中山大學
SUN YAT-SEN UNIVERSITY

本科生实验报告

学生姓名：　丁晓琪

学生学号：　22336057

专业名称：　计科

一：实验任务：

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。

2.在students表上演示死锁。

3.讨论如何避免死锁以及死锁的处理方法。

二：实验过程：

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) 建立第一个连接：隔离级别为可重复读级别，更新STUDENTS表sid='1'的grade，事务不提交

```
use School
SET TRANSACTION ISOLATION LEVEL REPEATABLE READ
begin tran
update STUDENTS set grade=7 where sid='1'
```

(2) 建立第二个连接：隔离级别为可重复读级别，查询STUDENTS表

```
use School
SET TRANSACTION ISOLATION LEVEL REPEATABLE READ
begin tran
    select * from STUDENTS where sid='1'
commit tran
```

(3) 执行结果：第一个连接执行成功但是未提交，第二个连接被第一个连接更新操作的排它锁阻塞

- 第一个连接执行成功但是未提交

```
use School
SET TRANSACTION ISOLATION LEVEL REPEATABLE READ
begin tran
update STUDENTS set grade=7 where sid='1'
```

100 %

消息

(1 行受影响)

完成时间: 2024-12-02T17:39:18.0248689+08:00

- 第二个连接的select65号进程被第一个连接的52号进程阻塞

52	52	0	sleeping	丁晓...	丁晓...	0	School	AWAITING COMMAND	0
53	53	0	sleeping	丁晓...	丁晓...	0	master	AWAITING COMMAND	0
54	54	0	sleeping	sa		0	master	TASK MANAGER	0
55	55	0	sleeping	sa		0	master	TASK MANAGER	0
56	56	0	sleeping	sa		0	master	TASK MANAGER	0
57	57	0	sleeping	sa		0	master	TASK MANAGER	0
58	58	0	sleeping	sa		0	master	TASK MANAGER	0
59	59	0	sleeping	sa		0	master	TASK MANAGER	0
60	60	0	sleeping	sa		0	master	TASK MANAGER	0
61	61	0	sleeping	sa		0	master	TASK MANAGER	0
62	65	0	suspended	丁晓...	丁晓...	52	School	SELECT	0

- 第二个连接始终正在执行，无法获得结果

```
use School
SET TRANSACTION ISOLATION LEVEL REPEATABLE READ
begin tran
    select * from STUDENTS where sid='1'
commit tran
```

100 %

结果 消息

(4).第二个连接设置lock_timeout锁定超时时间间隔

```
use School
SET TRANSACTION ISOLATION LEVEL REPEATABLE READ
set lock_timeout 2000
begin tran
    select * from STUDENTS where sid='1'
commit tran
```

100 %

结果 消息

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

完成时间: 2024-12-02T17:48:01.2910861+08:00

2. 在students表上演示死锁。

(1) 同时打开两个连接，执行一样的事务：先查询STUDENTS表sid='1'的表项，延时5s后更新操作，再次查询

```
USE School
SET TRANSACTION ISOLATION LEVEL REPEATABLE READ
BEGIN TRANSACTION
SELECT * FROM STUDENTS WHERE SID='1'
WAITFOR DELAY '00:00:05'
UPDATE STUDENTS SET GRADE=7 WHERE SID='1'
SELECT * FROM STUDENTS WHERE SID='1'
COMMIT TRANSACTION
```

(2) 执行结果：一个连接可以成功查询，一个连接因为死锁牺牲

```
USE School
SET TRANSACTION ISOLATION LEVEL REPEATABLE READ
BEGIN TRANSACTION
SELECT * FROM STUDENTS WHERE SID='1'
WAITFOR DELAY '00:00:05'
UPDATE STUDENTS SET GRADE=7 WHERE SID='1'
SELECT * FROM STUDENTS WHERE SID='1'
COMMIT TRANSACTION
```

100 %

结果 消息

```
(1 行受影响)
消息 1205，级别 13，状态 51，第 6 行
事务(进程 ID 65)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。

完成时间：2024-12-02T17:51:23.7150978+08:00
```

```
USE School
SET TRANSACTION ISOLATION LEVEL REPEATABLE READ
BEGIN TRANSACTION
SELECT * FROM STUDENTS WHERE SID='1'
WAITFOR DELAY '00:00:05'
UPDATE STUDENTS SET GRADE=7 WHERE SID='1'
SELECT * FROM STUDENTS WHERE SID='1'
COMMIT TRANSACTION
```

100 %

结果 消息

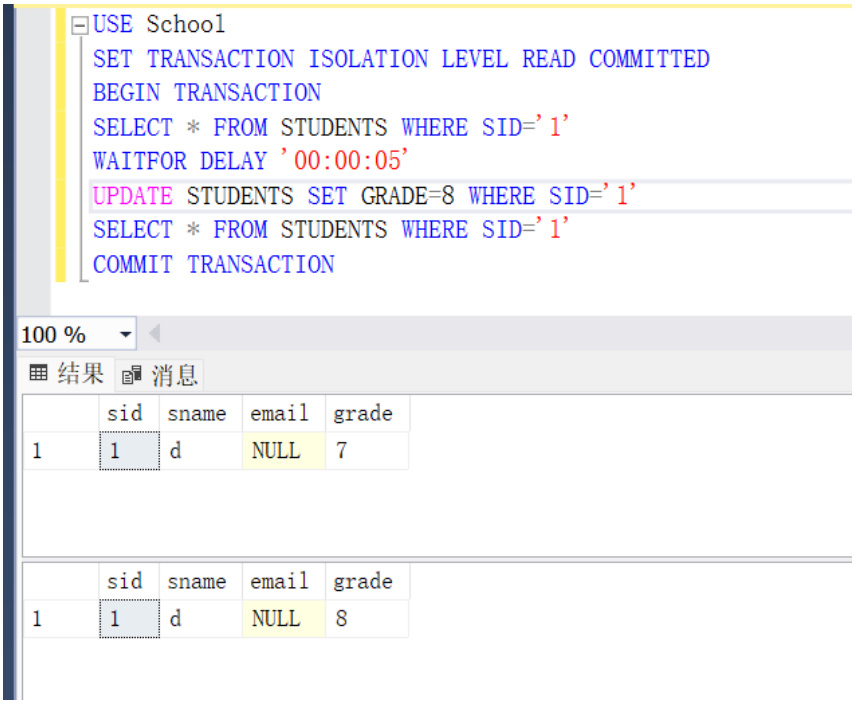
	sid	sname	email	grade
1	1	d	NULL	6

	sid	sname	email	grade
1	1	d	NULL	7

3. 讨论如何避免死锁以及死锁的处理方法。

- 死锁避免：

- 用较低的隔离级别避免使用REPEATABLE READ，将2的隔离级别降为COMMITED 不会发生死锁，但是执行结果可能和需求的不一样



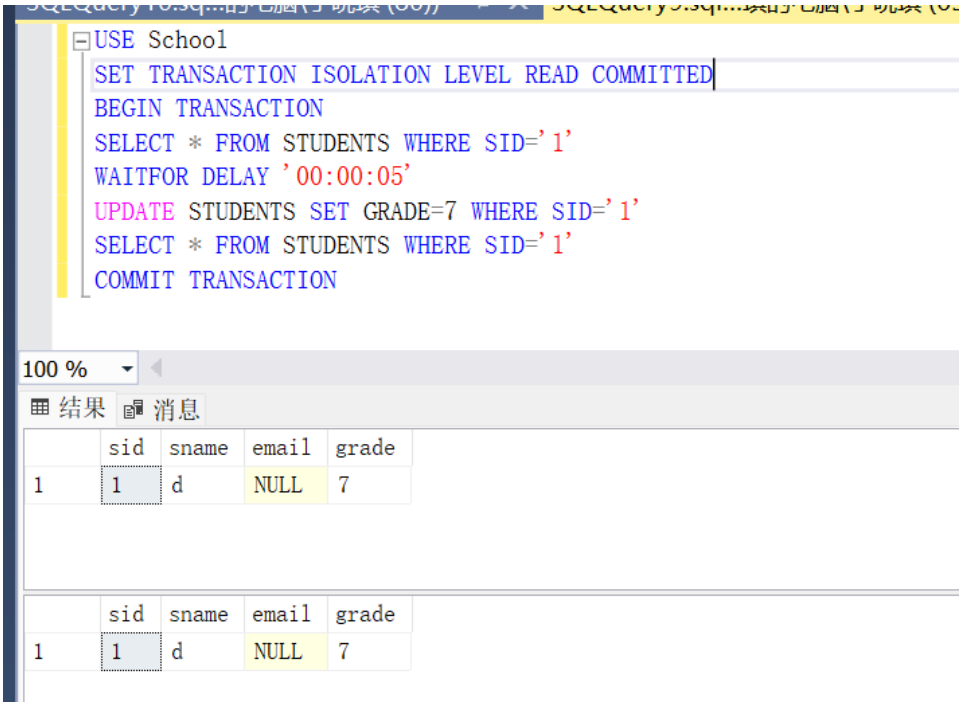
```
USE School
SET TRANSACTION ISOLATION LEVEL READ COMMITTED
BEGIN TRANSACTION
SELECT * FROM STUDENTS WHERE SID='1'
WAITFOR DELAY '00:00:05'
UPDATE STUDENTS SET GRADE=8 WHERE SID='1'
SELECT * FROM STUDENTS WHERE SID='1'
COMMIT TRANSACTION
```

100 %

结果 消息

	sid	sname	email	grade
1	1	d	NULL	7

	sid	sname	email	grade
1	1	d	NULL	8



```
USE School
SET TRANSACTION ISOLATION LEVEL READ COMMITTED
BEGIN TRANSACTION
SELECT * FROM STUDENTS WHERE SID='1'
WAITFOR DELAY '00:00:05'
UPDATE STUDENTS SET GRADE=7 WHERE SID='1'
SELECT * FROM STUDENTS WHERE SID='1'
COMMIT TRANSACTION
```

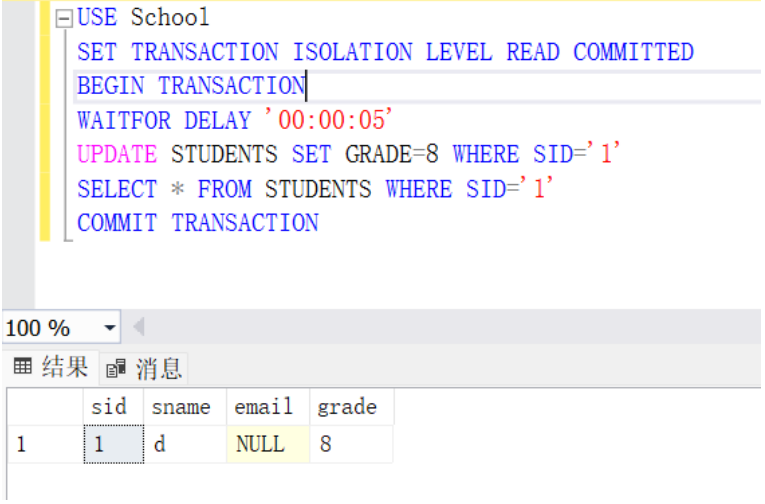
100 %

结果 消息

	sid	sname	email	grade
1	1	d	NULL	7

	sid	sname	email	grade
1	1	d	NULL	7

- 避免不必要的select操作，避免两个事务执行过程中同时持有共享锁，且后续还有更新操作，3中的代码都去掉第一个select后不会出现死锁情况



```
USE School
SET TRANSACTION ISOLATION LEVEL READ COMMITTED
BEGIN TRANSACTION
WAITFOR DELAY '00:00:05'
UPDATE STUDENTS SET GRADE=8 WHERE SID='1'
SELECT * FROM STUDENTS WHERE SID='1'
COMMIT TRANSACTION
```

100 %

结果 消息

	sid	sname	email	grade
1	1	d	NULL	8

```
USE School
SET TRANSACTION ISOLATION LEVEL READ COMMITTED
BEGIN TRANSACTION
WAITFOR DELAY '00:00:05'
UPDATE STUDENTS SET GRADE=7 WHERE SID='1'
SELECT * FROM STUDENTS WHERE SID='1'
COMMIT TRANSACTION
```

100 %

结果 消息

	sid	sname	email	grade
1	1	d	NULL	7

- 死锁处理：超时等待则回滚事务，释放锁，在数据库中提供了这样的自动检测死锁机制，死锁超时后会有事务回滚牺牲

```
USE School
SET TRANSACTION ISOLATION LEVEL REPEATABLE READ
BEGIN TRANSACTION
SELECT * FROM STUDENTS WHERE SID='1'
WAITFOR DELAY '00:00:05'
UPDATE STUDENTS SET GRADE=7 WHERE SID='1'
SELECT * FROM STUDENTS WHERE SID='1'
COMMIT TRANSACTION
```

100 %

结果 消息

```
(1 行受影响)
消息 1205，级别 13，状态 51，第 6 行
事务(进程 ID 65)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。

完成时间：2024-12-02T17:51:23.7150978+08:00
```

三：实验总结：

- repeatable read隔离级别：事务获取用于保护行的排它锁，在事务完成前一直保持
- 死锁场景：

  假设有两个事务，事务A和事务B，它们分别持有对同一数据的共享锁：

  1. **事务A**：
     - 执行 `SELECT * FROM table WHERE condition FOR UPDATE` （或在某些情况下，简单的 `SELECT` 语句在可重复读隔离级别下也会持有共享锁直到事务结束）。
     - 持有共享锁，开始读取数据。
     - 尝试更新这些数据，因此需要转换为更新锁（U锁），然后到排它锁（X锁）。
  2. **事务B**：
     - 同时，事务B也执行类似的查询和更新操作。
     - 也持有共享锁，开始读取相同的数据。
     - 尝试更新这些数据，同样需要转换为更新锁（U锁），然后到排它锁（X锁）。

  由于两个事务都持有共享锁并且都试图获取更新锁（U锁），它们会互相等待对方释放共享锁，从而导致死锁。