**是什么？**

Elasticsearch是一个基于[Apache Lucene(TM)](https://lucene.apache.org/core/)的开源搜索引擎。无论在开源还是专有领域，Lucene可以被认为是迄今为止最先进、性能最好的、功能最全的搜索引擎库。

Elasticsearch不仅仅是Lucene和全文搜索，我们还能这样去描述它：

* 分布式的实时文件存储，每个字段都被索引并可被搜索
* 分布式的实时分析搜索引擎
* 可以扩展到上百台服务器，处理PB级结构化或非结构化数据

**面向文档**

应用中的对象很少只是简单的键值列表，更多时候它拥有复杂的数据结构，比如包含日期、地理位置、另一个对象或者数组。

总有一天你会想到把这些对象存储到数据库中。将这些数据保存到由行和列组成的关系数据库中，就好像是把一个丰富，信息表现力强的对象拆散了放入一个非常大的表格中：你不得不拆散对象以适应表模式（通常一列表示一个字段），然后又不得不在查询的时候重建它们。

Elasticsearch是**面向文档(document oriented)**的，这意味着它可以存储整个对象或**文档(document)**。然而它不仅仅是存储，还会**索引(index)**每个文档的内容使之可以被搜索。在Elasticsearch中，你可以对文档（而非成行成列的数据）进行索引、搜索、排序、过滤。这种理解数据的方式与以往完全不同，这也是Elasticsearch能够执行复杂的全文搜索的原因之一。

使用案例：

* 维基百科使用Elasticsearch来进行全文搜做并高亮显示关键词，以及提供search-as-you-type、did-you-mean等搜索建议功能。
* 英国卫报使用Elasticsearch来处理访客日志，以便能将公众对不同文章的反应实时地反馈给各位编辑。
* StackOverflow将全文搜索与地理位置和相关信息进行结合，以提供more-like-this相关问题的展现。
* GitHub使用Elasticsearch来检索超过1300亿行代码。
* 每天，Goldman Sachs使用它来处理5TB数据的索引，还有很多投行使用它来分析股票市场的变动。

但是Elasticsearch并不只是面向大型企业的，它还帮助了很多类似DataDog以及Klout的创业公司进行了功能的扩展。

Elasticsearch 与 Solr 的比较总结

* 二者安装都很简单；
* Solr 利用 Zookeeper 进行分布式管理，而 Elasticsearch 自身带有分布式协调管理功能;
* Solr 支持更多格式的数据，而 Elasticsearch 仅支持json文件格式；
* Solr 官方提供的功能更多，而 Elasticsearch 本身更注重于核心功能，高级功能多有第三方插件提供；
* Solr 在传统的搜索应用中表现好于 Elasticsearch，但在处理实时搜索应用时效率明显低于 Elasticsearch。

Solr 是传统搜索应用的有力解决方案，但 Elasticsearch 更适用于新兴的实时搜索应用。

**index、type**

在Elasticsearch中，文档归属于一种**类型(type)**,而这些类型存在于**索引(index)**中，我们可以画一些简单的对比图来类比传统关系型数据库：

Relational DB -> Databases -> Tables -> Rows -> Columns

Elasticsearch -> Indices -> Types -> Documents -> Fields

Elasticsearch集群可以包含多个**索引(indices)**（数据库），每一个索引可以包含多个**类型(types)**（表），每一个类型包含多个**文档(documents)**（行），然后每个文档包含多个**字段(Fields)**（列）。

**什么是mapping**

ES的mapping非常类似于静态语言中的数据类型：声明一个变量为int类型的变量， 以后这个变量都只能存储int类型的数据。同样的， 一个number类型的mapping字段只能存储number类型的数据。

同语言的数据类型相比，mapping还有一些其他的含义，mapping不仅告诉ES一个field中是什么类型的值， 它还告诉ES如何索引数据以及数据是否能被搜索到。

当你的查询没有返回相应的数据， 你的mapping很有可能有问题。当你拿不准的时候， 直接检查你的mapping。



**剖析mapping**

一个mapping由一个或多个analyzer组成， 一个analyzer又由一个或多个filter组成的。当ES索引文档的时候，它把字段中的内容传递给相应的analyzer，analyzer再传递给各自的filters。

filter的功能很容易理解：一个filter就是一个转换数据的方法， 输入一个字符串，这个方法返回另一个字符串，比如一个将字符串转为小写的方法就是一个filter很好的例子。

一个analyzer由一组顺序排列的filter组成，执行分析的过程就是按顺序一个filter一个filter依次调用， ES存储和索引最后得到的结果。

总结来说， mapping的作用就是执行一系列的指令将输入的数据转成可搜索的索引项。

**默认analyzer**

回到我们的例子， ES猜测description字段是string类型，于是默认创建一个string类型的mapping，它使用默认的全局analyzer， 默认的analyzer是[标准analyzer](http://www.elasticsearch.org/guide/reference/index-modules/analysis/standard-analyzer.html)。

我们可以在做查询的时候键入\_analyze关键字查看分析的过程。使用以下指令查看description字段的转换过程：



可以看到， 我们的description字段的值转换成了[pretty], [cool], [guy]， 在转换过程中大写的A， 标点符号都被filter过滤掉了， Pretty也转成了全小写的pretty， 这里比较重要的是， 即使ES存储数据的时候仍然存储的是完整的数据， 但是可以搜索到这条数据的关键字只剩下这三个单词了， 其他的都是抛弃掉了。



现在就能得到正确的结果，这是一个公认的简单例子， 但是它描述了ES是如何工作的， 不要把mapping想成是数据类型， 把它想象成是搜索数据的指令集合。如果你不想字符"a"被删除， 你需要修改你的analyzer。

mapping配置

可以修改的项：

* 增加新的类型定义
* 增加新的字段
* **增加新的分析器**

不允许修改的项：

* **更改字段类型(比如文本改为数字)**
* **更改存储为不存储，反之亦然**
* **更改索引属性的值**
* **更改已索引文档的分析器**

**注意的是新增字段或更改分析器之后，需要再次对所有文档进行索引重建**

字段的数据类型

简单类型

* string**(指定分词器)**
* date(默认使用UTC保持,也可以使用format指定格式)
* 数值类型(byte,short,integer,long,float,double)
* boolean
* binary(存储在索引中的二进制数据的base64表示，比如图像，只存储不索引)
* ip(以数字形式简化IPV4地址的使用，可以被索引、排序并使用IP值做范围查询).

有层级结构的类型

比如object 或者 nested.

特殊类型

比如geo\_point, geo\_shape, or completion.

**动态模板：**

使用dynamic\_templates可以完全控制新字段的映射，你设置可以通过字段名或数据类型应用一个完全不同的映射。

例子：我们为/my\_index/my\_type 分别创建

es:字段名以\_es结尾的且是string类型的，需要使用spanish分词器

enn:其他字段的且是string类型的，需要使用english分词器

PUT /**my\_index**

{

"mappings": {

"my\_type": {

"dynamic\_templates": [

{

"es": {

=>模板名称，随意，一般要有语义"match": "\*\_es",

=>匹配字段名称"match\_mapping\_type": "string"=>匹配字段类型"mapping": {

=>当匹配到之后，该字段的具体设置"type": "string",

"anaylzer": "spanish"

}

}

},

{

"en": {

=>模板名称，随意，一般要有语义"match": "\*",

=>匹配字段名称(任意，通用的顺序要在之后)"match\_mapping\_type": "string"=>匹配字段类型"mapping": {

=>当匹配到之后，该字段的具体设置"type": "string",

"anaylzer": "english"

}

}

}

]

}

}

}

**index别名设置**

一个别名能够指向多个索引，因此当我们将别名指向新的索引时，我们还需要删除别名原来到旧索引的指向。这个改变需要是原子的，即意味着我们需要使用\_aliases端点：

POST /\_aliases { "actions": [ { "remove": { "index": "my\_index\_v1", "alias": "my\_index" }}, { "add": { "index": "my\_index\_v2", "alias": "my\_index" }} ] }

现在你的应用就在**零停机**时间的前提下，实现了旧索引到新索引的透明切换。

问题：

1、全文索引（json全文爬数据）

2、聚合无法根据匹配度排序

3、聚合无法真分页

ES后台管理 ： <http://checklist-formtastic-admin.debug.web.nd/>

{

"dynamic": "false",

"dynamic\_templates": [

{

"indexes": {

"mapping": {

"type": "string",

"fields": {

"raw": {

"index": "not\_analyzed",

"null\_value": "",

"type": "string"

}

}

},

"match\_mapping\_type": "string",

"path\_match": "indexes.\*"

}

}

],

"properties": {

"indexes": {

"dynamic": "strict",

"properties": {

"application": {

"type": "string",

"index": "not\_analyzed"

},

"attribute\_name": {

"type": "string",

"fields": {

"raw": {

"type": "string",

"index": "not\_analyzed",

"null\_value": ""

}

}

},

"category": {

"type": "string",

"index": "not\_analyzed"

},

"create\_time": {

"type": "long"

},

"data\_md5": {

"type": "string",

"index": "not\_analyzed"

},

"disable": {

"type": "boolean"

},

"keyword": {

"type": "string",

"fields": {

"raw": {

"type": "string",

"index": "not\_analyzed",

"null\_value": ""

}

}

},

"project\_code": {

"type": "string",

"fields": {

"raw": {

"type": "string",

"index": "not\_analyzed",

"null\_value": ""

}

}

},

"project\_id": {

"type": "string",

"index": "not\_analyzed"

},

"project\_title": {

"type": "string",

"fields": {

"raw": {

"type": "string",

"index": "not\_analyzed",

"null\_value": ""

}

}

},

"project\_type": {

"type": "string",

"index": "not\_analyzed",

"fields": {

"raw": {

"type": "string",

"index": "not\_analyzed",

"null\_value": ""

}

}

},

"reference\_count": {

"type": "long"

},

"tenant": {

"type": "long"

},

"top\_reference\_count": {

"type": "long"

},

"type": {

"type": "long"

},

"update\_time": {

"type": "long"

},

"user\_id": {

"type": "long"

},

"user\_name": {

"type": "string",

"fields": {

"raw": {

"type": "string",

"index": "not\_analyzed",

"null\_value": ""

}

}

}

}

},

"tenant": {

"type": "long"

}

}

}