文章
姚 鑫 · 三月 10, 2021 阅读大约需 分钟

第章 SQL表之间的关系

第章 SQL表之间的关系

要在表之间强制执行引用完整性，以定义外键。将修改包含外键约束的表时，将检查外键约束。

定义外键

有几种方法可以在InterSystems SQL中定义外键：

- 可以定义两个类之间的关系。定义关系会自动将外键约束投影到SQL。
- 可以在类定义中添加显式外键定义（对于关系未涵盖的情况）。
- 可以使用CREATE TABLE或ALTER TABLE命令添加外键。可以使用ALTER TABLE命令删除外键。

用作外键引用的RowID字段必须是公共的。引用隐藏的RowID？有关如何使用公用（或专用）RowID字段定义表的信息。

一个表（类）的外键最大数目为400。

外键引用完整检查

外键约束可以指定更新或删除时的引用操作。
在CREATE TABLE reference action子句中描述了使用DDL定义这个引用操作。
在类定义引用的OnDelete和OnUpdate外键关键字中定义了一个持久化类来定义这个引用操作，该类投射到一个表。
在创建分片表时，这些引用操作必须设置为无操作。

默认情况下，InterSystemsIRIS®数据平台对INSERT，UPDATE和DELETE操作执行外键引用完整性检查。如果该操作将违反参照完整性则不会执行；该操作将发出SQLCODE
-121，-122，-123或-124错误。参照完整性检查失败会生成如下错误：

```
???5540?SQLCODE?-124????'HealthLanguage.FKey2'?????1???????NewIndex1-????'NewForeignKey1'???'Pointer1'??NO ACTION??????[Execute + 5 ^ IRISSql16?USER]
```

可以使用$SYSTEM.SQL.SetFilerRefIntegrity()方法在系统范围内禁止此检查。若要确定当前设置，请调用$SYSTEM.SQL.CurrentSettings()。

默认情况下，当删除带有外键的行时，InterSystems IRIS将在相应的被引用表的行上获取长期（直到事务结束）共享锁。这样可以防止在引用行上的DELETE事务完成前对引用行进行更新或删除。这样可以防止删除引用行，然后回退删除引用行的情况。如果发生这种情况，外键将引用不存在的行。如果使用NoCheck定义外键，或者使用%NOCHECK或%NOLOCK指定引用行的DELETE，则不会获取此锁定。

使用持久性类定义定义表时，可以使用NoCheck关键字定义外键，以禁止将来对该外键进行检查。CREATE TABLE不提供此关键字选项。

可以使用%NOCHECK关键字选项禁止检查特定操。

默认情况下, InterSystems
IRIS还对以下操执行外键引用完整性检查。如果指定的操违反了引用完整性则不执行该命令:
- ALTER TABLE DROP COLUMN。
- ALTER TABLE DROP CONSTRAINT删除约束
问题-317 SQLCODE。
可以使用SET选项COMPILEMODE=NOCHECK来抑制外键完整性检查。
- 删除表。问题-320 SQLCODE。可以使用SET选项COMPILEMODE = NOCHECK来抑制外键插入检查。
- 触发器事件, 包括事件之前。
例如, 如果删除操因违反外键引用完整性而不能执行, 则不会执行BEFORE DELETE触发器。

在父/子关系中, 没有定义子元素的顺序。
应用程序代码不能依赖于任何特定的顺序。

父表和子表

定义父表和子表

在定义映射到表的持久类时, 可以使用relationship属性指定两个表之间的父/子关系。

下面的例子定义了父表:

```
Class Sample.Invoice Extends %Persistent
{
Property Buyer As %String(MAXLEN=50) [Required];
Property InvoiceDate As %TimeStamp;
Relationship Pchildren AS Sample.LineItem [ Cardinality = children, Inverse = Cparent
 ];
}
```

下面的例子定义了一个子表:

```
Class Sample.LineItem Extends %Persistent
{
Property ProductSKU As %String;
Property UnitPrice As %Numeric;
Relationship Cparent AS Sample.Invoice [ Cardinality = parent, Inverse = Pchildren ];
}
```

注 意这两句话:
- Relationship Pchildren AS Sample.LineItem [Cardinality = children, Inverse = Cparent];
- Relationship Cparent AS Sample.Invoice [Cardinality = parent, Inverse = Pchildren];

在Management Portal SQL interface Catalog Details选项卡中, 表信息提供了子表和/或父表的名称。
如果是子表, 则提供对父表的引用, 如:parent->Sample.Invoice。

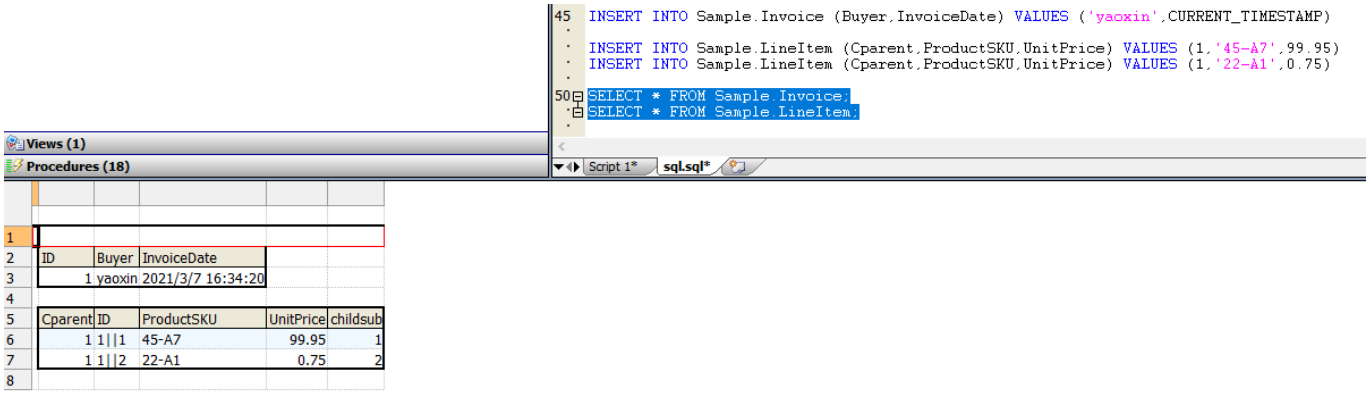
子表本身可以是子表的父表。
(子表的子表被称为"孙"表。)
在本例中, 表Info提供了父表和子表的名称。

向父表和子表插入数据

在将相应的记录插入子表之前，必须将每个记录插入父表。
例如:

```
INSERT INTO Sample.Invoice (Buyer,InvoiceDate) VALUES ('yaoxin',CURRENT_TIMESTAMP)


INSERT INTO Sample.LineItem (Cparent,ProductSKU,UnitPrice) VALUES (1,'45-A7',99.95)
INSERT INTO Sample.LineItem (Cparent,ProductSKU,UnitPrice) VALUES (1,'22-A1',0.75)
```



尝试插入没有对应父记录ID的子记录时，会使用%msg子表'Sample生成SQLCODE -104错误。
LineItem'引用父表中不存在的行。

在子表上的插入操作期间，在父表的相应行上获得共享锁。
在插入子表行时，该行被锁定。
然后，锁被释放(直到事务结束时才被持有)。
这确保在插入操作期间引用的父行不会被更改。

标识父表和子表

在嵌入式SQL中，可以使用主机变量数组来标识父表和子表。
在子表中，主机变量数组的标0被设置为父引用(Cparent)，格式为parentref，标1被设置为子记录ID，格式为par
entref|| childf。
在父表中，没有定义标0。
如下面的例子所示:

```
/// d ##class(PHA.TEST.SQL).FatherChildTable()
ClassMethod FatherChildTable()
{
    KILL tflds,SQLCODE,C1
    &sql(DECLARE C1 CURSOR FOR
        SELECT *,%TABLENAME INTO :tflds(),:tname
        FROM Sample.Invoice)
    &sql(OPEN C1)
    IF SQLCODE<0 {
        WRITE "???SQL??:",SQLCODE," ",%msg  QUIT
    }
    &sql(FETCH C1)
    IF SQLCODE=100 {
        WRITE "The ",tname," ???????",!  QUIT
    }
    WHILE $DATA(tflds(0)) {
```

```
        WRITE tname," ?????",!,"parent ref: ",tflds(0)," %ID: ",tflds(1),!
        &sql(FETCH C1)
        IF SQLCODE=100 {QUIT}
    }
    IF $DATA(tflds(0))=0 {
        WRITE tname," ???",!
    }
    &sql(CLOSE C1)
    IF SQLCODE<0 {
        WRITE "??????:",SQLCODE," ",%msg  QUIT
    }
}


DHC-APP> d ##class(PHA.TEST.SQL).FatherChildTable()
Sample.Invoice ???


/// d ##class(PHA.TEST.SQL).FatherChildTable1()
ClassMethod FatherChildTable1()
{
    KILL tflds,SQLCODE,C2
    &sql(DECLARE C2 CURSOR FOR
        SELECT *,%TABLENAME INTO :tflds(),:tname
        FROM Sample.LineItem)
    &sql(OPEN C2)
    IF SQLCODE<0 {
        WRITE "???SQL??::",SQLCODE," ",%msg  QUIT
    }
    &sql(FETCH C2)
    IF SQLCODE=100 {
        WRITE "The ",tname," ???????",! QUIT
    }
    WHILE $DATA(tflds(0)) {
        WRITE tname," ?????",!,"parent ref: ",tflds(0)," %ID: ",tflds(1),!
        &sql(FETCH C2)
        IF SQLCODE=100 {QUIT}
    }
    IF $DATA(tflds(0))=0 {
        WRITE tname," ???",!
    }
    &sql(CLOSE C2)
    IF SQLCODE<0 {
        WRITE "??????::",SQLCODE," ",%msg  QUIT
    }
}
```

对于子表, tflds(0)和tflds(1)迥如¨值:

```
DHC-APP>d ##class(PHA.TEST.SQL).FatherChildTable1()
Sample.LineItem ?????
parent ref: 1 %ID: 1||1
Sample.LineItem ?????
parent ref: 1 %ID: 1||2
```

对于"孙"表(即子表的子表), tflds(0)和tflds(1)返回如下值:

```
parent ref: 1||1 %ID: 1||1||1
parent ref: 1||1 %ID: 1||1||7
parent ref: 1||1 %ID: 1||1||8
parent ref: 1||2 %ID: 1||2||2
parent ref: 1||2 %ID: 1||2||3
parent ref: 1||2 %ID: 1||2||4
parent ref: 1||2 %ID: 1||2||5
parent ref: 1||2 %ID: 1||2||6
```

#SQL #Caché #InterSystems IRIS #InterSystems IRIS for Health

源 URL: https://cn.community.intersystems.com/post/%E7%AC%AC%E4%B8%83%E7%AB%A0-sql%E8%A1%A8%E4%B9%8B%E9%97%B4%E7%9A%84%E5%85%B3%E7%B3%BB