文章
姚 鑫 · 三月 8, 2021 阅读大约需 9 分钟

第五章 SQL定义表（三）

第五章 SQL定义表（三）

使用DDL定义表

可以使用标准DDL命令在InterSystems SQL中定义表：

InterSystems SQL中可用的DDL命令

- ALTER命令 ALTER TABLE，ALTER VIEW
- CREATE 命令 CREATE TABLE，CREATE VIEW，CREATE INDEX，CREATE TRIGGER
- DROP 命令 DROP TABLE，DROP VIEW，DROP INDEX，DROP TRIGGER

可以通过多种方式执行DDL命令，包括：

- 使用动态SQL。
- 使用嵌入式SQL。
- 使用DDL脚本文件。
- 使用ODBC调用。
- 使用JDBC调用。

在嵌入式SQL中使用DDL

在ObjectScript方法或例程中，可以使用嵌入式SQL来调用DDL命令。

例如，以下方法创建一个Sample.Employee表：

```
/// d ##class(PHA.TEST.SQL).CreateTable()
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, returning SQLCODE"  RETURN SQLCODE_" "_%msg}
}
```

```
DHC-APP>d ##class(PHA.TEST.SQL).CreateTable()
Table already exists
```

调用此方法时，它将尝试创建Sample.Employee表（以及相应的Sample.Employee类）。如果成功，则将SQLCODE变量设置为0。如果失败，则SQLCODE包含指示错误原因的SQL错误代码。

这样的DDL命令失败的最常见原因是：
- SQLCODE -99（违反权限）：此错误表明没有执行所需DDL命令的权限。通常，这是因为应用程序尚未确定当前用户是谁。可以使用$SYSTEM.Security.Login（）方法以编程方式执行此操作：

```
DHC-APP>w $SYSTEM.Security.Login("yx","123456")
0
```

SQLCODE -201（表或视图名称不是唯一的）：此错误表明正在尝试使用已经存在的表的名称创建新表。

使用类方法执行DDL

在ObjectScript中，可以使用Dynamic SQL%SQL.Statement对象使用Dynamic SQL准备和执行DDL命令。

下面的示例定义了一个使用动态SQL创建表的类方法：

```
 ClassMethod DefTable(user As %String,pwd As %String) As %Status [Language=objectscri
pt]
{
    DO ##class(%SYSTEM.Security).Login(user,pwd)
    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 tStatus=tStatement.%Prepare(.myddl)
    IF qStatus'=1 {WRITE "%Prepare failed:" DO $System.Status.DisplayError(qStatus) Q
UIT}
    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}
}
```

与嵌入式SQL示例一样，如果当前没有用户登录，则此方法将失败。

通过导入和执行DDL脚本定义表

可以使用IRIS（）方法从终端会话中交互式地导入InterSystems SQL DDL脚本文件，也可以使用DDLImport（“IRIS”）方法作为后台作业来导入InterSystems SQL DDL脚本文件。此方法可以导入和执行多个SQL命令，使可以使用txt脚本文件来定义表和视图，并用数据填充它们。

如果要将表从另一供应商的关系数据库迁移到InterSystems IRIS，则文本文件中可能包含一个或多个DDL脚本。InterSystems IRIS提供了几种%SYSTEM.SQL方法来帮助将此类表加载到InterSystems IRIS中。可以使用通用的DDLImport（）方法或特定供应商的%SYSTEM.SQL方法。供应商特定的SQL转换为InterSystems SQL并执行。错误和不支持的功能记录在日志文件中。

例如，从ObjectScript命令行加载一个Oracle DDL文件:

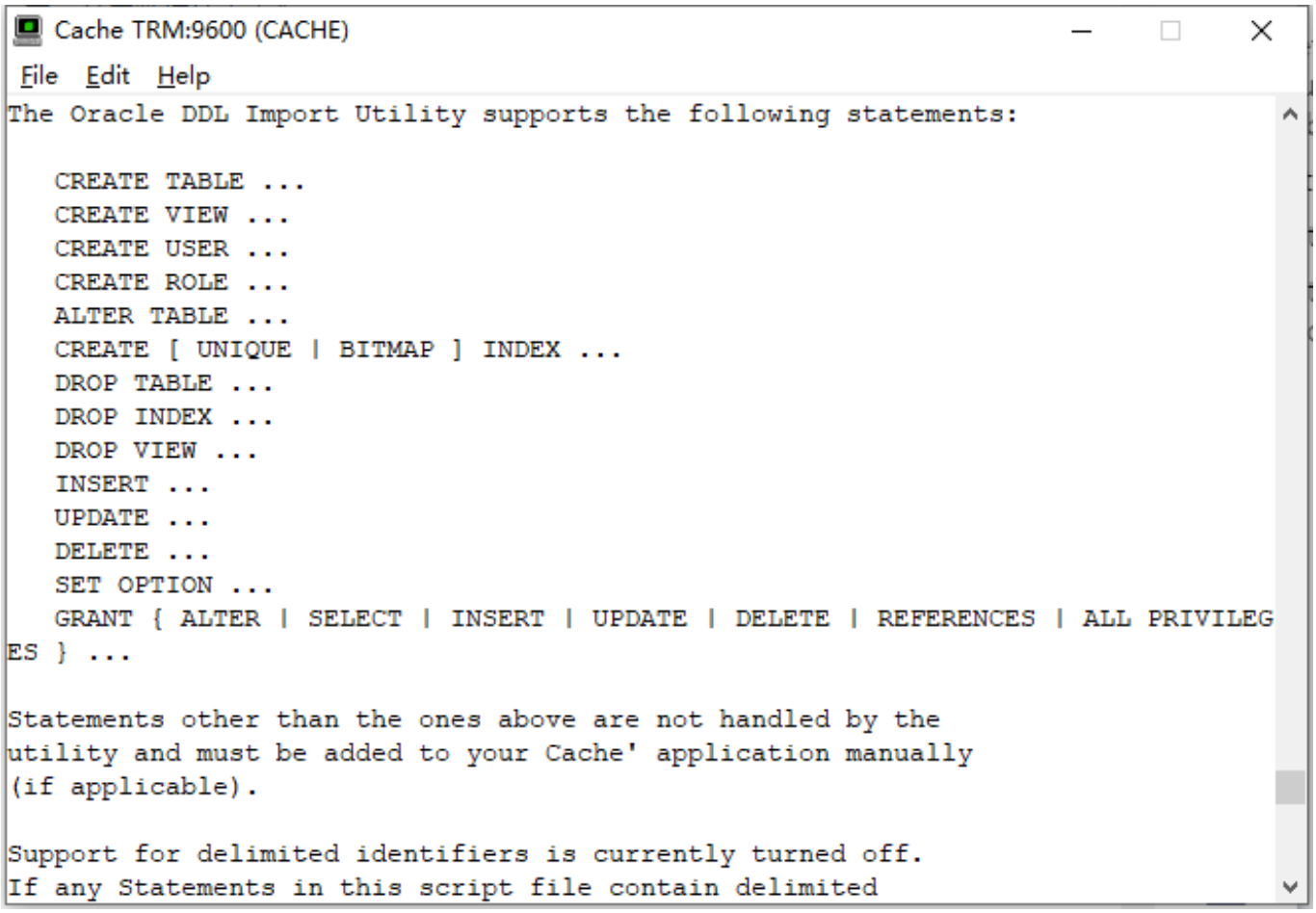
1. 使用InterSystems IRIS启动器菜单中的 “ 终端 ” 命令启动终端会话。
2. 切换到希望在其中加载表定义的名称空间：

```
SET $namespace = "MYNAMESPACE"
```

3. 调用所需的DDL导入方法：

```
DO $SYSTEM.SQL.Oracle()
```

并按照终端上显示的说明进行操作。



定义分片表

创建分片表有三个要求。
1. 许可证密钥必须支持分片。使用管理门户，系统管理，许可，许可证密钥显示当前许可证或激活新许可证。
2. 必须在IRIS实例上启用分片。必须具有%AdminSecure特权才能启用分片。使用 “ 管理门户 ”，“ 系统管理 ”，“ 配置 ”，“ 系统配置 ”，“ 分片配置 ” 来选择 “ 启用分片 ” 按钮。这使当前的InterSystems IRIS实例可以在分片群集中使用。选择 “ 为任何角色启用此实例 ” 或 “ 仅对碎片主机角色启用此实例 ”。按确定。重新启动您的InterSystems IRIS实例。
3. 必须在IRIS实例上部署分片群集。此分片群集包含一个分片主名称空间。如果未为分片配置当前名称空间，则尝试定义分片表失败，并显示错误＃9319：当前名称空间%1没有配置分片。

然后，可以在Shard Master命名空间中定义一个分片表，该表已定义为分片集群的一部分。可以使用CREATE TABLE通过指定分片键来定义分片表。或者，可以创建一个持久化类，该持久化类投影到分片表。

通过查询现有表定义表

可以使用$SYSTEM.SQL.QueryToTable（）方法基于一个或多个现有表来定义和填充新表。指定一个查询和一个新的表名称。现有表名和/或新表名可以是合格的或不合格的。该查询可以包含JOIN语法。该查询可以提供列名别名，这些别名将成为新表中的列名。

1. QueryToTable（）复制现有表的DDL定义，并为其指定指定的新表名。它复制查询中指定的字段的定义，包括数据类型，maxlength和minval /
   maxval。它不复制字段数据约束，例如默认值，必需值或唯一值。它不会将引用从字段复制到另一个表。

如果查询指定SELECT
*或SELECT%ID，则将原始表的RowID字段复制为数据类型为整数的非必需，非唯一数据字段。
QueryToTable（）为新表生成唯一的RowID字段。如果复制的RowID名为ID，则生成的RowID名为ID1。

QueryToTable（）为此新表创建一个对应的持久化类。持久类定义为DdlAllowed。新表的所有者是当前用户。

不管源表中的这些设置如何，新表都将使用Default Storage = YES定义，并且Supports Bitmap Indices = YES。

为新表创建的唯一索引是IDKEY索引。没有位图范围索引生成。复制字段的索引定义不会复制到新表中。

2. QueryToTable（）然后使用查询选择的字段中的数据填充新表。它将表格的 “ 范围大小 ” 设置为100,000。它估计IDKEY块计数。运行 “ 音调表 ” 以设置实际的 “ 范围大小 ” 和 “ 块计数 ”，以及每个字段的 “ 选择性 ” 和 “ 平均字段小 ” 值。

QueryToTable（）既创建表定义，又用数据填充新表。如果只希望创建表定义，请在查询WHERE子句中指定一个不选择任何数据行的条件。例如，WHERE Age < 20 AND Age > 20.

下面的示例从Sample.Person复制 “ 名称 ” 和 “ 年龄 ” 字段，并创建一个AVG（Age）字段。这些字段定义用于创建名为Sample.Youth的新表。然后，该方法where Age < 21.
的那些记录的Sample.Person数据填充Sample.Youth。AvgInit字段包含创建表时所选记录的合计值。

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

```
DHC-APP>  DO $SYSTEM.SQL.QueryToTable("SELECT Name,Age,AVG(Age) AS AvgInit FROM Sampl
e.Person WHERE Age < 21","Sample.Youth",1,.errors)

Preparing query...
Creating class...
Compiling class...
Copying data...
```

外部表

在InterSystems SQL中，还可以具有 “ 外部表 ”，这些表在SQL词典中定义但存储在外部关系数据库中。外部表的行为就像它们是本机InterSystems IRIS表一样：可以对它们发出查询并执行INSERT，UPDATE和DELETE操作。
InterSystems SQL网关提供对外部数据库的访问，该网关使用ODBC或JDBC提供透明的连接。

List表

INFORMATION.SCHEMA.TABLES持久类显示有关当前名称空间中所有表（和视图）的信息。它提供了许多属性，包括模式和表名称，表的所有者以及是否可以插入新记录。 TABLETYPE属性指示它是基表还是视图。

以下示例返回当前名称空间中所有表和视图的表类型，架构名称，表名称和所有者：

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

INFORMATION.SCHEMA.CONSTRAINTTABLEUSAGE持久类为为当前名称空间中的每个表定义的每个主键（显式或隐式），外键或唯一性约束显示一行。 INFORMATION.SCHEMA.KEYCOLUMNUSAGE为定义为当前名称空间中每个表的这些约束之一的一部分的每个字段显示一行。

列出列名和数字

可以通过以下四种方式列出指定表的所有列名（字段名）：
- GetColumns（）方法。这列出了所有列名和列号，包括隐藏的列。 ID（RowID）字段可以隐藏也可以不隐藏。 xclassname列始终是隐藏的；除非使用Final class关键字定义了持久类，否则它将自动定义。
- 管理门户网站SQL界面（系统资源管理器，SQL）架构内容的“ 目录详细信息 ”选项卡。它列出了所有列名和列号（包括隐藏的列）以及其他信息，包括数据类型和指示列是否被隐藏的标志。
- SELECT TOP 0 * FROM表名。这将按列号顺序列出所有非隐藏的列名。请注意，由于隐藏的列可以按列号顺序出现在任何位置，因此您无法通过计算这些非隐藏的列名来确定列号。
- INFORMATION.SCHEMA.COLUMNS持久类为当前名称空间中每个表或视图中的每个非隐藏列列出一行。 INFORMATION.SCHEMA.COLUMNS提供了大量属性，用于列出表和视图列的特征。请注意，ORDINALPOSITION与列号不同，因为不计算隐藏字段。 GetColumns（）方法同时计算隐藏字段和非隐藏字段。

下面的示例使用INFORMATION.SCHEMA.COLUMNS列出一些列属性：

```
SELECT TABLE_NAME,COLUMN_NAME,ORDINAL_POSITION,DATA_TYPE,CHARACTER_MAXIMUM_LENGTH,
       COLUMN_DEFAULT,IS_NULLABLE,UNIQUE_COLUMN,PRIMARY_KEY
       FROM INFORMATION_SCHEMA.COLUMNS WHERE TABLE_SCHEMA='Sample'
```

GetColumns（）方法

要以列号顺序列出表中的列名，可以使用GetColumns（）方法，如下所示：

```
/// d ##class(PHA.TEST.SQL).GetColumn()
ClassMethod GetColumn()
{
    SET stat=##class(%SYSTEM.SQL).GetColumns("Sample.Person",.byname,.bynum)
    IF stat=1 {
        SET i=1
        WHILE $DATA(bynum(i)) {
            WRITE "name is ",bynum(i),"   col num is ",i,!
            SET i=i+1
            }
    }
    ELSE { WRITE "GetColumns??????????" }
}
```

GetColumns（）列出所有已定义的列，包括隐藏的列。如果表引用了嵌入式%SerialObject类，则GetColumns（）

首先列出持久性类中的所有列，包括引用%SerialObject的属性，然后列出所有%SerialObject属性。在下面的GetColumns（）结果中显示了这一点：

```
DHC-APP>d ##class(PHA.TEST.SQL).GetColumn()
name is ID   col num is 1
name is Age   col num is 2
name is DOB   col num is 3
name is FavoriteColors   col num is 4
name is Home   col num is 5
name is Name   col num is 6
name is Office   col num is 7
name is SSN   col num is 8
name is Spouse   col num is 9
name is x__classname   col num is 10
name is Home_City   col num is 11
name is Home_State   col num is 12
name is Home_Street   col num is 13
name is Home_Zip   col num is 14
name is Office_City   col num is 15
name is Office_State   col num is 16
name is Office_Street   col num is 17
name is Office_Zip   col num is 18
```

还可以使用此方法确定指定列名的列号，如下所示：

```
/// d ##class(PHA.TEST.SQL).GetColumn1()
ClassMethod GetColumn1()
{
    SET stat=##class(%SYSTEM.SQL).GetColumns("Sample.Person",.byname)
    IF stat=1 {
        WRITE "Home_State is column number ",byname("Home_State"),!
    } ELSE {
        WRITE "GetColumns??????????"
    }
}
```

```
DHC-APP>d ##class(PHA.TEST.SQL).GetColumn1()
Home_State is column number 12
```

#Caché #InterSystems IRIS #InterSystems IRIS for Health

源URL:
https://cn.community.intersystems.com/post/%E7%AC%AC%E4%BA%94%E7%AB%A0-sql%E5%AE%9A%E4%B9%89%E8%A1%A8%EF%BC%88%E4%B8%89%EF%BC%89