我们在操作数据库的时候，可能会由于并发问题而引起的数据的不一致性（数据冲突）

**乐观锁（和CAS很类似）**

乐观锁不是数据库自带的，需要我们自己去实现。乐观锁是指操作数据库时**(更新操作**)，想法很乐观，认为这次的操作不会导致冲突，在操作数据时，并不进行任何其他的特殊处理（也就是不加锁），而在进行更新后，再去判断是否有冲突了。

通常实现是这样的：在表中的数据进行操作时(更新)，先给数据表加一个版本(version)字段，每操作一次，将那条记录的版本号加1。也就是先查询出那条记录，获取出version字段,如果要对那条记录进行操作(更新),则先判断此刻version的值是否与刚刚查询出来时的version的值相等，如果相等，则说明这段期间，没有其他程序对其进行操作，则可以执行更新，将version字段的值加1；如果更新时发现此刻的version值与刚刚获取出来的version的值不相等，则说明这段期间已经有其他程序对其进行操作了，则不进行更新操作。(防止ABA这种情况出现)

举例：

下单操作包括3步骤：

1.查询出商品信息

select (status,status,version) from t\_goods where id=#{id}

2.根据商品信息生成订单

3.修改商品status为2

update t\_goods

set status=2,version=version+1

where id=#{id} and version=#{version};

除了自己手动实现乐观锁之外，现在网上许多框架已经封装好了乐观锁的实现，如hibernate，需要时，可能自行搜索"hiberate 乐观锁"试试看。

**悲观锁**

与乐观锁相对应的就是悲观锁了。悲观锁就是在操作数据时，认为此操作会出现数据冲突，所以在进行每次操作时都要通过获取锁才能进行对相同数据的操作，这点跟java中的synchronized很相似，所以悲观锁需要耗费较多的时间。另外与乐观锁相对应的，悲观锁是由数据库自己实现了的，要用的时候，我们直接调用数据库的相关语句就可以了。

说到这里，由悲观锁涉及到的另外两个锁概念就出来了，它们就是共享锁与排它锁。共享锁和排它锁是悲观锁的不同的实现，它俩都属于悲观锁的范畴。

**共享锁**

共享锁指的就是对于多个不同的事务，对同一个资源共享同一个锁。就像这样，你家有一个大门，大门只有一个锁头，锁头的钥匙有好几把，你有一把，你女朋友有一把，你们都可能通过这把钥匙进入你们家，进去啪啪啪啥的，一下理解了哈，没错，这个就是所谓的共享锁（大门对应资源，锁头对应资源共享的那个锁，钥匙对应不同的事务）。

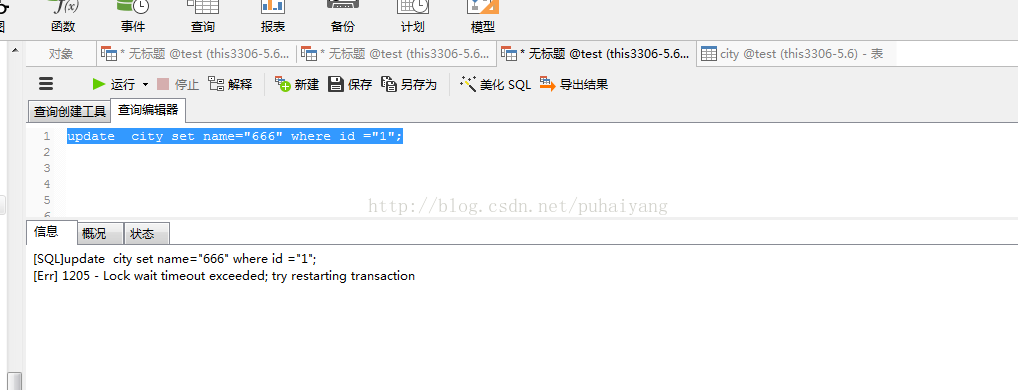
刚刚说了，对于悲观锁，一般数据库已经实现了，**共享锁也属于悲观锁的一种**，那么共享锁在mysql中是通过什么命令来调用呢。通过查询资料，了解到通过在执行语句后面加上lock in share mode就代表对某些资源加上共享锁了。

比如，我这里通过mysql打开两个查询编辑器，在其中开启一个事务，并不执行commit语句

表DDL如下：

1. CREATE TABLE `city` (
2. `id` bigint(20) NOT NULL AUTO\_INCREMENT,
3. `name` varchar(255) DEFAULT NULL,
4. `state` varchar(255) DEFAULT NULL,
5. PRIMARY KEY (`id`)
6. ) ENGINE=InnoDB AUTO\_INCREMENT=18 DEFAULT CHARSET=utf8;

begin;  
SELECT \* from city where id = "1"  lock in share mode;

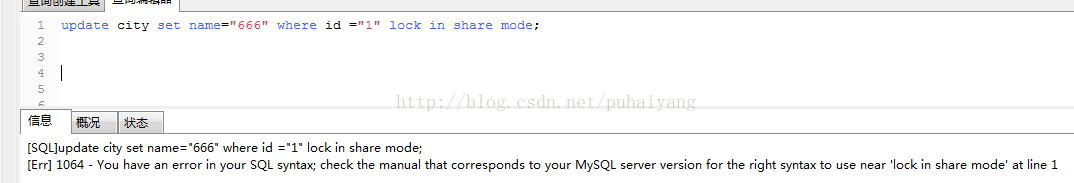
然后在另一个查询窗口中，对id为1的数据进行更新update  city set name="666" where id ="1";

此时，操作界面进入了卡顿状态，过几秒后，也提示错误信息

**[SQL]update  city set name="666" where id ="1";  
[Err] 1205 - Lock wait timeout exceeded; try restarting transaction**

那么证明，对于id=1的记录加锁成功了，在上一条记录还没有commit之前**（事务没执行完）**，这条id=1的记录被锁住了，只有在上一个事务释放掉锁后才能进行操作，或用共享锁才能对此数据进行操作。

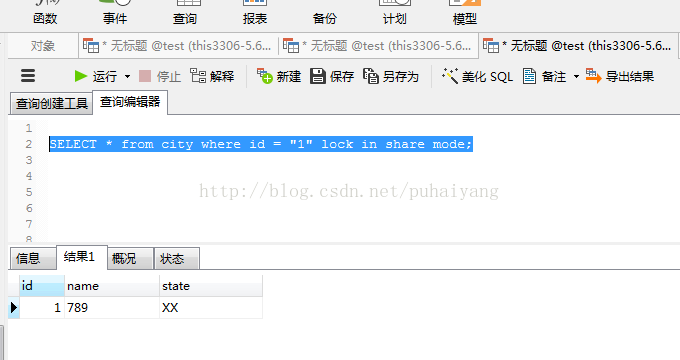
再实验一下：

update city set name="666" where id ="1" lock in share mode;

**[Err] 1064 - You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'lock in share mode' at line 1**

加上共享锁后，也提示错误信息了，通过查询资料才知道，对于update,insert,delete语句会自动加排它锁的原因

于是，我又试了试SELECT \* from city where id = "1" lock in share mode;

这下成功了。

**在一个事务加入共享锁后，其他事务未加入共享锁时，无法查询，加入后，只有查询，select加入共享锁才会有效。**

**而且只有查询select加入的共享锁才是共享锁，如果有update，等更改数据的操作，锁会直接升级为排它锁。此时如果有一个事务在修改数据，不允许其他事务再进行。**

**共享锁就是多个事务对于同一数据可以共享一把锁，都能访问到数据，但是只能读不能修改。**

**直到已释放所有共享锁。**

update,insert,delete语句会自动加排它锁的原因

### 排它锁

排它锁与共享锁相对应，就是指对于多个不同的事务，对同一个资源只能有一把锁。

与共享锁类型，在需要执行的语句后面加上for update就可以了

共享锁又称为读锁，简称S锁，顾名思义，**共享锁就是多个事务对于同一数据可以共享一把锁，都能访问到数据，但是只能读不能修改。**

排他锁又称为写锁，简称X锁，顾名思义，排他锁就是不能与其他所并存，如一个事务获取了一个数据行的排他锁，其他事务就不能再获取该行的其他锁，包括共享锁和排他锁，**但是获取排他锁的事务是可以对数据就行读取和修改。**

## 行锁

行锁，由字面意思理解，就是给某一行加上锁，也就是一条记录加上锁。

比如之前演示的共享锁语句

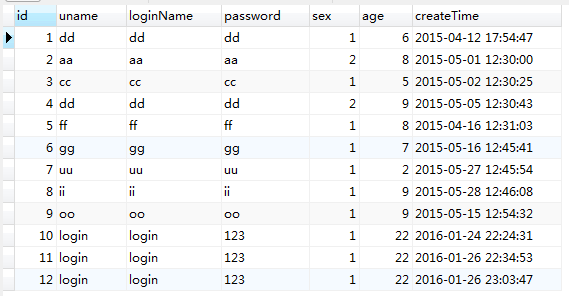
SELECT \* from city where id = "1"  lock in share mode;

由于对于city表中,id字段为主键，就也相当于索引。执行加锁时**，会将id这个索引为1的记录加上共享锁，那么这个锁就是行锁**。

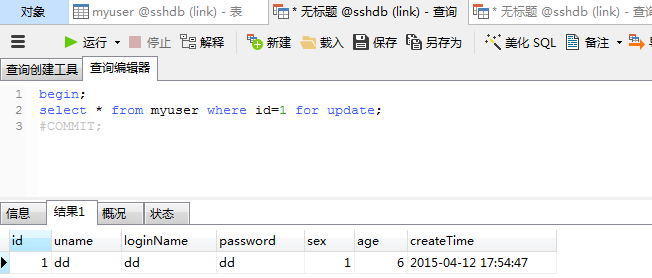
## 表锁

表锁，和行锁相对应，给这个表加上锁。

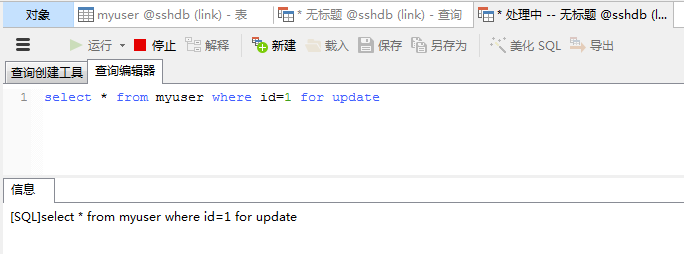
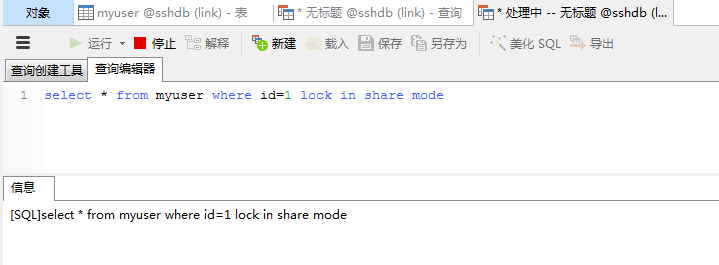
**再给出一个共享锁和排他锁的例子：**

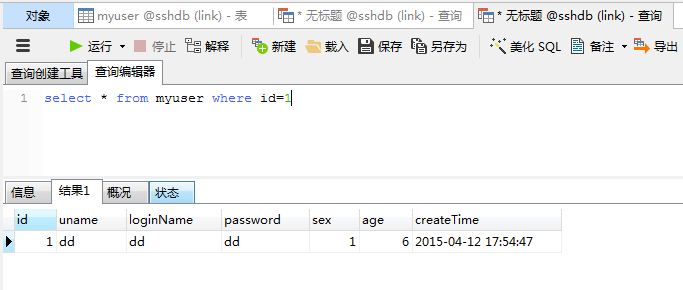
现在我们对id=1的数据行排他查询，这里会使用begin开启事务，而不会看见我关闭事务，这样做是用来测试，因为提交事务或回滚事务就会释放锁。

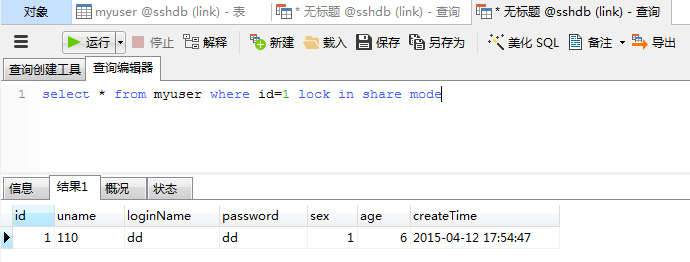
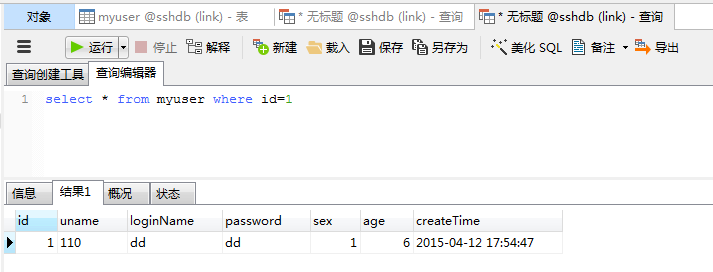
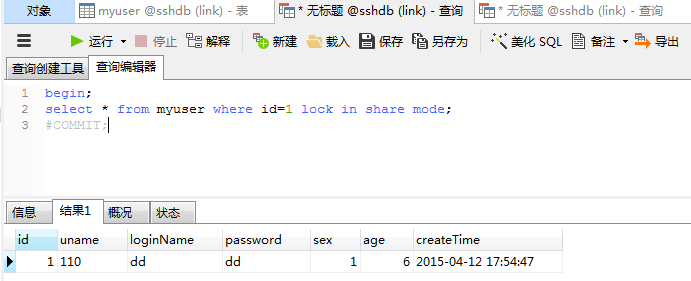
打开一个查询窗口

会查询到一条数据，现在打开另一个查询窗口，对同一数据分别使用排他查和共享锁查询两种方式查询**（加了排他锁）**

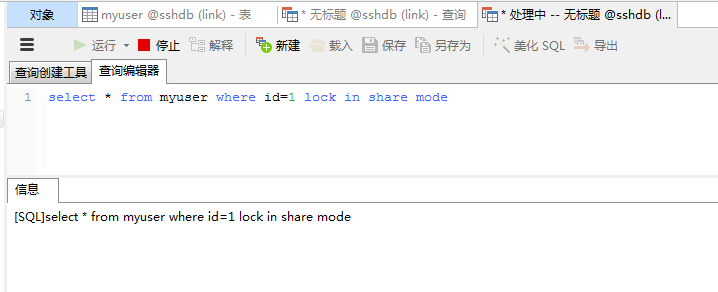
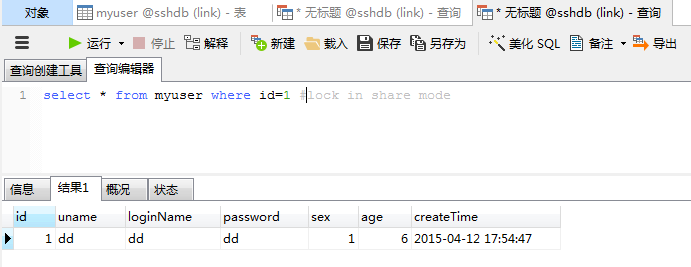
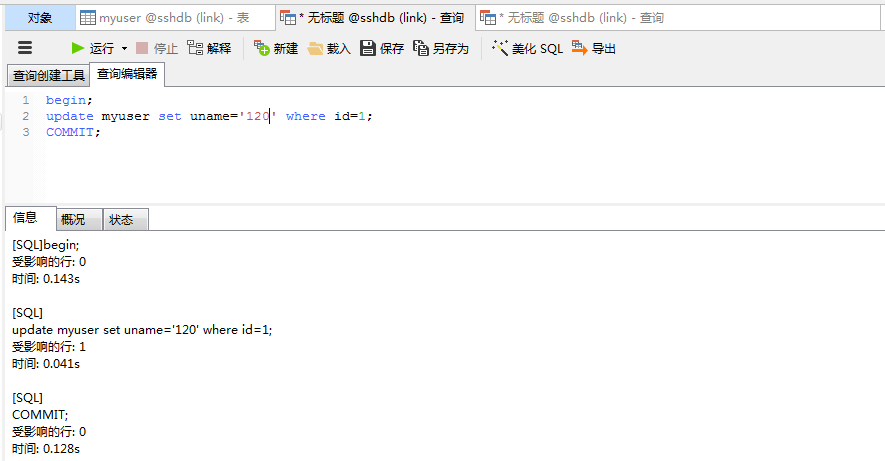
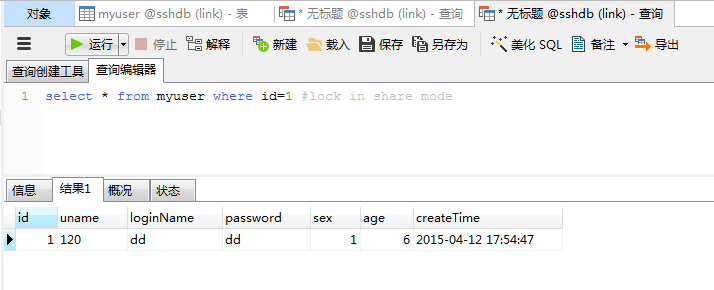
排他查

共享查我们看到开了排他锁查询和共享锁查询都会处于阻塞状态，因为id=1的数据已经被加上了排他锁，**此处阻塞是等待排他锁释放，无法加入其他锁了**。

**如果我们直接使用以下查询呢**我们看到是可以查询到数据的，但加排他锁就查不到，因为排他锁与共享锁不能存在同一数据上。所以加入了共享锁反而查不到。

我们再看一下一个事务获取了**共享锁，在其他查询中也只能加共享锁或不加锁。**我们看到是可以查询数据的，但加排他锁就查不到，**因为排他锁与共享锁不能存在同一数据上。**

最后我们验证下上面说的mysql InnoDb引擎中**update,delete,insert**语句自动加排他锁的问题，

 **此时共享查询处于阻塞，等待排它锁的释放，但是用普通查询能查到数据，因为没用上锁机制不与排他锁互斥，但查到的数据是修改数据之前的老数据。**然后我们提交数据，**释放排他锁看下修改后的数据，此时可用排他查，共享查和普通查询，** 因为事务提交后该行数据**释放排他锁**，下面就只显示普通查询，其他的同学们自己去验证。**可以看到结果与预期的一样。（都查到了最新的结果）**

**总结：由于Mysql等自己的增删改操作加了排它锁，所以在进行此等事务没有完成时，所有加了锁的增删改查都发挥不了，但是不加锁的普通查询还是可以查询得到，但是是查到的旧数据，加了排它锁更新的数据还未能查询到。**