

数据库的事务创建与运行实验

【实验七】



目录

[一． 数据库的事务创建与运行实验 1](#_Toc515788504)

[1． 实验目的 1](#_Toc515788505)

[2． 实验环境 1](#_Toc515788506)

[3． 实验内容与要求 1](#_Toc515788507)

[4． 实验步骤及结果分析 2](#_Toc515788508)

[5． 实验小结 6](#_Toc515788509)

2018-5-29

[裴子祥 计科七班 学号2015211921]

[指导老师：杜军平]

# 数据库的事务创建与运行实验

1. 实验目的

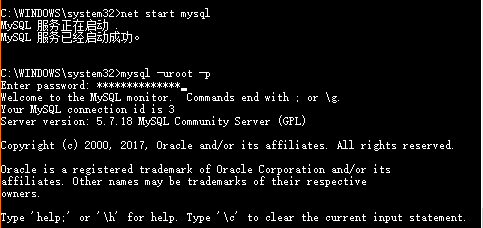
通过实验，了解mysql数据库系统中各类数据库事务的定义机制和基于锁的并发控制机制，掌握mysql数据库系统的事务控制机制。

1. 实验环境

**Microsoft Windows 10 专业版 64位**



**数据库版本：5.7.18 MySQL Community Server (GPL)**



1. 实验内容与要求
2. **定义三种模式的数据库事务**
3. 显式事务。

（2） 自动提交事务。

（3） 隐式事务。

**2. 察看事务的隔离级别**

（1） innodb系统级别的事务隔离级别

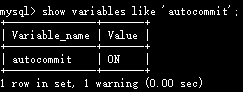
（2） innodb会话级别的事务隔离级别

1. 实验步骤及结果分析

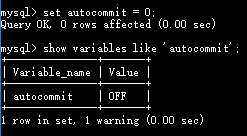
**1、定义三种模式的数据库事务**

（1）显式事务

mysql默认采用autocommit模式。



修改autocommit变量。



创建一个innodb类型表格course2

create table course2(

cno varchar(3) NOT NULL,

cname varchar(12) DEFAULT NULL,

lhour int(11) DEFAULT NULL,

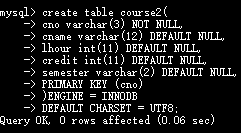
credit int(11) DEFAULT NULL,

semester varchar(2) DEFAULT NULL,

PRIMARY KEY (cno)

)ENGINE = INNODB

DEFAULT CHARSET = UTF8;

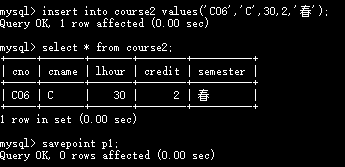


insert into course2 values('C06','C',30,2,'春');

select \* from course2;

savepoint p1;

插入任意一个元组，并设置一个回滚点p1。

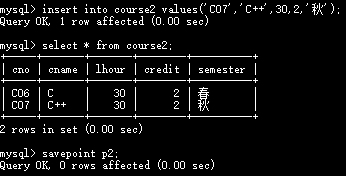


insert into course2 values('C07','C++',30,2,'秋');

select \* from course2;

savepoint p2;

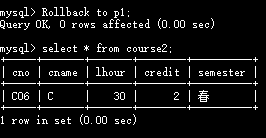
再插入一个元组，并设置一个回滚点p2。



Rollback to p1;

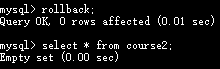
select \* from course2;

回滚到p1。



Rollback;

回滚到当前没有执行上一步的时候。

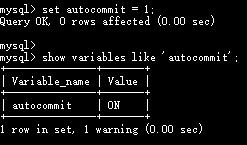


（2）自动提交事务

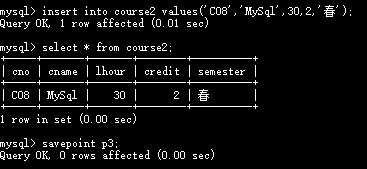
设置autocommit值为自动提交

set autocommit = 1;

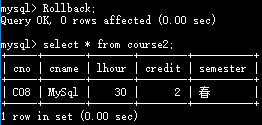
show variables like 'autocommit';



插入元组，设置回滚点p3。

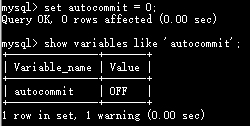


回滚，发现并未回滚成功。



（3）隐式事务

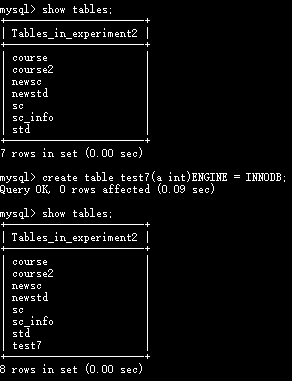
设置autocommit值为0



show tables;

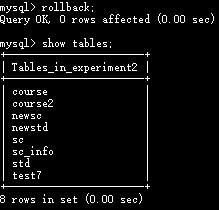
create table test7(a int)ENGINE = INNODB;

show tables;



rollback;

show tables;

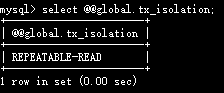


回滚失败。

**2、察看事务的隔离级别**

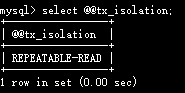
（1）innodb系统级别的事务隔离级别

select @@global.tx\_isolation;



（2）innodb会话级别的事务隔离级别

select @@tx\_isolation;



1. 实验小结

**实验结果：**

1. 建立显示事务，插入元组，设置回滚点，并回滚，发现操作都被回滚。显式事务执行成功。事务不自动提交，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。可以使用“rollback to point\_name”回滚到指定节点。
2. 建立自动提交事务，插入元组后，设置回滚点，回滚后，发现结果并没有改变，说明回滚失败，数据已经自动提交。每当执行一条SQL语句，这条语句就被当做一个完整的事务并提交，而被自动提交的事务没有回滚机制的支持。
3. 隐式事务中，autocommit=off，但某些语句会隐含地结束一个事务，直接commit结果，比如create table，alter funciton，drop table等。

**事务隔离级别：**

1）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）。

2）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其间可能会有新的commit，所以同一select可能返回不同结果。

3）Repeatable Read（可重读）：这是MySQL的默认事务隔离级别，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，这会导致另一个棘手的问题：幻读（Phantom Read）。简单的说，幻读指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）机制解决了该问题。

4）Serializable（可串行化）：这是最高的隔离级别，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。

**心得总结：**

这次本实验加深了我对数据库事务和隔离级别的理解。通过实验实践课堂学习的知识，并加以验证，慢慢掌握，收获颇丰。事务是对数据库操作提出一些约束的机制，其性质为原子性（语句组的不可分割）、一致性（修改的一致性，为rollback提供前提）、隔离性（每个事务有自己的空间，不同事务之间不发生影响）、持久性（事务一旦提交，对数据的修改就是永久的）。这种机制为数据库的可靠性提供保障，例如回滚机制，很好地控制和处理并发请求问题。事务越独立，并发副作用越小，但付出代价更大，因为隔离的本质是串行，这与并发是冲突的。隔离级别是用来平衡独立性与并发性以适应不同场景需求。