Redis端口6379

启动redis服务端：redis-server [\*\*\*\*.conf]

service redis start

启动redis客户端：redis-cli

停止redis服务端

service redis stop

启动后，查看redis进程的端口：ps -ef|grep redis

查看是否启动成功：-a:认证密码，-p：端口。

redis-cli -a myredis -p 6379

## key pattern 查询相应的key

（1）redis允许模糊查询key　　有3个通配符 \*、?、[]

（2）randomkey：返回随机key

（3）type key：返回key存储的类型

（4）exists key：判断某个key是否存在

（5）del key：删除key

（6）rename key newkey：改名

（7）renamenx key newkey：如果newkey不存在则修改成功

（8）move key 1：将key移动到1数据库

（9）ttl key：查询key的生命周期（秒）

（10）expire key 整数值：设置key的生命周期以秒为单位

（11）pexpire key 整数值：设置key的生命周期以毫秒为单位

（12）pttl key：查询key 的生命周期（毫秒）

（13）perisist key：把指定key设置为永久有效

## 字符串类型的操作

### 1、新增

#### a）set

语法：set key value

解释：把值value赋给key，如果key不存在，新增；否则，更新

[plain] [view plain](http://blog.csdn.net/love__coder/article/details/8274331) [copy](http://blog.csdn.net/love__coder/article/details/8274331)

[root@xsf001 ~]# redis-cli

redis 127.0.0.1:6379> set user.1.name zhangsan #设置user.1.name 为zhangsan

OK

redis 127.0.0.1:6379> set user.name 45      #设置user.1.name 为45

OK

#### b）setnx

语法：setnx key value

解释：只insert不update，即，仅仅key不存在时，则设置key的值为value，并返回1，否则返回0  。setnx 是set if not exists 的缩写

[plain] [view plain](http://blog.csdn.net/love__coder/article/details/8274331) [copy](http://blog.csdn.net/love__coder/article/details/8274331)

redis 127.0.0.1:6379> setnx user.1.name zhangsan   #user.1.name已经存在，则返回0

(integer) 0

redis 127.0.0.1:6379> setnx user.2.name zhangsan  #user.2.name不存在，则设置

(integer) 1

#### c）setex

语法： setex key seconds value

解释：设置key的过期时间和值。过期时间seconds单位是秒。设置过期时间和值是原子操作，如果redis仅仅当做缓存，这个很命令很有用。

[plain] [view plain](http://blog.csdn.net/love__coder/article/details/8274331) [copy](http://blog.csdn.net/love__coder/article/details/8274331)

redis 127.0.0.1:6379> setex user.2.age 2 14  #把user.2.age 的值设14 并且2秒后过期失效

OK

redis 127.0.0.1:6379> get user.2.age  #失效前

"14"

redis 127.0.0.1:6379> get user.2.age #失效后

(nil)

#### d）mset

语法：mset key value [key value ...]

解释：同时设置多个key-value

[plain] [view plain](http://blog.csdn.net/love__coder/article/details/8274331) [copy](http://blog.csdn.net/love__coder/article/details/8274331)

redis 127.0.0.1:6379> mset user.4.name lisi user.4.age 34  #设置user.4.name=lisi,user.4.age=34

OK

redis 127.0.0.1:6379> get user.4.name

"lisi"

redis 127.0.0.1:6379> get user.4.age

"34"

#### e）msetnx

语法：msetnx key value [key value ...]

解释：所有key都不存在才执行set操作

[plain] [view plain](http://blog.csdn.net/love__coder/article/details/8274331) [copy](http://blog.csdn.net/love__coder/article/details/8274331)

redis 127.0.0.1:6379> msetnx user.4.name lisi user.4.age 34  #key 都设置过，无法再次set

(integer) 0

redis 127.0.0.1:6379> msetnx user.4.name lisi user.4.std 3   #key user.4.name 曾设置过，无法再次设置

(integer) 0

redis 127.0.0.1:6379> msetnx user.4.tech lisi user.4.std 3   #key都没有设置过，可以再次设

(integer) 1  

### 2、查询

#### a）get

语法：get key

解释：获取key所set的值

[plain] [view plain](http://blog.csdn.net/love__coder/article/details/8274331) [copy](http://blog.csdn.net/love__coder/article/details/8274331)

redis 127.0.0.1:6379> get user.4.name  #获取user.4.name 的值

"lisi"

redis 127.0.0.1:6379> get user.4.age

"34"

redis 127.0.0.1:6379> get user.4.tech

"lisi"

redis 127.0.0.1:6379> get user.4.std

"3"

#### b）mget

语法： get key [key]

解释：批量获取key的值。程序一次获取多个值，可以减少网络连接损耗。

[plain] [view plain](http://blog.csdn.net/love__coder/article/details/8274331) [copy](http://blog.csdn.net/love__coder/article/details/8274331)

redis 127.0.0.1:6379> mget user.4.name user.4.age user.4.std #批量获取存在key的值

1) "lisi"   #user.4.name的值

2) "34"     #user.4.age 的值

3) "3"      #user.4.std 的值

redis 127.0.0.1:6379> mget user.4.name user.4.age user.4.std user.4.fri  #key user.4.fri 不存在仍然可以返回

1) "lisi"

2) "34"

3) "3"

4) (nil) #user.4.fri 的值

#### c）getrange

语法：getrange key star end

解释：获取存 储在key中value的字串。字符串的截取有star和end决定，字符串的第一个字符编号是0，第二个是1，一次类推；如果是负数，-1是最后一个字符，-2是倒数第二个字符，一次类推。

[plain] [view plain](http://blog.csdn.net/love__coder/article/details/8274331) [copy](http://blog.csdn.net/love__coder/article/details/8274331)

redis 127.0.0.1:6379> get user.4.name

"lisi"

redis 127.0.0.1:6379> getrange user.4.name 0 3  # 0 表示开始

"lisi"

redis 127.0.0.1:6379> getrange user.4.name 1 2

"is"

redis 127.0.0.1:6379> getrange user.4.name 1 -2 #-2 表示倒数第二

"is"

redis 127.0.0.1:6379> getrange user.4.name -1 -2  # end 》 start

""

redis 127.0.0.1:6379> getrange user.4.name 1 66  #end 》 值的长度

"isi"

### 3、修改

#### a）getset

语法：getset key value

解释：设置key的值，并返回key的旧值。

[plain] [view plain](http://blog.csdn.net/love__coder/article/details/8274331) [copy](http://blog.csdn.net/love__coder/article/details/8274331)

redis 127.0.0.1:6379> get user.4.name   #存在的key

"lisi"

redis 127.0.0.1:6379> getset user.4.name wangwu  #把存在的user.4.name设置为wagnwu

"lisi"

redis 127.0.0.1:6379> get user.4.name

"wangwu"

redis 127.0.0.1:6379> get user.5.name  #不存在的key

(nil)

redis 127.0.0.1:6379> getset user.5.name lisi

(nil)

redis 127.0.0.1:6379> get user.5.name

"lisi"

#### b）append

语法：append key value

解释：key存在，在旧值的后面追加value；key不存在，直接set

[plain] [view plain](http://blog.csdn.net/love__coder/article/details/8274331) [copy](http://blog.csdn.net/love__coder/article/details/8274331)

redis 127.0.0.1:6379> get user.4.name #存在的key

"wangwu"

redis 127.0.0.1:6379> append user.4.name 01

(integer) 8

redis 127.0.0.1:6379> get user.4.name

"wangwu01"

[plain] [view plain](http://blog.csdn.net/love__coder/article/details/8274331) [copy](http://blog.csdn.net/love__coder/article/details/8274331)

redis 127.0.0.1:6379> get user.6.name #不能存在的key

(nil)

redis 127.0.0.1:6379> append user.6.name jim

(integer) 3

redis 127.0.0.1:6379> get user.6.name

"jim"

#### c）setrange

语法：setrange key offset value

解释：用value重写key值的一部分，偏移量由offset指定

[plain] [view plain](http://blog.csdn.net/love__coder/article/details/8274331) [copy](http://blog.csdn.net/love__coder/article/details/8274331)

redis 127.0.0.1:6379> get user.4.name  #key存在

"wangwu01"

redis 127.0.0.1:6379> setrange user.4.name 0 lisi

(integer) 8

redis 127.0.0.1:6379> get user.4.name

"lisiwu01"

redis 127.0.0.1:6379> setrange user.4.name 9 lisi  # offset 》字符串长度

(integer) 13

redis 127.0.0.1:6379> get user.4.name

"lisiwu01\x00lisi"

redis 127.0.0.1:6379> setrange user.4.name 8 lisi

(integer) 13

redis 127.0.0.1:6379> get user.4.name

"lisiwu01lisii"

redis 127.0.0.1:6379> get user.6.std #key 不存在

(nil)

redis 127.0.0.1:6379> setrange user.6.std 0 3

(integer) 1

redis 127.0.0.1:6379> get user.6.std

"3"

#### d）incr

语法：incr key

解释：key中如果存储的是数字，则可以通过incr递增key的值，返回递增后的值。如果key不能存在，视为初始值为0

[plain] [view plain](http://blog.csdn.net/love__coder/article/details/8274331) [copy](http://blog.csdn.net/love__coder/article/details/8274331)

redis 127.0.0.1:6379> get user.7.age #key不存在 ，初始值视为0,

(nil)

redis 127.0.0.1:6379> incr user.7.age

(integer) 1

redis 127.0.0.1:6379> get user.7.age

"1"

redis 127.0.0.1:6379> incr user.7.age

(integer) 2

#### e）incrby

语法：incrby key increment

解释：用指定的步长增加key存储的数字。如果步长increment是负数，则减

[plain] [view plain](http://blog.csdn.net/love__coder/article/details/8274331) [copy](http://blog.csdn.net/love__coder/article/details/8274331)

redis 127.0.0.1:6379> get user.7.age

"3"

redis 127.0.0.1:6379> incrby user.7.age 15  #增加15

(integer) 18

redis 127.0.0.1:6379> get user.7.age

"18"

redis 127.0.0.1:6379> incrby user.7.fri 18  #key不能存在，原值视为0

(integer) 18

redis 127.0.0.1:6379> get user.7.fri

"18"

redis 127.0.0.1:6379> incrby  user.7.age -1  #步长为负

(integer) 17

redis 127.0.0.1:6379> get user.7.age

"17"

#### f）decr

语法：decr key

解释：递减key保存的数字，如果key不存在，初始值视为0

[plain] [view plain](http://blog.csdn.net/love__coder/article/details/8274331) [copy](http://blog.csdn.net/love__coder/article/details/8274331)

redis 127.0.0.1:6379> get user.7.age

"17"

redis 127.0.0.1:6379> decr user.7.age

(integer) 16

redis 127.0.0.1:6379> decr user.7.num #key 不存在，初始值视为0

(integer) -1

redis 127.0.0.1:6379> decr user.7.num

(integer) -2

#### g）decrby

语法：decrby key decrement

解释：用指定的步长递减key的值，如果步长decrment是负值，则递增

[plain] [view plain](http://blog.csdn.net/love__coder/article/details/8274331) [copy](http://blog.csdn.net/love__coder/article/details/8274331)

redis 127.0.0.1:6379> decrby user.7.num 4  #递减

(integer) -6

redis 127.0.0.1:6379> decrby user.7.num -9 #负值，递增

(integer) 3

注意：递增递减系列的函数，只能对保存的是数字的key操作，不能是字符串

### 4、删除

语法：del key [key]

解释：删除指定的key，返回删除key的个数

[plain] [view plain](http://blog.csdn.net/love__coder/article/details/8274331) [copy](http://blog.csdn.net/love__coder/article/details/8274331)

redis 127.0.0.1:6379> del user.7.num

(integer) 1

redis 127.0.0.1:6379> get user.7.num

(nil)

redis 127.0.0.1:6379> del user.7.age user.7.fri #删除多个key

(integer) 2

redis 127.0.0.1:6379> mget user.7.age user.7.fri

1) (nil)

2) (nil)

### 5、其他

语法：strlen key

解释：获取key中所存储值的长度

[plain] [view plain](http://blog.csdn.net/love__coder/article/details/8274331) [copy](http://blog.csdn.net/love__coder/article/details/8274331)

redis 127.0.0.1:6379> get user.1.name

"45"

redis 127.0.0.1:6379> strlen user.1.name  #user.1.name 长度

(integer) 2

redis 127.0.0.1:6379> strlen user.8.name #key不存在

(integer) 0

## 链表操作

Redis的list类型其实就是一个每个子元素都是string类型的双向链表，链表的最大长度是2^32。list既可以用做栈，也可以用做队列。

list的pop操作还有阻塞版本，主要是为了避免轮询

（1）lpush key value：把值插入到链表头部

（2）rpush key value：把值插入到链表尾部

（3）lpop key ：返回并删除链表头部元素

（4）rpop key： 返回并删除链表尾部元素

（5）lrange key start stop：返回链表中[start, stop]中的元素

（6）lrem key count value：从链表中删除value值，删除count的绝对值个value后结束

count > 0 从表头删除　　count < 0 从表尾删除　　count=0 全部删除

（7）ltrim key start stop：剪切key对应的链接，切[start, stop]一段并把改制重新赋给key

（8）lindex key index：返回index索引上的值

（9）llen key：计算链表的元素个数

（10）linsert key after|before search value：在key 链表中寻找search，并在search值之前|之后插入value

（11）rpoplpush source dest：把source 的末尾拿出，放到dest头部，并返回单元值

应用场景： task + bak 双链表完成安全队列

业务逻辑： rpoplpush task bak

接收返回值并做业务处理

如果成功则rpop bak清除任务，如果不成功，下次从bak表取任务

（12）brpop，blpop key timeout：等待弹出key的尾/头元素

timeout为等待超时时间，如果timeout为0则一直等待下去

应用场景：长轮询ajax，在线聊天时能用到

在总结list之前，先要弄明白几个跟list相关的概念：

列表：一个从左到右的队列，个人理解更类似于一个栈，常规模式下，先进列表的元素，后出。

表头元素：列表最左端第一个元素。

表尾元素：列表最右端的最后一个元素。不包含任何元素的列表成为空列表。

### 1）新增

#### a) lpush

语法：lpush key value[value]

解释：把一个或多个元素插入表头。如果是多个value时，按照从左到右的次序插。返回插入元素的个数

[plain] [view plain](http://blog.csdn.net/love__coder/article/details/8293225) [copy](http://blog.csdn.net/love__coder/article/details/8293225)

[root@xsf001 ~]# redis-cli

redis 127.0.0.1:6379> lpush lst.user zhangsan   #插入一个元素

(integer) 1

redis 127.0.0.1:6379> lpush lst.user zhangsan lisi   #插入多个元素，list中允许插入重复的元素

(integer) 3

#### b）lpushx

语法：lpushx key value

解释：插入一个表头元素，当且仅当列表key存在时，才能插入。返回列表中元素的个数

[plain] [view plain](http://blog.csdn.net/love__coder/article/details/8293225) [copy](http://blog.csdn.net/love__coder/article/details/8293225)

redis 127.0.0.1:6379> lpushx lst.user wangwu   #列表存在

(integer) 4

redis 127.0.0.1:6379> lpushx lst.tech wangwu   #列表不存在，不插入

(integer) 0

#### c）rpush

语法：rpush key [value]

解释：将一个或多个值插入到队列的队尾。多值时，从左到右依次添加。返回列表中元素个数

[plain] [view plain](http://blog.csdn.net/love__coder/article/details/8293225) [copy](http://blog.csdn.net/love__coder/article/details/8293225)

redis 127.0.0.1:6379> rpush lst.user ls005     #列表存在

(integer) 5

redis 127.0.0.1:6379> rpush lst.tech tec01 tec02 tec03  #列表不存在

(integer) 3

#### d) rpushx

语法：rpushx key value

解释：讲值插入到列表队尾，当且仅当列表key存在时，才添加。返回列表元素个数

[plain] [view plain](http://blog.csdn.net/love__coder/article/details/8293225) [copy](http://blog.csdn.net/love__coder/article/details/8293225)

redis 127.0.0.1:6379> rpushx lst.tech tec04  #key 存在

(integer) 4

redis 127.0.0.1:6379> rpushx lst.sub englist #key 不存在

(integer) 0

#### e）linsert

语法：linsert key before|after pivot value

解释：将值插入到pivot的前面或后面。返回列表元素个数。如果参照点pivot不存在不插入。如果有多个pivot，以离表头最近的为准

[plain] [view plain](http://blog.csdn.net/love__coder/article/details/8293225) [copy](http://blog.csdn.net/love__coder/article/details/8293225)

redis 127.0.0.1:6379> linsert lst.tech after tec04 tec06  #后面插

(integer) 5

redis 127.0.0.1:6379> linsert lst.tech before tec06 tec05 #前面插

(integer) 6

redis 127.0.0.1:6379> linsert lst.tech before tec08 tec07 #参照点不存在，不插

(integer) -1

redis 127.0.0.1:6379> linsert lst.user after zhangsan zhangsan01  #列表中有多个pivot，以从左到右的第一个为准

(integer) 6

### 2）查询

#### a) lindex

语法：lindex key index

解释：通过索引index获取列表的元素。 index>=0时， 0 表头，1 第二个元素，依次类推；index<0时，-1，表尾，-2倒数第二个元素，依次类推

[plain] [view plain](http://blog.csdn.net/love__coder/article/details/8293225) [copy](http://blog.csdn.net/love__coder/article/details/8293225)

redis 127.0.0.1:6379> lindex lst.user 0  #表头

"wangwu"

redis 127.0.0.1:6379> lindex lst.user -1 #表尾

"ls005"

redis 127.0.0.1:6379> lindex lst.user 2 #第三个元素

"zhangsan"

redis 127.0.0.1:6379> lindex lst.user -2 #倒数第二个元素

"zhangsan"

#### b）lrange

语法：lrange key start stop

解释：获取指定开始和结束范围的一些列元素。0：表头，-1：表尾。如果stop指定的元素在start的左边，返回空列表

[plain] [view plain](http://blog.csdn.net/love__coder/article/details/8293225) [copy](http://blog.csdn.net/love__coder/article/details/8293225)

redis 127.0.0.1:6379> lrange lst.user 0 -1 #返回所有

1) "wangwu"

2) "lisi"

3) "zhangsan"

4) "zhangsan01"

5) "zhangsan"

6) "ls005"

redis 127.0.0.1:6379> lrange lst.user -1 0  #返回空

(empty list or set)

redis 127.0.0.1:6379> lrange lst.user 1 2  #返回多个

1) "lisi"

2) "zhangsan"

redis 127.0.0.1:6379> lrange lst.user 1 1   #返回一个元素

1) "lisi"

### 3）修改

lset

语法：lset key index value

解释：设置列表指定索引的值，如果指定索引不存在则报错

[plain] [view plain](http://blog.csdn.net/love__coder/article/details/8293225) [copy](http://blog.csdn.net/love__coder/article/details/8293225)

redis 127.0.0.1:6379> lset lst.user 2 zhangsan1  #设置第三个元素为zhangsan1

OK

redis 127.0.0.1:6379> lrange lst.user 0 -1

1) "wangwu"

2) "lisi"

3) "zhangsan1"

4) "zhangsan01"

5) "zhangsan"

6) "ls005"

redis 127.0.0.1:6379> lset lst.user 6 ls006  #指定索引不存在

(error) ERR index out of range

### 4）删除

#### a）ltrim

语法：ltrim key start stop

解释：保留指定区域的元素，其他元素全部删除

[plain] [view plain](http://blog.csdn.net/love__coder/article/details/8293225) [copy](http://blog.csdn.net/love__coder/article/details/8293225)

redis 127.0.0.1:6379> ltrim lst.user 0 -2

OK

redis 127.0.0.1:6379> lrange lst.user 0 -1

1) "wangwu"

2) "lisi"

3) "zhangsan1"

4) "zhangsan01"

5) "zhangsan"

#### b）lrem

语法：lrem key count value

解释：移除等于value的元素，当count>0时，从表头开始查找，移除count个；当count=0时，从表头开始查找，移除所有等于value的；当count<0时，从表尾开始查找，移除|count| 个。

[plain] [view plain](http://blog.csdn.net/love__coder/article/details/8293225) [copy](http://blog.csdn.net/love__coder/article/details/8293225)

redis 127.0.0.1:6379> lrange lst.user 0 -1

 1) "zhangsan"

 2) "wangwu"

 3) "lisi"

 4) "zhangsan1"

 5) "zhangsan01"

 6) "zhangsan"

 7) "lisi"

 8) "zhangsan"

 9) "lisi"

10) "zhangsan"

redis 127.0.0.1:6379> lrem lst.user 2 zhangsan #移除前两个zhangsan

(integer) 2

redis 127.0.0.1:6379> lrange lst.user 0 -1

1) "wangwu"

2) "lisi"

3) "zhangsan1"

4) "zhangsan01"

5) "lisi"

6) "zhangsan"

7) "lisi"

8) "zhangsan"

redis 127.0.0.1:6379> lrange lst.user -1 zhangsan #移除最后一个zhangsan

(empty list or set)

redis 127.0.0.1:6379> lrange lst.user 0 -1

1) "wangwu"

2) "lisi"

3) "zhangsan1"

4) "zhangsan01"

5) "lisi"

6) "zhangsan"

7) "lisi"

redis 127.0.0.1:6379> lrem lst.user 0 lisi  #移除所有lisi

(integer) 3

redis 127.0.0.1:6379> lrange lst.user 0 -1

1) "wangwu"

2) "zhangsan1"

3) "zhangsan01"

4) "zhangsan"

redis 127.0.0.1:6379>

#### c）rpop

语法：rpop key

解释：移除并返回表尾元素

[plain] [view plain](http://blog.csdn.net/love__coder/article/details/8293225) [copy](http://blog.csdn.net/love__coder/article/details/8293225)

redis 127.0.0.1:6379> rpop lst.user

"zhangsan"

#### d）lpop

语法：lpop key

解释：移除并返回表头元素

[plain] [view plain](http://blog.csdn.net/love__coder/article/details/8293225) [copy](http://blog.csdn.net/love__coder/article/details/8293225)

redis 127.0.0.1:6379> lpop lst.user

"wangwu"

### 5）其他

#### a）llen

语法：llen key

解释：获取列表长度

[plain] [view plain](http://blog.csdn.net/love__coder/article/details/8293225) [copy](http://blog.csdn.net/love__coder/article/details/8293225)

redis 127.0.0.1:6379> llen lst.user

(integer) 2

## hashes类型及操作

　　Redis hash 是一个string类型的field和value的映射表，它的添加、删除操作都是O(1)（平均）。hash特别适用于存储对象，将一个对象存储在hash类型中会占用更少的内存，并且可以方便的存取整个对象。

配置： hash\_max\_zipmap\_entries 64 #配置字段最多64个

hash\_max\_zipmap\_value 512 #配置value最大为512字节

（1）hset myhash field value：设置myhash的field为value

（2）hsetnx myhash field value：不存在的情况下设置myhash的field为value

（3）hmset myhash field1 value1 field2 value2：同时设置多个field

（4）hget myhash field：获取指定的hash field

（5）hmget myhash field1 field2：一次获取多个field

（6）hincrby myhash field 5：指定的hash field加上给定的值

（7）hexists myhash field：测试指定的field是否存在

（8）hlen myhash：返回hash的field数量

（9）hdel myhash field：删除指定的field

（10）hkeys myhash：返回hash所有的field

（11）hvals myhash：返回hash所有的value

（12）hgetall myhash：获取某个hash中全部的field及value

hash是一些列key value（field value）的映射表。常常用其存储一些对象实例。相对于把一个对象的各个字段存储为[string](http://blog.csdn.net/love__coder/article/details/8274331)，存储为hash会占用更少的内存。为什么会更省内存呢？需要搞清楚两个配置（hash-max-zipmap-entries和hash-max-zipmap-value）的含义，配置的详细介绍，我打算放在最后的配置优化环节讲。

### 1）新增

#### a）hset

语法：hset key field value

解释：设置hash表key中的field的值。如果hash表不存在，则创建，并执行设置field的值，如果hash表存在，值field的值覆盖或新增

[plain] [view plain](http://blog.csdn.net/love__coder/article/details/8274886) [copy](http://blog.csdn.net/love__coder/article/details/8274886)

1. [root@xsf001 ~]# redis-cli
2. redis 127.0.0.1:6379> hset user.1 name zhangsan    #设置key user.1 name域 的值
3. (integer) 1
4. redis 127.0.0.1:6379> hset user.1 age 45 #设置age域
5. (integer) 1
6. redis 127.0.0.1:6379> hset user.1 tech lisi
7. (integer) 1

#### b）hmset

语法：hash key field value[key value]

解释：批量设置hash表key的域

[plain] [view plain](http://blog.csdn.net/love__coder/article/details/8274886) [copy](http://blog.csdn.net/love__coder/article/details/8274886)

1. redis 127.0.0.1:6379> hmset user.2 name niuer age 34 #同时设置name 和age域
2. OK

#### c）hsetnx

语法：hsetnx key field value

解释：仅仅当field域不存在时，设置hash表field的值

[plain] [view plain](http://blog.csdn.net/love__coder/article/details/8274886) [copy](http://blog.csdn.net/love__coder/article/details/8274886)

1. redis 127.0.0.1:6379> hsetnx user.1 name lisi  #由于name域已经设置过，所以返回0
2. (integer) 0
3. redis 127.0.0.1:6379> hsetnx user.1 fri 5
4. (integer) 1   #fri域没有设置过，所以hset并返回1

### 2）查询

#### a）hget

语法：hget key field

解释：获取哈希表key的field值

[plain] [view plain](http://blog.csdn.net/love__coder/article/details/8274886) [copy](http://blog.csdn.net/love__coder/article/details/8274886)

1. redis 127.0.0.1:6379> hget user.1 name  #存在的hash表及域
2. "zhangsan"
3. redis 127.0.0.1:6379> hget user.3 name  #不存在的hash表
4. (nil)
5. redis 127.0.0.1:6379> hget user.1 bb  #不存在的域
6. (nil)

#### b）hmget

语法：hmget key field[field]

解释：批量获取hash表的filed

[plain] [view plain](http://blog.csdn.net/love__coder/article/details/8274886) [copy](http://blog.csdn.net/love__coder/article/details/8274886)

1. redis 127.0.0.1:6379> hmget user.1 name age fri tech
2. 1) "zhangsan"
3. 2) "45"
4. 3) "5"
5. 4) "lisi"
6. redis 127.0.0.1:6379> hmget user.1 name age fri tech nofield #存在hash表中包含不存在的域nofield
7. 1) "zhangsan"
8. 2) "45"
9. 3) "5"
10. 4) "lisi"
11. 5) (nil)
12. redis 127.0.0.1:6379> hmget user.3 name age fri #不存在的hash表
13. 1) (nil)
14. 2) (nil)
15. 3) (nil)

#### c）hgetall

语法：hgetall key

解释：获取hash表的所有域值

[plain] [view plain](http://blog.csdn.net/love__coder/article/details/8274886) [copy](http://blog.csdn.net/love__coder/article/details/8274886)

1. redis 127.0.0.1:6379> hgetall user.2  #存在的hash表
2. 1) "name"   #域
3. 2) "niuer"  #域name的值
4. 3) "age"    #域
5. 4) "34"     #域age的值
6. redis 127.0.0.1:6379> hgetall user.3  #不存在的hansh表
7. (empty list or set)

#### d）hexists

语法：hexists key field

解释：判断hash表中是否存在某个域

[plain] [view plain](http://blog.csdn.net/love__coder/article/details/8274886) [copy](http://blog.csdn.net/love__coder/article/details/8274886)

1. redis 127.0.0.1:6379> hexists user.1 name  #存在
2. (integer) 1
3. redis 127.0.0.1:6379> hexists user.1 nofield  #不存在
4. (integer) 0
5. redis 127.0.0.1:6379> hexists use1 nofield #hash表不存在
6. (integer) 0

#### e）hkeys

语法：hkeys key

解释：获取hash表的所有域

[plain] [view plain](http://blog.csdn.net/love__coder/article/details/8274886) [copy](http://blog.csdn.net/love__coder/article/details/8274886)

1. redis 127.0.0.1:6379> hkeys user.1  #存在的hash表
2. 1) "name"
3. 2) "age"
4. 3) "tech"
5. 4) "fri"
6. redis 127.0.0.1:6379> hkeys user.4 #不存在的hash
7. (empty list or set)

#### f）hvals

语法：hvals key

解释：获取hash表的所有域值

[plain] [view plain](http://blog.csdn.net/love__coder/article/details/8274886) [copy](http://blog.csdn.net/love__coder/article/details/8274886)

1. redis 127.0.0.1:6379> hvals user.1  #存在hash
2. 1) "zhangsan"
3. 2) "45"
4. 3) "lisi"
5. 4) "5"
6. redis 127.0.0.1:6379> hvals user.4 #不存在
7. (empty list or set)

### 3）修改

语法：hincrby key field increment

解释：hash表field域的数值增加步长increment，如果increment是负值，则是递减。如果域不存在，初始值视为0

[plain] [view plain](http://blog.csdn.net/love__coder/article/details/8274886) [copy](http://blog.csdn.net/love__coder/article/details/8274886)

1. redis 127.0.0.1:6379> hincrby user.1 age 2  #增加2
2. (integer) 47
3. redis 127.0.0.1:6379> hincrby user.1 age -3  #减少3
4. (integer) 44
5. redis 127.0.0.1:6379> hincrby user.1 age2 -3 #域不能存在，初始值是0
6. (integer) -3

### 4）删除

语法：hdel key field[field]

解释：删除hash的域，如果指定多个field，则删除多个

[plain] [view plain](http://blog.csdn.net/love__coder/article/details/8274886) [copy](http://blog.csdn.net/love__coder/article/details/8274886)

1. redis 127.0.0.1:6379> hkeys user.1
2. 1) "name"
3. 2) "age"
4. 3) "tech"
5. 4) "fri"
6. 5) "age2"
7. redis 127.0.0.1:6379> hdel user.1 age2  #删除一个域
8. (integer) 1
9. redis 127.0.0.1:6379> hkeys user.1
10. 1) "name"
11. 2) "age"
12. 3) "tech"
13. 4) "fri"
14. redis 127.0.0.1:6379> hdel user.1 fri tech #删除2个域
15. (integer) 2
16. redis 127.0.0.1:6379> hkeys user.1
17. 1) "name"
18. 2) "age"
19. redis 127.0.0.1:6379> hdel user.1 bb #删除一个不存在的域
20. (integer) 0 #返回0

### 5）其他

语法：hlen key

解释：获取hash的域数量

[plain] [view plain](http://blog.csdn.net/love__coder/article/details/8274886) [copy](http://blog.csdn.net/love__coder/article/details/8274886)

1. redis 127.0.0.1:6379> hkeys user.1
2. 1) "name"
3. 2) "age"
4. redis 127.0.0.1:6379> hlen user.1 #存在2个域
5. (integer) 2
6. redis 127.0.0.1:6379> hlen user.4  #不存在的hash
7. (integer) 0

## 集合结构操作

特点：无序性、确定性、唯一性

（1）sadd key value1 value2：往集合里面添加元素

（2）smembers key：获取集合所有的元素

（3）srem key value：删除集合某个元素

（4）spop key：返回并删除集合中1个随机元素（可以坐抽奖，不会重复抽到某人）

（5）srandmember key：随机取一个元素

（6）sismember key value：判断集合是否有某个值

（7）scard key：返回集合元素的个数

（8）smove source dest value：把source的value移动到dest集合中

（9）sinter key1 key2 key3：求key1 key2 key3的交集

（10）sunion key1 key2：求key1 key2 的并集

（11）sdiff key1 key2：求key1 key2的差集

（12）sinterstore res key1 key2：求key1 key2的交集并存在res里

redis set 是string类型对象的无序集合，set不管存储多少对象，对存储对象的add，remove和test操作的时间复杂度是O(1)。set最多能包含 232 - 1 个member。

### 增加

#### a) sadd

语法：sadd key member[member...]

解释：对特定key的set增加一个或多个值，返回是增加元素的个数。注意：对同一个member多次add，set中只会保留一份。

[plain] [view plain](http://blog.csdn.net/love__coder/article/details/8497497) [copy](http://blog.csdn.net/love__coder/article/details/8497497)

[root@xsf001 ~]# redis-cli

redis 127.0.0.1:6379> sadd stu zhangsan lisi wangwu #新增

(integer) 3

redis 127.0.0.1:6379> smembers stu    #得到set的所有member

1) "wangwu"

2) "lisi"

3) "zhangsan"

redis 127.0.0.1:6379> sadd stu zhangsan #增加存在的member

(integer) 0

redis 127.0.0.1:6379> smembers stu

1) "wangwu"

2) "lisi"

3) "zhangsan"

redis 127.0.0.1:6379> sadd tech wangwu liming joe

(integer) 3

redis 127.0.0.1:6379> sadd tech jim

(integer) 1

redis 127.0.0.1:6379> smembers tech

1) "jim"

2) "liming"

3) "wangwu"

4) "joe"

### 2、查询

#### a）smembers

语法：smembers key

解释：获取set中的所有member

[plain] [view plain](http://blog.csdn.net/love__coder/article/details/8497497) [copy](http://blog.csdn.net/love__coder/article/details/8497497)

redis 127.0.0.1:6379> smembers stu

1) "wangwu"

2) "lisi"

3) "zhangsan"

redis 127.0.0.1:6379> smembers tech

1) "jim"

2) "liming"

3) "wangwu"

4) "joe"

#### b）sismember

语法：sismember key member

解释：判断值是否是set的member。如果值是set的member返回1，否则，返回0

[plain] [view plain](http://blog.csdn.net/love__coder/article/details/8497497) [copy](http://blog.csdn.net/love__coder/article/details/8497497)

redis 127.0.0.1:6379> sismember tech jim #jim 是set的member

(integer) 1

redis 127.0.0.1:6379> sismember tech jim001 #jim001 不是set的member

(integer) 0

#### c）scard

语法：scard key

解释：返回set的member个数，如果set不存在，返回0

[plain] [view plain](http://blog.csdn.net/love__coder/article/details/8497497) [copy](http://blog.csdn.net/love__coder/article/details/8497497)

redis 127.0.0.1:6379> scard tech  # tech 存在

(integer) 4

redis 127.0.0.1:6379> scard stud #stud 不存在

(integer) 0

redis 127.0.0.1:6379> scard stu

(integer) 4

#### d）srandmember

语法：srandmember key

解释：从set中返回一个随机member

[plain] [view plain](http://blog.csdn.net/love__coder/article/details/8497497) [copy](http://blog.csdn.net/love__coder/article/details/8497497)

redis 127.0.0.1:6379> srandmember stu

"zhangsan"

redis 127.0.0.1:6379> srandmember stu

"zhangsan"

redis 127.0.0.1:6379> srandmember stu

"wangwu"

redis 127.0.0.1:6379> srandmember stu

"zhangsan01"

### 3、删除

#### a）spop

语法：spop key

解释：移除并返回一个随机member

[plain] [view plain](http://blog.csdn.net/love__coder/article/details/8497497) [copy](http://blog.csdn.net/love__coder/article/details/8497497)

redis 127.0.0.1:6379> smembers stu #pop前

1) "zhangsan01"

2) "wangwu"

3) "lisi"

4) "zhangsan"

redis 127.0.0.1:6379> spop stu  #移除一个随机member

"lisi"

redis 127.0.0.1:6379> smembers stu #pop后

1) "zhangsan01"<span style="white-space:pre">   </span>

2) "wangwu"

3) "zhangsan"

#### b）srem

语法：srem key member [member ...]

解释：移除一个或多个member

[plain] [view plain](http://blog.csdn.net/love__coder/article/details/8497497) [copy](http://blog.csdn.net/love__coder/article/details/8497497)

redis 127.0.0.1:6379> smembers tech

1) "jim"

2) "liming"

3) "wangwu"

4) "joe"

redis 127.0.0.1:6379> srem tech jim   #移除jim

(integer) 1

redis 127.0.0.1:6379> smembers tech

1) "liming"

2) "wangwu"

3) "joe"

redis 127.0.0.1:6379> srem tech liming joe #移除多个member

(integer) 2

redis 127.0.0.1:6379> smembers tech

1) "wangwu"

#### c）smove

语法：smove source destination member

解释：将source中的member移动到destination

[plain] [view plain](http://blog.csdn.net/love__coder/article/details/8497497) [copy](http://blog.csdn.net/love__coder/article/details/8497497)

redis 127.0.0.1:6379> smembers tech   #smove前

1) "wangwu"

redis 127.0.0.1:6379> smembers stu

1) "zhangsan01"

2) "wangwu"

3) "zhangsan"

redis 127.0.0.1:6379> smove stu tech zhangsan  #将zhangsan 从stu移动到tech

(integer) 1

redis 127.0.0.1:6379> smembers stu #smove后

1) "zhangsan01"

2) "wangwu"

redis 127.0.0.1:6379> smembers tech

1) "wangwu"

2) "zhangsan"

### 4、其他

#### a）并集

语法：sunion key[key...]

解释：多个set的并集

[plain] [view plain](http://blog.csdn.net/love__coder/article/details/8497497) [copy](http://blog.csdn.net/love__coder/article/details/8497497)

redis 127.0.0.1:6379> smembers stu

1) "zhangsan01"

2) "wangwu"

redis 127.0.0.1:6379> sunion stu

1) "zhangsan01"

2) "wangwu"

redis 127.0.0.1:6379> smembers tech

1) "wangwu"

2) "zhangsan"

redis 127.0.0.1:6379> sunion stu tech

1) "zhangsan01"

2) "wangwu"

3) "zhangsan"

#### b）把并集结果存储到set

语法：sunionstore destination key [key ...]

解释：求多个set并集，并把结果存储到destination

[plain] [view plain](http://blog.csdn.net/love__coder/article/details/8497497) [copy](http://blog.csdn.net/love__coder/article/details/8497497)

redis 127.0.0.1:6379> sunionstore same stu tech #把stu tech并集结果存储在union

(integer) 3

redis 127.0.0.1:6379> smembers union

1) "zhangsan01"

2) "wangwu"

3) "zhangsan"

#### c）交集

语法：sinter key[key...]

解释：多个set的交集

[plain] [view plain](http://blog.csdn.net/love__coder/article/details/8497497) [copy](http://blog.csdn.net/love__coder/article/details/8497497)

redis 127.0.0.1:6379> smembers stu

1) "zhangsan01"

2) "wangwu"

redis 127.0.0.1:6379> smembers tech

1) "wangwu"

2) "zhangsan"

redis 127.0.0.1:6379> sinter stu tech

1) "wangwu"

#### d）把交集结果存储到指定set

语法：sinterstore destination key [key ...]

解释：把多个set的交集结果存储到destination

[plain] [view plain](http://blog.csdn.net/love__coder/article/details/8497497) [copy](http://blog.csdn.net/love__coder/article/details/8497497)

redis 127.0.0.1:6379> sinterstore inter stu tech

(integer) 1

redis 127.0.0.1:6379> smembers inter

1) "wangwu"

#### e) set中在其他set中不存在member

语法：sdiff key[key ...]

[plain] [view plain](http://blog.csdn.net/love__coder/article/details/8497497) [copy](http://blog.csdn.net/love__coder/article/details/8497497)

redis 127.0.0.1:6379>

redis 127.0.0.1:6379> smembers stu

1) "zhangsan01"

2) "wangwu"

redis 127.0.0.1:6379> smembers tech

1) "wangwu"

2) "zhangsan"

redis 127.0.0.1:6379> sdiff stu tech

1) "zhangsan01"

redis 127.0.0.1:6379> sdiff tech stu

1) "zhangsan"

#### f）把set中在其他set中不存在的member存储到新的set

语法：sdiffstore key[key...]

[plain] [view plain](http://blog.csdn.net/love__coder/article/details/8497497) [copy](http://blog.csdn.net/love__coder/article/details/8497497)

redis 127.0.0.1:6379> sdiffstore diff stu tech

(integer) 1

redis 127.0.0.1:6379> smembers diff

1) "zhangsan01"

## 有序集合

　　概念：它是在set的基础上增加了一个顺序属性，这一属性在添加修改元素的时候可以指定，每次指定后，zset会自动按新的值调整顺序。可以理解为有两列的mysql表，一列存储value，一列存储顺序，操作中key理解为zset的名字。

　　和set一样sorted，sets也是string类型元素的集合，不同的是每个元素都会关联一个double型的score。sorted set的实现是skip list和hash table的混合体。

　　当元素被添加到集合中时，一个元素到score的映射被添加到hash table中，所以给定一个元素获取score的开销是O(1)。另一个score到元素的映射被添加的skip list，并按照score排序，所以就可以有序地获取集合中的元素。添加、删除操作开销都是O(logN)和skip list的开销一致，redis的skip list 实现是双向链表，这样就可以逆序从尾部去元素。sorted set最经常使用方式应该就是作为索引来使用，我们可以把要排序的字段作为score存储，对象的ID当元素存储。

（1）zadd key score1 value1：添加元素

（2）zrange key start stop [withscore]：把集合排序后,返回名次[start,stop]的元素 默认是升续排列 withscores 是把score也打印出来

（3）zrank key member：查询member的排名（升序0名开始）

（4）zrangebyscore key min max [withscores] limit offset N：集合（升序）排序后取score在[min, max]内的元素，并跳过offset个，取出N个

（5）zrevrank key member：查询member排名（降序 0名开始）

（6）zremrangebyscore key min max：按照score来删除元素，删除score在[min, max]之间

（7）zrem key value1 value2：删除集合中的元素

（8）zremrangebyrank key start end：按排名删除元素，删除名次在[start, end]之间的

（9）zcard key：返回集合元素的个数

（10）zcount key min max：返回[min, max]区间内元素数量

（11）zinterstore dest numkeys key1[key2..] [WEIGHTS weight1 [weight2...]] [AGGREGATE SUM|MIN|MAX]

求key1，key2的交集，key1，key2的权值分别是weight1，weight2

聚合方法用 sum|min|max

聚合结果 保存子dest集合内

注意：weights,aggregate如何理解？

答：如果有交集，交集元素又有score，score怎么处理？aggregate num->score相加，min最小score，max最大score，另外可以通过weights设置不同的key的权重，交集时 score\*weight

redis支持有序集合，即sorted set。sorted set在[set](http://blog.csdn.net/love__coder/article/details/8497497)的基础上，增加了排序属性，是set的升级版。这里简要谈谈sorted set的常用函数：

### 1、增加

#### a）zadd

语法：zadd key score member [[score member] [score member] ...]

解释：增加一个或多个member【根据score排序】到有序集key中，如果member已经存在，只更新score。返回增加member个数，不包含已经存在的member

[plain] [view plain](http://blog.csdn.net/love__coder/article/details/8676223) [copy](http://blog.csdn.net/love__coder/article/details/8676223)

[root@xsf001 ~]# redis-cli

redis 127.0.0.1:6379> ZADD score 230 zhangsan

(integer) 1

redis 127.0.0.1:6379> zadd score 240 lisi 250 wangwu

(integer) 2

redis 127.0.0.1:6379> zadd score 100 liuli

redis 127.0.0.1:6379> zadd score 249 wangwu

(integer) 0

注意：有些redis版本，只能一次添加一个member

### 2、查询

#### a）zrange

语法：zrange key start stop [withscores]

解释：返回有续集key中指定范围【通过索引 start stop】的member[及其score]；

[plain] [view plain](http://blog.csdn.net/love__coder/article/details/8676223) [copy](http://blog.csdn.net/love__coder/article/details/8676223)

redis 127.0.0.1:6379> zrange score 0 -1  #所有元素

1) "liuli"

2) "zhangsan"

3) "lisi"

4) "wangwu"

redis 127.0.0.1:6379> zrange score 0 -1 withscores  #所有member及score

1) "liuli"

2) "100"

3) "zhangsan"

4) "230"

5) "lisi"

6) "240"

7) "wangwu"

8) "249"

redis 127.0.0.1:6379> zrange score 0 2 withscores  #前三个元素

1) "liuli"

2) "100"

3) "zhangsan"

4) "230"

5) "lisi"

6) "240"

redis 127.0.0.1:6379> zrange score 0 -2  #第一个元素到

1) "liuli"

2) "zhangsan"

3) "lisi"

注意：0，表示第一个元素，-1 表示最后一个元素，-2 表示倒数第二个元素

#### b）zcount

         语法：zcount key min max

解释：统计key中score值介于min 和max之间的member个数

[plain] [view plain](http://blog.csdn.net/love__coder/article/details/8676223) [copy](http://blog.csdn.net/love__coder/article/details/8676223)

redis 127.0.0.1:6379> zrange score 0 -1 withscores

1) "liuli"

2) "100"

3) "zhangsan"

4) "230"

5) "lisi"

6) "240"

7) "wangwu"

8) "249"

redis 127.0.0.1:6379> zcount score 230 240  # count（score >=230 && score <=240）

(integer) 2

#### c) zscore

语法：zscore key member

解释：返回有续集key中member的score

[plain] [view plain](http://blog.csdn.net/love__coder/article/details/8676223) [copy](http://blog.csdn.net/love__coder/article/details/8676223)

redis 127.0.0.1:6379> zscore score liuli

"100"

#### d）zrevrange

语法：zrevrange key start stop [withscores]

解释：返回有续集key中指定范围[通过索引start stop]的member[及score]，返回member根据score按降序排列

[plain] [view plain](http://blog.csdn.net/love__coder/article/details/8676223) [copy](http://blog.csdn.net/love__coder/article/details/8676223)

redis 127.0.0.1:6379> zrevrange score 0 -1 withscores

1) "wangwu"

2) "249"

3) "lisi"

4) "240"

5) "zhangsan"

6) "230"

7) "liuli"

8) "100"

redis 127.0.0.1:6379> zrevrange score 1 -2

1) "lisi"

2) "zhangsan"

注意：索引0 表示第一个元素，-1 最后一个元素，-2 倒数第二个元素；依次类推

#### e）zrangebyscore

语法：zrangebyscore key min max [withscores] [limit offset count]

解释：返回有续集key中，score大于等于min并且小于等于 max的member。返回结果按照score递增的次序排列。可选withscores决定返回结果集中仅仅返回member还是返回member和 score；可选参数limit 指定返回结果的数量区间。

[plain] [view plain](http://blog.csdn.net/love__coder/article/details/8676223) [copy](http://blog.csdn.net/love__coder/article/details/8676223)

redis 127.0.0.1:6379> zrangebyscore score 100 10000 withscores

1) "liuli"

2) "100"

3) "zhangsan"

4) "230"

5) "lisi"

6) "240"

7) "wangwu"

8) "249"

redis 127.0.0.1:6379> zrangebyscore score 100 10000  limit 1 3

1) "zhangsan"

2) "lisi"

3) "wangwu"

#### f）zrevrangebyscore

语法：zrevrangescore key max min [withscores] [limit offset count]

解释：返回有续集key中score<=max并且score>=min 的元素，返回结果根据score从大到小顺序排列。可选参数withscores决定结果集中是否包含score，可选参数limit 指定返回结果集范围。

[plain] [view plain](http://blog.csdn.net/love__coder/article/details/8676223) [copy](http://blog.csdn.net/love__coder/article/details/8676223)

redis 127.0.0.1:6379> zrevrangebyscore score 0 10000

(empty list or set)

redis 127.0.0.1:6379> zrevrangebyscore score 10000 0

1) "wangwu"

2) "lisi"

3) "zhangsan"

4) "liuli"

redis 127.0.0.1:6379> zrevrangebyscore score 10000 0 withscores limit 0, 2

1) "wangwu"

2) "249"

3) "lisi"

4) "240"

注意：max在min之前

#### g）zrank

语法：zrank key member

解释：根据score从低到高，返回member在有续集中的index

[plain] [view plain](http://blog.csdn.net/love__coder/article/details/8676223) [copy](http://blog.csdn.net/love__coder/article/details/8676223)

redis 127.0.0.1:6379> zrange score 0 -1

1) "liuli"

2) "zhangsan"

3) "lisi"

4) "wangwu"

redis 127.0.0.1:6379> zrank score liuli

(integer) 0

redis 127.0.0.1:6379> zrank score wangwu

(integer) 3

#### h）zrevrank

语法：zrevrank key member

解释：根据score从高到低排序，返回member在有序集key中的index

[plain] [view plain](http://blog.csdn.net/love__coder/article/details/8676223) [copy](http://blog.csdn.net/love__coder/article/details/8676223)

redis 127.0.0.1:6379> zrange score 0 -1

1) "liuli"

2) "zhangsan"

3) "lisi"

4) "wangwu"

redis 127.0.0.1:6379> zrevrank score liuli

(integer) 3

redis 127.0.0.1:6379> zrevrank score wangwu

(integer) 0

#### i)zcard

语法：zcard key

解释：返回有续集key的基数

[plain] [view plain](http://blog.csdn.net/love__coder/article/details/8676223) [copy](http://blog.csdn.net/love__coder/article/details/8676223)

redis 127.0.0.1:6379> zcard score

(integer) 4

redis 127.0.0.1:6379> zcard stdu  #有续集不存在返回0

(integer) 0

### 3、修改

#### a）zincrby

语法：zincrby key increment member

解释：有续集key的member增加增量increment，返回增加后的score

[plain] [view plain](http://blog.csdn.net/love__coder/article/details/8676223) [copy](http://blog.csdn.net/love__coder/article/details/8676223)

redis 127.0.0.1:6379> zscore score liuli

"100"

redis 127.0.0.1:6379> zincrby score 300 liuli

"400"

redis 127.0.0.1:6379> zscore score liuli

"400"

redis 127.0.0.1:6379> <span style="font-family: Arial, Helvetica, sans-serif;">zincrby </span>score 500 xie

"500"

注意：如果member在key中不存在，则新增个member

### 4、删除

#### a）zrem

语法：zrem key member [member ...]

解释：移除有续集中的一个或多个member，返回移除member的个数

[plain] [view plain](http://blog.csdn.net/love__coder/article/details/8676223) [copy](http://blog.csdn.net/love__coder/article/details/8676223)

redis 127.0.0.1:6379> zrange score 0 -1

1) "zhangsan"

2) "lisi"

3) "wangwu"

4) "liuli"

5) "xie"

redis 127.0.0.1:6379> zrem score xie wwww  #www不存在

(integer) 1

redis 127.0.0.1:6379> zrange score 0 -1

1) "zhangsan"

2) "lisi"

3) "wangwu"

4) "liuli"

注意:如果member不存在，则忽略

#### b）zremrangebyrank

语法：zremrangebyrank key start stop

解释：移除有续集中指定排名范围【start stop】的元素，返回移除元素个数

[plain] [view plain](http://blog.csdn.net/love__coder/article/details/8676223) [copy](http://blog.csdn.net/love__coder/article/details/8676223)

redis 127.0.0.1:6379> zrange score 0 -1

1) "zhangsan"

2) "lisi"

3) "wangwu"

4) "liuli"

redis 127.0.0.1:6379> zremrangebyrank score 0 1

(integer) 2

redis 127.0.0.1:6379> zrange score 0 -1

1) "wangwu"

2) "liuli"

注意：如果stop在start之前，则移除0个

#### c）zremrangebyscore

语法：zremrangebyscore key min max

解释：移除有续集中的member，移除member的score大于等于min小于等于max；返回移除元素个数

[plain] [view plain](http://blog.csdn.net/love__coder/article/details/8676223) [copy](http://blog.csdn.net/love__coder/article/details/8676223)

redis 127.0.0.1:6379> zrange score 0 -1 withscores

1) "wangwu"

2) "249"

3) "liuli"

4) "400"

redis 127.0.0.1:6379> zremrangebyscore score 248 250

(integer) 1

redis 127.0.0.1:6379> zrange score 0 -1 withscores

1) "liuli"

2) "400"

### 5、其他

#### a）zinterstore

语法：zinterstore destination numkeys  key[key...] [weights weight] [aggregate sum|min|max]

解释：统计多个有续集的交集，其中有续集key的个数必须以 numkeys参数指定，并将统计结果存储到destination。默认情况下，destination中元素的score是各个有续集key中元素的 score之和。使用weights为每个有续集指定个乘法因子，每个有续集的score在传递给集合函数（aggregate）之前，先乘以乘法因子。 如果没指定乘法因子weight，默认是1；使用aggregate选项指定交集的聚合方式。min，最小score，max，最大score，sum， 求和。

[plain] [view plain](http://blog.csdn.net/love__coder/article/details/8676223) [copy](http://blog.csdn.net/love__coder/article/details/8676223)

redis 127.0.0.1:6379> zrange score 0 -1 withscores

1) "lisi"

2) "20"

3) "liuli"

4) "400"

redis 127.0.0.1:6379> zrange score1 0 -1 withscores

1) "wangwu"

2) "20"

3) "liuli"

4) "30"

redis 127.0.0.1:6379> zinterstore dest 2 score score1 #交集求和 sum（score.member.score \* 1 + score1.member.score \* 1）

(integer) 1

redis 127.0.0.1:6379> zrange dest 0 -1 withscores

1) "liuli"

2) "430"

redis 127.0.0.1:6379> zinterstore dest 2 score score1 weights 1 2 # 交集求和，sum（score.member.score \* 1 + score1.member.score \* 2）

(integer) 1

redis 127.0.0.1:6379> zrange dest 0 -1 withscores

1) "liuli"

2) "460"

redis 127.0.0.1:6379> zinterstore dest 2 score score1 aggregate min # 交集求最小值，min（score.member.score \* 1 , score1.member.score \* 1）

(integer) 1

redis 127.0.0.1:6379> zrange dest 0 -1 withscores

1) "liuli"

2) "30"

redis 127.0.0.1:6379> zinterstore dest 2 score score1 aggregate max # 交集求最大值 max（score.member.score \* 1 , score1.member.score \* 1）

(integer) 1

redis 127.0.0.1:6379> zrange dest 0 -1 withscores

1) "liuli"

2) "400"

redis 127.0.0.1:6379> zinterstore dest 2 score score1 aggregate sum  # 交集求和，sum（score.member.score \* 1 , score1.member.score \* 1）

(integer) 1

redis 127.0.0.1:6379> zrange dest 0 -1 withscores

1) "liuli"

2) "430"

redis 127.0.0.1:6379> zinterstore dest 2 score score1 weights 1 2 aggregate min # 交集求最小值 min（score.member.score \* 1 , score1.member.score \* 2）

(integer) 1

redis 127.0.0.1:6379> zrange dest 0 -1 withscores

1) "liuli"

2) "60"

#### b）zunionstore

语法：zunionstore destination numkeys  key[key...] [weights weight] [aggregate sum|min|max]

解释：统计多个有续集的并集，其中有续集key的个数必须以 numkeys参数指定，并将统计结果存储到destination。。默认情况下，destination中元素的score是各个有续集key中元素 的score之和。weights指定乘法因子，aggregate指定聚合方式，详情，请参阅 zinterstore 中介绍

[plain] [view plain](http://blog.csdn.net/love__coder/article/details/8676223) [copy](http://blog.csdn.net/love__coder/article/details/8676223)

redis 127.0.0.1:6379> zrange score 0 -1 withscores

1) "lisi"

2) "20"

3) "liuli"

4) "400"

redis 127.0.0.1:6379> zrange score1 0 -1 withscores

1) "wangwu"

2) "20"

3) "liuli"

4) "30"

redis 127.0.0.1:6379> zunionstore dest 2 score score1 #交集求和 sum（score.member.score \* 1 , score1.member.score \* 1）

(integer) 3

redis 127.0.0.1:6379> zrange dest 0 -1 withscores

1) "lisi"

2) "20"

3) "wangwu"

4) "20"

5) "liuli"

6) "430"

redis 127.0.0.1:6379> zunionstore dest 2 score score1  weights 1 2 aggregate sum # 交集求和 sum（score.member.score \* 1 , score1.member.score \* 2）

(integer) 3

redis 127.0.0.1:6379> zrange dest 0 -1 withscores

1) "lisi"

2) "20"

3) "wangwu"

4) "40"

5) "liuli"

6) "460"

redis 127.0.0.1:6379> zunionstore dest 2 score score1  weights 1 2 aggregate max # 交集最大值 max（score.member.score \* 1 , score1.member.score \* 2）

(integer) 3

redis 127.0.0.1:6379> zrange dest 0 -1 withscores

1) "lisi"

2) "20"

3) "wangwu"

4) "40"

5) "liuli"

6) "400"

redis 127.0.0.1:6379> zunionstore dest 2 score score1  weights 2 4 aggregate min # 交集最小值 min（score.member.score \* 2 , score1.member.score \* 4）

(integer) 3

redis 127.0.0.1:6379> zrange dest 0 -1 withscores

1) "lisi"

2) "40"

3) "wangwu"

4) "80"

5) "liuli"

6) "120"

提醒:sorted set的所有函数，请用下面方式查看

[plain] [view plain](http://blog.csdn.net/love__coder/article/details/8676223) [copy](http://blog.csdn.net/love__coder/article/details/8676223)

[root@xsf001 ~]# redis-cli

redis 127.0.0.1:6379> help @sorted\_set

## 服务器相关命令

（1）ping：测定连接是否存活

（2）echo：在命令行打印一些内容

（3）select：选择数据库

（4）quit：退出连接

（5）dbsize：返回当前数据库中key的数目

（6）info：获取服务器的信息和统计

（7）monitor：实时转储收到的请求

（8）config get 配置项：获取服务器配置的信息

config set 配置项 值：设置配置项信息

（9）flushdb：删除当前选择数据库中所有的key

（10）flushall：删除所有数据库中的所有的key

（11）time：显示服务器时间，时间戳（秒），微秒数

（12）bgrewriteaof：后台保存rdb快照

（13）bgsave：后台保存rdb快照

（14）save：保存rdb快照

（15）lastsave：上次保存时间

（16）shutdown [save/nosave]

注意：如果不小心运行了flushall，立即shutdown nosave，关闭服务器，然后手工编辑aof文件，去掉文件中的flushall相关行，然后开启服务器，就可以倒回原来是数据。如果flushall之后，系统恰好bgwriteaof了，那么aof就清空了，数据丢失。

（17）showlog：显示慢查询

问：多慢才叫慢？

答：由slowlog-log-slower-than 10000，来指定（单位为微秒）

问：服务器存储多少条慢查询记录

答：由slowlog-max-len 128，来做限制