图文 76 对订单系统核心流程引入 幂等性机制，保证数据不会重复

详情 评论

对订单系统核心流程引入幂等性机制，保证数据不会重复

继《从零开始带你成为JVM实战高手》后，救火队长携新作再度出山，重磅推荐：

（点击下方蓝字试听）

《从零开始带你成为MySQL实战优化高手》

1、到底什么是幂等性机制?

上一次我们详细的分析了MQ的消息出现重复的问题，那么这一次我们就来分析一下到底应该如何去避免MQ中的消息进行重复处理。

要解决这个问题，我们就要先给大家讲一个概念，叫做幂等性机制。

这个幂等性机制，其实就是用来避免对同一个请求或者同一条消息进行重复处理的机制，所谓的幂等，他的意思就是，比如你有一个接口，然后如果别人对一次请求重试了多次，来调用你的接口，你必须保证自己系统的数据是正常的，不能多出来一些重复的数据，这就

是幂等性的意思。

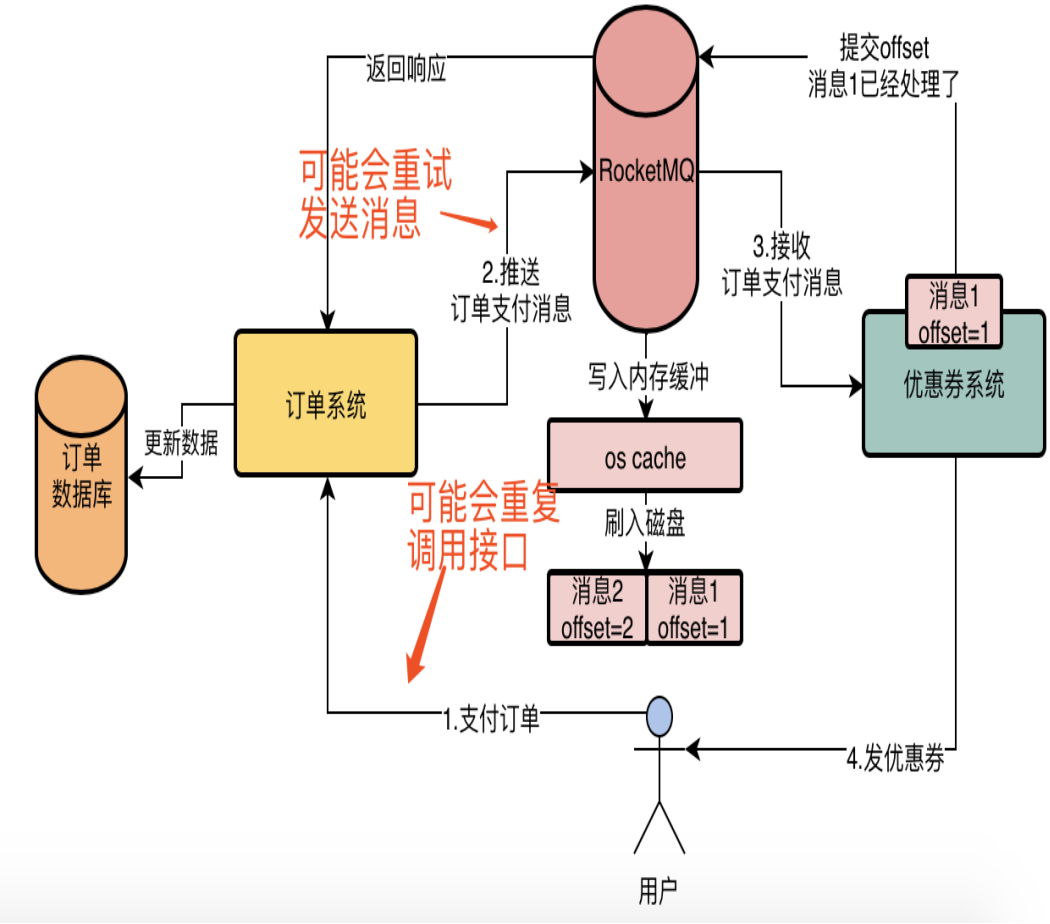
那么对于我们的MQ而言，就是你从MQ里获取消息的时候，要保证对同一个消息只能处理一次，不能重复处理多次，导致出现重复的数据。

因此要解决MQ的消息重复问题，关键就是要引入幂等性机制。

2、发送消息到MQ的时候如何保证幂等性?

现在我们先来看第一个问题，当我们的订单系统发送消息到MQ的时候需要保证幂等性吗?

我们都知道，订单系统的接口可能会被重复调用导致发送重复的消息到MQ去，也可能自己有重试机制导致发送重复的消息到MQ，如下图所示



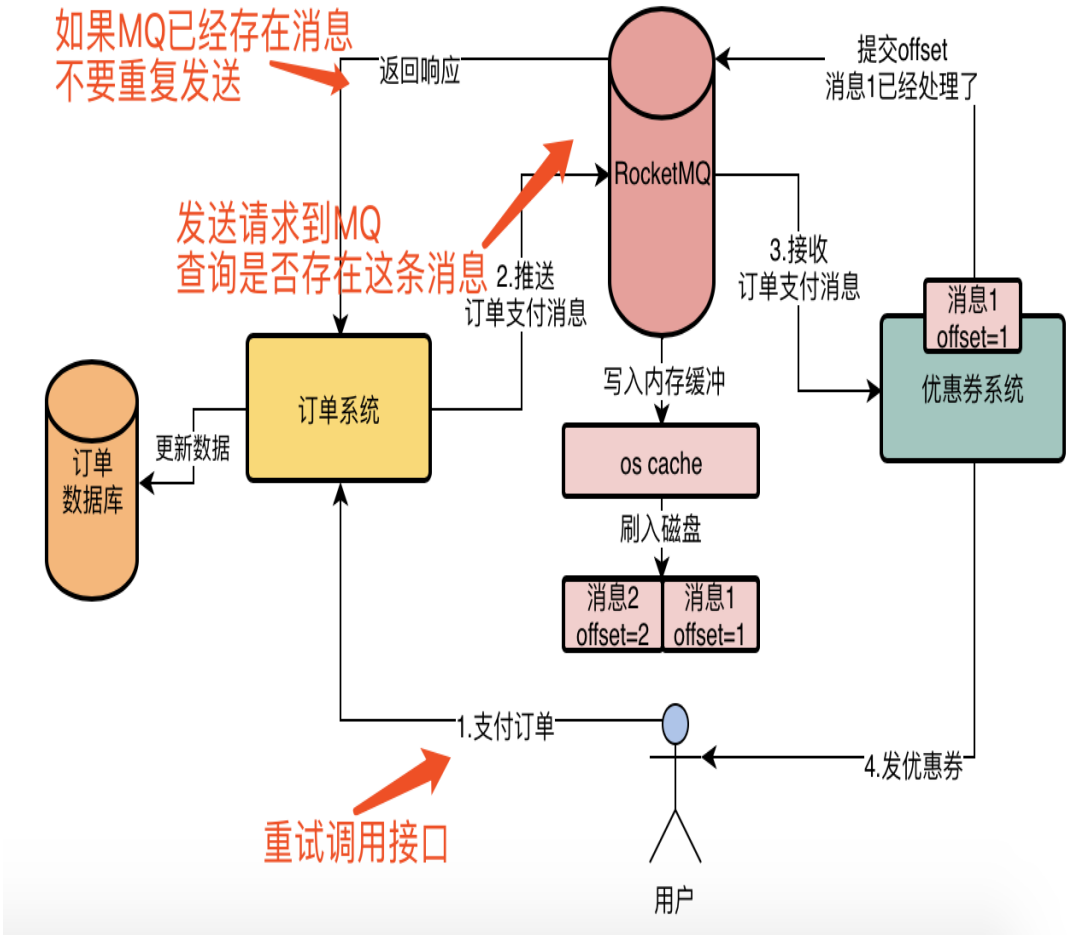
那么我们如果想要让订单系统别发送重复的消息到MQ去，应该怎么做呢?

大体上来说，常见的方案有两种。

第一个方案就是业务判断法，也就是说你的订单系统必须要知道自己到底是否发送过消息到MQ去，消息到底是否已经在MQ里了。

我们举个例子，当支付系统重试调用你的订单系统的接口时，你需要发送一个请求到MQ去，查询一下当前MQ里是否存在针对这个订单的支付消息?

如果MQ告诉你，针对id=1100这个订单的支付成功消息，在我这里已经有了，你之前已经写入进来了，那么订单系统就可以不要再次发送这条消息到MQ去了，我们看下图的示意。



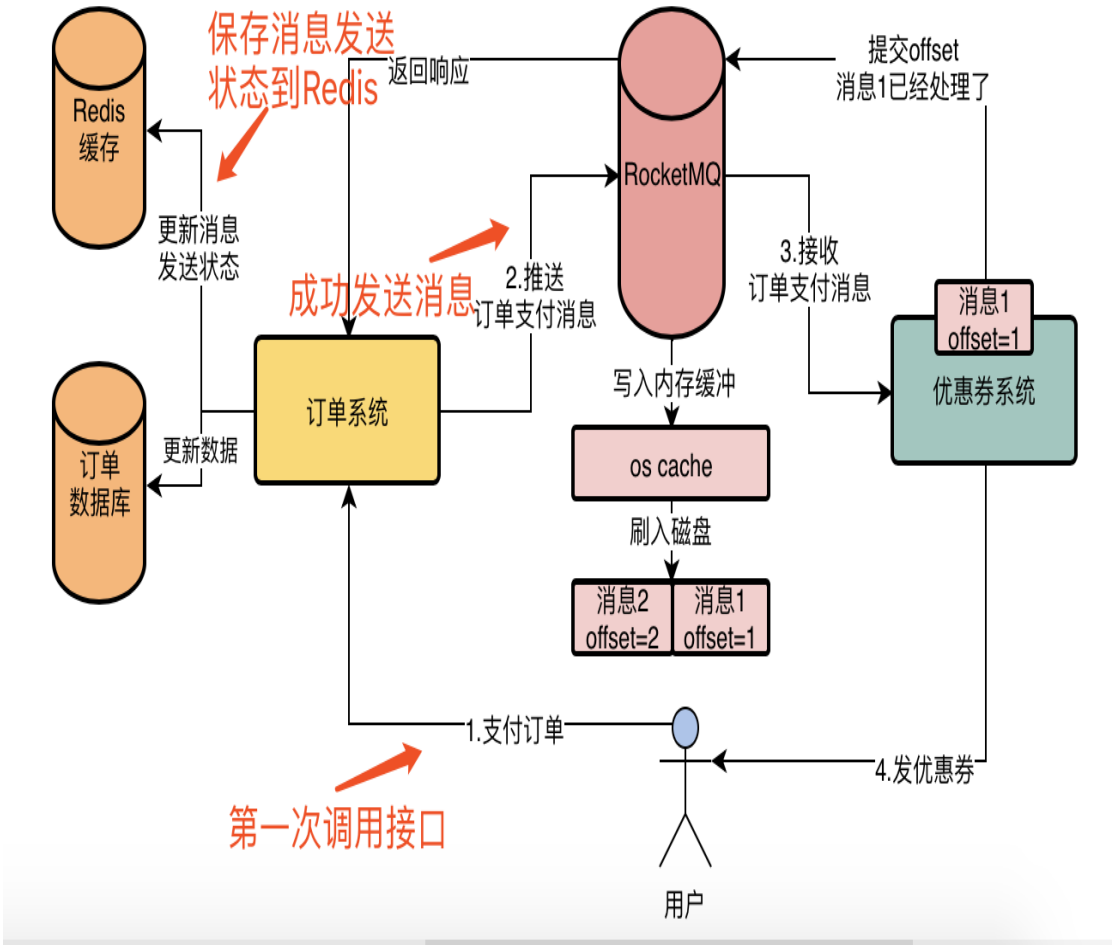
这个业务判断法的核心就在于，你的消息肯定是存在于MQ里的，到底发没发送过，只有MQ知道。如果没发送过这个消息，MQ里肯定没有这个消息，如果发送过这个消息，MQ里肯定给有这个消息。

所以当你的订单系统的接口被重试调用的时候，你这个接口上来就应该发送请求到MQ里去查询一下，比如对订单id=1100这个订单的支付成功消息，在你MQ那里有没有？如果有的话，我就不再重复发送消息了！

3、基于Redis缓存的幂等性机制

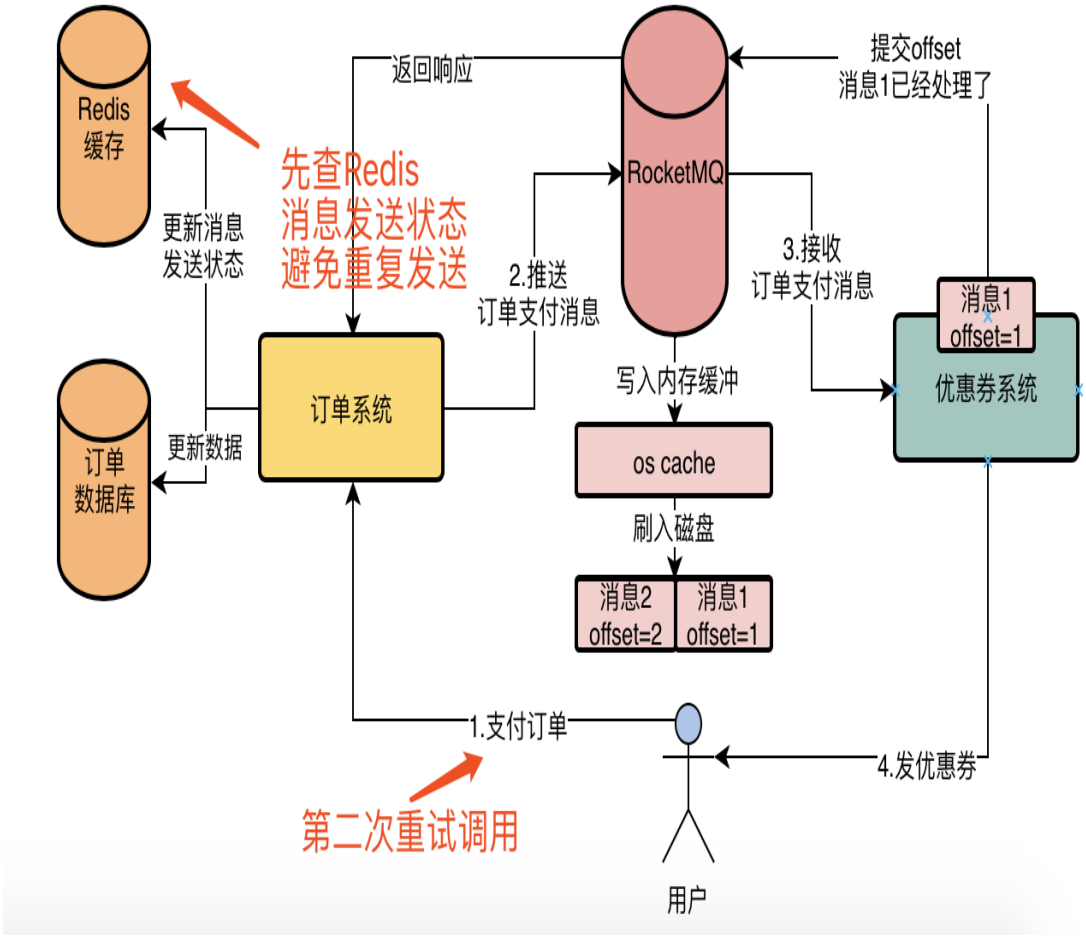
接着我们来讲第二种方法，就是状态判断法

这个方法的核心在于，你需要引入一个Redis缓存来存储你是否发送过消息的状态，如果你成功发送了一个消息到MQ里去，你得在Redis缓存里写一条数据，标记这个消息已经发送过，我们看下图。



那么当你的订单接口被重复调用的时候，你只要根据订单id去Redis缓存里查询一下，这个订单的支付消息是否已经发送给MQ了，如果发送过了，你就别再次发送了！

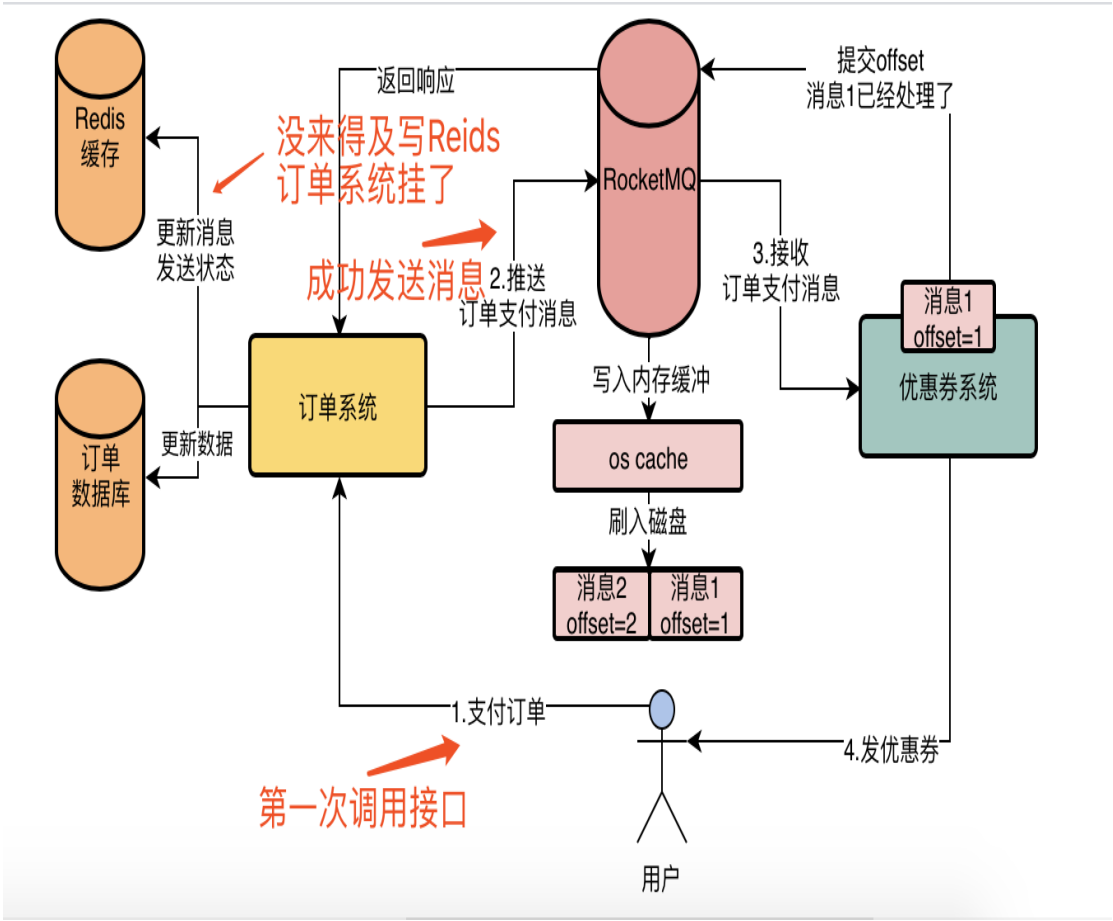
我们看下图的示意。



其实两种幂等性机制都是很常用的，但是大家这里一定要知道一个事情，那就是对于基于Redis的状态判断法，有可能没办法完全做到幂等性。

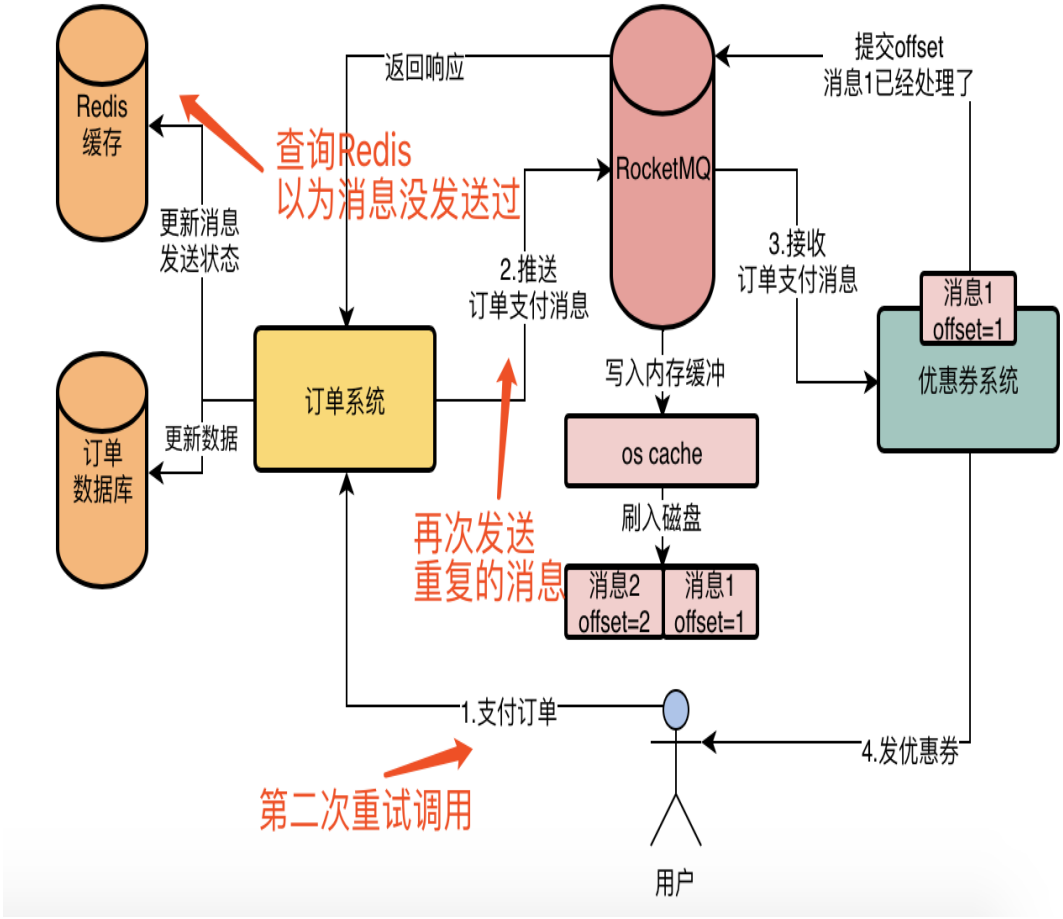
举个例子，你的支付系统发送请求给订单系统，然后已经发送消息到MQ去了，但是此时订单系统突然崩溃了，没来得及把消息发送的状态写入Redis

我们看下图



这个时候如果你的订单系统在其他机器上部署了，或者他重启了，那么这个时候订单系统被重试调用的时候，他去找Redis查询消息发送状态，会以为消息没发送过，然后会再次发送重复消息到MQ去

我们看下图



所以这种方案一般情况下是可以做到幂等性的，但是如果有时候你刚发送了消息到MQ，还没来得及写Redis，系统就挂了，之后你的接口被重试调用的时候，你查Redis还以为消息没发过，就会发送重复的消息到MQ去。

4、有没有必要在订单系统环节保证消息不重复发送?

所以在我们这个场景中，如果在订单系统环节要保证消息不重复发送，要不然就是直接通过查询MQ来判断消息是否发过，要不然就是通过引入Redis来保存消息发送状态。其实这两种方案都不是太好。

因为RocketMQ虽然是支持你查询某个消息是否存在的，这个功能我们后面的案例会讲他的功能使用和底层原理，但是在这个环节你直接从MQ查询消息是没这个必要的，他的性能也不是太好，会影响你的接口的性能。

另外基于Redis的消息发送状态的方案，在极端情况下还是没法100%保证幂等性，所以也不是特别好的一个方案。

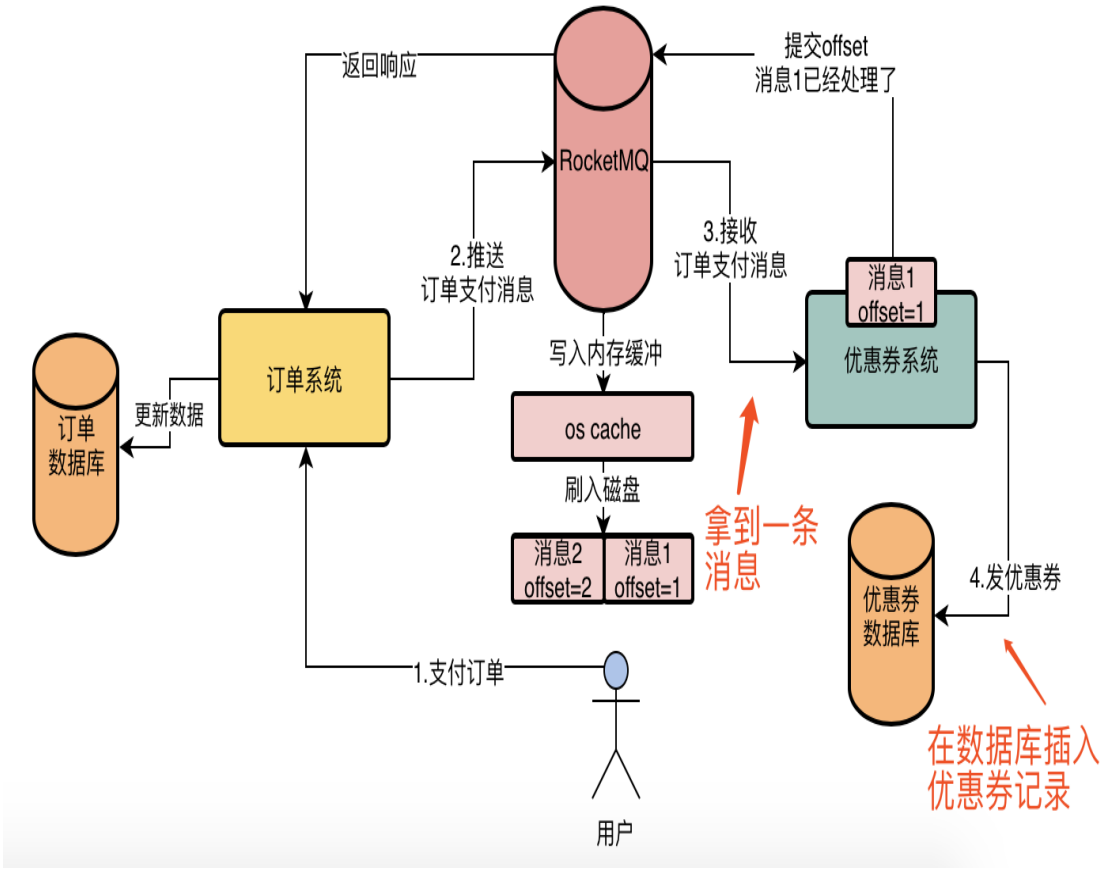
所以对于我们而言，在这里建议是不用在这个环节保证幂等性，也就是我们可以默许他可能会发送重复的消息到MQ里去。

5、优惠券系统如何保证消息处理的幂等性?

接着我们来看优惠券系统假设会拿到重复的消息，那么如何保证消息处理的幂等性?

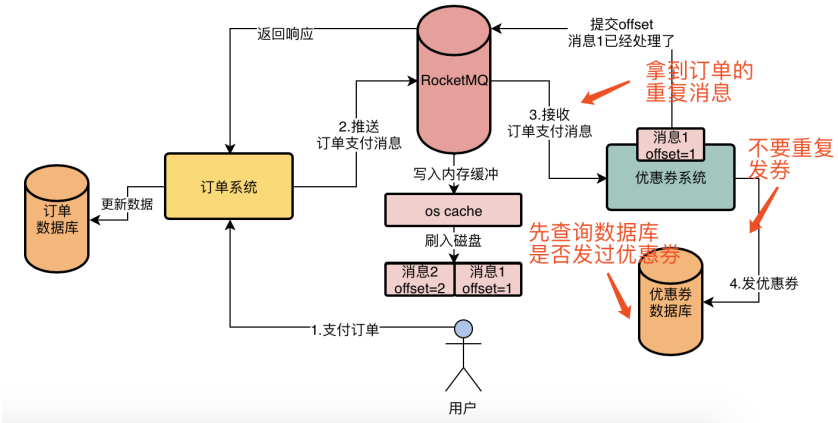
其实这里就比较简单了，直接基于业务判断法就可以了，因为优惠券系统每次拿到一条消息后给用户发一张优惠券，实际上核心就是在数据库里给用户插入一条优惠券记录

我们看下图



那么如果优惠券系统从MQ那里拿到一个订单的两条重复的支付成功消息，这个时候其实很简单，他只要先去优惠券数据库中查询一下，比如对订单id=1100的订单，是否已经发放过优惠券了，是否有优惠券记录，如果有的话，就不要重复发券了！

通过这个业务判断的方法，就可以简单高效的避免消息的重复处理了，我们看下图。



6、MQ消息幂等性的方案总结

一般来说，对于MQ的重复消息问题而言，我们往MQ里重复发送一样的消息其实是还可以接收的，因为MQ里有多条重复消息，他不会对系统的核心数据直接造成影响，但是我们关键要保证的，是你从MQ里获取消息进行处理的时候，必须要保证消息不能重复处理。

这里的话，要保证消息的幂等性，我们优先推荐的其实还是业务判断法，直接根据你的数据存储中的记录来判断这个消息是否处理过，如果处理过了，那就别再次处理了。因为我们要知道，基于Redis的消息发送状态的方案，在一些极端情况下还是没法完全保证幂等性的。

7、小作业：如果给你们的系统设计消息幂等性方案，如何做?

上次给大家留了一个小作业，让大家思考一下自己的系统中是否会出现消息重复的问题。

那么今天给大家留一个小作业，结合你自己的业务思考一下，如果你的消息有重复问题，如何基于业务判断法去保证消息处理的幂等性?

大家可以踊跃在评论区留言自己的方案设计，跟大家一起交流。

End

狸猫技术窝精品专栏及课程推荐：

- 《从零开始带你成为JVM实战高手》
- 《21天互联网Java进阶面试训练营》（分布式篇）
- 《互联网Java工程师面试突击》（第1季）
- 《互联网Java工程师面试突击》（第3季）

重要说明：

如何提问：每篇文章都有评论区，大家可以尽情留言提问，我会逐一答疑

如何加群：购买狸猫技术窝专栏的小伙伴都可以加入狸猫技术交流群，一个非常纯粹的技术交流的地方

具体加群方式，请参见目录菜单下的文档：《付费用户如何加群》（**购买后可见**）