

## 文章

姚鑫 · 九月 9, 2021 阅读大约需 7 分钟

## 第十一章 SQL命令 CREATE PROCEDURE (二)

## 第十一章 SQL命令 CREATE PROCEDURE (二)

## characteristics

用于创建方法的特征与用于创建查询的特征不同。

如果指定的特征无效，系统将生成SQLCODE -47错误。  
指定重复的特征将导致SQLCODE -44错误。

可用的方法特征关键字如下：

方法关键字

FOR className

FINAL

PRIVATE

RESULT SETS , DYNAMIC RESULT SETS [n]

RETURNS datatype

SELECTMODE mode

可用的查询特征关键字如下：

Query查询关键字

CONTAINID integer

FOR className

FINAL

RESULTS (resultset)

## 含义

指定要在其中创建方法的类的名称。如果这个类不存在，它将被创建。还可以通过限定方法名来指定类名。FOR子句中指定的类名通过限定方法名重写指定的类名。如果使用FOR my.class语法指定类名，IRIS将用Sqlname=procname定义类方法。因此，该方法应该作为my.procname()调用(而不是my.classprocname())。

指定子类不能重写该方法。默认情况下，方法不是final。FINAL关键字由子类继承。

指定该方法只能由它自己的类或子类的其他方法调用。默认情况下，方法是公共的，可以不受限制地调用。这个限制由子类继承。

指定创建的方法将包含ReturnResultsets关键字。这一特征短语的所有形式都是同义词。

指定调用该方法返回的值的数据类型。如果省略RETURNS，则该方法不能返回值。这个规范由子类继承，并且可以由子类修改。该数据类型可以指定类型参数，如MINVAL、MAXVAL和SCALE。例如RETURNS DECIMAL(19,4)。注意，当返回一个值时，IRIS会忽略数据类型的长度；例如，RETURNS VARCHAR(32)可以接收由调用方法返回的任意长度的字符串。

仅当LANGUAGE为SQL(默认)时使用。当指定时，IRIS将#SQLCOMPILE SELECT=mode语句添加到相应的类方法中，从而生成使用指定的SELECTMODE在方法中定义的SQL语句。可能的模式值是LOGICAL、ODBC、RUNTIME和DISPLAY。默认为LOGICAL。

## 含义

指定返回ID的字段(如果有的话)。将CONTAINID设置为返回ID的列的编号，如果没有列返回ID，则设置为0。IRIS不验证命名字段是否实际包含ID，因此此处的用户错误会导致数据不一致。

指定要在其中创建方法的类的名称。如果这个类不存在，它将被创建。还可以通过限定方法名来指定类名。FOR子句中指定的类名通过限定方法名重写指定的类名。

指定子类不能重写该方法。默认情况下，方法不是final。FINAL关键字由子类继承。

按照查询返回的顺序指定数据字段。如果指定RESULTS

Query查询关键字

含义

子句，则必须将查询返回的所有字段用括号括起来的逗号分隔列表列出。在SQLCODE -76基数不匹配错误中，指定比查询结果返回的字段少或多。为每个字段指定一个列名(将用作列标题)和一个数据类型。如果使用SQL语言，则可以省略RESULTS子句。如果忽略RESULTS子句，则会在类编译期间自动生成ROWSPEC。

SELECTMODE mode

指定用于编译查询的模式。可能的值是LOGICAL、ODBC、RUNTIME和DISPLAY。默认是RUNTIME。

SELECTMODE子句用于SELECT查询操作以及INSERT和UPDATE操作。

它指定编译时选择模式。

为SELECTMODE指定的值添加在ObjectScript类方法代码的开头，如:#SQLCompile Select=mode。

- 在SELECT查询中，SELECTMODE指定返回数据的模式。

如果模式值为LOGICAL，则返回逻辑(内部存储)值。

例如，日期以\$HOROLOG格式返回。

如果模式值为ODBC，则应用逻辑到ODBC的转换，并返回ODBC格式值。

如果模式值为DISPLAY，则应用逻辑到显示的转换，并返回显示格式值。

如果mode值为RUNTIME，则可以在执行时设置显示模式(LOGICAL、ODBC或display)。

- 在INSERT或UPDATE操作中，SELECTMODE

RUNTIME选项支持将输入数据值从显示格式(display或ODBC)自动转换为逻辑存储格式。

只有当SQL代码执行时的选择模式设置为LOGICAL(这是所有

SQL执行接口的默认设置)时，才会应用这个已编译的从显示到逻辑的数据转换代码。

RESULTS子句指定查询的结果。

RESULTS子句中的SQL数据类型参数被转换为查询的ROWSPEC中相应的 IRIS数据类型参数。

例如，RESULTS子句RESULTS (Code VARCHAR(15))生成ROWSPEC规范:ROWSPEC = "Code:%Library.String(MAXLEN=15)"。

## LANGUAGE

指定过程代码语言的关键字子句。可用的选项是:

- 语言OBJECTSCRIPT(用于OBJECTSCRIPT)或语言SQL。过程代码在codebody中指定。

- Language Java、Language Python或Language

DotNet用于使用这些语言之一调用外部存储过程的SQL过程。外部存储过程的语法如下:

```
LANGUAGE langname EXTERNAL NAME external-routine-name
```

其中，langname是JAVA、PYTHON或DOTNET，而external-routine-

name是一个引号括起来的字符串，包含指定语言中的外部例程的名称。

SQL过程调用现有的例程;

不能在CREATE PROCEDURE语句中用这些语言编写代码。

这些语言中的存储过程库存储在IRIS外部，因此不必在IRIS内打包、导入或编译。

下面是一个CREATE过程调用现有JAVA外部存储过程的示例:

```
CREATE PROCEDURE updatePrice (item_name VARCHAR, new_price INTEGER)
LANGUAGE JAVA
EXTERNAL NAME 'Orders.updatePrice'
```

如果省略LANGUAGE子句，则默认为SQL。

## codebody

要创建的方法或查询的程序代码。可以在SQL或ObjectScript中指定此代码。使用的语言必须与language子句匹配。

但是，ObjectScript中指定的代码可以包含嵌入式SQL。IRIS使用提供的代码来生成方法或查询的实际代码。

- SQL程序代码以BEGIN关键字开头，后面跟着SQL代码本身。

在每个完整的SQL语句的末尾，指定一个分号(;)。

一个查询只包含一条SQL语句——一条SELECT语句。

还可以创建插入、更新或删除数据的过程。

SQL程序代码以END关键字结束。

输入参数在SQL语句中作为主机变量指定，形式为:name。

(注意，在SQL代码中不应该使用问号(?)来指定输入参数。

过程将成功构建，但在调用过程时，不能传递这些参数或接受默认值。)

- ObjectScript程序代码用花括号括起来:{code}。  
代码行必须缩进。  
如果指定了，标签或#include预处理器命令必须以冒号作为前缀，并出现在第一列，如下所示:

```
CREATE PROCEDURE SP123()  
    LANGUAGE OBJECTSCRIPT  
{  
:Top  
:#Include %occConstant  
    WRITE "Hello World"  
    IF 0=$RANDOM(2) { GOTO Top }  
    ELSE {QUIT $$$OK }  
}
```

系统自动包含%occlude。

如果程序代码包含 IRIS Macro Preprocessor语句(# commands, ## functions, 或\$\$\$ Macro references)，则这些语句的处理和扩展是过程方法定义的一部分，并在方法编译时进行处理和扩展。

IRIS在生成过程时提供额外的代码行，该过程将SQL嵌入到ObjectScript “包装器” 中，提供过程上下文处理程序，并处理返回值。

下面是iris生成的包装器代码的示例:

```
NEW SQLCODE,%ROWID,%ROWCOUNT,title  
&sql(  
    -- code_body  
)  
QUIT $GET(title)
```

如果指定的代码是OBJECTSCRIPT，则必须显式定义 “包装器” (该NEWs变量并使用QUIT val在完成时返回一个值。

## 示例

下面的示例分为使用SQL codebody的示例和使用ObjectScript codebody的示例。

### 使用SQL代码的示例

下面的示例创建了一个名为PersonStateSP的简单查询，该查询作为存储过程公开。

它没有声明任何参数，并接受特征和语言的默认值:

```
ClassMethod CreateProcedure()
```

```
{
    &sql(
        CREATE PROCEDURE PersonStateSP()
        BEGIN
            SELECT Name,Home_State FROM Sample.Person ;
        END
    )
    if SQLCODE=0 {
        w !,"???????"
    } elseif SQLCODE=-361 {
        w !,"?????????"
    } else {
        w !,"SQL ?? ",SQLCODE
    }
}
```

可以转到Management Portal，选择Classes选项，然后选择SAMPLES名称空间。  
在这里，将找到上述示例创建的存储过程:User.procPersonStateSP.cls。

```
Class User.procPersonStateSP Extends %Library.RegisteredObject [ ClassType = "", DdlAllowed, Owner = {yx}, Not ProcedureBlock ]
{

Query PersonStateSP() As %Library.SQLQuery [ SqlName = PersonStateSP, SqlProc ]
{
    SELECT Name,Home_State FROM Sample.Person
}

}
```

在重新运行上面的程序示例之前，可以从这个显示中删除这个过程。  
当然，可以使用DROP PROCEDURE来删除一个过程：

```
ClassMethod CreateProcedure1()
{
    &sql(
        DROP PROCEDURE PersonStateSP
    )
    if SQLCODE=0 {
        w !,"???????"
    } elseif SQLCODE=-362 {
        w !,"?????????"
    } else {
        w !,"SQL??r: ",SQLCODE
    }
}
```

下面的示例创建一个更新数据的过程。它使用CREATE PROCEDURE在Sample.Employee类中生成方法UpdateSalary：

```
CREATE PROCEDURE UpdateSalary ( IN SSN VARCHAR(11), IN Salary INTEGER )
```

---

```
FOR Sample.Employee
BEGIN
    UPDATE Sample.Employee SET Salary = :Salary WHERE SSN = :SSN;
END
```

### 使用ObjectScript代码的示例

下面的示例创建生成随机大写字母的RandomLetterSP()存储过程方法。然后，可以在SELECT语句中将此方法作为函数调用。提供了一个删除RandomLetterSP()方法的删除过程。

```
CREATE PROCEDURE RandomLetterSP()
RETURNS INTEGER
LANGUAGE OBJECTSCRIPT
{
:Top
    SET x=$RANDOM(90)
    IF x<65 {GOTO Top}
    ELSE {QUIT $CHAR(x)}
}

SELECT Name FROM Sample.Person
WHERE Name %STARTSWITH RandomLetterSP()

DROP PROCEDURE RandomLetterSP
```

下面的CREATE PROCEDURE示例使用ObjectScript调用Execute()，Fetch()。和Close()方法。此类过程还可以包含FetchRows()和GetInfo()方法调用：

```
CREATE PROCEDURE GetTitle()
FOR Sample.Employee
RESULTS (ID %Integer)
CONTAINID 1
LANGUAGE OBJECTSCRIPT
Execute(INOUT qHandle %Binary)
{ QUIT 1 }
Fetch(INOUT qHandle %Binary, INOUT Row %List, INOUT AtEnd %Integer)
{ QUIT 1 }
Close(INOUT qHandle %Binary)
{ QUIT 1 }
```

下面的CREATE PROCEDURE示例使用ObjectScript调用%SQL.Statement结果集类：

```
CREATE PROCEDURE Sample_Employee.GetTitle(
    INOUT Title VARCHAR(50) )
RETURNS VARCHAR(30)
FOR Sample.Employee
LANGUAGE OBJECTSCRIPT
{
SET myquery="SELECT TOP 10 Name,Title FROM Sample.Employee"
SET tStatement = ##class(%SQL.Statement).%New()
SET qStatus = tStatement.%Prepare(myquery)
```

```
IF qStatus'=1 {WRITE "%Prepare failed:" DO $System.Status.DisplayError(qStatus) QUIT}
SET rset = tStatement.%Execute()
DO rset.%Display()
WRITE !,"End of data"
}
```

如果ObjectScript代码块将数据提取到局部变量(例如, Row)中, 则必须以行set Row=""结束代码块, 以指示数据结束条件。

下面的示例将CREATE PROCEDURE与调用嵌入式SQL的ObjectScript代码一起使用。它在Sample.Employee类中生成方法GetTitle, 并将Title值作为参数传出:

```
CREATE PROCEDURE Sample_Employee.GetTitle(
  IN SSN VARCHAR(11),
  INOUT Title VARCHAR(50) )
  RETURNS VARCHAR(30)
  FOR Sample.Employee
  LANGUAGE OBJECTSCRIPT
  {
    NEW SQLCODE,%ROWCOUNT
    &sql(SELECT Title INTO :Title FROM Sample.Employee
      WHERE SSN = :SSN)
    IF $GET(%sqlcontext)'= "" {
      SET %sqlcontext.%SQLCODE=SQLCODE
      SET %sqlcontext.%ROWCOUNT=%ROWCOUNT }
    QUIT
  }
```

它使用%sqlcontext对象, 并使用相应的SQL变量设置它的%SQLCODE和%ROWCOUNT属性。注意, 在过程的LANGUAGE ObjectScript关键字后面的花括号中包含ObjectScript代码。在ObjectScript代码中有嵌入式SQL代码, 用&sql标记, 用括号括起来。

[#SQL #Caché](#)

---

### 源

URL:

<https://cn.community.intersystems.com/post/%E7%AC%AC%E5%8D%81%E4%B8%80%E7%AB%A0-sql%E5%91%BD%E4%BB%A4-create-procedure%E4%BA%8C%E4%BC%89>