Redis

教学导航

教学目标	
教学方法	

一、Redis 简介

1. 关于关系型数据库和 nosql 数据库

关系型数据库是基于关系表的数据库，最终会将数据持久化到磁盘上，而 nosql 数据库是基于特殊的结构，并将数据存储到内存的数据库。从性能上而言，nosql 数据库要优于关系型数据库，从安全性上而言关系型数据库要优于 nosql 数据库，所以在实际开发中一个项目中 nosql 和关系型数据库会一起使用 达到性能和安全性的双保证。

2. 为什么要使用 Redis

3．redis 在 Linux 上的安装

1）安装 redis 编译的 c 环境，yum install gcc-c++

2）将 redis-2.6.16.tar.gz 上传到 Linux 系统中

3）解压到/usr/local 下　tar -xvf redis-2.6.16.tar.gz -C /usr/local

4）进入 redis-2.6.16 目录 使用 make 命令编译 redis

5）在 redis-2.6.16 目录中 使用 make PREFIX=/usr/local/redis install 命令安装 redis 到/usr/local/redis 中

6）拷贝 redis-2.6.16 中的 redis.conf 到安装目录 redis 中

7）启动 redis 在 bin 下执行命令 redis-server redis.conf

8）如需远程连接 redis，需配置 redis 端口 6379 在 linux 防火墙中开发

/sbin/iptables -I INPUT -p tcp --dport 6379 -j ACCEPT

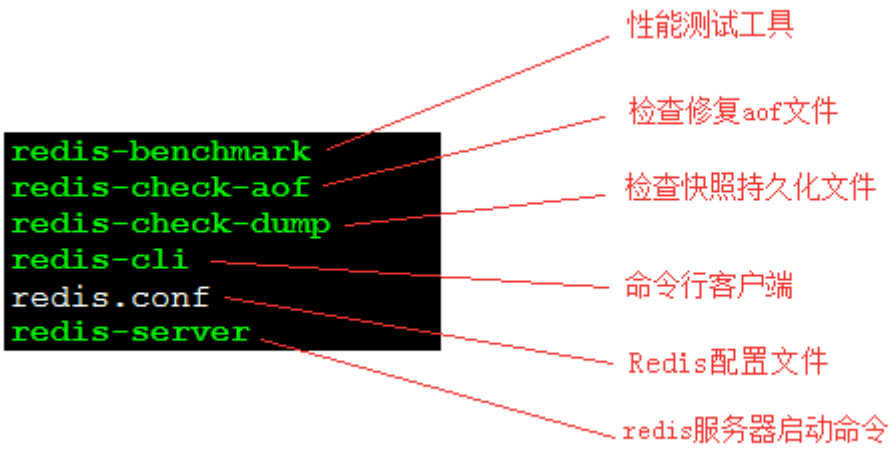
/etc/rc.d/init.d/iptables save

启动后看到如上欢迎页面，但此窗口不能关闭，窗口关闭就认为 redis 也关闭了(类似 Tomcat 通过 bin 下的 startup.bat 的方式)

解决方案：可以通过修改配置文件 配置 redis 后台启动，即服务器启动了但不会穿件控制台窗口

将 redis.conf 文件中的 daemonize 从 false 修改成 true 表示后台启动

使用命令查看 6379 端口是否启动 ps -ef | grep redis



二、使用 java 去操作 Redis

三、Redis 的常用命令

redis 是一种高级的 key-value 的存储系统

其中的 key 是字符串类型，尽可能满足如下几点：

1）key 不要太长，最好不要操作 1024 个字节，这不仅会消耗内存还会降低查找效率

2）key 不要太短，如果太短会降低 key 的可读性

3）在项目中，key 最好有一个统一的命名规范（根据企业的需求）

其中 value 支持五种数据类型：

1）字符串型 string

2）字符串列表 lists

3）字符串集合 sets

4）有序字符串集合 sorted sets

5）哈希类型 hashs

我们对 Redis 的学习，主要是对数据的存储，下面将来学习各种 Redis 的数据类型的存储操作：

1．存储字符串 string

字符串类型是 Redis 中最为基础的数据存储类型，它在 Redis 中是二进制安全的，这便意味着该类型可以接受任何格式的数据，如 JPEG 图像数据或 Json 对象描述信息等。在 Redis 中字符串类型的 Value 最多可以容纳的数据长度是 512M

key1	value1
key2	value2
key3	value3

1）**set key value**：设定 key 持有指定的字符串 value，如果该 key 存在则进行覆盖操作。总是返回”OK”

2）**get key**：获取 key 的 value。如果与该 key 关联的 value 不是 String 类型，redis 将返回错误信息，因为 get 命令只能用于获取 String value；如果该 key 不存在，返回 null。

```
redis 127.0.0.1:6379> set name zhangsan
OK
redis 127.0.0.1:6379> get name
"zhangsan"
redis 127.0.0.1:6379> set name lisi
OK
redis 127.0.0.1:6379> get name
"lisi"
redis 127.0.0.1:6379>
```

3）**getset key value**：先获取该 key 的值，然后在设置该 key 的值。

```
redis 127.0.0.1:6379> getset name wangwu
"lisi"
redis 127.0.0.1:6379> get name
"wangwu"
redis 127.0.0.1:6379>
```

4）**incr key**：将指定的 key 的 value 原子性的递增 1.如果该 key 不存在，其初始值为 0，在 incr 之后其值为 1。如果 value 的值不能转成整型，如 hello，该操作将执行失败并返回相应的错误信息。

5）**decr key**：将指定的 key 的 value 原子性的递减 1.如果该 key 不存在，其初始值为 0，在 incr 之后其值为-1。如果 value 的值不能转成整型，如 hello，该操作将执行失败并返回相应的错误信息。

```
redis 127.0.0.1:6379> incr num
(integer) 1
redis 127.0.0.1:6379> get num
"1"
redis 127.0.0.1:6379> incr num
(integer) 2
redis 127.0.0.1:6379> get num
"2"
redis 127.0.0.1:6379> decr num
(integer) 1
redis 127.0.0.1:6379> get num
"1"
redis 127.0.0.1:6379>
```

6）**incrby key increment**：将指定的 key 的 value 原子性增加 increment，如果该 key 不存在，器初始值为 0，在 incrby 之后，该值为 increment。如果该值不能转成整型，如 hello 则失败并返回错误信息

7）**decrby key decrement**：将指定的 key 的 value 原子性减少 decrement，如果该 key 不存在，器初始值为 0，在 decrby 之后，该值为 decrement。如果该值不能转成整型，如 hello 则失败并返回错误信息

```
redis 127.0.0.1:6379> incrby num 5
(integer) 6
redis 127.0.0.1:6379> incrby num 5
(integer) 11
redis 127.0.0.1:6379> get num
"11"
redis 127.0.0.1:6379> decrby num 5
(integer) 6
redis 127.0.0.1:6379> decrby num 5
(integer) 1
redis 127.0.0.1:6379> get num
"1"
redis 127.0.0.1:6379>
```

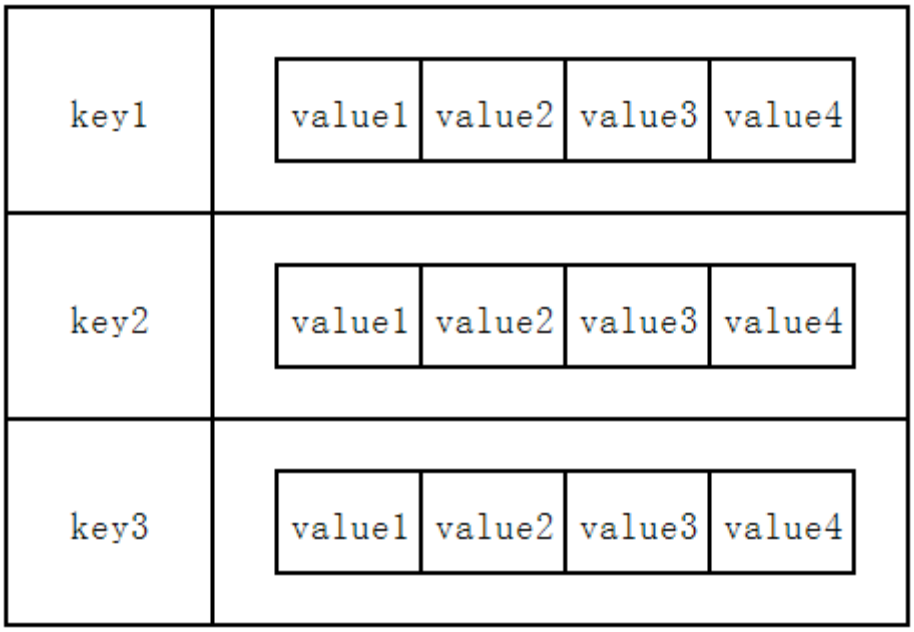
8）**append key value**：如果该 key 存在，则在原有的 value 后追加该值；如果该 key不存在，则重新创建一个 key/value

```
redis 127.0.0.1:6379> append addr beijing
(integer) 7
redis 127.0.0.1:6379> get addr
"beijing"
redis 127.0.0.1:6379> append addr tianjin
(integer) 14
redis 127.0.0.1:6379> get addr
"beijingtianjin"
redis 127.0.0.1:6379>
```

2．存储 lists 类型

在 Redis 中，List 类型是按照插入顺序排序的字符串链表。和数据结构中的普通链表一样，我们可以在其头部(left)和尾部(right)添加新的元素。在插入时，如果该键并不存在，Redis 将为该键创建一个新的链表。与此相反，如果链表中所有的元素均被移除，那么该键也将会被从数据库中删除。List 中可以包含的最大元素数量是4294967295。

从元素插入和删除的效率视角来看，如果我们是在链表的两头插入或删除元素，这将会是非常高效的操作，即使链表中已经存储了百万条记录，该操作也可以在常量时间内完成。然而需要说明的是，如果元素插入或删除操作是作用于链表中间，那将会是非常低效的。相信对于有良好数据结构基础的开发者而言，这一点并不难理解。



1）**lpush key value1 value2...**：在指定的 key 所关联的 list 的头部插入所有的

values，如果该 key 不存在，该命令在插入的之前创建一个与该 key 关联的空链表，之后再向该链表的头部插入数据。插入成功，返回元素的个数。

2）**rpush key value1、value2...**：在该 list 的尾部添加元素

3）**lrange key start end**：获取链表中从 start 到 end 的元素的值，start、end 可为负数，若为-1 则表示链表尾部的元素，-2 则表示倒数第二个，依次类推...

```
redis 127.0.0.1:6379> lpush mylist jerry lucy
(integer) 2
redis 127.0.0.1:6379> lrange mylist 0 -1
1) "lucy"
2) "jerry"
redis 127.0.0.1:6379> rpush mylist tom
(integer) 3
redis 127.0.0.1:6379> lrange mylist 0 -1
1) "lucy"
2) "jerry"
3) "tom"
redis 127.0.0.1:6379>
```

4）**lpushx key value**：仅当参数中指定的 key 存在时（如果与 key 管理的 list 中没有值时，则该 key 是不存在的）在指定的 key 所关联的 list 的头部插入 value。

5）**rpushx key value**：在该 list 的尾部添加元素

```
redis 127.0.0.1:6379> lrange mylist 0 -1
1) "lucy"
2) "jerry"
3) "tom"
redis 127.0.0.1:6379> lpushx mylist rose
(integer) 4
redis 127.0.0.1:6379> rpushx mylist mary
(integer) 5
redis 127.0.0.1:6379> lrange mylist 0 -1
1) "rose"
2) "lucy"
3) "jerry"
4) "tom"
5) "mary"
redis 127.0.0.1:6379>
```

6）**lpop key**：返回并弹出指定的 key 关联的链表中的第一个元素，即头部元素。

7）**rpop key**：从尾部弹出元素。

```
redis 127.0.0.1:6379> lrange mylist 0 -1
1) "rose"
2) "lucy"
3) "jerry"
4) "tom"
5) "mary"
redis 127.0.0.1:6379> lpop mylist
"rose"
redis 127.0.0.1:6379> rpop mylist
"mary"
redis 127.0.0.1:6379> lrange mylist 0 -1
1) "lucy"
2) "jerry"
3) "tom"
```

8）**rpoplpush resource destination**：将链表中的尾部元素弹出并添加到头部

```
redis 127.0.0.1:6379> lrange mylist 0 -1
1) "lucy"
2) "jerry"
3) "tom"
redis 127.0.0.1:6379> rpoplpush mylist mylist2
"tom"
redis 127.0.0.1:6379> rpoplpush mylist mylist2
"jerry"
redis 127.0.0.1:6379> lrange mylist 0 -1
1) "lucy"
redis 127.0.0.1:6379> lrange mylist2 0 -1
1) "jerry"
2) "tom"
redis 127.0.0.1:6379>
```

9）**llen key**：返回指定的 key 关联的链表中的元素的数量。

```
redis 127.0.0.1:6379> del mylist
(integer) 1
redis 127.0.0.1:6379> lpush mylist tom jerry lucy rose mary
(integer) 5
redis 127.0.0.1:6379> lrange mylist 0 -1
1) "mary"
2) "rose"
3) "lucy"
4) "jerry"
5) "tom"
redis 127.0.0.1:6379> llen mylist
(integer) 5
```

10）**lset key index value**：设置链表中的 index 的脚标的元素值，0 代表链表的头元 素，-1 代表链表的尾元素。

```
redis 127.0.0.1:6379> lrange mylist 0 -1
1) "mary"
2) "rose"
3) "lucy"
4) "jerry"
5) "tom"
redis 127.0.0.1:6379> lset mylist 2 davy
OK
redis 127.0.0.1:6379> lrange mylist 0 -1
1) "mary"
2) "rose"
3) "davy"
4) "jerry"
5) "tom"
redis 127.0.0.1:6379> lset mylist -2 peter
OK
redis 127.0.0.1:6379> lrange mylist 0 -1
1) "mary"
2) "rose"
3) "davy"
4) "peter"
5) "tom"
redis 127.0.0.1:6379>
```

11）**lrem key count value**：删除 count 个值为 value 的元素，如果 count 大于 0，从头向尾遍历并删除 count 个值为 value 的元素，如果 count 小于 0，则从尾向头遍历并删除。如果 count 等于 0，则删除链表中所有等于 value 的元素。

```
redis 127.0.0.1:6379> lrange mylist 0 -1
1) "mary"
2) "rose"
3) "davy"
4) "peter"
5) "tom"
redis 127.0.0.1:6379> lrem mylist 0 peter
(integer) 1
redis 127.0.0.1:6379> lrange mylist 0 -1
1) "mary"
2) "rose"
3) "davy"
4) "tom"
redis 127.0.0.1:6379>
```

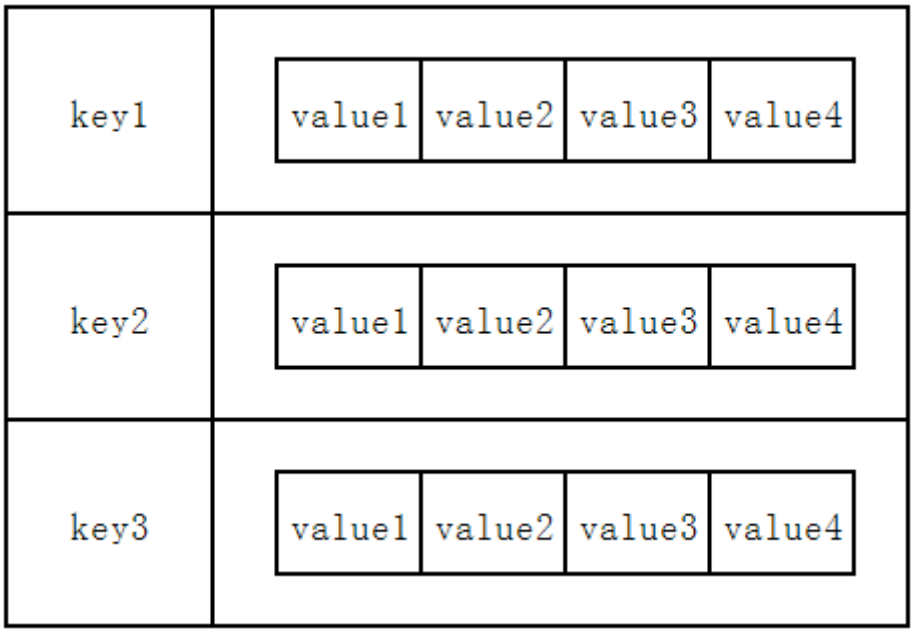
12) **linsert key before|after pivot value**：在 pivot 元素前或者后插入 value 这个元素。

```
redis 127.0.0.1:6379> lrange mylist 0 -1
1) "mary"
2) "rose"
3) "davy"
4) "tom"
redis 127.0.0.1:6379> linsert mylist before rose jerry
(integer) 5
redis 127.0.0.1:6379> lrange mylist 0 -1
1) "mary"
2) "jerry"
3) "rose"
4) "davy"
5) "tom"
redis 127.0.0.1:6379>
```

3．存储 sets 类型

在 Redis 中，我们可以将 Set 类型看作为没有排序的字符集合，和 List 类型一样，我

们也可以在该类型的数据值上执行添加、删除或判断某一元素是否存在等操作。需要说明的是，这些操作的时间是常量时间。Set 可包含的最大元素数是 4294967295。和 List 类型不同的是，Set 集合中不允许出现重复的元素。和 List 类型相比，Set 类型在功能上还存在着一个非常重要的特性，即在服务器端完成多个 Sets 之间的聚合计算操作，如 unions、intersections 和 differences。由于这些操作均在服务端完成，因此效率极高，而且也节省了大量的网络 IO 开销



- 1）sadd key value1、value2...：向 set 中添加数据，如果该 key 的值已有则不会重复添加
- 2）smembers key：获取 set 中所有的成员
- 3）scard key：获取 set 中成员的数量

```
redis 127.0.0.1:6379> sadd myset tom lucy jerry rose
(integer) 4
redis 127.0.0.1:6379> smembers myset
1) "rose"
2) "lucy"
3) "jerry"
4) "tom"
redis 127.0.0.1:6379> scard myset
(integer) 4
redis 127.0.0.1:6379>
```

- 4）sismember key member：判断参数中指定的成员是否在该 set 中，1 表示存在，0 表示不存在或者该 key 本身就不存在
- 5）srem key member1、member2...：删除 set 中指定的成员

```
redis 127.0.0.1:6379> sismember myset jerry
(integer) 1
redis 127.0.0.1:6379> sismember myset davy
(integer) 0
redis 127.0.0.1:6379> srem myset jerry
(integer) 1
redis 127.0.0.1:6379> smembers myset
1) "rose"
2) "lucy"
3) "tom"
redis 127.0.0.1:6379>
```

6）srandmember key：随机返回 set 中的一个成员

```
redis 127.0.0.1:6379> srandmember myset
"rose"
redis 127.0.0.1:6379> srandmember myset
"tom"
redis 127.0.0.1:6379> srandmember myset
"rose"
redis 127.0.0.1:6379> srandmember myset
"lucy"
```

✦7）sdiff sdiff key1 key2：返回 key1 与 key2 中相差的成员，而且与 key 的顺序有关。即返回差集。

```
redis 127.0.0.1:6379> smembers myset
1) "rose"
2) "lucy"
3) "tom"
redis 127.0.0.1:6379> smembers myset2
1) "lucy"
2) "zhangsan"
3) "tom"
4) "lisi"
5) "wangwu"
redis 127.0.0.1:6379> sdiff myset myset2
1) "rose"
redis 127.0.0.1:6379> sdiff myset2 myset
1) "wangwu"
2) "lisi"
3) "zhangsan"
redis 127.0.0.1:6379>
```

✦8）sdiffstore destination key1 key2：将 key1、key2 相差的成员存储在 destination 上

```
redis 127.0.0.1:6379> smembers myset
1) "rose"
2) "lucy"
3) "tom"
redis 127.0.0.1:6379> smembers myset2
1) "lucy"
2) "zhangsan"
3) "tom"
4) "lisi"
5) "wangwu"
redis 127.0.0.1:6379> sdiffstore myset3 myset myset2
(integer) 1
redis 127.0.0.1:6379> smembers myset3
1) "rose"
redis 127.0.0.1:6379>
```

✦9）sinter key[key1,key2…]：返回交集。

✦10）sinterstore destination key1 key2：将返回的交集存储在 destination 上

```
redis 127.0.0.1:6379> smembers myset
1) "rose"
2) "lucy"
3) "tom"
redis 127.0.0.1:6379> smembers myset2
1) "lucy"
2) "zhangsan"
3) "tom"
4) "lisi"
5) "wangwu"
redis 127.0.0.1:6379> sinter myset myset2
1) "lucy"
2) "tom"
redis 127.0.0.1:6379> sinterstore myset4 myset myset2
(integer) 2
redis 127.0.0.1:6379> smembers myset4
1) "lucy"
2) "tom"
redis 127.0.0.1:6379>
```

✦11）sunion key1、key2：返回并集。

```
redis 127.0.0.1:6379> smembers myset
1) "rose"
2) "lucy"
3) "tom"
redis 127.0.0.1:6379> smembers myset2
1) "lucy"
2) "wangwu"
3) "lisi"
4) "tom"
5) "zhangsan"
redis 127.0.0.1:6379> sunion myset myset2
1) "tom"
2) "rose"
3) "lucy"
4) "wangwu"
5) "lisi"
6) "zhangsan"
redis 127.0.0.1:6379>
```

✦12）sunionstore destination key1 key2：将返回的并集存储在 destination 上

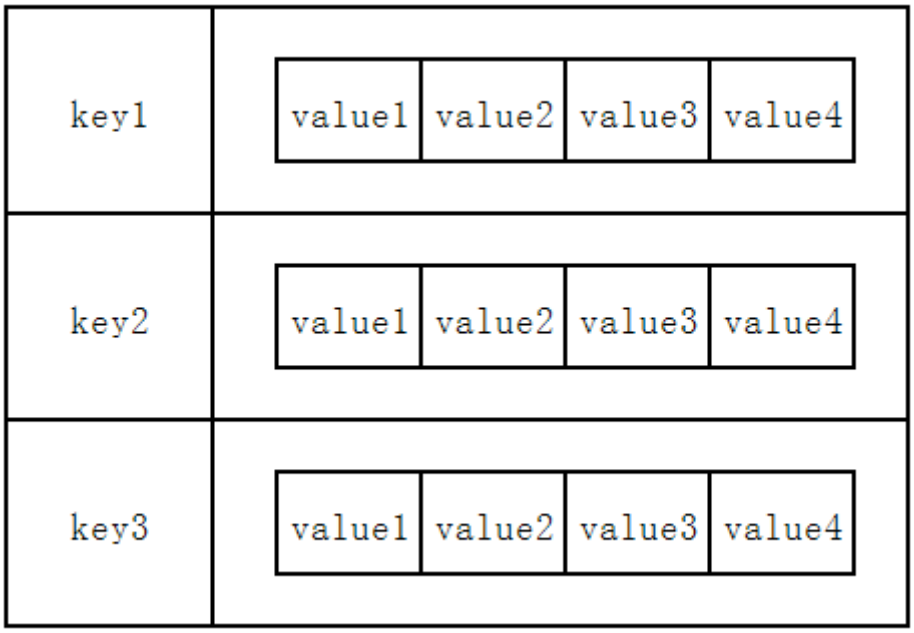
```
redis 127.0.0.1:6379> sunionstore myset5 myset myset2
(integer) 6
redis 127.0.0.1:6379> smembers myset5
1) "tom"
2) "rose"
3) "lucy"
4) "wangwu"
5) "lisi"
6) "zhangsan"
```

4．存储 sortedset

Sorted-Sets 和 Sets 类型极为相似，它们都是字符串的集合，都不允许重复的成员出现在一个 Set 中。它们之间的主要差别是 Sorted-Sets 中的每一个成员都会有一个分数(score)与之关联，Redis 正是通过分数来为集合中的成员进行从小到大的排序。然而需要额外指出的是，尽管 Sorted-Sets 中的成员必须是唯一的，但是分数(score)却是可以重复的。

在 Sorted-Set 中添加、删除或更新一个成员都是非常快速的操作，其时间复杂度为集合中成员数量的对数。由于 Sorted-Sets 中的成员在集合中的位置是有序的，因此，即便是访问位于集合中部的成员也仍然是非常高效的。事实上，Redis 所具有的这一特征在很多其它类型的数据库中是很难实现的，换句话说，在该点上要想达到和 Redis 同样的高效，在其它数据库中进行建模是非常困难的。

例如：游戏排名、微博热点话题等使用场景。



- 1）**zadd key score member score2 member2 …** ：将所有成员以及该成员的分数存放到 sorted-set 中
- 2）**zcard key**：获取集合中的成员数量

```
redis 127.0.0.1:6379> zadd mysort 10 zhangsan 30 lisi 20 wuangwu
(integer) 3
redis 127.0.0.1:6379> zcard mysort
(integer) 3
redis 127.0.0.1:6379>
```

- 3）**zcount key min max**：获取分数在[min,max]之间的成员
- zincrby key increment member：设置指定成员的增加的分数。
- zrange key start end [withscores]：获取集合中脚标为 start-end 的成员，[withscores]参数表明返回的成员包含其分数。
- zrangebyscore key min max [withscores] [limit offset count]：返回分数在[min,max]的成员并按照分数从低到高排序。[withscores]：显示分数；[limit offset count]：offset，表明从脚标为 offset 的元素开始并返回 count 个成员。
- zrank key member：返回成员在集合中的位置。
- zrem key member[member…]：移除集合中指定的成员，可以指定多个成员。
- zscore key member：返回指定成员的分数

5．存储 hash

Redis 中的 Hashes 类型可以看成具有 String Key 和 String Value 的 map 容器。所以该类型非常适合于存储值对象的信息。如 Username、Password 和 Age 等。如果 Hash 中包含很少的字段 那么该类型的数据也将仅占用很少的磁盘空间。每一个 Hash 可以存储 4294967295 个键值对。

key1	filed1 value1 filed2 value2 filed3 value3
key2	filed1 value1 filed2 value2 filed3 value3
key3	filed1 value1 filed2 value2 filed3 value3

1）**hset key field value**：为指定的 key 设定 field/value 对（键值对）。

2）**hgetall key：**获取 key 中的所有 filed-vaule

```
redis 127.0.0.1:6379> hset myset1 name zhangsan
(integer) 1
redis 127.0.0.1:6379> hset myset1 age 28
(integer) 1
redis 127.0.0.1:6379> hset myset1 addr beijing
(integer) 1
redis 127.0.0.1:6379> hgetall myset1
1) "name"
2) "zhangsan"
3) "age"
4) "28"
5) "addr"
6) "beijing"
redis 127.0.0.1:6379>
```

✦3）**hget key field**：返回指定的 key 中的 field 的值

```
redis 127.0.0.1:6379> hget myset1 addr
"beijing"
redis 127.0.0.1:6379> hget myset1 age
"28"
redis 127.0.0.1:6379>
```

✦4）**hmset key fields**：设置 key 中的多个 filed/value

✦5）**hmget key fileds**：获取 key 中的多个 filed 的值

✦6）**hexists key field**：判断指定的 key 中的 filed 是否存在

✦7）**hlen key**：获取 key 所包含的 field 的数量

✦8）**hincrby key field increment**：设置 key 中 filed 的值增加 increment，如：age 增加 20

✦

四、Redis 的通用操作(见 pdf 文档)

五、Redis 的特性(见 pdf 文档)

六、Redis 的事务(见 pdf 文档)

七、Redis 的持久化(见 pdf 文档)

总结：

1、nosql
2、redis 安装----linux（重点）
3、jedis（重点）
4、redis 的数据操作类型 5 中（了解）--- string 和 hash
5、redis 的其他