

苏州大学实验报告

院、系	计科院	年级专业	软件工程	姓名	高歌	学号	2030416018
课程名称	数据库课程实践					同组实验者	/
指导教师	瞿剑锋	实验日期	2022-04-27	成绩			

实验名称

SQL 语言（试验十四）锁

试验十四 锁

目的：理解锁的概念及锁的作用。

一、利用帮助系统了解 SQL-Server 的下列语句的含义

1 锁的隔离级别

SET TRANSACTION ISOLATION LEVEL Serializable

在数据集上设置范围锁，防止其他用户在事务完成之前更新数据集或将行插入数据集内。

2 设置锁定超时时间

SET LOCK_TIMEOUT 5000

设定语句等待锁释放的毫秒数。执行上面的语句后，若执行语句时等待锁释放超过五秒仍未释放，则会返回错误。

3 SP_LOCK

查询数据库中锁有关的全部信息。常用于诊断死锁问题。

RESULTS								CTRL
	spid	dbid	Objid	IndId	Type	Resource	Mode	Status
1	53	5	0	0	DB	...	S	GRANT
2	54	1	1787153412	0	TAB	...	IS	GRANT
3	54	32767	-571204656	0	TAB	...	Sch-S	GRANT
4	54	5	0	0	DB	...	S	GRANT

二、观察封锁

1 执行语句序列 A

BEGIN TRANsaction

Update Student set sage=sage+1 where sno='0001'

Select * from Student where sno='0002'

RESULTS					
	Sno	Sname	Ssex	Sage	Sdept
1	0002	李文庆	男	23	JSJ

2 在查询分析器中打开第二个连接(连接 school)[文件-连接]，输入下列语句：

1) Select * from Student where sno='0002'

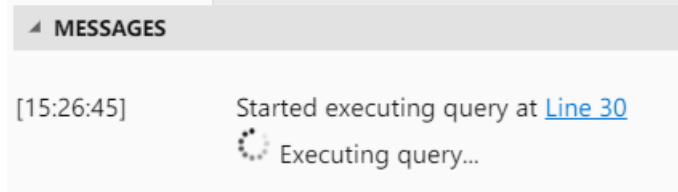
记录执行结果，说明原因。

RESULTS					
	Sno	Sname	Ssex	Sage	Sdept
1	0002	李文庆	男	23	JSJ

和上面的结果没有区别，因为 0002 并未在事务中被锁住。

2) `Select * from Student where sno=' 0001'`

记录执行结果, 说明原因。(如上一步没有停止, 则强行终止)



无法得出结果。这是由于 0001 的信息已在事务中被更改, 并且尚未提交或撤回, 因此被锁住, 无法查询。

3) `update student set sname=' aaa' where sno=' 0002'`

记录执行结果, 说明原因。(如上一步没有停止, 则强行终止)



正常更新。因为这次是在另一个查询中提交的事务, 且 0002 没有在事务中被锁住, 因此直接提交了对 0002 的更新。(当前为隔离级别为 Read Committed, 若隔离级别为更高的 Serializable, 将无法更新)

4) 强行终止上一步的命令, 然后执行语句:

`DBCC opentran`

记录结果



`DBCC OPENTRAN` 返回数据库中最早开始的事务。

思考: 如何知道此事务是那一台计算机发出的?

可根据查询到的 Session ID 定位。例如这里使用上面查询到的 Session ID 55:

1. `SELECT *`
2. `FROM sys.dm_exec_Sessions`

3. WHERE session_id = 55;

RESULTS							CTRL+ALT
	session_id	login_time	host_name	program_name	host_process_id	client_version	client_interfac
1	55	2022-04-27 15:24:20.873	Y7000	Microsoft SQL Server Management Studio - 查询	40396	7	.Net SqlClient

5) 执行:

```
select * from student where sno=' 0001'
```

记录执行结果, 说明原因

MESSAGES

[15:40:56] Started executing query at [Line 50](#)
Executing query...

同 2), 自然还是无法得出结果。

然后回到第一个连接中, 执行语句:

```
COMMIT TRANSACTION
```

观察并记录第二个连接窗口中的现象, 说明原因

RESULTS					
	Sno	Sname	Ssex	Sage	Sdept
1	0001	周志林	男	21	SX

由于事务已提交, 0001 不再被锁住, 查询得出结果。

三、了解锁的类型

1 执行下列语句

```
BEGIN TRAN
```

```
Select * from student where sno=' 0001'
```

```
Print 'server process ID (spid) : '
```

```
Print @@spid
```

RESULTS					
	Sno	Sname	Ssex	Sage	Sdept
1	0001	周志林	男	21	SX

MESSAGES	
[15:50:00]	Started executing query at Line 62 (1 行受到影响) server process ID (spid) : 54 Total execution time: 00:00:00.102

1) 然后执行下列语句

```
exec sp_lock
```

RESULTS								CTRL
	spid	dbid	ObjId	IndId	Type	Resource	Mode	Status
1	53	5	0	0	DB	...	S	GRANT
2	54	1	1787153412	0	TAB	...	IS	GRANT
3	54	32767	-571204656	0	TAB	...	Sch-S	GRANT
4	54	5	0	0	DB	...	S	GRANT

注意根据事务中输出的 spid ,观察结果中相应 spid 的记录, 观察加锁。

2) 然后执行下列语句

```
commit tran
exec sp_lock
```

RESULTS								CTRL
	spid	dbid	ObjId	IndId	Type	Resource	Mode	Status
1	53	5	0	0	DB	...	S	GRANT
2	54	1	1787153412	0	TAB	...	IS	GRANT
3	54	32767	-571204656	0	TAB	...	Sch-S	GRANT
4	54	5	0	0	DB	...	S	GRANT

注意根据事务中输出的 spid ,观察结果中相应 spid 的记录, 观察加锁。

2 执行下列语句

```
BEGIN TRAN
```

```
Update student set sage=sage + 1 where sno=' 1001'
```

```
Print 'server process ID (spid) : '
```

```
Print @@spid
```

MESSAGES	
[15:54:51]	Started executing query at Line 81 (0 行受到影响) server process ID (spid) : 54 Total execution time: 00:00:00.043

1) 然后执行下列语句

```
exec sp_lock
```

RESULTS								CTRL
	spid	dbid	ObjId	IndId	Type	Resource	Mode	Status
1	53	5	0	0	DB	...	S	GRANT
2	54	5	581577110	1	PAG	1:240	IX	GRANT
3	54	5	581577110	0	TAB	...	IX	GRANT
4	54	1	1787153412	0	TAB	...	IS	GRANT
5	54	32767	-571204656	0	TAB	...	Sch-S	GRANT
6	54	5	0	0	DB	...	S	GRANT

注意根据事务中输出的 spid ,观察结果中相应 spid 的记录, 观察加锁。

2) 然后执行下列语句

```
commit tran
exec sp_lock
```

RESULTS								CTRL
	spid	dbid	ObjId	IndId	Type	Resource	Mode	Status
1	53	5	0	0	DB	...	S	GRANT
2	54	1	1787153412	0	TAB	...	IS	GRANT
3	54	32767	-571204656	0	TAB	...	Sch-S	GRANT
4	54	5	0	0	DB	...	S	GRANT

注意根据事务中输出的 spid ,观察结果中相应 spid 的记录, 观察加锁。

3 使用 SET TRANSACTION ISOLATION LEVEL Serializable

然后重新执行三和四步, 观察与原来有何不同。

执行“二、观察封锁”的事务前加上“SET TRANSACTION ISOLATION LEVEL Serializable; GO;”, 重复“二、观察封锁”中所有任务, 观察到第三步中无法更新 0002 的信息。

4 了解表级锁 (查看帮助文件)

```
BEGIN TRAN
```

```
Select * from student (TABLOCKX) where sno=' 1002'
```

```
Print 'Server Process ID (spid): '
```

```
Print @@spid
```

RESULTS						CTRL
	Sno	Sname	Ssex	Sage	Sdept	
MESSAGES						
[16:19:54] Started executing query at Line 90						
(0 行受到影响)						
Server Process ID (spid):						
54						
Total execution time: 00:00:00.055						

然后执行:

```
exec sp_lock
```

RESULTS								CTRL
	spid	dbid	ObjId	IndId	Type	Resource	Mode	Status
1	53	5	0	0	DB	...	S	GRANT
2	54	5	581577110	0	TAB	...	X	GRANT
3	54	1	1787153412	0	TAB	...	IS	GRANT
4	54	32767	-571204656	0	TAB	...	Sch-S	GRANT
5	54	5	0	0	DB	...	S	GRANT

注意根据事务中输出的 spid ,观察结果中相应 spid 的记录, 观察加锁类型。

5 了解锁定超时

a) 执行下列语句, 设置锁定超时为 1000 ms

```
set lock_timeout 1000
```

```
go
```

```
BEGIN TRAN
```

```
Select * from student (TABLOCKX) where sno=' 1002'
```

RESULTS						CTRL
	Sno	Sname	Ssex	Sage	Sdept	

MESSAGES

[16:23:49] Started executing query at [Line 104](#)
(0 行受到影响)
Total execution time: 00:00:00.004


2) 打开第二个连接

执行:

```
select * from student
```

记录观察到的现象。

MESSAGES

[16:27:57] Started executing query at [Line 111](#)
 Executing query...

仍无法得到查询结果，并且刚才的超时设置并未生效。

3) 在打开的第二个连接中

```
set lock_timeout 10000
```

```
go
```

```
select * from student
```

记录观察到的现象。

MESSAGES

[16:31:34] Started executing query at [Line 117](#)
命令已成功完成。
[16:31:34] Started executing query at [Line 119](#)
Msg 1222, Level 16, State 56, Line 3
已超过了锁请求超时时段。
Total execution time: 00:00:10.012

十秒后查询超时，返回错误。因此可知 SET LOCK_TIMEOUT 仅在当前查询中生效，并不全局生效。

四 编程实例

1 编写存储过程 usp_update1，传入参数为课程号,处理逻辑:

对传入的这门课,进行如下处理:

如某学生该门课成绩>80，则加 2 分

如某学生该门课成绩>60，则加 1 分

如某学生该门课成绩<=60，扣 1 分

要求: 在存储过程中,要么全部学生的成绩被处理成功,要么全部不处理

```
1. CREATE PROCEDURE usp_update1  
2. (@Cno CHAR(10))  
3. AS
```

```

4. BEGIN
5. BEGIN TRANSACTION
6. UPDATE SC
7. SET Grade = Grade + 2
8. WHERE Cno = @Cno
9. AND Grade > 80;
10. UPDATE SC
11. SET Grade = Grade + 1
12. WHERE Cno = @Cno
13. AND Grade > 60;
14. UPDATE SC
15. SET Grade = Grade - 1
16. WHERE Cno = @Cno
17. AND Grade <= 60;
18. COMMIT TRANSACTION
19. END

```

思考：在调试中，采用那些措施，使存储过程运行时执行回滚操作（rollback）。

当任意需要被修改的记录被锁住时，存储过程都将进行回滚操作。例如启动事务修改 SC 中的某一条记录而不提交，此时启动另一个查询执行上述存储过程。

2 编写触发器，

对 insert、update 语句进行监控，当学生的年龄超过 40 岁时，把该学生的系科改为 ‘BAK’,同时删除该学生的所有选课记录。（注意，利用事务，使修改系科及删除选课记录要么全做，要么全不做）

```

1. CREATE TRIGGER ust_updateStudent
2. ON Student
3. FOR INSERT, UPDATE
4. AS
5. BEGIN
6. IF ((SELECT Sage FROM inserted) > 40)
7. BEGIN
8. BEGIN TRANSACTION
9. UPDATE Student
10. SET Sdept = 'BAK'
11. WHERE Sno = (SELECT Sno FROM inserted);
12. DELETE SC
13. WHERE Sno = (SELECT Sno FROM inserted);
14. COMMIT TRANSACTION
15. END
16. END

```