文章
姚 鑫 · 五月 22, 2022 阅读大约需 5 分钟

第150章 SQL函数 TRUNCATE

第150章 SQL函数 TRUNCATE

标量数值函数，按指定位数截断一个数。

大纲

```
{fn TRUNCATE(numeric-expr,scale)}
```

参数

- numeric-expr - 要截断的数字。数字或数字表达式。
- scale - 计算结果为一个整数的表达式，该整数指定要截断的位数，从小数点开始计算。可以是零、正整数或负整数。如果比例是小数，会将其舍入为最接近的整数。

Truncate返回NUMERIC或DOUBLE数据类型。如果NUMERIC-EXPR的数据类型为DOUBLE，则TRUNCATE返回DOUBLE；否则返回NUMERIC。

描述

TRUNCATE通过从小数点开始按小数位数截断NUMERIC-EXPR。它不对数字进行四舍五入，也不添加填充零。在截断操作之前，将删除前导零和尾随零。

- 如果小数位数为正数，则在小数点右侧的位数处执行截断。如果小数位数等于或大于小数位数，则不会发生截断或零填充。
- 如果Scale为零，则该数字将被截断为整数。换句话说，在小数点右侧的零位数处执行截断；所有小数位和小数点本身都被截断。
- 如果小数位数为负数，则在小数点左侧的位数处执行截断。如果小数位数等于或大于数字中的整数位数，则返回零。
- 如果NUMERIC-EXPR为零(但表示为00.00、-0等)。TRUNCATE返回0(零)，不带小数位数，无论小数位数是多少。
- 如果NUMERIC-EXPR或SCALE为NULL，则TRUNCATE返回NULL。

TRUNCATE只能用作ODBC标量函数(使用花括号语法)。

ROUND可用于对数字执行类似的截断操作。Trim可用于对字符串执行类似的截断操作。

TRUNCATE, ROUND, and $JUSTIFY

TRUNCATE 和 ROUND 是执行类似操作的数值函数；它们都可用于减少数字的有效小数位数或整数位数。 ROUND 允许指定舍入（默认）或截断； TRUNCATE 不执行舍入。 ROUND 返回与 numeric-expr 相同的数据类型； TRUNCATE 返回 numeric-expr 作为数据类型 NUMERIC，除非 numeric-expr 是数据类型

DOUBLE，在这种情况下它返回数据类型 DOUBLE。

TRUNCATE 截断到指定数量的小数位数。如果截断导致尾随零，则保留这些尾随零。但是，如果 scale 大于 numeric-expr 规范形式的小数位数，则 TRUNCATE 不会填充零。

ROUND
舍入（或截断）到指定数量的小数位数，但其返回值始终是标准化的，删除尾随零。例如，ROUND(10.004,2) 返回 10，而不是 10.00。

当舍入到固定的小数位数很重要时使用 $JUSTIFY - 例如，在表示货币金额时。 $JUSTIFY 在舍入操作之后返回指定数量的尾随零。当要舍入的位数大于小数位数时，$JUSTIFY 补零。 $JUSTIFY 还右对齐数字，以便 DecimalSeparator 字符在一列数字中对齐。 $JUSTIFY 不会截断。

示例

以下两个示例都将数字截断为两位小数。第一个（使用动态 SQL）将比例指定为整数；第二个（使用嵌入式 SQL）将 scale 指定为解析为整数的主变量：

```
/// d ##class(PHA.TEST.SQLFunction).Truncate()
ClassMethod Truncate()
{
    s myquery = "SELECT {fn TRUNCATE(654.321888,2)} AS trunc"
    s tStatement = ##class(%SQL.Statement).%New()
    s qStatus = tStatement.%Prepare(myquery)
    if qStatus'=1 {WRITE "%Prepare failed:" DO $System.Status.DisplayError(qStatus) Q
UIT}
    s rset = tStatement.%Execute()
    d rset.%Display()
}


DHC-APP>d ##class(PHA.TEST.SQLFunction).Truncate()
trunc
654.32

1 Rows(s) Affected


/// d ##class(PHA.TEST.SQLFunction).Truncate1()
ClassMethod Truncate1()
{
    s x=2
    &sql(
        SELECT
            {fn TRUNCATE(654.321888,:x)}
        INTO
            :a
    )
    if SQLCODE '= 0 {
        w !,"Error code ",SQLCODE
    } else {
        w !,"truncated value is: ",a
    }
}
```

```
DHC-APP>d ##class(PHA.TEST.SQLFunction).Truncate1()

truncated value is: 654.32
```

这两个示例都返回 654.32（截断到小数点后两位）。

以下动态 SQL 示例指定的比例大于十进制位数：

```
/// d ##class(PHA.TEST.SQLFunction).Truncate2()
ClassMethod Truncate2()
{
    s myquery = "SELECT {fn TRUNCATE(654.321000,9)} AS trunc"
    s tStatement = ##class(%SQL.Statement).%New()
    s qStatus = tStatement.%Prepare(myquery)
    if qStatus'=1 {WRITE "%Prepare failed:" DO $System.Status.DisplayError(qStatus) Q
UIT}
    s rset = tStatement.%Execute()
    d rset.%Display()
}



DHC-APP>d ##class(PHA.TEST.SQLFunction).Truncate2()
trunc
654.321

1 Rows(s) Affected
```

它返回 654.321（ 在截断操作之前删除了尾随零；没有发生截断或零填充）。

以下动态 SQL 示例指定零比例：

```
/// d ##class(PHA.TEST.SQLFunction).Truncate3()
ClassMethod Truncate3()
{
    s myquery = "SELECT {fn TRUNCATE(654.321888,0)} AS trunc"
    s tStatement = ##class(%SQL.Statement).%New()
    s qStatus = tStatement.%Prepare(myquery)
    if qStatus'=1 {WRITE "%Prepare failed:" DO $System.Status.DisplayError(qStatus) Q
UIT}
    s rset = tStatement.%Execute()
    d rset.%Display()
}



DHC-APP>d ##class(PHA.TEST.SQLFunction).Truncate3()
trunc
654

1 Rows(s) Affected
```

它返回 654（所有小数位和小数点都被截断）。

以下动态 SQL 示例指定负比例：

```
/// d ##class(PHA.TEST.SQLFunction).Truncate4()
ClassMethod Truncate4()
{
    s myquery = "SELECT {fn TRUNCATE(654.321888,-2)} AS trunc"
    s tStatement = ##class(%SQL.Statement).%New()
    s qStatus = tStatement.%Prepare(myquery)
    if qStatus'=1 {WRITE "%Prepare failed:" DO $System.Status.DisplayError(qStatus) Q
UIT}
    s rset = tStatement.%Execute()
    d rset.%Display()
}


DHC-APP>d ##class(PHA.TEST.SQLFunction).Truncate4()
trunc
600

1 Rows(s) Affected
```

它返回 600（两个整数位已被截断并替换为零；请注意，未进行四舍五入）。

以下动态 SQL 示例指定与数字的整数部分一样大的负数：

```
/// d ##class(PHA.TEST.SQLFunction).Truncate5()
ClassMethod Truncate5()
{
    s myquery = "SELECT {fn TRUNCATE(654.321888,-3)} AS trunc"
    s tStatement = ##class(%SQL.Statement).%New()
    s qStatus = tStatement.%Prepare(myquery)
    if qStatus'=1 {WRITE "%Prepare failed:" DO $System.Status.DisplayError(qStatus) Q
UIT}
    s rset = tStatement.%Execute()
    d rset.%Display()
}



DHC-APP>d ##class(PHA.TEST.SQLFunction).Truncate5()
trunc
0

1 Rows(s) Affected
```

#SQL #Caché

源 URL: https://cn.community.intersystems.com/post/%E7%AC%AC150%E7%AB%A0-sql%E5%87%BD%E6%95%B0-truncate