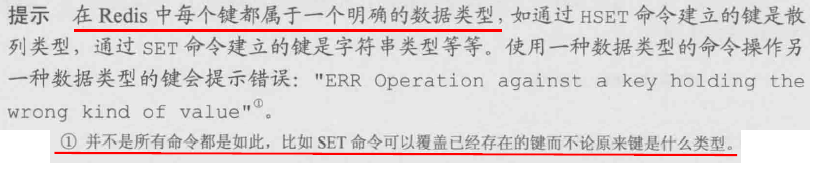
## 3 、redis的5种数据类型及相应命令

redis不区分命令大小写。



|  |
| --- |
| string 512m  一个散列类型键可包含至多232-1个字段  一个列表类型键最多能容纳232-1个元素  一个集合类型键最多能容纳232-1个元素 |

### 3.1、一些实用的基础命令

|  |  |
| --- | --- |
| keys pattern |  |
| exists key | 返回值：存在返回1，不存在返回0 |
| del key1 [key2 key3 ...] | 返回值：删除的键的个数 |
| type key | 返回值：string、hash、list、set、zset |

|  |  |
| --- | --- |
| **技巧：删除所有复合规则的键**（注意del不支持通配符） | |
| 方案1  （推荐） | 在linux命令行执行下面命令：  [root@tkafka ~]# redis-cli del `redis-cli keys "a\*"`  注意用的反单引号，将redis-cli keys "a\*"命令的结果作为redis-cli del命令的参数。 |
| 方案2 | 在linux命令行执行下面命令：  [root@tkafka ~]# redis-cli keys "a\*" | xargs redis-cli del  利用的是linux管道和xargs命令 |

### 3.2、字符串类型[string]

字符串类型是redis中最基本的数据类型，是其它4种数据类型的基础，例如列表类型是以列表的形式组织字符串，集合类型是用集合的形式组织字符串。

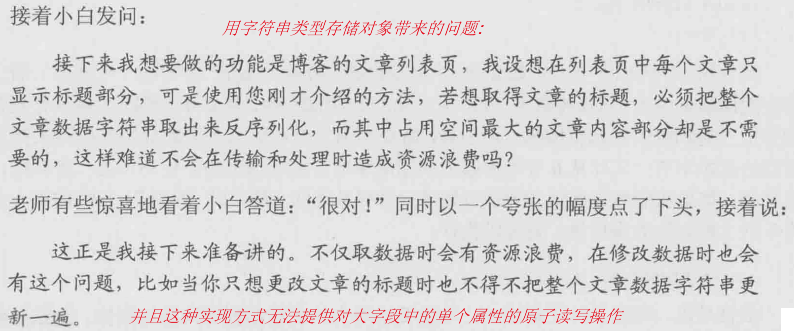
string可存储任何形式的字符串，比如数值形式的字符串，二进制形式的字符串，json格式的字符串等，也就是说string能存储数值、图片、json化对象等。

|  |  |  |
| --- | --- | --- |
| **基础命令** | | |
| set key value  get key |  | |
| mset key value [key2 value2 ...]  mget key [key2 ...] |  | |
| append key value | 描述：向键值的末尾追加value，如果键不存在则相当于set key value。  返回值：追加后的字符串长度。 | |
| strlen key | 你好(6) ab(2) | |
| **-- 当字符串类型存储的是【整数形式的字符串】时，才可以用的命令，否则报错：** | | |
| incr key  incrby key increment | 注意：若key不存在，incr命令会默认key的值为0，再进行递增；若key的值不是整数形式的字符串，redis会报错。  返回值：返回递增后的值。 | |
| decr key  decrby key decrement | 这两个指令功能都可以用 incrby key increment 实现 | |
| incrbyfloat key increment |  | |
| **--位操作命令:** | | |
| getbit key offset | |  |
| setbit key offset value | |  |
| bitcount key [startByteIndex] [endByteIndex] | | 统计字符串类型键中值是1的二进制位的个数。  可以加两个参数来限制统计的字节范围[0开始]。 |
| bitop operation destkey key1 key2 … keyN | | 对多个字符串类型键进行位运算，并将结果存储destkey参数指定的键中，operation可选AND、OR、XOR、NOT。 |
| bitpos key bitValue [startByteIndex] [endByteIndex] | | 获得指定键中第一个位值为0或1的二进制位的偏移量[0开始]。bitValue可选0、1。  可以加两个参数来限制查询的字节范围[0开始]。 |

**注意：包括incr在内的所有redis命令都是原子操作，无论多少客户端同时连接，都不会出现并发性的安全问题**。但是如果你希望在程序自己写一个方法来实现incr命令的效果，就得自己在代码层面保证该方法是线程安全的。

|  |
| --- |
| **实践** |
| **1、键命名实践**：【对象类型名:对象id:对象属性名】；如user:1:age  多个单词用“.”分隔，如user:2:first.name |
| 1. 文章访问量   对每篇文章都用一个键article:articleId:visit.count来记录该篇文章的访问量，访问一次就incr一次。 |
| 3、自增id  为一类对象如article定义一个键如articles:count，用于保存该类对象的数量。  每要增加一个该类对象时都先做一次incr，incr的返回值即可作为该新增对象的id。 |
| 4、存储文章数据（下节会讲这种实现方式的缺陷）  键名article:articleId:article.Data 键值：articleTitle,articleAuthor,aticleContent,articleTime序列化后的字符串。  获取文章数据时，获取键值后进行反序列化即可。 |
| 5、位操作命令实践 |

### 3.3、散列类型[hash]



redis是采用字典结构<key, value>存储数据的，在此之上，散列类型键的键值也是字典结构的**<key, <field, value>>**,其中的value只能是基础的字符串类型。**redis中的数据类型不支持数据类型嵌套，也就是说hash、list、set、zset中的元素只能是基础的字符串类型。**

**散列类型适合存储对象。键命名：对象类名:对象ID**。

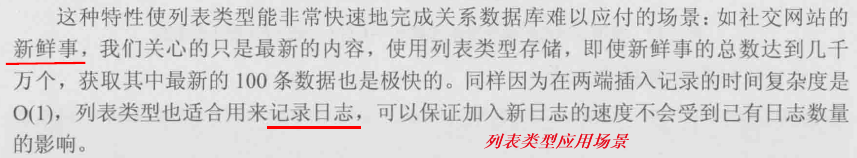
对于关系型数据库中的表，由于其是二维表结构的，表中每条记录拥有的字段是一致的，无法单独为某条记录增减属性而不影响其它记录；散列类型则没有此限制，例如user:1的字段为name、age、phone，但user:2的字段为name、sex、city。

|  |  |
| --- | --- |
| hset key field value  hsetnx key field value  hget key field | 调用hset时不用区分是插入属性还是更新属性，且当key不存在时，hset还会自动创建key。  返回值：1表示是执行的是插入操作，0表示执行的是更新操作。  hsetnx只在字段不存在时才进行赋值[not exists]。 |
| hmset key field value [field2 value2 ...]  hmget key field [field2 ...] |  |
| hexist key field | 返回值：1或0 |
| hdel key field [field2 …] |  |
| hincrby key field increment | 返回值为增长后的字段值。  当key不存在时，hincrby命令会自动创建key，并默认field在增长前的字段值为0。 |
| hgetall key  hkeys key  hvals key  hlen key | 很多语言的redisClient会将hgetall的返回结果封装成编程语言中的对象 |

|  |
| --- |
| **实践** |
| 1. 存储文章数据（本节开始有讲到用字符串来存储对象的缺点，这里实践下用散列来存储对象）   <article:articleId,<文章各字段名,字段值>> 字段有articleTitle,articleAuthor,aticleContent,articleTime |
| 1. 存储文章缩略名【slug】   **缩略名**用于构成文章网址的一部分，每篇文章的缩略名必须是唯一的，发布文章时需要验证用户输入的缩略名是否已被占用；系统还应提供根据缩略名获取文章id的功能。  <slug.to.id,<slugname,articleid>>用于存储文章缩略名和文章id之间的映射关系。”hexists slug.to.id slugname”可以验证缩略名是否已被占用；”hget slug.to.id slugname”可以根据缩略名获取文章id；另外修改文章缩略名时要修改<slug.to.id,<slugname,articleid>>中相应字段。 |

### 3.4、列表类型[list]

列表类型（list）内部使用**双向链表**结构实现的。向链表两端添加元素的时间复杂度为O(1)，获取元素时获取越接近两端的元素就越快；跟索引有关的操作较慢。

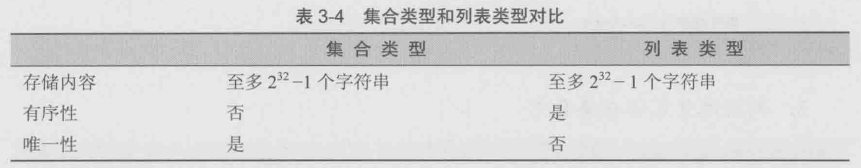


|  |  |
| --- | --- |
| 基础命令 | |
| lPush|rPush key value [value2 ...] | 返回值：增加元素后列表的长度 |
| lPop|rPop key | 返回值：被移除的元素值 |
| lLen key | 实现上，redis会直接读取现成的值，所以时间复杂度为O(1) |
| lRem key count value | 删除列表中前count个值为value的元素  返回值：实际删除的元素个数 |
| lInsert key before|AFTER pivot value | 描述：先从左到右的从列表中查找值为pivot的元素，然后根据第二个参数是BEFORE还是AFTER来决定将value插入到该元素之前还是之后。  返回值：插入后，列表中元素个数 |
| 用作数组【效率较低】 | |
| lRange key start end | 描述：获取列表片段**[**startIndex, endIndex**]**，索引从0开始  注意：lrange支持负索引，表示从最右边开始数，-1表示最右边第一个元素，即”lrange key 0 -1”可以获取列表中所有元素。 |
| lTrim key start end | 删除指定索引范围之外的所有元素 |
| lIndex key index |  |
| lSet key index value |  |
| 用做栈（**lpush + lpop 或 rpush + rpop**） | |
| 用作队列（**lpush + rpop 或 rpush + lpop**） | |
| rpoplpush srcQueue destQueue | 描述：从srcQueue列表类型键的右边弹出一个元素，然后将其加入到destQueue列表类型键的左边，并返回这个元素，当然整个过程是原子的。 **rpoplpush可用于在多个队列间传递数据**。 |

|  |
| --- |
| **实践** |
| 1、存储按时间排序的文章id列表  用列表类型键articles:list来存储文章id列表，发布新文章时要使用lpush将新文章的id加入此列表中，删除文章时要使用lrem把列表中的相应文章id移除掉，这样就可以用lrange来实现文章分页了。  “lrange articles:list (pageIndex-1)\*pageSize pageIndex\*pageSize-1” |
| 2、存储评论列表  用列表类型键article:articleId:comments来存储某篇文章的所有评论。 |

### 3.5、集合类型[set]

集合类型（set）内部是使用值为空的hashtable<E,~>实现的，常用的操作如add(E)、remove(E)、contains(E)时间复杂度都是O(1)。



|  |  |  |
| --- | --- | --- |
| 基础命令 | | |
| sAdd key member [member2 ...] | 如果key不存在则会自动创建  返回值：成功插入的元素数量（插入操作前集合中已存在的元素会被忽略插入，不参与计数） | |
| sRem key member [member2 ...] | 返回删除成功的元素个数 | |
| sMembers key | 返回集合中所有元素 | |
| sIsMember key member | 判断元素是否在集合中 | |
| scard key | 返回集合中元素个数 | |
| 随机操作 | | |
| sPop key | 从集合中随机弹出一个元素 | |
| sRandMember key [count] | 随机从集合中获取一个元素，可以指定count参数来随机获取多个元素。 | |
| 集合间运算命令 | | |
| sDiff key [key2 …] | | 差集（A-B：所有属于A但不属于B的元素）。  “sDiff A B C”表示(A-B)-C |
| sInter key [key2 …] | | 交集 |
| sUnion key [key2 …] | | 并集 |
| sDiffStore destCollection key1 [key2 ...] | | 这3个命令会将运算结果存储在destCollection键中而不会返回，常用于需要进行多步集合运算的场景中，如需要先进行差集再将结果和其它键计算交集。 |
| sInterStore destCollection key1 [key2 ...] | |
| sUnionStore destCollection key1 [key2 ...] | |

|  |
| --- |
| **实践** |
| 1、存储文章标签  由于一个文章的所有标签不会重复，且在展示时没要求标签顺序，所以可以用集合类型键article:articleId:tags来存储文章标签。 |
| 2、通过标签搜索文章  为每个标签定义一个集合类型键tag:tagname:articles来存储该标签下的文章ID列表。用sMember可得到一个标签下所有文章；用sInter可得到同属于某几个标签的文章。 |

### 3.6、有序集合类型[zset]

在集合类型的基础上，有序集合类型**为集合中的每个元素都关联了一个分数**，所以比集合类型多了一些**与分数有关的操作**。有序集合中的元素不能重复，但是不同元素的分数可以相同。

|  |  |
| --- | --- |
| **有序集合类型与列表类型** | |
| **相似点** | **区别** |
| 1)有序  2)可以获取某一范围的元素 | 1)实现方式不同    2)列表中不能简单地调整某个元素的位置，但是有序集合可以（通过更改这个元素的分数）  3)有序集合比列表更耗内存 |

|  |  |
| --- | --- |
| 基础命令 | |
| zAdd key score member [score2 member2 ..] | 描述：加入一个指定分数的元素，若元素已存在则更新该元素的分数。分数可以是整数、double、+inf（正无穷）、-inf（负无穷）。  返回值：新加入到集合中的元素个数 |
| zScore key member | 返回元素的分数 |
| zIncrBy key increment member | 描述：增加某个元素的分数，分数可为负数。  返回值：更改后的分数 |
| zCard key | 返回元素数量，类比sCard |
| zRem key member [member2 ...] | 返回成功删除的元素数量（不包含本来就不存在的元素） |
| 跟顺序相关的命令 | |
| zRange key start end [withScores] | zRange命令会**按照元素分数从小到大的顺序**返回**索引**从start到end之间的所有元素[start, end]，**zReRange则是从大到小**。加上**”withScores”**参数表示需要同时获取元素的分数。  zRange可类比列表的LRange，索引都是从0开始，负数索引表示从最右边开始往前数（-1表示最右边的元素）  zRange命令时间复杂度为**O(log(n+m))**，n为有序集合的基数，m为要返回的元素个数。  对于分数相同的元素，redis会按照**字典序(**0<9<A<Z<a<z**)**来进行排列。 |
| zRevRange key start end [withScores] |
| zRangeByScore key **min max**  [withScores] [limit offset count] | zRangeByScore命令按元素分数从小到大的顺序返回**分数**在min和max之间的所有元素[min, max]。  如果希望分数范围不包含端点值，可以在分数前加上”(”，例如”80 (100”表示[80, 100)。min和max支持无穷大+inf和-inf，例如”(80 +inf”表示80分以上。  limit offset count和在sql中的语义一样，表示在获得元素列表的基础上向后偏移offset个元素，然后获取前count个元素。 |
| zRevRangeByScore key **max min**  [withScores] [limit offset count] |
| zRemRangeByRank key start end | **按排名范围删除**，返回删除的元素数量 |
| zRemRangeByScore key min max | **按分数范围删除**，返回删除的元素数量 |
| zCount key min max | 获取指定分数范围的元素个数 |
| zRank key member | 获得指定元素的排名（0开始） |
| zRevRank key member |
| **集合间运算命令（用到再查）** | |
| zInterStore … | 计算有序集合的交集 |
| zUnionStore … | 计算有序集合的并集 |

|  |
| --- |
| **实践** |
| **1按文章点击量顺序获取文章列表**  集合类型键articles:article.visit.count，以文章ID作为集合元素，以文章访问量作为元素分数。每当一篇文章被访问时，就用”zIncrBy articles:article.visit.count 1 aticleId”来更新文章访问量。  这样，通过”zReRange articles:article.visit.count (pageIndex-1)\*pageSize pageIndex\*pageSize-1”命令即可实现按照文章点击量顺序获取文章列表。通过”zScore articles:article.visit.count articleId”可获取某篇文章的访问量。有了这个键，就不再用3.2节定义的字符串类型键article:articleId:visit.count来记录单个文章的访问量了。 |
| **2改进按时间顺序排列文章**  3.4节的列表类型键实现，在更改元素顺序上比较麻烦；为了能够自由更改文章发布时间，这里采用有序集合类型代替列表类型，以文章ID作为集合元素，文章发布的Unix时间(一个秒数)作为元素分数；通过修改元素的分数来实现更改文章发布时间，通过zRevRangeByScore来实现获取指定时间范围的文章列表。 |