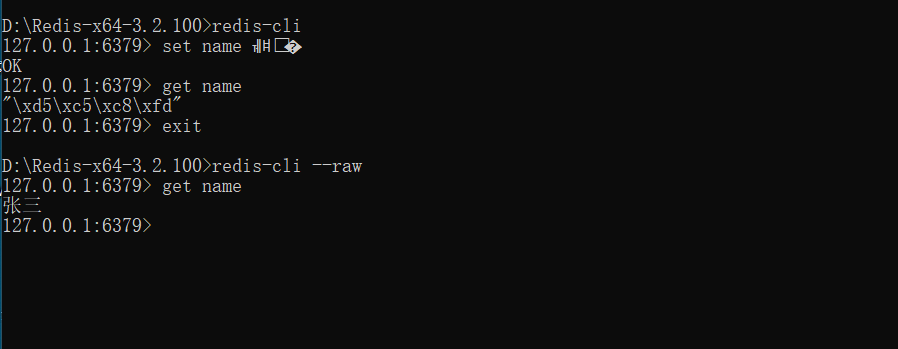
# Redis数据类型简介

1. Redis支持5种数据类型：String（字符串）、Hash（哈希）、List（列表）、Set（集合）以及SortedSet（有序集合）。
2. Redis命令不区分大小写。
3. 在获取键值对中的值时，如果值是中文的，则会返回编码后的字符串。如果你希望返回的是中文，那么在客户端使用命令连接服务器时加上参数--raw；如下：

**redis-cli - -raw**

示例如下：



（关于设置中文时也出现乱码的问题有待解决）

# String类型及命令

1. String（字符串）类型是Redis中最基本的数据类型，它是二进制安全的，任何形式的字符串都可以存储，包括二进制数据、序列化后的数据、JSON化的对象，甚至是一张Base64编码后的图片；
2. String（字符串）类型的键最大能存储512MB的数据

## 设置键值对

### set

set命令用于设置一个键值对，语法如下：

**set key value [ex seconds] [px milliseconds] [nx|xx]**

使用set命令将字符串值value设置到key中；如果key已存在，将会覆盖掉旧值，并且忽略类型；针对某个带生存时间的key来说，当set命令成功执行时，这个key上的生存周期也会被清空。

set命令可选参数如下：

1. ex seconds：用于设置key的过期时间为多少秒(seconds)；

**set key value ex seconds**

等同于**setex key seconds value**

1. px milliseconds：用于设置key的过期时间为多少毫秒(millseconds)；

**set key value px milliseconds**

等同于**psetex key milliseconds value**

1. nx：表示当key不存在时才对key进行设置操作；

**set key value nx**

1. xx：表示当key存在时才对key进行设置操作；

**set key value xx**

set命令的返回值：当set命令设置成功，则会返回OK；如果设置了NX或XX，因条件不足而设置失败，则会返回空批量回复（NULL Bulk Reply）

### mset

mset命令可以同时设置多个键值对，语法如下：  
 **mset key value [key value …]**

mset命令设置多个键值对时，如果某个key已经存在，那么mset命令会用新值覆盖旧值；

mset命令具有原子性，所有给定的key都会在同一时间内被设置更新，不存在某些key被更新了而另一些key没有被更新的情况；

返回值：总是返回OK，因为mset命令不可能设置失败。

### setnx

setnx命令set if not exists的缩写，用于设置不存在的键值对，语法如下：

**setnx key value**

如果key不存在，则设置值，返回1；如果key存在，则该条setnx命令无效，返回0。

### msetnx

msetnx命令用于设置多个不存在的键值对，语法如下：

**msetnx key value [key value…]**

使用mesetnx设置多个不存在的键值对时，当且仅当所有给定的key都不存在时才会设置成功；如果有一个key存在，则该条命令失效。

它也具有原子性，即要么全部被设置，要么全部设置失败。

返回值：所有key设置成功后返回1；否则返回0。

## 获取键的值

### get

get命令可以获取指定key对应的字符串值，语法如下：

**get key**

当key存在时，返回key对应的值；如果key对应的值不是字符串类型的，则返回一个错误；

如果key不存在，则返回nil或空；

### mget

mget命令可以同时获取多个给定key对应的值，key之间使用空格隔开，语法如下：  
 **mset key [key…]**

如果给定的key中有不存在的key，那么这个key对应的返回值为nil或空；

返回：以行的形式显示每一个key对应的值，key不存在则该key对于的值为nil或空

### getrange

如果key对应的值是字符串类型，则可以使用getrange命令通过字符串截取的方式来获取值的某一部分。语法如下：

**getrange key start end**

start表示开始坐标，end表示结束坐标；

start和end参数的值可以是正整数，也可以是负整数；当为负整数时，表示从字符串最后开始计数，如-1表示最后一个字符串，-2表示倒数第二个字符串，以此类推；

返回值：返回截取的字符串

## 修改键的值

### setrange

setrange命令可以从指定的位置开始将key的值替换为新的字符串，语法如下：

**setrange key offset value**

offset表示开始下标，最大值为536870911（因为Redis字符串的大小被限制在512MB以内）；

如果key不存在，则当作空白字符串处理，创建该key， key对应的值就是value；

如果给定key原来存储的字符串长度比偏移量小（比如，原字符长度为4，而设置的offset为9），那么原字符和偏移量之间的空白将用零字节（Zerobytes，”\x00”）来填充；

返回值：返回替换成功后值的长度。

### getset

getset命令用于修改key的值，并返回key的旧值，语法如下：

**getset key value**

如果key存在，并且值是字符串类型的，则返回该key的旧值；

如果key存在，但值不是字符串类型，则返回错误；

如果key不存在，则创建key并设置值，执行结果返回nil；

### append

append命令用于为键追加值，语法如下：

**append key value**

如果key存在，并且值是字符串类型的，则将value值追加到key旧值的末尾；

如果key不存在，则创建key并设置值；

返回值：返回追加value之后，值的总长度；

## 键的偏移量

### setbit

setbit命令可以设置键的偏移量，语法如下：

**setbit key offset value**

使用setbit命令对key所存储的字符串值进行设置指定偏移量上的位或清除指定偏移上的位的操作；

value参数决定了位的设置或清除，value值取0或1；

当key不存在时，自动生成一个新的字符串值，这个字符串是动态的，它可以扩展，以确保将value保存到指定的偏移量上；当这个字符串串扩展时，使用0来填充空白位置；

offset参数必须是大于或等于0，并且小于2^32 (bit映射被限制在512MB之内) 的正整数；

默认情况下，bit初始化为0；

返回值：返回指定偏移量原来存储的位

### getbit

getbit命令可以获取键的偏移量，语法如下：

**getbit key offset**

对key所存储的字符串值，使用getbit命令来获取指定偏移量上的位(bit)；

返回值：当offset值超过了字符串的最大长度或者key不存在时，返回0；否则返回字符串指定偏移量上的位(bit)

## 键值对的计算

### bitcount

bitcount命令用于计算key的值（字符串类型的）中被设置为1的比特位数量，如下：

**bitcount key [start] [end]**

start和end为可选参数，如果不设置，则表示对整个字符串进行计数，如果设置，则可以对指定下标的子字符串进行计数；start和end取值都是整数值，也可以取负数，比如，-1表示字符串的最后一个字节，-2表示字符串的倒数第二个字节，以此类推；

如果key不存在，就会被当做空字符串来处理，其计数结果为0；

返回值：返回被设置为1的位的数量；

### bitop

bittop命令可以对key保存的字符串值进行元运算，语法如下：

**bittop operation destkey key [key..]**

destkey用于保存运算结果；

operation表示位元操作符，可以是AND、OR、NOT、XOR任意一种；

·bittop and destkey key [key…]：表示对一个或多个key求逻辑并，并将结果保存到destkey中

·bittop or destkey key [key…]：表示对一个或多个key求逻辑或，并将结果保存到destkey中；

·bittop not destkey key：表示对给定的key求逻辑非，并将结果保存到destkey中；

·bittop xor destkey key [key…]：表示对一个或多个key求逻辑异或，并将结果保存到destkey中；

除了not元运算符外，其余的运算符都可以接收一个或多个key作为参数；

当使用bittop命令进行不同长度的字符串的位元运算时，较短的那个字符串所缺少的部分将会被当作0；

不存在的key也会被看作包含0的字符串序列

返回值：返回保存在destkey中的字符串长度，这个长度和输入key中最长的字符串的长度相等。

### strlen

strlen命令用于统计键对应的值的字符长度，语法如下：  
 **strlen key**

如果key的值不是字符串类型，则返回一个错误；

如果key不存在，则返回0；

## 键的生存时间

在设置键值对时，可以通过可选参数来设置其生存时间，以秒或毫秒为单位；当键值对的生存时间结束，Redis将自动删除该键值对。

### setex

setex命令以秒为单位，设置键值对的生存时间，语法如下：

**setex key seconds value**

key表示key名称，如果key不存在，则创建

seconds表示秒数；

value表示要给key设置的值；

返回：设置成功返回OK，否则就是设置失败。

### ttl

ttl命令可以查询某个key的生存时间还有多少秒，语法如下：

**ttl key**

结果返回一个正整数，以秒为单位。如果返回一个负数，则表示该key不存在或已过期。

### psetex

psetex命令以毫秒为单位，设置key的生存时间，语法如下：  
 **psetex key milliseconds value**

key表示键名称，如果键不存在，则创建；

milliseconds表示毫秒；

value表示要给key设置的值；

返回：设置成功返回OK，否则表示设置失败；

### pttl

pttl命令用于查看一个key的生存时间还有多少毫秒，语法如下：  
 **pttl key**

结果返回一个正整数，以毫秒为单位。

## 值的增减

### decr

decr命令用于将key中存储的数字值减1，语法如下：  
 **decr key**

如果key不存在，则key的值先被初始化为0，在执行减1操作；

如果key存在，但对应的值不能表示为整数数字，则返回一个错误；

返回值：返回减1之后的值；

### decrby

decrby命令用于将key存储的数字值减去减量值，语法如下：

**decrby key decrement**

decrement表示要减去的值；

如果key不存在，则key的值先被初始化为0，再执行减法操作；

如果key存在，但对应的值不能表示为整数数字，则返回一个错误；

decrby操作的数值限制在64位(bit)有符号数字表示范围之内；

返回值：返回减去减量值key的新值；

### incr

incr命令用于将key存储的数值加1，语法如下：

**incr key**

如果key不存在，则key的值先被初始化为0，在执行加1操作；

如果key存在，但不能表示为整数数字，则返回一个错误；

返回值：返回执行加1操作后的key的新值；

### incrby

incrby命令用于将key所存储的值加上一个增量值，语法如下：

**incrby key increment**

参数increment表示增量；

如果key不存在，则创建key并将值设置为0，然后再执行incrby命令；

如果key的值不能表示为整数数字，则返回一个错误；

如果执行成功，则返回加上增量值后key对应的新值。

### incrbyfloat

Incrbyfloat命令可以将key所存储的值加上一个浮点数增量值，语法如下：

**incrbyfloat key increment**

increment为增量值；

如果key不存在，则创建key并将值设置为0，然后再执行incrbyfloat命令；

如果key的值不能表示为数字，则返回一个错误；

如果incrbyfloat命令执行成功，则返回加上增量值后的值；

注意：小数末尾的0会被忽略；并且有特定需要时，浮点数也会被转化为整数，如2.0转换为2；在执行incrbyfloat命令后，无论产生的浮点数的实际精度有多长，计算结果最多也只能保留小数点后15位。

# Hash类型及命令

1. Redis的Hash类型是一个String类型的域（field）和值（value）的映射表，可以将哈希表看作为一个对象；

2）Hash类型常常用来存储对象信息；

3）在Redis中，每个哈希表可以存储2^32-1个键值对，也就是40多亿个数据。

## 设置哈希表中域的值

### hset

hset命令用于设置一个哈希表key（对象）的域（属性）和值，语法如下：

**hset key field value**

key表示哈希表名，可以理解为对象名；

field表示域，可以理解为对象的属性；

value表示值，可以理解为对象属性的值；

如果key不存在，则创建；

如果field已经存在，那么新值将覆盖旧值

返回值：如果field是新建域，并且hset命令成功，则返回1；如果哈希表中已经存在field，那么新值将覆盖旧值，返回0；

### hsetnx

hsetnx命令可以设置为不存在的域值，语法如下：

**hsetnx key field value**

key表示哈希表名，可以理解为对象名；

field表示域，可以理解为该对象的属性名；

value表示值，可以理解为属性的值；

如果哈希表不存在，则创建哈希表，再执行该命令；

如果哈希表中的域不存在，则设置该哈希表的域和值；

如果哈希表中的域已存在，则该条命令执行无效；

返回：设置成功则返回1，如果field已存在，则设置失败，返回0。

### hmset

hmset命令可以为一个哈希表设置多个域值对，语法如下：

**hmset key field value [field value ……]**

key表示哈希表名，可以理解为对象名；

field表示域，可以理解为属性名；

value表示值，可以理解为属性值；

如果哈希表key不存在，则创建；

如果域已存在，则新值覆盖旧值；

返回值：执行成功返回OK，否则表示执行失败。

## 获取哈希表中的域值

### hget

hget命令可以获取哈希表中域的值，语法如下：

**hset key field**

field表示域，可以理解为属性名；

如果哈希表key不存在，或者域field不存在，则返回nil，否则返回域对应的值。

### hgetall

hgetall命令用于获取哈希表中所有的域和值，语法如下：  
  **hgetall key**

key表示哈希表名，可以理解为对象名；

返回值：以行的形式返回哈希表中的域field和值value，如果哈希表不存在，则返回空行。

### hmget

hmget命令可以一次性获取哈希表中一个或多个域的值，语法如下：  
 **hmget key field [field……]**

key表示哈希表名，可以理解为对象名；

field表示域，可以理解为属性名；

返回值：以行的形式返回，行的顺序与给定域的排序顺序一致；如果某个域fileld不存在，则该域field的值返回nil；如果哈希表不存在，则每个域都返回nil。

### hkeys

hkeys命令可以获取哈希表中所有的域，不包括值，语法如下：

**hkeys key**

key表示哈希表，可以理解为对象；

返回值：以行的形式返回哈希表中所有的域；如果哈希表不存在，则提示为空。

### hvals

hvals命令用于获取哈希表中所有域的值，语法如下：

**hval key**

key表示哈希表名，可以理解为对象名；

返回：以行的形式返回所有的值；如果哈希表不存在，则提示为空。

## 哈希表的统计

### hlen

hlen命令可以得到哈希表中域的数量，语法如下：

**hlen key**

key表示哈希表名，可以理解为对象名；

返回：返回域的数量（一个整数值）；如果哈希表不存在，返回0。

### hstrlen

hstrlen命令可以知道哈希表中域的值的长度，语法如下：

**strlen key field**

key表示哈希表名，可以理解为对象名；

field表示域，可以理解为属性名；

返回：返回一个整数，表示值的长度；如果哈希表或域不存在，则返回0。

## 值的加法操作

### hincrby

hincrby命令用于给域的值加上整数增量值，语法如下：

**hincrby key field increment**

key表示哈希表，可以理解为对象；

field表示域，可以理解为属性；

increment表示增量，是一个不分正负的整数；当为负数时，可以实现减法操作；

如果哈希表不存在，则先创建它，再执行hincrby命令；

如果域不存在，则先创建域并初始化值为0，再执行hincrby命令；

如果域的值不能表示为整数数字，则返回一个错误；

如果执行成功，返回执行操作后域的新值；

### hincrbyfloat

hincrbyfloat命令用于给域的值加上浮点数增量值，语法如下：

**hincrbyfloat key field increment**

key表示哈希表，可以理解为对象；

field表示域；可以理解为对象的属性；

increment表示浮点数增量值；

如果哈希表不存在，则先创建哈希表，再执行hincrbyfloat命令；

如果域不存在，则先创建域并初始化值为0，再执行hincrbyfloat命令；

如果域的值不能表示为数字，则报错；

如果执行成功，则返回执行操作后域的新值；

## 域的删除与判断

### hdel

hdel命令可以删除哈希表中的一个或多个域，语法如下：

**hdel key field [field……]**

key表示哈希表名，可以理解为对象名；

field表示域，可以理解为属性名；

如果删除的field不存在，则忽略；

如果哈希表不存在，则返回0；

返回值：返回成功删除的域的数量，不包括被忽略的。

### hexists

hexists命令可以判断某个域是否存在，语法如下：

**hexists key field**

key表示哈希表，可以理解为对象；

field表示域，可以理解为属性；

返回值：如果这个域field存在，则返回1；如果域不存在或哈希表也不存在，则返回0。

# List类型及命令

1. Redis中列表类型可以被看作是一个字符串列表，列表中的每一个元素都是字符串类型；
2. 在操作Redis中的列表时，可以将一个元素插入到这个列表的头部或尾部；
3. 一个列表大约可以存储2^32-1个元素。

## 向列表中插入值

### lpush

lpush命令可以将一个或多个value值插入到列表的头部（类似于入栈操作），语法如下：

**lpush key value [value…]**

key表示列表名；

value表示要插入到列表中的值；

如果列表不存在时，则创建一个列表，再执行lpush命令；

如果指定的列表名不是列表类型，则返回一个错误；

返回值：成功执行后，返回该列表的长度（列表中元素的个数）。

### rpush

rpush命令用于将一个或多个value插入值到列表的尾部，语法如下：

rpush key value [value…]

key表示列表名；

value表示要向列表中插入的值；

如果列表不存在，则创建一个列表，再执行rpush命令；

如果指定的列表名不是列表数据类型，则返回一个错误；

返回值：成功执行后，返回该列表的长度（列表中元素的个数）。

### linsert

linesert命令可以在列表中某个值的前面或后面插入一个值，语法为：

**linsert key before|after privot value**

key表示列表；

before或after表示插入到前面还是后面；

privot为指定的元素；

value表示要插入的值；

如果指定的值privot不存在，该命令无效，返回-1；

如果列表key不存在，该命令无效，返回0；

如果列表key不是列表类型，返回一个错误；

返回值：执行成功，返回列表的长度（列表的元素个数）；

### lpushx

lpushx命令可以在列表头部插入一个值，但前提是列表已存在，并且是列表类型，语法如下：

**lpushx key value**

key表示列表名；

value表示要插入的值；

如果指定的列表key不存在，则返回0；

如果指定的列表key不是列表类型，则返回一个错误；

返回值：执行成功后，返回列表的长度（列表的元素个数）。

### rpushx

rpushx命令可以在列表尾部插入插入一个值，前提是列表已存在，并且是列表类型，语法如下：

**rpushx key value**

key表示列表名；

value表示要插入的值；

如果指定的列表key不存在，则返回0；

如果指定的列表key不是列表类型，则返回一个错误；

返回值：执行成功后，返回列表的长度（列表的元素个数）。

### lset

lset命令可以修改列表中指定下标的值，如下：

**lset key index value**

key表示列表名；

index表示下标，从0开始；

value表示要修改的值；

如果指定的列表key不存在，则返回错误信息；

如果指定的列表不是列表类型，则返回错误信息；

如果指定的下标index超出列表范围，则返回错误信息；

返回值：成功执行后，返回OK。

## 获取列表值

### llen

llen命令可以得到列表的长度，语法如下：  
 **llen key**

key表示列表名；

如果指定的列表key不存在，则返回0；

如果指定的列表key不是列表类型，则返回错误信息；

返回值：成功执行后，返回列表的长度（列表中元素的个数）。

### lindex

lindex命令可以获取列表中指定下标的值，语法如下：

**lindex key index**

key表示列表名；

index表示下标，从0开始；也可以为负数，如-1表示最后一个元素值，以此类推；

如果指定的列表key不是列表类型，则返回错误信息；

如果指定的列表key不存在，则返回nil；

返回值：返回指定下标的值；

### lrange

lrange命令用于获取列表中指定范围内的值，语法如下：

**lrange key start end**

key表示列表名；

start表示开始下标；可以为负数，如-1表示最后一个元素；

end表示结束下标；可以为负数，如-1表示最后一个元素；

如果指定的列表key不存在或者start参数值大于列表的长度，则返回空列表；

如果指定的列表key不是列表类型，则返回错误信息；

返回值：成功执行后，以行的形式返回列表中指定范围中的值，每个值占一行。

## 删除列表元素

### lpop

lpop命令可以删除列表的头元素并得到它的值，语法如下：

**lpop key**

key表示列表名；

如果指定的列表key不存在，则返回nil；

如果指定的列表key不是列表类型，则返回错误；

返回值：返回被删除的值。

### rpop

ropo命令可以删除列表的尾元素并得到它的值，语法如下：

**rpop key**

key表示列表名；

如果指定的列表key不存在，则返回nil；

如果指定的列表key不是列表类型，则返回错误；

返回值：返回被删除的值；

### blpop

blpop命令可以在指定的时间内删除列表的头元素，语法如下：

**blpop key [key…] timeout**

key表示列表，可以有多个；

timeout表示时间，以秒为单位；

执行该命令会有两种情况：阻塞行为和非阻塞行为；

当给定的列表key都不存在时，进入阻塞状态，直到等待超时（等待时间超过了指定时间），阻塞状态结束，并返回nil与等待时间；或者是另一个客户端使用了lpush或rpush命令向给定的列表key添加了元素，阻塞状态结束，并删除另一个客户端添加的头元素，返回列表名和它被删除的头元素；

当给定的列表key中但凡有一个列表存在，则删除其头元素，并返回该列表名与被删除的元素；

如果给定的列表key都存在或存在多个，则只删除第一个列表的头元素，并返回该列表名与被删除的头元素。

### brpop

brpop命令可以在指定的时间内删除列表的尾元素，语法如下：

**brpop key [key…] timeout**

key表示列表名；

timeout表示时间，以秒为单位；

执行该命令会有两种情况：阻塞行为和非阻塞行为；

当给定的列表key都不存在时，进行阻塞状态，直到等待超时（等待时间超过了指定时间），阻塞状态结束，并返回nil和等待时间；或者是另一个客户端使用了lpush或rpush命令向给定的列表key添加了元素，阻塞状态结束，并删除另一个客户端添加的尾元素，返回列表名和它被删除的尾元素；

当给定的列表key中但凡有一个列表存在，则删除其尾元素，并返回该列表名与被删除的尾元素；

如果给定的列表key都存在或存在多个，则只删除第一个列表的尾元素，并返回该列表名与被删除的尾元素；

### lrem

lrem命令可以删除列表中与指定值相同的值，并可以指定要删除的数量，语法如下：

**lrem key count value**

key表示列表名；

value表示指定的值；

count表示要删除的个数；当count为0时，删除列表中所有与value相等的值；当count大于0时，表示从表头向表尾搜索，删除count个与value相等的值；当count小于0时，表示从表尾向表头搜索，删除count绝对值个与value相等的值；

如果指定的列表key不是列表类型，则返回错误信息；

如果指定的列表key不存在，则返回0；

返回值：返回被删除的值的数量。

### ltrim

ltrim命令可以删除列表中指定范围外的值，也就是只保留列表中指定范围内的值，语法如下：

**ltrim key start stop**

key表示列表名；

start表示开始下标；

stop表示结束下标；

如果指定的列表key不是列表类型，则返回错误信息；

如果指定的列表key不存在，则返回空列表；

如果stop值超过列表长度，则返回OK，列表不会产生影响；

如果start值和stop值都超过列表长度，则清空列表；返回OK；

## 移动列表元素

### rpoplpush

rpoplpush命令可以将列表中的尾部元素移动到另一个列表中的表头，语法如下：

**rpoplpush source destination**

source表示列表源，也就是要移动尾元素的列表；

destination表示目标列表，也就是要接收元素到表头的列表；

如果指定的列表源不存在，则返回nil；

如果列表源与目标列表是同一个列表，则表示将表尾元素移动到表头，并返回该元素；

如果目标列表不存在，则创建空的目标列表，再执行rpoplpush命令，并返回该元素；

返回值：返回被移动的元素；

### brpoplpush

该命令用于在指定的时间内移动列表元素到另一列表中。

# Set类型及命令

1. Redis的Set类型是String类型值的集合；集合无序并且不能存在重复元素，每一元素都是唯一的；
2. 集合是通过哈希表来实现的，所以使用集合进行增加、删除、查询操作时的效率特别高；
3. 一个集合所能存储的最大容量是2^32-1个元素；

## 向集合中添加元素

### sadd

sadd命令用于向集合中添加一个或多个元素，语法如下：

**sadd key member [member……]**

key表示集合名；

member表示要添加的元素；

如果指定的集合key不存在，则创建一个空的集合，再执行sadd命令；

如果添加的元素在集合中已存在，则该元素会被忽略；

如果指定的集合key不是集合类型，则返回错误信息；

返回值：执行成功后，返回成功添加到集合中的元素个数（不包括被忽略的）；

### smove

smove命令用于将集合中某个元素移动到另个集合中，语法如下：

**smove source destination member**

source表示集合源；

member表示集合源中的某一个元素，也就是要移动的元素；

destination表示目标集合，也就是接收元素的集合；

如果指定的集合源不存在或元素不存在，则返回0；

如果指定的目标集合不存在，则创建一个空的集合，再执行smove命令；

如果指定的集合源或者目标集合不是集合类型，则返回错误信息；

返回值：返回1表示执行成功，返回0表示集合源或元素不存在；

### sunionstore

sunionstore命令用于将一个或多个集合中的全部元素复制出来组成一个新的集合，语法如下：

**sunionstore destination key [key……]**

destination表示新集合的名称；

key表示要复制的集合，可以有多个；

如果指定的某个集合key不是集合类型，则返回错误信息；

如果指定的新集合已存在，则向新集合中追加内容；

返回值：成功执行后，返回新集合中元素的个数。

## 获取集合元素

### sismember

sismember命令可以判断某个元素是否在集合中，语法如下：

**sismember key number**

key表示集合名；

number表示元素值；

返回值：如果集合中存在该元素，则返回1。

### scard

scard命令用于获取集合中元素的数量，语法如下：

**scard key**

key表示集合名；

返回值：返回集合中元素的个数；集合不存在时，返回0。

### smembers

smembers命令用于获取集合中的所有元素，语法如下：

**smembers key**

key表示集合名；

返回值：返回集合中的所有元素；集合不存在时返回空列表。

### srandmember

srandmember命令用于随机获取集合中的一个或指定个数个元素，语法如下：

**srandmember key [count]**

key表示集合名；

count表示指定个数；当为正数且小于集合元素个数时，返回随机count个元素；当count大于等于集合元素个数时，返回集合的所有元素；当count为负数时，返回count个元可能重复多次的元素；

返回值：

如果只设置了key参数，则返回一个随机的元素；此时如果集合为空，则返回nil。

如果设置了count参数，则返回count个元素；此时如果集合为空，则返回一个空列表。

### sunion

sunion命令用于获取一个或多个集合中的所有元素，语法如下：

**sunion key [key……]**

key表示集合名；

返回值：返回指定集合们的全部元素的并集，会忽略掉多余重复的元素。

## 集合的并集与差集

### sdiff

sdiff命令用于获取某个集合的与其他集合们的差集，语法格式如下：

**sdiff key [key…..]**

key表示集合名，可以是多个；

如果只指定了一个集合，则返回该集合的所有元素；

如果指定了多个集合，则返回第一个集合中的其他集合没有的元素；

### sdiffstore

sdiffstore命令用于将某个指定的集合与其他集合们的差集组成一个新的集合，语法如下：

**disffstore destination key [key……]**

destination表示新集合名；

key表示真正要操作的集合；

如果只指定了一个集合key，则将该集合的所有元素复制到新集合destination中；

如果只指定了多个集合key，则将第一个集合key中的其他集合没有的元素（也就是差集）复制到新集合destination中；

如果指定的新集合已存在，则清空新集合中的内容，再执行disffstore命令；如果不存在，则创建；

返回值：返回新集合中元素的个数（差集的个数）

### sinter

sinter命令用于获取一个或多个集合的并集（相同的元素），如下：

**sinter key [key……]**

key表示集合名；

如果指定的集合key不存在，则视它为空集合；

返回值：如果没有交集，则返回空列表，否则返回它们的交集（相同元素）。

### sinterstore

sinterstore命令用于将一个或多个集合的并集（相同的元素）存入一个新的集合中，如下：

**sinterstore destination key [key……]**

destination表示新集合名；

key表示集合名；

如果只指定了一个集合key，则将该集合中的全部元素复制一份到新集合中；

如果新集合已存在，则清空所有元素，再执行sinterstore命令；

返回值：返回新集合中元素的个数（并集的个数）；

## 删除集合元素

### spop

spop命令用于随机删除集合中一个或指定个数个元素，语法如下：

**spop key [count]**

key表示集合名；

count表示指定的个数，可选，如不指定，则随机删除一个；

返回值：返回被删除的随机元素；如果集合不存在，返回nil。

### srem

srem命令用于删除集合中一个或多个指定的元素，语法如下：

**srem key member [member…]**

key表示集合名；

member表示要删除的元素，可以有多个；

如果给定的多个元素存在重复的，则会忽略掉重复的元素，如果集合中不存在给定的元素，则该元素也会被忽略；

返回值：返回成功删除的元素个数（不包括被忽略掉的）。

# SortedSet类型及命令

1. 有序集合类型也是String类型的集合，有序集合中不存在重复的元素，每个集合元素都有一个对应的double类型的小数，Redis通过元素对应的小数来为集合进行从小到的排序，有序集合中的元素是唯一的，但元素对应的double小数不是唯一的，它可以重复。
2. 有序集合采用哈希表实现，当面对增加、删除、查询操作时，效率特别高。
3. 有序集合中所能存储的最大元素数量是2^32-1个。

## 添加元素到有序集合中

### zadd

zadd命令用于向有序集合中添加一个或多个元素及其对应的小数值，语法如下：

**zadd key score member [ [score member] ……]**

key表示有序集合名；

score表示小数值，可以是double类型的浮点数，也可以是整数；

member表示要插入的元素；

如果有序集合key不存在，则创建一个新的空有序集合，再执行zadd命令；

如果有序集合中已经存在要插入的元素，则只会更新它的小数值；

返回值：返回成功添加元素的数量，不包括已存在的元素；

### zincrby

zincrby命令用于将有序集合中元素对应的小数值进行增量操作，语法如下：

**zincrby key increment member**

key表示有序集合名；

increment表示增量值，可正可负，为负数时可实现减量操作；

member表示元素；

如果有序集合key不存在，或者元素member不存在，则该命令等同于zadd key increment member，即等同于zadd命令；

返回值：返回进行增量操作后，更新的小数数值；

## 取有序集合的元素

### zcard

zcard命令用于获取有序集合中元素的数量，语法如下：

**zcard key**

key表示有序集合；

返回值：返回有序集合的元素数量；有序集合不存在则返回0。

### zcount

zcount命令用于获取有序集合中元素的小数值在指定区间的元素数量，语法如下：

**zcount key min max**

key表示有序集合

min表示开始区间（包括）；

max表示结束区间（包括）；

返回值：返回小数值在指定区间的元素数量；

### zlexcount

zlexcount命令用于获取有序集合中，在指定区间内的元素数量，语法如下：

**zexxcount key min max**

key表示有序集合

min和max是一个区间，使用“（”或“[”表示，其中“（”表示开区间，指定的值不会包含在范围之内；“[”表示闭区间，指定的值会被包含在范围之内。

min和max的值除了是区间，还可以是-（负无穷）或+（正无穷），如zelcount z1 - +表示获取z1有序集合中所有元素的数量。

返回值：返回有序集合中在指定区间内的元素数量

### zrange

zrange命令用于获取有序集合的指定下标范围内（包括）的元素，并进行升序排序。语法如下：  
 **zrange key start stop [withscores]**

key表示有序集合；

start表示开始下标，可以为负数值，如-1表示最后一个元素；

stop表示结束下标，也可以为负数值；

withscores表示是否获取元素对应的小数；如果需要，直接在命令后面加上winthscores；

如果开始下标start大于有序集合的最大下标、或者开始下标start大于结束下标stop，则返回空列表；

如果开始下标合法，但结束下标stop大于有序集合的最大下标时，则会返回有序集合的所有元素；

返回值：返回指定下标范围的元素值，并且会按照升序进行排序；如果追加了withscores参数，则连同元素对应的小数一起返回。

### zrevrange

zrevrange命令用于获取有序集合中指定下标范围内（包括）的元素，并进行降序排序，语法如下：

**zrevrange key start stop [withscores]**

key表示有序集合；

start表示开始下标，可以为负数；

stop表示结束下标，可以为负数；

withscores表示是否返回元素对应的小数，如加上，则返回；

如果开始下标start大于有序集合的最大下标、或者开始下标start大于结束下标stop，则返回空列表；

如果开始下标合法，但结束下标stop大于有序集合的最大下标，则返回有序集合的全部元素；

返回值：返回指定下标范围的元素值，并且会按照降序进行排序；如果追加了withscores参数，则连同元素对应的小数一起返回；

### zscore

zscore命令用于获取有序集合中元素对应的小数，语法如下：

**zscore key member**

key表示有序集合；

member表示元素；

如果有序集合key不存在，或元素member不存在，则返回nil；

返回值：返回元素对应的小数值。

### zrangebylex

zrangebylex命令用于获取有序集合中在指定区间内 的元素。语法如下：  
 **zrangebylex key min max [limit offset count]**

key表示有序集合；

min表示开始区间；

max表示结束区间；

min和max是一个区间，使用“（”或“[”表示，其中“（”表示开区间，指定的值不会包含在范围之内；“[”表示闭区间，指定的值会被包含在范围之内。

min和max的值除了是区间，还可以是-（负无穷）或+（正无穷），如zelcount z1 - +表示获取z1有序集合中所有的元素。

返回值：返回有序集合中在指定区间内的所有元素

### zrangebyscore

zrangebyscore命令用于获取有序集合中在指定分数区间的元素，并将返回的所有元素按照小数值从小到大（升序）进行排序。语法如下：

**zrangebyscore key min max [withscores] [limit offset count]**

key表示有序集合；

min表示开始区间

max表示结束区间；

withscores表示是否获取元素对应的小数，如果需要，直接追加在命令后面；

min和max是一个区间，使用“（”或“[”表示，其中“（”表示开区间，指定的值不会包含在范围之内；“[”表示闭区间，指定的值会被包含在范围之内；

如果没有指定区间，则默认为“[”闭区间，指定的小数会被包含进来；

返回值：返回有在指定小数区间的所有元素，并根据小数进行升序排序。

### zrevrangebyscore

zrevrangebyscore命令用户获取有序集合中在指定分数区间的所有元素，并将返回的所有元素按照小数值从大到小（降序）进行排序，语法如下：

**zrevrangebyscore key max min [withscores] [limit offset count]**

key表示有序集合；

max表示小数的开始区间；

min表示小数的结束区间；

withscores参数表示是否获取元素对应的小数，如果需要，直接追加在命令后面；

min和max是一个区间，使用“（”或“[”表示，其中“（”表示开区间，指定的值不会包含在范围之内；“[”表示闭区间，指定的值会被包含在范围之内；

如果没有指定区间，则默认为“[”闭区间，指定的小数会被包含进来；

在降序排序时，如果具有相同小数的元素，则按照元素在字典序的逆序排序。

返回值：返回在指定区间内的所有元素，并根据小数进行降序排序。

## 获取元素的排名

### zrank

zrank命令用于获取指定的元素在有序集合中的排名（从小到大），语法如下：

**zrank key member**

key表示有序集合；

member表示元素；

rank命令的排名是按照元素对应的小数值由小到大进行排序的，排名以0为底，也就是小数值最小的元素排在最后；

返回值：返回元素在有序集合中的排名；如果有序集合或元素不存在，则返回nil。

### zrevrank

zrevrank命令用户获取指定的元素在有序集合中的排名（从大到小），语法如下：

**zrevrank key member**

key表示有序集合；

member表示元素；

zrevrank命令的排名是按照元素对应的小数值从大到小进行排序的，排名以0为头，也就是小数值最大的元素排在最前；

返回值：返回元素在有序集合中的排名；如果有序集合或元素不存在，则返回nil

## 有序集合的交集与并集

### zinterstore

zinterstore命令用于将一个或多个有序集合的交集（相同的元素）复制出来保存到另一个指定的有序集合中，语法如下：

**zinterstore destination numkeys key [key……]**

**[weights weight [weight……] ]**

**[aggregate|sum|min|max ]**

destination表示要保存交集的有序集合；

key表示要提取交集的有序集合；

numkeys表示有序集合的数量，它的值也就是有序集合key的数量；

如果要保存交集的有序集合已经存在，则清空其元素，再执行zinterstore命令；

如果指定的有序集合中有的不存在，则视为空；

在默认情况下，交集中的每个元素的小数值是所有给定有序集合中该元素的小数值之和。通过可选参数weights和aggregate可以改变小数值；

可选参数weights用于为每个给定的有序集合分别指定一个乘数，每个给定的有序集合中的所有元素的小数值在传递给聚合函数之前都会乘以这个乘数，如果没有指定weights，则默认为1；

可选参数aggregate用于指定并集元素的聚合方式，可以取以下值

·sum：默认的聚合方式，将所有有序集合中并集元素的小数值之和作为并集结果中该元素的小数值；

·min：将所有有序集合中并集元素的最小小数值作为并集结果中该元素的小数值；

·max：将所有有序集合中并集元素的最大小数值作为并集结果中该元素的小数值；

返回值：返回有序集合中的元素数量，也就是并集元素数量。

### zunionstore

zunionstore命令用于将给定的一个或多个有序集合中的并集（将所有元素合体）复制并保存到另一个有序集合中，语法如下：

**zunionstore destination numkeys key [key……]**

**[weights weights [weigths…..]**

**[ aggregate|sum|min|max ]**

destination表示要保存并集的有序集合；

key表示要提取并集元素的有序集合；

numkeys表示有提取几个有序集合的并集，参数值为有序集合key的数量；

如果要保存并集的有序集合已经存在，则清空其元素，再执行zunionstore命令；

如果要提取并集元素的有序集合不存在，则视为空；

默认情况下，并集中元素对应的小数值是所有有序集合中该元素的小数值之和；通过可选参数weights和aggregate可以改变小数值；

可选参数weights用于为每个给定的有序集合分别指定一个乘数，每个给定的有序集合中的所有元素的小数值在传递给聚合函数之前都会乘以这个乘数，如果没有指定weights，则默认为1；

可选参数aggregate用于指定并集元素的聚合方式，可以取以下值

·sum：默认的聚合方式，将所有有序集合中并集元素的小数值之和作为并集结果中该元素的小数值；

·min：将所有有序集合中并集元素的最小小数值作为并集结果中该元素的小数值；

·max：将所有有序集合中并集元素的最大小数值作为并集结果中该元素的小数值；

返回值：返回有序集合中的元素数量，也就是并集元素数量。

## 删除有序集合的元素

### zrem

zrem命令用于删除有序集合中的一个或多个元素，语法如下：

**zrem key member [member……]**

key表示有序集合；

member表示要删除的元素，可以是多个；

如果要删除的元素不存在，则忽略该元素；

如果有序集合不存在，则返回0；

返回值：返回被成功删除元素的个数，不包括被忽略的元素。

### zremrangebylex

zremarangebylex命令用于删除有序集合中，在指定区间内（包括）的所有元素，语法如下：  
  **zremrangebylex key min max**

key表示有序集合；

min表示开始区间；

max表示结束区间；

min和max是一个区间，使用“（”或“[”表示，其中“（”表示开区间，指定的值不会包含在范围之内；“[”表示闭区间，指定的值会被包含在范围之内。

min和max的值除了是区间，还可以是-（负无穷）或+（正无穷），如zelcount z1 - +表示删除z1有序集合中所有的元素；

如果没有指定区间，则默认为“[”闭区间；

返回值：返回被成功删除元素的个数。

### zremrangebyrank

zremrangebyrank命令用于删除有序集合中在指定排名区间内（包括）的所有元素，语法如下：

**zremrangebyrank key start stop**

key表示有序集合；

start表示开始排名；

stop表示结束排名；

返回值：成功执行后，返回被成功删除元素的个数。

### zremrangebyscore

zremrangebyscore命令用于删除有序集合中指定小数区间内（包括）的所有元素，语法如下：

**zremrangebyscore key min max**

key表示有序集合；

min表示小数的最小值；

max表示小数的最大值；

如果有序集合不存在，返回0；

返回值：执行成功后，返回被成功删除元素的个数。