

# 《数据库系统原理》实验报告

实验名称     实验七 数据库的事务创建与运行

班     级     2015211307

学     号     2017526019

姓     名     刘禾子

## 实验七 数据库的事务创建与运行实验

### ■ 实验目的

通过实验，了解 MySQL 数据库系统中各类数据库事务的定义机制和基于锁的并发控制机制，掌握 MySQL 数据库系统的事务控制机制。

### ■ 实验平台及环境

Windows10、MySQL 5.7、MySQL Workbench 6.3CE

### ■ 实验内容

1. 定义三种模式的数据库事务
2. 查看事务的隔离级别

### ■ 实验步骤及结果分析

#### 1. 定义三种模式的数据库事务

事务是由相关操作构成的一个完整的操作单元。两次连续成功的COMMIT或ROLLBACK之间的操作，称为一个事务。

对数据库所做的一系列修改，在修改过程中，暂时不写入数据库，而是缓存起来，用户在自己的终端可以预览变化，直到全部修改完成，并经过检查确认无误后，一次性提交并写入数据库，在提交之前，必要的话所做的修改都可以取消。提交之后，就不能撤销，提交成功就其他用户才可以通过查询浏览数据的变化。

#### 事务的特点

ACID：原子性（atomicity）、一致性（consistency）、隔离性（isolation）、持久性（durability）。一个有效的事务处理系统必须满足相关标准。

✚ 原子性：一个事务必须被视为一个单独的内部“不可分”的工作单元，以确保整个事务要么全部执行要么全部回滚

✚ 一致性：数据库总是从一种一致性状态转换到另一种一致性状态。

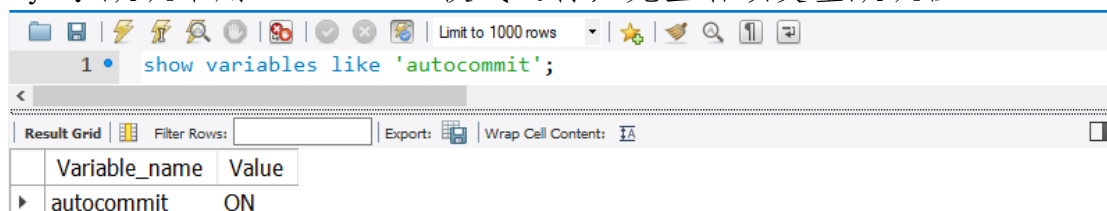
✚ 隔离性：某个事务的结果只有在完成之后才对其他事务可见。在上述例子中，当数据库执行完insert语句，还未执行delete语句时，如果此时另一个客户端对数据库的访问也同时运行，它将仍视符合条件的记录在b表中。

✚ 持久性：一旦一个事务提交事务所做的数据改变是永久的

## (1) 显示事务

显式事务，由用户指定，允许用户决定哪批工作必须成功完成，否则所有的部分都不完成。操作包括start transaction,rollback,commit。

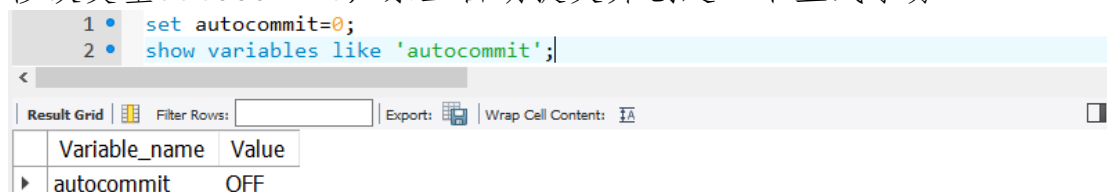
MySQL默认采用autocommit模式运行，先查看该变量默认值



```
1 • show variables like 'autocommit';
```

Variable_name	Value
autocommit	ON

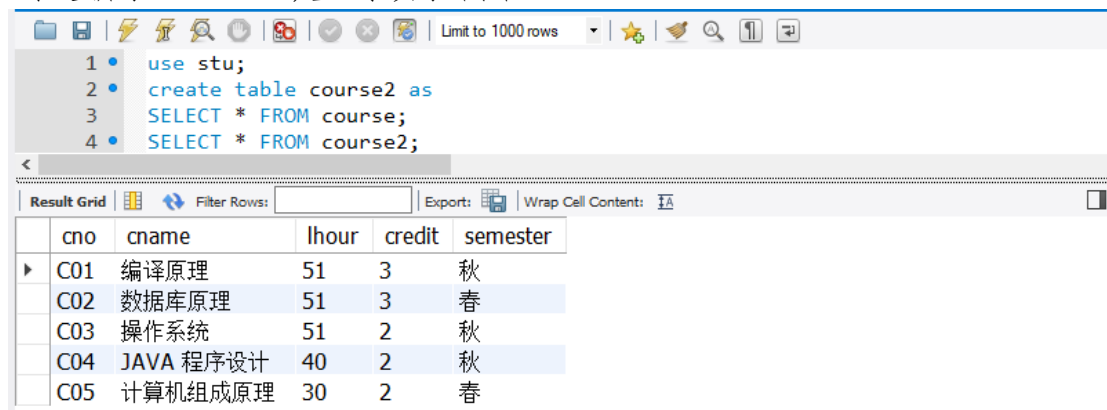
修改变量autocommit，禁止自动提交并创建一个显式事务



```
1 • set autocommit=0;
2 • show variables like 'autocommit';
```

Variable_name	Value
autocommit	OFF

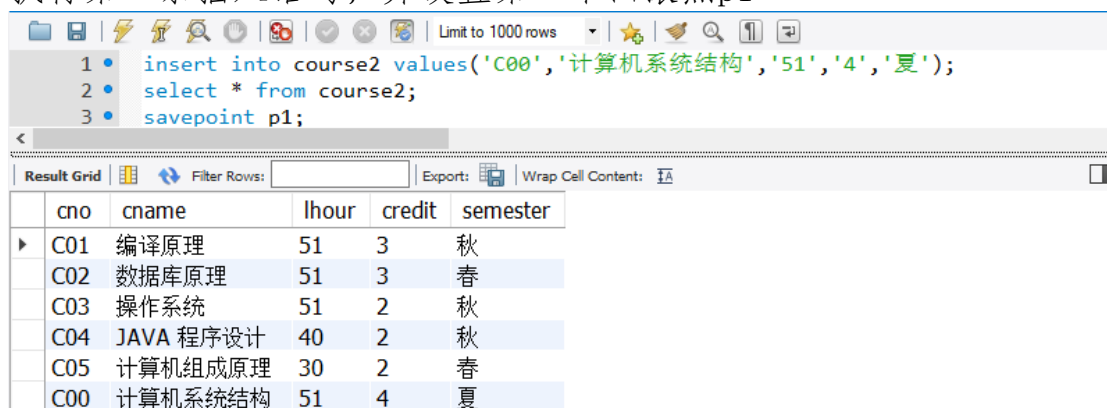
创建新表course2，查询该表内容



```
1 • use stu;
2 • create table course2 as
3 • SELECT * FROM course;
4 • SELECT * FROM course2;
```

cno	cname	lhour	credit	semester
C01	编译原理	51	3	秋
C02	数据库原理	51	3	春
C03	操作系统	51	2	秋
C04	JAVA 程序设计	40	2	秋
C05	计算机组成原理	30	2	春

执行第一条插入语句，并设置第一个回滚点p1



```
1 • insert into course2 values('C00','计算机系统结构','51','4','夏');
2 • select * from course2;
3 • savepoint p1;
```

cno	cname	lhour	credit	semester
C01	编译原理	51	3	秋
C02	数据库原理	51	3	春
C03	操作系统	51	2	秋
C04	JAVA 程序设计	40	2	秋
C05	计算机组成原理	30	2	春
C00	计算机系统结构	51	4	夏

执行第二条插入语句，并设置第二个回滚点p2

1	•	insert into course2 values('C10','软件工程','51','4','夏');
2	•	select * from course2;
3	•	savepoint p2;

	cno	cname	lhour	credit	semester
▶	C01	编译原理	51	3	秋
	C02	数据库原理	51	3	春
	C03	操作系统	51	2	秋
	C04	JAVA 程序设计	40	2	秋
	C05	计算机组成原理	30	2	春
	C00	计算机系统结构	51	4	夏
	C10	软件工程	51	4	夏

回滚到p1,如下图所示p2自动被丢弃，回到只插入C00课程的状态：

1	•	rollback to p1;
2	•	select * from course2;

	cno	cname	lhour	credit	semester
▶	C01	编译原理	51	3	秋
	C02	数据库原理	51	3	春
	C03	操作系统	51	2	秋
	C04	JAVA 程序设计	40	2	秋
	C05	计算机组成原理	30	2	春
	C00	计算机系统结构	51	4	夏

回滚到原始点，即事务开始的点。如下图所示所有操作都被回滚。显示事务执行成功

1	•	rollback;
2	•	select * from course2;

	cno	cname	lhour	credit	semester
▶	C01	编译原理	51	3	秋
	C02	数据库原理	51	3	春
	C03	操作系统	51	2	秋
	C04	JAVA 程序设计	40	2	秋
	C05	计算机组成原理	30	2	春

分析可知，显式事务即事务没有自动提交，可以回滚到原始点，在rollback和commit之前对数据库的修改都可以挽回，而不是永久写入。

## (2) 自动提交事务

MySQL默认采用autocommit模式进行，当执行一个用于更新表的语句后，MySQL立刻把更新存储到磁盘中。默认级别为不可重复读。

将变量autocommit的值重新设置为1后，重复上述（1）的操作。

1	•	set autocommit=1;
2	•	savepoint p3;
3	•	insert into course2 values('C06','体系结构','51','4','夏');
4	•	select * from course2;
5	•	savepoint p4;

cno	cname	lhour	credit	semester
C01	编译原理	51	3	秋
C02	数据库原理	51	3	春
C03	操作系统	51	2	秋
C04	JAVA 程序设计	40	2	秋
C05	计算机组成原理	30	2	春
C06	体系结构	51	4	夏

执行rollback to p3报错，即使显示savepoint p3成功

17 10:15:30 savepoint p3 0 row(s) affected 0.000 sec

但是实际上语句都已经自动提交了且永久写入所以事务无法回滚

1 • rollback to p3;  
2 • select \* from course2;

21 10:18:07 rollback to p3 Error Code: 1305. SAVEPOINT p3 does not exist 0.016 sec

### (3) 隐式事务

虽然设置 autocommit=1，但是事务中如果有 create table,alter function,drop index 等语句时，则隐含地结束一个事务，相当于在执行语句前进行了 commit。

在（2）中设置完自动提交后，首先查看所有的表

1	•	show tables;
---	---	--------------

Tables_in_stu
course
course2
n_student
sc
student

建立一个新表 test,再查看所有的表

1	•	create table test(id int);
1	•	show tables;

Tables_in_stu
course
course2
n_student
sc
student
test

执行 rollback 语句再次查看所有的表报错，回滚失败，由于 create table 隐含了一个事务的结束。即使 rollback 语句执行成功，但是隐式事务无法挽回

1 • rollback;  
2 • show tables;

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Tables_in_stu	
course	
course2	
n_student	
sc	
student	
test	

Result 9 x

Output

Action Output

#	Time	Action	Message	Duration / Fetc
21	10:18:07	rollback to p3	Error Code: 1305. SAVEPOINT p3 does no...	0.016 sec
22	10:21:17	show tables	5 row(s) returned	0.000 sec / 0.0
23	10:23:56	create table test(id int)	0 row(s) affected	0.141 sec
24	10:24:31	show tables	6 row(s) returned	0.000 sec / 0.0
25	10:28:22	rollback	0 row(s) affected	0.000 sec
26	10:28:22	show tables	6 row(s) returned	0.000 sec / 0.0

## 2. 查看事务的隔离级别

### (1) 事务的锁信息

查看系统上表锁定争夺：

1 • show status like 'table%';

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Variable_name	Value
Table_locks_immediate	195
Table_locks_waited	0
Table_open_cache_hits	186
Table_open_cache_misses	4
Table_open_cache_overflows	0

查看系统上的行锁的争夺情况：

1 • show status like 'innodb\_row\_lock%';

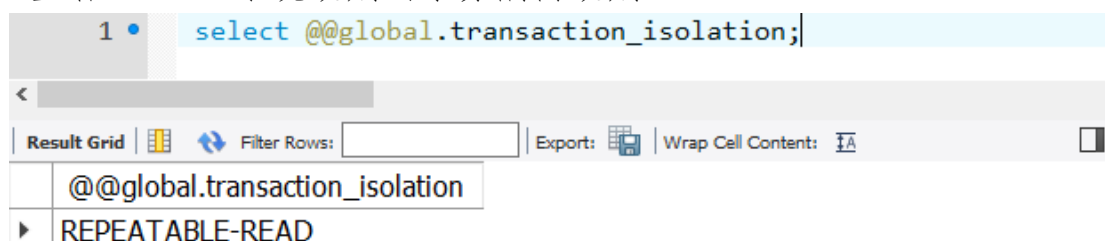
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Variable_name	Value
Innodb_row_lock_current_waits	0
Innodb_row_lock_time	0
Innodb_row_lock_time_avg	0
Innodb_row_lock_time_max	0
Innodb_row_lock_waits	0

## (2) 事务的隔离级别

查看 innodb 系统级别的事务隔离级别：

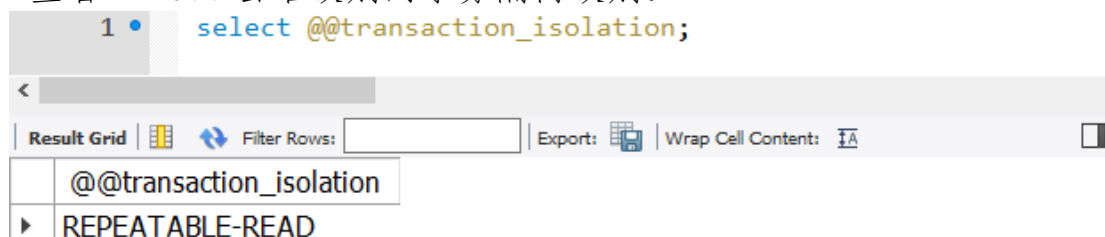
```
1 • select @@global.transaction_isolation;
```



@@global.transaction_isolation
REPEATABLE-READ

查看 innodb 会话级别的事务隔离级别：

```
1 • select @@transaction_isolation;
```



@@transaction_isolation
REPEATABLE-READ

SQL 标准定义了 4 类隔离级别。低级别的隔离级一般支持更高的并发处理，并拥有更低的系统开销。

- ✚ **Read Uncommitted**(读取未提交内容)：在该隔离级别，所有事务都可以看到其他未提交事务的执行结果。本隔离级别很少用于实际应用，因为它的性能也不比其他级别好多少。读取未提交的数据，也被称之为脏读 (Dirty Read)。
- ✚ **Read Committed**(读取提交内容)：这是大多数数据库系统的默认隔离级别（但不是 MySQL 默认的）。它满足了隔离的简单定义：一个事务只能看见已经提交事务所做的改变。这种隔离级别也支持所谓的不可重复读 (Nonrepeatable Read)，因为同一事务的其他实例在该实例处理期间可能会有新的 commit，所以同一 select 可能返回不同结果。
- ✚ **Repeatable Read**(可重读)：MySQL 默认隔离级别，它确保同一事务的多个实例在并发读取数据时，会看到同样的数据行。不过理论上，这会导致幻读 (Phantom Read)，即当用户读取某一范围的数据行时，另一个事务又在该范围内插入了新行，当用户再读取该范围的数据行时，会发现有了新的“幻影”行。InnoDB 和 Falcon 存储引擎通过多版本并发控制 (MVCC, Multiversion Concurrency Control) 机制得以解决。
- ✚ **Serializable**(可串行化)：最高的隔离级别，可读，不可写，它通过强制事务排序，使之不可能相互冲突，从而解决幻读问题。简言之，它是在每个读的数据行上加上共享锁。在这个级别，可能导致大量的超时现象和锁竞争。

## ■ 实验总结

通过本次实验，我理解和掌握了MySQL数据库中三种模式的事务，通过比较理解了他们之间的区别，加强了自身对事务的理解。不过在实验过程中碰到了一些问题，没有事先了解到MySQL中的事务是属于自动提交的特点而出现了错误，在查询事务隔离级别时由于版本问题，新版本的tx\_isolation变量名变为transaction\_isolation从而导致1193错误，修改变量名之后方可顺利完成实验。