
黑马程序员™
www.itheima.com
传智播客旗下
高端IT教育品牌
RabbitMQ

目录 Contents

1. RabbitMQ 内容介绍

RabbitMQ高级特性

- 消息可靠性投递
- Consumer ACK
- 消费端限流
- TTL
- 死信队列
- 延迟队列
- 日志与监控
- 消息可靠性分析与追踪
- 管理

RabbitMQ应用问题

- 消息可靠性保障
- 消息幂等性处理

RabbitMQ集群搭建

- RabbitMQ高可用集群

1. RabbitMQ 高级特性

1.1 消息的可靠投递

在使用 RabbitMQ 的时候，作为消息发送方希望杜绝任何消息丢失或者投递失败场景。RabbitMQ 为我们提供了两种方式用来控制消息的投递可靠性模式。

- confirm 确认模式
- return 退回模式

rabbitmq 整个消息投递的路径为：

producer--->rabbitmq broker--->exchange--->queue--->consumer

- 消息从 producer 到 exchange 则会返回一个 confirmCallback 。
- 消息从 exchange-->queue 投递失败则会返回一个 returnCallback 。

我们将利用这两个 callback 控制消息的可靠性投递

1. RabbitMQ 高级特性

1.1 消息的可靠投递小结

- 设置ConnectionFactory的publisher-confirms="true" 开启 确认模式。
- 使用rabbitTemplate.setConfirmCallback设置回调函数。当消息发送到exchange后回调confirm方法。在方法中判断ack，如果为true，则发送成功，如果为false，则发送失败，需要处理。
- 设置ConnectionFactory的publisher-returns="true" 开启 退回模式。
- 使用rabbitTemplate.setReturnCallback设置退回函数，当消息从exchange路由到queue失败后，如果设置了rabbitTemplate.setMandatory(true)参数，则会将消息退回给producer。并执行回调函数returnedMessage。
- 在RabbitMQ中也提供了事务机制，但是性能较差，此处不做讲解。

使用channel下列方法，完成事务控制：

txSelect(), 用于将当前channel设置成transaction模式
txCommit()，用于提交事务
txRollback(),用于回滚事务

1.2 Consumer Ack

ack指Acknowledge，确认。 表示消费端收到消息后的确认方式。

有三种确认方式：

- 自动确认：acknowledge="none"
- 手动确认：acknowledge="manual"
- 根据异常情况确认：acknowledge="auto"，（这种方式使用麻烦，不作讲解）

其中自动确认是指，当消息一旦被Consumer接收到，则自动确认收到，并将相应 message 从 RabbitMQ 的消息缓存中移除。但是在实际业务处理中，很可能消息接收到，业务处理出现异常，那么该消息就会丢失。如果设置了手动确认方式，则需要在业务处理成功后，调用channel.basicAck()，手动签收，如果出现异常，则调用channel.basicNack()方法，让其自动重新发送消息。

1.2 Consumer Ack 小结

- 在rabbit:listener-container标签中设置acknowledge属性，设置ack方式 none：自动确认，manual：手动确认
- 如果在消费端没有出现异常，则调用channel.basicAck(deliveryTag,false);方法确认签收消息
- 如果出现异常，则在catch中调用 basicNack或 basicReject，拒绝消息，让MQ重新发送消息。

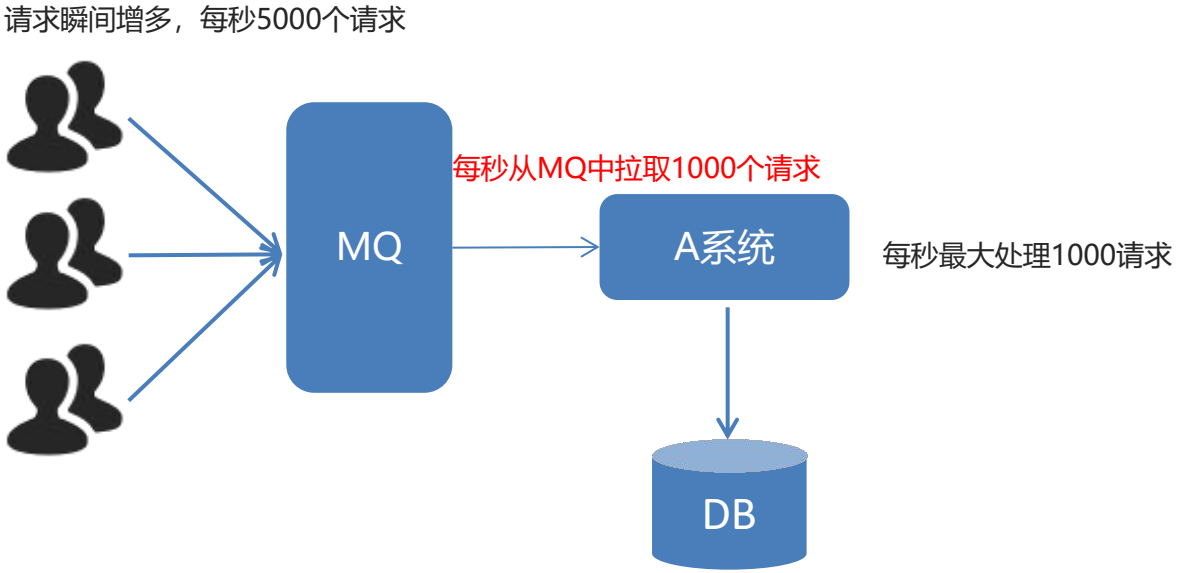
1. RabbitMQ 高级特性

1.2 消息可靠性总结

1. 持久化
    - exchange要持久化
    - queue要持久化
    - message要持久化
2. 生产方确认Confirm
3. 消费方确认Ack
4. Broker高可用

1. RabbitMQ 高级特性

1.3 消费端限流



1. RabbitMQ 高级特性

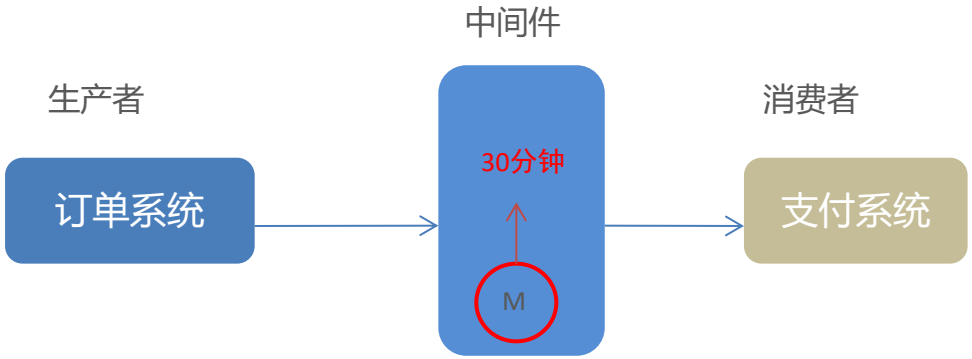
1.3 消费端限流小结

- 在<rabbit:listener-container> 中配置 prefetch属性设置消费端一次拉取多少消息
- 消费端的确认模式一定为手动确认。acknowledge="manual"

1. RabbitMQ 高级特性

1.4 TTL

- TTL 全称 Time To Live（存活时间/过期时间）。
- 当消息到达存活时间后，还没有被消费，会被自动清除。
- RabbitMQ可以对消息设置过期时间，也可以对整个队列（Queue）设置过期时间。



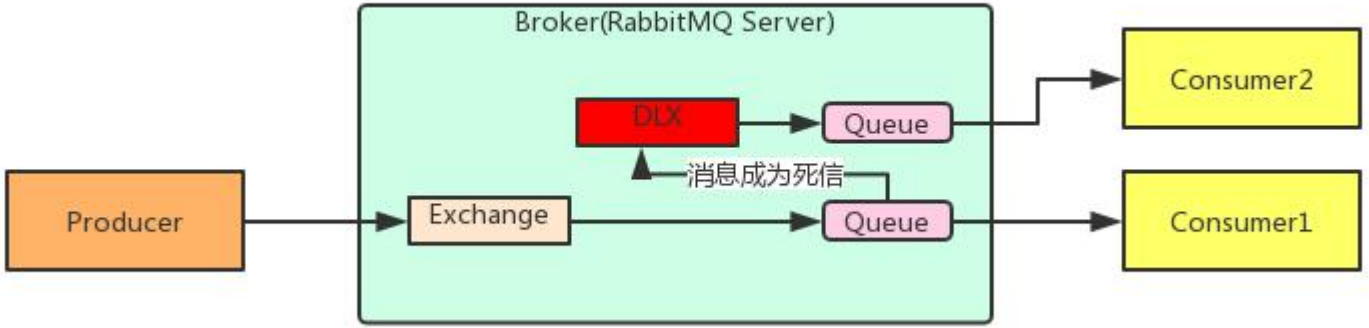
1.4 TTL 小结

- 设置队列过期时间使用参数：x-message-ttl，单位：ms(毫秒)，会对整个队列消息统一过期。
- 设置消息过期时间使用参数：expiration。单位：ms(毫秒)，当该消息在队列头部时（消费时），会单独判断这一消息是否过期。
- 如果两者都进行了设置，以时间短的为准。

1. RabbitMQ 高级特性

1.5 死信队列

死信队列，英文缩写：DLX 。Dead Letter Exchange（死信交换机），当消息成为Dead message后，可以被重新发送到另一个交换机，这个交换机就是DLX。



1. RabbitMQ 高级特性

1.5 死信队列

消息成为死信的三种情况：

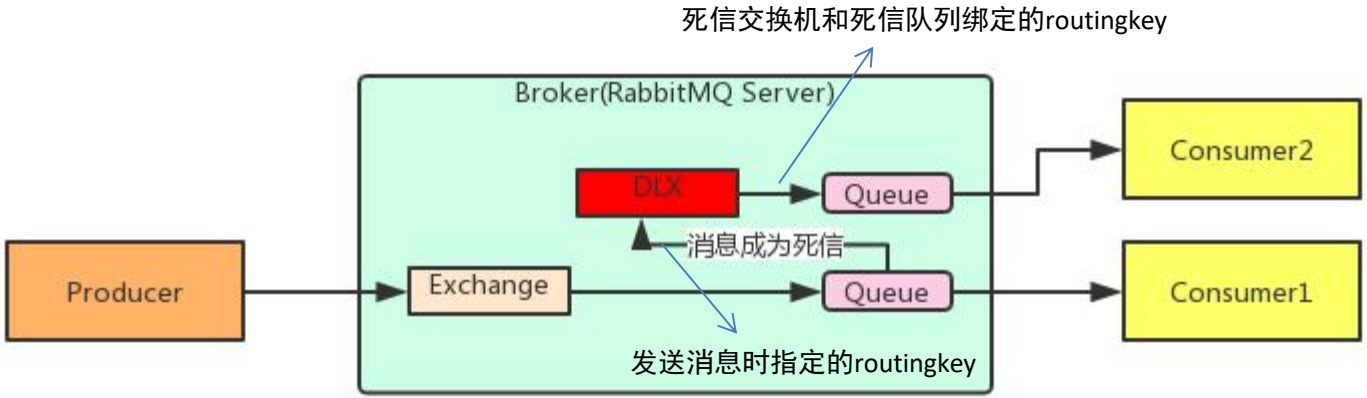
1. 队列消息长度到达限制；
2. 消费者拒接消费消息，basicNack/basicReject,并且不把消息重新放入原目标队列,requeue=false；
3. 原队列存在消息过期设置，消息到达超时时间未被消费；

1. RabbitMQ 高级特性

1.5 死信队列

队列绑定死信交换机：

给队列设置参数： x-dead-letter-exchange 和 x-dead-letter-routing-key



1. RabbitMQ 高级特性

1.5 死信队列小结

1. 死信交换机和死信队列和普通的没有区别

2. 当消息成为死信后，如果该队列绑定了死信交换机，则消息会被死信交换机重新路由到死信队列

3. 消息成为死信的三种情况：
    1. 队列消息长度到达限制；
    2. 消费者拒接消费消息，并且不重回队列；
    3. 原队列存在消息过期设置，消息到达超时时间未被消费；

1. RabbitMQ 高级特性

1.6 延迟队列

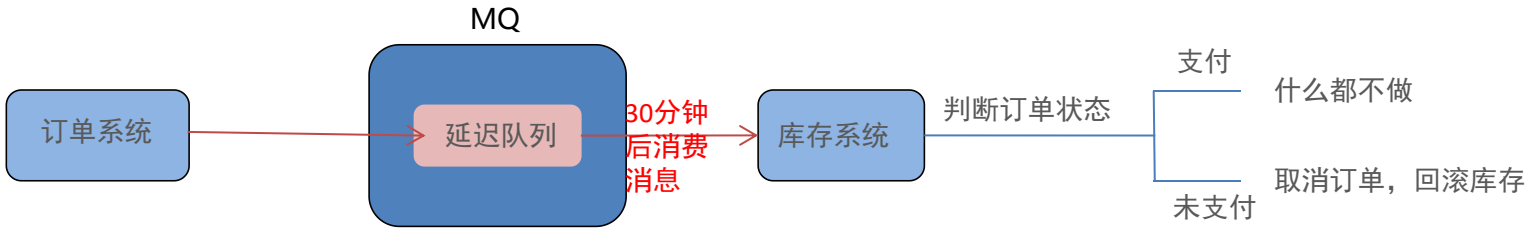
延迟队列，即消息进入队列后不会立即被消费，只有到达指定时间后，才会被消费。

需求：

1. 下单后，30分钟未支付，取消订单，回滚库存。
2. 新用户注册成功7天后，发送短信问候。

实现方式：

1. 定时器
2. 延迟队列

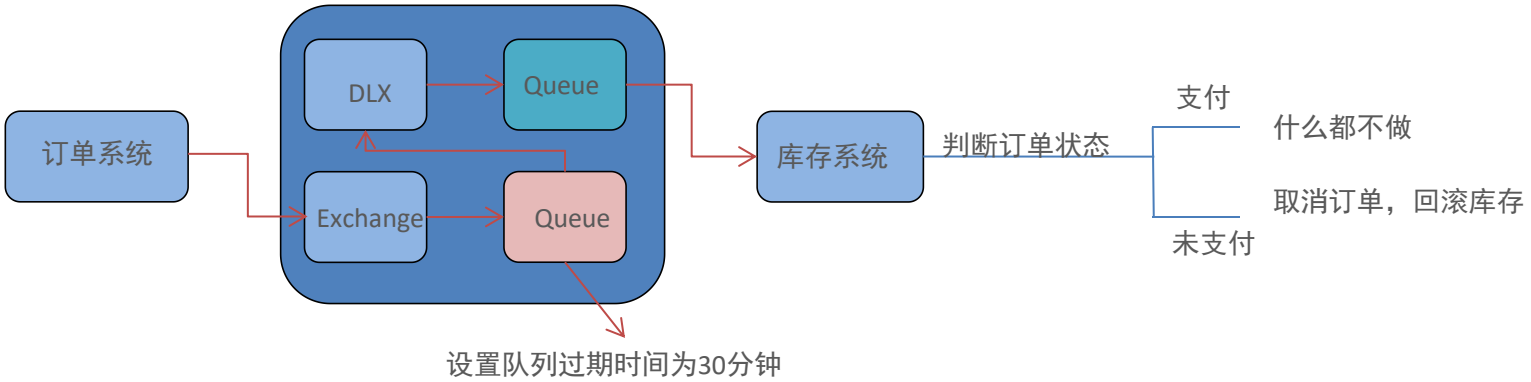


1. RabbitMQ 高级特性

1.6 延迟队列

很可惜，在RabbitMQ中并未提供延迟队列功能。

但是可以使用：TTL+死信队列 组合实现延迟队列的效果。



1. RabbitMQ 高级特性

1.6 延迟队列小结

1. 延迟队列 指消息进入队列后，可以被延迟一定时间，再进行消费。
2. RabbitMQ没有提供延迟队列功能，但是可以使用： TTL + DLX 来实现延迟队列效果。

1.7 日志与监控

1.7.1 RabbitMQ日志

RabbitMQ默认日志存放路径： /var/log/rabbitmq/rabbit@xxx.log

日志包含了RabbitMQ的版本号、Erlang的版本号、RabbitMQ服务节点名称、cookie的hash值、RabbitMQ配置文件地址、内存限制、磁盘限制、默认账户guest的创建以及权限配置等等。

1. RabbitMQ 高级特性

1.7 日志与监控

1.7.2 web管控台监控

1. RabbitMQ 高级特性

1.7 日志与监控

1.7.3 rabbitmqctl管理和监控

查看队列
rabbitmqctl list_queues

查看exchanges
rabbitmqctl list_exchanges

查看用户
rabbitmqctl list_users

查看连接
rabbitmqctl list_connections

查看消费者信息
rabbitmqctl list_consumers

查看环境变量
rabbitmqctl environment

查看未被确认的队列
rabbitmqctl list_queues name messages_unacknowledged

查看单个队列的内存使用
rabbitmqctl list_queues name memory

查看准备就绪的队列
rabbitmqctl list_queues name messages_ready

1. RabbitMQ 高级特性

1.8 消息追踪

在使用任何消息中间件的过程中，难免会出现某条消息异常丢失的情况。对于RabbitMQ而言，可能是因为生产者或消费者与RabbitMQ断开了连接，而它们与RabbitMQ又采用了不同的确认机制；也有可能是因为交换器与队列之间不同的转发策略；甚至是交换器并没有与任何队列进行绑定，生产者又不感知或者没有采取相应的措施；另外RabbitMQ本身的集群策略也可能导致消息的丢失。这个时候就需要有一个较好的机制跟踪记录消息的投递过程，以此协助开发和运维人员进行问题的定位。

在RabbitMQ中可以使用Firehose和rabbitmq_tracing插件功能来实现消息追踪。

1. RabbitMQ 高级特性

1.8 消息追踪-Firehose

firehose的机制是将生产者投递给rabbitmq的消息，rabbitmq投递给消费者的消息按照指定的格式发送到默认的exchange上。这个默认的exchange的名称为amq.rabbitmq.trace，它是一个topic类型的exchange。发送到这个exchange上的消息的routing key为 publish.exchangename 和 deliver.queuename。其中exchangename和queuename为实际exchange和queue的名称，分别对应生产者投递到exchange的消息，和消费者从queue上获取的消息。

注意：打开 trace 会影响消息写入功能，适当打开后请关闭。

rabbitmqctl trace_on：开启Firehose命令

rabbitmqctl trace_off：关闭Firehose命令

1. RabbitMQ 高级特性

1.8 消息追踪-rabbitmq_tracing

rabbitmq_tracing和Firehose在实现上如出一辙，只不过rabbitmq_tracing的方式比Firehose多了一层GUI的包装，更容易使用和管理。

启用插件：rabbitmq-plugins enable rabbitmq_tracing

目录 Contents

2. RabbitMQ 应用问题

RabbitMQ应用问题

1. 消息可靠性保障
    - 消息补偿机制
2. 消息幂等性保障
    - 乐观锁解决方案

2. RabbitMQ 应用问题

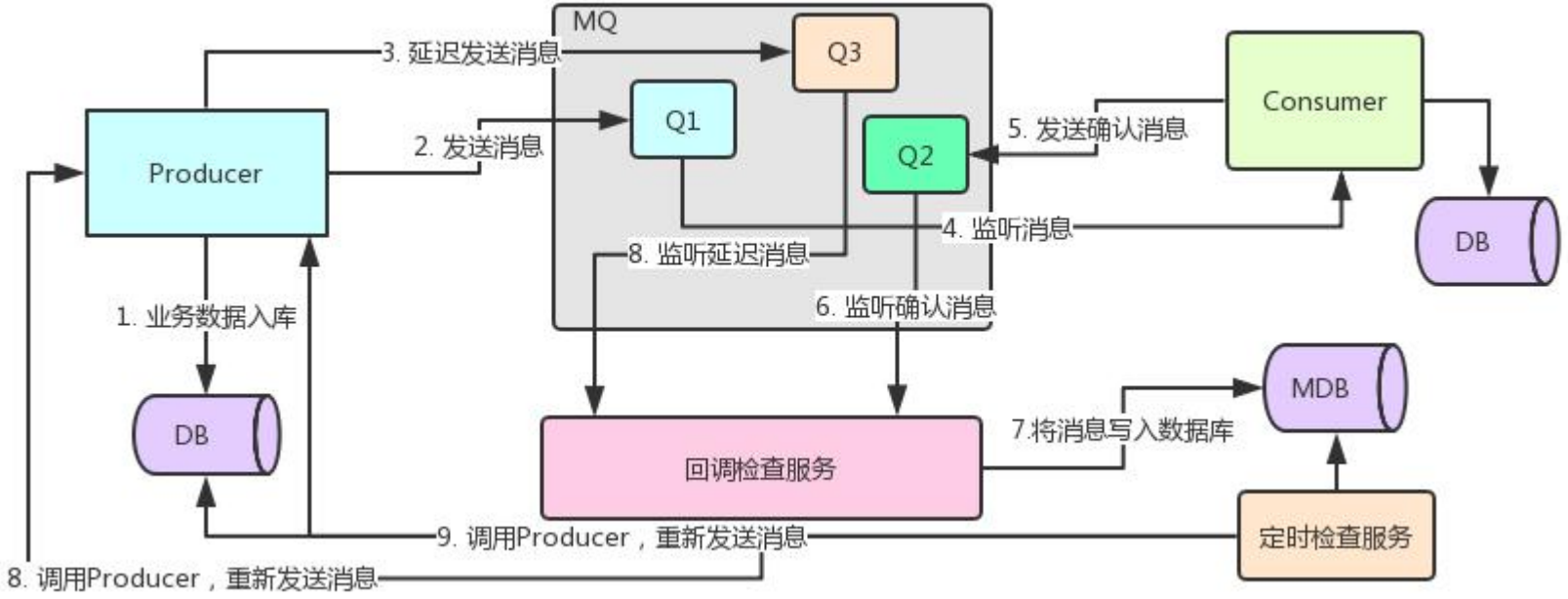
2.1 消息可靠性保障

需求:

100%确保消息发送成功

2. RabbitMQ 应用问题

2.1 消息可靠性保障--消息补偿



2. RabbitMQ 应用问题

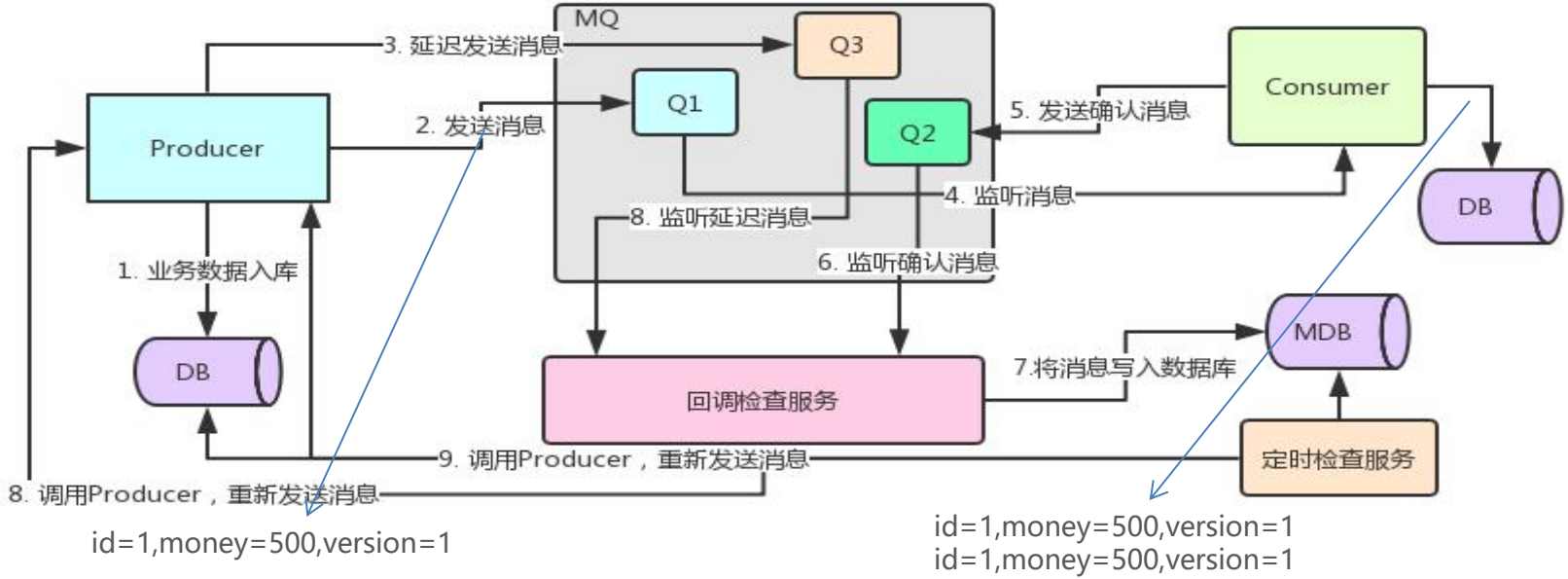
2.2 消息幂等性保障

幂等性指一次和多次请求某一个资源，对于资源本身应该具有同样的结果。也就是说，其任意多次执行对资源本身所产生的影响均与一次执行的影响相同。

在MQ中指，消费多条相同的消息，得到与消费该消息一次相同的结果。

2. RabbitMQ 应用问题

2.2 消息幂等性保障--乐观锁机制



第一次执行：version=1
update account set money = money - 500 , version = version + 1
where id = 1 and version = 1

第二次执行：version=2
update account set money = money - 500 , version = version + 1
where id = 1 and version = 1

传智播客旗下高端IT教育品牌