

中山大学数据科学与计算机学院

移动信息工程专业-数据库系统

本科生实验报告

(2017-2018 学年秋季学期)

课程名称：数据库系统实验

教学班级	15M1	专业（方向）	移动互联网
学号	15352408	姓名	张镓伟

一、实验目的

1. 学会识别锁冲突，学会检查和处理死锁。

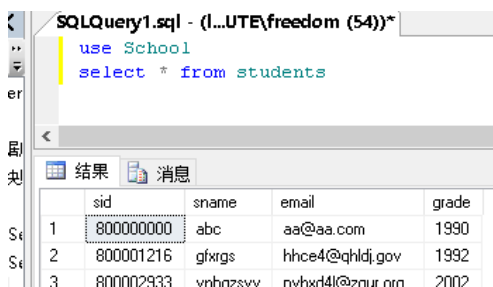
二、实验内容

1. 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。
2. 设计实验制造事务之间的死锁，分析造成死锁的原因。

三、实验过程及结果

(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。

先查询一下 students 表，准备用第一条记录作演示：



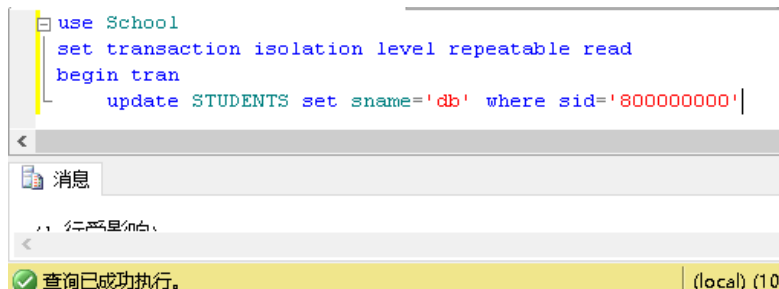
The screenshot shows a SQL query window with the following SQL code:

```
use School
select * from students
```

The results pane displays the following data:

	sid	sname	email	grade
1	800000000	abc	aa@aa.com	1990
2	800001216	gixrgs	hhce4@qhldj.gov	1992
3	800002933	vnhrzsvv	nvhrxrl4l@znur.nm	2002

我们更新 sid 为 800000000 的学省的 grade，但是为了制造锁争夺，更新事务没提交（这个地方没有像 ppt 一样显示正在执行查询，可能是版本问题）：



The screenshot shows a SQL query window with the following SQL code:

```
use School
set transaction isolation level repeatable read
begin tran
update STUDENTS set sname='db' where sid='800000000'
```

The status bar at the bottom indicates: 查询已成功执行。 (local) (10)

新建一个查询事务，查询该 sid 的学生的信息：

```
SQLQuery1.sql - (\\...\\UTE\\freedom (54))* SQLQuery2.sql - (\\...\\om (51)) 正在执行...
set transaction isolation level repeatable read
begin tran
    select * from STUDENTS where sid='800000000'
commit tran
```

正在执行查询... (local) (10.0)

发现查询被阻塞一直在等待，这是因为设置了 repeatable read 隔离级别，该学生的记录被更新事务作了修改，这个事务在完成前会获得并保持一个用于保护该行的排他锁，即不允许其他事务对该行读取或者修改，因此这里我们一直没提交更新事务，那么查询事务就会一直被阻塞，上演锁争夺现象。

验证查询进程是否被阻塞：

```
SQLQuery3.sql - (\\...\\UTE\\freedom (52))* SQLQuery1.sql - (\\...\\UTE\\freedom (54))* SQLQuer
exec sp_who
```

	spid	ecid	status	loginame	hostname	blk	dbname	cmd	request_id
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	51	0	suspended	FREE...	FREED...	54	School	SELECT	0
25	52	0	runnable	FREE...	FREED...	0	School	SELECT	0
26	53	0	sleeping	FREE...	FREED...	0	master	AWAITING COMMAND	0
27	54	0	sleeping	FREE...	FREED...	0	School	AWAITING COMMAND	0
28	55	0	sleeping	FREE...	FREED...	0	Repor...	AWAITING COMMAND	0

由图可以看到 编号为 51 的进程被编号为 54 的进程阻塞，证实了我们前面所述。

通过 lock_timeout 设置锁定超时时间解决永久等待问题，超时后，锁定管理器将自动解除锁的争夺：

```
use School
set transaction isolation level repeatable read
set lock_timeout 2000
begin tran
    select * from students where sid='800000000'
commit tran
```

消息 1222，级别 16，状态 51，第 5 行
已超过了锁请求超时时段。

(2) 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的工作，并回滚之前的事务。



连接 1 查询成功:

```
SQLQuery1.sql - (Microsoft SQL Server)
set transaction isolation level repeatable read
begin tran
select * from students where sid='800000000'
waitfor delay '00:00:05'
update STUDENTS set grade=2000 where sid='800000000'
commit tran
select * from students where sid='800000000'
```

	sid	sname	email	grade
1	800000000	abc	aa@aa.com	1990

	sid	sname	email	grade
1	800000000	abc	aa@aa.com	2000

连接 2 由于死锁报错:

```
SQLQuery1.sql - (Microsoft SQL Server)
set transaction isolation level repeatable read
begin tran
select * from students where sid='800000000'
waitfor delay '00:00:05'
update STUDENTS set grade=2000 where sid='800000000'
commit tran
select * from students where sid='800000000'
```

(1 行受影响)
消息 1205, 级别 13, 状态 51, 第 5 行
事务 (进程 ID 52) 与另一个进程被死锁在 锁 资源上, 并且已被选作死锁牺牲品。请重新运行该事务。

出现死锁的原因: 因为两个连接都通过设置共享锁 (shared lock) 对同一数据进行查询, 并尝试转换为更新锁 (update lock), 进而到排它锁 (exclusive) 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。

避免死锁的方法:

1. 为了避免死锁, 存取资源顺序最好相同。如连接 A 先存取甲数据库对象, 再存取乙数据库对象, 如果连接 B 的存取顺序刚好相反, 则有可能发生死锁。
2. 避免事务中的用户交互, 因为运行没有用户交互的批处理的速度要远远快于用户手动响应查询的速度。
3. 保持事务简短并在一个批处理中, 在同一数据库中并发执行多个需要长时间运行的事务时通常发生死锁。事务运行时间越长, 其持有排它锁或更新锁的时间也就越长, 从而堵塞了其它活动并可能导致死锁。保持事务在一个批处理中, 可以最小化事务的网络通信往返量, 减少完成事务可能的延迟并释放锁。
4. 使用低隔离级别。确定事务是否能在更低的隔离级别上运行。执行提交读允许

事务读取另一个事务已读取（未修改）的数据，而不必等待第一个事务完成。使用较低的隔离级别（例如提交读）而不使用较高的隔离级别（例如可串行读）可以缩短持有共享锁的时间，从而降低了锁定争夺。

5. 使用绑定连接。使用绑定连接使同一应用程序所打开的两个或多个连接可以相互合作。次级连接所获得的任何锁可以象由主连接获得的锁那样持有，反之亦然，因此不会相互阻塞

死锁的处理方法：

1. SQL Server 对付死锁的办法是牺牲掉其中的一个，抛出异常，并且回滚事务
2. 使用 try catch 语句处理死锁异常，如下：

```

set transaction isolation level repeatable read
begin tran
begin try
  select * from students where sid='800000000'
  waitfor delay '00:00:05'
  update STUDENTS set grade=2000 where sid='800000000'
commit tran
end try
begin catch
  SELECT 'There was an error! ' + ERROR_MESSAGE()
  return
end catch
  
```

结果	消息
sid	sname email grade
800000000	abc aa@aa.com 2000

```

set transaction isolation level repeatable read
begin tran
begin try
  select * from students where sid='800000000'
  waitfor delay '00:00:05'
  update STUDENTS set grade=2000 where sid='800000000'
commit tran
end try
begin catch
  SELECT 'There was an error! ' + ERROR_MESSAGE()
  rollback
end catch
  
```

结果	消息
sid	sname email grade
1 800000000	abc aa@aa.com 2000

(无列名)
1 There was an error! 事务(进程 ID 52)与另一个进程被死锁在锁资源上，...

如图，第一个事务成功操作，第二个事务因为死锁触发了异常，然后转到处理 catch 语句中的事件。

四、实验感想

通过这次实验，我认识到了锁冲突和死锁现象，同时也知道了如何检查这两种情况以及处理他们。对于锁冲突，我们可以通过设置锁定超时时间间隔来自动解除锁的争夺，而对于死锁，sql server 默认会牺牲掉一个事务，当然我们也可以使用 try catch 语句来处理。实际中，我们应该尽量避免发生这两种情况。通过调整存取资源顺序等方式去避免。