# 一 常用命令

redis命令不区分大小写

## 1 启动命令 与 服务端相关命令

redis-server 启动redis服务,参数 --help用来查看文档，-v 查看版本

按照生产环境部署后，推荐从配置文件启动:

sudo redis-server /etc/redis/redis.conf

sudo service redis start

sudo service redis stop

sudo servive redis restart

ps -ef|grep redis 查看redis进程

sudo kill -9 pid 杀死redis进程

## 2 客户端命令

连接： redis-cli -h 127.0.0.1 -p 6379

带密码：redis-cli -h 127.0.0.1 -p 6379 -a myPassword

ping: redis-cli Ping

redis-cli 启动redis客户端

redis-cli -p 6379 shutdown 关闭redis服务

select 1 启用1号库

注意：如果只是简单的解压redis，要使用上述命令，需要带上 ./文件目录名

## 3 其他命令

redis-benchmark redis性能测试工具

redis-check-aof AOF文件修复工具

redis-check-rdb RDB文件检索工具

# 二 数据操作

## 1 数据类型

redis基于key-value，每个数据都是一个键值对，键类型是字符串，值的类型有五种：

字符串string、

哈希hash、

列表list、

无序集合set、

有序集合zset。

## 2 键的命令

keys 键名 返回该键所对应的值

keys \* 返回所有键

keys ? 匹配一个字符

keys [] 匹配括号中任一字符，如：a[b-d]匹配的是 ab ac ad

keys \x 匹配字符x，\用于转义

exists key 判断键key是否存在，存在返回1，不存在返回0

del key 删除键，删除成功返回1，失败返回0

注意： del命令不支持通配符，但是可以利用linux管道：

删除所有以user:开头的键：redis-cli del `redis-cli keys “user:\*”`

type key 获取键值类型7

Redis没有对键命名进行约束，但一般采用 对象类型:对象ID:对象属性来命名一个键，如：user:1001:name 。 多个单词使用 . 分隔。

EXPIRE key seconds 设置过期时间（秒），没有则一直在，直到DEL移除

TTL key 查看有效时间

## 3 字符串string类型

### 3.1 string类型的设置与获取

string类型可以存储任何形式的字符串，包括二进制数据，最大容量为512M。

设置：

SET key value 设置键值

SETEX key seconds value 设置键值及过期时间，以秒为单位

MSET key value [key value ...] 设置多个键值

获取：

GET key 根据键获取值，如果不存在此键则返回nil

### 3.2 string类型多数据操作

MGET key1 key2 ... 根据多个键获取多个值

MSET k1 v1 [k2,v2...] 设置多个键值

### 3.3 string类型的运算

运算：值是数字 虽然用的是tring类型，但是如果是数字可以进行运算

INCR key 将key对应的value加1

注意：INCR命令创建的键的初始值都是1

INCRBY key increment 将key对应的value加整数值increment

DECR key 将key对应的value减1

DECRBY key decrement 将key对应的value减整数值increment

INCRBYFLOAT key increment 将key对应的value增加浮点精度的值

APPEND key value 值末尾添加value，键不存在则创建。返回值为总长度。

STRLEN key 获取字符串长度

注意：decrby 5 和 incrby -5 是一样的

注意：redis的操作是原子性的，即：当2个客户端同时读取到了值为1的一个key，并对该key执行了incr操作，虽然会执行2次操作，但是结果将会是2，不是3。

其他：

APPEND key value 追加值

STRLEN key 获取值长度

注意：每个字符串类型的键只能存储一个字符串，而如果使用字符串类型的键存储文章带来的问题是有很多的字符串，这时可以使用序列化函数，如JS中的JSON.stringify将其转换成一个字符串。但是这样做性能很低，一般推荐MessagePack序列化为二进制数据（http://msgpack.org）。

### 3.4 redis实现自增主键方式

对每一类对象使用名为 对象类型（复数形式）:count的键，如：users:count 来存储当前类型对象的数量，每增加一个新对象时都使用incr命令递增该键的值。

由于incr命令建立的键初始值是1，那么很容易得到对象总数，且该总数也是最后一个新增对象的ID。

### 3.5 位操作

一个字节由8个二进制位组成，redis提供了4个命令可以直接对二进制位操作。

比如使用redis设置 set foo bar 后，存储结构如下：

b a r

0110 0010 01100001 01110010

GETBIT foo 0 //返回结果0

GETBIT foo 6 //返回结果1

注意：如果索引长度超过了值得二进制位长度，返回默认值0

SETBIT foo 0 1 //设置第0位位1

注意：如果设置的位置超过了二进制位长度，则将中间的二进制位设置为0，如果设置了不存在的键的二进制位的值，则前面的位赋值为0

SETCOUNT foo //获取位值为1的二进制位个数，该案例返回10

SETCOUNT foo 0 1 //只统计前2个字节，该案例返回6

位运算命令BITOP，支持：AND OR XOR NOT运算

SET foo1 bar

SET foo2 aar

BITOP OR res foo1 foo2 //foo1 OR foo2 的值后存储在键res中

BITOPS foo 1 //获得键的第一个位值为1的偏移量

BITOPS foo 1 1 2 //第二个第三个参数分别制定要查询的起始字节和结束字节，该 案例表示要查询 第二个字节到第三个字节之间出现的第一个位 值为1的二进制位偏移量

注意：如果不设置结束字节且键值的所有二进制位都是1，当要查询值为0的二进制位偏移量时，返回结果是键值长度的下一个字位的偏移量。这是因为redis会认为键值长度之后的二进制位都是0。

位实战：记录用户的性别，0 1 ，那么100万个用户只会占用100kb。

注意：SETBIT时，如果当前键的值长度小于要设置的二进制位的偏移量时，redis会自动分配内存并将键值得当前长度到制定的偏移量之间的二进制位都设置为0。如果要分配的内存过大，则可能会造成服务器阻塞而无法接受同一时间其他请求。举例：在2014款mac上设置偏移量位232-1，即分配500M内存，需要耗费1秒时间。举例：如果网站用户的ID是从100000001开始，那么会造成10多M的浪费，正确做法是给每个用户减去100000001再存储。

## 4 散列 hash存储对象

hash一般可以理解为用来存储对象，将对象的键值一起做了存储。一个散列类型的键可以包含至多2的32次方-1个字段。

注意：字段值只能是字符串。同样其他数据类型也不支持数据类型嵌套，比如集合类型的每个元素都只能是字符串，不能是另一个集合或者散列。

设置：

HSET key field value 设置单个属性：hset person name 'lisi'

HMSET key field value field value ... 设置多个属性

HSETNX key field value 与HSET一致，但是如果字段存在，则不执行任何操作

案例：

hset person name ‘zs’

hset person age 13

hgetall person

获取：

HGET key field 获取一个属性的值

HMGET key field field ... 获取多个属性的值

HGETALL key 获取所有属性和值

HLEN key 返回包含属性的个数

HKEYS key 获取所有属性

HVALS key 获取所有属性的值

其它:

HEXISTS key field 判断属性是否存在，存在返回1，否则返回0，

键不存在也返回0

HDEL key field field ... 删除属性及值，返回值是删除的字段个数

HSTRLEN key field 返回值的字符串长度

HINCRBY key field increment 字段数字+1

hincrby person score 60

不存在score，则创建，且执行命令前的值为0，命令

返回值是增值后的字段值。

HSET命令不区分插入和更新，修改数据时无需事先判断字段是否存在来决定是更新还是插入。执行插入返回1，执行更新返回0，当然如果键不存在，则会创建键。

注意：在redis中，键的类型和命令有关，SET创建的键是字符串类型，HSET命令创建的hash类型，如果使用一种数据类型的命令操作另外一个数据类型的键，会提示错误：

ERR Operation against a key holding the wrong kind of value

## 5 列表 list 存储数组

列表类型可以用来存储一个有序的字符串列表，常用的操作是向列表两端添加元素，或者获得一个列表的片段。

列表的内部使用双向链表实现，所以向列表两端添加元素的复杂度为O(1)，获取越接近两端的匀速速度越快。案例：向一个有几千万元素的列表获取头部、尾部10条记录和向一个有20个元素的列表中获取头部或者尾部的10条记录的速度是一样的。

使用链表的代价是依据索引获取元素非常慢。需要从第一个元素一直查找到索引元素。

列表的使用场景：社交网站的新鲜事，只需要获取最新内容、记录日志、队列。

与散列类型的键能容纳的字段数相同，也是2的32次方-1个。

设置：

LPUSH key value 在头部插入数据

RPUSH key value 在尾部插入数据

以上两个命令返回值为增加元素后的长度，且支持多个值插入：

lpush number 2 5 先往左边插入2，再从最左边插入3

LSET key index value 设置指定索引的元素值，索引是基于0的下标，可以是 负数，表示偏移量从list尾部开始计数，如-1表示列表 的最后一个元素

LINSERT key BEFORE|AFTER pivot value

在一个元素的前|后插入新元素，返回值是插入后个数

首先会在列表中从左往右查找值为pivot的元素，然后插入

获取：

LPOP key 移除并且返回 key 对应的 list 的第一个元素

RPOP key 移除并返回存于 key 的 list 的最后一个元素

LRANGE key start stop 返回存储在 key 的列表里指定范围内的元素

start 和 end 偏移量都是基于0的下标

支持负索引，表示从右边开始计数，-1表示最右边第一个元素

LINDEX key index 返回列表里索引对应的元素

删除：

LREM key count value 删除列表中前count个值为value的元素，返回值是时间删除元 素个数，根据count值不同，LREM执行方式不同。

count > 0 从列表左边开始删除 count 个值为value的元素

count < 0 从列表右边开始删除 |count|个值为value的元素

count = 0 删除所有值为value的元素

其它：

LTRIM key start stop 删除索引范围外的元素，改为原集合的一个子集

start 和 end 偏移量都是基于0的下标

案例：每次写日志希望只保留最近100条日志，在插入日志后， 使用LRTIM logs 0 99

LLEN key 返回存储在 key 里的list的长度（个数）

RPOPLPUSH source destination

将元素从一个列表转义到另一个列表。先执行rpop，再执行lpush

即：先从source列表类型的键最右边弹出一个元素，然后加入 到destination列表类型键的左边，并返回这个元素的值。

实战：当列表类型作为队列时，source 和destination如果相同，RPOPLPUSH 命令会不断的将队尾的元素移动到队首。例如使用该特性实现一个网站监控系统：对垒存储需要监控的网站，程序不断使用RPOPLPUSH 循环取出一个网址来测试可用性，好处是在于在程序执行过程中仍然可以不断的向网址列表中加入新网址，整个系统易扩展，允许多个客户端同时处理队列。

## 6 set存储无序集合

元素为string类型，同样存储内容最多不能超过2的32次方-1个。

与列表不同的是：元素具有唯一性，不重复。

集合在redis内部是使用值为空的散列表实现，所以操作的复杂度为O（1），最方便的是多个集合类型键之间进行并集、交集、差集运算。

设置：

SADD key member member ... 添加元素，键不存在自动创建。因为在一个集合中不能 有相同元素，所以如果加入的元素已存在，则忽略，命 令返回值是成功加入元素个数

获取：

SMEMBERS key 返回key集合所有的元素

SCARD key 返回集合元素个数

删除：

SREM key member member 删除元素，返回成功删除的个数

SPOP key 从列表左边删除一个元素，返回该被删除元素

由于set无序，则会随机删除

运算：

SINTER key key ... 求多个集合的交集

SDIFF key key ... 求某集合与其它集合的差集

SUNION key key ... 求多个集合的合集

SINTERSTORE destination key ... 求交集后后存储结果于destination 中

SDIFFSTORE destination key ... 求差集后后存储结果

SUNIONSTORE destination key ... 求合集后后存储结果

其它：

SISMEMBER key member 判断元素是否在集合中，由于是个O（1）操作，无论 集合数目有多少，都能很快返回结果，存在返回1，不 存在返回0

SRANDMEMBER key count 随机从集合中获取key元素，可选参数count来确定获 取多个。count为正数时，获取count个不同元素，count 大于集合元素个数时，返回全部集合元素，为负数时， 获取 |count|个元素，元素有可能相同。

实战：博客的标签都是不相同的，且展示这些标签的时候排序没有要求，那么可以使用集合存储标签。

## 7 zset存储有序集合

元素为string类型，元素具有唯一性，不重复。

与set不同的是每个元素都会关联一个double类型的score，表示权重，通过权重将元素从小到大排序，所以zset可以随意调整位置，通过更改score。

元素的score可以相同。

由于有序集合使用散列表和跳跃表实现，读取中间的数据速度也很快，时间复杂度是O(log(N))

增加：

ZADD key score member score member ...

如果元素已存在，则修改分数为新分数，分数可以是正数、小数。

ZADD test +inf c +inf -inf分别表示正无穷和负无穷。

ZINCRBY key increment member 增加分数，返回值是更改后的分数，如果不存在该元素，则 先建立，并分数默认赋值为0

获取：

ZRANGE key start stop 按照分数大小，返回指定范围内的元素，可选参数是

withscores 不但返回元素，还会返回对应的分数

类似的命令还有 ZREVRANGE key start stop，按照分数从大到小排列。

如果元素分数相同，按照0-9-A-Z-a-z排序，中文按照UTF-8编码排序。

ZCARD key 返回元素个数

ZRANGEBYSCORE key min max [WITHSCORES] [LIMIT offset count]

按照元素分数从小到大返回分数在min-max的元素，如果不希望包含两端的元素使用(min max)，位置支持-inf和+inf

ZCOUNT key min max 返回有序集key中，score在min和max之间的个数

ZSCORE key member 返回有序集key中，成员member的score值

删除：

ZREM key member member... 删除一个或者多个元素，返回删除元素数量，不包括不 存在元素

ZREMRANGEBYRANK key start stop 按照元素分数排名从小到大范围删除元素

ZREMRANGEBYSCROE key min max 按照元素分数范围删除元素，返回值是删除元素数量

其他：

ZRANK key member 从小到大获得元素排名，最小排名0

ZREVARANK key member 从大到小获得元素排名，最大排名0

运算：

ZINTERSCORE destination numkeys key key key ... [WEIGHTS weight weight weight ...] [AGGREGATE SUM|MIN|MAX]

计算读个有序集合的交集，并将结果存储在destination键中，返回值为destination的元素个数。destination键中的元素的分数由AGGREGATE 参数决定。

当AGGREGATE 时SUM（默认）时，结果键中元素的分数是每个参与计算的集合中该元素的分数的和，是MIN时，是元素分数最小值，MAX同理。

通过WIGHTS参数可以设置每个集合的权重，每个集合在参与运算时分数会被乘以该集合的权重。

ZINTERSTORE命令与上面的命令用法一致，用来计算并集。

## 8 list set zset区别

list：有顺序的集合，元素不能重复，存储的集合按照我们自己添加的顺序排序；

set：没顺序的集合，元素可以重复，不依赖于添加、大小顺序去，且值可以重复；

zset：有顺序的集合，元素不能重复，根据权重大小从小到大排列；