# **如何保证消息不被重复消费?**

要保证消息不被重复消费，需要保证消息消费时的幂等性，保证了幂等性，重复消费了也不会造成系统异常。

幂等性，通俗的说，无论你重复请求多少次，你得确保对应的数据是不会改变的。

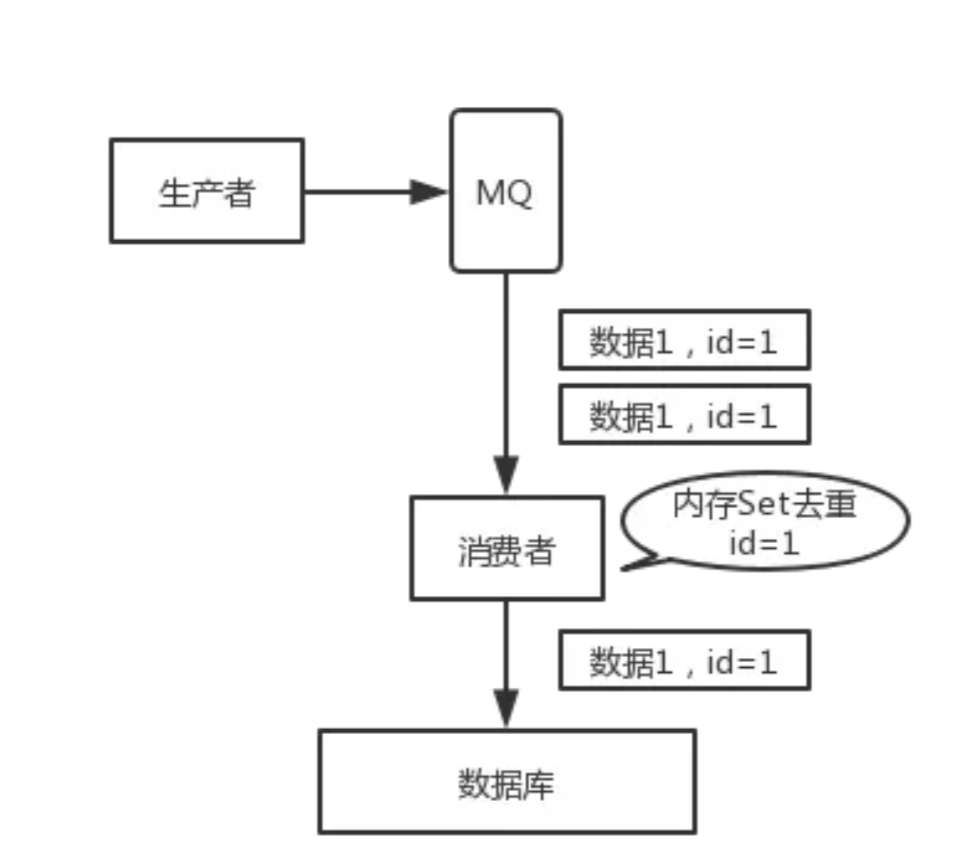
一条数据重复出现两次，数据库里就只有一条数据，这就保证了系统的幂等性。

举个例子吧。假设你有个系统，消费一条消息就往数据库里插入一条数据，要是你一个消息重复两次，你不就插入了两条，这数据不就错了？但是你要是消费到第二次的时候，自己判断一下是否已经消费过了，若是就直接扔了，这样不就保留了一条数据，从而保证了数据的正确性。

**所以第二个问题来了，怎么保证消息队列消费的幂等性？**

其实还是得结合业务来思考，我这里给几个思路：

* 比如你拿个数据要写库，你先根据主键查一下，如果这数据都有了，你就别插入了，update 一下好吧。
* 比如你是写 Redis，那没问题了，反正每次都是 set，天然幂等性。
* 比如你不是上面两个场景，那做的稍微复杂一点，你需要让生产者发送每条数据的时候，里面加一个全局唯一的 id，类似订单 id 之类的东西，然后你这里消费到了之后，先根据这个 id 去比如 Redis 里查一下，之前消费过吗？如果没有消费过，你就处理，然后这个 id 写 Redis。如果消费过了，那你就别处理了，保证别重复处理相同的消息即可。
* 比如基于数据库的唯一键来保证重复数据不会重复插入多条。因为有唯一键约束了，重复数据插入只会报错，不会导致数据库中出现脏数据。



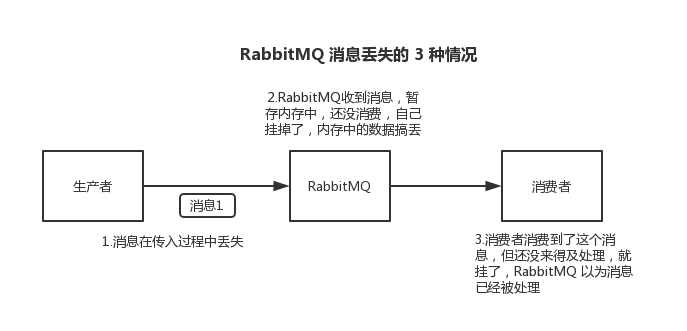
mq-11

当然，如何保证 MQ 的消费是幂等性的，需要结合具体的业务来看。

# 如何处理消息丢失的问题？

数据的丢失问题，可能出现在生产者、MQ、消费者中。

RabbitMQ是怎么处理的?



## 1. 生产者弄丢了数据

         生产者将数据发送到 RabbitMQ 的时候，可能数据就在半路给搞丢了，因为网络问题啥的，都有可能。

         RabbitMQ提供事务机制和confirm模式来确保生产者不丢消息。

         RabbitMQ 事务机制（同步）就是说，发送消息前开启事务，发送过程中如果出现什么异常，事务就会回滚，如果发送成功则提交事务，它的缺点是吞吐量会下降，因为是同步的，提交一个事务之后会阻塞在哪儿，太耗性能。

       所以一般生产上都是用confirm模式的居多。在生产者那里设置开启confirm模式之后，每次写的消息都会分配一个唯一的id，然后如果写入了RabbitMQ中，RabbitMQ会告诉你这个消息是否接受成功，如果接收失败，你可以重试。confirm机制是异步的，你发送消息之后就可以发送下一个消息，然后RabbitMQ接收之后会异步回调你的一个接口通知你这个消息接收到了。

## 消息队列弄丢数据

         这个一般是开启 RabbitMQ 的持久化，写入消息之后会自动持久化到磁盘，哪怕RabbitMQ 挂了，恢复之后会自动读取之前存储的数据，一般数据不会丢。如果RabbitMQ 还没持久化就挂了，这个可以配合confirm模式，你可以在消息持久化后发送给生产者一个Ack确认信号，告诉生产者已经持久成功，如果生产者没有收到确认信号，生产者会自动重发。

**如何进行持久化？**

         1.创建队列时，将持久化标识durable设置为true。

         2. 第二个是发送消息的时候将消息的 deliveryMode设置为 2

       这个就是将消息设置为持久化的。

       必须要同时设置这两个持久化才行，RabbitMQ 哪怕是挂了，再次重启，也会从磁盘上重启恢复 queue，恢复这个 queue 里的数据。

## 消费者丢数据

       启用手动确认模式可以解决这个问题。

① 自动确认模式，消费者挂掉，待ack的消息回归到队列中。消费者抛出异常，消息会不断的被重发，直到处理成功。不会丢失消息，即便服务挂掉，没有处理完成的消息会重回队列，但是异常会让消息不断重试。

② 手动确认模式，如果消费者来不及处理就死掉时，没有响应ack时会重复发送一条信息给其他消费者；如果监听程序处理异常了，且未对异常进行捕获，会一直重复接收消息，然后一直抛异常；如果对异常进行了捕获，但是没有在finally里ack，也会一直重复发送消息(重试机制)。

③ 不确认模式，acknowledge="none" 不使用确认机制，只要消息发送完成会立即在队列移除，无论客户端异常还是断开，只要发送完就移除，不会重发。