文章
姚 鑫 · 九月 13, 2021 阅读大约需 9 分钟

第十五章 SQL命令 CREATE TABLE（二）

第十五章 SQL命令 CREATE TABLE（二）

全局临时表

指定GLOBAL TEMPORARY关键字将表定义为全局临时表。表定义是全局的(对所有进程都可用)；表数据是临时的(在进程期间持续存在)。相应的类定义包含一个附加的类参数SQLTABLETYPE=“ GLOBAL TEMPORARY ”。与标准的IRIS表一样，ClassType=Persistent，并且类包含Final关键字，表示它不能有子类。

无论哪个进程创建临时表，临时表的所有者都会自动设置为PUBLIC。这意味着所有用户都可以访问缓存的临时表定义。例如，如果存储过程创建了一个临时表，则允许调用该存储过程的任何用户都可以访问该表定义。这仅适用于临时表定义；临时表数据特定于调用，因此只能由当前用户进程访问。

全局临时表的表定义与基表相同。全局临时表必须具有唯一的名称；尝试为其提供与现有基表相同的名称会导致SQLCODE-201错误。该表将一直存在，直到显式删除(使用DROP TABLE)。可以使用ALTER TABLE更改表定义。

下面的嵌入式SQL示例创建一个全局临时表：

```
ClassMethod CreateTable3()
{
    d $SYSTEM.Security.Login("_SYSTEM","SYS")
    n SQLCODE,%msg
    &sql(
        CREATE GLOBAL TEMPORARY TABLE TempEmp
        (
            EMPNUM     INT NOT NULL,
            NAMELAST   CHAR(30) NOT NULL,
            NAMEFIRST  CHAR(30) NOT NULL,
            CONSTRAINT EMPLOYEEPK PRIMARY KEY (EMPNUM)
        )
    )
    if SQLCODE=0 {
        w !,"???"
    } else {
        w !,"SQLCODE=",SQLCODE,": ",%msg
    }
}
```

```
///
Class User.TempEmp Extends %Persistent [ ClassType = persistent, DdlAllowed, Final, O
wner = {_PUBLIC}, ProcedureBlock, SqlRowIdPrivate, SqlTableName = TempEmp ]
{

Property EMPNUM As %Library.Integer(MAXVAL = 2147483647, MINVAL = -2147483648) [ Requ
ired, SqlColumnNumber = 2 ];
```

```
Property NAMELAST As %Library.String(MAXLEN = 30) [ Required, SqlColumnNumber = 3 ];

Property NAMEFIRST As %Library.String(MAXLEN = 30) [ Required, SqlColumnNumber = 4 ];

Parameter SQLTABLETYPE = "GLOBAL TEMPORARY";

/// DDL Primary Key Specification
Index EMPLOYEEPK On EMPNUM [ PrimaryKey, Type = index, Unique ];
}
```

%DESCRIPTION, %FILE, %EXTENTSIZE / %NUMROWS, %ROUTINE

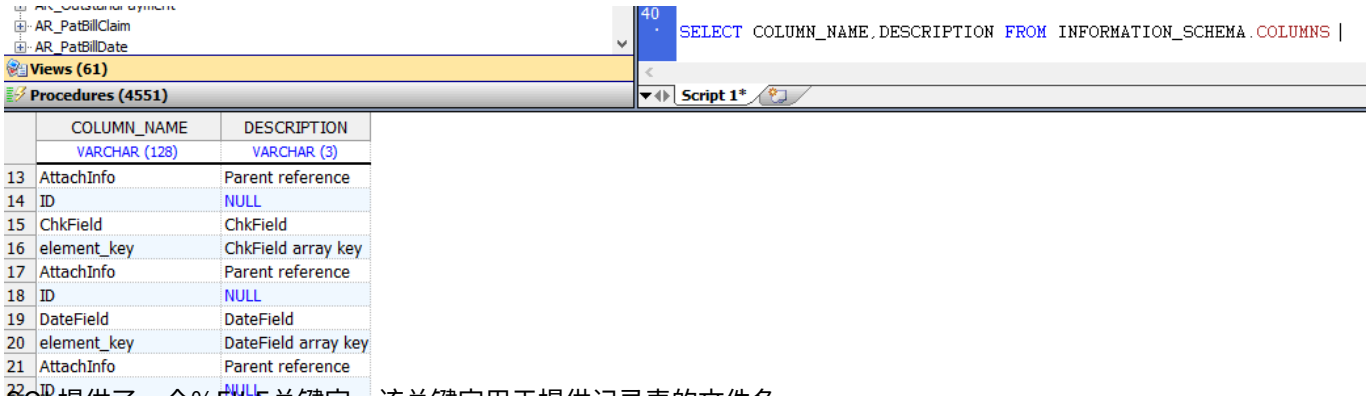
这些可选关键字短语可以在逗号分隔的表元素列表中的任何位置指定。

SQL提供了一个%DESCRIPTION关键字，可以使用该关键字为记录表或字段提供描述。%DESCRIPTION后面跟着用单引号括起来的文本字符串。这个文本可以是任意长度的，可以包含任何字符，包括空格。
(描述中的单引号字符由两个单引号表示。
例如: “ Joe' s Table ” 。)
一个表可以有%DESCRIPTION。
表的每个字段都可以有自己的%DESCRIPTION，在数据类型之后指定。
如果为一个表指定多个表宽%DESCRIPTION, IRIS将发出SQLCODE -82错误。
如果您为一个字段指定了多个%DESCRIPTION，系统只保留最后一个指定的%DESCRIPTION。
不能使用ALTER TABLE更改现有的描述。

在相应的持久化类定义中，在对应的类(表)或属性(字段)语法之前的一行中出现了以三个斜杠开头的描述。
例如:/// Joe's Table。
在对应持久化类的类引用中，表描述出现在类名和SQL表名之后;
字段说明出现在相应的属性语法之后。

可以使用INFORMATION.SCHEMA.TABLES或INFORMATION.SCHEMA.COLUMNS的DESCRIPTION属性显示%DESCRIPTION文本。
例如:

```
SELECT COLUMN_NAME,DESCRIPTION FROM INFORMATION_SCHEMA.COLUMNS WHERE TABLE_NAME='MyTable'
```



	COLUMN_NAME	DESCRIPTION
	VARCHAR (128)	VARCHAR (3)
13	AttachInfo	Parent reference
14	ID	NULL
15	ChkField	ChkField
16	element_key	ChkField array key
17	AttachInfo	Parent reference
18	ID	NULL
19	DateField	DateField
20	element_key	DateField array key
21	AttachInfo	Parent reference
22	ID	NULL

SQL提供了一个%FILE关键字，该关键字用于提供记录表的文件名。
%FILE后面跟着用单引号括起来的文本字符串。
一个表定义只能有一个%FILE关键字;
指定多个会产生SQLCODE -83错误。

SQL提供了可选的%EXTENTSIZE和%NUMROWS关键字，它们用于存储一个整数，记录该表中预期的行数。
这两个关键词是同义词;
%EXTENTSIZE是首选术语。
当创建一个表来保存已知的数据行数时，特别是当初始的行数不太可能随后更改时(比如包含州和省的表)，设置%EXTENTSIZE可以节省空间并提高性能。
如果未指定，则标准表的初始分配值为100,000，临时表的初始分配值为50。
一个表定义只能有一个%EXTENTSIZE或%NUMROWS关键字;
指定多个会导致SQLCODE -84错误。
一旦用数据填充了表，就可以通过运行Tune table将这个%EXTENTSIZE值更改为实际的行数。

SQL提供了一个%ROUTINE关键字，它允许为为这个基表生成的例程指定例程名称前缀。
%ROUTINE后面跟着用单引号括起来的文本字符串。
例如，%ROUTINE 'myname'在名为myname1、myname2等的例程中生成代码。
不能从%ROUTINE调用用户定义的(“ 外部 ”)函数。
一个表定义只能有一个%ROUTINE关键字;
指定多个会导致SQLCODE -85错误。
在Studio中，例程名称前缀显示为SqlRoutinePrefix值。

仅支持兼容性选项

SQL仅接受以下CREATE TABLE选项用于解析，以帮助将现有SQL代码转换为 SQL。
这些选项不提供任何实际的功能。

```
{ON | IN} dbspace-name

LOCK MODE [ROW | PAGE]

[CLUSTERED | NONCLUSTERED]

WITH FILLFACTOR = literal

MATCH [FULL | PARTIAL]

CHARACTER SET identifier

COLLATE identifier  /* But note use of COLLATE keyword, described below */

NOT FOR REPLICATION
```

字段定义

在表名之后，一组圆括号包含表中所有字段(列)的定义。字段定义用逗号分隔。按照惯例，每个字段定义通常在单独的行上显示，并使用缩进；建议这样做，但不是必需的。定义最后一个字段后，请记住为字段定义提供右括号。

字段定义的各个部分由空格分隔。首先列出字段名称，然后列出其数据特征。字段的数据特征按以下顺序显示：数据类型、(可选)数据大小，然后是(可选)数据约束。然后，可以附加一个可选的字段%DESCRIPTION来记录该字段。

字段定义可以引用定义多个字段(属性)的现有嵌入式串行对象，而不是定义字段。字段名后面是串行对象的包名和类名。例如，Office Sample.Address。不要指定数据类型或数据约束；可以指定%DESCRIPTION。不能使用CREATE TABLE创建嵌入式串行对象。

注：我们建议避免创建列超过400列的表。重新设计数据库，以便：这些列变成行；列在几个相关的表中划分；或者数据以字符流或位流的形式存储在较少的列中。

字段名称

字段名遵循标识符约定，具有与表名相同的命名限制。应避免以%字符开头的字段名(允许以%z或%Z开头的字段名)。字段名称不应超过128个字符。默认情况下，字段名是简单标识符。它们不区分大小写。尝试创建与同一表中的另一个字段仅在字母大小写上不同的字段名会生成SQLCODE-306错误。

IRIS使用该字段名生成相应的类属性名。特性名称仅包含字母数字字符(字母和数字)，最大长度为96个字符。要生成此属性名，
IRIS首先从字段名中删除标点符号，然后生成96个(或更少)字符的唯一标识符。当创建唯一的属性名需要时，
IRIS会用整数(从0开始)替换字段名的最后一个字符。

下面的示例显示
IRIS如何处理仅标点符号不同的字段名称。这些字段对应的类属性分别命名为PatNum、PatNu0和PatNu1：

```
CREATE TABLE MyPatients (
     _PatNum VARCHAR(16),
     %Pat@Num INTEGER,
     Pat_Num VARCHAR(30),
     CONSTRAINT Patient_PK PRIMARY KEY (_PatNum))
```

```
///
Class User.MyPatients Extends %Persistent [ ClassType = persistent, DdlAllowed, Final
, Owner = {yx}, ProcedureBlock, SqlRowIdPrivate, SqlTableName = MyPatients ]
{

Property PatNum As %Library.String(MAXLEN = 16) [ SqlColumnNumber = 2, SqlFieldName =
 _PatNum ];

Property PatNu0 As %Library.Integer(MAXVAL = 2147483647, MINVAL = -2147483648) [ SqlC
olumnNumber = 3, SqlFieldName = %Pat@Num ];

Property PatNu1 As %Library.String(MAXLEN = 30) [ SqlColumnNumber = 4, SqlFieldName =
 Pat_Num ];

/// Bitmap Extent Index auto-
generated by DDL CREATE TABLE statement.  Do not edit the SqlName of this index.
Index DDLBEIndex [ Extent, SqlName = "%%DDLBEIndex", Type = bitmap ];

/// DDL Primary Key Specification
Index PatientPK On PatNum [ PrimaryKey, SqlName = Patient_PK, Type = index, Unique ];

}
```

CREATE TABLE中指定的字段名称在class属性中显示为SqlFieldName值。

在动态选择操作期间，
IRIS可以生成属性名称别名，以便于常见的字母大小写变体。例如，在给定字段名HomeStreet的情况下， IRIS可能会为特性名称指定别名HOMESTREAT、HOMESTREAT和HomeStreet。如果别名与另一字段名的名称冲突，或与分配给另一字段名的别名冲突， IRIS不会分配别名。

数据类型

每个字段定义都必须指定一个数据类型，该数据类型映射到字段定义所基于的数据类型类。指定的数据类型将字段允许的数据值限制为适合该数据类型的值。
SQL支持大多数标准SQL数据类型。本参考的数据类型部分提供了支持的数据类型的完整列表。

CREATE TABLE允许使用 SQL数据类型(例如，VARCHAR(24)或CHARACTER VARING(24))或通过直接指定它映射到的数据类型类(例如，%Library.String(MAXLEN=24)或%String(MAXLEN=24))来指定数据类型。(对于所有数据类型类，语法形式%Library.Datatype和%Datatype 是同义词。)

通常， SQL(如CREATE TABLE命令)指定数据类型。可以直接指定数据类型类来定义其他数据定义参数，例如允许的数据值的枚举列表、允许的数据值的模式匹配、最大和最小数值以及超过最大长度(MAXLEN)的数据值的自动截断。

注：数据类型类参数默认值可能不同于 SQL数据类型默认值。例如，VARCHAR()和CHARACTER VARING()缺省为MAXLEN=1；相应的数据类型类%Library.String默认为MAXLEN=50。

IRIS通过提供SQL.SystemDataTypes映射表和SQL.UserDataTypes映射表，将这些标准SQL数据类型映射到IRIS数据类型。用户可以添加SQL.UserDataTypes以包括其他用户定义的数据类型。

要查看和修改当前数据类型映射，请转到管理门户，选择系统管理、配置、SQL和对象设置、系统DDL映射。要创建其他数据类型映射，请转到管理门户，选择系统管理、配置、SQL和对象设置、用户DDL映射。

如果在SQL中指定的数据类型不存在相应的 IRIS数据类型，则SQL数据类型名称将用作相应类属性的数据类型。必须在DDL运行时(SQLExecute)之前创建此用户定义的IRIS数据类型。

还可以覆盖单个参数值的数据类型映射。例如，假设不希望VARCHAR(100)映射到提供的标准映射%string(MAXLEN=100)。可以通过将DDL数据类型 ‘ VARCHAR(100) ’ 添加到表中，然后指定其相应 IRIS类型来覆盖它。例如：

```
VARCHAR(100) maps to MyString100(MAXLEN=100)
```

注：定义分割表的CREATE TABLE不能包含ROWVERSION数据类型或SERIAL(%Library.Counter)数据类型的字段。

数据大小

在数据类型之后，可以在括号中表示允许的数据大小。允许使用数据类型名称和数据大小括号之间的空格，但不是必需的。

对于字符串，数据大小表示最大字符数。例如：

```
ProductName VARCHAR (64)
```

对于允许使用小数的数字，这表示为一对整数(p，s)。第一个整数(P)是数据类型精度，但它与数值精度(数字中的位数)不同。这是因为底层IRIS数据类型类没有精度，而是使用此数字来计算MAXVAL和MINVAL；第二个整数是小数位数，它指定最大小数位数。例如：

```
UnitPrice NUMERIC(6,2)  /* maximum value 9999.99 */
```

要确定字段的最大允许值和最小允许值，请使用以下ObjectScript函数：

```
  WRITE $$maxval^%apiSQL(6,2),!
  WRITE $$minval^%apiSQL(6,2)
```

请注意，因为p不是数字计数，所以它可以小于刻度的值：

```
ClassMethod CreateTable4()
{
    for i = 0 : 1 : 6 {
        w "Max for (",i,",2)=",$$maxval^%apiSQL(i,2),!
    }
}
```

```
DHC-APP>d ##class(PHA.TEST.SQLCommand).CreateTable4()
Max for (0,2)=.99
Max for (1,2)=.99
Max for (2,2)=.99
Max for (3,2)=9.99
Max for (4,2)=99.99
Max for (5,2)=999.99
Max for (6,2)=9999.99
```

#SQL #Caché

源URL: https://cn.community.intersystems.com/post/%E7%AC%AC%E5%8D%81%E4%BA%94%E7%AB%A0-sql%E5%91%BD%E4%BB%A4-create-table%EF%BC%88%E4%BA%8C%EF%BC%89