链滴

Redis 各个数据类型理解

作者：yhm

原文链接：https://ld246.com/article/1642580741973

来源网站：链滴

1、redis分5大数据类型

String （字符串类型，SDS结构的，可变长度的字符串）

list （列表）

set （无序，没有重复的集合）

zset（有序）

hash （哈希数据结构）

2、公共redis结构体

redis针对每种数据类型都由RedisObject对象

```
typedef struct redisObject{
    //类型
    unsigned type:4;
    //编码
    unsigned encoding:4;
    //指向底层数据结构的指针
    void *ptr;
    //引用计数
    int refcount;
    //记录最后一次被程序访问的时间
    unsigned lru:22;

}robj
```

redis中每个基本数据类型都由此公共结构体描述

type的取值范围：

对象	对象 type 属性的值	*TYPE* 命令的输出
字符串对象	REDIS_STRING	"string"
列表对象	REDIS_LIST	"list"
哈希对象	REDIS_HASH	"hash"
集合对象	REDIS_SET	"set"
有序集合对象	REDIS_ZSET	"zset"

redis命令行操作，显示key的Type类型

```
type key
```

```
127.0.0.1:6379> set str1 v1
OK
127.0.0.1:6379> lpush list1 v1 v2 v3
(integer) 3
127.0.0.1:6379> type str1
string
127.0.0.1:6379> type list1
list
127.0.0.1:6379>
```

注意：在Redis中，键总是一个字符串对象，而值可以是字符串、列表、集合等对象，所以我们通常的键为字符串键，表示的是这个键对应的值为字符串对象，我们说一个键为集合键时，表示的是这个对应的值为集合对象。

2.1、encoding 属性和 *prt 指针

prt指针类似于C语言中的指针，指向Redis底层真实的数据结构的头地址

而具体是什么数据类型的数据结构，则由encoding决定

编码常量	编码所对应的底层数据结构
REDIS_ENCODING_INT	long 类型的整数
REDIS_ENCODING_EMBSTR	embstr 编码的简单动态字符串
REDIS_ENCODING_RAW	简单动态字符串
REDIS_ENCODING_HT	字典
REDIS_ENCODING_LINKEDLIST	双端链表
REDIS_ENCODING_ZIPLIST	压缩列表
REDIS_ENCODING_INTSET	整数集合
REDIS_ENCODING_SKIPLIST	跳跃表和字典

每种类型的数据结构都至少使用了2种编码，编码方式清单如下：

类 型	编 码	对 象
REDIS_STRING	REDIS_ENCODING_INT	使用整数值实现的字符串对象
REDIS_STRING	REDIS_ENCODING_EMBSTR	使用 embstr 编码的简单动态字符串实现的字符串对象
REDIS_STRING	REDIS_ENCODING_RAW	使用简单动态字符串实现的字符串对象
REDIS_LIST	REDIS_ENCODING_ZIPLIST	使用压缩列表实现的列表对象
REDIS_LIST	REDIS_ENCODING_LINKEDLIST	使用双端链表实现的列表对象
REDIS_HASH	REDIS_ENCODING_ZIPLIST	使用压缩列表实现的哈希对象
REDIS_HASH	REDIS_ENCODING_HT	使用字典实现的哈希对象
REDIS_SET	REDIS_ENCODING_INTSET	使用整数集合实现的集合对象
REDIS_SET	REDIS_ENCODING_HT	使用字典实现的集合对象
REDIS_ZSET	REDIS_ENCODING_ZIPLIST	使用压缩列表实现的有序集合对象
REDIS_ZSET	REDIS_ENCODING_SKIPLIST	使用跳跃表和字典实现的有序集合对象

原文链接：Redis 各个数据类型理解

查看Value使用的编码，命令如下

OBJECT ENCODING key

```
127.0.0.1:6379> set k1 str
OK
127.0.0.1:6379> set k2 123
OK
127.0.0.1:6379> OBJECT ENCODING k1
"embstr"
127.0.0.1:6379> OBJECT ENCODING k2
"int"
127.0.0.1:6379>
```

比如String类型的数据结构，可以由int、embstr或者Raw编码方式实现

3、String类型

字符串类型是很常用的类型 。所有key-value对的Key是string类型，字符串类型长度不超过512MB。

字符串类型的三种编码格式

- int：使用字符串类型存储数字类型，比如put intkey 8876
- raw：保存长字符串，长度大于44字节的字符串（redis3.2版本之前是39字节，之后是44字节）
- embstr：保存不超过44字节的字符串（redis3.2版本之前是39字节，之后是44字节）

int 编码是用来保存整数值，而raw和embstr用来存储字符串（长度大小不同使用不同编码方式）。

raw和embstr的存储区别：

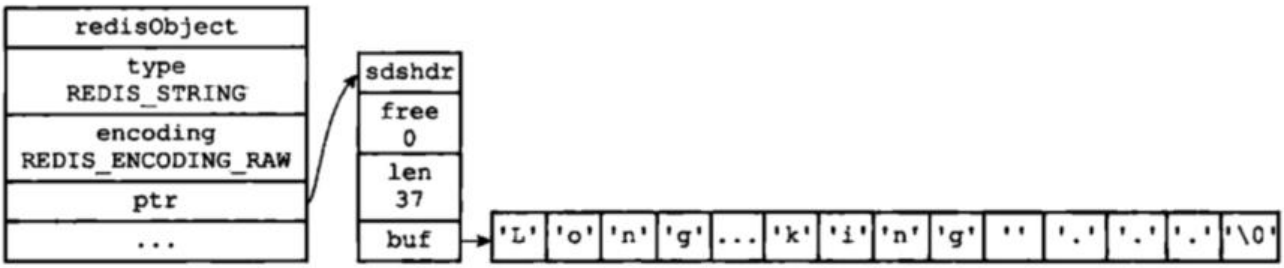


图 8-2　raw 编码的字符串对象

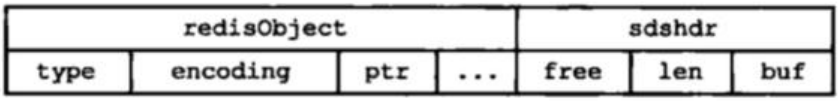


图 8-3　embstr 编码创建的内存块结构

embstr编码方式，存放的RedisObject头信息和一定长度的后缀（Sds实现）是一次性申请内存空间。因此raw查找快（因为直接可以在Redisobject后通过偏移量获取），但是eMbstr的字符串如果需变长了，则需要重新申请并分配RedisObject头节点空间和sds空间，因此，Embstr设置为只读的。

而raw不一样，他的 RedisObject头信息节点和后面的sds实现的字符串是分2次申请内存空间的，申

内存空间代价相比embstr高，而且后面节点也需要头信息，因此存储效率略微低些。但是存储字符串度，是最大512MB

* 注意

如果Redis的字符串采用的Int编码存储的数据的大小超过Long的最大值，则Redis的字符串类型编码动变成Raw。

对于 embstr 编码，由于 Redis 没有对其编写任何的修改程序（embstr 是只读的），在对embstr对进行修改时，都会先转化为raw再进行修改，因此，只要是修改embstr对象，修改后的对象一定是ra的，无论是否达到了44个字节。

4、List数据类型

list存储简单的字符串对象，其实是一个队列，可以在队列头和队列尾分别插入数据。底层是 用链表现的队列。

4.1、List编码格式

4.1.1、Ziplist编码格式

即同时满足如下条件时，使用zipList编码方式存储List：

- 队列元素个数小于512个
- 每个队列元素数据大小，小于64字节
-

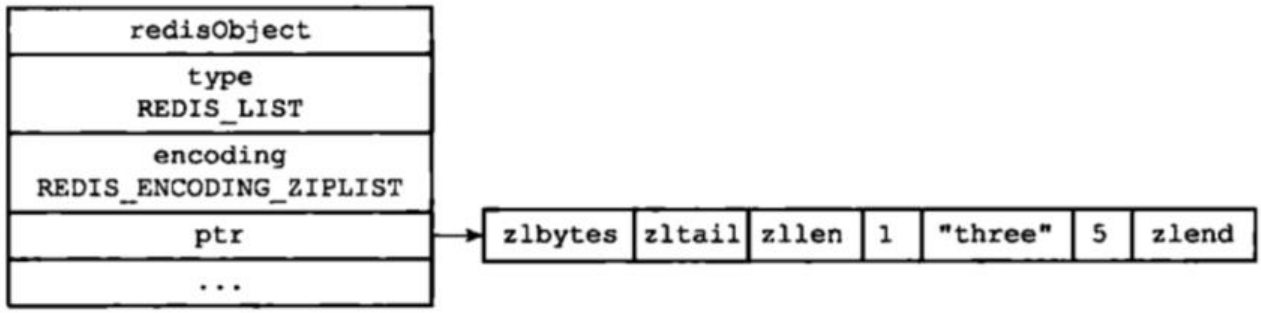


图 8-5　ziplist 编码的 numbers 列表对象

4.1.2、linkedlist编码格式

不同时都满足上述2个条件数据，即采用Linkedlist存储，结构如下：

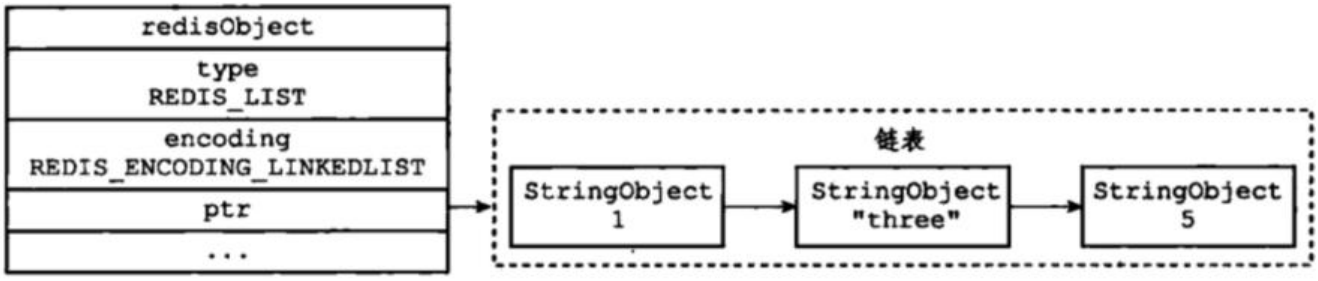


图 8-6　linkedlist 编码的 numbers 列表对象

原文链接：Redis 各个数据类型理解

上面两个条件可以在redis.conf 配置文件中的 list-max-ziplist-value选项和 list-max-ziplist-entries选项进行配置。

5、哈希数据结构

hashs数据结构，key是字符串，Value可能是一个键值对集合

底层编码方式：

5.1、Ziplist编码格式

为了节省空间，Ziplist把加入进去的键值对，每次都往列表的尾部插入

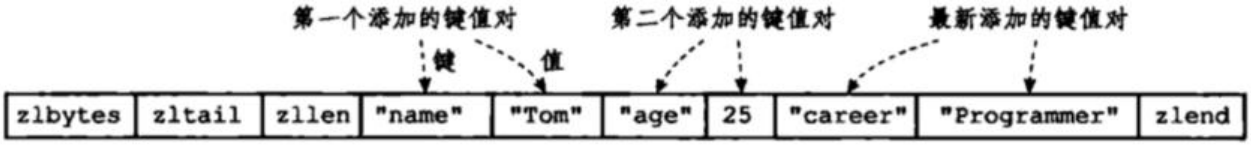


图 8-10　profile 哈希对象的压缩列表底层实现

5.2、Hashtable编码格式

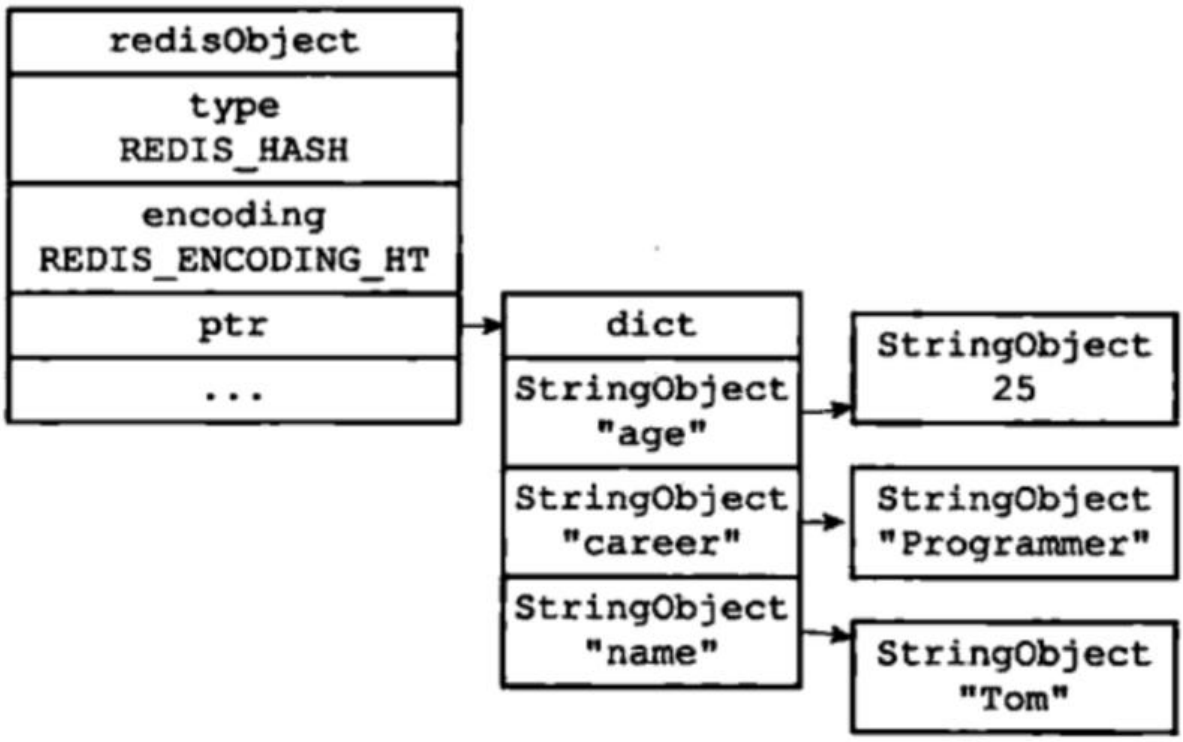


图 8-11　hashtable 编码的 profile 哈希对象

底层采用数据字典方式，通过hash算法，计算Key在hash字典表中的位置，然后通过引用，找到Valu对象：

在前面介绍压缩列表时，我们介绍过压缩列表是Redis为了节省内存而开发的，是由一系列特殊编码连续内存块组成的顺序型数据结构，相对于字典数据结构，压缩列表用于元素个数少、元素长度小的景。其优势在于集中存储，节省空间。

5.3、hash数据结构 的 底层编码转换 规则

同时满足下面2个条件

- 列表中元素个数小于512
- 列表中每个元素占用空间大小，都小于64字节

Redis自动将Ziplist转为Hashtable存储

6、集合数据结构

7、有序集合数据结构

8、Redis各个数据结构的应用场景

string 存储字符串，数字类型自加，计数等。基于Sds机制实现，是二进制安全的（二进制安全参考https://www.zhihu.com/question/28705562），可以存放视频，音频，图片等

list 当作简单消息队列使用；使用lrange实现分页查询

hash 用来存储用户登陆session数据，做单点登陆功能的后端实现

set 无序，不重复集合，用来去重

Zset 有序集合，用来排序，排行榜，使用TOP N 命令操作

9、Redis的内存回收和共享

Redis内存回收策略：

1）volatile-lru　利用LRU算法移除设置过过期时间的key (LRU:最近使用 Least Recently Used)

2）allkeys-lru　利用LRU算法移除任何key

3）volatile-random 移除设置过过期时间的随机key

4）allkeys-random　移除随机key

5）volatile-ttl　移除即将过期的key(minor TTL)

6）noeviction　noeviction　不移除任何key，只是返回一个写错误 ，默认为此选项

10、对象的空转时长

见redisObject对象：

```
typedef struct redisObject{
    //类型
    unsigned type:4;
    //编码
    unsigned encoding:4;
    //指向底层数据结构的指针
    void *ptr;
    //引用计数
```

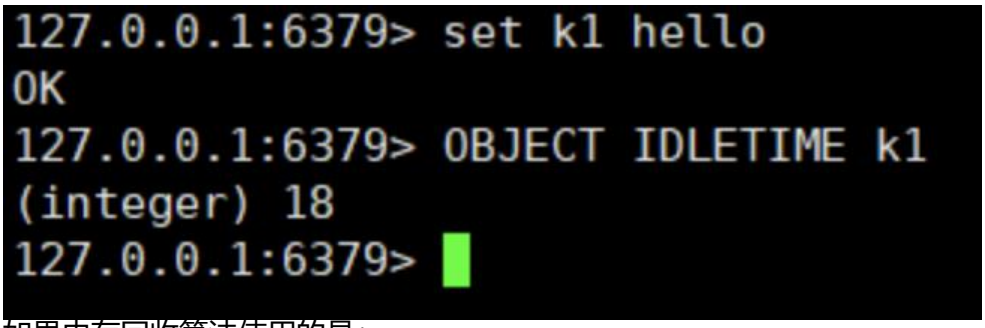
```
    int refcount;
    //记录最后一次被程序访问的时间
    unsigned lru:22;

}robj
```

最后一个属性：lru

该属性记录了对象最后一次被命令程序访问的时间。

使用命令OBJECT IDLETIME key 查询Key的lru



如果内存回收算法使用的是：

1）volatile-lru　利用LRU算法移除设置过过期时间的key (LRU:最近使用 Least Recently Used)

2）allkeys-lru　利用LRU算法移除任何key

则有使用到lru属性支撑算法。