分布式事务的几种解决方案

标签：比较 分布式 有一个 检查 public 数据库

前言

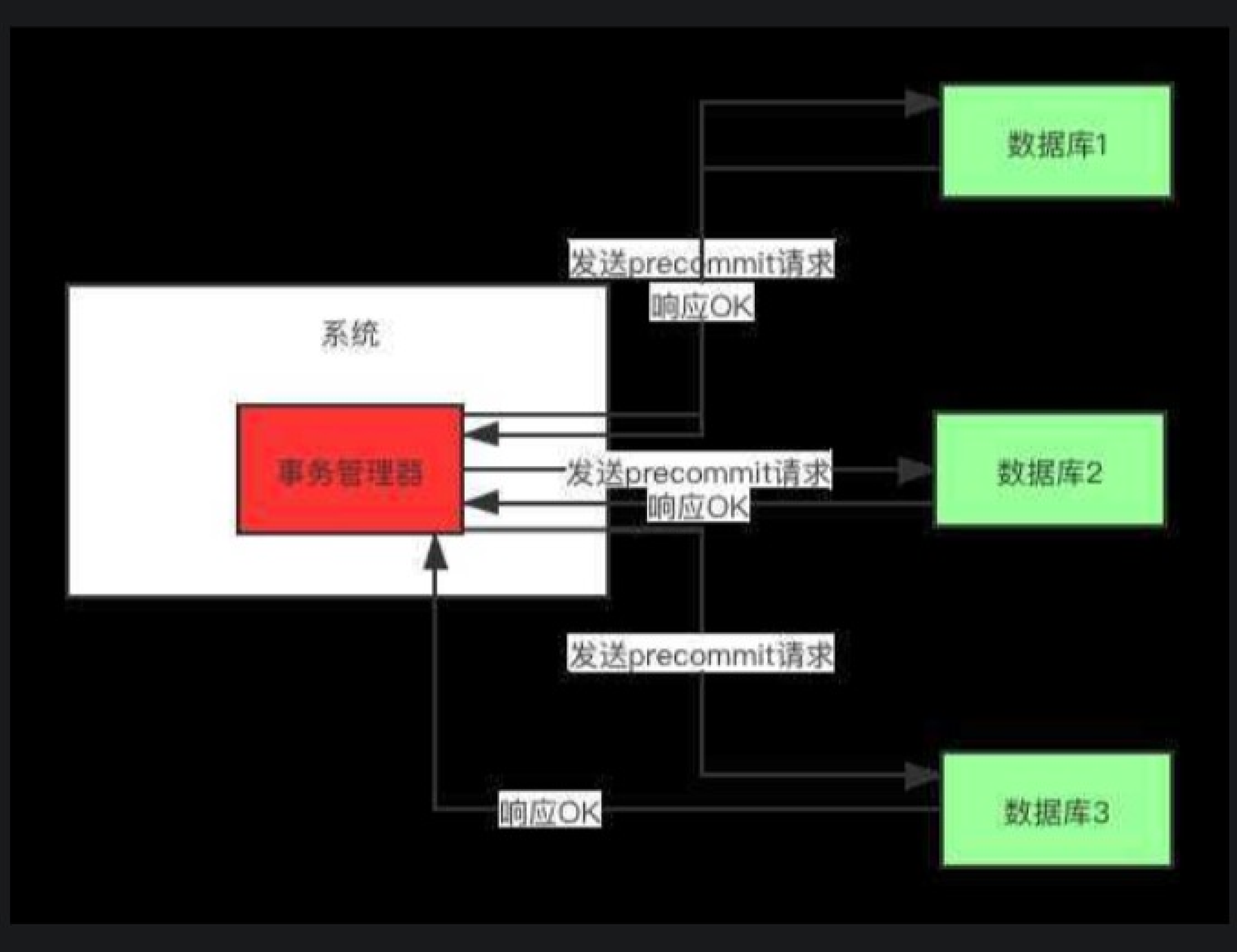
????随着现在分布式，微服务的普及，怎样保证微服务之间的数据一致性就成了一个很大的问题，也就是怎样解决分布式事务。不像之前系统都是单点的，操作的都是同一个数据库，这样系统对数据库的操作都可以放在一个事务中，并不需要跨系统调用服务。而分布式的出现，一个大型的系统下面可能会有多个子系统模块，这时候就会出现跨系统调用，这时就会出现一个问题，如果我本地系统事务执行正常，而我去调用系统A的时候系统A出现异常，就会导致我们两边的数据出现不一致的情况。下面主要讲一下几种常见的解决分布式事务的方案

XA方案/两阶段提交方案

????XA方案也被称为两阶段提交，基于2PC理论实现的。是有个事务管理器的概念，事务管理器负责协调多个数据库的事务。在XA方案中分为两阶段：

????????第一阶段：事务管理器首先向各个数据库发送一个precommit预提交操作，然后由各个数据库反馈是否可以正式提交事务

????????第二阶段：事务管理器收到各个数据库的反馈，如果各个数据库都响应ok，则表明可以提交commit，如果有一个数据库回答不ok，那么事务管理器就会发送回滚事务请求



????XA方案应用场景可以在sharding-jdbc中有体现，sharding-jdbc是用来分库分表的工具，必然就会存在多个数据库，多个数据库源，需要保证多个数据库中的事务要么都成功，要么都失败，可以通过看个demo了解一下：

```
@ShardingTransactionType(TransactionType.XA)
@Transactional(rollbackFor = Exception.class)
public void testTransaction() {
  // step1: 先查询库1中的user 然后进行更新操作
  // step2: 在查询库2中的user 然后进行更新操作，更新失败，看两个库的数据会不会进行回滚
  try {
    // step1
    User user1 = userMapper.selectUserById(1);
    user1.setUsername("测试事务1");
    userMapper.updateUser(user1);
    // step2
    User user0 = userMapper.selectUserById(2);
    user0.setUsername("测试事务2");
    userMapper.updateUser(user0);
    int result = 1/0;
  } catch (Exception e) {
    log.error(e.getMessage(), e);
    TransactionAspectSupport.currentTransactionStatus().setRollbackOnly();
    log.info("报错，事务回滚");
  }
}
```

????我们从代码中可以看到sharding-jdbc提供了@ShardingTransactionType注解，传入的事务类型是XA，try{}中分别从两个数据库中查出了user1,user0。而更新了user1对象，user0对象。最后result=1/0抛出异常，catch(){}中则采用TransactionAspectSupport回滚操作。此时应该两个库中的user对象应该被回滚，网友可以去试下，写一个sharding-jdbc针对分布式事务的demo

????XA方案在企业应用中还是用的比较少，因为XA方案还是只是在单个系统中，并没有出现跨系统间的接口调用，比较适合单块应用里，跨多个库的分布式事务。而且还严重依赖于事务管理器，一旦执行到第二个阶段，事务管理器宕机了，数据库就会一直等待commit请求，从而被阻塞住。还会出现一个问题就是：各个数据库之间数据不一致，加入数据库1和数据库2收到了commit请求，而数据库3因为网络原因没有收到commit请求，这时就会出现数据库3与其他两个库之间的数据不一致

TCC 方案

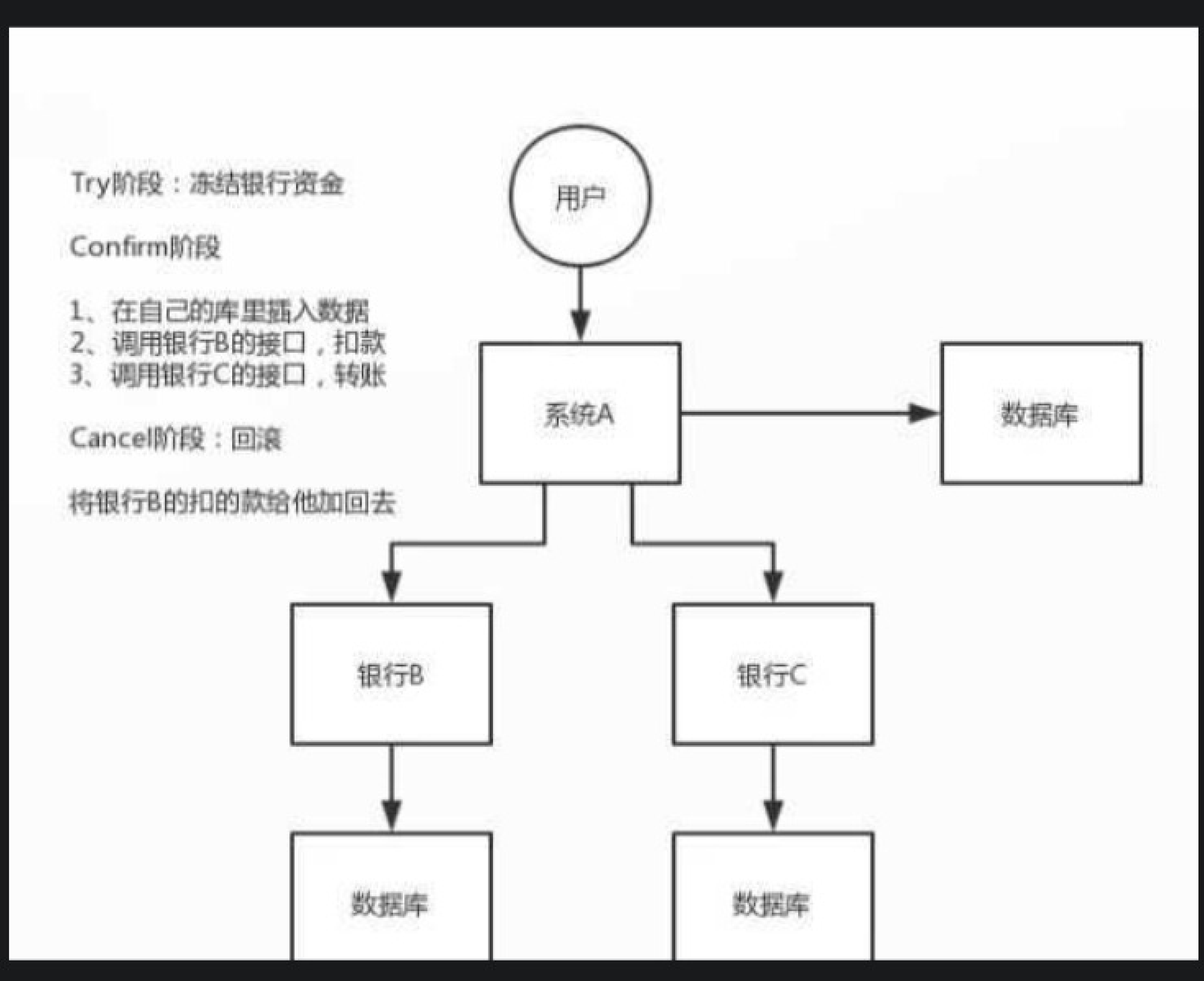
????TCC全称是：Try、Confirm、Cancel

????????**Try阶段：是对各个服务的资源做检查以及对资源进行锁定和预留，比如我要支付100元的物品，首先需要检查你的账户是否够100元进行支付，如果够则进行锁定**

????????**Confirm阶段：这个阶段说的是各个服务中执行实际的操作，我支付了100元，那么我的银行账户需要扣减100元，商家账户就需要增加100元**

????????**Cancel阶段：如果任何一个服务的业务执行操作失败，这里就需要将成功的进行回滚，我的账户扣减100元成功，商家账户增加100元失败，那么成功扣减100元也需要进行回滚**

????这种方案用的也比较少，主要是后面如果出现失败，需要自己手动进行回滚，严重依赖于自己写的回滚代码，但是一般涉及到钱，支付的场景，TCC方案用的比较多，需要严格保证分布式事务要么全部成功，要么全部失败，严格保证钱数据的一致性，还有就是各个业务之间执行的时间比较短，不然会出现资源一直被锁定状态



本地消息表

????本地消息表大概的意思就是在数据库中建立一张消息表，这张消息表维护执行的事务状态信息，大概的步骤：

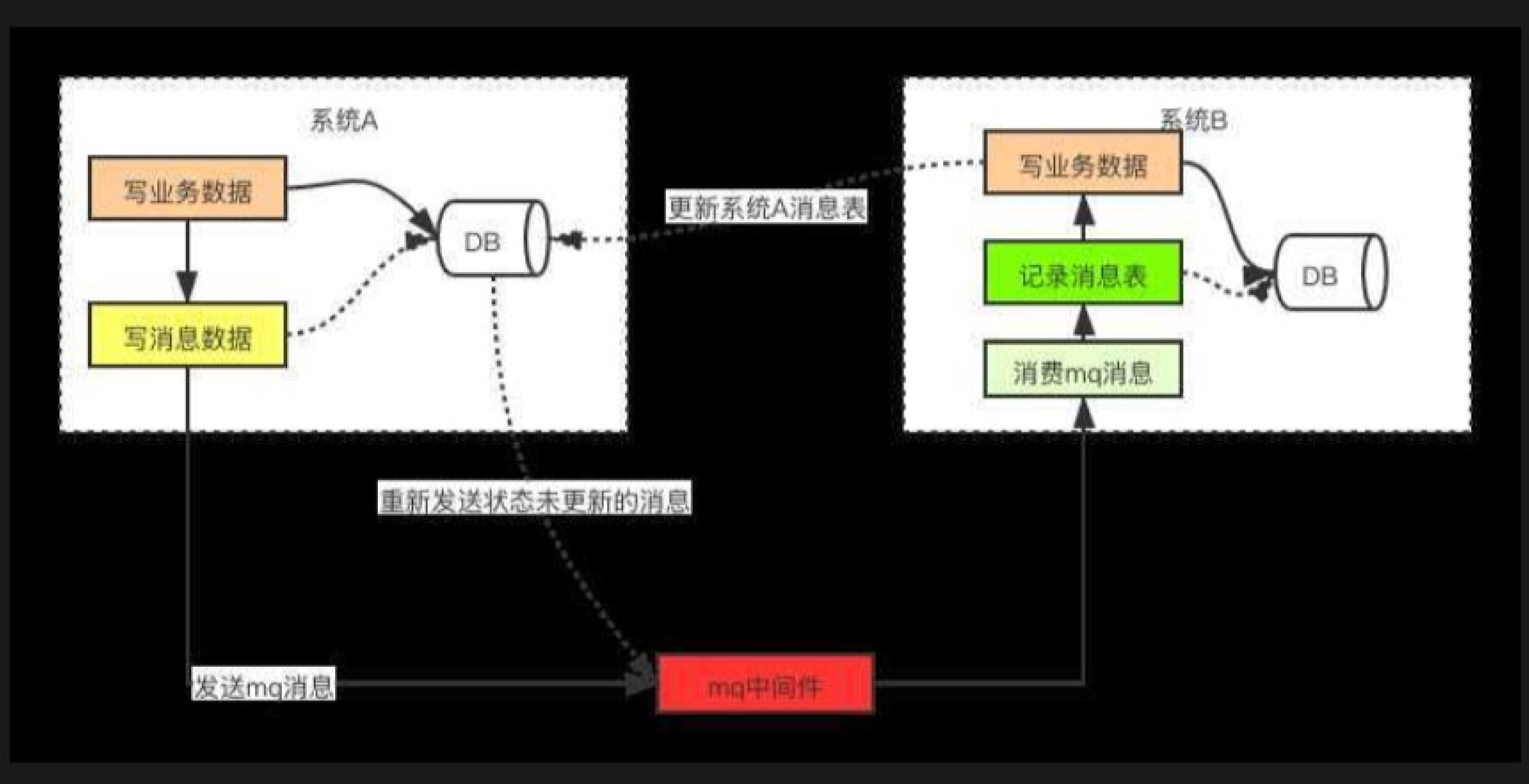
????????1. **系统A在执行本地事务的同时，会向消息表中插入一条数据**

????????2. **接着系统A如果需要请求系统B，这时就会发送一条消息到mq**

????????3. **系统B接受到了系统A发送过来的mq消息，首先会在自己的本地消息表中插入一条数据，同时执行其他业务操作，如果执行成功，则会更新自己的本地消息表的状态和更新系统A的消息表的状态，表示自己处理成功**

????????4. **如果系统B执行失败，则不会更新自己的本地消息表和系统A的消息表的状态，那么系统A会不断的轮询扫描自己的消息表，看那些消息状态没有被更新过来，消息状态没有被更新过来的，会再次发送mq消息给系统B消费，让系统B再次处理**

????本地消息表是基于BASE理论实现的，保证了最终一致性，哪怕系统B执行失败了，也会一直重发消息，直到系统B执行成功。适用于对一致性要求不高的场景，还需要注意的一点就是重试消费mq消息的幂等性



????本地消息表主要的问题在于严重依赖于数据库的消息表来管理两边的事务，如果碰上高并发场景，数据库会成为一个瓶颈，扩展性不是很高，本地消息表用到的场景也比较少

可靠消息最终一致性

????可靠消息最终一致性方案跟本地消息表不同的是直接把消息表砍掉，直接依赖于MQ来实现事务，像阿里的RocketMQ就支持事务，大致的意思就是：

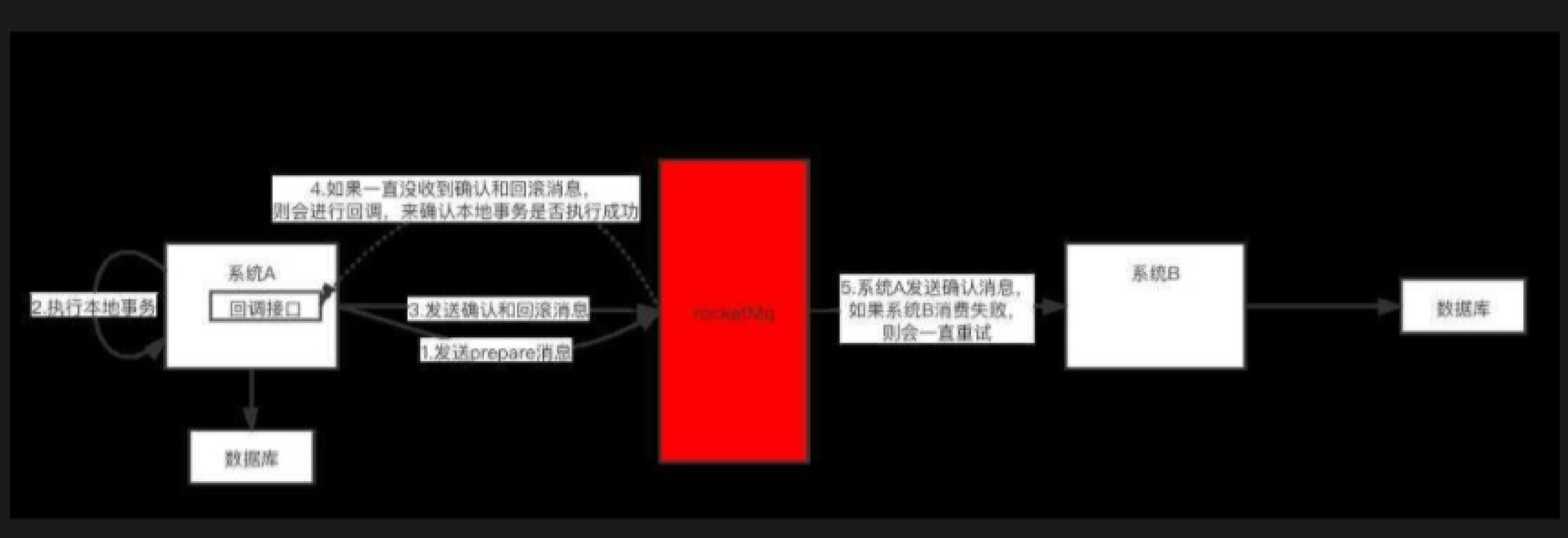
????????1. **首先系统A会发送一个prepare消息到mq，如果prepare消息发送失败，后面的操作也别执行了**

????????2. **系统A发送prepare消息成功，开始执行本地事务，如果本地事务执行成功则告诉mq发送确认消息，如果本地事务执行失败，则告诉mq执行回滚消息，需要将prepare消息回撤掉**

????????3. **如果是发送确认消息，系统B接受到mq发送过来的消息，执行本地的事务，如果系统B执行本地事务失败，自动不断重试直到成功。如果还是一直不成功，则发送报警由人工来手工回滚和补偿，只能人工手动的修数据。**

????????4. **假设mq接受到了系统A发送过来的prepare消息，但是一直没收到确认发送还是回滚操作，那么mq会定时的轮询所有prepare消息来回调系统A的接口，来检查系统A执行本地事务的时候是不是失败了，自己在回调接口中的业务可以自定义如果本地事务执行失败，那么发送到mq的prepare消息也进行回滚操作**

????这种方案在企业应用中还是使用的比较多的，一些大型互联网企业都是依赖于mq来实现最终消息的一致性，像rocketmq本身就支持事务，提供了TransactionListener接口，我们只需要实现接口中的checkLocalTransaction的方法，也就是mq没有收到确认发送还是回滚的操作需要回调的接口。虽然RabbitMQ、ActiveMQ没有现成的支持事务，但是也可以基于rocketmq的思路来封装一套类似的逻辑出来



总结

????上面几种方案需要真正落地的话，实现起来还是比较麻烦的，如果要实现分布式事务，需要考虑到的点还是比较多的，复杂度也比较大。能不用分布式事务的话就不用，如果非得使用的话，看能不能使用一些补偿机制去实现，最后再结合自己公司的业务分析，来看看哪一种方案适合自己的业务

分布式事务的几种解决方案

标签：比较 分布式 有一个 检查 public 数据库

原文：https://www.cnblogs.com/semi-sub/p/13172812.html

请输入关键词 搜索

分享档案 更多>

2021年04月01日 (328)
2021年03月31日 (342)
2021年03月30日 (655)
2021年03月29日 (666)
2021年03月28日 (659)
2021年03月27日 (507)
2021年03月26日 (120)
2021年03月25日 (30)
2021年03月24日 (24)
2021年03月23日 (31)

最新文章 更多>

1. 基于MATLAB的手写公式识别(9) 2021-04-01
2. 面试题-python3 找出一个字符串中所有子串是回文的次数 2021-04-01
3. 阿里最强 Python 自动化工具开源了！ 2021-04-01
4. 弦长公式 2021-04-01
5. 寻找无序数组中的前k大元素 2021-04-01
6. pm2 管理服务 2021-04-01
7. QueryWrapper的关键字查询匹配多个字段 2021-04-01
8. inline， block， inline-blockd的区别 2021-04-01
9. SEO工具ahrefs各参数详解 2021-04-01
10. python字典 2021-04-01

教程昨日排行 更多>

1. MySQL 及 SQL 注入
2. FILE *fopen(const char *filename, const char...
3. HTML 框架
4. JavaScript 比较
5. HTML DOM 修改 HTML 内容
6. jQuery 效果方法
7. jQuery UI 简介
8. jEasyUI 表单的 CRUD 应用
9. SQL SELECT DISTINCT
10. MySQL 连接

友情链接

汇智网 PHP教程 插件网 布布教程

关于我们 - 联系我们 - 留言反馈