文章
姚 鑫 · 九月 10, 2021 阅读大约需 10 分钟

第十二章 SQL命令 CREATE QUERY

第十二章 SQL命令 CREATE QUERY

创建Query

大纲

```
CREATE QUERY queryname(parameter_list) [characteristics]
   [ LANGUAGE SQL ]
   BEGIN
code_body ;
   END

CREATE QUERY queryname(parameter_list) [characteristics]
    LANGUAGE OBJECTSCRIPT
   { code_body }
```

参数

- queryname - 要在存储过程类中创建的查询的名称。queryname必须是有效的标识符。过程名可以是限定的(schema.procname)，也可以是非限定的(procname)。非限定过程名接受默认模式名。即使没有指定参数，queryname也必须后跟括号。
- parameterlist - 可选-传递给查询的参数列表。参数列表用圆括号括起来，列表中的参数用逗号分隔。即使没有指定参数，括号也是必须的。
- characteristics - 可选-指定查询特征的一个或多个关键字。允许的关键字有结果、容器ID、FOR、FINAL、PROCEDURE、SELECTMODE。多个特征由空白(空格或换行符)分隔。特性可以以任何顺序指定。
- LANGUAGE OBJECTSCRIPT，LANGUAGE SQL - 可选—指定用于codebody的编程语言的关键字子句。指定语言对象脚本或语言SQL。如果省略了LANGUAGE子句，则默认为SQL。
- codebody - 查询的程序代码。SQL程序代码以BEGIN关键字开头，以END关键字结尾。查询的codebody只包含一个完整的SQL语句(一个SELECT语句)。该SELECT语句以分号(;)结束。ObjectScript程序代码用花括号括起来。ObjectScript代码行必须缩进。

描述

CREATE QUERY语句在类中创建一个查询。
默认情况下，名为MySelect的查询将被存储为User.queryMySelect或SQLUser.queryMySelect。

CREATE QUERY创建的查询可能作为存储过程公开，也可能不作为存储过程公开。
要创建公开为存储过程的查询，必须指定procedure关键字作为其特征之一。
还可以使用CREATE PROCEDURE语句创建作为存储过程公开的查询。

为了创建查询，必须拥有%CREATEQUERY管理权限，如GRANT命令所指定的。如果试图为已定义所有者的现有类创建查询，则必须以该类的所有者身份登录。否则，操作将失败，并出现SQLCODE -99错误。

如果类定义是已部署的类，则不能在类中创建查询。此操作失败，出现SQLCODE -400错误，出现%msgUnable to execute DDL that modifies a deployed class: 'classname'。

参数

queryname

要创建为存储过程的查询的名称。此名称可以是非限定名称(StoreName)并采用默认架构名称，也可以通过指定架构名称(Patient.StoreName)进行限定。可以使用$SYSTEM.SQL.Schema.Default()方法来确定当前系统范围内的默认架构名称。系统范围内的初始默认模式名是SQLUser，它对应于类包名User。

注意，FOR特征(将在下面描述)覆盖queryname中指定的类名。
如果已经存在具有此名称的方法，则操作将失败，并出现SQLCODE -361错误。

生成的类的名称是对应于架构名称的包名，后跟一个点，后跟 “ query ”，后跟指定的queryname。例如，如果非限定查询名RandomLetter采用初始默认模式SQLUser，则得到的类名将是:User.queryRandomLetter。

SQL不允许指定只以字母大小写不同的查询名。
指定一个与现有查询名称仅在字母大小写上不同的查询名称将导致SQLCODE -400错误。

如果指定的queryname已经存在于当前命名空间中，系统将生成SQLCODE -361错误。

parameter-list

用于将值传递给查询的参数的参数声明列表。
形参列表用圆括号括起来，列表中的形参声明用逗号分隔。
括号是必须的，即使没有指定参数。

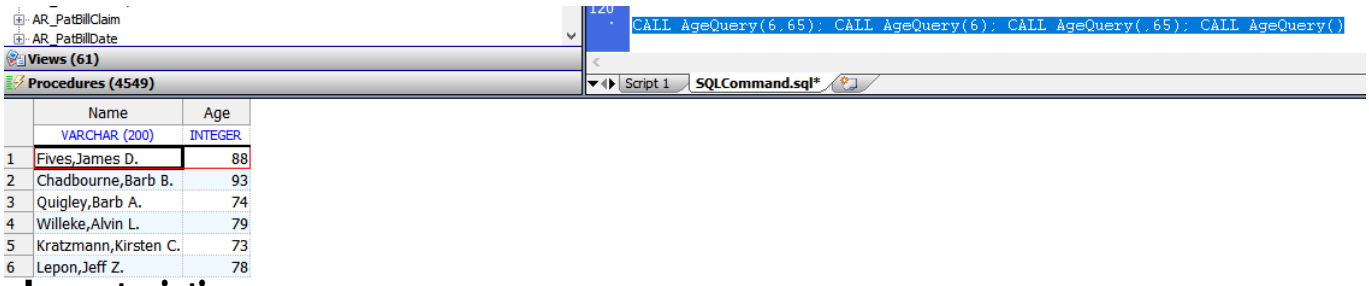
列表中的每个参数声明由(按顺序)组成:
- 一个可选关键字，指定参数模式是IN(输入值)、OUT(输出值)还是INOUT(修改值)。
如果省略，默认参数模式为IN。
- 参数名称。
参数名称区分大小写。
- 参数的数据类型。
- 可选:默认值。可以指定DEFAULT关键字后跟一个默认值;DEFAULT关键字是可选的。如果没有指定默认值，则假定默认值为NULL。

下面的示例创建了一个公开为存储过程的查询，该存储过程具有两个输入参数，这两个参数都具有默认值。
topnum输入参数指定可选的DEFAULT关键字;
minage输入参数忽略了这个关键字:

```
CREATE QUERY AgeQuery(IN topnum INT DEFAULT 10,IN minage INT 20)
   PROCEDURE
   BEGIN
   SELECT TOP :topnum Name,Age FROM Sample.Person
   WHERE Age > :minage ;
   END
```

以下是该查询的所有有效CALL语句：Call AgeQuery(6，65)；Call AgeQuery(6)；Call AgeQuery(，65)；Call AgeQuery()。

```
CALL AgeQuery(6,65); CALL AgeQuery(6); CALL AgeQuery(,65); CALL AgeQuery()
```



	Name	Age
	VARCHAR (200)	INTEGER
1	Fives,James D.	88
2	Chadbourne,Barb B.	93
3	Quigley,Barb A.	74
4	Willeke,Alvin L.	79
5	Kratzmann,Kirsten C.	73
6	Lepon,Jeff Z.	78

characteristics

可用的特征关键字如下：

- CONTAINID integer -
  指定返回ID的字段(如果有)。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确实包含ID，因此此处的用户错误会导致数据不一致。
- FOR className - 指定要在其中创建方法的类的名称。如果该类不存在，则会创建它。还可以通过限定方法名称来指定类名。在FOR子句中指定的类名将覆盖通过限定方法名指定的类名。
- FINAL - 指定子类不能重写该方法。默认情况下，方法不是最终的。Final关键字由子类继承。
- PROCEDURE - 指定查询为SQL存储过程。存储过程由子类继承。(此关键字可以缩写为proc。)
- RESULTS (resultset) - 按查询返回数据字段的顺序指定数据字段。如果指定RESULTS子句，则必须将查询返回的所有字段作为逗号分隔的列表列出，并将其括在圆括号中。指定比查询返回的字段少或多的字段会导致SQLCODE-76基数不匹配错误。为每个字段指定列名(将用作列标题)和数据类型。如果使用SQL语言，则可以省略RESULTS子句。如果省略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。可能的值有Logical、ODBC、Runtime和Display。默认值为运行时。

如果指定的方法关键字(如PRIVATE或RETURNS)对查询无效，系统将生成SQLCODE-47错误。指定重复特征会导致SQLCODE-44错误。

SELECTMODE子句指定返回数据的模式。如果模式值是逻辑值，则返回逻辑值(内部存储)。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。如果模式值为RUNTIME，则可以通过设置%SQL.Statement类%SelectMode属性在执行时设置模式(设置为LOGICAL、ODBC或DISPLAY)，运行时模式的值为Logical。为SELECTMODE指定的值将添加到ObjectScript类方法代码的开头：#SQLCompile select=mode。

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CODEBODY使用的语言的关键字子句。允许的子句是Language OBJECTSCRIPT或Language SQL。如果省略LANGUAGE子句，则默认为SQL。

如果语言是SQL，则会生成%Library.SQLQuery类型的类查询。如果语言是OBJECTSCRIPT，则会生成%Library.Query类型的类查询。

codebody

要创建的查询的程序代码。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，在ObjectScript中指定的代码可以包含嵌入式SQL。

如果指定的代码是SQL，则它必须由单个SELECT语句组成。SQL中查询的程序代码以BEGIN关键字开头，后跟程序代码(SELECT语句)。在程序代码的末尾，指定分号(；)，然后指定END关键字。

如果指定的代码是OBJECTSCRIPT，则它必须包含对 IRIS提供的%Library.Query类的Execute()和Fetch()类方法的

调用，并且可以包含Close()、FetchRows()和GetInfo()方法调用。ObjectScript代码用大括号括起来。如果EXECUTE()或FETCH()丢失，则编译时会生成SQLCODE-46错误。

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。

如果查询公开为存储过程(通过在Characteristic中指定PROCEDURE关键字)，则它使用过程上下文处理程序在过程及其调用方之间来回传递过程上下文。

调用存储过程时，%Library.SQLProcContext类的对象在%sqlcontext变量中实例化。这用于在过程及其调用者(例如，ODBC服务器)之间来回传递过程上下文。

%sqlcontext由几个属性组成，包括错误对象、SQLCODE错误状态、SQL行数和错误消息。下面的示例显示了用于设置其中几个值的值：

```
  SET %sqlcontext.%SQLCODE=SQLCODE
  SET %sqlcontext.%ROWCOUNT=%ROWCOUNT
  SET %sqlcontext.%Message=%msg
```

SQLCODE和%ROWCOUNT的值由SQL语句的执行自动设置。每次执行前都会重置%sqlcontext对象。

或者，可以通过实例化%SYSTEM.Error对象并将其设置为%sqlcontext.Error来建立错误上下文。

IRIS使用提供的代码生成查询的实际代码。

示例

下面的嵌入式SQL示例创建名为DocTestPersonState的查询。它不声明任何参数，设置SELECTMODE特征，并采用默认语言(SQL)：

```
ClassMethod CreateQuery()
{
    &sql(
        CREATE QUERY DocTestPersonState() SELECTMODE RUNTIME
            BEGIN
                SELECT Name,Home_State FROM Sample.Person ;
            END
    )
    if SQLCODE=0 {
        w !,"????"
    } elseif SQLCODE=-361 {
        w !,"????: ",%msg
    } else {
        w !,"?? QUERY ?? ",SQLCODE
    }
}
```

可以转到管理门户，选择Classes选项，然后选择Samples命名空间。将在那里找到由上面的示例创建的查询：User.queryDocTestPersonState.cls。在重新运行上面的程序示例之前，您可以从该显示中删除此查询。当然，可以使用DROP QUERY删除创建的查询。

```
Class User.queryDocTestPersonState Extends %Library.RegisteredObject [ ClassType = ""
```

```
, DdlAllowed, Owner = {yx}, Not ProcedureBlock ]
{

Query DocTestPersonState() As %Library.SQLQuery(SELECTMODE = "RUNTIME")
{
    SELECT Name,Home_State FROM Sample.Person
}

}
```

下面的嵌入式SQL示例创建一个名为DocTestSQLCODEList的基于方法的查询，该查询获取SQLCODE及其说明的列表。它设置结果结果集特征，将语言设置为ObjectScript，并调用Execute()、Fetch()和Close()方法：

```
ClassMethod CreateQuery1()
{
    &sql(
        CREATE QUERY DocTestSQLCODEList()
            RESULTS
            (
                SQLCODE SMALLINT, Description VARCHAR(100)
            )
            PROCEDURE
            LANGUAGE OBJECTSCRIPT
            Execute(INOUT QHandle BINARY(255))
            {
                s QHandle=1, %i(QHandle)=""
                q ##lit($$$OK)
            }
            Fetch(INOUT QHandle BINARY(255), INOUT Row %List, INOUT AtEnd INT)
            {
                s AtEnd = 0, Row = ""
                s %i(QHandle) = $o(^%qCacheSQL("SQLCODE", %i(QHandle)))
                if %i(QHandle) = "" {
                    s AtEnd = 1 q ##lit($$$OK)
                }
                s Row = $lb(%i(QHandle), ^%qCacheSQL("SQLCODE", %i(QHandle), 1, 1))
                q ##lit($$$OK)
            }
            Close(INOUT QHandle BINARY(255))
            {
                k %i(QHandle)
                q ##lit($$$OK)
            }
    )
    if SQLCODE=0 {
        w !,"????"
    } elseif SQLCODE=-361 {
        w !,"????: ",%msg
    } else {
        w !,"?? QUERY ?? ",SQLCODE _ " "_%msg
    }
}
```

可以转到管理门户，选择Classes选项，然后选择Samples命名空间。将在那里找到由上面的示例创建的查询：User.queryDocTestSQLCODEList.cls。在重新运行上面的程序示例之前，可以从该显示中删除此查询。当然，可以使用D

ROP QUERY删除创建的查询。

下面的动态SQL示例创建名为DocTest的查询，然后使用%SQL.Statement类的%PrepareClassQuery()方法执行此查询：

```
ClassMethod CreateQuery2()
{
    s SQLCODE = 0
    /* ?? Query */
    s myquery=4
    s myquery(1) = "CREATE QUERY DocTest() SELECTMODE RUNTIME "
    s myquery(2) = "BEGIN "
    s myquery(3) = "SELECT TOP 5 Name,Home_State FROM Sample.Person ; "
    s myquery(4) = "END"
    s tStatement = ##class(%SQL.Statement).%New()
    s qStatus = tStatement.%Prepare(.myquery)
    if qStatus '= 1 {
        w "%Prepare ???" DO $System.Status.DisplayError(qStatus)
        q
    }
    s rset = tStatement.%Execute()
    if SQLCODE = 0 {
        w !,"????"
    } elseif SQLCODE=-361 {
        w !,"????: ",%msg
    } else {
        w !,"?? QUERY ?? ",SQLCODE _ " "_%msg
    }
    /* ?? Query */
    w !,"?? Query",!
    s cqStatus = tStatement.%PrepareClassQuery("User.queryDocTest","DocTest")
    if cqStatus'=1 {
        w "%PrepareClassQuery ???"
        d $System.Status.DisplayError(cqStatus)
    }
    s rset = tStatement.%Execute()
    w "Query ??",!,!
    while rset.%Next() {
        d rset.%Print()
    }
    w !,"????"
    /* ?? Query */
    &sql(DROP QUERY DocTest)
    if SQLCODE = 0 {
        w !,"?? Query"
    }
}
```

#SQL #Caché

源URL:
https://cn.community.intersystems.com/post/%E7%AC%AC%E5%8D%81%E4%BA%8C%E7%AB%A0-sql%E5%91%BD%E4%BB%A4-create-query