

Redis7核心数据结构

实战版

-- 楼兰



Redis7有哪些数据结构

核心版

Core

Bitmap

Cluster management

Connection management

Generic

Geospatial indices

Hash

HyperLogLog

List

Pub/Sub

Scripting and functions

Server management

Set

Sorted set

Stream

String

Transactions

扩展版

Stack

Bloom filter

Cuckoo filter

Count-min sketch

JSON

Search and query

Auto-suggest

T-digest

Time series

Top-k



永远的神: help

To get help about Redis commands type:

- "help @<group>" to get a list of commands in <group>

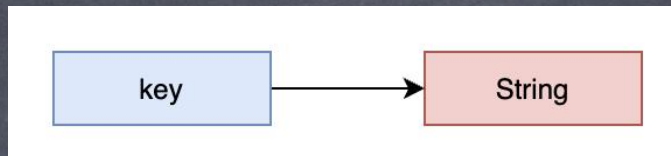
- "help <command>" for help on <command>

- "help <tab>" to get a list of possible help topics

- "quit" to exit



String结构



• 字符串常用操作

SET key value

MSET key value [key value ...]

SETNX key value

GET key

MGET key [key ...]

DEL key [key ...]

EXPIRE key seconds

//存入字符串键值对

//批量存储字符串键值对

//存入一个不存在的字符串键值对

//获取一个字符串键值

//批量获取字符串键值

//删除一个键

//设置一个键的过期时间(秒)

• 原子加减

INCR key

DECR key

INCRBY key increment

DECRBY key decrement

//将key中储存的数字值加1

//将key中储存的数字值减1

//将key所储存的值加上increment

//将key所储存的值减去decrement



String常见应用场景

- 单值缓存

SET key value

APPEND key value

- 对象缓存

1) set user:1 '{"name":"roy","balance":1888}'

2) MSET user:1:name roy user:1:balance 1888

MGET user:1:name user:1:balance

- 分布式锁

SETNX product:10001 true

//返回1代表获取锁成功

SETNX product:10001 true

//返回0代表获取锁失败

。。。 执行业务操作

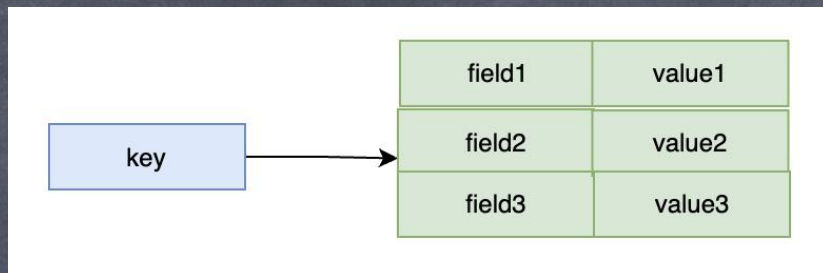
DEL product:10001

//执行完业务释放锁

SET product:10001 true ex 10 nx //防止程序意外终止导致死锁



Hash结构



- Hash常用操作

HSET key field value

HSETNX key field value

HMSET key field value [field value ...]

HGET key field

HMGET key field [field ...]

HDEL key field [field ...]

HLEN key

HGETALL key

HINCRBY key field increment

//存储一个哈希表key的键值

//存储一个不存在的哈希表key的键值

//在一个哈希表key中存储多个键值对

//获取哈希表key对应的field键值

//批量获取哈希表key中多个field键值

//删除哈希表key中的field键值

//返回哈希表key中field的数量

//返回哈希表key中所有的键值

//为哈希表key中field键的值加上增量increment



Hash应用场景

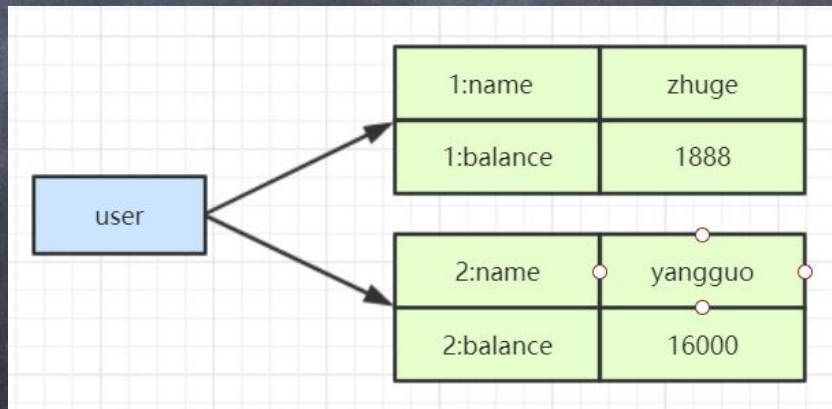
- 对象缓存

HSET user:1 name roy balance 1888

HMGET user:1 name balance 1888

HSET user 1:name roy 1:balance 1888

HMGET user 1:name 1:balance



Hash应用场景

- 电商购物车

- 1) 以用户id为key
- 2) 商品id为field
- 3) 商品数量为value

- 购物车操作

- 1) 添加商品→hset cart:1001 10088 1
- 2) 增加数量→hincrby cart:1001 10088 1
- 3) 商品总数→hlen cart:1001
- 4) 删除商品→hdel cart:1001 10088
- 5) 获取购物车所有商品→hgetall cart:1001





Hash结构优缺点

- 优点

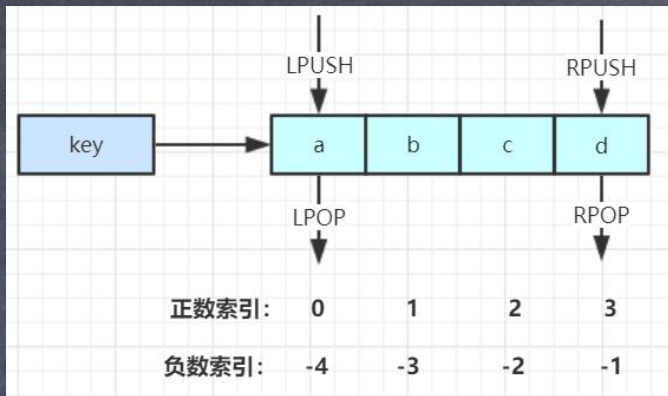
- 1) 同类数据归类整合储存，方便数据管理
- 2) 相比string操作消耗内存与cpu更小
- 3) 相比string储存更节省空间

- 缺点

- 1) 过期功能不能使用在field上，只能用在key上
- 2) Redis集群架构下不适合大规模使用



List类型



• List常用操作

LPUSH key value [value ...]

//将一个或多个值value插入到key列表的表头(最左边)

RPUSH key value [value ...]

//将一个或多个值value插入到key列表的表尾(最右边)

LPOP key

//移除并返回key列表的头元素

RPOP key

//移除并返回key列表的尾元素

LRANGE key start stop

//返回列表key中指定区间内的元素，区间以偏移量start和stop指定

BLPOP key [key ...] timeout

//从key列表表头弹出一个元素，若列表中没有元素，阻塞等待
timeout秒,如果timeout=0,一直阻塞等待

BRPOP key [key ...] timeout

//从key列表表尾弹出一个元素，若列表中没有元素，阻塞等待
timeout秒,如果timeout=0,一直阻塞等待



List类型应用场景

• 常用数据结构

Stack(栈) = LPUSH + LPOP

Queue(队列) = LPUSH + RPOP

Blocking MQ(阻塞队列) = LPUSH + BRPOP

• 常见应用场景

视频列表、签到列表

排队机

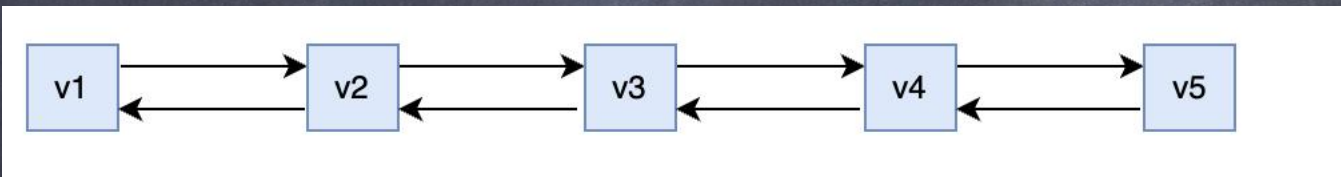
简化版的MQ

 <p>大数据流式计算设计精髓</p> <p>1857 2022-2-13</p>	 <p>大数据系列课程-用户画像项目实战</p> <p>823 2022-1-16</p>	 <p>楼兰Java宝藏图2021盘点与寄语</p> <p>149 2022-1-14</p>	 <p>Log4j2紧急漏洞</p> <p>619 2021-12-12</p>	 <p>大数据风控项目实战</p> <p>3018 2021-12-1</p>
 <p>ClickHouse全解析</p> <p>3668 2021-11-24</p>	 <p>Flink流式计算实战</p> <p>1075 2021-11-24</p>	 <p>短小精悍的ETL工具: Sqoop快速上手</p> <p>1703 2021-10-9</p>	 <p>HBase实战解析-轻松掌握大数据存储基石</p> <p>327 2021-9-10</p>	 <p>DDD微服务设计实战</p> <p>2586 2021-7-5</p>
 <p>大数据数仓基石-Hive3.12实战解析</p> <p>199 2021-6-21</p>	 <p>最简单 最全面 最硬核的 SpringBoot核心机制介绍</p> <p>334 2021-5-10</p>	 <p>Http与Https的区别</p> <p>349 2021-4-16</p>	 <p>Activiti7整合Spring快速精通</p> <p>455 2021-4-16</p>	 <p>Java面试专题第一期</p> <p>634 2021-4-11</p>

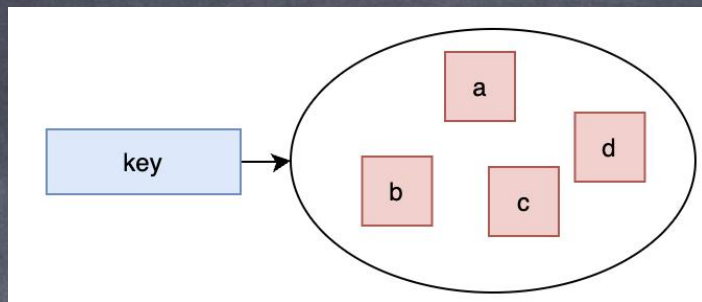


List类型注意点

- 1) 一个list的容量是2的32次方减1个元素，大概40多亿。但是在应用时，要注意大key的问题。
- 2) list的底层是一个双向链表，对双端的操作性能很高。但是通过索引下表直接操作某一个中间节点的性能就会比较低。



Set类型



Set常用操作

SADD key member [member ...]

SREM key member [member ...]

SMEMBERS key

SCARD key

SISMEMBER key member

SRANDMEMBER key [count]

SPOP key [count]

Set运算操作

SINTER key [key ...]

SINTERSTORE destination key [key ..]

SUNION key [key ..]

SUNIONSTORE destination key [key ...]

SDIFF key [key ...]

SDIFFSTORE destination key [key ...]

//往集合key中存入元素，元素存在则忽略，
若key不存在则新建

//从集合key中删除元素

//获取集合key中所有元素

//获取集合key的元素个数

//判断member元素是否存在于集合key中

//从集合key中选出count个元素，元素不从key中删除

//从集合key中选出count个元素，元素从key中删除

//交集运算

//将交集结果存入新集合destination中

//并集运算

//将并集结果存入新集合destination中

//差集运算

//将差集结果存入新集合destination中



Set应用场景

- 微信抽奖小程序

1) 点击参与抽奖加入集合

SADD key {userID}

2) 查看参与抽奖所有用户

SMEMBERS key

3) 抽取count名中奖者

SRANDMEMBER key [count] / SPOP key [count]





Set应用场景

- 微信微博点赞，收藏，标签

1) 点赞

SADD like:{消息ID} {用户ID}

2) 取消点赞

SREM like:{消息ID} {用户ID}

3) 检查用户是否点过赞

SISMEMBER like:{消息ID} {用户ID}

4) 获取点赞的用户列表

SMEMBERS like:{消息ID}

5) 获取点赞用户数

SCARD like:{消息ID}





Set应用场景

- 集合操作

SINTER set1 set2 set3 $\rightarrow \{c\}$ 共同关注的人

SUNION set1 set2 set3 $\rightarrow \{a,b,c,d,e\}$ 朋友圈的人

SDIFF set1 set2 set3 $\rightarrow \{a\}$ 推荐好友

微关系

共同关注(53)



黄章晋...



梁斌pe...



粉丝群



蛋NiKa

我关注的人也关注她(13)



享受独...



BjFant...



安卓大...



Tutuu...

查看更多 >



ZSet有序列表类型

			正数索引:	负数索引:
<div>key</div>	80	Steven	0	-4
	86	Eric	1	-3
	90	Lilei	2	-2
	98	Hanmeimei	3	-1

- ZSet常用操作

ZADD key score member [[score member] ...]

//往有序集合key中加入带分值元素

ZREM key member [member ...]

//从有序集合key中删除元素

ZSCORE key member

//返回有序集合key中元素member的分值

ZINCRBY key increment member

//为有序集合key中元素member的分值加上increment

ZCARD key

//返回有序集合key中元素个数

ZRANGE key start stop [WITHSCORES]

//正序获取有序集合key从start下标到stop下标的元素

ZREVRANGE key start stop [WITHSCORES]

//倒序获取有序集合key从start下标到stop下标的元素

- Zset集合操作

ZUNIONSTORE destkey numkeys key [key ...] //并集计算

ZINTERSTORE destkey numkeys key [key ...] //交集计算

ZSet应用场景

- Zset集合操作实现排行榜

- 1) 点击新闻

ZINCRBY hotNews:20190819 1 守护香港

- 2) 展示当日排行前十

ZREVRANGE hotNews:20190819 0 9 WITHSCORES

- 3) 七日搜索榜单计算

ZUNIONSTORE hotNews:20190813-20190819 7

hotNews:20190813 hotNews:20190814... hotNews:20190819

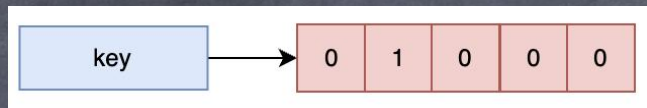
- 4) 展示七日排行前十

ZREVRANGE hotNews:20190813-20190819 0 9 WITHSCORES





Bitmap类型



- Bitmap常用操作

SETBIT key offset value

//将一个二进制数组的offset位置设置成value。value只能是0或者1。

GETBIT key offset

//返回一个二进制数组的offset位置的值。

BITCOUNT key [start end [BYTE|BIT]] //返回二进制数组中1的个数

BITPOS key bit [start [end [BYTE|BIT]]] //返回bitmap中第一个值为bit的offset位置。

BITOP AND|OR|XOR|NOT destkey key [key ...] //对两个bitmap做二进制的与或非计算。



Bitmap应用场景

- 每日签到

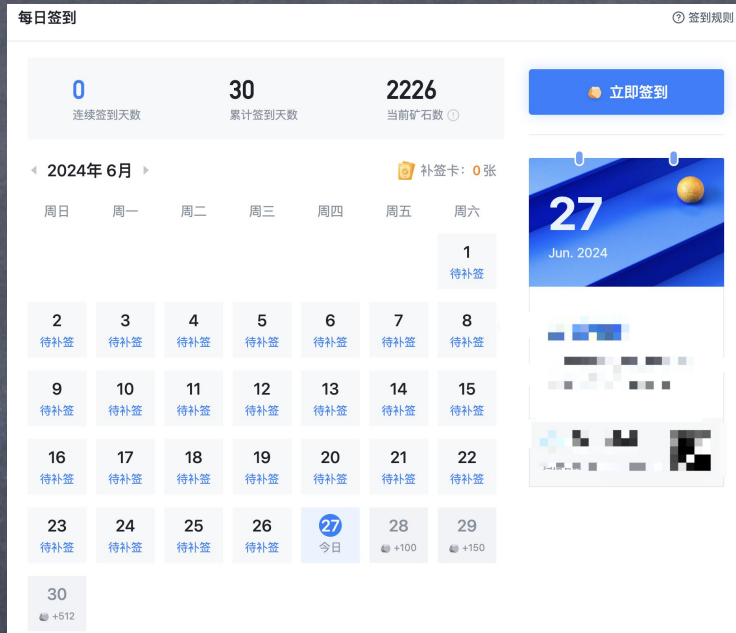
SETBIT dailycheck:1 100 1 1号用户第100天完成了签到

BITCOUNT dailycheck:1 统计1号用户的签到次数

BITPOS dailycheck:1 统计1号用户第一天签到的时间

- 优点

快速、高效、节省空间





Hyperloglog类型

- 作用介绍:

用于统计一个集合中不重复的元素个数。

典型应用场景例如根据用户访问记录统计网站的UV。

- Hyperloglog常用操作

PFADD visitlog 192.168.65.111 192.168.65.112 192.168.65.111 //添加用户访问记录

PFCOUNT visitlog //统计不同的独立访客

- Hyperloglog其他操作

PFMERGE destkey [sourcekey [sourcekey ...]] //将多个hyperloglong数据整合成一条记录。

Geo类型

- 常用操作



GEOADD key [NX|XX] [CH] longitude latitude member [longitude latitude member ...] //添加一个或多个地点

GEOPOS key [member [member ...]] //返回地址的经纬度

GEODIST key member1 member2 [M|KM|FT|MI] //计算两个地点之间的距离

GEORADIUS key longitude latitude radius M|KM|FT|MI [WITHCOORD] [WITHDIST] [WITHHASH] [COUNT count [ANY]] [ASC|DESC] [STORE key|STOREDIST key] //查询某个经纬度地址附近的地点

GEOSEARCH key FROMMEMBER member|FROMLONLAT longitude latitude BYRADIUS radius M|KM|FT|MI|BYBOX width height M|KM|FT|MI [ASC|DESC] [COUNT count [ANY]] [WITHCOORD] [WITHDIST] [WITHHASH] //查询某个地点附近的地点

Geo应用场景

- 获取经纬度

<https://api.map.baidu.com/lbsapi/getpoint/index.html>

- 添加商家地址

GEOADD changsha 113.017489 28.200454 火车站

112.96903 28.201195 橘子洲 113.017031 28.199706 赛格广

场 113.017004 28.197677 国储

- 查询距离

GEODIST changsha 火车站 橘子洲 M

- 查找火车站附近的景点

GEORADIUSBYMEMBER changsha 火车站 2 KM withdist

withcoord count 4 withhash





stream类型

- 作用介绍:

Redis版的MQ -- 阻塞队列 + pub/sub
了解即可，企业应用比较少。

- 常用操作

`XADD key [NOMKSTREAM] [MAXLEN|MINID [=|~] threshold [LIMIT count]] *|id field value [field value ...]`

//往对列的末尾发布一条消息

`XDEL key id [id ...]`

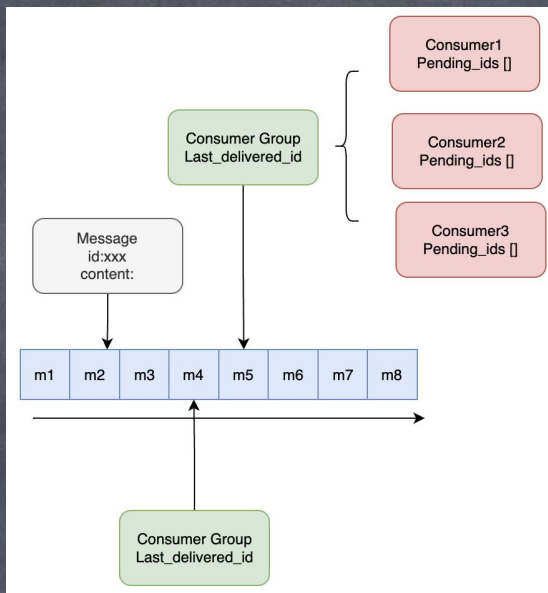
// 删除队列中的一条消息

`XLEN key`

//获取队列的长度

`XRANGE key start end [COUNT count]`

//查询队列中的消息





stream应用示例

- 创建队列，并添加消息 *表示让系统自动生成ID

```
XADD mystream * name loulan name roy name admin
```

- 查看对列消息 - 对列开始 + 对列结尾

```
XRANGE mystream - +
```

- 创建消费者组 0 从队列头部开始消费。\$ 从队列尾部开始消费

```
XGROUP CREATE mystream groupA 0
```

- 消费消息 > 表示从第一条未被消费过的消息消费。也可以指定ID

```
XREADGROUP GROUP groupA consumer1 count 2 STREAMS mystream >
```

- 查看消费者组的消费进度

```
XPENDING mystream groupA
```



补充: SpringBoot集成Redis

Maven依赖:

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-data-redis</artifactId>
</dependency>
```

核心配置:

```
spring:
  data:
    redis:
      host: 192.168.65.214
      port: 6379
      password: 123qweasd
      .....
```




补充: RestTemplate快速上手

记住一个对象:

@Resource

```
private RedisTemplate<String,Object> redisTemplate;
```

按组操作

```
redisTemplate.opsForValue().xxx //string类型
```

```
redisTemplate.opsForSet().xxx //set类型
```

```
redisTemplate.opsForHash().xxx //hash类型
```

```
redisTemplate.opsForList().xxx //list类型
```

```
redisTemplate.opsForZset().xxx //Zset类型
```

```
redisTemplate.opsForGeo().xxx //Geo类型
```

```
redisTemplate.opsForHyperLogLog().xxx //hyperLogLog类型
```


```
redisTemplate.opsForStream().xxx //stream类型
```

```
redisTemplate.opsForValue().setBit() //bit类型 为什么bit没有一个单独的操作类型?
```



补充: RedisTemplate中文乱码问题

```
@Bean
public RedisTemplate<String, Object> redisTemplate(RedisConnectionFactory redisConnectionFactory){
    RedisTemplate<String, Object> redisTemplate = new RedisTemplate<>();
    redisTemplate.setConnectionFactory(redisConnectionFactory);
    //    GenericJackson2JsonRedisSerializer jsonSerializer = new GenericJackson2JsonRedisSerializer();
    StringRedisSerializer stringRedisSerializer = new StringRedisSerializer();
    GenericToStringSerializer<String> genericToStringSerializer = new
GenericToStringSerializer<>(String.class);
    //指定key和value的序列化方式
    redisTemplate.setKeySerializer(stringRedisSerializer);
    redisTemplate.setValueSerializer(genericToStringSerializer);
    redisTemplate.setHashKeySerializer(stringRedisSerializer);
    redisTemplate.setHashValueSerializer(stringRedisSerializer);
    redisTemplate.afterPropertiesSet();
    return redisTemplate;
}
```

理解 » 熟练 » 记忆