文章
姚 鑫 · 三月 23, 2021 阅读大约需 12 分钟

第十三章 使用动态SQL（一）

第十三章 使用动态SQL（一）

动态SQL简介

动态SQL是指在运行时准备并执行的SQL语句。在动态SQL中，准备和执行SQL命令是单独的操作。通过动态SQL，可以以类似于ODBC或JDBC应用程序的方式在InterSystems IRIS中进行编程（除了要在与数据库引擎相同的进程上下文中执行SQL语句）。动态SQL是从ObjectScript程序调用的。

动态SQL查询是在程序执行时准备的，而不是在编译时准备的。这意味着编译器无法在编译时检查错误，并且不能在Dynamic SQL中使用预处理器宏。这也意味着执行程序可以响应用户或其他输入而创建专门的Dynamic SQL查询。

动态SQL可用于执行SQL查询。它也可以用于发出其他SQL语句。本章中的示例执行SELECT查询。

动态SQL用于执行InterSystems IRIS SQL Shell，InterSystems IRIS管理门户网站“ 执行查询 ”界面，SQL代码导入方法以及“ 数据导入和导出实用程序 ”。

在Dynamic SQL（和使用它的应用程序）中，行的最大大小为3,641,144个字符。

动态SQL与嵌入式SQL

动态SQL与嵌入式SQL在以下方面有所不同：

- **动态SQL查询的初始执行效率比嵌入式SQL稍低，因为它不会生成查询的内联代码。但是，动态SQL和嵌入式SQL的重新执行比第一次执行查询要快得多，因为它们都支持缓存的查询。**
- 动态SQL可以通过两种方式接受输入到查询的文字值：使用“ ？ ”指定的输入参数。字符和输入主机变量（例如:var）。嵌入式SQL使用输入和输出主机变量（例如:var）。
- 使用结果集对象（即Data属性）的API检索动态SQL输出值。嵌入式SQL将主机变量（例如：var）与SELECT语句的INTO子句一起使用以输出值。
- **动态SQL设置%SQLCODE，%Message，%ROWCOUNT和%ROWID对象属性。嵌入式SQL设置相应的SQLCODE，%msg，%ROWCOUNT和%ROWID局部变量。动态SQL不会为SELECT查询设置%ROWID；嵌入式SQL为基于游标的SELECT查询设置%ROWID。**
- 动态SQL提供了一种简单的方法来查找查询元数据（例如列的数量和名称）。
- 动态SQL执行SQL特权检查；必须具有适当的权限才能访问或修改表，字段等。Embedded SQL不执行SQL特权检查。
- 动态SQL无法访问私有类方法。要访问现有的类方法，必须将该方法公开。这是一般的SQL限制。但是，嵌入式SQL克服了此限制，因为嵌入式SQL操作本身是同一类的方法。

动态SQL和嵌入式SQL使用相同的数据表示形式（默认情况下为逻辑模式，但是可以更改）和NULL处理。

%SQL.Statement类

动态SQL的首选接口是%SQL.Statement类。要准备和执行动态SQL语句，请使用%SQL.Statement的实例。执行动态SQL语句的结果是一个SQL语句结果对象，该对象是%SQL.StatementResult类的实例。 SQL语句结果对象可以是单一值，结果集或上下文对象。在所有情况下，结果对象都支持标准接口。每个结果对象都会初始化%SQLCODE，

%Message和其他结果对象属性；这些属性设置的值取决于发出的SQL语句。对于成功执行的SELECT语句，对象是结果集（特别是%SQL.StatementResult的实例），并且支持预期的结果集功能。

以下ObjectScript代码准备并执行动态SQL查询：

```
/// d ##class(PHA.TEST.SQL).DynamicSQL()
ClassMethod DynamicSQL()
{
    /* ????SQL.Statement?? */
    SET myquery = "SELECT TOP 5 Name,DOB FROM Sample.Person"
    SET tStatement = ##class(%SQL.Statement).%New()
    SET qStatus = tStatement.%Prepare(myquery)
    IF qStatus'=1 {
        WRITE "%Prepare ??"
        DO $System.Status.DisplayError(qStatus)
        QUIT
    }
    SET rset = tStatement.%Execute()
    DO rset.%Display()
    WRITE !,"End of data"
}
```

```
DHC-APP>d ##class(PHA.TEST.SQL).DynamicSQL()
Name    DOB
yaoxin  54536
xiaoli
??    63189
??    63189
??    50066

5 Rows(s) Affected
End of data
```

本章中的示例使用与%SQL.Statement和%SQL.StatementResult类关联的方法。

创建一个对象实例

可以使用%New（）类方法创建%SQL.Statement类的实例：

SET tStatement = ##class(%SQL.Statement).%New()

此时，结果集对象已准备好准备SQL语句。创建%SQL.Statement类的实例后，可以使用该实例发出多个动态SQL查询和/或INSERT，UPDATE或DELETE操作。

%New（）按以下顺序接受三个可选的逗号分隔参数：

1. %SelectMode，它指定用于数据输入和数据显示的模式。
2. %SchemaPath，它指定用于为无限定的表名提供架构名称的搜索路径。
3. %Dialect，它指定Transact-SQL（TSQL）Sybase或MSSQL方言。默认值为IRIS（InterSystems SQL）。

还有一个%ObjectSelectMode属性，不能将其设置为%New（）参数。
%ObjectSelectMode指定字段到其相关对象属性的数据类型绑定。

在下面的ObjectScript示例中，%SelectMode为2（显示模式），%SchemaPath将“ Sample ”指定为默认架构：

```
SET tStatement = ##class(%SQL.Statement).%New(2,"Sample")
```

在下面的ObjectScript示例中，未指定%SelectMode（请注意占位符逗号），并且%SchemaPath指定包含三个架构名称的架构搜索路径：

```
SET tStatement = ##class(%SQL.Statement).%New(,"MyTests,Sample,Cinema")
```

%SelectMode属性

%SelectMode属性指定以下模式之一：0 =Logical逻辑（默认），1 = ODBC，2 =Display.显示。这些模式指定如何输入和显示数据值。模式最常用于日期和时间值以及显示%List数据（包含编码列表的字符串）。数据以逻辑模式存储。

SELECT查询使用%SelectMode值确定用于显示数据的格式。

INSERT或UPDATE操作使用%SelectMode值来确定允许的数据输入格式。

%SelectMode用于数据显示。 SQL语句在内部以逻辑模式运行。例如，无论%SelectMode设置如何，ORDER BY子句均根据记录的逻辑值对记录进行排序。 SQL函数使用逻辑值，而不管%SelectMode设置如何。映射为SQLPROC的方法也可以在逻辑模式下运行。在SQL语句中称为函数的SQL例程需要以逻辑格式返回函数值。

- 对于SELECT查询，%SelectMode指定用于显示数据的格式。将%SelectMode设置为ODBC或Display也会影响用于指定比较谓词值的数据格式。某些谓词值必须以%SelectMode格式指定，而其他谓词值必须以逻辑格式指定，而与%SelectMode无关。
  - %SelectMode = 1（ODBC）中的时间数据类型数据可以显示小数秒，这与实际的ODBC时间不同。InterSystems IRIS Time数据类型支持小数秒。相应的ODBC TIME数据类型（TIMESTRUCT标准标头定义）不支持小数秒。 ODBC TIME数据类型将提供的时间值截断为整秒。 ADO DotNet和JDBC没有此限制。
  - %SelectMode = 0（逻辑）中的%List数据类型数据不会显示内部存储值，因为%List数据是使用非打印字符编码的。而是，Dynamic SQL将%List数据值显示为$LISTBUILD语句，例如：$lb("White","Green")。 %SelectMode = 1（ODBC）中的%List数据类型数据显示用逗号分隔的列表元素；此元素分隔符指定为CollectionOdbcDelimiter参数。 %SelectMode = 2中的%List数据类型数据（显示）显示由$ CHAR（10,13）分隔的列表元素（换行，回车）；此元素分隔符指定为CollectionDisplayDelimiter参数。
- 对于INSERT或UPDATE操作，%SelectMode指定将转换为逻辑存储格式的输入数据的格式。为了进行此数据转换，必须使用RUNTIME（默认）的选择模式编译SQL代码，以便在执行INSERT或UPDATE时使用Display或ODBC %SelectMode。有关日期和时间的允许输入值，请参考日期和时间数据类型。

可以将%SelectMode指定为%New（）类方法的第一个参数，或直接对其进行设置，如以下两个示例所示：

```
SET tStatement = ##class(%SQL.Statement).%New(2)
```

```
SET tStatement = ##class(%SQL.Statement).%New()
SET tStatement.%SelectMode=2
```

下面的示例返回%SelectMode的当前值：

```
/// d ##class(PHA.TEST.SQL).DynamicSQL1()
```

```
ClassMethod DynamicSQL1()
{
    SET tStatement = ##class(%SQL.Statement).%New()
    WRITE !,"??????=",tStatement.%SelectMode
    SET tStatement.%SelectMode=2
    WRITE !,"??????=",tStatement.%SelectMode
}


DHC-APP>d ##class(PHA.TEST.SQL).DynamicSQL1()

??????=0
??????=2
```

可以使用$SYSTEM.SQL.Util.GetOption("SelectMode") 方法为当前进程确定SelectMode默认设置。当n可以为0 =逻辑，1 = ODBC或2 = Display时，可以使用$SYSTEM.SQL.Util.SetOption("SelectMode",n) 方法来更改当前进程的SelectMode默认设置。设置%SelectMode会覆盖当前对象实例的默认设置。它不会更改SelectMode进程的默认值。

%SchemaPath属性

%SchemaPath属性指定用于为非限定的表名，视图名或存储过程名提供架构名的搜索路径。模式搜索路径用于数据管理操作，例如SELECT，CALL，INSERT和TRUNCATE TABLE；数据定义操作（例如DROP TABLE）将忽略它。

搜索路径被指定为带引号的字符串，其中包含模式名称或逗号分隔的一系列模式名称。 InterSystems IRIS以从左到右的顺序搜索列出的模式。 InterSystems IRIS会搜索每个指定的架构，直到找到第一个匹配的表，视图或存储过程名称。因为模式是按指定顺序搜索的，所以不会检测到歧义的表名。仅搜索当前名称空间中的架构名称。

模式搜索路径可以包含文字模式名称以及CURRENTPATH，CURRENTSCHEMA和DEFAULTSCHEMA关键字。
- CURRENTPATH指定当前模式搜索路径，如先前的%SchemaPath属性中所定义。这通常用于将架构添加到现有架构搜索路径的开头或结尾。
- 如果%SQL.Statement调用是从类方法中进行的，则CURRENTSCHEMA指定当前模式容器的类名称。如果在类方法中定义了#SQLCompile Path宏指令，则CURRENTSCHEMA是映射到当前类包的架构。否则，CURRENTSCHEMA与DEFAULTSCHEMA相同。
- DEFAULTSCHEMA指定系统范围的默认架构。使用此关键字，可以在搜索其他列出的架构之前，在架构搜索路径中将系统范围的默认架构作为一个项目进行搜索。如果已经搜索了路径中指定的所有模式而没有匹配项，则在搜索模式搜索路径后始终会搜索系统范围内的默认模式。

%SchemaPath是InterSystems IRIS在架构中搜索匹配表名的第一位。如果未指定%SchemaPath，或者未列出包含匹配表名的架构，则InterSystems IRIS将使用系统范围的默认架构。

可以通过指定%SchemaPath属性或指定%New（）类方法的第二个参数来指定模式搜索路径，如以下两个示例所示：

```
  SET path="MyTests,Sample,Cinema"
  SET tStatement = ##class(%SQL.Statement).%New(,path)


  SET tStatement = ##class(%SQL.Statement).%New()
  SET tStatement.%SchemaPath="MyTests,Sample,Cinema"
```

可以在使用它的%Prepare（）方法之前的任何位置设置%SchemaPath。

下面的示例返回%SchemaPath的当前值：

```
/// d ##class(PHA.TEST.SQL).DynamicSQL2()
ClassMethod DynamicSQL2()
{
    SET tStatement = ##class(%SQL.Statement).%New()
    WRITE !,"?? path=",tStatement.%SchemaPath
    SET tStatement.%SchemaPath="MyTests,Sample,Cinema"
    WRITE !,"?? path=",tStatement.%SchemaPath
}
```

```
DHC-APP>d ##class(PHA.TEST.SQL).DynamicSQL2()

?? path=
?? path=MyTests,Sample,Cinema
```

可以使用%ClassPath（）方法将%SchemaPath设置为为指定的类名定义的搜索路径：

```
/// d ##class(PHA.TEST.SQL).DynamicSQL3()
ClassMethod DynamicSQL3()
{
    SET tStatement = ##class(%SQL.Statement).%New()
    SET tStatement.%SchemaPath=tStatement.%ClassPath("Sample.Person")
    WRITE tStatement.%SchemaPath
}
```

```
DHC-APP>d ##class(PHA.TEST.SQL).DynamicSQL3()
Sample
```

%Dialect属性

%Dialect属性指定SQL语句方言。可以指定Sybase，MSSQL或IRIS（InterSystems SQL）。Sybase或MSSQL设置导致使用指定的Transact-SQL方言处理SQL语句。

Sybase和MSSQL方言在这些方言中支持SQL语句的有限子集。它们支持SELECT，INSERT，UPDATE，DELETE和EXECUTE语句。他们支持CREATE TABLE语句用于永久表，但不支持临时表。支持创建视图。支持CREATE TRIGGER和DROP TRIGGER。但是，如果CREATE TRIGGER语句部分成功，但是在类编译时失败，则此实现不支持事务回滚。支持CREATE PROCEDURE和CREATE FUNCTION。

Sybase和MSSQL方言支持IF控制流语句。 IRIS（InterSystems SQL）方言不支持此命令。

默认值为InterSystems SQL，由空字符串（“ ”）表示，或指定为“ IRIS ”

可以将%Dialect指定为%New（）类方法的第三个参数，或者将其直接设置为属性，或者使用方法进行设置，如以下三个示例所示：

在%New（）类方法中设置%Dialect：

```
/// d ##class(PHA.TEST.SQL).DynamicSQL4()
```

```
ClassMethod DynamicSQL4()
{
    SET tStatement = ##class(%SQL.Statement).%New(,,"Sybase")
    WRITE "???????=",tStatement.%Dialect
}


DHC-APP>d ##class(PHA.TEST.SQL).DynamicSQL4()
???????=Sybase
```

直接设置%Dialect属性：

```
/// d ##class(PHA.TEST.SQL).DynamicSQL5()
ClassMethod DynamicSQL5()
{
    SET tStatement = ##class(%SQL.Statement).%New()
    SET defaultdialect=tStatement.%Dialect
    WRITE "??????=",defaultdialect,!
    SET tStatement.%Dialect="Sybase"
    WRITE "???????=",tStatement.%Dialect,!
    SET tStatement.%Dialect="IRIS"
    WRITE "??????????=",tStatement.%Dialect,!
}


DHC-APP>d ##class(PHA.TEST.SQL).DynamicSQL5()
??????=
???????=Sybase
??????????=iris
```

使用%DialectSet（）实例方法设置%Dialect属性，该方法将返回错误状态：

```
/// d ##class(PHA.TEST.SQL).DynamicSQL6()
ClassMethod DynamicSQL6()
{
    SET tStatement = ##class(%SQL.Statement).%New()
    SET tStatus = tStatement.%DialectSet("Sybase")
    IF tStatus'=1 {
        WRITE "%DialectSet ??:"
        DO $System.Status.DisplayError(tStatus) QUIT
    }
    WRITE "???????=",tStatement.%Dialect
}


DHC-APP>d ##class(PHA.TEST.SQL).DynamicSQL6()
???????=Sybase
```

%DialectSet（）方法返回%Status值：成功返回状态1。失败返回以0开头的对象表达式，后跟编码错误信息。因此，无法执行tStatus = 0测试是否失败；您可以执行$$ISOK（tStatus）= 0宏测试以检查失败

%ObjectSelectMode属性

%ObjectSelectMode属性是一个布尔值。如果%ObjectSelectMode = 0（默认），则SELECT列表中的所有列都将绑定到结果集中具有文字类型的属性。如果%ObjectSelectMode = 1，则SELECT列表中的列将绑定到具有关联属性定义中定义的类型的属性。

%ObjectSelectMode允许指定如何在从SELECT语句生成的结果集类中定义类型类为swizzleable类的列。如果%ObjectSelectMode = 0，则将在结果集中将与swizzleable列相对应的属性定义为与SQL表的RowID类型相对应的简单文字类型。如果%ObjectSelectMode = 1，则将使用列的声明类型定义属性。这意味着访问结果集属性将触发swizzling。

无法将%ObjectSelectMode设置为%New（）的参数。

下面的示例返回%ObjectSelectMode默认值，设置%ObjectSelectMode，然后返回新的%ObjectSelectMode值：

```
/// d ##class(PHA.TEST.SQL).DynamicSQL7()
ClassMethod DynamicSQL7()
{
    SET myquery = "SELECT TOP 5 %ID AS MyID,Name,Age FROM Sample.Person"
    SET tStatement = ##class(%SQL.Statement).%New()
    WRITE !,"?? ObjectSelectMode=",tStatement.%ObjectSelectMode
    SET tStatement.%ObjectSelectMode=1
    WRITE !,"?? ObjectSelectMode=",tStatement.%ObjectSelectMode
}
```

```
DHC-APP>d ##class(PHA.TEST.SQL).DynamicSQL7()

?? ObjectSelectMode=0
?? ObjectSelectMode=1
```

当使用字段名称属性从结果集中返回值时，主要使用%ObjectSelectMode = 1。本章“ 从结果集中返回特定值 ”部分的字段名属性中的示例对此进行了进一步说明。

当SELECT列表中的字段链接到集合属性时，可以使用%ObjectSelectMode = 1。%ObjectSelectMode将使集合swizzle。如果%SelectMode = 1或2，则系统在转换前将收集序列值转换为逻辑模式形式。生成的oref支持完整的收集接口。

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

源URL:
https://cn.community.intersystems.com/post/%E7%AC%AC%E5%8D%81%E4%B8%89%E7%AB%A0-%E4%BD%BF%E7%94%A8%E5%8A%A8%E6%80%81sql%EF%BC%88%E4%B8%80%EF%BC%89