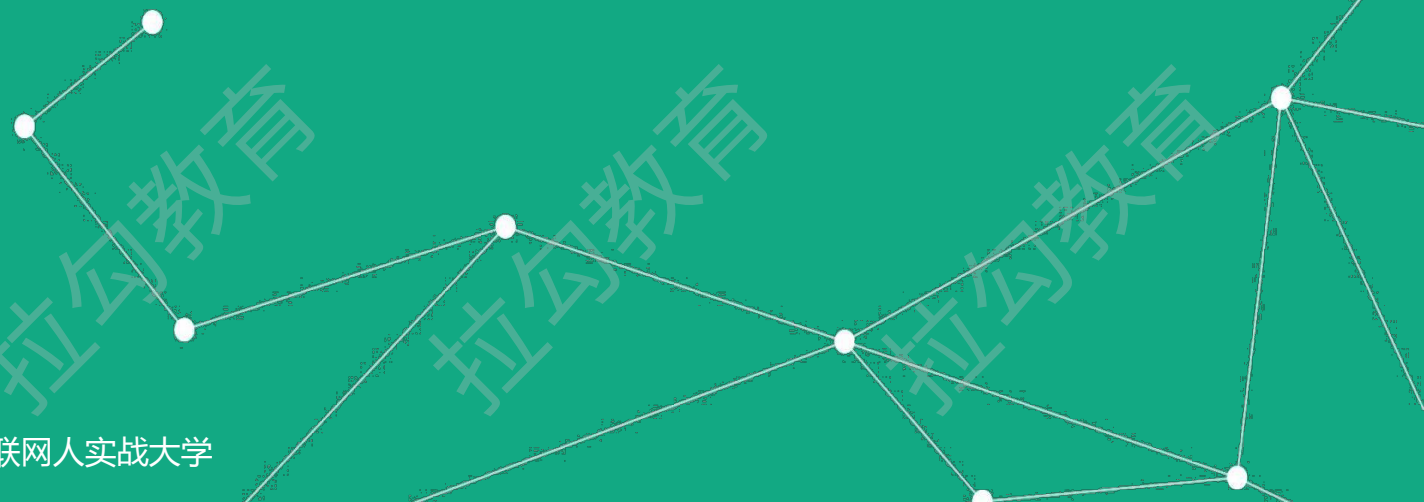
课时 02

位数组在 Redis 中的应用

1. 统计每个月学习专栏的用户活跃度
2. 比特与字节
3. 位数组
4. 位数组的实现
5. Redis 中的 Bitmap 数据结构



统计每个月学习专栏的用户活跃度

一个关于用户行为分析的问题：

2019 年 11 和 12 这两个月内都在学习《数据结构精讲：从原理到实战》这个专栏的用户有多少

```
SELECT COUNT(*) FROM nov_stats
INNER JOIN dec_stats
ON nov_stats.user_id = dec_stats.user_id
WHERE nov_stats.logged_in_month = "2019-11"
AND dec_stats.logged_in_mont = "2019-12"
```

这种做法需要进入到数据库中去读取数据并且做内连接

效率不是那么高

比特与字节

计算机处理信息时的最小单位其实就是一个二进制单位，即比特（Bit）

每 8 个二进制比特位都可以组成另一个单位，即字节（Byte），字节是内存空间中的一个基本单位

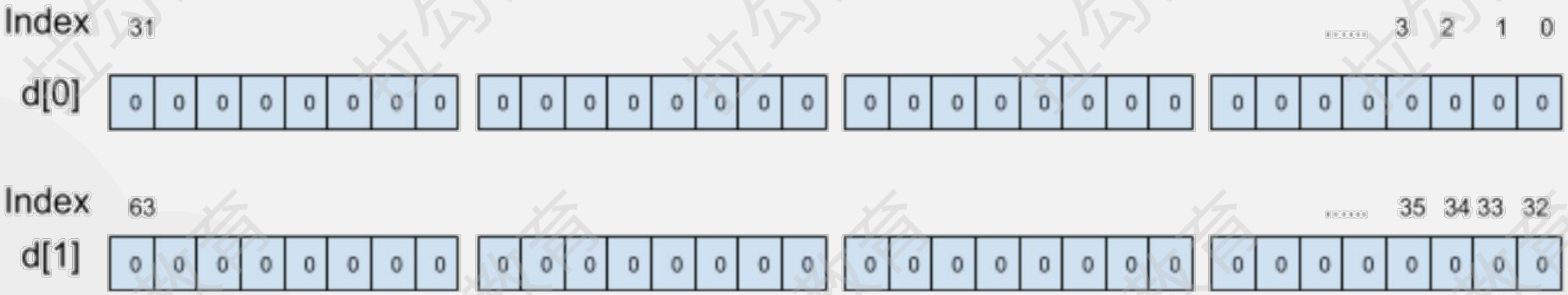
以 Java 为例，假设我们要声明一个大小为 2 的“比特数组”

```
int[] d = new int[2];
```

位数组

将每个元素中的每一个比特位都作为状态信息存储的数组称之为位数组（Bit Array）或者位图（Bit Map）

来看看上面声明的拥有两个元素的 int 数组的内存模型是怎么样的



位数组

当我们要操作位数组中在位置为“i”这个比特位的时候，应该如何定位它呢？

很简单，可以按照下面的公式来定位：

```
所在数组中的元素为： 32 / 32 = 1
比特位在元素中的位置为：35 % 32 = 3
```

一般来说因为位数组的元素只保存“0”或者“1”两种状态，所以对于位数组的操作有以下几种：

- 获取某个位置的比特位状态
- 设置某个位置的比特位，也就是将那个位置的比特位设置为“1”
- 清除某个位置的比特位，也就是将那个位置的比特位设置为“0”

位数组的实现

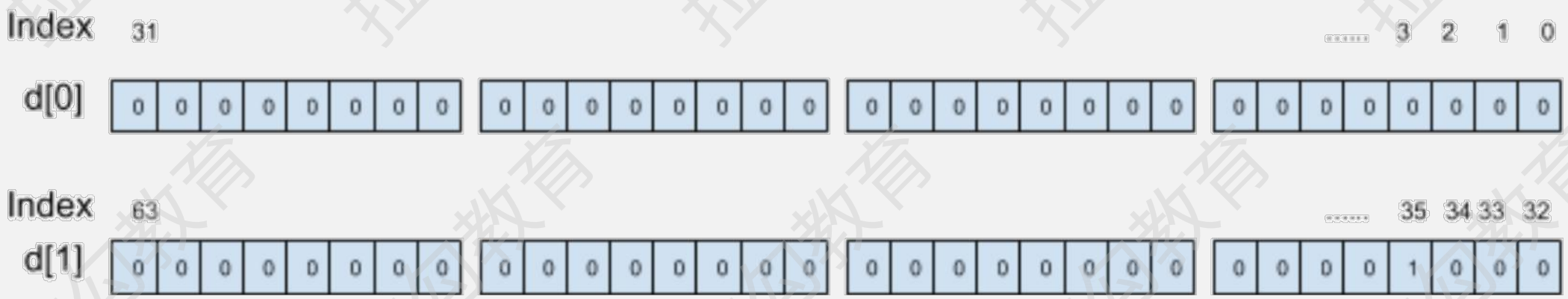
声明 GetBit 的方法签名为 `boolean GetBit(int[] array, long index);`

```
boolean GetBit(int[] array, int index) {
    int elementIndex = index / 32;
    int position = index % 32;
    long flag = 1;
    flag = flag << position;
    if ((array[elementIndex] & flag) != 0) {
      return true;
    } else {
      return false;
    }
}
```

这个方法将用于获取在 array 位数组中 index 位上的比特位是什么状态

- 如果为“1”则返回 true
- 如果为“0”则返回 false

位数组的实现



如果调用了 GetBit(d, 35) 这条语句，将得到 elementIndex 为 1、position 为 3、flag 为 0x08
将 d[1] 和 0x08 进行位操作的与运算，最后可以得出一个非 0 的结果，所以这个函数返回 true

如果调用了 GetBit(d, 32) 这条语句，我们将得到 elementIndex 为 1、position 为 0、flag 为 0x1，将 d[1] 和 0x1 进行位操作的与运算，最后可以得出一个 0 的结果，所以这个函数返回 false

位数组的实现

声明 SetBit 的方法签名为

```
void SetBit(int[] array, long index);
```

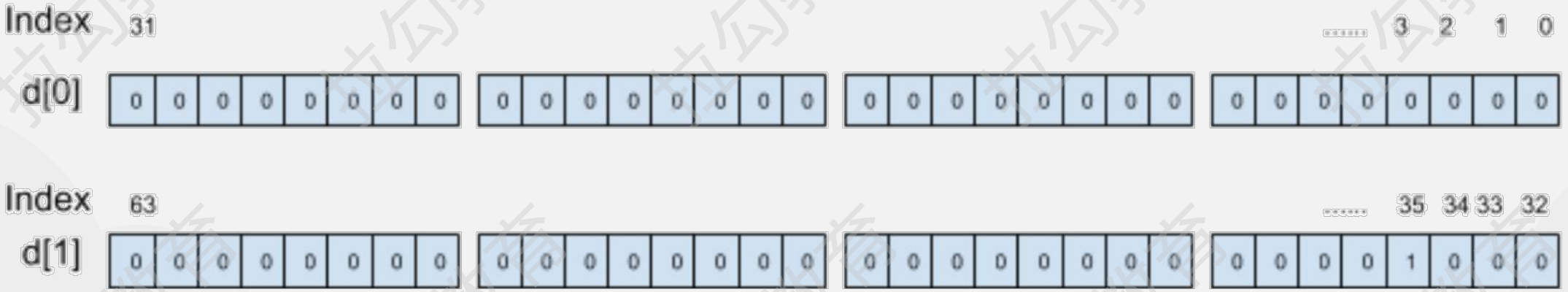
```
void SetBit(int[] array, int
index) {
    ...
    int elementIndex = index / 32;
    int position = index % 32;
    long flag = 1;
    flag = flag << position;
    array[elementIndex] | flag;
}
```

位数组的实现

如果调用了 SetBit(d, 35) 这条语句

我们将得到 elementIndex 为1、position 为 3、flag 为 0x08，将 d[1] 和 0x08 进行位操作的或运算

设置完之后位数组的状态如下图所示：

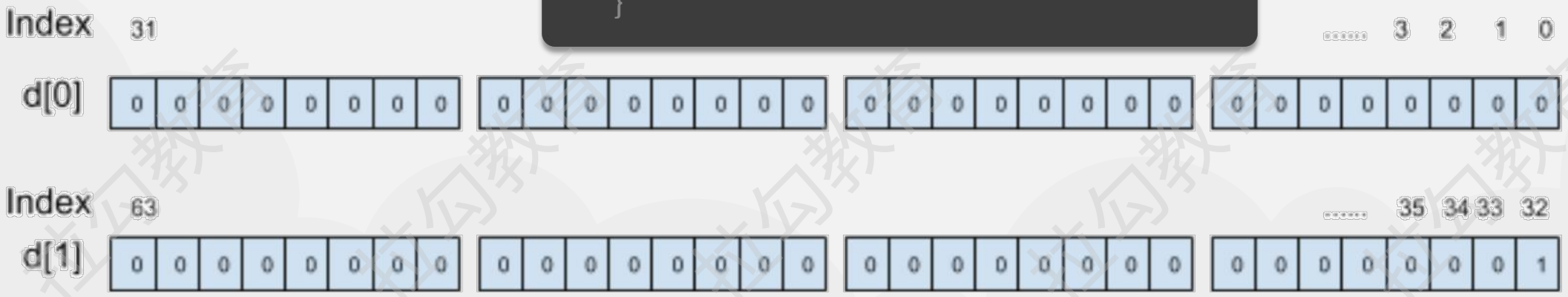


位数组的实现

声明 ClearBit 的方法签名为

```
void ClearBit(int[] array, long index);
```

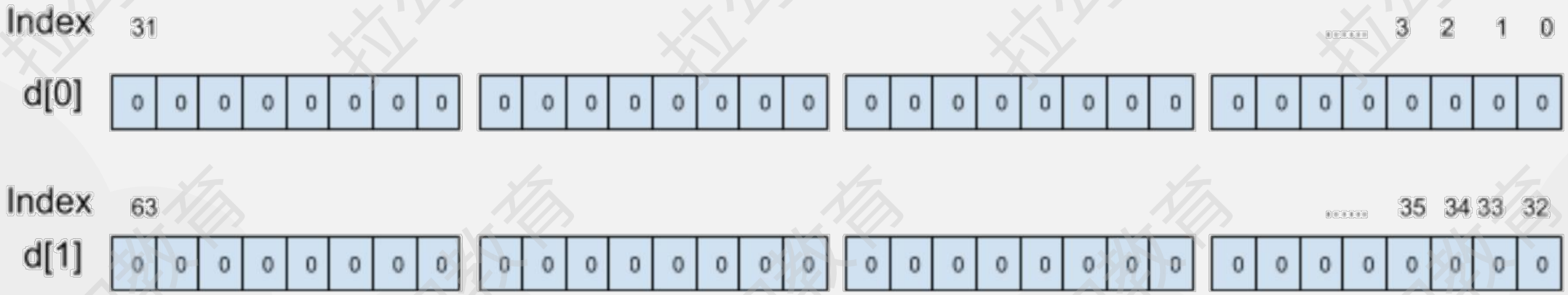
```
void ClearBit(int[] array, int index) {
    ...
    int elementIndex = index / 32;
    int position = index % 32;
    long flag = 1;
    flag = ~(flag << position);
    array[elementIndex] & flag;
}
```



位数组的实现

如果调用了 ClearBit(d,32) 这条语句，我们将得到 elementIndex 为1、position 为 1、flag 为 0xFFFFFFFE，将 d[1] 和 0xFFFFFFFE 进行位操作的与运算

设置完之后位数组的状态如下图所示：



Redis 中的 Bitmap 数据结构

Redis 是一个开源的并且使用内存来作为存储空间的高效数据库

感兴趣的同学可以到官网 https://redis.io 上查看相关文档

Redis 中的 Bitmap 数据结构

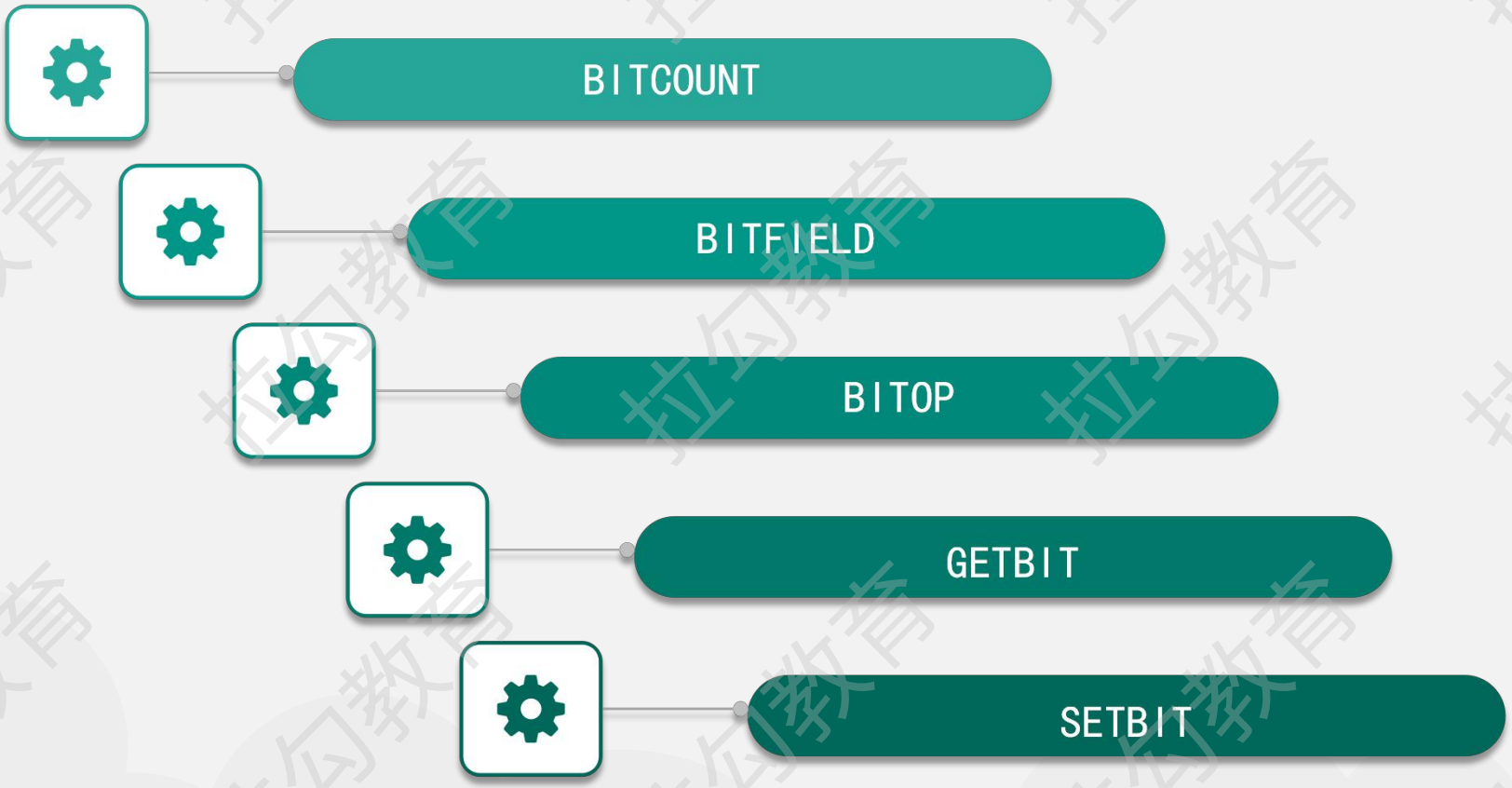
Bitmap，在这里其实就是位数组

Bitmap 的本质其实是在 Redis 里面通过一个 Strings 类型来表示的

在 Redis 中，Strings 的最大长度可以是 512MB

$$512MB = 2^{29} \text{ bytes} = 2^{32} \text{ bits} = 4294967296$$

Redis 中的 Bitmap 数据结构



Redis 中的 Bitmap 数据结构

如果想知道同时在 2019 年 11 和 12 月学习这个专栏的用户有多少，可以做怎样的优化呢？

可以用 Redis 里的 BITCOUNT、SETBIT 和 BITOP 来完成

BITCOUNT 这个命令其实是可以计算一个位数组里有多少比特位是为“1”的，而 BITOP 可以针对位数组进行“与”、“或”、“非”、“异或”这样的操作

Redis 中的 Bitmap 数据结构

首先针对 11 月学习的用户和 12 月学习的用户，可以为它们创建单独的位数组

例如 logins:2019:11 和 logins:2019:12

在 11 月，每当有用户登录学习时，程序会自动调用“SETBIT logins:2019:11 user_id 1”

同理，对于 12 月登录学习的用户，我们也可以调用“SETBIT logins:2019:12 user_id 1”

SETBIT命令可以将user_id在位数组中相应的位设为“1”

Redis 中的 Bitmap 数据结构

当要获得在这两个月内同时都学习了这个专栏的用户数时

可以调用“BITOP AND logins:2019:11-12 logins:2019:11 logins:2019:12”

将 logins:2019:11 和 logins:2019:12 这两个位数组做位运算中的与操作

将结果存放在“logins:2019:11-12”这个位数组中

最后调用“BITCOUNT logins:2019:11-12”来得到结果

Redis 的 Bitmap 操作之所以强大，是因为所有操作都是位运算以及发生在内存中，所以速度极快