文章
姚 鑫 · 九月 15, 2021 阅读大约需 10 分钟

第十六章 SQL命令 CREATE TABLE（三）

第十六章 SQL命令 CREATE TABLE（三）

字段数据约束

数据约束控制字段允许使用的值、字段的默认值以及数据值使用的排序规则类型。所有这些数据约束都是可选的。可以按任何顺序指定多个数据约束，并以空格分隔。

NULL和NOT NULL

NOT NULL数据约束关键字指定该字段不接受空值；换句话说，每条记录都必须为该字段指定一个值。NULL和空字符串('') IRIS中是不同的值。可以在接受字符串的字段中输入空字符串，即使该字段定义了NOT NULL限制也是如此。不能在数值字段中输入空字符串。

NULL数据约束关键字显式指定此字段可以接受空值；这是字段的默认定义。

UNIQUE

唯一数据约束指定此字段仅接受唯一值。因此，没有两条记录可以包含该字段的相同值。SQL空字符串('')被视为数据值，因此在应用了UNIQUE数据约束的情况下，任何两条记录都不能包含此字段的空字符串值。NULL不被视为数据值，因此唯一数据约束不适用于多个NULL。要限制字段使用NULL，请使用NOT NULL关键字约束。
- 唯一数据约束要求指定字段的所有值都是唯一值。
- UNIQUE fields constraint(使用CONSTRAINT关键字)要求一组指定字段的所有值在串联在一起时产生唯一值。不需要将各个字段限制为唯一值。

定义为分片表的表对UNIQUE数据约束的使用有额外的限制。
不包含shard键的字段或字段组上的唯一约束为插入和更新增加了显著的性能成本。
因此，当插入和更新性能是一个重要的考虑因素时，建议避免这种类型的唯一约束。

DEFAULT

默认数据约束指定 IRIS在INSERT操作期间自动为此字段提供的默认数据值(如果INSERT未为此字段提供数据值)。如果插入操作为字段数据值提供NULL，则采用NULL而不是默认数据值。因此，为同一字段同时指定DEFAULT和NOT NULL数据约束是很常见的。

默认值可以作为文字值或关键字选项提供。作为文字默认值提供的字符串必须用单引号引起来。数字默认值不需要单引号。例如：

```
CREATE TABLE membertest
(MemberId INT NOT NULL,
Membership_status CHAR(13) DEFAULT 'M',
Membership_term INT DEFAULT 2)
```

创建表时不会验证默认值。定义后，默认值可以忽略数据类型、数据长度和数据约束限制。但是，当使用INSERT向

表提供数据时，缺省值是受约束的；它不受数据类型和数据长度限制，而是受数据约束限制。例如，定义了Ordernum int Unique Default ‘ No Number ’ 的字段可以采用默认值一次，忽略int数据类型限制，但不能第二次采用缺省值，因为这将违反唯一字段数据约束。

如果未指定默认值，则隐含的默认值为NULL。如果字段具有非空数据约束，则必须显式或默认地为该字段指定值。不要将SQL零长度字符串(空字符串)用作非空默认值。

DEFAULT Keywords

默认数据约束可以接受关键字选项来定义其值。支持以下选项：NULL、USER、CURRENTUSER、SESSIONUSER、SYSTEMUSER、CURRENTDATE、CURRENTTIME、CURRENTTIMESTAMP、SYSDATE和OBJECTSCRIPT。

USER、CURRENTUSER和SESSIONUSER默认关键字将字段值设置为ObjectScript $USERNAME特殊变量。

CURRENTDATE、CURRENTTIME、CURRENTTIMESTAMP、GETDATE、GETUTCDATE和SYSDATE SQL函数也可以用作默认值。它们在各自的参考页中进行了描述。当用作默认值时，可以指定CURRENTTIME或TIMESTAMP函数，有没有精度值。如果未指定精度，则将使用SQL配置设置 “ GETDATE()、CURRENTTIME和CURRENTTIMESTAMP的默认时间精度 ” 的精度，默认为0。DEFAULT函数在准备/编译CREATE TABLE语句时(而不是在执行语句时)使用有效的时间精度设置。

可以将CURRENTTIMESTAMP指定为数据类型为%Library.PosiTime或%Library.TimeStamp；的字段的默认值。当前日期和时间以字段数据类型指定的格式存储。可以将CURRENTTIMESTAMP、GETDATE、GETUTCDATE和SYSDATE指定为%Library.TimeStamp字段(数据类型TIMESTAMP或DATETIME)的默认值。
IRIS将日期值转换为适合该数据类型的格式。

```
CREATE TABLE mytest
(
    TestId INT NOT NULL,
    CREATE_DATE DATE DEFAULT CURRENT_TIMESTAMP(2),
    WORK_START DATE DEFAULT SYSDATE
)
```

可以使用TODATE函数作为数据类型DATE的默认数据约束。可以使用TOTIMESTAMP函数作为数据类型TIMESTAMP的默认数据约束。

OBJECTSCRIPT文字关键字短语使您能够通过提供包含ObjectScript代码的带引号的字符串来生成默认值，如下例所示：

```
CREATE TABLE mytest
(
    TestId INT NOT NULL,
    CREATE_DATE DATE DEFAULT OBJECTSCRIPT '+$HOROLOG' NOT NULL,
    LOGNUM NUMBER(12,0) DEFAULT OBJECTSCRIPT '$INCREMENT(^LogNumber)'
)
```

ON UPDATE

ON UPDATE子句使字段的计算值为%%UPDATE。这是定义字段的快捷语法，每当表中的行被更新时，该字段总是被计算。此功能最常见的用途是在表中定义一列，该列包含上次更新该行的时间戳值。

可用的更新规格选项有:

```
CURRENT_DATE | CURRENT_TIME[(precision)] | CURRENT_TIMESTAMP[(precision)] | GETDATE([
prec]) | GETUTCDATE([prec]) | SYSDATE |
USER | CURRENT_USER | SESSION_USER | SYSTEM_USER |
 NULL | <literal> | -<number>
```

以下示例在插入行以及每次更新该行时，将行字段设置为当前时间戳值:

```
CREATE TABLE mytest
(
    Name VARCHAR(48),
    RowTS TIMESTAMP DEFAULT Current_Timestamp(6) ON UPDATE Current_Timestamp(6)
)
```

在本例中，如果没有为RowTS字段指定显式值，则DEFAULT关键字将RowTS设置为插入时的当前时间戳。如果UPDATE为RowTS字段指定了显式值，则ON UPDATE关键字将验证但忽略指定值，并使用当前时间戳更新RowTS。如果指定的值未通过验证，则会生成SQLCODE-105错误。

下面的示例将HasBeenUpdateed字段设置为布尔值：

```
CREATE TABLE mytest
(Name VARCHAR(48),
 HasBeenUpdated TINYINT DEFAULT 0 ON UPDATE 1 )
```

下面的示例将whLastUpdateed字段设置为当前用户名：

```
CREATE TABLE mytest
(Name VARCHAR(48),
 WhoLastUpdated VARCHAR(48) DEFAULT CURRENT_USER ON UPDATE CURRENT_USER )
```

如果该字段还具有COMPUTECODE数据约束，则不能指定ON
UPDATE子句。尝试这样做会在编译/准备时导致SQLCODE-1错误。

Collation Parameters

可选的排序规则参数指定对字段的值进行排序时要使用的字符串排序规则类型。
SQL支持十种类型的排序规则。如果未指定排序规则，则默认为%SQLUPPER排序规则，不区分大小写。

为便于编程，建议在COLLATION参数之前指定可选关键字COLLATE，但此关键字不是必需的。各种排序参数关键字的百分号(%)前缀是可选的。

%Exact排序规则遵循ANSI(或Unicode)字符排序规则序列。这提供区分大小写的字符串排序，并识别前导和尾随空格以及制表符。

%SQLUPPER归类将所有字母转换为大写以进行归类。

%SPACE和%SQLUPPER排序规则会在数据后追加一个空格。这将强制对空值和数字值进行字符串排序。

%SQLSTRING、%SQLUPPER和%TRUNCATE排序规则提供了一个可选的maxlen参数，该参数必须用圆括号括起来。Maxlen是一个截断整数，它指定执行排序时要考虑的最大字符数。当创建包含大数据值的字段的索引时，此参数非常有用。

%PLUS和%MINUS排序规则将NULL处理为0(0)值。

注意:shard键字段只能接受%EXACT、%SQLSTRING或%SQLUPPER排序，没有截断。

ObjectScript为数据排序规则转换提供了%SYSTEM.Util类的Colation()方法。

注意：要将命名空间默认排序规则从%SQLUPPER(不区分大小写)更改为另一种排序规则类型，如%SQLSTRING(区分大小写)，请使用以下命令：

```
WRITE $$SetEnvironment^%apiOBJ("collation","%Library.String","SQLSTRING")
```

发出此命令后，必须清除索引，重新编译所有类，然后重建索引。当其他用户正在访问表的数据时，不要重建索引。这样做可能会导致不准确的查询结果。

%DESCRIPTION

可以为字段提供描述文本。此选项遵循与为表格提供描述文本相同的约定。上面使用其他表元素对其进行了描述。

计算字段

可以定义一个或多个计算其值的字段，而不是用户提供的字段。计算字段值的事件取决于以下关键字选项：
- COMPUTECODE：值在插入时计算并存储，值在更新时不变。
- COMPUTECODE WITH
COMPUTEONCHANGE：VALUE在INSERT时计算并存储，在UPDATE时重新计算并存储。
- COMPUTECODE WITH DEFAULT和COMPUTEONCHANGE：默认值在插入时存储，值在更新时计算和存储。
- COMPUTECODE WITH COMPUTTECODE WITH
COMPUTED或TRANSPENT：值不存储，而是在每次查询字段时生成。

COMPUTECODE

COMPUTECODE数据约束指定ObjectScript代码来计算此字段的默认数据值。ObjectScript代码在大括号内指定。在ObjectScript代码中，可以使用大括号分隔符指定SQL字段名称。ObjectScript代码可以由多行代码组成。它可以包含嵌入式SQL。允许在ObjectScript代码大括号分隔符之前或之后使用空格和回车。

COMPUTECODE指定SqlComputeCode字段名称及其值的计算。在COMPUTECODE或SqlComputeCode类属性中指定计算字段名称时，必须指定SQL字段名称，而不是相应的生成的表属性名称。

计算机代码提供的默认数据值必须处于逻辑(内部存储)模式。计算机代码中的嵌入式SQL被自动编译并以逻辑模式运行。

以下示例定义了Birthday COMPUTECODE字段。它使用ObjectScript代码从道布字段值计算其默认值：

```
CREATE TABLE MyStudents
(
   Name VARCHAR(16) NOT NULL,
   DOB DATE,
   Birthday VARCHAR(12) COMPUTECODE {SET {Birthday}=$PIECE($ZDATE({DOB},9),",")},
   Grade INT
)
```

COMPUTECODE可以包含伪字段引用变量{%%CLASSNAME}、{%%CLASSNAMEQ}、{%%OPERATION}、{%%T

ABLENAME}和{%%ID}。这些伪字段在类编译时被转换为特定值。所有这些伪字段关键字都不区分大小写。

COMPUTECODE值是默认值；只有在未向该字段提供值的情况下才会返回该值。COMPUTECODE值不受数据类型限制。COMPUTECODE值受到唯一数据约束和其他数据约束限制的限制。如果同时指定DEFAULT和COMPUTECODE，则始终采用默认值。

COMPUTECODE可以选择接受COMPUTEONCHANGE、COMPUTEONCHANGE或TEMPUTEONCHANGE关键字。支持以下关键字组合行为：

如果ObjectScript COMPUTECODE代码中存在错误，则在第一次执行代码之前，SQL不会检测到此错误。因此，如果在INSERT时首先计算值，则INSERT操作失败，出现SQLCODE-415错误；如果在更新时首先计算值，则UPDATE操作失败，出现SQLCODE-415错误；如果在查询时首先计算值，则SELECT操作失败并出现SQLCODE-350错误。

可以索引COMPUTECODE存储值。应用程序开发人员负责确保根据计算字段存储值的数据类型验证和标准化计算字段存储值(规范化形式的数字)，特别是在为计算字段定义(或打算定义)索引的情况下。

COMPUTEONCHANGE

COMPUTECODE本身会导致在INSERT过程中计算字段值并将其存储在数据库中；该值在后续操作中保持不变。默认情况下，后续的更新或触发器代码操作不会更改计算值。指定COMPUTEONCHANGE关键字会导致后续的UPDATE或触发器代码操作重新计算并替换此存储值。

如果使用COMPUTEONCHANGE子句指定一个字段或以逗号分隔的字段列表，则对其中一个字段的值所做的任何更改都会导致 SQL重新计算COMPUTECODE字段值。

如果COMPUTEONCHANGE中指定的字段不是表规范的一部分，则会生成SQLCODE-31。

在下面的示例中，生日是根据DOB(出生日期)值进行插入计算的。
更新DOB时重新计算生日:

```
CREATE TABLE SQLUser.MyStudents (
   Name VARCHAR(16) NOT NULL,
   DOB DATE,
   Birthday VARCHAR(40) COMPUTECODE {
        SET {Birthday}=$PIECE($ZDATE({DOB},9),",")
        _" changed: "_$ZTIMESTAMP }
        COMPUTEONCHANGE (DOB)
     )
```

COMPUTEONCHANGE用与字段定义相对应的类属性的%%UPDATE值定义SqlComputeOnChange关键字。
该属性值最初是作为INSERT操作的一部分计算的，并在UPDATE操作期间重新计算。

CALCULATED和TRANSIENT

指定COMPUTECODE或TEMPUTE关键字指定COMPUTECODE字段值不保存在数据库中；它作为访问它的每个查询操作的一部分进行计算。这会减小数据存储的大小，但可能会降低查询性能。因为这些关键字导致 IRIS不存储COMPUTECODE字段值，所以这些关键字和COMPUTEONCHANGE关键字是互斥的。以下是计算字段的示例：

```
CREATE TABLE MyStudents (
   Name VARCHAR(16) NOT NULL,
   DOB DATE,
   Days2Birthday INT COMPUTECODE{SET {Days2Birthday}=$ZD({DOB},14)-$ZD($H,14)} CALCUL
ATED
```

)

Computed为与字段定义对应的类属性定义计算的布尔关键字。瞬态定义与字段定义对应的类属性的瞬态布尔关键字。

计算和瞬态提供了几乎相同的行为，但有以下差异。
TRANSIENT意味着IRIS不存储该属性。
计算意味着IRIS不为属性分配任何实例内存。
因此，当指定calculate时，将隐式设置TRANSIENT。

瞬态属性不能被索引。
除非属性也是SQLComputed，否则无法为计算属性建立索引。

#SQL #Caché

源URL: https://cn.community.intersystems.com/post/%E7%AC%AC%E5%8D%81%E5%85%AD%E7%AB%A0-sql%E5%91%BD%E4%BB%A4-create-table%EF%BC%88%E4%B8%89%EF%BC%89