文章
姚 鑫 · 十一月 25, 2021 阅读大约需 4 分钟

第八十七章 SQL命令 VALUES

第八十七章 SQL命令 VALUES

指定字段中使用的数据值的INSERT/UPDATE子句。

大纲

```
(field1{,fieldn})
     VALUES (value1{,valuen})
```

参数

- field - 字段名或以逗号分隔的字段名列表。
- value - 值或以逗号分隔的值列表。
  每个值被赋给相应的字段。

描述

VALUES子句用于INSERT、UPDATE或INSERT or UPDATE语句中，以指定要插入到字段中的数据值。
通常:

- INSERT查询使用以下语法:

```
INSERT INTO tablename (fieldname1,fieldname2,...)
     VALUES (value1,value2,...)
```

- UPDATE查询使用以下语法:

```
UPDATE tablename (fieldname1,fieldname2,...)
     VALUES (value1,value2,...)
```

VALUES子句中的元素依次对应于表名后面指定的字段。
注意，如果在VALUES子句中只指定了一个value元素，则没有必要将元素括在括号中。

下面的嵌入式SQL示例显示了一个INSERT语句，它向 “ Employee ” 表添加了一行:

```
   &sql(INSERT INTO Employee (Name,SocSec,Telephone)
        VALUES("Boswell",333448888,"546-7989"))


   &sql(INSERT INTO Employee (Name,SocSec,Telephone)
        VALUES ('Boswell',333448888,'546-7989'))
```

插入和更新查询可以使用VALUES子句，而不需要在表名之后显式指定字段名列表。
为了省略表名后面的字段名列表，查询必须满足以下两个条件:
- values子句中指定的值的数量与表中字段的数量相同(不包括ID字段)。
- values子句中的值按字段的内部列号顺序列出，从列2开始。
列1总是为系统生成的ID字段保留，而不是在VALUES子句中指定。

例如，查询:

```
INSERT INTO Sample.Person VALUES (5,'John')
```

等价于查询:

```
INSERT INTO Sample.Person (Age,Name) VALUES (5,'John')
```

如果表 “ Sample.Person ” 正好有两个用户定义的字段。

在本例中，将值5分配给列号较低的字段，将值 “ John ” 分配给另一个字段。

VALUES子句可以指定数组的元素，如下面的嵌入式SQL示例所示：

```
   &sql( UPDATE Person(Tel)
        VALUES :per('tel',)
        WHERE ID = :id )
```

UPDATE查询还可以引用具有未指定最后一个下标的数组。INSERT使用数组元素的存在和不存在来为新创建的行赋值和默认值，而UPDATE使用数组元素的存在来指示应该更新相应的字段。例如，假设有六列的表使用以下数组：

```
emp("profile",2)="Smith"
emp("profile",3)=2
emp("profile",3,1)="1441 Main St."
emp("profile",3,2)="Cableton, IL 60433"
emp("profile",5)=NULL
emp("profile",7)=25
emp("profile","next")="F"
```

列1始终保留给ID字段，并且不是用户指定的。插入的 “ Employee ” 行将第2列 “ name ” 设置为 “ Smith ” ；将第3列 “ Address ” 设置为两行值；未指定第4列 “ Department ” ，因此将其设置为默认值；将第5列 “ Location ” 设置为NULL。由于相应的数组元素定义为空值，因此不使用 “ location ” 的默认值。数组元素 “ 7 ” 和 “ NEXT ” 与 “ Employee ” 表中的列号不对应，因此查询会忽略它们。下面是使用此数组的UPDATE语句：

```
  &sql(UPDsqlATE Employee
       VALUES :emp('profile',)
       WHERE Employee = 379)
```

根据上面的定义和数组值，该语句将更新row ID = 379的 “ Employee ” 行的 “ Name ” 、 “ Address ” 和 “ Location ” 字段的值。

然而，完全省略下标将导致SQLCODE -54错误:在VALUES之后需要数组指示符(省略了最后一个下标)。

你也可以使用一个数组引用的UPDATE查询，目标多行，例如:

```
  &sql(UPDATE Employee
       VALUES :emp('profile',)
       WHERE Type = 'PART-TIME')
```

VALUES子句变量不能使用点语法。
因此，下面的嵌入式SQL示例是正确的:

```
   SET sname = state.Name
   &sql(INSERT INTO StateTbl VALUES :sname)
```

以下是不正确的:

```
     &sql(INSERT INTO State VALUES :state.Name)
```

NULL和空字符串值是不同的。
为了向后兼容，旧数据中的所有空字符串(")值都被认为是NULL值。
在新数据中，空字符串以$CHAR(0)的形式存储在数据中。
通过SQL, NULL被引用为'NULL'。
例如:

```
INSERT INTO Sample.Person
(SSN,Name,Home_City) VALUES ('123-45-6789','Doe,John',NULL)
```

通过SQL，空字符串被引用为"(两个单引号)。
例如:

```
INSERT INTO Sample.Person
(SSN,Name,Home_City) VALUES ('123-45-6789','Doe,John','')
```

不能为ID字段插入NULL值。

示例

下面的嵌入式SQL示例将 “ Doe,John ” 的一条记录插入到示例中。
人表。
然后它选择这个记录，然后删除这个记录。
第二个SELECT确认删除。

```
ClassMethod Value()
{
    s x = "Doe,John",y = "123-45-6789",z = "Metropolis"
    s (a, b, c, d, e)=0
    n SQLCODE,%ROWCOUNT,%ROWID
```

```
    &sql(
        INSERT INTO Sample.Person
        (
            Name, SSN, Home_City
        )
        VALUES
        (
            :x, :y, :z
        )
    )
    if SQLCODE '= 0 {
        w !,"INSERT Error code ",SQLCODE
        q
    }
    &sql(
        SELECT Name, SSN, Home_City
            INTO :a, :b, :c
            FROM Sample.Person
            WHERE Name = :x
    )
    if SQLCODE '= 0 {
        w !,"Error code ",SQLCODE
    } else {
        w !,"After INSERT:"
        w !,"Name = ", a," SSN = ", b," City = ",c
        w !,"SQL code = ", SQLCODE," Number of rows = ", %ROWCOUNT
    }
    &sql(
        DELETE FROM Sample.Person
            WHERE Name = :x
    )
    &sql(
        SELECT Name,SSN
            INTO :d,:e
            FROM Sample.Person
            WHERE Name='Doe,John'
    )
    if SQLCODE <0 {
        w !,"Error code ",SQLCODE
    } else {
        w !,"After DELETE:"
        w !,"Name=",d," SSN=",e
        w !,"SQL code=",SQLCODE," Number of rows=",%ROWCOUNT
    }
}
```

#SQL #Caché

源URL:
https://cn.community.intersystems.com/post/%E7%AC%AC%E5%85%AB%E5%8D%81%E4%B8%83%E7%AB%A0-sql%E5%91%BD%E4%BB%A4-values