1.搜索质量:查全率,衡量搜索系统查找相关文档的能力;

查准率,衡量搜索系统过滤非相关文档的能力.

2.索引过程中的核心类

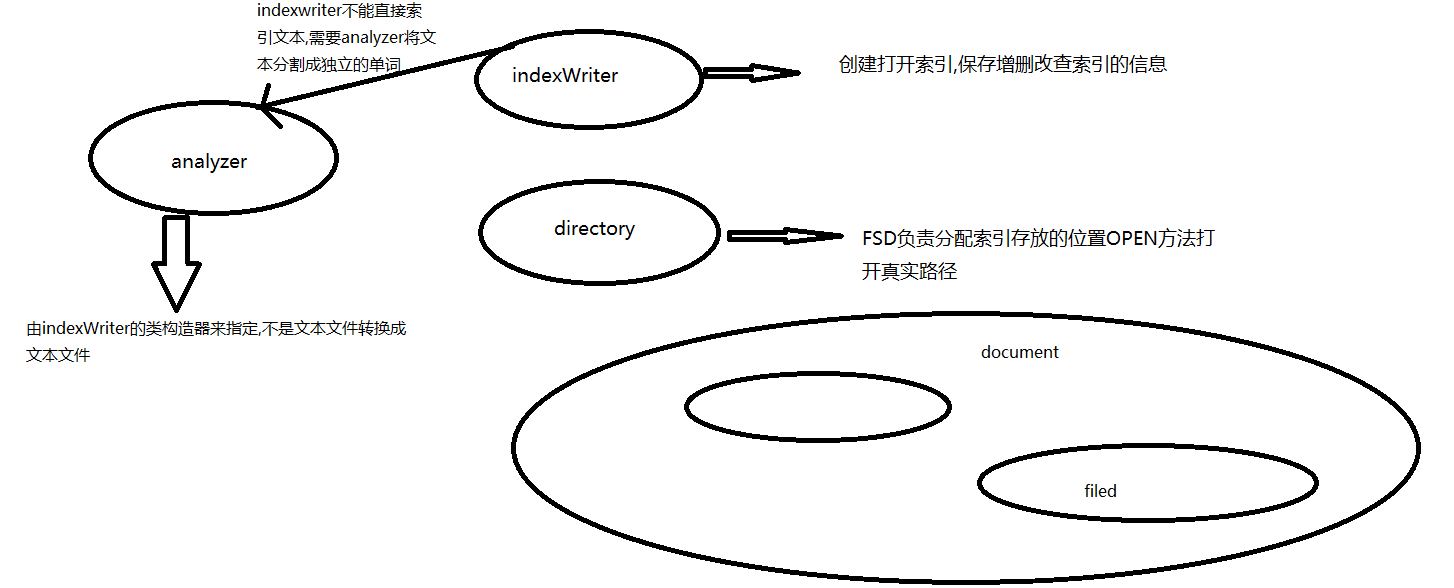
IndexWriter:索引的核心组件,负责创建新索引,打开已有索引,以及向索引中添加,删除或更新被索引文档的信息,提供对索引文件尚未写入操作,但不能用于读取或搜索索引,需要有一定的空间来存储索引,功能有directory完成.

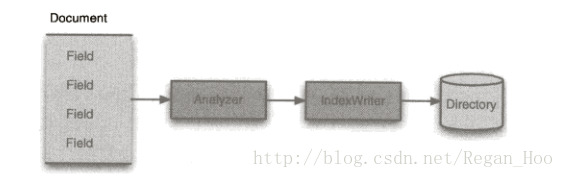
Directory:描述了Lucene索引的存放位置,是一个抽象类,子类具体负责制定索引的存储路径,用FSDirectory.open 方法获取真实文件的存储路径,然后将他们一次传递给IndexWriter的类构造方法,indexWriter不能直接索引文本,需要Analyzer现将文本分割成独立的单词.

Analyzer:文件文本被索引之前,需要Analyzer分析器处理,Analyzer是由indexWriter的类构造器来制定的,负责从文本中提取语汇单元,提出剩余无用信息,如果索引内容不是文本文件,那么需要先转换成文本文件,

document:对象代表一些域(field)的集合,文档的域代表文档或者文档相关的一些元数据,元数据作为文档的不同域单独存储并被索引,Document对象结构比较简单,为一个包含多个域对象的容器,域表示能被索引的文本的类.

field:索引的每个文档都包含一个或者多个不同命名的域,这些域包含在field类中,每个与对应一个域名和相应的阈值,以及一组选项来精确控制Lucene所以操作各种阈值





3.搜索过程中的核心类

indexSearcher:去搜索indexWriter创建的索引,是连接索引的中间环节,可以将其看作一只读方式,打开索引的类,需要利用directory实例来掌控前期创建的索引,提供搜索方法,

Term:是搜索功能的基本单元,Term对象包含一对字符串元素,:域名和单词(域名文本值),

Query:包含了一些非常有用的方法,TermQuery是他的子类

TermQuery:提供最基本的查询,用来匹配指定域中包含的特定向的文档.

TopDocs:该类是一个简单的指针容器,指针一般指向前N个排名的搜索结果,结果是匹配查询的文档

4.Lucene对搜索内容进行建模,

文档和域:

文档是Lucene索引和搜索的基本单位,文档为包含一个或者多个域的容器,而域则依次包含真正被搜索的内容.每个与都有一个标识名称,该名称为一个文本值或者二进制值,如用户在输入title:Lucene时 搜索的结果为标题阈值含有Lucene的所有文档.

Lucene针对与可以进行的三种操作:

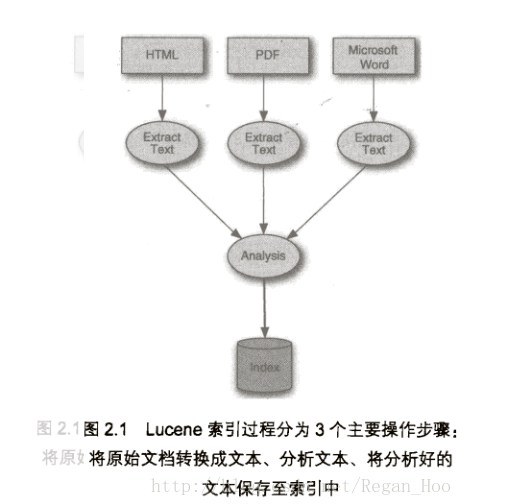
---阈值可以被索引.如果需要搜索一个域,则必须首先对他进行索引.被索引的阈值必须是文本格式的(二进制格式的阈值只能被存储,不能被索引).在索引一个域时,需要首先使用分析过程将阈值转化为语汇单元,然后将语汇单元加入到索引中.

---域被索引后,还可以选择性地存储项向量,项向量可以看做域的一个小型反向索引集合,通过项向量能够检索该域的所有语汇单元,这个机制有助于实现一些高级功能,比如搜索与当前文档相似的文档.

---阈值可以被单独存储,,也就是被分析前的阈值也可以写进索引,以便后续的检索,这个机制可以将原始值展现给用户,比如文文档的标题或摘要.

当搜索程序通过索引检索文档时,只有被存储的域才会被作为搜索结果展示,

Lucene要求在进行索引操作时简单化或者规格化原始数据.



在索引操作期间,文本先从原始数据中提取出来,并允许创建对应的Document实例,该实例包含多个FIeld实例,,他们都用来保存原始数据,随后的分析过程将文本处理成大量语汇单元,,最后将语汇单元加入到段结构中.

----------------------------------------------------------------

提取文本和创建文档,使用Lucene索引数据时,必须先从数据中提取存文本格式信息,以便Lucene识别该文本并建立对应的LLucene文档.

分析文档,一旦建立Lucene文档和域,就可以调用IndexWriter对象的addDocument方法将数据传递给Lucene进行索引操作了,在索引操作时,Lucene首先分析文本,将文本数据分割成语汇单元,进行操作.

向索引添加文档,对输入数据分析完毕后.,就可以将分析文件写入索引结果中,Lucene将输入数据以一种倒排序索引的数据结构进行存储.进行关键字查询时,这种数据结构可以有效地利用磁盘空间,Lucene使用倒排数据结构的原因是,把文档中提出的语汇单元最为查询关键字,而不是将文档作为中心实体,,相当于所以与页码的对应关系.

索引段,Lucene索引都包含一个或多个段,每个段都是一个独立的索引,他包含整个文档索引的一个子集.每当writer刷新缓存区增加的文档,以及挂起目录删除操作时,索引文件都会建立一个新段..在搜索索引时,每个段都是单独访问的,但搜索结果是合并后返回的.

每个段都包含多个文件,文件格式是\_X,这里X代表段名称,为扩展名,用来标识该文件对应索引的某个部分,如果使用混合文件格式(Lucene默认的处理方式),文件都会被压缩成一个单一的文件:\_X.cfs.这种方式能够在搜索期间减少打开的文件数量.

java完整搭建Lucene

创建索引

1.创建一个model对象

2.通过FSDirectory.open创建索引库(指定路径不含文件名)

Directory directory = FSDirectory.open(new File("E:\\s"));

3.创建分词器,StardardAnalyz(版本)

Analyzer analyzer = new StandardAnalyzer(Version.LUCENE\_30);

4.创建索引写入类directory和anaylzer和域对象的最大长度

IndexWriter indexWriter = new IndexWriter(directory, analyzer,

MaxFieldLength.LIMITED);

5.将model对象转化为document

Document document=new Document();

Field 自定义的field=new Filed("自定义的名称","名称的熟悉");

将filed域对象写入document中

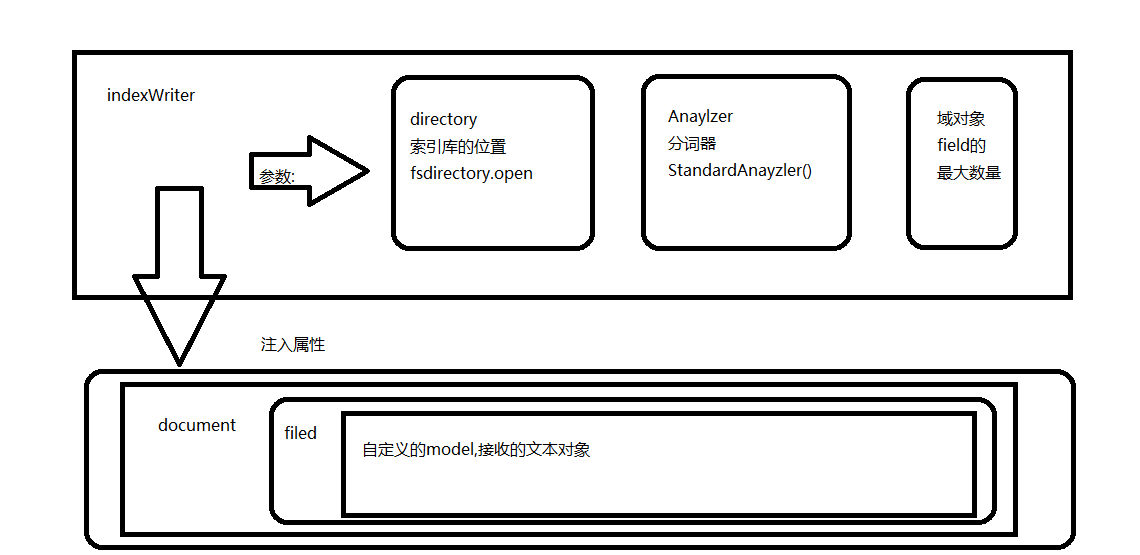
document.add(field);

将document写入indexwriter中

indexwriter.addDocument(document)

6.关闭indexwriter

任何读写流都需要关闭



读索引

1.创建(理解为打开)索引库

Directory directory = FSDirectory.open(new File("E:\\s"));

2.创建indexSearcher(创建索引读取器)参数只有一个(索引库)

IndexSearcher indexSearcher = new IndexSearcher(directory);

3.创建分词器

Analyzer analyzer = new StandardAnalyzer(Version.LUCENE\_30);

4.创建查询对象queryParser(分词器版本,查询的内容,分词器)

QueryParser queryParser = new QueryParser(Version.LUCENE\_30, "content", analyzer);

5.通过QueryParser创建Query(理解为要进行查询的对象)

Query query = queryParser.parse(查询的关键词);

6.定义显示的结果条数(使用所以查询器去查找)理解(与查询结果的权重有关)

TopDocs topDocs = indexSearcher.search(查询结果, 条数);

topDocs.totalHits(),查询结果的条数.

7.SoreDocs表示一个结果的相关得分度与文档编号等信息的对象.

topDocs.soreDocs();

float score = scoreDoc.score;// 关键词得分

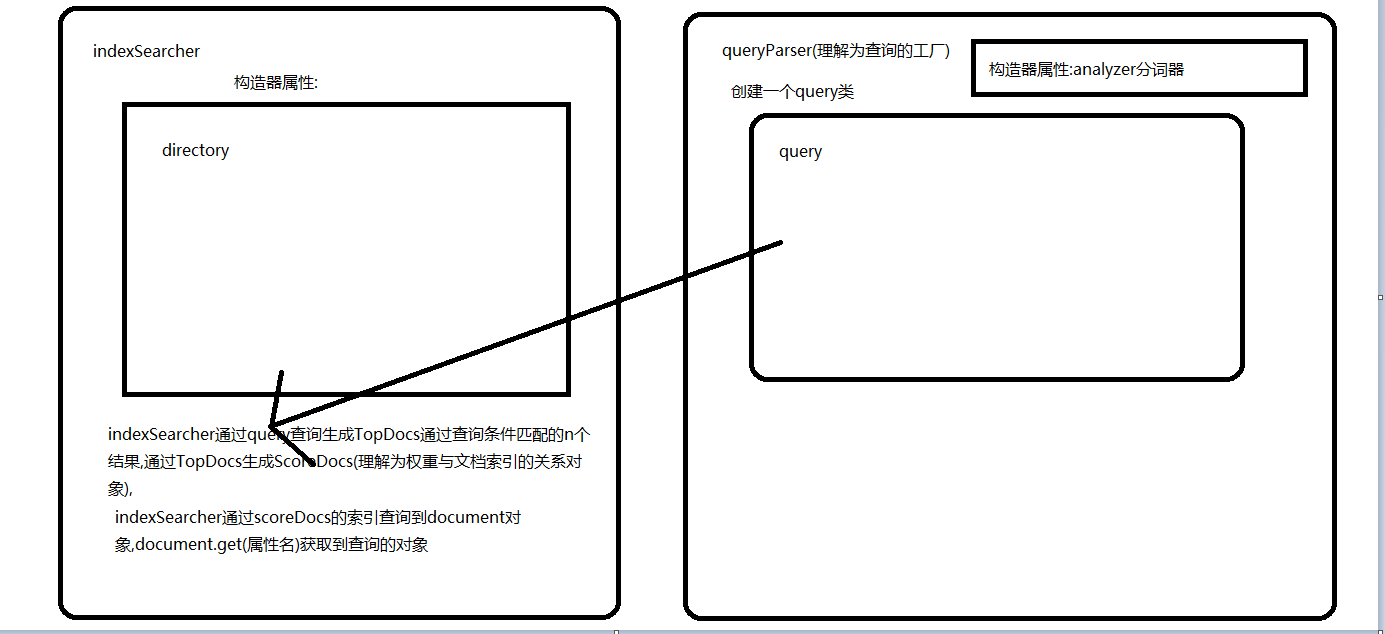
int index = scoreDoc.doc;// 索引的下标

8.获取document对象

Document document = indexSearcher.doc(index);

9.通过document对象获取到article

(????????????????不需要关闭,能关闭)



solr的使用

1.启动solr ,使用命令为java -jar start.jar --module=http

2.默认的导入数据 java -jar post.jar solr.xml moniter.xml

3.数据的导入

使用DIH从数据库导入数据,

支持CSV文件导入,因此Excel数据能导入

支持JSON格式文档,

二进制文档:word,PDF

支持以编程的方式自定义导入

4.更新数据

solr根据文档的唯一id创建索引,文档id存在,覆盖,

5.删除数据

通过id删除文档,或者同查询删除匹配的文档.

solr删除数据也有事务,可以设置commit

6.查询排序,倒序(正序,倒序,多个字段排序)

高亮:h1=true h1.f1=name 需要高亮的字段 文本分析(例如将连续的字段携程a\_B\_A,就可以搜索 ab和a-b)

相关配置:

1.schema.xml solr模式的关联文件

1.1 fieldType用来定义数据类型

name用来定义节点名称 class指定定义的类型名称

可以自定义当前类型建立索引,指定分析器(包括分词器和过滤器)

1.2 field 指定建立索引和查询数据的字段

indexed,表示是否被索引

stored表示是否被存储

version 和root节点是必须保留的,不能删除

1.3 copyfield 将一些字段复制到另一个字段中,可以通过一个字段搜索全部

1.4 dynamicField 表示动态字段,可以动态定义一个字段,只要符合规则的字段都可以相当于通配符\*

1.5 uniqueKey 是文档的唯一标识,相当于主键

1.6 defaultSearchField 搜索的时候默认搜索的值

1.7 solrQueryParser 指定搜索多个词时的关系

1.8 性能优化

只用于搜索不返回结果的filed设置为不存储

不用于搜索,只返回结果的field不设置索引

删除不必要的复制字段,为了索引字段的最小化和搜索的效率

所有text的index都设置为false,将其复制到一个总的field中,进行搜索

2. solrConfig.xml 定义了一些solr的处理规则,包括索引的存放位置,更新删除,查询的规则配置

2.1 datadir 定义了索引数据和日志数据的存放位置

2.2 LuceneMatchVersion 表示solr底层使用Lucene的版本

2.3 lib solr引用包的位置,

2.4 directoryFactory 索引的存储方式

2.4.1 standard 基于文件系统存储目录,基于操作系统和Java虚拟机版本

2.4.2 simpleFS 适用于小型系统,不适于大数据和多线程

2.4.3 NIOFS 适用于多线程,不适于windows(慢),jvm存在bug

2.4.4 MMap linux上的事项

2.4.5NRTCaching 可以做缓存,将索引存储在内存中,

2.4.6 RAM 是一个内存存储方案,不能持久化,不支持索引复制

2.5 CodecFactory 编解码工厂

2.6 indexconfig 设置索引的低级别的属性

token的长度,write等待锁的时间,生成索引的最大线程数,是否使用混合文件,缓存的大小,合并策略,合并因子,合并调度器,锁工厂.....

索引库的锁设置,single只适合只读的索引库,不能更改

native,使用本地操作系统的文件锁方式,不能多个solr共用一个索引库

simple,使用简单的文件锁

2.7 updateHandler 更新索引库的日志

2.8 Query查询

2.9 RequestDispatcher

3 加入IK中文分词器