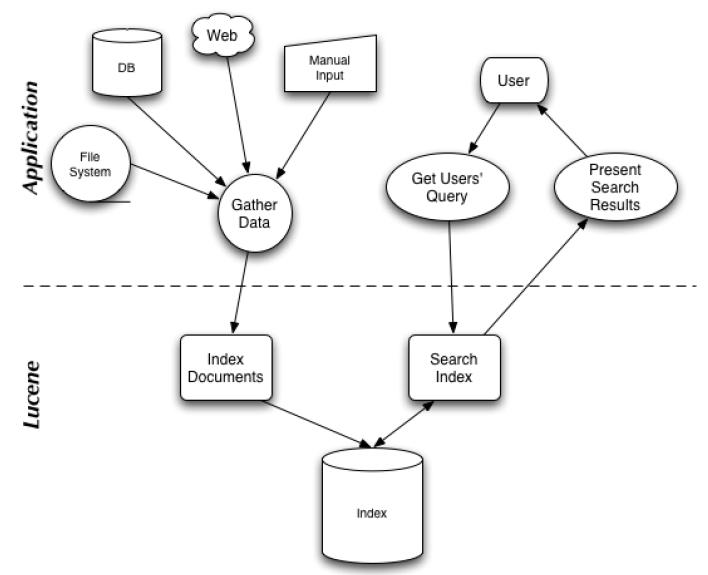
**Lucene 简介**

Lucene 是一个基于 Java 的开源的全文信息检索（基本思路：将非结构化数据中的一部分信息提取出来，重新组织，使其变得有一定结构，然后对此有一定结构的数据进行搜索，从而达到相对较好的搜索性能）工具包，它不是一个完整的搜索应用程序，而是为你的应用程序提供文本分析、索引（从非结构化数据中提取出的并重新组织的信息，就是索引）和搜索功能。Lucene 目前是 Apache Jakarta 家族中的一个开源项目。也是目前最为流行的基于 Java 开源全文检索工具包。Lucene采用倒排索引（inverted index）的机制。倒排索引就是说我们维护了一个词 / 短语表，对于这个表中的每个词 / 短语，都有一个链表描述了有哪些文档包含了这个词 / 短语。



我们可以看到，Lucene的使用主要体现在两个步骤：

　　1 创建索引，通过IndexWriter对不同的文件进行索引的创建，并将其保存在索引相关文件存储的位置中。

　　2 通过索引查寻关键字相关文档。

实现时 lucene将上面三列分别作为词典文件（Term Dictionary）、频率文件(frequencies)、位置文件 (positions)保存。其中词典文件不仅保存有每个关键词，还保留了指向频率文件和位置文件的指针，通过指针可以找到该关键字的频率信息和位置信息。为了减小索引文件的大小，Lucene对索引还使用了压缩技术，对词典中的关键词进行压缩。

Lucene原理：

Lucene是一个高性能的java全文检索工具包，它使用的是倒排文件索引结构。该结构及相应的生成算法如下：  
　　  
　　0）设有两篇文章1和2  
　　文章1的内容为：Tom lives in Guangzhou,I live in Guangzhou too.  
　　文章2的内容为：He once lived in Shanghai.  
　　  
　　1)由于lucene是基于关键词索引和查询的，首先我们要取得这两篇文章的关键词，通常我们需要如下处理措施  
　　a.我们现在有的是文章内容，即一个字符串，我们先要找出字符串中的所有单词，即分词。英文单词由于用空格分隔，比较好处理。中文单词间是连在一起的需要特殊的分词处理。  
　　b.文章中的”in”, “once” “too”等词没有什么实际意义，中文中的“的”“是”等字通常也无具体含义，这些不代表概念的词可以过滤掉  
　　c.用户通常希望查“He”时能把含“he”，“HE”的文章也找出来，所以所有单词需要统一大小写。  
　　d.用户通常希望查“live”时能把含“lives”，“lived”的文章也找出来，所以需要把“lives”，“lived”还原成“live”  
　　e.文章中的标点符号通常不表示某种概念，也可以过滤掉  
　　在lucene中以上措施由Analyzer类完成  
　　  
　　经过上面处理后  
　　 文章1的所有关键词为：[tom] [live] [guangzhou] [live] [guangzhou]  
　　 文章2的所有关键词为：[he] [live] [shanghai]  
　　  
　　2) 有了关键词后，我们就可以建立倒排索引了。上面的对应关系是：“文章号”对“文章中所有关键词”。倒排索引把这个关系倒过来，变成：“关键词”对“拥有该关键词的所有文章号”。文章1，2经过倒排后变成  
　　关键词               文章号  
　　guangzhou         1  
　　he                     2  
　　i                        1  
　　live                    1,2  
　　shanghai            2  
　　tom                   1  
　　  
　　通常仅知道关键词在哪些文章中出现还不够，我们还需要知道关键词在文章中出现次数和出现的位置，通常有两种位置：a)字符位置，即记录该词是文章中第几个字符（优点是关键词亮显时定位快）；b)关键词位置，即记录该词是文章中第几个关键词（优点是节约索引空间、词组（phase）查询快），lucene 中记录的就是这种位置。  
　　  
　　加上“出现频率”和“出现位置”信息后，我们的索引结构变为：  
  
倒排表  
　　关键词             文章号[出现频率]                     出现位置  
　　guangzhou        1[2]                                        3，6  
　　he                     2[1]                                       1  
　　i                        1[1]                                       4  
　　live                    1[2],2[1]                                 2，5，2  
　　shanghai            2[1]                                       3  
　　tom                   1[1]                                        1  
　　  
　　以live 这行为例我们说明一下该结构：live在文章1中出现了2次，文章2中出现了一次，它的出现位置为“2,5,2”这表示什么呢？我们需要结合文章号和出现频率来分析，文章1中出现了2次，那么“2,5”就表示live在文章1中出现的两个位置，文章2中出现了一次，剩下的“2”就表示live是文章2中第 2个关键字。  
　　  
　　以上就是lucene索引结构中最核心的部分。我们注意到关键字是按字符顺序排列的（lucene没有使用B树结构），因此lucene可以用二元搜索算法快速定位关键词。  
　　  
　　实现时 lucene将上面三列分别作为词典文件（Term Dictionary）、频率文件(frequencies)、位置文件 (positions)保存。其中词典文件不仅保存有每个关键词，还保留了指向频率文件和位置文件的指针，通过指针可以找到该关键字的频率信息和位置信息。  
　　  
　　 Lucene中使用了field的概念，用于表达信息所在位置（如标题中，文章中，url中），在建索引中，该field信息也记录在词典文件中，每个关键词都有一个field信息(因为每个关键字一定属于一个或多个field)。

Lucene压缩技术：

1. 采用可变字节编码的方式对其索引文件的整型数据进行压缩，该编码是一种变长编码，它是一种字节对齐的编码方式。它用不同字节数来表示不同范围的数，每个字节的最高位是标志位，如果最高位是0，则表示其后续没有字节；如果最高位是1，则表示其有后续字节。
2. 采用前缀压缩技术对倒排索引文件进行压缩，压缩后存储的信息为<前缀长度，后缀>，例如：当前词为“阿拉伯语”，上一个词为“阿拉伯”，那么“阿拉伯语”压缩为<3，语>。其次大量用到的是对数字的压缩，数字只保存与上一个值的差值（这样可以减小数字的长度，进而减少保存该数字需要的字节数）。例如当前文章号是16389（不压缩要用3个字节保存），上一文章号是16382，压缩后保存7（只用一个字节）。

Lucene索引模型：

二分搜索来快速定位关键词，再通过指向频率文件的指针读出所有文章的ID，然后将结果按相关排序规则输出。

Lucene软件包分析：

Package: org.apache.lucene.document

这个包提供了一些为封装要索引的文档所需要的类，比如 Document, Field。这样，每一个文档最终被封装成了一个 Document 对象。

Package: org.apache.lucene.analysis

这个包主要功能是对文档进行分词，因为文档在建立索引之前必须要进行分词，所以这个包的作用可以看成是为建立索引做准备工作。

Package: org.apache.lucene.index

这个包提供了一些类来协助创建索引以及对创建好的索引进行更新。这里面有两个基础的类：IndexWriter 和 IndexReader，其中 IndexWriter 是用来创建索引并添加文档到索引中的，IndexReader 是用来删除索引中的文档的。

Package: org.apache.lucene.search

这个包提供了对在建立好的索引上进行搜索所需要的类。比如 IndexSearcher 和 Hits, IndexSearcher 定义了在指定的索引上进行搜索的方法，Hits 用来保存搜索得到的结果。

## 建立索引

为了对文档进行索引，Lucene 提供了五个基础的类，他们分别是 Document, Field, IndexWriter, Analyzer, Directory。下面我们分别介绍一下这五个类的用途：

**Document**

Document 是用来描述文档的，这里的文档可以指一个 HTML 页面，一封电子邮件，或者是一个文本文件。一个 Document 对象由多个 Field 对象组成的。可以把一个 Document 对象想象成数据库中的一个记录，而每个 Field 对象就是记录的一个字段。

**Field**

Field 对象是用来描述一个文档的某个属性的，比如一封电子邮件的标题和内容可以用两个 Field 对象分别描述。

* Field.Index 🡪表示Field的索引方式
  + NO 🡪表示该Field不需要索引,也就是用户不需要去查找该Field的值
  + NO\_NORMS 🡪表示对该Field进行索引,但是不使用Analyzer,同时禁止它参加评分,主要是为了减少内存的消耗
  + TOKENIZED 🡪表示该Field先被分词再索引
  + UN\_TOKENIZED 🡪像链接地址URL、文件系统路径信息、时间日期、人名、居民身份证、电话号码等等通常将被索引并且完整的存储在索引中，但一般不需要切分词
  + Field.Store 🡪表示Field的存储方式
  + COMPRESS🡪压缩存储
  + NO 🡪原文不存储在索引文件中，搜索结果命中后，再根据其他附加属性如文件的Path，数据库的主键等，重新连接打开原文，适合原文内容较大的情况。
  + YES🡪索引文件本来只存储索引数据, 此设计将原文内容直接也存储在索引文件中，如文档的标题。

**Analyzer**

在一个文档被索引之前，首先需要对文档内容进行分词处理，这部分工作就是由 Analyzer 来做的。Analyzer 类是一个抽象类，它有多个实现。针对不同的语言和应用需要选择适合的 Analyzer。Analyzer 把分词后的内容交给 IndexWriter 来建立索引。

**IndexWriter**

IndexWriter 是 Lucene 用来创建索引的一个核心的类，他的作用是把一个个的 Document 对象加到索引中来。

**Directory**

这个类代表了 Lucene 的索引的存储的位置，这是一个抽象类，它目前有两个实现，第一个是 FSDirectory，它表示一个存储在文件系统中的索引的位置。第二个是 RAMDirectory，它表示一个存储在内存当中的索引的位置。

熟悉了建立索引所需要的这些类后，我们就开始对某个目录下面的文本文件建立索引了，清单 1 给出了对某个目录下的文本文件建立索引的源代码。

## 搜索文档

利用 Lucene 进行搜索就像建立索引一样也是非常方便的。在上面一部分中，我们已经为一个目录下的文本文档建立好了索引，现在我们就要在这个索引上进行搜索以找到包含某个关键词或短语的文档。Lucene 提供了几个基础的类来完成这个过程，它们分别是呢 IndexSearcher, Term, Query, TermQuery, Hits. 下面我们分别介绍这几个类的功能。

**Query**

这是一个抽象类，他有多个实现，比如 TermQuery, BooleanQuery, PrefixQuery. 这个类的目的是把用户输入的查询字符串封装成 Lucene 能够识别的 Query。

**Term**

Term 是搜索的基本单位，一个 Term 对象有两个 String 类型的域组成。生成一个 Term 对象可以有如下一条语句来完成：Term term = new Term(“fieldName”,”queryWord”); 其中第一个参数代表了要在文档的哪一个 Field 上进行查找，第二个参数代表了要查询的关键词。

**TermQuery**

TermQuery 是抽象类 Query 的一个子类，它同时也是 Lucene 支持的最为基本的一个查询类。生成一个 TermQuery 对象由如下语句完成： TermQuery termQuery = new TermQuery(new Term(“fieldName”,”queryWord”)); 它的构造函数只接受一个参数，那就是一个 Term 对象。

**IndexSearcher**

IndexSearcher 是用来在建立好的索引上进行搜索的。它只能以只读的方式打开一个索引，所以可以有多个 IndexSearcher 的实例在一个索引上进行操作。

**Hits**

Hits 是用来保存搜索的结果的。

内建Query对象：

1. BooleanQuery
2. BooleanClause
3. RangeQuery
4. PrefixQuery
5. PhraseQuery
6. FuzzyQuery
7. WildcardQuery
8. SpanQuery
9. 其它