彻底解密企业级分布式事务设计与实践（中）
抵达技术巅峰篇

主讲人：孙玄

2020.07.08

我的职业成长路线

奈学教育

奈学教育

创始人&CEO

转 转

首席架构师
技术委员会主席
大中台技术负责人

58集团

技术委员会主席
高级系统架构师

百度

资深研发工程师

毕业

浙江大学

擅长领域

架构设计、大数据
机器学习、技术管理等

对外分享

业界顶级大会
百万年薪架构
直播大课品牌创始人

目录

NX 奈学教育

NX 奈学教育

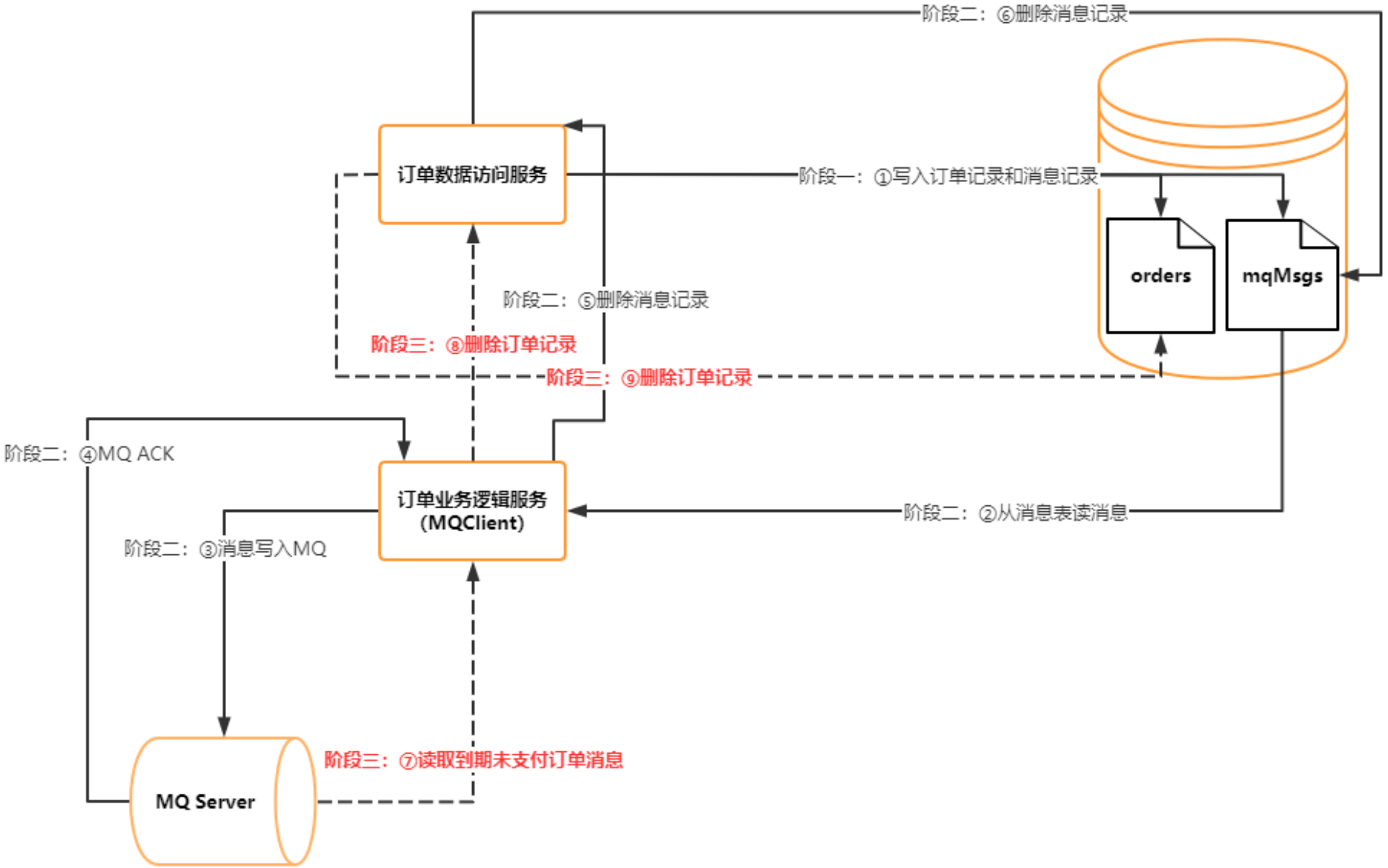
01.企业级异步业务分布式事务设计与实践（下）（分布式锁、幂等）

企业级异步业务场景分布式事务设计与实践（下）

奈学教育

- 上次课程思考
  - MQ下游消费服务如何保证事务的成功处理?
  - 异步场景分布式事务存在性能瓶颈吗？如何提升性能?

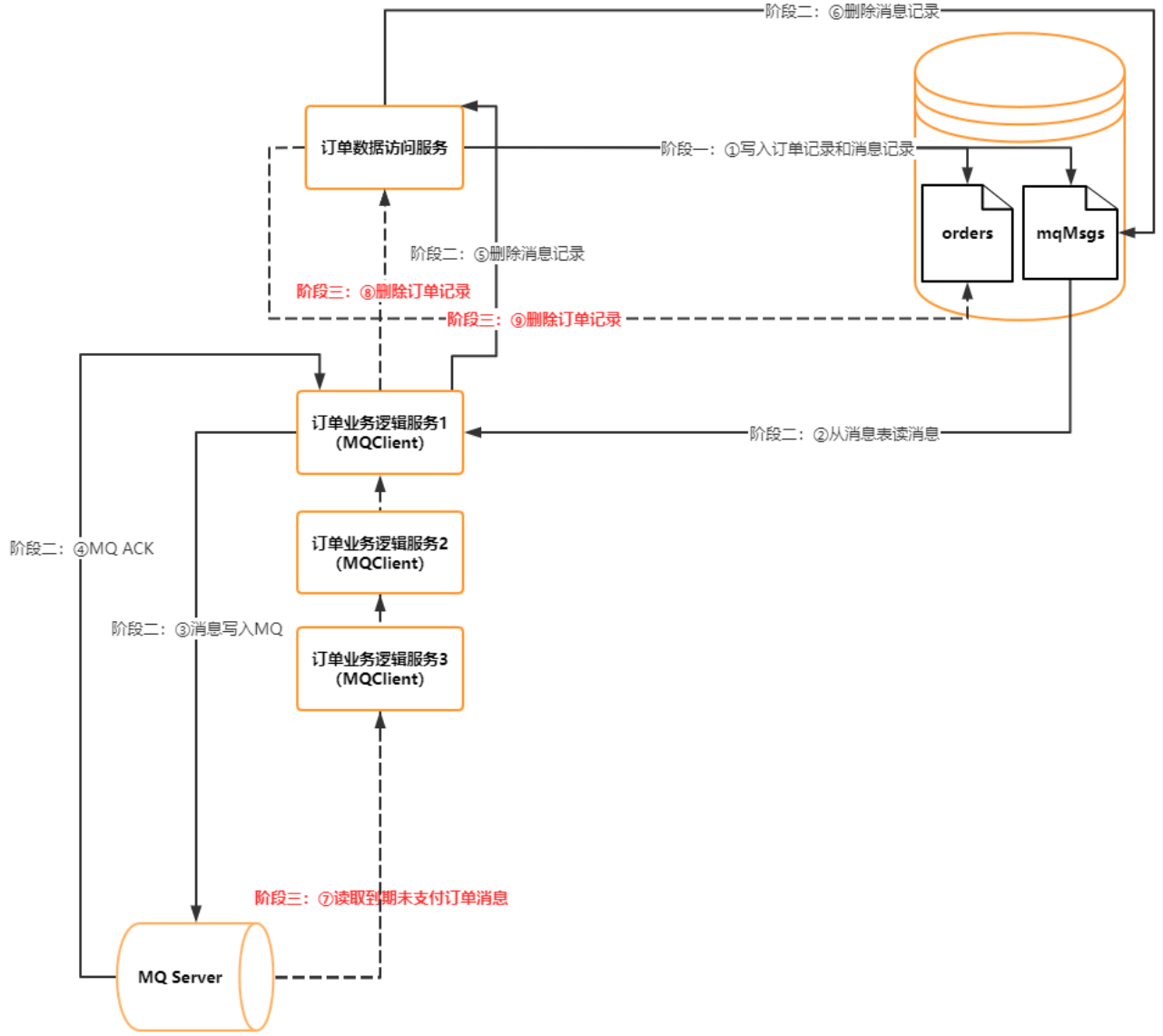
企业级异步业务场景分布式事务设计与实践（下）



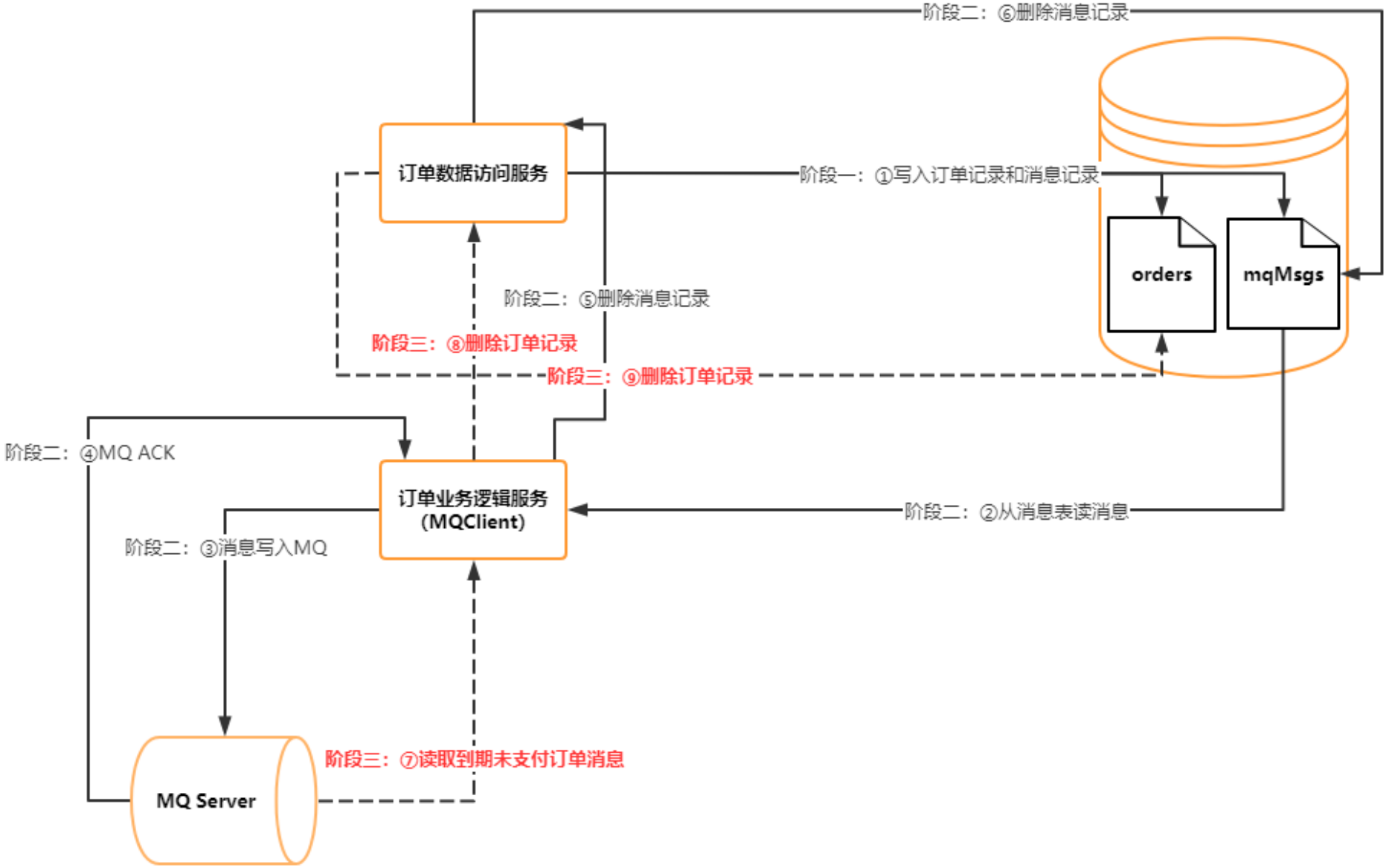
企业级异步业务场景分布式事务设计与实践（下）

奈学教育

- 上次课程思考
  - MQ下游消费服务如何保证事务的成功处理?
    - 是否需要幂等处理?
    - 是否需要去重处理?
  - 场景二
    - 支付订单



企业级异步业务场景分布式事务设计与实践（下）

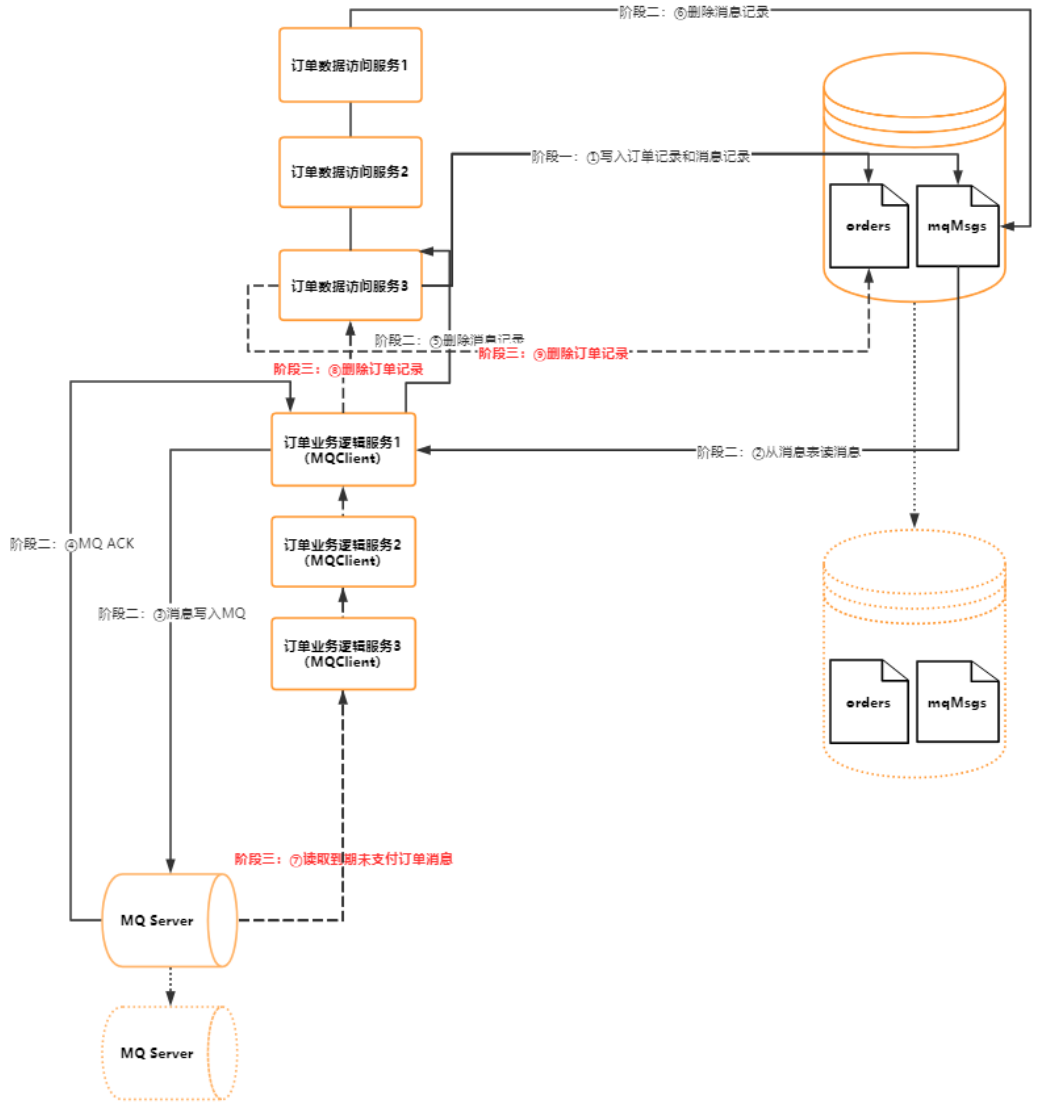


企业级异步业务场景分布式事务设计与实践（下）

奈学教育

- 上次课程思考
  - 异步场景分布式事务存在性能瓶颈吗？如何提升性能？
    - 吞吐量涉及环节
    - 响应延迟涉及环节

不存在性能瓶颈



NX 奈学教育

02.企业级同步业务场景分布式事务设计与实践（SAGAS/Seata等）

企业级同步业务场景分布式事务设计与实践（SAGAS/Seata等）

- 业务场景分类
  - 拼多多购买商品
    - 同步场景
      - 减商品库存、建立订单
      - 前台支付
    - 异步场景
      - 前台超时未支付

企业级同步业务场景分布式事务设计与实践（SAGAS/Seata等）

- Saga模型剖析
  - 起源于1987年 Hector Garcia-Molina，Kenneth Salem 发表论文《Sagas》
  - 把一个分布式事务拆分为多个本地事务，每个本地事务都有相应的执行模块和补偿模块
  - 当Saga事务中任意一个本地事务出错时，通过调用补偿方法恢复之前的事务，达到事务最终一致性
  - 当每个Saga子事务 T1, T2, ⋯, Tn 都有对应的补偿定义 C1, C2, ⋯, Cn-1,那么Saga系统可以保证：
    - 子事务序列 T1, T2, ⋯, Tn得以完成 (最佳情况)
    - 或者序列 T1, T2, ⋯, Tj, Cj-1, ⋯, C2, C1, 0 < j < n, 得以完成

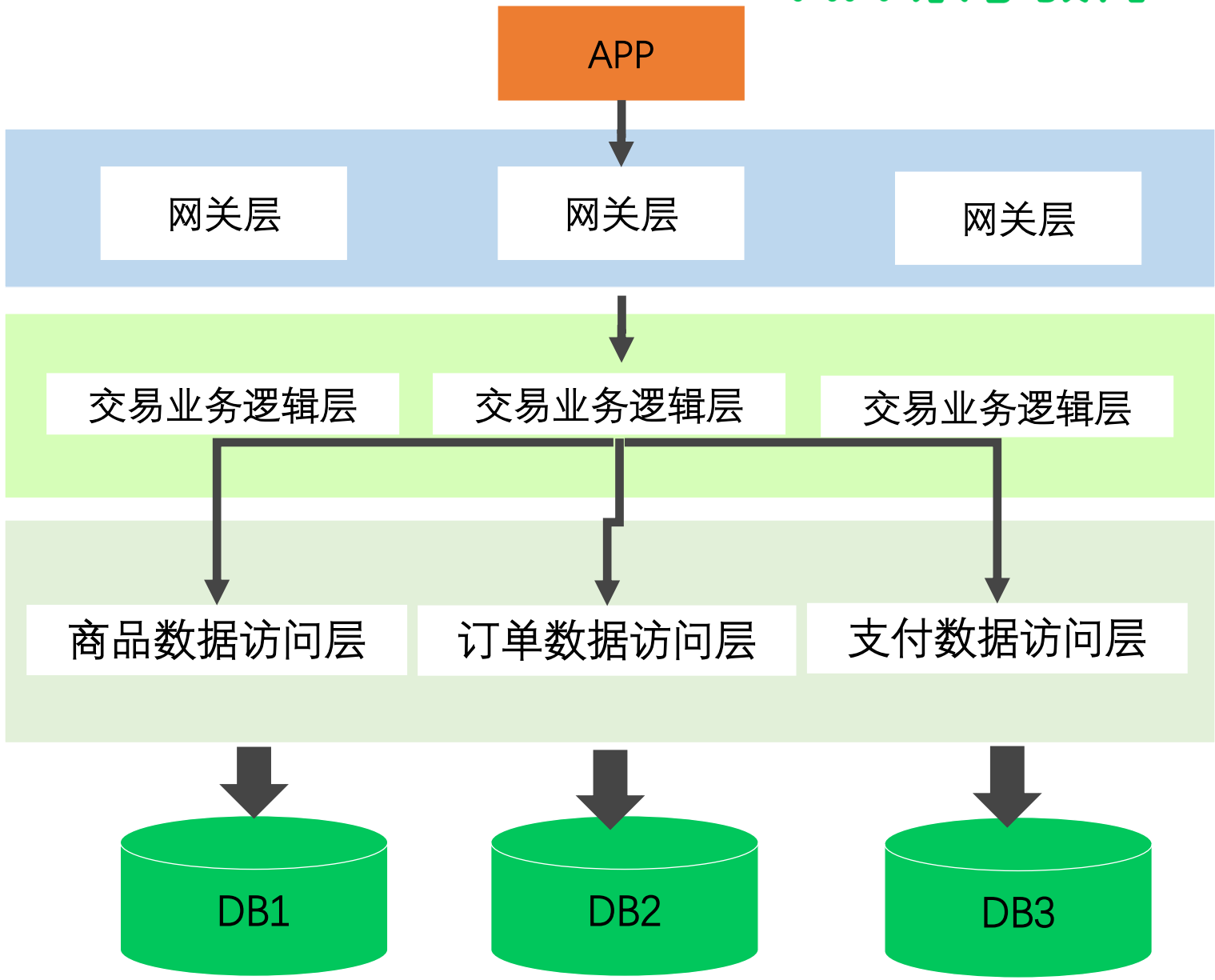
  - Saga隔离性
    - 业务层控制并发
      - 在应用层加锁
      - 应用层预先冻结资源等

  - Saga恢复方式
    - 向后恢复：补偿所有已完成的事务，如果任一子事务失败 **回滚已提交事务**
    - 向前恢复：重试失败的事务，假设每个子事务最终都会成功 **retry**

生产中使用的多一些 →

企业级同步业务场景分布式事务设计与实践（SAGAS/Seata等）

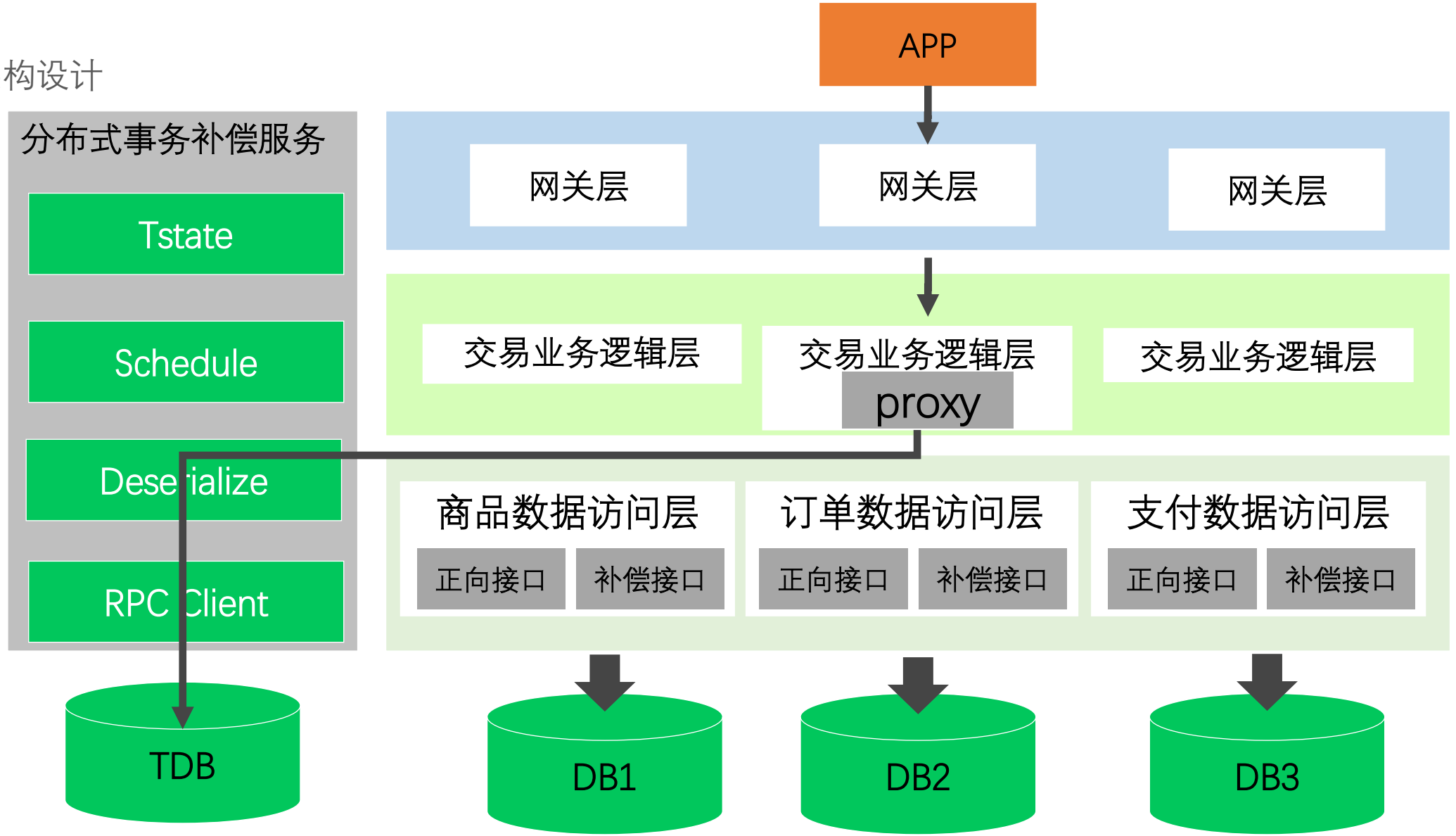
- 业务场景分类
  - 拼多多购买商品
    - 同步场景
      - 减商品库存、建立订单
      - 前台支付
  - 架构设计



企业级同步业务场景分布式事务设计与实践（SAGAS/Seata等）

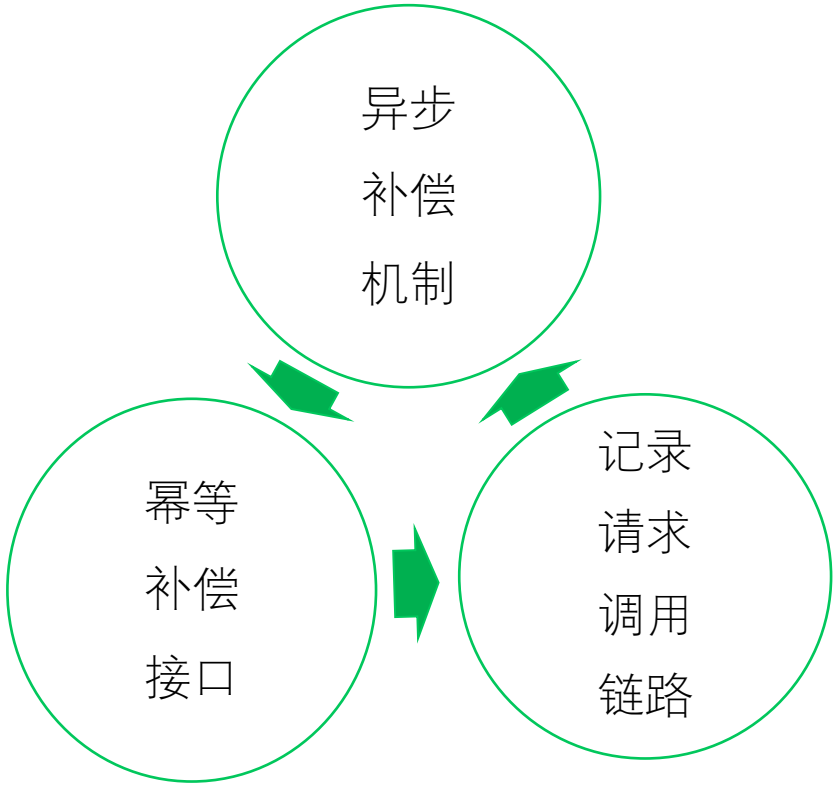
NX 奈学教育

- Sagas架构设计



企业级同步业务场景分布式事务设计与实践（SAGAS/Seata等）

- Sagas架构设计三大关键技术



企业级同步业务场景分布式事务设计与实践（SAGAS/Seata等）

- 业务逻辑层Proxy设计（**基于拦截技术**）
  - 交易业务逻辑层加事务注解@Around(“execution(* *(..)) && @annotation(TX)”)
  - Proxy在真正业务逻辑被调用之前, 生成一个全局唯一 TXID 标示事务组，TXID保存在全局变量里，事前拦截器写入，并向TDB写入 TXID 并把事务组置为开始状态，完成业务操作后，清除TXID标识
  - 交易业务逻辑层调用数据访问层前，通过RPCProxy代理记录当前调用请求上下文参数
  - 如果业务成功，调用记录删除
  - 如果调用异常，根据调用记录反向补偿

企业级同步业务场景分布式事务设计与实践（SAGAS/Seata等）

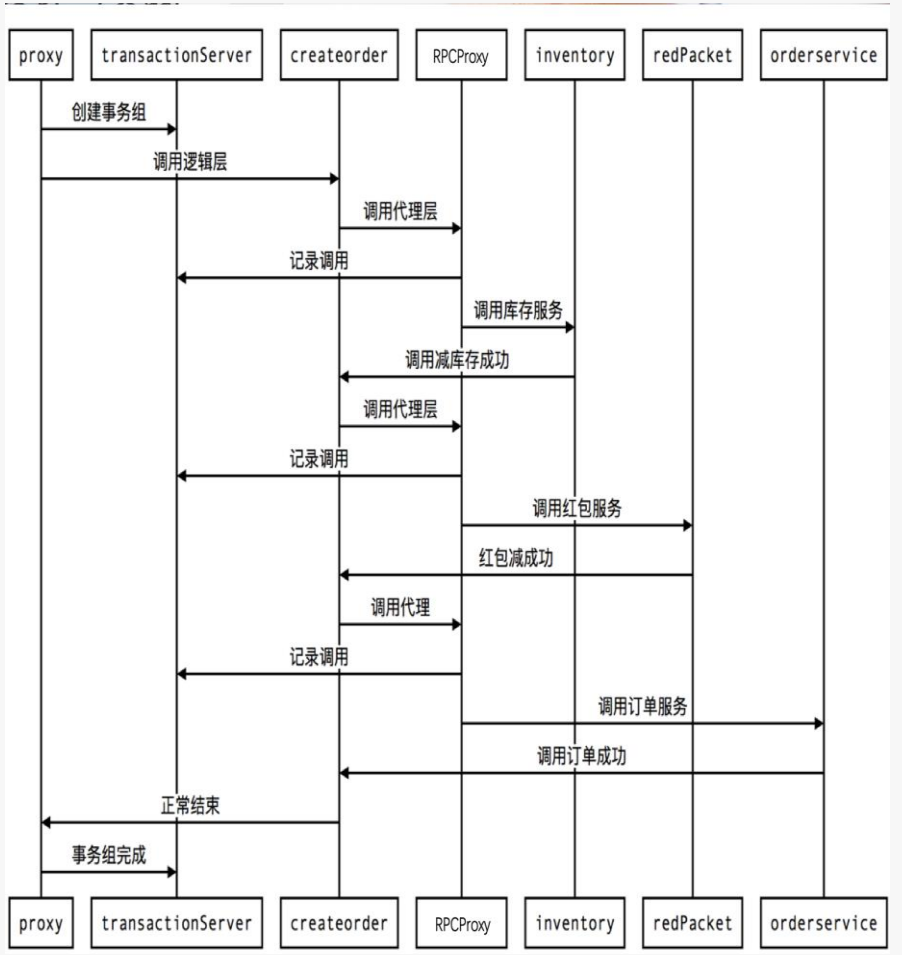
- 分布式事务补偿服务
  - 事务组表（数据库表TDB）
    - 记录事务组状态
    - txid、state、timestamp
  - 事务调用组表（数据库表TDB）
    - 记录事务组内的每一次调用以及相关参数
    - txid、actionid、callmethod、pramatype、params
  - 补偿策略
    - 调用执行失败，修改事务组状态
    - 分布式事务补偿服务异步执行补偿

企业级同步业务场景分布式事务设计与实践（SAGAS/Seata等）

Sagas成功案例

- 二手交易创建订单事务组正常流程
  - 锁库存->减红包->创建订单
  - RPCProxy透明记录调用请求参数
    - 记录事务域的开始与结束
    - 在所有远程调用成功时
    - 对业务逻辑不做侵入

Sagas成功案例

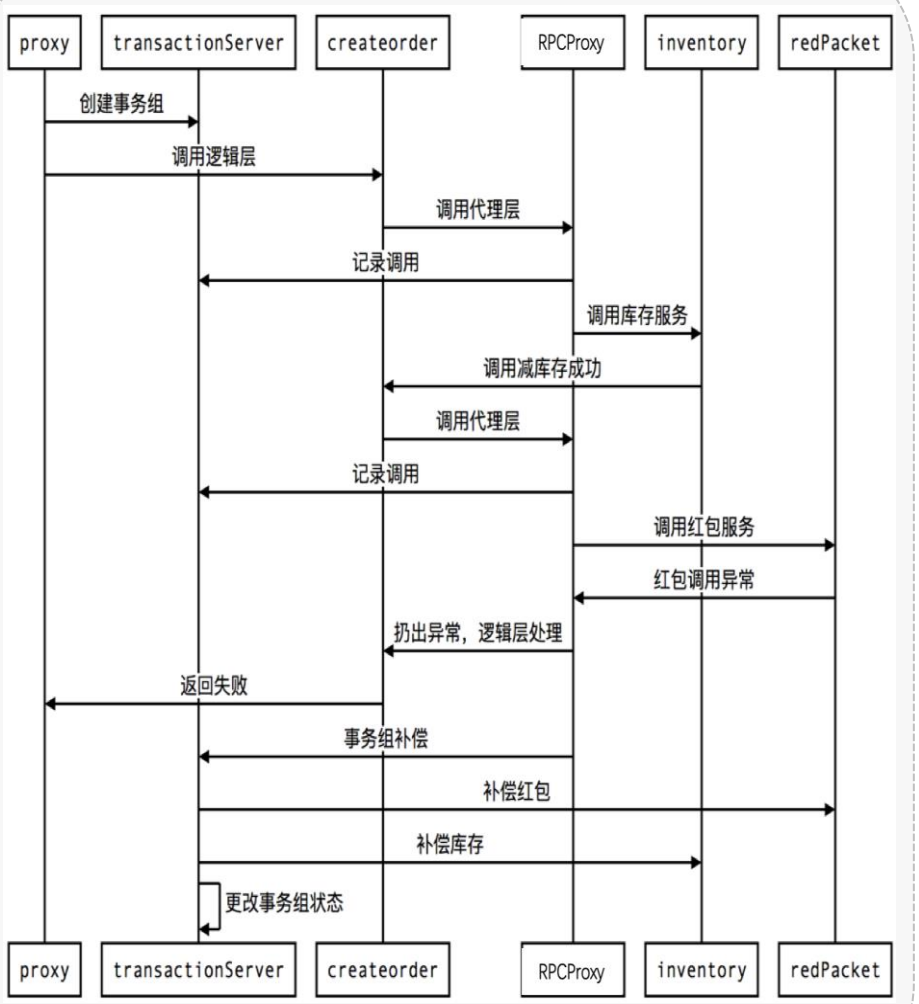


企业级同步业务场景分布式事务设计与实践（SAGAS/Seata等）

Sagas失败案例

- 二手交易创建订单事务组异常流程
  - 微服务数据访问层失败，RPCProxy更改事务组状态
  - 微服务业务正常执行
  - 事务补偿服务异步执行补偿

Sagas失败案例



- Sagas失败
  - 记录错误日志
  - 报警
  - 人工智能
    - 人工介入

企业级同步业务场景分布式事务设计与实践（SAGAS/Seata等）

- 分

的分

补偿

企业级同步业务场景分布式事务设计与实践（SAGAS/Seata等）

- 企业级Sagas源码剖析

项目组成

- nx-saga
  - nx-saga-common
    saga api和基类，jar组件
  - nx-saga-dubbo
    saga dubbo客户端，jar组件
  - nx-saga-nxrpc
    saga nxrpc客户端，jar组件
  - nx-saga-tm
    saga服务端(TM)，SB应用

TDB表结构

- t_txgroup
  - txid、state(状态)、priority (优先级) 、createTime、updateTime
- t_txrecord
  - id、txid、mannerName (方式名) 、className (类名) 、compensateName (补偿方法名) 、methodName (方法名) 、 params (参数) 、paramTypes (参数class) 、regAddress (注册地址) 、serviceName (服务名) 、 step (步骤) 、 version 、createTime
- t_txcompensate
  - id、step (步骤) 、txid 、success、createTime、updateTime

- 企业级Sagas源码剖析
  - 事务拦截器（proxy）：生成事务ID，插入事务TDB的t_txgroup表

```
@Around("execution(* *(..)) && @annotation(com.nx.arth.tx.saga.common.annotation.SagaTransactional)")
public Object around(ProceedingJoinPoint point)
    throws Throwable {
    // 开启AP
    TransactionManager.startTransaction();
    Logger.info("desc=startTransaction txid=" + TransactionManager.getTransactionId());
    // 创建事务主表对象
    Date now = new Date();
    TransactionGroup transactionGroup = new TransactionGroup();
    transactionGroup.setTxid(TransactionManager.getTransactionId());
    transactionGroup.setState(TransactionState.start.getState());
    transactionGroup.setPriority(2);
    transactionGroup.setCreateTime(now);
    transactionGroup.setUpdateTime(now);
    transactionGroupDao.insert(transactionGroup);

    // 执行方法
    Object result = point.proceed();

    // 正常事务结束
    TransactionGroup transactionGroupInThread = transactionGroupDao.select(transactionGroup.getTxid());
    if (transactionGroupInThread.getState().equals(TransactionState.start.getState())) {
        transactionGroup.setState(TransactionState.end.getState());
        transactionGroup.setUpdateTime(new Date(System.currentTimeMillis()));
        transactionGroupDao.update(transactionGroup);
        Logger.info("k=s act=endTransaction txid=" + TransactionManager.getTransactionId());
    }
    TransactionManager.endTransaction();
    return result;
}
```

企业级同步业务场景分布式事务设计与实践（SAGAS/Seata等）

- 企业级Sagas源码剖析
  - 事务步骤拦截器：获取每一步的上下文，插入事务TDB的t_txrecord表

```
@Around("execution(* *(..)) && @annotation(com.nx.arth.tx.saga.common.annotation.Compensate)")
public Object around(ProceedingJoinPoint point)
    throws Throwable {
    // 1、参数校验及获取
    if (StringUtils.isEmpty(TransactionManager.getTransactionId())) {
        return point.proceed();
    }
    Class<?> targetClass = point.getTarget().getClass();
    MethodSignature methodSignature = (MethodSignature)point.getSignature();
    Method method = targetClass.getMethod(methodSignature.getName(), methodSignature.getParameterTypes());
    Compensate compensate = method.getAnnotation(Compensate.class);
    Object[] arguments = point.getArgs();

    // 2、创建事务步骤对象
    TransactionRecord record = new TransactionRecord();
    record.setId(UUID.randomUUID().toString());
    record.setTxid(TransactionManager.getTransactionId());
    record.setMethodName(method.getName());
    record.setClassName(targetClass.getName());
    record.setParamTypes(Base64Utils.encodeToString(SerializationUtils.serialize(methodSignature.getParameterTypes())));
    record.setParams(Base64Utils.encodeToString(SerializationUtils.serialize(arguments)));
    record.setCreateTime(new Date());
    // 3、dubbo服务参数获取
    refMap.entrySet().parallelStream().forEach(item -> {
        Class<?> clazz = item.getValue().getInterfaceClass();
        if (clazz.isInstance(point.getTarget())) {
            record.setClassName(clazz.getName());
            record.setVersion(item.getValue().getVersion());
            record.setServiceName(item.getValue().getApplication().getName());
            record.setMannerName(item.getValue().getRegistry().getProtocol());
            record.setRegAddress(item.getValue().getRegistry().getAddress());
            return;
        }
    });
```

- 企业级Sagas源码剖析
  - 事务调度补偿器：获取需要补偿的记录进行补偿

```
 * @类描述 dubbo泛化调用
@Component
public class DubboInvoker {
    /**
     * @方法名称 compensate
     * @功能描述 对指定事务步骤进行补充泛化调用
     * @param record 事务步骤
     * @return 补充成功：true-正常ACK，false-异常返回
     */
    public Boolean compensate(TransactionRecord record) {
        try {
            String version = StringUtils.isEmpty(record.getVersion()) ? "1.0.0" : record.getVersion();
            GenericService service = DubboGenericServiceUtil.getDubboService(record.getRegAddress(), record.getMannerName(), record.getClassName(), version);
            Object[] params = (Object[])SerializationUtils.deserialize(Base64Utils.decodeFromString(record.getParams()));
            Class<?>[] paramType = (Class<?>[])SerializationUtils.deserialize(Base64Utils.decodeFromString(record.getParamTypes()));
            String[] parameterTypes = new String[paramType.length];
            for (int i = 0; i < paramType.length; i++) {
                parameterTypes[i] = paramType[i].getName();
            }
            service.$invoke(record.getCompensateName(), parameterTypes, params);
            return true;
        } catch (Exception e) {
            return false;
        }
    }
}
```

百万架构师VIP学员获取全部企业级可运行代码！！

- 思考
  - 高并发情况下，TM数据库面临单点压力，如何实现Sagas模式下的分布式数据源?

明天直播课程继续讲解！！！

要点回顾

NX 奈学教育

01 企业级异步业务场景分布式事务设计与实践（下）（分布式锁、幂等）

02 企业级同步业务场景分布式事务设计与实践（TCC/SAGAS/Seata等）

NX 奈学教育

欢迎关注本人公众号
“架构之美”