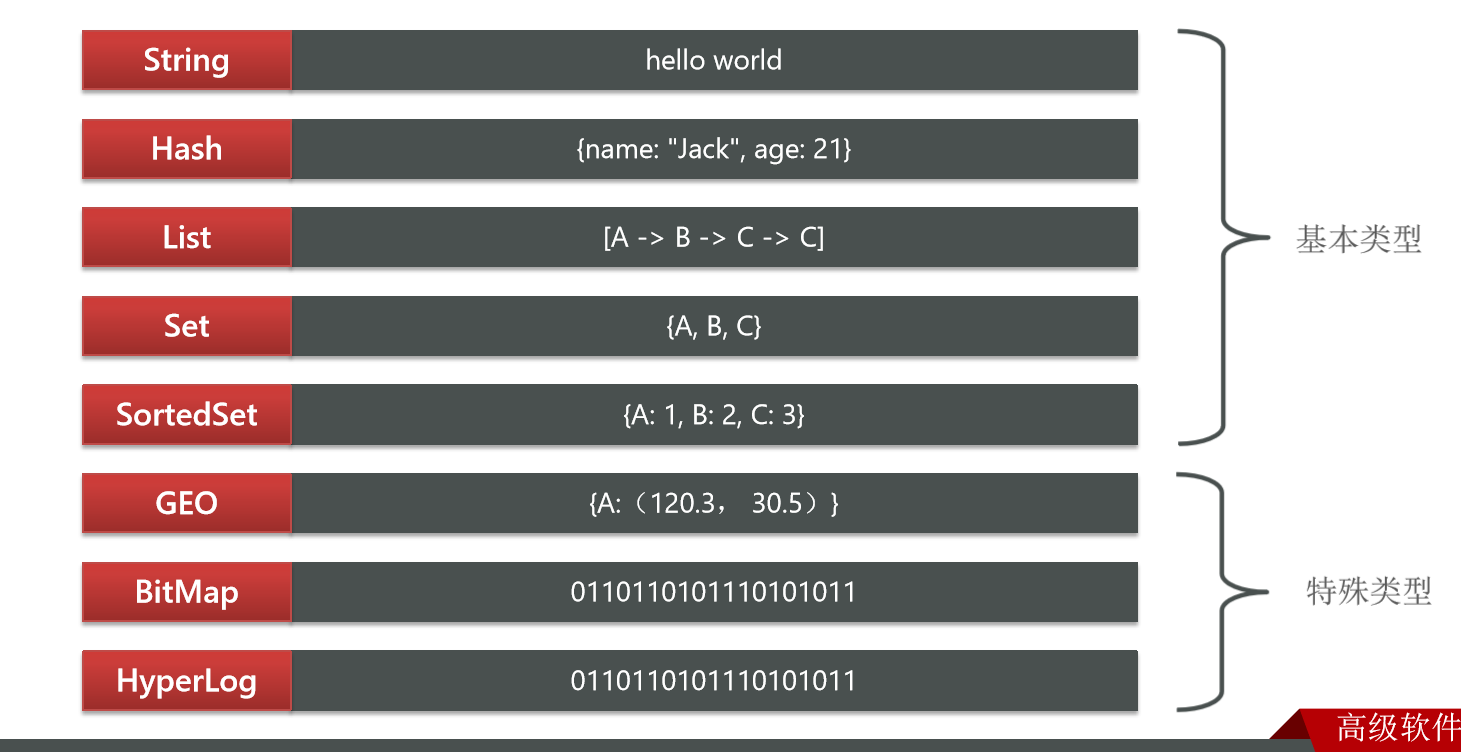
# Redis命令

## Redis数据结构介绍

Redis是一个key-value的数据库，key一般是String类型，不过value的，类型多种多样。常见的有下面8种，最常用的是基本类型中的5种：String、Hash、List、Set、SortedSet。



### Redis基本类型：是值value的类型

1：string（就是字符串类型，）

2：hash（就是一个Map）

3：list（其实是一个链表，a指向b，b指向c，c指向d）

4：set（其实就是一个无序不重复的列表）

5：zset（又叫SortedSet，其实就是一个有序不重复的列表）

### Key的结构的最佳实践：层级存储

Redis的key允许有多个单词形成层级结构，多个单词之间用冒号“：”隔开，格式如：项目名:业务名:类型:id

当然这个格式并非固定，可以根据自己的需求来删除或添加词条。

例如我们的项目名称叫heima，有user和product两种不同类型的模块，我们可以这样定义key：

User相关的key：heima:user:1

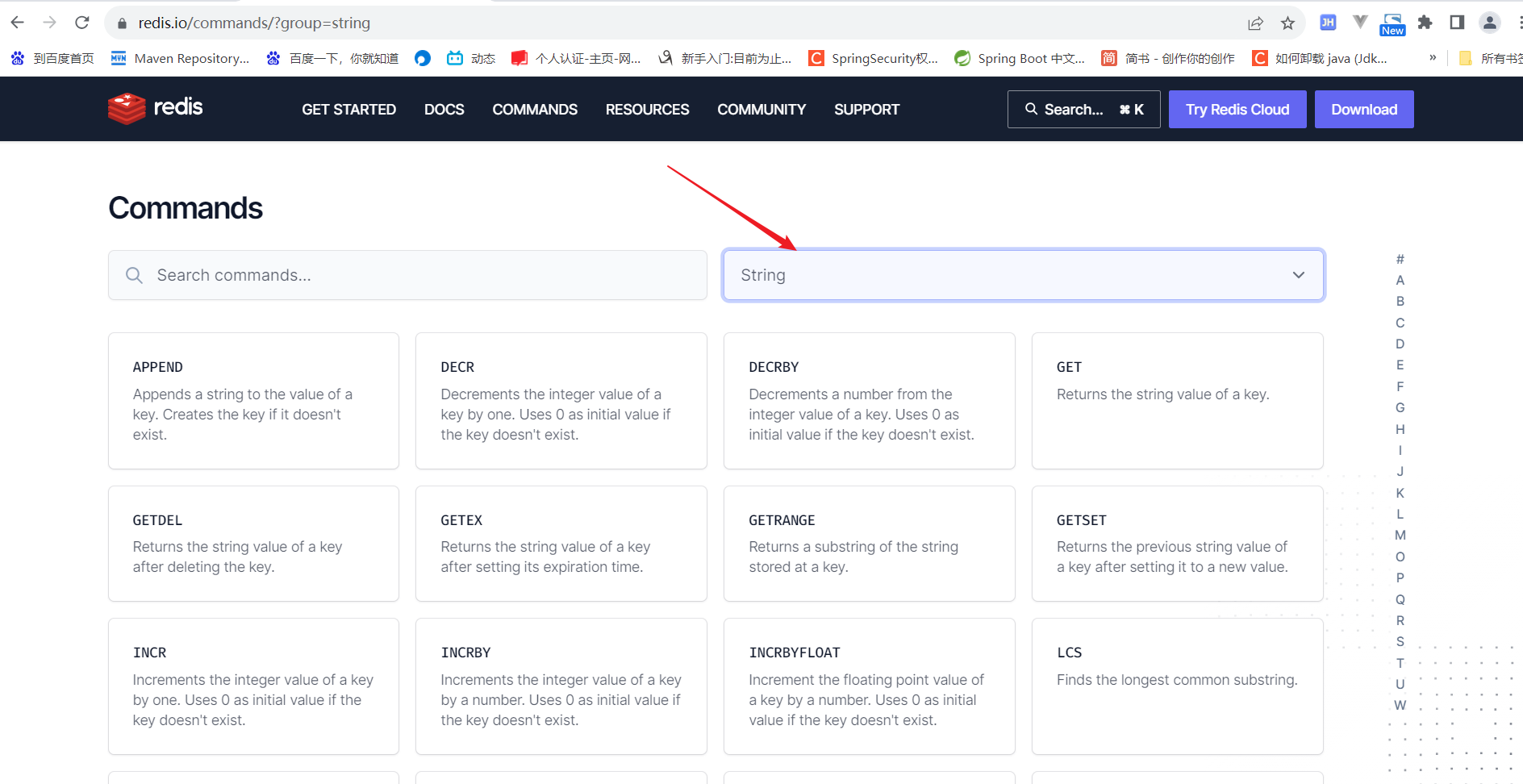
product相关的key：heima:product:1

如果value是一个java对象，例如一个User对象，则可以将对象序列化为JSON字符串后存储：

heima:user:1 {“id”:1, “name”:”jack”, “age”:28}

## Redis通用命令：

通用指令是部分数据类型的，都可以使用的指令，可以在官方文档查看学习：<https://redis.io/commands/>



也可以使用客户端控制台帮助文档：help @generic

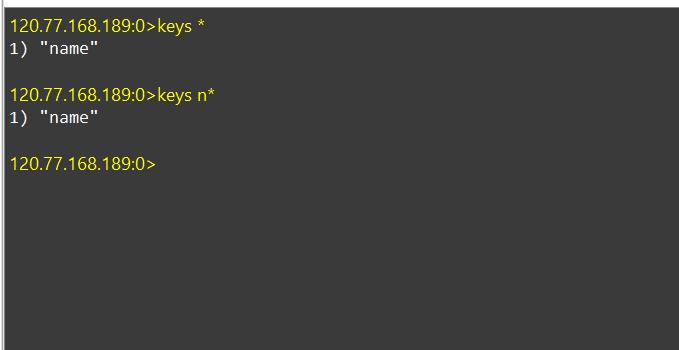
HELP COMMAND：查看命令的帮助文档，通过help [command]可以查看一个命令的具体用法

Eg：help keys

### KEYS：查看符合模板的所有key

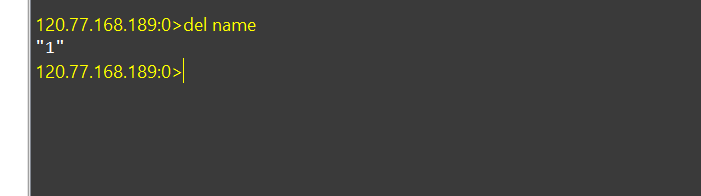
Eg：keys \*

慎用，数据量大，耗时；Redis单线程，阻塞。



### DEL：删除一个或多个指定的key，返回删除的键的数量

Eg：del key



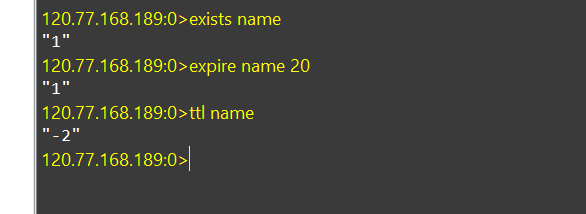
### Exists：判断key是否存在

Eg：exists name



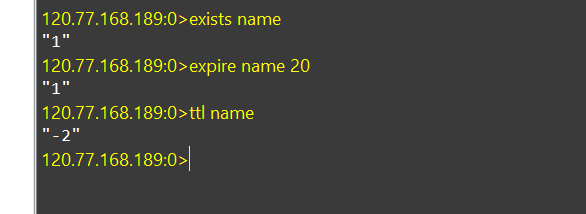
### EXPIRE：给key设置一个有限期，有效期到期时该key会被自动删除

Eg：expire name 20



### TTL：查看一个key的剩余有效期，返回秒，-1永久有效，-2失效

Eg：ttl key



## String类型：

我们知道Redis是一个键值型的非关系型数据库，只不过value的类型五花八门，而value为String类型时，就是我们理解的字符串，因此String类型可以认为是Redis中最简单的数据类型。使用起来也非常地简单方便。但是都是字符串，字符串表现出来的形式还是会有一些差异的。

比如下面有三个键值对，key为msg的value是“hello word”，这就是普通的字符串；第二个key为num的value是10，尽管这里10也是一个字符串，但是看起来的样子是一个数字，第三个key为score的value为92.5，同样的意思，是一个看起来是浮点数的字符串。

Redis在做数据存储的时候，这些value都是字符串，所以底层都是由字节数组去存储的，只不过为了节省空间，数值类型的字符串在做编码的时候会有不同，会把数字字节转为二进制的形式转为字节去存储，这样一来一个字节就能表示很大的数字了，所以就会更节省空间，而字符串类型value就只能把字符转成对应的字节码，然后再去存储，相对来讲占用内存多一些。

所以字符串表现形式不一样，底层编码不一样，但本质都是数组。如果你愿意的话，你甚至可以把一张图片编译成字节存到我们的String类型里。



### 常见命令：

#### 单个增set、单个查get：

SET：添加或者修改已经存在的一个String类型的键值对

GET：根据key获取String类型的value

#### 批量增mset、批量查mget：

MSET：批量添加多个String类型的键值对

MGET：根据多个key获取多个String类型的value

#### 操作value数值自增自减INCR：

除了基本的存取之外，既然Redis存储数值表现类型字符串的时候，编码有不同，所以，对于数值类型的value，Redis就有特殊的功能了，比如自增、自减。

INCR：让一个整形的key自增1

INCRBY：让一个整形的key自增并指定步长，比如 incrby num 2 让num值自增2

INCRBYFLOAT：让一个浮点类型的数字自增并指定步长

组合命令：

#### SETNX：添加一个String类型的简直对，前提是这个key不存在，否则不执行，即是真正的新增，如果这个key存在就不添加，不存在就添加。

setnx其实是一个组合命令哦，先是set，然后是nx，即：

**setnx name 黄远辉**

等价于

**set name 黄远辉 nx**

#### SETEX：添加一个String类型的简直对，并且指定有效期

setex其实也是一个组合命令哦，先是set，然后是ex，即：

**setex name 10 黄远辉**

等价于

**set name 黄远辉 ex 10**

### Key的结构的最佳实践：层级存储

Redis的key允许有多个单词形成层级结构，多个单词之间用冒号“：”隔开，格式如：项目名:业务名:类型:id

当然这个格式并非固定，可以根据自己的需求来删除或添加词条。

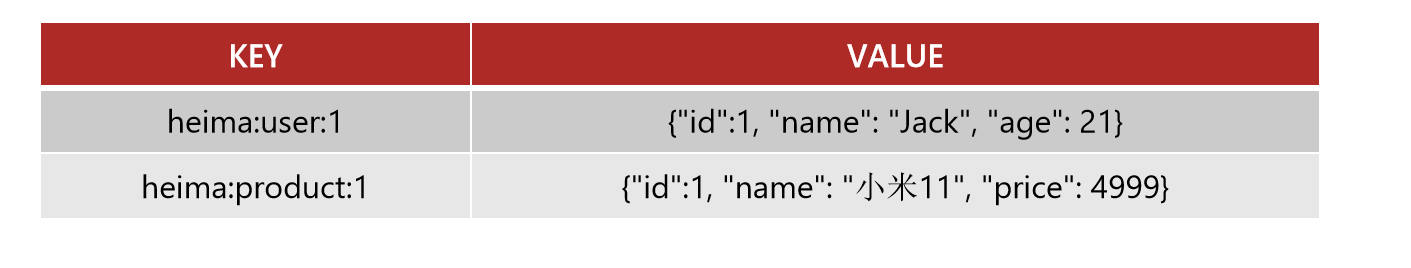
例如我们的项目名称叫heima，有user和product两种不同类型的模块，我们可以这样定义key：

User相关的key：heima:user:1

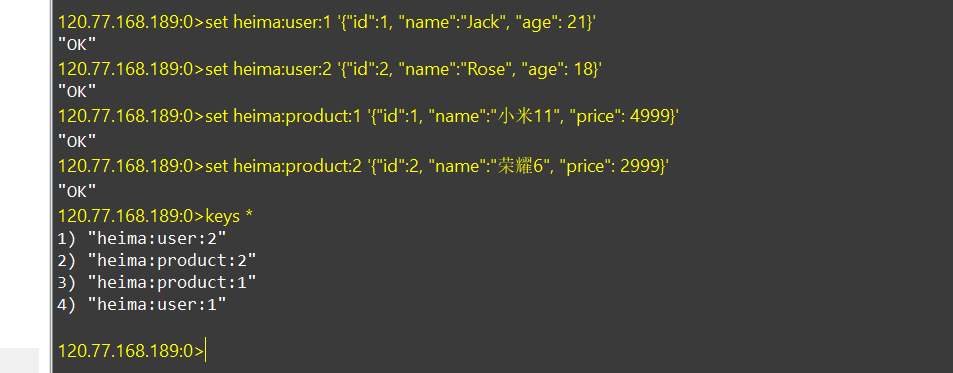
product相关的key：heima:product:1

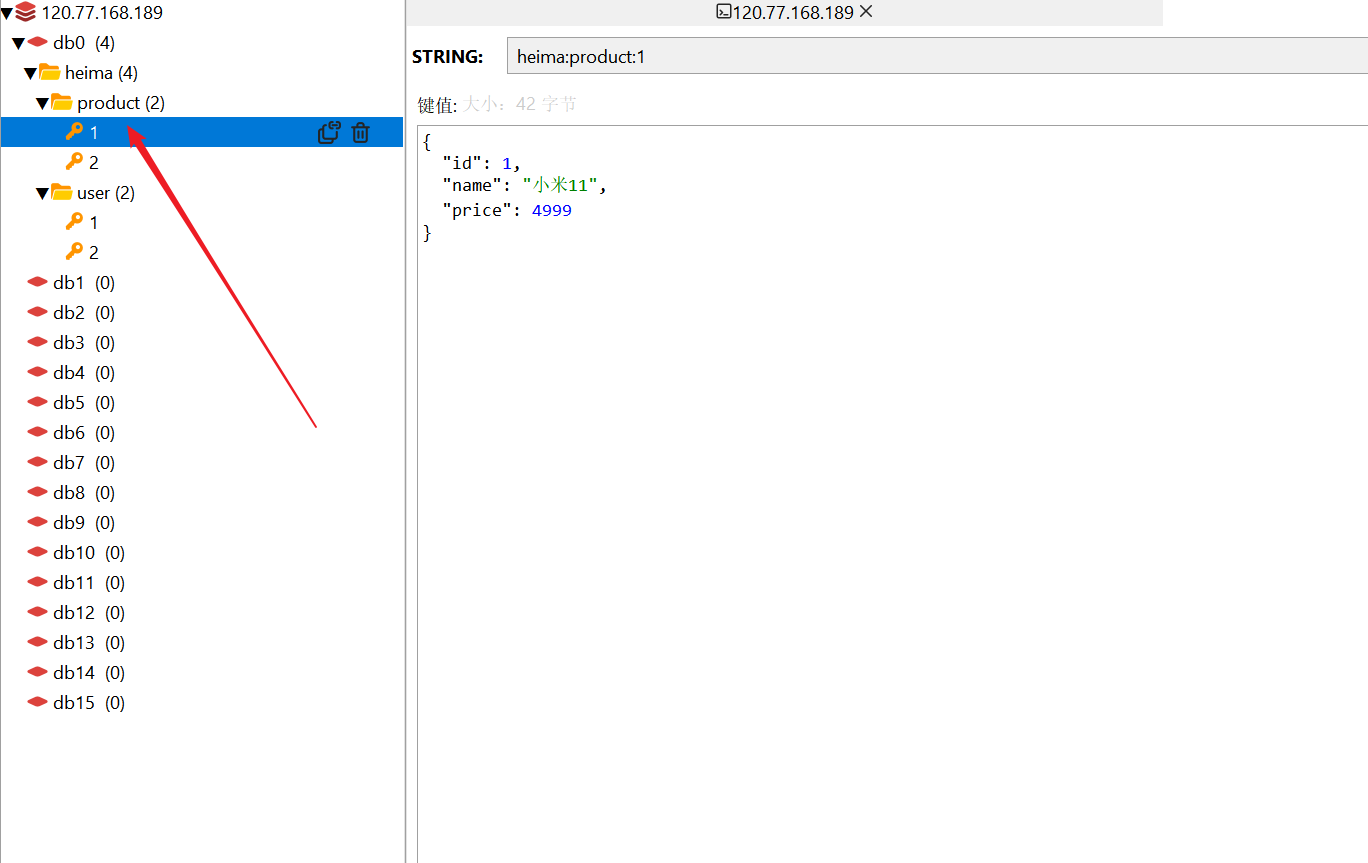
如果value是一个java对象，例如一个User对象，则可以将对象序列化为JSON字符串后存储：

heima:user:1 {“id”:1, “name”:”jack”, “age”:28}



你看，我们的key以冒号隔开时，是不是很自然就形成了层级结构啊！可以避免id相同时的数据冲突，自然就把数据分离了。

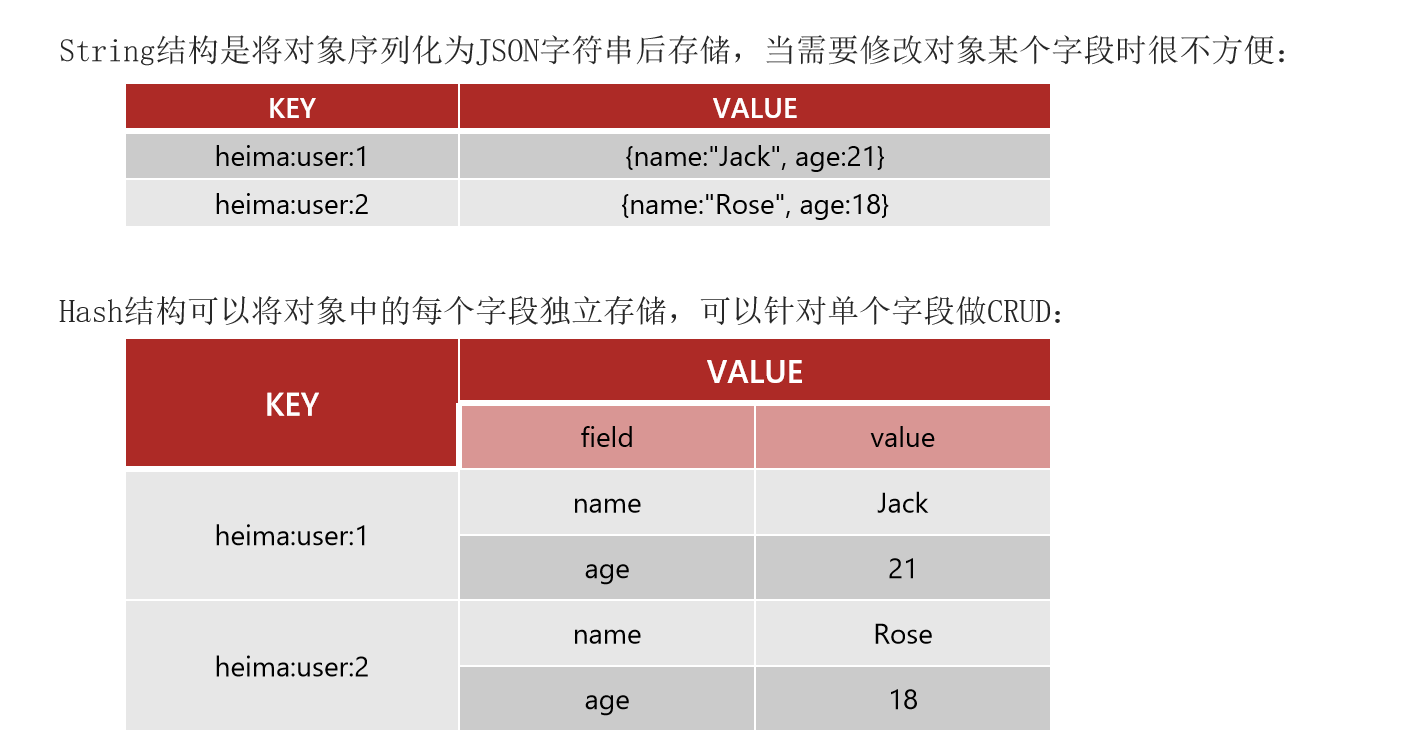




## Hash类型：

Hash类型，也叫散列，其value是一个无序字典，类似与Java中的HashMap结构！也就是说我们的Redis本身是一个key-value，而hash类型的value又是一个哈希表，key-value的结构！

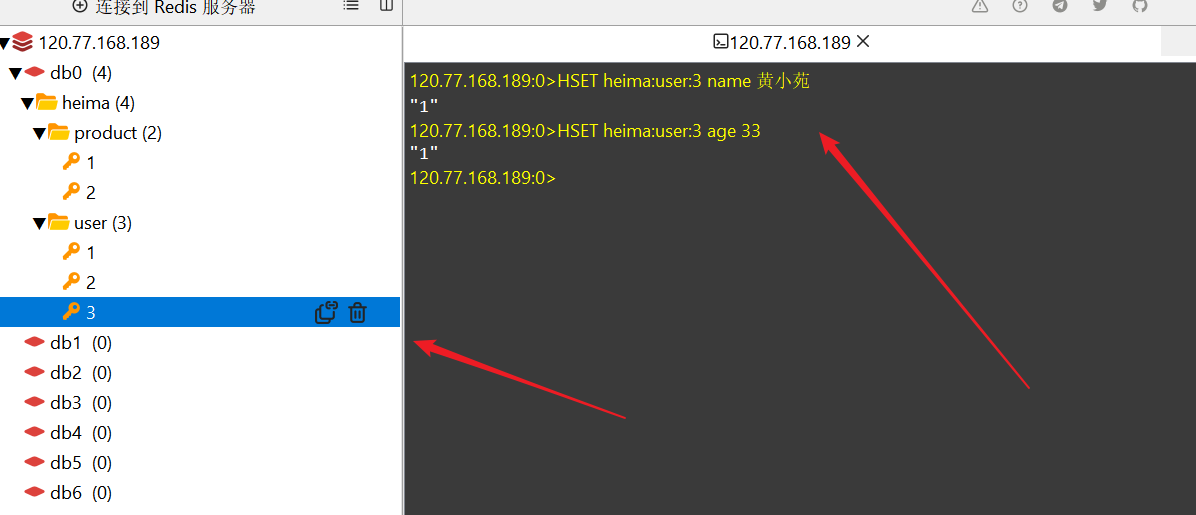
Redis中并没有MySql中的Table的概念，String类型是将对象序列化为JSON字符串后存储，但当需要修改对象某个字段时很不方便；Hash结构可以将对象中的每个字段独立存储，可以针对单个字段做CRUD。

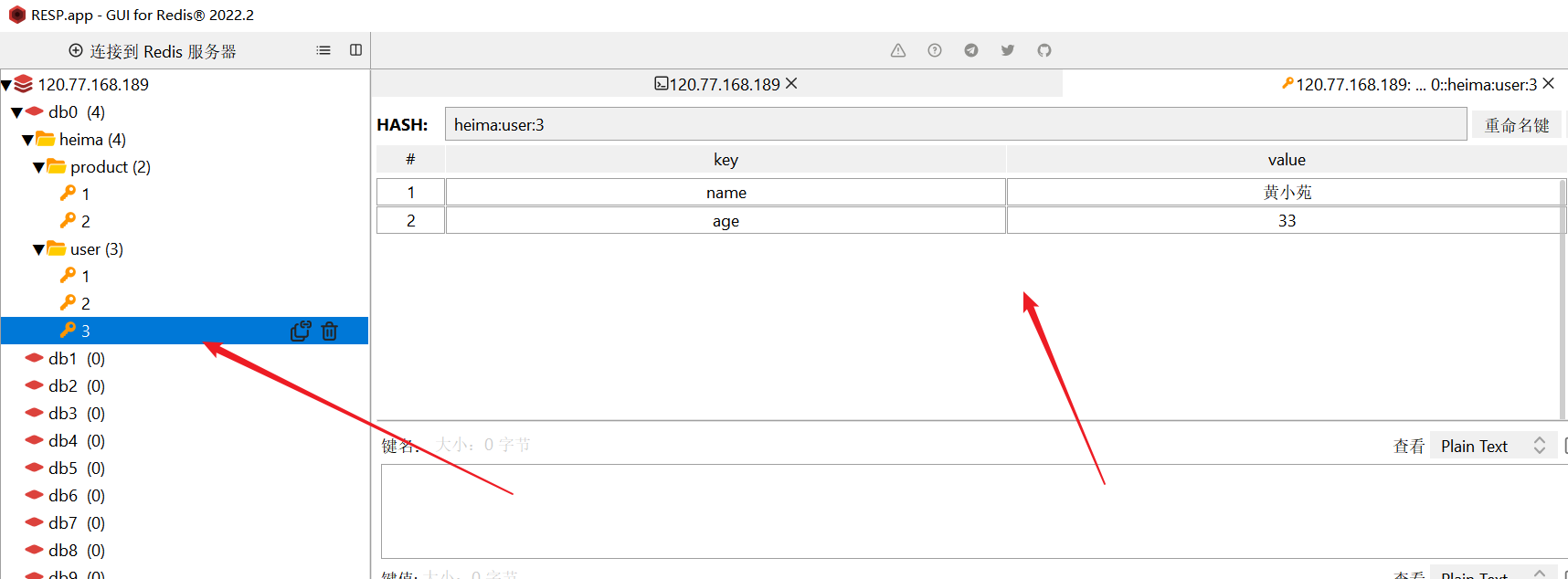


### 常见命令：

#### 单个增HSET、单个查HGET：

HSET key field value：添加或者修改hash类型的key的field值





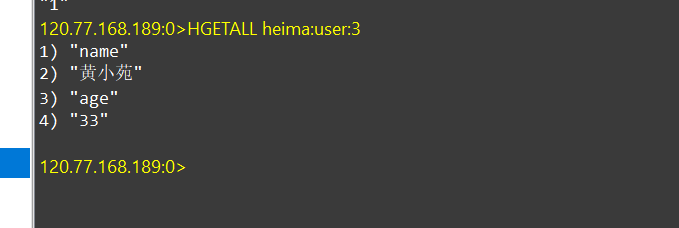
HGET key field：获取一个hash类型key的field值

#### 批量增HMSET、批量查HMGET：

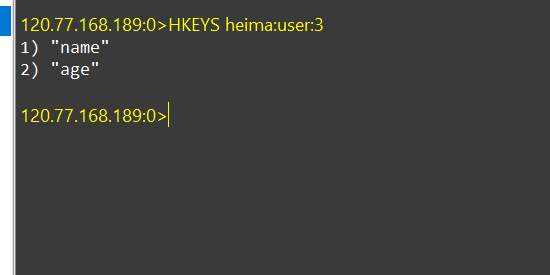
HMSET key field2 value1 field2 value2：批量添加多个hash类型的key的field值

HMGET key field1 field2：根据多个hash类型key的field值

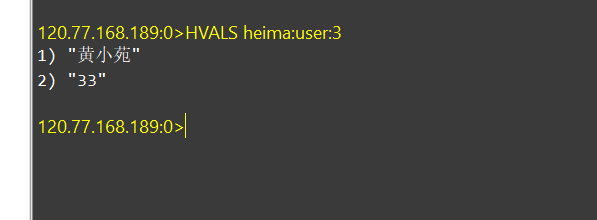
#### HGETALL key：获取一个hash类型的key中的所有的field和value



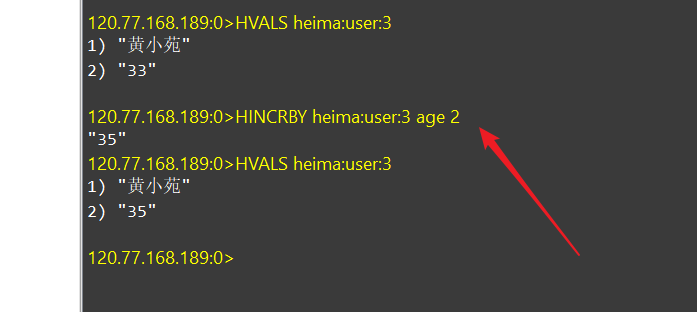
#### HKEYS key：获取一个hash类型的key中的多有的field



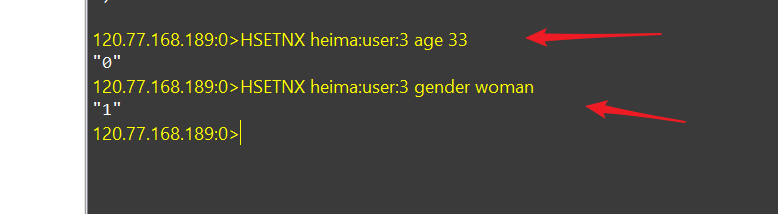
#### HVALS key：取一个hash类型的key中的多有的value



#### HINCRBY key filed 步长：让一个hash类型key的字段值自增并指定步长



#### HSETNX：添加一个hash类型的key的field值，前提是这个field不存在，否则不执行



## List类型：

reids中的List类型与java中的LinkedList类似，可以看作是一个双向链表结构。既可以支持正向检索，也可以支持反向检索。

特征也与java中的LinkedList类似：

有序；

元素可以重复；

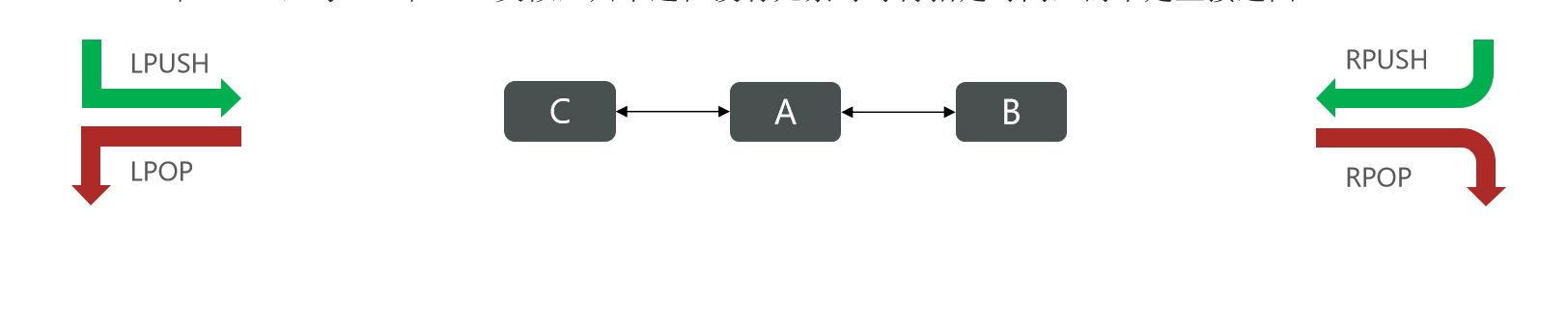
插入和删除速度快；

查询速度一般；

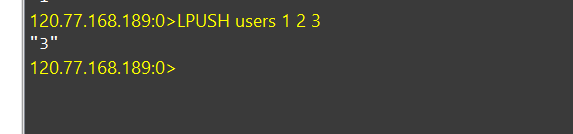
适用场景：对顺序有要求的场景，比如朋友圈点赞、朋友圈的评论，有谁先谁后的

### 常见命令：

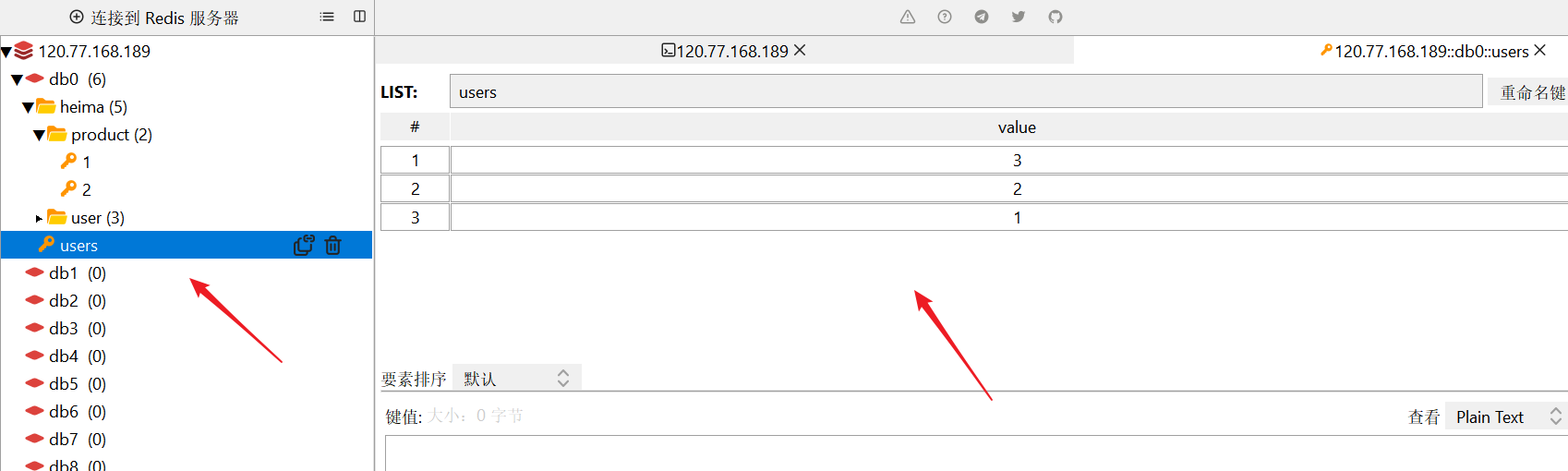
L是左侧，即队首！R是右侧，即队尾！



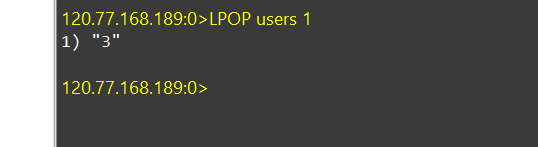
#### LPUSH key element...：向列表左侧插入一个或多个元素



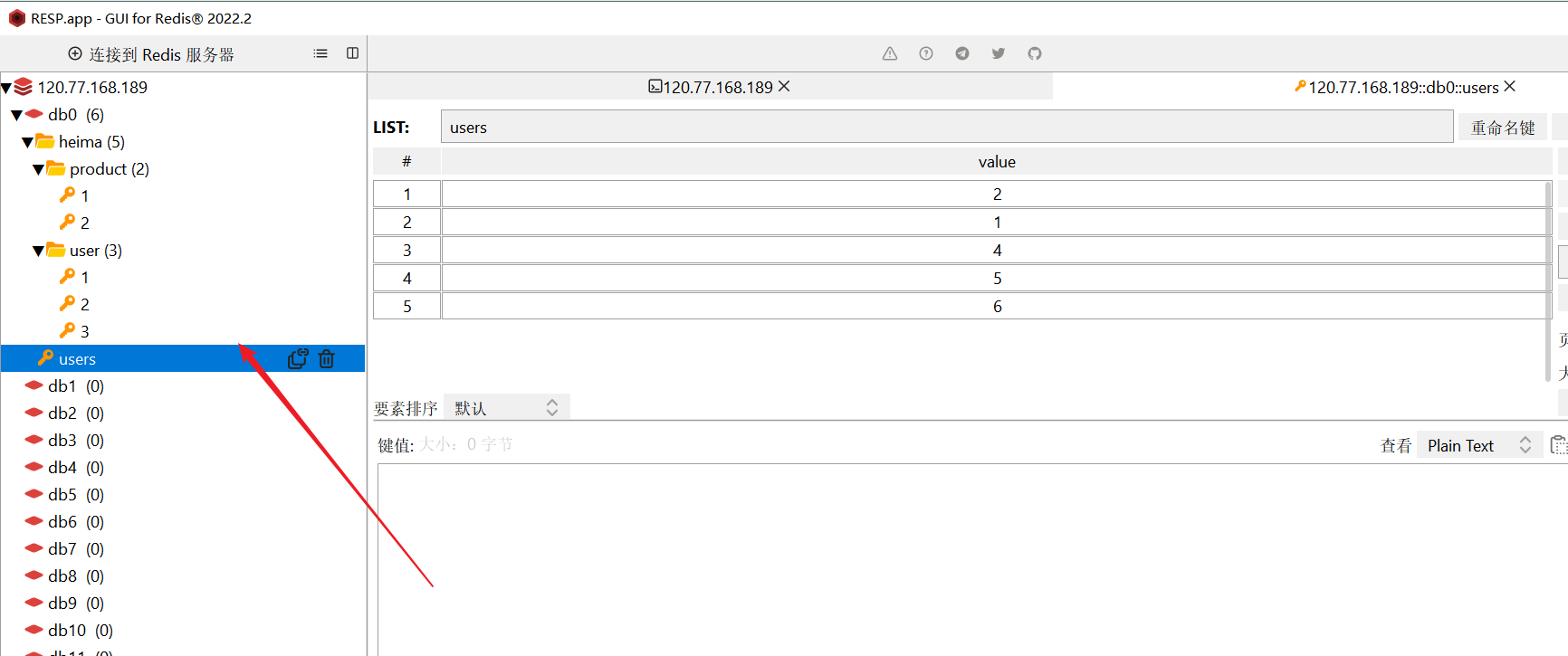
队首是3！



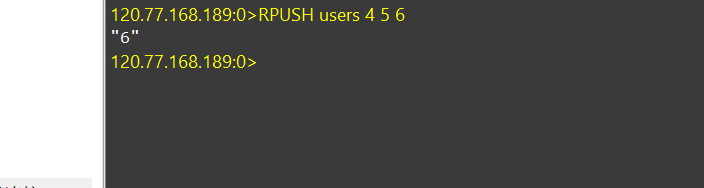
#### LPOP key：移除并返回列表左侧的第一个元素，没有则返回nil



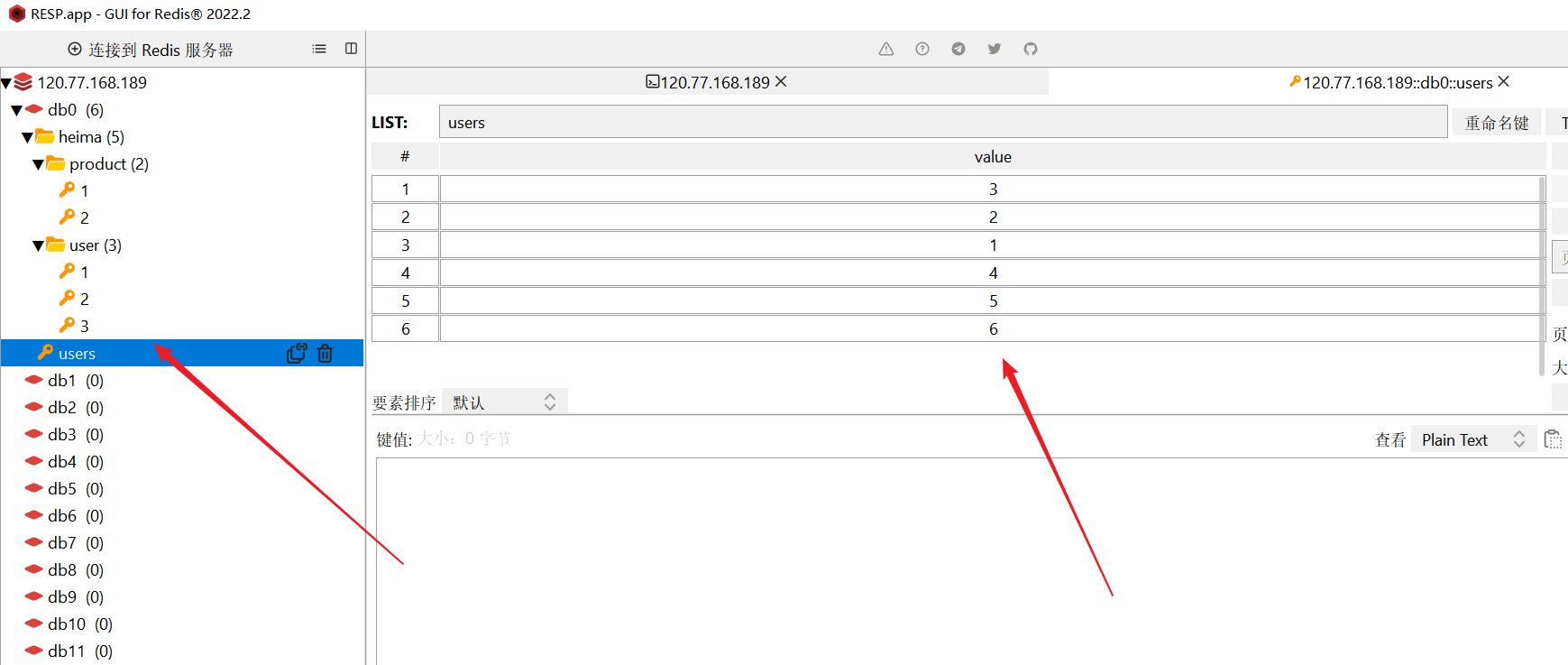
会移除队首的3！



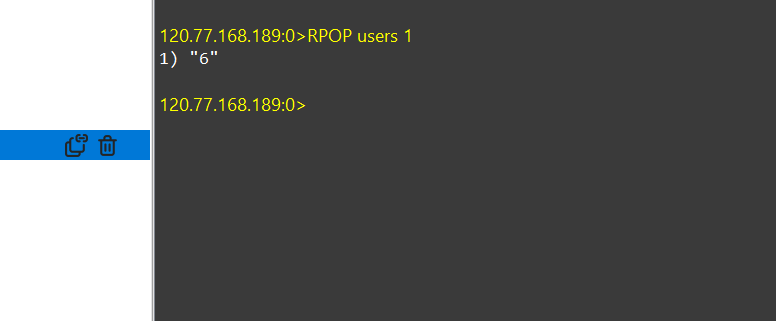
#### RPUSH key element...：向列表右侧插入一个或多个元素



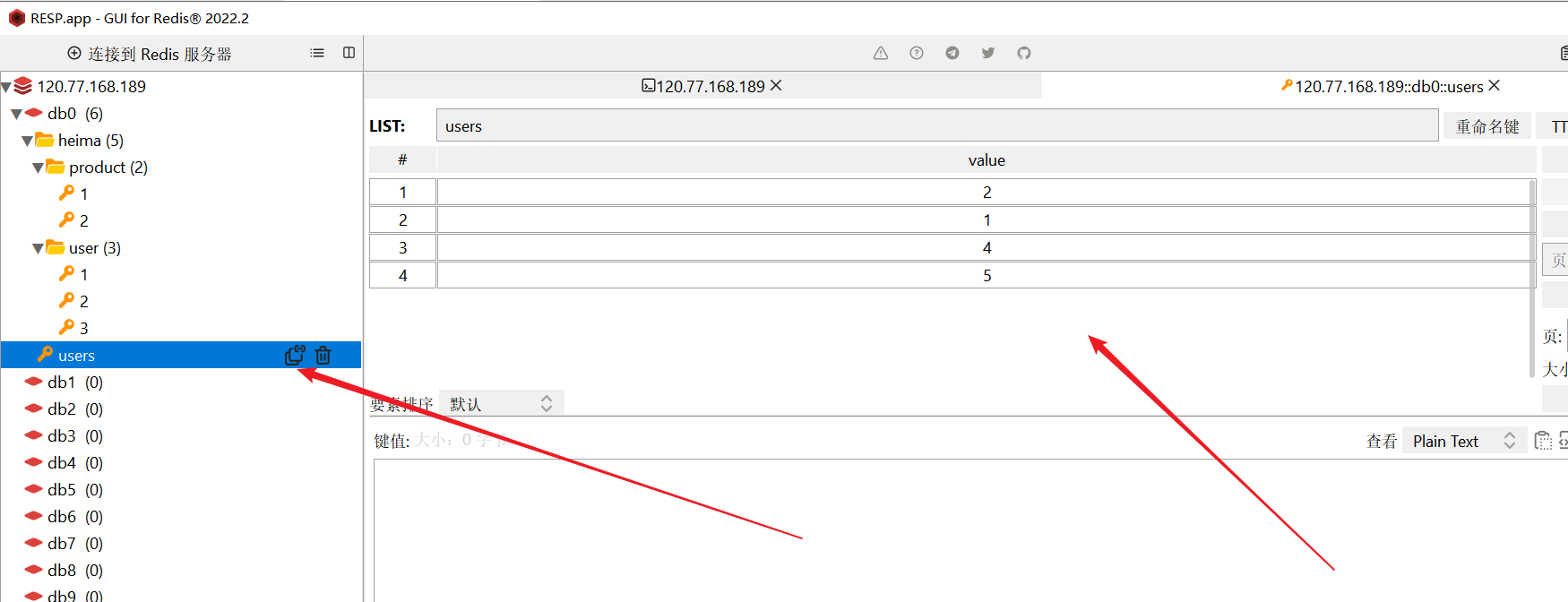
6是队尾！



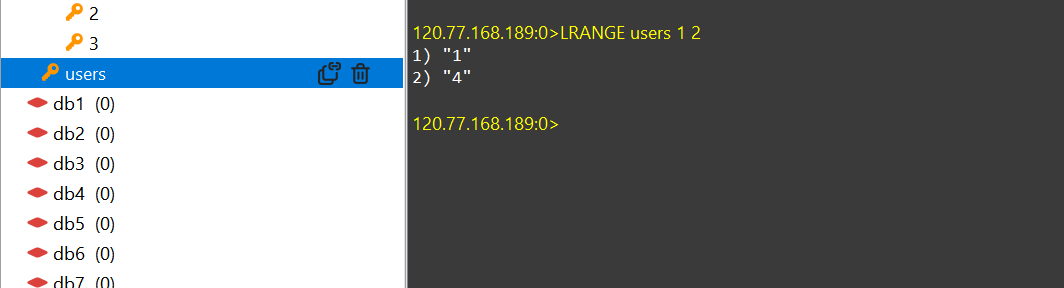
#### RPOP key：移除并返回列表右侧的第一个元素，没有则返回nil



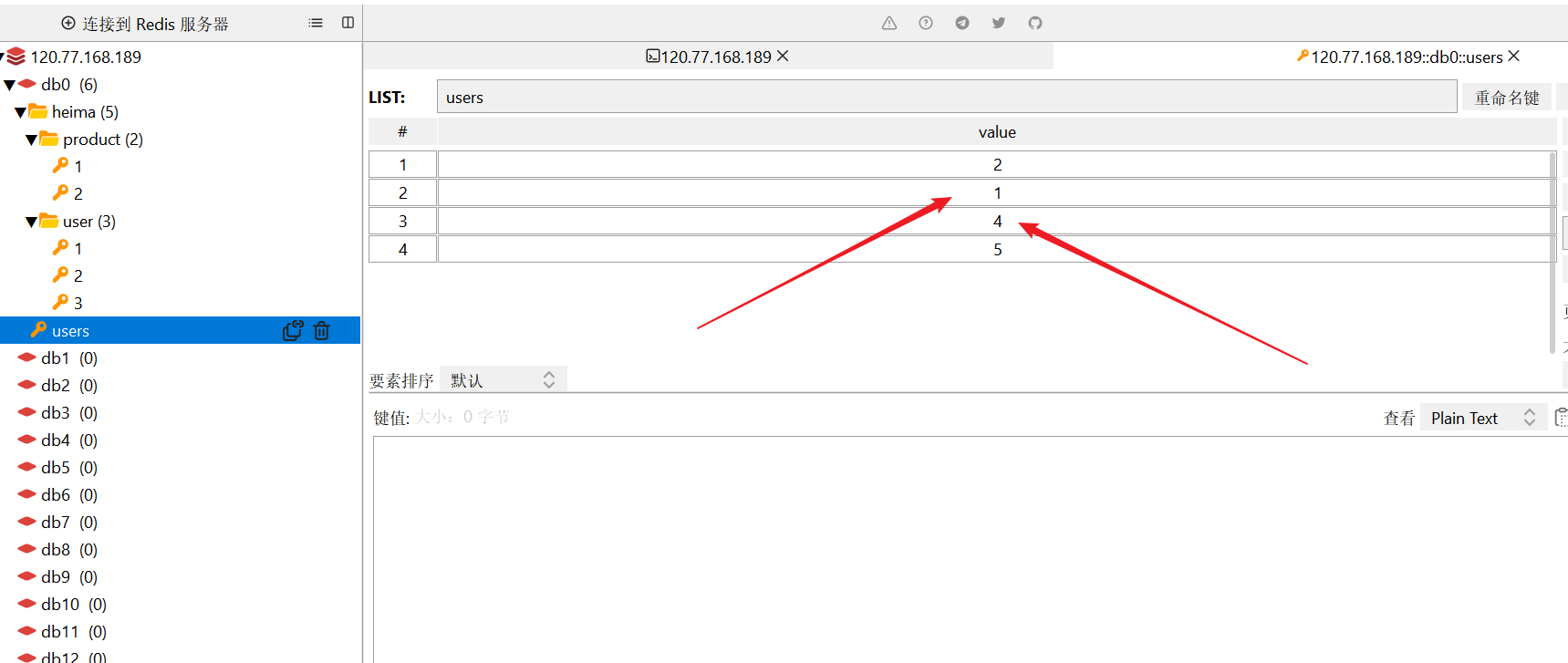
会移除队尾的6！



#### LRANGE key star end：返回一段角标范围内的所有元素（注意没有RRANGE！）



队首是0开始！



#### BLPOP key和BRPOP key：与LPOP和RPOP类似，只不过在没有元素时会等待指定时间，而不是直接返回nil

### 总结：

#### 1：如何利用List结构模拟一个栈？

答：入口和出口在同一边！

#### 2：如何利用List结构模拟一个队列？

答：入口和出口在不同边！

#### 3：如何利用List结构模拟一个阻塞队列？

答：入口和出口在不同边，出队列时采用BLPOP 或者 BRPOP

## Set类型：

Redis中的Set结构类型与java中的HashSet类似，可以看作是一个value为null的HashMap。因为也是一个hash表，因此具备与Java中的HashSet类型的特征：

无序；

元素不可以重复；

查询速度快；

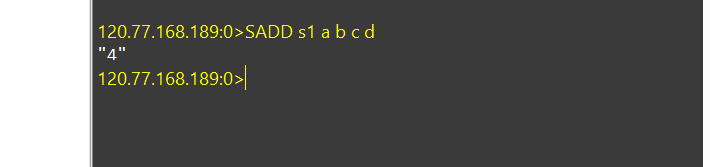
支持交集、并集、差集等功能

适用场景：好友列表、共同好友、关注列表等，社交型的应用中使用比较广泛。

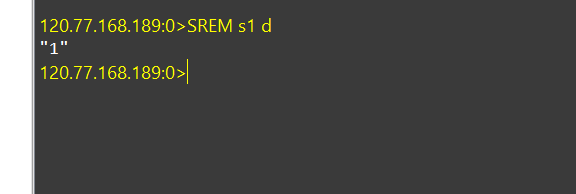
### 常见命令：

#### 单个集合的操作：

##### SADD key member...：向set中添加一个或多个元素



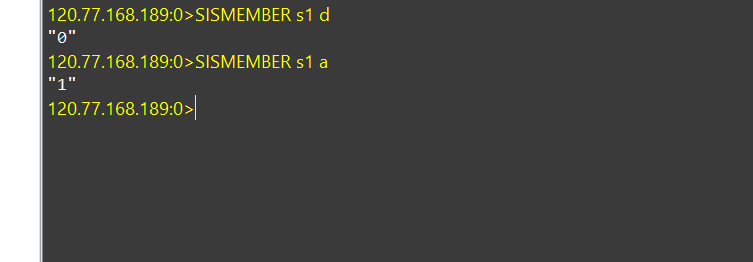
##### SREM key member...：移除set中的指定元素



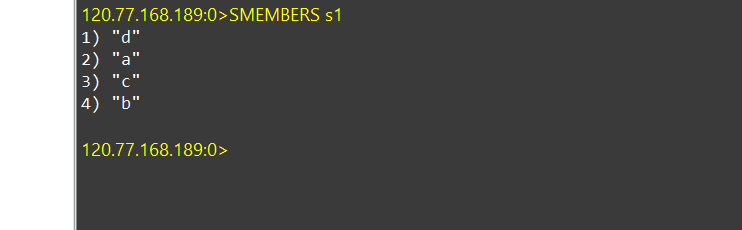
##### SCARD key：返回set中元素的个数



##### SISMEMBER key member：判断一个元素是否存在于set中

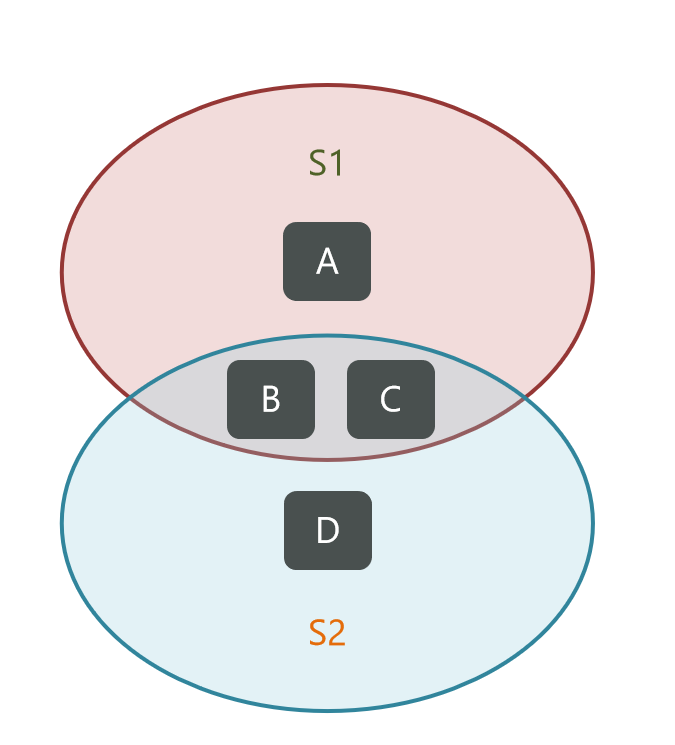


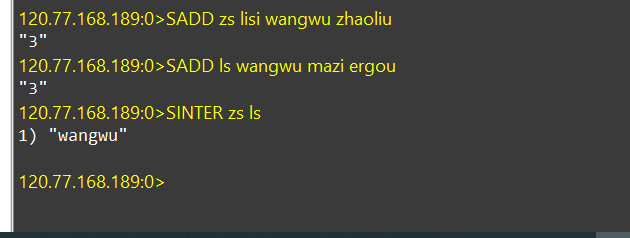
##### SISMEMBERS：获取set中的多有元素



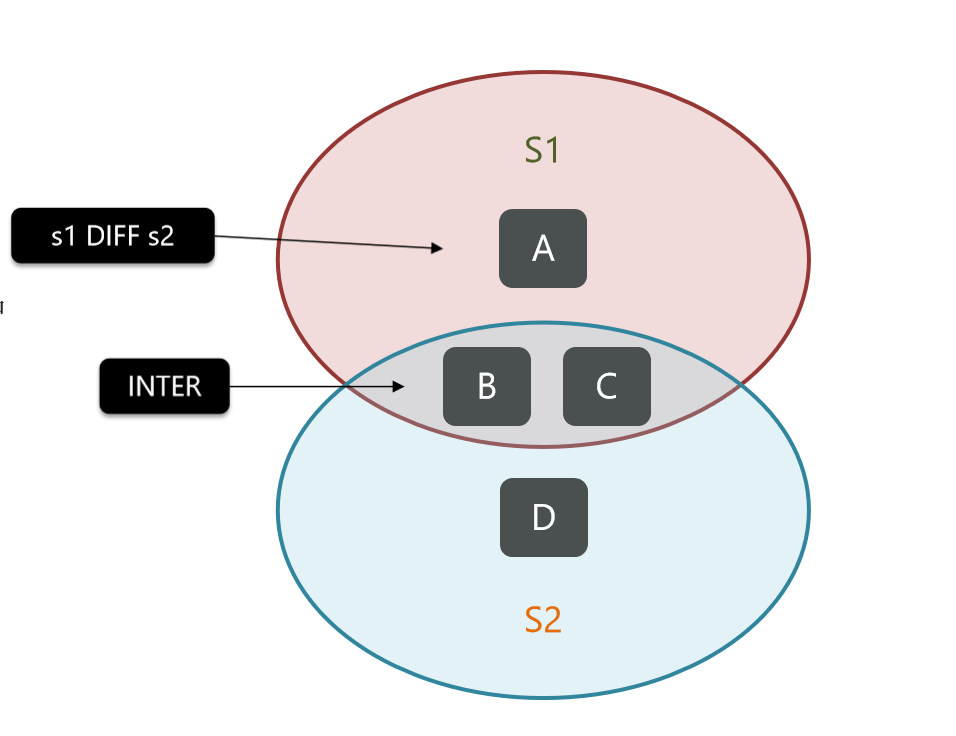
#### 多个集合的操作：

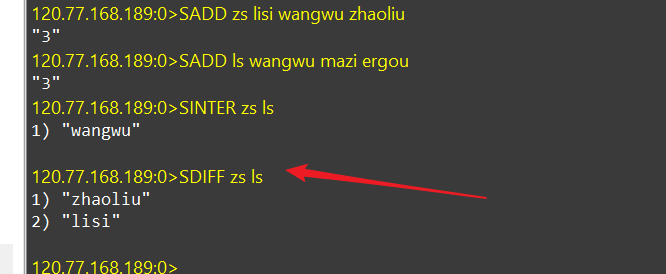
##### SINTER key1 key2....：求key1与key2的交集



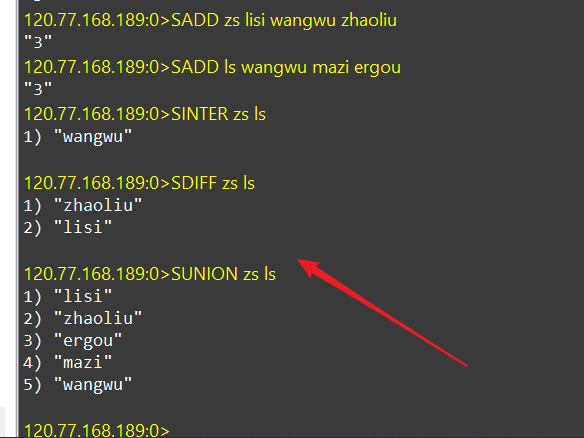


##### SDIFF key1 key2....：求key1与key2的差集：就是key1有 key2没有





##### SUNION key1 key2....：求key1与key2的并集



## SortedSet类型：

Redis中的SortedSet是一个可排序的set集合，与java中的TreeSet有些类似，TreeSet正是一种可排序的集合。但底层数据结构却区别很大，java中的TreeSet底层是红黑树来实现的，而Redis中的SortedSet则不是！SortedSet中的每一个元素都带有一个score属性，可以基于score属性对元素排序，底层的实现是一个跳表（SkipList）加hash表。

SortedSet具备的特性：

可排序；

元素不重复；

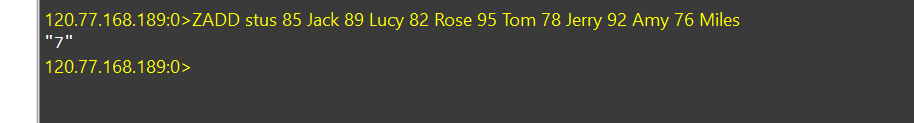
查询速度快

因为SortedSet的可排序性，经常被用来实现排行榜这样的功能。

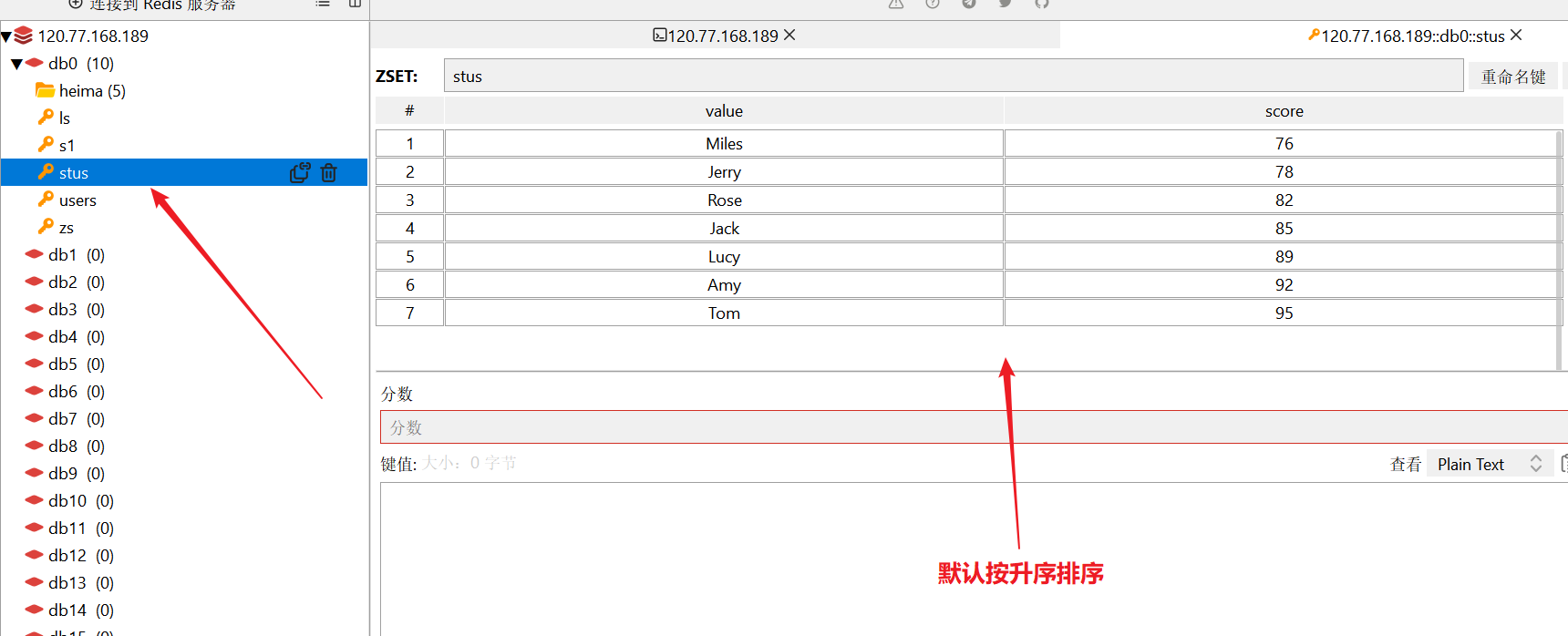
### 常见命令：（排名默认都是按照升序的排名，如果要降序则在命令的Z后面添加REV（反转的意思）即可）

#### 单个集合的操作：

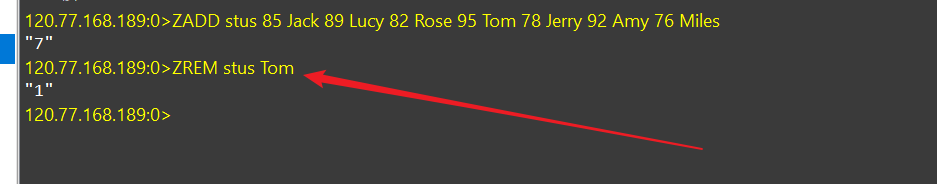
##### ZADD key score member：添加一个或者多个元素到sorted set，如果已经存在则更新其score值中。



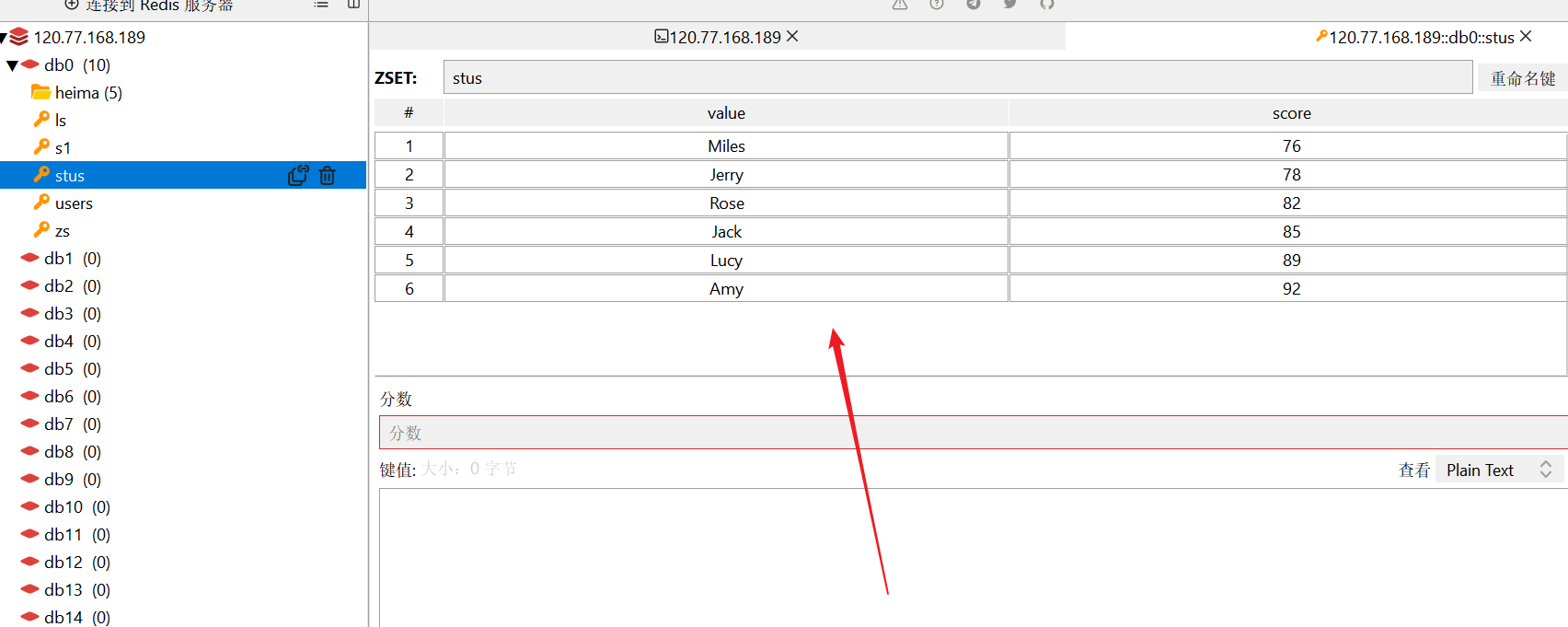
会自动根据score帮你做排序！



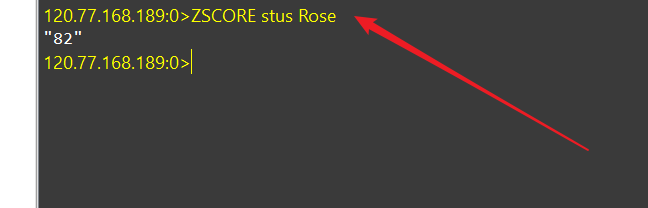
##### ZREM key member：移除sorted set中的一个指定元素。



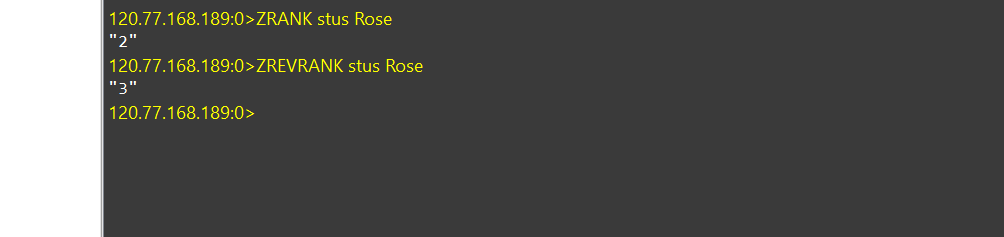
Tom不见了！



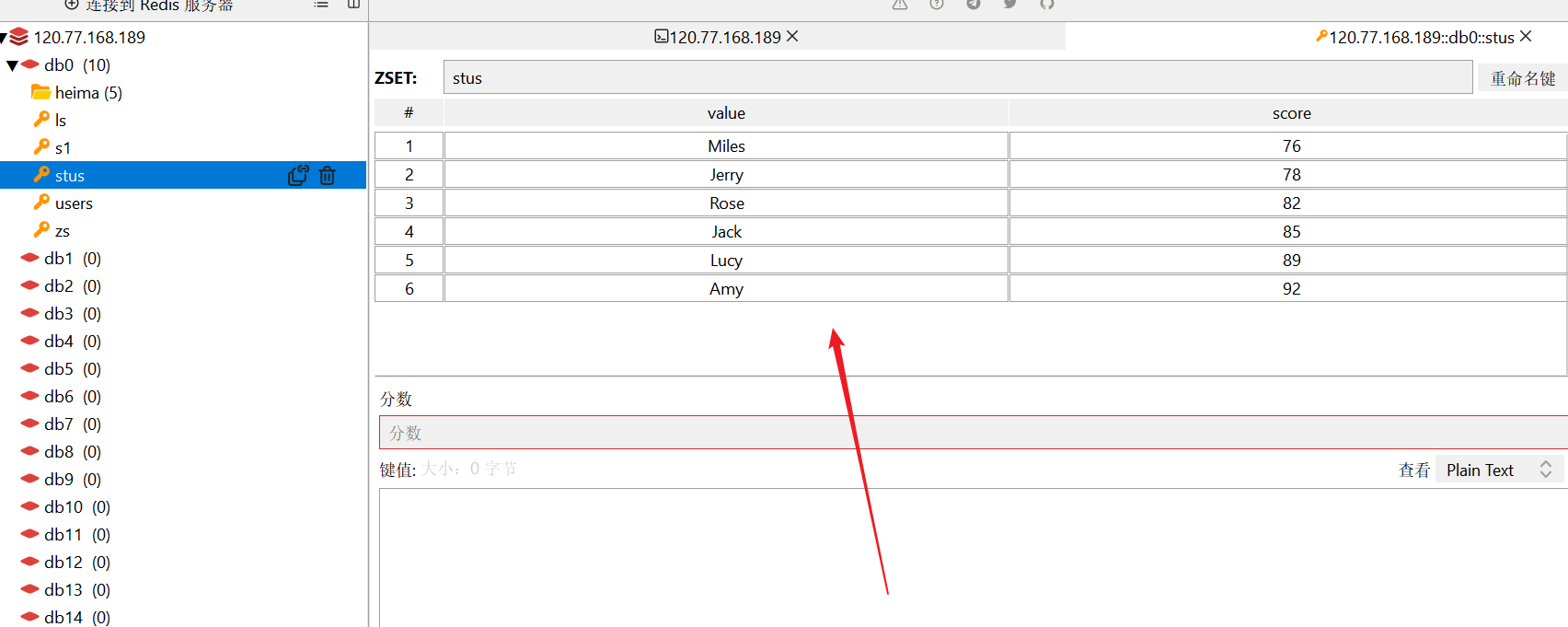
##### ZSCORE key member：获取sorted set中的指定元素的score值。



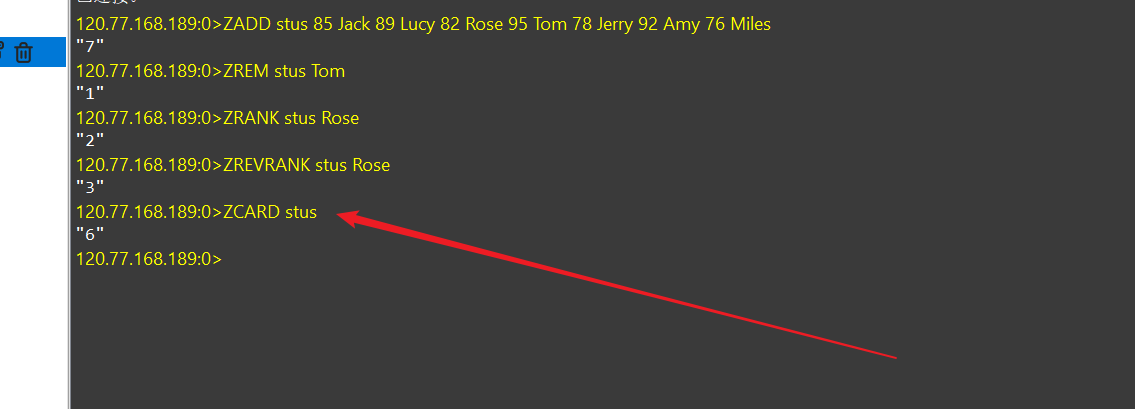
##### ZRANK key member：获取sorted set中的指定元素的排名。



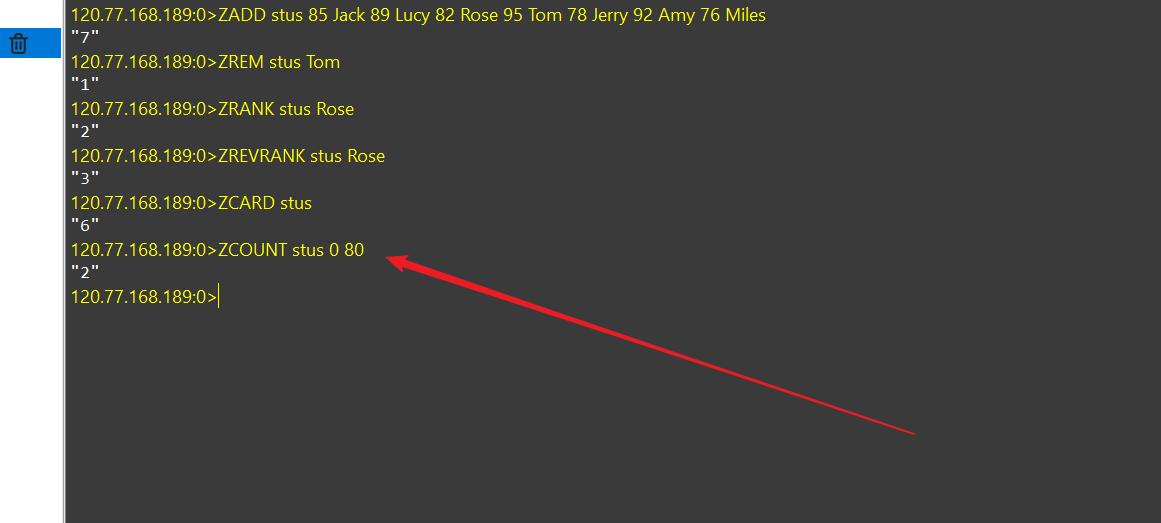
从0开始！



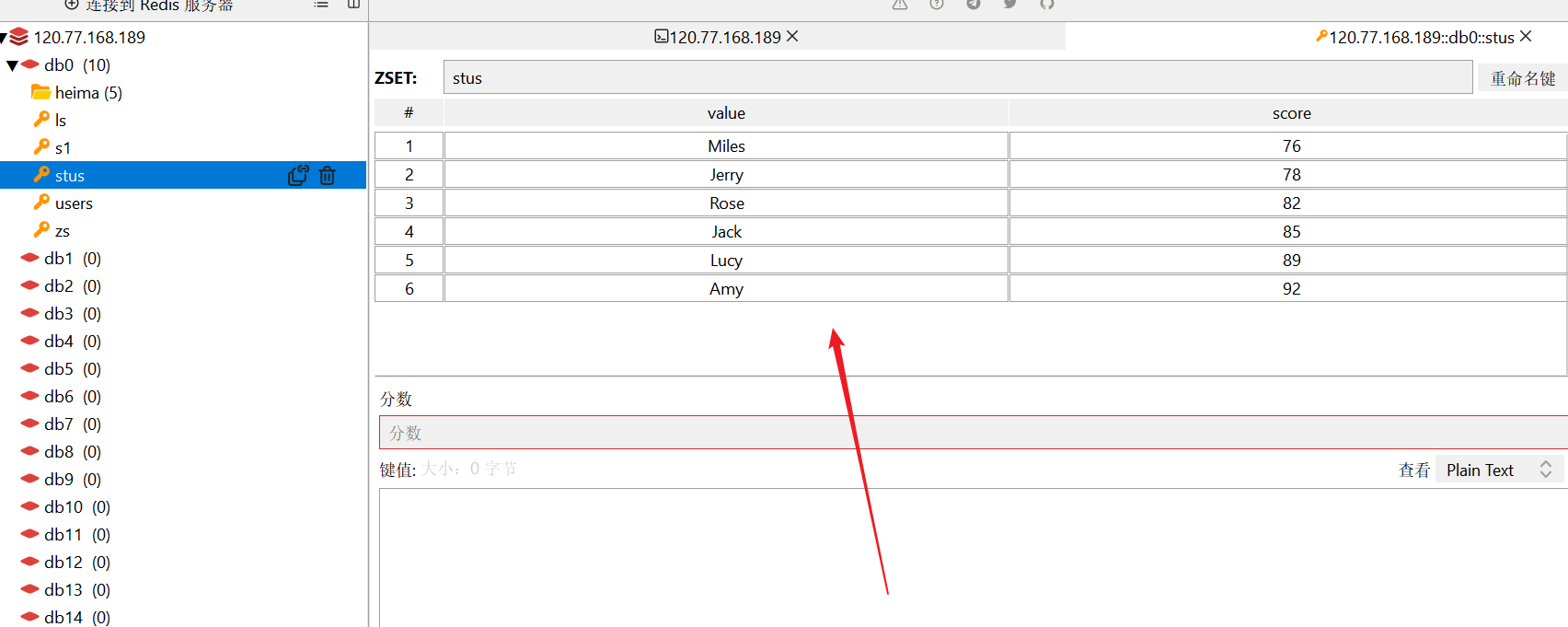
##### ZCARD key：返回sorted set中元素的个数。



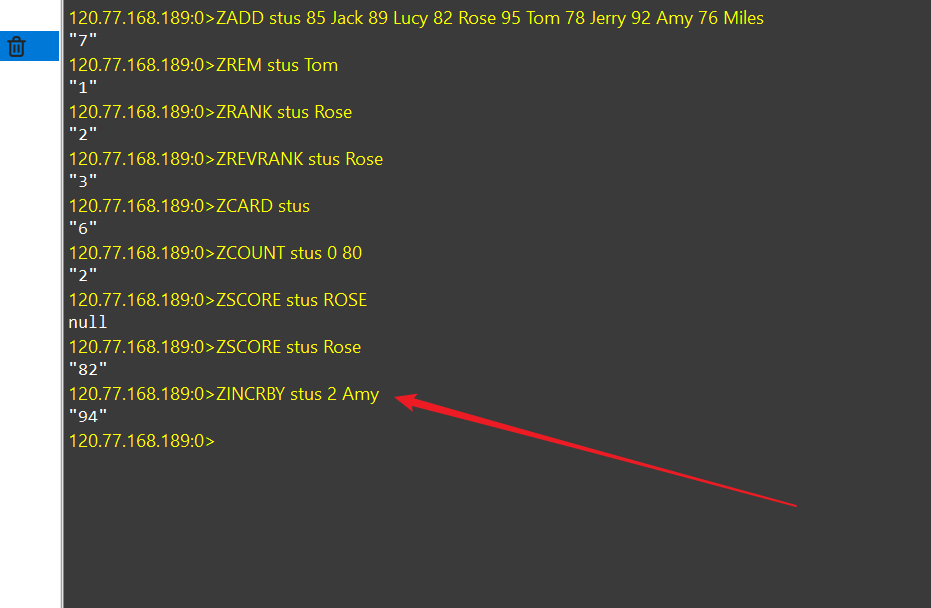
##### ZCOUNT key min max：统计sorted set中score值在给定范围内的所有元素的个数。



可以看到score值在0到80分的确实有两个：

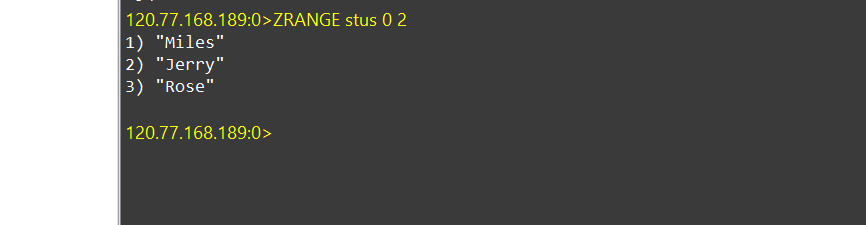


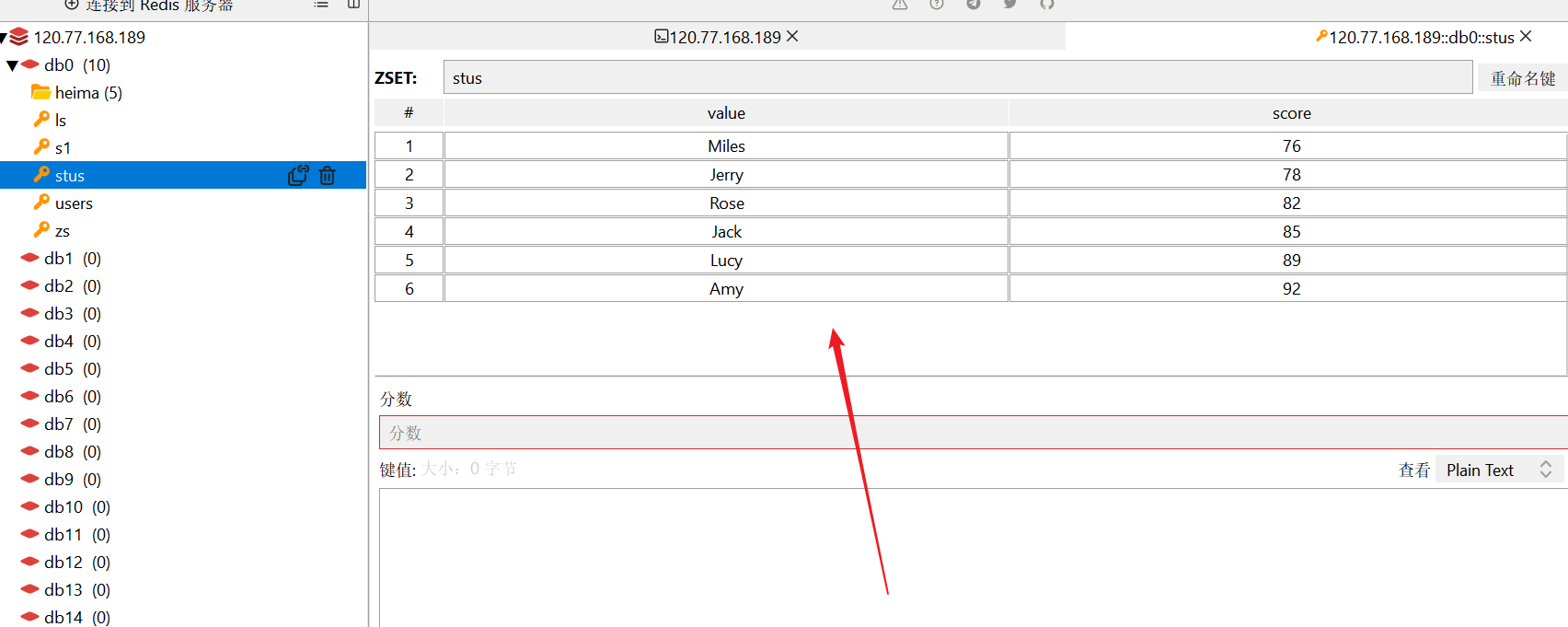
##### ZINCRBY key increment meMber：让sorted set的指定元素自增，步长为指定的increment值。



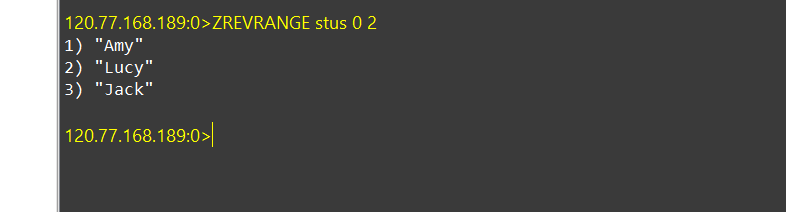
##### ZRANGE key min max：按照score排序后，获取指定排名范围内的元素。

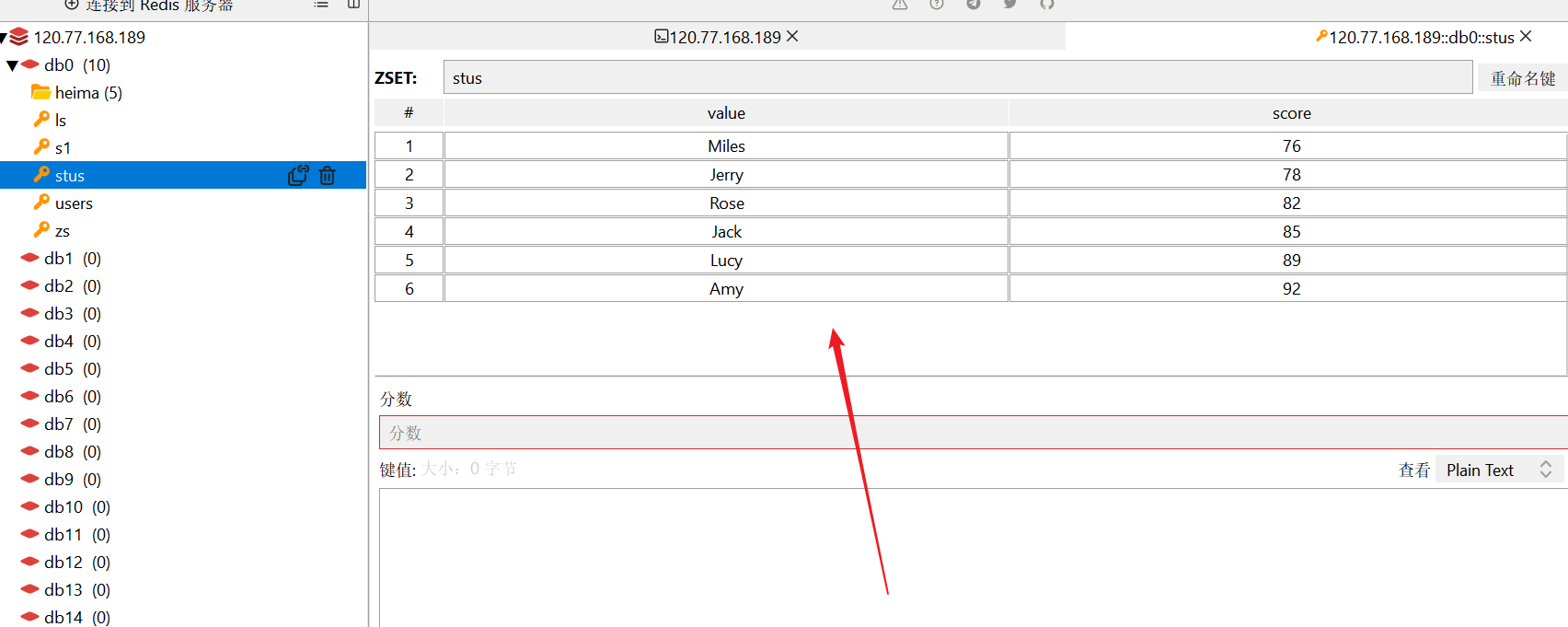
获取排序前三名！





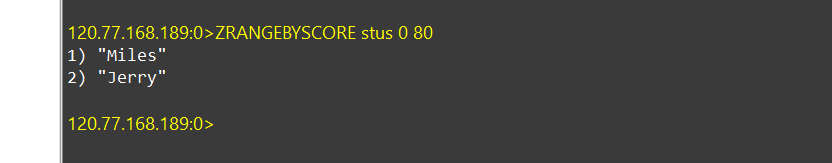
获取排序后三名！

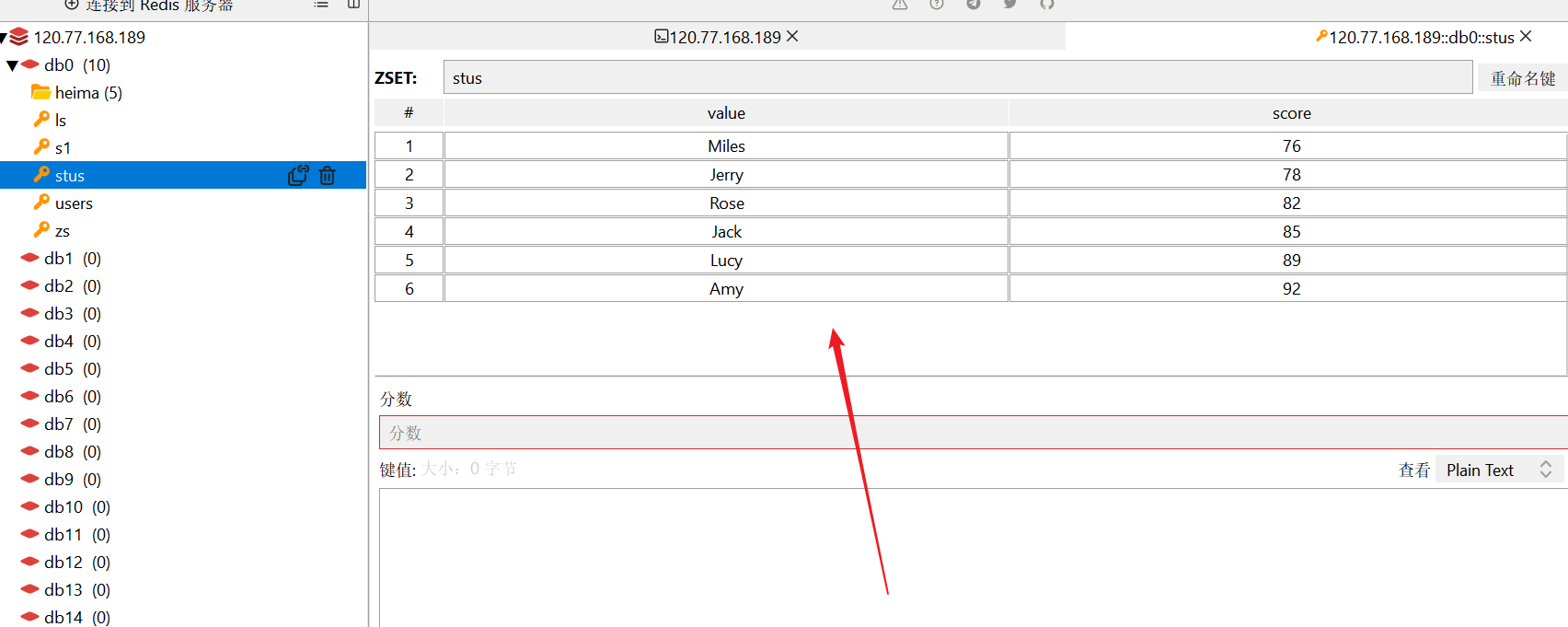




##### ZRANGEBYSCORE key min max：按照score排序后，获取指定score范围内的元素。

获取score在0到80范围内的元素：





#### 多个集合的操作：

ZDIFF、ZINTER、ZUNION：求差集、交集、并集