数据库与缓存一致性解决方案

1. **概述**

我们读取数据时，采用的原则为：“先缓存后数据库”。即先从缓存中获取数据，如果缓存中不存在（缓存的key不存在）时，从数据库获取数据，并且写入缓存中。当我们更新数据时，无论我们是先更新数据库再更新缓存，还是先更新缓存再更新数据库，都可能会出现缓存不一致的情况。有的人说更新缓存时，可以采用先删除缓存，然后更新数据库，对于这种情况，也可能出现缓存不一致的情况。

1. **异常情况分析**
2. 先更新数据库再更新缓存：

当更新数据时，如更新某商品的库存，当前商品的库存是100，现在要更新为99，先更新数据库更改成99，然后更新缓存，发现更新缓存失败了，这意味着数据库存的是99，而缓存是100，这导致数据库和缓存不一致。

1. 先更新缓存再更新数据库：

当更新数据时，如更新某商品的库存，当前商品的库存是100，现在要更新为99，先更新缓存库存为99，然后更新数据库，发现数据库操作失败了，这意味着数据库存的是100，而缓存是99，这导致数据库和缓存不一致。

1. 先删除缓存再更新数据库，这里需要考虑2个问题：

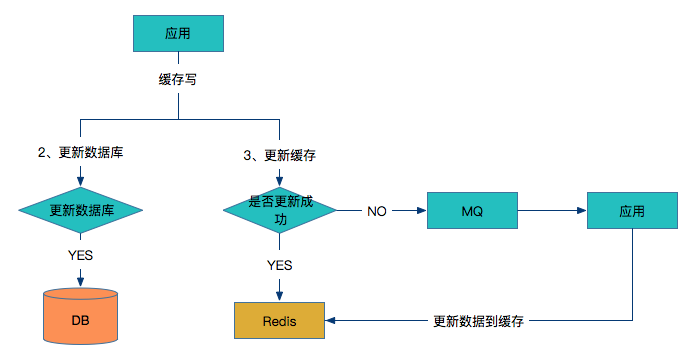
* 在高并发的情况下，如果当删除完缓存的时候，这时去更新数据库，但还没有更新完，另外一个请求来查询数据，发现缓存里没有，就去数据库里查，还是以上面商品库存为例，如果数据库中产品的库存是100，那么查询到的库存是100，然后插入缓存，插入完缓存后，原来那个更新数据库的线程把数据库更新为了99，导致数据库与缓存不一致的情况。

解决办法：遇到这种情况，可以用队列的去解决这个问题，创建几个队列，如20个，根据商品的ID去做hash值，然后对队列个数取摸，当有数据更新请求时，先把它丢到队列里去，当更新完后在从队列里去除，如果在更新的过程中，遇到以上场景，先去缓存里看下有没有数据，如果没有，可以先去队列里看是否有相同商品ID在做更新，如果有也把查询的请求发送到队列里去，然后同步等待缓存更新完成。  
这里有一个优化点，如果发现队列里有一个查询请求了，那么就不要放新的查询操作进去了，用一个while（true）循环去查询缓存，循环个200MS左右，如果缓存里还没有则直接取数据库的旧数据，一般情况下是可以取到的。

* 在高并发情况下，如果数据库是读写分离，写库更新成功，缓存也删除了，但是读库还有没同步到数据，这时从读库中读取的数据会是旧的数据，再用这个旧的数据加入缓存，就会导致缓存的数据与数据库中的数据不一致。

如果我们在更新数据库之后，最后加一步立即更新缓存的操作，如果更新失败，会存在上面说的2个问题。这里需要注意的是，在高并发情况下，删除缓存是存在缓存穿透的风险。

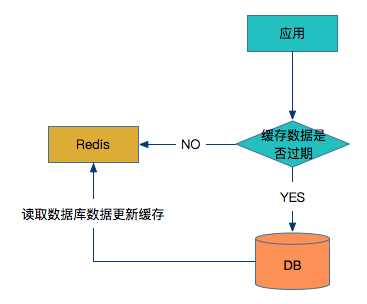
1. **解决方案之重客户端**



* 应用同时更新数据库和缓存
* 如果数据库更新成功，则开始更新缓存，否则如果数据库更新失败，则整个更新过程失败。
* 判断更新缓存是否成功，如果成功则返回
* 如果缓存没有更新成功，则将数据发到MQ中
* 应用监控MQ通道，收到消息后继续更新Redis。

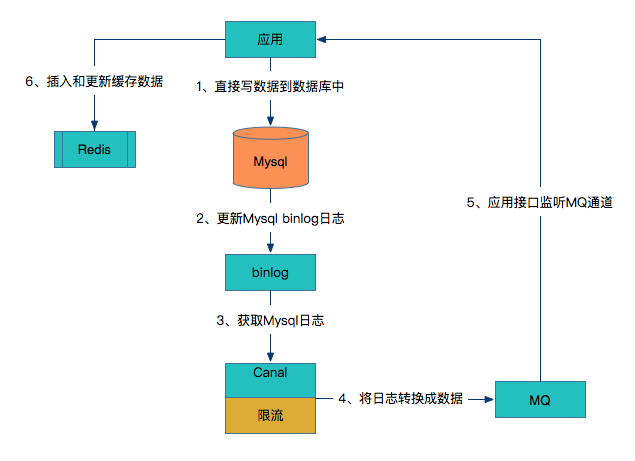
**问题点：**如果更新Redis失败，同时在将数据发到MQ之前的时间，应用重启了，这时候MQ就没有需要更新的数据，如果Redis对所有数据没有设置过期时间，同时在读多写少的场景下，只能通过人工介入来更新缓存。

如何来解决这个问题？那么在写入Redis数据的时候，在数据中增加一个时间戳插入到Redis中。在从Redis中读取数据的时候，首先要判断一下当前时间有没有过期，如果没有则从缓存中读取，如果过期了则从数据库中读取最新数据覆盖当前Redis数据并更新时间戳。具体过程如下图所示：



1. **解决方案之数据库与缓存解耦**

上述方案对于应用的研发人员来讲比较重，需要研发人员同时考虑数据库和Redis是否成功来做不同方案，如何让研发人员只关注数据库层面，而不用关心缓存层呢？请看下图：



* 应用直接写数据到数据库中。
* 数据库更新binlog日志。
* 利用Canal中间件读取binlog日志。
* Canal借助于限流组件按频率将数据发到MQ中。
* 应用监控MQ通道，将MQ的数据更新到Redis缓存中。

可以看到这种方案对研发人员来说比较轻量，不用关心缓存层面，而且这个方案虽然比较重，但是却容易形成统一的解决方案。

1. **附录**

<https://blog.csdn.net/simba_1986/article/details/77823309>

https://www.jianshu.com/p/fbe6a7928229?utm\_source=oschina-app