

Lecture 13, Fall 2017/2018

数据库系统实验

Yubao Liu (刘玉葆)

School of Data and Computer Science

Sun Yat-sen University

- 本节课提纲

- 实验目的

- 实验示例

- 练习

- 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。

实验示例

1.假设学校将学生的银行卡和校园卡进行了绑定，允许学生直接从银行卡转账到校园卡中。假设某学号为05212222的学生需要从银行卡中转账100元到校园卡中，编写事务处理程序，实现这一操作。

```
4.1.2.sql - (local)...\NBRTDQO\dgn (53))*  SQLQuery1.sql - ...NBRTDQO\dgn (52))
1  use school
2  go
3  begin tran      事务开始
4  update icbc_card
5  set restored_money=restored_money-100
6  where stu_card_id='05212222'
7
8  update stu_card
9  set remained_money =remained_money+100
10 where card id='05212222'
11 commit tran    事务结束
```

消息

(1 行受影响)

(1 行受影响)

实验示例

2.事务与批处理的差别：批处理是由一条或多条SQL语句组成，用go语句来终止语句组。批处理与事务的差别在于：批处理中每条语句单独完成或失败，不会影响其他语句的执行。

SQLQuery2.sql - (...BRTDQO\dgn (57))* 4.1.3.sql - (local)...-NBRTDQO\dgn (54))*

```
1 select * from courses where cid='10002'
```

结果 消息

	cid	cname	hour
1	10002	operating system	96

1).首先查询课程10002的课时为96

SQLQuery2.sql - (...BRTDQO\dgn (57))* 4.1.3.sql - (local)...-NBRTDQO\dgn (54))*

```
1 Update courses set hour=90 where cid='10002'
2 Insert teachers values ('1234567895', 'MY', 'MY@ZSU.EDU.CN', 3000)
3 go
```

消息

(1 行受影响)
消息 2627, 级别 14, 状态 1, 第 2 行
违反了 PRIMARY KEY 约束 'PK_TEACHERS'。不能在对象 'dbo.TEACHERS' 中插入重复键。
语句已终止。

SQLQuery2.sql - (...BRTDQO\dgn (57))* 4.1.3.sql - (local)...-NBRTDQO\dgn (54))*

```
1 select * from courses where cid='10002'
```

结果 消息

	cid	cname	hour
1	10002	operating system	90

2).执行批处理语句：
将课程10002的课时改为90，然后插入一条记录。

3).再次查询课时10002的课时，课程10002的课时已修改成功。

【可以发现，批处理语句虽然插入记录的语句执行失败，但并不影响第一句更新操作的执行。如果将批处理换为事务，则修改失败，同学们可自行验证】⁴

实验示例

3. 嵌套事务的示例。

全局变量，用于记录SQL Server当前等待提交的事务数

```
1 SELECT 'BEFORE TRANSACTION:' AS HINT, @@TRANSCOUNT AS TRANSACTIONCOUNT
2 BEGIN TRAN
3 SELECT 'THE FIRST TRANSACTION STARTS:' AS HINT, @@TRANSCOUNT AS TRANSACTIONCOUNT
4     SELECT TOP 1 * FROM CHOICES
5     BEGIN TRAN
6         SELECT 'THE SECOND TRANSACTION STARTS:' AS HINT, @@TRANSCOUNT AS TRANSACTIONCOUNT
7     COMMIT TRAN
8     SELECT 'THE SECOND TRANSACTION COMMITS' AS HINT, @@TRANSCOUNT AS TRANSACTIONCOUNT
9 ROLLBACK TRAN
10 SELECT 'THE FIRST TRANSACTION ROLL BACK' AS HINT, @@TRANSCOUNT AS TRANSACTIONCOUNT
```

HINT	TRANSACTIONCOUNT
1 BEFORE TRANSACTION:	0
1 THE FIRST TRANSACTION STARTS:	1
no sid tid cid score	
1 500000058 823069829 249596497 10037 76	
1 THE SECOND TRANSACTION STARTS:	2
1 THE SECOND TRANSACTION COMMITS	1
1 THE FIRST TRANSACTION ROLL BACK	0

第一个事务

嵌套第二个事务

嵌套的事务执行完毕后，还剩下外部一个事务

查询已成功执行。 (local) (10.0 RTM) DESKTOP-NBRTDQO\dgn (58) School 00:00:00 6 行

实验示例

4. 触发器被视为数据修改事务的一部分。为表Courses的删除操作创建一个触发器，然后执行一个删除操作，观察事务数目的变化。

```
4.1.5.sql - (local...P-NBRTDQO\dgn (59)) 4.1.4.sql - (local)...-NBRTDQO\dgn (58))*
1 CREATE TRIGGER TD_COURSE ON COURSES
2   FOR DELETE
3   AS
4   DECLARE @INFO VARCHAR(255)
5   SELECT @INFO='触发器中的事务数据为: ' + CONVERT(VARCHAR(2), @@TRANCOUNT)
6   PRINT @INFO
7   RETURN
```

select用于变量赋值

Convert函数，用于转换数据格式

消息
命令已成功完成。

再执行以下代码：

```
SQLQuery3.sql - (...BRTDQO\dgn (61))* 4.1.6.sql - (local...P-NBRTDQO\dgn (60))
1 print '删除操作以前触发器中事务数为:' + convert(varchar(2), @@TRANCOUNT);
2
3 delete from COURSES
4 where cid='10052'
5
6 print '删除操作之后触发器中事务数为:' + convert(varchar(2), @@TRANCOUNT);
```

消息
删除操作以前触发器中事务数为: 0
触发器中的事务数据为: 1
(0 行受影响)
删除操作之后触发器中事务数为: 0

可以发现，在删除操作执行过程中，触发器得到执行，而且事务数为1。从而验证，触发器事务是数据修改事务的一部分。

实验示例

5.在存储过程中使用事务的示例。

1).创建存储过程

```
SQLQuery3.sql - (...BRTDQO\dgn (55))* 4.1.8.sql - (local)...-NBRTDQO\dgn (54))* SQLQue
1 CREATE PROCEDURE INSERTCOURSEINFO
2     @courseid char(10),
3     @coursename varchar(30),
4     @hour int,
5     @returnString varchar(100) out
6 AS
7 BEGIN TRAN
8     IF EXISTS( SELECT CID FROM COURSES WHERE CID=@COURSEID)
9     BEGIN
10         SELECT @returnString='课程信息已经存在'
11         GOTO ONERROR
12     END
13     --新增课程信息
14     INSERT INTO COURSES VALUES (@courseid,@courseName,@hour)
15     IF @@ERROR<>0
16     BEGIN
17         SELECT @returnString='新增课程信息失败'
18         GOTO ONERROR
19     END
20     -----错误处理
21     SELECT @returnString='新增课程信息成功'
22     PRINT @returnString
23     COMMIT TRAN
24 ONERROR:
25     PRINT @returnString
26     ROLLBACK TRAN
27 GO
```

全局变量，如果语句执行失败则不为0

2).调用存储过程，插入已存在的记录，显示“课程信息已经存在”

```
SQLQuery3.sql - (...BRTDQO\dgn (55))* 4.1.8.sql - (local)...-NBRTDQO\dgn (54))* SQLQue
1 declare @courseid char(10)
2 declare @coursename varchar(30)
3 declare @hour int
4 declare @returnString varchar(100)
5 exec INSERTCOURSEINFO '10001','english',90,@returnString out
```

消息
课程信息已经存在

3).调用存储过程，插入记录成功，显示“新增课程信息成功”

```
SQLQuery3.sql - (...BRTDQO\dgn (55))* 4.1.8.sql - (local)...-NBRTDQO\dgn (54))* SQLQue
1 declare @courseid char(10)
2 declare @coursename varchar(30)
3 declare @hour int
4 declare @returnString varchar(100)
5 exec INSERTCOURSEINFO '10051','english',90,@returnString out
```

消息
(1 行受影响)
新增课程信息成功
新增课程信息成功
消息 3903, 级别 16, 状态 1, 过程 INSERTCOURSEINFO, 第 26 行
ROLLBACK TRANSACTION 请求没有对应的 BEGIN TRANSACTION。

实验示例

6.命名事务。通过对事务命名，使事务易于识别，特别是事务嵌套时，可提高代码可读性。下面定义两个事务，外层事务更新表Courses，内层事务更新表Teachers。

```
SQLQuery4.sql - (...BRTDQO\dgn (63))* 4.1.6.sql - (local)...-NBRTDQO\dgn (60))*
1 BEGIN TRAN tran_upd_courses --外层事务命名
2   update courses
3   set hour=60
4   where cid='10001'
5   BEGIN TRAN tran_upd_teachers --内层事务命名
6     insert into teachers
7     values ('1234567890', 'zs', 'my@zsu.edu.cn', 3000)
8   IF @@ERROR!=0
9     BEGIN
10      -----撤消事务
11      ROLLBACK TRAN tran_upd_teacher
12      PRINT '更新教师表失败'
13      RETURN
14    END
15  --提交内层事务
16  COMMIT TRAN tran_upd_teachers
17  -----提交外层事务
18  COMMIT TRAN tran_upd_courses
```

消息

(1 行受影响)
消息 2627, 级别 14, 状态 1, 第 6 行
违反了 PRIMARY KEY 约束 'PK_TEACHERS'。不能在对象 'dbo.TEACHERS' 中插入重复键。
语句已终止。
消息 6401, 级别 16, 状态 1, 第 11 行
无法回滚 tran_upd_teacher。找不到该名称的事务或保存点。
更新教师表失败

实验示例

若插入未重复的记录，事务执行成功。



```
SQLQuery4.sql - (...BRTDQO\dgn (63))* 4.1.6.sql - (local)...-NBRTDQO\dgn (60))*
1 BEGIN TRAN tran_upd_courses
2   update courses
3   set hour=60
4   where cid='10001'
5   BEGIN TRAN tran_upd_teachers
6     insert into teachers
7     values ('1234567899','zs','my@zsu.edu.cn',3000)
8   IF @@ERROR!=0
9     BEGIN
10      -----撤消事务
11      ROLLBACK TRAN tran_upd_teacher
12      PRINT '更新教师表失败'
13      RETURN
14    END
15  --提交内层事务
16  COMMIT TRAN tran_upd_teachers
17  ----提交外层事务
18  COMMIT TRAN tran_upd_courses
```

插入正确记录，事务执行成功

消息

(1 行受影响)

(1 行受影响)

实验示例

7.事务保存点。因为回滚操作代价很大，所以保存点提供一种机制，用于回滚部分事务。

```
SQLQuery4.sql - (...BRTDQO\dgn (54))* SQLQuery3.sql - (...BRTDQO\dgn (55))*
3 select * from COURSES
4 where cid='10001'
5
6 BEGIN TRAN tran_upd_courses
7 update courses
8 set hour=46
9 where cid='10001'
10 ----设置事务保存点
11 SAVE TRAN tran_upd_teachers_done
12 insert into teachers
13 values ('1234567890','zs','my@zsu.edu.cn',3000)
14 IF @@ERROR!=0 OR @@ROWCOUNT>1
15 BEGIN
16 ----撤消事务
17 ROLLBACK TRAN tran_upd_teachers_done
18 PRINT '更新教师表信息失败!'
19 RETURN
20 END
21 ---提交事务
22 COMMIT TRAN tran_upd_courses
23 go
24
25 select * from COURSES
26 where cid='10001'
```

改变的行数

课时更新操作依然成功。

可以发现，事务内部虽然插入数据失败，但只回滚到保存点，所以数据更新操作成功。

总结：使用 “SAVE TRAN savePoint_name” 语句创建保存点。然后执行 “ROLLBACK TRANSACTION savePoint_name” 语句以回滚到保存点，而不是回滚到事务的起点。

结果 消息

消息 2627，级别 14，状态 1，第 10 行
违反了 PRIMARY KEY 约束 'PK_TEACHERS'。不能在对象 'dbo.TEACHERS' 中插入重复键。
语句已终止。
更新教师表信息失败！
(1 行受影响)

• 练习

以下练习均在school数据库上进行。

- (1) **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。
- (2) **编写**一个带有保存点的事务。更新teachers表中数据后，设置事务保存点，然后在表courses中插入数据，如果courses插入数据失败，则回滚到事务保存点。演示courses插入失败，但teachers表更新成功的操作。
- (3) **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。