《数据库系统实验》

实验报告

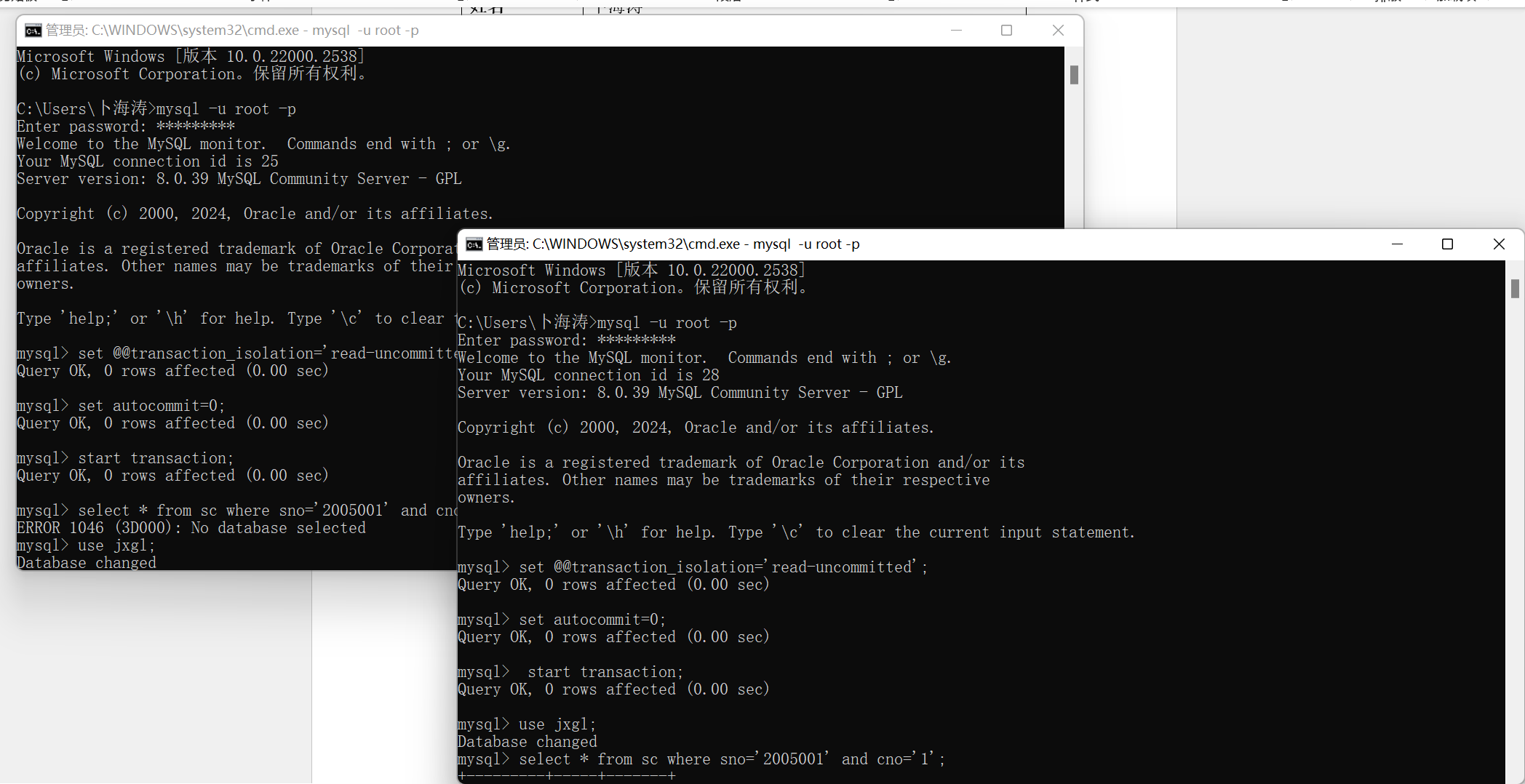
|  |  |
| --- | --- |
| **题目** | 实验十二 |
| **姓名** | 卜海涛 |
| **学号** | 22336016 |
| **班级** | 22级计算机科学与技术（人工智能与大数据） |

1. 实验环境：

操作系统：Windows64位

图形界面：MySQL Workbench

1. 实验内容与完成情况：
2. 同时打开两个cmd窗口连接mysql数据库，每个窗口执行一个事务



1. 复现表12-5

执行事务1

代码：

set @@transaction\_isolation='read-uncommitted';

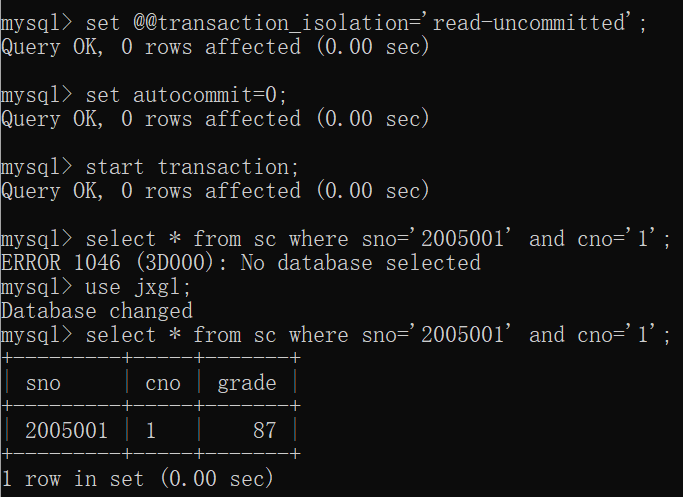
set autocommit=0;

start transaction;

use jxgl;

select \* from sc where sno='2005001' and cno='1';

结果：



执行事务2

代码：

set @@transaction\_isolation='read-uncommitted';

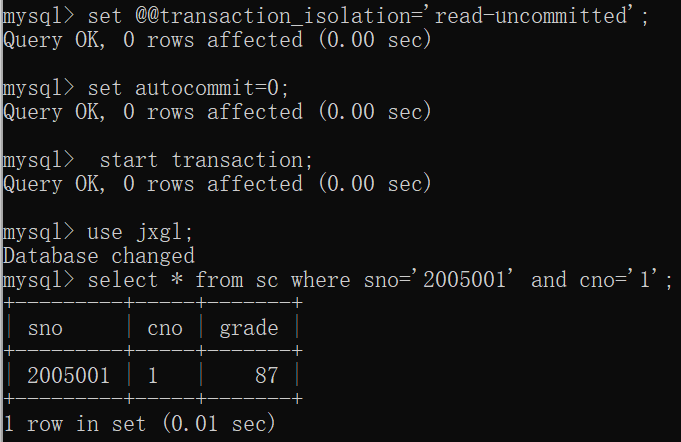
set autocommit=0;

start transaction;

use jxgl;

select \* from sc where sno='2005001' and cno='1';

结果：



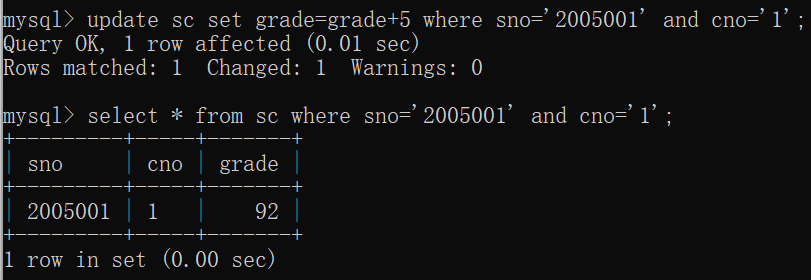
执行事务1

代码：

update sc set grade=grade+5 where sno='2005001' and cno='1';

select \* from sc where sno='2005001' and cno='1';

结果：



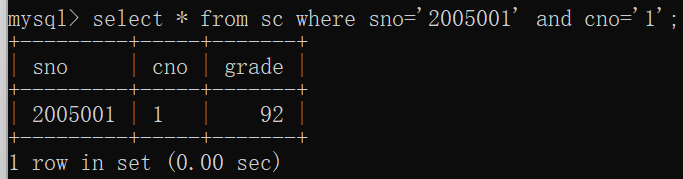
此时sc表在事务1中被更新，但是还未提交到数据库

执行事务2

代码：

select \* from sc where sno='2005001' and cno='1';

结果：



事务2读取了事务1还未提交的更新结果

由于两个事务采用的隔离级别为read-uncommitted（未提交读），当一个事务的更新结果还

未提交时，另一个事务仍然能够从缓存中读到其更新的结果，从而产生了脏读，这违反了

数据库的隔离性

执行事务1

代码：

rollback;

结果：



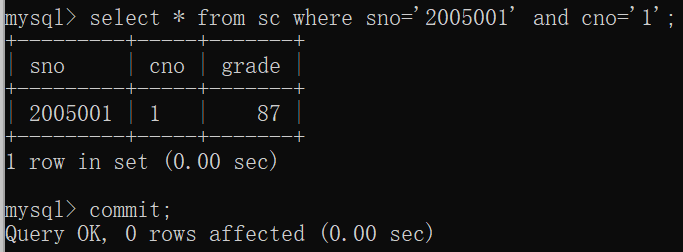
执行事务2

代码：

select \* from sc where sno='2005001' and cno='1';

commit;

结果：



同理，由于事务1的隔离级别为read-uncommited，因此事务2能读到其rollback后的数据

1. 复现表12-6

执行事务1

代码：

set @@transaction\_isolation='read-uncommitted';

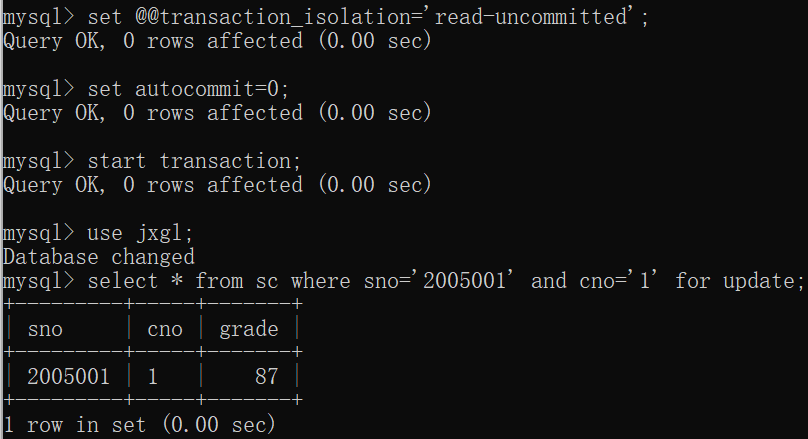
set autocommit=0;

start transaction;

use jxgl;

select \* from sc where sno='2005001' and cno='1' for update;

结果：



事务1通过for update设置了排他锁，上锁之后事务2既不可读也不可写，需要等待事务

1释放锁才能继续执行

执行事务2

代码：

set @@transaction\_isolation='read-uncommitted';

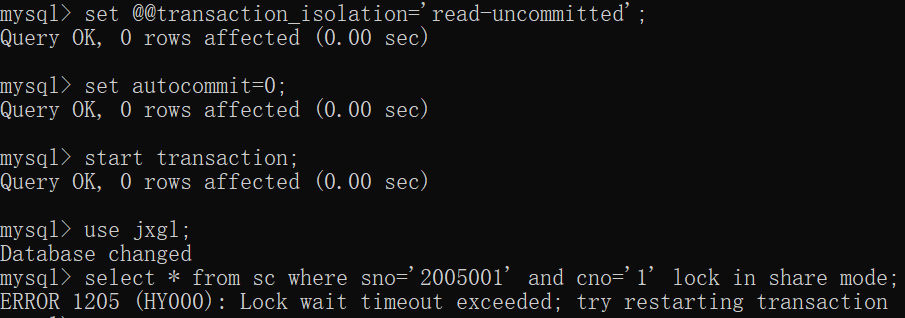
set autocommit=0;

start transaction;

use jxgl;

select \* from sc where sno='2005001' and cno='1' lock in share mode;

结果：



由于事务1使用了排他锁，事务2需等待事务1释放锁后才能执行查询操作，而事务2在

查询时使用了共享锁，即在查询过程中不允许其他事务的写操作，共享锁在等待一段时间

后未获取结果就报错

执行事务1

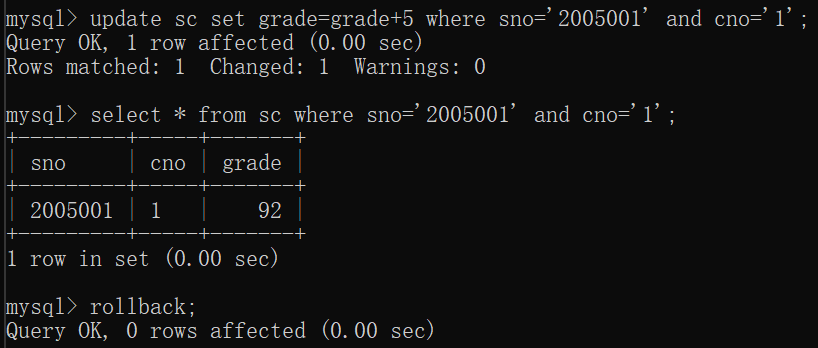
代码：

update sc set grade=grade+5 where sno='2005001' and cno='1';

select \* from sc where sno='2005001' and cno='1';

rollback;

结果：



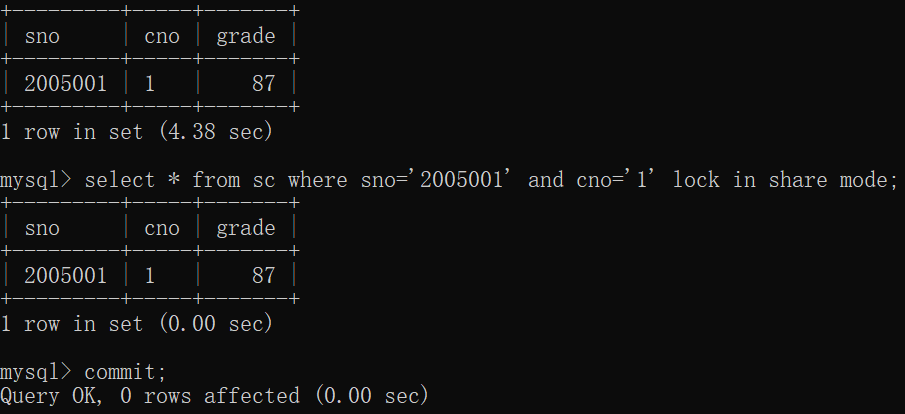
执行事务2

代码：

select \* from sc where sno='2005001' and cno='1' lock in share mode;

commit;

结果：



事务1回滚之后排他锁才被释放，事务2获得锁后继续执行未完成的查询操作，这样事务

2就不会读到事务1修改但还未提交的数据，避免了脏读的发生