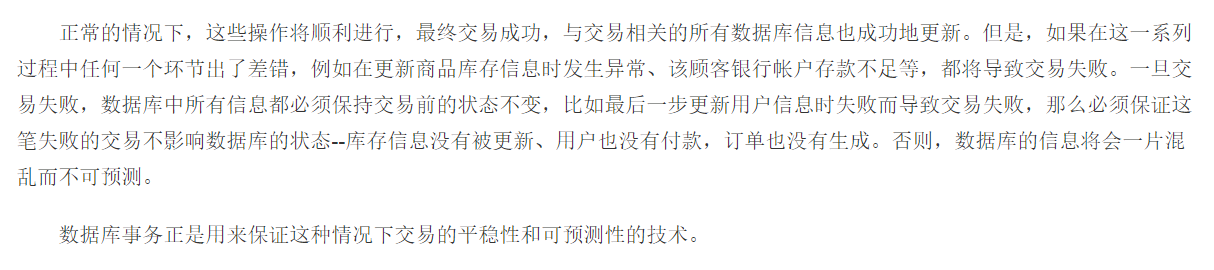
1. 事务的提交概念
2. 第一阶段提交是假提交，可以回滚到原始状态
3. 把假提交编程真正的提交，rollback不回去了
4. 数据库事务概念
5. 数据库事务，是指单个逻辑工作单元执行的一系列操作要么完全执行，要么完全不执行

事务处理可以确保非事务单元内所有操作都成功完成，否则不会永久更新面向数据的资源。通过将一组相关操作组合成一个要么全部成功要么全部失败的单元，可以简化错误恢复并使应用程序更加可靠

1. 一个事务的工作逻辑单元，必须满足所谓的ACID（原子性，一致性，隔离性和持久性）属性
2. 操作流程
3. 设想网上购物的一次交易，其付款过程至少包括几步数据库操作：
4. 更新客户所购商品的库存信息
5. 保存客户付款信息—可能包括与银行系统的交互
6. 生成订单并且保存到数据库中
7. 更新用户相关信息，购物数据等



1. 相关属性
2. 原子性

事务必须使原子工作单元，对于其数据修改，要么全部执行，要么全部失败通常，与某个事务关联的操作具有共同的目标，并且是相互依赖的。如果系统只执行这些操作的一个子集，则可能会破坏事务的总体目标。原子性消除了系统处理操作子集的可能性。

1. 一致性

事务在完成时，必须使所有的数据都保持一致状态。某些维护一致性的责任由应用程序开发人员承担，他们必须确保应用程序已强制所有已知的完整性约束。例如，当开发用于转帐的应用程序时，应避免在转帐过程中任意移动小数点。

1. 隔离性

由并发事务所作的修改必须与任何其它并发事务所作的修改隔离。事务查看数据时数据所处的状态，要么是另一并发事务修改它之前的状态，要么是另一事务修改它之后的状态，事务不会查看中间状态的数据。这称为隔离性。当事务可序列化时将获得最高的[隔离级别](https://baike.baidu.com/item/%E9%9A%94%E7%A6%BB%E7%BA%A7%E5%88%AB)。在此级别上，从一组可并行执行的事务获得的结果与通过连续运行每个事务所获得的结果相同。由于高度隔离会限制可并行执行的事务数，所以一些应用程序降低隔离级别以换取更大的吞吐量。

1. 持久性

事务完成之后，它对于系统的影响是永久的

1. 简单介绍一下和老爸通电话后对事务和分布式事务的理解
2. DAO层操作数据库，service通过注解配置事务



查询的时候才能够加只读事务:

@Transaction(readOnly=true)

Public Page findByPageRequest(PageRequest pr) {  
 return employeesAssignedDao.findByPageRequest(pr);

//假如这边还有一个dao都执行select，事务能够保证这两条sql语句是一个原子操作就行了

}

Try{

java.sql.connection.setAutoCommit(false);

updateSales = con.prepareStatement(updateString);//这两条语句有事务的原子特性

updateTotal = con.prepareStatement(updateStatement);

java.sql.connection.commit();

}catch(Exception e){

If(connection != null)

Java.sql.connection.rollback();

}finally{

If(updateSales != null)

updateSales.close();

If(updateTotal != null)

updateTotal.close();

Connection.setAutoCommit(true);

}

In this method, the auto-commit mode is disabled for the connection con, which means that the two prepared statements updateSales and updateTotal are committed together when the method commit is called.

It is advisable to disable the auto-commit mode only during the transaction mode. This way, you avoid holding database locks for multiple statements, which increases the likelihood of conflicts with other users.

在connection.commit() 和 transaction.rollback()的时候都会释放事务的锁

分布式事务：

1. 从一个银行转账到另外一个银行，可以看成是一个完整的事务；比如说从一个银行扣钱是一个操作，然后发一个消息到另外一个银行，另外一个银行操作完，会发一个回执给这个银行，如果没有收到回执重复发送3次，如果再没有收到就记录日志。整个操作可以看成是一个事务
2. 车管所，有一个总行的数据库存储所有全省的车辆信息，下面一堆市和县每一个车管所都有一个数据库。现在市和县的车辆信息定时向总行的数据库同步最新的车辆信息。

在同步时，建立一个字段叫sync\_success 如果同步成功字段设置成true，否则设置成false；在同步完后如果有失败的，就定时再取出sync\_success为false的再同步一次；这个操作也可以看做是一个事务