

一、KEY

```
1 127.0.0.1:6379> keys *    #查看所有的key
2 (empty array)
3 127.0.0.1:6379> set k1 v1 #设置键保存字符串值。
4 OK
5 127.0.0.1:6379> exists k1 #判断某个key是否存在
6 (integer) 1
7 127.0.0.1:6379> del k1 #删除key
8 (integer) 1
9 127.0.0.1:6379> keys *
10 (empty array)
11 127.0.0.1:6379> set k1 v1 EX 10 #设置过期时间
12 OK
13 127.0.0.1:6379> ttl k1 #查看剩余时间
14 (integer) 5
15 127.0.0.1:6379> ttl k1
16 (integer) 3
17 127.0.0.1:6379> ttl k1
18 (integer) 2
19 127.0.0.1:6379> set k2 v2
20 OK
21 127.0.0.1:6379> get k2 #获取key对应的value
22 "v2"
23 127.0.0.1:6379>
```

二、String

```
1 127.0.0.1:6379> keys *    #查看所有的key
2 1) "k2"
3 2) "num"
4 127.0.0.1:6379> get k2
5 "v2"
6 127.0.0.1:6379> get num
7 "2"
8 127.0.0.1:6379> incr num #自增 +1
9 (integer) 3
10 127.0.0.1:6379> incr num
```

```
11 (integer) 4
12 127.0.0.1:6379> decr num #减 -1 incr、decr 一定要是数字才能进行加减, +1 和 -1。
13 (integer) 3
14 127.0.0.1:6379> incrby num 10 #incrby、decrby 命令将 key 中储存的数字加上指定的增量值。
15 (integer) 13
16 127.0.0.1:6379> decrby num 5
17 (integer) 8
18 127.0.0.1:6379> set k3 asdfghj
19 OK
20 127.0.0.1:6379> getrange k3 0 -1 #getrange 获取指定区间范围内的值
21 "asdfghj"
22 127.0.0.1:6379> getrange k3 0 3
23 "asdf"
24 127.0.0.1:6379> get k3
25 "asdfghj"
26 127.0.0.1:6379> setrange k3 1 xx #setrange 设置指定区间范围内的值, 格式是setrange key
    值 具体值
27 (integer) 7
28 127.0.0.1:6379> get k3
29 "axxfghj"
30 127.0.0.1:6379> setex k3 10 exptime #设置过期时间
31 OK
32 127.0.0.1:6379> ttl k3 #查看剩余的时间
33 (integer) 6
34 127.0.0.1:6379> setnx k4 "redis" #如果不存在就设置, 成功返回1
35 (integer) 1
36 127.0.0.1:6379> get k4
37 "redis"
38 127.0.0.1:6379> setnx k4 "redis"
39 (integer) 0
40 127.0.0.1:6379> mset k10 v10 k11 v11 k12 v12 #同时设置一个或多个 key-value 对。
41 OK
42 127.0.0.1:6379> keys *
43 1) "k2"
44 2) "k12"
45 3) "k4"
46 4) "k10"
47 5) "k11"
48 6) "num"
```

```
49 127.0.0.1:6379> mget k10 k11 k12 k13 # 命令返回所有(一个或多个)给定 key 的值。如果给定的 key 里面, 有某个 key 不存在, 那么这个 key 返回特殊值 nil
50 1) "v10"
51 2) "v11"
52 3) "v12"
53 4) (nil)
54 127.0.0.1:6379> msetnx k10 v10 k15 v15 #当所有 key 都成功设置, 返回 1。如果所有给定 key 都设置失败(至少有一个 key 已经存在), 那么返回 0。原子操作。
55 (integer) 0
56 127.0.0.1:6379> get key15
57 (nil)
58 127.0.0.1:6379> set user:1 value(json数据) #传统对象缓存, 可以用来缓存对象
59 OK
60 127.0.0.1:6379> keys *
61 1) "k2"
62 2) "user:1"
63 3) "k12"
64 4) "k4"
65 5) "k10"
66 6) "k11"
67 7) "num"
68 127.0.0.1:6379> get user:1
69 "value(json\xe6\x95\xb0\xe6\x8d\xae)"
70 127.0.0.1:6379> mset user:1:name zhangsan user:1:age 16 #批量设置键和值
71 OK
72 127.0.0.1:6379> mget user:1:name user:1:age #批量获取值
73 1) "zhangsan"
74 2) "16"
75 127.0.0.1:6379> getset db mongodb #没有旧值, 返回 nil
76 (nil)
77 127.0.0.1:6379> get db
78 "mongodb"
79 127.0.0.1:6379> getset db redis
80 "mongodb"
81 127.0.0.1:6379> get db #返回旧值 mongodb
82 "redis"
83 127.0.0.1:6379> strlen db #获取字符串的长度 (integer)
84 (integer) 5
85
```

三、List

```
1
2 # =====
3 # lpush: 将一个或多个值插入到列表头部。(左)
4 # rpush: 将一个或多个值插入到列表尾部。(右)
5 # lrange: 返回列表中指定区间内的元素，区间以偏移量 START 和 END 指定。 # 其中 0 表示列表的第一个元素， 1 表示列表的第二个元素，以此类推。
6
7 # 你也可以使用负数下标，以 -1 表示列表的最后一个元素， -2 表示列表的倒数第二个元素，以此类推。
8 # =====
9 127.0.0.1:6379> LPUSH list "one"
10 (integer) 1
11 127.0.0.1:6379> LPUSH list "two"
12 (integer) 2
13 127.0.0.1:6379> RPUSH list "right"
14 (integer) 3
15 127.0.0.1:6379> Lrange list 0 -1
16 1) "two"
17 2) "one"
18 3) "right"
19 127.0.0.1:6379> Lrange list 0 1
20 1) "two"
21 2) "one"
22 # =====
23 # lpop 命令用于移除并返回列表的第一个元素。当列表 key 不存在时，返回 nil 。
24 # rpop 移除列表的最后一个元素，返回值为移除的元素。
25 # =====
26 127.0.0.1:6379> Lpop list
27 "two"
28 127.0.0.1:6379> Rpop list
29 "right"
30 127.0.0.1:6379> Lrange list 0 -1
31 1) "one"
32 # =====
33 # Lindex, 按照索引下标获得元素(-1代表最后一个，0代表是第一个)
34 # =====
35 127.0.0.1:6379> Lindex list 1
36 (nil)
```

```
37 127.0.0.1:6379> Lindex list 0
38 "one"
39 127.0.0.1:6379> Lindex list -1
40 "one"
41 # =====
42 # llen 用于返回列表的长度。
43 # =====
44 127.0.0.1:6379> flushdb
45 OK
46 127.0.0.1:6379> Lpush list "one" (integer) 1
47 127.0.0.1:6379> Lpush list "two" (integer) 2
48 127.0.0.1:6379> Lpush list "three" (integer) 3
49 127.0.0.1:6379> Llen list # 返回列表的长度 (integer) 3
50 # =====
51 # lrem key 根据参数 COUNT 的值, 移除列表中与参数 VALUE 相等的元素。 #
    =====
52 127.0.0.1:6379> lrem list 1 "two"
53 (integer) 1
54 127.0.0.1:6379> Lrange list 0 -1
55 1) "three"
56 2) "one"
57 # =====
58 # Ltrim key 对一个列表进行修剪(trim), 就是说, 让列表只保留指定区间内的元素, 不在指定区
    间的元素都将被删除。
59 # =====
60 127.0.0.1:6379> RPUSH mylist "hello"
61 (integer) 1
62 127.0.0.1:6379> RPUSH mylist "hello"
63 (integer) 2
64 127.0.0.1:6379> RPUSH mylist "hello2"
65 (integer) 3
66 127.0.0.1:6379> RPUSH mylist "hello3"
67 (integer) 4
68 127.0.0.1:6379> ltrim mylist 1 2
69 OK
70 127.0.0.1:6379> lrange mylist 0 -1
71 1) "hello"
72 2) "hello2"
73 # =====
```

```
74 # rpoplpush 移除列表的最后一个元素，并将该元素添加到另一个列表并返回。 #
=====
75 127.0.0.1:6379> rpush mylist "hello"
76 (integer) 1
77 127.0.0.1:6379> rpush mylist "foo"
78 (integer) 2
79 127.0.0.1:6379> rpush mylist "bar"
80 (integer) 3
81 127.0.0.1:6379> rpoplpush mylist myotherlist
82 "bar"
83 127.0.0.1:6379> lrange mylist 0 -1
84 1) "hello"
85 2) "foo"
86 127.0.0.1:6379> lrange myotherlist 0 -1
87 1) "bar"
88 # =====
89 # lset key index value 将列表 key 下标为 index 的元素的值设置为 value 。
90 # =====
91 127.0.0.1:6379> exists list # 对空列表(key 不存在)进行 LSET (integer) 0
92 127.0.0.1:6379> lset list 0 item # 报错
93 (error) ERR no such key
94 127.0.0.1:6379> lpush list "value1" # 对非空列表进行 LSET (integer) 1
95 127.0.0.1:6379> lrange list 0 0
96 1) "value1"
97 127.0.0.1:6379> lset list 0 "new" # 更新值 OK
98 127.0.0.1:6379> lrange list 0 0
99 1) "new"
100 127.0.0.1:6379> lset list 1 "new" # index 超出范围报错 (error) ERR index out of
    range
101 # =====
102 # linsert key before/after pivot value 用于在列表的元素前或者后插入元素。
103 # 将值 value 插入到列表 key 当中，位于值 pivot 之前或之后。
104 # =====
105 redis> RPUSH mylist "Hello"
106 (integer) 1
107 redis> RPUSH mylist "World"
108 (integer) 2
109 redis> LINSERT mylist BEFORE "World" "There"
110 (integer) 3
111 redis> LRANGE mylist 0 -1
```

```
112 1) "Hello"
113 2) "There"
114 3) "World"
115
```

四、Set

```
1 # =====
2 # sadd 将一个或多个成员元素加入到集合中，不能重复
3 # smembers 返回集合中的所有的成员。
4 # sismember 命令判断成员元素是否是集合的成员。
5 # =====
6 127.0.0.1:6379> sadd myset "hello"
7 (integer) 1
8 127.0.0.1:6379> sadd myset "daming"
9 (integer) 1
10 127.0.0.1:6379> sadd myset "daming"
11 (integer) 0
12 127.0.0.1:6379> SMEMBERS myset
13 1) "daming"
14 2) "hello"
15 127.0.0.1:6379> SISMEMBER myset "hello"
16 (integer) 1
17 127.0.0.1:6379> SISMEMBER myset "world"
18 (integer) 0
19 # =====
20 # scard, 获取集合里面的元素个数
21 # =====
22 127.0.0.1:6379> scard myset
23 (integer) 2
24 # =====
25 # srem key value 用于移除集合中的一个或多个成员元素
26 # =====
27 127.0.0.1:6379> srem myset "daming"
28 (integer) 1
29 127.0.0.1:6379> SMEMBERS myset
30 1) "hello"
31 # =====
32 # srandmember key 命令用于返回集合中的一个随机元素。
```

```
33 # =====
34 127.0.0.1:6379> SMEMBERS myset
35 1) "daming"
36 2) "world"
37 3) "hello"
38 127.0.0.1:6379> SRANDMEMBER myset
39 "hello"
40 127.0.0.1:6379> SRANDMEMBER myset 2
41 1) "world"
42 2) "daming"
43 127.0.0.1:6379> SRANDMEMBER myset 2
44 1) "daming"
45 2) "hello"
46 # =====
47 # spop key 用于移除集合中的指定 key 的一个或多个随机元素
48 # =====
49 127.0.0.1:6379> SMEMBERS myset
50 1) "daming"
51 2) "world"
52 3) "hello"
53 127.0.0.1:6379> spop myset
54 "world"
55 127.0.0.1:6379> spop myset
56 "daming"
57 127.0.0.1:6379> spop myset
58 "hello"
59 # =====
60 # smove SOURCE DESTINATION MEMBER
61 # 将指定成员 member 元素从 source 集合移动到 destination 集合。
62 # =====
63 127.0.0.1:6379> sadd myset "hello"
64 (integer) 1
65 127.0.0.1:6379> sadd myset "world"
66 (integer) 1
67 127.0.0.1:6379> sadd myset "daming"
68 (integer) 1
69 127.0.0.1:6379> sadd myset2 "set2"
70 (integer) 1
71 127.0.0.1:6379> smove myset myset2 "daming"
72 (integer) 1
```



```

73 127.0.0.1:6379> SMEMBERS myset
74 1) "world"
75 2) "hello"
76 127.0.0.1:6379> SMEMBERS myset2
77 1) "daming"
78 2) "set2"
79 # =====
80 - 数字集合类
81 - 差集: sdiff - 交集: sinter
82 - 并集: sunion
83 # =====
84 127.0.0.1:6379> sadd key1 "a"
85 (integer) 1
86 127.0.0.1:6379> sadd key1 "b"
87 (integer) 1
88 127.0.0.1:6379> sadd key1 "c"
89 (integer) 1
90 127.0.0.1:6379> sadd key2 "c"
91 (integer) 1
92 127.0.0.1:6379> sadd key2 "d"
93 (integer) 1
94 127.0.0.1:6379> sadd key2 "e"
95 (integer) 1
96 127.0.0.1:6379> SDIFF key1 key2 # 差集
97 1) "a"
98 2) "b"
99 127.0.0.1:6379> SINTER key1 key2 # 交集
100 1) "c"
101 127.0.0.1:6379> SUNION key1 key2 # 并集
102 1) "a"
103 2) "b"
104 3) "c"
105 4) "e"
106 5) "d"
107

```

五、Hash

```

1 # =====

```

```
2 # hset、hget 命令用于为哈希表中的字段赋值 。
3 # hmset、hget 同时将多个field-value对设置到哈希表中。会覆盖哈希表中已存在的字段。
4 # hgetall 用于返回哈希表中，所有的字段和值。
5 # hdel 用于删除哈希表 key 中的一个或多个指定字段
6 # =====
7 127.0.0.1:6379> hset myhash field1 "daming"
8 (integer) 1
9 127.0.0.1:6379> hget myhash field1
10 "daming"
11 127.0.0.1:6379> HMSET myhash field1 "Hello" field2 "World"
12 OK
13 127.0.0.1:6379> HGET myhash field1
14 "Hello"
15 127.0.0.1:6379> HGET myhash field2
16 "World"
17 127.0.0.1:6379> hgetall myhash
18 1) "field1"
19 2) "Hello"
20 3) "field2"
21 4) "World"
22 127.0.0.1:6379> HDEL myhash field1
23 (integer) 1
24 127.0.0.1:6379> hgetall myhash
25 1) "field2"
26 2) "World"
27 # =====
28 # hlen 获取哈希表中字段的数量。
29 # =====
30 127.0.0.1:6379> hlen myhash
31 (integer) 1
32 127.0.0.1:6379> HMSET myhash field1 "Hello" field2 "World"
33 OK
34 127.0.0.1:6379> hlen myhash
35 (integer) 2
36 # =====
37 # hexists 查看哈希表的指定字段是否存在。
38 # =====
39 127.0.0.1:6379> hexists myhash field1
40 (integer) 1
41 127.0.0.1:6379> hexists myhash field3
```

```

42 (integer) 0
43 # =====
44 # hkeys 获取哈希表中的所有域(field)。
45 # hvals 返回哈希表所有域(field)的值。
46 # =====
47 127.0.0.1:6379> HKEYS myhash
48 1) "field2"
49 2) "field1"
50 127.0.0.1:6379> HVALS myhash
51 1) "World"
52 2) "Hello"
53 # =====
54 # hincrby 为哈希表中的字段值加上指定增量值。
55 # =====
56 127.0.0.1:6379> hset myhash field 5
57 (integer) 1
58 127.0.0.1:6379> HINCRBY myhash field 1
59 (integer) 6
60 127.0.0.1:6379> HINCRBY myhash field -1
61 (integer) 5
62 127.0.0.1:6379> HINCRBY myhash field -10
63 (integer) -5
64 # =====
65 # hsetnx 为哈希表中不存在的的字段赋值 。
66 # =====
67 127.0.0.1:6379> HSETNX myhash field1 "hello" (integer) 1 # 设置成功, 返回 1 。
68 127.0.0.1:6379> HSETNX myhash field1 "world" (integer) 0 # 如果给定字段已经存在, 返回 0 。
69 127.0.0.1:6379> HGET myhash field1
70 "hello"
71

```

六、Sorted set

```

1 # =====
2 # zadd 将一个或多个成员元素及其分数值加入到有序集当中。
3 # zrange 返回有序集中, 指定区间内的成员
4 # =====
5 127.0.0.1:6379> zadd myset 1 "one"

```

```
6 (integer) 1
7 127.0.0.1:6379> zadd myset 2 "two" 3 "three"
8 (integer) 2
9 127.0.0.1:6379> ZRANGE myset 0 -1
10 1) "one"
11 2) "two"
12 3) "three"
13 # =====
14 # zrangebyscore 返回有序集合中指定分数区间的成员列表。有序集成员按分数值递增(从小到大) 次序排列。
15 # =====
16 127.0.0.1:6379> zadd salary 2500 xiaoming
17 (integer) 1
18 127.0.0.1:6379> zadd salary 5000 xiaohong
19 (integer) 1
20 127.0.0.1:6379> zadd salary 500 daming
21 (integer) 1
22 # Inf无穷大量+∞,同样地,-∞可以表示为-Inf。
23 127.0.0.1:6379> ZRANGEBYSCORE salary -inf
24 1) "daming"
25 2) "xiaoming"
26 3) "xiaohong"
27 127.0.0.1:6379> ZRANGEBYSCORE salary -inf
28 1) "daming"
29 2) "500"
30 3) "xiaoming"
31 4) "2500"
32 5) "xiaohong"
33 6) "5000"
34 127.0.0.1:6379> ZREVRANGE salary 0 -1 WITHSCORES # 递减排列
35 1) "xiaohong"
36 2) "5000"
37 3) "xiaoming"
38 4) "2500"
39 5) "daming"
40 6) "500"
41 127.0.0.1:6379> ZRANGEBYSCORE salary -inf 2500 WITHSCORES # 显示工资 <=2500 的所有成员
42 1) "daming"
43 2) "500"
```

```
44 3) "xiaoming"
45 4) "2500"
46 # =====
47 # zrem 移除有序集中的一个或多个成员
48 # =====
49 127.0.0.1:6379> ZRANGE salary 0 -1
50 1) "daming"
51 2) "xiaoming"
52 3) "xiaohong"
53 127.0.0.1:6379> zrem salary daming
54 (integer) 1
55 127.0.0.1:6379> ZRANGE salary 0 -1
56 1) "xiaoming"
57 2) "xiaohong"
58 # =====
59 # zcard 命令用于计算集合中元素的数量。
60 # =====
61 127.0.0.1:6379> zcard salary
62 (integer) 2
63 OK
64 # =====
65 # zcount 计算有序集合中指定分数区间的成员数量。
66 # =====
67 127.0.0.1:6379> zadd myset 1 "hello"
68 (integer) 1
69 127.0.0.1:6379> zadd myset 2 "world" 3 "daming"
70 (integer) 2
71 127.0.0.1:6379> ZCOUNT myset 1 3
72 (integer) 3
73 127.0.0.1:6379> ZCOUNT myset 1 2
74 (integer) 2
75 # =====
76 # zrank 返回有序集中指定成员的排名。其中有序集成员按分数值递增(从小到大)顺序排列。 #
    =====
77 127.0.0.1:6379> zadd salary 2500 xiaoming
78 (integer) 1
79 127.0.0.1:6379> zadd salary 5000 xiaohong
80 (integer) 1
81 127.0.0.1:6379> zadd salary 500 daming
82 (integer) 1
```

```
83 127.0.0.1:6379> ZRANGE salary 0 -1 WITHSCORES # 显示所有成员及其 score 值
84 1) "daming"
85 2) "500"
86 3) "xiaoming"
87 4) "2500"
88 5) "xiaohong"
89 6) "5000"
90 127.0.0.1:6379> zrank salary daming # 显示 daming 的薪水排名, 最少 (integer) 0
91 127.0.0.1:6379> zrank salary xiaohong # 显示 xiaohong 的薪水排名, 第三 (integer) 2
92 # =====
93 # zrevrank 返回有序集中成员的排名。其中有序集成员按分数值递减(从大到小)排序。
94 # =====
95 127.0.0.1:6379> ZREVRANK salary daming # 大明第三
96 (integer) 2
97 127.0.0.1:6379> ZREVRANK salary xiaohong # 小红第一 (integer) 0
```