链滴

Redis-02- 常用五大数据类型

作者：Anileh
原文链接：https://ld246.com/article/1652088045808
来源网站：链滴

常用命令查询网址
<h3 id="1-Redis键-key-">1.Redis 键（key）</h3>

<code>keys *</code> ： 查看当前库的所有 key
<pre><code class="highlight-chroma">127.0.0.1:6379> keys *
1) "k2"
2) "k1"
3) "k3"

</code></pre>

<code>exists key</code> ： 判断某个 key 是否存在
<pre><code class="highlight-chroma">127.0.0.1:6379> exists k3
(integer) 1
127.0.0.1:6379> exists k4
(integer) 0

</code></pre>

<code>type key</code>： 查看 key 的数据类型
<pre><code class="highlight-chroma">127.0.0.1:6379> type k1
string
127.0.0.1:6379> type k3
string

</code></pre>

<code>del key</code> ： 删除指定的 key 数据
<pre><code class="highlight-chroma">127.0.0.1:6379> del k2
(integer) 1

</code></pre>

<code>unlink key</code>： 根据 value 选择非阻塞删除

<blockquote>
仅将 keys 从 keyspace 元数据中删除，真正的删除会在后续异步操作
</blockquote>

<code>expire key 10</code>：为给定的 key 设置过期时间

<code>ttl key</code>:　　查看剩余过期时间 -1 表示永不过期，-2 表示已经过期
<pre><code class="highlight-chroma"><span class="highlight
cl">127.0.0.1:6379> ttl k1
(integer) -1

</code></pre>

<code>select key</code> ：　切换数据库

<code>dbsize</code> ：　　查看当前数据库 key 的数量
<pre><code class="highlight-chroma"><span class="highlight
cl">127.0.0.1:6379> dbsize
(integer) 2

</code></pre>

<code>flushdb</code>:　　清空数据库

<code>flushall</code>:　　通杀全部库

<h3 id="2-Redis字符串-string-">2.Redis 字符串(string)</h3>
<h4 id="2-1-简介">2.1 简介</h4>
String 是 Redis 最基本是的数据类型，可以理解为与 Memcached 一样的类型，一个 key 对应个 vale
String 类型是二进制安全的，意味着 Redis 的 String 可以包含任何数据，比如 jpg 的图片或序化的对象
String 类型是 Redis 最基本的数据类型，一个字符串 value 最多可以是 512M
<h4 id="2-2常用命令">2.2 常用命令</h4>
<code>get <key></code>:　　查询对应键值
<pre><code class="highlight-chroma"><span class="highlight
cl">127.0.0.1:6379> get k2
"v2"

</code></pre>
<code>append <key> <value></code> ：将给定的 value 追加到原值的末尾</p

<pre><code class="highlight-chroma"><span class="highlight
cl">127.0.0.1:6379> append k2 v22
(integer) 5
127.0.0.1:6379>
get k2
"v2v22"

</code></pre>
<code>strlen <key></code> ：　　key 的长度
<pre><code class="highlight-chroma"><span class="highlight
cl">127.0.0.1:6379> strlen k1

(integer) 2

</code></pre>
<code>setnx <key><value></code>： 在 key 不存在时，设置 key-value</p

<pre><code class="highlight-chroma"><span class="highlight
cl">127.0.0.1:6379> setnx k4 v4
(integer) 1

</code></pre>
<code>incr <key></code>： 将 Key 存储的数字值赠一，只能最数字值操
，如果为空，新增值为 1
<pre><code class="highlight-chroma"><span class="highlight
cl">127.0.0.1:6379> set k1 1
OK
127.0.0.1:6379>
incr k1
(integer) 2
127.0.0.1:6379>
get k1
"2"
</code></pre>
<code>decr <key></code>： 将 key 中存储的数字值减一，只能对数字值操
，如果为空，新增值为-1
<pre><code class="highlight-chroma"><span class="highlight
cl">127.0.0.1:6379> decr k1
(integer) 1
127.0.0.1:6379>
get k1
"1"

</code></pre>
<code>incrby / decrby <key><step></code>：将 key 中存储的数字值增减 自义步长、
<pre><code class="highlight-chroma"><span class="highlight
cl">127.0.0.1:6379> incrby k1 10
(integer) 11
127.0.0.1:6379>
get k1
"11"

</code></pre>
<h4 id="2-3-数据结构">2.3 数据结构</h4>
String 的数据结构为简单动态字符串，内部结构实现上类似于 Java 的 ArrayList，采用预分配冗空间的方法来减少内存的频繁分配。
<img src="https://ld246.com/images/img-loading.svg" alt="image.png" data-src="https:/b3logfile.com/file/2022/05/image-272587af.png?imageView2/2/interlace/1/format/jpg"></
>
如图所示，内部为当前字符串实际分配的空间 capacity 一般要高于实际字符串长度 len。当字串长度小于 1M 时，扩容都是加倍现有的空间，如果超过 1M，则每次只多扩容 1M。字符串的最大度是 512M.
<h3 id="3-Redis列表-List-">3 Redis 列表（List）</h3>
<h4 id="3-1-简介">3.1 简介</h4>
单键多值，Redis 列表是简单的字符串列表。

按照插入顺序排序。你可以添加一个元素到列表的头部（左边）或者尾部（右边）。
它的底层实际是个双向链表，对两端的操作性能很高，通过索引下标的操
中间的节点性能会较差。
<img src="https://ld246.com/images/img-loading.svg" alt="image.png" data-src="https:
/b3logfile.com/file/2022/05/image-38a4671f.png?imageView2/2/interlace/1/format/jpg"></
>
<h4 id="3-2-常用命令">3.2 常用命令</h4>
<code>lpush/rpush <key><value1><value2><value3> .... </code>
从左边/右边插入一个或多个值。
<code>lrange mylist 0 -1</code>：0 左边第一个，-1 右边第一个，（0-1 表示获取所有）<
p>
<pre><code class="highlight-chroma"><span class="highlight
cl">127.0.0.1:6379> lpush k1 1 2 3
(integer) 3
127.0.0.1:6379>
lrange k1 0 -1
1) "3"
2) "2"
3) "1"
127.0.0.1:6379>
rpush k2 4 5 6
(integer) 3
127.0.0.1:6379>
lrange k2 0 -1
1) "4"
2) "5"
3) "6"
</code></pre>
<code>llen <key></code>：获得列表长度
<pre><code class="highlight-chroma"><span class="highlight-line"><span class="highlight
cl">127.0.0.1:6379> llen k1
(integer) 3

</code></pre>
<code>lpop/rpop <key></code>：从左边/右边吐出一个值。值在键在，值光键亡。</
>
<pre><code class="highlight-chroma"><span class="highlight
cl">127.0.0.1:6379> lpop k1
"3"

</code></pre>
<code>rpoplpush <key1><key2></code>：从 key1 列表右边吐出一个值，插到
ey2 列表左边。
<pre><code class="highlight-chroma"><span class="highlight
cl">127.0.0.1:6379> rpoplpush k1 k2
"1"
127.0.0.1:6379>
lrange k2 0 -1
1) "1"
2) "4"
3) "5"
4) "6"
</code></pre>
<code>linsert <key> before <value><newvalue></code>：在 value 的

面插入 newvalue 值
<pre><code class="highlight-chroma">127.0.0.1:6379> linsert k2 before 5 3
(integer) 5
127.0.0.1:6379> lrange k2 0 -1
1) "1"
2) "4"
3) "3"
4) "5"
5) "6"

</code></pre>
<code>lrem <key><n><value></code>：从左边删除 n 个 value(从左到右
<code>lset<key><index><value></code>：将列表 key 下标为 index 的替换成 value
<pre><code class="highlight-chroma">127.0.0.1:6379> lset k1 0 2
OK
127.0.0.1:6379> lrange k1 0 -1
1) "2"
</code></pre>
<h4 id="3-3-数据结构">3.3 数据结构</h4>
List 的数据结构为快速链表 quickList。
首先在列表元素较少的情况下会使用一块连续的内存存储，这个结构是 ziplist，也即是压缩列表。</>
它将所有的元素紧挨着一起存储，分配的是一块连续的内存。
当数据量比较多的时候才会改成 quicklist。
因为普通的链表需要的附加指针空间太大，会比较浪费空间。比如这个列表里存的只是 int 类型的数，结构上还需要两个额外的指针 prev 和 next。
<img src="https://ld246.com/images/img-loading.svg" alt="image.png" data-src="https://b3logfile.com/file/2022/05/image-0a71c2f3.png?imageView2/2/interlace/1/format/jpg"></>
Redis 将链表和 ziplist 结合起来组成了 quicklist。也就是将多个 ziplist 使用双向指针串起来使。这样既满足了快速的插入删除性能，又不会出现太大的空间冗余。
<h3 id="4-Redis集合-Set-">4.Redis 集合（Set）</h3>
<h4 id="4-1简介">4.1 简介</h4>
set 对外提供的功能是类似于 list 的列表，但是 set 可以自动排重，并提供了判断元素是否存在接口
Redis 的 Set 是 string 类型的无序集合。它底层其实是一个 value 为 null 的 hash 表，所以添加，除，查找的复杂度都是 O(1)。
<h4 id="4-2-常用命令">4.2 常用命令</h4>
<code>sadd key value1 value2 ...</code>：将一个或多个 member 元素加入到集合 key 中已经存在的元素将被覆盖
<pre><code class="highlight-chroma">127.0.0.1:6379> sadd k1 1 2 3 1
(integer) 3
`
</code></pre>
<code>smembers key</code>： 取出该集合的所有元素
<pre><code class="highlight-chroma">127.0.0.1:6379> smembers k1

1) "1"
2) "2"
3) "3"
</code></pre>
<code>sismember key value</code>: 判断集合 key 中是否有 value
<pre><code class="highlight-chroma">127.0.0.1:6379> sismember k1 1
(integer) 1
127.0.0.1:6379> sismember k1 5
(integer) 0

</code></pre>
<code>scard key</code>：返回该集合的元素个数
<pre><code class="highlight-chroma">127.0.0.1:6379> scard k1
(integer) 3

</code></pre>
<code>srem key value1 value2 ...</code>：删除集合中的某个元素
<code>spop key</code> 随机从该集合中吐出某个元素
<code>srandmember key n</code>：随机从该集合中取出 n 个值，不会从集合中删除
<code>smove source destination value</code>：从集合中把一个值移动到另一个集合中</
>
<code>sinter key1 key2</code>：返回两个集合的交集
<code>sunion key1 key2</code>：返回两个集合的并集
<code>sdiff key1 key2</code>：返回两个集合的差集
<h4 id="4-3数据结构">4.3 数据结构</h4>
Set 数据结构是 dict 字典，字典是用哈希表实现的。
Java 中 HashSet 的内部实现使用的是 HashMap，只不过所有的 value 都指向同一个对象。Redis 的 set 结构也是一样，它的内部也使用 hash 结构，所有的 value 都指向同一个内部值。
<h3 id="5-Redis哈希-hash-">5 Redis 哈希(hash)</h3>
<h4 id="5-1-简介">5.1 简介</h4>
Redis hash 是一个键值对集合。
Redis hash 是一个 string 类型的 field 和 value 的映射表，hash 特别适合用于存储对象。
类似 Java 里面的 Map<String,Object>
用户 ID 为查找的 key，存储的 value 用户对象包含姓名，年龄，生日等信息，如果用普通的 key/value 结构来存储
主要有以下两种存储方式：
<img src="https://ld246.com/images/img-loading.svg" alt="image.png" data-src="https://b3logfile.com/file/2022/05/image-0ba4828f.png?imageView2/2/interlace/1/format/jpg"></
>
<h4 id="5-2-常用命令">5.2 常用命令</h4>
<code>hset key field value</code> ：给 key 集合中的 field 键赋值 value
<code>hmset key </code>：field1 value1 field2 value2 ... 批量设置 hash 的值
<pre><code class="highlight-chroma">127.0.0.1:6379> hset k1 name tom gender female
(integer) 2
127.0.0.1:6379> hset k2 name jerry gender female
(integer) 2

</code></pre>
<code>hget key field</code> ：从 key 集合 field 取出 value

<pre><code class="highlight-chroma">127.0.0.1:6379> hget k1 name
"tom"
</code></pre>
<code>hexists key</code>： field 查看哈希表 key 中，给定域 field 是否存在
<pre><code class="highlight-chroma">
127.0.0.1:6379> hexists k1 name
(integer) 1
127.0.0.1:6379> hexists k1 age
(integer) 0

</code></pre>
<code>hkeys key </code>：列出该 hash 集合的所有 field
<pre><code class="highlight-chroma">
127.0.0.1:6379> hkeys k1
1) "name"
2) "gender"

</code></pre>
<code>hvals key</code> ：列出该 hash 集合的所有 value
<pre><code class="highlight-chroma">
127.0.0.1:6379> hvals k1
1) "tom"
2) "female"

</code></pre>
<code>hincrby key field increment</code> ：为 key 中的域 field 的值加上增量 1 -1
<code>hsetnx key field value </code>：将哈希表 key 中的域 field 的值设置为 value，当仅当域 field 不存在
<pre><code class="highlight-chroma">
127.0.0.1:6379> hsetnx tom gender male
(integer) 1
127.0.0.1:6379> hget tom name
(nil)
127.0.0.1:6379> hsetnx tom age 10
(integer) 1
127.0.0.1:6379> hget tom age
"10"

</code></pre>
<h4 id="5-3-数据结构">5.3 数据结构</h4>
Hash 类型对应的数据结构是两种：ziplist（压缩列表），hashtable（哈希表）。当 field-value 长度较短且个数较少时，使用 ziplist，否则使用 hashtable
<h3 id="6-Redis有序集合Zset">6.Redis 有序集合 Zset</h3>
<h4 id="6-1简介">6.1 简介</h4>
Redis 有序集合 zset 与普通集合 set 非常相似，是一个没有重复元素的字符串集合。
不同之处是有序集合的每个成员都关联了一个评分（score）,这个评分（score）被用来按照从最低分最高分的方式排序集合中的成员。集合的成员是唯一的，但是评分可以是重复了 。
因为元素是有序的, 所以你也可以很快的根据评分（score）或者次序（position）来获取一个范围的

素。
访问有序集合的中间元素也是非常快的,因此你能够使用有序集合作为一个没有重复成员的智能列表。
/p>
<h4 id="6-2-常用命令">6.2 常用命令</h4>
zadd key score1 value1 score2 value2... ： 将一个或多个 member 元素及其 score 值加入有集合 key 中
<pre><code class="highlight-chroma">127.0.0.1:6379> zadd k1 100 java 200 c++ 300 mysql 400 web
(integer) 4

</code></pre>
zrange key start stop 返回有序集合 key 中下表在 start 与 stop 之间的元素、
带 WITHSCORES 可以显示 scores
<pre><code class="highlight-chroma">127.0.0.1:6379> zrange k1 1 3
1) "c++"
2) "mysql"
3) "web"
127.0.0.1:6379> zrange k1 1 3 withscores
1) "c++"
2) "200"
3) "mysql"
4) "300"
5) "web"
6) "400"

</code></pre>
zrangebyscore key minmax [withscores][limit offset count
返回集合 key 中所有 score 介于 min 和 max 之家的 member，按 score 值递增排列’
<pre><code class="highlight-chroma">127.0.0.1:6379> zrangebyscore k1 200 300
1) "c++"
2) "mysql"
</code></pre>
zrevrangebyscore key maxmin [withscores][limit offset count] 递减排列
<pre><code class="highlight-chroma">127.0.0.1:6379> zrevrangebyscore k1 300 100
1) "mysql"
2) "c++"
3) "java"

</code></pre>
<h4 id="6-3-数据结构">6.3 数据结构</h4>
SortedSet(zset)是 Redis 提供的一个非常特别的数据结构，
一方面它等价于 Java 的数据结构 Map<String, Double>，可以给每一个元素 value 赋予个权重 score，
另一方面它又类似于 TreeSet，内部的元素会按照权重 score 进行排序，可以得到每个元素的名，还可以通过 score 的范围来获取元素的列表。
zset 底层使用了两个数据结构
（1）hash，hash 的作用就是关联元素 value 和权重 score，保障元素 value 的唯一性，可以通过素 value 找到 相应的 score 值。
（2）跳跃表，跳跃表的目的在于给元素 value 排序，根据 score 的范围获取元素列表。