**String**

字符串类型是Redis最基础的数据结构

字符串类型的值实际可以是

字符串（简单字符串、复杂字符串（例如JSON、XML））

数字（整数、浮点数）

二进制（图片、音频、视频），但是值最大不能超过512MB。

**内部编码**

int：8个字节的长整型。

embstr：小于等于39个字节的字符串。

raw：大于39个字节的字符串。

**使用场景**

1.缓存功能

2.计数

incr key:value

3.共享Session

Spring Session+Redis 实现 Session共享

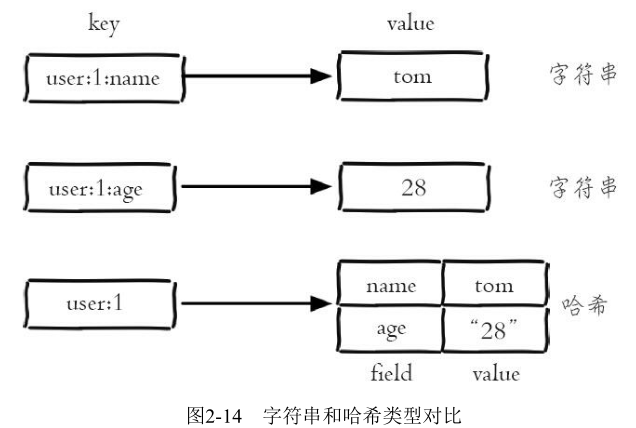
4.限速

5.全局唯一

incrby orderId 1000

## Hash

在Redis中，哈希类型是指键值本身又是一个键值对结构，形如value={{field1，value1}，...{fieldN，valueN}}



哈希类型中的映射关系叫作field-value，注意这里的value是指field对应的值，不是键对应的值，请注意value在不同上下文的作用。

**内部编码**

**ziplist（压缩列表）**

当哈希类型元素个数小于hash-max-ziplist-entries配置（默认512个）、同时所有值都小于hash-max-ziplist-value配置（默认64字节）时，Redis会使用ziplist作为哈希的内部实现，ziplist使用更加紧凑的结构实现多个元素的连续存储，所以在节省内存方面比hashtable更加优秀。

**hashtable（哈希表）**

当哈希类型无法满足ziplist的条件时，Redis会使用hashtable作为哈希的内部实现，因为此时ziplist的读写效率会下降，而hashtable的读写时间复杂度为O（1）。

**使用场景**

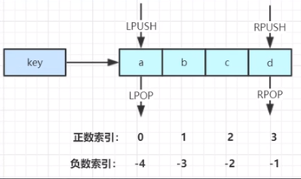
模拟关系型数据库，例如购物车场景

## List

列表（list）类型是用来存储多个有序的字符串，一个列表最多可以存储2^32 -1个元素.

在Redis中，可以对列表两端插入（push）和弹出（pop）,还可以获取指定范围的元素列

表、获取指定索引下标的元素.



可以充当栈和队列的角色

特点：

列表中的元素是**有序**，**可重复**的，可通过**索引下标**获取某个元素或者某个范围内的元素列表

**内部编码**

**ziplist（压缩列表）**：

当列表的元素个数小于list-max-ziplist-entries配置（默认512个），同时列表中每个元素的值都小于list-max-ziplist-value配置时（默认64字节），Redis会选用ziplist来作为列表的内部实现来减少内存的使用。

**linkedlist（链表）**：

当列表类型无法满足ziplist的条件时，Redis会使用linkedlist作为列表的内部实现.

**使用场景**

1.消息队列

2.文章列表

## Set

集合（set）类型保存多个字符串元素，元素**无序**，**不可重复**，不能通过索引下标获取。

一个集合（set）最多可以存储2^32 -1个元素。

**内部编码**

intset（整数集合）

当集合中的元素都是整数且元素个数小于set-max-intset-entries配置（默认512个）时，Redis会选用intset来作为集合的内部实现，从而减少内存的使用。

hashtable（哈希表）

当集合类型无法满足intset的条件时，Redis会使用hashtable作为集合的内部实现。

**使用场景**

集合类型比较典型的使用场景是标签（tag）

抽奖

## ZSet

有序集合：元素是有序，不可重复，元素设置一个分数(score)作为排序的依据。score可以重复

**内部编码**

ziplist（压缩列表）

当有序集合的元素个数小于zset-max-ziplist-entries配置（默认128个），同时每个元素的值都小于zset-max-ziplist-value配置（默认64字节）时，Redis会用ziplist来作为有序集合的内部实现，ziplist可以有效减少内存的使用。

skiplist（跳跃表）

当ziplist条件不满足时，有序集合会使用skiplist作为内部实现，因为此时ziplist的读写效

率会下降。

**使用场景**

有序集合比较典型的使用场景就是排行榜系统

缓存穿透，是指查询一个数据库一定不存在的数据。

解决办法：缓存空值

缓存雪崩，是指在某一个时间段，缓存集中过期失效。

解决办法：过期时间不固定

缓存击穿，是指一个key非常热点，在不停的扛着大并发，大并发集中对这一个点进行访问，当这个key在失效的瞬间，持续的大并发就穿破缓存，直接请求数据库，就像在一个屏障上凿开了一个洞

解决办法：

1.永不过期

2.使用互斥锁(mutex key)。对缓存查询加锁，如果KEY不存在，就加锁，然后查DB入缓存，然后解锁；其他进程如果发现有锁就等待，然后等解锁后返回数据或者进入DB查询