# 数据结构与对象

## 简单动态字符串SDS

### 1.1.1 SDS的定义

Struct sdshdr {

int len;

int free;

char buf[];

};

1.1.2 SDS与c字符串的区别

* 常数复杂度获取字符串长度
* 杜绝缓冲区溢出
* 减少修改字符串时带来的内存重分配次数
  + 空间预分配
  + 惰性空间释放
* 二进制安全
* 兼容部分C字符串函数

## 链表

由于C语言没有实现链表结构，Redis实现了自己的链表。

Typedef struct listNode {

struct listNode \*prev;

struct listNode \*next;

void \* value;

} listNode;

## 1.3 字典

Redis的字典使用hash表作为底层实现，一个hash表里面可以有多个hash表节点，而每个hash表节点就保存了字典中的一个键值对。

### 1.3.1 Redis里面使用的哈希算法

|  |
| --- |
| hash = dict->type->hashFunction(key);  index = hash & dict->ht[x].sizemask; |

采用的hash算法是murmurhash2

### 1.3.2 与hash相关

#### 1.3.2.1 解决键冲突

链地址法。

#### 1.3.2.2 rehash

在对hash表进行扩展或者收缩操作时，程序需要将现有hash表包含的所有键值对rehash到新hash表里面，并且这个rehash过程并不是一次性完成的，而是渐进式地完成的。

## 跳跃表

跳跃表是一种有序数据结构，它通过每个节点中维持多个指向其他节点的指针，从而达到快速访问节点的目的。

## 1.5 整数集合

## 1.6 压缩列表

## 1.7 对象

# 二、单机数据库

## 2.1数据库

* Redis服务器的所有数据库都保存在redisServer.db数组中，而数据库的数量则由redisServer.dbnum属性保存。
* 客户端通过修改目标数据库指针，让它指向redisServer.db数组中的不同元素来切换不同的数据库。
* 数据库主要由dict和expires两个字典构成，其中dict字典负责保存键值对，而exoires字典则负责保存键的过期时间。
* 因为数据库由字典构成，所以对数据库的操作都是建立在字典操作之上。
* 数据库的键总是一个字符串对象，而值则可以是任意一种Redis对象类型，包括字符串对象、hash表对象、集合对象、列表对象和有序集合对象，分别对应字符串键、集合键、列表键和有序集合键。
* Expires字典的键指向数据库中的某个键，而值则记录了数据库键的过期时间，过期时间是一个以毫秒为单位的UNIX时间戳。
* Redis使用惰性删除和定期删除两种策略来删除过期键。
* 执行SAVE命令或者BGSAVE命令所产生的RDB文件不会包含已经过期的键。
* 执行BGREWRITEAOF命令所产生的重写AOF文件不会包含已经过期的键。
* 当一个过期键被删除之后，服务器会追加一条DEL命令到现有的AOF文件的末尾，显式的删除过期键。
* 当主服务器删除一个过期键之后，它会向所有从服务器发送一条DEL命令，显示的删除过期键。
* 从服务器即使发现过期键也不会自作主张的删除它，而是等待主节点发来DEL命令，这种统一、中心化的过期键删除策略可以保证主从服务器数据的一致性。
* 当Redis命令对数据库进行修改之后，服务器会根据配置向客户端发送数据库通知。

## 2.2 RDB持久化

* RDB文件用于保存和还原Redis服务器所有数据库中的所有键值对数据。
* SAVE命令由服务器进程直接执行保存操作，所以该命令会阻塞服务器。
* BGSAVE命令由子进程执行保存操作，所以该命令不会阻塞服务器。
* 服务器状态中会保存所有用save选项设置的保存条件，当任意一个保存条件被满足时，服务器会自动执行BGSAVE命令。
* RDB文件是一个经过压缩的二进制文件，由多个部分组成。
* 对于不同类型的键值对，RDB文件会使用不同的方式来保存它们。

## 2.3 AOF持久化

## 2.4事件

## 2.5客户端

## 2.6服务器