数据库和缓存双写最终一致性补偿方案

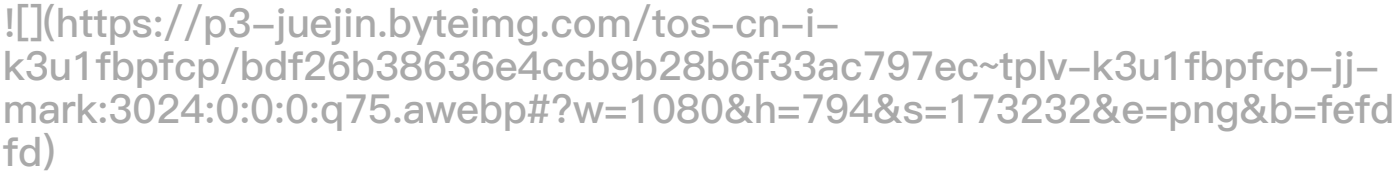
> 在高并发的场景下，数据库和缓存（比如Redis）双写可能会发生数据不一致

> **`这个问题在面试或工作中经常碰到。常见的四种方案相信大家都已经熟悉，这里也不在赘述。不清楚的可以看我这一期视频号：数据库和缓存双写一致性。今天文章主要讲的是我们普遍采用的“先更新数据库，再删除缓存”这种方案下，缓存删失败了，有哪些补偿重试机制。`**

1. 接口重试方案

通过spring的@Retry注解或者Guava提供的重试方法功能，进行接口自动重试。这种对于接口并发量不是很大的情况可以这么用，如果接口并发量很高的情况下，你在接口中直接同步重试，可能有点影响接口性能。

2. 重试记录表方案

将重试数据写表，然后使用分布式定时任务进行重试；使用定时任务重试的话，有个缺点就是实时性没那么高，对于实时性要求特别高的业务场景，该方案不太适用。但是对于一般场景，还是可以满足的。但它有一个很大的优点，即数据是落库的，不会丢数据。
具体方案见下图：


3. 消息队列MQ方案

RocketMQ消息重试机制：

1. 消息消费失败;
2. 消息重新投递回 Broker;
3. 利用RocketMQ消息重新投递机制，如果达到重试次数后消息还没有成功被消费，直接进入死信队列主题（`%DLQ%+消费组名称`）
4. 否则进入重试队列主题（`%RETRY%+消费组名称`），按照MQ延迟级别依次进行消费

具体方案见下图：

当然这里还有个问题：就是左边MQ生产者这里，如果发送消息到MQ失败了，怎么办?
因为发生发送消息到MQ失败的概率还是比较小的，所以这里最简单的方法就是加spring的@Retry注解或者Guava提供的重试方法进行再次重试处理。

4. 监听Binlog方案
==========

前面我们聊过的，无论是定时任务，还是mq（消息队列），做重试机制，对业务都有一定的侵入性。在使用定时任务的方案中，需要在业务代码中增加额外逻辑，如果删除缓存失败，需要将数据写入重试表。而使用mq的方案中，如果删除缓存失败了，需要在业务代码中发送mq消息到mq服务器。其实，还有一种方案，即利用canal等中间件监听binlog，最后我们订阅binlog数据消费。

具体方案见下图：

在binlog订阅者中做缓存删除工作时，如果按照图中的方案进行删除缓存，只删除了一次，也可能会失败。所以这里就需要加上前面聊过的重试机制了：如果删除缓存失败，写入重试表，使用定时任务重试；或者写入mq，让mq自动重试。
这里再列举个使用mq自动重试机制的方案：

在binlog订阅者中如果删除缓存失败，则发送一条mq消息到mq服务器，在mq消费者中自动重试5次。如果有任意一次成功，则直接返回成功；否则重试5次后还是失败，**可设置达到MQ最大重试次数后，则该消息自动被放入死信

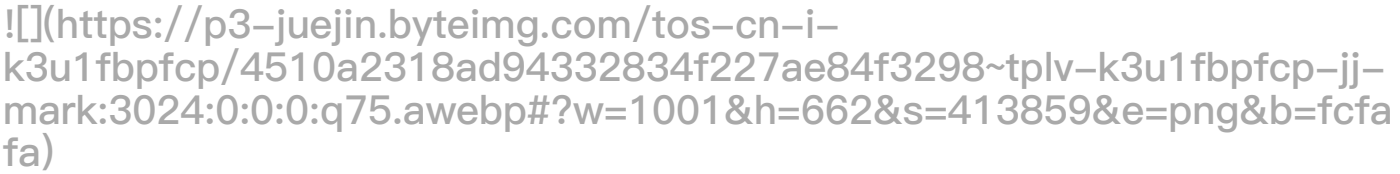
队列**，后面再进行人工介入或补偿处理死信队列消息。

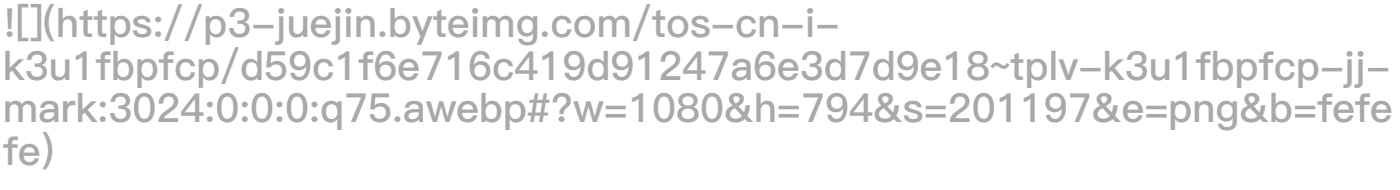
5. 数据库字段标识方案（用Redis做队列场景）
==========

我们当前业务下，需要实现给审核员派单这么一种场景，需要使用到Redis实现队列的数据结构，将数据放入到Redis物理队列中实现出队分派；如果修改了队列其他信息（比如说派单规则），则需要删除Redis队列缓存全部案件清空，如果删除缓存失败，可以利用（`案件单表字段version + 定时任务重试`）的方式，保证数据最终一致性。

如下图，先操作数据库，每次更新数据库时，版本号都会+1，并且状态更新为初始态，然后再删除Redis缓存队列。这样的话即使缓存删除失败（Redis队列中老的记录没被删除），只要数据库的已经更新成功，后面业务再消费redis队列记录的时候，可以通过version版本号进行判断记录是否过期或者是否是重复的数据。

后面通过定时任务捞取状态为初始态的记录，把最新的记录也重新放回Redis队列; 这时候如果Redis有新老数据共存，则比较最新版本号判断是否过期或者重复的数据，将其丢弃后继续重新获取Redis队列的下一个数据，最终只保证最新的记录能被消费处理。



具体方案见下图：


更多优质技术分享，请下面[老马Hony]公众号

原文链接: https://juejin.cn/post/7383363236682039331