INSTEAD OF 触发器

可以在表或视图上指定 INSTEAD OF 触发器。执行这种触发器就能够替代原始的触发动作。 INSTEAD OF 触发器扩展了视图更新的类型。对于每一种触发动作 (INSERT 、UPDATE 或 DELETE) ，每一个表或视图只能有一个 INSTEAD OF 触发器。

INSTEAD OF 触发器被用于更新那些没有办法通过正常方式更新的视图。例如，通常不能在一个基于连接的视图上进行 DELETE 操作。然而，可以编写一个 INSTEAD OF DELETE 触发器来实现删除。上述触发器可以访问那些如果视图是一个真正的表时已经被删除的数据行。 将被删除的行存储在一个名为 deleted 的工作表中，就像 AFTER 触发器一样。相似地，在 UPDATE INSTEAD OF 触发器或者 INSERT INSTEAD OF 触发器中，你可以访问 inserted 表中的新行。

不能在带有 WITH CHECK OPTION 定义的视图中创建 INSTEAD OF 触发器。

INSTEAD OF触发器的主要优点是可以使不能更新的视图支持更新。基于多个基表的视图必须使用。 INSTEAD OF触发器来支持引用多个表中数据的插入、更新和删除操作。 INSTEAD OF触发器的另一个优点是使您得以编写这样的逻辑代码：在允许批处理的其他部分成功的同时拒绝批处理中的某些部分。

Transact-SQL 语句创建两个基表、一个视图和视图上的 INSTEAD OF 触发器。以下表将个人数据和业务数据分开并且是视图的基表。

```
1.  /* 在视图上定义 instead of insert 触发器以在一个或多个基表中插入数据。 */
2.  -- 部门表
3.  create table dept
4.  (
5.      d_id int primary key,
6.      d_name varchar(20)
7.  )
8.  -- 员工表
9.  create table emp
10. (
11.     e_id int primary key,
12.     e_name varchar(20),
13.     d_id int references dept(d_id)
```

```
14.  )
15.
16.  if exists(select * from sysobjects where name='v')
17.  drop view v
18.  go
19.  create view v
20.  as
21.  select e_id,e_name,d.d_id,d_name from emp e,dept d where e.d_id=d.d_id
22.  go
23.
24.  /******************          使用  instead of      触发器 ******************/
25.  if exists(select * from sysobjects where name='de_em_insert')
26.  drop trigger de_em_insert
27.  go
28.  create trigger de_em_insert
29.  on v
30.  instead of insert
31.  as
32.  begin
33.  if (not exists (select d.d_id from dept d, inserted i where d.d_id = i.d
     _id))
34.  insert into dept select d_id,d_name from inserted
35.  if (not exists (select e.d_id from emp e, inserted i where e.d_id = i.d_
     id))
36.  insert into emp select e_id,e_name,d_id from inserted
37.  else
38.  update emp set e_id = i.e_id,e_name = i.e_name from emp e, inserted i wh
     ere e.d_id = i.d_id
39.  end
40.
```

```
41.  select * from v
42.  insert into v values(1001,'          张珊 ',101,'    销售部 ')
43.
44.  select * from v
45.  select * from emp
46.  select * from dept
47.
```

可以在视图或表中定义 INSTEAD OF DELETE触发器，以代替 DELETE 语句的标准操作。 通常，在视图上定义 INSTEAD OF DELETE触发器以便在一个或多个基表中修改数据。

可在视图上定义 INSTEAD OF UPDATE触发器以代替 UPDATE语句的标准操作。通常，在视图上定义 INSTEAD OF UPDATE触发器以便修改一个或多个基表中的数据。

INSTEAD OF触发器

Instead Of 触发器与 After 触发器的工作流程是不一样的。 After 触发器是在 SQL Server 服务器接到执行 SQL语句请求之后，先建立临时的 Inserted 表和 Deleted 表，然后实际更改数据，最后才激活触发器的。而 Instead Of 触发器看起来就简单多了，在 SQL Server 服务器接到执行 SQL语句请求后，先建立临时的 Inserted 表和 Deleted 表，然后就触发了 Instead Of 触发器，至于那个 SQL语句是插入数据、 更新数据还是删除数据， 就一概不管了， 把执行权全权交给了 Instead Of 触发器，由它去完成之后的操作。

Instead Of 触发器的使用范围

Instead Of 触发器可以同时在数据表和视图中使用，通常在以下几种情况下，建议使用 Instead Of 触发器：

>> 数据库里的数据禁止修改：例如电信部门的通话记录是不能修改的，一旦修改，则通话费用的计数将不正确。在这个时候，就可以用 Instead Of 触发器来跳过 Update 修改记录的 SQL语句。

>> 有可能要回滚修改的 SQL语句：用 After 触发器并不是一个最好的方法，如果用 Instead Of 触发器，在判断折扣大于 0.6 时，就中止了更新操作，避免在修改数据之后再回滚操作，减少服务器负担。

>> 在视图中使用触发器：因为 After 触发器不能在视图中使用，如果想在视图中使用触发器，就只能用 Instead Of 触发器。

>> 用自己的方式去修改数据：如不满意 SQL直接的修改数据的方式，可用 Instead Of 触发器来控制数据的修改方式和流程。

一个例子：

```
-- 创建表

CREATE TABLE [tt] (
[a] [int] NOT NULL ,
[b] [nvarchar] (50) COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
[c] [nvarchar] (50) COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
CONSTRAINT [PK_tt] PRIMARY KEY CLUSTERED
(
   [a]
) ON [PRIMARY]
) ON [PRIMARY]
GO

-- 创建触发器

CREATE TRIGGER ta ON [dbo].[tt]
INSTEAD OF INSERT
AS
SET NOCOUNT ON
if (SELECT a from inserted)>3
   print N'       不能插入大于  3 的'
else
   insert into tt select * from inserted

-- 执行

NSERT INTO tt values(5,'a','aa')

select * from tt
```

可以看到 insert   语句并没有把数据插入进去。

虽然视图通常不能动态修改，但是，使用   INSTEAD OF 触发器您可以指定操作，保持视图最新，同时修改视图基础基表中的数据。例如，您可在视图上定义 INSTEAD OF INSERT触发器，以替换标准的   INSERT 语句。

假定在   pubs 数据库中以下面的视图开始：

```
CREATE VIEW AuthorsNames
AS
SELECT au_id, au_fname, au_lname
FROM authors
```

如果直接对 authors 表应用 INSERT 事件，则该视图将是不正确的，因为没有向该视图通知新插入的作者。避免此问题的方法是在该视图上创建 INSTEAD OF 触发器来处理插入。

```
CREATE TRIGGER ShowInsert on AuthorsNames
INSTEAD OF INSERT
AS
BEGIN
INSERT INTO authors
  SELECT address, au_fname, au_id, au_lname, city, contract, phone,
state, zip
  FROM inserted
END
```

有关 INSTEADOF 触发器的更多信息和示例，请参见数据库服务器的文档。如果当前所使用的是 Microsoft SQLServer，则请参见“ SQLServer 联机丛书”中的“ INSTEAD OF”。

PUBLISHERS表结构

PUB_ID PUB_NAME CITY STATE COUNTRY

以下为代码部分 ：

```
USE PUBS
GO
IF EXISTS(SELECT * FROM sysobjects WHERE NAME='PUBLISHERS_VIEW' AND
TYPE='V')

DROP VIEW PUBLISHERS_VIEW

go

-- 创建视图

CREATE VIEW PUBLISHERS_VIEW
AS
SELECT PUB_NAME,CITY,STATE,COUNTRY FROM PUBLISHERS

GO
INSERT INTO PUBLISHERS_VIEW
VALUES('工业出版社 ','北京','亚洲','中国')

-- 可以看到视图无法往基础表插入数据，因为视图不包括 pub_id， 而他又不能为空
```

提示：服务器： 消息 515，级别 16，状态 2，行 1
无法将 NULL 值插入列 'pub_id' ，表 'pubs.dbo.publishers' ；该列不允许空值。 INSERT 失败。
语句已终止。

以下为代码部分 ：

```
-- 下面创建一个 INSTEAD 触发器，在视图望基础表加数据时补充 PUB_ID
go

if exists(select name from sysobjects where
name='tri_ins_PUBLISHERS_view' and type='tr')

drop tri_ins_PUBLISHERS_view

go

create tri_ins_PUBLISHERS_view

on PUBLISHERS_VIEW

instead of insert

as
-- 声明变量
declare
@spub_id char(4),
@spub_name varchar(40),
@scity varchar(20),
@sstate char(2),
@sCountry varchar(30),
@n integer

-- 从 inserted 表读取要插入的数据
select
@spub_name=pub_name,
@scity=city,
@sstate=state,
@sCountry=Country
from inserted

-- 补充缺少的值
select @n='9900',@spub_id='9900'
```

```
while @spub_id in (select pub_id from PUBLISHERS)

begin
select @n=@n+1
select @spub_id=cast(@n as char(4))
end

insert into PUBLISHERS
values(@spub_id,@spub_name,@scity,@sstate,@sCountry)
```

现在执行

```
INSERT INTO PUBLISHERS_VIEW VALUES('S','出版社 ',' 北京 ','YZ',' 中国 ')
```

结果

（所影响的行数为 1 行）