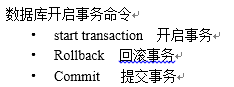
# 事务

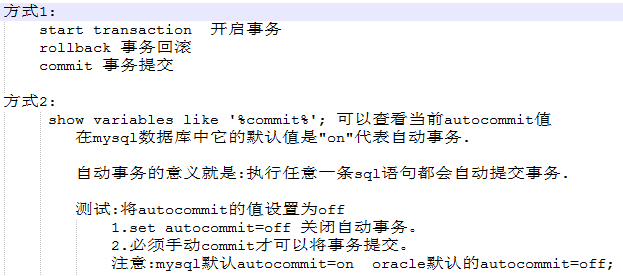
## 事务的概念

事务指逻辑上的一组操作，组成这组操作的各个单元，要不全部成功，要不全部不成功。



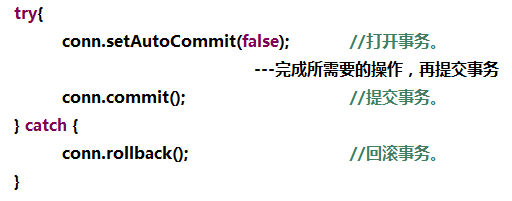
在开发中，有事务的存在，可以保证数据完整性。

## mysql下怎样操作



## Jdbc怎样操作

### 事务处理



//jdbc中事务操作

public class TransactionTest2 {

public static void main(String[] args) {

// 修改id=2这个人的money=500;

String sql = "update account set money=500 where id=2";

Connection con = null;

Statement st = null;

try {

con = JdbcUtils.getConnection();

con.setAutoCommit(false); // 开启事务，相当于 start transaction;

st = con.createStatement();

st.executeUpdate(sql);

} catch (SQLException e) {

e.printStackTrace();

// 事务回滚

try {

con.rollback();

} catch (SQLException e1) {

e1.printStackTrace();

}

} finally {

try {

con.commit(); // 事务提交

st.close();

con.close();

} catch (SQLException e) {

e.printStackTrace();

}

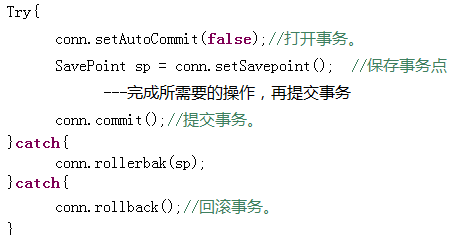
}

}

}

### 事务保存点（SavePoint）

只回滚一部分事务，当只想撤销事务中的部分操作时可使用SavePoint



## 性质ACID

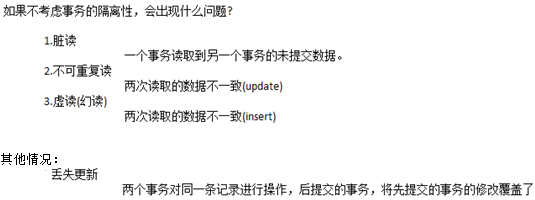
**原子性（Atomicity）**  
 原子性是指事务是一个不可分割的工作单位，事务中的操作要么都发生，要么都不发生。

**一致性（Consistency）**  
 事务前后数据的完整性必须保持一致。(数据库数据完整性约束)

**隔离性（Isolation）--事务与事务之间**  
 事务的隔离性是指多个用户并发访问数据库时，一个用户的事务不能被其它用户的事务所干扰，多个并发事务之间数据要相互隔离。

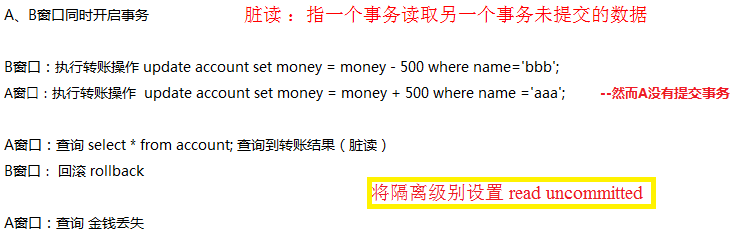
**持久性（Durability）**  
 持久性是指一个事务一旦被提交，它对数据库中数据的改变就是永久性的，接下来即使数据库发生故障也不应该对其有任何影响。

# 并发问题



## 脏读

脏读 ：指一个事务读取另一个事务未提交的数据



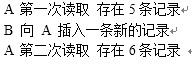
## 不可重复读

不可重复读：在一个事务先后两次读取发生数据不一致情况，第二次读取到另一个事务已经提交数据 （强调数据更新 update）



## 虚读(幻读) ：

在一个事务中，第二次读取发生数据记录数的不同 ，读取到另一个事务已经提交数据 （强调数据记录变化 insert ）



## 丢失更新

# 事务的隔离级别

## 作用

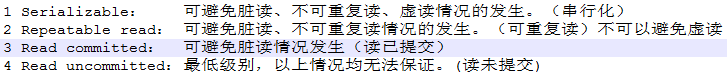
数据库内部定义了四种隔离级别，用于解决三种隔离问题

## 隔离级别

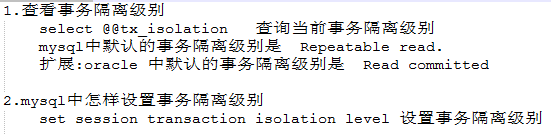
connection.setTransactionIsolation(Connection.TRANSACTION\_READ\_COMMITTED);

V:可能出现，X:不会出现

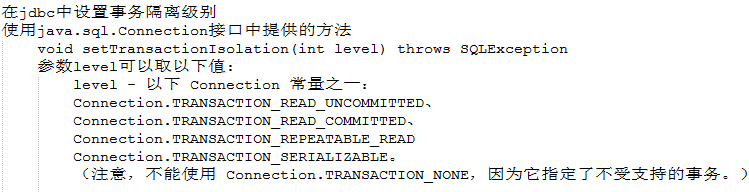




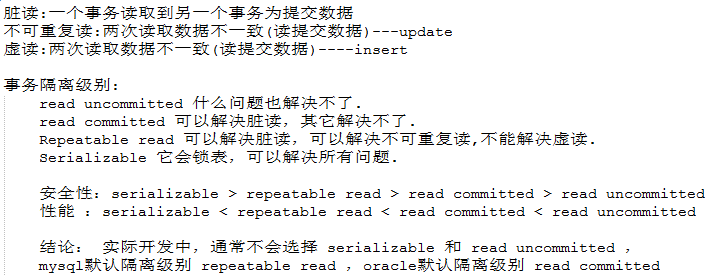
## mysql中设置隔离级别



## jdbc中设置隔离级别



## 小结

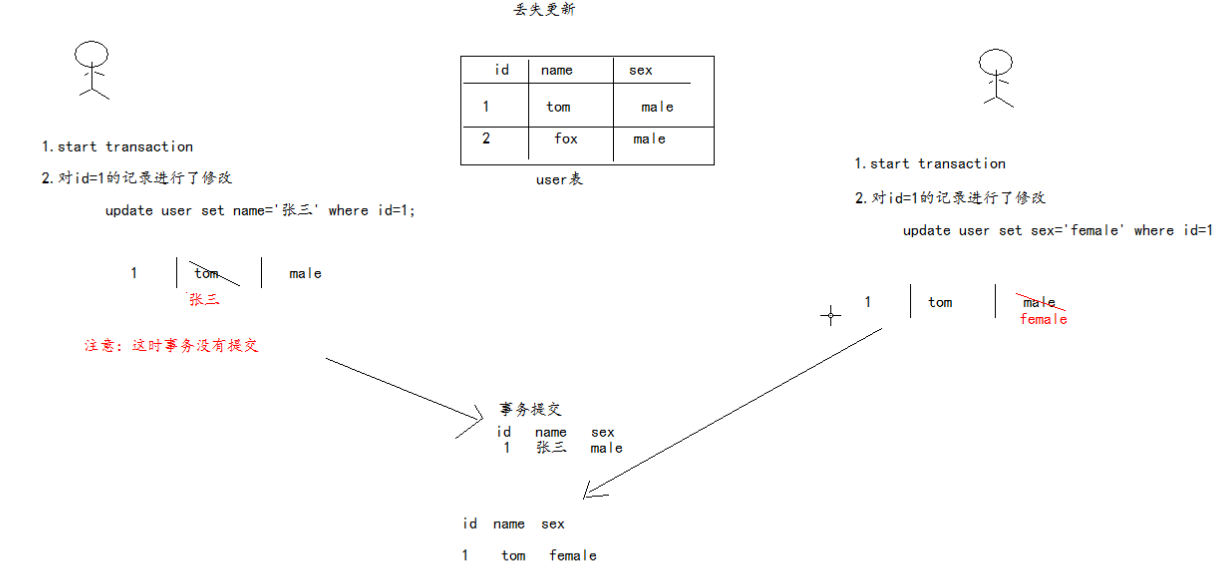


# 丢失更新

## 概述

多个事务对同一条记录进行了操作，后提交的事务将先提交的事务操作覆盖了。

## 图解



## 解决方法

### 悲观锁



允许一张数据表中数据记录，添加多个共享锁，添加共享锁记录，对于其他事务可读不可写的

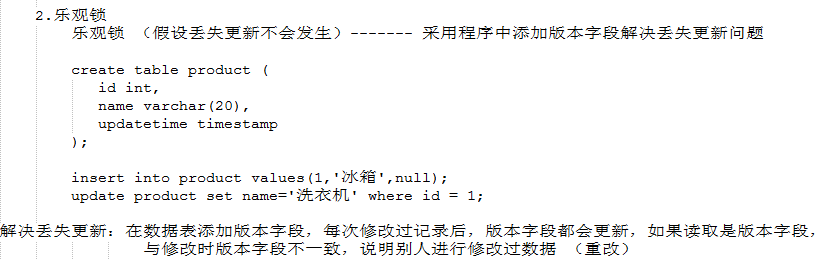
一张数据表中数据记录，只能添加一个排它锁，在添加排它锁的数据 不能再添加其他共享锁和排它锁的 ，对于其他事务可读不可写的

锁必须在事务中添加 ，如果事务结束了 锁就释放了

解决丢失更新：事务在修改记录过程中，锁定记录，别的事务无法并发修改

### 乐观锁





timestamp 在插入和修改时 都会自动更新为当前时间

解决丢失更新：在数据表添加版本字段，每次修改过记录后，版本字段都会更新，如果读取是版本字段，与修改时版本字段不一致，说明别人进行修改过数据 （重改）

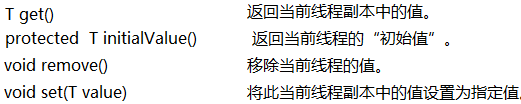
# 绑定本地线程—线程池

## ThreadLocal类

### 构造方法



### 常用方法



### MyThreadLocal<T>

**public** **class** MyThreadLocal<T> {

**private** Map<Thread, T> map;

**private** MyThreadLocal() {

map = **new** HashMap<Thread, T>();

}

**public** **void** set(T t) {

map.put(Thread.*currentThread*(), t);

}

**public** T get() {

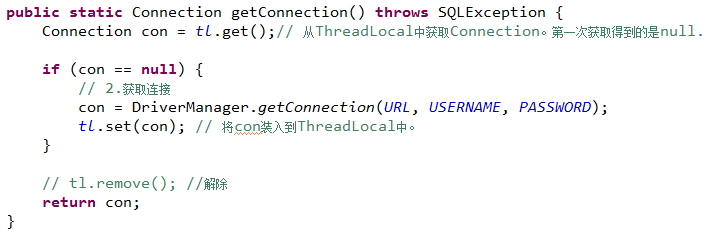
**return** map.get(Thread.*currentThread*());

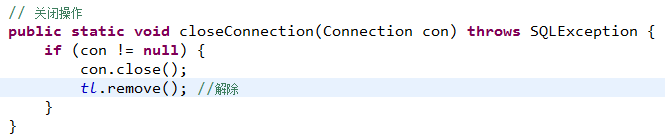
}

}

## 关于JdbcUtils中使用ThreadLocal







# 案例—存钱取钱

