redis五种数据类型：String，Hash，List，Set，ZSet。五种数据类型各有优点，各司其职。最后两种数据类型Set和ZSet基本上是一种，ZSet就是给Set加了个排序而已（相当于加了个属性）。

Set ZSet

Set:无序集合，通过hash table实现，添加，删除，查找复杂度都是O(1)

命令：

sadd myset hello 向myset集合添加hello元素（不能重复，重复返回0）

smembers myset 查看myset集合中所有元素

srem myset hello 删除myset中的hello元素

spop myset 从myset中随机删除一个元素

sdiff myset1 myset2 集合myset1和集合myset2的差集（1有而2没有的）

sdiffstore myset3 myset1 myset2 将myset1和myset2的差集存入myset3

sinter myset1 myset2 获取myset1和myset2的交集

sinterstore myset3 myset1 myset2 将myset1和myset2的交集存入myset3

sunion myset1 myset2 获取myset1和myset2的并集

sunionstore myset3 myset1 myset2 将myset1和myset2的并集存入myset3

smove myset1 myset2 jeyson 将jeyson元素从myset1移到myset2中

scard myset1 获取myset1中元素个数

sismember myset1 hello 测试hello是否是myset1的元素，是返回1 否则0

srandmember myset1 随机返回myset1一个元素，但不删除

ZSet:有序集合，为Set的升级版本，增加了一个顺序属性

命令：

zadd myset 1 one 向myset集合添加顺序为1的元素one

更新顺序也是这个命令zadd myset 2 one 把one的顺序号改为2

zrange myset 0 -1 withscores 获取集合中所有元素（带序号）(升序)

zrevrange myset 0 -1 withscores 获取集合中所有元素（带序号）(降序）

zrange myset 0 -1获取集合中所有元素（不带序号）(升序)

zrem myset one 删除集合中one元素

zincrby myset 3 three 将three的顺序号增加3（如果没有three则创建three，顺序为3）

zrank myset one 返回one的索引（不是顺序，索引从0开始）（升序）

zrevrank myset one返回one的索引（从大到小）（降序）

zrangebyscore myset 2 5 withscores 获取顺序为2-5的元素

zcount myset 2 5 返回集合中顺序在2-5的元素数量

zcard myset 返回集合中所有元素个数

zremrangebyrank myset 2 5 删除指定索引区间内（2-5）的元素

zremrangebyscore myset 2 5删除指定顺序区间内（2-5）的元素

**Hash**

HSET

HSET key field value

将哈希表key中的域field的值设为value。

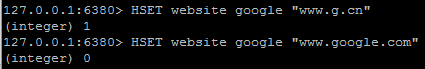
如果key不存在，一个新的哈希表被创建并进行HSET操作。

如果域field已经存在于哈希表中，旧值将被覆盖。

返回值：

\*如果field是哈希表中的一个新建域，并且值设置成功，返回1。

如果哈希表中域field已经存在且旧值已被新值覆盖，返回0。\*



HSETNX

HSETNX key field value （not exist）

将哈希表key中的域field的值设置为value，当且仅当域field不存在。

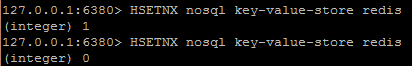
若域field已经存在，该操作无效。

如果key不存在，一个新哈希表被创建并执行HSETNX命令。

返回值：

\*设置成功，返回1。

如果给定域已经存在且没有操作被执行，返回0。\*



HMSET

HMSET key field value [field value …]

同时将多个field - value(域-值)对设置到哈希表key中。

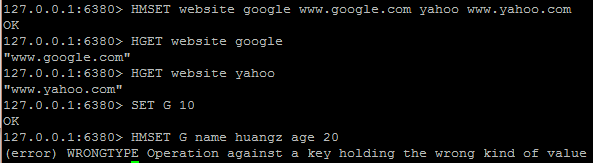
此命令会覆盖哈希表中已存在的域。

如果key不存在，一个空哈希表被创建并执行HMSET操作。

返回值：

\*如果命令执行成功，返回OK。

当key不是哈希表(hash)类型时，返回一个错误。\*



HGET

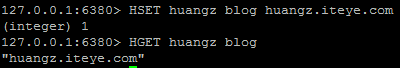
HGET key field

返回哈希表key中给定域field的值。

返回值：

\*给定域的值。

当给定域不存在或是给定key不存在时，返回nil。\*



HMGET

HMGET key field [field …]

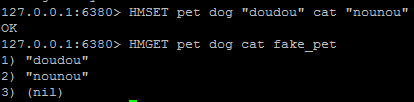
返回哈希表key中，一个或多个给定域的值。

如果给定的域不存在于哈希表，那么返回一个nil值。

因为不存在的key被当作一个空哈希表来处理，所以对一个不存在的key进行HMGET操作将返回一个只带有nil值的表。

返回值：

一个包含多个给定域的关联值的表，表值的排列顺序和给定域参数的请求顺序一样。



HGETALL

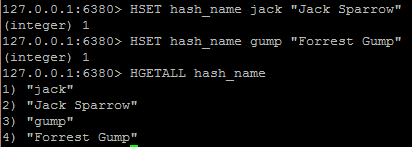
HGETALL key

返回哈希表key中，所有的域和值。

在返回值里，紧跟每个域名(field name)之后是域的值(value)，所以返回值的长度是哈希表大小的两倍。

返回值：

以列表形式返回哈希表的域和域的值。 若key不存在，返回空列表。



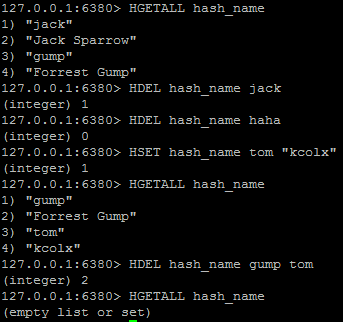
HDEL

HDEL key field [field …]

删除哈希表key中的一个或多个指定域，不存在的域将被忽略。

返回值:

被成功移除的域的数量，不包括被忽略的域。



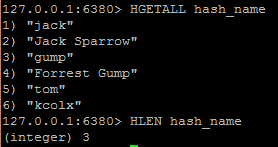
HLEN

HLEN key返回哈希表key中域的数量。

返回值：

\*哈希表中域的数量。

当key不存在时，返回0。\*



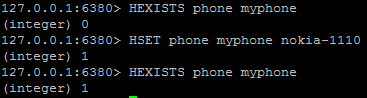
HEXISTS

HEXISTS key field查看哈希表key中，给定域field是否存在。

返回值：

\*如果哈希表含有给定域，返回1。

如果哈希表不含有给定域，或key不存在，返回0。\*



HINCRBY

HINCRBY key field increment

为哈希表key中的域field的值加上增量increment。

增量也可以为负数，相当于对给定域进行减法操作。

如果key不存在，一个新的哈希表被创建并执行HINCRBY命令。

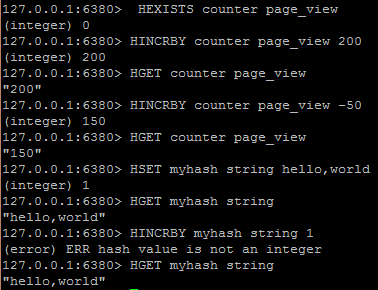
如果域field不存在，那么在执行命令前，域的值被初始化为0。

对一个储存字符串值的域field执行HINCRBY命令将造成一个错误。

本操作的值限制在64位(bit)有符号数字表示之内。

返回值：

执行HINCRBY命令之后，哈希表key中域field的值。



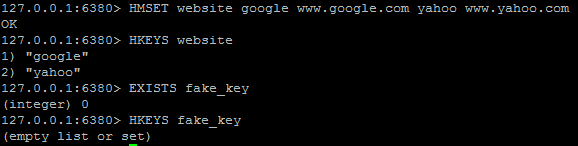
HKEYS

HKEYS key返回哈希表key中的所有域。

返回值：

\*一个包含哈希表中所有域的表。

当key不存在时，返回一个空表。\*



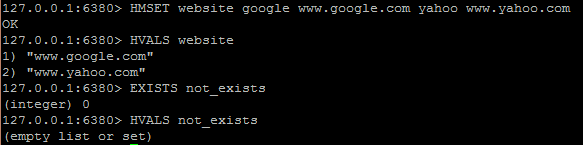
HVALS

HVALS key返回哈希表key中的所有值。

返回值：

\*一个包含哈希表中所有值的表。

当key不存在时，返回一个空表\*。



排序

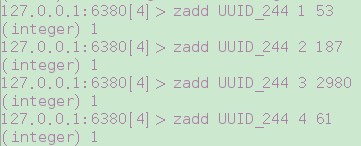
对zset进行排序很简单

zrange UUID\_244 0 -1 withscores 对UUID\_244的set按照scores进行升序排序,并取出对应的scores

zrevrange UUID\_244 0 -1 withscores 对UUID\_244的set按照scores进行降序排序,并取出对应的scores

zrevrange UUID\_244 0 -1 对UUID\_244的set按照scores进行降序排序，只显示value

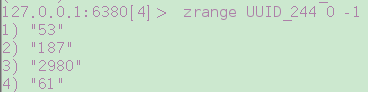
zrange UUID\_244 0 -1 withscores 对UUID\_244的set按照scores进行降序排序，只显示value







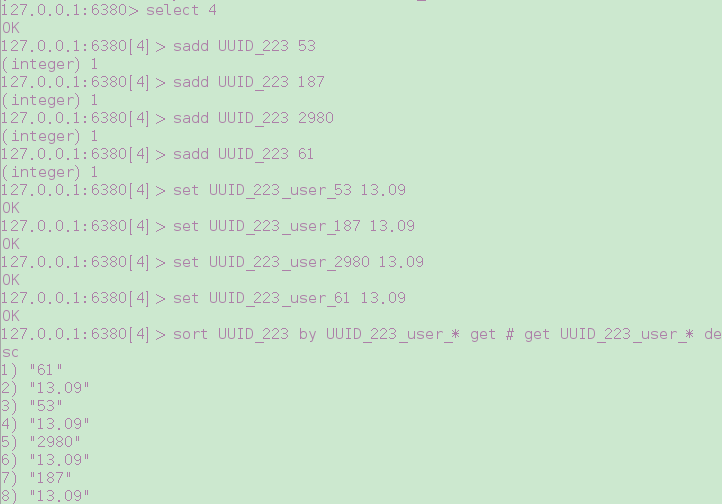




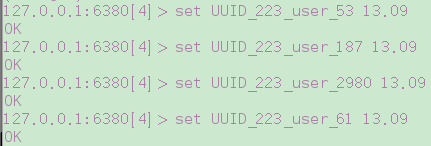
当key-value中的key中的key与set、zset的scores在名称上有关连时，可以根据key-value中的key对value进行排序，取出set、zset中与关连的key 对应的scores

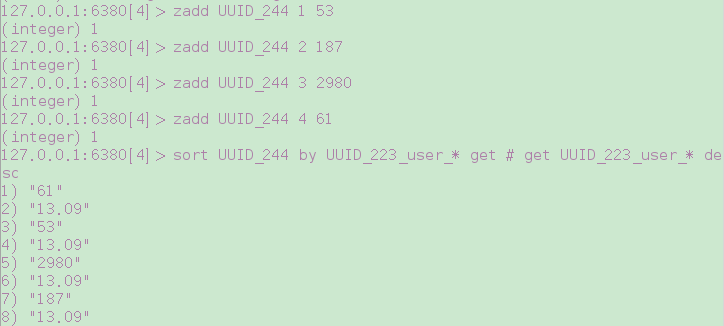
例1：UUID——223是set，有4个scores，分别关连每个scores，创建了4和key-value对，

现在对key-value对中的value进行排序，并取出与key关连的scores



例2：仍然使用上面的4个key-value对，再建一个UUID\_244的zset，zset的scores与4个key的名称有关连。

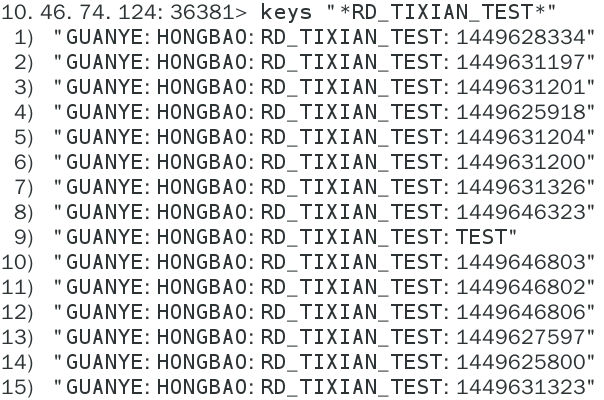




小结：redis只能通过key进行排序，操作对象不管是set还是zset，排序方式不管是升序还是降序，排序结果都与创建该元素的时间无关。

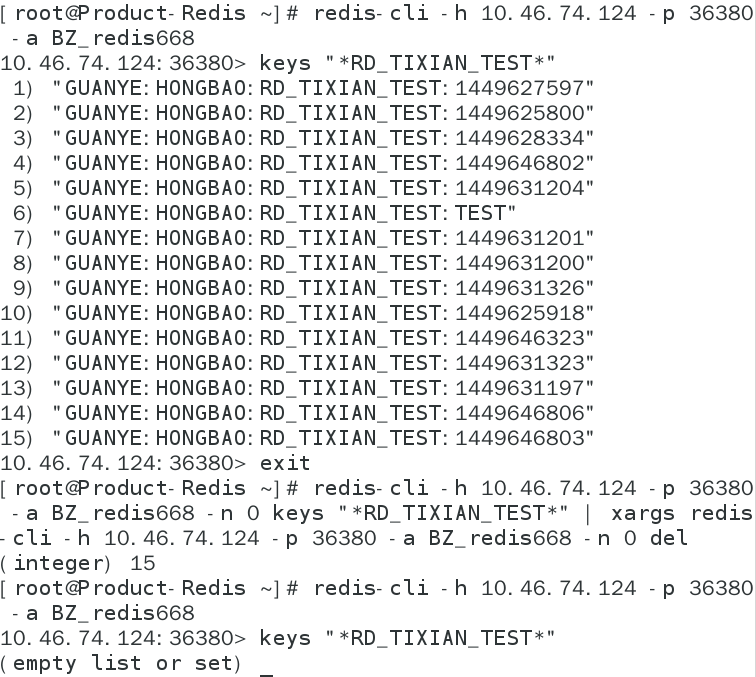
模糊查询

keys "\*RD\_TIXIAN\_TEST\*"



批量删除

redis-cli -h 10.46.74.124 -p 36380 -a BZ\_redis668 -n 0 keys "\*RD\_TIXIAN\_TEST\*" | xargs redis-cli -h 10.46.74.124 -p 36380 -a BZ\_redis668 -n 0 del



**Redis命令总结**

|  |  |  |
| --- | --- | --- |
| KEYS pattern | 获得符合规则的健名列表 | |
|  | EXISTS key | 判断一个键是否存在 |
|  | DEL key [key …] | 删除键（不支持通配符） |
|  | TYPE key | 获得键值的数据类型。返回值可能是string（字符串类型）、hash（散列类型）、list（列表类型）、set（集合类型）、zset（有序集合类型） |
| 字符串操作 | SET key value  GET key | 字符串类型赋值和取值 |
|  | INCR key  INCRBY key increment | 增加、增加指定的整数 |
|  | DECR key  DECRBY key decrement | 减少、减少指定的整数 |
|  | INCRBYFLOAT key increment | 增加指定的浮点数 |
|  | APPEND key value | 向尾部追加值 |
|  | STRLEN key | 获取字符串长度 |
|  | MGET key [key ...]  MSET key value [key value ...] | 同时获得/设置多个键值 |
|  | GETBIT key offset  SETBIT key offset value  BITCOUNT key [start] [end]  BITOP operation destkey key [key ...] | 位操作。BITOP命令可以对多个字符串类型键进行位操作，并将结果存储在destkey参数指定的键中。BITOP命令支持的运算操作有AND、OR、XOR和NOT。 |
| 散列表操作 | HSET key field value  HGET key field  HMSET key field value [field value ...]  HMGET key field [field ...]  HGETALL key | 散列类型赋值和取值 |
|  | HEXISTS key field | 判断字段是否存在 |
|  | HSETNX key field value | 当字段不存在时赋值 |
|  | HINCRBY key field increment | 增加数字 |
|  | HDEL key field [field ...] | 删除字段 |
|  | HKEYS key  HVALS key | 只获取字段名或字段值 |
|  | HLEN key | 获得字段数量 |
| 列表操作 | LPUSH key value [value ...]  RPUSH key value [value ...] | 向列表两端添加元素 |
|  | LPOP key  RPOP key  BLPOP key [key ...] timeout(s)  BRPOP key [key ...] timeout(s) | 从列表两端弹出元素  BLPOP/BRPOP是阻塞式，同时检测多个键，如果所有键都没有元素则阻塞，如果其中有一个键有元素则会从该键中弹出元素，返回值为键名和对应的元素值。如果多个键都有元素则按照从左到右的顺序取第一个键中的一个元素 |
|  | LLEN key | 获取列表中元素个数 |
|  | LRANGE key start stop | 获得列表片段（索引从0开始，支持负值，-1表示最右边第一个元素） |
|  | LREM key count value | 删除列表中指定的值。LREM命令会删除列表中前count个值为value的元素，根据count值不同，LREM命令的执行方式会略有差异：  当count>0时，从左边开始  当count<0时 从右边开始  当count=0时，删除所有值为value的元素 |
|  | LINDEX key index  LSET key index value | 获得/设置指定索引的元素值，索引从0开始，支持负数，-1是最右边元素的索引 |
|  | LTRIM key start end | 只保留列表指定片段 |
|  | LINSERT key BEFORE|AFTER pivot value | 向列表中插入元素。LINSERT命令首先会在列表中从左到右查找值为pivot的元素，然后根据第二个参数是BEFORE还是AFTER来决定将value插入到该元素的前面还是后面。 |
|  | RPOPLPUSH source destination | 将元素从一个列表转到另一个列表 |
| 集合操作 | SADD key member [member ...]  SREM key member [member ...] | 增加删除集合元素 |
|  | SMEMBERS key | 获得集合中的所有元素 |
|  | SISMEMBER key member | 判断元素是否在集合中 |
|  | SDIFF key [key ...]  SINTER key [key ...]  SUNION key [key ...] | 集合间运算  差集运算 A-B={x|x属于A且x不属于B}  交集运算  并集运算 |
|  | SCARD key | 获得集合中元素个数 |
|  | SDIFFSTORE destination key [key ...]  SINTERSTORE destination key [key ...]  SUNIONSTORE destination key [key ...] | 进行集合运算并将结果存储 |
|  | SRANDMEMBER key [count] | 随机获得集合中的元素  当count>0，随机从集合里获得count个不重复的元素  当count<0，随机从集合里获得|count|个元素，元素有可能相同 |
|  | SPOP key | 从集合中弹出一个元素  由于集合是无序的，所以SPOP命令会从集合中随机选择一个元素弹出 |
| 有序集合操作 | ZADD key score member [score member ...] | 有序集合增加元素  如果元素已经存在，则修改元素的分数。分数不仅可以是整数，还支持双精度浮点数。+inf和-inf表示正无穷和负无穷  redis> ZADD testboard 17E+307 a  redis> ZADD testboard +inf c  redis> ZADD testboard -inf d |
|  | ZSCORE key member | 获得元素的分数 |
|  | ZRANGE key start stop [WITHSCORES]  ZREVRANGE key start stop [WITHSCORES] | 获得排名在某个范围的元素列表  ZRANGE命令会按照元素分数从小到大的顺序返回索引从start到stop之间的所有元素。包括两端。-1表示最后一个元素。加上后面的WITHSCORES，返回的结果会跟上元素的分数。ZREVRANGE按照从大到小的顺序返回结果。 |
|  | ZRANGEBYSCORE key min max [WITHSCORES] [LIMIT offset count]  ZREVRANGEBYSCORE key max min [WITHSCORES] [LIMIT offset count] | 获得指定分数范围的元素  如果希望分数范围不包含端点值，可以在分数前加上“(”符号。redis> ZRANGEBYSCORE scoreboard 80 (100 |
|  | ZINCRBY key increment member | 增加某个元素的分数 |
|  | ZCARD key | 获得有序集合中元素数量 |
|  | ZCOUNT key min max | 获得指定分数范围内的元素个数 |
|  | ZREM key member [member ...] | 删除一个或多个元素 |
|  | ZREMRANGEBYRANK key start stop | 按照排名范围删除元素 |
|  | ZREMRANGEBYSCORE key min max | 按照分数范围删除元素 |
|  | ZRANK key member  ZREVRANK key member | 获得元素排名 |
|  | ZINTERSTORE destination numkeys key [key ...] [WEIGHTS weight [weight ...]] [AGGREGATE SUM|MIN|MAX]  ZUNIONSTORE destination numkeys key [key ...] [WEIGHTS weight [weight ...]] [AGGREGATE SUM|MIN|MAX] | 计算有序集合的交集/并集  destination键中元素的分数是由AGGREGATE参数决定的，默认是SUM。SUM是各参与计算的集合中该元素分数的和，MIN是最消息，MAX是最大值。  通过WEIGHTS参数设置每个集合的权重，每个集合在参与计算时元素的分数会被乘上该集合的权重。  redis> ZINTERSTORE result 2 sortedset1 sortedset2 WEIGHTS 1 0.1 |
| 事务 | MULTI  EXEC | 事务开始  事务执行 |
|  | WATCH | WATCH命令可以监控一个或多个键，一旦其中又一个键被修改（或删除），之后的事务就不会执行。监控一直持续到EXEC命令。 |
| 生存时间 | EXPIRE  PEXPIRE  EXPIREAT  PEXPIREAT | 设置一个键的生存时间 |
|  | TTL | 获取键的剩余时间 |
|  | PERSIST | 清除一个键的生存时间 |
| 排序 | SORT key [DESC] [LIMIT offset count] BY \* GET \* STORE \* | 可以对列表类型、集合类型和有序集合类型键进行排序 |
| 消息通知 | PUBLISH channel message | 发布者发布一条消息  返回值是接收到这条消息的订阅者数量 |
|  | SUBSCRIBE channel [channel ...]  UNSUBSCRIBE channel [channel ...] | 订阅/取消订阅频道 |
|  | PSUBSCRIBE pattern [pattern ...]  PUNSUBSCRIBE pattern [pattern ...] |  |