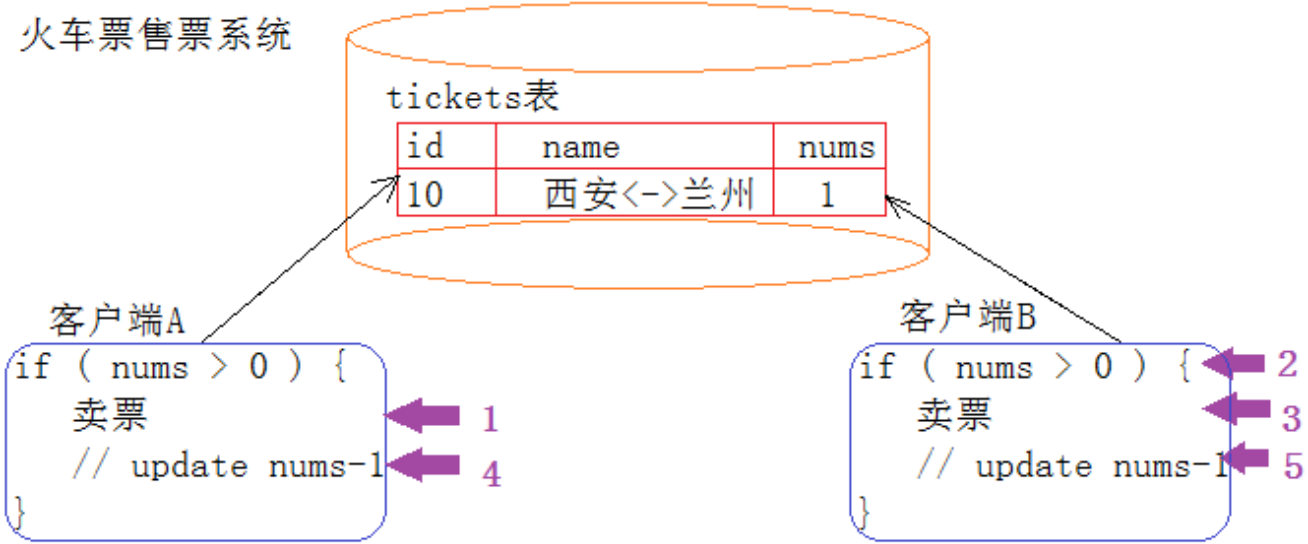


11. 事务管理（重点）

事务就是一组DML语句组成，这些语句在逻辑上存在相关性，这一组DML语句要么全部成功，要么全部失败，是一个整体。MySQL提供一种机制，保证我们达到这样的效果。事务还规定不同的客户端看到的数据是不相同的。

事务理论的深度理解参见推荐书籍：《数据库系统概念》的第14章。



当客户端A检查还有一张票时，将票卖掉，还没有执行更新数据库时，客户端B检查了票数，发现大于0，于是又卖了一次票。然后A将票数更新回数据库。这是就出现了同一张票被卖了两次。

11.1 事务基本操作

案例：

银行转账的例子更需要考虑事务的问题。

- 创建测试表

```
create table account(  
    id int primary key,  
    name varchar(50) not null default '',  
    balance decimal(10, 2) not null default 0.0  
);
```

- 开始一个事务
`start transaction;`
- 创建一个保存点
`savepoint 保存点名;`
- 回到保存点（根据具体情况）
`rollback to 保存点名;`

- 代码演示

```
mysql> start transaction; --开启事务
Query OK, 0 rows affected (0.00 sec)

mysql> savepoint aa; --设置保存点aa
Query OK, 0 rows affected (0.00 sec)

mysql> insert into account values(1, '张三', 10); --添加一条记录
Query OK, 1 row affected (0.00 sec)

mysql> savepoint bb; -- 设置保存点bb
Query OK, 0 rows affected (0.00 sec)

mysql> insert into account values(2, '李四', 10000); --再添加一条记录
Query OK, 1 row affected (0.00 sec)

mysql> select * from account; --两条记录都在了
+----+-----+-----+
| id | name  | balance |
+----+-----+-----+
| 1  | 张三  | 10.00   |
| 2  | 李四  | 10000.00 |
+----+-----+-----+
2 rows in set (0.00 sec)

mysql> rollback to bb; -- 发现后来添加这一条记录是误操作。所以回滚到bb状态
Query OK, 0 rows affected (0.01 sec)

mysql> select * from account; -- 第二条记录没有了
+----+-----+-----+
| id | name  | balance |
+----+-----+-----+
| 1  | 张三  | 10.00   |
+----+-----+-----+
1 row in set (0.03 sec)
```

11.2 事务操作注意事项

- 如果没有设置保存点，也可以回滚，只能回滚到事务的开始。直接使用 rollback(前提是事务还没有提交)
- 如果一个事务被提交了 (commit) ， 则不可以回退 (rollback)
- 可以选择回退到哪个保存点
- InnoDB支持事务， MyISAM不支持事务
- 开始事务可以使 start transaction

11.3 事务的隔离级别

当我们有多个客户端同时操作数据库的某张表，如何进行隔离操作？MySQL提供了隔离级别。

当MySQL表被多个线程或者客户端开启各自事务操作数据库中的数据时，MySQL提供了一种机制，可以让不同的事务在操作数据时，具有隔离性。从而保证数据的一致性。

11.3.1 无隔离性的问题

- 脏读

是指当一个事务正在访问数据，并且对数据进行了修改，而这种修改还没有提交到数据库中，这时，另外一个事务也访问这个数据，然后使用了这个数据。

举例：

1. Mary的原工资为1000，财务人员将Mary的工资改为了8000(但未提交事务)
2. Mary读取自己的工资，发现自己的工资变为了8000，欢天喜地！
3. 而财务发现操作有误，回滚了事务，Mary的工资又变为了1000
像这样，Mary读取的工资数8000是一个脏数据。

- 不可重复读

是指在一个事务内，多次读同一数据。在这个事务还没有结束时，另外一个事务也访问该同一数据。那么，在第一个事务中的两次读数据之间，由于第二个事务的修改，那么第一个事务两次读到的数据可能是不一样的。这样就发生了在一个事务内两次读到的数据是不一样的，因此称为是不可重复读。（即不能读到相同的数据内容）

举例：

1. 在事务1中，Mary 读取了自己的工资为1000，操作并没有完成
2. 在事务2中，这时财务人员修改了Mary的工资为2000，并提交了事务。
3. 在事务1中，Mary 再次读取自己的工资时，工资变为了2000

解决办法：如果只有在修改事务完全提交之后才可以读取数据，则可以避免该问题

- 幻读

是指当事务不是独立执行时发生的一种现象，例如第一个事务对一个表中的数据进行了修改，这种修改涉及到表中的全部数据行。同时，第二个事务也修改这个表中的数据，这种修改是向表中插入一行新数据。那么，以后就会发生操作第一个事务的用户发现表中还有没有修改的数据行，就好象发生了幻觉一样。

举例：

目前工资为1000的员工有10人。
1. 事务1，读取所有工资为1000的员工。
2. 这时事务2向employee表插入了一条员工记录，工资也为1000
3. 事务1再次读取所有工资为1000的员工 共读取到了11条记录，

解决办法：如果在操作事务完成数据处理之前，任何其他事务都不可以添加新数据，则可避免该问题

备注：不可重复读的重点是修改：同样的条件，你读取过的数据，再次读取出来发现值不一样了 幻读的重点在于新增或者删除：同样的条件，第1次和第2次读出来的记录数不一样

11.3.2 事务的隔离级别

隔离级别	脏读	不可重复读	幻读	加锁读
读未提交 (read uncommitted)	✓	✓	✓	不加锁
读已提交 (read committed)	✗	✓	✓	不加锁
可重复读 (repeatable read)	✗	✗	✗	不加锁
可串行化 (serializable)	✗	✗	✗	加锁

✓: 会发生该问题
✗: 不会发生该问题

- 设置事务的隔离级别

语法:

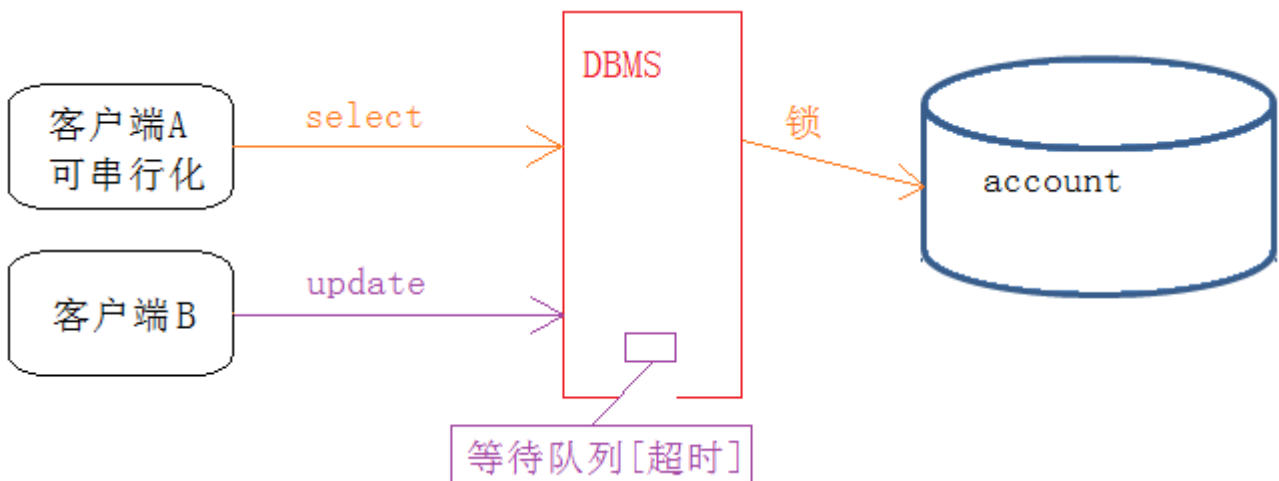
```
set session transaction isolation level read uncommitted;
```

- 查看当前的隔离级别:

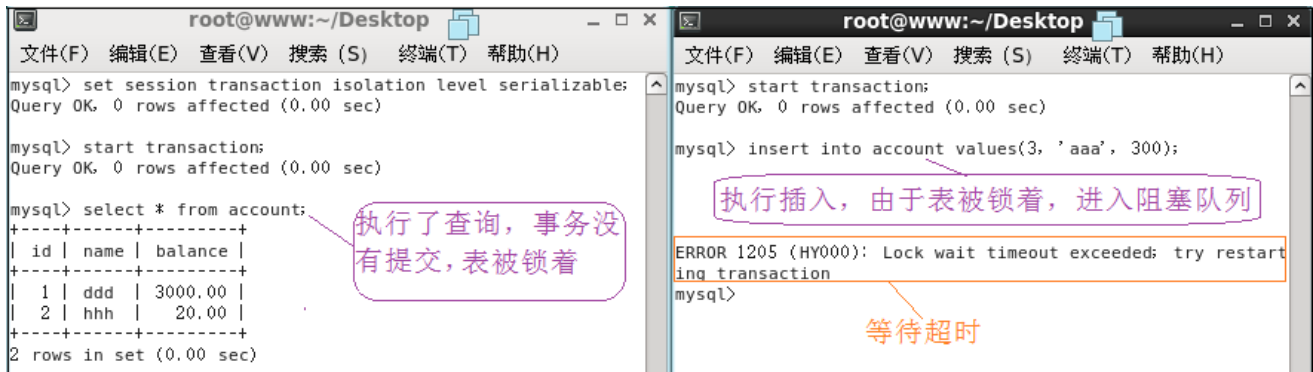
```
mysql> select @@tx_isolation;
```

```
+-----+
| @@tx_isolation |
+-----+
| READ-UNCOMMITTED |
+-----+
```

- 隔离级别: 可串行化 案例:



- 当客户端A在执行select过程中, DBMS会对库加锁, 如果这时客户端B执行插入, 只要还没释放锁, 插入不进去, 会将B的update语句放入等待队列, 直到释放了锁或超时。



```
mysql> set session transaction isolation level serializable;
Query OK, 0 rows affected (0.00 sec)

mysql> start transaction;
Query OK, 0 rows affected (0.00 sec)

mysql> select * from account;
+----+-----+-----+
| id | name | balance |
+----+-----+-----+
| 1  | ddd  | 3000.00 |
| 2  | hhh  | 20.00   |
+----+-----+-----+
2 rows in set (0.00 sec)

mysql> start transaction;
Query OK, 0 rows affected (0.00 sec)

mysql> insert into account values(3, 'aaa', 300);
ERROR 1205 (HY000): Lock wait timeout exceeded; try restarting transaction
mysql>
```

说明：mysql默认的隔离级别是可重复读,一般情况下不要修改

11.4 事务的ACID特性

- **原子性(Atomicity):**

事务是应用中最小的执行单位，就如原子是自然界的最小颗粒，具有不可再分的特征一样，事务是应用中不可再分的最小逻辑执行体。

- **一致性(Consistency):**

事务执行的结果，必须使数据库从一个一致性状态，变到另一个一致性状态。当数据库只包含事务成功提交的结果时，数据库处于一致性状态。如果系统运行发生中断，某个事务尚未完成而被迫中断，而改未完成的事务对数据库所做的修改已被写入数据库，此时数据库就处于一种不正确（不一致）的状态。因此一致性是通过原子性来保证的。

- **隔离性(Isolation):**

各个事务的执行互不干扰，任意一个事务的内部操作对其他并发事务都是隔离的。也就是说，并发执行的事务之间不能看到对方的中间状态，并发执行的事务之间不能互相影响。

- **持久性(Durability):**

持久性是指一个事务一旦被提交，它对数据库所做的改变都要记录到永久存储其中（如：磁盘）。