文章
Jingwei Wang · 七月 14, 2022 阅读大约需 13 分钟

InterSystems SQL 的使用 - 第三部分 - 表

创建表

可以通过以下方式定义表：

0. 通过DDL定义表
    - 使用任意数据库管理工具执行DDL（使用ODBC，JDBC连接）

MyApp.Person表可以使用DDL CREATE TABLE语句来定义，指定SQL schema.table名称。成功执行这个SQL语句会生成一个相应的持久化类，包名MyApp，类名Person。当使用DDL命令定义一个表时，你不需要指定USEEXTENTSET或创建一个位图范围索引。InterSystems SQL会自动应用这些设置，并将它们包含在预测的持久化类中。默认情况下，CREATE TABLE在相应的类定义中指定了Final类的关键字，表示它不能有子类。

```
CREATE TABLE MyApp.Person (
  Name VARCHAR(50) NOT NULL,
  SSN VARCHAR(15) DEFAULT 'Unknown',
  DateOfBirth DATE,
  Sex VARCHAR(1)
)
```

- 使用Objectscript执行DDL
    - 在ObjectScript中使用 Embedded SQL.

```
ClassMethod CreateTable() As %String
{
 &sql(CREATE TABLE Sample.Employee (
   EMPNUM              INT NOT NULL,
   NAMELAST            CHAR (30) NOT NULL,
   NAMEFIRST           CHAR (30) NOT NULL,
   STARTDATE           TIMESTAMP,
   SALARY              MONEY,
   ACCRUEDVACATION     INT,
   ACCRUEDSICKLEAVE    INT,
   CONSTRAINT EMPLOYEEPK PRIMARY KEY (EMPNUM)))
 IF SQLCODE=0{ WRITE "Table created" RETURN "Success"}
 ELSEIF SQLCODE=-201 { WRITE "Table already exists"  RETURN SQLCODE}
 ELSE { WRITE "Serious SQL Error, returing SQLCODE " RETURN SQLCODE_
" "_%msg}
}
```

这个方法试图创建一个Sample.Employee表（以及相应的Sample.Employee类）。如果成功，SQLCODE变量被设置为0；如果不成功，SQLCODE包含一个SQL错误代码，表明失败的原因。 像这样的DDL命令，最常见的失败原因是: SQLCODE -99（特权侵犯）。这个错误表明你没有权限执行所需的DDL命令。一般来说，这是因为应用程序没有确定谁是当前用户。你可以使用$SYSTEM.Security.Login()方法以编程方式完成这个任务。

```
DO $SYSTEM.Security.Login(username,password)
```

- 使用 Dynamic SQL.

```
Class Sample.NewT
{
 ClassMethod DefTable(user As %String, pws As %String) [Language = o
bjectscript]
{
   Do ##class(%SYSTEM.Security).Login(user,pws)
   SET myddl = 2
   SET myddl(1)="CREATE TABLE Sample.MyTest "
   SET myddl(2)="(NAME VARCHAR(30) NOT NULL,SSN VARCHAR(15) NOT NULL
)"
   SET tStatement = ##class(%SQL.Statement).%New()
   SET qStatus = tStatement.%Prepare(.myddl)
     IF qStatus '= 1 { WRITE "%Prepare failed:" DO $System.Status.Di
splayError(qStatus) QUIT }
   SET rset = tStatement.%Execute()
   IF rset.%SQLCODE =0 {WRITE "Created a table"}
   ELSEIF rset.%SQLCODE =-201 {WRITE "table already exists"}
ELSE {WRITE "Unexpected error SQLCODE=",rset.%SQLCODE}
}
}
```

◦ 批量执行DDL脚本文件
- 使用**$SYSTEM.SQL.Schema.Run()**方法从终端会话中交互式地导入InterSystems SQL DDL脚本文件
- 使用**$SYSTEM.SQL.Schema.ImportDDL("IRIS")**
方法作为后台作业。这个方法可以导入和执行多个SQL命令，使你能够使用一个txt脚本文件来定义表和视图，并为它们填充数据。
- 针对特定供应商的%SYSTEM.SQL.Schema Load方法。特定供应商的SQL被转换为InterSystems的SQL并执行。错误和不支持的功能被记录在日志文件中。例如下面的Oracle示例。

```
SET $namespace = "MYNAMESPACE"
DO $SYSTEM.SQL.Schema.LoadOracle()
```

1. 通过持久化类定义表 当编译时，这个持久化类会自动投射到一个与类定义相对应的关系表中：每个类代表一个表；每个属性代表一个列，以此类推。这个定义在MyApp schema中创建了MyApp.Person持久化类和相应的SQL表Person。持久类的名称Person被用作SQL表的名称。要提供一个不同的SQL表名，你可以使用SqlTableName类的关键字。

```
Class MyApp.Person Extends %Persistent
{
Parameter USEEXTENTSET = 1; //USEEXTENTSET ??????????1??????????????????globals???
Property Name As %String(MAXLEN=50) [Required];
Property SSN As %String(MAXLEN=15) [InitialExpression = "Unknown"];
Property DateOfBirth As %Date;
Property Sex As %String(MAXLEN=1);
Index BitmapExtent [Extent, Type = bitmap ] //??????????????ID????????????????????
}
```

使用持久化类定义在编译时可以创建的相应的表，但是这个表定义不能使用SQL DDL命令进行修改或删除（或者使用管理门户的Drop操作），这些命令会给你提示"'schema.name'类不启用DDL...")。你必须在表类定义中指定[DdlAllowed]以允许这些操作。

```
Class MyApp.Person Extends %Persistent [DdlAllowed]
```

2. 根据现有的表（或表或视图）来定义和填充一个新的表。 你指定一个查询和一个新的表名。现有的表名和/或新的表名可以是限定的或非限定的。查询可以包含JOIN语法。查询可以提供列名别名，成为新表中的列名。可以使用：
    - CREATE TABLE AS SELECT命令

      ```
      CREATE TABLE Sample.YoungPeople
      AS SELECT Name,Age
      FROM Sample.People
      WHERE Age < 21
      ```

    - $SYSTEM.SQL.Schema.QueryToTable()方法执行。

      ```
      DO $SYSTEM.SQL.Schema.QueryToTable("SELECT Name,Age,AVG(Age) AS AvgInit F
      ROM Sample.Person WHERE Age < 21","Sample.Youth",1,.errors)
      ```

        - QueryToTable()复
          制了现有表的DDL定义，并将其指定为
          新表的名称。它
          复制查询中指定的字段的定义，包括数据类型、最大长度和最小值/最大值，但不复制字段的数据约束，如默认值、要求值或唯一值。它不会从一个字段复制引用到另一个表中。
          如果查询指定SELECT *或SELECT %ID，原始表的RowID字段将被复制为数据类型为整数的非必填、非唯一的数据字段。QueryToTable()为新表生成了一个唯一的RowID字段。如果复制的RowID被命名为ID，生成的RowID被命名为ID1。
        - QueryToTable()为这个新表创建一个相应的持久化类。该持久化类被定义为DdlAllowed。新表的所有者是当前用户。
          新表被定
          义为默认存储=YES
          和支持位图索引=YES，不管源表中的
          这些设置如何。
          为新表创建的唯一索引是IDKEY索引，没有生成位图范围索引。被复制的字段的索引定义不会被复制到新表中。

$SYSTEM.SQL.Schema.TableExists()方法可以用来确定一个表的名字是否已经存在。

RowID

在SQL中，每条记录都由一个唯一的整数值来标识，称为RowID。

在InterSystems SQL中，你不需要指定一个RowID字段。当你创建一个表并指定所需的数据字段时，会自动创建一个RowID字段。这个RowID在内部使用，但没有被映射到一个类属性。默认情况下，只有当一个持久化类被投射到一个SQL表时，它的存在才是可见的。在这个投射表中，会出现一个额外的RowID字段。

默认情况下，这个字段被命名为 "ID "并被分配到第1列，且这个字段是自增的，不会被重复使用。因此，如果记录被插入和删除，RowID值将是升序的数字序列，但可能不是数字连续的。**且ALTER
TABLE不能修改或删除RowID字段的定义。**RowID计数器可以通过**TRUNCATE
TABLE**
命令被重置，但它不会被DELETE命令重置，即使DELETE命令删除了表中的所有记录。如果没有数据被插入到表中，或者TRUNCATE TABLE被用来删除所有表的数据，IdLocation存储关键字的global是未定义的。

默认情况下，InterSystems IRIS将这个字段命名为
"ID"，然而这个字段名并不保留。RowID字段名在每次编译表的时候都会重新建立。如果用户定义了一个名为 "ID "的字段，当表被编译时，InterSystems IRIS将RowID命名为 "ID1"。例如，如果用户随后使用ALTER
TABLE来定义一个名为 "ID1 "的字段，表的编译就会将RowID重命名为 "ID2"，以此类推。在一个持久化的类定义中，你可以使用SqlRowIdName类关键字来直接指定这个类所投射的表的RowID字段名。由于这些原因，应该避免用名字来引用RowID字段。

InterSystems
SQL提供了%ID伪列名（别名），它总是返回RowID值，不管分配给RowID的字段名是什么。(InterSystems

TSQL提供了$IDENTITY伪列名，它也做同样的事情)。

默认情况下，使用CREATE TABLE定义的表使用$SEQUENCE执行ID分配，允许多个进程同时快速填充表。当$SEQUENCE被用来填充表时，一个RowID值的序列被分配给一个进程，然后按顺序分配它们。因为并发的进程使用他们自己分配的序列来分配RowID，所以不能假设由一个以上的进程插入的记录是按照插入的顺序进行的。

当使用CREATE TABLE创建一个表时，RowID默认是隐藏的。一个隐藏的字段不会被SELECT *显示，并且是PRIVATE。当你创建一个表的时候，你可以指定%PUBLICROWID关键字来使RowID不被隐藏并且是公开的。这个可选的%PUBLICROWID关键字可以在CREATE TABLE逗号分隔的表元素列表中的任何地方指定，但不能在ALTER TABLE中指定。

当创建一个投射为表的持久化类时，RowID默认不会被隐藏。它通过SELECT *显示，并且是公共的。你可以通过指定类的关键字SqlRowIdPrivate来定义一个RowID为隐藏和PRIVATE的持久化类。

作为**外键**引用的RowID必须是**公共的**。默认情况下，一个具有公共RowID的表不能被用作源表或目标表，例如使用INSERT INTO Sample.DupTable SELECT * FROM Sample.SrcTable 将数据复制到一个重复的表。

你可以使用管理门户SQL界面来查看RowID是否被隐藏。

基于其他字段的RowID

通过定义一个投射表的持久化类，你可以将RowID定义为来自一个字段或一个字段组合的值。要做到这一点，用IdKey索引关键字指定一个索引。例如，通过PatientName字段的值指定索引定义

```
IdxId On PatientName [IdKey];
```

或者通过PatientName和SSN字段的合并值指定索引定义

```
IdxId On (PatientName,SSN) [IdKey]
```

但是，基于字段的RowID比采取系统分配的连续正整数的RowId效率低。

在INSERT中：为构成RowId的字段或字段组合指定的值必须是唯一的。指定一个非唯一的值会产生一个SQLCODE -119 "UNIQUE或PRIMARY KEY约束在INSERT时唯一性检查失败"。

在UPDATE中：默认情况下，组成RowId的每个字段的值是不可修改的。试图修改这些字段之一的值会产生一个SQLCODE -107 "不能UPDATE RowID或基于字段的RowID"。

当一个RowID基于多个字段时，RowID的值是由||操作符连接的每个组成字段的值。例如，Ross,Betsy||123-45-6789。InterSystems
IRIS试图确定基于多个字段的RowID的最大长度；如果它不能确定最大长度，RowID长度默认为512。

主键

InterSystems IRIS提供两种方法来唯一地识别表中的行：**RowID**和**主键**。

可选的主键是一个有意义的值，应用程序可以用它来唯一地识别表中的一行（例如在连接中）。一个主键可以是用户指定的数据字段，也可以是一个以上的数据字段的组合。主键值必须是唯一的，但不要求是整数值。RowID是一个内部使用的整数值，用于识别表中的一行。通常情况下，主键是一个由应用程序生成的值，而RowID是一个由InterSystems IRIS生成的唯一整数值。

系统会自动创建一个master

map来访问使用RowID字段的数据行。如果你定义了一个主键字段，系统会自动创建并维护一个主键索引。

显然，有两个不同的字段和索引来识别行，这种双重性不一定是好事。你可以通过以下两种方式中的任何一种解决单一的行标识符和索引：

- 使用应用程序生成的主键值作为IDKEY。你可以通过在类定义中使用关键字PrimaryKey和IdKey来识别主键索引，当然，也可以从DDL中这样做。这使得主键索引成为表的主映射。因此，主键将被用作行的主要内部地址。**如果主键由一个以上的字段组成，或者主键值不是整数，这可能会降低效率**。
- 不要使用应用程序生成的主键值，而是使用应用程序内系统生成的RowID整数作为应用程序使用的主键（例如在连接中）。这样做的好处是，整数的RowID适合于更有效的处理，包括使用位图索引。

根据应用程序的性质，你可能希望解决一个单一的行标识符和索引，或者为应用程序生成的主键和系统生成的RowID设置单独的索引。

特殊字段：RowVersion字段 , 自增字段, 和 Serial 计数器字段

InterSystems SQL支持三种特殊用途的数据类型，用于自动递增计数器的值。这三种数据类型都是扩展%Library.BigInt数据类型类的子类。

%Library.RowVersion: 计算对所有RowVersion表的插入和更新的命名空间。只有包含ROWVERSION字段的表的插入和更新才会增加这个计数器。ROWVERSION值是唯一的，不可修改的。这个全命名空间的计数器从不重置。可以通过指定一个数据类型为ROWVERSION（%Library.RowVersion）的字段来创建一个RowVersion字段。你只能在每个表中指定一个ROWVERSION数据类型字段。试图创建一个有一个以上ROWVERSION字段的表会导致5320编译错误。RowVersion字段不应该被包含在唯一键或主键中。RowVersion字段不能成为IDKey索引的一部分。

%Library.Counter
（也被称为SERIAL计数器字段）。计算插入到表中的次数。默认情况下，这个字段接收一个自动递增的整数。然而，用户可以为这个字段指定一个非零的整数值。用户可以指定一个重复的值。如果用户提供的值大于系统提供的最高值，自动递增计数器被设置为从用户指定的值开始递增。这个计数器通过TRUNCATE
TABLE命令被重置
为1。它不会被DELETE命令重置，即使DELETE命令删除了表中的所有记录。**分片的表不能有SERIAL计数器字段**。

%Library.AutoIncrement: 计算插入到表中的次数。默认情况下，这个字段接收一个自动递增的整数。然而，用户可以为这个字段指定一个非零的整数值。用户可以指定一个重复的值。指定一个用户值对自动递增计数器没有影响。

所有这三个字段和IDENTITY字段都返回AUTOINCREMENT = YES，如下面的例子所示。

查看表定义

0. 表信息 **INFORMATION.SCHEMA.TABLES**
   持久化类显示当前命名空间中所有表（和视图）的信息。包括模式和表名，表的所有者，以及是否可以插入新记录。TABLETYPE属性表明它是一个基础表还是一个视图。

   ```
   SELECT Table_Type,Table_Schema,Table_Name,Owner FROM INFORMATION_SCHEMA.TABLES
   ```

   INFORMATION.SCHEMA.CONSTRAINTTABLEU
   SAGE持久化类为当前命名空间中的每个表定义的每个主键（显性或隐性）、外键或唯一约束。
   INFORMATION.SCHEMA.KEYCOLUMNUSAGE为当前命名空间中的每个表的每个已定义的约束。
   也可以使用管理门户SQL界面中的目录详情来查看表信息。
1. 列信息
   - GetColumn 方法

```
SET stat = ##class(%SYSTEM.SQL.Schema).GetAllColumns("MyApp.Person",.byna
me,.bynum)
IF stat=1
{
 SET i=1
 WHILE $DATA(bynum(i))
{
   WRITE "name is ",bynum(i), "column is ", i, !
}
}ELSE{ WRITE "GetAllColumns() cannot locate specified table"}
```

生成结果：

```
name is ID   col num is 1
name is Age   col num is 2
name is Home   col num is 3
name is Name   col num is 4
```

- SQL脚本 **INFORMATION_SCHEMA.COLUMNS** 可以列出指定schema的所有列名:

```
SELECT TABLE_NAME,COLUMN_NAME,ORDINAL_POSITION,DATA_TYPE,CHARACTER_MAXIMU
M_LENGTH,COLUMN_DEFAULT,IS_NULLABLE,UNIQUE_COLUMN,PRIMARY_KEY
FROM INFORMATION_SCHEMA.COLUMNS WHERE TABLE_SCHEMA='MyApp'
?
??
?
SELECT TOP 0 * FROM tablename.
```

2. Constraints信息
   - **INFORMATION.SCHEMA.TABLECONSTRAINTS**
     持久化类列出了表名，约束类型和约束名称。约束类型包括UNIQUE, PRIMARY KEY, 和 FOREIGN KEY。

     ```
     SELECT Table_Schema,Table_Name,Constraint_Type,Constraint_Name FROMINFORM
     ATION_SCHEMA.TABLE_CONSTRAINTS
     ```

   - **INFORMATION.SCHEMA.CONSTRAIN TCOLUMNUSAGE**
     持久化类列出了表名、列名和约束名称。如果一个约束涉及到多个列，那么每一个列都会列出一个单独的项目。

     ```
     SELECT Table_Schema,Table_Name,Column_Name,Constraint_Name FROMINFORMATIO
     N_SCHEMA.CONSTRAINT_COLUMN_USAGE
     ```

   - **INFORMATION.SCHEMA. REFERENTIALCONSTRAINTS**持久化类列出了外键约束，包括引用表（CONSTRAINTSCHEMA, CONSTRAINTTABLENAME），被引用表（UNIQUECONSTRAINTSCHEMA, UNIQUECONS TRAINTTABLE），外键名称（CONSTRAINTNAME），以及UPDATE和DELETE参考动作（UPD ATERULE，DELETERULE），值为NO ACTION，SET DEFAULT，SET NULL，或CASCADE。

     ```
     SELECT Constraint_Table_Name,Unique_Constraint_Table,Constraint_Name,Upda
     te_Rule,Delete_Rule FROMINFORMATION_SCHEMA.REFERENTIAL_CONSTRAINTS
     ```

连接外部表

在InterSystems SQL中，你也可以有 "外部表"，即在SQL字典中定义的表，但存储在一个外部关系数据库中。外部表的作用就像它们是本地的InterSystems
IRIS表一样：你可以对它们发出查询，执行INSERT、UPDATE和DELETE操作。

对外部数据库的访问是由**InterSystems SQL Gateway**提供的，它使用**ODBC**或**JDBC**提供透明的连接。

#SQL #InterSystems IRIS for Health

源URL:https://cn.community.intersystems.com/post/intersystems-sql-%E7%9A%84%E4%BD%BF%E7%94%A8-%E7%AC%AC%E4%B8%89%E9%83%A8%E5%88%86-%E8%A1%A8