# 缓存不一致性

## 引用缓存的好处

   1）减缓数据库压力；2）提高系统并发处理能力

## 引用缓存的问题

        1）处理逻辑变得复杂；2）使用不当，容易引起缓存和数据库数据不一致的问题

## 数据不一致的原因

        缓存操作与数据库操作不是原子操作，当一方操作成功、另一方操作失败时就会造成数据不一致问题。

## 更新策略

* 1. 先更新数据库，再更新缓存
  2. 先删除缓存，再更新数据库
  3. 先更新数据库，再删除缓存

### 4.1先更新数据库，再更新缓存

这种方式会出现线程安全问题。

例如，同时有请求A和请求B进行更新操作，那么会出现

（1）线程A更新了数据库

（2）线程B更新了数据库

（3）线程B更新了缓存

（4）线程A更新了缓存

这就出现请求A更新缓存应该比请求B更新缓存早才对，但是因为网络等原因，B却比A更早更新了缓存。这就导致了脏数据，因此不考虑。

### 4.2先删缓存，再更新数据库

该方案会导致不一致的原因是。同时有一个请求A进行更新操作，另一个请求B进行查询操作。那么会出现如下情形:

（1）请求A进行写操作，删除缓存

（2）请求B查询发现缓存不存在

（3）请求B去数据库查询得到旧值

（4）请求B将旧值写入缓存

（5）请求A将新值写入数据库

上述情况就会导致不一致的情形出现。而且，如果不采用给缓存设置过期时间策略，该数据永远都是脏数据。

那么，如何解决呢？采用延时双删策略

伪代码如下

|  |
| --- |
| public void write(String key,Object data){  redis.delKey(key);  db.updateData(data);  Thread.sleep(1000);  redis.delKey(key);  } |

转化为中文描述就是

（1）先淘汰缓存

（2）再写数据库（这两步和原来一样）

（3）休眠1秒，再次淘汰缓存

这么做，可以将1秒内所造成的缓存脏数据，再次删除。

那么，这个1秒怎么确定的，具体该休眠多久呢？

针对上面的情形，读者应该自行评估自己的项目的读数据业务逻辑的耗时。然后写数据的休眠时间则在读数据业务逻辑的耗时基础上，加几百ms即可。这么做的目的，就是确保读请求结束，写请求可以删除读请求造成的缓存脏数据。

采用这种同步淘汰策略，吞吐量降低怎么办？

那就将第二次删除作为异步的。自己起一个线程，异步删除。这样，写的请求就不用沉睡一段时间后了，再返回。这么做，加大吞吐量。

### 4.3先更新数据库，再删缓存

假设这会有两个请求，一个请求A做查询操作，一个请求B做更新操作，那么会有如下情形产生

（1）缓存刚好失效

（2）请求A查询数据库，得一个旧值

（3）请求B将新值写入数据库

（4）请求B删除缓存

（5）请求A将查到的旧值写入缓存

如果发生上述情况，确实是会发生脏数据。

发生上述情况有一个先天性条件，就是步骤（3）的写数据库操作比步骤（2）的读数据库操作耗时更短，才有可能使得步骤（4）先于步骤（5）。可是，大家想想，数据库的读操作的速度远快于写操作的（不然做读写分离干嘛，做读写分离的意义就是因为读操作比较快，耗资源少），因此步骤（3）耗时比步骤（2）更短，这一情形很难出现。

一定要解决怎么办？如何解决上述并发问题？

首先，给缓存设有效时间是一种方案。其次，采用策略（2）里给出的异步延时删除策略，保证读请求完成以后，再进行删除操作。

### 4.4还有其他造成不一致的原因么？

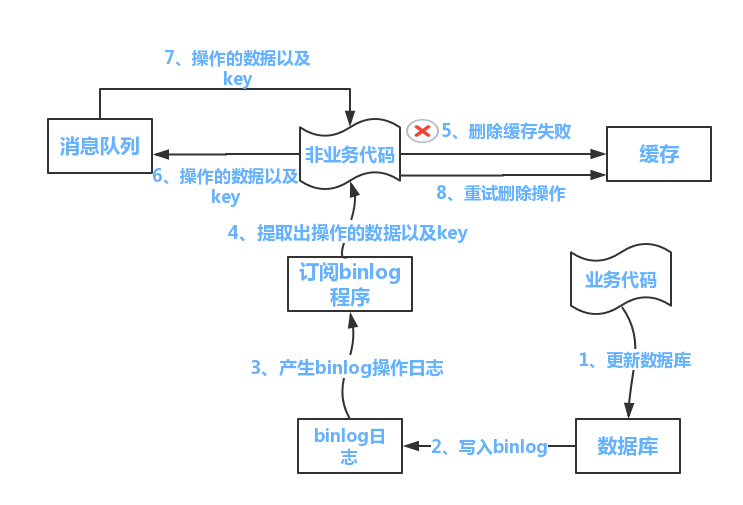
删缓存失败了，这会就出现不一致的情况了。

关于于删除失败导致的数据不一致的解决方案大概有以下几种：

1. 对删除缓存进行重试，数据的一致性要求越高，我越是重试得快。
2. 定期全量更新，简单地说，就是我定期把缓存全部清掉，然后再全量加载。
3. 给所有的缓存一个失效期，任何不一致，都可以靠失效期解决，失效期越短，数据一致性越高。但是失效期越短，查数据库就会越频繁。因此失效期应该根据业务来定。

### 4.5基于订阅binlog的同步机制实现重试

阿里巴巴的一款开源框架canal，提供了一种发布/ 订阅模式的同步机制，通过该框架我们可以对MySQL的binlog进行订阅，这样一旦MySQL中产生了新的写入、更新、删除等操作，就可以把binlog相关的消息推送至Redis，Redis再根据binlog中的记录，对Redis进行更新。



**流程如下图所示：**

（1）更新数据库数据

（2）数据库会将操作信息写入binlog日志当中

（3）订阅程序提取出所需要的数据以及key

（4）另起一段非业务代码，获得该信息

（5）尝试删除缓存操作，发现删除失败

（6）将这些信息发送至消息队列

（7）重新从消息队列中获得该数据，重试操作。