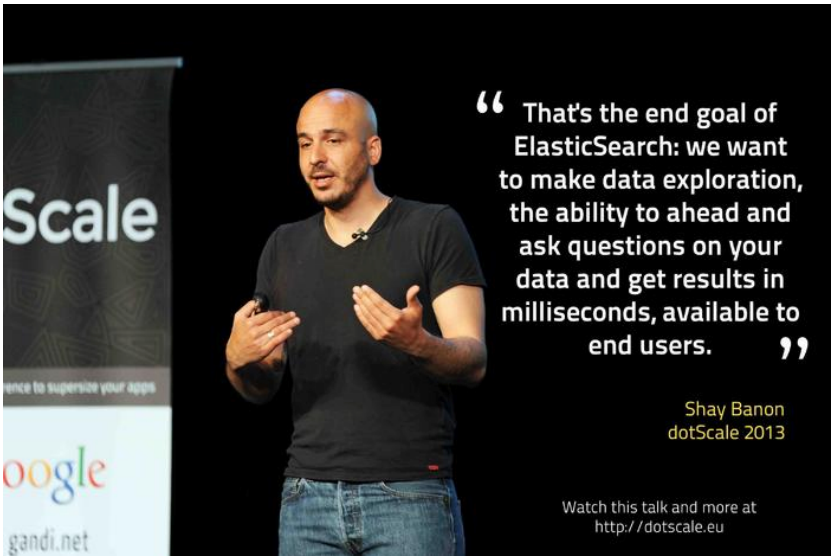
who:谁发明了它

就是下面这位大哥 Shay Banon



What:elasticsearch是什么

elasticsearch开源并且免费(很关键)的全文检索和分析的引擎，可以快速存储，搜索数据，还可以科学的分析数据

然额，elasticsearch本质上就是一个分布式的数据库，允许多台服务器协同工作，每台服务器可以运行多个elasticsearch实例,单个的elasticsearch实例就是一个节点 node ，一组节点构成cluster集群

隐藏luence 复杂性，提供简单易用的restful api接口、java api接口（还有其他语言的api接口

原理:http://wangnan.tech/post/elkstack-es01/

luence 了解的必要性

需求总是来源于生活

结构化数据 ：数据库

非结构化数据:word文档 ，邮件等

对于非结构化数据常用的是顺序扫描法，势必由于数据量过多造成查询速度过慢

由此 我们对非结构化数据抽取出部分有结构的数据 重新组织 称之为索引

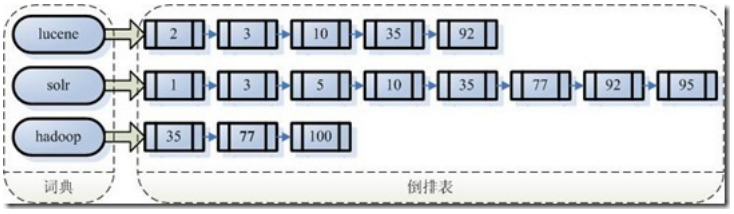
好比如字典中的词汇很多，但是很多词汇的拼旁部首，韵母等抽取出来的结构化数据就是索引

一.索引里面存在着什么

比如顺序扫描，从文件查询字符串的过程中，也即是我们已经知道所有的文件，去查询我们所需要的字符串，就是文件到字符串的一个映射过程；但是现在我们需要做的是 一直字符串去寻求文件，也就是从字符串到文件的映射过程，然而如果索引能够保存字符串到文件的映射,将会大大的提高搜索效率

由于从字符串到文件的映射是文件到字符串映射的反向过程，于是保存这种信息的索引称为**反向索引。**

当我们查询100个文档列表



lucene->文件链表1

solr->文件链表2

hadoop-> 文件链表3

lucene,solr,hadoop一系列字符串，构成一个词典

每个字符串指向包含此字符串的文档链表 ，称之为倒排表（Posting list)

当我们查询 lucene 和solr

1.查询lucene 指向的文件链表1
2.查询solr 指向的文件链表2
3.通过合并文件链表1和文件链表2,找到一个既包含lucene也包含solr的文件链表4
创建索引的过程也许会和顺序扫描相比并不一定快很多 ，但是当文件量过大 ，由于索引是一劳永逸的过程，当我们下次在进行搜索时就会大大加快速度

这也是全文搜索相对于顺序扫描的优势之一：一次索引，多次使用

二.如何创建索引

1.需要一些源文档
① Students should be allowed to go out with their friends, but not allowed to drink beer.
② My friend Jerry went to school to see his students but found them drunk which is not allowed.

2,利用分词组件去得到词元

对于一些挺词，也就是普通的词 ，大多情况 不能作为关键词，因此分词组件会进行筛选
所以，对于上面的两个源文档我们得到词元信息为
“Students” ，“allowed” ，“go” ，“their” ，“friends” ，“allowed” ，“drink” ，“beer” ，“My” ，“friend” ，“Jerry” ，“allowed”

3.将得到的词元传给语言处理组件（Linguistic Processor)
对于英语
1. 变为小写(Lowercase)。
2. 将单词缩减为词根形式，如“cars”到“car”等。这种操作称为：stemming。
3. 将单词转变为词根形式，如“drove”到“drive”等。这种操作称为：lemmatization。
将上面的词元处理后的结果称之为词

4，将词传给索引组件Indexer创建词典

1. 利用得到的词(Term)创建一个字典。

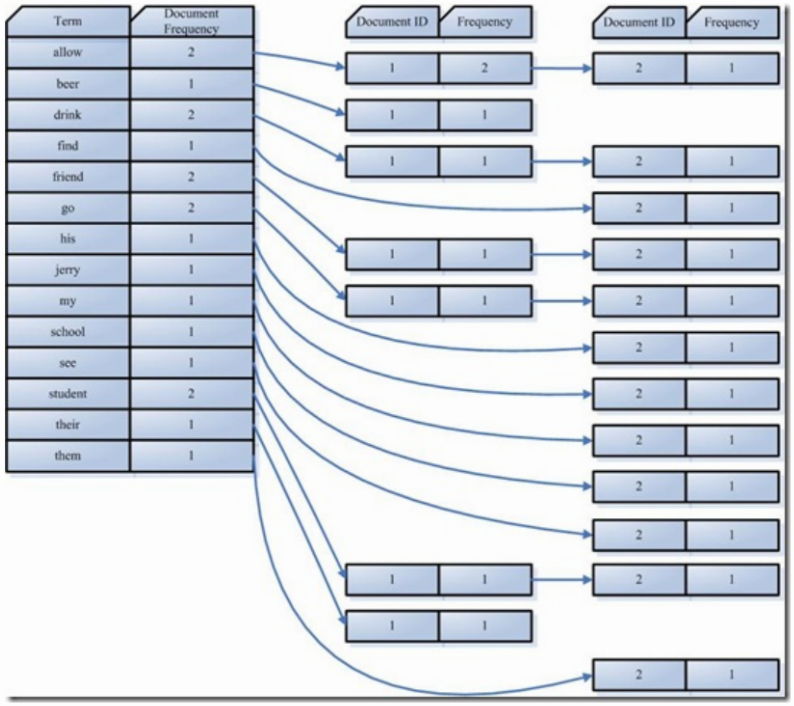
在我们的例子中字典如下：

Term	Document ID
student	1
allow	1
go	1
their	1
friend	1
allow	1
drink	1
beer	1
my	2
friend	2
jerry	2
go	2
school	2
see	2
his	2
student	2
find	2
them	2
drink	2
allow	2

2. 对字典按字母顺序进行排序。

Term	Document ID
allow	1
allow	1
allow	2
beer	1
drink	1
drink	2
find	2
friend	1
friend	2
go	1
go	2
his	2
jerry	2
my	2
school	2
see	2
student	1
student	2
their	1
them	2

3. 合并相同的词(Term)成为文档倒排(Posting List)链表。



在此表中，有几个定义：

- Document Frequency 即文档频次，表示总共有多少文件包含此词(Term)。
- Frequency 即词频率，表示此文件中包含了几个此词(Term)。

所以对词(Term) “allow”来讲，总共有两篇文档包含此词(Term)，从而词(Term)后面的文档链表总共有两项，第一项表示包含“allow”的第一篇文档，即1号文档，此文档中，“allow”出现了2次，第二项表示包含“allow”的第二个文档，是2号文档，此文档中，“allow”出现了1次。

到此为止，索引已经创建好了，我们可以通过它很快的找到我们想要的文档。

而且在此过程中，我们惊喜地发现，搜索“drive”，“driving”，“drove”，“driven”也能够被搜到。因为在我们的索引中，“driving”，“drove”，“driven”都会经过语言处理而变成“drive”，在搜索时，如果您输入“driving”，输入的查询语句同样经过我们这里的一到三步，从而变为查询“drive”，从而可以搜索到想要的文档。

三、如何对索引进行搜索？

这里我们就似乎就已经找到想要的文档了，但问题是文档查询出来的结果成百上千个，哪一个使我们最想要的结果，我们并不知道，如何将我们通过搜索最想要的结果展示在最前面是一个问题

于是，我们需要去解决这个问题

例如我们查询一个

lucene AND learned NOT hadoop。

第二步：对查询语句进行词法分析，语法分析，及语言处理。

由于查询语句有语法，因而也要进行语法分析，语法分析及语言处理。

1. 词法分析主要用来识别单词和关键字。

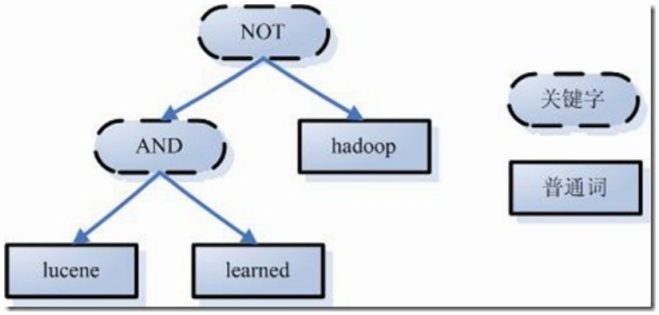
如上述例子中，经过词法分析，得到单词有lucene，learned，hadoop，关键字有AND，NOT。

如果在词法分析中发现不合法的关键字，则会出现错误。如lucene AMD learned，其中由于AND拼错，导致AMD作为一个普通的单词参与查询。

2. 语法分析主要是根据查询语句的语法规则来形成一棵语法树。

如果发现查询语句不满足语法规则，则会报错。如lucene NOT AND learned，则会出错。

如上述例子，lucene AND learned NOT hadoop形成的语法树如下：



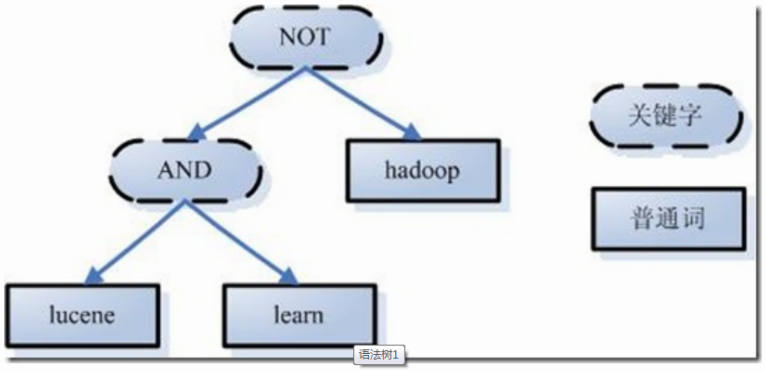
3. 语言处理同索引过程中的语言处理几乎相同。

如learned变成learn等。

经过第二步，我们得到一棵经过语言处理的语法树。

如learned变成learn等。

经过第二步，我们得到一棵经过语言处理的语法树。



第三步：搜索索引，得到符合语法树的文档。

此步骤有分几小步：

1. 首先，在反向索引表中，分别找出包含lucene，learn，hadoop的文档链表。
2. 其次，对包含lucene，learn的链表进行合并操作，得到既包含lucene又包含learn的文档链表。
3. 然后，将此链表与hadoop的文档链表进行差操作，去除包含hadoop的文档，从而得到既包含lucene又包含learn而且不包含hadoop的文档链表。
4. 此文档链表就是我们要找的文档。

四.//如何计算文档之间的相关性

我们首先需要找到文档中那些词对文档之间的相关性更重要，比如 this,what,a 等词当然是不重要的，不会影响文档之间的相关性

这个找出词对文档的重要性称之为计算词的权重

计算词的权重 ： 两个参数 一个词 ，一个文档 越重要的词权重越大

这种词到文档之间权重的计算方式称之为向量空间模型算法(Vector Space Model)

影响一个词在文档中的重要性有两个因素，1，文档中出现的次数 这个参数越大越重要

2，多少文档包含，这个从参数越小越好

解释一下：在本文档中出现的次数越多，说明这个文档就是主要说这件事情的 重要，然而，如果包含这个词的文档很多，说明这个词很普通，大家都有，不足以区别，因而重要性降低

这也如我们程序员所学的技术，对于程序员本身来说，这项技术掌握越深越好（掌握越深说明花时间看的越多，tf越大），找工作时越有竞争力。然而对于所有程序员来说，这项技术懂得的人越少越好（懂得的人少df小），找工作越有竞争力。人的价值在于不可替代性就是这个道理。

权重计算公式如下

$$w_{t,d} = tf_{t,d} \times \log(n / df_t)$$

我们把一个文档看做一系列词 看做一个向量

Document = {term1, term2, …… ,term N}

Document Vector = {weight1, weight2, …… ,weight N}

计算它的权重值

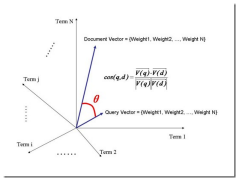
同样把查询的语句当做一个文档 同样适用向量表示

Query = {term1, term 2, …… , term N}

Query Vector = {weight1, weight2, …… , weight N}

我们将结果放到一个N维空间

如图:



们认为两个向量之间的夹角越小，相关性越大。

所以我们计算夹角的余弦值作为相关性的打分，夹角越小，余弦值越大，打分越高，相关性越大

维度不同时，取出并集

相关性计算公式：

$$score(q,d) = \frac{\vec{V}_q \cdot \vec{V}_d}{|\vec{V}_q||\vec{V}_d|} = \frac{\sum_{i=1}^{n} w_{i,q} w_{i,d}}{\sqrt{\sum_{i=1}^{n} w_{i,q}^2} \sqrt{\sum_{i=1}^{n} w_{i,d}^2}}$$

1. 索引过程：

1）有一系列被索引文件

2）被索引文件经过语法分析和语言处理形成一系列的词（Term）

3）经过索引创建形成词典和反向索引表

4） 通过索引存储将索引写入磁盘

2：搜索过程:

A.用户输入查询语句

B.对查询语句经过语法分析和语言分析得到一系列的词(Term)

C.通过语法分析得到一个查询树

D.通过索引存储将索引存入内存

E.通过查询树搜索索引，得到文档链表，经过交差并得到结果文档

F.将得到结果进行相关性排序

G:返回给用户结果

why:我们为什么使用它

首先咱们了解一下数据库的搜索场景

它可以快速地储存、搜索和分析海量数据。维基百科、Stack Overflow、Github 都采用它。

1、比方说，每条记录的指定字段的文本，可能会很长，比如说“商品描述”字段的长度，有长达数千个，甚至数万个字符，这个时候，每次都要对每条记录的所有文本进行扫描，来判断说，你包不包含我指定的这个关键词（比如说“牙膏”）

2、还不能将搜索词拆分开来，尽可能去搜索更多的符合你的期望的结果，比如输入“生化机”，就搜索不出来“生化危机”

再来看看咱们的elasticsearch

具体说说elasticsearch的好处

① 分词搜索，全文检索，不会在出现数据库的对文本一个一个扫描，它直接就通过节点的索引进行搜

② 自己维护冗余备份数据，某些机器宕机了，不会丢失数据
③ 当然还有强大搜索功能，聚合分析的功能
④ 结合kibana 还能进行建立图表，给各位展示科学化的业务数据,更好的分析业务
⑤ 免费的不要钱而且听说永久免费!
⑥提供了java api接口
⑦ 提供了Restful api接口 大家都可以访问
⑧请求和应答全是json的数据格式

where:elasticsearch使用场景

国外
（1）维基百科，类似百度百科，牙膏，牙膏的维基百科，全文检索，高亮，搜索推荐
（2）The Guardian（国外新闻网站），类似搜狐新闻，用户行为日志（点击，浏览，收藏，评论）+社交网络数据（对某某新闻的相关看法），数据分析，给到每篇新闻文章的作者，让他知道他的文章的公众反馈
（3）Stack Overflow（国外的程序异常讨论论坛），IT问题，程序的报错，提交上去，有人会跟你讨论和回答，全文检索，搜索相关问题和答案，程序报错了，就会将报错信息粘贴到里面去，搜索有没有对应的答案
（4）GitHub（开源代码管理），搜索上千亿行代码
（5）电商网站，检索商品
（6）日志数据分析，logstash采集日志，ES进行复杂的数据分析（ELK技术，elasticsearch+logstash+kibana）
（7）商品价格监控网站，用户设定某商品的价格阈值，当低于该阈值的时候，发送通知消息给用户，比如说订阅牙膏的监控，如果高露洁牙膏的家庭套装低于50块钱，就通知我，我就去买
（8）BI系统，商业智能，Business Intelligence。比如说有个大型商场集团，BI，分析一下某某区域最近3年的用户消费金额的趋势以及用户群体的组成构成，产出相关的数张报表，**区，最近3年，每年消费金额呈现100%的增长，而且用户群体85%是高级白领，开一个新商场。ES执行数据分析和挖掘，Kibana进行数据可视化
国内
（9）国内：
站内搜索（电商，招聘，门户，等等），IT系统搜索（OA，CRM，ERP，等等），数据分析（ES热门的一个使用场景）

Elasticsearch的特点
（1）可以作为一个大型分布式集群（数百台服务器）技术，处理PB级数据，服务大公司；也可以运行在单机上，服务小公司
（2）Elasticsearch不是什么新技术，主要是将全文检索、数据分析以及分布式技术，合并在了一起，才形成了独一无二的ES；lucene（全文检索），商用的数据分析软件（也是有的），分布式数据库（mycat）
（3）对用户而言，是开箱即用的，非常简单，作为中小型的应用，直接3分钟部署一下ES，就可以作为生产环境的系统来使用了，数据量不大，操作不是太复杂
（4）数据库的功能面对很多领域是不够用的（事务，还有各种联机事务型的操作）；特殊的功能，比如全文检索，同义词处理，相关度排名，复杂数据分析，海量数据的近实时处理；Elasticsearch作为传统数据库的一个补充，提供了数据库所不能提供的很多功能

when:什么时候使用它

对于搜索比较频繁的应用，我个人觉得对于有很多很多的数据的时候就可以用它，比如资源搜索
然后还可以为咱们做日志分析

How

一.搭建环境

环境搭建地址

注意:数据库jdk版本必须是1.8以上

http://note.youdao.com/noteshare?id=f0e48260fc7f789442ad20c8241c4728&sub=61AB58C4662E466097E6A0B88C46C182

安装执行，在网页上浏览 es:9200

出现如下数据说明正确


```
{
  "name" : "U82GWHd",
  "cluster_name" : "elasticsearch",
  "cluster_uuid" : "OzmrQH7qQbShom2659NTFQ",
  "version" : {
    "number" : "6.0.0",
    "build_hash" : "23ffa4f",
    "build_date" : "2017-08-29T01:53:03.739Z",
    "build_snapshot" : false,
    "lucene_version" : "7.0.0",
    "minimum_wire_compatibility_version" : "5.6.0",
    "minimum_index_compatibility_version" : "5.0.0"
  },
  "tagline" : "You Know, for Search"
}
```


然后下载kibana logstash 具体的详细配置请看下面的地址文档

二.文件配置

上面的网址的配置流程走完以后请详细看一下我的配置

sqlserver.conf配置文件

```
input {
    jdbc {
        jdbc_driver_library => "D:/soft/elasticsearch/logstash-6.0.0-beta2/bin/sqlserverdriver/mssql-jdbc-6.2.1.jre8.jar"
        jdbc_driver_class => "com.microsoft.sqlserver.jdbc.SQLServerDriver"
        jdbc_connection_string =>          "jdbc:sqlserver://10.0.0.130:1433;databaseName=Webi_WeLiveDBTest;"
   jdbc_user => "speakhi_user"
        jdbc_password => "speakhi_user123"
        schedule => "* * * * *"
        jdbc_default_timezone => "Asia/Shanghai"
        statement => "SELECT * FROM Lesson_Homework where CreateTime < GETDATE()"
    }
}
output {
        stdout { codec => rubydebug }
    elasticsearch {
        index => "sselkdb"
        document_type => "sselktable"
        document_id => "%{id}"
        hosts => ["127.0.0.1:9200"]
    }
}
```

sqlserver.conf的部分解释

schedule => "* * * * *"代表每分钟都在执行同步数据库数据

statement => "SELECT * FROM Lesson_Homework where CreateTime < GETDATE()"

代表查询当前的最新数据（注:不支持删除操作，对于数据的数据被物理删除的数据，elasticsearch是无法再查询的时候进行自己删除，蛋因为现在都是逻辑删除,elasticsearch是可以查询出来的）

index => "sselkdb" 索引名称

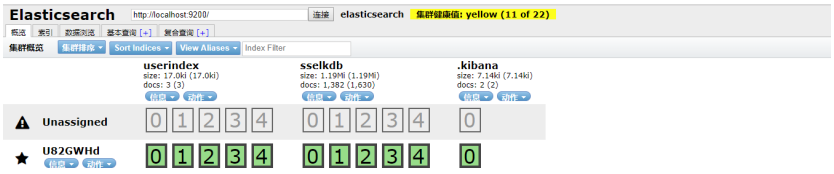
document_type => "sselktable"类型

document_id => "%{id}"对应的 数据库的id

下载一个google插件**Elasticsearch head**

这个插件主要是让我们更好的查询看到数据来自强大的google

在此插件中输入http://locahost:9200配置



elasticsearch -head的状态介绍

green：每个索引的primary shard和replica shard都是active状态的

yellow：每个索引的primary shard都是active状态的，但是部分replica shard不是active状态，处于不可用的状态

red：不是所有索引的primary shard都是active状态的，部分索引有数据丢失了

这些都弄好之后

三.代码开发

开启springboot项目中的代码编写

首先我这里选择的是jestClient操作elasticsearch这里还有一种方式是通过

ElasticsearchRepostiry类似jpa的一种工具接口，但会随着ela的版本的修改而变化代码，索引首选jestClient

ok！第一步先导入依赖

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-elasticsearch</artifactId>
    <version>1.5.4.RELEASE</version>
</dependency>
<dependency>
    <groupId>io.searchbox</groupId>
    <artifactId>jest</artifactId>
</dependency>
<dependency>
    <groupId>net.java.dev.jna</groupId>
    <artifactId>jna</artifactId>
</dependency>
```

这里需要注意 ①

springboot对应的elasticsearch的版本

这里sprigboot是1.5.4，ela依赖也是1.5.4

版本对应参照请看下面

Spring Boot Version (x)	Spring Data Elasticsearch Version (y)	Elasticsearch Version (z)
x <= 1.3.5	y <= 1.3.4	z <= 1.7.2*
x >= 1.4.x	2.0.0 <=y < 5.0.0**	2.0.0 <= z < 5.0.0**

然后我引入jest

Jest是一个为ES操作Java Http Rest 的客户端，及使用它来捕获、存储以及操作数据

Jest提供JavaAPI，可以很好的处理Java对象序列

此处必须要引入jna ，不然会报错的

Java Native Access

JNA提供一组Java工具类用于在运行期动态访问系统本地库

东西弄好之后我们开始写代码

第一步在application.properties中配置ela 服务地址 es:9200

第二步在实体上建立索引以及类型

```
@Document(indexName = "sselkdb", type = "sselktable")
```

```
org.springframework.data.elasticsearch.annotations.Document;
```

第三步拿出jestClient对象，用该对象操作elasticsearch拿出数据

这里重点使用的是ela的全文检索的方法

这里因为要过滤数据，所以我拼接查询的条件(查询的方法会单独出一个文档供大家参考)

如下:

```
String queryStr = " {\n "
        + " \"query\":{\n"
        + " \"query_string\":{\n"
        + " \"query\":\"" + query + "\"\n"
        + "}\n"
        + " },\n"
        + "\"post_filter\":{\n"
        + "\"term\" :{\n"
        + "\"isdeleted\" :\"false\"\n"
        + "}\n"
        + "}\n"
        + "}\n";
```

拼接以后执行，根据官网的文档来的

https://www.elastic.co/blog/found-java-clients-for-elasticsearch

具体的拼接数据方式可以现在kibana中拼出查询语句，然后在使用到代码中

获取数据的方式解析方式有两种

https://blog.csdn.net/u011781521/article/details/77853824

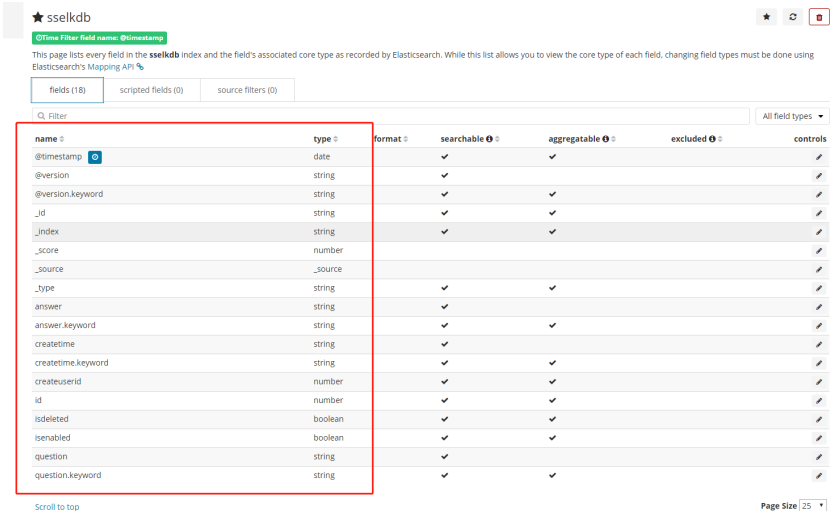
ElasticSearch已经使用了jackson，可以直接使用它把javabean转为json.

我这里使用了手动解析的方式，因为发现自动解析不好用

注:因为数据在导入elasticsearch的时候类型会是通过jackJson序列化过去的

我们在解析数据的时候需要保持参数类型是从elasticsearch过来的

恩，这里直接查看kinaba的数据 mapping映射



他的参数以及类型已经存在，我们需要解析出来，手动的解析方式如下

```
LessonHomeworkParam lessonHomeworkParam = new LessonHomeworkParam();
lessonHomeworkParam.setId(Integer.parseInt(sourceObject.get("id").getAsNumber().toString()));
lessonHomeworkParam.setAnswer(sourceObject.get("answer").getAsString());
lessonHomeworkParam.setQuestion(sourceObject.get("question").getAsString());
lessonHomeworkParam.setCreateTime(sourceObject.get("createtime").getAsString());
lessonHomeworkParam.setCreateUserId(Integer.parseInt(sourceObject.get("createuserid").getAsNumber().toString()));
lessonHomeworkParam.setDeleted(sourceObject.get("isdeleted").getAsBoolean());
lessonHomeworkParam.setEnabled(sourceObject.get("isenabled").getAsBoolean());
lessonHomeworkParams.add(lessonHomew
```

Typo: In word 'isenabled' more... (Ctrl+F1)

ok,大功告成，已经可以查询出数据了，这里只是使用了全文检索，过滤，还有很多查询方式有时间再去学习

核心代码附件

LessonHomework... e.java
8.47KB

四.后续

kibana上查询出来的数据


```
{
  "took": 40,
  "timed_out": false,
  "_shards": {
    "total": 27,
    "successful": 27,
    "skipped": 0,
    "failed": 0
  },
  "hits": {
    "total": 1529,
    "max_score": 5.710427,
    "hits": [
      {
        "_index": "catalog",
```

```
      "_type": "catalogtable",
      "_id": "2406",
      "_score": 5.710427,
      "_source": {
        "@timestamp": "2018-06-25T01:23:00.031Z",
        "sequence": 8,
        "id": 2406,
        "isdeleted": false,
        "parentid": 2197,
        "createuserid": 10,
        "name": "2",
        "@version": "1",
        "type": "lesson_catalog",
        "createtime": "2018-06-23T08:20:35.183Z"
      }
    }
```


took字段表示该操作的耗时（单位为毫秒），timed_out字段表示是否超时，hits字段表示命中的记录　total：返回记录数。max_score：最高的匹配程度

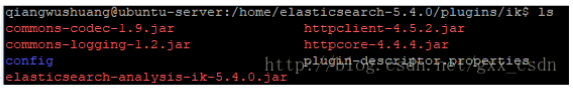
hits：返回的记录组成的数组。

这里的都是我在完成基本的全文检索功能之后添加上去，有兴趣可以看一下

添加ik分词器

在es目录下的`plugins`目录下创建一个新文件夹，命名为`ik`，然后把上面的压缩包中的内容解压到该目录中。

比如在Ubuntu中，把解压出来的内容放到es/plugins/ik中：



kinaba的详细使用：https://www.cnblogs.com/zhangs1986/p/7325504.html

ik版本的下载地址 https://github.com/medcl/elasticsearch-analysis-ik/releases

下载IK分词器注意对应的es的版本

ik分词器的安装以及介绍使用 https://blog.csdn.net/gxx_csdn/article/details/79110384

elasticsearch 试用终极详细介绍 http://www.sojson.com/blog/90.html 它里面的东西写的很专业

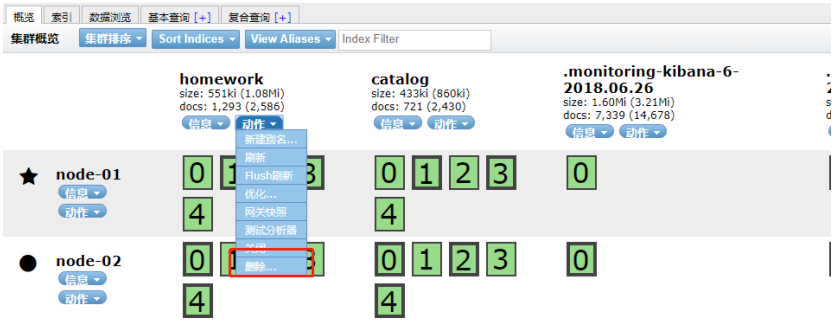
monitoring-kibana kibana 监控 这个是我在linux上导入数据时会产生的监控文件

https://www.elastic.co/guide/en/kibana/current/xpack-monitoring.html

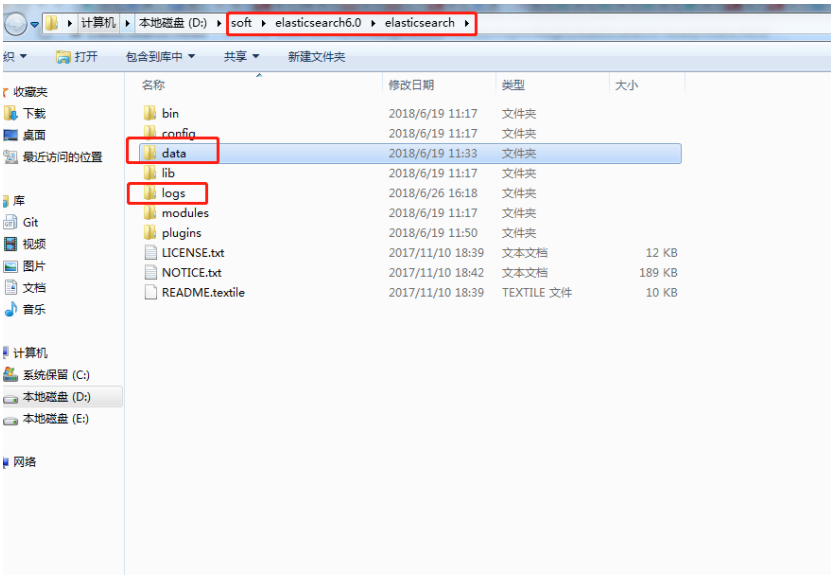
五.清除elasticsearch数据

①windows版

首先需要在elasticsearch上将索引删除

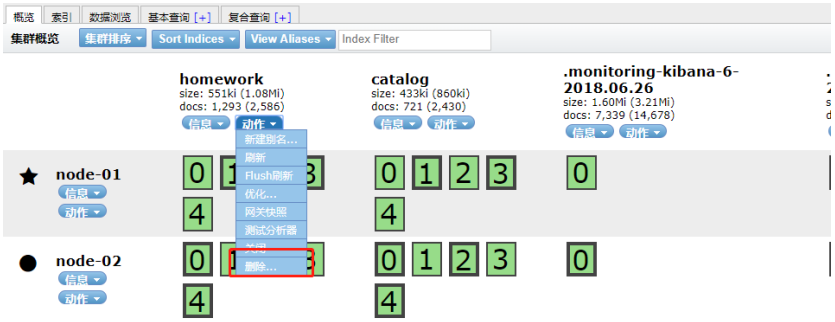


然后进入elasticsearch文件目录下面将这两个文件删除



②linux版本

在elasticsearch-head将索引删除,然后在配置的文件保存路径中删除文件(建议删除索引即可,当然在安装的配置文件中也可以删除数据)



附:开发过程中遇到的坑

第一个就是使用什么办法解析，刚开始找了很多种方式，最后在官网找到了使用jestClient的方法

query拼接起来去执行

第二个就是解析查询出来的数据的时候很坑，本来想自动解析，但是类型转换不过来，因为我这里的实体类使用的java的语法的类型，但是elasticsearch有它自己的类型，所以最后选择了手动解析数据查出来这些数据

第三个坑就是刚开始的时候引入的依赖一定要注意点，经常会出现版本不对，而且我上面提出的三个依赖全部引入，springboot才能正确执行没有bug

查看对应版本问题

https://www.cnblogs.com/xuwujing/p/8998168.html

@Document 什么意思

Spring-data-elasticsearch为我们提供了@Document、@Field等注解，如果某个实体需要建立索引，只需要加上这些注解即可

9300是TCP通讯端口，集群间和的TcpClient都走的它，9200是HTTP协议的RESTful的接口

elasticsearch

索引文件的问题：

配置文件的问题：