title	date	categories	tags
Redis 数据结构之 list	Sat May 28 2016 21:23:18 GMT+0800 (CST)	Redis	redis

前言

Redis作为cache服务器，支持多种数据结构，String、List、Hash、Set、Zset。多种数据结构的存在，使得`Redis`适用于多种业务,Redis的适用也越发广泛，本文就介绍Redis中最简单的数据结构`List`的操作命令。

简介

`List`顾名思义，即链表、队列。即一个`key`对应的`value`为一个队列,支持头插、尾插等操作。

命令简介

[LR]PUSH 命令

`语法`: [LR]PUSH key value1 [value2 ...]
`作用`: 以头插或尾插方式插入指定`key`队列中一个或多个元素
`返回`: 成功返回插入后的队列中元素个数

头插法插入N个元素

> 127.0.0.1:6379> LPUSH list val1 val2 val3
> (integer) 3

尾插法插入N个元素

```
127.0.0.1:6379> RPUSH list val1 val2 val3
(integer) 6
```

`List` 中的所有元素

```
127.0.0.1:6379> LRANGE list 0 -1
1) "val3"
2) "val2"
3) "val1"
4) "val1"
5) "val2"
6) "val3"
```

[LR]PUSHX 命令

`语法`: [LR]PUSHX key value `作用`: 以头插或尾插方式插入指定`key`队列中一个或多个元素
`返回`: 当指定的队列不存在时，返回`0`,否则插入成功返回插入后队列中的元素个数

向一个不存在的队列中以头插法插入一个节点

```
127.0.0.1:6379> LPUSHX list value
(integer) 0
```

向一个存在的队列中以头插法插入一个节点

```
127.0.0.1:6379> LRANGE list 0 -1
1) "value"
127.0.0.1:6379> LPUSHX list value
(integer) 2
```

尾插法命令使用方式与头插法类似，此处不做过多介绍

LINSERT 命令

`语法`：LINSERT key where refVal value
`作用`：在已存在的队列`key`中指定的节点`refval`位置的前或后插入节点`value`。`where`为`after`时在指定节点后插入，`before`为在指定节点前插入
`返回`：队列不存在则返回0;如果指定节点`refVal`不存在，则不做任何操作并返回`-1`，否则按照指定位置插入新元素，并返回插入后节点个数

向一个不存在的队列中插入元素

```
127.0.0.1:6379> LINSERT list after value value1
(integer) 0
```

向一个已经存在的队列中,指定一个不存在节点位置插入

```
127.0.0.1:6379> LRANGE list 0 -1
1) "value"
2) "value"
127.0.0.1:6379> LINSERT list after value1 value11
(integer) -1
```

向一个已经存在的队列中，指定一个存在的节点位置后插入新元素

```
127.0.0.1:6379> LINSERT list after value value11
(integer) 3
127.0.0.1:6379> LRANGE list 0 -1
1) "value"
2) "value11"
3) "value"
```

向一个已经存在的队列中，指定一个存在的节点位置前插入新元素

```
127.0.0.1:6379> LINSERT list before value value11
(integer) 4
127.0.0.1:6379> LRANGE list 0 -1
1) "value11"
2) "value"
3) "value11"
4) "value"
```

`注意`:当指定的节点在队列中存在多个时,队列按照从头到尾的优先级方式进行匹配

LLEN 命令

`语法`: LLEN key
`作用`: 返回队列中元素的个数
`返回`: 队列不存在返回`0`,存在则返回队列中元素个数

```
127.0.0.1:6379> LLEN listlist
(integer) 0
127.0.0.1:6379> LLEN list
(integer) 4
```

LINDEX 命令

`语法`: LINDEX key index
`作用`: 返回指定队列指定下标`index`处的元素
`返回`: 队列不存在，返回`NULL`，`index`大于队列元素个数，也返回`NULL`,否则返回对应的节点元素。`index`如果小于0,则下标逆向遍历

队列不存在的情况

> 127.0.0.1:6379> LINDEX list111 1
> (nil)

指定下标的元素不存在

> 127.0.0.1:6379> LINDEX list 100
> (nil)

获取指定下标的元素

> 127.0.0.1:6379> LINDEX list 1
> "value"

LSET 命令

`语法`: LSET key index value
`作用`: 设置队列指定下标出的元素为新值`value`
`返回`: 队列不存在或下标越界均返回`错误`,否则设置成功返货`OK`

队列不存在

> 127.0.0.1:6379> LSET listlist 2 value
> (error) ERR no such key

下标越界

> 127.0.0.1:6379> LSET list 100 value
> (error) ERR index out of range

成功设置

> 127.0.0.1:6379> LRANGE list 0 -1

```
1) "value11"
2) "value"
3) "value11"
4) "value"
127.0.0.1:6379> LSET list 0 value123
OK
127.0.0.1:6379> LRANGE list 0 -1
1) "value123"
2) "value"
3) "value11"
4) "value"
```

[LR]POP 命令

`语法`: [LR]POP key
`作用`: 从队列的头或未弹出节点元素(返回该元素并从队列中删除)
`返回`: 成功则返回元素,失败则返回 `NULL`

失败

```
127.0.0.1:6379> LPOP listlist
(nil)
127.0.0.1:6379> LPOP list
"value123"
127.0.0.1:6379> RPOP list
"value"
```

LRANGE 命令

`语法`: LRANGE key start end

`作用`：返回队列指定区间的元素，`start`、`end`可以为负数,逆向遍历
`返回`：成功则返回元素,失败则返回`NULL`

队列不存在

> 127.0.0.1:6379> LRANGE listlist 0 -1 s (empty list or set)

区间指定错误

> 127.0.0.1:6379> LRANGE list 0 -1
> 1) "val6"
> 2) "val5"
> 3) "val4"
> 4) "val3"
> 5) "val2"
> 6) "val1"
> 127.0.0.1:6379> LRANGE list 100 200
> (empty list or set)

获取指定区间的值

> 127.0.0.1:6379> LRANGE list 2 4
> 1) "val4"
> 2) "val3"
> 3) "val2"
> 127.0.0.1:6379> LRANGE list -4 -2
> 1) "val4"
> 2) "val3"
> 3) "val2"

获取整个队列的值

> 127.0.0.1:6379> LRANGE list 0 -1
> 1) "val6"

```
2) "val5"
3) "val4"
4) "val3"
5) "val2"
6) "val1"
```

LTRIM 命令

`语法`：LTRIM key start end
`作用`：截取队列指定区间的元素,其余元素都删除
`返回`：成功则返回 `OK`

测试链表元素值

```
127.0.0.1:6379> LRANGE list 0 -1
1) "val6"
2) "val5"
3) "val4"
4) "val3"
5) "val2"
6) "val1"
```

截取指定区间的值

```
127.0.0.1:6379> LTRIM list 2 4
OK
127.0.0.1:6379> LRANGE list 0 -1
1) "val4"
2) "val3"
3) "val2"
```

区间值错误

```
127.0.0.1:6379> LTRIM list -1 0
OK
127.0.0.1:6379> LRANGE list 0 -1
(empty list or set)
```

区间值错误，导致整个链表为空

LREM 命令

`语法`: LTRIM key toremove value
`作用`: 删除队列中指定值的节点，删除方式由`toremove`决定,
`toremove`大于`0`则从头遍历删除,最多删除`toremove`个节点;
`toremove`小于`0`则从尾遍历删除,最多删除`toremove`个节点; 为`0`则删除所有值相同的节点
`返回`: 返回删除的节点个数

从头删除指定值为`val`的节点

```
127.0.0.1:6379> LRANGE list 0 -1
1) "val"
2) "val1"
3) "val"
4) "val2"
5) "val"
6) "val3"
7) "val"
8) "val4"
9) "val"
10) "val5"
127.0.0.1:6379> LREM list 3 val
(integer) 3
```

```
127.0.0.1:6379> LRANGE list 0 -1
1) "val1"
2) "val2"
3) "val3"
4) "val"
5) "val4"
6) "val"
7) "val5"
```

队列中值为val的节点数大于需要删除的数，值删除指定数量的节点，如果队列中节点数量不足呢?

```
127.0.0.1:6379> LREM list 3 val
(integer) 2
127.0.0.1:6379> LRANGE list 0 -1
1) "val1"
2) "val2"
3) "val3"
4) "val4"
5) "val5"
```

`toremove`小于0的情况与大于0的情况类似，不做介绍，下面简单介绍下为0的情况

```
127.0.0.1:6379> LRANGE list 0 -1
1) "val"
2) "val1"
3) "val"
4) "val2"
5) "val"
6) "val3"
7) "val"
```

```
8) "val4"
9) "val"
10) "val5"
127.0.0.1:6379> LREM list 0 val
(integer) 5
127.0.0.1:6379> LRANGE list 0 -1
1) "val1"
2) "val2"
3) "val3"
4) "val4"
5) "val5"
```

RPOPLPUSH 命令

`语法`：RPOPLPUSH srcKey dstKey
`作用`：将源队列`src`的尾部元素插入`dst`队列的头部
`返回`：成功返回插入的元素

源队列

```
127.0.0.1:6379> LRANGE src 0 -1
1) "val1"
2) "val2"
3) "val3"
```

目标队列(可以不存在的,如果不存在则会创建一个新的)

```
127.0.0.1:6379> LRANGE dst 0 -1
(empty list or set)
```

执行命令

```
127.0.0.1:6379> RPOPLPUSH src dst
"val3"
127.0.0.1:6379> LRANGE src 0 -1
1) "val1"
2) "val2"
127.0.0.1:6379> LRANGE dst 0 -1
1) "val3"
```

B[LR]POP 命令

`语法`: B[LR]POP key1 [key2 ...] timeout
`作用`: 从一个或多个队列的头或尾部弹出一个元素,如果指定的队列集合中有一个队列有元素则立即返回数据，否则阻塞等待`timeout`时间,如果在指定时间内,队列集合中有元素新增,则该操作返回。否则超时时间到，自动返回
`返回`: 成功返回元素以及元素所属队列,失败返回错误

队列集合中有元素

```
127.0.0.1:6379> BLPOP src dst 30
1) "src"
2) "val1"
```

队列集合中没有元素且超时

```
127.0.0.1:6379> BLPOP src dst 30
(nil) (30.04s)
```

队列集合中没有元素,阻塞过程中队列集合中新增了元素

```
127.0.0.1:6379> BLPOP src dst 300
1) "src"
```

> 2) "kkk"
> (6.67s)
> 新增元素操作
> 127.0.0.1:6379> rpush src kkk
> (integer) 1

阻塞的队列尾部操作类似，不做介绍

BRPOPLPUSH 命令

`语法`：BRPOPLPUSH src dst timeout
`作用`：将指定队列`src`尾部的元素插入到`dst`头部，如果队列`src`为空，则操作阻塞`timeout`时间，在阻塞时间内,如果队列`src`有新增元素，则将`src`队列尾部元素插入到`dst`头部并返回，否则超时返回
`返回`：返回操作的元素

源队列不为空

> 127.0.0.1:6379> BRPOPLPUSH src dst 300
> "kkkk"

源队列为空

> 127.0.0.1:6379> BRPOPLPUSH src dst 300
> "jjj"
> (11.68s)

新增操作

> 127.0.0.1:6379> rpush src jjj
> (integer) 1

总结

`List`是用得比较多的数据结构之一,使用list可以轻易的构造队列，比如存储消息队列之类的，以完成`FIFO`或`FILO`的功能。