

数据库实验lab12

曹广杰

15352015 数据科学与计算机

授课教师：刘玉葆

Content

数据库实验lab12

Content

实验目的

句法汇总

实验内容

嵌套事务

事务保存点

事务储存

附录

实验目的

1. 熟悉SQL Server的事务控制语言；
2. 能够熟练使用事务控制语言来编写事务处理程序

句法汇总

1. 事务创建格式与整体回滚设置；
2. 回滚到保存点；
3. 设置储存过程与报错信息变量 `@returnString`；
4. 调用储存过程；
5. varchar与char的不同。
6. 设置主键的语法格式。
7. 创建事务与设置回滚选项。
8. 外键的建立条件。

实验内容

三道题的代码详见附录，以下是针对代码的分析。

嵌套事务

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。

首先需要应用事务的格式：

```

1  select 'BEFORE TRANSACTION:' as hint, @@TRANSCOUNT as TRANSACTIONCOUNT
2  set XACT_ABORT ON
3  begin transaction outter
4      -- Update
5      -- Inner transaction
6      select 'INNER TRANSACTION:' as hint, @@TRANSCOUNT as TRANSACTIONCOUNT
7      set XACT_ABORT on
8      begin transaction iner
9          -- Insert
10     commit transaction iner
11     -- Inner transaction done
12 commit transaction outter

```

题意分析：

- 要求

1. 使用嵌套的事务方案；
2. 出现失误则整体回滚；

嵌套的事务方案只需要套用事务实现的模板即可，而对于整体回滚则可以使用语句 `set XACT_ABORT on` 对其进行设置，这也正是在每一个事务开始的时候都使用该语句的原因。

句法：

1. 事务的创建格式；
2. 事务的整体回滚设置；

运行结果详见附录。

事务保存点

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。

```

1  -- Transaction edit begin
2      save transaction edit_tea_done
3      print 'Update teachers done.'
4
5      -- Transaction insert begin
6      -- insert operation
7      if @@ERROR <> 0 or @@ROWCOUNT > 1
8      begin
9          Rollback TRANSACTION edit_tea_done
10         print 'Insert courses fail.'
11         return
12     end
13     -- Transaction insert done
14 -- Transaction edit done

```

题意分析：

1. 添加保存点；
2. 内部事务失误则回滚到保存点；

对于第一个要求，依然是在嵌套事务的整体架构中，添加保存点的操作就是一行语句 `save transaction Tranname`（Tranname表示事务的名字）。

对于第二个要求，则需要保证两点：

1. 回滚到保存点；
2. 不回滚到事务的开始；

为了回滚到保存点，需要使用语句 `Rollback TRANSACTION Tranname`（Tranname 表示事务的名字）。而为了不回滚到事务的开始状态，则需要将 `XACT_ABORT` 的参数设为 `off`。

句法：

- 添加保存点： `save transaction Tranname`
- 回滚到保存点： `Rollback TRANSACTION Tranname`
- 关闭回滚到事务开始的语句： `set XACT_ABORT on`

运行结果详见附录。

事务储存

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。

储存过程的编写如下：

```
1  create procedure INSERTCOURSEINFO
2      -- variable to deal
3      @returnString varchar(100) out -- error info
4  as -- define for procedure
5  begin tran
6      if exists(case) --cond is a case to judge
7      begin
8          select @returnString = 'Course exists.'
9          goto OneError
10     end
11
12     if @@ERROR <> 0 --there is something wrong
13     begin
14         select @returnString = 'Something Wrong'
15         goto OneError
16     end
17
18     select @returnString = 'Insert success'
19     print @returnString
20 commit tran
21
22 OneError: -- error case
23     print @returnString
24     Rollback TRAN
25 go
```

调用储存过程的代码如下：

```

1 declare x --x is variables
2 declare @returnString varchar(100) -- error info
3 -- FORMAT: exec /procedure /values
4 exec INSERTCOURSEINFO '10001', 'english', 90, @returnString out

```

题意分析：

1. 设置储存过程 + 添加条件信息 + 添加报错；
2. 调用过程；

设置储存过程。

只是使用create语句添加一些变量，并使用as语句将内部的过程实现向当前过程信息的封装，故而as前后分为两个部分，至于内部的设计就与过程的设计无关了。

添加条件信息以及报错信息。

使用if语句对条件进行限定，并将输出变量@returnString的值针对不同的错误采取不同的修改，再使用print对其进行输出。

调用储存过程。

1. 使用declare声明变量；
2. 使用exec /procedure /values调用过程信息，对于输出的变量其后需要添加一个属性out；

运行结果详见附录。

附录

Q1:

```

1 select 'BEFORE TRANSACTION:' as hint, @@TRANCOUNT as TRANSACTIONCOUNT
2 set XACT_ABORT ON
3 begin transaction outter
4     update students
5     set sname = "tom"
6     where sid = '1000'
7
8     select 'INNER TRANSACTION:' as hint, @@TRANCOUNT as TRANSACTIONCOUNT
9     set XACT_ABORT on
10    begin transaction inner
11        insert into teachers
12        values('200000000', 'handson', null, null);
13    commit transaction inner
14
15 ROLLBACK transaction outter

```

Q1的查询语句：

```

1  set XACT_ABORT on
2  go
3  begin transaction tmp
4      select top 3 *
5      from students
6
7      select *
8      from teachers
9      where tid = '200000000'
10 commit transaction tmp

```

运行结果如下:

结果				
sid	sname	email	grade	
1	1000	tom	NULL	NULL
2	800001216	gtxrgs	hhce4@qhldj.gov	1992
3	800002933	vnbqzsvv	pvhwd4l@zqur.org	2002

tid	tname	email	salary
1	200000000	handson	NULL

Q2:

```

1  select 'Edit teachers:' as hint, @@TRANCOUNT as TRANSACTIONCOUNT
2  set XACT_ABORT off
3  begin transaction edit_tea
4      update teachers
5      set email = 'handson@magic.club'
6      where tid = '200000000'
7      save transaction edit_tea_done
8      print 'Update teachers done.'
9
10 select 'Insert courses:' as hint, @@TRANCOUNT as TRANSACTIONCOUNT
11 set XACT_ABORT on
12 begin transaction inst_cou
13     insert into courses
14     values('10050', 'Magic', 55);
15
16     if @@ERROR <> 0 or @@ROWCOUNT > 1
17     begin
18         Rollback TRANSACTION edit_tea_done
19         print 'Insert courses fail.'
20         return
21     end

```

```
22      commit transaction inst_cou
23      commit transaction edit_tea
```

Q3:

```
1  create procedure INSERTCOURSEINFO
2      @courseid char(10),
3      @coursename char(30),
4      @hour int,
5      @returnString varchar(100) out
6  as
7  begin tran
8      if exists
9          (select cid
10             from courses
11             where cid = @courseid)
12      begin
13          select @returnString = 'Course exists.'
14          goto OneError
15      end
16
17      insert into courses
18      values(@courseid, @coursename, @hour);
19
20      if @@ERROR <> 0
21      begin
22          select @returnString = 'Something Wrong'
23          goto OneError
24      end
25
26      select @returnString = 'Insert success'
27      print @returnString
28      commit tran
29
30  OneError:
31      print @returnString
32      Rollback TRAN
33  go
```

调用代码:

```
1  declare @courseid char(10)
2  declare @coursename char(30)
3  declare @hour int
4  declare @returnString varchar(100)
5  exec INSERTCOURSEINFO '10001', 'english', 90, @returnString out
```

运行结果如下:

