# 1.Redis介绍

## 1.1什么是Redis

Redis 是一个支持nosql，key-value数据类型、内存、缓存、分布式的、可持久化、非关系型数据库。

## 1.2 nosql,key-value,非关系型数据库

### 1.2.1 结构化数据和非结构化数据

将一批数据的共有特性经过提取后的有结构的数据就是结构化数据(例如一个国家的所有学生信息)

不能以同一个结构描述的一个数据集合,统称为非结构化数据(网页,日志是典型的非结构化数据)

### 1.2.2 key-value/nosql

存储结构化数据使用二维表;

典型的关系型数据库（MySQL、Oracle）,处理的是结构化数据，使用的操作语言sql(structured query language);

存储非结构化数据使用的常用结构就是key-value;

对于非结构化的海量数据处理(增删查改),使用nosql(not only structured query language);

典型的非关系型数据库redis,mongoDB

## 1.3内存

redis在运行期间,使用的是内存资源作为存储介质存储数据;具有内存的致命缺陷(容量上限有限,断电丢失数据);优点是内存存储快极了;

## 1.4可持久化

存储在redis内存中的数据,可以设置持久化策略,在断电内存数据丢失时,重启redis可以从磁盘文件重新读取数据(占据市场重要的技术点)

## 1.5支持分布式集群

3.0之后的cluster技术,1000个节点

可以使用redis搭建传统结构的分布式,或者利用redis-cluster搭建集群

## 1.6 Redis支持的各种功能

* 1. 缓存,提升系统访问数据的性能(减少了层级之间的调用)
  2. 低配版本的消息队列
  3. 实现各种逻辑锁
  4. session共享使用redis解决

# 2.Redis的安装和启动

## 2.1解压安装

在砺儒云下载安装包**redis-3.2.11.tar.gz**

也可以**在官方网站上下载**

官网下载地址：

<http://download.redis.io/releases/redis-3.2.11.tar.gz>

### 2.1.1 上传安装包

启动虚拟机，进入/home/software

手机屏幕截图

描述已自动生成

打开图片包含 徽标

描述已自动生成，通过sftp上传安装包**redis-3.2.11.tar.gz**到虚拟机的/home/software下

图形用户界面, 应用程序, 表格

描述已自动生成

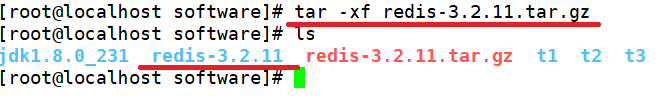
### 2.1.2 解压

[root@localhost software]# tar -xf redis-3.2.11.tar.gz

文本

描述已自动生成

解压后得到redis-3.2.11 文件夹



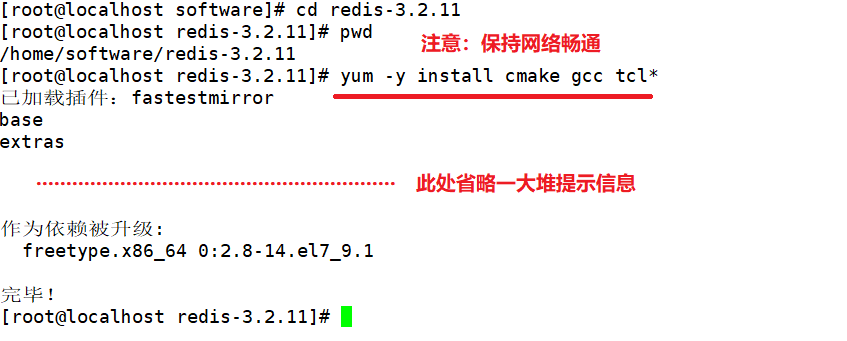
### 2.1.3 执行编译，编译安装

#### 1.进入redis根目录

[root@localhost software]# cd redis-3.2.11

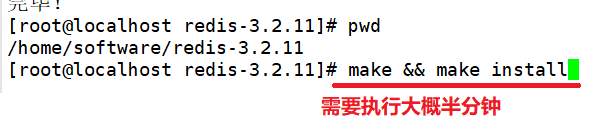
#### 2.在执行编译和编译安装之前，需要先安装插件

[root@localhost redis-3.2.11]# **yum -y install cmake gcc tcl\***



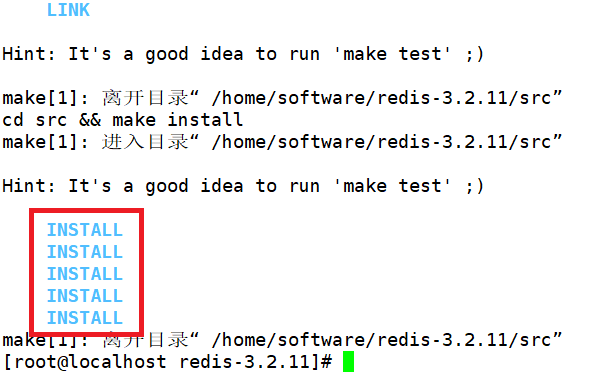
#### 3.执行 make && make install

[root@localhost redis-3.2.11]# **make && make install**



#### 4.安装成功

执行完毕，看到5个蓝色的install,表示安装成功



## 2.2 redis的默认启动和默认登录

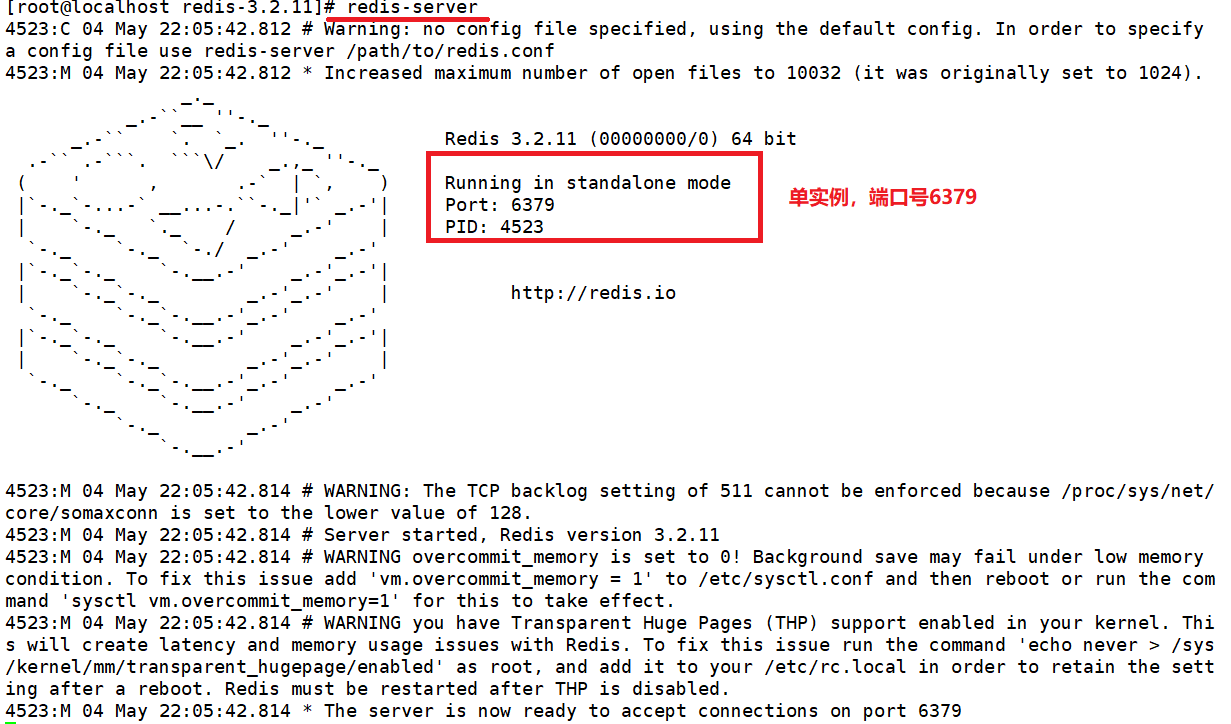
### 2.2.1 启动redis一个单节点服务

执行命令启动redis一个单节点服务

在当前虚拟机的任何位置，执行命令,启动redis一个单节点服务

语法：redis-sever 配置文件（配置文件默认为空）

[root@localhost redis-3.2.11]# redis-server



看到上图所述内容，表示Redis服务端启动完毕（单实例，单线程，表示只有单个服务端），客户端可以很多个。

### 2.2.2 启动一个客户端

单独打开一个xshell窗口，启动一个客户端

语法：redis-cli -p 连接的服务器端口(默认6379) -h连接的服务器ip地址（默认127.0.0.1）

[root@localhost ~]# redis-cli

图标

低可信度描述已自动生成

## 2.3 测试Redis命令-nosql的增删改查命令

### 2.3.1.添加

127.0.0.1:6379> set teachername xulaoshi

OK

### 2.3.2.查询

127.0.0.1:6379> get teachername

"xulaoshi"

### 2.3.3.修改

127.0.0.1:6379> set teachername wulaoshi

OK

### 2.3.4.删除

127.0.0.1:6379> del teachername

(integer) 1

127.0.0.1:6379> get teachername

(nil)

## 2.4 Redis的命令参数

文本

描述已自动生成

### 2.4.1 EX seconds

设置过期时间，以秒为单位

文本

描述已自动生成

### 2.4.2 PX milliseconds

设置过期时间，以毫秒为单位

文本, 信件

描述已自动生成

### 2.4.3 NX

仅当key不存在时，可以设置改key值

文本

描述已自动生成

### 2.4.4 XX

仅当key存在时，可以设置该key值（修改）

文本, 信件

描述已自动生成

# 3.Redis五种数据类型及基本命令

Redis具备五种数据类型，每种数据类型都是针对key-value结构中的value而言。

String 普通数据的缓存

Hash 存放对象，应用场景：存放游客的购物车

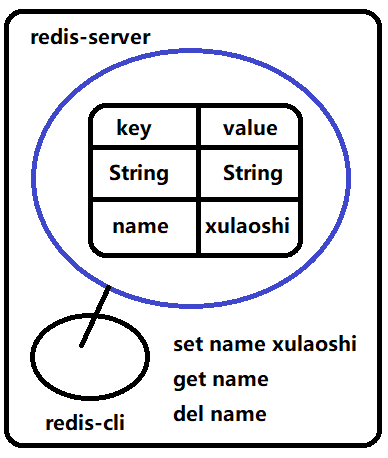
List 秒杀

Set 兴趣爱好；关注，粉丝，共同好友

Zset 排行榜

## 3.1 String类型

### 3.1.1 String类型数据结构

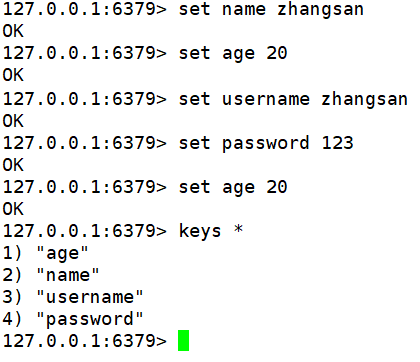
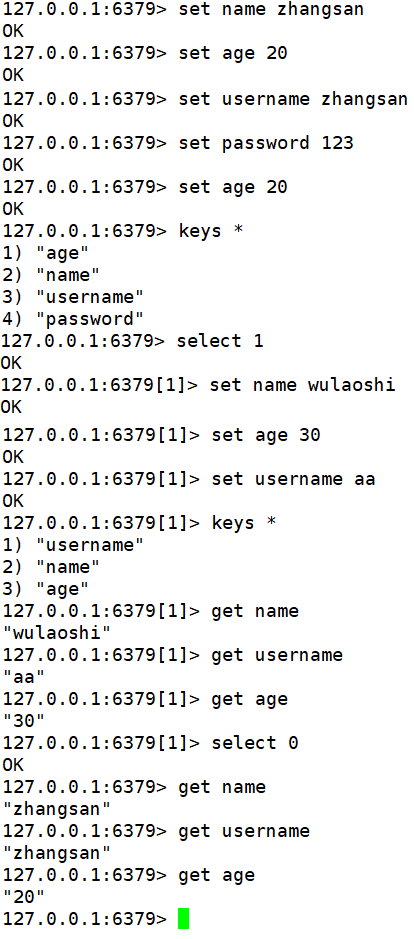


### 3.1.2命令

此处会带有一些跟数据类型无关的命令

#### keys \*

查看当前redis节点中所有已经存在的key值。\*是一个pattern,可以 name\*，ag\* 等等。



#### set key value

存储一个key-value结构的数据（String）,覆盖同名key的value值.

#### get key

获取当前key对应的value数据

#### select 整数

**redis中默认存在16个数据分库（database），index号0-15，库与库之间是不互通的。**

**redis默认登录的是0号分库，可以选择1号库等.**

#### exists key

判断当前节点是否包含key的数据。

get也可以完成这个任务。redis在key-value结构的value存储容量是1G。使用get判断存在会先读再判断数据是否为空，浪费时间

文本

描述已自动生成

#### del key

删除key值对应的key-value数据

#### type key

查看当前key的类型

文本, 信件

描述已自动生成

#### 文本 描述已自动生成flushall

清除所有分库中的数据。包括清空dump.rdb文件

#### save

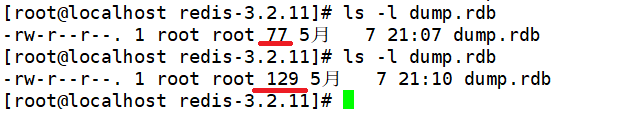
将当前的内存数据保存到磁盘文件（根目录dump.rdb）

关闭服务器端，再次打开，看看数据是否还在。

关闭redis服务器端 Ctrl + C 或用 quit命令

save前和save后观察redis根目录的dump.rdb大小

save后，可以发现已经存储到持久化文件



#### flushdb

清空一个分库的数据，不删除持久化文件内容（进行测试使用）

#### 文本 描述已自动生成incr decr key

自增1，自减1

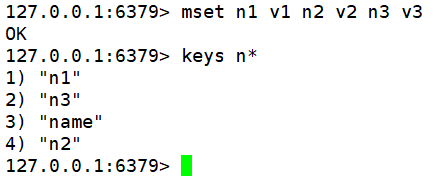
redis中string字符串如果是纯数字，也有一些简单的数字操作命令

#### incrby decrby key

多步计步器：incrby decrby key 整数：对key的value数字自增自减去整数步数

#### mset 和 mget 批量操作

单节点批量操作（不支持多节点分布式，只能对本节点数据进行处理）(测试时确定当前节点拥有的数据)

 图表, 瀑布图

描述已自动生成

#### expire key:

对key添加超时过期的设定。

redis作为内存缓存数据库虽然速度快,还是有缺点的,内存容量,不可能比硬盘容量大的，可以通过expired key 设置数据的过期时间

文本

描述已自动生成

redis最常见的保护内存的机制(热点数据保存,冷点数据超时清除)。**访问次数多的数据，超时时间越长，访问次数少的数据，超时时间短。**

#### ttl key

配合ttl key可以查看 key值剩余时间,已经超时的内容删除，ttl剩余时间是-2 ；-1表示永久数据。

文本

描述已自动生成 文本

低可信度描述已自动生成

#### pttl key

pttl key也可以查看 key值剩余时间，返回是毫秒。

文本

描述已自动生成

#### pexpire

设置超时，以毫秒为单位

文本

描述已自动生成

#### append key value

对key对应的value进行追加数据操作;

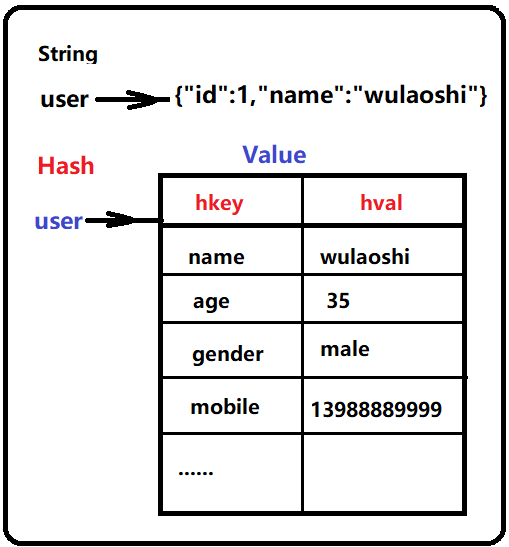
文本

描述已自动生成

## 3.2 Hash数据类型

### 3.2.1 hash类型数据结构

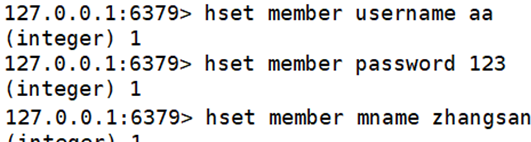
对应value为hash结构的数据都是面向对象的数据结构



### 3.2.2 命令

#### hset key field value

key 是 相当于变量名，field是value中的key-value对中的key，面向对象时表示一个对象的属性名称， value表示属性的值



#### hget key field

文本

中度可信度描述已自动生成

#### hexists key field

判断hash类型中的属性值是否存在

图片包含 文本

描述已自动生成

#### hmset和hmget

批量设置和批量获取，不支持分布式

图形用户界面, 文本, 应用程序

中度可信度描述已自动生成

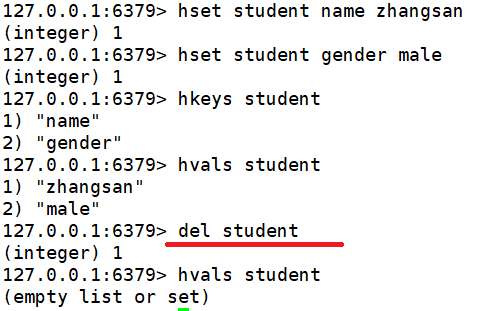
#### Hdel key field:

删除属性和值

文本

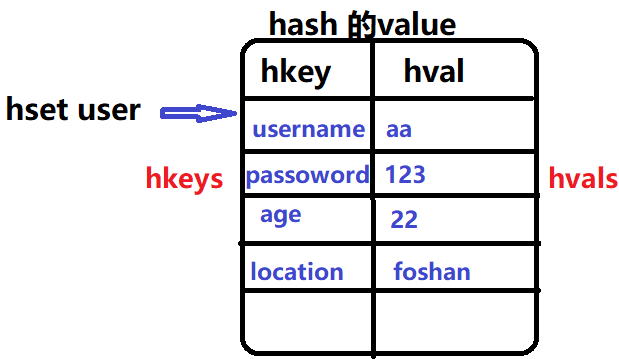
中度可信度描述已自动生成

删除整个对象 del



#### Hkeys 和Hvals:

单独获取对象的属性名，或者属性的值



文本

描述已自动生成

#### Hlen key

获取属性个数（长度）

文本

描述已自动生成

#### Hincrby

**只有hincrby 没有hdecrby,也没有incr和decr**

和String类型特性区别的一个体现

String对应存储对象数据：需要在redis中修改对象某个属性的数值（18->22）

{“id”: "20182005105","age"：18,“name”:"zhangsan"}

不能直接在redis中获取age将18变成22，

必须从redis读出来，转化成对象，通过对象的set属性修改，然后重新放回redis。

hash存储对象数据

user id 20182005105

age 18

name zhangsan

Hincreby user age 4

## 3.3 List链表数据类型

### 3.3.1 List 类型数据结构

**双向链表**



list01--{“200”,"300","400"}

### 3.3.2 命令

查看链表list和设置链表数据lpush（左表示上，右表示下）

#### lpush key value [values]

对于一个链表,从左(上)开始插入数据, lpush插入的数据在最上面

文本

中度可信度描述已自动生成

#### lrange key 起始下标 结束下标

（展示范围（从上到下/左到右），结束下标是-1表示到尾部）

图片包含 表格

描述已自动生成

#### rpush key value

从下向上插入数据，rpush插入的数据在最下面

文本

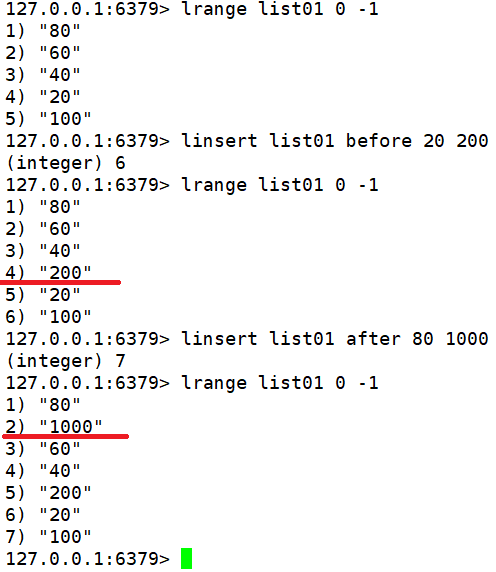
描述已自动生成

#### linsert

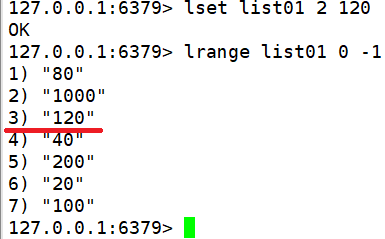
从链表中间操作数据,从左侧（上面）开始找到第一个相同元素插入数据（before after）

linsert list01 before 100 300

在list01链表中，找到100这个元素，从其前面插入数据one 300 200 one 100 400



#### lset key index value对下标对应的数据修改



#### lrem 从下往上删除

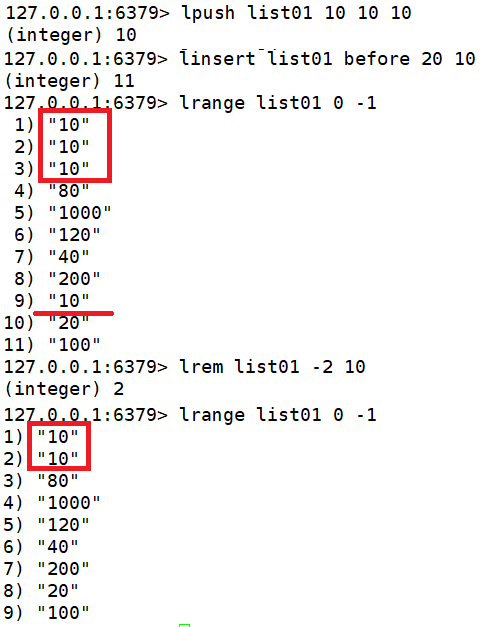
这个命令没有对应的rrem命令

lrem key count value：从左向右寻找count个value相同值元素删除。

count>0从左往右删（从上往下找）

count=0 全删

count<0从右往左删（从下往上找）



#### ltrim 保留链表指定范围内的数据元素剩余的删除

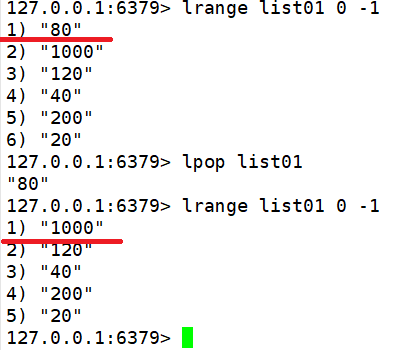
ltrim key 起始位置 结束位置（-1表示到尾部）

图表, 文本, 瀑布图

中度可信度描述已自动生成

#### lpop 从list头部删除元素

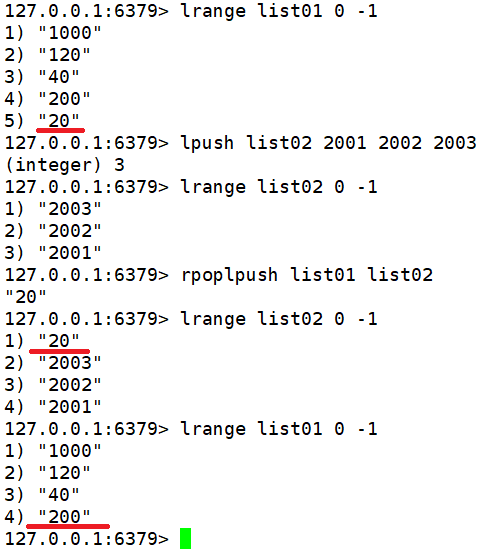
返回删除结果（类似remove） 秒杀逻辑,可以利用list对象实现阻止超卖现象



### 3.3.3两个链表的数据交互

#### rpoplpush

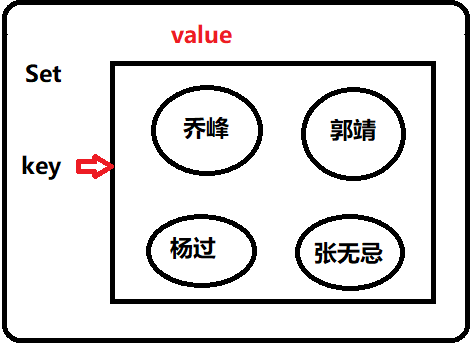
从第一个list的尾部删除数据，移动到第二个list的头部添加



## 3.4 Set结构

集合用来保存多个字符串元素，和列表不同的是不允许有重复元素，并且集合中元素是无序的。

### 3.4.1 Set 类型数据结构



最常见的应用场景，比如兴趣爱好，社交平台中关注人存在一个集合中，将其所有粉丝存在一个集合

### 3.4.2单集合操作

#### sadd key member添加

127.0.0.1:6379> sadd favor dancing songing reading

(integer) 3

#### smembers key获取所有元素

文本

中度可信度描述已自动生成

#### srem key member删除

文本

中度可信度描述已自动生成

#### scard key 返回元素个数

 127.0.0.1:6379> scard favor

(integer) 2

#### sismember key element判断元素是否在集合中

文本

描述已自动生成

#### srandmember key 随机抽取几个元素

图表, 文本

描述已自动生成

### 3.4.3集合间的操作

#### sinter set1 set2 交集

图形用户界面, 文本, 应用程序

描述已自动生成

#### suinon set1 set2 并集

文本

描述已自动生成

#### sdiff set1 set2 差集属于set1不属于set2的元素为差集

文本

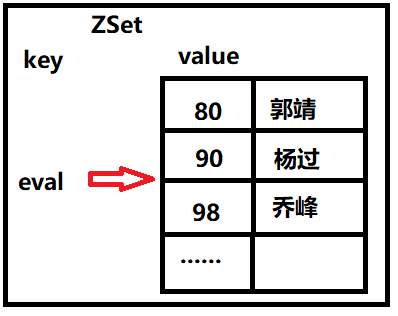
中度可信度描述已自动生成

## 3.5 ZSet结构（sorted set）

### 3.5.1 ZSet数据结构

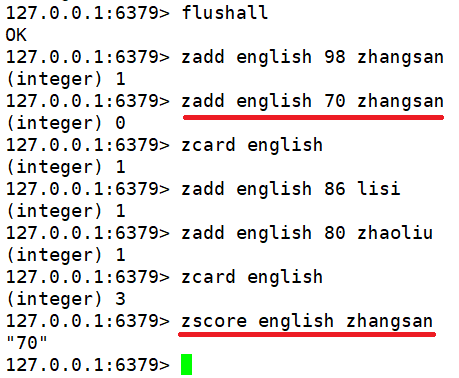
在集合的基础上绑定一个score作为排序的依据

比较典型的使用场景是**排行榜系统**。例如视频网站需要对用户上传的视频做排行榜



### 3.5.2单集合（单个集合内操作）

#### zadd key score member添加成员



#### zscore key member查看某个成员的分数

文本

描述已自动生成

#### 计算成员的升序排名 zrank key member

文本

描述已自动生成

 升序排列，最低分在最前面。

#### 删除成员 zrem key member

文本

描述已自动生成

原先是3个元素，删除一个后剩下两个元素。

#### 增加成员的 分数zincrby key increment member

文本

描述已自动生成

#### 返回指定排名范围的成员 zrange key start end（-1表示最后一个）

文本

描述已自动生成

#### 返回指定分数范围的成员 zrangebysore key min max

文本

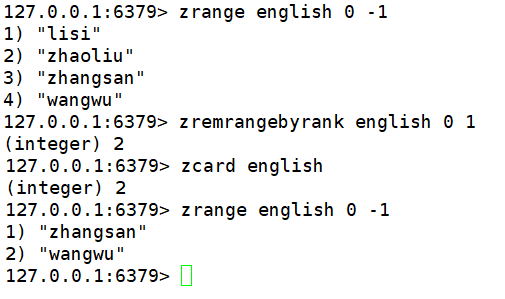
中度可信度描述已自动生成

#### 返回指定分数范围成员个数 zcount key min max

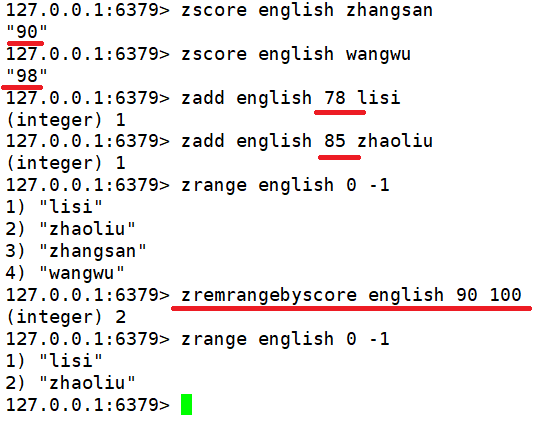
图片包含 图表

描述已自动生成

#### 删除指定排名内的升序元素 zremrangebyrank key start end



#### 删除指定分数范围的成员 zremrangebyscore key min max



### 3.5.3 集合间的操作

#### 交集 zinterstore destination numkeys key1 key2

制定一个交集并集查询结果的输出集合desitination

指定查询交集并集的总zset个数numkeys ,给定符合数量的多个zset集合

发现：交集得到的新集合元素的得分是所有相同元素的得分求和的结果。

