Redis的数据结构与内部编码

# Redis的5种数据结构

Redis中存储的都是**键值对key-value**，其中所有的**键key**都是字符串string类型；而通常说Redis有5种数据结构，它们都是对应键值对中的**值**来说的，这5种数据结构分别为**string、list、hash、set、zset**。(字符串、列表、哈希、集合、有序集合)

内部编码：每种数据结构都有对应的超过2种的内部编码。

# Redis的键管理命令

## Redis键管理命令分类

Redis的键管理命令有上百个，虽然多，但是很有规律：可以分为**全局命令**和**数据结构特有的命令**。

* **全局命令**：由于Redis中存储的都是键值对，且键key都是字符串类型，这些键值对都是通过key操作的，都可以设置过期时间等，因此存在一些命令对任何数据类型都通用，称为**全局命令**。
* **数据结构特有的命令：**针对不同的数据结构存在不同的增加、修改、获取等命令，但是很有规律可循**。**

## 全局命令

既然**Redis**中存储的都是**键值对**，那么必然存在一些通用的命令，对任何数据结构都适用。按照**单个键、遍历键、数据库管理**三个方面介绍这些通用命令：

## 数据库管理命令(3个)：select num、dbsize、flushdb/flushall

### 切换数据库：**select dbIndex**

配置文件databases num 配置了数据库的个数，默认是16个。

**注意**：Redis的分布式实现Redis Cluster只允许使用0号数据库。

若需要使用多个数据功能，完全可以在一台机器上部署多个Redis实例，彼此之间利用端口号来做区分。

1. **当前数据库的键总数：dbsize**

**注意**:keys的复杂度为O(n),而dbsize的复杂度为O(1)。keys命令会遍历所有的键，而dbsize获取的是Reids内置的键总数变量。

### 清空数据库：**flushdb/flushall**

flushdb只清除当前数据库；flushall会清除所有数据库。

注意：**flushdb/flushall命令会将所有**数据清除，一旦误操作，后果不堪设想。一般在配置文件中，通过rename-command配置规避这个命令。另外，若数据较多，存在阻塞Redis的可能性。

## 单个键管理命令(多)

1. **type key ②exists key、③del key…、④rename key newkey⑤randomkey ⑥expire/ pexpire key seconds/milliseconds、expireat/ pexpireat key timestamp/milliseconds-timestamp、⑥persist key** ⑦**ttl/pttl key**

**注意：del 后可以跟多个key，支持批量删除。**

### **获取键的数据结构：type key**

返回字符串，分别为string、list、hash、set、zset

1. **检查键是否存在: exists key**

返回1表示存在，0表示不存在。

1. **删除键: del key [key …]** 一次可以删除多个键；返回成功删除键的个数，若返回0，则表示没有删除任何一个键，当然删除一个本不存在的键就返回0.

Redis中存储的都是键值对，无论什么数据类型，都可以通过del key命令删除掉。

### **键重命名**：**rename/renamenx** key newkey

**存在问题**：如果newkey已经存在，那么原来newkey的值将被覆盖(原newkey键值对被删除)。

解决办法：为了防止强行rename，Redis提供了**renamenx**命令，确保**只有newkey不存在**时候才被覆盖（类似**setnx**）。 **renamenx** key newkey

注意：**rename key key** (名字一样)在3.2版本返回Ok，之前版本报错。

### 随机返回一个键：**randomkey**

1. 键过期：

**expire** key seconds :对键添加过期时间；键在seconds秒后过期

**pexpire key milliseconds:**键在milliseconds毫秒后过期；

**expire key timestamp:**键在秒级时间戳timestamp后过期；

**pexpireat key milliseconds-timestamp：**键在毫秒级时间戳timestamp后过期；

无论是使用过期时间还是时间戳，秒级还是毫秒级，在Redis内部最终使用的都是**pexpireat**。

setex是**set+expire**的组合命令，是原子执行的，且减少一次网络通讯时间。

若expire key seconds返回0，表示键不存在；返回负值，键立即被删除。

Redis不支持二级数据结构(如哈希、列表)内部元素的过期功能。

1. 清除键的过期时间：**persist key**

**注意：string类型的set命令会自动去掉过期时间，源码中：最后执行了removeExpire(db,key)操作。P63**

1. 获取键的剩余过期时间：**ttl** key(秒)、pttl key(毫秒级)

有3种返回值：

**大于等于0**的整数表示键剩余的过期时间；**-1**表示没有设置过期时间；**-2**表示键不存在。

### 迁移键：move、dump+restore、migrate

move key db:用于Redis内部，数据库之间的迁移；

**dump key**

**restore key ttl value**

## 遍历键命令

### **查看相匹配的键：keys pattern**

若查看所有的键：keys \*

## 数据结构特有命令

针对每种数据结构都有自己特有的操作命令，在以下各个部分单独介绍。

# string数据结构及操作命令

**string最简单的类型，键值对中，一个key对应一个value，string类型是二进制安全的。Redis中的string可以包含任何数据，比如jpg图片、序列化的对象等等。**

（1）设置key对应的值为**string类型的value：**

添加一个 key1=zhaohong的键值对：利用set。

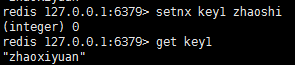


修改此key1对应的value：（**一个键只能对应一个value**）



（2） **setnx： nx 表示not exist，不存在。即只有key不存在时，才会成功。**

**如果key已经存在，则返回0，且修改不成功，如果不存在，设置成功，返回1。**



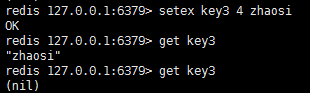


（3）如果没有这个键，直接get，则会返回nil：



（4） **setex ： 设置键值对的生存时间。**

**使用方法 setex key time value**

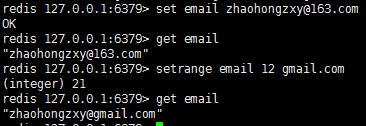


**3秒后，就消失了。**

（5） **setrange** ： 设置指定key的value 的子字符串，相当于修改value值。

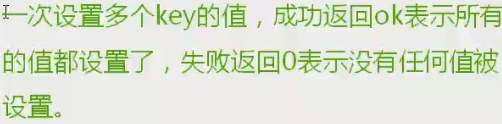
**setrange key index subValue #index从0开始**

**返回的是修改后value的字符个数。**

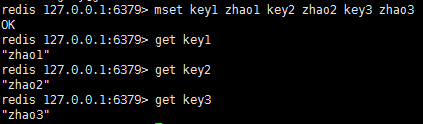


**注意**：如果替代的字符串较短，不能把原来的字符串覆盖掉。如：

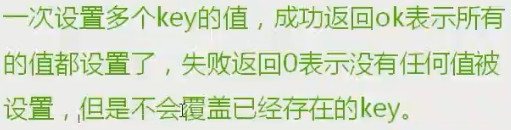
**（6）mset： 一次设置多个值。**

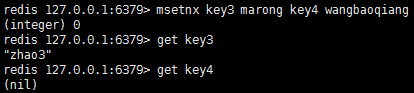


**mset key1 value1 key2 value2 key3 value3 ….**



**msetnx : 返回0，则都不成功。**



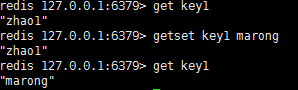


1. （1）**get**： 获取key对应的value值，如果key不存在，则返回nil。

get key1

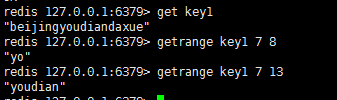
（2）**getset** ： 设置key的值，并返回key的旧值。获取旧值，并设置新值。

**getset key newvalue**



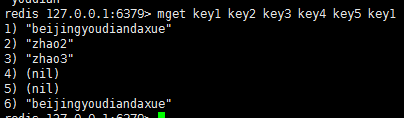
（3）getrange: 获取key对应的value值得子字符串。

**getrange key fromindex endindex**



（4） **mget** ：一次获取多个key的值，如果对应的key不存在，则返回nil。

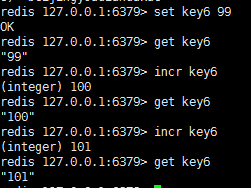
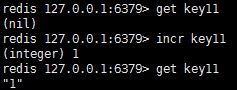
mget key1 key2 key3 ….



1. **incr** : 对**key的值**做加加操作，并返回新的值。

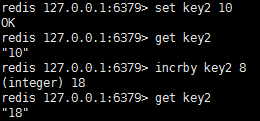
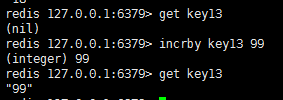
incr key

如果key本身不存在，则默认为原来的值 为0，并子增1.

**incrby** ： 同incr类似，加指定值num，key不存在时，会设置key，并默认为原来的值为0，在0基础上加num。

incry key num #num可以为负值。

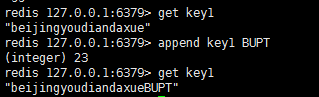
1. 类似的，有decr和decrby。

decr key： 对key对应的值做减操作。

decrby key num：减去设定的值num。

1. append ：给key对应的字符串添加子字符串，返回新字符串的长度。

append key subvalue

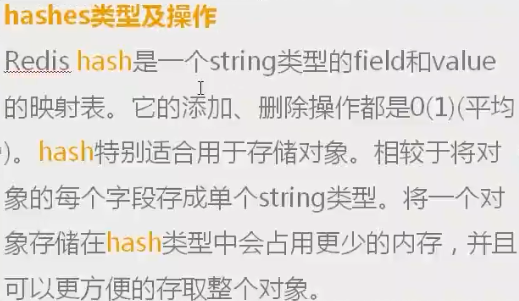
****

1. **strlen ： 查看key对应的value的长度。 strlen key**



# hash数据结构及操作命令

1. 第二种数据类型：hash：



# list数据结构及操作命令

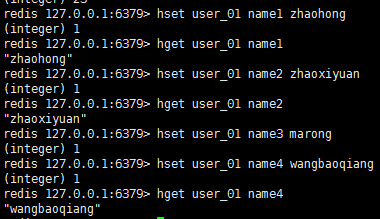
# set数据结构及操作命令

# zset数据结构及操作命令

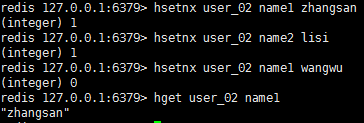
## 关于set的方法：

（1）hset : 设置hash field 为指定值，如果key不存在，则先创建。设置成功则返回1，否则返回0。

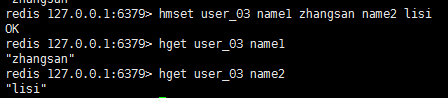
hset hashtablename field1 value



（2）**hsetnx ： 设置成功返回1，否则返回0.**



（3）hmset ： 批处理，同时设置多个。



注意没有 hmsetnx这个命令。

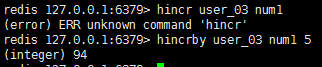
1. 关于get的方法：
2. hget ： 获取key对应的值。



1. hmget ： 获取多个key对应的值。

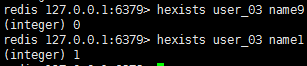


1. hincrby ：增量。对hash上某个field对应的value增加一定值。返回增量后的value值。



没有hincr方法。

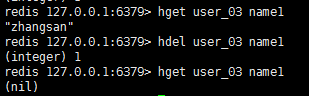
1. hexists： 判断hash中某个field是否存在。存在则返回1，否则返回0.



1. hlen ： 获取hash中几个field，返回个数。

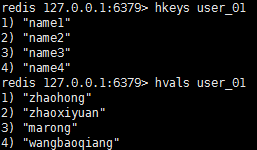


1. hdel ： 删除hash中指定的field。删除成功返回1，否则返回0.

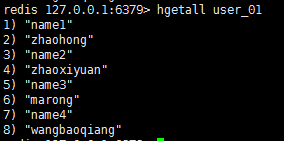


1. hkeys：返回hash中所有的key。

hvals：返回hash中所有的value。



1. hgetall：获取某个hash中全部的field和value。



1. 注意：hash中没有 hmsetnx 、hsetex、hsetrange、hincr等方法。