## 前言

我是邢开春，有过几年搜索引擎开发的经验，现在搞了一个网站，一是交流分享，二是希望利用自己的工作经验赚取一些外快。现在已将自己多年搜索引擎的经验做成了一套视频（包含lucene、solr、elasticsearch），这套视频从入门使用到深入原理，面面俱到，可以帮助初学者快速入门，帮助使用者务实基础。本人的课程(博客文章、视频教程等)都力求在时长上尽量简短，在知识上尽量直击本质。

## lucene是什么

Lucene是一款开源的、高性能的、可拓展的信息检索Java工具库(也就是说lucene是提供搜索功能的Java jar包)。具体是下图的东西，也可以下载下来看一眼是什么。(下载链接)。将lucene引入Java项目中，你可以利用它做出类似百度、google提供的搜索服务。

## 什么时候应该考虑使用搜索引擎lucene?

第一点：有搜索功能的需求

想为App、网站添加类似”百度搜索的服务”搜索自家资源,想为用户提供类似”百度搜索的服务”搜索自家资源

第二点: 对搜索速度有要求

当mysql、oracle数据库搜索的速度太慢时，举一个例子，假设你有一亿篇文章，使用数据库提供的like ‘%关键词%’，将会耗时几小时才能查找出结果。而搜索引擎则能做到几十毫秒查询到结果。

mysql、oracle数据库的like查询会顺序扫描每一个数据记录，所谓顺序扫描，就是一个记录一个记录的查询扫描，对于每一个记录，从头看到尾，一个字符一个字符的检测是否包含被查询字符串。假设每一万记录耗时一秒，扫描一亿条记录，约耗时一万秒，约三小时。

第三点：对搜索结果的质量有要求

假设我们是一电商平台，用户搜索出来5000个商品，因为商品数据太多，可以用分页技术把数据展示给用户，一般用户也就查看前几页的几十个商品[这说明只有极小一部分商品数据才能被用户看到]，那么这总计5000个商品应当如何排序展示，才能给用户最好的体验(用户获取到心怡的商品)

[说句题外话，对于不同用户，由于身份，阅历不同，对待同一商品的满意程度是不一致的。当然，这篇文章不讨论这个问题]。

Lucene会通过评分模型为关键词与每一个被搜索出来商品一一计算一个相关度，然后商品按照与关键词的相关度排序，和搜索关键词密切相关的排在前面，越靠前的商品与搜索关键词相关度越高。mysql、oracle等数据库则不能很好地完成这个功能。

**lucene、mysql、oracle的区别？**

lucene是搜索引擎，mysql、oracle是关系型数据库，问lucene与mysql的区别，不如问搜索引擎与关系型数据库的区别

|  |  |  |
| --- | --- | --- |
|  | 搜索引擎 | 关系型数据库 |
| 搜索速度 | 亿级数据量，可以保持在几十毫秒内查询出结果 | 亿级数据量，不能很好支持查询like查询 |
| 搜索质量 | 搜索质量高。有评分模型，使得搜索结果中与搜索词相关度较高的结果排前面返回 | 搜索质量较差。数据库一般不关注搜索质量。数据库着重于对数据资源的管理。具体来说是增删改查, ACID。一般，数据库提供按照数字大小，字符编码进行排序的功能。 |
| 维护数据与数据之间的关系的能力 | 弱 | 专业 |

**lucene、solr、elasticsearch的区别？**

lucene是什么？（这里拍一个视频）

solr是什么？（这里拍一个视频）链接上服务器solr页面

elasticsearch是什么？（这里拍一个视频）链接上服务器elasticsearch页面

Lucene是什么？

lucene单纯的是一个jar包。它仅仅只提供搜索最核心的的开发接口(创建索引与利用索引进行搜索)，说直白点，它只是一个java语言的开发工具jar包。它没有用户操作界面，只支持Java编程语言开发，缺少其它编程语言的sdk[链接]，不能提供分布式搜索服务。

solr是什么？(链接)elasticsearch是什么？(链接)

之所以把solr与elasticsearch放在一起说，是因为它们都是企业级搜索引擎(solr、elasticsearch)。打个比方，它们的关系好比是oracle与mysql(它们都是关系型数据库)。企业级搜索引擎都有用户操作界面，有sdk(多种编程语言的开发包)，也就是说它可以被多种编程语言使用。提供分布式搜索功能。而它们的底层都是lucene，利用lucene提供的功能进行了二次开发，附加了众多的功能，形成企业级的搜索平台。

solr与elasticsearch的区别？

既然solr与elasticsearch都是企业级的搜索平台，它们的相同点有很多: 它们的底层都是lucene，它们都是企业级搜索平台，它们都提供分布式搜索功能，都能轻易处理亿级的数据量，都提供实时搜索功能......

那它们有什么区别呢？据我的观察，它们最大的区别是生态。solr是apache开源的软件，它开源，完全免费的企业级搜索平台。而elasticsearc是公司企业在维护推广，它部分产品收费，同时企业围绕着elasticsearch推出了一套完善的数据分析框架，也就是大名鼎鼎的ELK。

**lucene原理是什么？**

这个问题问的不够准确，一般人其实想问的问题包含两个:

一是lucene快速检索的原理是什么？

二是lucene的搜索效果为什么好？

第一个问题: lucene快速检索的原理是什么？

假设，我们有一个问题，如何在一亿个从小到大排序好的**有序**数据集中查找一个特定的数存不存在？

基本原理是：首先在有序的数字中找到中值(最中间的那个数的值)，将要查找的目标与中值进行比较，如果目标等于中值，则数据集中存在要查找的数；如果目标小于中值，若目标存在，则一定在前半部分数据中(这样，我们待查找的数据范围缩小了一半，在一亿条数据中查找和在五千万条数据中查找，差别还是蛮大的)，如果目标大于中值，则在后半部分查找；

如何在剩下的五千万条中查找呢？

同样的方法，先找这五千万条数据中的中值，如果目标等于中值，则数据集中存在要查找的数；如果目标小于中值，则在前半部分找；如果目标大于中值，则在后半部分找。

如何在剩下的二千五百万条数据中查找呢？

同样的方法，先找中值，如果目标等于中值，则数据集中存在要查找的数；如果目标小于中值，则在前半部分找；如果目标大于中值，则在后半部分找。

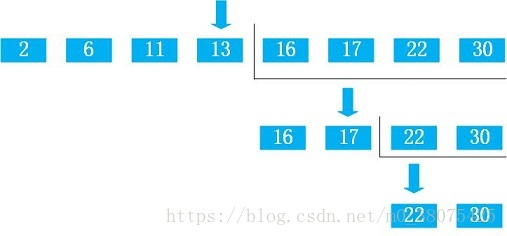
如何在剩下的一千二百五十万条数据中查找呢？

以此类推，直到找到目标为止。

用此方法，至多查询27次，就可确定一亿个有序的数中是否包含某个数。

案例

假设我们要在 2，6，11，13，16，17，22，30中查找22，上图所示，则查找步骤为：



 首先找到中值：中值为13（下标：int middle = (0+7)/2），将22与13进行比较，发现22比13大，则在13的后半部分找；

 在后半部分 16，17，22，30中查找22，首先找到中值，中值为17（下标：int middle=(0+3)/2），将22与17进行比较，发现22比17大，则继续在17的后半部分查找；

 在17的后半部分 22，30查找22，首先找到中值，中值为22（下标：int middle=(0+1)/2），将22与22进行比较，查找到结果。

二分查找大大降低了比较次数，二分查找的时间复杂度为：IMG_256，即IMG_257。

在一亿个数中查找某个数是否存在，仅仅需要查找27次。

假设，我们有一个问题，如何在一亿个**无序**的数据集中查找一个特定的数存不存在？

因为是无序、毫无规则，最容易想到，顺序的拿数据集中的每一个数字与待查询的数字比较。虽然可以解决这个问题，但是这需要比较一亿次。

换一种思路:如果我把无序的数据集整理成有序的数据集，则最多27次就可以查询到结果了。

我们先把无序的数字集整理成一份有序的数据集[这份有序的数据集被称为索引。]。然后我们在索引【有序的数字集】上查找待查询的数字。

补充一下:把无序的数字集整理成有序的数据集的动作被称为[创建索引]。

整理一下思路:由于无序的数据查找数字效率太低，我们创建了一份可以进行快速查找数据的有着特定数据结构的数据【索引】，在进行数字查找的时候，我们是在索引上进行查找的，所以查找速度提升了成千上万倍。

假设，我们有一个问题，如何在一亿篇文章中查找包含’lucene’的文章？

一种传统的做法叫做顺序扫描法：所谓顺序扫描，比如要找内容包含某一个字符串的文件，就是一个文档一个文档的看，对于每一个文档，从头看到尾，如果此文档包含此字符串，则此文档为我们要找的文件，接着看下一个文件，直到扫描完所有的文件。对于小数据量的文件，这种方法还是最直接，最方便的。但是对于大量的文件，这种方法就很慢了。

我们能不能仿照查找无序数字集中是否有某个数的方式创建一种利于搜索的数据结构的索引，然后利用索引进行搜索，从而加快搜索速度？

我们的目标是创造一种适合查找”关键词”被哪些文章包含的数据结构。那好，我就直接创建一种关键词到文件ID的映射的索引，不就完美解决问题了吗?

如果我先把一亿篇文章，先做成关键词到文件ID的映射，每个关键词都对应着一系列的文件，这些文件中都出现这个关键词。那么我在查找”关键词”被哪些文章时，是不是一下子就可以找到结果？



这个问题lucene的定义"lucene是基于java的开源全文索引工具包"已经给出了答案。通过lecene提供的工具，你对需要进行搜索的原数据（文档，网页，数据库等）进行索引操作，生成一份全文索引数据。在进行检索的时候，你不是对原数据, 而是对索引数据进行搜索的，所以你的检索效率才有了指数级提高(索引数据是有利于检索的数据结构)。



什么是全文索引？

全文索引，又称倒排索引(反向索引)(inverted index)，与之相对应的是正排索引(正向索引)(forward index)。

正排索引：无论课本，杂志，还是报纸，它们都拥有一个目录。假如我们想看某篇文章，通过目录，我们可以快速的找到这篇文章所在的页面，而不是笨拙的一页页的去翻看这本书。这里，目录就是一个索引。它有文章标题到文章位置(页码)映射的特殊数据结构，可以帮助我们去定位一个文章所在书本的位置。书籍的目录即是相对书籍中的文章的一份索引信息。像目录这种索引，因为通过文章名称(文章名称相当于文章的唯一标识)去找文章，是一个很自然的操作，所以目录这种索引，被称为正向索引（通过人名找人；通过手机品牌找手机）。

倒排索引是什么：与正排索引相对立，如果我想通过文章中的一部分内容去找这篇文章（例如："[http://bbkmgt.com](https://link.zhihu.com/?target=http://bbkmgt.com)"这个网址被包含在哪个文章之中？），就是一个反向操作了，不那么自然了，而且查找速度会特别慢（一本几百页的书籍，如果不做任何处理的话，想找到这个网址在哪里,你需要一行行去查找匹配，查找效率自然很慢）。好在我们可以针对书籍先做处理，找出所有包含"[http://bbkmgt.com](https://link.zhihu.com/?target=http://bbkmgt.com)"的文章，建立一个从"[http://bbkmgt.com](https://link.zhihu.com/?target=http://bbkmgt.com)"到文章的映射，通过映射，还是可以快速检索"[http://bbkmgt.com](https://link.zhihu.com/?target=http://bbkmgt.com)"在哪篇文章之中，这种不太自然的映射结构，被称为倒排索引（例如：如何通过一颗胎记去找人？建立一个胎记到人的映射。通过一颗零件去找手机？建立手机零件到手机的映射。）。虽然建立这个映射的过程比较耗时，但是如果我的查询次数比较多，这个前置的消耗可以被认为是值得的。

理解了全文索引，我们可以总结出，使用的lucene基本流程是①建立索引:为原文档建立索引。 ②检索：在索引中进行检索。下面，我们详细解说这两步，并引出一系列的细节操作与相关名词。

第二个问题: lucene的搜索效果为什么好？

这个问题比较复杂，我做了视频，详细讲解了lucene对搜索结果进行了打分的打分公式。

## lucene快速使用

lucene Maven地址

创建索引

利用索引进行搜索

lucene搜索为什么快？lucene做了哪些努力?

lucene调优？

lucene索引管理？

lucene打分算法

lucene源码解读