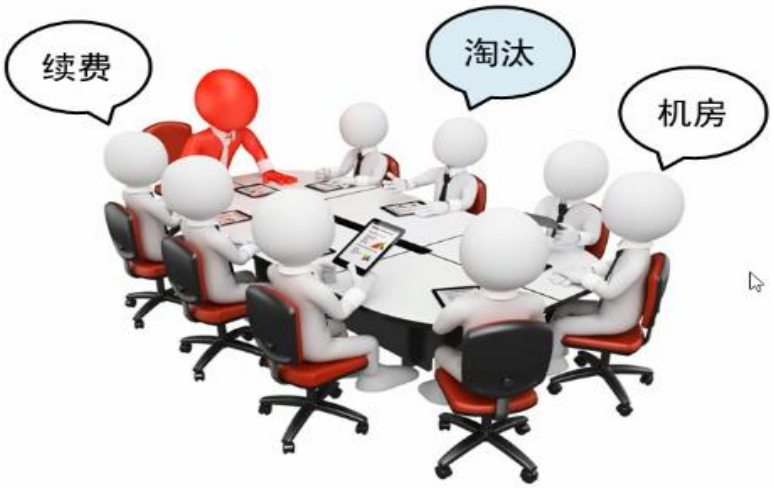
目录 Contents

- Bitmaps
- HyperLogLog
- GEO

Bitmaps

公司的年度总结会



Bitmaps

公司的年度总结会第二天

Bitmaps

存储需求

党员	党员	党员
是	Y	1

01001010

Bitmaps

存储需求

党员	党员	党员
是	Y	1

Bitmaps类型的基础操作

- 获取指定key对应偏移量上的bit值

```
getbit key offset
```

- 设置指定key对应偏移量上的bit值，value只能是1或0

```
setbit key offset value
```

Bitmaps类型的扩展操作

业务场景

电影网站

- 统计每天某一部电影是否被点播
- 统计每天有多少部电影被点播
- 统计每周/月/年有多少部电影被点播
- 统计年度哪部电影没有被点播

业务分析

01010011

Bitmaps

Bitmaps类型的扩展操作

业务场景

电影网站

- 统计每天某一部电影是否被点播
- 统计每天有多少部电影被点播
- 统计每周/月/年有多少部电影被点播
- 统计年度哪部电影没有被点播

业务分析

01010011

《非诚勿扰》

id:5

offset:4

01010011

01010011

01010011
or 11011001
11011001

11011001

Bitmaps

Bitmaps类型的扩展操作

- 对指定key按位进行交、并、非、异或操作，并将结果保存到destKey中

```
bitop op destKey key1 [key2...]
```

  - and：交
  - or：并
  - not：非
  - xor：异或
- 统计指定key中1的数量

```
bitcount key [start end]
```

Tips 21：

- redis 应用于信息状态统计

HyperLogLog

统计独立UV

- 原始方案：set
  - 存储每个用户的id（字符串）
- 改进方案：Bitmaps
  - 存储每个用户状态（bit）
- 全新的方案：Hyperloglog

HyperLogLog

基数

- 基数是数据集去重后元素个数
- HyperLogLog 是用来做基数统计的，运用了LogLog的算法

{1, 3, 5, 7, 5, 7, 8}　　基数集：{1, 3, 5 ,7, 8}　　基数：5

{1, 1, 1, 1, 1, 7, 1}　　基数集：{1,7}　　基数：2

HyperLogLog

LogLog算法

$$DV_{LL} = \text{constant} * m * 2^{\overline{R}}$$

$$DV_{HLL} = \text{constant} * m^2 * \left(\sum_{j=1}^{m} 2^{-R_j} \right)^{-1}$$

HyperLogLog

LogLog算法

Let $h : \mathcal{D} \to \{0,1\}^{32}$ *hash data from* $\mathcal{D}$ *to binary 32–bit words.*
Let $\rho(s)$ *be the position of the leftmost 1-bit of* s*: e.g.,* $\rho(1\cdots) = 1$, $\rho(0001\cdots) = 4$, $\rho(0^K) = K+1$.
define $\alpha_{16} = 0.673$; $\alpha_{32} = 0.697$; $\alpha_{64} = 0.709$; $\alpha_m = 0.7213/(1+1.079/m)$ for $m \geq 128$;
Program HYPERLOGLOG (**input** $\mathcal{M}$: multiset of items from domain $\mathcal{D}$).
assume $m = 2^b$ with $b \in [4\,..\,16]$.
initialize a collection of m registers, $M[1], \ldots, M[m]$, to 0;

for $v \in \mathcal{M}$ **do**
  set $x := h(v)$;
  set $j = 1 + \langle x_1 x_2 \cdots x_b \rangle_2$; {*the binary address determined by the first* b *bits of* x}
  set $w := x_{b+1} x_{b+2} \cdots$;
  set $M[j] := \max(M[j], \rho(w))$;
compute $E := \alpha_m m^2 \cdot \left(\sum_{j=1}^{m} 2^{-M[j]} \right)^{-1}$; {*the "raw" HyperLogLog estimate*}
if $E \leq \frac{5}{2}m$ **then**
  let V be the number of registers equal to 0;
  if $V \neq 0$ **then set** $E^* := m \log(m/V)$ **else set** $E^* := E$; {*small range correction*}
if $E \leq \frac{1}{30} 2^{32}$ **then**
  set $E^* := E$; {*intermediate range—no correction*}
if $E > \frac{1}{30} 2^{32}$ **then**
  set $E^* := -2^{32} \log(1 - E/2^{32})$; {*large range correction*}
return *cardinality estimate* E^* *with typical relative error* $\pm 1.04/\sqrt{m}$.

HyperLogLog类型的基本操作

- 添加数据

```
pfadd key element [element ...]
```

- 统计数据

```
pfcount key [key ...]
```

- 合并数据

```
pfmerge destkey sourcekey [sourcekey...]
```

Tips 22：

- redis 应用于独立信息统计

HyperLogLog

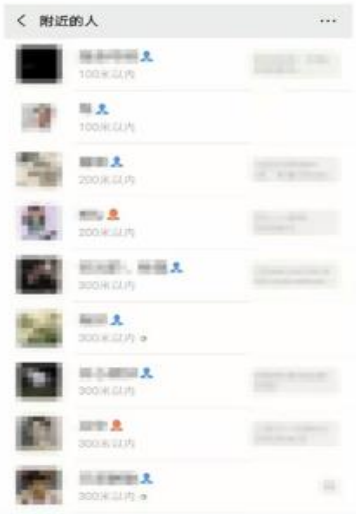
相关说明

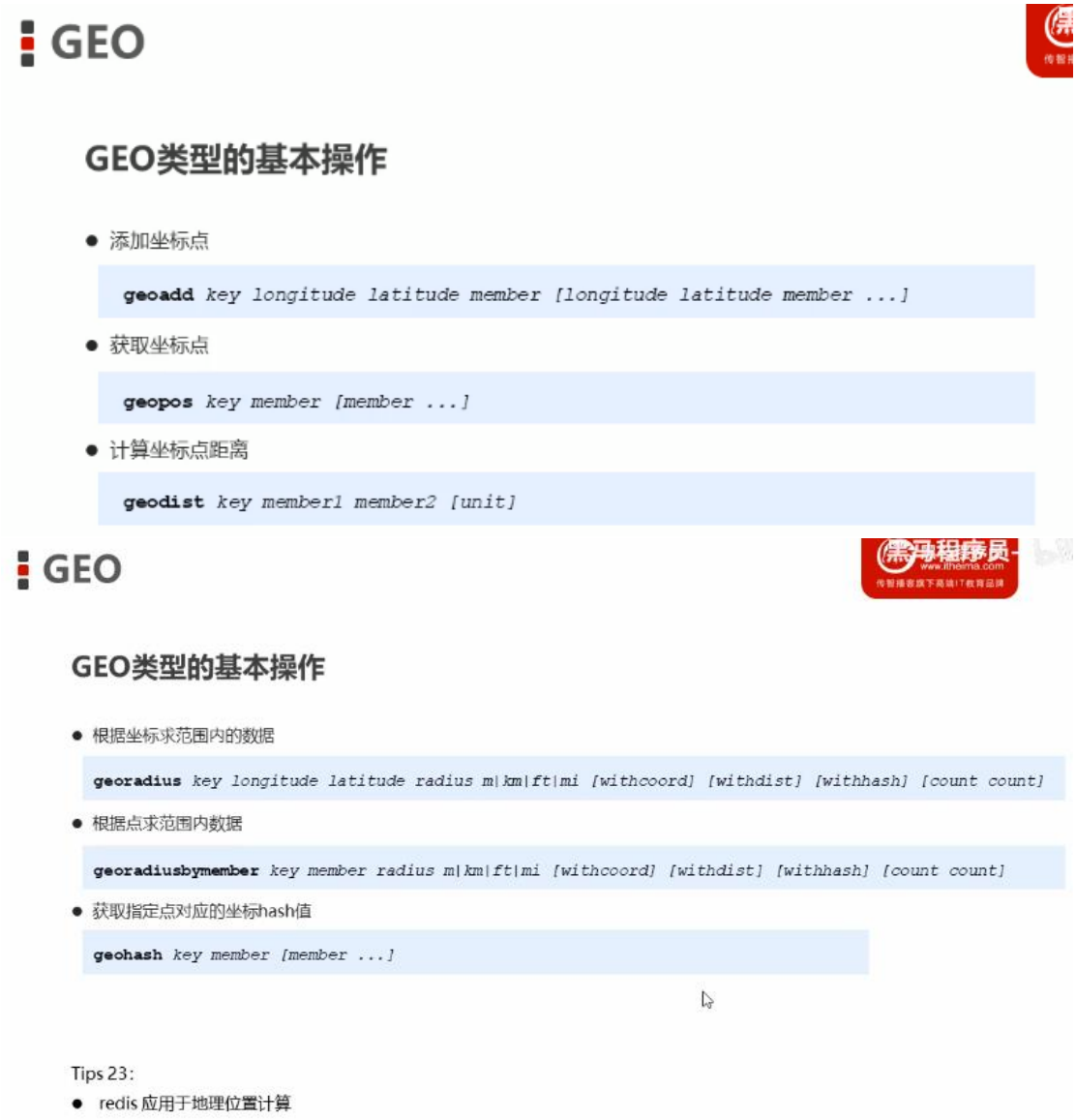
- 用于进行基数统计，不是集合，不保存数据，只记录数量而不是具体数据
- 核心是基数估算算法，最终数值存在一定误差
- 误差范围：基数估计的结果是一个带有 0.81% 标准错误的近似值
- 耗空间极小，每个hyperloglog key占用了12K的内存用于标记基数
- pfadd命令不是一次性分配12K内存使用，会随着基数的增加内存逐渐增大
- Pfmerge命令合并后占用的存储空间为12K，无论合并之前数据量多少

GEO

火热的生活服务类软件

- 微信 / 陌陌
- 美团 / 饿了么
- 携程 / 马蜂窝
- 高德 / 百度
- ……





同时上面两个函数还可以加 asc、desc 升序降序，案例：微信附近的人。