redis设计简介

多数据库

redis是字典结构的存储服务器，一个redis实例提供多个用来存储数据的字典，客户端可以指定将数据存储在哪个字典中。

redis不支持自定数据库名字，每个数据库都以编号命名。另外，一个redis实例不支持为每个数据库设置不同的访问密码，所以一个客户端要么可以访问全部数据库，要么链一个数据库页没有权限访问。最重要的多个数据库之间不是完全隔离的，比如flushall可以清空一个redis实例中所有数据库中的数据。

简单动态字符串SDS

```
struct sdshdr {
    // 记录buf数组中已经使用字节的数量
    //等于SDS所保存字符串长度
    int len;
    // 记录buf数组中未使用字节数量
    int free;
    //字节数组，用于保存字符串
    char buf[]
}
```

1. 使用len获取字符串长度
2. 空间预分配足够，减少修改字符串时带来的内存重分配次数
3. 惰性空间释放，字符串缩短操作不直接设计内存空间的释放。

链表

redis中的列表键是用链表实现的。当一个列表键中包含了数量比较多的元素，又或者列表中包含的元素都是比较长的字符串时，redis就会使用链表作为列表键的底层实现。

字典

使用哈希表实现，一个哈希表里面有多个哈希表结点，而每个哈希表结点就保存了字典中的一个键值对。

1. 使用哈希链表解决冲突
2. rehash方式采用渐进式hash

   当哈希表中键值对数量太多后者太少时，程序需要对哈希表的大小进行相应的扩展或者收缩。

   哈希表扩展收缩时间：

   1. 服务器目前没有实行BGSAVE或者BGREWRITEAOF命令，并且哈希表的负载因子大于等于1.
   2. 服务器目前证字啊执行BGSAVE或者BGREWRITE命令，并且哈希表的负载因子大于等于5
   3. 负载因子小于0.1时哈希收缩。

跳跃表

跳跃表是一种有序的数据结构，它通过在每个结点中维持多个指向其他结点的指针，从而达到快速访问结点的目的。

Redis使用跳跃表作为有序集合键的底层实现之一，如果一个有序集合包含的元素数量比较多，redis会使用跳跃表作为底层实现。

redis还在集群结点中把跳跃表作为内部数据结构。

整数集合

当一个几何中值包含整数值元素，并且这个集合的元素数量不多时，Redis就会使用整数集合作为集合键的底层实现。

```
typedef struct intset{
    //编码方式，确定数组存放的整数类型
    uint32_t encoding;
    //集合中包含的元素数量
    uint32_t length;
    //保存元素的数组
    int8_t contents[];
}intset
```

1. 升级。添加的新元素类型长度大于原来的整数集合元素，需要分配内存空间添加到原来的数组上，然后移动原来的元素至合适的位置。
2. 不存在降级操作。

压缩列表

压缩列表是列表键和哈希键的底层实现之一。当一个列表见只包含少量列表项，并且每个列表键要么就是小整数，要么就是长度比较短的字符串，那么Redis就会使用压缩列表作为底层实现。

压缩列表是为了节约内存开发的，是由一系列特殊编码的连续内存块组成的顺序型数据结构。一个压缩列表可以包含任意多个结点，每个结点可以保存一个字节数组或者一个整数值。

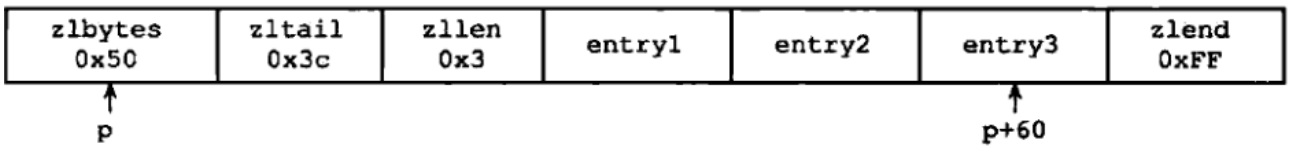

图 7-2　包含三个节点的压缩列表

结点数据结构如下：