文章
姚 鑫 · 三月 29, 2021 阅读大约需 16 分钟

第十三章 使用动态SQL（七）

第十三章 使用动态SQL（七）

SQL元数据

动态SQL提供以下类型的元数据：
- 在“ 准备 ”之后，描述查询类型的元数据。
- 在“ 准备 ”之后，描述查询中选择项的元数据（“ 列 ”和“ 扩展列信息 ”）。
- 在准备之后，描述查询参数的元数据：参数，：var参数和常量。 （语句参数，形式参数和对象）
- 执行之后，描述查询结果集的元数据。在执行Prepare操作（%Prepare（），%PrepareClassQuery（）或%ExecDirect（））之后，可以使用%SQL.StatementMetadata属性值。
- 可以直接为最新的%Prepare（）返回%SQL.Statement元数据属性。
- 可以返回包含%SQL.StatementMetadata属性的oref的%SQL.Statement%Metadata属性。这使可以返回多个准备操作的元数据。

SELECT或CALL语句返回所有这些元数据。 INSERT，UPDATE或DELETE返回语句类型元数据和形式参数。

语句类型元数据

使用%SQL.Statement类进行Prepare之后，可以使用%SQL.StatementMetadata statementType属性来确定准备哪种类型的SQL语句，如以下示例所示。本示例使用%SQL.Statement%Metadata属性来保存和比较两个Prepare操作的元数据：

```
/// d ##class(PHA.TEST.SQL).MetaData()
ClassMethod MetaData()
{
    SET tStatement = ##class(%SQL.Statement).%New()
    SET myquery1 = "SELECT TOP ? Name,Age,AVG(Age),CURRENT_DATE FROM Sample.Person"
    SET myquery2 = "CALL Sample.SP_Sample_By_Name(?)"
    SET qStatus = tStatement.%Prepare(myquery1)
    IF qStatus'=1 {
        WRITE "%Prepare failed:" DO $System.Status.DisplayError(qStatus) QUIT
    }
    SET meta1 = tStatement.%Metadata
    SET qStatus = tStatement.%Prepare(myquery2)
    IF qStatus'=1 {
        WRITE "%Prepare failed:" DO $System.Status.DisplayError(qStatus) QUIT
    }
    SET meta2 = tStatement.%Metadata
    WRITE "????query 1: ",meta1.statementType,!
    WRITE "????query 2: ",meta2.statementType,!
    WRITE "End of metadata"
}
```

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

```
????query 1: 1
????query 2: 45
End of metadata
```

statementType属性的“ 类引用 ”条目列出了语句类型整数代码。最常见的代码是1（SELECT查询）和45（CALL到存储的查询）。

可以使用%GetImplementationDetails（）实例方法返回相同的信息，如成功准备的结果中所述。

执行查询后，可以从结果集中返回语句类型名称（例如SELECT）。

选择项目Select-item元数据

使用%SQL.Statement类准备SELECT或CALL语句之后，可以通过显示所有元数据或指定各个元数据项来返回有关查询中指定的每个选择项列的元数据。此列元数据包括ODBC数据类型信息，以及客户端类型和InterSystems Objects属性的起源以及类类型信息。

以下示例返回最近准备的查询中指定的列数：

```
/// d ##class(PHA.TEST.SQL).MetaData1()
ClassMethod MetaData1()
{
    SET myquery = "SELECT %ID AS id,Name,DOB,Age,AVG(Age),CURRENT_DATE,Home_State FRO
M Sample.Person"
    SET tStatement = ##class(%SQL.Statement).%New()
    SET qStatus = tStatement.%Prepare(myquery)
    IF qStatus'=1 {WRITE "%Prepare failed:" DO $System.Status.DisplayError(qStatus) Q
UIT}
    WRITE "Number of columns=",tStatement.%Metadata.columnCount,!
    WRITE "End of metadata"
}
```

```
DHC-APP>d ##class(PHA.TEST.SQL).MetaData1()
Number of columns=7
End of metadata
```

以下示例返回列名称（或列别名），ODBC数据类型，最大数据长度（精度），以及每个SELECT项目字段的比例：

```
/// d ##class(PHA.TEST.SQL).MetaData2()
ClassMethod MetaData2()
{
    SET $NAMESPACE="SAMPLES"
    SET myquery=2
    SET myquery(1)="SELECT Name AS VendorName,LastPayDate,MinPayment,NetDays,"
    SET myquery(2)="AVG(MinPayment),$HOROLOG,%TABLENAME FROM Sample.Vendor"
    SET rset = ##class(%SQL.Statement).%New()
    SET qStatus = rset.%Prepare(.myquery)
    IF qStatus'=1 {
        WRITE "%Prepare failed:" DO $System.Status.DisplayError(qStatus) QUIT
    }
    SET x=rset.%Metadata.columns.Count()
    SET x=1
    WHILE rset.%Metadata.columns.GetAt(x) {
```

```
        SET column=rset.%Metadata.columns.GetAt(x)
        WRITE !,x," ",column.colName," ????? ",column.ODBCType
        WRITE " ??? ",column.precision," ?? = ",column.scale
        SET x=x+1
    }
    WRITE !,"End of metadata"
}
```

```
DHC-APP>d ##class(PHA.TEST.SQL).MetaData2()

1 VendorName ????? 12 ??? 50 ?? = 0
2 LastPayDate ????? 9 ??? 10 ?? = 0
3 MinPayment ????? 8 ??? 6 ?? = 0
4 NetDays ????? 4 ??? 3 ?? = 0
5 Aggregate_5 ????? 8 ??? 20 ?? = 0
6 Expression_6 ????? 12 ??? 255 ?? = 0
7 Literal_7 ????? 12 ??? 13 ?? = 0
End of metadata
```

下面的示例使用%SQL.StatementMetadata%Display（）实例方法显示所有列元数据：

```
/// d ##class(PHA.TEST.SQL).MetaData3()
ClassMethod MetaData3()
{
    SET tStatement = ##class(%SQL.Statement).%New()
    SET qStatus = tStatement.%Prepare("SELECT %ID AS id,Name,DOB,Age,AVG(Age),CURRENT
_DATE,Home_State FROM Sample.Person")
    IF qStatus'=1 {WRITE "%Prepare failed:" DO $System.Status.DisplayError(qStatus) Q
UIT}
    DO tStatement.%Metadata.%Display()
    WRITE !,"End of metadata"
}
```

```
DHC-APP>d ##class(PHA.TEST.SQL).MetaData3()


Columns (SQLRESULTCOL, property 'columns'):

Column Name  Type Prec Scale Null Label        Table        Schema       CType
-----------  ---- ---- ----- ---- ------------ ------------ ------------ -----
id              4   10     0    0 id           Person       Sample           5
Name           12   50     0    0 Name         Person       Sample          10
DOB             9   10     0    1 DOB          Person       Sample           2
Age             4   10     0    1 Age          Person       Sample           5
Aggregate_5     2   20     8    1 Aggregate_5                               14
Expression_6    9   11     0    2 Expression_6                               2
Home_State     12    2     0    1 Home_State   Person       Sample          10


Extended Column Info (SQLRESULTCOL)
     Flags: 1:AutoIncrement,2:CaseSensitive,3:Currency,4:ReadOnly,5:RowVersion,
            6:Unique,7:Aliased,8:Expression,9:Hidden,10:Identity,11:KeyColumn,
            12:RowId
```

```
Column Name  Linked Prop           Type Class            Flags
------------ --------------------- --------------------- ------------------------
id                                 Sample.Person         Y,N,N,Y,N,Y,Y,N,N,Y,Y,Y
Name         Sample.Person.Name    %Library.String       N,N,N,N,N,N,N,N,N,N,N,N
DOB          Sample.Person.DOB     %Library.Date         N,N,N,N,N,N,N,N,N,N,N,N
Age          Sample.Person.Age     %Library.Integer      N,N,N,N,N,N,N,N,N,N,N,N
Aggregate_5                        %Library.Numeric      N,N,N,Y,N,N,Y,N,N,N,N,N
Expression_6                       %Library.Date         N,N,N,Y,N,N,Y,Y,N,N,N,N
Home_State   Sample.Address.State
                                   %Library.String       N,N,N,N,N,N,N,N,N,N,N,N


Statement Parameters (property 'parameters'):

Nbr. Type precision scale nullable colName      columntype
---- ---- --------- ----- -------- ------------ ----------


Formal Parameters (property 'formalParameters'):

Nbr. Type precision scale nullable colName      columntype
---- ---- --------- ----- -------- ------------ ----------


Objects:

Col Column Name Extent            ExportCall
--- ----------- ----------------- ------------------------------
  1 id          Sample.Person     ##class(Sample.Person).%SQLQuickLoad
```

这将返回所选字段的两个表列表。第一列元数据表列出了列定义信息：

显示标题	%SQL.StatementColumn属性	描述
Column Name	colName	列的SQL名称。如果为该列提供了别名，则会在此处列出该列的别名，而不是字段名称。名称和别名将被截断为12个字符。对于表达式，聚合，文字，主机变量或子查询，列出了分配的“ Expression<u>n</u> ”，“ Aggregate<u>n</u> ”，“ Literal<u>n</u> ”，“ HostVar<u>n</u> ”或“ Subquery<u>n</u> ”标签（n为SELECT项序列号）。如果为表达式，聚合，文字，主机变量或子查询分配了别名，则在此处列出该别名。
Type	ODBCType	ODBC数据类型的整数代码。请注意，这些ODBC数据类型代码与CType数据类型代码不同。
Prec	precision	精度或最大长度（以字符为单位）。日期，时间，PosixTime和TimeStamp数据类型中描述了TIME数据类型的精度和小数位元数据。
Scale	scale	小数位数的最大数目。对于整数或非数值返回0。日期，时间，PosixTime和TimeStamp数据类型中描述了TIME数据类型的精度和小数位元数据。
Null	isNullable	一个整数值，指示是否将列定义为Non-NULL（0），或者是否允许NULL（1）。 RowID返回0。如果SELECT项是可能导致NULL的聚合或子查询，或者

显示标题	%SQL.StatementColumn属性	描述
		如果它指定NULL文字，则该项设置为1。如果SELECT项是表达式或主机变量，则设置此项到2（无法确定）。
Label	label	列名或列别名（与列名相同）。
Table	tableName	SQL表名称。即使为表指定了别名，也始终在此处列出实际的表名。如果SELECT项是表达式或聚合，则不会列出任何表名。如果SELECT项是子查询，则列出子查询表名称。
Schema	schemaName	表的架构名称。如果未指定架构名称，则返回系统范围的默认架构。如果SELECT项是表达式或聚合，则不会列出任何模式名称。如果SELECT项是子查询，则不会列出任何架构名称。
CType	clientType	客户端数据类型的整数代码。

第二列元数据表列出了扩展列信息。扩展列信息表列出了具有十二个布尔标志（SQLRESULTCOL）的每一列，这些标志被指定为Y（是）或N（否）：

显示标题	%SQL.StatementColumn属性	描述
1: AutoIncrement	isAutoIncrement	TRowID和IDENTITY字段返回Y。
2: CaseSensitive	isCaseSensitive	具有%EXACT归类的字符串数据类型字段返回Y。引用%SerialObject嵌入式对象的属性返回Y。
3: Currency	isCurrency	使用%Library.Currency数据类型定义的字段，例如MONEY数据类型。
4: ReadOnly	isReadOnly	表达式，聚合，文字，HostVar或子查询返回Y。RowID，IDENTITY和RowVersion字段返回Y。
5: RowVersion	isRowVersion	RowVersion字段返回Y。
6: Unique	isUnique	定义为具有唯一值约束的字段。RowID和IDENTITY字段返回Y。
7: Aliased	isAliased	系统为非字段选择项提供别名。因此，无论用户是否通过指定列别名替换了系统别名，表达式，聚合，文字，HostVar或子查询都将返回Y。此标志不受用户指定的列别名的影响。
8: Expression	isExpression	表达式返回Y。
9: Hidden	isHidden	如果使用%PUBLICROWID或SqlRowIdPrivate = 0（默认值）定义表，则RowID字段返回N。否则，RowID字段返回Y。引用%SerialObject嵌入式对象的属性返回Y。
10: Identity	isIdentity	定义为IDENTITY字段的字段返回Y。如果未隐藏RowID，则RowID字段返回Y。
11: KeyColumn	isKeyColumn	定义为主键字段或外键约束目标的字段。 RowID字段返回Y。
12: RowID	isRowId	ROWID和Identity字段返回Y.

扩展列信息元数据表列出了每个选定字段的列名称（SQL名称或列别名），链接属性（链接的持久性类属性）和类型类（数据类型类）。请注意，链接属性列出了持久性类名（不是SQL表名）和属性名（不是列别名）。

- 对于普通表字段（SELECT Name FROM Sample.Person）： Linked Prop=Sample.Person.Name, Type Class=%Library.String.
- 对于表格的RowID (SELECT %ID FROM Sample.Person): Linked Prop= [none], Type Class=Sample.Person.
- 对于表达式，聚合，文字，HostVar或子查询 (SELECT COUNT(Name) FROM Sample.Person): Linked Prop= [none], Type Class=%Library.BigInt.
- 供参考%Serial Object嵌入式对象属性 (SELECT HomeState FROM Sample.Person). Linked

Prop=Sample.Address.State, Type Class=%Library.String.
- 对于引用%SerialObject嵌入式对象的字段(SELECT Home FROM Sample.Person). Linked Prop=Sample.Person.Home, Type Class=Sample.Address.

在此示例中，Sample.Person中的HomeState字段引用%SerialObject类Sample.Address的State属性。

下面的示例返回带有一个形式参数（也就是语句参数）的被调用存储过程的元数据：

```
/// d ##class(PHA.TEST.SQL).MetaData4()
ClassMethod MetaData4()
{
    SET $NAMESPACE="SAMPLES"
    SET mysql = "CALL Sample.SP_Sample_By_Name(?)"
    SET tStatement = ##class(%SQL.Statement).%New()
    SET qStatus = tStatement.%Prepare(.mysql)
    IF qStatus'=1 {
        WRITE "%Prepare failed:" DO $System.Status.DisplayError(qStatus) QUIT
    }
    DO tStatement.%Metadata.%Display()
    WRITE !,"End of metadata"
}
```

它不仅返回列（字段）信息，还返回语句参数，形式参数和对象的值。

以下示例返回具有三个形式参数的的元数据。这三个参数之一用问号（？）指定，使其成为语句参数：

```
/// d ##class(PHA.TEST.SQL).MetaData5()
ClassMethod MetaData5()
{
    SET $NAMESPACE="SAMPLES"
    SET mycall = "CALL personsets(?,'MA')"
    SET tStatement = ##class(%SQL.Statement).%New(0,"sample")
    SET qStatus = tStatement.%Prepare(mycall)
    IF qStatus'=1 {WRITE "%Prepare failed:" DO $System.Status.DisplayError(qStatus) Q
UIT}
    DO tStatement.%Metadata.%Display()
    WRITE !,"End of metadata"
}
```

```
DHC-APP>d ##class(PHA.TEST.SQL).MetaData5()


Columns (SQLRESULTCOL, property 'columns'):

Column Name  Type Prec Scale Null Label        Table        Schema       CType
-----------  ---- ---- ----- ---- ------------ ------------ ------------ -----


Extended Column Info (SQLRESULTCOL)
     Flags: 1:AutoIncrement,2:CaseSensitive,3:Currency,4:ReadOnly,5:RowVersion,
            6:Unique,7:Aliased,8:Expression,9:Hidden,10:Identity,11:KeyColumn,
            12:RowId
```

```
Column Name  Linked Prop           Type Class            Flags
------------ --------------------- --------------------- -----------------------


Statement Parameters (property 'parameters'):

Nbr. Type precision scale nullable colName      columntype
---- ---- --------- ----- -------- ------------ ----------
   1   12     50        0     2    name            1


Formal Parameters (property 'formalParameters'):

Nbr. Type precision scale nullable colName      columntype
---- ---- --------- ----- -------- ------------ ----------
   1    4      4        0     2    _isc_sp_ret_val 5
   2   12     50        0     2    name            1
   3   12     50        0     2    state           1


Objects:

Col Column Name Extent            ExportCall
--- ----------- ----------------- ----------------------------

End of metadata
```

请注意，此元数据不返回任何列信息，但是“ 语句参数 ”，“ 形式参数 ”列表包含列名称和数据类型。

Query参数元数据

使用%SQL.Statement类进行Prepare之后，您可以返回有关查询参数的元数据：输入参数（指定为问号（？）），输入主机变量（指定为：varname）和常量（文字值）。可以返回以下元数据：

- ？参数：parameterCount属性
- ODBC数据类型为？参数：%SQL.StatementMetadata%Display（）实例方法“ 语句参数 ”列表。
- ？，v（：var）和c（常量）参数的列表：%GetImplementationDetails（）实例方法，如成功准备的结果中所述。
- ？，v（：var）和c（常量）参数的ODBC数据类型：formalParameters属性。%SQL.StatementMetadata%Display（）实例方法“ 形式参数 ”列表。
- 查询文本，其中显示以下参数：%GetImplementationDetails（）实例方法，如成功准备结果中所述。

语句元数据%Display（）方法列出了“ 语句参数 ”和“ 形式参数 ”。对于每个参数，它列出了顺序参数号，ODBC数据类型，精度，小数位数，该参数是否可为空（2表示始终提供一个值）及其对应的属性名称（colName）和列类型。

请注意，某些ODBC数据类型以负整数形式返回。

下面的示例按顺序返回每个查询参数（？，：var和常量）的ODBC数据类型。请注意，TOP参数以数据类型12（VARCHAR）而不是数据类型4（INTEGER）返回，因为可以指定TOP ALL：

```
/// d ##class(PHA.TEST.SQL).MetaData6()
ClassMethod MetaData6()
{
    SET myquery = 4
    SET myquery(1) = "SELECT TOP ? Name,DOB,Age+10 "
```

```
    SET myquery(2) = "FROM Sample.Person"
    SET myquery(3) = "WHERE %ID BETWEEN :startid :endid AND DOB=?"
    SET myquery(4) = "ORDER BY $PIECE(Name,',',?)"
    SET tStatement = ##class(%SQL.Statement).%New()
    SET qStatus = tStatement.%Prepare(.myquery)
    IF qStatus'=1 {
        WRITE "%Prepare failed:" DO $System.Status.DisplayError(qStatus) QUIT
    }
    SET prepmeta = tStatement.%Metadata
    WRITE "Number of ? parameters=",prepmeta.parameterCount,!
    SET formalobj = prepmeta.formalParameters
    SET i=1
    WHILE formalobj.GetAt(i) {
        SET prop=formalobj.GetAt(i)
        WRITE prop.colName," type= ",prop.ODBCType,!
        SET i=i+1
    }
    WRITE "End of metadata"
}
```

执行Execute之后，无法从查询结果集元数据中获取参数元数据。在结果集中，所有参数均已解析。因此parameterCount = 0，formalParameters不包含任何数据。

Query结果集元数据

使用%SQL.Statement类执行Execute之后，可以通过调用返回结果集元数据：

- %SQL.StatementResult类的属性。
- %SQL.StatementResult%GetMetadata（）方法，访问%SQL.StatementMetadata类属性。

%SQL.StatementResult属性

执行查询操作后，%SQL.StatementResult返回：
- %StatementType属性返回与最近执行的SQL语句相对应的整数代码。以下是这些整数代码的部分列表：1 = SELECT; 2 = INSERT; 3 = UPDATE; 4 = DELETE or TRUNCATE TABLE; 9 = CREATE TABLE; 15 = CREATE INDEX; 45 = CALL.
- %StatementTypeName计算的属性基于%StatementType返回最近执行的SQL语句的命令名称。此名称以大写字母返回。请注意，TRUNCATE TABLE操作将作为DELETE返回。即使执行了更新操作，INSERT OR UPDATE也将作为INSERT返回。
- %ResultColumnCount属性返回结果集行中的列数。

下面的示例显示这些属性：

```
/// d ##class(PHA.TEST.SQL).MetaData7()
ClassMethod MetaData7()
{
    SET myquery = "SELECT TOP ? Name,DOB,Age FROM Sample.Person WHERE Age > ?"
    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(10,55)
    IF rset.%SQLCODE=0 {
        WRITE "Statement type=",rset.%StatementType,!
```

```
        WRITE "Statement name=",rset.%StatementTypeName,!
        WRITE "Column count=",rset.%ResultColumnCount,!
        WRITE "End of metadata"
    } ELSE {
        WRITE !,"SQLCODE=",rset.%SQLCODE," ",rset.%Message
    }
}
```

%SQL.StatementResult %GetMetadata（）

执行之后，可以使用%SQL.StatementResult %GetMetadata（）方法访问%SQL.StatementMetadata类属性。这些是在Prepare之后由%SQL.Statement%Metadata属性访问的相同属性。

以下示例显示了属性：

```
/// d ##class(PHA.TEST.SQL).MetaData8()
ClassMethod MetaData8()
{
    SET myquery=2
    SET myquery(1)="SELECT Name AS VendorName,LastPayDate,MinPayment,NetDays,"
    SET myquery(2)="AVG(MinPayment),$HOROLOG,%TABLENAME FROM Sample.Vendor"
    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()
    IF rset.%SQLCODE=0 {
    SET rsmeta=rset.%GetMetadata()
    SET x=rsmeta.columns.Count()
    SET x=1
    WHILE rsmeta.columns.GetAt(x) {
        SET column=rsmeta.columns.GetAt(x)
        WRITE !,x," ",column.colName," is data type ",column.ODBCType
        WRITE " with a size of ",column.precision," and scale = ",column.scale
        SET x=x+1 }
    } ELSE {
        WRITE !,"SQLCODE=",rset.%SQLCODE," ",rset.%Message
    }
    WRITE !,"End of metadata"
}
```

请注意，结果集元数据不提供参数元数据。这是因为Execute操作会解析所有参数。因此，在结果集中，parameterCount = 0，而formalParameters不包含任何数据。

审核动态SQL

InterSystems IRIS支持动态SQL语句的可选审核。启用%System /%SQL / DynamicStatement系统审核事件时，将执行动态SQL审核。默认情况下，未启用此系统审核事件。

如果启用%System /%SQL / DynamicStatement，则系统将自动审核在系统范围内执行的每个%SQL.Statement动态语句。审核将信息记录在审核数据库中。

要查看审核数据库，请依次转到管理门户，系统管理，选择安全性，审核，然后查看审核数据库。可以将 “ 事件名称 ”

过滤器设置为DynamicStatement，以将View Audit Database限制为Dynamic SQL语句。审核数据库列出了时间（本地时间戳），用户，PID（进程ID）和事件的描述。说明指定动态SQL语句的类型。例如，SQL SELECT语句（%SQL.Statement）或SQL CREATE VIEW语句（%SQL.Statement）。

通过选择事件的详细信息链接，可以列出其他信息，包括事件数据。事件数据包括执行的SQL语句和该语句的任何参数的值。例如：

```
SELECT TOP ? Name , Age FROM Sample . MyTest WHERE Name %STARTSWITH ?
/*#OPTIONS {"DynamicSQLTypeList":",1"} */
Parameter values:
%CallArgs(1)=5
%CallArgs(2)="Fred"
```

事件数据的总长度（包括语句和参数）为3,632,952个字符。如果该语句和参数长于3632952，则事件数据将被截断。

InterSystems IRIS还支持ODBC和JDBC语句的审核（事件名称= XDBCStatement），以及嵌入式SQL语句的审核（事件名称= EmbeddedStatement）。

#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%83%EF%BC%89