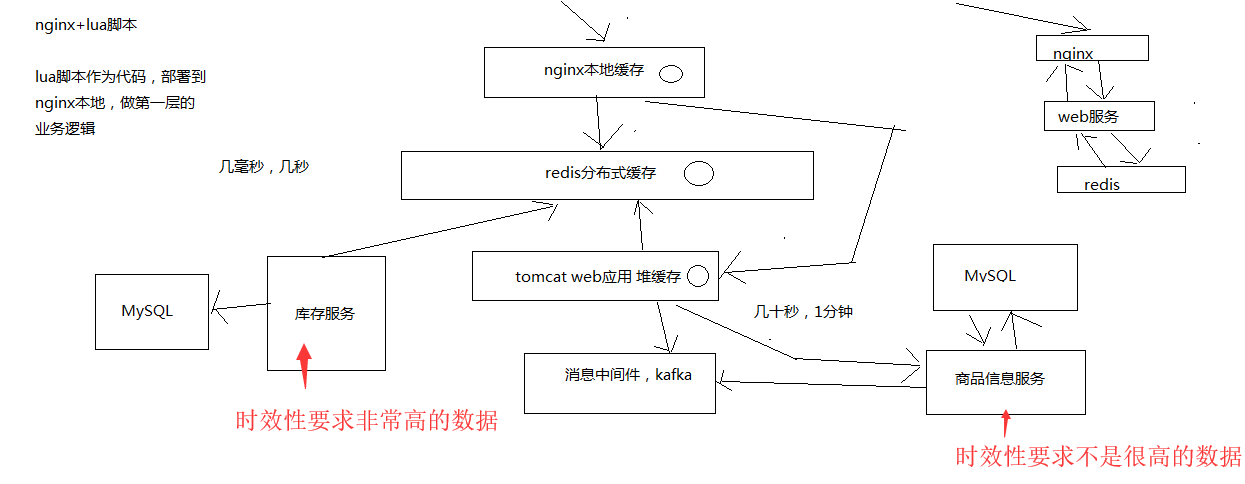
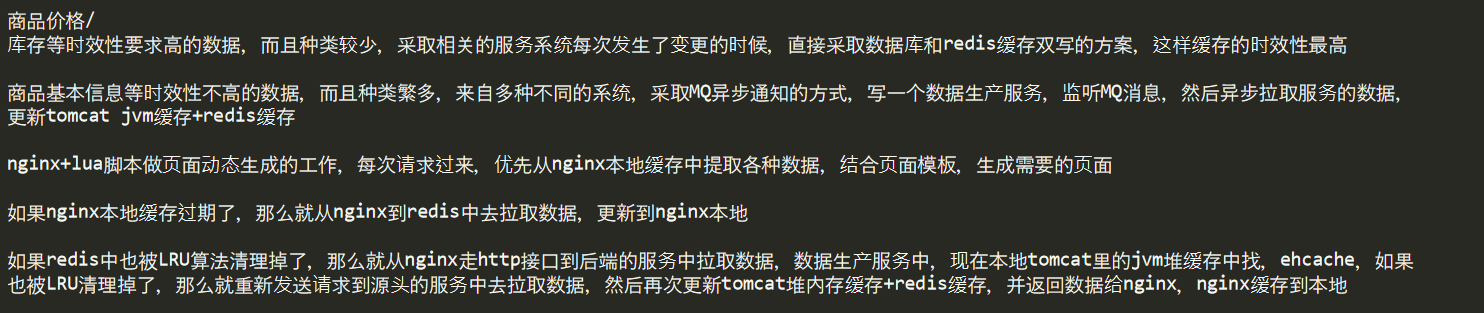
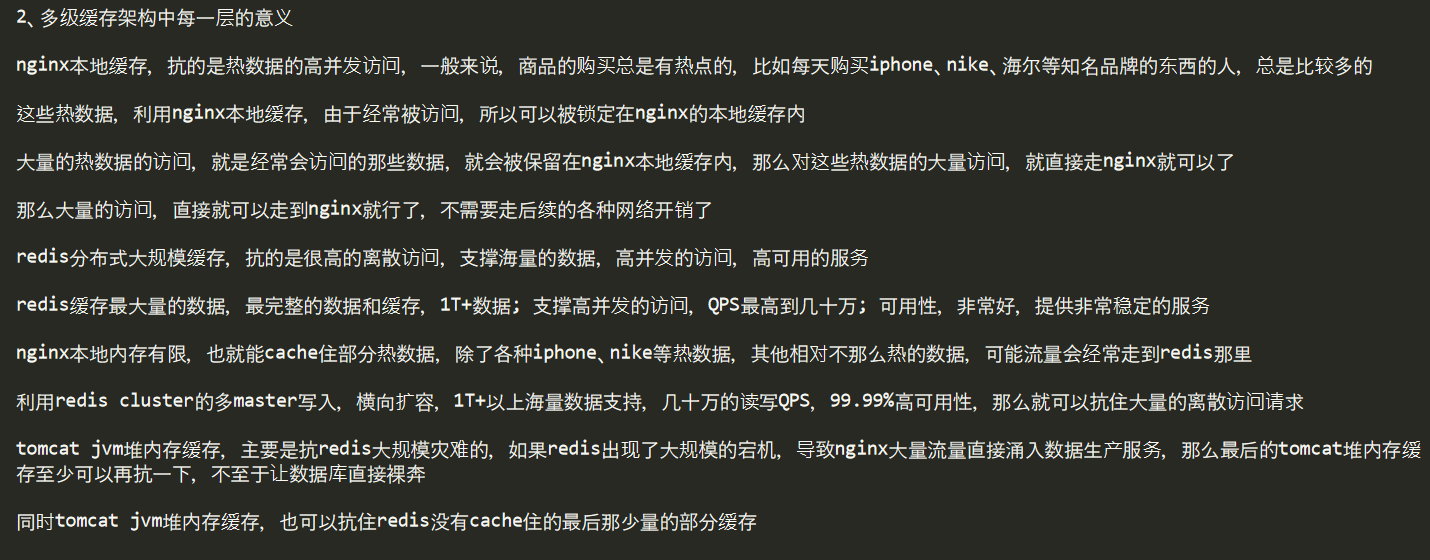
1. 多级缓存架构简介
   1. 商品详情页系统的多级缓存架构





* 1. 每一层的意义



* 1. Cache Aside Pattern

读的时候，先读缓存，缓存没有的话，那么就读数据库，然后取出数据后放入缓存，同时返回响应

更新的时候，先删除缓存，然后再更新数据库

为什么是删除缓存，而不是更新缓存呢？

原因很简单，很多时候，复杂点的缓存的场景，因为缓存有的时候，不简单是数据库中直接取出来的值，更新缓存的代价是很高的。28法则，黄金法则，20%的数据，占用了80%的访问量。

其实删除缓存，而不是更新缓存，就是一个lazy计算的思想，不要每次都重新做复杂的计算，不管它会不会用到，而是让它到需要被使用的时候再重新计算

1. 缓存+数据库双写不一致问题分析与解决方案设计
   1. 问题：先修改数据库，再删除缓存，如果删除缓存失败了，那么会导致数据库中是新数据，缓存中是旧数据，数据出现不一致

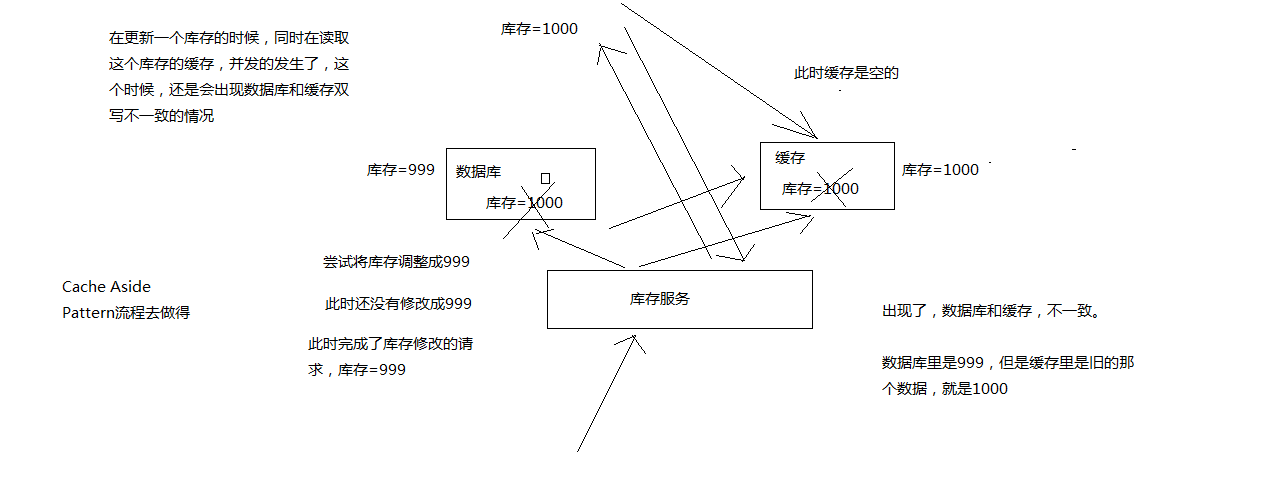
解决思路：

先删除缓存，再修改数据库，如果删除缓存成功了，如果修改数据库失败了，那么数据库中是旧数据，缓存中是空的，那么数据不会不一致

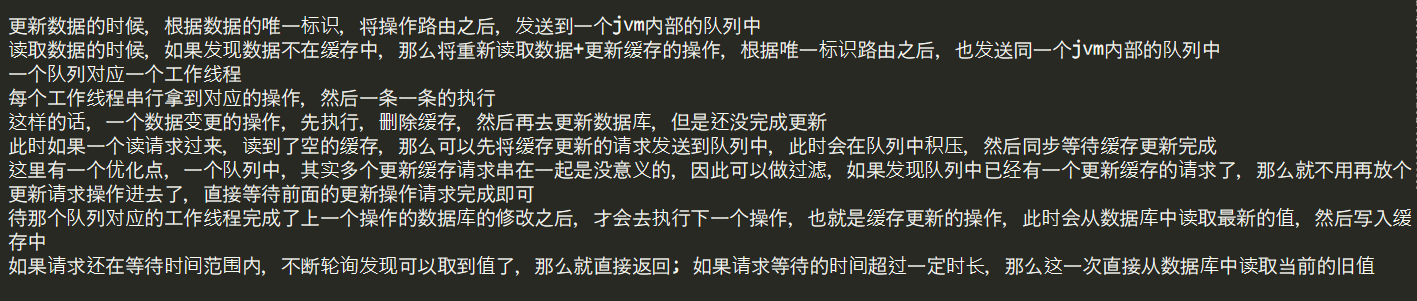
* 1. 读写并发的时候复杂的数据库+缓存双写不一致的场景

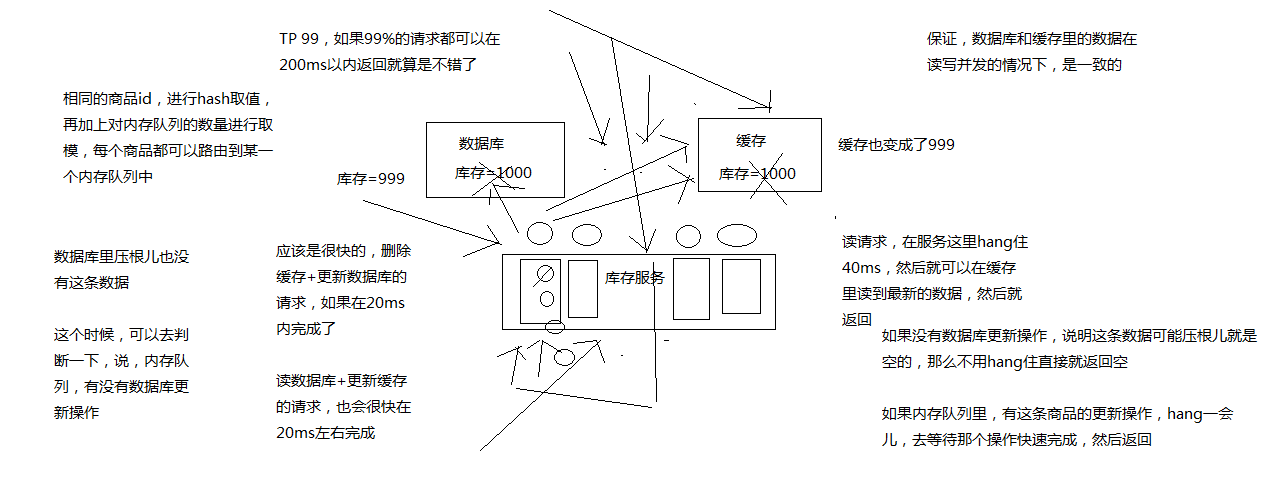
数据发生了变更，先删除了缓存，然后要去修改数据库，此时还没修改

一个请求过来，去读缓存，发现缓存空了，去查询数据库，查到了修改前的旧数据，放到了缓存中



解决方案：





[高并发的场景下，该解决方案要注意的问题](双写不一致解决方案需要注意的问题.txt)