项目 12 触发器的应用

一、目的和要求

1. 理解触发器的作用
2. 应能熟练创建，修改，删除触发器
3. 在实际应用开发时能够灵活运用触发器完成业务规则以达到简化系统整体设计的目的。

二、实训内容

创建，修改，删除触发器
在实际应用开发时能够灵活运用触发器

三、实验环境

WindowsXP/windows 7 和 SQL Server Management Studio 的安装环境

四、知识点学习：

触发器的概念

触发器是一种特殊的存储过程，它不能被显式地调用，而是在往表中插入记录、更新记录或者删除记录时被自动地激活。 所以触发器可以用来实现对表实施复杂的完整性约束。

SQL Server 为每个触发器都创建了两个专用临时表： INSERTED 表和 DELETED 表。

这两个表的结构与激发触发器的表的结构相同。用户不能对它们进行修改，只能在触发器程序中查询表中的内容。触发器执行完毕后，与该触发器相关的这两个表也会被删除。

当执行 INSERT 语句时， INSERTED 表存放要向表中插入的所有行。当执行 DELETE 语句时，DELETED 表存放要从表中删除的所有行。当执行 UPDATE 语句时，相当于先执行一个 DELETE 操作，再执行一个 INSERT 操作。所以旧的行被移动到 DELETED 表，而新的行插入到 INSERTED 表。

使用触发器的优点

（1）多张表的级联修改。
（2）强于 CHECK 约束的复杂限制。
（3）比较数据修改前后的差别。
（4）强制表的修改要合乎业务规则。

创建和应用触发器

在创建触发器时，需要指定触发器的名称、包含触发器的表、引发触发器的条件以及当触发器启动后要执行的语句等内容。

使用 CREATE TRIGGER 命令创建触发器的语法格式如下。

```
CREATE TRIGGER 触发器名
ON {表|视图}
[ WITH ENCRYPTION ]
{ FOR | AFTER | INSTEAD OF } { [ INSERT ] [ , ]
[ UPDATE ] [ , ] [DELETE] }
[ NOT FOR REPLICATION ]
AS
[{IF UPDATE(列名)[{AND|OR} UPDATE(列名)][ ...n ]}
SQL 语句
```

注意：

AFTER 或 FOR 关键字，创建的是后触发，即当引起触发器执行的修改语句完成后，并通过了各种约束检查后，才执行触发器中的语句。后触发只能建在表上，不能建在视图上。

INSTEAD OF 关键字，创建的是替代触发。替代触发的特征是引起触发器执行的修改语句只起到启动触发器的作用，而并没有执行，取而代之的是执行触发器中的语句。替代触发可以建在表上或视图上。

INSERT 触发器

通过 INSERT 触发器检查添加操作的业务规则。

UPDATE 触发器

对于 UPDATE 触发器，当 UPDATE 操作在表上执行时，则产生触发。

在触发器程序中，有时只关心某些列的变化，则可以使用 IF UPDATE（列名），仅对指定列的修改作出反应，这点是其他两种触发器没有的。

DELETE 触发器

当对表执行 DELETE 操作时，激发该表的 DELETE 触发器。

查看触发器的定义信息

在 SQL Server Management Studio 对象资源管理器下，可以轻松创建触发器、查看触发器的定义信息和修改触发器。

修改和删除触发器

1. 修改触发器

修改触发器的语法格式如下。

```
ALTER TRIGGER 触发器名
ON {表|视图}
[ WITH ENCRYPTION ]
{ FOR | AFTER | INSTEAD OF } { [ INSERT ] [ , ] [ UPDATE ] [ , ]
[DELETE] }
[ NOT FOR REPLICATION ]
AS
[{IF UPDATE(列名)[{AND|OR} UPDATE(列名)][ ...n ]}
SQL 语句
```

2. 删除触发器

触发器的删除是通过 DROP 语句来实现的，在 SQL Server Management Studio 也同样可以进行删除。

禁止或启用触发器

禁止和启用触发器的语法格式如下。

```
ALTER TABLE 表名
{ENABLE|DISABLE} TRIGGER
{ALL|触发器名[,...n]}
```

使用该语句可以禁用或启用指定表上的某些触发器或所有触发器。

总结：

1 inserted 、deleted 表
2 创建触发器：after、for 触发器的不同
3 如何检查触发器的效果
4 修改、查看触发器

五、实验步骤

（一）附加数据库 student 和 XK 数据库
打开 SQL Server Management Studio
选择数据库节点，右键单击选择附加数据库
找到 student_data.mdf 和 student_log.ldf 所在目录
将给定的数据库附加到 SQL Server Management Studio 中
（二）打开 student 数据库，完成以下操作

教师演示操作题：

```
--创建 insert 触发器
use student
go
create trigger student_tr1
on student
for insert
as
begin
print '禁止你插入记录'
rollback tran
end
--以下是演示 insert 触发器的结果
USE student
GO
INSERT INTO student
    (sno,sname,ssex,sbirthday,sscore,classno)
VALUES
     ('0701011101','孙龙','男','1988-6-4',479,'07010111')
--创建 delete 触发器
use student
go
create trigger student_tr2
on student
for delete
as
begin
print '禁止你删除记录'
rollback tran
end
--以下是演示 insert 触发器的结果
delete from student
--创建 insert 触发器:因为用了 instead of 所以不用事务回滚了。
use student
go
create trigger student_tr3
on student
instead of insert
as
print '禁止你插入记录'
go
--以下是演示 insert 触发器的结果
select * from student

--创建 update 触发器
use student
go
```

```
create trigger student_tr5
on student
for update
as
begin
if update(sname)
print '你不能修改记录'
rollback tran
end
go
--以下是演示 insert 触发器的结果
update student
set sname='abc'
where sname='孙晓龙'


-- 创建 delete,insert,update 触发器
use student
go
create trigger student_delete_inset_update on student
for delete,insert,update
as
begin
print'你不能删除，插入，修改记录'
rollback tran
end

--创建 insert 触发器
use student
go
create trigger student_insert on student
for insert
as
select count(*) as 共计人数 from student
go

select count(*)from student
select *from student
--以下是演示 insert 触发器的结果
USE student
GO
INSERT INTO student
    (sno,sname,ssex,sbirthday,sscore,classno)
VALUES
('0701011101','孙龙','男','1988-6-4',479,'07010111')


--创建一个 delete 触发器：当我们在 class 表中删除了班级编号，那么对应的 student 表
也要删除。
select * from student
select * from class
use student
go
create trigger student_s_c_tr on class
for delete
as
begin
delete from student where classno in(select classno from deleted)
```

```
print '相应的记录被删除了'
end
go
 --以下是演示 insert 触发器的结果
delete from class   where classno='07010211'


--触发器的管理:
sp_help student_insert  --查一般信息
sp_depends student_insert  -- 查依赖关系
sp_helptext student_insert --查代码
sp_helptrigger student --查一个表中有多少触发器
sp_rename student_s_c_tr,s_c_tr --改名
drop trigger s_c_tr  --删除
alter table student disable trigger student_insert  --禁用触发器
alter table student enable trigger student_insert --使用触发器
```

学生操作题：实验内容与步骤

任务 1 创建一个触发器，实现每当修改 student 表的数据时，向客户端显示一条：记录已修改！的消息。（最基本触发器示例）

任务 2 在 management studio 中查看触发器 TEST1 的属性。

任务 3 将上面中 test1 触发器中的 for update 修改为 inseted of update 并进行比较。

任务 4 创建一个触发器，实现当插入，更新或删除 stucou 表的选课数据行时，能同时更新表中相应的报名人数。

任务 5 知识点：使用 if update(colum)。创建一人触发器，完成功能是：每当在 student 表中修改 pwd 列的数据时，将向客户端显示一条消息。

任务 6 知识点：多个触发器。创建一人触发器，完成功能是：每当在 student 表中修改数据时，将向客户端显示一条消息：二次触发！。

任务 7 使用 transact-sql 语句删除触发器 test1

任务 8 使用 management studio 删除触发器 test2

任务 9 使用 transact-sql 语句将触发器 setwillnum 更名为：updatewillnum.

任务 10 知识点，禁用触发器。使用 transact-sql 语句禁用触发器 updatewillnum

任务 11 知识点，恢复使用触发器。使用 transact-sql 语句恢复使用触发器

任务 12 使用 transact-sql 语句查询 xk 数据库中有哪些触发器。

作业：

1 创建触发器，订单信息表中如果插入新的订单给予提示
2 创建触发器，如果插入或更新的订货日期大于现在的日期，则提示错误，并且插入或更新数据失败。
3 创建触发器，如果修改销售人员的部门号，给予提示：“销售人员×××的部门号已经修改”

同学们可以按以下内容学习：

1. 概述

触发器是一种特殊的存储过程，它不能被显式地调用，而是在往表中插入记录、更新记录或者删除记录时被自动地激活。 所以触发器可以用来实现对表实施复杂的完整性约束。

2. 触发器的分类

SQL Server2000 提供了两种触发器:“Instead of” 和 “After” 触发器。
一个表或视图的每一个修改动作(Insert、Update 和 Delete)都可以有一个 “Instead of” 触发器，一个表的每个修改动作都可以有多个 “After”触发器。

2.1 “Instead of”触发器

“Instead of”触发器在执行真正 “插入” 之前被执行。除表之外， “Instead of” 触发器也可以用于视图，用来扩展视图可以支持的更新操作。
“Instead of”触发器会替代所要执行的 SQL 语句，言下之意就是所要执行 SQL 并不会 “真正执行”

```
alter trigger trigger_学生_Delete
on 学生
instead of Delete
as
begin
      select 学号, 姓名  from deleted
end

delete from 学生  where 学号  = 4
```

上例中定义了 “trigger 学生_Delete”触发器，该触发器从 “delete”表中打印出所要删除的学生.在执行 “delete”操作后，会发现 “学号 = 4”的学生并未被删除， 原因在于 “trigger 学生 Delete”替代了所要执行的 “delete from 学生 where 学号 = 4”语句，而在 “trigger 学生_Delete”中并未真正删除学生。

2.2 “After”触发器

“After”触发器在 Insert、Update 或 Deleted 语句执行之后被触发。“After”触发器只能用于表。

“After”触发器主要用于表在修改后（insert、update 或 delete 操作之后），来修改其他表

3. Inserted 和 Deleted 表

SQL Server 为每个触发器都创建了两个专用表:Inserted 表和 Deleted 表。

这两个表由系统来维护，它们存在于内存中而不是在数据库中，可以理解为一个虚拟的表。

这两个表的结构总是与被该触发器作用的表的结构相同。

触发器执行完成后，与该触发器相关的这两个表也被删除。

Deleted 表存放由于执行 Delete 或 Update 语句而要从表中删除的所有行。

Inserted 表存放由于执行 Insert 或 Update 语句而要向表中插入的所有行。

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

4. 触发器的执行过程

如果一个 Insert、update 或者 delete 语句违反了约束，那么这条 SQL 语句就没有执行成功，因此“After”触发器也不会被激活。

“Instead of” 触发器可以取代激发它的操作来执行。它在 Inserted 表和 Deleted 表刚刚建立，其它任何操作还没有发生时被执行。因为“Instead of” 触发器在约束之前执行，所以它可以对约束进行一些预处理。

5. 创建触发器

```
create trigger trigger_name
on  {table_name|view_name}
{After|Instead of} {insert|update|delete}
as 相应 T-SQL 语句
```

6. 修改触发器：

```
alter trigger trigger_name
on  {table_name|view_name}
{After|Instead of} {insert|update|delete}
as 相应 T-SQL 语句
```

7. 删除触发器：

```
drop trigger trigger_name
```

8. 查看数据库中已有触发器：

8.1 查看数据库中所有触发器

```
select * from sysobjects where xtype='TR'
```

8.2 查看单个触发器

```
exec sp_helptext '触发器名'
```

9. “Instead of”相关示例:

两张表：学生(学号 int, 姓名 varchar)、借书记录(学号 int, 图书编号 int)

实现功能：在删除学生表时，如果该学生仍有借书记录（未还）则不能删除

```
alter trigger trigger_学生_Delete
on 学生
instead of Delete
as
begin
      if not exists(select * from 借书记录, deleted where 借书记录.学号  = deleted.学号)
            delete from 学生  where 学生.学号  in (select 学号  from deleted)
end
```

10. “After”触发器

10.1 在“订单”表中建立触发器，当向“订单”表中插入一条订单记录时，检查“商品”表的货品状态“状态”是否为1(正在整理)，则不能往“订单”表加入该订单。

```
create trigger trigger_订单_insert
on 订单
after insert
as
    if (select 状态  from 商品, inserted where 商品.pid = inserted.pid)=1
    begin
        print 'the goods is being processed'
        print 'the order cannot be committed'
        rollback transaction --回滚，避免加入
    end
```

该示例中“pid”为商品编码

该示例的 if 判断严格来讲是不准确的，因为“订单”表如果每次插入一条记录，该判断没有问题；如果一次插入多条记录，则“select 状态”返回的是多行。

10.2 在“订单”表建立一个插入触发器，在添加一条订单时，减少“商品”表相应的货品记录中的库存。

```
create trigger trigger_订单_insert2
on 订单
after insert
as
    update 商品  set 数量  = 数量  - inserted.数量
    from 商品, inserted
    where 商品.pid = inserted.pid
```

10.3 在“商品”表建立删除触发器，实现“商品”表和“订单”表的级联删除。

```
create trigger goodsdelete trigger_商品_delete
on 商品
after delete
as
    delete from 订单  where 订单.pid in (select pid from deleted)
```

10.4 在“订单”表建立一个更新触发器，监视“订单”表的“订单日期”列，使其不能被“update”.

```
create trigger trigger_订单_update
on 订单
after update
as
    if update(订单日期)
    begin
        raiserror('订单日期不能手动修改',10,1)
        rollback transaction
    end
```

10.5 在“订单”表建立一个插入触发器，保证向“订单”表插入的货品必须要在“商品”表中一定存在。

```
create trigger trigger_订单_insert3
on 订单
after insert
as
    if (select count(*) from 商品, inserted where 商品.pid = inserted.pid)=0
    begin
        print '商品不存在'
```

```
            rollback transaction
        end
```

10.6“订单”表建立一个插入触发器，保证向“订单”表插入的货品信息要在“订单日志”表中添加

```
alter trigger trigger_订单_insert
on 订单
for insert
as
        insert into 订单日志  select inserted.Id, inserted.pid,inserted.数量  from inserted
```

Sqlserver 示例

触发器 insert

```
1     create trigger tri_insert
2     on student
3     for insert
4     as
5     declare @student_idchar(10)
6     select @student_id=s.student_id from students
7     inner join insertedion s.student_id=i.student_id
8     if @student_id='0000000001'
9     begin
10    raiserror('不能插入 1 的学号!',16,8)
11    rollbacktran
12    end
13    go
```

触发器 update

?

```
1     create trigger tri_update
2     on student
3     for update
4     as
5     if update(student_id)
6     begin
7     raiserror('学号不能修改!',16,8)
8     rollbacktran
9     end
10    go
```

触发器 delete

```
1     create trigger tri_delete
2     on student
3     for delete
4     as
5     declare @student_idvarchar(10)
6     select @student_id=student_id from deleted
7     if @student_id='admin'
8     begin
9     raiserror('错误',16,8)
```

```
10 rollbacktran
11 end
```