redis

#### **hash （对field进行数值计算）应用场景：点赞、收藏、详情页（）**

set sean：：name ‘kj’ 存储成功 键为 sean：：name 值为kj

get sean：：name 返回 值 kj （sean：：name 是一个完整的键）

set sean：：age 18

keys sean\* 结果 返回sean 开始的所有键 “sean：：name” “sean：：age ”

#### **(存储一个)**

hset sean name kj 存储sean 的name为kj

#### **(存储多个）**

hmset sean age 18 address bj 存储sean 的age为18， address 为bj

#### **（获取一个）**

hget sean name 返回结果 kj

#### **(获取多个)**

hmget sean name age 返回结果 kj 18

#### **（获取所有键）**

hkeys sean 返回结果 name age address

#### **（获取所有值）**

hvals sean 返回结果 kj 18 bj

#### **（获取所有键值对）**

HGETALL sean 返回结果 name kj age 18 address bj

#### **（对hash数值进行操作）**

HINCRBYFLOAT sean age 0.5 返回结果 18.5 （增加一个浮点数）

hget sean age 返回结果 18.5

HINCRBYFLOAT sean age -1 返回结果 17.5 支持负数操作

#### **SET （去重集合）无序、去重**

SMEMBERS 命令 (会消耗网卡吞吐量)

sadd k1 tom sean peter ooxx tom xxoo 返回结果 5 新增五条信息

SMMBERS k1 返回结果 sean tom ooxx petere xxoo （把重复的tom 去掉了）

srem k1 ooxx xxoo 返回结果 2 删除两条信息

SMEMBERS k1 返回结果 tom peter sean 删除了ooxx 和xxoo两条信息

#### **交 并 差 集**

sadd k1 4 5

sadd k2 1 2 3 4 5 sadd k3 4 5 6 7 8

#### **交集**

SINTER k1 k2 k3 做交集 返回 4 5

#### **返回结果存储到新的键里边（可以减少IO）**

SINTERSTORE dest k2 k3 返回结果 把 k2 k3的交集 存储到dest 里边

SMEMBERS dest 返回结果 4 5

#### **并集**

SUNION k2 k3 返回结果 1 2 3 4 5 6 7 8 （返回结果也回去重）

SUNIONSTORE 带有store 同样可以把返回的结果集放到 新的键值里边

#### **差集（有方向性）**

SDIFF k2 k3 返回结果 1 2 3 （第一个键值 为主 ）

SDIFF k3 k2 返回结果 6 7 8 （第一个键值 为主 ）

#### **产生随机事件**

sadd k1 tom1 tom2 tom3 tom4 tom5 tom6 tom7

SRANDMEMBER key count

count 是正数 ，取出有一个去重的结果集，数量为count，不能超过已有集合

count 是负数，取出一个带重复的结果集，一定满足数量

count为0 不返回

解决问题 ： 抽奖（10个奖品 ，用户数量：>10 或<10 ， 中奖：是否重复 ）

情况 ：多少种组合方式 人多还是奖品多 重复还是不允许重复

假 value 存储的是 人

SRANDMEMBER k1 3 返回结果 tom xoox oxox

SRANDMEMBER k1 3 返回结果 ooxx tom oxox

SRANDMEMBER k1 3 返回结果 ooxx tom xoox 三个结果一定不重复

但是

SRANDMEMBER k1 -3 返回结果 tom xoox oxox

SRANDMEMBER k1 3 返回结果 tom oxox xoox

SRANDMEMBER k1 3 返回结果 tom oxox tom 就允许出现一样的 （谢谢惠顾）

#### **(往外取出值,取出一个）**

SPOP k1 返回结果 oxox

SPOP k1 返回结果 ooxx

SPOP k1 返回结果 nil (所有信息全部弹出，返回空值)

SPOP k1 返回结果 nil (所有信息全部弹出，返回空值)

#### **sorted\_set zset(去重 对元素排序)（有正反向索引）**

底层排序两种方式：字典序 和 数值序 、

#### **一、 ：物理内存：左小右大，不随命令行改变**

假设想让 苹果 香蕉 鸭梨 去排序

排序方式： 按名称、按含糖量等

8 2 3 分值

苹果 香蕉 橘子 元素

0 1 2

-3 -2 -1 索引

zadd k1 8 apple 2 banna 3 orange

（增加命令 k1是健 8是分值 apple 是元素 ）

ZRWANGE k1 0 -1 返回结果 banana orange apple （ 按照 分值进行排序）

ZRANGE k1 0 -1 withscores 返回结果 banana 2 orange 3 apple 8 (按索引返回数据 带着分值)

ZRANGEBYSCORE k1 3 8 返回结果 orange apple （按数值返回结果 ）

ZRANGE k1 0 1 返回结果 banana orange (正向索引 获取数据)

Zrevrange k1 0 1 返回结果 apple orange (反向按索引 获取数据）

ZINCRBY k1 2.5 banana 返回结果 4.5 （banana的 分值增加2.5 成为4.5 ）

#### **二、有集合操作 （并集，交集）**

zadd k1 80 tom 60 sean 70 baby

zadd k2 60 tom 100 sean 40 yiming

#### **ZUNIONSTORE unkey 2 k1 k2**

#### **目标key 几个key 第一个key 第二个key 权重 聚合指令(默认是sun 相加操作)**

ZRANGE unkey 0 -1 withscores 返回结果 yiming 40 baby 70 tom 140 sean 160

（只有k2 有 ） （只有k1有） k1 k2 都有 所以相加

#### **权重操作**

ZUNIONSTORE unkey1 2 k1 k2 weight 1 0.5

ZRANGE unkey1 0 -1 withscores 返回结果 yiming 20 baby 70 sean 110 tom 110

k2 的所有分值 乘 0.5 k1的乘0.1

#### **并集**

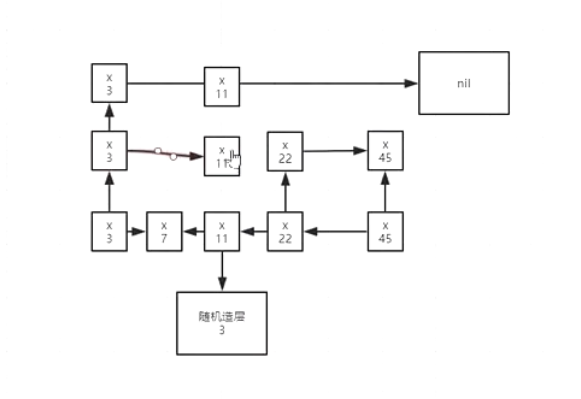
ZUNIONSTORE unkey1 2 k1 k2 aggregate max

ZRANGE unkey1 0 -1 withscores 返回结果 yiming 40 baby 70 tom 80 sean 100 （取两个分值最大的那个）

#### **交集（不存在 直接踢掉）**

#### **三、Zset排序是怎么实现的**

（ship list 存储结构 跳跃表 ）（类平衡树）



数据量比较多 平均值相对最优

#### **redis 进阶使用**