**Redis**

# 缓存问题

## 缓存雪崩

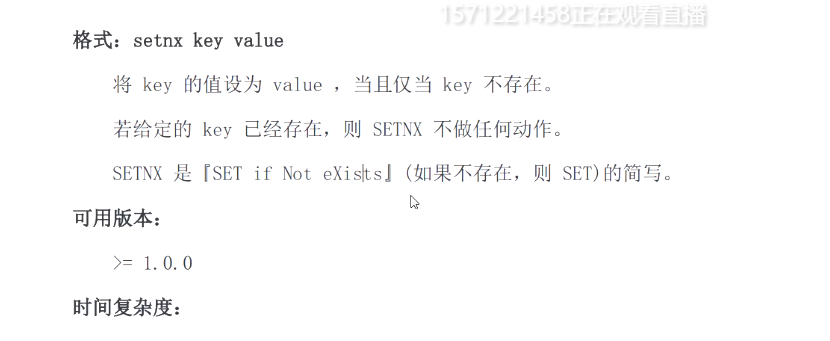
## 缓存击穿

## 缓存穿透

缓存穿透，是指查询一个数据库一定不存在的数据。正常的使用缓存流程大致是，数据查询先进行缓存查询，如果key不存在或者key已经过期，再对数据库进行查询，并把查询到的对象，放进缓存。如果数据库查询对象为空，则不放进缓存。

解決方法:布隆过滤器

# 分布式锁



* 放了防止redis宕机的时候，key一直存在，所以要设置超时时间。  
  设置了超时时间，如果过了超时时间，业务还没有执行完，那么key自动失效，就会有其他的线程来争夺锁，等到该线程执行完的时候，会释放锁，但是是释放的是其他线程加的锁，所以还是出现超卖的现象。
* 可以每隔一段时间(超时时间的三分之一，例如redission)检查锁是否失效，没有失效，就更新超时时间，为锁续命

## 分布式锁的问题

1. 由单机锁演进到分布式锁
2. 分布式锁可以由redis中setnx来实现
3. 使用这条命令，如果服务器宕机了，那么设置的值将一直存在

解决方法:可以设置超时时间

如果设置的超时时间过短，A线程的业务代码还没有执行完，锁就被redis删除了，此时B线程获取了锁，开始执行业务代码，刚好在这时B线程的锁就被A线程释放了，其他线程又可以获取锁继续执行，那么这还是会导致超卖问题

解决方法:可以写个定时任务，定时（最好是超时时间的三分之一）检查锁是否持有，如果持有，就延长锁的时间，可以使用redision

# 数据结构

Key设计的时候一般遵循：功能模块 表名 业务模块

## 存储上限

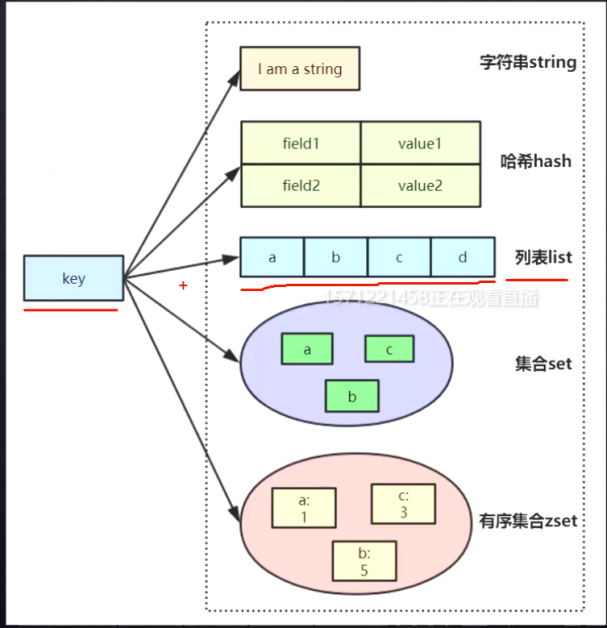
Strings类型：一个String类型的value最大可以存储512M

Lists类型：list的元素个数最多为2^32-1个，也就是4294967295个。

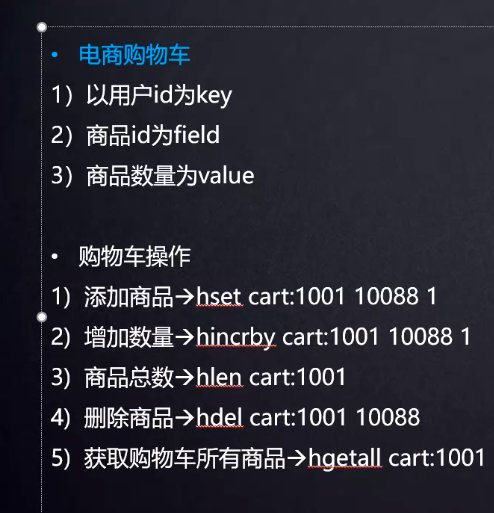
Sets类型：元素个数最多为2^32-1个，也就是4294967295个。

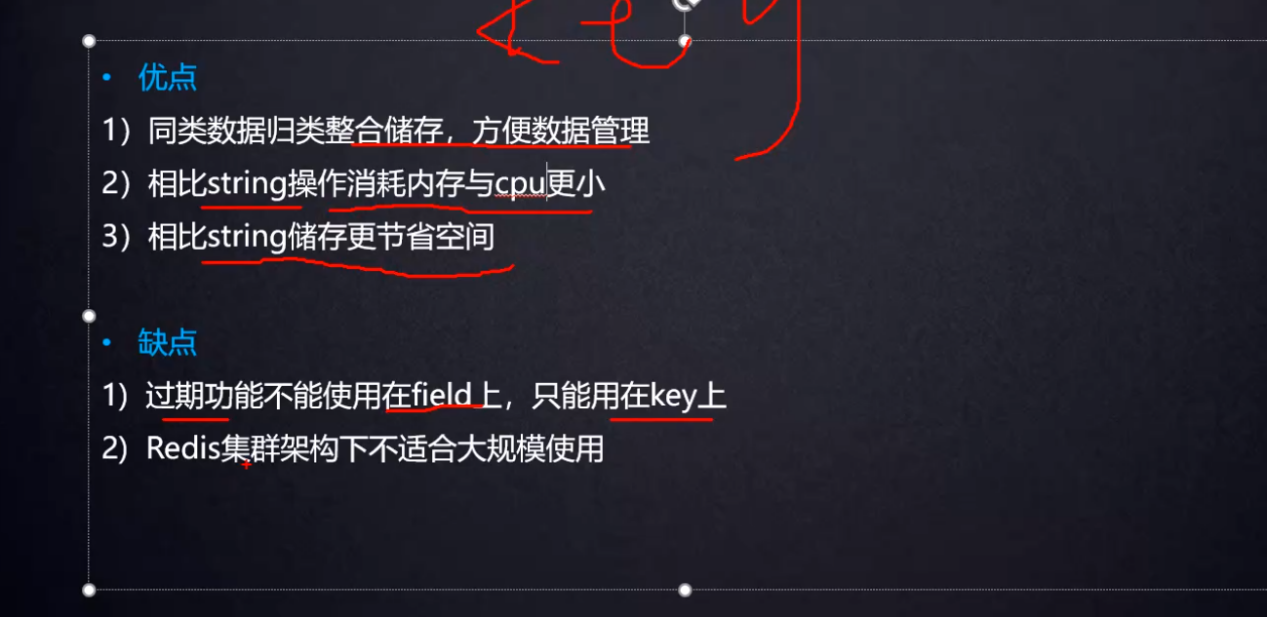
Hashes类型：键值对个数最多为2^32-1个，也就是4294967295个。

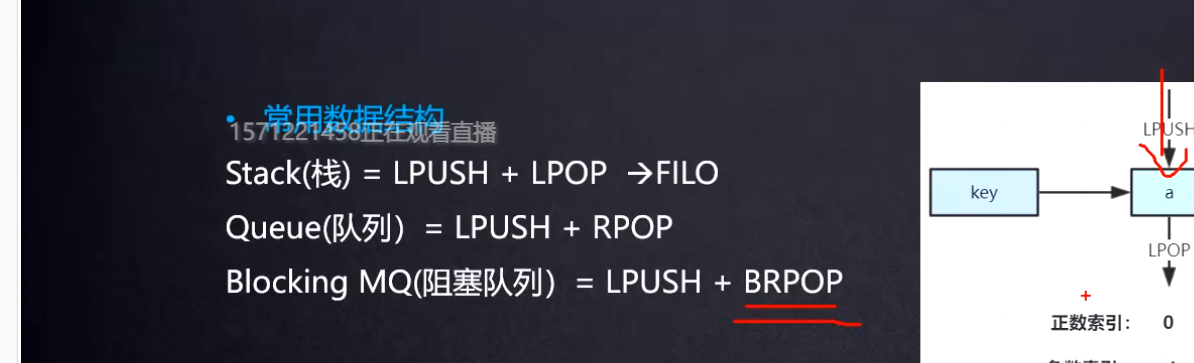
Sorted sets类型：跟Sets类型相似。

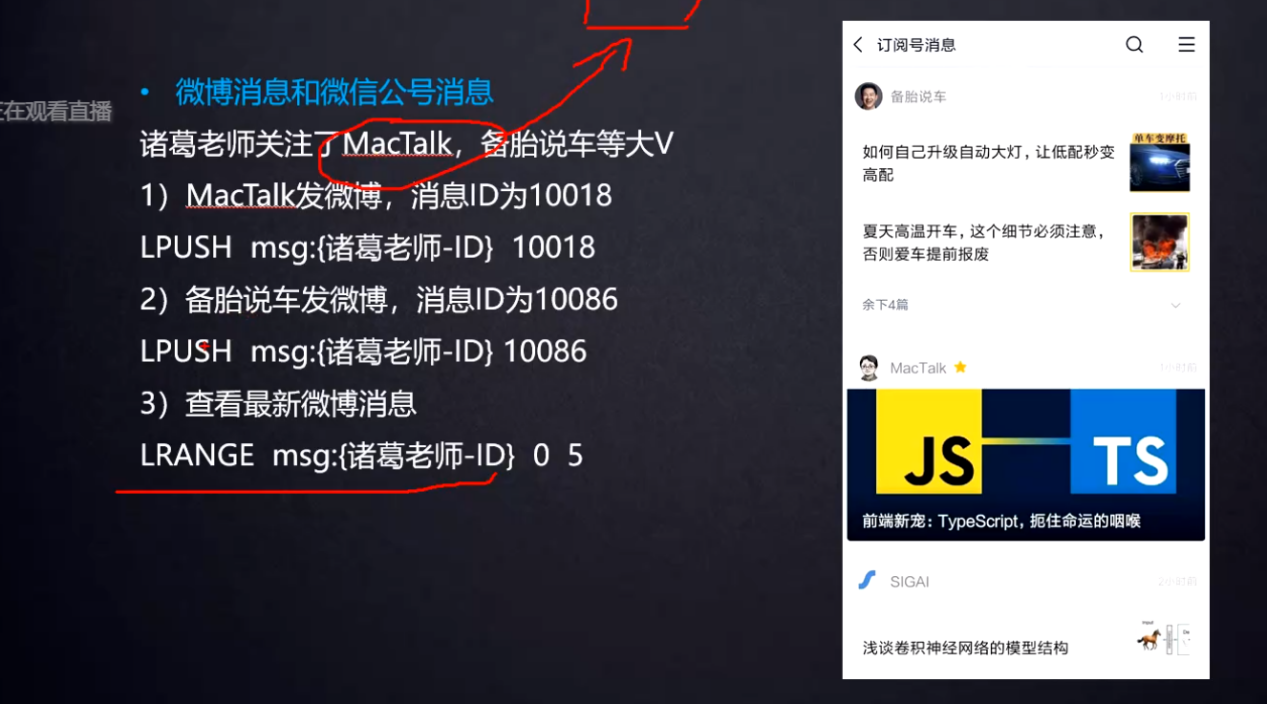


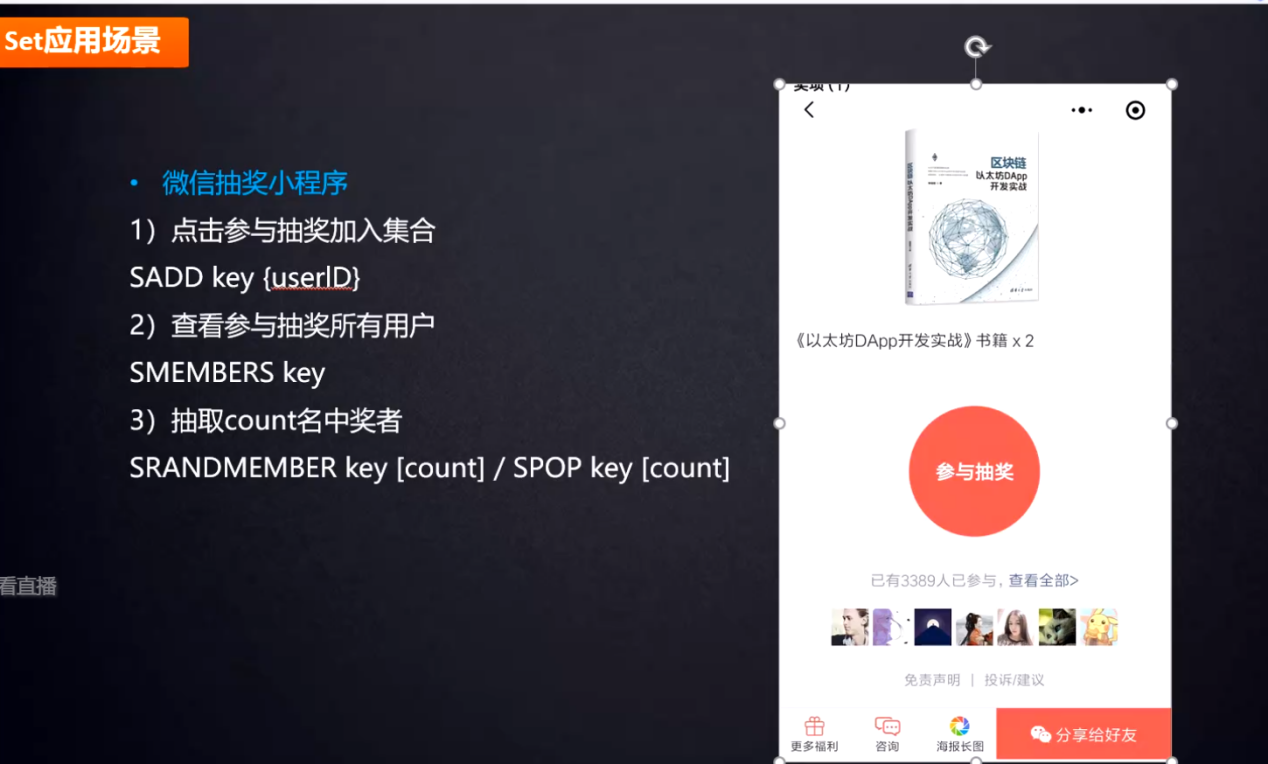
## Hash

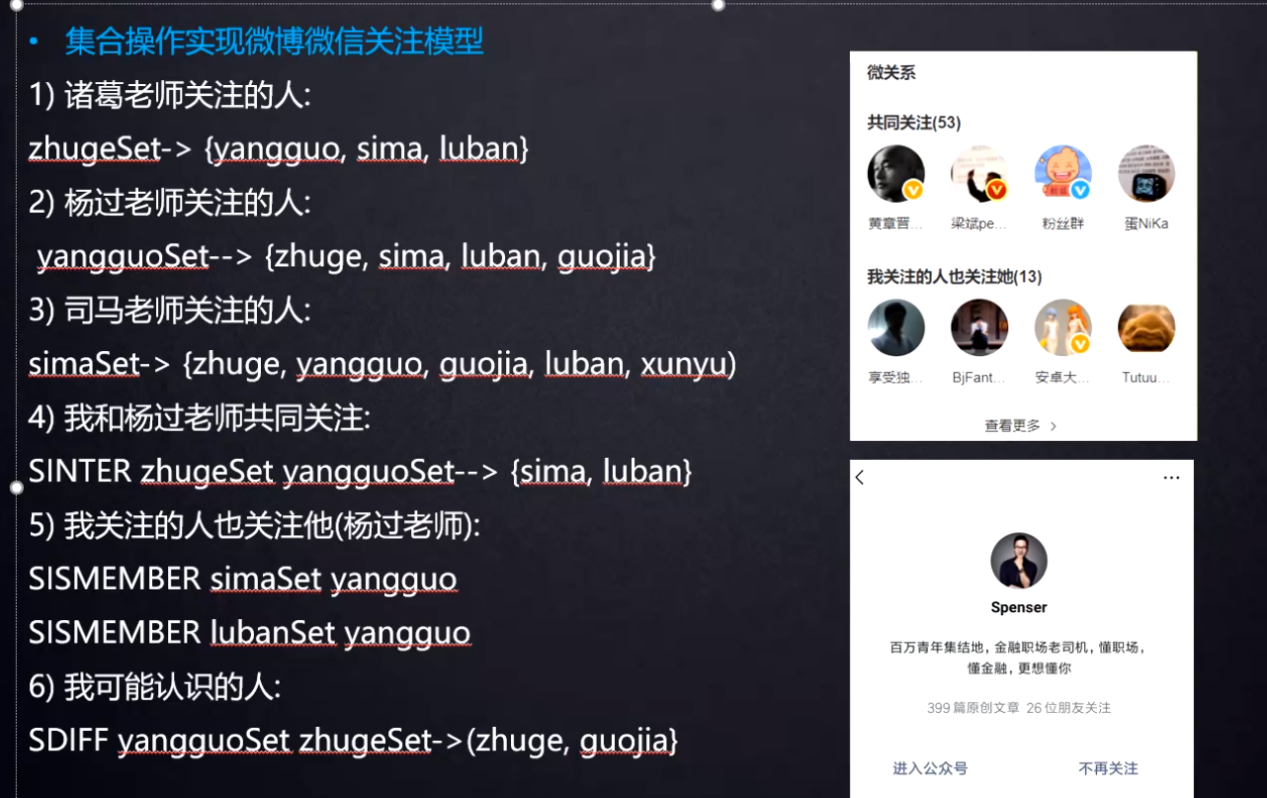


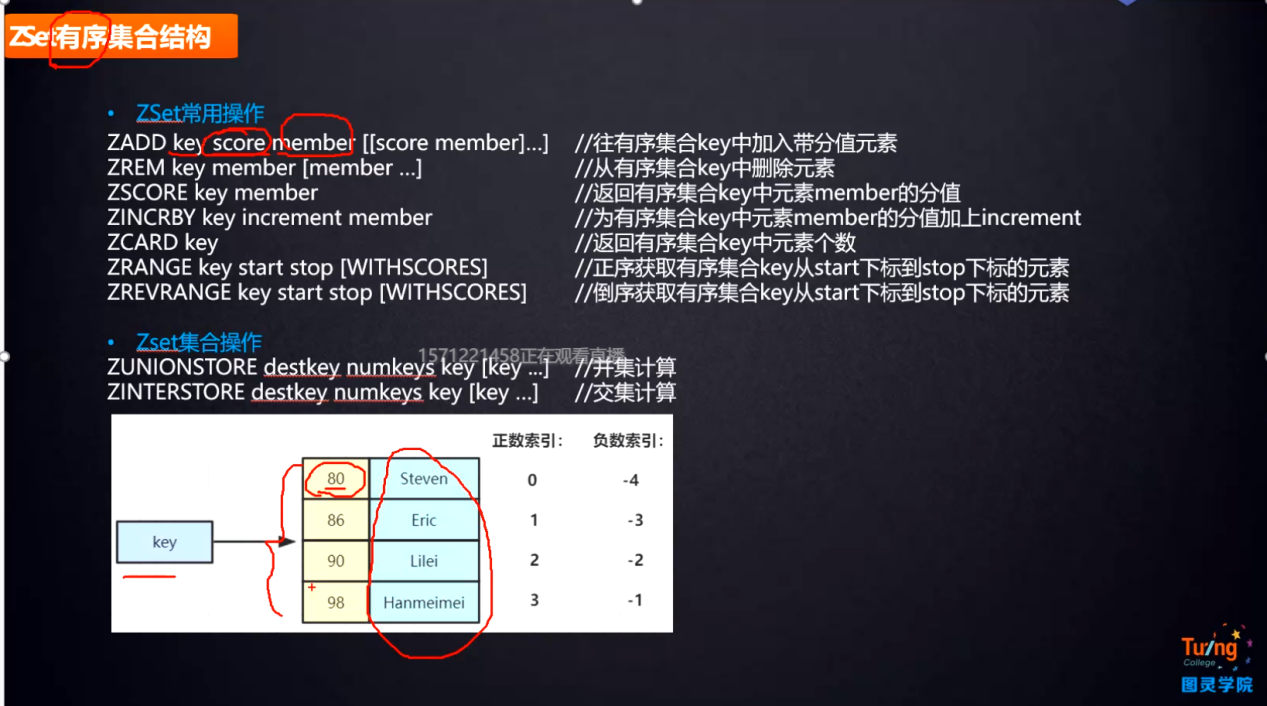






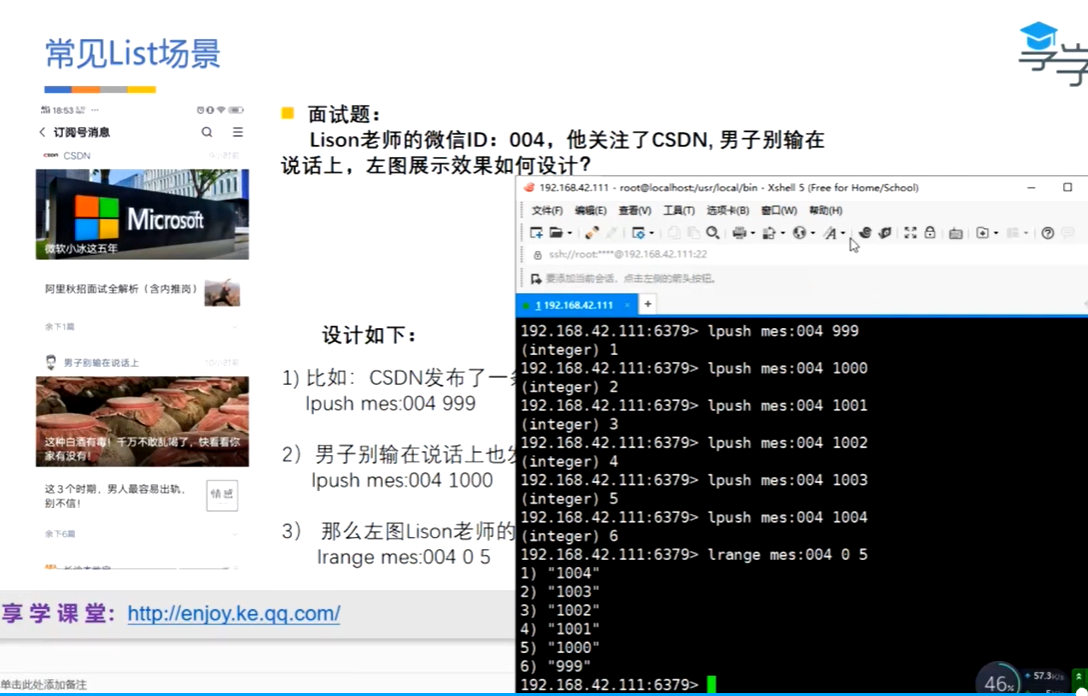






## LIst

使用场景：消息推送



## Set

抽奖:**srandmember**

## Zset

可以做热榜

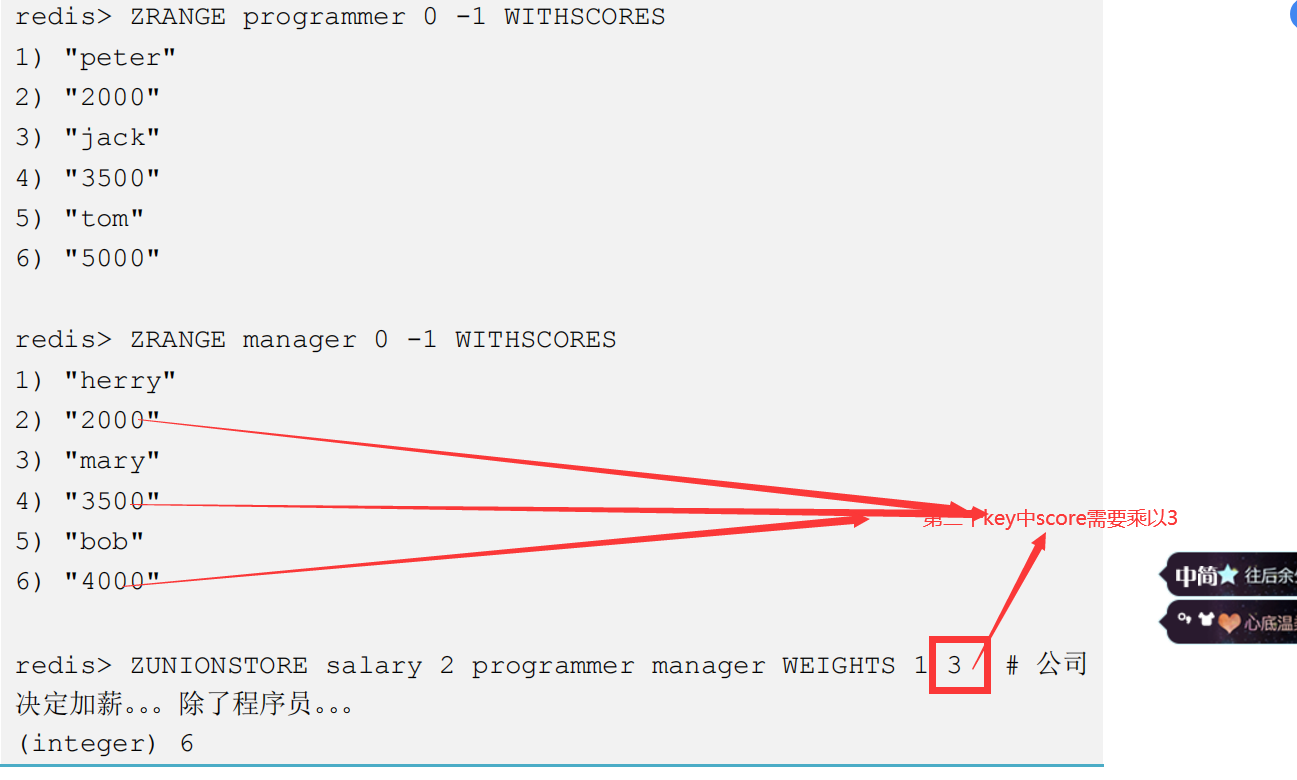
### ZUNIONSTORE

可以用来将两个key中相同的元素的score相加

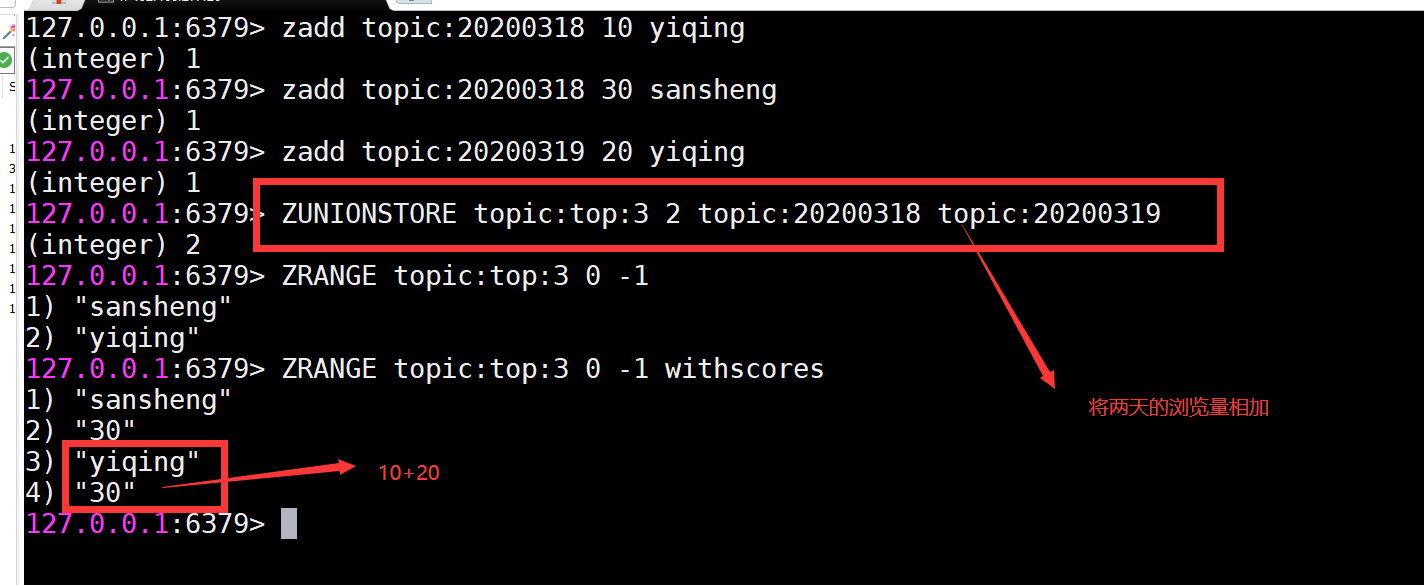
比如keya：1 one

Keyb: 2 one

用这个命令之后，就变为了3=1+2



可以用来将几天的浏览量相加

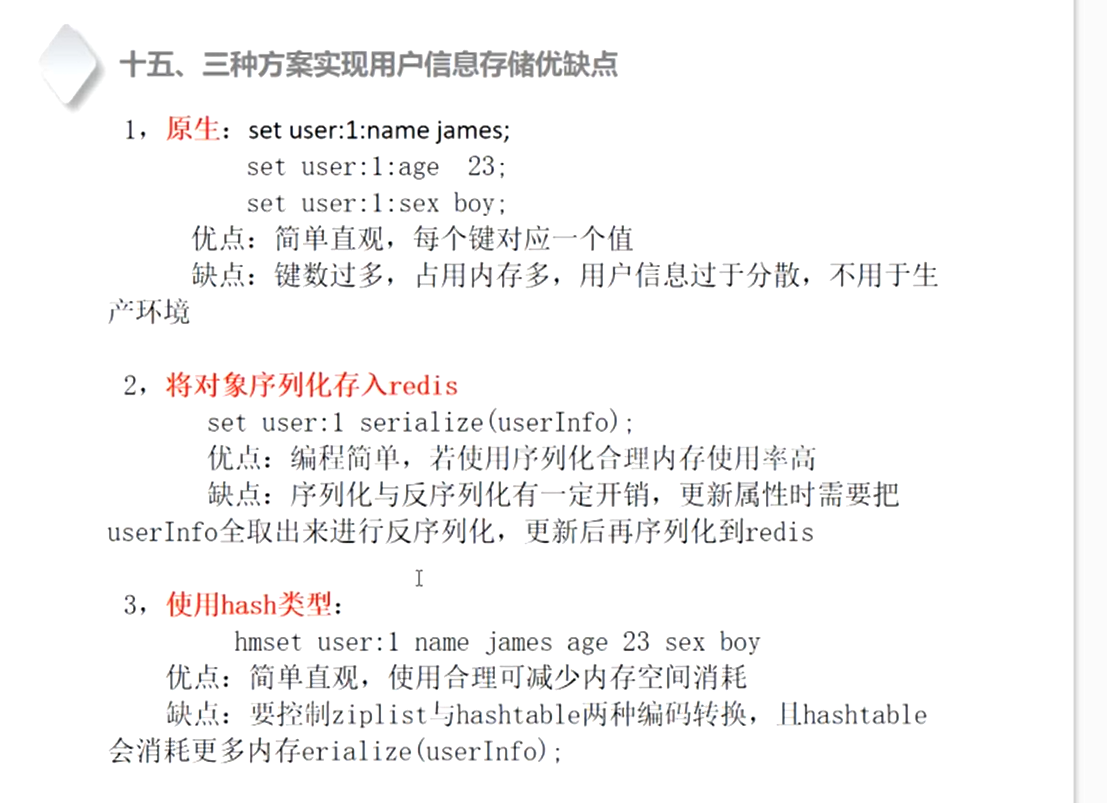


# 特性

速度快:数据存储到内存中，多路复用



# 数据存储的优缺点



# Redis持久化

## rdb

## AOF

# Lua

## 安装教程

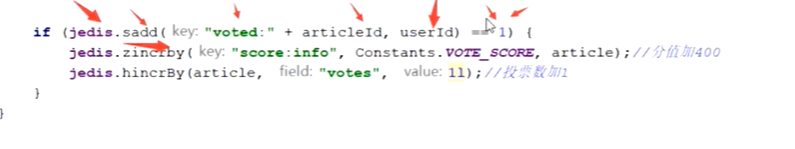
1. 到http://www.lua.org/ftp/下载包
2. 解压
3. yum install libtermcap-devel ncurses-devel libevent-devel readline-devel
4. Make linux test
5. Make install

# 应用

## 投票

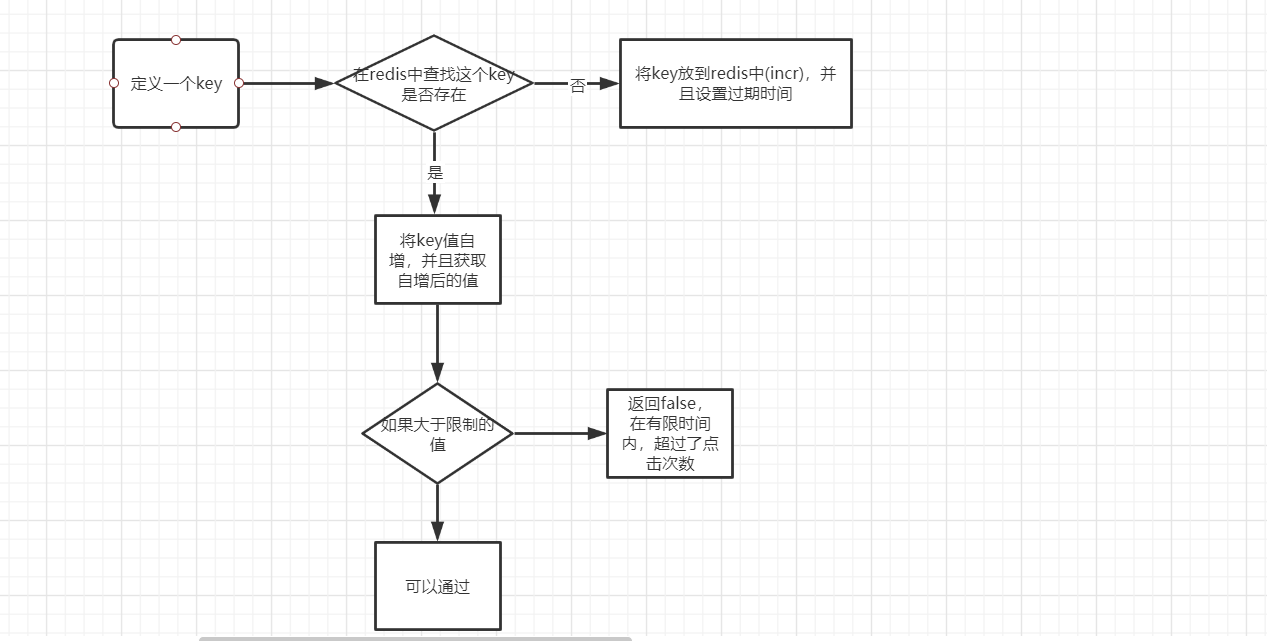
给文章投票

1. 创建文章，将文章保存到redis中
2. 给文章投票，防止一个人头多次，利用set类型来保存
3. 投一票要加分，分数需要排序，用到了zset
4. 还要记录文章的创建时间，按照时间排序，利用zset来保存



# 限流

## 流程图



## 限流脚本

|  |
| --- |
| local key = KEYS[1] local limit = tonumber(ARGV[1]) local expire\_time = ARGV[2]  local is\_exists = redis.call("EXISTS", key)  if is\_exists == 1 then   if redis.call("INCR", key) > limit then  return 0  else  return 1  end  else  redis.call("SET", key, 1);  redis.call("EXPIRE", key, expire\_time)  return 1 end |

执行:EVALSHA 744f229f0fc3c2765f4942cf2ef517a5f98f9d5a 1 192.168.42.111 5 2

