Redis的数据结构

以下提到的具体操作命令可以在本地的redis中进行验证测试

一、字符串String

String是redis的最基本的数据结构，String类型可以是简单的字符串，同时可以是json，xml，数字（整数，浮点），二进制（图片，音频，视频）等，但最大不能查过512兆。

常用的命令：

`set [key] [value]` 存储一对键值到redis中

`setnx [key] [value]` 在指定的 key 不存在时，为 key 设置指定的值。返回值，存储成功1，失败0

`setex [key] [timeout] [value]` 存储一对键值到redis，并设置过期时间（单位：秒）

`incr [key]` 存储一个计数的值到redis，从1开始计数，返回key当前存储的值

`getset [key] [value]` 设置key新的value值，同时返回key原来的value值

使用场景：

1. 对数据进行缓存，缓存通常能起到加速读写和降低 后端压力的作用。
2. 计数器，可以快速实现一个计数查询功能，比如点赞，视频播放量等。
3. 接口防刷，比如短信验证码可以缓存在redis中，5分钟内直接从缓存中取。
4. 限流，一个接口被恶意调用，通过计数功能，当一个接口被调用了10次以后，就做限流操作。

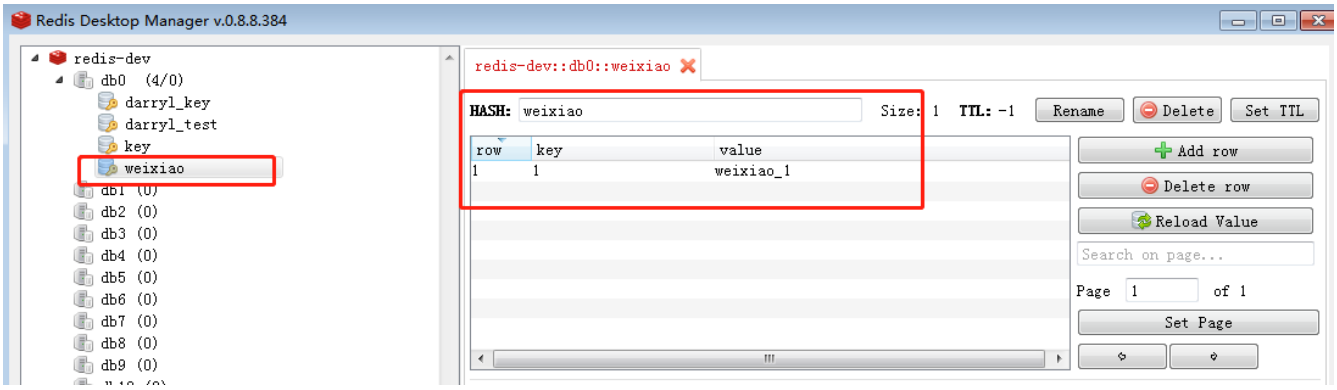
二、哈希hash

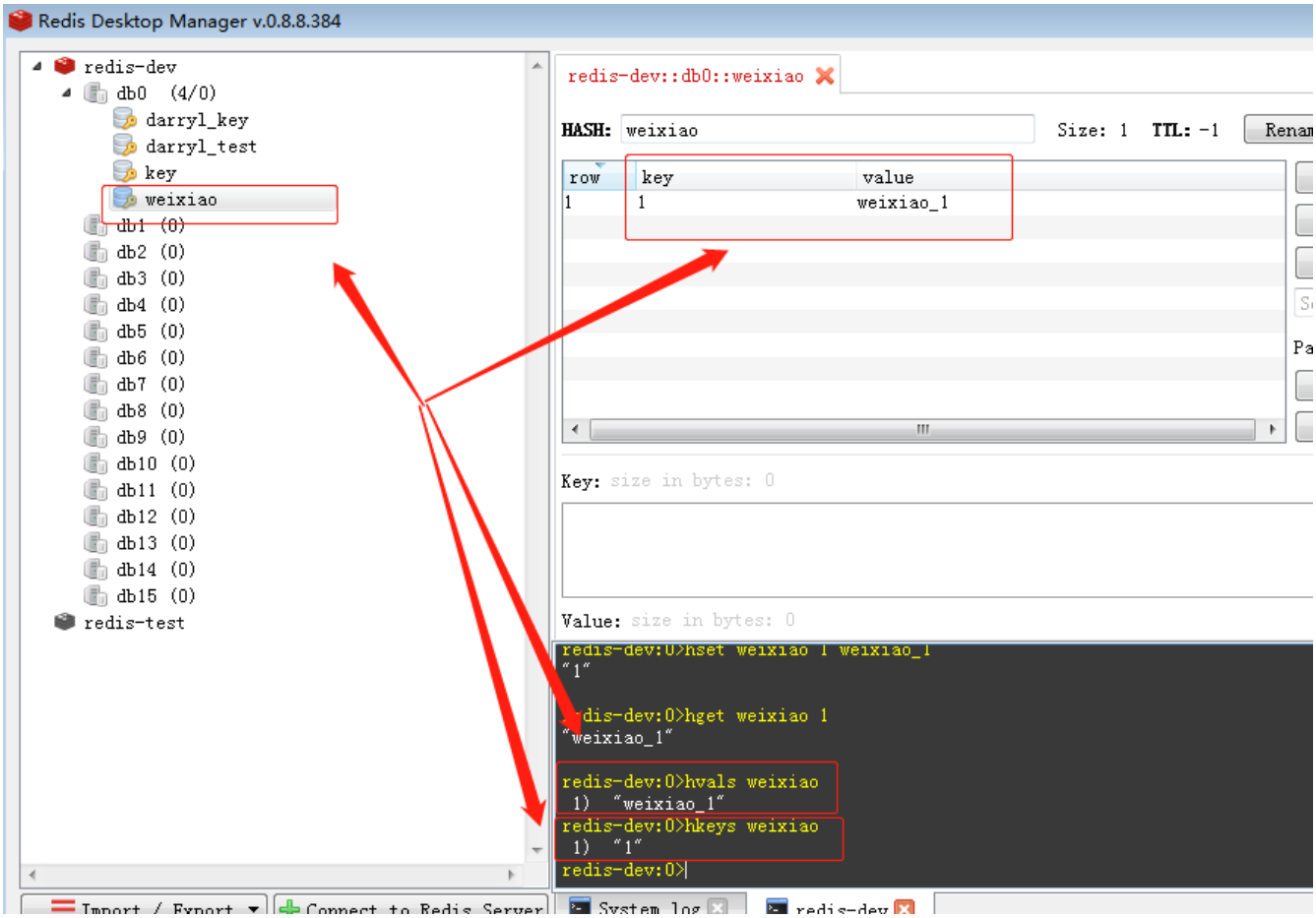
在redis中哈希类型是指值本身又是一种键值对结构，如 value={{field1,value1},.....{fieldN,valueN}}

`hset [key] [field] [value]` 存储值到redis

`hkeys [key]` 获取当前key下面存储的hash键值对中的所有key

`hvals [key]` 获取当前key下面存储的hash键值对中的所有value





i. ziplist: 当field个数不超过hash-max-ziplist-entries(默认为512个)时,并且没有大value(64个字节以上算大) ii. hashtable:ziplist的两个条件只要有一个不符合就会转换为hashtable

使用场景：

1. 存储用户信息，因为有结构关系，方便用于用户信息的存储
2. 数据存储简单直观，同时减少了内存空间

三、列表List

用来存储多个有序的字符串,列表中的每个字符串称为元素（element），一个列表最多可以存储2^32 -1个元素。

i. 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。 ii. 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。

`rpush [key] [value]` 从右边存储到这个key中的value值，返回list的长度。

`lpush [key] [value]` 从左边存储到这个key中的value值，返回list的长度。

`lpop [key]` 从左边弹出这个key中的value值，该value值将会从list中删除。

`lrem [key] [count] [value]` 删除指定元素,count>0从左到右，count<0从右到左,count=0,删除所有

`blpop [key] [timeout]` 阻塞式弹出，列表为空，则按照设置的timeout值进行阻塞，列表不为空，则会立即返回，这个阻塞会将整个redis进行阻塞。

使用场景：

1. 消息队列:Redis的lpush+brpop命令组合即可实现阻塞队列。

四、集合set

用来保存多个的字符串元素，但和列表类型不一样的是，集合中不允许有重复元素，并且集合中的元素是无序的，不能通过索引下标获取元素。

i. 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。 ii. 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。

`sadd [key] [value]` 将key-value向redis中存储到一个set集合中，返回这个set集合的大小

`scard [key]` 计数，返回这个key下面的存储的set集合的大小

`sismember [key] [value]` 判断set集合中的value值是否存在，返回0，不存在，返回1，存在

`sinter [key ...]` 求多个set集合的交集

`suinon [key ...]` 求多个set集合的并集

`sdiff [key ...]` 求多个set集合的差集

`spop [key]` 从set集合中随机弹出一个value值

使用场景：

1. 给用户添加标签，通过set集合的不可重复性和无序性
2. 抽奖，生成随机数,集合不能存放相同的元素,因此随机pop出一个元素,可以作为中奖的号码
3. 社交需求:可以给用户推荐有相同兴趣的人,用于交友

五、有序集合zset

不能有重复的元素,而且还可以排序,它和列表使用索引下标作为排序依据不同的是,它给每个元素设置一个分数(score)作为排序的依据

`zadd [key] [score] [value]` 向redis中存储一个有序set集合

`zcard [key]` 返回该key下的set集合的大小

`zscore [key] [value]` 返回key下的set集合的value值对应的score分数

`zincrby [key] [increment] [value]` 对key下的set有序集合中的value值的score基础上再加 increment值，来提高的排名

i. 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。 ii. 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：

1. 用户点赞数排行，排行榜
2. 展示用户信息及用户分数(例如学校中按学生分数排序)

六、HyperLogLog

Redis HyperLogLog 是用来做基数统计的算法，HyperLogLog 的优点是，在输入元素的数量或者体积非常非常大时，计算基数所需的空间总是固定 的、并且是很小的。

在 Redis 里面，每个 HyperLogLog 键只需要花费 12 KB 内存，就可以计算接近 2^64 个不同元素的基 数。这和计算基数时，元素越多耗费内存就越多的集合形成鲜明对比。

但是，因为 HyperLogLog 只会根据输入元素来计算基数，而不会储存输入元素本身，所以 HyperLogLog 不能像集合那样，返回输入的各个元素。

什么是基数?

比如数据集 {1, 3, 5, 7, 5, 7, 8}， 那么这个数据集的基数集为 {1, 3, 5 ,7, 8}, 基数(不重复元素)为5。 基数估计就是在误差可接受的范围内，快速计算基数。

`PFADD key [value...]` 添加指定元素到 HyperLogLog 中

`PFCOUNT [key]` 返回给定 HyperLogLog 的基数估算值

使用场景：

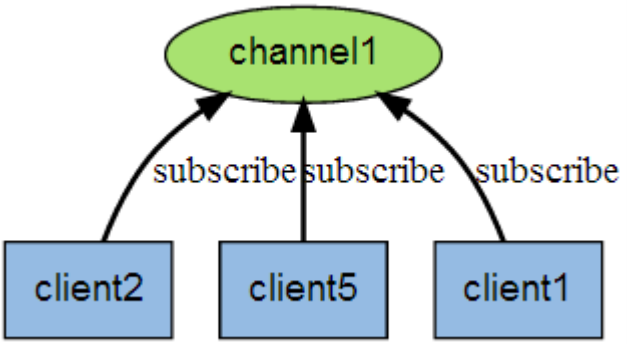
1. 例如统计 UV（user view）用户访问，PV（page view）页面访问，一般统计每个页面的UV是要去重的，可以用hyperloglog来实现统计，其实用set也可以统计一个页面的用户访问，因为set集合中是不会重复的。

七、发布订阅

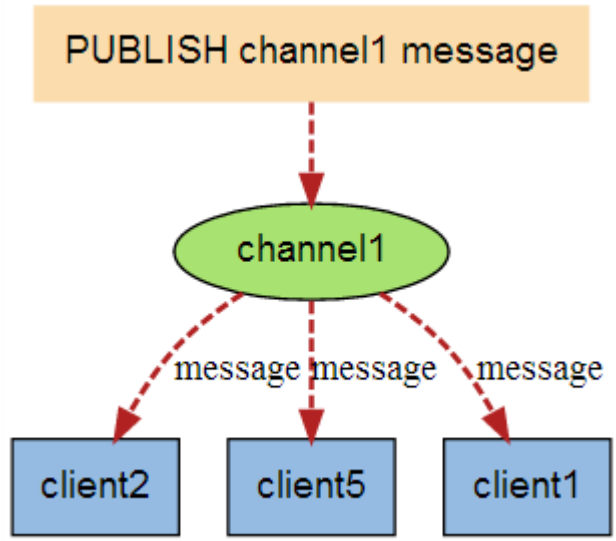
Redis 发布订阅(pub/sub)是一种消息通信模式：发送者(pub)发送消息，订阅者(sub)接收消息。

Redis 客户端可以订阅任意数量的频道。

下图展示了频道 channel1 ， 以及订阅这个频道的三个客户端 —— client2 、 client5 和 client1 之间的关系：



当有新消息通过 PUBLISH 命令发送给频道 channel1 时， 这个消息就会被发送给订阅它的三个客户端：



`SUBSCRIBE [channel ...]` 命令用于订阅给定的一个或多个频道的信息。

`PUBLISH [channel] [message]` 将信息发送到指定的频道，返回值接收到信息的订阅者数量。

`UNSUBSCRIBE [channel ...]` 退订给定的频道。