阿里高频面试题：如何快速判断元素是不是在集合里？

Java技术迷 今天

以下文章来源于Hollis ， 作者zyz1992

Hollis

Hollis，一个对Coding有着独特追求的人。《Java工程师成神之路》系列作者、《程序员的三门课...

如何快速判断一个元素是不是在一个集合里？这个题目是我最近面试的时候常问的一个问题，这个问题不同人都有很多不同的回答。

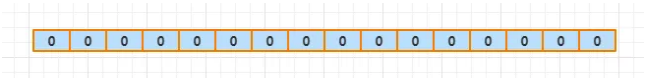
今天想介绍一个很少有人会提及到的方案，那就是借助布隆过滤器。

什么叫布隆过滤器

布隆过滤器（Bloom Filter）是一个叫做 Bloom 的老哥于1970年提出的。

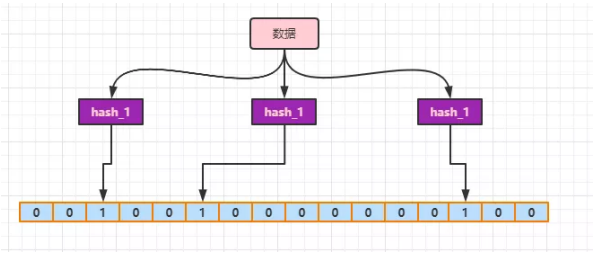
实际上可以把它看作由二进制向量（或者说位数组）和一系列随机映射函数（哈希函数）两部分组成的数据结构。

它的优点是空间效率和查询时间都比一般的算法要好的多，缺点是有一定的误识别率和删除困难。



实现原理

先来一张图



布隆过滤器算法主要思想就是利用 n 个哈希函数进行 hash 过后，得到不同的哈希值，根据 hash 映射到数组（这个数组的长度可能会很长很长）的不同的索引位置上，然后将相应的索引位上的值设置为1。

判断该元素是否出现在集合中，就是利用k个不同的哈希函数计算哈希值，看哈希值对应相应索引位置上面的值是否是1，如果有1个不是1，说明该元素不存在在集合中。

但是也有可能判断元素在集合中，但是元素不在，这个元素所有索引位置上面的1都是别的元素设置的，这就导致一定的误判几率（这就是为什么上面是活**可能**在一个集合中的根本原因，因为会存在一定的 hash 冲突）。

注意：误判率越低，相应的性能就会越低。

作用

布隆过滤器是可以用于判断一个元素是不是（**可能**）在一个集合里，并且相比于其它的数据结构，布隆过滤器在空间和时间方面都有巨大的优势。

注意上面的一个词：可能。这里先预留一个悬念，下文会详细分析到。

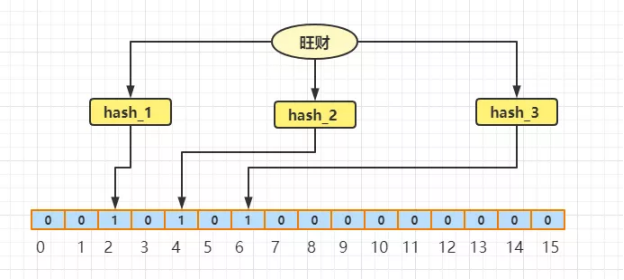
使用场景

- 判断给定数据是否存在
- 防止缓存穿透（判断请求的数据是否有效避免直接绕过缓存请求数据库）等等、邮箱的垃圾邮件过滤、黑名单功能等等。

具体实现

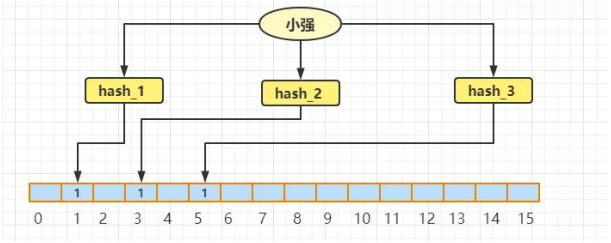
看完了布隆过滤器的算法思想，那就开始具体的实现的讲解。

我先来举个例子，假设有旺财和小强两个字符串，他们分别经过三次的 hash 算法，然后根据 hash 的结果将对应的数组（假设数组长度为 16）的索引位置的值置为1，先来看下**旺财**这个词组:

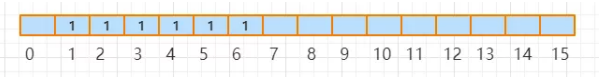


旺财经过三次 hash 过后，值分别为2,4,6 那么根据可以得到索引值分别为 2、4、6，于是就将该数组的索引（2、4、6）位置的值置为1，其余当做是0，现在假设需要查找旺财 ，同样经过这个三个hash然后发现得到的索引 2、4、6对应的位置的值都为1，那么可以判断**旺财**可能是存在的。

接着有将小强插入到布隆过滤器中，实际的过程和上面的一样，假设得到的下标是 1、3、5



抛开旺财的存在，小强此时是这样子在布隆过滤器中的，结合旺财和小强实际的数组是这样子的：



现在有来一个数据：**9527**，现在要求是判断 9527 是否存在，假设9527 经过三次 hash 过后得到的下标分别为：5、6、7。结果发现下标为 7 的位置的值为0，那么可以肯定的判断出，9527 一定不存在。

接着又来了一个 **国产007**，经过三次 hash 过后得到的下标分别为：2、3、5，结果发现 2、3、5下标对应的值全是1，于是可以大致判断出 **国产007**可能存在。但是实际上经过我们刚刚的演示，国产007根本就不存在，之所以 2、3、5 索引位置的值为1 ，那是因为其他的数据设置的。

说到这里，不知道大家有没有明白布隆过滤器的作用。

代码的实现

作为 java 程序员，我们真的是很幸福了，我们使用到很多的框架和工具，基本都被封装好了，布隆过滤器，我们就使用 google 封装好的工具类。

首先添加依赖

```
<!--布隆过滤依赖-->
<dependency>
    <groupId>com.google.guava</groupId>
    <artifactId>guava</artifactId>
    <version>25.1-jre</version>
</dependency>
```

代码的实现

```
import com.google.common.hash.BloomFilter;
import com.google.common.hash.Funnels;
import java.nio.charset.Charset;
public class BloomFilterDemo {
    public static void main(String[] args) {
        /**
         * 创建一个插入对象为一亿，误报率为0.01%的布隆过滤器
```

```
         * 不存在一定不存在
         * 存在不一定存在
         * ----------------
         *  Funnel 对象：预估的元素个数，误判率
         *  mightContain ： 方法判断元素是否存在
         */
        BloomFilter<CharSequence> bloomFilter = BloomFilter.create(Funnels.stringFunnel(Charset.forName("u
        bloomFilter.put("死");
        bloomFilter.put("磕");
        bloomFilter.put("Redis");
        System.out.println(bloomFilter.mightContain("Redis"));
        System.out.println(bloomFilter.mightContain("Java"));
    }
}
```

具体的解释已经写在注释中了。到这里相信大家一定明白了布隆过滤器和其怎么使用了。

实战

我们来模拟这样的场景：通过布隆过滤器来解决缓存穿透。

首先你的知道什么叫缓存穿透吧？

缓存穿透是指用户访问一个缓存和数据库中都没有的数据，因为缓存中不存在，所以就会去访问数据库，如果并发很高。很容易会击垮数据库

那布隆过滤器是如何解决这个问题的呢？他

的原理是这样子的：将数据库中所有的查询条件，放入布隆过滤器中，当一个查询请求过来时，先经过布隆过滤器进行查，如果判断请求查询值存在，则继续查；如果判断请求查询不存在，直接丢弃。

其代码如下：

```
String get(String key) {
    String value = redis.get(key);
    if (value  == null) {
        if(!bloomfilter.mightContain(key)){
```

```
            return null;
        }else{
            value = db.get(key);
            redis.set(key, value);
        }
    }
    return value;
}
```

小结

本文详细介绍了布隆过滤器是什么？有什么作用？实现原理以及从代码层面多方面来阐述布隆过滤器。学习能为各位在学习进阶的路上添砖加瓦

往 期 推 荐

1、致歉！抖音Semi Design承认参考阿里Ant Design

2、对比7种分布式事务方案，还是偏爱阿里开源的Seata，真香！

3、Redis存储结构体信息，选hash还是string?

4、扫盲 docker 常用命令

5、最全分布式Session解决方案

6、21 款 yyds 的 IDEA插件

7、真香！用 IDEA 神器看源码，效率真高！