文章
姚 鑫 · 三月 4, 2021 阅读大约需 11 分钟

第三章 SQL语言元素（二）

第三章 SQL语言元素（二）

算术运算符和函数

InterSystems SQL支持以下算术运算符:

- + 加法操作符。
  例如，17+7 = 24。
- – 减法运算符。
  例如，17-7等于10。
  注意，这些字符中的一对是InterSystems SQL注释指示器。
  因此，要指定两个或多个减法操作符或负号，必须使用空格或圆括号。
  例如，17- -7或17-(-7)等于24。

运算符	描述
+	加法操作符。
–	减法运算符。例如，17-7等于10。注意，这些字符中的一对是InterSystems SQL注释指示器。因此，要指定两个或多个减法操作符或负号，必须使用空格或圆括号。
*	乘法运算符。例如，17*7等于119。
/	除法操作符。例如:17/7 = 2.428571428571428571。
\	整数除法运算符。例如，17\7等于2。
#	模运算符。例如，17 #7等于3。注意，因为#字符也是一个有效的标识符字符，要将它用作模运算符，应该指定它与操作数之间用前后空格分隔
E	求幂(科学记数法)运算符。可以是大写或小写。例如:7E3 = 7000。指数过大会导致SQLCODE -7 “ 指数超出范围 ” 错误。例如1E309、7E308。
()	分组操作符。用于嵌套算术运算。除非使用了圆括号，否则在InterSystems SQL中算术操作的执行顺序是严格的从左到右的顺序。例如，17+7*2等于48，但17+(7 * 2)等于31。
||	连接运算符。例如，17||7等于177。

算术运算是对标准形式的数字进行的。

产生的数据类型

当对两个具有不同数据类型的数值执行算术运算时，结果数据类型确定如下:

对于加法(+)，减法(-)，整数除法(\)，和取模(#):

描述	NUMERIC	INTEGER	TINYINT	SMALLINT	BIGINT	DOUBLE
NUMERIC	NUMERIC	NUMERIC	NUMERIC	NUMERIC	NUMERIC	DOUBLE

描述	NUMERIC	INTEGER	TINYINT	SMALLINT	BIGINT	DOUBLE
INTEGER	NUMERIC	BIGINT	BIGINT	BIGINT	BIGINT	DOUBLE
TINYINT	NUMERIC	BIGINT	SMALLINT	INTEGER	BIGINT	DOUBLE
SMALLINT	NUMERIC	BIGINT	INTEGER	INTEGER	BIGINT	DOUBLE
BIGINT	NUMERIC	BIGINT	BIGINT	BIGINT	BIGINT	DOUBLE
DOUBLE	DOUBLE	DOUBLE	DOUBLE	DOUBLE	DOUBLE	DOUBLE

对于乘法(*)或除法(/):

描述	NUMERIC	INTEGER	TINYINT	SMALLINT	BIGINT	DOUBLE
NUMERIC	NUMERIC	NUMERIC	NUMERIC	NUMERIC	NUMERIC	DOUBLE
INTEGER	NUMERIC	NUMERIC	NUMERIC	NUMERIC	NUMERIC	DOUBLE
TINYINT	NUMERIC	NUMERIC	NUMERIC	NUMERIC	NUMERIC	DOUBLE
SMALLINT	NUMERIC	NUMERIC	NUMERIC	NUMERIC	NUMERIC	DOUBLE
BIGINT	NUMERIC	NUMERIC	NUMERIC	NUMERIC	NUMERIC	DOUBLE
DOUBLE	DOUBLE	DOUBLE	DOUBLE	DOUBLE	DOUBLE	DOUBLE

连接任意数据类型的两个数字将产生VARCHAR字符串。

在动态SQL中，可以使用SQL列元数据来确定结果集字段的数据类型。

运算符优先级

SQL-92标准在操作符优先级方面不精确;
关于这个问题的假设在不同的SQL实现中有所不同。
InterSystems SQL可以配置为支持任意一种优先级:
- 在InterSystems IRIS 2019.1及其后续版本中，InterSystems SQL默认支持算术运算符的ANSI优先级。
这是一个系统范围的配置设置。
当配置ANSI优先级时，"*"、"\"、"/"和"#"操作符的优先级高于"+"、"-"和"||"操作符。
优先级高的操作符在优先级低的操作符之前执行。
因此，3+3*5 = 18。
如果需要，可以使用括号覆盖优先级。
因此，(3+3)*5 = 30。

在安装InterSystems IRIS 2019.1时，支持默认的ANSI优先级;
升级InterSystems IRIS 2018.1到InterSystems IRIS 2019.1时，操作符优先级仍然配置为InterSystems IRIS 2018.1 default:严格从左到右的顺序。

- 在InterSystems IRIS 2018.1中，InterSystems SQL默认不提供算术运算符的优先级。
  默认情况下，InterSystems SQL严格按照从左到右的顺序执行算术表达式，没有操作符优先级。
  这与ObjectScript中使用的约定相同。
  因此，3+3*5 = 30。
  可以使用括号来强制要求的优先级。
  因此，3+(3*5)= 18。
  谨慎的开发人员应该使用圆括号来明确地表达他们的意图。

可以使用$SYSTEM.SQL.SetANSIPrecedence()方法在系统范围内配置任意一种SQL操作符优先级。
1 = ANSI优先;
0 =严格从左到右的计算。
要确定当前设置，调用$SYSTEM.SQL.CurrentSettings()。
更改此SQL选项将立即在系统范围内生效。
更改此选项将导致在系统范围内清除所有缓存的查询。

更改SQL优先级对ObjectScript没有影响。
ObjectScript总是严格遵循从左到右的算术运算符执行。

精度和等级

数字结果的精度(数字中存在的最大数字数)为:

- 使用以下算法确定加减:resultprecision=max(scale1, scale2)+ max(precision1-scale1, precision2-scale2)+1。
  当计算结果精度大于36时，将精度值设置为36。
- 乘法使用以下算法确定:resultprecision=min(36, precision1+precision2+1)。
- 除法(value1 / value2)通过以下算法确定:resultprecision=min(36, precision1-scale1 +scale2+max(6, scale1+precision2+1))。

以下数字结果的比例(最大小数位数):

- 加法或减法使用以下算法确定:resultscale=max(scale1, scale2)。
- 乘法使用如下算法确定:resultscale=min(17, scale1+scale2)。
- 除法(value1 / value2)通过如下算法确定:resultscale=min(17, max(6, scale1+precision2+1))。

算术和三角函数

InterSystems SQL支持以下算术函数:

代码	描述
ABS	返回数字表达式的绝对值。
CEILING	返回大于或等于数字表达式的最小整数。
EXP	返回数值表达式的对数指数(以e为底)值。
FLOOR	返回小于或等于数字表达式的最大整数。
GREATEST	从逗号分隔的数字列表中返回最大的数字。
ISNUMERIC	返回一个布尔码，指定表达式是否为有效数字。
LEAST	从逗号分隔的数字列表中返回最小的数字。
LOG	返回数字表达式的自然对数(以e为基数)值。
LOG10	返回数字表达式的以10为基数的日志值。
MOD	返回除法运算的模值(余数)。与#操作符相同。
PI	返回数值常量pi。
POWER	返回数值表达式的指定幂的值
ROUND	返回四舍五入(或截断)到指定数字数目的数字表达式。
SIGN	返回数值代码，指定数值表达式的计算结果是正、零还是负。
SQRT	返回数值表达式的平方根。
SQUARE	返回数值表达式的平方。
TRUNCATE	返回截断为指定数字数目的数字表达式。

InterSystems SQL支持下列三角函数。

代码	描述
ACOS	返回数值表达式的反余弦值。
ASIN	返回数字表达式的反正弦值。
ATAN	返回数值表达式的正切。
COS	返回数值表达式的余弦值。
COT	返回数值表达式的余切。
SIN	返回数值表达式的正弦值。
TAN	返回数值表达式的切线。
DEGREES	将弧度转换为角度。
RADIANS	将角度转换为弧度。

关系运算符

条件表达式的计算结果为布尔值。条件表达式可以使用以下关系运算符：

代码	描述
=	等于运算符。
!= <>	不等于运算符。这两种句法形式在功能上是相同的。
<	少于运算符。
>	大于运算符。

代码	描述
<=	小于或等于运算符。
>=	大于或等于运算符。

比较表格字段值时，这些相等运算符将使用字段的默认排序规则。 InterSystems IRIS默认值不区分大小写。比较两个文字时，比较区分大小写。

比较浮点数时，应避免使用等号运算符（等于或不等于）。浮点数（数据类型为%Library.Decimal和%Library.Double类）存储为二进制值，而不是固定精度的数字。在转换过程中，舍入运算可能会导致两个浮点数不完全相等，这些浮点数旨在表示相同的数字。使用小于/大于测试来确定两个浮点数是否“ 相同 ”至所需的精度。

包含并跟随运算符

InterSystems SQL还支持“ 包含 ”和“ 跟随 ”比较运算符：
- [包含运算符。返回包含操作数的所有值，包括等于该操作数的值。该运算符使用EXACT（区分大小写）排序规则。取反是NOT [。
  - Contains运算符确定一个值是否包含指定的字符或字符串。区分大小写。
  - %STARTWITH谓词条件确定值是否以指定的字符或字符串开头。它不区分大小写。
  - InterSystems SQL搜索可用于确定值是否包含指定的单词或短语。 SQL Search执行上下文感知匹配。它不区分大小写。
-]
跟随运算符。返回排序规则序列中跟随操作数的所有值。排除操作数值本身。该运算符使用字段的默认排序规则。InterSystems IRIS默认值不区分大小写。反之则不是]。

例如，SELECT Age FROM MyTable，其中Age]
88返回89或更大的值，但也返回9，因为在排序序列中9在88之后。
SELECT Age FROM MyTable WHERE Age > 88返回89以上;
它不会返回9。
字符串操作数，如' ABC '，排序在任何包含附加字符的字符串(如' ABCA ')之前;
因此，要从[操作符或>操作符中排除操作数字符串，必须指定整个字符串。
Name]‘ Smith,John ’包含‘ Smith,John ’不包含‘ Smith,John P. ’

逻辑运算符

SQL逻辑运算符用于评估为True或False的条件表达式中。这些条件表达式在SELECT语句WHERE和HAVING子句，CASE语句WHEN子句，JOIN语句ON子句和CREATE TRIGGER语句WHEN子句中使用。

非一元运算符

可以使用NOT一元逻辑运算符来指定条件的逻辑逆，如以下示例所示：

```
sql
 SELECT Name,Age FROM Sample.Person
WHERE NOT Age>21
ORDER BY Age
```

```
sql
 SELECT Name,Age FROM Sample.Person
WHERE NOT Name %STARTSWITH('A')
ORDER BY Name
```

可以将NOT运算符放在条件之前（如上所示）。或者，不能将NOT放在单字符运算符之前；例如，NOT <，NOT [等。请注意，NOT和它求反的单字符运算符之间必须没有空格。

AND和OR运算符

可以在一系列两个或多个条件下，在两个操作数之间使用AND和OR逻辑运算符。这些逻辑运算符可以用关键字或符号指定：

代码	描述

AND	&
OR	!

在符号运算符及其操作数之间不需要空格（尽管为了可读性建议使用空格）。关键字运算符前后需要空格。

这些逻辑运算符可以与NOT一元逻辑运算符一起使用，例如：WHERE Age<65 & NOT Age=21.。

以下两个示例使用逻辑运算符根据年龄安排计算。每三年对20至40岁的人群进行计算，每两年对40至64岁的人群进行计算，每年对65岁及65岁以上的人群进行计算。这些示例给出了相同的结果。第一个示例使用关键字，第二个示例使用符号：

```
SELECT Name,Age FROM Sample.Person
WHERE Age>20
      AND Age<40 AND (Age # 3)=0
      OR Age>=40 AND (Age # 2)=0
      OR Age>=65
ORDER BY Age
```

```
SELECT Name,Age FROM Sample.Person
WHERE Age>20
      & Age<40 & (Age # 3)=0
      ! Age>=40 & (Age # 2)=0
      ! Age>=65
ORDER BY Age
```

可以使用括号将逻辑运算符分组。这将建立分组级别；评估从最低的分组级别到最高的分组级别进行。在下面的第一个示例中，“ 与 ”条件仅应用于第二个“ 或 ”条件。它返回来自MA的任何年龄的人，以及来自NY的小于25岁的人：

```
SELECT Name,Age,Home_State FROM Sample.Person
WHERE Home_State='MA' OR Home_State='NY' AND Age < 25
ORDER BY Age
```

使用括号对条件进行分组会得出不同的结果。以下示例返回来自MA或NY的年龄小于25的人员：

```
SELECT Name,Age,Home_State FROM Sample.Person
WHERE (Home_State='MA' OR Home_State='NY') AND Age < 25
ORDER BY Age
```

- SQL执行使用短路逻辑。如果条件失败，将不会测试其余的AND条件。如果条件成功，则将不会测试其余的OR条件。
- 但是，由于SQL优化了WHERE子句执行，因此无法预测并且不应该依赖多个条件（在同一分组级别）的执行顺序。

注释

InterSystems
SQL支持单行注释和多行注释。注释文本可以包含任何字符或字符串，当然，指示注释结尾的字符除外。

注意：使用嵌入式SQL标记语法(&sql<marker>(...)<reversemarker>) 对SQL注释的内容强加了限制。如果使用标记语法，则SQL代码中的注释可能不包含字符序列“ ）<reversemarker> ”。

可以使用preparse（）方法返回去除注释的SQL DML语句。 preparse（）方法还用？替换每个查询参数。字符并返回这些参数的%List结构。以下示例中的preparse（）方法返回查询的解析版本，除去单行和多行注释以及空格：

```
/// d ##class(PHA.TEST.SQL).Null5()
ClassMethod Null5()
{
    SET myq=4
    SET myq(1)="SELECT TOP ? Name /* first name */, Age "
    SET myq(2)="   FROM Sample.MyTable -- this is the FROM clause"
    SET myq(3)="   WHERE  /* various conditions "
    SET myq(4)="apply */ Name='Fred' AND Age > 21 -- end of query"
    DO ##class(%SQL.Statement).preparse(.myq,.stripped,.args)
    WRITE stripped,!
    WRITE $LISTTOSTRING(args)
}
```

```
DHC-APP>d ##class(PHA.TEST.SQL).Null5()
 SELECT TOP ? Name , Age FROM Sample . MyTable WHERE Name = ? AND Age > ?
?,?,c,Fred,c,21
```

单行注释

单行注释由两个连字符前缀指定。注释可以在单独的行上，也可以与SQL代码显示在同一行上。当注释在同一行上跟随SQL代码时，至少一个空格必须将代码与双连字符注释运算符分隔开。注释可以包含任何字符，包括连字符，星号和斜杠。注释继续到该行的末尾。

下面的示例包含多个单行注释：

```
-- This is a simple SQL query
-- containing -- (double hyphen) comments
SELECT TOP 10 Name,Age, -- Two columns selected
Home_State -- A third column
FROM Sample.Person -- Table name
-- Other clauses follow
WHERE Age > 20 AND -- Comment within a clause
Age < 40
ORDER BY Age,  -- Comment within a clause
Home_State
-- End of query
```

多行注释

多行注释由/ * 开头定界符和 * /结束定界符指定。注释可以出现在一个或多个单独的行上，或者可以与SQL代码在同一行上开始或结束。注释定界符应与SQL代码分隔至少一个空格。注释可以包含任何字符，包括连字符，星号和斜杠，但* /字符对显然是例外。

下面的示例包含多个多行注释：

```
/* This is
   a simple
   SQL query. */
```

```
SELECT TOP 10 Name,Age /* Two fields selected */
FROM Sample.Person  /* Other clauses
could appear here */ ORDER BY Age
/* End of query */
```

当注释掉嵌入式SQL代码时，请始终在＆sql指令之前或括号内开始注释。下面的示例正确注释掉了两个嵌入式SQL代码块：

```
    SET a="default name",b="default age"
    WRITE "(not) Invoking Embedded SQL",!
    /*&sql(SELECT Name INTO :a FROM Sample.Person) */
    WRITE "The name is ",a,!
    WRITE "Invoking Embedded SQL (as a no-op)",!
    &sql(/* SELECT Age INTO :b FROM Sample.Person */)
    WRITE "The age is ",b
```

SQL代码保留为注释

嵌入式SQL语句可以保留为例程的.INT代码版本中的注释。这是通过设置$SYSTEM.SQL.SetRetainSQL（）方法完成的。若要确定当前设置，请调用$
SYSTEM.SQL.CurrentSettings（），它将显示“将SQL保留为注释”设置。默认值为1（“是”）。

将此选项设置为“是”以将SQL语句保留为例程的.INT代码版本中的注释。将此选项设置为“是”还会在注释文本中列出SQL语句使用的所有非%变量。这些列出的变量也应该在ObjectScript过程的PUBLIC变量列表中列出，并使用NEW命令重新初始化。

#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%BA%8C%EF%BC%89