问题1：tcc分布式事务下，主业务和从业务必须有事务管理器。目前我们系统基于spring的事务管理器。如果从零实现tcc，是否能保证我们对spring事务管理器的掌握程度足够？如果一个业务涉及多个事务或者嵌套事务，将如何处理？

举第一个例子，spring事务管理器在接受异常后，默认回滚数据库的写入操作。那么当我们的业务在try阶段异常时，如何保证spring事务管理器不回滚数据库的写入操作而是直接进入我们定义的cancel阶段的回滚操作？

举第二个例子（假设第一个例子我们已经解决）。以后我们将在业务表基础上引入流水表。假设业务已经实现了tcc，当一个业务发起时，流水表写入初始状态【处理中】，业务表在try阶段写入状态【准备中】。当业务在try阶段异常回滚后，如何管理流水表的状态？

举第三个例子，与第一个例子类似，一个业务的事务逻辑为A(REQUIRED)->B(REQUIRES\_NEW)->C(REQUIRED)，A的事务已经提交，数据库已经修改，但B由于异常需要回滚。如何保证嵌套事务的异常处理优先进入我们自定义的tcc的cancel阶段而不是进入spring默认的事务管理器进行了回滚？

问题2：假设我们已经完全解决了问题1，接入了spring事务框架。我们如何实现以下机制？

Try阶段发生异常，数据库事务未提交，我们需要直接回滚try阶段任务而不去调用cancel；try阶段发生异常，数据库事务提交成功，我们需要调用cancel操作。

问题3：tcc分布式事务下，由于多个业务之间相互调用，可能存在一段时间网络宕机、故障之类的问题。当网络恢复后，异步任务需要继续完成已有的tcc事务或者有详细的日志记录以供人工介入。如果从零实现tcc，是否能保证我们对异步任务、消息中间件等知识的掌握程度足够，以完成故障恢复后，保证数据正常？

举第一个例子，如何设计tcc事务日志？

举第二个例子，在confirm/cancel阶段中服务宕机，服务恢复后，如何保证继续进行confirm/cancel事务的提交？如果不提供服务自动恢复的能力，如何保证日志信息完善记录而不是丢失在内存中？

问题4：tcc分布式事务需要提供幂等性保障。即一次分布式事务只能被confirm/cancel一次。我们目前的幂等控制由尚未完全掌握的流水表处理，是否能保证流水表、tcc事务下，我们的考虑完善足够？

问题5：tcc分布式事务中，调用try后，再调用cancel。但由于网络等问题，从业务可能先收到cancel，并进行了空回滚，接下来执行了try，如何保证这种事情不会发生？

问题6：tcc分布式事务框架中，只需要对外提供try阶段的接口，如果我们从零实现tcc，那么其他服务与我们对接时，我们如何提供对外接口？或者如何让其他系统对接我们系统？