

---

### 文章

[姚鑫](#) · 三月 14, 2022 阅读大约需 5 分钟

## 第八十一章 SQL函数 \$LISTBUILD

## 第八十一章 SQL函数 \$LISTBUILD

从字符串构建列表的列表函数。

### 大纲

`$LISTBUILD(element [,...])`

### 参数

- element - 任何表达式，或逗号分隔的表达式列表

### 描述

`$LISTBUILD` 接受一个或多个表达式，并为每个表达式返回一个包含一个元素的列表。

以下函数可用于创建列表：

- `$LISTBUILD`，它从多个字符串创建一个列表，每个元素一个字符串。
- `$LISTFROMSTRING`，它从包含多个分隔元素的单个字符串创建一个列表。
- `$LIST`，从现有列表中提取子列表。

`$LISTBUILD` 与其他 SQL

列表函数一起使用：`$LIST`、`$LISTDATA`、`$LISTFIND`、`$LISTFROMSTRING`、`$LISTGET`、`$LISTLENGTH` 和 `$LISTTOSTRING`。

注意：`$LISTBUILD` 和其他 `$LIST`

函数使用优化的二进制表示来存储数据元素。出于这个原因，等效性测试可能无法对某些 `$LIST` 数据按预期工作。在其他情况下可能被视为等效的数据可能具有不同的内部表示。例如，`$LISTBUILD(1)` 不等于 `$LISTBUILD('1')`。

出于同样的原因，`$LISTBUILD` 返回的列表字符串值不应用于使用分隔符的字符搜索和解析函数，例如 `$PIECE` 和 `$LENGTH` 的两个参数形式。由 `$LISTBUILD` 创建的列表中的元素不使用字符分隔符进行标记，因此可以包含任何字符。

### 示例

以下嵌入式 SQL 示例采用三个字符串并生成一个三元素列表：

```
/// d ##class(PHA.TEST.SQLFunction).Lb()  
ClassMethod Lb()
```

```
{
  s x = "Red"
  s y = "White"
  s z = "Blue"
  &sql(
    SELECT $LISTBUILD(:x, :y, :z)
    INTO :listout
  )
  if SQLCODE = 0 {
    w listout," length ", $LISTLENGTH(listout)
  } else {
    w "Error code:", SQLCODE
  }
}
```

```
DHC-APP>d ##class(PHA.TEST.SQLFunction).Lb()
RedWhiteBlue length 3
```

## 注意

### 省略参数

省略元素表达式会产生一个值为 NULL 的元素。例如，以下嵌入式 SQL 包含两个 \$LISTBUILD 语句，它们都生成一个三元素列表，其第二个元素具有未定义 (NULL) 值：

```
/// d ##class(PHA.TEST.SQLFunction).Lb1()
ClassMethod Lb1()
{
  s x = "Red"
  s y = "White"
  s z = "Blue"
  &sql(
    SELECT $LISTBUILD(:x, ,:z),
           $LISTBUILD(:x, '', :z)
    INTO :list1, :list2)
  if SQLCODE = 0 {
    w list1," length ", $LISTLENGTH(list1),!
    w list2," length ", $LISTLENGTH(list2)
  } else {
    w "Error code:", SQLCODE
  }
}
```

```
DHC-APP>d ##class(PHA.TEST.SQLFunction).Lb1()
RedBlue length 3
RedBlue length 3
```

此外，如果 \$LISTBUILD 表达式未定义，则相应的列表元素具有未定义的值。以下嵌入式 SQL 示例生成一个双元素列表，其第一个元素为“Red”，第二个元素具有未定义的值：

```
/// d ##class(PHA.TEST.SQLFunction).Lb2()
```

```
ClassMethod Lb2()  
{  
    &sql(  
        SELECT $LISTBUILD('Red',:z)  
        INTO :list1  
    )  
    if SQLCODE = 0 {  
        w list1," length ",$LISTLENGTH(list1)  
    } else {  
        w "Error code:",SQLCODE  
    }  
}
```

```
DHC-APP>d ##class(PHA.TEST.SQLFunction).Lb2()  
Red length 2
```

以下嵌入式 SQL 示例生成一个包含两个元素的列表。尾随逗号表示第二个元素具有未定义的值：

```
/// d ##class(PHA.TEST.SQLFunction).Lb3()  
ClassMethod Lb3()  
{  
    &sql(  
        SELECT $LISTBUILD('Red',)  
        INTO :list1  
    )  
    if SQLCODE = 0 {  
        w list1," length ",$LISTLENGTH(list1)  
    } else {  
        w "Error code:",SQLCODE  
    }  
}
```

```
DHC-APP>d ##class(PHA.TEST.SQLFunction).Lb3()  
Red length 2
```

## 不提供任何参数

调用不带参数的 \$LISTBUILD 函数会返回一个列表，其中包含一个数据值未定义的元素。这与 NULL 不同。以下是创建“空”列表的有效 \$LISTBUILD 语句：

```
/// d ##class(PHA.TEST.SQLFunction).Lb4()  
ClassMethod Lb4()  
{  
    &sql(  
        SELECT $LISTBUILD(),  
               $LISTBUILD(NULL)  
        INTO :list1, :list2  
    )  
    if SQLCODE = 0 {  
        ZZDUMP list1  
        w !,"length ",$LISTLENGTH(list1),!  
        ZZDUMP list2  
    }  
}
```

```

        w !,"length ", $LISTLENGTH(list2),!
    } else {
        w "Error code:", SQLCODE
    }
}

```

```
DHC-APP>d ##class(PHA.TEST.SQLFunction).Lb4()
```

```

0000: 01
length 1
.

0000: 02 01
length 1
..

```

以下是创建包含空字符串的列表元素的有效 \$LISTBUILD 语句：

```

/// d ##class(PHA.TEST.SQLFunction).Lb5()
ClassMethod Lb5()
{
    &sql(
        SELECT $LISTBUILD(''),
               $LISTBUILD(CHAR(0))
        INTO :list1, :list2
    )
    if SQLCODE = 0 {
        ZZDUMP list1
        w !,"length ", $LISTLENGTH(list1),!
        ZZDUMP list2
        w !,"length ", $LISTLENGTH(list2),!
    } else {
        w "Error code:", SQLCODE
    }
}

```

```
DHC-APP> d ##class(PHA.TEST.SQLFunction).Lb5()
```

```

0000: 03 01 00
length 1
...

0000: 03 01 00
length 1
...

```

## 嵌套List

列表的元素本身可能是一个列表。例如，以下语句生成一个三元素列表，其第三个元素是二元素列表 “Walnut,Pecan ”：

```

SELECT $LISTBUILD('Apple','Pear',$LISTBUILD('Walnut','Pecan'))

0x07014170706C650601506561721101080157616C6E75740701506563616E

```

### 连接List

使用 SQL 连接运算符 (||) 连接两个列表的结果是另一个列表。例如，以下 SELECT 项生成相同的列表 “ A,B,C ”：

```
SELECT $LISTBUILD('A','B','C') AS List,  
       $LISTBUILD('A','B') || $LISTBUILD('C') AS CatList
```

```
0x030141030142030143      ABC
```

在以下示例中，前两个选择项生成相同的二元素列表；第三个选择项导致 NULL（因为将 NULL 连接到任何内容都会导致 NULL）；第四个和第五个选择项产生相同的三元素列表：

```
SELECT  
  $LISTBUILD('A','B') AS List,  
  $LISTBUILD('A','B') || '' AS CatEStr,  
  $LISTBUILD('A','B') || NULL AS CatNull,  
  $LISTBUILD('A','B') || $LISTBUILD('') AS CatEList,  
  $LISTBUILD('A','B') || $LISTBUILD(NULL) AS CatNList
```

### Unicode

如果列表元素中的一个或多个字符是宽 (Unicode) 字符，则该元素中的所有字符都表示为宽字符。为了确保跨系统的兼容性，\$LISTBUILD 总是以相同的顺序存储这些字节，而不管硬件平台如何。宽字符表示为字节字符串。

[#SQL](#) [#Caché](#)

---

#### 源

URL:

<https://cn.community.intersystems.com/post/%E7%AC%AC%E5%85%AB%E5%8D%81%E4%B8%80%E7%AB%A0-sql%E5%87%BD%E6%95%B0-listbuild>