文章
姚 鑫 · 三月 19, 2021 阅读大约需 11 分钟

第十二章 使用嵌入式SQL（三）

第十二章 使用嵌入式SQL（三）

主机变量

主机变量是将文字值传入或传出嵌入式SQL的局部变量。
最常见的是，主机变量用于将本地变量的值作为输入值传递给Embedded SQL，或者将SQL查询结果值作为输出主机变量传递给Embedded SQL查询。

主机变量不能用于指定SQL标识符，例如架构名称，表名称，字段名称或游标名称。主机变量不能用于指定SQL关键字。

- 输出主机变量仅在嵌入式SQL中使用。它们在INTO子句中指定，INTO子句是仅嵌入式SQL支持的SQL查询子句。
- 输入主机变量可以在嵌入式SQL或动态SQL中使用。在动态SQL中，还可以使用“ ？ ”向SQL语句输入文字。输入参数。这 ” ？ ”语法不能在Embedded SQL中使用。

在嵌入式SQL中，可以在可以使用文字值的任何位置使用输入主机变量。使用SELECT或FETCH语句的INTO子句指定输出主机变量。

注意：当SQL NULL输出到ObjectScript时，它由一个ObjectScript空字符串（“ ”）表示，该字符串的长度为零。

要将变量或属性引用用作宿主变量，请在其前面加上一个冒号（:)。 嵌入式InterSystems SQL中的主机变量可以是以下之一：
- 一个或多个ObjectScript局部变量，例如：myvar，指定为以逗号分隔的列表。局部变量可以完全形成并且可以包含下标。像所有局部变量一样，它区分大小写，并且可以包含Unicode字母字符。
- 单个ObjectScript局部变量数组，例如：myvars（）。局部变量数组只能从单个表（而不是联接表或视图）中接收字段值。
- 对象引用，例如：oref.Prop，其中Prop是属性名称，带有或不带有前导%字符。这可以是简单属性或多维数组属性，例如：oref.Prop（1）。它可以是一个实例变量，例如：i%Prop或：i %% Data。属性名称可以定界。例如：Person."Home City".即使停用了对分隔标识符的支持，也可以使用分隔属性名称。多维属性可以包括：i%Prop（）和：m%Prop（）主机变量引用。对象引用主机变量可以包含任意数量的点语法级别；例如，例如，：Person.Address.City。

当oref.Prop用作过程块方法内的宿主变量时，系统会自动将oref变量（而不是整个oref.Prop引用）添加到PublicList并对其进行更新。

主机变量中的双引号指定文字字符串，而不是带分隔符的标识符。例如，：request.GetValueAt("PID:SetIDPID") or :request.GetValueAt("PID:PatientName(1).FamilyName").

主机变量应在ObjectScript过程的PublicList变量列表中列出，并使用NEW命令重新初始化。您可以配置InterSystems IRIS以便在注释文本中列出Embedded SQL中使用的所有主机变量。使用InterSystems SQL的注释部分对此进行了描述。

主机变量值具有以下行为：
-
输入主机变量永远不会被SQL语句代码修改。即使嵌入式SQL运行后，它们仍保留其原始值。但是，输入主机变量值在提供给SQL语句代码之前会被“ 轻度格式化 ”：有效数字值将去除前导和尾随零，单个前导加号和尾随小数点。时间

戳记值将除去尾随空格，以小数秒为单位的尾随零和（如果没有小数秒的话）尾随的小数点。
- 当SQLCODE = 0时，即返回有效行时，将设置INTO子句中指定的输出主机变量。如果执行SELECT语句或FETCH语句导致SQLCODE = 100（没有数据与查询匹配），则INTO子句中指定的输出主机变量将设置为null（“ ”）。如果在执行SELECT语句或FETCH语句之前未定义INTO变量，导致SQLCODE = 100，则该变量将保持未定义状态。主机变量值仅应在SQLCODE = 0时使用。在DECLARE ... SELECT ... INTO语句中，请勿在两个FETCH调用之间修改INTO子句中的输出主机变量，因为这可能会导致不可预测的查询结果。

在处理输出主机变量之前，必须检查SQLCODE值。仅当SQLCODE = 0时才应使用输出主机变量值。

当在INTO子句中使用逗号分隔的主机变量列表时，必须指定与选择项数量相同的主机变量数量（字段，集合函数，标量函数，算术表达式，文字）。宿主变量太多或太少都会在编译时导致SQLCODE -76基数错误。

在嵌入式SQL中使用SELECT *时，这通常是一个问题。例如，SELECT * FROM Sample.Person仅对以逗号分隔的15个主机变量列表有效（非隐藏列的确切数目，具体取决于表定义，该数目可能包含也可能不包含系统生成的RowID）（ID）列）。

因为列数可以更改，所以用单个宿主变量的INTO子句列表指定SELECT *通常不是一个好主意。使用SELECT *时，通常最好使用主机变量下标数组，例如：

```
/// d ##class(PHA.TEST.SQL).EmbedSQL9()
ClassMethod EmbedSQL9()
{
    NEW SQLCODE
    &sql(SELECT %ID,* INTO :tflds() FROM Sample.Person )
    IF SQLCODE<0 {
        WRITE "SQLCODE?? ",SQLCODE," ",%msg  QUIT
    } ELSEIF SQLCODE=100 {
        WRITE "??????"  QUIT
    }
    FOR i=0:1:25 {
        IF $DATA(tflds(i)) {
            WRITE "field ",i," = ",tflds(i),!
        }
    }
}
```

```
DHC-APP>d ##class(PHA.TEST.SQL).EmbedSQL9()
field 1 = 1
field 2 = 30
field 3 = 54536
field 4 = ReOrangYellow
field 6 = yaoxin
field 8 = 111-11-1117
field 9 = 13
field 11 = St Louis
field 12 = WI
field 13 = 889 Clinton Drive
field 14 = 78672
field 15 = Ukiah
field 16 = AL
field 17 = 9619 Ash Avenue
field 18 = 56589
```

本示例使用%ID返回RowID作为字段号1，无论RowID是否隐藏。 注意，在此示例中，字段编号下标可能不是连续序列；有些字段可能被隐藏并被跳过。包含NULL的字段以空字符串值列出。
**

退出嵌入式SQL后立即检查SQLCODE值是一种良好的编程习惯。仅当SQLCODE =
0时才应使用输出主机变量值。**

主机变量示例

在下面的ObjectScript示例中，Embedded
SQL语句使用输出主机变量将名称和归属状态地址从SQL查询返回到ObjectScript：

```
/// d ##class(PHA.TEST.SQL).EmbedSQL10()
ClassMethod EmbedSQL10()
{
    &sql(SELECT Name,Home_State
        INTO :CName,:CAddr
        FROM Sample.Person)
    IF SQLCODE<0 {
        WRITE "SQLCODE?? ",SQLCODE," ",%msg  QUIT
    } ELSEIF SQLCODE=100 {
        WRITE "??????"  QUIT
    }

    WRITE !,"Name is: ",CName
    WRITE !,"State is: ",CAddr
}
```

```
DHC-APP>d ##class(PHA.TEST.SQL).EmbedSQL10()

Name is: yaoxin
State is: WI
```

嵌入式SQL使用INTO子句指定主机变量：CName和：CAddr，以在局部变量CName中返回所选客户的姓名，并在局部变量CAddr中返回主目录状态。

下面的示例使用带下标的局部变量执行相同的操作：

```
/// d ##class(PHA.TEST.SQL).EmbedSQL11()
ClassMethod EmbedSQL11()
{
    &sql(SELECT Name,Home_State
    INTO :CInfo(1),:CInfo(2)
    FROM Sample.Person)
    IF SQLCODE<0 {
        WRITE "SQLCODE?? ",SQLCODE," ",%msg  QUIT
    } ELSEIF SQLCODE=100 {
        WRITE "??????"  QUIT
    }
    WRITE !,"Name is: ",CInfo(1)
    WRITE !,"State is: ",CInfo(2)
}
```

```
DHC-APP>d ##class(PHA.TEST.SQL).EmbedSQL11()

Name is: yaoxin
State is: WI
```

这些主机变量是带有用户提供的下标（：CInfo（1））的简单局部变量。但是，如果省略下标（：CInfo（））， 则InterSystems IRIS使用SqlColumnNumber填充主机变量下标数组，如下所述。

在下面的ObjectScript示例中，嵌入式SQL语句同时使用输入主机变量（在WHERE子句中）和输出主机变量（在INTO子句中）：

```
/// d ##class(PHA.TEST.SQL).EmbedSQL12()
ClassMethod EmbedSQL12()
{
    SET minval = 10000
    SET maxval = 50000
    &sql(SELECT Name,Salary INTO :outname, :outsalary
    FROM Sample.Employee
    WHERE Salary > :minval AND Salary < :maxval)
    IF SQLCODE<0 {
        WRITE "SQLCODE?? ",SQLCODE," ",%msg  QUIT
    } ELSEIF SQLCODE=100 {
        WRITE "??????"  QUIT
    }
    WRITE !,"Name is: ",outname
    WRITE !,"Salary is: ",outsalary
}
```

```
DHC-APP>d ##class(PHA.TEST.SQL).EmbedSQL12()

Name is: Chadwick,Phyllis L.
Salary is: 16377
```

以下示例在输入主机变量上执行 “ light normalization ” 。请注意，InterSystems IRIS将输入变量值视为字符串，并且不对其进行规范化，但是Embedded SQL将此数字规范化为65，以在WHERE子句中执行相等比较：

```
/// d ##class(PHA.TEST.SQL).EmbedSQL13()
ClassMethod EmbedSQL13()
{
    SET x="+065.000"
        &sql(SELECT Name,Age
        INTO :a,:b
        FROM Sample.Person
        WHERE Age=:x)
    IF SQLCODE<0 {
        WRITE "SQLCODE?? ",SQLCODE," ",%msg  QUIT
    } ELSEIF SQLCODE=100 {
        WRITE "??????"  QUIT
    }
    WRITE !,"Input value is: ",x
    WRITE !,"Name value is: ",a
    WRITE !,"Age value is: ",b
}
```

```
DHC-APP>d ##class(PHA.TEST.SQL).EmbedSQL13()
```

```
Input value is: +065.000
Name value is: Houseman,Martin D.
Age value is: 65
```

在下面的ObjectScript示例中，嵌入式SQL语句使用对象属性作为宿主变量：

```
   &sql(SELECT Name, Title INTO :obj.Name, :obj.Title
        FROM MyApp.Employee
        WHERE %ID = :id )
```

在这种情况下，obj必须是对具有可变（即可以修改）属性Name和Title的对象的有效引用。请注意，如果查询包含INTO语句并且没有返回任何数据（即SQLCODE为100），则执行查询可能会导致修改主机变量的值。

用列号下标的主机变量

如果FROM子句包含一个表，则可以为从该表中选择的字段指定带下标的主机变量；否则，可以为该表指定一个下标主机变量。例如，本地数组：myvar（）。 InterSystems IRIS使用每个字段的SqlColumnNumber作为数字下标填充本地数组。请注意，SqlColumnNumber是表定义中的列号，而不是选择列表序列。
（不能将带下标的宿主变量用于视图的字段。）

主机变量数组必须是省略了最低级别下标的局部数组。因此，:myvar(), :myvar(5,), and :myvar(5,2,)都是有效的主机变量下标数组。
- 主机变量下标数组可以用于INSERT，UPDATE或INSERT OR UPDATE语句VALUES子句中的输入。当在INSERT或UPDATE语句中使用时，主机变量数组使您可以定义在运行时而不是在编译时更新哪些列。
- 主机变量下标数组可以用于SELECT或DECLARE语句INTO子句中的输出。在下面的示例中显示了SELECT中的下标数组用法。

在下面的示例中，SELECT使用指定字段的值填充Cdata数组。 Cdata（）的元素对应于表列定义，而不是SELECT元素。因此，在Sample.Person中，“ 名称 ”字段是第6列，“ 年龄 ”字段是第2列，“ 出生日期 ”（DOB）字段是第3列：

```
/// d ##class(PHA.TEST.SQL).EmbedSQL14()
ClassMethod EmbedSQL14()
{
    &sql(SELECT Name, Age, DOB
        INTO :Cdata()
        FROM Sample.Person)
    IF SQLCODE<0 {
        WRITE "SQLCODE?? ",SQLCODE," ",%msg  QUIT
    } ELSEIF SQLCODE=100 {
        WRITE "??????"  QUIT
    }
    WRITE !,"Name is: ",Cdata(6)
    WRITE !,"Age is: ",Cdata(2)
    WRITE !,"DOB is: ",$ZDATE(Cdata(3),1)
}
```

```
DHC-APP>d ##class(PHA.TEST.SQL).EmbedSQL14()

Name is: yaoxin
Age is: 30
DOB is: 04/25/90
```

以下示例使用带下标的数组主机变量返回行的所有字段值：

```
/// d ##class(PHA.TEST.SQL).EmbedSQL15()
ClassMethod EmbedSQL15()
{
    &sql(SELECT * INTO :Allfields()
        FROM Sample.Person)
    IF SQLCODE<0 {
        WRITE "SQLCODE?? ",SQLCODE," ",%msg  QUIT
    } ELSEIF SQLCODE=100 {
        WRITE "??????"  QUIT
    }
    SET x=1
    WHILE x '="" {
        WRITE !,x," field is ",Allfields(x)
        SET x=$ORDER(Allfields(x))
    }
}
```

```
DHC-APP>d ##class(PHA.TEST.SQL).EmbedSQL15()

1 field is 1
2 field is 30
3 field is 54536
4 field is ReOrangYellow
6 field is yaoxin
8 field is 111-11-1117
9 field is 13
11 field is St Louis
12 field is WI
13 field is 889 Clinton Drive
14 field is 78672
15 field is Ukiah
16 field is AL
17 field is 9619 Ash Avenue
18 field is 56589
```

请注意，此WHILE循环使用$ORDER而不是简单的x = x + 1进行递增。这是因为在许多表（例如Sample.Person）中，可能存在隐藏的列。这些导致列号序列不连续。

如果SELECT列表包含不是该表中的字段的项，例如表达式或箭头语法字段，则INTO子句还必须包含逗号分隔的非数组主机变量。下面的示例组合了一个带下标的数组主机变量，以返回与定义的表列对应的值，而主机变量组合为返回与定义的表列不对应的值：

```
/// d ##class(PHA.TEST.SQL).EmbedSQL16()
ClassMethod EmbedSQL16()
{
    &sql(SELECT Name, Home_City, {fn NOW}, Age, ($HOROLOG-DOB)/365.25, Home_State
        INTO :Allfields(), :timestmp('now'), :exactage
        FROM Sample.Person)
    IF SQLCODE<0 {
        WRITE "SQLCODE?? ",SQLCODE," ",%msg  QUIT
    } ELSEIF SQLCODE=100 {
        WRITE "??????"  QUIT
    }
```

```
    SET x = $ORDER(Allfields(""))
    WHILE x '="" {
        WRITE !,x," field is ",Allfields(x)
        SET x=$ORDER(Allfields(x))
    }
    WRITE !,"date & time now is ",timestmp("now")
    WRITE !,"exact age is ",exactage
}


DHC-APP> d ##class(PHA.TEST.SQL).EmbedSQL16()

1 field is 1
2 field is 30
3 field is 54536
4 field is ReOrangYellow
6 field is yaoxin
8 field is 111-11-1117
9 field is 13
11 field is St Louis
12 field is WI
13 field is 889 Clinton Drive
14 field is 78672
15 field is Ukiah
16 field is AL
17 field is 9619 Ash Avenue
18 field is 56589
date & time now is 2021-03-13 16:00:40
exact age is 30.88295687885010267
```

请注意，非数组主机变量必须在数量和顺序上与非列SELECT项匹配。

将主机变量用作下标数组受以下限制：

- **只有在FROM子句的单个表中选择字段时，才可以使用带下标的列表。这是因为从多个表中选择字段时，SqlColumnNumber值可能会发生冲突。**
- 下标列表只能在选择表字段时使用。它不能用于表达式或聚合字段。这是因为这些选择列表项没有SqlColumnNumber值。

NULL和未定义的主机变量

如果指定未定义的输入主机变量，则嵌入式SQL将其值视为NULL。

```
/// d ##class(PHA.TEST.SQL).EmbedSQL17()
ClassMethod EmbedSQL17()
{
    NEW x
    &sql(SELECT Home_State,:x
        INTO :a,:b
        FROM Sample.Person)
    IF SQLCODE<0 {
        WRITE "SQLCODE?? ",SQLCODE," ",%msg  QUIT
    } ELSEIF SQLCODE=100 {
        WRITE "??????"  QUIT
    }
    WRITE !,"Home_State????: ",$LENGTH(a)
```

```
    WRITE !,"x????: ",$LENGTH(b)
}


DHC-APP>d ##class(PHA.TEST.SQL).EmbedSQL17()

Home_State????: 2
x????: 0
```

SQL NULL等效于ObjectScript “ ” 字符串（长度为零的字符串）。

如果将NULL输出到主机变量，则Embedded
SQL会将其值视为ObjectScript “ ” 字符串（零长度字符串）。例如，Sample.Person中的某些记录具有NULL
Spouse字段。执行此查询后：

```
/// d ##class(PHA.TEST.SQL).EmbedSQL18()
ClassMethod EmbedSQL18()
{
    &sql(SELECT Name,Spouse
        INTO :name, :spouse
        FROM Sample.Person
        WHERE Spouse IS NULL)
    IF SQLCODE<0 {
        WRITE "SQLCODE?? ",SQLCODE," ",%msg  QUIT
    } ELSEIF SQLCODE=100 {
        WRITE "??????"  QUIT
    }
    WRITE !,"Name: ",name," of length ",$LENGTH(name)," defined: ",$DATA(name)
    WRITE !,"Spouse: ",spouse," of length ",$LENGTH(spouse)," defined: ",$DATA(spouse
)
}


DHC-APP>d ##class(PHA.TEST.SQL).EmbedSQL18()

Name: xiaoli of length 6 defined: 1
Spouse:  of length 0 defined: 1
```

宿主变量spouse将设置为 “ ” （长度为零的字符串）以指示NULL值。因此，不能使用ObjectScript $DATA函数来确定
SQL字段是否为NULL。当传递带有NULL值的SQL字段的输出主机变量时，$DATA返回true（定义了变量）。

在极少数情况下，表字段包含SQL零长度字符串（''），例如，如果应用程序将字段显式设置为SQL
''字符串，则主机变量将包含特殊标记值$CHAR（0）（长度为1的字符串，仅包含一个ASCII
0字符），它是SQL零长度字符串的ObjectScript表示形式。强烈建议不要使用SQL零长度字符串。

下面的示例比较SQL NULL和SQL零长度字符串输出的主机变量：

```
/// d ##class(PHA.TEST.SQL).EmbedSQL19()
ClassMethod EmbedSQL19()
{
    &sql(SELECT '',Spouse
        INTO :zls, :spouse
        FROM Sample.Person
        WHERE Spouse IS NULL)
    IF SQLCODE<0 {
```

```
        WRITE "SQLCODE?? ",SQLCODE," ",%msg  QUIT
    } ELSEIF SQLCODE=100 {
        WRITE "??????"  QUIT
    }
    WRITE "In ObjectScript"
    WRITE !,"ZLS is of length ",$LENGTH(zls)," defined: ",$DATA(zls)
    /* Length=1, Defined=1 */
    WRITE !,"NULL is of length ",$LENGTH(spouse)," defined: ",$DATA(spouse)
    /* Length=0, Defined=1 */
}


DHC-APP>d ##class(PHA.TEST.SQL).EmbedSQL19()
In ObjectScript
ZLS is of length 1 defined: 1
NULL is of length 0 defined: 1
```

请注意，此主机变量NULL行为仅在基于服务器的查询（嵌入式SQL和动态SQL）中为true。在ODBC和JDBC中，使用ODBC或JDBC接口显式指定NULL值。

主机变量的有效性

- **嵌入式SQL永远不会修改输入主机变量。**
- **仅当SQLCODE = 0时，输出主机变量才在Embedded SQL之后可靠地有效。**

例如，以下OutVal的用法不可靠:

```
/// d ##class(PHA.TEST.SQL).EmbedSQL20()
ClassMethod EmbedSQL20()
{
InvalidExample
    SET InVal = "1234"
    SET OutVal = "None"
    &sql(SELECT Name
        INTO :OutVal
        FROM Sample.Person
        WHERE %ID=:InVal)
    IF OutVal="None" {
        WRITE !,"??????"
        WRITE !,"SQLCODE=",SQLCODE
    } ELSE {
        WRITE !,"Name is: ",OutVal
    }
}



DHC-APP>d ##class(PHA.TEST.SQL).EmbedSQL20()

??????
SQLCODE=100
```

调用嵌入式SQL之前设置的OutVal的值在从嵌入式SQL返回之后不应该被IF命令引用。

相反，应该使用SQLCODE变量编写如下示例:

```
/// d ##class(PHA.TEST.SQL).EmbedSQL21()
ClassMethod EmbedSQL21()
{
ValidExample
    SET InVal = "1234"
    &sql(SELECT Name
        INTO :OutVal
        FROM Sample.Person
        WHERE %ID=:InVal)
    IF SQLCODE'=0 {
        SET OutVal="None"
        IF OutVal="None" {
        WRITE !,"??????"
        WRITE !,"SQLCODE=",SQLCODE }
    } ELSE {
        WRITE !,"Name is: ",OutVal
    }
}


DHC-APP>d ##class(PHA.TEST.SQL).EmbedSQL21()

??????
SQLCODE=100
```

嵌入式SQL将SQLCODE变量设置为0，以指示成功地检索输出行。
SQLCODE值为100表示没有找到与SELECT条件匹配的行。
SQLCODE负数表示SQL错误条件。

主机变量和程序块

如果嵌入式SQL在过程块内，则所有输入和输出主机变量必须是公共的。可以通过在过程块开始处的PUBLIC部分中声明它们，或用一个初始%字符命名它们（自动使它们公开）来完成它们。但是请注意，用户定义的%主机变量是自动公开的，但不是自动更新的。用户有责任根据需要对这些变量执行NEW。如嵌入式SQL变量中所述，某些SQL%变量（例如%ROWCOUNT，%ROWID和%msg）既自动公开又自动更新。必须将SQLCODE声明为public。

在以下过程块示例中，主机变量zip，city和state以及SQLCODE变量被声明为PUBLIC。 SQL系统变量%ROWCOUNT，%ROWID和%msg已经公开，因为它们的名称以%字符开头。然后，过程代码对SQLCODE，其他SQL系统变量和状态局部变量执行NEW。

```
/// d ##class(PHA.TEST.SQL).EmbedSQL22()
ClassMethod EmbedSQL22()
{
UpdateTest(zip,city)
    [SQLCODE,zip,city,state] PUBLIC {
    NEW SQLCODE,%ROWCOUNT,%ROWID,%msg,state
    SET state="MA"
    &sql(UPDATE Sample.Person
        SET Home_City = :city, Home_State = :state
        WHERE Home_Zip = :zip)
    IF SQLCODE<0 {
        WRITE "SQLCODE error ",SQLCODE," ",%msg  QUIT
    }
        QUIT %ROWCOUNT
```

```
    }
}
```

#SQL #Caché #InterSystems IRIS #InterSystems IRIS for Health

源URL:
https://cn.community.intersystems.com/post/%E7%AC%AC%E5%8D%81%E4%BA%8C%E7%AB%A0-%E4%BD%BF%E7%94%A8%E5%B5%8C%E5%85%A5%E5%BC%8Fsql%EF%BC%88%E4%B8%89%EF%BC%89