大数据库系统

# 3.3 Redis数据类型及操作

# 3.3 Redis数据类型及操作

- 主要内容

3.3.1 通用命令

3.3.2 STRING（字符串）

3.3.3 LIST（列表）

3.3.4 SET（无序集合）

3.3.5 ZSET（有序集合）

3.3.6 HASH（哈希表）

# 3.3.3 LIST（列表）

- LIST及相关命令
  - Redis的列表（List）数据类型可以被看作简单的字符串列表，列表按照插入顺序排序
  - 在操作Redis的列表时，可以将一个元素插入/移除这个列表的头部或尾部
  - 即可以看成一个堆栈
  - 也可以看出一个队列

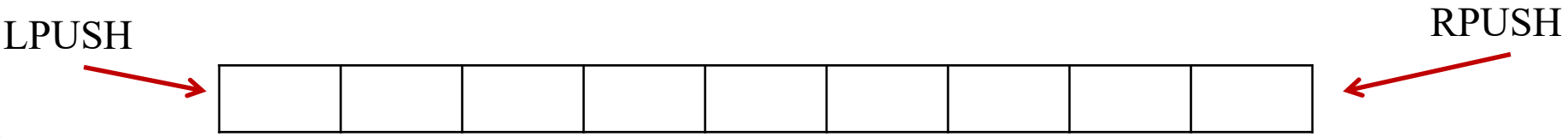
# 3.3.3 LIST（列表）

- LPUSH、RPUSH命令

LPUSH/ RPUSH key value1 [value2 ...]

将多个值插入列表头部/尾部

- LPUSH/ RPUSH将一个或多个value值插入列表key的头部/尾部。
- 可以想象一个水平放置的列表
- 返回值为列表的长度
- 如果命令后跟多个value，先后插入顺序为：value1、value2、……



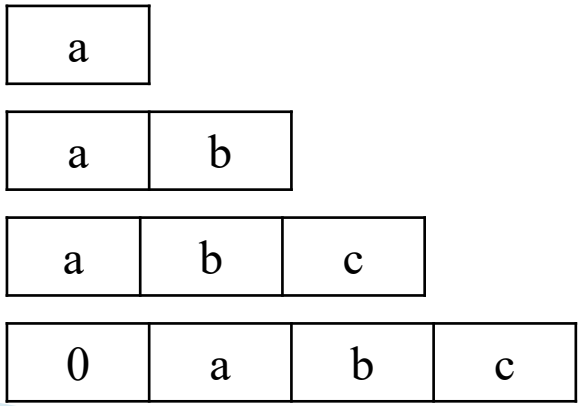
# 3.3.3 LIST（列表）

例：character里存放的value是怎样排列的？

```
redis 127.0.0.1:6379> keys *
(empty list or set)
redis 127.0.0.1:6379> lpush character a
(integer) 1
redis 127.0.0.1:6379> rpush character b
(integer) 2
redis 127.0.0.1:6379> rpush character c
(integer) 3
redis 127.0.0.1:6379> lpush character 0
(integer) 4
redis 127.0.0.1:6379>
```

可以看到lpush与rpush返回的数字为当前列表的长度

目前character的内容为：



# 3.3.3 LIST（列表）

- LRANGE命令

LRANGE key start end

获取列表指定区间内的元素，区间从start开始，到end结束，左数从0开始，右数从-1开始

| h | e | l | l | o |
|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 |
| -5 | -4 | -3 | -2 | -1 |

# 3.3.3 LIST（列表）

例1：获取character列表的内容

```
redis 127.0.0.1:6379> lrange character 0 3
1) "0"
2) "a"
3) "b"
4) "c"
```

可以看到跟我们刚才列出的内容一致

问：由于刚刚已经知道character列表的长度，所以们可以指定其实位置是从0到3，

如果不知道character的长度该怎样查询character的全部内容呢?

答：由于从右数是从-1开始，因此可以使用lrange character 0 -1

```
redis 127.0.0.1:6379> lrange character 0 -1
1) "0"
2) "a"
3) "b"
4) "c"
redis 127.0.0.1:6379>
```

这种方法适用于不知道列表长度情况下获取列表的全部内容

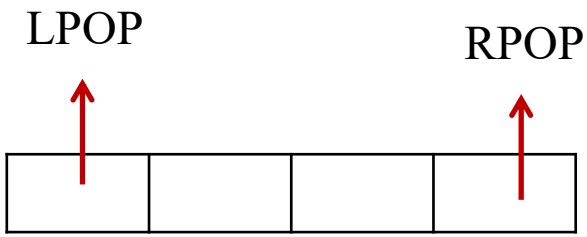
# 3.3.3 LIST（列表）

- LPOP/RPOP命令

LPOP/RPOP key

LPOP/RPOP命令用于返回列表key的头/尾元素，同时把这个头/尾元素删除[1]

返回值：返回列表的头/尾元素。如果key不存在，则将会返回nil




```
redis 127.0.0.1:6379> lrange character 0 -1
1) "0"
2) "a"
3) "b"
4) "c"
redis 127.0.0.1:6379> lpop character
"0"
redis 127.0.0.1:6379> lrange character 0 -1
1) "a"
2) "b"
3) "c"
redis 127.0.0.1:6379> rpop character
"c"
redis 127.0.0.1:6379> lrange character 0 -1
1) "a"
2) "b"
redis 127.0.0.1:6379>
```

# 3.3.3 LIST（列表）

- LREM命令

LREM KEY COUNT VALUE

删除指定COUNT个数的VALUE元素

- 当count=0时，表示删除列表key中所有与value相等的元素[2]
- 当count>0时，表示从列表key的表头开始向表尾搜索，删除与value相等的元素，删除的数量为count个
- 当count<0时，表示从列表key的表尾开始向表头搜索，删除与value相等的元素，删除的数量为count的绝对值个

- 例1：从左侧开始删除1个b

```
redis 127.0.0.1:6379> flushdb
OK
redis 127.0.0.1:6379> lpush answer a b c a b d a
(integer) 7
redis 127.0.0.1:6379> lrange answer 0 -1
1) "a"
2) "d"
3) "b"
4) "a"
5) "c"
6) "b"
7) "a"
redis 127.0.0.1:6379> lrem answer 1 b
(integer) 1
redis 127.0.0.1:6379> lrange answer 0 -1
1) "a"
2) "d"
3) "a"
4) "c"
5) "b"
6) "a"
redis 127.0.0.1:6379>
```

- 例2：从右侧开始，删除2个a

```
redis 127.0.0.1:6379> lrange answer 0 -1
1) "a"
2) "d"
3) "a"
4) "c"
5) "b"
6) "a"
redis 127.0.0.1:6379> lrem answer -2 a
(integer) 2
redis 127.0.0.1:6379> lrange answer 0 -1
1) "a"
2) "d"
3) "c"
4) "b"
redis 127.0.0.1:6379>
```

# 3.3.3 LIST（列表）

- LTRIM命令

LTRIM key start stop

用于对一个列表进行修剪（trim），切[start,stop]一段，并把该段重新赋给key

- 参数start和stop也可以是负数，用-1表示列表中的最后一个元素
- 如果参数start的值比列表下标的最大值还要大，清空这个列表

- 构造列表character: a b c d e f
- 例1：对character进行剪切，保留c d e f

```
redis 127.0.0.1:6379> rpush character a b c d e f
(integer) 6
redis 127.0.0.1:6379> lrange character 0 -1
1) "a"
2) "b"
3) "c"
4) "d"
5) "e"
6) "f"
redis 127.0.0.1:6379> ltrim character 2 5
OK
redis 127.0.0.1:6379> lrange character 0 -1
1) "c"
2) "d"
3) "e"
4) "f"
redis 127.0.0.1:6379>
```

- 例2：从表尾开始剪切，保留d $e^3$

```
redis 127.0.0.1:6379> lrange character 0 -1
1) "c"
2) "d"
3) "e"
4) "f"
redis 127.0.0.1:6379> ltrim character 1 -2
OK
redis 127.0.0.1:6379> lrange character 0 -1
1) "d"
2) "e"
redis 127.0.0.1:6379>
```

# 3.3.3 LIST（列表）

- LINDEX命令

LINDEX key index

获取列表key中下标为index的元素，下标从0开始

```
redis 127.0.0.1:6379> lrange character 0 -1
1) "d"
2) "e"
redis 127.0.0.1:6379> lindex character 0
"d"
redis 127.0.0.1:6379> lindex character 1
"e"
redis 127.0.0.1:6379> lindex character 2
(nil)
redis 127.0.0.1:6379>
```

# 3.3.3 LIST（列表）

- LSET

命令格式：

LSET key index value

LSET命令用于设置列表key中下标为index的值为value

当下标index参数超出范围时，将会返回错误；当列表key为空时，也会返回错误

```
127.0.0.1:6379> rpush name zhangsan lisi wangwu
(integer) 3
127.0.0.1:6379> lset name 1 lily
OK
127.0.0.1:6379> lrange name 0 -1
1) "zhangsan"
2) "lily"
3) "wangwu"
127.0.0.1:6379>
```

思考下，如何删掉列表中索引值为2的元素呢？

# 3.3.3 LIST（列表）

- LLEN命令

LLEN key

LLEN命令用于统计列表key的长度

```
redis 127.0.0.1:6379> lrange character 0 -1
1) "d"
2) "e"
redis 127.0.0.1:6379> llen character
(integer) 2
redis 127.0.0.1:6379> rpush character b c d
(integer) 5
redis 127.0.0.1:6379> llen character
(integer) 5
```

# 3.3.3 LIST（列表）

- LINSERT命令

LINSERT key BEFORE | AFTER pivot value

用于向列表中插入一个值，将值value插入列表key当中，这个值的位置在值pivot之前或之后

如果成功，则返回插入操作完成之后的列表长度，如果只有pivot不存在，则返回-1；如果key不存在，或者是空列表，则返回0

注意：一旦找到一个pivot后，命令就结束了，因此不会插入多个value

准备列表num：1 3 6 8 9

例1：在列表num的3之前插入2

```
redis 127.0.0.1:6379> rpush num 1 3 6 8 9
(integer) 5
redis 127.0.0.1:6379> linsert num before 3 2
(integer) 6
redis 127.0.0.1:6379> lrange num 0 -1
1) "1"
2) "2"
3) "3"
4) "6"
5) "8"
6) "9"
redis 127.0.0.1:6379>
```

例2：在列表num的9之后插入10

```
redis 127.0.0.1:6379> linsert num after 9 10
(integer) 7
redis 127.0.0.1:6379> lrange num 0 -1
1) "1"
2) "2"
3) "3"
4) "6"
5) "8"
6) "9"
7) "10"
```

例3：在列表num的99之后插入10

```
redis 127.0.0.1:6379> linsert num after 99 10
(integer) -1
redis 127.0.0.1:6379>
```

# 3.3.3 LIST（列表）

- 例4：在列表num的9之后插入10，再在列表num的10之后插入11

```
redis 127.0.0.1:6379> linsert num after 9 10
(integer) 8
redis 127.0.0.1:6379> lrange num 0 -1
1) "1"
2) "2"
3) "3"
4) "6"
5) "8"
6) "9"
7) "10"
8) "10"
redis 127.0.0.1:6379> linsert num after 10 11
(integer) 9
redis 127.0.0.1:6379> lrange num 0 -1
1) "1"
2) "2"
3) "3"
4) "6"
5) "8"
6) "9"
7) "10"
8) "11"
9) "10"
redis 127.0.0.1:6379>
```

- 有两个10，但只在第一个10后面插入了11

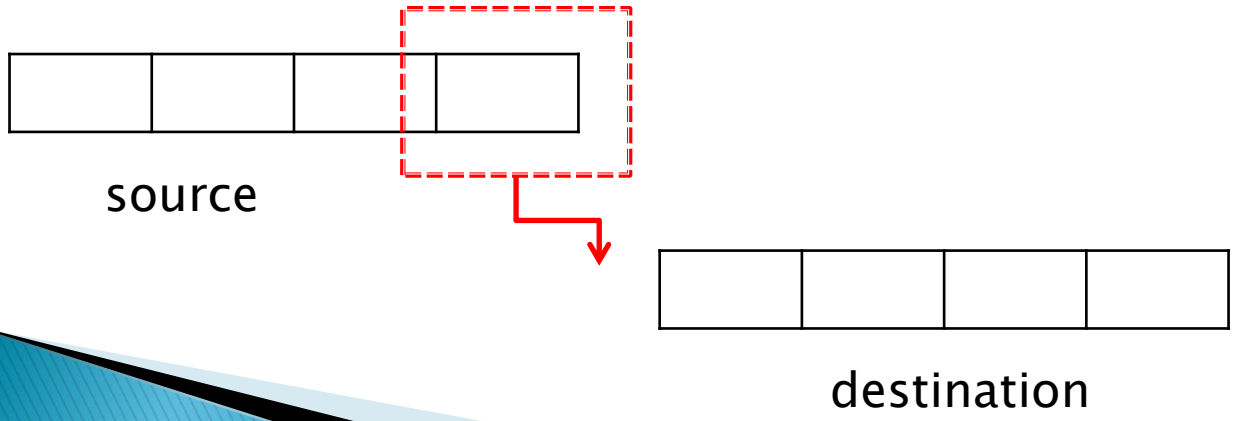
# 3.3.3 LIST（列表）

## ◆ RPOPLPUSH命令

RPOPLPUSH source destination

将列表source中的尾元素弹出，并返回给客户端。这个被返回的元素将会插入列表destination中，作为该列表的头元素[4]

- 如果列表source与列表destination相同，那么列表的尾元素将被移动到表头，并返回该元素。



# 3.3.3 LIST（列表）

创建task列表 a b c d，创建空列表job

例1：将task列表的d，移到job列表的头部

```
redis 127.0.0.1:6379> rpush task a b c d
(integer) 4
redis 127.0.0.1:6379> rpoplpush task job
"d"
redis 127.0.0.1:6379> lrange task 0 -1
1) "a"
2) "b"
3) "c"
redis 127.0.0.1:6379> lrange job 0 -1
1) "d"
redis 127.0.0.1:6379>
```

# 3.3.3 LIST（列表）

- BLPOP、 BRPOP命令

BLPOP/ BRPOP key [key ...] timeout

在指定时间内删除列表的头/尾元素， timeout为等待超时时间

即当给定列表内没有任何元素可以返回时，连接将被BRPOP命令阻塞，直到等待超时或发现可返回的元素为止

例：清空列表job，等待job列表20秒

```
redis 127.0.0.1:6379> lrange job 0 -1
1) "d"
redis 127.0.0.1:6379> rpop job
"d"
redis 127.0.0.1:6379> rpop job
(nil)
redis 127.0.0.1:6379> brpop job 20
```

终端1

由于列表job已经为空，因此使用brpop后，就一直处在等待状态，这时再开一个终端，lpush一个e

```
[root@localhost redis]# ./bin/redis-cli
redis 127.0.0.1:6379> lpush job e
(integer) 1
redis 127.0.0.1:6379>
```

终端2

这时再回去看brpop job 20后会有什么反应

```
redis 127.0.0.1:6379> brpop job 20
1) "job"
2) "e"
(12.15s)
redis 127.0.0.1:6379>
```

终端1

可以看到brpop返回了值，并返回了时间

适用于在线聊天、轮询的场景中，发送信息即PUSH，接收信息即POP，如果发送端没发送信息，则一直等着

# 3.3.3 LIST（列表）

- LIST及相关命令

LPUSH/RPUSH
LPOP/RPOP
LRANGE
LREM
LTRIM
LSET
LINDEX
LLEN
LINSERT
BRPOP/BLPOP
RPOPLPUSH

# 3.3 Redis数据类型及操作

- 主要内容

3.3.1 通用命令

3.3.2 STRING（字符串）

3.3.3 LIST（列表）

3.3.4 SET（无序集合）

3.3.5 ZSET（有序集合）

3.3.6 HASH（哈希表）

# 3.3.4 SET（无序集合）

- SET类型的相关命令

SET即集合，String类型的无序集合

- 集合无序
- 不存在重复的元素，每个元素都是唯一的

# 3.3.4 SET（无序集合）

- SADD命令

  SADD key member [member ...]

  SADD命令用于将一个或多个member元素添加到集合key中

  如果这个集合key中已经存在这个member元素，那么它将会被忽略

  执行命令成功后，返回被添加到集合中的新元素的数量，不包含被忽略的元素

# 3.3.4 SET（无序集合）

- 例1：向gender集合添加两个元素：male和female

```
127.0.0.1:6379> sadd gender male female
(integer) 2
127.0.0.1:6379>
```

- 例2：向gender集合随意添加两个重复的元素

```
127.0.0.1:6379> sadd gender yahoo yahoo
(integer) 1
127.0.0.1:6379> sadd gender yahoo
(integer) 0
127.0.0.1:6379>
```

- 可以发现，由于集合元素具有唯一性，因此添加的yahoo元素只保存一个，第一行返回1，指添加了一个元素，第二次添加yahoo元素返回为0，指没添加任何元素

# 3.3.4 SET（无序集合）

## ◆ SMEMBERS命令

SMEMBERS key

用于获取集合key中的所有元素

- 例：查看gender集合里的所有元素
- 问：对于gender集合，添加的顺序为 male、female、yahoo，为什么SMEMBERS 结果为female、 male、yahoo ?
- 答：集合的无序性

```
127.0.0.1:6379> sadd gender male female
(integer) 2
127.0.0.1:6379> sadd gender yahoo yahoo
(integer) 1
127.0.0.1:6379> sadd gender yahoo
(integer) 0
127.0.0.1:6379> smembers gender
1) "female"
2) "male"
3) "yahoo"
127.0.0.1:6379>
```

# 3.3.4 SET（无序集合）

- SREM命令

SREM key member [member ...]

用于删除集合key中的一个或多个member元素。该命令在执行过程中会忽略不存在的member元素

```
127.0.0.1:6379> smembers gender
1) "female"
2) "male"
3) "yahoo"
127.0.0.1:6379> srem gender yahoo
(integer) 1
127.0.0.1:6379> smembers gender
1) "female"
2) "male"
127.0.0.1:6379> srem gender x c
(integer) 0
127.0.0.1:6379> srem gender male x c
(integer) 1
127.0.0.1:6379>
```

为什么返回1？[4]

# 3.3.4 SET（无序集合）

- SPOP命令

SPOP key [count]

用于随机删除集合key中的一个或多个元素[5]

SPOP命令成功执行后，返回被删除的随机元素

集合无序，不能按某种顺序删除，适用于抽奖等场景[6]

例：创建集合gender a b c d e f，使用spop随机删除元素

```
127.0.0.1:6379> del gender
(integer) 1
127.0.0.1:6379> sadd gender a b c d e f
(integer) 6
127.0.0.1:6379> spop gender
"c"
127.0.0.1:6379> smembers gender
1) "a"
2) "f"
3) "e"
4) "b"
5) "d"
127.0.0.1:6379>
```

```
127.0.0.1:6379> spop gender
"d"
127.0.0.1:6379> spop gender
"f"
127.0.0.1:6379> spop gender
"e"
```

# 3.3.4 SET（无序集合）

- SRANDMEMBER函数

SRANDMEMBER key

随机返回集合key中的一个元素

命令SPOP在从集合中随机删除元素的同时返回这个元素；

而SRANDMEMBER命令只随机返回元素，并不会改动这个集合的内容

```
127.0.0.1:6379> del gender
(integer) 1
127.0.0.1:6379> sadd gender a b c d e f
(integer) 6
127.0.0.1:6379> srandmember gender
"d"
127.0.0.1:6379> srandmember gender
"b"
127.0.0.1:6379> srandmember gender
"f"
127.0.0.1:6379> srandmember gender
"a"
127.0.0.1:6379> srandmember gender
"d"
```

# 3.3.4 SET（无序集合）

- SISMEMBER命令

SISMEMBER key number

判断元素number是否在集合key中

返回值：如果集合key中存在元素number，则返回1；如果集合key中不存在元素number，或者集合key不存在，就返回0

```
127.0.0.1:6379> smembers gender
1) "c"
2) "a"
3) "f"
4) "e"
5) "b"
6) "d"
127.0.0.1:6379> sismember gender Q
(integer) 0
127.0.0.1:6379> sismember gender f
(integer) 1
```

# 3.3.4 SET（无序集合）

- SCARD命令

SCARD key

获取集合key中元素的数量[6]

例：输出gender集合元素个数

```
127.0.0.1:6379> scard gender
(integer) 6
127.0.0.1:6379>
```

# 3.3.4 SET（无序集合）

## ◆ SMOVE命令

SMOVE source destination member

将集合source中的member元素移动到集合destination中

注意：当集合destination中已经包含member元素时，SMOVE命令只是简单地将集合source中的member元素删除，而不会移动

·例：构造两个集合upper={A, B, C} 、lower={a,b,c}将A从upper移动至lower

```
127.0.0.1:6379> sadd upper A B C
(integer) 3
127.0.0.1:6379> sadd lower a b c
(integer) 3
127.0.0.1:6379> smove upper lower A
(integer) 1
127.0.0.1:6379> smembers upper
1) "C"
2) "B"
127.0.0.1:6379> smembers lower
1) "b"
2) "c"
3) "a"
4) "A"
```

```
127.0.0.1:6379> smove lower upper A
(integer) 1
127.0.0.1:6379> smembers upper
1) "C"
2) "A"
3) "B"
127.0.0.1:6379>
```

# 3.3.4 SET（无序集合）

- SINTER、SUNION、SDIFF命令

SINTER/ SUNION/ SDIFF key [key ...]

获取给定的一个或多个集合key中的全部元素，该集合是所有给定集合的交集/并集/差集。

注意：被返回的交集/并集/差集中的元素并不会被保存

- 创建集合lisi、wang、poly

```
127.0.0.1:6379> sadd lisi a b c d
(integer) 4
127.0.0.1:6379> sadd wang a c d e f
(integer) 5
127.0.0.1:6379> sadd poly a c d g
```

- 例1：求三者交集

```
127.0.0.1:6379> sinter lisi wang poly
1) "c"
2) "d"
3) "a"
```

- 例2：求三者并集

```
127.0.0.1:6379> sunion lisi wang poly
1) "c"
2) "a"
3) "g"
4) "f"
5) "e"
6) "b"
7) "d"
```

- 例3：求lisi与wang的差集

```
127.0.0.1:6379> sdiff lisi wang
1) "b"
```

# 3.3.4 SET（无序集合）

- SINTERSTORE 、SUNIONSTORE 、SDIFFSTORE 命令

SINTERSTORE/ SUIONSTORE/ SDIFFSTORE destination key [key ...]

获取给定的一个或多个集合key中的交集/并集/差集的全部元素，并将这些元素保存到集合destination中，返回结果集destination中的成员数量

如果集合destination已经存在，则会被新产生的集合覆盖

```
127.0.0.1:6379> sinter lisi wang poly
1) "c"
2) "d"
3) "a"
127.0.0.1:6379> sinterstore result lisi wang poly
(integer) 3
127.0.0.1:6379> smembers result
1) "c"
2) "a"
3) "d"
127.0.0.1:6379>
```

# 总结

- LIST及相关命令

LPUSH/RPUSH
LPOP/RPOP
LRANGE
LREM
LTRIM
LSET
LINDEX
LLEN
LINSERT
BRPOP/BLPOP
RPOPLPUSH

- SET类型的相关命令

SADD
SREM
SPOP
SRANDMEMBER
SISMEMBER
SMEMBERS
SCARD
SMOVE
SINTER /SUNION /SDIFF
SINTERSTORE/SUNIONSTORE/SDIFFSTORE