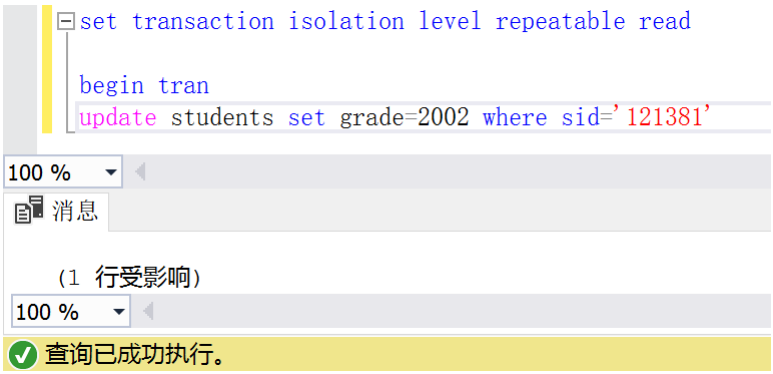
19335262 张航悦 实验 15

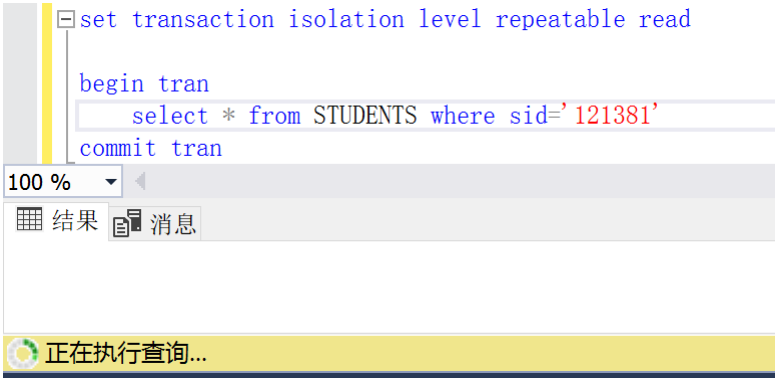
练习一

> 在students表上演示锁争夺，通过sp_who查看阻塞的进程。通过设置lock_timeout解除锁争夺。

- 建立一个连接，更新students表中sid='121381'的学生的grade，但该事务未提交。
- 但是我的sql server中的显示结果与PPT略有不同，不会显示'正在查询'，而是会显示'查询成功'，仍可以成功阻塞其他进程。



- 建立第二个连接，执行查询事务，发现被连接1阻塞。



- 通过sp_who查看阻塞的进程

  可以看到54号进程被阻塞。

```
exec sp_who
```

100 %

结果 消息

	spid	ecid	status	loginame	hostname	blk	dbname	cmd	request_id
1	1	0	background	sa		0	NULL	RESOURCE MONITOR	0
2	2	0	background	sa		0	NULL	XE TIMER	0
3	3	0	background	sa		0	NULL	XE DISPATCHER	0
4	4	0	background	sa		0	NULL	LOG WRITER	0
5	5	0	background	sa		0	NULL	LAZY WRITER	0
6	6	0	background	sa		0	NULL	LOCK MONITOR	0
7	7	0	background	sa		0	master	SIGNAL HANDLER	0
8	8	0	sleeping	sa		0	master	TASK MANAGER	0
9	9	0	background	sa		0	master	TRACE QUEUE TASK	0
10	10	0	sleeping	sa		0	master	TASK MANAGER	0
11	11	0	background	sa		0	master	BRKR TASK	0
12	12	0	background	sa		0	tempdb	CHECKPOINT	0
13	13	0	background	sa		0	master	TASK MANAGER	0
14	14	0	background	sa		0	master	BRKR EVENT HNDLR	0
15	15	0	background	sa		0	master	BRKR TASK	0
16	16	0	background	sa		0	master	BRKR TASK	0
17	17	0	sleeping	sa		0	master	TASK MANAGER	0
18	18	0	sleeping	sa		0	master	TASK MANAGER	0
19	19	0	sleeping	sa		0	master	TASK MANAGER	0
20	20	0	sleeping	sa		0	master	TASK MANAGER	0
21	21	0	sleeping	sa		0	master	TASK MANAGER	0
22	22	0	sleeping	sa		0	master	TASK MANAGER	0
23	23	0	sleeping	sa		0	master	TASK MANAGER	0
24	24	0	sleeping	sa		0	master	TASK MANAGER	0
25	25	0	sleeping	sa		0	master	TASK MANAGER	0
26	30	0	sleeping	sa		0	master	TASK MANAGER	0
27	51	0	sleeping	NT AU...	LAPTO...	0	msdb	AWAITING COMMAND	0
28	52	0	sleeping	NT AU...	LAPTO...	0	msdb	AWAITING COMMAND	0
29	53	0	sleeping	LAPTO...	LAPTO...	0	School	AWAITING COMMAND	0
30	54	0	suspended	LAPTO...	LAPTO...	53	School	SELECT	0
31	55	0	sleeping	LAPTO...	LAPTO...	0	School	AWAITING COMMAND	0
32	56	0	runnable	LAPTO...	LAPTO...	0	School	SELECT	0

- 设置lock_timeout

```
set transaction isolation level repeatable read
set lock_timeout 2000
begin tran
    select * from STUDENTS where sid='121381'
commit tran
```

100 %

结果 消息

```
消息 1222，级别 16，状态 51，第 4 行
已超过了锁请求超时时段。
```

练习二

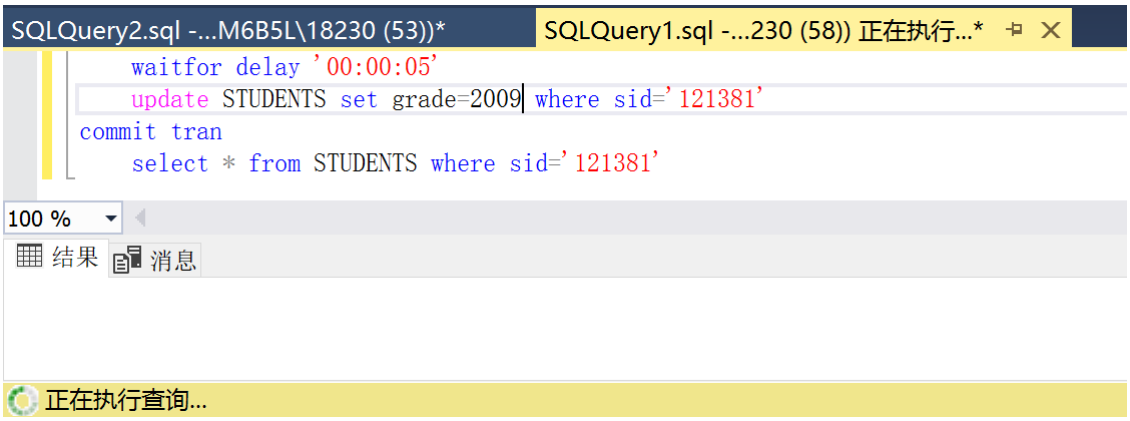
> 在students表上演示死锁。

- 打开两个连接，同时执行下面的代码。可以发现有一个连接可以查询，另一个连接由于死锁，直接停掉了当前程序工作，并回滚之前的事务。

```
set transaction isolation level repeatable read
begin tran
    select * from STUDENTS where sid='121381'
    waitfor delay '00:00:05'
    update STUDENTS set grade=2009 where sid='121381'
commit tran
    select * from STUDENTS where sid='121381'
```

- 实验结果

  我的两个查询一直都是显示'正在执行查询'，未截到死锁报错的信息。



练习三

> 讨论如何避免死锁以及死锁的处理方法。

预防死锁有两种方法。一种方法是通过对加锁请求进行排序或要求同时获得所有的锁发生循环等待。另一种方法比较接近死锁恢复，每当等待有可能导致死锁时，进行事物回滚还不是等待加锁。

当一个检测算法判定存在死锁时，系统必须从死锁中恢复。解除死锁最通常的做法是回滚一个或多个事物。而采取的动作有三个。

1. 选择牺牲者：给定处于死锁状态的事物集，为解除死锁，我们必须决定回滚哪一个事物可以打破死锁。
2. 回滚。一旦决定了要回滚哪个事物，就必须决定该事物回滚多远。可以选择彻底回滚和部分回归这两种方法。
3. 饿死。在系统中，如果选择牺牲者主要基于代价因素，有可能同一事物总是被选为牺牲者。这样一来，该事物总是不能完成其指定任务，这样就会发生饿死。因此必须保证一个事物被选为牺牲者的次数有限。最常用的方法是在代价因素中包含回滚次数。