北京邮电大学数据库系统实验报告

实验名称: 数据库的事务创建与运行实验

计算机科学与技术系
2015211306班
魏晓
学号:2015211301

实验目的

- 通过实验，了解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，掌握数据库系统的事务控制机制。

实验内容

1. 定义三种模式的数据库事务
2. 察看事务的隔离级别

实验平台及环境

实验平台:MySQL 14.14 with Python 3.6
运行环境:Mac OS High Sierra 10.13(17A405)
可视界面:MySQL WorkBench 6.3.10

```
Xiaos-MacBook-Pro:~ weixiao$ mysql --version
mysql  Ver 14.14 Distrib 5.7.20, for macos10.12 (x86_64) using  EditLine wrapper
```

实验步骤及结果分析

MySQL中三种数据库事务模式

显示事务

- 使用set autocommit = 0 的命令来使MySQL工作在显示事务模式下.在这种模式下,关键字begin表示一个事务的开始,commit表示提交这个事务,rollback表示删除这个事务,有的也叫回滚事务
  - 初始course

```
mysql> select * from course;
+-----+----------------------+-------+--------+----------+
| cno | cname                | lhour | credit | semester |
+-----+----------------------+-------+--------+----------+
| C01 | 编译原理             |    51 |      3 | 球       |
| 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 | 数据结构             |    60 |      3 | 春       |
+-----+----------------------+-------+--------+----------+
6 rows in set (0.00 sec)
```

  - 设置显示事务

```
mysql> set autocommit= 0;
Query OK, 0 rows affected (0.00 sec)
```

  - 定义如下事务

```
mysql> begin;
Query OK, 0 rows affected (0.01 sec)

mysql> insert into course values ('C07','math',0,0,'spring');
ERROR 1406 (22001): Data too long for column 'semester' at row 1
mysql> insert into course values ('C07','math',0,0,'sp');
Query OK, 1 row affected (0.01 sec)

mysql> savepoint s1;
Query OK, 0 rows affected (0.00 sec)

mysql> insert into course values ('C08','science',1,1,'aut');
ERROR 1406 (22001): Data too long for column 'semester' at row 1
mysql> insert into course values ('C08','science',1,1,'au');
Query OK, 1 row affected (0.01 sec)

mysql> savepoint s2;
Query OK, 0 rows affected (0.00 sec)
```

  - 现在,在存储点1我们只增加了C07,在s2又增加了C08,此时事务并没有被提交,我们可以使用select来检查当前的状态

```
mysql> select * from course;
+-----+----------------------+-------+--------+----------+
| cno | cname                | lhour | credit | semester |
+-----+----------------------+-------+--------+----------+
| C01 | 编译原理             |    51 |      3 | 球       |
| 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 | 数据结构             |    60 |      3 | 春       |
| C07 | math                 |     0 |      0 | sp       |
| C08 | science              |     1 |      1 | au       |
+-----+----------------------+-------+--------+----------+
8 rows in set (0.00 sec)
```

- 我们在回滚到点s2,再查看当前数据库

```
mysql> rollback to s1;
Query OK, 0 rows affected (0.00 sec)

mysql> select * from course;
+-----+----------------------+-------+--------+----------+
| cno | cname                | lhour | credit | semester |
+-----+----------------------+-------+--------+----------+
| C01 | 编译原理             |    51 |      3 | 球       |
| 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 | 数据结构             |    60 |      3 | 春       |
| C07 | math                 |     0 |      0 | sp       |
+-----+----------------------+-------+--------+----------+
7 rows in set (0.00 sec)
```

- 可以看见回滚能够返回到过去的某个状态,相当于没有被执行
- 如果想要确认当前的状态可以使用commit来提交

```
mysql> commit;
Query OK, 0 rows affected (0.00 sec)

mysql> select * from course;
+-----+----------------------+-------+--------+----------+
| cno | cname                | lhour | credit | semester |
+-----+----------------------+-------+--------+----------+
| C01 | 编译原理             |    51 |      3 | 球       |
| 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 | 数据结构             |    60 |      3 | 春       |
| C07 | math                 |     0 |      0 | sp       |
+-----+----------------------+-------+--------+----------+
7 rows in set (0.00 sec)
```

隐式事务(自动提交事务)

- 使用set autocommit=1 的时候,就是隐士事务
- 每当我们执行一条SQL语句,这条语句就被当作一个完整的事务并提交,因此在之前的实验的操作均属于被自动提交的事务,隐式事务并没有回滚机制,因为每一次commit都会直接提交到数据库
- 即使是显示事务模式,有一部分SQL语句也被视为隐式事务.比如DROP table:

查看事务的隔离级别

- MysQL提供如下四种隔离级别:
- SERIALIZABLE

> 以序列的形式处理事务,只有事务提交后,用户才能看到,但是该级别的孤立会影响MySQL的性能,因为需要占用大量的资源,以保证大量事务在人意时间不被用户看到

1. REPEATABLE READ

> 相比序列化,这个级别在应用的安全性上做了妥协.

2. READ COMMITTED

> 提交后可读的安全性比可读性还要低,在这一级别的事务,用户可以看到其他事务添加的新纪录,在事务处理的时候,如果存在其他用户同时对事务的响应表进行修改,那么统一事务在不同时间使用SELECT查询的到的结果集可能不同

3. READ UNCOMMITED

> 比提交后读更低,同时该孤立集也是事务之间的最小的间隔,该孤立级容易产生虚幻读操作,其他用户可以在这个孤立集上看到未提交的事务

- 查看事务的隔离级别:

```
mysql> select @@tx_isolation;
+-----------------+
| @@tx_isolation  |
+-----------------+
| REPEATABLE-READ |
+-----------------+
1 row in set, 1 warning (0.01 sec)
```

实验总结

- 什么是事务?
  - 我认为事务是一种被设计出来的机制,该机制对数据库的一系列操作提出了一些约束,比如原子性约束,一致性,隔离型和持久性.这种极致为数据库的可靠性提供了保障,比如提出回滚机制,并很好的控制和处理并发请求
  - 事务越独立,并发的副作用就越小,但付出的代价就越大,这是因为隔离的本质是串行,这与并发是冲突的.为了平衡独立性和并发性,数据库用隔离级别来平衡这两个指标以适应不同情境下的应用