链滴

Redis 五大基础数据结构

作者：DongXiaokai0819

原文链接：https://ld246.com/article/1570239796909

来源网站：链滴

许可协议：署名-相同方式共享 4.0 国际 (CC BY-SA 4.0)

《Redis深度历险》读书笔记 -- 之基本数据结构

markdown语法学习：https://www.jianshu.com/p/191d1e21f7ed

一： redis安装与启动

redis下载与安装参考 博客：https://www.cnblogs.com/jylee/p/9844965.html

在将redis解压缩后的目录下，打开cmd

redis服务**启动**命令： redis-server.exe redis.windows.conf

连接redis：redis-cli.exe -h 127.0.0.1 -p 6379

或者直接在文件夹中打开**redis-cli.exe**

如果没配置环境变量的话连接redis的指令，必须在redis解压路径下打开cmd

二： Redis基本数据类型

参考教程：https://www.runoob.com/redis/redis-data-types.html

参考书籍：https://book.douban.com/subject/30386804/

我这不能算参考了，基本上算是比着书敲了一遍，第一次写博文，也分不太清重点不重点的，见谅。谅。。

1、String（字符串）

- 简介：String是redis最基本，最简单的数据类型，内部就是一个字符数组 。redis所有的数据结构是以唯一的key字符串作为名称，然后通过这个唯一的key值来获取相应的value数据。（key都是字串，value有五种表示）**key-value**，最大可存储512M
- 特性： String可以包含任何数据，二进制安全，如jpg图片类型，java序列化对象
- 应用： 缓存用户信息

Redis的字符串是动态的、可以修改的，内部机构的实现类似于Java的ArrayList，采用**预分配冗余空间**方式来减少内存的频繁分配

基本操作

键值对：相当于字典的key、value，支持简单的crud，下面的name--key codehole--value

```
127.0.0.1:6379> set name codehole
OK
127.0.0.1:6379> get name
"codehole"
127.0.0.1:6379> exists name
(integer) 1
127.0.0.1:6379> del name
(integer) 1
127.0.0.1:6379> get name
```

```
(nil)
```

批量键值对： 可对多个字符串进行批量操作，节省网络耗时开销

```
127.0.0.1:6379> set name1 codehole
OK
127.0.0.1:6379> set name2 holycoder
OK
127.0.0.1:6379> mget name1 name2 name3
1) "codehole"
2) "holycoder"
3) (nil)
127.0.0.1:6379> mset name1 boy name2 girl name3 unkonwn
OK
127.0.0.1:6379> mget name1 name2 name3
1) "boy"
2) "girl"
3) "unkonwn"
```

过期和set命令扩展： 可对key设置过期时间，到时间自动删除，常用做控制缓存的实效时间

```
127.0.0.1:6379> set name codehole
OK
127.0.0.1:6379> expire name 5 #设置name的过期时间5秒
(integer) 1
127.0.0.1:6379> get name
(nil)
127.0.0.1:6379> setex name 5 codehole #set 与expire 结合起来用
OK
127.0.0.1:6379> get name
(nil)
127.0.0.1:6379> setnx name codehole #如果name不存在就执行set创建
(integer) 1
127.0.0.1:6379> get name
"codehole"
127.0.0.1:6379> setnx name holycoder #如果存在就不执行
(integer) 0
127.0.0.1:6379> get name
"codehole"
```

计数： 若value是整数， 还可进行自增操作，范围在signed long最大值与最小值之间

```
127.0.0.1:6379> set age 30
OK
127.0.0.1:6379> incr age
(integer) 31
127.0.0.1:6379> incrby age 5
(integer) 36
127.0.0.1:6379> incrby age -5
(integer) 31
```

基本操作整理

- set key value　储存一个键值对

- get key 根据key获得该key所对应的value
- exists key 判断该键值对是否存在
- del key 根据key删除键值对
- mset key1 value1 key2 value2... 储存多个key value
- mget key1 key2 根据多个key ，返回一个value列表
- expire key second 设置某个key的过期时间
- setex key second value 等价于set+expire
- setnx key value如果key存在就不执行，不存在则执行
- incr keyvalue 为整数时 +1
- incrby key +-number value +-number

2、Hash（哈希字典）

在看这一部分时，讲到了rehash，以及redis中的rehash与java中rehash的区别，这里没看懂，也不楚rehash是什么，回头需要再补一补。

- 简介： 键值对集合，相当于Java语言中的HashMap，无序字典，数组+链表的二维结构
- 特性： 适合存储对象，可以只更新map中的某个属性值而不用取出整个map
- 应用： 可以用作存储用户信息，hash可以对用户结构中的每个字段单独存储，获取时可按需获取

hash结构的存储消耗要高于单个字符串，到底要使用hash还是字符串需要根据实际情况再三权衡

基本操作

```
127.0.0.1:6379> hset books java "think in java"
(integer) 1
127.0.0.1:6379> hset books golang "concurrency in go"
(integer) 1
127.0.0.1:6379> hset books python "python cookbook"
(integer) 1
127.0.0.1:6379> hgetall books
1) "java"
2) "think in java"
3) "golang"
4) "concurrency in go"
5) "python"
6) "python cookbook"
127.0.0.1:6379> hlen books
(integer) 3
127.0.0.1:6379> hget books java
"think in java"
127.0.0.1:6379> hset books golang "learning go programming"
(integer) 0
127.0.0.1:6379> hget books golang
"learning go programming"
```

```
127.0.0.1:6379> hmset books java "effective java" python "learning python"
OK
```

基本操作整理

- hset hashKey key value 储存一个hash 如果字符串包含空格，要用引号引起来，若hset 相同的ke的value ，则视为更新操作
- hgetall hashKey 返回hash内所有key value（key与value间隔出现）
- hlen hashKey 返回hash的长度
- hget hashKey key 从hashKey内获取key所对应的value
- hmset hashKey key1 value1 key2 value2 批量set
- hincrby hashKey key number value 是数字的的可进行计数

3、List(列表)

- 简介： 链表（双向链表）,相当于java中的LinkedList
- 特性： 增删快，时间复杂度为O(1)，但是索引定位很慢，为O(n)， 提供了操作某一段元素的API
- 应用： 最新的消息，排行榜等时间线相关的，常用做异步队列

当列表弹出了最后一个元素之后，该数据结构被自动删除，内存被回收

基本操作

右边进左边出：队列先进先出，常用做消息排队和异步逻辑处理，确保了元素的访问顺序性

```
127.0.0.1:6379> rpush books python java golang
(integer) 3
127.0.0.1:6379> llen books
(integer) 3
127.0.0.1:6379> lpop books
"python"
127.0.0.1:6379> lpop books
"java"
127.0.0.1:6379> lpop books
"golang"
127.0.0.1:6379> lpop books
(nil)
```

右边进右边出：栈后进后出，业务中并不太常见

```
127.0.0.1:6379> rpush books python java golang
(integer) 3
127.0.0.1:6379> rpop books
"golang"
127.0.0.1:6379> rpop books
"java"
127.0.0.1:6379> rpop books
"python"
```

```
127.0.0.1:6379> rpop books
(nil)
```

慢操作

```
127.0.0.1:6379> rpush books python java golang
(integer) 3
127.0.0.1:6379> lindex books 1 #同java链表的get（index）,需要对链表进行遍历
"java"
127.0.0.1:6379> lrange books 0 -1 #获取所有元素
1) "python"
2) "java"
3) "golang"
127.0.0.1:6379> ltrim books 1 -1 #保留区间内值
OK
127.0.0.1:6379> lrange books 0 -1
1) "java"
2) "golang"
127.0.0.1:6379> ltrim books 1 0 #清空列表，因为长度为负
OK
127.0.0.1:6379> llen books
(integer) 0
```

基本操作整理

- rpush key value1 value2 value3 储存一个列表，从右边进
- llen key 返回列表长度
- lpop key 从key列表中左边弹出一个值，并清内存
- rpop 从列表右边弹出一个值，并清内存
- lindex key index 根据index获得key列表中的值，O（n）
- lrange key key findex eindex 根据区间获取列表值，返回列表，O(n)
- ltrim books findex eindex 根据区间保留列表中的值

4、Set（集合）

- 简介： 哈希表的实现，元素不重复，无序，相当于java中的hashSet。
- 特性： 增删查复杂度都为O(1)，为集合提供了求交、并、差等操作
- 应用： 共同好友，统计网站独立ip，某活动的中奖id

内部实现相当于一个特殊的字典，字典中所有的value都是一个值null。

当集合中的最后一个元素被移除之后秘书局结构被自动删除，内存被回收

基本操作

```
127.0.0.1:6379> sadd books python
(integer) 1
127.0.0.1:6379> sadd books python
```

```
(integer) 0
127.0.0.1:6379> sadd books java golang
(integer) 2
127.0.0.1:6379> smembers books
1) "golang"
2) "java"
3) "python"
127.0.0.1:6379> sismember books java
(integer) 1
127.0.0.1:6379> sismember books rust
(integer) 0
127.0.0.1:6379> scard books
(integer) 3
127.0.0.1:6379> spop books
"python"
127.0.0.1:6379> scard books
(integer) 2
```

基本操作整理

- sadd key value 储存一个set
- smembers books 返回set内所有值，无序
- sismember key value 查询某个value是否存在，存在返回1，否则返回0
- scard key 获取set长度
- spop key 从set中弹出一个值

5、zSet（有序集合/有序列表）

- 简介： 将Set中的value元素增加了一个权重score元素，set内元素按score排序。类似于java中的sotedSet和HashMap的结合体
- 特性： 数据插入集合时，已经进行了排序
- 应用： 排行榜 ， 储存学生成绩 ， 粉丝列表 ， 带权重的消息队列

内部实现是一种叫作“跳跃列表”的数据结构

zset中最后一个value被移除后，数据结构被自动删除，内存被回收

基本操作

```
127.0.0.1:6379> zadd books 9.0 "think in java"
(integer) 1
127.0.0.1:6379> zadd books 8.9 "java concurrency"
(integer) 1
127.0.0.1:6379> zadd books 8.6 "java cookbook"
(integer) 1
127.0.0.1:6379> zrange books 0 -1
1) "java cookbook"
2) "java concurrency"
```

```
3) "think in java"
127.0.0.1:6379> zrevrange books 0 -1
1) "think in java"
2) "java concurrency"
3) "java cookbook"
127.0.0.1:6379> zcard books
(integer) 3
127.0.0.1:6379> zscore books "java concurrency"
"8.9000000000000004"
127.0.0.1:6379> zrank books "java concurrency"
(integer) 1
127.0.0.1:6379> zrangebyscore books 0 8.91
1) "java cookbook"
2) "java concurrency"
127.0.0.1:6379> zrangebyscore books -inf 8.91 withscores
1) "java cookbook"
2) "8.5999999999999996"
3) "java concurrency"
4) "8.9000000000000004"
127.0.0.1:6379> zrem books "java concurrency"
(integer) 1
127.0.0.1:6379> zrange books 0 -1
1) "java cookbook"
2) "think in java"
```

基本操作整理

- zadd key score value　储存一个zset ,传入score
- zrange key 0 -1 按score排序列出，参数区间为排名范围
- zrevrange key 0 -1 按score逆序列出，参数区间为排名范围
- zcard key 获取zset长度
- zscore key value　获取指定value 的score ，内部使用double类型进行存储score
- zrank key value获取value所在排名
- zrangebyscore key findex eindex根据分值区间遍历zset
- zrangebyscore key -inf index withscores根据分值区间遍历zset，同时返回分值，-inf为负无穷大
- zrem key value删除指定value

今日未解决问题

1. redis中的rehash
2. redis中的rehash与java中rehash的区别
3. 跳跃列表