# **[SQL Server 触发器](http://www.cnblogs.com/hoojo/archive/2011/07/20/2111316.html)**

触发器是一种特殊类型的存储过程，它不同于之前的我们介绍的存储过程。触发器主要是通过事件进行触发被自动调用执行的。而存储过程可以通过存储过程的名称被调用。

Ø 什么是触发器

    触发器对表进行插入、更新、删除的时候会自动执行的特殊存储过程。触发器一般用在check约束更加复杂的约束上面。触发器和普通的存储过程的区别是：触发器是当对某一个表进行操作。诸如：update、insert、delete这些操作的时候，系统会自动调用执行该表上对应的触发器。SQL Server 2005中触发器可以分为两类：DML触发器和DDL触发器，其中DDL触发器它们会影响多种数据定义语言语句而激发，这些语句有create、alter、drop语句。

    DML触发器分为：

    1、 after触发器（之后触发）

        a、 insert触发器

        b、 update触发器

        c、 delete触发器

    2、 instead of 触发器 （之前触发）

    其中after触发器要求只有执行某一操作insert、update、delete之后触发器才被触发，且只能定义在表上。而instead of触发器表示并不执行其定义的操作（insert、update、delete）而仅是执行触发器本身。既可以在表上定义instead of触发器，也可以在视图上定义。

    触发器有两个特殊的表：插入表（instered表）和删除表（deleted表）。这两张是逻辑表也是虚表。有系统在内存中创建者两张表，不会存储在数据库中。而且两张表的都是只读的，只能读取数据而不能修改数据。这两张表的结果总是与被改触发器应用的表的结构相同。当触发器完成工作后，这两张表就会被删除。Inserted表的数据是插入或是修改后的数据，而deleted表的数据是更新前的或是删除的数据。

|  |  |  |
| --- | --- | --- |
| 对表的操作 | Inserted逻辑表 | Deleted逻辑表 |
| 增加记录（insert） | 存放增加的记录 | 无 |
| 删除记录（delete） | 无 | 存放被删除的记录 |
| 修改记录（update） | 存放更新后的记录 | 存放更新前的记录 |

    Update数据的时候就是先删除表记录，然后增加一条记录。这样在inserted和deleted表就都有update后的数据记录了。注意的是：触发器本身就是一个事务，所以在触发器里面可以对修改数据进行一些特殊的检查。如果不满足可以利用事务回滚，撤销操作。

Ø 创建触发器

    语法

create trigger tgr\_name  
on table\_name  
with encrypion –加密触发器  
 for update...  
as  
 Transact-SQL

    # 创建insert类型触发器

--创建insert插入类型触发器  
if (object\_id('tgr\_classes\_insert', 'tr') is not null)  
 drop trigger tgr\_classes\_insert  
go  
create trigger tgr\_classes\_insert  
on classes  
 for insert --插入触发  
as  
 --定义变量  
 declare @id int, @name varchar(20), @temp int;  
 --在inserted表中查询已经插入记录信息  
 select @id = id, @name = name from inserted;  
 set @name = @name + convert(varchar, @id);  
 set @temp = @id / 2;   
 insert into student values(@name, 18 + @id, @temp, @id);  
 print '添加学生成功！';  
go  
--插入数据  
insert into classes values('5班', getDate());  
--查询数据  
select \* from classes;  
select \* from student order by id;

     insert触发器，会在inserted表中添加一条刚插入的记录。

    # 创建delete类型触发器

--delete删除类型触发器  
if (object\_id('tgr\_classes\_delete', 'TR') is not null)  
 drop trigger tgr\_classes\_delete  
go  
create trigger tgr\_classes\_delete  
on classes  
 for delete --删除触发  
as  
 print '备份数据中……';   
 if (object\_id('classesBackup', 'U') is not null)  
 --存在classesBackup，直接插入数据  
 insert into classesBackup select name, createDate from deleted;  
 else  
 --不存在classesBackup创建再插入  
 select \* into classesBackup from deleted;  
 print '备份数据成功！';  
go  
--  
--不显示影响行数  
--set nocount on;  
delete classes where name = '5班';  
--查询数据  
select \* from classes;  
select \* from classesBackup;

   delete触发器会在删除数据的时候，将刚才删除的数据保存在deleted表中。

    # 创建update类型触发器

--update更新类型触发器  
if (object\_id('tgr\_classes\_update', 'TR') is not null)  
 drop trigger tgr\_classes\_update  
go  
create trigger tgr\_classes\_update  
on classes  
 for update  
as  
 declare @oldName varchar(20), @newName varchar(20);  
 --更新前的数据  
 select @oldName = name from deleted;  
 if (exists (select \* from student where name like '%'+ @oldName + '%'))  
 begin  
 --更新后的数据  
 select @newName = name from inserted;  
 update student set name = replace(name, @oldName, @newName) where name like '%'+ @oldName + '%';  
 print '级联修改数据成功！';  
 end  
 else  
 print '无需修改student表！';  
go  
--查询数据  
select \* from student order by id;  
select \* from classes;  
update classes set name = '五班' where name = '5班';

     update触发器会在更新数据后，将更新前的数据保存在deleted表中，更新后的数据保存在inserted表中。

    # update更新列级触发器

if (object\_id('tgr\_classes\_update\_column', 'TR') is not null)  
 drop trigger tgr\_classes\_update\_column  
go  
create trigger tgr\_classes\_update\_column  
on classes  
 for update  
as  
 --列级触发器：是否更新了班级创建时间  
 if (update(createDate))  
 begin  
 raisError('系统提示：班级创建时间不能修改！', 16, 11);  
 rollback tran;  
 end  
go  
--测试  
select \* from student order by id;  
select \* from classes;  
update classes set createDate = getDate() where id = 3;  
update classes set name = '四班' where id = 7;

     更新列级触发器可以用update是否判断更新列记录；

    # instead of类型触发器

       instead of触发器表示并不执行其定义的操作（insert、update、delete）而仅是执行触发器本身的内容。

       创建语法

create trigger tgr\_name  
on table\_name  
with encryption  
 instead of update...  
as  
 T-SQL

      # 创建instead of触发器

if (object\_id('tgr\_classes\_inteadOf', 'TR') is not null)  
 drop trigger tgr\_classes\_inteadOf  
go  
create trigger tgr\_classes\_inteadOf  
on classes  
 instead of delete/\*, update, insert\*/  
as  
 declare @id int, @name varchar(20);  
 --查询被删除的信息，病赋值  
 select @id = id, @name = name from deleted;  
 print 'id: ' + convert(varchar, @id) + ', name: ' + @name;  
 --先删除student的信息  
 delete student where cid = @id;  
 --再删除classes的信息  
 delete classes where id = @id;  
 print '删除[ id: ' + convert(varchar, @id) + ', name: ' + @name + ' ] 的信息成功！';  
go  
--test  
select \* from student order by id;  
select \* from classes;  
delete classes where id = 7;

      # 显示自定义消息raiserror

if (object\_id('tgr\_message', 'TR') is not null)  
 drop trigger tgr\_message  
go  
create trigger tgr\_message  
on student  
 after insert, update  
as raisError('tgr\_message触发器被触发', 16, 10);  
go  
--test  
insert into student values('lily', 22, 1, 7);  
update student set sex = 0 where name = 'lucy';  
select \* from student order by id;

    # 修改触发器

alter trigger tgr\_message  
on student  
after delete  
as raisError('tgr\_message触发器被触发', 16, 10);  
go  
--test  
delete from student where name = 'lucy';

    # 启用、禁用触发器

--禁用触发器  
disable trigger tgr\_message on student;  
--启用触发器  
enable trigger tgr\_message on student;

    # 查询创建的触发器信息

--查询已存在的触发器  
select \* from sys.triggers;  
select \* from sys.objects where type = 'TR';  
  
--查看触发器触发事件  
select te.\* from sys.trigger\_events te join sys.triggers t  
on t.object\_id = te.object\_id  
where t.parent\_class = 0 and t.name = 'tgr\_valid\_data';  
  
--查看创建触发器语句  
exec sp\_helptext 'tgr\_message';

    # 示例，验证插入数据

if ((object\_id('tgr\_valid\_data', 'TR') is not null))  
 drop trigger tgr\_valid\_data  
go  
create trigger tgr\_valid\_data  
on student  
after insert  
as  
 declare @age int,  
 @name varchar(20);  
 select @name = s.name, @age = s.age from inserted s;  
 if (@age < 18)  
 begin  
 raisError('插入新数据的age有问题', 16, 1);  
 rollback tran;  
 end  
go  
--test  
insert into student values('forest', 2, 0, 7);  
insert into student values('forest', 22, 0, 7);  
select \* from student order by id;

    # 示例，操作日志

if (object\_id('log', 'U') is not null)  
 drop table log  
go  
create table log(  
 id int identity(1, 1) primary key,  
 action varchar(20),  
 createDate datetime default getDate()  
)  
go  
if (exists (select \* from sys.objects where name = 'tgr\_student\_log'))  
 drop trigger tgr\_student\_log  
go  
create trigger tgr\_student\_log  
on student  
after insert, update, delete  
as  
 if ((exists (select 1 from inserted)) and (exists (select 1 from deleted)))  
 begin  
 insert into log(action) values('updated');  
 end  
 else if (exists (select 1 from inserted) and not exists (select 1 from deleted))  
 begin  
 insert into log(action) values('inserted');  
 end  
 else if (not exists (select 1 from inserted) and exists (select 1 from deleted))  
 begin  
 insert into log(action) values('deleted');  
 end  
go  
--test  
insert into student values('king', 22, 1, 7);  
update student set sex = 0 where name = 'king';  
delete student where name = 'king';  
select \* from log;  
select \* from student order by id;

https://www.cnblogs.com/hoojo/archive/2011/07/20/2111316.html