
和对象设置，然后选择 SQL。可以查看和设置 TODATE 默认格式选项。

要确定当前设置，请调用 $SYSTEM.SQL.CurrentSettings()，它会显示 TODATE() 默认格式设置。

格式元素

格式是根据以下规则指定的一个或多个格式元素的字符串：
- 格式元素不区分大小写。
- 几乎任何顺序或数量的格式元素都是允许的。
- 格式字符串使用与 datestring 中的分隔符匹配的非字母数字分隔符（例如，空格、斜杠或连字符）分隔它们的元素。指定日期分隔符的这种使用不依赖于为您的 NLS 语言环境定义的 DateSeparator。
- 以下日期格式字符串不需要分隔符：MMDDYYYY、DDMMYYYY、YYYYMMDD 和
YYYYDDMM。还支持不完整的日期格式 YYYYMM，并假定 DD 值为 01。请注意，在这些情况下，必须为 MM 和 DD 值提供前导零。

下表列出了 format 参数的有效日期格式元素：

Format	Code Meaning
DD	两位数的月份日期 (01-31)。不需要前导零，除非格式不包含日期分隔符。
MM	两位数的月份编号（01-12；01 = JAN）。
MON	月份的缩写名称，由当前语言环境中的 MonthAbbr 属性指

Format	Code Meaning
	定。默认情况下，在英文中，这是月份名称的前三个字母。在其他语言环境中，月份缩写可能超过三个字母长和/或可能不包含月份名称的第一个字母。不允许使用句点字符。不区分大小写。
MONTH	月份的全名，由当前语言环境中的 MonthName 属性指定。默认值为： January February March April May June July August September October November December. Not case-sensitive.
YYYY	四位数年份。
YY	年份的最后两位数。 2 位数年份的前 2 位数默认为 19。
RRRR	四位数年份。
RR	L年份的最后 2 位数字。
DDD	一年中的某一天（见下文）。
J	儒略日期（自公元前 4712 年 1 月 1 日 (BCE) 以来的天数）。

TODATE 格式还可以包含 D（星期几号）、DY（星期几缩写）或
DAY（星期几名称）元素。但是，这些格式元素未经过验证或用于确定返回值。

单个日期元素的日期格式

可以将 DD、DDD、MM 或 YYYY
指定为完整的日期格式。因为这些格式字符串省略了月份、年份或同时省略了月份和年份，
将它们解释为指的是当前月份和年份：
- DD 返回当前年份当前月份中指定日期的日期。
- DDD 返回当前年份中指定日期的日期。
- MM 返回当前年份中指定月份的第一天的日期。
- YYYY - 返回指定年份当前月份第一天的日期。

以下嵌入式 SQL 示例显示了这些格式：

```
/// d ##class(PHA.TEST.SQLFunction).ToDate()
ClassMethod ToDate()
{
    NEW SQLCODE
    &sql(
        SELECT
            TO_DATE('300','DDD'),
            TO_DATE('24','DD')
        INTO
            :a,:b
    )
    if SQLCODE '= 0 {
        w !,"Error code ",SQLCODE
    } else {
        w "DDD format: ",a," = ",$ZDATE(a,1,,4),!
        w "DD format: ",b," = ",$ZDATE(b,1,,4)
    }
}


DHC-APP>d ##class(PHA.TEST.SQLFunction).ToDate()
DDD format: 66409 = 10/27/2022
DD format: 66253 = 05/24/2022


/// d ##class(PHA.TEST.SQLFunction).ToDate1()
```

```
ClassMethod ToDate1()
{
    NEW SQLCODE
    &sql(
        SELECT
            TO_DATE('8','MM'),
            TO_DATE('2018','YYYY')
        INTO
            :a,:b
    )
    if SQLCODE '= 0 {
        w !,"Error code ",SQLCODE
    } else {
        w "MM format: ",a," = ",$ZDATE(a,1,,4),!
        w "YYYY format: ",b," = ",$ZDATE(b,1,,4),!
        w "done"
    }
}


DHC-APP>d ##class(PHA.TEST.SQLFunction).ToDate1()
MM format: 66322 = 08/01/2022
YYYY format: 64769 = 05/01/2018
done
```

两位数年份转换（RR 和 RRRR 格式）

YY 格式通过简单地附加 19 将两位数的年份值转换为四位数字。因此 07 变为 1907，93 变为 1993。

RR 格式提供更灵活的两位数到四位数年份转换。此转换基于当年。如果当前年份在上半世纪（例如，2000 到 2050），则从 00 到 49 的两位数年份扩展到当前世纪的四位数字年份，从 50 到 2 位数年份99 年扩大到上个世纪的四位数年份。如果当前年份在世纪下半叶（例如，2050 年到 2099 年），则所有两位数年份都将扩展为当前世纪中的四位数年份。下面的嵌入式 SQL 示例显示了两位数年份到四位数年份的这种扩展：

```
/// d ##class(PHA.TEST.SQLFunction).ToDate2()
ClassMethod ToDate2()
{
    NEW SQLCODE
    &sql(
        SELECT
            TO_DATE('29 September 00','DD MONTH RR'),
            TO_DATE('29 September 18','DD MONTH RR'),
            TO_DATE('29 September 49','DD MONTH RR'),
            TO_DATE('29 September 50','DD MONTH RR'),
            TO_DATE('29 September 77','DD MONTH RR')
        INTO
            :a,:b,:c,:d,:e
    )
    if SQLCODE = 0 {
        w a," = ",$ZDATE(a,1,,4),!
        w b," = ",$ZDATE(b,1,,4),!
        w c," = ",$ZDATE(c,1,,4),!
        w d," = ",$ZDATE(d,1,,4),!
        w e," = ",$ZDATE(e,1,,4)
    } else {
        w "error:",SQLCODE
```

```
    }
}


DHC-APP>d ##class(PHA.TEST.SQLFunction).ToDate2()
58346 = 09/29/2000
64920 = 09/29/2018
76243 = 09/29/2049
40083 = 09/29/1950
49945 = 09/29/1977
```

RRRR 格式允许输入两位数和四位数字的混合年份。四位数年份不变（与 YYYY相同）。使用 RR 格式算法将两位数年份转换为四位数年份。这显示在以下嵌入式 SQL 示例中：

```
/// d ##class(PHA.TEST.SQLFunction).ToDate3()
ClassMethod ToDate3()
{
    n SQLCODE
    &sql(
        SELECT
            TO_DATE('29 September 2018','DD MONTH RRRR'),
            TO_DATE('29 September 18','DD MONTH RRRR'),
            TO_DATE('29 September 1949','DD MONTH RRRR'),
            TO_DATE('29 September 49','DD MONTH RRRR'),
            TO_DATE('29 September 1950','DD MONTH RRRR'),
            TO_DATE('29 September 50','DD MONTH RRRR')
        INTO
            :a,:b,:c,:d,:e,:f
    )
    if SQLCODE = 0 {
        w a," 4-digit = ",$ZDATE(a,1,,4),!
        w b," 2-digit = ",$ZDATE(b,1,,4),!
        w c," 4-digit = ",$ZDATE(c,1,,4),!
        w d," 2-digit = ",$ZDATE(d,1,,4),!
        w e," 4-digit = ",$ZDATE(e,1,,4),!
        w f," 2-digit = ",$ZDATE(f,1,,4)
    } else {
        w "error:",SQLCODE
    }
}



DHC-APP>d ##class(PHA.TEST.SQLFunction).ToDate3()
64920 4-digit = 09/29/2018
64920 2-digit = 09/29/2018
39718 4-digit = 09/29/1949
76243 2-digit = 09/29/2049
40083 4-digit = 09/29/1950
40083 2-digit = 09/29/1950
```

#Apple macOS #SQL #Caché

源

URL:
https://cn.community.intersystems.com/post/%E7%AC%AC143%E7%AB%A0-sql%E5%87%BD%E6%95%B0-todate%EF%BC%88%E4%B8%80%EF%BC%89