kafka 如何保证消息幂等

tojson 老师

1、 经典面试场景

面试官：你好，请问你做过的项目中，是如何进行系统间解耦的？

候选人：我们系统之间基本都是通过 kafka 消息来实现解耦的，kafka 除了用于系统的解耦，还能解决流量的削峰，高并发的时候，可以极大减轻数据库的压力。

面试官：既然你聊到 kafka，那我们聊一下生产环境经常遇到的一些问题，比如你的上游因为网络抖动同一条消息发送了两次，这个时候可能会产生一些脏数据，那你说一下这类问题的解决方案？

候选人：这个其实是我们经常说的消息幂的问题，对于消息的幂等有很多种方案，常见的解决方案有四种：

1）给业务设置唯一 key，比如订单业务的订单号、支付业务的支付流水号等常见的唯一 key，下游针对 key 加一个分布式锁，最后通过数据库表设置唯一健兜底来保证消息的幂等。

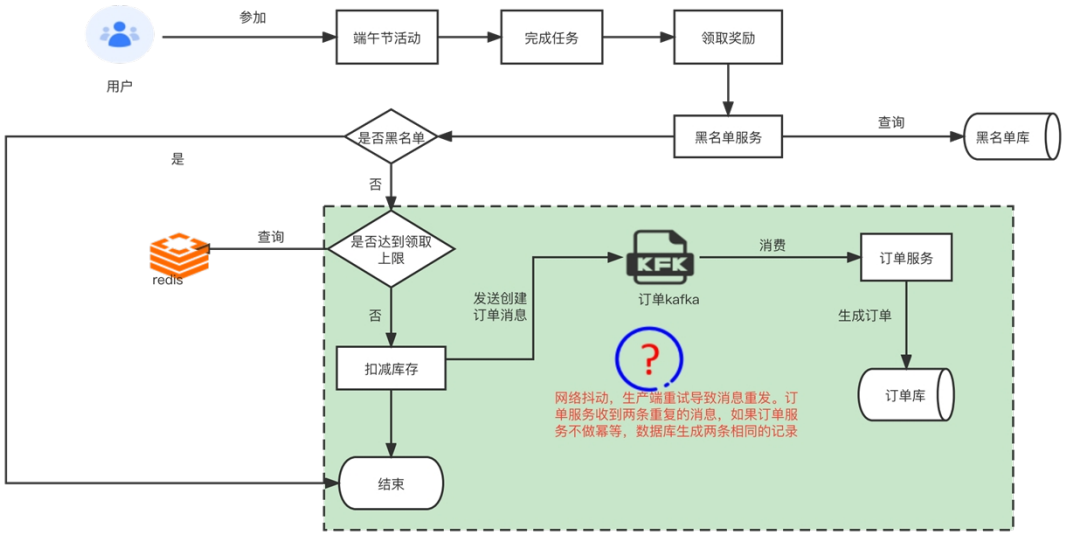
2）发送消息的时候构造一个全局的唯一 Id，下游消费的时候判断全局的唯一 Id 是否存在，如果已经存在，这条消息不用处理。

3）对于有状态流转的业务，如果状态机已经处于下一个状态，这时候来了一个上一个状态的变更消息，这个消息不用处理。

4）对于一些更新的业务，可以采用基于版本号的乐观锁，更新 SQL 的时候我们判断传入的当前的版本号与数据库表的版本号是否相等，如果相等更新成功，不相等更新失败。

面试官：OK，这样听起来比较抽象，你能举一个实际应用的例子吗？

候选人：好的，我这里举一个我们最近做了一个端午节活动的例子，如下所示：



候选人：为了保证消息幂等，我们可以在生产端的消息体给业务设置唯一 key，也就是场景中的订单号，同时在消费端使用这个 key 做分布式锁，最后在订单表中对订单号这个字段设置一个唯一健约束做最终的兜底确保订单表不会产生相同的记录。

面试官：好的，我这边没问题了，恭喜你进入下一轮！

2、基本概念及产生消息重复的原因

2.1 幂等的基本概念

幂等简单点讲，就是用户对于同一操作发起的一次请求或者多次请求的结果是一致的，不会产生任何副作用。幂等分很多种，比如接口的幂等、消息的幂等，它是分布式系统设计时必须要考虑的一个方面。

✓ 查询操作（天然幂等）

查询一次和查询多次，在数据不变的情况下，查询结果是一样的。查询是天然的幂等操作。

✓ 删除操作（天然幂等）

删除操作也是幂等的，删除一次和删除多次都是把数据删除(注意可能返回结果不一样，删除的数据不存在，返回 0，删除的数据多条，返回结果多个)。

✓ 新增操作

新增操作，这种情况下多次请求，可能会产生重复数据。

✓ 修改操作

修改操作，如果只是单纯的更新数据，比如：update account set money=100 where id=1，是没有问题的。如果还有计算，比如：update account set money=money+100 where id=1，这种情况下多次请求，可能

会导致数据错误。

总的来说：当出现消费者对某条消息重复消费的情况时，重复消费的结果与消费一次的结果是相同的，并且多次消费并未对业务系统产生任何负面影响，那么这个消费者的处理过程就是幂等的。

例如，在支付场景下，消费者消费扣款消息，对一笔订单执行扣款操作，扣款金额为 100 元。如果因网络不稳定等原因导致扣款消息重复投递，消费者重复消费了该扣款消息，但最终的业务结果是只扣款一次，扣费 100 元，且用户的扣款记录中对应的订单只有一条扣款流水，不会多次扣除费用。那么这次扣款操作是符合要求的，整个消费过程实现了消费幂等。

2.2 产生消息重复的原因

在互联网应用中，尤其在网络不稳定的情况下，消息队列的消息有可能会出现重复。如果消息重复消费会影响您的业务处理，请对消息做幂等处理。消息重复的可能原因如下：

◆ 发送时消息重复

当一条消息已被成功发送到服务端并完成持久化，此时出现了网络闪断或者生产者宕机，导致服务端对生产者应答失败。 如果此时生产者 Producer 意识到消息发送失败并尝试再次发送消息，消费者 Consumer 后续会收到两条内容相同的消息。

◆ 投递时消息重复

消息消费的场景下，消息已投递到消费者 Consumer 并完成业务处理，当消费者给服务端反馈应答的时候网络闪断。为了保证消息至少被消费一次，消息队列的服务端将在网络恢复后再次尝试投递之前已被处理过的消息，消费者 Consumer 后续会收到两条内容相同的消息。

◆ 负载均衡时消息重复

当消息队列的服务端或消费者重启、扩容或缩容时，都有可能会触发 rebalance，此时消费者 Consumer 可能会收到重复消息。

3、解决方案及案例分析

既然消息可能会产生重复，那如何解决消息幂等的问题呢？我们需要从生产者、中间件、消费者这几个不同层面，来保证消息的幂等，消息的幂等业界有很多种方案，我这里列出常见的几种方案供大家参考：

3.1 设置业务唯一 key 方案（应用最广泛）

业务唯一 key 可以是单个字段或者组合字段，这个方案是怎么实现的呢？

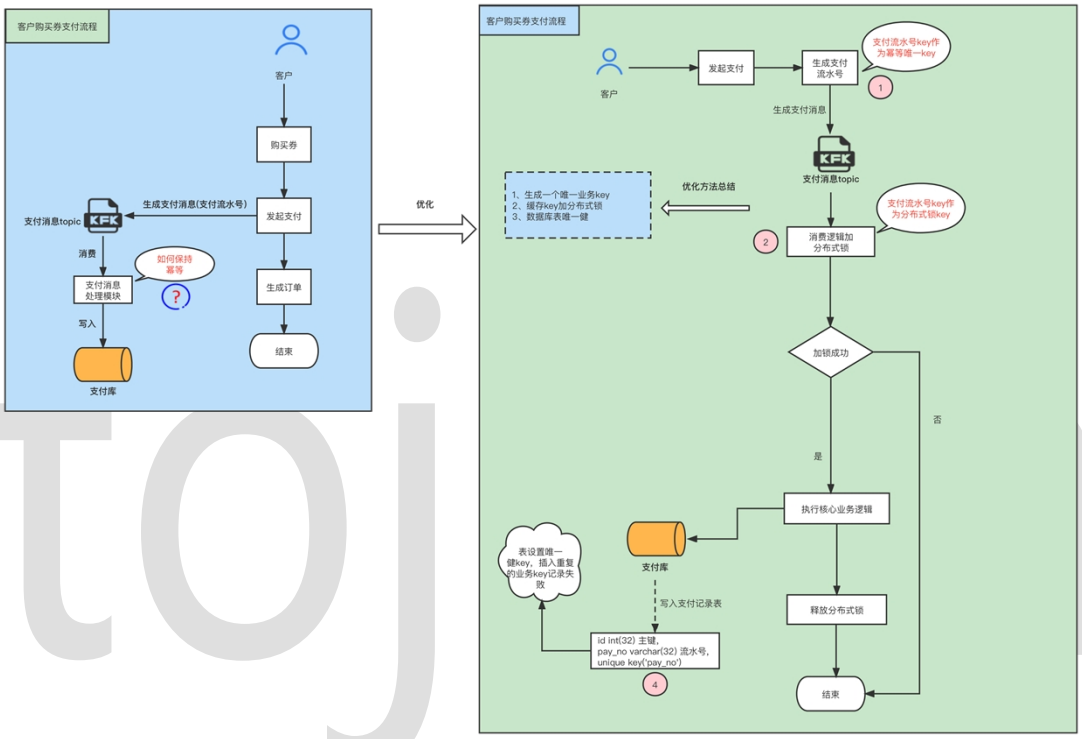
1）生产者消息体构造业务唯一 key，消息端针对这个 key 加分布式锁

2）在消费端，创建一个消息防重表，利用插入记录唯一健约束控制

与业务有一定的耦合，另外高并发下频繁对消息防重表进行操作，性能比较低，不太建议使用。

3）数据库业务表加唯一索引（数据库）

案例分析图如下所示：



具体业务流程：

1、客户发起支付流程。

2、生产者生产消息构造一个支付流水号作为消息体幂等的唯一 key。

3、发送消息给 broker，broker 持久化消息到磁盘。

4、消费者开始消费消息，我们在消费逻辑中加一个分布式锁，防止短时间内消息重复投递。

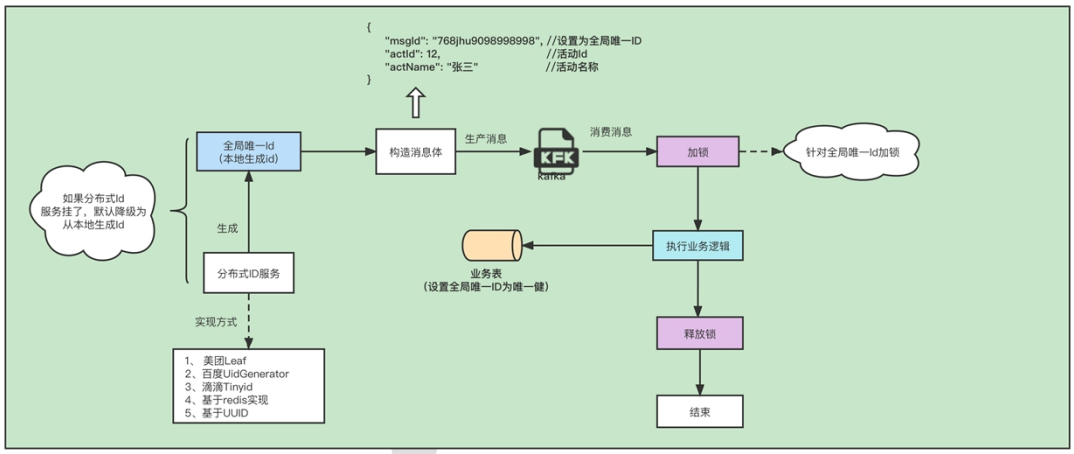
5、当加锁成功后，执行核心业务逻辑，然后释放分布式锁，当加锁失败，直接结束。

6、最后，为了防止后续生产者重复推送相同唯一 key 的消息，我们需要在数据库的业务表设置一个针对这个 key 的唯一键，通过唯一健约束来保证数据库表不会出现两条相同的记录，从而实现消息幂等。

3.2 设置全局唯一 id 方案

消息生产者生成一个唯一的消息 Id（利用分布式 Id 生成服务或本地 Id 生成一个 Id），我们在消息投递时，给每条业务消息附加一个唯一的消息 Id，然后就可以在消费者利用类似分布式锁的机制，实现唯一性的消费。

案例分析图：



具体业务流程：

1、生产者生产消息构造消息体，这里会通过分布式 Id 服务生成一个全局唯一 Id，如果分布式 ID 服务因为高并发挂掉了，降级从本地生成一个消息 Id。

2、发送消息给 broker，broker 持久化消息到磁盘。

3、消费者开始消费消息，我们在消费逻辑中加一个分布式锁，防止短时间内消息重复投递。

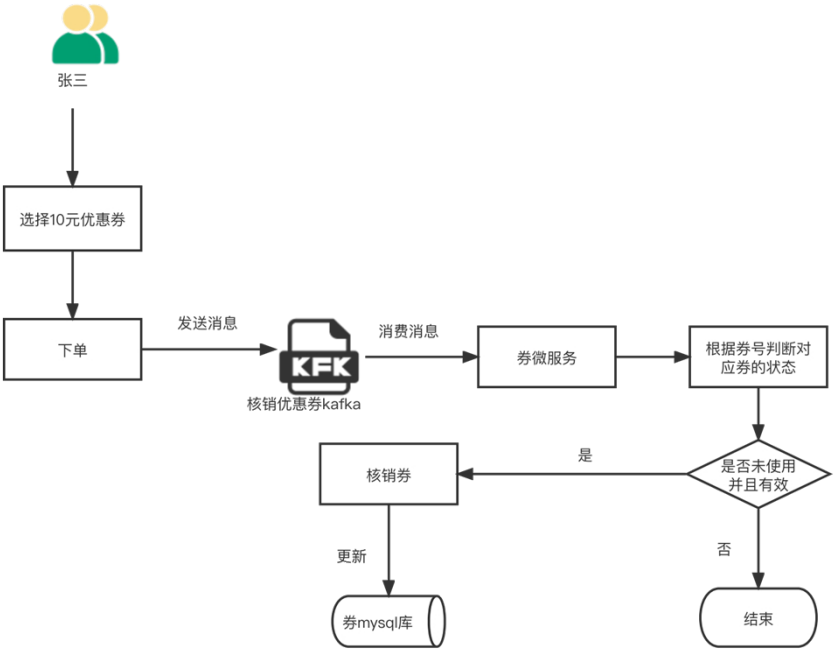
4、当加锁成功后，执行核心业务逻辑，然后释放分布式锁，当加锁失败，直接结束。

5、最后，为了防止后续生产者重复推送相同唯一 id 的消息，我们需要在数据库的业务表设置一个针对这个唯一 Id 的唯一键，通过唯一健约束来保证数据库表不会出现两条相同的记录，从而实现消息幂等。

3.3 基于业务的状态机方案

在设计单据相关的业务，或者是任务相关的业务，肯定会涉及到状态机(状态变更图)，我们以业务单据为例，在业务单据上面会有个状态，状态在不同的情况下会发生变更，一般情况下存在有限状态机，当消费业务消息的时候，如果状态机已经处于下一个状态，这时候来了一个上一个状态的消息，直接丢弃消息不处理，保证了有限状态机的幂等。

案例分析图：



具体业务流程：

1、客户张三选择一个 10 元的优惠券。

2、客户下单完成，发送一条核销优惠券的消息。

3、券微服务消费这条消息，根据券号判断对应券的状态是否未使用并且有效。

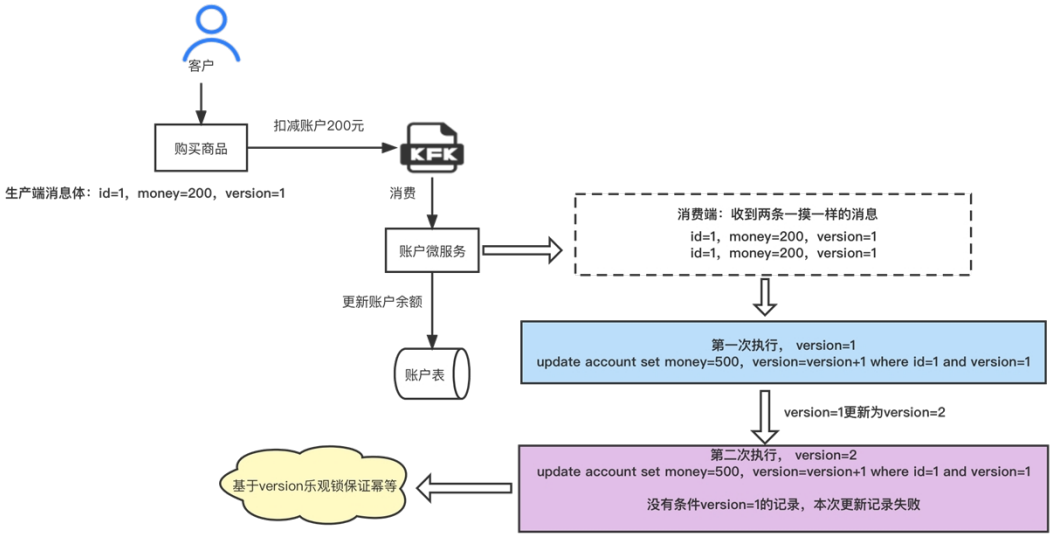
4、如果是，对券进行核销，更新对应券的状态。

5、如果不是，直接返回结束。

3.4 基于 version 版本号的乐观锁方案

此方案一般是适用于更新业务的场景，更新表的时候通过版本号对比来保证消息的幂等。

案例分析图：



具体业务流程：

1、客户购买商品，完成后准备发送一条扣减账户 200 的消息。

2、生产端开始生产消息，构造消息体{id=1，money=200，version=1}。

3、发送消息给 broker，broker 持久化这条消息后，返回确认消息给生产者，此时出现了网络闪断或者生产者宕机，导致 broker 对生产者响应失败。 如果此时生产者意识到消息发送失败并尝试再次发送消息，消费者后续会收到两条内容相同的消息。

4、消费者收到相同的消息，开始消费第一条消息{id=1，money=200，version=1}，根据 version=1 更新记录，更新成功。

5、接着开始消费第二条消息{id=1，money=200，version=1}，根据 version=1 更新记录，但是此时 version 已经被更新为 2，条件不满足更新失败。

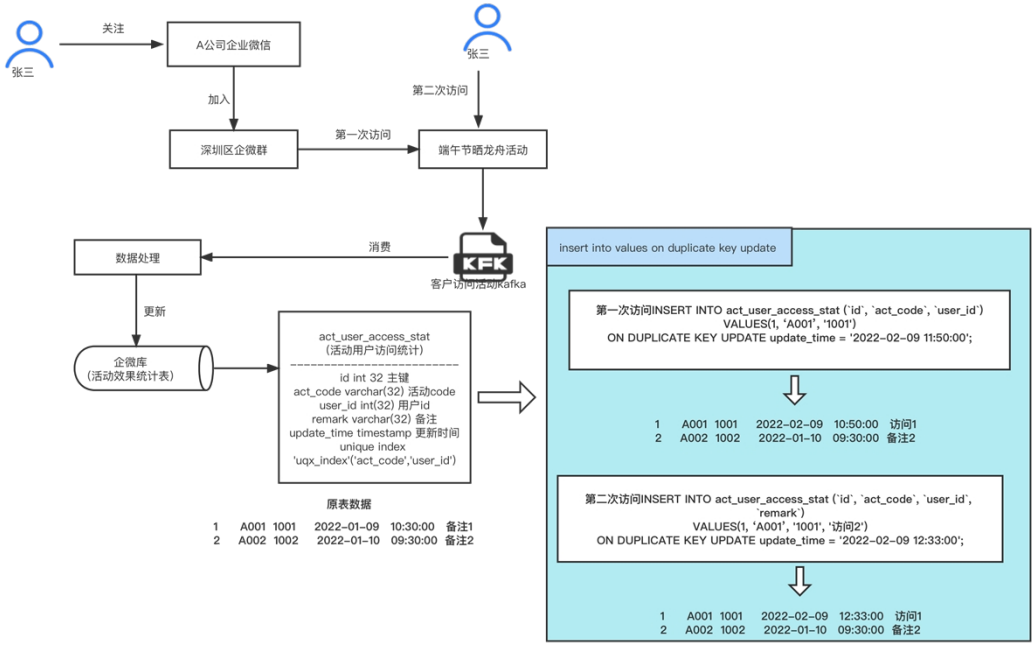
6、消费者通过基于 version 的乐观锁保证了消息幂等。

3.5 insert ...on duplicate key update 更新方案

此方案一般适合一些统计更新类的业务或者定时同步第三方平台数据到自己数据库的场景。

例如：定时同步企业微信的成员数据到自己企微库的成员表，就可以采用这种方案实现。

案例分析图：



具体业务流程：

1、张三关注某 A 公司企业微信，加入深圳区企微群。

2、管理员在深圳区企微群投放一个活动，张三第一次点击这个活动，这个时候活动模块发送一条 kafka 消息。

3、在数据库创建一张活动效果统计表，act_code 和 user_id 两个字段作为联合唯一索引

4、数据处理模块消费这条消息，通过 insert.. on duplicate key update 插入一条记录

5、过了几小时，张三第二次点击这个活动，这个时候活动模块再发送一条 kafka 消息，数据处理模块再次消费这条消息，通过 insert.. on duplicate key update 更新对应唯一索引的这条记录的更新时间字段。

6、通过 insert.. on duplicate key update 命令，可以实现数据库表不会出现重复的记录，还能实现业务的更新逻辑。

4、思考

1）消费者做好优雅退出处理。这是为了尽可能避免消息消费到一半程序退出导致的消息重试。

2）消息消费失败的时候，可以做好监控报警，以便进行人工干预。

3）消费消息的方法，确保在同一个事务，以便消费失败的时候，可以回滚。