# redis\_day01回顾

## Redis的特点

1、基于key-value的非关系型数据库  
2、基于内存存储，速度很快  
3、基于内存存储，经常当作缓存型数据库使用，常用信息缓存在redis数据库中  
4、单进程单线程

## 五大数据类型

1、字符串类型（string）  
2、列表类型（list）  
3、哈希类型（hash）  
4、集合类型（set）  
5、有序集合类型（sorted set）

### 字符串类型

# 设置key相关操作  
1、set key value  
2、set key value nx  
3、mset k1 v1 k2 v2 k3 v3  
4、set key value ex seconds  
5、set key value  
5、expire key 5  
5、pexpire key 5  
5、ttl key  
5、persist key  
# 获取key相关操作  
6、get key  
7、mget k1 k2 k3  
8、strlen key   
# 数字相关操作  
7、incrby key 步长  
8、decrby key 步长  
9、incr key  
10、decr key  
11、incrbyfloat key number

### 列表类型

# 插入元素相关操作  
1、LPUSH key value1 value2   
2、RPUSH key value1 value2  
3、RPOPLPUSH source destination  
4、LINSERT key after|before value newvalue  
# 查询相关操作  
5、LRANGE key start stop  
6、LLEN key  
# 删除相关操作  
7、LPOP key  
8、RPOP key  
9、BLPOP key timeout  
10、BRPOP key timeout  
11、LREM key count value  
12、LTRIM key start stop  
# 修改指定元素相关操作  
13、LSET key index newvalue

**思考：**

**Redis列表如何当做共享队列来使用？？？**

1、生产者消费者模型  
2、生产者进程在列表中 LPUSH | RPUSH 数据，消费者进程在列表中 BRPOP | BLPOP 数据

### Python与redis交互注意

1、r.set('name','Tom',ex=5,nx=True)  
2、r.mset({'user1:name':'Tom','user1:age':'25'})  
# 有元素时返回弹出元素,否则返回None  
3、r.brpop('mylist',3)

# redis\_day02笔记

## ==位图操作bitmap==

**定义**

1、位图不是真正的数据类型，它是定义在字符串类型中  
2、一个字符串类型的值最多能存储512M字节的内容，位上限：2^32  
# 1MB = 1024KB  
# 1KB = 1024Byte(字节)  
# 1Byte = 8bit(位)

**强势点**

可以实时的进行统计，极其节省空间。官方在模拟1亿2千8百万用户的模拟环境下，在一台MacBookPro上，典型的统计如“日用户数”的时间消耗小于50ms, 占用16MB内存

**设置某一位上的值（setbit）**

# 设置某一位上的值（offset是偏移量，从0开始）  
setbit key offset value  
# 获取某一位上的值  
GETBIT key offset  
# 统计键所对应的值中有多少个 1   
BITCOUNT key

**示例**

# 默认扩展位以0填充  
127.0.0.1:6379> set mykey ab  
OK  
127.0.0.1:6379> get mykey  
"ab"  
127.0.0.1:6379> SETBIT mykey 0 1  
(integer) 0  
127.0.0.1:6379> get mykey  
"\xe1b"  
127.0.0.1:6379>

**获取某一位上的值**

GETBIT key offset

127.0.0.1:6379> GETBIT mykey 3  
(integer) 0  
127.0.0.1:6379> GETBIT mykey 0  
(integer) 1  
127.0.0.1:6379>

**bitcount**

统计键所对应的值中有多少个 1

127.0.0.1:6379> SETBIT user001 1 1  
(integer) 0  
127.0.0.1:6379> SETBIT user001 30 1  
(integer) 0  
127.0.0.1:6379> bitcount user001  
(integer) 2  
127.0.0.1:6379>

**应用场景案例**

# 网站用户的上线次数统计（寻找活跃用户）  
 用户名为key，上线的天作为offset，上线设置为1  
# 示例  
 用户名为 user1:login 的用户，今年第1天上线，第30天上线  
 SETBIT user1:login 0 1   
 SETBIT user1:login 29 1  
 BITCOUNT user1:login

**代码实现**

## ==Hash散列数据类型==

* **定义**

1、由field和关联的value组成的键值对  
2、field和value是字符串类型  
3、一个hash中最多包含2^32-1个键值对

* **优点**

1、节约内存空间 - 特定条件下 【1，字段小于512个，2：value不能超过64字节】  
2、可按需获取字段的值

* **缺点（不适合hash情况）**

1，使用过期键功能：键过期功能只能对键进行过期操作，而不能对散列的字段进行过期操作  
2，存储消耗大于字符串结构

* **基本命令操作**

# 1、设置单个字段  
HSET key field value  
HSETNX key field value  
# 2、设置多个字段  
HMSET key field value field value  
# 3、返回字段个数  
HLEN key  
# 4、判断字段是否存在（不存在返回0）  
HEXISTS key field  
# 5、返回字段值  
HGET key field  
# 6、返回多个字段值  
HMGET key field filed  
# 7、返回所有的键值对  
HGETALL key  
# 8、返回所有字段名  
HKEYS key  
# 9、返回所有值  
HVALS key  
# 10、删除指定字段  
HDEL key field   
# 11、在字段对应值上进行整数增量运算  
HINCRBY key filed increment  
# 12、在字段对应值上进行浮点数增量运算  
HINCRBYFLOAT key field increment

**Hash与python交互**

# 1、更新一条数据的属性，没有则新建  
hset(name, key, value)   
# 2、读取这条数据的指定属性， 返回字符串类型  
hget(name, key)  
# 3、批量更新数据（没有则新建）属性,参数为字典  
hmset(name, mapping)  
# 4、批量读取数据（没有则新建）属性  
hmget(name, keys)  
# 5、获取这条数据的所有属性和对应的值，返回字典类型  
hgetall(name)  
# 6、获取这条数据的所有属性名，返回列表类型  
hkeys(name)  
# 7、删除这条数据的指定属性  
hdel(name, \*keys)

**Python代码hash散列**

**应用场景：微博好友关注**

1、用户ID为key，Field为好友ID，Value为关注时间  
 key field value  
 user:10000 user:606 20190520  
 user:605 20190521  
2、用户维度统计  
 统计数包括：关注数、粉丝数、喜欢商品数、发帖数  
 用户为key，不同维度为field，value为统计数  
 比如关注了5人  
 HSET user:10000 fans 5  
 HINCRBY user:10000 fans 1

**应用场景: redis+mysql+hash组合使用**

* **原理**
* 用户想要查询个人信息  
  1、到redis缓存中查询个人信息  
  2、redis中查询不到，到mysql查询，并缓存到redis  
  3、再次查询个人信息
* **代码实现**

**mysql数据库中数据更新信息后同步到redis缓存**

用户修改个人信息时，要将数据同步到redis缓存

## 集合数据类型（set）

* **特点**

1、无序、去重  
2、元素是字符串类型  
3、最多包含2^32-1个元素

* **基本命令**

# 1、增加一个或者多个元素,自动去重；返回值为成功插入到集合的元素个数  
SADD key member1 member2  
# 2、查看集合中所有元素  
SMEMBERS key  
# 3、删除一个或者多个元素，元素不存在自动忽略  
SREM key member1 member2  
# 4、元素是否存在  
SISMEMBER key member  
# 5、随机返回集合中指定个数的元素，默认为1个  
SRANDMEMBER key [count]  
# 6、弹出成员  
SPOP key [count]  
# 7、返回集合中元素的个数，不会遍历整个集合，只是存储在键当中了  
SCARD key  
# 8、把元素从源集合移动到目标集合  
SMOVE source destination member  
  
# 9、差集(number1 1 2 3 number2 1 2 4 结果为3)  
SDIFF key1 key2   
# 10、差集保存到另一个集合中  
SDIFFSTORE destination key1 key2  
  
# 11、交集  
SINTER key1 key2  
SINTERSTORE destination key1 key2  
  
# 11、并集  
SUNION key1 key2  
SUNIONSTORE destination key1 key2

**案例: 新浪微博的共同关注**

# 需求: 当用户访问另一个用户的时候，会显示出两个用户共同关注过哪些相同的用户  
# 设计: 将每个用户关注的用户放在集合中，求交集即可  
# 实现:  
 user001 = {'peiqi','qiaozhi','danni'}  
 user002 = {'peiqi','qiaozhi','lingyang'}  
   
user001和user002的共同关注为:  
 SINTER user001 user002  
 结果为: {'peiqi','qiaozhi'}

**python操作set**

**python代码实现微博关注**

## ==有序集合sortedset==

* **特点**

1、有序、去重  
2、元素是字符串类型  
3、每个元素都关联着一个浮点数分值(score)，并按照分值从小到大的顺序排列集合中的元素（分值可以相同）  
4、最多包含2^32-1元素

* 示例
* **一个保存了水果价格的有序集合**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 分值 | 2.0 | 4.0 | 6.0 | 8.0 | 10.0 |
| 元素 | 西瓜 | 葡萄 | 芒果 | 香蕉 | 苹果 |

**一个保存了员工薪水的有序集合**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 分值 | 6000 | 8000 | 10000 | 12000 |  |
| 元素 | lucy | tom | jim | jack |  |

**一个保存了正在阅读某些技术书的人数**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 分值 | 300 | 400 | 555 | 666 | 777 |
| 元素 | 核心编程 | 阿凡提 | 本拉登 | 阿姆斯特朗 | 比尔盖茨 |

* **有序集合常用命令**

# 在有序集合中添加一个成员 返回值为 成功插入到集合中的元素个数  
zadd key score member  
# 查看指定区间元素（升序)  
zrange key start stop [withscores]  
# 查看指定区间元素（降序）  
ZREVRANGE key start stop [withscores]  
# 查看指定元素的分值  
ZSCORE key member  
  
# 返回指定区间元素  
# offset : 跳过多少个元素  
# count : 返回几个  
# 小括号 : 开区间 zrangebyscore fruits (2.0 8.0  
zrangebyscore key min max [withscores] [limit offset count]  
# 每页显示10个成员,显示第5页的成员信息:   
# limit 40 10  
# MySQL: 每页显示10条记录,显示第5页的记录  
# limit 40,10  
# limit 2,3 显示: 第3 4 5条记录  
  
# 删除成员  
zrem key member  
# 增加或者减少分值  
zincrby key increment member  
# 返回元素排名  
zrank key member  
# 返回元素逆序排名  
zrevrank key member  
# 删除指定区间内的元素  
zremrangebyscore key min max  
# 返回集合中元素个数  
zcard key  
# 返回指定范围中元素的个数  
zcount key min max  
zcount salary 6000 8000   
zcount salary (6000 8000# 6000<salary<=8000  
zcount salary (6000 (8000#6000<salary<8000   
# 并集  
zunionstore destination numkeys key [weights 权重值] [AGGREGATE SUM|MIN|MAX]  
# zunionstore salary3 2 salary salary2 weights 1 0.5 AGGREGATE MAX  
# 2代表集合数量,weights之后 权重1给salary,权重0.5给salary2集合,算完权重之后执行聚合AGGREGATE  
   
# 交集：和并集类似，只取相同的元素  
ZINTERSTORE destination numkeys key1 key2 WEIGHTS weight AGGREGATE SUM(默认)|MIN|MAX

**python操作sorted set**