文章
姚 鑫 · 三月 17, 2021 阅读大约需 分钟

第十二章 使用嵌入式SQL(一)

第十二章 使用嵌入式SQL(一)

可以将SQL语句嵌入InterSystemsIRIS®数据平台使用的ObjectScript代码中。这些嵌入式SQL语句在运行时转换为优化的可执行代码。

嵌入式SQL有两种：

- 一个简单的嵌入式SQL查询只能返回单行中的值。简单嵌入式SQL还可以用于单行插入，更新和删除以及其他SQL操作。
- 基于游标的嵌入式SQL查询可以遍历查询结果集，并从多行中返回值。基于游标的嵌入式SQL也可以用于多行更新和删除SQL操作。

注意：嵌入式SQL不能输入到Terminal命令行，也不能在XECUTE语句中指定。要从命令行执行SQL，请使用$SYSTEM.SQL.Execute()方法或SQL Shell接口。

编译嵌入式SQL

当包含嵌入式SQL的例程被编译时，嵌入式SQL不会被编译。
相反，嵌入式SQL的编译发生在SQL代码的第一次执行(运行时)。
第一次执行定义了一个可执行的缓存查询。
这与动态SQL的编译类似，在动态SQL中，直到执行SQL Prepare操作才编译SQL代码。

直到第一次执行例程，嵌入式SQL代码才会根据SQL表和其他实体进行验证。
因此，可以编译包含嵌入式SQL的持久化类的例程或方法，这些SQL引用在例程编译时不存在的表或其他SQL实体。
由于这个原因，大多数SQL错误是在运行时执行时返回的，而不是编译时返回的。

**在例程编译时，对嵌入式SQL执行SQL语法检查。
ObjectScript编译器失败，并为嵌入式SQL中的无效SQL语法生成编译错误。**

可以使用Management Portal SQL接口测试嵌入式SQL中指定的SQL实体是否存在，而不需执行SQL代码。
这在验证嵌入式SQL代码中进行了描述，该代码既验证SQL语法，又检查是否存在SQL实体。
可以选择在运行时执行之前验证嵌入式SQL代码，方法是使用/compileembedded=1限定符编译包含嵌入式SQL代码的例程，如验证嵌入式SQL代码中所述。

对执行的嵌入式SQL语句将生成一个缓存的查询。该嵌入式SQL的后续执行将使用缓存的查询，而不是重新编译嵌入式SQL源。这提供了对嵌入式SQL的缓存查询的性能优势。

当首次使用OPEN命令打开游标时，会执行基于游标的Embedded SQL语句的运行时执行。在执行的这一点上，将生成它的缓存查询计划，如管理门户中的“SQL语句”列表中所示。列出的“SQL语句”位置是包含嵌入式SQL代码的例程的名称。请注意，执行嵌入式SQL不会在“缓存的查询”列表中生成一个条目。这些清单(带有类名称，例如%sqlcq.USER.cls1)是由Dynamic SQL查询创建的。

注意：较早版本的IRIS中使用的#SQLCompile Mode预处理程序语句已被弃用。它已被解析，但不再对大多数嵌入式SQL命令执行任何操作。无论#SQLCompile Mode设置如何，大多数嵌入式SQL命令都会在运行时进行编译。但是，设置#SQLCompile Mode = deferred对于少量的嵌入式SQL命令仍然有意义，因为它会强制在运行时编译所有类型的嵌入式SQL命令。

嵌入式SQL和宏预处理器

可以在方法内和触发器内（前提是它们已定义为使用ObjectScript）或在ObjectScript MAC例程内使用嵌入式SQL。MAC例程由InterSystems IRIS宏预处理器处理，并转换为INT（中间）代码，随后将其编译为可执行的OBJ代码。这些操作是在包含嵌入式SQL的例程的编译时执行的，而不是在嵌入式SQL代码本身上执行的，嵌入式SQL代码本身直到运行时才进行编译。

如果嵌入式SQL语句本身包含InterSystems IRIS宏预处理器语句（# 命令，## 函数或$$macro引用），则在编译例程时将编译这些语句，并在运行时将其提供给SQL代码。 这可能会影响包含ObjectScript代码主体CREATE PROCEDURE，CREATE FUNCTION，CREATE METHOD，CREATE QUERY或CREATE TRIGGER语句。

在嵌入式SQL中包含文件

嵌入式SQL语句要求它们引用的任何宏包含文件都必须在运行时加载到系统上。

因为嵌入式SQL的编译将推迟到首次引用之前进行，所以嵌入式SQL类的编译上下文将是运行时环境，而不是包含类或例程的编译时环境。如果运行时当前名称空间与包含例程的编译时名称空间不同，则编译时名称空间中的包含文件可能在运行时名称空间中不可见。在这种情况下，将发生以下情况：

1. 如果在运行时名称空间中找不到包含文件，则嵌入式SQL编译将删除所有包含文件。由于SQL编译很少需要包含文件，因此如果没有这些文件，运行时嵌入式SQL编译通常会成功。
2. 如果删除包含文件后编译失败，则InterSystems IRIS错误将报告例程编译时名称空间，嵌入式SQL运行时名称空间以及从运行时名称空间找不到的包含文件列表。

#SQLCompile宏指令

宏预处理器提供了三个与嵌入式SQL一起使用的预处理器指令：

- #SQLCompile Select指定从Select语句返回时数据显示的格式，或者指定插入或更新语句时数据输入所需格式，或者指定Select输入主机变量。
  它支持以下6个选项:Logical(默认值)、Display、ODBC、Runtime、Text(与Display相同)和FDBMS(见下文)。
  如果#SQLCompile Select=Runtime，可以使用$SYSTEM.SQL.Util.SetOption("SelectMode", n)方法来更改数据的显示方式。n取值为0=Logical、1=ODBC、2=Display。

  无论指定了#SQLCompile Select选项，INSERT或UPDATE都会自动将指定的数据值转换为相应的逻辑格式进行存储。

  不管指定了#SQLCompile Select选项，Select都会自动将输入的主机变量值转换为谓词匹配的相应逻辑格式。

  使用#SQLCompile Select进行查询显示如下示例所示。
  这些示例显示DOB(出生日期)值，然后将SelectMode更改为ODBC格式，然后再次显示DOB。
  在第一个例子中，改变SelectMode对显示没有影响;
  在第二个示例中，因为#SQLCompile Select=Runtime，更改SelectMode将更改显示:

```
/// d ##class(PHA.TEST.SQL).EmbedSQL()
ClassMethod EmbedSQL()
```

```
{
    #SQLCompile Select=Display
    &sql(SELECT DOB INTO :a FROM Sample.Person)
    IF SQLCODE<0 {
        WRITE "SQLCODE error ",SQLCODE," ",%msg
        QUIT
    } ELSEIF SQLCODE=100 {
        WRITE "Query returns no results"
        QUIT
    }
    WRITE "1st date of birth is ",a,!
    DO $SYSTEM.SQL.Util.SetOption("SelectMode",1)
    WRITE "changed select mode to: ",$SYSTEM.SQL.Util.GetOption("SelectMode"),!
    &sql(SELECT DOB INTO :b FROM Sample.Person)
    WRITE "2nd date of birth is ",b
}


DHC-APP>d ##class(PHA.TEST.SQL).EmbedSQL()
1st date of birth is 04/25/1990
2nd date of birth is 04/25/1990


/// d ##class(PHA.TEST.SQL).EmbedSQL1()
ClassMethod EmbedSQL1()
{
   #SQLCompile Select=Runtime
   &sql(SELECT DOB INTO :a FROM Sample.Person)
      IF SQLCODE<0 {WRITE "SQLCODE error ",SQLCODE," ",%msg  QUIT}
      ELSEIF SQLCODE=100 {WRITE "Query returns no results"  QUIT}
   WRITE "1st date of birth is ",a,!
   //DO $SYSTEM.SQL.Util.SetOption("SelectMode",1)
   //WRITE "changed select mode to: ",$SYSTEM.SQL.Util.GetOption("SelectMode"),!
   &sql(SELECT DOB INTO :b FROM Sample.Person)
   WRITE "2nd date of birth is ",b
}


DHC-APP>d ##class(PHA.TEST.SQL).EmbedSQL1()
1st date of birth is 1990-04-25
2nd date of birth is 1990-04-25
```

- 提供#SQLCompile Select=FDBMS是为了使嵌入式SQL能够以与FDBMS相同的方式格式化数据。
  如果一个查询在WHERE子句中有一个常量值，FDBMS模式假定它是一个显示值，并使用DisplayToLogical转换对它进行转换。
  如果一个查询在WHERE子句中有一个变量，FDBMS模式使用FDBMSToLogical conversion对它进行转换。
  应该设计FDBMS转换方法来处理三种FDBMS变量格式:Internal、Internal_$c(1)_External和$c(1)_External。
  如果查询选择一个变量，它将调用LogicalToFDBMS转换方法。
  这个方法返回Internal_$c(1)_External。

  - #SQLCompile Path(或#Import)指定模式搜索路径，用于解析SELECT、CALL、INSERT、UPDATE、DELETE和TRUNCATE表等数据管理命令中未限定的表、视图和存储过程名称。
    如果没有指定模式搜索路径，或者在指定的模式中找不到表，InterSystems IRIS将使用默认模式。
    数据定义语句如ALTER TABLE、DROP VIEW、CREATE INDEX或CREATE TRIGGER会忽略#SQLCompile Path和#Import。
    数据定义语句使用默认模式来解析非限定名称。

- #SQLCompile
  Audit计是一个布尔开关，指定嵌入式SQL语句的执行是否应该记录在系统事件审计日志中。

嵌入式SQL语法

&sql指令

嵌入式SQL语句由&sql() 指令与其余代码分开，如以示例所示：

```
/// d ##class(PHA.TEST.SQL).EmbedSQL2()
ClassMethod EmbedSQL2()
{
    NEW SQLCODE,a
    WRITE "?????SQL",!
    &sql(SELECT Name INTO :a FROM Sample.Person)
    IF SQLCODE<0 {
        WRITE "SQLCODE?? ",SQLCODE," ",%msg  QUIT
    } ELSEIF SQLCODE=100 {
        WRITE "??????"  QUIT
    }
    WRITE "??? ",a
}
```

```
DHC-APP>d ##class(PHA.TEST.SQL).EmbedSQL2()
?????SQL
??? Adams,Diane F.
```

使用指定一个或个主机变量的INTO子句返回结果。在这种情况，主机变量名为：a。

&sql指令不区分大小写，可以使用&sql，&SQL，&Sql等。
&sql指令必须后跟一个开放的括号，并且中间没有空格，换行符或 注 释。
&sql指令可以与标签在同一行上使用，如以示例所示：

```
/// d ##class(PHA.TEST.SQL).EmbedSQL3()
ClassMethod EmbedSQL3()
{
Mylabel  &sql(
            SELECT Name INTO :a
            FROM Sample.Person
            )
}
```

&sql指令的主体包含一个有效的Embedded SQL语句，并用括号括起来。可以按照自己喜欢的任何方式设置SQL语句的格式：SQL会忽略空格和换行符。Studio可以识别&sql指令，并使用可识别SQL的着色器对SQL代码语句进行语法着色。

当宏预处理器遇到&sql指令时，它将随附的SQL语句交给SQL查询处理器。查询处理器返回执行查询所需代码（ObjectScript INT格式）。然后，宏预处理器用此代码（或对包含该代码的标签的调用）替换&sql指令。在Studio中，可以根据需要查看生成代码，方法是查看为类或例程生成的INT代码（使用“查看”菜单中的“查看其他代码”选项）
。

如果&sql指令包含无效的Embedded SQL语句，则宏预处理器会生成编译错误。无效的SQL语句可能具有语法错误，或者引用了在编译时不存在的表或列。

&sql指令可以在括号内的任何位置包含SQL样式的 注 释，可以不包含SQL代码，或仅包含 注 释文本。如果&sql指令不包含SQL代码或仅包含 注 释文本，则将该指令解析为无操作，并且未定义SQLCODE变量。

```
NEW SQLCODE
WRITE !,"Entering Embedded SQL"
&sql()
WRITE !,"Leaving Embedded SQL"
```

```
NEW SQLCODE
WRITE !,"Entering Embedded SQL"
&sql(/* SELECT Name INTO :a FROM Sample.Person */)
WRITE !,"Leaving Embedded SQL"
```

&sql嵌入代语法

由于复杂的嵌入式SQL程序可能包含多个&sql指令（包括嵌套的&sql指令），因此提供了以下代语法格式：
- ## sql(...)：此指令在功能上等同于&sql。它提供了另一种语法来使代码清晰。但是，它不能包含标记语法。
- &sql <marker>(...)<reversemarker>：此伪指令允许指定多个&sql伪指令，并使用用户选择的标记字符或字符串标识每个伪伪指令。下一节将介绍此标记语法。

&sql标记语法

可以使用用户定义的标记语法来标识特定的&sql指令。该语法由在“&sql”和右括号之间指定的字符或字符串组成在嵌入式SQL的结尾处，在右括号后必须立即显示此标记的相反内容。语法如下：

```
&sql<marker>( SQL statement )<reverse-marker>
```

请 注 意，在&sql，标记和右括号之间不允许有空格（空格，制表符或行返回），并且在右括号和反向标记之间不允许有空格。

标记可以是单个字符或一系列字符。标记不能包含以下标点符号：

```
( + - / \ | * )
```

标记不能包含空格字符（空格，制表符或换行符）。它可能包含所有其他可打印字符和字符组合，包括Unicode字符。标记和反向标记区分大小写。

相应的反向标记必须包含与反向标记相同的字符。例如：&sqlABC(...)CBA。
如果标记包含[或{字符，则反向标记必须包含相应的]或}字符。以下是有效的&sql标记和反向标记对的示例：

```
&sql@@( ... )@@
&sql[( ... )]
&sqltest( ... )tset
&sql[Aa{( ... )}aA]
```

选择标记字符或字符串时，请 注 意以下重要的SQL限制：SQL代码不能在代码中的任何位置（包括文字字符串和 注 释）包含字符序列“）<reversemarker>”。例如，如果标记“ABC，则字符串“）CBA”不能出现在嵌入式SQL代码中的任何位置。如果发生这种情况，有效标记和有效SQL代码的组合将使编译失败。因此，在选择标记字符或字符串时要格外小心，以防止发生这种冲突，这一点很重要。

嵌入式SQL和行偏移量

嵌入式SQL的存在会影响ObjectScript行偏移量，如下所示：

- 嵌入式SQL在例程中的该点处将INT代码行的总数加（至少）2。因此，嵌入式SQL的单行计为3行，嵌入式SQL的两行计为4行，依此类推。调用其他代码的嵌入式SQL可以向INT代码添加更多行。

一个虚拟的嵌入式SQL语句，仅包含一个 注 释，算作2条INT代码行，如以下示例所示：＆sql（/ *供将来使用* /）。

- 嵌入式SQL中的所有行都计为行偏移，包括 注 释和空白行。

可以使用^ROUTINE全局显示INT代码行。

#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%80%EF%BC%89