一. 实验目的

- 学会识别锁冲突，学会检查和处理死锁。

二. 实验内容

- 设计实验造成事务对资源的争夺，分析原因，讨论解决锁争夺的办法。
- 设计实验制造事务之间的死锁，分析造成死锁的原因。

三. 实验结果

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。**

   首先我们先查询当前的STUDENTS表

```
SELECT * FROM STUDENTS
```

结果 消息

	sid	sname	email	grade
1	800001216	gfxrgs	hhce4@qhldj.gov	1992
2	800002933	vnbqzsvv	pvhxd4l@zqur.org	2002
3	800005753	waqcj	hlhq0h8@jdba.gov	1992
4	800006682	fiiluommh	ihzd6_k@kzvft.gov	1992
5	800006941	ogvmu	62sfbd@lrt.gov	1995
6	800007595	uxqqbkjn	cr8g@zrvgt.edu	1997

查询已成功执行。 | (local)\CHONOR (10.0 RTM) | DESKTOP-CHONOR\Chonor ... | School | 00:00:00 | 100000 行

   我们选择第一条记录进行修改，我们将第一条记录的的grade设置为2017，设置repeatable read隔离级别，此时在此事务完成前（此事务没有提交所以一直处于未完成）不允许其他事务读取或者修改。

```
SET TRANSACTION ISOLATION LEVEL REPEATABLE READ
BEGIN TRAN
UPDATE STUDENTS SET GRADE='2017' WHERE SID='800001216'
```

   执行后效果

消息

(1 行受影响)

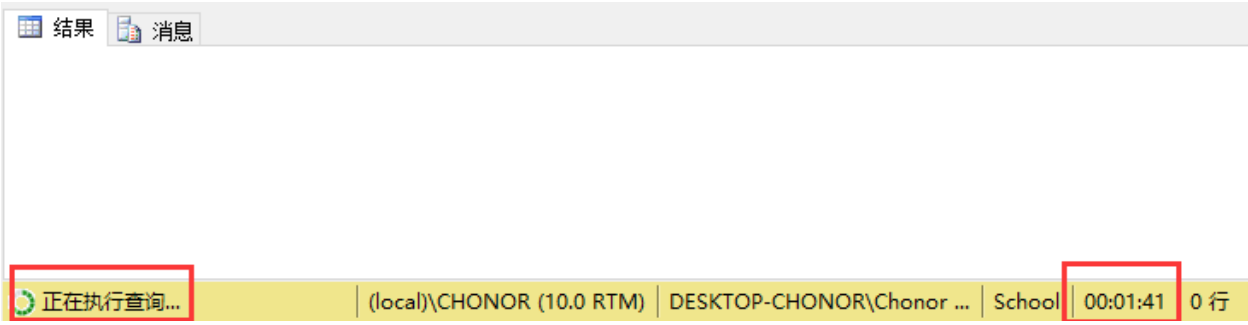
查询已成功执行。 | (local)\CHONOR (10.0 RTM) | DESKTOP-CHONOR\Chonor ... | School | 00:00:00 | 0 行

   建立第二个事务进行查询

```
SET TRANSACTION ISOLATION LEVEL REPEATABLE READ

BEGIN TRAN
    SELECT * FROM STUDENTS WHERE sid='800001216'
COMMIT TRAN
```

此时结果如下

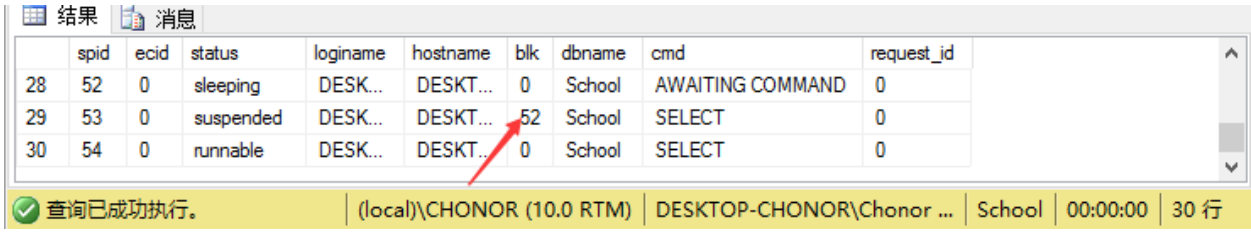


我们可以看到此时的查询被阻塞一直在等待。

使用指令

```
EXEC sp_who
```

查看结果如下



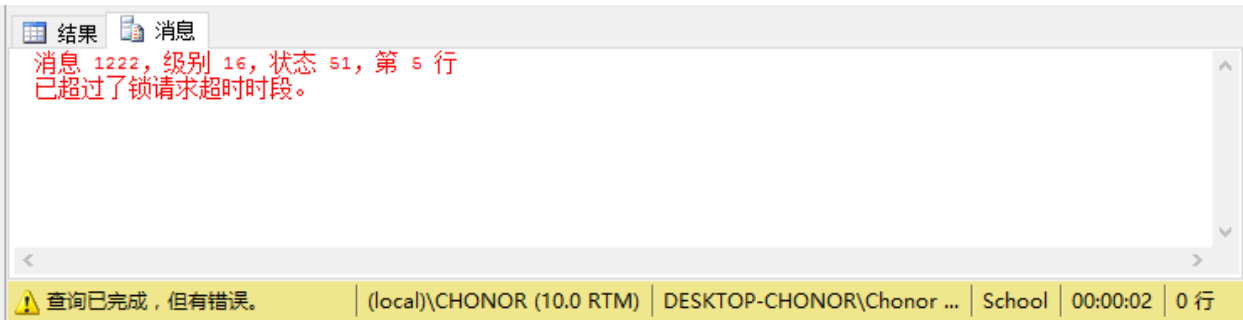
	spid	ecid	status	loginame	hostname	blk	dbname	cmd	request_id
28	52	0	sleeping	DESK...	DESKT...	0	School	AWAITING COMMAND	0
29	53	0	suspended	DESK...	DESKT...	52	School	SELECT	0
30	54	0	runnable	DESK...	DESKT...	0	School	SELECT	0

此时我们看到进程53被进程52阻塞。

为了防止永久等待我们可以设置lock_timeout来设置锁定超时时间间隔

```
SET TRANSACTION ISOLATION LEVEL REPEATABLE READ
SET LOCK_TIMEOUT 2000

BEGIN TRAN
    SELECT * FROM STUDENTS WHERE sid='800001216'
COMMIT TRAN
```

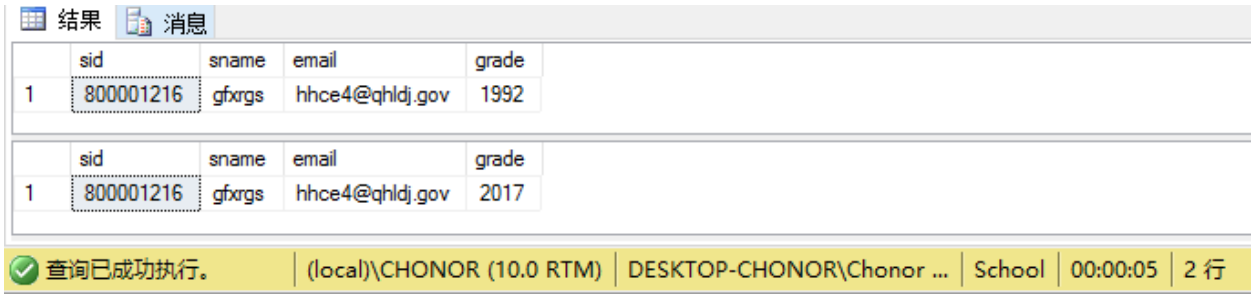


2. **在students表上演示死锁**

使用两个链接同时查询如下代码

```
SET TRANSACTION ISOLATION LEVEL REPEATABLE READ
BEGIN TRAN
    SELECT * FROM STUDENTS WHERE sid='800001216'
    WAITFOR DELAY '00:00:05'
    UPDATE STUDENTS SET grade='2017' WHERE sid='800001216'
COMMIT TRAN
    SELECT * FROM STUDENTS WHERE sid='800001216'
```

此时第一个连接结果如下：



此时第二个连接查询结果如下：

此时出现错误号1205，出现在事务牺牲，通过牺牲事务解决死锁，因为两个连接都通过设置共享锁对同一数据进行查询，并尝试转换为更新锁，进而到排它锁以完成更新操作。但隔离级别为“可重复读”，在事务完成之前，两个连接不可能释放放共享锁而永远无法更新，因而导致死锁。

3. **讨论如何避免死锁以及死锁的处理方法**

死锁的避免

- 尽量使查询对资源的使用顺序保持一致，避免出现循环访问。
- 尽量减少事务执行的时间。
- 保持事务简短并在一个批处理中。
- 避免事务中的用户交互，即锁的竞争。
- 减少资源的占用时间从而减少锁的竞争。
- 在合理的范围内降低隔离级别，减少锁的持有时间。
- 同一个对象尽量采用select 在update 前来使用。
- 对于实时性要求不高的可以使用with(nolock)来实现对表的查询，但是可能会差生脏读。

死锁的处理

- 可以使用然后用kill spid处理掉处于导致死锁的事务
- 使用SET LOCK_TIMEOUT 设定锁请求超时
- SQL Server内部有一个锁监视器线程执行死锁检查，锁监视器对特定线程启动死锁搜索时，会标识线程正在等待的资源；然后查找特定资源的所有者，并递归地继续执行对那些线程的死锁搜索，直到找到一个构成死锁条件的循环。检测到死锁后，数据库引擎 选择运行回滚开销最小的事务的会话作为死锁牺牲品，返回1205 错误，回滚死锁牺牲品的事务并释放该事务持有的所有锁，使其他线程的事务可以请求资源并继续运行。
- 修改产生锁死的语句

四. 实验感想

通过这次实验了解了锁死的原理，实践了怎样的代码会产生死锁，以及在sql server中如何对死锁进行检测，同时对于sql server处理死锁的方式也有了一定的了解，也实践了通过设定锁请求超时来处理死锁。通过上课所学的内容，和实践结合，多于实际中如何辨识死锁，从代码编写层面避免死锁避免死锁的方式有了一定了解