文章
姚 鑫 · 四月 5, 2021 阅读大约需 6 分钟

第十七章 使用触发器

第十七章 使用触发器

本章介绍如何在Intersystems SQL中定义触发器。触发器是响应某些SQL事件执行的代码行。本章包括以下主题：

定义触发器

有几种方法可以为特定表定义触发器：

- 在将投影到SQL表的持久性类定义中包含触发定义。例如，MyApp.person类的此定义包括Loggevent触发器的定义，在每个成功的数据插入到MyApp.person表之后，将在每个成功的数据插入后调用：

```
Class MyApp.Person Extends %Persistent [DdlAllowed]
{
    // ... Class Property Definitions

    Trigger LogEvent [ Event = INSERT, Time = AFTER ]
    {
        // Trigger code to log an event
    }
 }
```

- 使用SQL创建触发命令创建触发器。这在相应的持久性类中生成触发对象定义。
  SQL触发器名称按照标识符命名约定进行操作。
  IntersystemsIris®数据平台使用SQL触发名称生成相应的触发类实体名称。

必须拥有%createtrigger管理级别权限来创建触发器。必须具有删除触发器的%droptrigger管理级别权限。

类的最大用户定义触发器数为200。

注意：Intersystems Iris不支持收集投影的表上的触发。用户无法定义这样的触发器，并且作为子表的集合的投影不认为涉及该基本集合的触发。
Intersystems Iris不支持修改Security.Roles和Security.Users表的触发器。

触发器的类型

触发器由以下内容定义：

- 导致它执行的事件类型。触发器可以是单个事件触发器或多事件触发。定义单个事件触发器以在指定表上发生插入，更新或删除事件时执行。定义多事件触发器以执行当在指定的表中发生多个指定的事件中的任何一个时执行。可以使用类定义或创建触发命令定义插入/更新，更新/删除或插入/更新/删除多事件触发器。事件类型在Class定义中指定了所需的事件触发器关键字。
- 触发器执行的时间:在事件发生之前或之后。
  这是由可选的Time trigger关键字在类定义中指定的。
  默认为Before。

- 可以将多个触发器与同一事件和时间相关联;在这种情况下，可以使用order trigger关键字来控制触发多个触发器的顺序。先触发顺序较低的触发器。
  如果多个触发器具有相同的Order值，则不指定它们的触发顺序。
- 可选的Foreach trigger关键字提供了额外的粒度。
  该关键字控制触发器是每一行触发一次(Foreach = row)，还是每一行或对象访问触发一次(Foreach = row/object)，还是每语句触发一次(Foreach = statement)。
  没有Foreach trigger关键字定义的触发器每一行触发一次。
  如果触发器是用Foreach = row/object定义的，那么触发器也会在对象访问期间的特定点被调用，如本章后面所述。
  可以使用INFORMATION.SCHEMA.TRIGGERS的ACTIONORIENTATION属性列出每个触发器的For each值

下面是可用的触发器及其等价的回调方法:

- BEFORE INSERT (等价于 %OnBeforeSave())
- AFTER INSERT (等价于 %OnAfterSave())
- BEFORE UPDATE (等价于 %OnBeforeSave())
- AFTER UPDATE (等价于 %OnAfterSave())
- BEFORE UPDATE OF specified column(s)
- AFTER UPDATE OF specified column(s)
- BEFORE DELETE (等价于 %OnDelete())
- AFTER DELETE (等价于 %OnAfterDelete())

注意:当触发器执行时，它不能直接修改正在处理的表中的属性值。
这是因为InterSystems IRIS在字段(属性)值验证代码之后执行触发代码。
例如，触发器不能将LastModified字段设置为正在处理的行中的当前时间戳。
但是，触发器代码可以对表中的字段值发出更新。
更新执行自己的字段值验证。

AFTER Triggers

在INSERT、UPDATE或DELETE事件发生后执行AFTER触发器:
- 如果SQLCODE=0(事件成功完成)，InterSystems IRIS将执行AFTER触发器。
- 如果SQLCODE是负数(事件失败)，系统间IRIS就不会执行AFTER触发器。
- 如果SQLCODE=100(没有发现要插入、更新或删除的行)，则系统间IRIS执行AFTER触发器。

递归触发器

触发器执行可以是递归的。
例如，如果表T1有一个对表T2执行插入操作的触发器，表T2也有一个对表T1执行插入操作的触发器。

当表T1有一个调用例程/过程的触发器，并且该例程/过程执行对T1的插入操作时，也可以发生递归。
触发器递归的处理取决于触发器的类型:

- 行和行/对象触发器:InterSystems IRIS不阻止行触发器和行/对象触发器递归地执行。
  处理触发器递归是程序员的责任。
  如果触发代码不处理递归执行，则可能发生runtime <FRAMESTACK>错误。
- 语句触发器:InterSystems IRIS阻止AFTER语句触发器递归执行。
  如果InterSystems IRIS检测到该触发器在执行堆栈中已经被调用，它将不会发出AFTER触发器。
  没有错误发出;
  触发器不会被第二次执行。

InterSystems IRIS不会阻止BEFORE语句触发器递归地执行。
在触发递归之前处理是程序员的责任。
如果BEFORE触发器代码不处理递归执行，可能会发生runtime <FRAMESTACK>错误。

Trigger Code

每个触发器包含执行触发操作的一行或多行代码。
每当与触发器关联的事件发生时，SQL引擎就会调用这段代码。
如果触发器是使用CREATE触发器定义的，则可以用ObjectScript或SQL编写此操作代码。
(InterSystems IRIS将SQL编写的代码转换为类定义中的ObjectScript。)
如果触发器是使用Studio定义的，那么这个操作代码必须用ObjectScript编写。

因为触发器的代码不是作为过程生成的，所以触发器中的所有局部变量都是公共变量。
这意味着触发器中的所有变量都应该用一个新语句显式声明;
这可以防止它们与调用触发器的代码中的变量发生冲突。

%ok, %msg, and %oper 系统变量

- %ok:仅在触发器代码中使用的变量。
  如果触发代码成功，它设置%ok=1。
  如果触发代码失败，它设置%ok=0。
  如果在触发器执行期间发出SQLCODE错误，InterSystems IRIS将设置%ok=0。
  当%ok=0时，触发器代码中止，触发器操作和调用触发器的操作被回滚。
  如果插入或更新触发器代码失败，并且表中定义了一个外键约束，InterSystems IRIS将释放外键表中相应行上的锁。

触发代码可以显式设置%ok=0。
这会创建一个运行时错误，中止触发器的执行并回滚操作。
通常，在设置%ok=0之前，触发器代码显式地将%msg变量设置为用户指定的字符串，用于描述这个用户定义的触发器代码错误。

%ok变量是一个必须显式更新的公共变量。
在完成非触发代码SELECT、INSERT、UPDATE或DELETE语句后，%ok的值与之前的值没有变化。
%ok仅在执行触发器代码时定义。

%msg:触发代码可以显式地将%msg变量设置为描述运行时错误原因的字符串。
设置变量%msg。

%oper:仅在触发器代码中使用的变量。
触发器代码可以引用变量%oper，该变量包含触发触发器的事件(插入、更新或删除)的名称。

{fieldname}语法

在触发器代码中，可以使用特殊的{fieldname}语法引用字段值(对于属于触发器关联的表的字段)。
例如，下面是MyApp中LogEvent触发器的定义。
Person类包含一个对ID字段的引用，如{ID}:

```
Class MyApp.Person Extends %Persistent [DdlAllowed]
{
    // ... Definitions of other class members

    /// This trigger updates the LogTable after every insert
    Trigger LogEvent [ Event = INSERT, Time = AFTER ]
    {
        // get row id of inserted row
        NEW id,SQLCODE,%msg,%ok,%oper
        SET id = {ID}

        // INSERT value into Log table
        &sql(INSERT INTO LogTable
            (TableName, IDValue)
            VALUES ('MyApp.Person', :id))
        IF SQLCODE<0 {SET baderr="SQLCODE ERROR:"_SQLCODE_" "_%msg
                      SET %ok=0
                    RETURN baderr }

      }
   // ... Definitions of other class members

}
```

这个{fieldname}语法支持统一字段。
它不支持%SerialObject集合属性。
例如，如果表引用了嵌入的串行对象类Address(其中包含属性City)，那么触发器语法{AddressCity}就是对字段的有效引用。
触发器语法{Address}是对集合属性的引用，不能使用。

触发器代码中的宏

触发器代码可以包含一个引用字段名的宏定义(使用{fieldname}语法)。
但是，如果你的触发代码包含一个#Include预处理器指令，用于一个引用字段名的宏(使用{fieldname}语法)，那么这个字段名就不能被访问。
这是因为InterSystems IRIS在代码被传递给宏预处理器之前，翻译触发器代码中的{fieldname}引用。
如果一个{fieldname}引用在#Include文件中，它不会在触发器代码中“ 看到 ”，因此不会被转换。

这种情况的解决方法是定义一个带参数的宏，然后将{fieldname}传递给触发器中的宏。
例如，#Include文件可以包含如下一行:

```
#Define dtThrowTrigger(%val) SET x=$GET(%val,"?")
```

然后在触发器中调用提供{fieldname}语法作为参数的宏:

```
  $$$dtThrowTrigger({%%ID})
```

{name*O}， {name*N}和{name*C}触发代码语法

在更新触发器代码中有三种语法快捷方式可用。

可以使用下面的语法引用旧的(预更新的)值:

```
{fieldname*O}
```

其中fieldname是字段的名称，星号后面的字符是字母 “ O ” (表示旧)。
对于插入触发器，{fieldname*O}总是空字符串("")。

你可以使用下面的语法来引用新的(更新后的)值:

```
{fieldname*N}
```

其中fieldname是字段的名称，星号后面的字符是字母 “ N ” (表示新字段)。
{fieldname*N}语法只能用于引用要存储的值;
它不能用来更改值。
不能在触发器代码中设置{fieldname*N}。
在插入或更新时计算字段的值应该通过其他方法实现，比如SqlComputeOnChange。

可以使用以下语法测试字段值是否被更改(更新):

```
{fieldname*C}
```

其中，fieldname是字段的名称，星号后面的字符是字母 “ C ” (表示已更改)。
{fieldname*C}的计算结果是1，如果字段已经被修改，0，如果它没有被修改。
对于插入触发器，InterSystems IRIS将{fieldname*C}设置为1。

对于具有流属性的类，如果SQL语句(INSERT或UPDATE)没有插入/更新流属性本身，则对流属性{stream *N}和{stream *O}的SQL触发器引用将返回流的OID。
然而，如果SQL语句确实插入/更新了stream属性，{stream *O}仍然是OID，但{stream *N}的值被设置为以下之一:

- 在触发器之前，将流字段的值以传递给更新或插入的任何格式返回。
  这可以是输入到stream属性中的文字数据值，也可以是临时stream对象的OREF或OID。
- AFTER trigger将流的Id作为{stream *N}的值返回。
  这是InterSystems IRIS的Id值，存储在流字段名为global的^classnameD中。
  该值根据流属性的CLASSNAME类型参数使用适当的Id格式。

如果一个流属性使用InterSystems IRIS对象更新，{stream *N}的值总是一个OID。

注意:对于由串行对象的数组集合创建的子表触发器，触发器逻辑与对象访问/保存一起工作，但与SQL访问(插入或更新)不工作。

附加触发器代码语法

在ObjectScript中编写的触发器代码可以包含伪域引用变量{%%CLASSNAME}、{%%CLASSNAMEQ}、{%%OPERATION}、{%%TABLENAME}和{%%ID}。
这些伪字段在类编译时被转换成特定的值。

可以从触发器代码、SQL计算代码和SQL映射定义中使用类方法，因为类方法不依赖于拥有开放对象。

必须使用##class(classname). methodname()语法从触发器代码中调用方法。
你不能使用..Methodname()语法，因为这个语法需要一个当前打开的对象。

可以将当前行字段的值作为类方法的参数传递，但是类方法本身不能使用字段语法。

Pulling Triggers

如果调用对应于该表的DML命令，则 “ 拉出 ” (执行)已定义的触发器。

对于DML命令成功插入、更新或删除的每一行，都会拉取一行或行/对象触发器。

对于每个成功执行的INSERT、UPDATE或DELETE语句，都会拉出一次语句触发器，而不管该语句是否实际更改了表数据中的任何行。

- INSERT语句拉动相应的插入触发器。
  插入可以通过指定%NOTRIGGER关键字来阻止触发相应的触发器。
  指定%NOJOURN关键字的插入不会记录该插入或相应的插入触发器。
  这意味着插入事件或触发事件都不可能回滚。

快速插入不能用于具有插入触发器的表。

- UPDATE语句拉动相应的更新触发器。
  更新可以通过指定%NOTRIGGER关键字来阻止触发相应的触发器。
  指定%NOJOURN关键字的更新不会记录该更新或相应的更新触发器。
  这意味着更新事件或触发事件都不可能回滚。

- 根据执行的DDL操作的类型，INSERT或UPDATE语句拉动相应的INSERT触发器或UPDATE触发器。
  要防止触发任何类型的触发器，请指定%NOTRIGGER关键字。

- DELETE语句拉动相应的DELETE触发器。
  DELETE可以通过指定%NOTRIGGER关键字来阻止触发相应的触发器。
  指定%NOJOURN关键字的删除不会记录删除或相应的删除触发器。
  这意味着删除事件或触发事件都不可能回滚。
- **TRUNCATE TABLE语句不会触发删除触发器。**

默认情况下，DDL语句和相应的触发操作被记录在日志中。
%NOJOURN关键字阻止DDL命令和触发动作的日志记录。

触发器和对象访问

如果触发器是用Foreach = row/object定义的，那么触发器也会在对象访问期间的特定点被调用，这取决于触发器定义的Event和Time关键字，如下所示:

Event	Time	此时也调用Trigger
INSERT	BEFORE	在新对象的%Save()之前
INSERT	AFTER	在新对象的%Save()后
UPDATE	BEFORE	在已存在对象的%Save()之前
UPDATE	AFTER	在已存在对象的%Save()后
DELETE	BEFORE	在现有对象的%DeleteId()之前
DELETE	AFTER	在现有对象的%DeleteId()后

因此，也没有必要为了保持SQL和对象行为同步而实现回调方法，

在对象访问期间没有拔出触发器

默认情况下，SQL对象使用%Storage.Persistent存储。
InterSystems IRIS也支持 %Storage.SQL storage。
SQL存储。

在使用 %Storage.SQL storage的类中保存或删除对象时。
SQL存储，所有语句(Foreach = statement)、行(Foreach = row)和行/对象(Foreach = row/object)触发器被拉出。
没有定义Foreach trigger关键字的触发器是行触发器。
提取所有触发器是默认行为。

但是，在使用%Storage.SQL storage保存或删除类中的对象时。
SQL，可以指定只有定义为Foreach = row/object的触发器应该被拉出。
定义为Foreach = statement或Foreach = row的触发器不会被拉取。
这是通过指定类参数OBJECTSPULLTRIGGERS = 0来实现的。
默认值是OBJECTSPULLTRIGGERS = 1。

此参数仅应用于使用%Storage.SQL定义的类。

触发器与事务

触发器在事务中执行触发器码。它设置事务级别，然后执行触发器代码。成功完成触发器代码后，触发器提交事务。

注意：使用事务的触发器的结果是，如果触发器调用提交事务的代码，则触发器的完成失败，因为事务级别已经递减为0.调用生产的业务服务时可能发生这种情况。

使用INSERT语句级别对象触发器后，如果触发器集%OK = 0，则使用SQLCODE
-131错误失败行的插入失败。如下所示，可能会发生交易回滚：
- 如果autocommit = on，则插入的事务将被回滚。
- 如果autocommit =off，则应用于回滚或提交输入的事务。
- 如果使用noautocommit模式，则不启动事务，因此插入件不能回滚。

AutoCommit模式是使用 SET TRANSACTION %COMMITMODE option或 SetOption()方法建立的，如下所示 SET status=$SYSTEM.SQL.Util.SetOption("AutoCommit",intval,.oldval).
可用方法INTVAL值为0（无），1（隐式）和2（显式）。

触发器可以在触发器中的%MSG变量中设置错误消息。此消息将返回给呼叫者，给出触发器失败的信息。

列出触发器

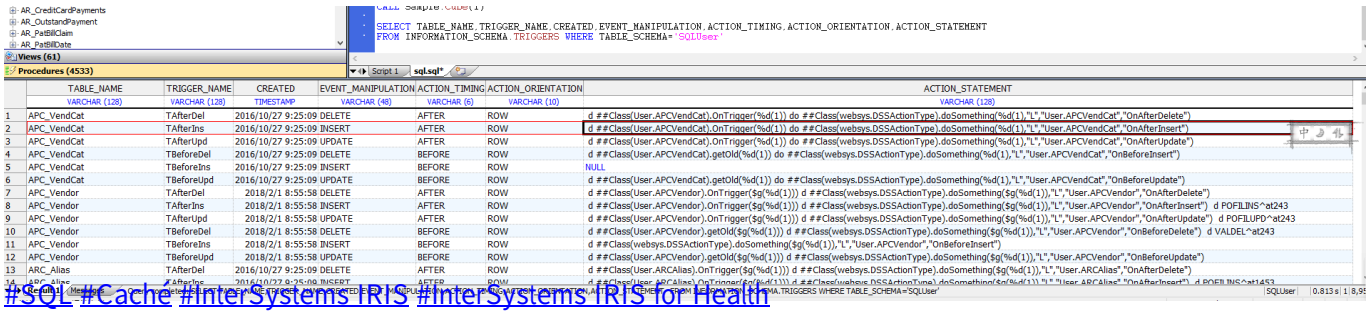
在管理门户SQL接口目录详细信息中列出了为指定表定义的触发器。这列出了每个触发器的基本信息。

Information.schema.triggers类列出了当前命名空间中的定义触发器。对于每个触发信息.Schema.triggers列出了各种属性，包括触发器的名称，关联的架构和表名称，EventManipulation属性（插入，更新，删除，插入/更新，ActionTiming属性（之前，之后），创建的属性（触发创建时间戳）和ActionStatement属性，它是生成的SQL触发器代码。

创建的属性从上次修改课程定义时派生触发创建时间戳。因此，随后使用此类（例如，定义其他触发器）可能导致创建属性值的意外更新。

可以从SQL查询中访问此信息.Schema.triggers信息，如下例所示：

```
SELECT TABLE_NAME,TRIGGER_NAME,CREATED,EVENT_MANIPULATION,ACTION_TIMING,ACTION_ORIENT
ATION,ACTION_STATEMENT
FROM INFORMATION_SCHEMA.TRIGGERS WHERE TABLE_SCHEMA='SQLUser'
```



#SQL #Cache #InterSystems IRIS #InterSystems IRIS for Health

源URL:
https://cn.community.intersystems.com/post/%E7%AC%AC%E5%8D%81%E4%B8%83%E7%AB%A0-%E4%BD%BF%E7%94%A8%E8%A7%A6%E5%8F%91%E5%99%A8