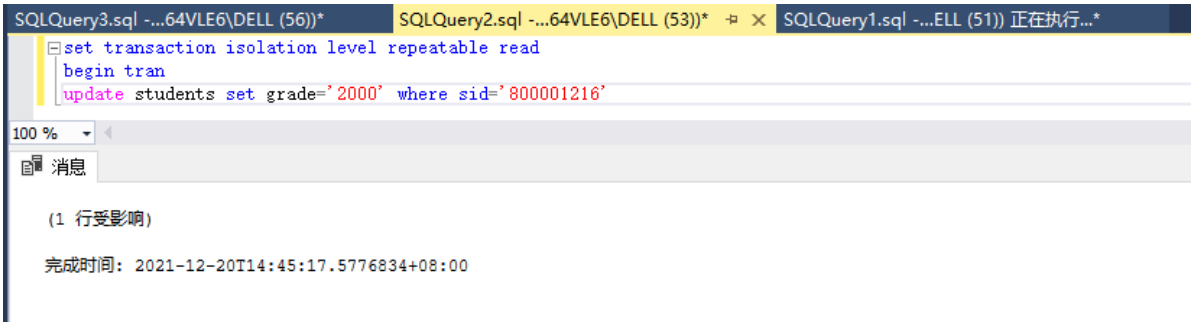
第十五次实验

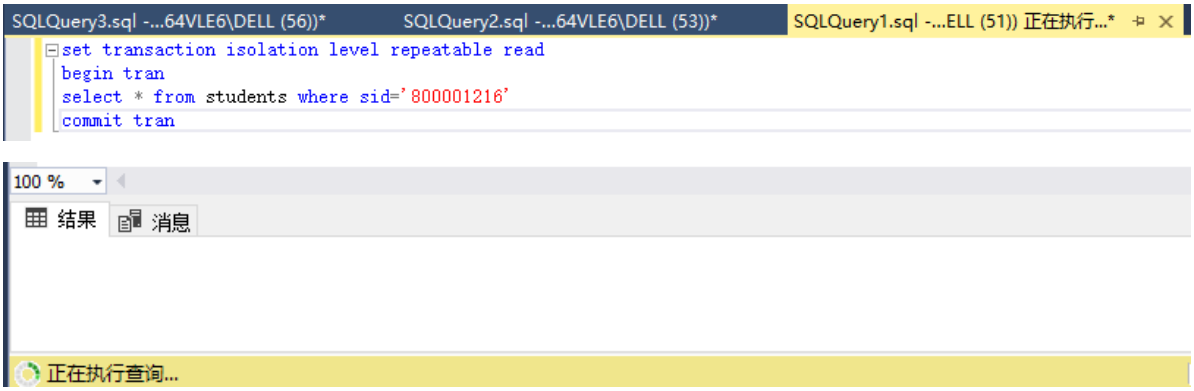
(1)在students表上演示锁争夺，通过sp_who查看阻塞的进程。通过设置lock_timeout解除锁争夺。

1. **通过sp_who查看阻塞的进程**

```
set transaction isolation level repeatable read
begin tran
update students set grade='2000' where sid='800001216'
```




```
set transaction isolation level repeatable read
begin tran
select * from students where sid='800001216'
commit tran
```




```
exec sp_who
```

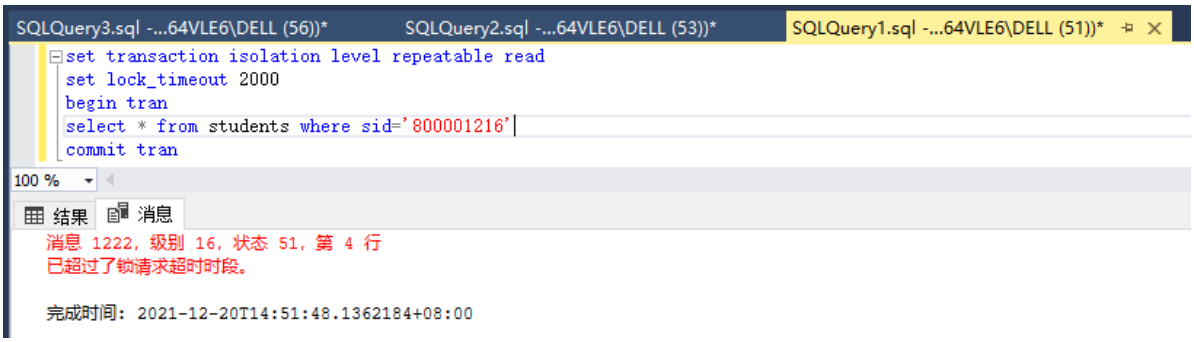
结果 消息

	spid	ecid	status	loginame	hostname	blk	dbname	cmd	request_id
47	47	0	sleeping	sa		0	NULL	TASK MANAGER	0
48	48	0	sleeping	sa		0	master	TASK MANAGER	0
49	49	0	sleeping	sa		0	master	TASK MANAGER	0
50	50	0	sleeping	sa		0	master	TASK MANAGER	0
51	51	0	suspended	DESKT...	DESKT...	53	School	SELECT	0

可以发现进程51被进程53阻塞

2. **设置lock_timeout解除锁争夺**

```
set transaction isolation level repeatable read
set lock_timeout 2000
begin tran
select * from students where sid='800001216'
commit tran
```

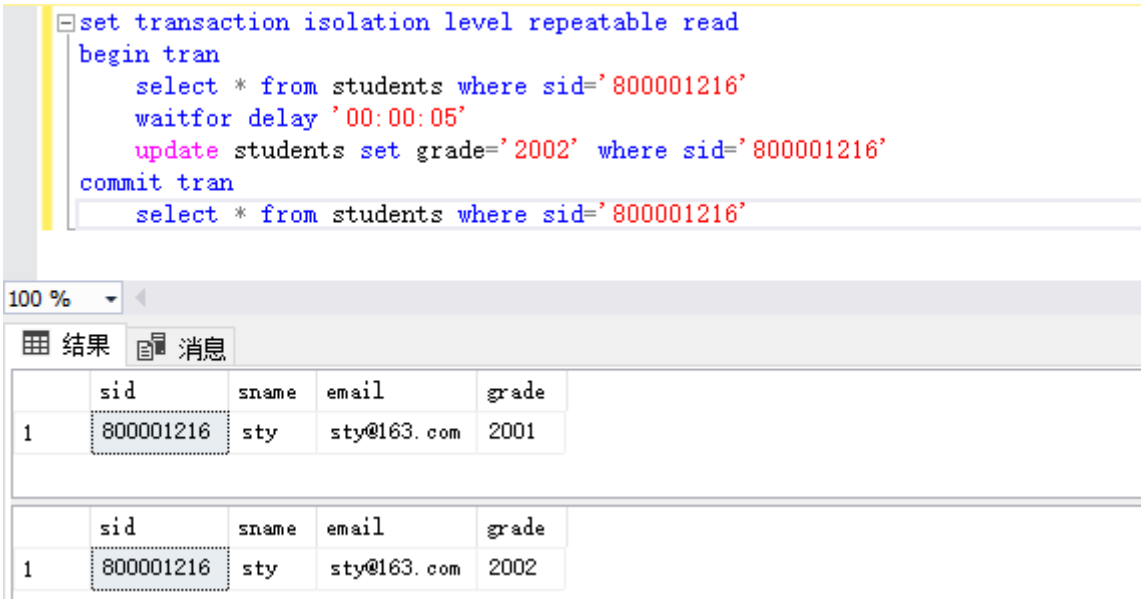


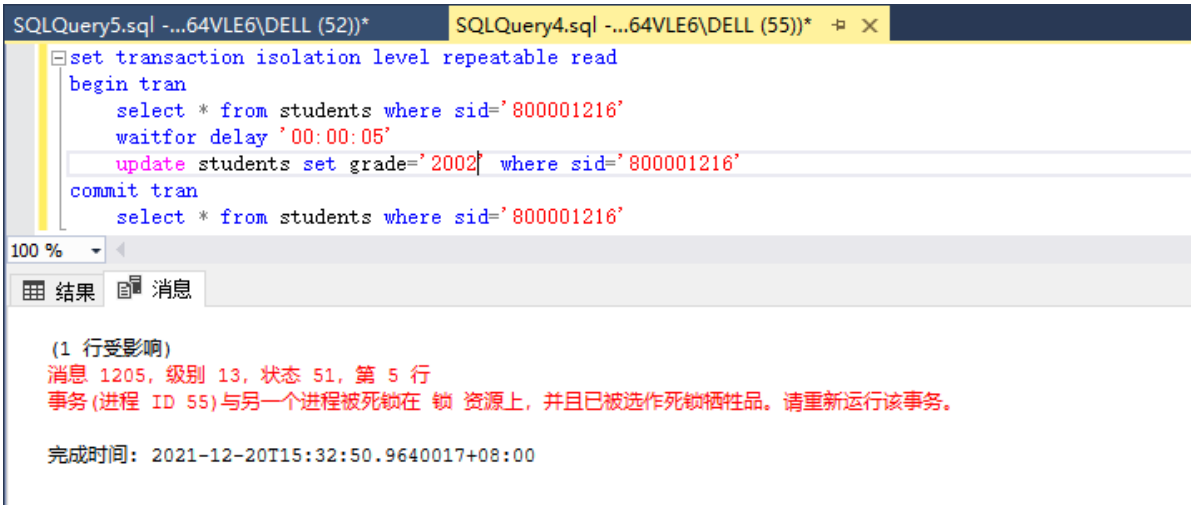
为进程51设置lock_timeout，解除锁的争夺。

(2)在students表上演示死锁。

同时在两个进程执行下面这段代码

```
set transaction isolation level repeatable read
begin tran
    select * from students where sid='800001216'
    waitfor delay '00:00:05'
    update students set grade='2002' where sid='800001216'
commit tran
    select * from students where sid='800001216'
```





(3)讨论如何避免死锁以及死锁的处理方法。

如何避免死锁（死锁预防）

- 同时获得所有锁
  - 每个事务都要在它执行前锁上它所有要用的数据（预声明）
  - 强加偏序条件（基于图的协议）
- 抢占与事务回滚
  - 抢占技术
  - 非抢占技术
- 锁超时，申请锁的事务至多等待一定时间，若此时间内未授予该事务锁，则事务超时回滚重启。

死锁的处理方法

- 死锁预防（上面已提到）
- 死锁检测

- **维护当前将数据项分配给事务的有关信息，以及任何尚未解决的数据项请求信息。**
- **提供一个使用这些信息判断系统是否进入死锁状态的算法。**
- **当检测算法判定存在死锁时，从死锁中恢复。**

- 死锁恢复
  - 选择牺牲者：决定回滚某一最小代价的事务
  - 回滚：一旦确定回滚事务，则需确定该事务回滚多远，包括彻底回滚和部分回滚
  - 饿死：如果选择牺牲者主要基于代价，则有可能同一事务总是被选为牺牲者，那该事务始终不能完成任务，就饿死了