为什么redis需要把所有数据放到内存中？（2017-11-25-wzz）

### Redis为了达到最快的读写速度将数据都读到内存中，并通过异步的方式将数据写入磁盘。所以redis具有快速和数据持久化的特征。如果不将数据放在内存中，磁盘I/O速度为严重影响redis的性能。在内存越来越便宜的今天，

### ****redis 主从复制、哨兵和集群这三个有什么区别****

**1.主从模式：**读写分离，备份，一个master可以有多个slaves。

**2.哨兵sentinel：**监控，自动转移，哨兵发现主服务器挂了后，就会从slave中重新选举一个主服务器。

**3.集群：**为了解决单机redis容量有限的问题，将数据按一定的规则分配到多台机器，内存/qps不受限于单机，可受益于分布式集群高扩展性。

**Redis是单线程**

redis利用队列技术将并发访问变为串行访问，消除了传统数据库串行控制的开销

### 缓存穿透

描述：

缓存穿透是指缓存和数据库中都没有的数据，而用户不断发起请求，如发起为id为“-1”的数据或id为特别大不存在的数据。这时的用户很可能是攻击者，攻击会导致数据库压力过大。 解决方案：

接口层增加校验，如用户鉴权校验，id做基础校验，id<=0的直接拦截；

从缓存取不到的数据，在数据库中也没有取到，这时也可以将key-value对写为key-null，缓存有效时间可以设置短点，如30秒（设置太长会导致正常情况也没法使用）。这样可以防止攻击用户反复用同一个id暴力攻击

### Redis单线程

避免多线程中上下文的切换,减少了cup的损耗

避免了并发情况下加锁解锁的操作

虽然cpu多核情况下效率高,我们也可以使用单cpu情况下开启多个redis的实例

### Redis是单线程

redis利用队列技术将并发访问变为串行访问，消除了传统数据库串行控制的开销

#### 6、Redis有哪几种数据淘汰策略？

noeviction:返回错误当内存限制达到并且客户端尝试执行会让更多内存被使用的命令（大部分的写入指令，但DEL和几个例外）

allkeys-lru: 尝试回收最少使用的键（LRU），使得新添加的数据有空间存放。

volatile-lru: 尝试回收最少使用的键（LRU），但仅限于在过期集合的键,使得新添加的数据有空间存放。

allkeys-random: 回收随机的键使得新添加的数据有空间存放。

volatile-random: 回收随机的键使得新添加的数据有空间存放，但仅限于在过期集合的键。将要过期的键,使得新添加的数据有空间存放。

#### Redis集群方案应该怎么做？都有哪些方案？

1.codis。

目前用的最多的集群方案，基本和twemproxy一致的效果，但它支持在 节点数量改变情况下，旧节点数据可恢复到新hash节点。

2.redis cluster3.0自带的集群，特点在于他的分布式算法不是一致性hash，而是hash槽的概念，以及自身支持节点设置从节点。具体看官方文档介绍。

4.在业务代码层实现，起几个毫无关联的redis实例，在代码层，对key 进行hash计算，然后去对应的redis实例操作数据。 这种方式对hash层代码要求比较高，考虑部分包括，节点失效后的替代算法方案，数据震荡后的自动脚本恢复，实例的监控，等等。

#### 11、Redis集群方案什么情况下会导致整个集群不可用？

有A，B，C三个节点的集群,在没有复制模型的情况下,如果节点B失败了，那么整个集群就会以为缺少5501-11000这个范围的槽而不可用。

* 集群主库半数宕机(根据 failover 原理，fail 掉一个主需要一半以上主都投票通过才可以)
* 集群某一节点的主从全数宕机

12.谈一谈对redis集群的了解

1. Redis集群使用CRC16对key进行hash，集群固定使用16384对hash出来的值取模。因为取模结果一定在16384之内，所以集群中的sharding（分片）实际就是如何将16384个值在n个主节点间分配（从节点是主节点的近似副本，原因见3），如何分配取决于你的配置。
2. Redis生产级集群需要容灾，为此，一般部署为n个主+n\*m个从。n大小主要取决于单机性能，m大小主要取决于机器稳定性。
3. Redis集群是弱一致性的，此处的一致，主要指主从之间的数据一致性。主要是因为redis在做数据更新时，不要求主从数据同步复制一定要成功。
4. 集群最小的主数量为3，主数量应为奇数，以便做选举判决

#### 12、MySQL里有2000w数据，redis中只存20w的数据，如何保证redis中的数据都是热点数据？

redis内存数据集大小上升到一定大小的时候，就会施行数据淘汰策略。

#### 13、Redis有哪些适合的场景？

（1）会话缓存（Session Cache）

最常用的一种使用Redis的情景是会话缓存（session cache）。用Redis缓存会话比其他存储（如Memcached）的优势在于：Redis提供持久化。当维护一个不是严格要求一致性的缓存时，如果用户的购物车信息全部丢失，大部分人都会不高兴的，现在，他们还会这样吗？

幸运的是，随着 Redis 这些年的改进，很容易找到怎么恰当的使用Redis来缓存会话的文档。甚至广为人知的商业平台Magento也提供Redis的插件。

（2）全页缓存（FPC）

除基本的会话token之外，Redis还提供很简便的FPC平台。回到一致性问题，即使重启了Redis实例，因为有磁盘的持久化，用户也不会看到页面加载速度的下降，这是一个极大改进，类似PHP本地FPC。

再次以Magento为例，Magento提供一个插件来使用Redis作为全页缓存后端。1

此外，对WordPress的用户来说，Pantheon有一个非常好的插件 wp-redis，这个插件能帮助你以最快速度加载你曾浏览过的页面。

（3）队列

Reids在内存存储引擎领域的一大优点是提供 list 和 set 操作，这使得Redis能作为一个很好的消息队列平台来使用。Redis作为队列使用的操作，就类似于本地程序语言（如Python）对 list 的 push/pop 操作。

如果你快速的在Google中搜索“Redis queues”，你马上就能找到大量的开源项目，这些项目的目的就是利用Redis创建非常好的后端工具，以满足各种队列需求。例如，Celery有一个后台就是使用Redis作为broker，你可以从这里去查看。

（4）排行榜/计数器

Redis在内存中对数字进行递增或递减的操作实现的非常好。集合（Set）和有序集合（Sorted Set）也使得我们在执行这些操作的时候变的非常简单，Redis只是正好提供了这两种数据结构。

所以，我们要从排序集合中获取到排名最靠前的10个用户–我们称之为“user\_scores”，我们只需要像下面一样执行即可：

当然，这是假定你是根据你用户的分数做递增的排序。如果你想返回用户及用户的分数，你需要这样执行：

ZRANGE user\_scores 0 10 WITHSCORES

Agora Games就是一个很好的例子，用Ruby实现的，它的排行榜就是使用Redis来存储数据的，你可以在这里看到。

（5）发布/订阅

最后（但肯定不是最不重要的）是Redis的发布/订阅功能。发布/订阅的使用场景确实非常多。我已看见人们在社交网络连接中使用，还可作为基于发布/订阅的脚本触发器，甚至用Redis的发布/订阅功能来建立聊天系统！

#### 14、Redis支持的Java客户端都有哪些？官方推荐用哪个？

Redisson、Jedis、lettuce等等，官方推荐使用Redisson。

读法: Zhaideisi

#### 18、说说Redis哈希槽的概念？

Redis集群没有使用一致性hash,而是引入了哈希槽的概念，Redis集群有16384个哈希槽，每个key通过CRC16校验后对16384取模来决定放置哪个槽，集群的每个节点负责一部分hash槽。使用哈希槽的好处就在于可以方便的添加或移除节点。当需要增加节点时，只需要把其他节点的某些哈希槽挪到新节点就可以了；当需要移除节点时，只需要把移除节点上的哈希槽挪到其他节点就行了；在这一点上，我们以后新增或移除节点的时候不用先停掉所有的 redis 服务。

#### 19、Redis集群的主从复制模型是怎样的？

为了使在部分节点失败或者大部分节点无法通信的情况下集群仍然可用，所以集群使用了主从复制模型,每个节点都会有N-1个复制品.

#### 20、Redis集群会有写操作丢失吗？为什么？

Redis并不能保证数据的强一致性，这意味这在实际中集群在特定的条件下可能会丢失写操作。

#### 21、Redis集群之间是如何复制的？

异步复制

#### 25、Redis中的管道有什么用？

一次请求/响应服务器能实现处理新的请求即使旧的请求还未被响应。这样就可以将多个命令发送到服务器，而不用等待回复，最后在一个步骤中读取该答复。

这就是管道（pipelining），是一种几十年来广泛使用的技术。例如许多POP3协议已经实现支持这个功能，大大加快了从服务器下载新邮件的过程。

#### 26、怎么理解Redis事务？

Redis事务是命令的集合,可以对命令集合做语法检查,没有事务回滚,没有原子性和隔离性

#### 27、Redis事务相关的命令有哪几个？

* multi，标记一个事务块的开始，返回 ok
* exec，执行所有事务块内，事务块内所有命令执行的先后顺序的返回值，操作被，返回空值 nil
* discard，取消事务，放弃执行事务块内的所有命令，返回 ok
* watch，监视 key 在事务执行之前是否被其他指令改动，若已修改则事务内的指令取消执行，返回 ok
* unwatch，取消 watch 命令对 key 的监视，返回 ok

#### 28、Redis key的过期时间和永久有效分别怎么设置？

EXPIRE和PERSIST命令。

#### 29、Redis如何做内存优化？

尽可能使用散列表（hashes），散列表（是说散列表里面存储的数少）使用的内存非常小，所以你应该尽可能的将你的数据模型抽象到一个散列表里面。

比如你的web系统中有一个用户对象，不要为这个用户的名称，姓氏，邮箱，密码设置单独的key,而是应该把这个用户的所有信息存储到一张散列表里面。

#### 30、Redis回收进程如何工作的？

一个客户端运行了新的命令，添加了新的数据。

Redi检查内存使用情况，如果大于maxmemory的限制, 则根据设定好的策略进行回收。

一个新的命令被执行，等等。

所以我们不断地穿越内存限制的边界，通过不断达到边界然后不断地回收回到边界以下。

如果一个命令的结果导致大量内存被使用（例如很大的集合的交集保存到一个新的键），不用多久内存限制就会被这个内存使用量超越。

aof重写

热点数据需要手动去写

Mybatis聚合函数

Spring父容器子容器

除了哪五种,其他Redis数据类型

位图bitmap

统计HyperLogLog

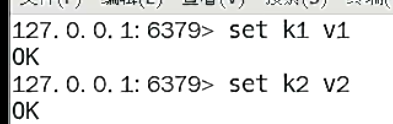
地理GEO

Steams

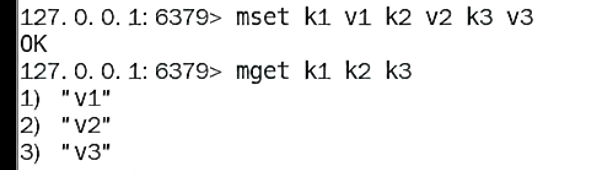
命令不区分大小写,但是key区分大小写

### String类型

存储 获取 Set k1 v1 get k1

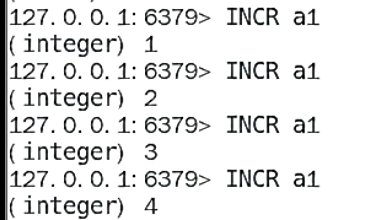


一次存储获取多个数据 mset k1 v1 k2 v2 k3 v3 mget k1 k2 k3



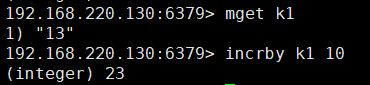
自增1

Incr a1



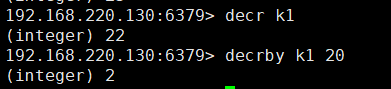
自增n ,n为随意正整数

Incrby k1 n

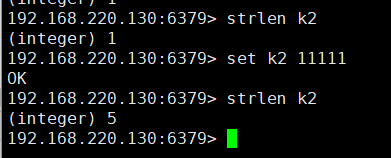


自减 decr k1

自减n decrby k1 5



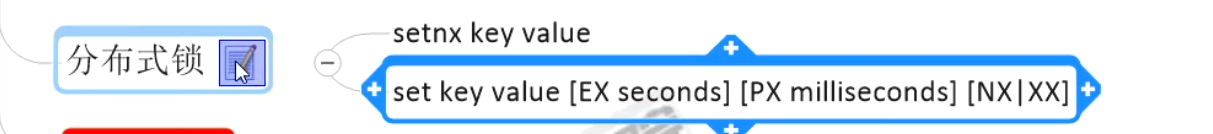
strlen 获取字符串长度

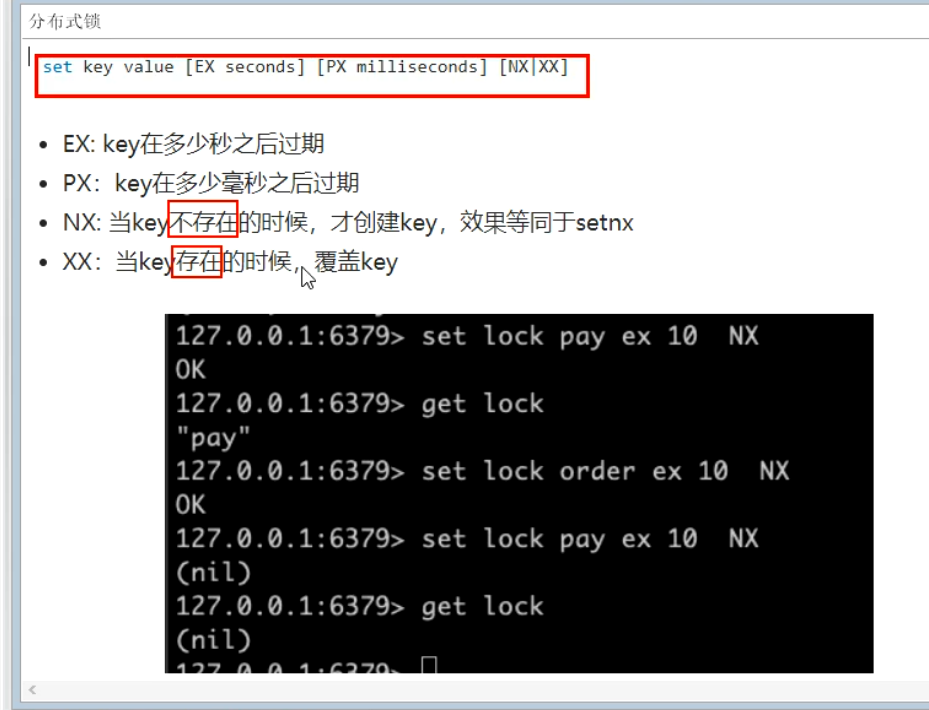


ttl k1 还有多久过期



分布式锁





string应用场景

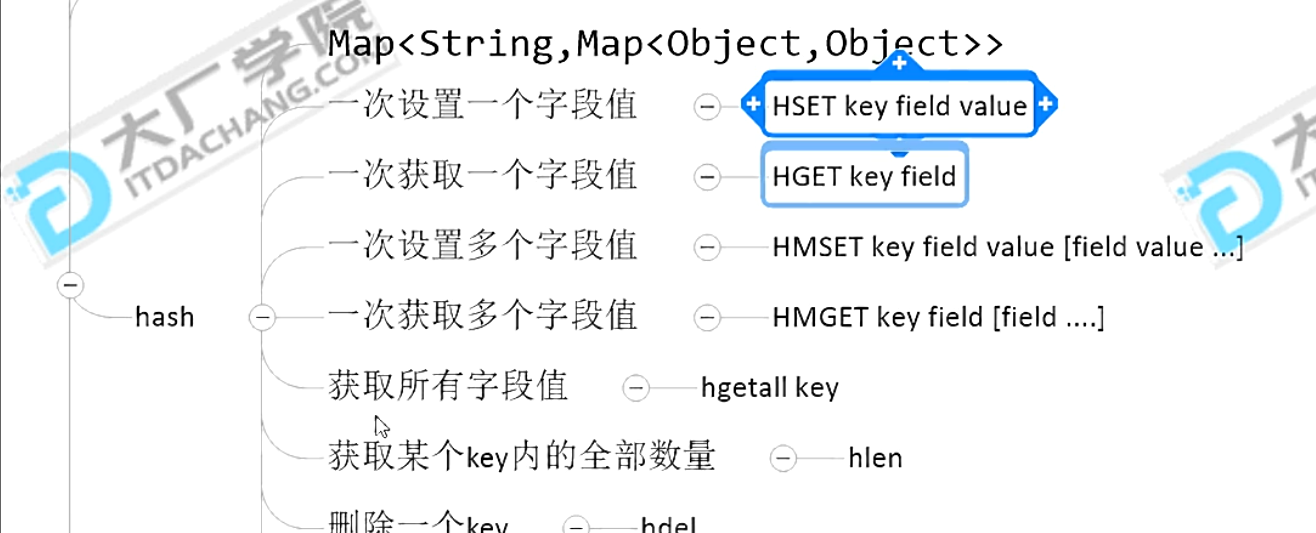
1商品编号,订单号采用incr命令生成

2是否喜欢的文章(就是喜欢和不喜欢的统计)

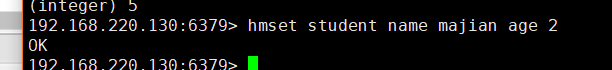


### Hash

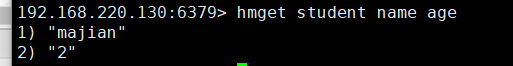
Redis中的hash对应Java中map<String , map<k , y>>



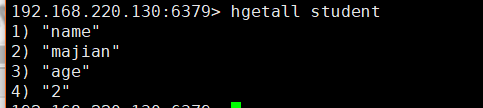
Hmset 给一个键值设置多个字段值



获取一个key的多个字段值 hmget key filed1 field2



获取一个key 的 所有字段值 hgetall key



获取key的字段数量



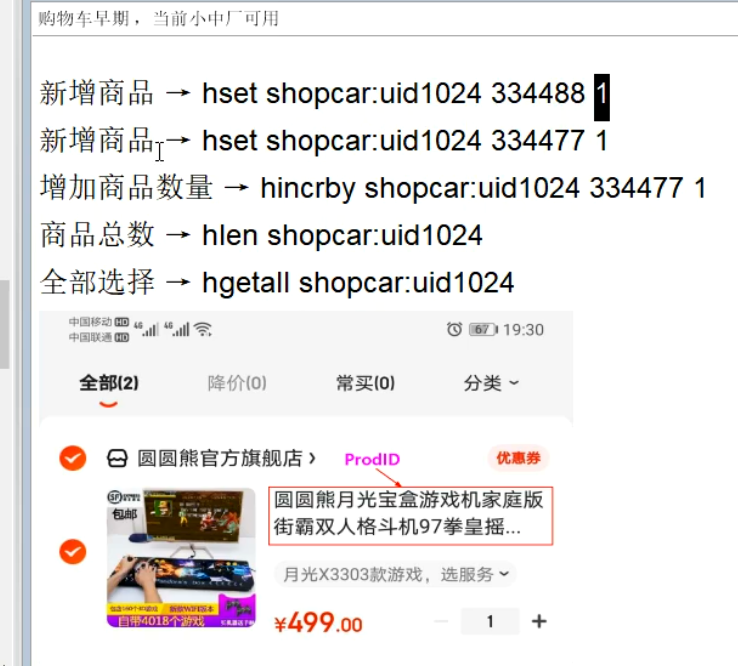
Hdel 删除一个key的一个字段



使一个key的一个字段加n

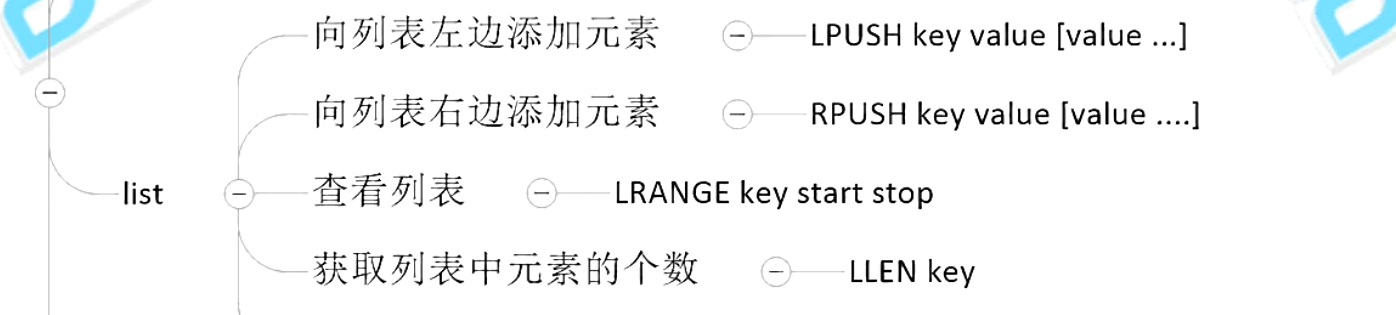


Hash使用场景 -----购物车

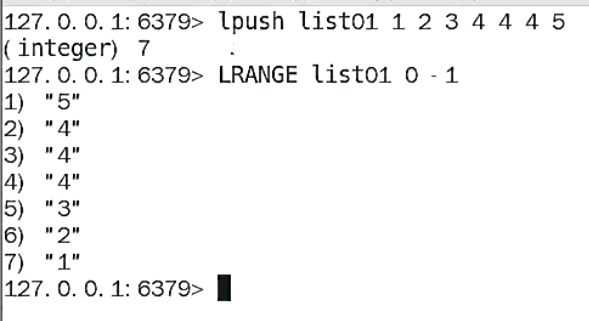


### List

List是有序可重复的双向链表



全部打印 值设为 0 到 -1



应用场景

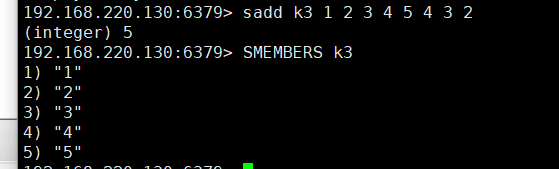
微信文章订阅公众号推送信息



Set 无序不重复



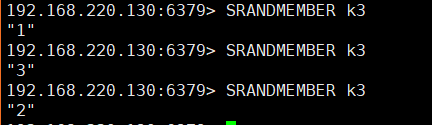
添加数据,重复的会覆盖



获取集合有多少数量 scard key



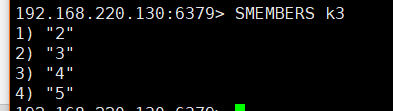
随机弹出一个,不删除



随机弹出一个然后删除



获取集合所有元素



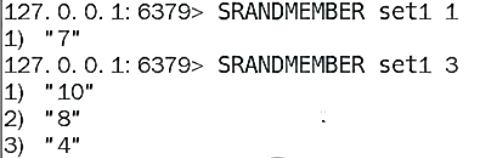
Set集合运算



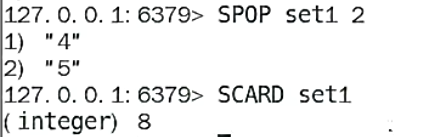
应用场景

1微信抽奖小程序’------------抽奖 随机弹出元素命令

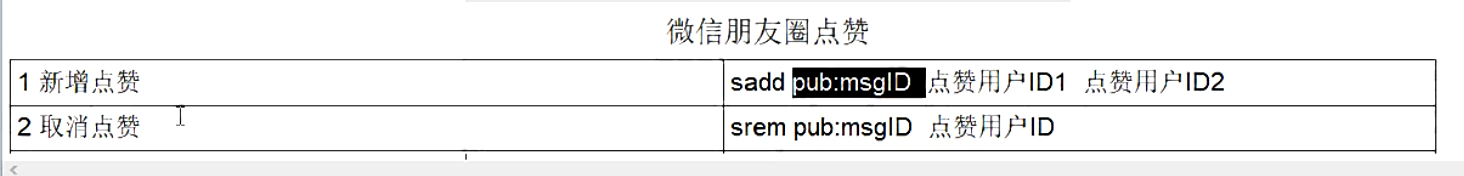
------使用srandmember key n 随机弹出n个不删除



------使用spop key n 随机弹出n个然后删除

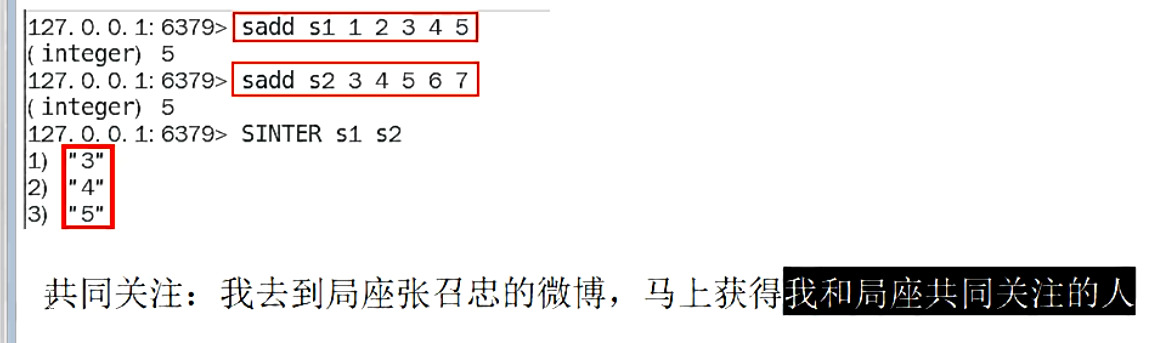


2微信朋友圈点赞---添加和删除命令

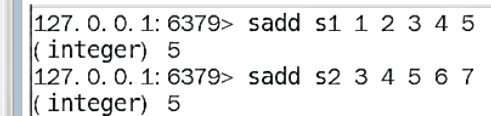


3微信好友关注社交关系--------取交集





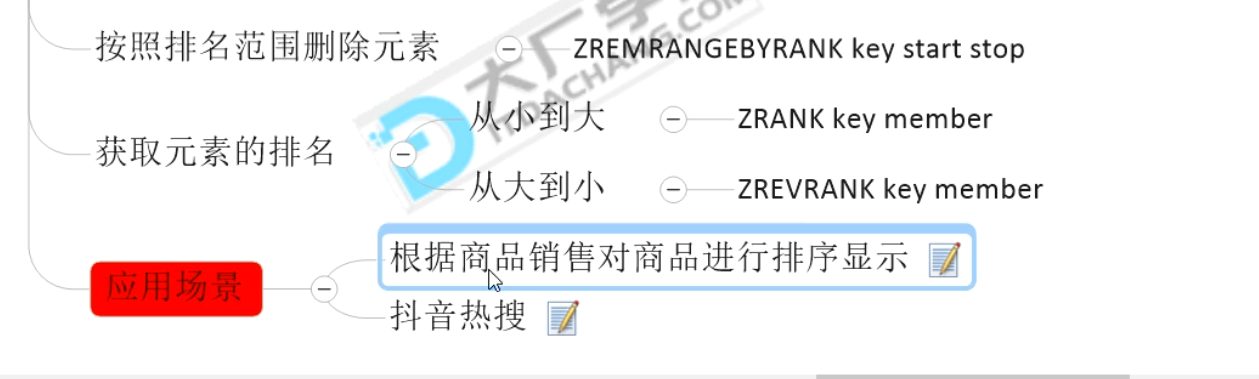
4 qq内推可能认识的人-----取一个差集 (我没有但是你有的好友)





### zset-------向一个有序的集合加入元素和该元素的分数





销售的商品进行排序显示



抖音热搜



**布隆过滤器（推荐）**

就是引入了k(k>1)k(k>1)个相互独立的哈希函数，保证在给定的空间、误判率下，完成元素判重的过程。

它的优点是空间效率和查询时间都远远超过一般的算法，缺点是有一定的误识别率和删除困难。

Bloom-Filter算法的核心思想就是利用多个不同的Hash函数来解决“冲突”。

Hash存在一个冲突（碰撞）的问题，用同一个Hash得到的两个URL的值有可能相同。为了减少冲突，我们可以多引入几个Hash，如果通过

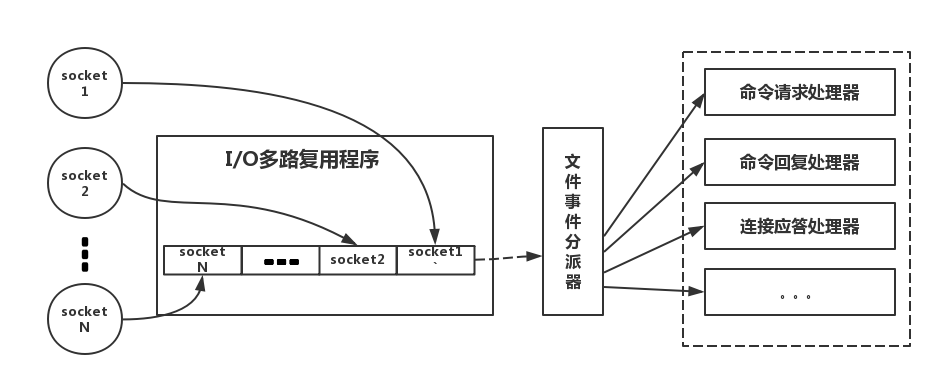
其中的一个Hash值我们得出某元素不在集合中，那么该元素肯定不在集合中。只有在所有的Hash函数告诉我们该元素在集合中时，才能确

定该元素存在于集合中。这便是Bloom-Filter的基本思想。

Bloom-Filter一般用于在大数据量的集合中判定某元素是否存在。

Redis多路复用

1、经营方式一就是传统的并发模型，每个I/O流都有一个新的线程管理。  
2、经营方式二就是I/O多路复用。只有单个线程，通过跟踪每个I/O流的状态，来管理多个I/O流



我们的redis-client在操作的时候，会产生具有不同事件类型的socket。在服务端，有一段I/0多路复用程序，将其置入队列之中。然后，文件事件分派器，依次去队列中取，转发到不同的事件处理器中

在请求比较多， 服务数量比较少的情况下， 使用多路复用可以 提高效率