测试service事务

注意

注意接口Service里面方法的Transaction注解

传播规则回答了这样一个问题：一个新的事务应该被启动还是被挂起，或者是一个方法是否应该在事务性上下文中运行。

1. **Propagation.REQUIRED**

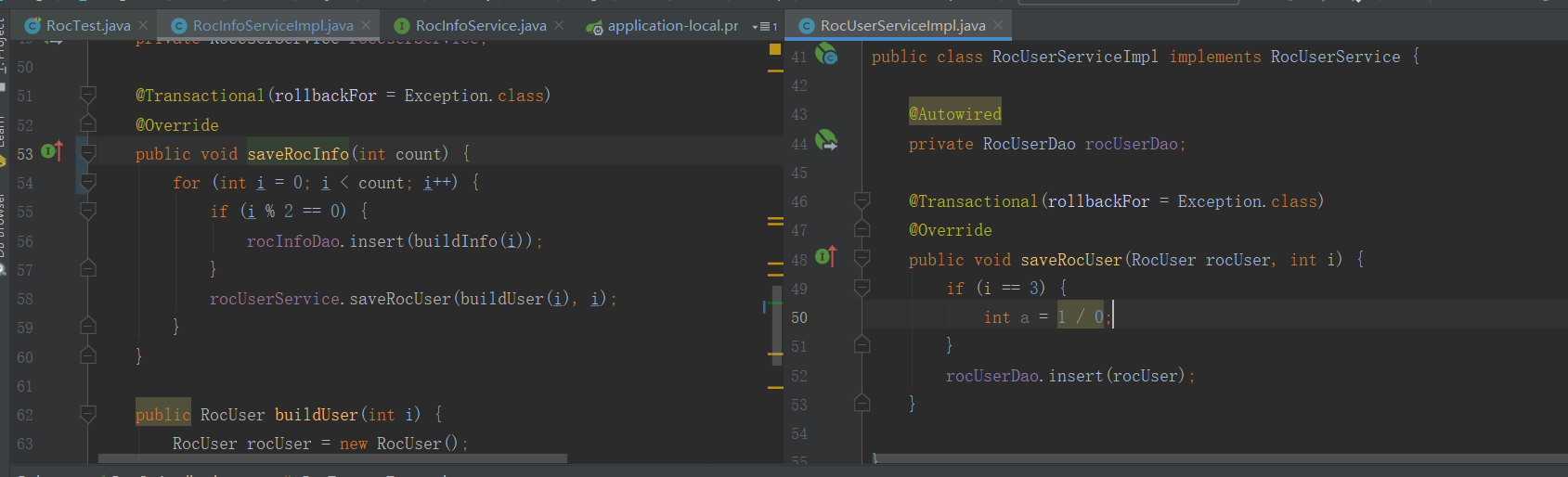
Spring默认的传播机制，能满足绝大部分业务需求，如果外层有事务，则当前事务加入到外层事务，一块提交，一块回滚。如果外层没有事务，新建一个事务执行

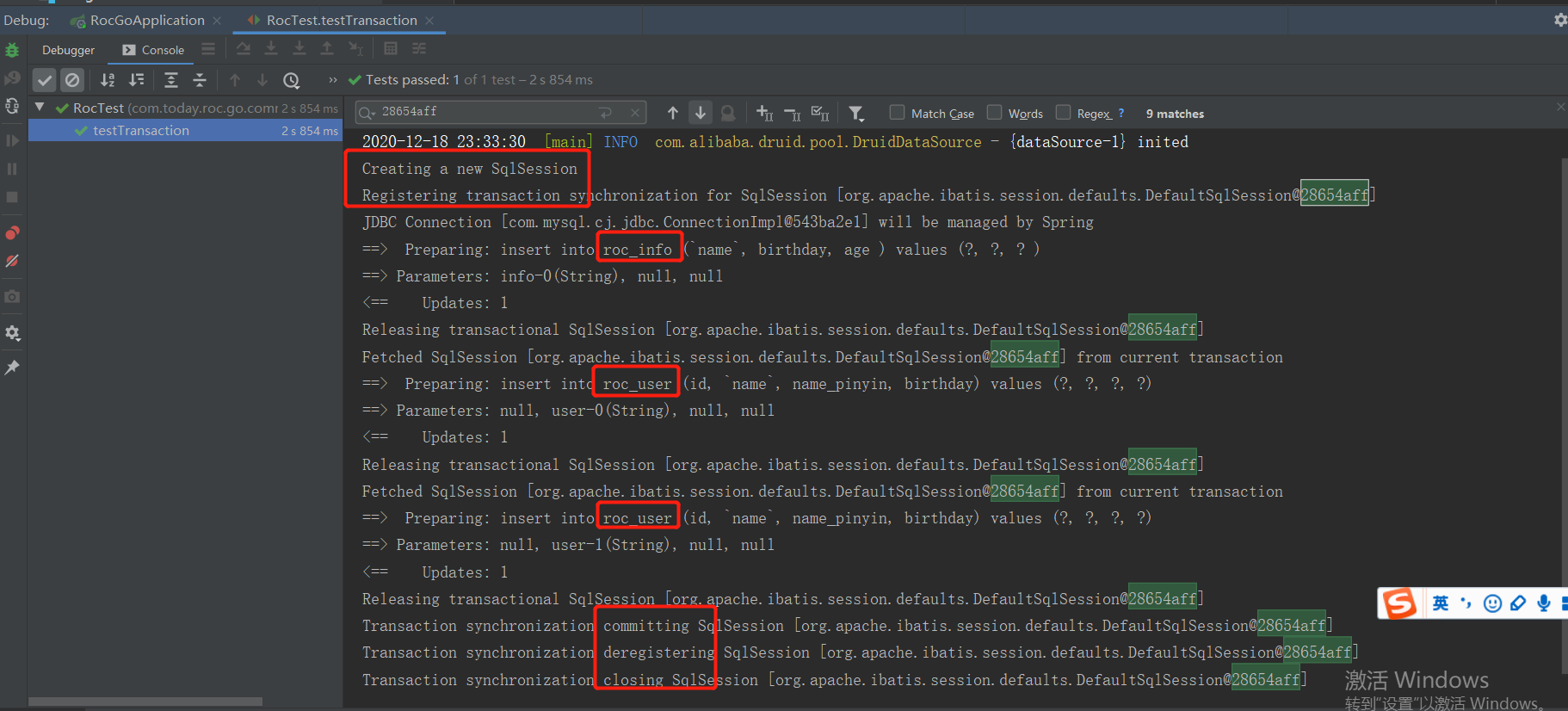
RocInfoService开启事务、RocUserService 开启事务

RocInfoService循环调用rocUserService新增RocUser方法

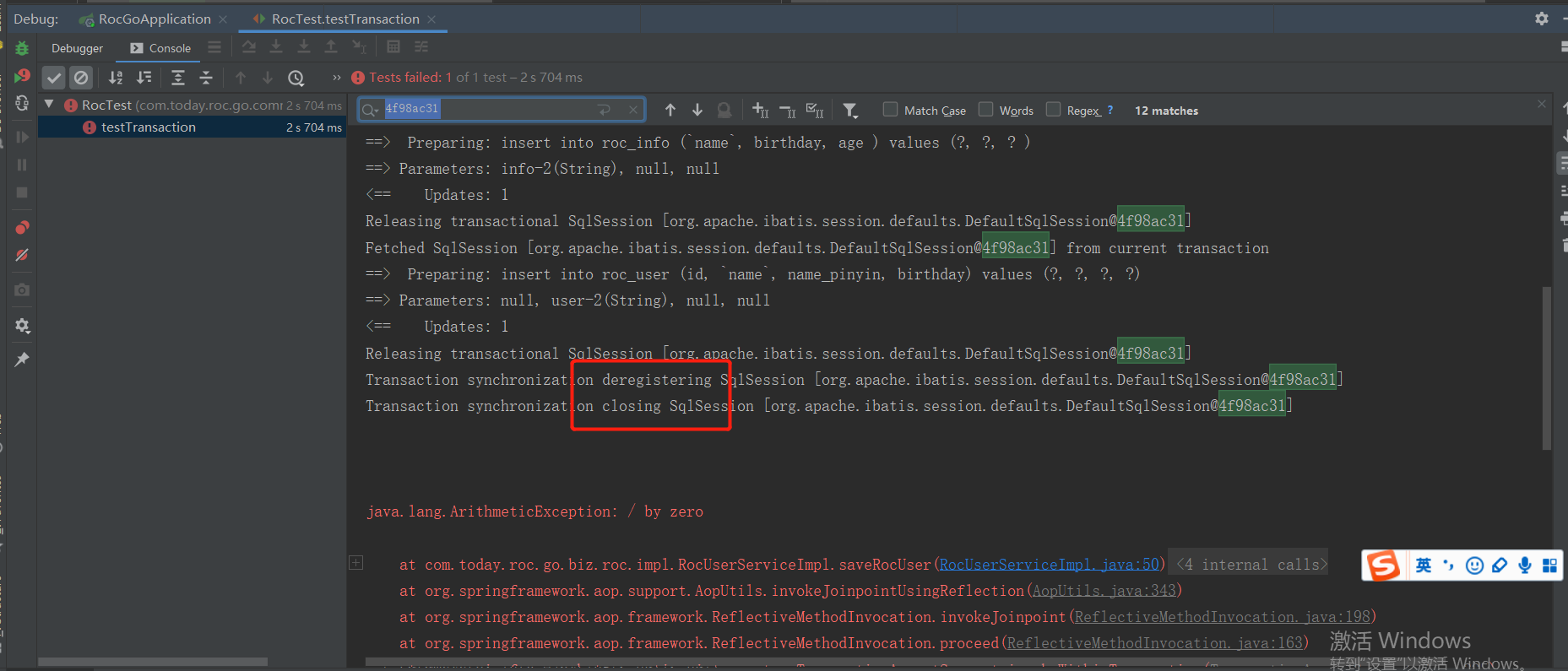
事务是嵌套的

Count = 2

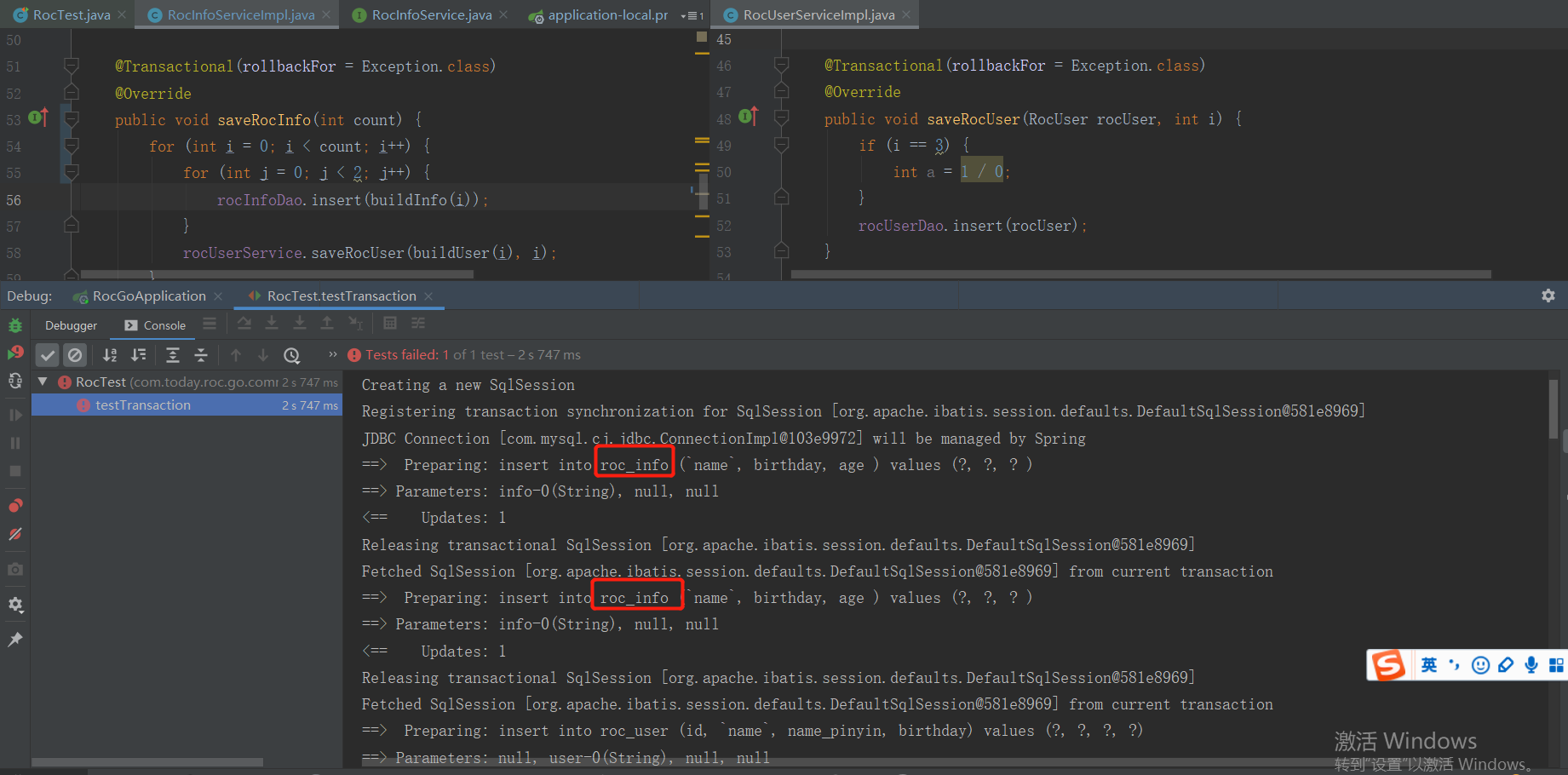




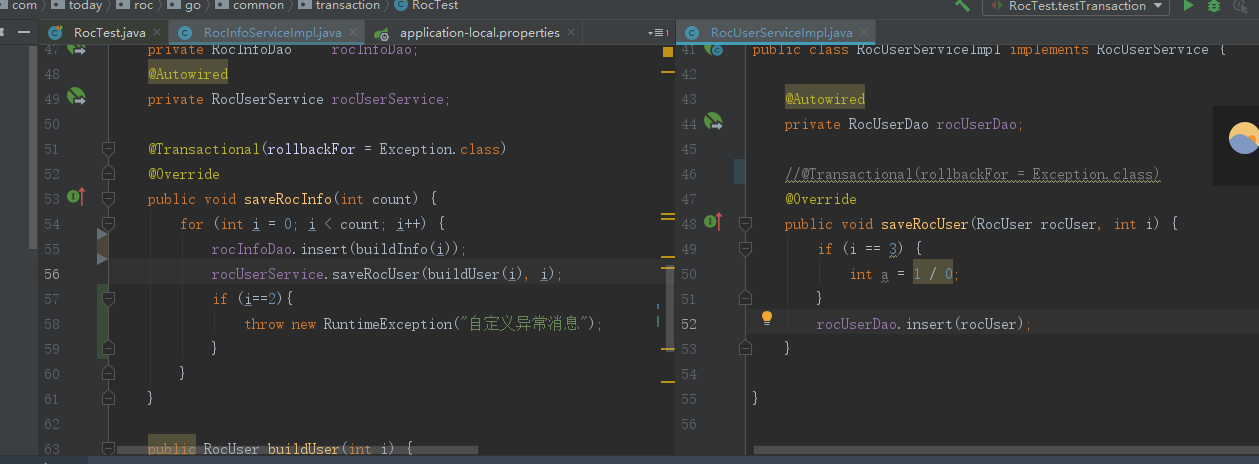
Count =4



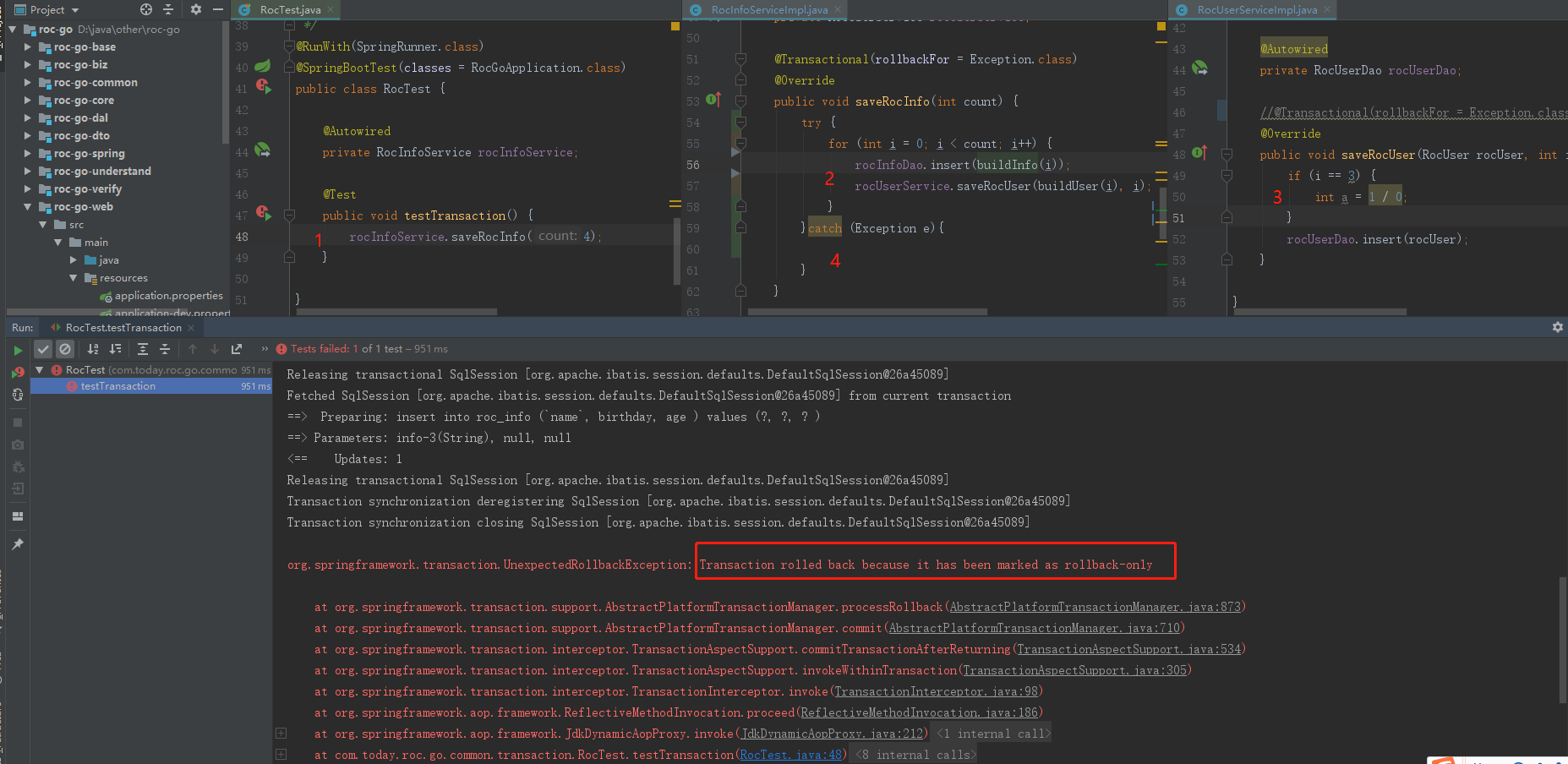
测试Fetched SqlSession



在外层RocInfoService抛出异常



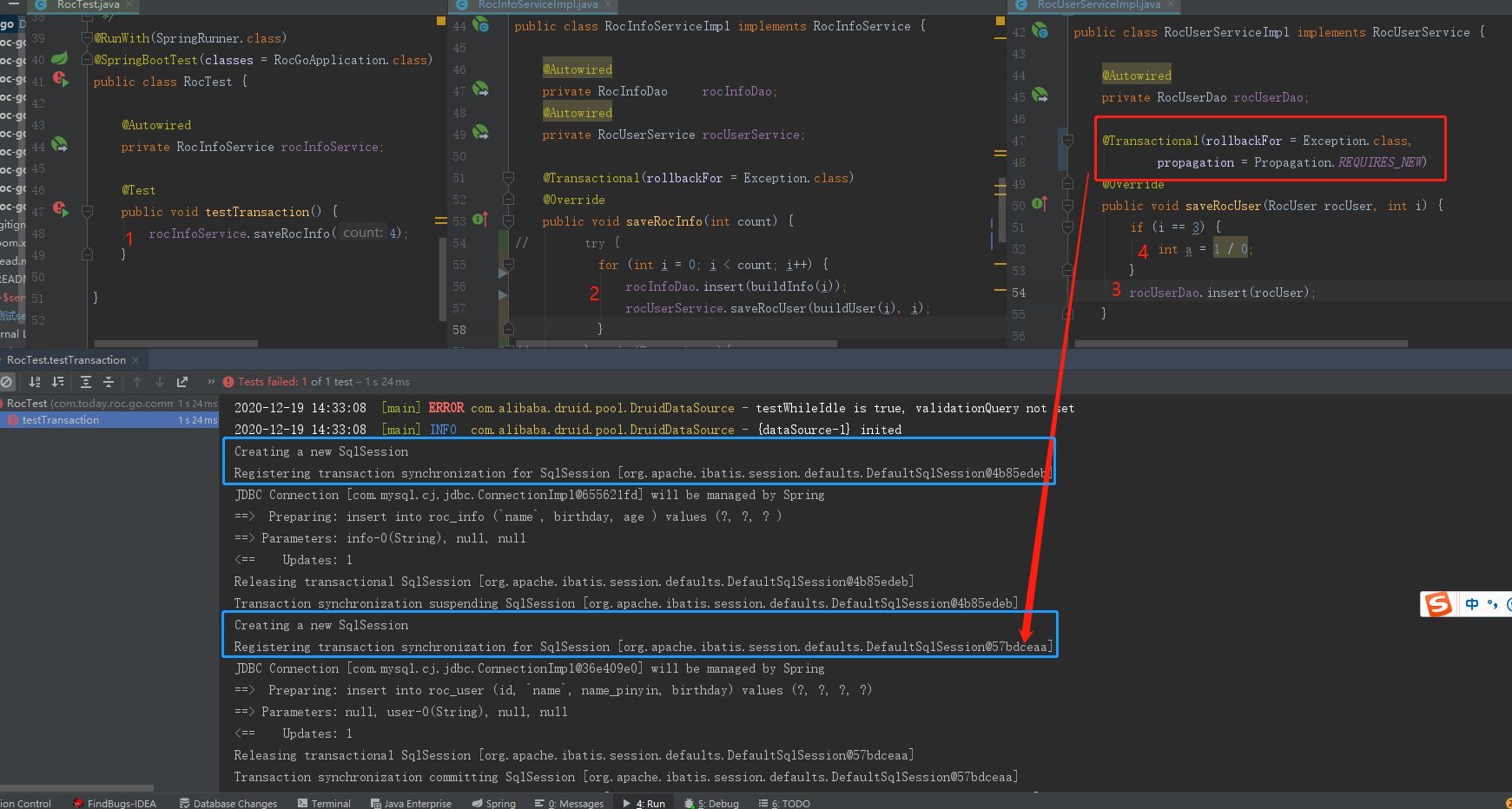
在嵌套事务场景中，内层事务的sql和外层事务的sql会在外层事务结束时进行提交或回滚。如果内层事务抛出异常e，在内层事务结束时，spring会把事务标记为“rollback-only”。这时如果外层事务捕捉了异常e，那么外层事务方法还会继续执行代码，直到外层事务也结束时，spring发现事务已经被标记为“rollback-only”，但方法却正常执行完毕了，这时spring就会抛出“org.springframework.transaction.UnexpectedRollbackException: Transaction rolled back because it has been marked as rollback-only”。

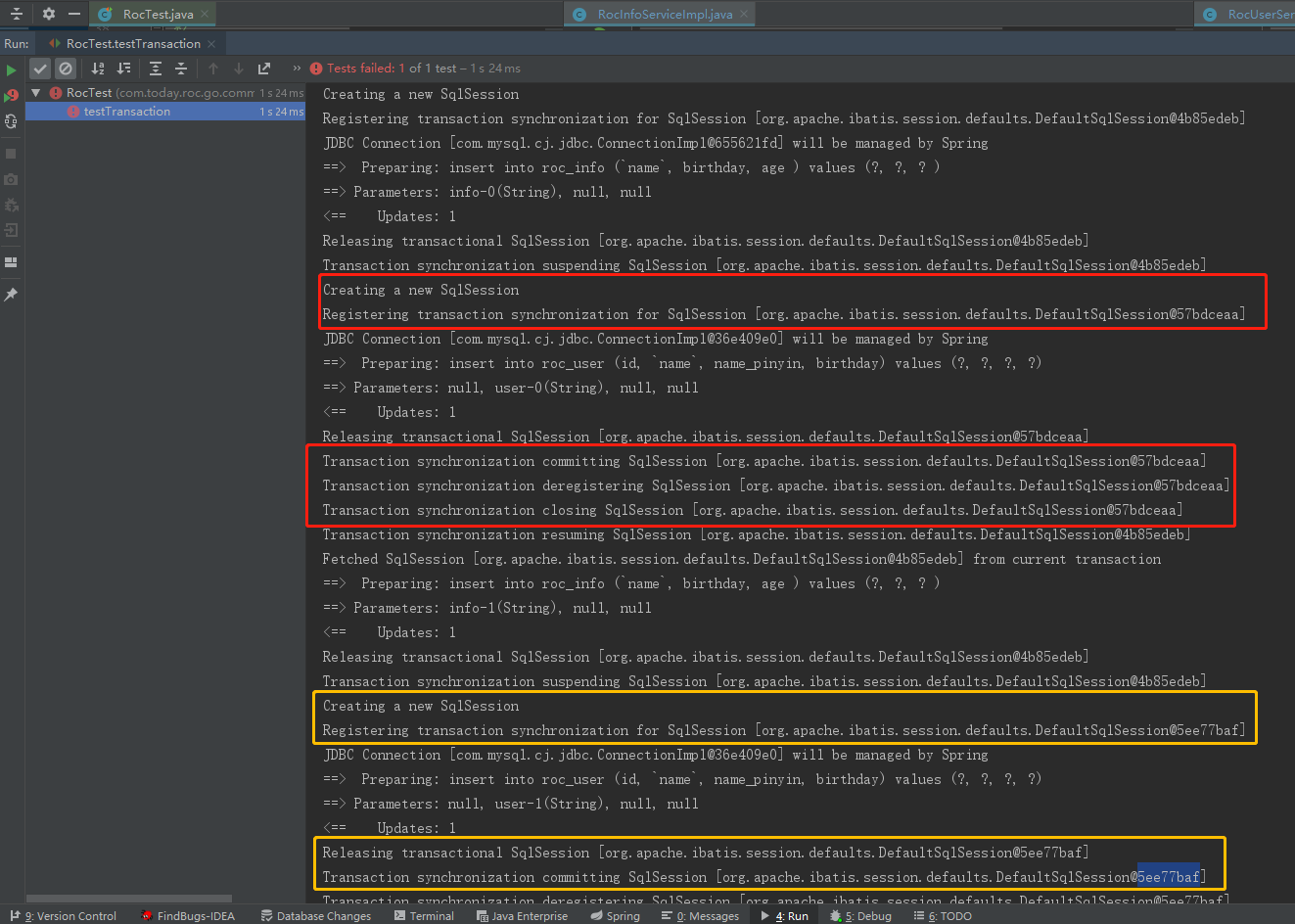


1. **Propagation.REQUIRES\_NEW**

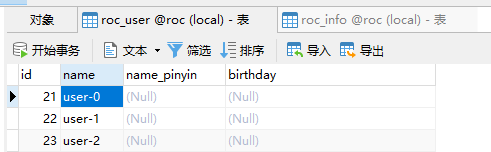
该事务传播机制是每次都会新开启一个事务，同时把外层事务挂起，当当前事务执行完毕，恢复上层事务的执行。如果外层没有事务，执行当前新开启的事务即可

RocUserService每次执行开启一个新事务，每次执行完都commit事务，如果内层事务报错未捕获，外层事务回滚



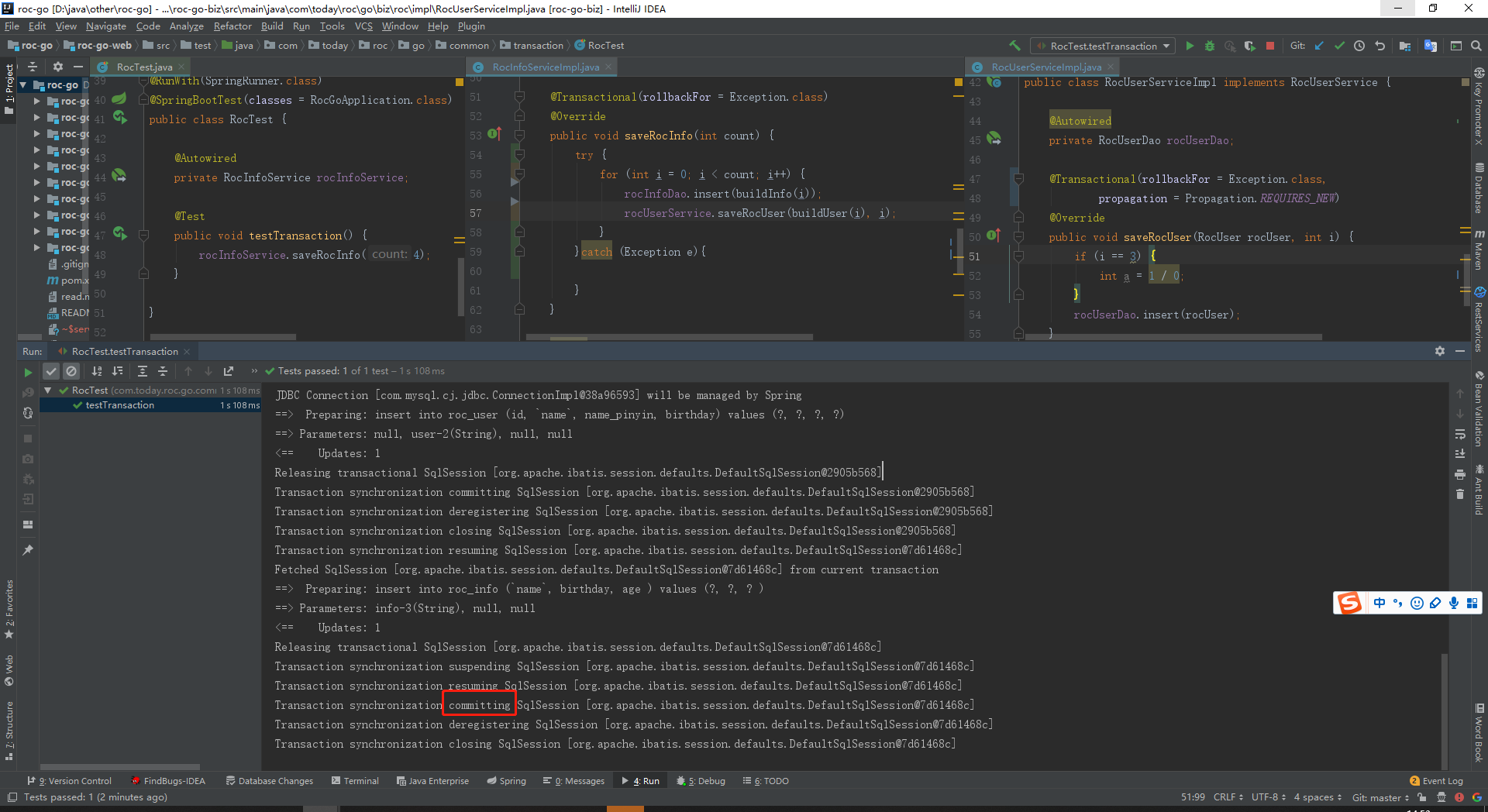


增加三条数据，第四条报错

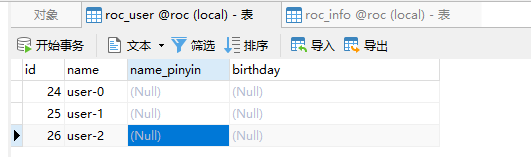


外层数据roc\_info为空

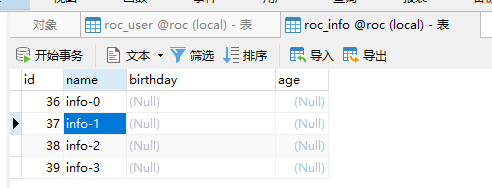
内层事务报错会回滚，如果外层捕获异常，外层事务不回滚，并插入数据



新增三条数据



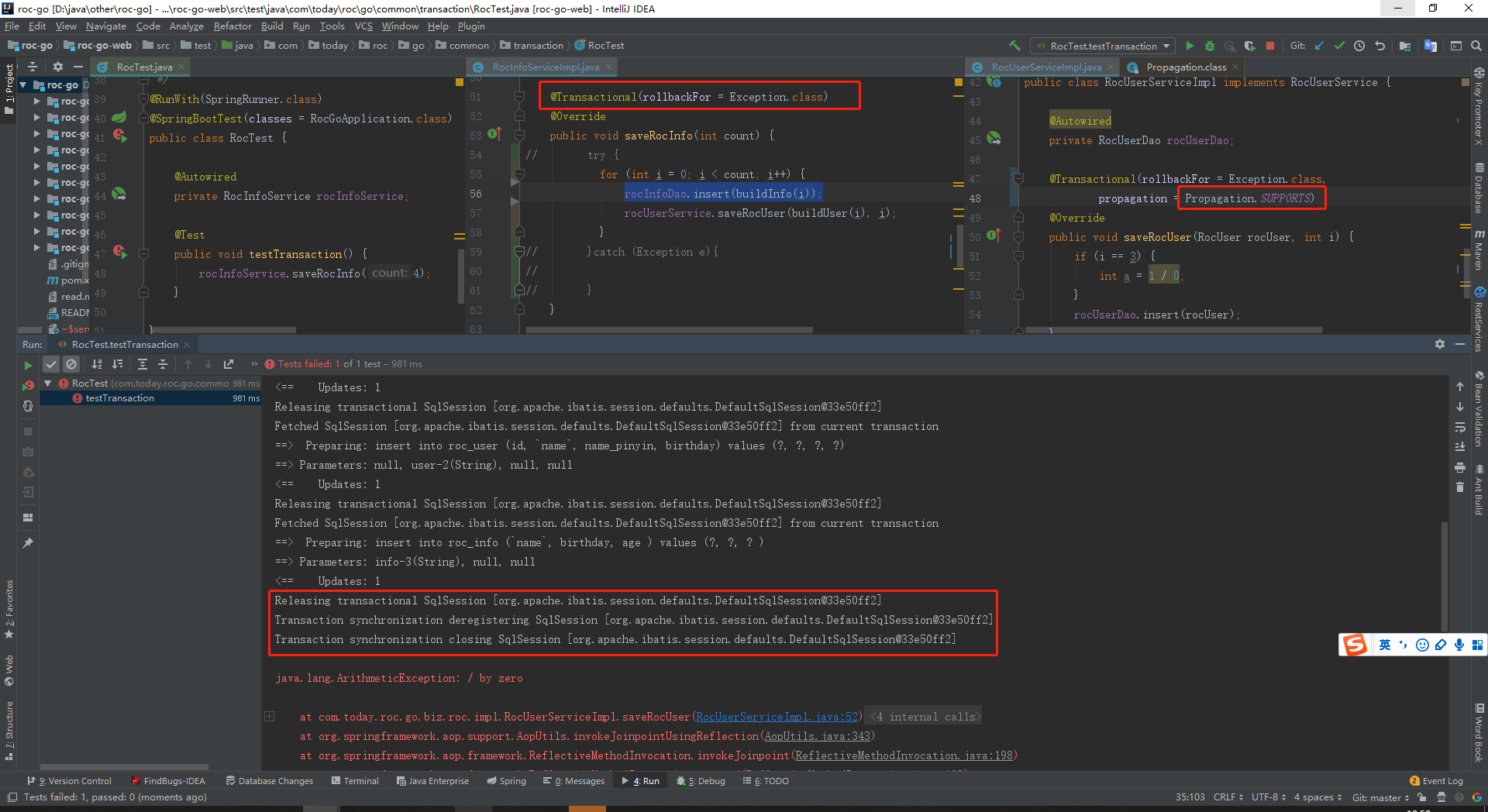
新增4条数据



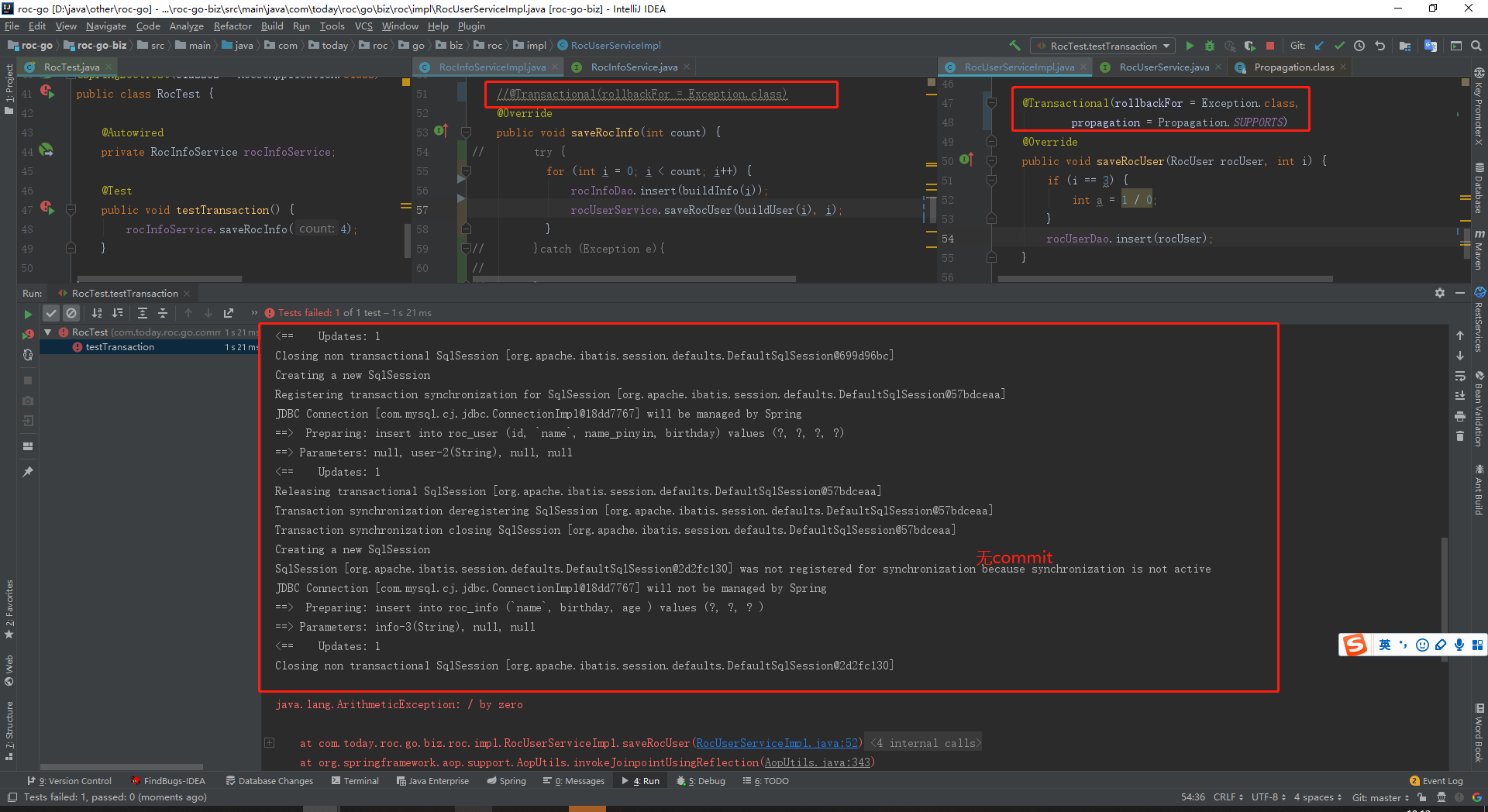
**3、Propagation.SUPPORTS**

如果外层有事务，则加入外层事务，如果外层没有事务，则直接使用非事务方式执行。完全依赖外层的事务

外层有事务，内层使用外层业务



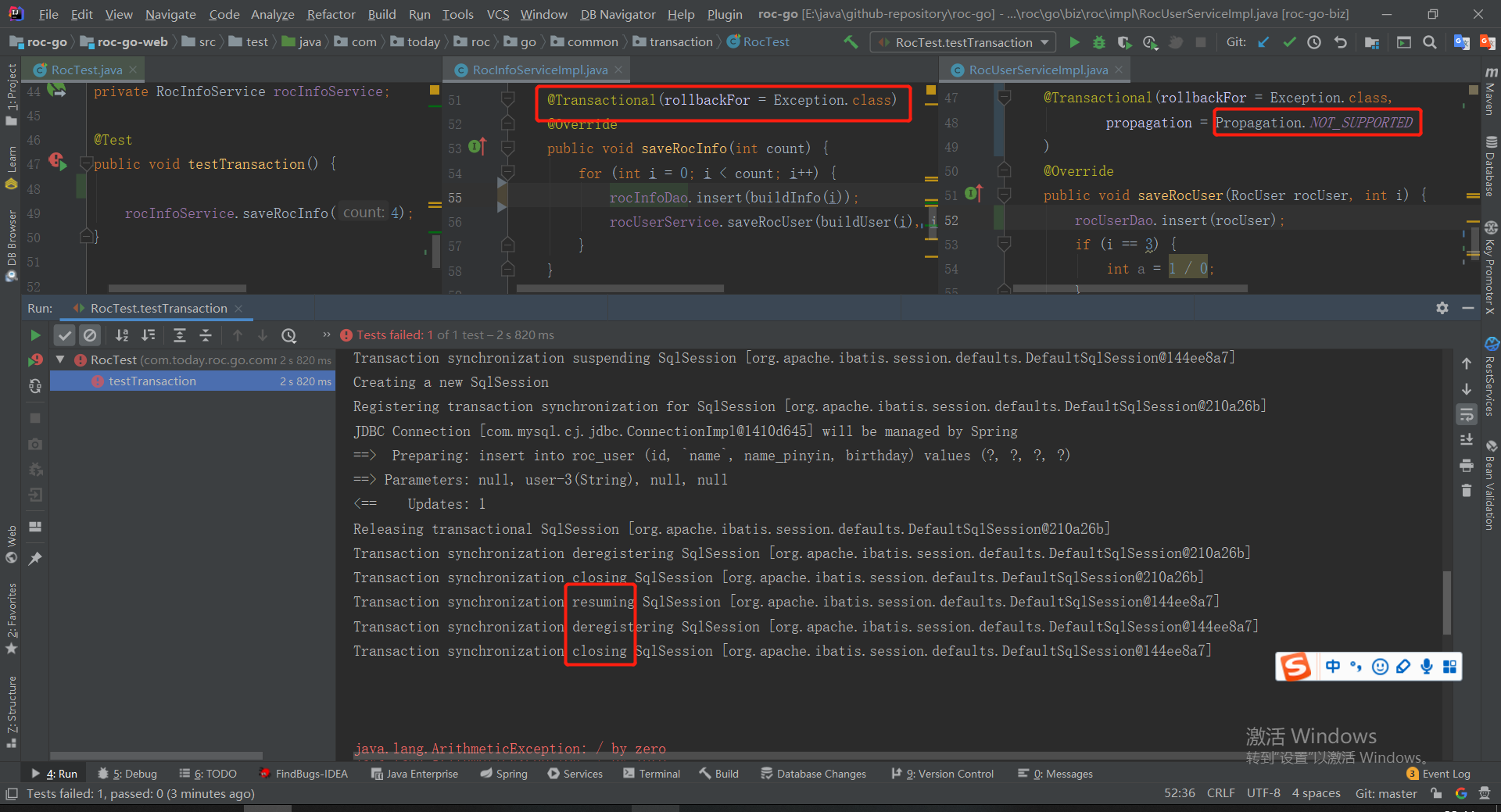
外层没有事务，内层事务事务不作用

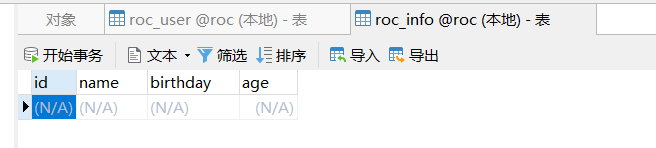
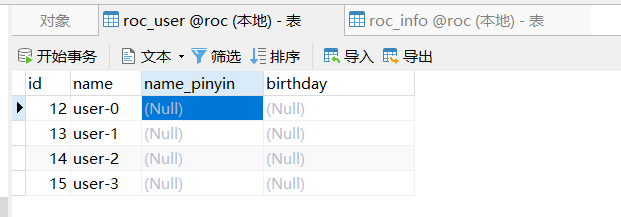


4、Propagation.NOT\_SUPPORTED

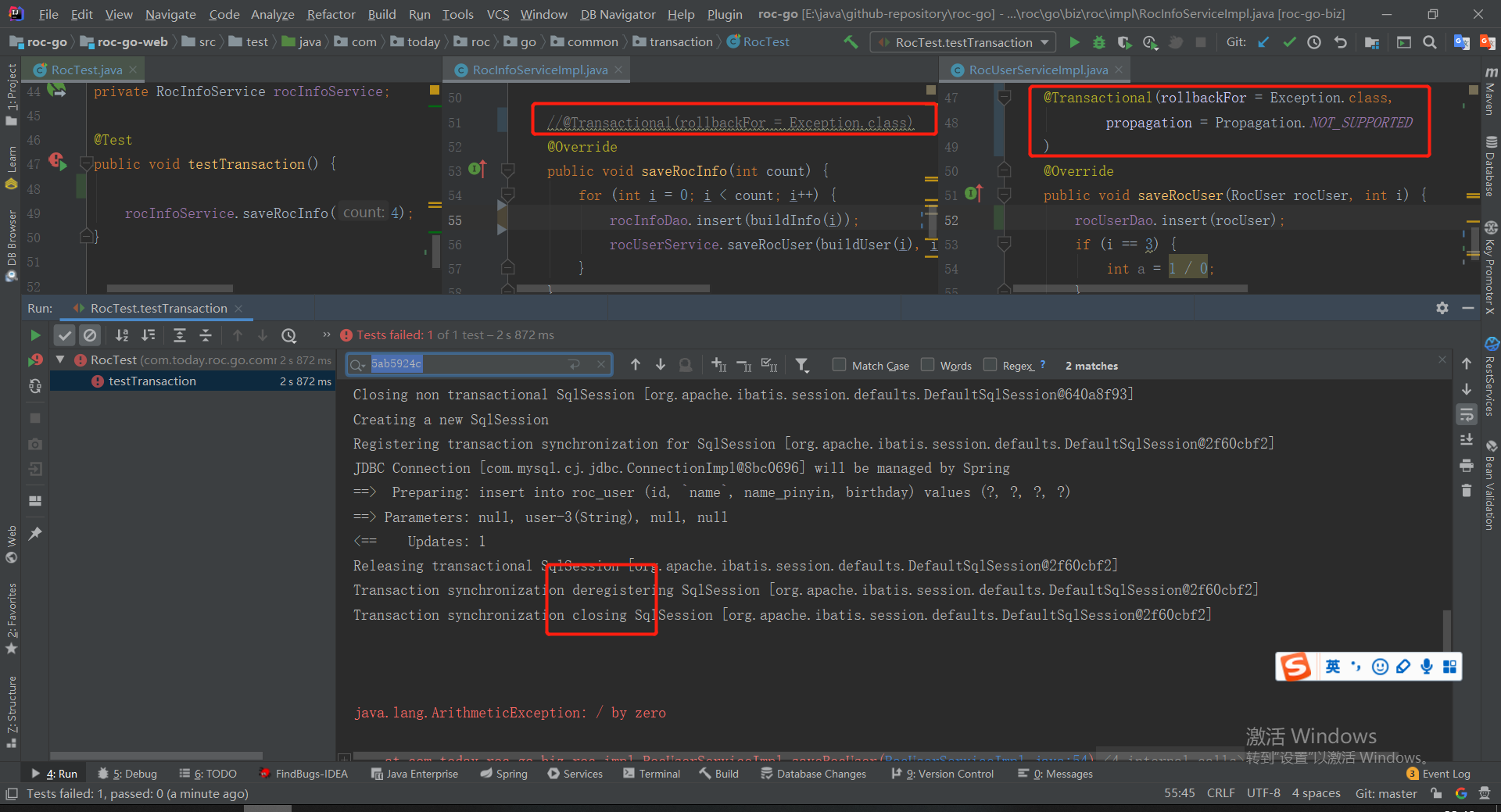
该传播机制不支持事务，如果外层存在事务则挂起，执行完当前代码，则恢复外层事务，无论是否异常都不会回滚当前的代码

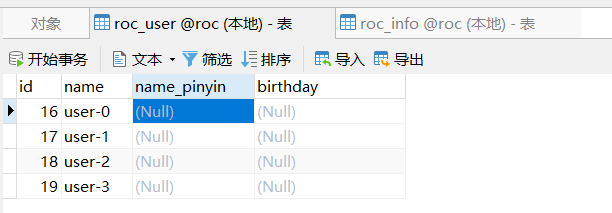
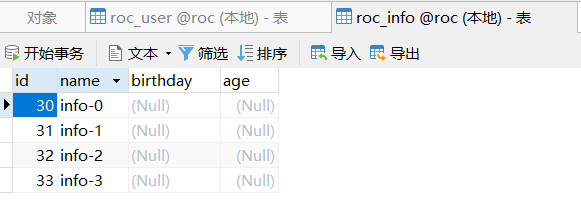
外层有事务，内层事务挂起，内层报错，抛出到外层，内层自身没有事务





外层没有事务，都不受事务影响

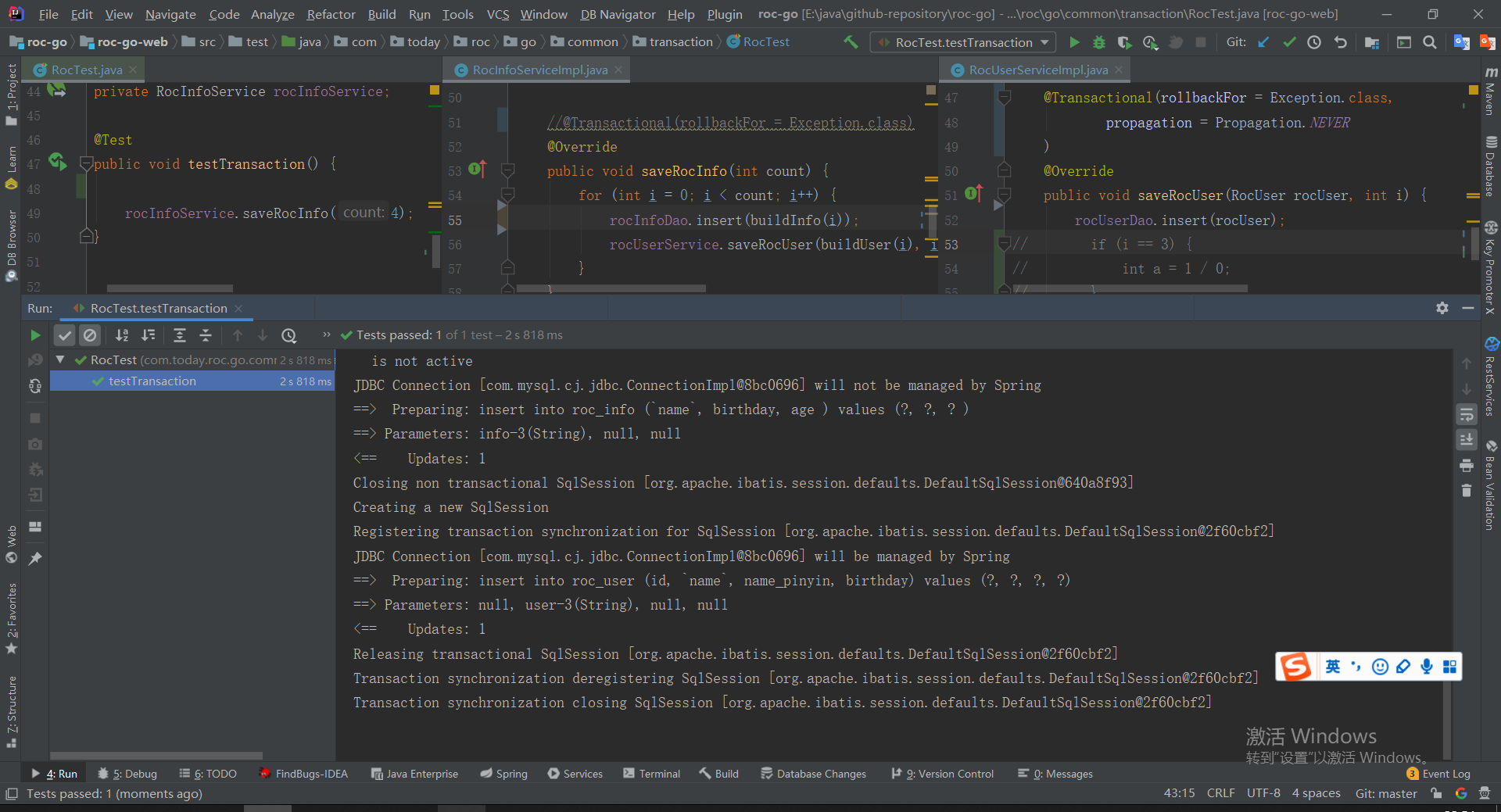




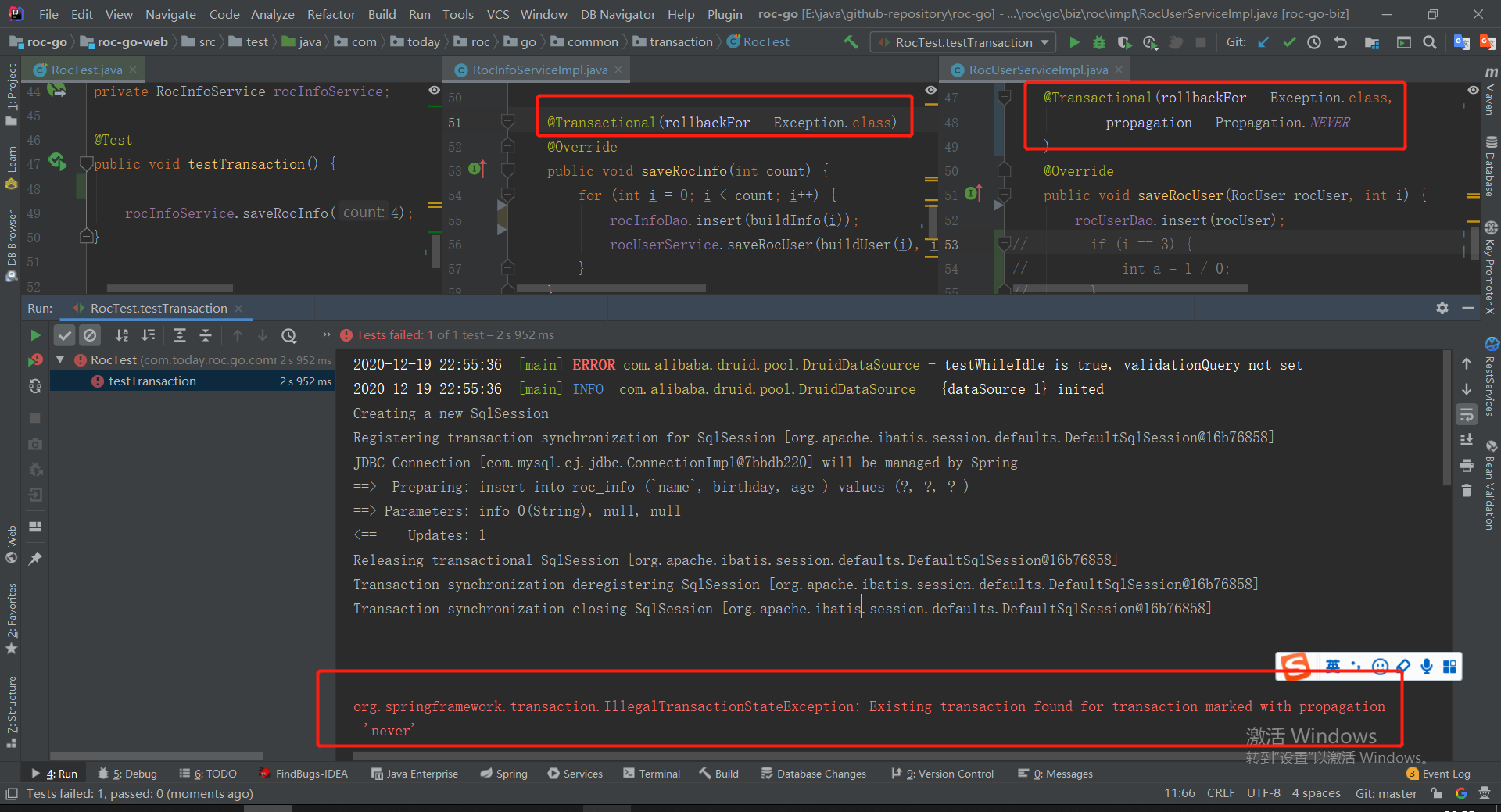
1. Propagation.NEVER

该传播机制不支持外层事务，即如果外层有事务就抛出异常

外层无事务，正常



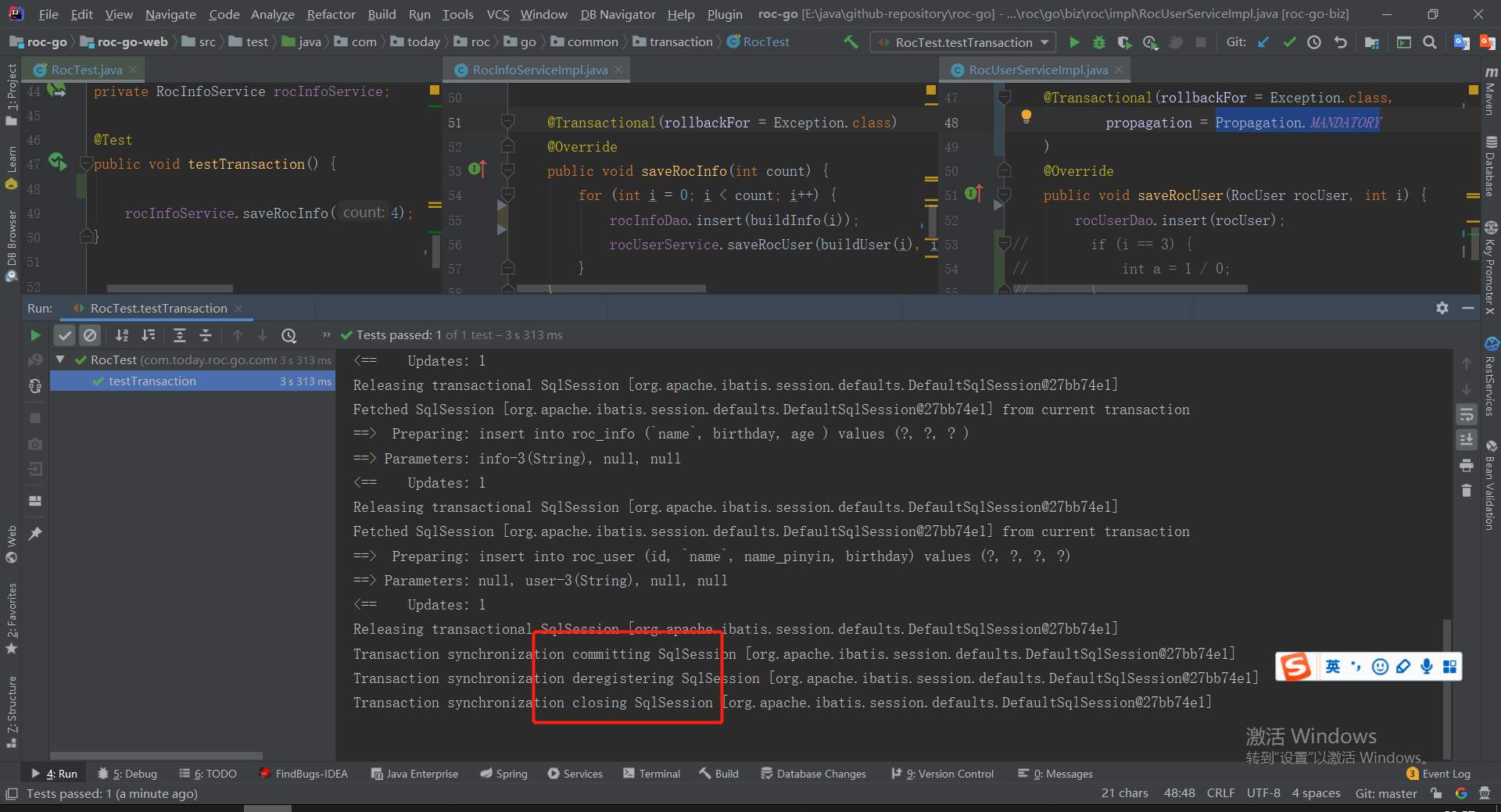
外层有事务，报错



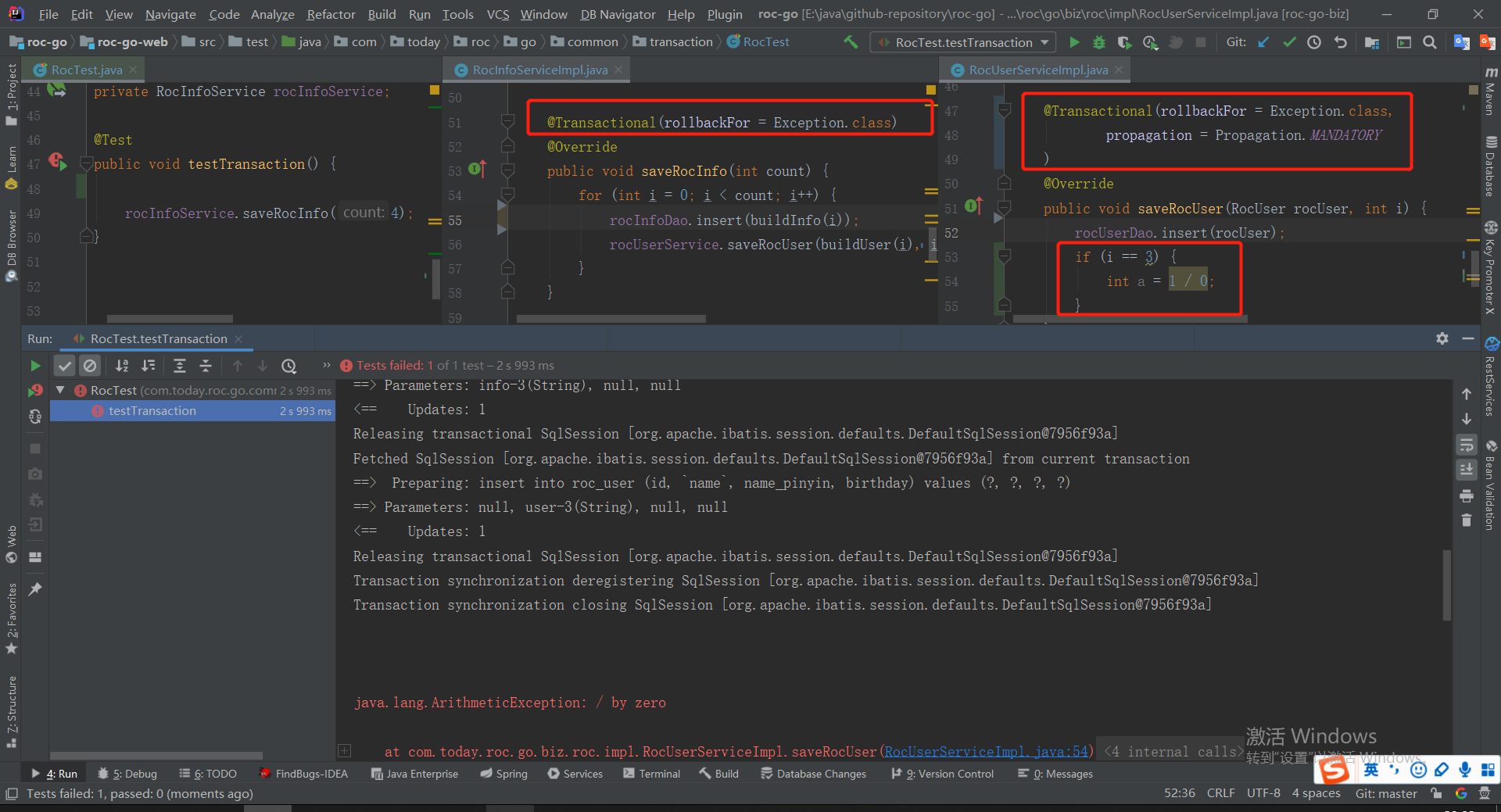
6、Propagation.MANDATORY

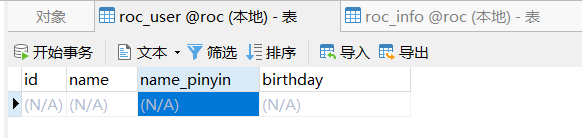
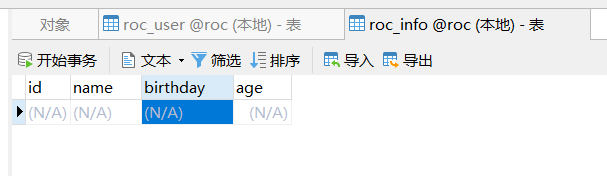
与NEVER相反，如果外层没有事务，则抛出异常

外层有事务，正常

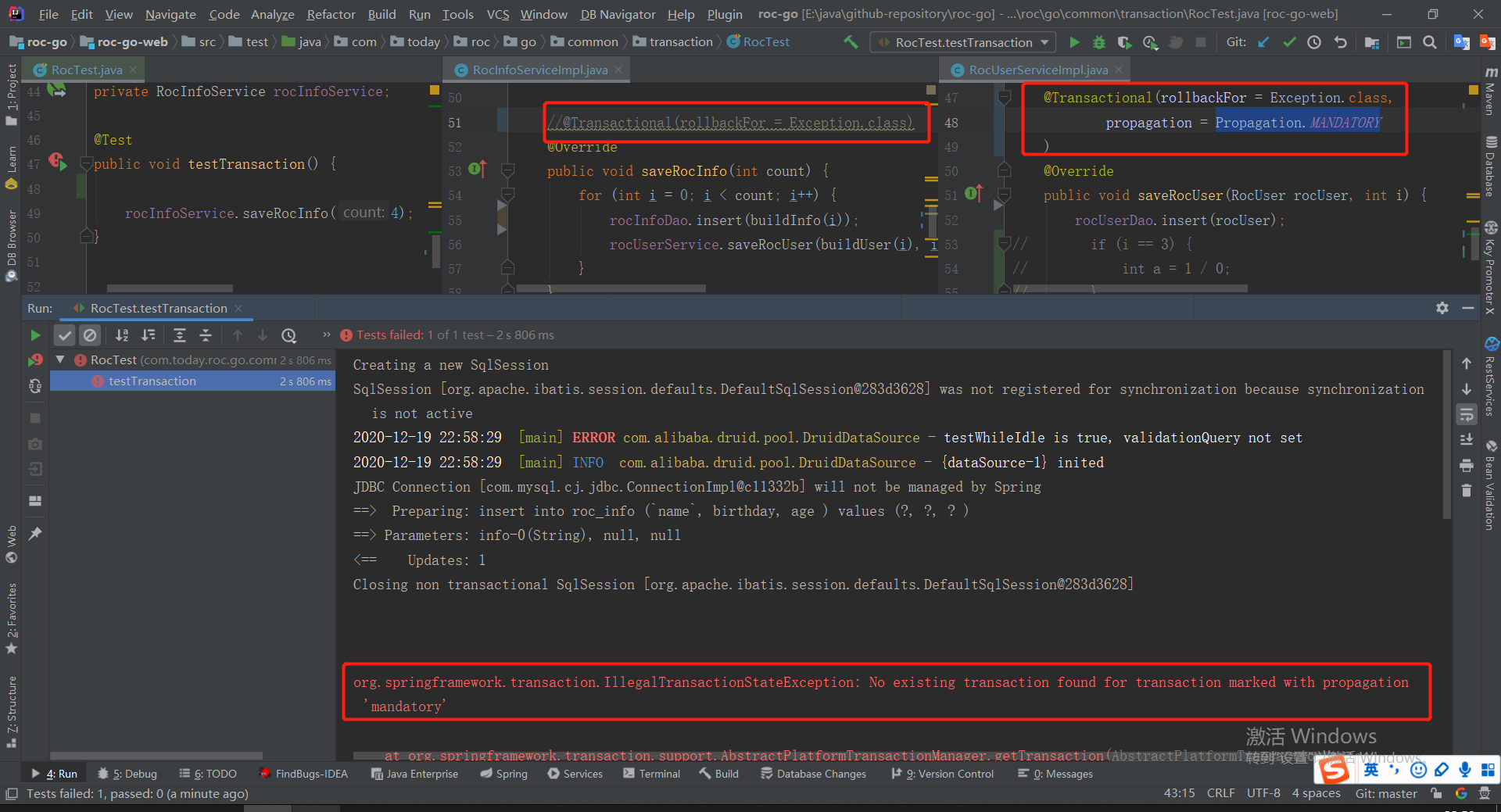


外层有事务，内层报错，全部回滚





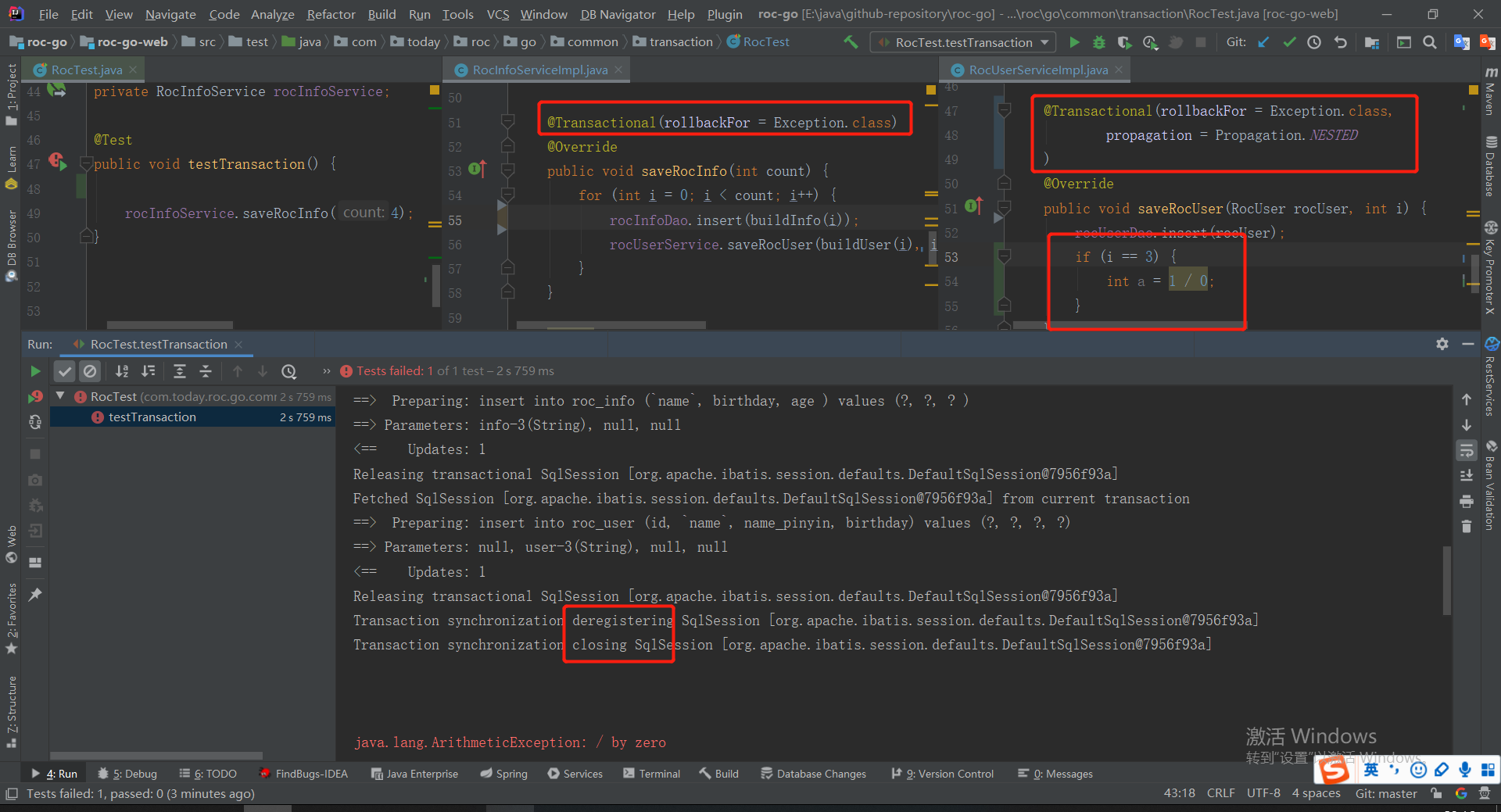
外层无事务，报错



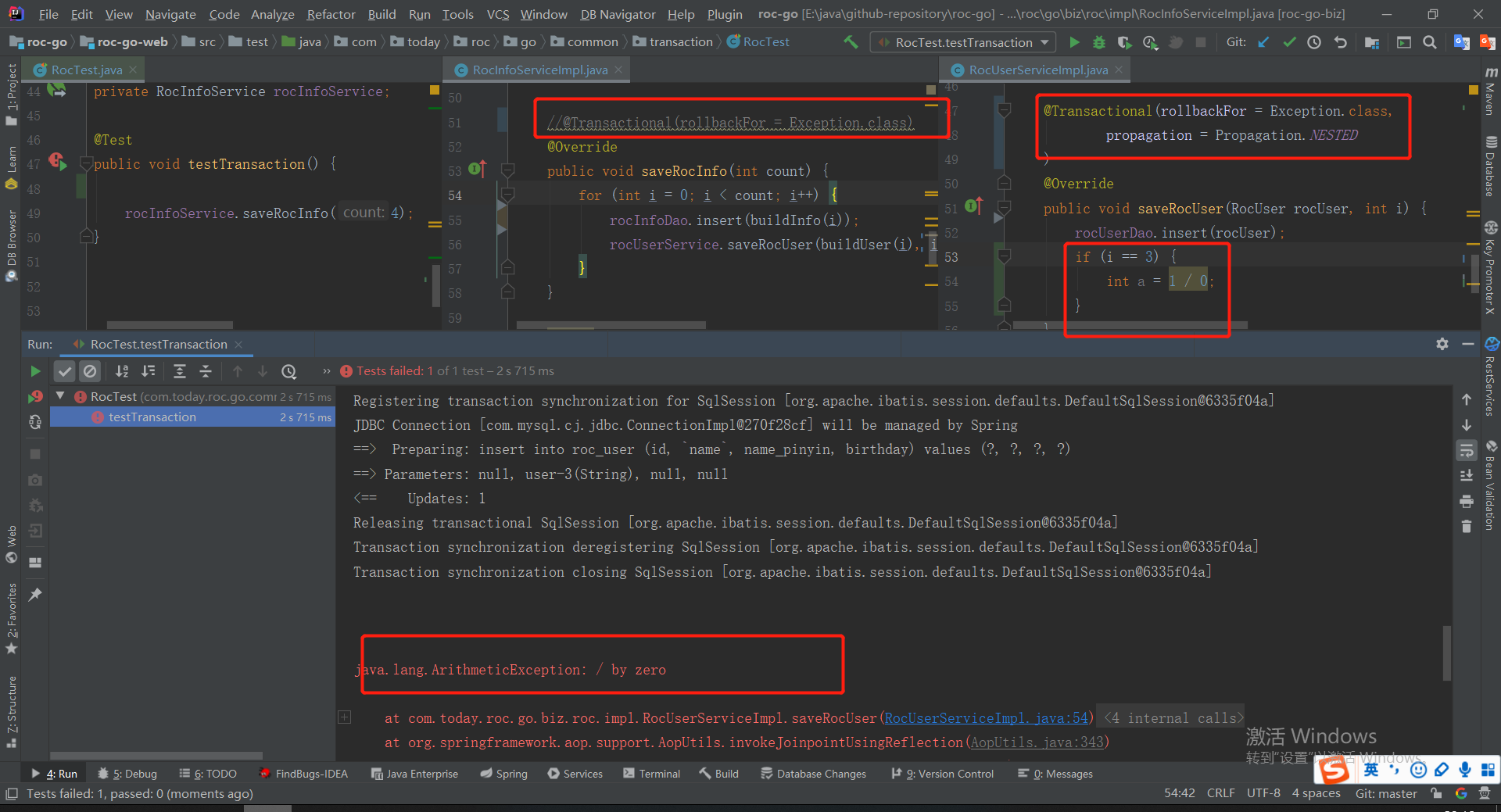
7、Propagation.NESTED

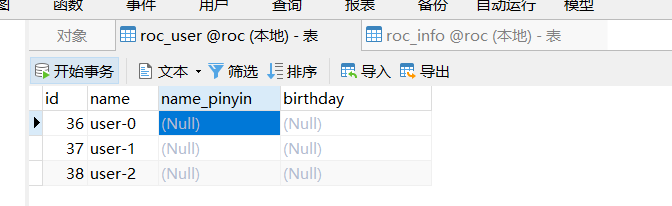
* 该传播机制的特点是可以保存状态保存点，当前事务回滚到某一个点，从而避免所有的嵌套事务都回滚，即各自回滚各自的，如果子事务没有把异常吃掉，基本还是会引起全部回滚的。

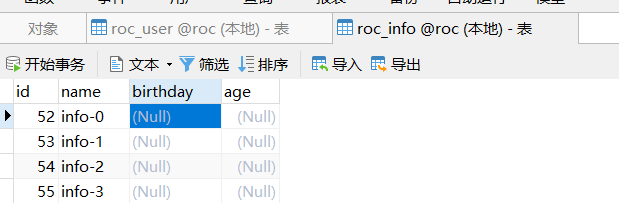
外层事务正常、嵌套事务



无外层事务，类似Propagation.REQUIRES\_NEW









PROPAGATION\_REQUIRES\_NEW 启动一个新的, 不依赖于环境的 "内部" 事务. 这个事务将被完全 commited 或 rolled back 而不依赖于外部事务, 它拥有自己的隔离范围, 自己的锁, 等等. 当内部事务开始执行时, 外部事务将被挂起, 内务事务结束时, 外部事务将继续执行.   
    另一方面, PROPAGATION\_NESTED 开始一个 "嵌套的" 事务,  它是已经存在事务的一个真正的子事务. 潜套事务开始执行时,  它将取得一个 savepoint. 如果这个嵌套事务失败, 我们将回滚到此 savepoint. 潜套事务是外部事务的一部分, 只有外部事务结束后它才会被提交.