文章
姚 鑫 · 九月 12, 2021 阅读大约需 9 分钟

第十四章 SQL命令 CREATE TABLE（一）

第十四章 SQL命令 CREATE TABLE（一）

创建表

大纲

```
CREATE [GLOBAL TEMPORARY] TABLE
table (table-element-commalist) [shard-key] [WITH table-option-commalist]

table-element ::=
     [%DESCRIPTION string]
     [%FILE string]
     [{%EXTENTSIZE | %NUMROWS} integer]
     [%PUBLICROWID]
     [%ROUTINE string]

    { fieldname datatype [AUTO_INCREMENT] | IDENTITY | SERIAL | ROWVERSION
            [ %DESCRIPTION string ]
           {
             [ [COLLATE] sqlcollation ]
             [ UNIQUE ]
             [ NULL | NOT NULL ]
             [ PRIMARY KEY ]
             [ REFERENCES table  (reffield-commalist) ]
             [ DEFAULT [(]default-spec[)] ]
             [ ON UPDATE update-spec ]
             [ COMPUTECODE { ObjectScript-code }
                   [ COMPUTEONCHANGE (field-commalist) |
                     CALCULATED | TRANSIENT ] ]
             } , }

     [{ [CONSTRAINT uname]
          UNIQUE (field-commalist) }]

    [ [CONSTRAINT pkname]
          PRIMARY KEY (field-commalist) ]

     [{ [CONSTRAINT fkname]
          FOREIGN KEY (field-commalist) REFERENCES table
              [(reffield-commalist)]
             [ON DELETE ref-action] [ON UPDATE ref-action]  [NOCHECK] }]


[ SHARD [ KEY (field-commalist)  [ COSHARD [ WITH ] [(]table[)] ] ] ]

[WITH table-option ::=
         { %CLASSPARAMETER paramname [=] value } , }
```

```
          [ STORAGETYPE [=] {ROW | COLUMN} ]
     ]

sqlcollation ::=
     { %EXACT | %MINUS | %MVR | %PLUS | %SPACE |
        %SQLSTRING [(maxlen)] | %SQLUPPER [(maxlen)] |
        %TRUNCATE[(maxlen)]  }
```

此摘要不包括仅为兼容性而分析的关键字，但不执行任何操作。下面单独一节列出了这些受支持的no-op关键字。

参数

- GLOBAL TEMPORARY - 可选-此关键字子句将表创建为临时表。
- table - 要创建的表的名称，指定为有效标识符。表名可以是限定的(schema.table)，也可以是非限定的(Table)。未限定的表名采用默认模式名。
- table-element - 一个或多个字段定义或关键字短语的逗号分隔列表。此逗号分隔的列表用圆括号括起来。每个字段定义(至少)由一个字段名(指定为有效标识符)和一个数据类型组成。关键字短语可以只由关键字(%PUBLICROWID)、关键字和文字组成。
- WITH table-option - 可选-
  一个或多个表选项(如一个或多个%CLASSPARAMETER子句或STORAGETYPE子句)的逗号分隔列表。
- COLLATE sqlcollation - 可选-指定以下SQL排序规则类型之一：%Exact、%Minus、%Plus、%SPACE、%SQLSTRING、%SQLUPPER、%TRUNCATE或%MVR。默认值为名称空间默认排序规则(除非更改，否则为%SQLUPPER)。%SQLSTRING、%SQLUPPER和%TRUNCATE可以使用可选的最大长度截断参数(括在圆括号中的整数)指定。这些排序参数关键字的百分号(%)前缀是可选的。COLLATE关键字是可选的。
- uname，pkname，fkname - 可选-
  约束的名称，指定为有效标识符。如果指定为分隔标识符，则约束名称可以包".", "^", ",", "->"
  字符。此可选约束名称在ALTER TABLE中用于标识已定义的约束。
- field-commalist - 字段名或逗号分隔的任意顺序的字段名列表。用于定义唯一、主键或外键约束。为约束指定的所有字段名也必须在字段定义中定义。必须用括号括起来。
- reffield-commalist - 可选-在FOREIGN KEY约束中指定的引用表中定义的字段名或现有字段名列表(以逗号分隔)。如果指定，必须用圆括号括起来。如果省略，则采用默认值，如定义外键中所述。
- ref-action - 可选-外键定义可以指定两个ref-action子句：ON DELETE REF-ACTION 或 ON UPDATE REF-ACTION。支持的引用操作选项有no action、set default、set null或CASCADE。

描述

CREATE TABLE命令创建指定结构的表定义。
IRIS自动创建与此表定义对应的持久化类，其属性与字段定义对应。CREATE TABLE将相应的类定义为DdlAllowed。它不在相应的类定义中指定显式StorageStrategy；它使用默认存储%Storage.Persistent。默认情况下，CREATE TABLE在相应的类定义中指定最终的CLASS关键字，指示它不能有子类。(默认值为1；可以使用$SYSTEM.SQL.Util.SetOption()方法设置status=$SYSTEM.SQL.Util.SetOption("DDLFinal",0,.oldval)在系统范围内更改此默认值；要确定当前设置，请调用$SYSTEM.SQL.CurrentSettings()方法)。

注：CREATE TABLE通过指定字段定义和其他元素创建表。使用CREATE TABLE AS SELECT命令通过从现有表复制字段定义和数据来定义表。

语法概述

CREATE TABLE命令具有以下总体语法：
- 表名，限定名(schema.tablename)或非限定名(Tablename)。
- 一对圆括号，用逗号分隔的表格元素列表括起来。这些表元素包括字段定义、约束、关键字子句以及主键和外键定义。元素可以按任何顺序指定。元素必须用逗号分隔。
- 可选的分片键定义，可以在右括号后指定。
- 可选的WITH子句，可以在右括号之后和分片键定义(如果存在)之后指定。WITH子句可以包含逗号分隔的%CLASSPARAMETER子句列表 和/或 STORAGETYPE子句。

较早的CREATE TABLE代码可能会将SHARD键定义和%CLASSPARAMETER子句作为逗号分隔的元素包含在表元素的圆括号内。首选语法是在结束表元素括号之后指定这些子句。指定这些子句的重复项会生成SQLCODE-327错误。

SQL安全和权限

CREATE TABLE命令是特权操作。用户必须具有%CREATETABLE管理权限才能执行CREATE TABLE。否则将导致 SQLCODE –99 %msg User 'name' does not have %CREATETABLE privileges。如果拥有适当的授予权限，则可以使用GRANT命令将%CREATETABLE权限分配给用户或角色。管理权限是特定于命名空间的。

默认情况下，将强制执行CREATE TABLE安全权限。此权限要求可使用$SYSTEM.SQL.Util.SetOption()方法在系统范围内配置 status=$SYSTEM.SQL.Util.SetOption("SQLSecurity",0,.oldval)。要确定当前设置，请调用$SYSTEM.SQL.CurrentSettings()方法，该方法显示an SQL security enabled setting.

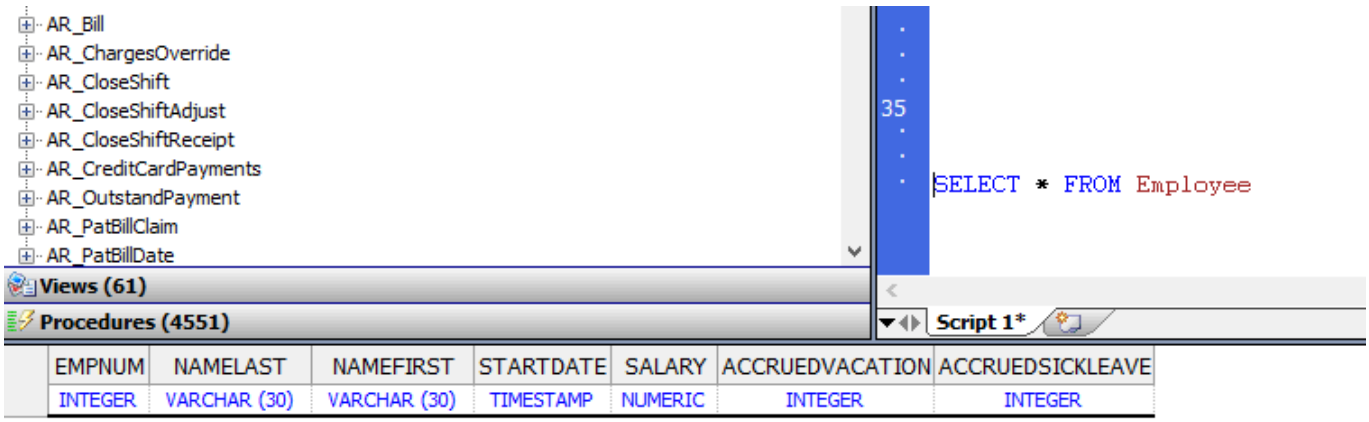
默认值为1(已启用)。启用SQL安全性后，用户只能对其已被授予权限的表或视图执行操作。这是此选项的推荐设置。

如果此方法设置为0，则对更改此设置后启动的任何新进程禁用SQL安全性。这意味着基于特权的表/视图安全性被抑制。可以在不指定用户的情况下创建表。在本例中，动态SQL将 “ SYSTEM ” 指定为用户，嵌入式SQL将""(空字符串)指定为用户。任何用户都可以对表或视图执行操作，即使该用户没有权限执行操作。

嵌入式SQL不使用SQL权限。在嵌入式SQL中，可以使用$SYSTEM.Security.Login()方法以具有适当权限的用户身份登录。必须具有%ServiceLogin：Use权限才能调用$SYSTEM.Security.Login()方法。

下面的嵌入式SQL示例创建Employee表：

```
ClassMethod CreateTable()
{
    d $SYSTEM.Security.Login("_SYSTEM","SYS")
    n SQLCODE,%msg
    &sql(
        CREATE TABLE 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 {
        w !,"???"
    } else {
        w !,"SQLCODE=",SQLCODE,": ",%msg
    }
}
```



这个名为Employee的表有许多已定义的字段。EMPNUM字段(包含员工的公司ID号)是一个不能为空的整数值；此外，它被声明为表的主键。员工的姓和名都有一个字段，这两个字段都是最大长度为30的字符串，不能为空。此外，还有员工的开始日期、累计假期时间和累计病假时间字段(使用TIMESTAMP和INT数据类型)。

使用下面的程序删除上一个示例中创建的表:

```
ClassMethod CreateTable1()
{
    d $SYSTEM.Security.Login("_SYSTEM","SYS")
    n SQLCODE,%msg
    &sql(
        DROP TABLE Employee
    )
    if SQLCODE=0 {
        w !,"????"
    } else {
        w !,"SQLCODE=",SQLCODE,": ",%msg
    }
}
```

表名

表名可以是限定的，也可以是非限定的。

非限定表名具有以下语法：tablename；它省略架构(和句点(.)。字符)。未限定的表名采用默认模式名。系统范围内的初始默认架构名称是SQLUser，它对应于默认的类包名称User。架构搜索路径值将被忽略。

可以配置系统范围的默认架构名称。

要确定当前系统范围内的默认架构名称，请使用$SYSTEM.SQL.Schema.Default()方法。

限定表名具有以下语法：schema.tablename。它可以指定现有的架构名称，也可以指定新的架构名称。指定现有架构名称会将该表放入该架构中。指定新的模式名称将创建该模式(以及关联的类包)，并将表放入该模式中。

表名和模式名遵循SQL标识符命名约定，受使用非字母数字字符、唯一性和最大长度的附加约束。以%字符开头的名称保留供系统使用。默认情况下，模式名和表名是简单标识符，不区分大小写。

IRIS使用表名生成相应的类名。 IRIS使用架构名称来生成相应的类包名称。类名仅包含字母数字字符(字母和数字)，并且在前96个字符内必须是唯一的。要生成类名， IRIS首先从表名中剔除符号(非字母数字)字符，然后生成唯一的类名，从而施加唯一性和最大长度限制。要生成包名，它然后对架构名中的符号(非字母数字)字符进行剥离或执行特殊处理。然后， IRIS生成唯一的包名，施加唯一性和最大长度限制。

可以对架构和表使用相同的名称。同一架构中的表和视图不能使用相同的名称。

架构名称不区分大小写；相应的类包名称区分大小写。如果指定的架构名称仅与现有类包名的大小写不同，并且包定义为空(不包含类定义)。 IRIS通过更改类包名称的大小写来协调这两个名称。

IRIS支持表名和字段名的16位(宽)字符。对于大多数区域设置，可以使用重音字母作为表名，并且重音符号包含在生成的类名中。以下示例对SQL表名执行验证测试：

```
ClassMethod CreateTable2()
{
    s tname = "MyTestTableName"
    s x = $SYSTEM.SQL.IsValidRegularIdentifier(tname)
    if x = 0 {
        if $length(tname) > 200 {
            w "????"
            q
        } elseif $SYSTEM.SQL.IsReservedWord(tname) {
            w "??????"
            q
        } else {
            w "????????",!
            s nls = ##class(%SYS.NLS.Locale).%New()
            if nls.Language [ "Japanese" {
                w "?????????????"
                q
            }
            q
        }
    } else {
        w tname," ???????"
    }
}
```

注意：日语区域设置不支持标识符中的重音字母字符。日语标识符可以包含(除日语字符外)拉丁字母字符A-Z和a-z(65-90和97-122)、下划线字符(95)和希腊大写字母字符(913-929和931-937)。Nls.language测试使用[(CONTAINS运算符)而不是=，因为不同的操作系统平台有不同的日语区域设置。

表存在

要确定当前命名空间中是否已存在表，请使用$SYSTEM.SQL.Schema.TableExists("schema.tname")

默认情况下，当创建与现有表同名的表时， IRIS拒绝CREATE TABLE尝试并发出SQLCODE-201错误。要确定当前系统范围的配置设置，请调用$SYSTEM.SQL.CurrentSettings()，它将显示Allow DDL CREATE TABLE or CREATE VIEW for existing table or view setting。默认值为0；这是此选项的推荐设置。如果此选项设置为1，IRIS将删除与该表关联的类定义，然后重新创建它。这与执行DROP TABLE、删除现有表，然后执行CREATE TABLE大致相同。在这种情况下，强烈建议$SYSTEM.SQL.CurrentSettings()，DDL DROP TABLE是否删除表的数据？值设置为1(默认值)。

在管理门户、系统管理、配置、SQL和对象设置中，通过选中忽略冗余DDL语句复选框，可以在系统范围内设置此选项(以及其他类似的创建、更改和删除选项)。

#SQL #Caché

源URL:

https://cn.community.intersystems.com/post/%E7%AC%AC%E5%8D%81%E5%9B%9B%E7%AB%A0-sql%E5%91%BD%E4%BB%A4-create-table%EF%BC%88%E4%B8%80%EF%BC%89