

# 一. 实验目的

- 熟悉SQL Server的事务控制语言，能够熟练使用事务控制语言来编写事务处理程序。

# 二. 实验结果

- 编写一个嵌套事务。外层修改students表某记录，内层在teachers表插入一条记录。演示内层插入操作失败后，外层修改操作回滚

我们先查看一下原来的数据

```
SELECT * FROM STUDENTS ORDER BY SID
SELECT * FROM TEACHERS ORDER BY TID
```

结果消息

	sid	sname	email	grade
1	800001216	gfxrgs	hhce4@qhldj.gov	1992
2	800002933	vnbnqzsvv	pvhxd4l@zqur.org	2002
3	800005753	waqcj	hlhq0h8@jdba.gov	1992

	tid	tname	email	salary
1	200003125	fqmmyi	wcjcj@glq.net	3928
2	200005322	udjom	gd5l8@ppmlf.gov	873
3	200009423	jhoul...	12cj@pzc.gov	3286

查询已成功执... | (local)\CHONOR (10.0 RTM) | DESKTOP-CHONOR\Chonor ... | School1 | 00:00:00 | 115000 行

之后我们更新第一条学生数据，然后插入个第一条教师数据相同tid的数据，此时因为tid不能重复所以会插入失败。

```
SET XACT_ABORT ON
BEGIN TRAN up_stu
UPDATE STUDENTS
SET grade='1992'
WHERE sid='800001216'
    BEGIN TRAN insert_tea
        INSERT INTO TEACHERS
        VALUES('200003125','test','xxx@xxx','9999')
    COMMIT TRAN insert_tea
COMMIT TRAN up_stu
```

结果如下：



```

UPDATE TEACHERS
SET salary='9999'
WHERE tid='200003125'
SAVE TRAN up_tea_done
INSERT INTO COURSES
VALUES ('10001','test','999')
IF @@ERROR!=0
BEGIN
    ROLLBACK TRAN up_tea_done
    PRINT '插入课程表失败'
    RETURN
END
COMMIT TRAN up_tea

```



(1 行受影响)  
 消息 2627, 级别 14, 状态 1, 第 7 行  
 违反了 PRIMARY KEY 约束 'PK\_COURSES'。不能在对象 'dbo.COURSES' 中插入重复键。  
 语句已终止。  
 插入课程表失败

⚠ 查询已完成, 但有错... | (local)\CHONOR (10.0 RTM) | DESKTOP-CHONOR\Chonor ... | School1 | 00:00:00 | 0 行

此时再次查询

```

SELECT * FROM TEACHERS ORDER BY TID
SELECT * FROM COURSES ORDER BY CID

```



	tid	tname	email	salary
1	200003125	fqmmyi	wjcg@glq.net	9999
2	200005322	udjom	gd518@ppmlf.gov	873
3	200009423	jhoulvtr	12cj@pzc.gov	3286
4	200010493	xzdkfko	zcf_m@def.com	3332

  

	cid	cname	hour
1	10001	database	96
2	10002	operating system	88

✅ 查询已成功执... | (local)\CHONOR (10.0 RTM) | DESKTOP-CHONOR\Chonor ... | School1 | 00:00:00 | 15000 行

此时教师工资更新成功没有回滚, 而课程表没有变化。

- 编写一个包含事务的存储过程, 用于更新courses表的课时。如果更新记录的cid不存在, 则输出“课程信息不存在”, 其他错误输出“修改课时失败”, 如果执行成功, 则输出“课时修改成功”。调用该存储过程, 演示更新成功与更新失败的操作。

```

CREATE PROCEDURE UPDATECOURSEINFO
@cid_ char(10),
@hour_ int,
@returnStr varchar(100) out
AS
BEGIN TRAN

```

```

IF NOT EXISTS(SELECT CID FROM COURSES WHERE CID=@cid_)
BEGIN
    SELECT @returnStr='课程信息不存在'
    GOTO ONERROR
END
UPDATE COURSES
SET hour=@hour_
WHERE cid=@cid_
IF @@ERROR<>0
BEGIN
    SELECT @returnStr='修改课时失败'
    GOTO ONERROR
END
SELECT @returnStr='修改课时成功'
PRINT @returnStr
COMMIT TRAN
ONERROR:
PRINT @returnStr
ROLLBACK TRAN
GO

```

结果如下：



命令已成功完成。

✓ 查询已成功执行。 | (local)\CHONOR (10.0 RTM) | DESKTOP-CHONOR\Chonor ... | School1 | 00:00:00 | 0 行

演示更新成功

```

declare @cid_ char(10)
declare @hour_ int
declare @returnStr varchar(100)
exec UPDATECOURSEINFO '10001',100,@returnStr out
SELECT * FROM COURSES ORDER BY CID

```



(1 行受影响)  
 修改课时成功  
 修改课时成功  
 消息 3903, 级别 16, 状态 1, 过程 UPDATECOURSEINFO, 第 25 行  
 ROLLBACK TRANSACTION 请求没有对应的 BEGIN TRANSACTION。

	cid	cname	hour
1	10001	database	100
2	10002	operating system	88
3	10003	computer graphics	48
4	10004	java	48
5	10005	c++	60

更新失败的操作，此时修改一个不存在的课程

```
declare @cid_ char(10)
declare @hour_ int
declare @returnStr varchar(100)
exec UPDATECOURSEINFO '11001',100,@returnStr out
```

消息

课程信息不存在

查询已成功执行。

(local)\CHONOR (10.0 RTM)

DESKTOP-CHONOR\Chonor ...

School1

00:00:00 0 行

演示修改课时失败的操作2，先加入一个约束要求课时大于0,之后进行更新hour=0

```
ALTER TABLE COURSES ADD CONSTRAINT Uhour CHECK (hour>0)

declare @cid_ char(10)
declare @hour_ int
declare @returnStr varchar(100)
exec UPDATECOURSEINFO '10001',0,@returnStr out
```

查询结果如下

消息

消息 547, 级别 16, 状态 0, 过程 UPDATECOURSEINFO, 第 12 行  
UPDATE 语句与 CHECK 约束"Uhour"冲突。该冲突发生于数据库"School1", 表"dbo.COURSES", column 'hour'。  
语句已终止。  
修改课时失败

查询已完成，但有错误。

(local)\CHONOR (10.0 RTM)

DESKTOP-CHONOR\Chonor ...

School1

00:00:00 0 行

结果 消息

	cid	cname	hour
1	10001	database	100
2	10002	operating system	88
3	10003	computer graphics	48
4	10004	java	48
5	10005	c++	60

查询已成功执行。

(local)\CHONOR (10.0 RTM)

DESKTOP-CHONOR\Chonor ...

School1

00:00:00 50 行

此时更新失败

### 三. 实验感想

---

这次实验是在学习事务的使用方式，之前实验中也有事务的内容，这次实验多增加了事务的嵌套，回滚，保存点和事务的存储过程，第三个任务中想要演示修改失败，结果发现原数据表中并没有对课时进行约束，也就是怎么修改都能成功，只要id存在，所以多加了一个约束进行修改失败的演示。