文章
姚 鑫 · 三月 4, 2021 阅读大约需 9 分钟

第三章 SQL语言元素（一）

第三章 SQL语言元素（一）

命令和关键字

InterSystems SQL命令（也称为SQL语句）以关键字开头，后跟一个或多个参数。其中一些参数可能是子句或函数，由它们自己的关键字标识。

- **InterSystems SQL命令没有命令终止符，除非在特殊情况下（例如SQL过程代码或触发代码），在这种情况下，SQL命令以单个分号（;）终止。否则，InterSystems SQL命令不需要或接受分号命令终止符**。在InterSystems SQL中指定分号命令终止符会导致SQLCODE -25错误。 TSQL的InterSystemsIRIS®数据平台实现（Transact-SQL）接受但不需要分号命令终止符。在将SQL代码导入Inter Systems SQL时，会去除分号命令终止符。
- **InterSystems SQL命令没有空格限制。如果命令项之间用空格隔开，则至少需要一个空格。** 如果命令项之间用逗号分隔，则不需要空格。算术运算符之前或之后不需要空格。可以在以空格分隔的项目之间，以逗号分隔的参数列表中的项目之间或在算术运算符之前或之后插入换行符或多个空格。

InterSystems SQL关键字包括命令名称，函数名称，谓词条件名称，数据类型名称，字段约束，优化选项和特殊变量。它们还包括AND，OR和NOT逻辑运算符，NULL列值指示符以及ODBC函数构造，例如{d dateval}和{fn CONCAT（str1，str2）}。

- 关键字不区分大小写。按照惯例，在本文档中，关键字用大写字母表示，但是InterSystems SQL没有大小写限制。
- 有许多关键字是SQL保留字。 InterSystems SQL仅保留那些不能明确解析的关键字。SQL保留字可用作分隔符。

函数:内在的和外在的

- 内在的:InterSystems SQL支持大量内在的(系统提供的)函数。这些函数包括数字函数、字符串函数以及日期和时间函数。

聚合函数是SQL固有函数，它计算列的所有值并返回单个聚合值。

- InterSystems SQL也可以支持用户提供的ObjectScript函数调用(外部函数)，如下所示:

这种写法只能在mac routine里，类文件里编译报错。

```
MySQL
     &sql(SELECT Name,$$MyFunc() INTO :n,:f FROM Sample.Person)
     WRITE "name is: ",n,!
     WRITE "function value is: ",f,!
     QUIT
MyFunc()
    SET x="my text"
```

```
    QUIT x
```

如果将用户提供的（外部）函数的使用配置为系统范围的选项，则该SQL语句只能调用用户提供的（外部）函数。默认为 “ 否 ” 。默认情况下，尝试调用用户提供的函数会发出SQLCODE -372错误。可以使用%SYSTEM.SQL类的SetAllowExtrinsicFunctions（）方法在系统范围内配置SQL对外部函数的使用。若要确定当前设置，请调用$SYSTEM.SQL.CurrentSettings（），该显示显示 “ 允许在SQL语句中使用外部函数 ” 选项。

不能使用用户提供的函数来调用%routine(名称以%字符开头的例程)。
尝试这样做会发出SQLCODE -373错误。

文字

InterSystems SQL文字具有以下语法：

```
literal ::=
number | string-literal
number ::=
 {digit}[.]digit{digit}[E[+|-]digit{digit}]
digit ::=
 0..9
string-literal ::=
std-string-literal | ObjectScript-empty-string
std-string-literal ::=
 ' {std-character-representation} '
std-character-representation ::=
nonquote-character | quote-symbol
quote-symbol ::=
 ''
ObjectScript-empty-string ::=
 ""
```

文字是一系列代表实际（文字）值的字符。它可以是数字或字符串。

- 数字不需要任何分隔符。它可以由数字0到9，小数点字符，指数符号以及加号和减号组成。数字中只能使用一个小数点字符。该小数点只能用于数字的基数部分，不能用于指数部分。小数点后不需要数字。允许前导零和尾随零。指数（科学符号）符号为字母E；大写字母E和小写字母E都可以接受，但是大写字母E是首选用法。加号或减号可以加一个底数或一个指数。多个加号和减号可以加上x个基数；
  SQL将这些符号视为运算符。 x只能有一个正负号。
  SQL将此符号视为文字的一部分。请勿在数字中使用逗号或空格。
- 字符串文字包含一对分隔符，其中包含任何类型的字符串。首选的定界符是单引号字符。要将分隔符指定为字符串中的文字，请将该字符加倍；例如： 'Mary's office'.

空字符串是文字字符串；它由两个单引号字符（''）表示。 NULL不是文字值；它表示没有任何值。

注意：在嵌入式SQL中，不允许在字符串文字中使用以##开头的一些字符序列，如 “ 使用嵌入式SQL ” 一章的 “ 文字值 ” 中所述。此限制不适用于其他SQL调用，例如动态SQL。

字符串分割符

使用单引号（'）字符作为字符串定界符。 SQL兼容性支持双引号字符（ “ ）的使用，但由于与定界标识符标准冲突，因此强烈建议不要使用。将一对双引号字符""解析为无效的定界标识符。并生成SQLCODE -1错误。

要在字符串中指定单引号字符作为字面字符，请指定一对这样的字符作为字面转义序列。
例如，'a 'normal' string'。

串联

双竖条(||)是首选的SQL连接操作符。
它可以用于连接两个数字、两个字符串或一个数字和一个字符串。

下划线(_)作为SQL连接操作符提供，以保证ObjectScript的兼容性。
此连接操作符只能用于连接两个字符串。

如果两个操作数都是字符串，并且两个字符串都具有相同的排序规则类型，则所得的级联字符串具有该排序规则类型。在所有其他情况下，连接的结果是排序类型EXACT。

NULL和空字符串

使用NULL关键字表示没有指定值。
在SQL中，NULL始终是表示数据值因任何原因未指定或不存在的首选方式。

SQL零长度字符串(空字符串)由两个单引号字符指定。
空字符串('')与空字符串是不同的。
空字符串是一个已定义的值，一个不包含字符的字符串，一个长度为0的字符串。
一个零长度的字符串在内部由非显示字符$CHAR(0)表示。

注意:不建议使用SQL零长度字符串作为字段输入值或字段默认值。
使用NULL表示数据值的缺失。

在SQL编码中应避免使用SQL零长度字符串。
但是，由于许多SQL操作都会删除末尾的空格，所以只包含空格字符(空格和制表符)的数据值可能会导致SQL的零长度字符串。

注意，不同的SQL length函数返回不同的值:length、CHAR_LENGTH和DATALENGTH返回SQL长度。
$LENGTH返回ObjectScript表示长度。
长度不计算尾随空格;
所有其他长度函数都计算末尾的空格。

null 处理

NOT NULL数据约束要求字段必须接收一个数据值;
不允许指定NULL而不是值。
这个约束不阻止使用空字符串值。

SELECT语句的WHERE或HAVING子句中的IS NULL谓词选择空值;
它不选择空字符串值。

IFNULL函数计算一个字段值，如果字段值为NULL，则返回第二个参数中指定的值。
它不会将空字符串值视为非空值。

COALESCE函数从提供的数据中选择第一个非空值。
它将空字符串值视为非空值。

当CONCAT函数或concenate操作符(||)连接一个字符串和一个NULL时，结果是NULL。
如下面的例子所示:

```
SELECT {fn CONCAT('fred',NULL)} AS FuncCat,    -- returns <null>
       'fred'||NULL AS OpCat                   -- returns <null>
```

AVG、COUNT、MAX、MIN和SUM聚合函数在执行操作时忽略NULL值。
(COUNT *统计所有行，因为不可能有一个所有字段都为空值的记录。)
SELECT语句的DISTINCT关键字在其操作中包含NULL;
如果指定的字段有空值，DISTINCT返回一个空行.

AVG、COUNT和MIN、聚合函数受空字符串值的影响。
MIN函数将空字符串视为最小值，即使存在值为0的行。
MAX和SUM聚合函数不受空字符串值的影响。

null 表达式

对大多数SQL函数提供NULL作为操作数将返回NULL。

任何以NULL作为操作数的SQL算术操作都返回NULL值。
因此,7 +零=零。
这包括二元运算加法(+)、减法(-)、乘法(*)、除法(/)、整数除法(\)和取模(#)，以及一元符号运算符加号(+)和减号(-)。

算术操作中指定的空字符串将被视为0(零)值。
除法(/)，整数除法(\)，或对空字符串(6/ '')取模(#)会导致<DIVIDE>错误。

NULL的长度

在SQL中，NULL的长度是没有定义的(它返回< NULL >)。
然而，空字符串的长度被定义为长度为0。
如下面的例子所示:

```
SELECT LENGTH(NULL) AS NullLen,   -- returns <null>
       LENGTH('') AS EmpStrLen    -- returns 0
```

如本例所示，SQL LENGTH函数返回SQL长度。

可以使用ASCII函数将SQL的零长度字符串转换为NULL，示例如下:

```
SELECT LENGTH(NULL) AS NullLen,                  -- returns <null>
       LENGTH({fn ASCII('')}) AS AsciiEmpStrLen, -- returns <null>
       LENGTH('') AS EmpStrLen                   -- returns 0
```

但是，对标准SQL的某些系统间IRIS扩展对NULL和空字符串的长度的处理是不同的。
$LENGTH函数返回这些值的InterSystems
IRIS内部表示:NULL表示为长度为0的已定义值，SQL空字符串表示为长度为0的字符串。
该功能与ObjectScript兼容。

```
SELECT $LENGTH(NULL) AS NullLen,    -- returns 0
$LENGTH('') AS EmpStrLen,           -- returns 0
$LENGTH('a') AS OneCharStrLen,      -- returns 1
$LENGTH(CHAR(0)) AS CharZero        -- returns 0
```

这些值的内部表示方式的另一个重要位置是%STRING、%SQLSTRING和%SQLUPPER函数，它们将空格附加到值中。
因为NULL实际上没有值，所以在它后面添加一个空格会创建一个长度为1的字符串。
但是一个空字符串确实有一个字符值，所以在它后面加上一个空格会创建一个长度为2的字符串。
如下面的例子所示:

```
SELECT CHAR_LENGTH(%STRING(NULL)) AS NullLen,  -- returns 1
CHAR_LENGTH(%STRING('')) AS EmpStrLen          -- returns 2
```

注意，这个例子使用的是CHARLENGTH，而不是LENGTH。
因为LENGTH函数删除了末尾的空格，所以LENGTH(%STRING(NULL))返回长度为0的字符串;
LENGTH(%STRING(''))返回长度为2的字符串，**因为%STRING追加的是前导空格，而不是尾随空格。**

ObjectScript和SQL

当SQL NULL输出到ObjectScript时，它由ObjectScript空字符串("")表示，长度为0的字符串。

当SQL零长度字符串数据输出到ObjectScript时，它由包含$CHAR(0)的字符串表示，该字符串长度为1。

```
/// d ##class(PHA.TEST.SQL).Null()
ClassMethod Null()
{
    &sql(SELECT NULL,'' INTO :a,:b)
    WRITE !,"NULL length: ",$LENGTH(a)         // returns 0
    WRITE !,"empty string length: ",$LENGTH(b) // returns 1
}
```

```
DHC-APP>d ##class(PHA.TEST.SQL).Null()

NULL length: 0
empty string length: 1
```

在ObjectScript中，没有值通常用空字符串("")表示。
当这个值被传递到嵌入式SQL中时，它会被视为空值，如下面的例子所示:

```
/// d ##class(PHA.TEST.SQL).Null1()
ClassMethod Null1()
{
    SET x=""
    SET myquery="SELECT NULL As NoVal,:x As EmpStr"
    SET tStatement=##class(%SQL.Statement).%New()
    SET qStatus=tStatement.%Prepare(myquery)
    IF qStatus'=1 {WRITE "%Prepare failed:" DO $System.Status.DisplayError(qStatus) Q
UIT}
    SET rset=tStatement.%Execute()
    WHILE rset.%Next() {
        WRITE "NoVal:",rset.%Get("NoVal")," length ",$LENGTH(rset.%Get("NoVal")),!
   // length 0
        WRITE "EmpStr:",rset.%Get("EmpStr")," length ",$LENGTH(rset.%Get("EmpStr")),!
   // length 0
    }
```

```
    WRITE "End of data"
}


DHC-APP>d ##class(PHA.TEST.SQL).Null1()
NoVal: length 0
EmpStr: length 0
End of data
```

如果指定了一个未定义的输入主机变量，嵌入式SQL将其值视为NULL。

当将NULL或空字符串值从嵌入式SQL传递到ObjectScript时，NULL被转换为长度为0的字符串，空字符串被转换为长度为1的字符串。
如下面的例子所示:

```
/// d ##class(PHA.TEST.SQL).Null2()
ClassMethod Null2()
{
    &sql(SELECT
        NULL,
        ''
        INTO :a,:b)
    WRITE !,"The length of NULL is: ",$LENGTH(a)         // length 0
    WRITE !,"The length of empty string is: ",$LENGTH(b) // length 1
}


DHC-APP>d ##class(PHA.TEST.SQL).Null2()

The length of NULL is: 0
The length of empty string is: 1
```

在下面的例子中，SQL的空字符串加上一个空格被传递为长度为2的字符串:

```
/// d ##class(PHA.TEST.SQL).Null3()
ClassMethod Null3()
{
    &sql(SELECT %SQLUPPER('')
        INTO :y )
    WRITE !,"SQL empty string length: ",$LENGTH(y)
}



DHC-APP> d ##class(PHA.TEST.SQL).Null3()

SQL empty string length: 2
```

#Caché #InterSystems IRIS #InterSystems IRIS for Health

源URL:

https://cn.community.intersystems.com/post/%E7%AC%AC%E4%B8%89%E7%AB%A0-sql%E8%AF%AD%E8%A8%80%E5%85%83%E7%B4%A0%EF%BC%88%E4%B8%80%EF%BC%89