# 分布式事务

事务(Transaction)是访问并可能更新数据库中各种数据项的一种程序执行单元。在关系数据库中，一个事务有一组sql语句组成。事务应该具有4个属性：原子性，一致性，隔离性，持久性，这四个属性通常称为ACID特性。

## 原子性(atomicity)

一个事务是一个不可分割的工作单位。事务中包括的诸多操作要么都做，要么都不做。

## 一致性(consistency)

事务必须是使数据库从一个一致性状态变成另一个一致性状态，事务的中间状态不能被观察到的。

## 隔离性(isolation)

一个事务的执行不能被其他事务干扰。即一个事务的内部操作及使用的数据对并发的其他事务的隔离的，并发操作的各个事务之间不能互相干扰。隔离性又分为四个级别：读未提交(read uncommitted)，读已提交(read committd, 解决脏读)，可重复读(repeatable read 解决虚读)，串行化(serializable 解决幻读)。

## 持久性(durability)

持久性也称永久性。指一个事务一旦提交，它对数据库中数据改变就应该是永久性的。接下来的其他操作或故障不应该对其有所影响。

# 本地事务

## @Transational

大多数场景下，我们的应用只需要操作单一的数据库。这种情况下的事务称之为本地事务

（Local Transacton）。本地事务的ACID特性是数据库直接提供的。

# 分布式事务

阿里巴巴Seata 分布式事务为用户带来了4中模式 AT，TCC，SAGA，XA 事务模式

Seata 三大角色 TC（事务协调者），TM（事务管理者），RM（资源管理器）

## 两阶段提交

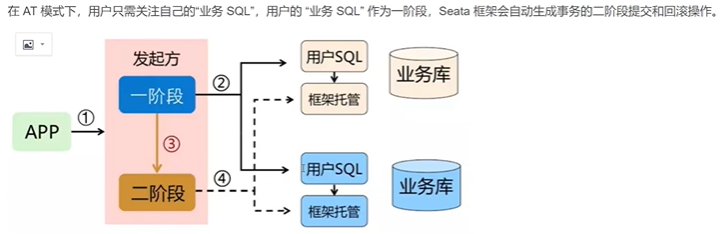
同步阻塞 参与者在等待协调者的指令时，其实是在等待其他参与者的响应

单点在2pc中 一切请求都来源与协调者

数据不一致

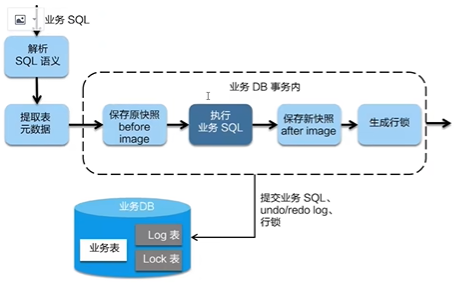
环境可靠性问题

### AT （auto transactional）



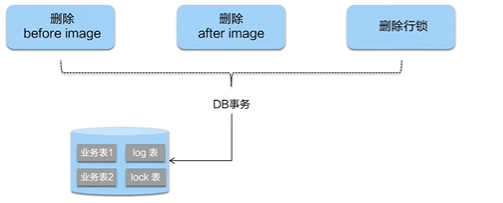
一阶段：Seata会拦截“业务SQL“，首先解析sql语义，找到”业务sql“要更新的业务数据，在业务数据被更新前，将其保存为”before image“ 然后执行行”业务SQL“更新业务数据，在业务数据更新之前，再将其保存成”after image“，最后生成行锁，以上操作全部在一个数据库事务内完成。这样保证了一阶段操作的原子性。

操作如下：



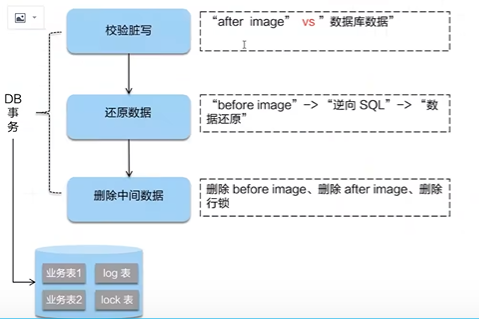
二阶段：

如果是提交的话，因为“业务sql“在一阶段已经提交至数据库，所有Seata框架只需将一阶段保存的快照数据和行锁删掉，完成数据清理即可。



二阶段回滚：

二阶段如果是回滚的话，Seata就需要回滚一阶段已经执行的“业务sql“，还原业务数据。回滚方式便是用”before image“还原业务数据；但在还原前首先检验脏读，对比”数据库当前业务数据“和”after image“，如果两份数据完全一致就说明没有脏写，可以还原也业务数据，如果不一致就说明有脏写，出现脏写就需要转人工处理。

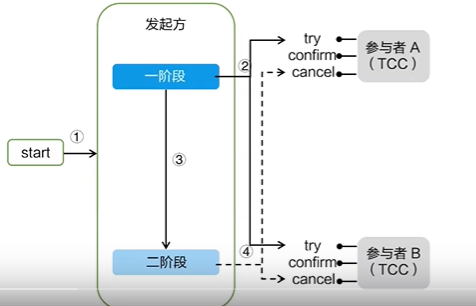


利用sql完成事务的提交

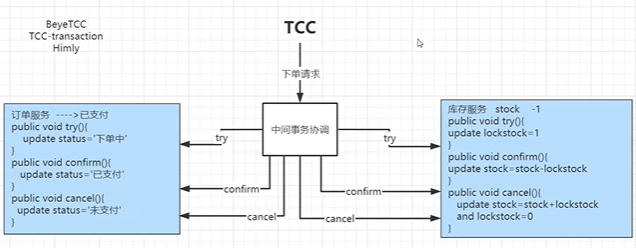
### TCC （try confirm cancel）

1. 侵入性比较强并且自己实现的相关的事务控制逻辑
2. 整个过程中基本上没有锁的一个概念，性能更强。

TCC模式用户根据自己的业务场景实现Try，Confirm和Cancel三个操作；事务发起方在一阶段执行Try方式，在二阶段提交执行Confirm方法，二阶段回滚执行Cannel方法。



业务场景



## MQ可靠消息最终一致性方案

### RocketMQ消息中间件

