映射和分析

当 弄索引里面的数据 ，我 一些奇怪的事情。一些事情看起来被打乱了：在我 的索引中有12条推文，其中只有一条包含日期 2014-09-15，但是看一看下面 命中的 数（total）：

```
GET /_search?q=2014              # 12 results
GET /_search?q=2014-09-15        # 12 results !
GET /_search?q=date:2014-09-15   # 1  result
GET /_search?q=date:2014         # 0  results !
```

什 在 _all 字段 日期返回所有推文，而在 date 字段只 年 却没有返回 果？ 什 我 在 _all 字段和 date 字段的 果有差 ？

推 起来， 是因 数据在 all 字段与 date 字段的索引方式不同。所以，通 求 gb 索引中 tweet 型的_映射（或模式定 ）， 我 看一看 Elasticsearch 是如何解 我 文 的：

```
GET /gb/_mapping/tweet
```

将得到如下 果：

```
{
   "gb": {
      "mappings": {
         "tweet": {
            "properties": {
               "date": {
                  "type": "date",
                  "format": "strict_date_optional_time||epoch_millis"
               },
               "name": {
                  "type": "string"
               },
               "tweet": {
                  "type": "string"
               },
               "user_id": {
                  "type": "long"
               }
            }
         }
      }
   }
}
```

基于 字段 型的 ， Elasticsearch 我 生了一个映射。 个 告 我 date 字段被 是 date 型的。由于 _all 是 字段，所以没有提及它。但是我 知道 _all 字段是 string 型的。

所以 date 字段和 string 字段索引方式不同，因此搜索 果也不一 。 完全不令人吃 。 可能会 核心数据 型 strings、numbers、Booleans 和 dates 的索引方式有 不同。没 ，他 有不同。

但是，到目前 止，最大的差 在于代表 精 （它包括 string 字段）的字段和代表 全文 的字段。 个区 非常重要——它将搜索引 和所有其他数据 区 来。

精 VS 全文

Elasticsearch 中的数据可以概括的分 ：精 和全文。

精 如它 听起来那 精 。例如日期或者用 ID，但字符串也可以表示精 ，例如用 名或 箱地址。 于精 来 ，Foo 和 foo 是不同的，2014 和 2014-09-15 也是不同的。

一方面，全文 是指文本数据（通常以人 容易 的 言 写），例如一个推文的内容或一封 件的内容。

NOTE

全文通常是指非 化的数据，但 里有一个 解：自然 言是高度 化的。 在于自然 言的 是 的， 致 算机 以正 解析。例如，考 条 句：

```
May is fun but June bores me.
```

它指的是月 是人？

精 很容易 。 果是二 制的：要 匹配 ，要 不匹配。 很容易用 SQL 表示：

```
WHERE name    = "John Smith"
  AND user_id = 2
  AND date    > "2014-09-15"
```

全文数据要微妙的多。我 的不只是“ 个文 匹配 ”，而是“ 文 匹配 的程度有多大？”句 ， 文 与 定 的相 性如何？

我 很少 全文 型的域做精 匹配。相反，我 希望在文本 型的域中搜索。不 如此，我 希望搜索能 理解我 的意 ：

- 搜索 UK ，会返回包含 United Kindom 的文 。
- 搜索 jump ，会匹配 jumped ， jumps ， jumping ，甚至是 leap 。
- 搜索 johnny walker 会匹配 Johnnie Walker ， johnnie depp 匹配 Johnny Depp 。
- fox news hunting 返回福克斯新 （ Foxs News ）中 于狩 的故事，同 ， fox hunting news 返回 于 狐的故事。

了促 在全文域中的 ，Elasticsearch 首先 分析 文 ，之后根据 果 建 倒排索引 。在接下来的 ，我 会 倒排索引和分析 程。

倒排索引

Elasticsearch 使用一 称 倒排索引 的 ，它 用于快速的全文搜索。一个倒排索引由文中所有不重 的列表 成， 于其中 个 ，有一个包含它的文 列表。

例如，假 我 有 个文 ， 个文 的 `content` 域包含如下内容：

1. The quick brown fox jumped over the lazy dog
2. Quick brown foxes leap over lazy dogs in summer

了 建倒排索引，我 首先将 个文 的 `content` 域拆分成 独的 （我 称它 条 或 `tokens` ），建一个包含所有不重 条的排序列表，然后列出 个 条出 在 个文 。 果如下所示：

```
Term      Doc_1  Doc_2
-------------------------
Quick   |       |  X
The     |   X   |
brown   |   X   |  X
dog     |   X   |
dogs    |       |  X
fox     |   X   |
foxes   |       |  X
in      |       |  X
jumped  |   X   |
lazy    |   X   |  X
leap    |       |  X
over    |   X   |  X
quick   |   X   |
summer  |       |  X
the     |   X   |
------------------------
```

在，如果我 想搜索 `quick brown` ，我 只需要 包含 个 条的文 ：

```
Term      Doc_1  Doc_2
-------------------------
brown   |   X   |  X
quick   |   X   |
------------------------
Total   |   2   |  1
```

个文 都匹配，但是第一个文 比第二个匹配度更高。如果我 使用 算匹配 条数量的相似性算法，那 ，我 可以 ， 于我 的相 性来 ，第一个文 比第二个文 更佳。

但是，我 目前的倒排索引有一些 ：

- `Quick` 和 `quick` 以独立的 条出 ，然而用 可能 它 是相同的 。
- `fox` 和 `foxes` 非常相似，就像 `dog` 和 `dogs` ；他 有相同的 根。

- jumped 和 leap, 尽管没有相同的 根，但他 的意思很相近。他 是同 。

使用前面的索引搜索 +Quick +fox 不会得到任何匹配文 。（ 住，+ 前 表明 个 必存在。）只有同 出 Quick 和 fox 的文 才 足 个 条件，但是第一个文 包含 quick fox，第二个文 包含 Quick foxes 。

我 的用 可以合理的期望 个文 与 匹配。我 可以做的更好。

如果我 将 条 准模式，那 我 可以 到与用 搜索的 条不完全一致，但具有足 相 性的文 。例如：

- Quick 可以小写化 quick 。
- foxes 可以 干提取 -- 根的格式-- fox 。 似的， dogs 可以 提取 dog 。
- jumped 和 leap 是同 ，可以索引 相同的 jump 。

在索引看上去像 ：

```
Term      Doc_1  Doc_2
-------------------------
brown   |   X   |  X
dog     |   X   |  X
fox     |   X   |  X
in      |       |  X
jump    |   X   |  X
lazy    |   X   |  X
over    |   X   |  X
quick   |   X   |  X
summer  |       |  X
the     |   X   |  X
-------------------------
```

不 。我 搜索 +Quick +fox 然 会失 ，因 在我 的索引中，已 没有 Quick 了。但是，如果我 搜索的字符串使用与 content 域相同的 准化 ，会 成 +quick +fox， 个文 都会匹配！

NOTE	非常重要。 只能搜索在索引中出 的 条，所以索引文本和 字符串必 准化 相同的格式。

分 和 准化的 程称 分析， 我 会在下个章 。

分析与分析器

分析 包含下面的 程：

- 首先，将一 文本分成 合于倒排索引的独立的 条，
- 之后，将 些 条 一化 准格式以提高它 的“可搜索性”，或者 *recall*

分析器 行上面的工作。 分析器 上是将三个功能封装到了一个包里：

字符 器

首先，字符串按 序通 个 字符 器 。他 的任 是在分 前整理字符串。一个字符 器可以用来去掉HTML，或者将 & 化成 and。

分 器

其次，字符串被 分 器 分 个的 条。一个 的分 器遇到空格和 点的 候，可能会将文本拆分成 条。

Token 器

最后， 条按 序通 个 *token* 器 。 个 程可能会改 条（例如，小写化 Quick ）， 除 条（例如， 像 a， and， the 等无用 ），或者 加 条（例如，像 jump 和 leap 同 ）。

Elasticsearch提供了 箱即用的字符 器、分 器和token 器。 些可以 合起来形成自定 的分析器以用于不同的目的。我 会在 [custom-analyzers] 章 。

内置分析器

但是， Elasticsearch 附 了可以直接使用的 包装的分析器。接下来我 会列出最重要的分析器。了 明它 的差 ，我 看看 个分析器会从下面的字符串得到 些 条：

```
"Set the shape to semi-transparent by calling set_trans(5)"
```

准分析器

准分析器是Elasticsearch 使用的分析器。它是分析各 言文本最常用的 。它根据 Unicode 盟定 的 界 分文本。 除 大部分 点。最后，将 条小写。它会 生

```
set, the, shape, to, semi, transparent, by, calling, set_trans, 5
```

分析器

分析器在任何不是字母的地方分隔文本，将 条小写。它会 生

```
set, the, shape, to, semi, transparent, by, calling, set, trans
```

空格分析器

空格分析器在空格的地方 分文本。它会 生

```
Set, the, shape, to, semi-transparent, by, calling, set_trans(5)
```

言分析器

特定 言分析器可用于 {ref}/analysis-lang-analyzer.html[很多 言]。它 可以考 指定 言的特点。例如， 英 分析器附 了一 英 无用 （常用 ，例如 and 或者 the ，它 相性没有多少影 ），它 会被 除。 由于理解英 法的 ， 个分 器可以提取英 的 干 。

英 分 器会 生下面的 条：

```
set, shape, semi, transpar, call, set_tran, 5
```

注意看 transparent、 calling 和 set_trans 已 根格式。

什 候使用分析器

当我 索引 一个文 ，它的全文域被分析成 条以用来 建倒排索引。 但是，当我 在全文域 搜索 的候，我 需要将 字符串通 相同的分析 程 ，以保 我 搜索的 条格式与索引中的条格式一致。

全文 ，理解 个域是如何定 的，因此它 可以做正 的事：

- 当 一个 全文 域 ， 会 字符串 用相同的分析器，以 生正 的搜索 条列表。
- 当 一个 精 域 ，不会分析 字符串，而是搜索 指定的精 。

在 可以理解在 始章 的 什 返回那 的 果：

- date 域包含一个精 ： 独的 条 2014-09-15。
- _all 域是一个全文域，所以分 程将日期 化 三个 条： 2014， 09， 和 15。

当我 在 _all 域 2014，它匹配所有的12条推文，因 它 都含有 2014 ：

```
GET /_search?q=2014              # 12 results
```

当我 在 all 域 2014-09-15，它首先分析 字符串， 生匹配 2014， 09， 或 15 中 _任意条的 。 也会匹配所有12条推文，因 它 都含有 2014 ：

```
GET /_search?q=2014-09-15        # 12 results !
```

当我 在 date 域 2014-09-15，它 精 日期，只 到一个推文：

```
GET /_search?q=date:2014-09-15   # 1  result
```

当我 在 date 域 2014，它 不到任何文 ，因 没有文 含有 个精 日志：

```
GET /_search?q=date:2014         # 0  results !
```

分析器

有些 候很 理解分 的 程和 被存 到索引中的 条，特 是 接触Elasticsearch。 了理解 生了什 ， 可以使用 analyze API 来看文本是如何被分析的。在消息体里，指定分析器和要分析的文本：

```
GET /_analyze
{
  "analyzer": "standard",
  "text": "Text to analyze"
}
```

果中 个元素代表一个 独的 条：

```
{
   "tokens": [
      {
         "token":        "text",
         "start_offset": 0,
         "end_offset":   4,
         "type":         "<ALPHANUM>",
         "position":     1
      },
      {
         "token":        "to",
         "start_offset": 5,
         "end_offset":   7,
         "type":         "<ALPHANUM>",
         "position":     2
      },
      {
         "token":        "analyze",
         "start_offset": 8,
         "end_offset":   15,
         "type":         "<ALPHANUM>",
         "position":     3
      }
   ]
}
```

token 是 存 到索引中的 条。 position 指明 条在原始文本中出 的位置。 start_offset 和 end_offset 指明字符在原始字符串中的位置。

TIP 个分析器的 type 都不一 ，可以忽略它 。它 在Elasticsearch中的唯一作用在于{ref}/analysis-keep-types-tokenfilter.html[keep_types token 器]。

analyze API 是一个有用的工具，它有助于我 理解Elasticsearch索引内部 生了什 ，随着深入，我会 一 它。

指定分析器

当Elasticsearch在 的文 中 到一个新的字符串域，它会自 置其 一个全文 字符串 域，使用 准分析器 它 行分析。

不希望 是 。可能 想使用一个不同的分析器， 用于 的数据使用的 言。有 候 想要一个字符串域就是一个字符串域—不使用分析，直接索引 入的精 ，例如用 ID或者一个内部的状 域或 。

要做到 一点，我 必 手 指定 些域的映射。

映射

了能 将 域 ，数字域 数字，字符串域 全文或精 字符串， Elasticsearch需要知道 个域中数据的 型。 个信息包含在映射中。

如 [data-in-data-out] 中解 的，索引中 个文 都有 型 。 型都有它自己的 映射 ，或者 模式定 。映射定 了 型中的域， 个域的数据 型，以及Elasticsearch如何 理 些域。映射也用于配置与 型有 的元数据。

我 会在 [mapping] 映射。本 ，我 只 足 入 的内容。

核心 域 型

Elasticsearch 支持如下 域 型：

- 字符串: `string`
- 整数 : `byte`, `short`, `integer`, `long`
- 浮点数: `float`, `double`
- 布 型: `boolean`
- 日期: `date`

当 索引一个包含新域的文 —之前未曾出 -- Elasticsearch 会使用 映射 ，通 JSON中基本数据型， 域 型，使用如下 ：

JSON type
: 域 **type**

布 型: `true` 或者 `false`
: `boolean`

整数: `123`
: `long`

浮点数: `123.45`
: `double`

字符串，有效日期: `2014-09-15`
: `date`

字符串: `foo bar`
: `string`

NOTE

意味着如果 通 引号("123")索引一个数字，它会被映射 string 型，而不是 long 。但是，如果 个域已 映射 long ，那 Elasticsearch 会 将 个字符串 化 long ，如果无法 化， 出一个 常。

看映射

通 /_mapping ，我 可以 看 Elasticsearch 在一个或多个索引中的一个或多个 型的映射。在 始章 ，我 已 取得索引 gb 中 型 tweet 的映射：

```
GET /gb/_mapping/tweet
```

Elasticsearch 根据我 索引的文 ， 域(称 属性) 生成的映射。

```
{
   "gb": {
      "mappings": {
         "tweet": {
            "properties": {
               "date": {
                  "type": "date",
                  "format": "strict_date_optional_time||epoch_millis"
               },
               "name": {
                  "type": "string"
               },
               "tweet": {
                  "type": "string"
               },
               "user_id": {
                  "type": "long"
               }
            }
         }
      }
   }
}
```

TIP

的映射，例如 将 age 域映射 string 型，而不是 integer ，会 致 出 令人困惑的 果。

一下！而不是假 的映射是正 的。

自定 域映射

尽管在很多情况下基本域数据 型已 用，但 常需要 独域自定 映射，特 是字符串域。自定 映射允 行下面的操作：

- 全文字符串域和精　　字符串域的区
- 使用特定　言分析器
- 　化域以　　部分匹配
- 指定自定　数据格式
- 　有更多

域最重要的属性是 type 。　于不是 string 的域，　一般只需要　置 type ：

```
{
    "number_of_clicks": {
        "type": "integer"
    }
}
```

　，　　string　　型域会被　　包含全文。就是　，它　的　在索引前，会通　一个分析器，　　于个域的　　在搜索前也会　　一个分析器。

string 域映射的　个最重要属性是 index 和 analyzer 。

index

index 属性控制　　索引字符串。它可以是下面三个　：

analyzed
　首先分析字符串，然后索引它。　句　　，以全文索引　个域。

not_analyzed
　索引　个域，所以它能　被搜索，但索引的是精　　。不会　它　行分析。

no
　不索引　个域。　个域不会被搜索到。

string　域　index　属性　　是　analyzed　。如果我　想映射　个字段　一个精　　，我　需要　置它 not_analyzed ：

```
{
    "tag": {
        "type":     "string",
        "index":    "not_analyzed"
    }
}
```

NOTE	其他　　型（例如 long ， double ， date 等）也接受 index 参数，但有意　的　只有 no 和 not_analyzed ， 因　它　永　不会被分析。

analyzer

于 analyzed 字符串域，用 analyzer 属性指定在搜索和索引 使用的分析器。 ， Elasticsearch 使用 standard 分析器， 但 可以指定一个内置的分析器替代它，例如 whitespace 、 simple 和 english ：

```
{
    "tweet": {
        "type":     "string",
        "analyzer": "english"
    }
}
```

在 [custom-analyzers] ，我 会展示 定 和使用自定 分析器。

更新映射

当 首次 建一个索引的 候，可以指定 型的映射。 也可以使用 /_mapping 新 型（或者 存在的 型更新映射） 加映射。

> **NOTE**
>
> 尽管 可以 加 一个存在的映射， 不能 修改 存在的域映射。如果一个域的映射已 存在，那 域的数据可能已 被索引。如果 意 修改 个域的映射，索引的数据可能会出 ，不能被正常的搜索。

我 可以更新一个映射来添加一个新域，但不能将一个存在的域从 analyzed 改 not_analyzed 。

了描述指定映射的 方式，我 先 除 gd 索引：

```
DELETE /gb
```

然后 建一个新索引，指定 tweet 域使用 english 分析器：

```
PUT /gb ①
{
  "mappings": {
    "tweet" : {
      "properties" : {
        "tweet" : {
          "type" :    "string",
          "analyzer": "english"
        },
        "date" : {
          "type" :   "date"
        },
        "name" : {
          "type" :   "string"
        },
        "user_id" : {
          "type" :   "long"
        }
      }
    }
  }
}
```

① 通　消息体中指定的 mappings　建了索引。

后，我　决定在 tweet 映射　加一个新的名　tag 的 not_analyzed 的文本域，使用 _mapping ：

```
PUT /gb/_mapping/tweet
{
  "properties" : {
    "tag" : {
      "type" :    "string",
      "index":    "not_analyzed"
    }
  }
}
```

注意，我　不需要再次列出所有已存在的域，因　无　如何我　都无法改　它　。新域已　被合并到存在的映射中。

映射

可以使用 analyze API　　字符串域的映射。比　下面　个　求的　出：

```
GET /gb/_analyze
{
  "field": "tweet",
  "text": "Black-cats" ①
}

GET /gb/_analyze
{
  "field": "tag",
  "text": "Black-cats" ①
}
```

① 消息体里面　　我　想要分析的文本。

tweet　域　生　个　条　black　和　cat　，　tag　域　生　独的　条　Black-cats　。　句　　，我　的映射正常工作。

核心域　型

除了我　提到的　　量数据　型，JSON　有 null　，数　，和　象，　些 Elasticsearch 都是支持的。

多　域

很有可能，我　希望 tag 域包含多个　　。我　可以以数　的形式索引　　：

```
{ "tag": [ "search", "nosql" ]}
```

于数　，没有特殊的映射需求。任何域都可以包含0、1或者多个　，就像全文域分析得到多个　条。

暗示　数　中所有的　必　是相同数据　型的　。　不能将日期和字符串混在一起。如果　通　索引数来　建新的域，Elasticsearch 会用数　中第一个　的数据　型作　　个域的　型 。

NOTE

当　从 Elasticsearch 得到一个文　，　个数　的　序和　当初索引文　　一　。　得到的 _source 域，包含与　索引的一模一　的 JSON 文　。

但是，数　是以多　域　索引的—可以搜索，但是无序的。　在搜索的　候，　不能指定　　“第一个”　　或者　　“最后一个”。　　更　切的　，把数　想象成 装在袋子里的　 。

空域

当然，数　可以　空。　相当于存在零　。　事　上，在　Lucene　中是不能存　　null　　的，所以我　　存在 null　的域　空域。

下面三　域被　　是空的，它　将不会被索引：

```
"null_value":                null,
"empty_array":               [],
"array_with_null_value":     [ null ]
```

多　　象

我　　的最后一个 JSON 原生数据　是　象 -- 在其他　言中称　哈希，哈希 map，字典或者　　数　。

内部　象　　常用于嵌入一个　体或　象到其它　象中。例如，与其在 `tweet`　文　中包含 `user_name`　和 `user_id` 域，我　也可以　　写：

```
{
    "tweet":            "Elasticsearch is very flexible",
    "user": {
        "id":           "@johnsmith",
        "gender":       "male",
        "age":          26,
        "name": {
            "full":     "John Smith",
            "first":    "John",
            "last":     "Smith"
        }
    }
}
```

内部　象的映射

Elasticsearch 会　　新的　象域并映射它　　象，在 `properties` 属性下列出内部域：

```
{
  "gb": {
    "tweet": { ①
      "properties": {
        "tweet":            { "type": "string" },
        "user": { ②
          "type":             "object",
          "properties": {
            "id":           { "type": "string" },
            "gender":       { "type": "string" },
            "age":          { "type": "long"   },
            "name":   { ②
              "type":         "object",
              "properties": {
                "full":     { "type": "string" },
                "first":    { "type": "string" },
                "last":     { "type": "string" }
              }
            }
          }
        }
      }
    }
  }
}
```

① 根 象

② 内部 象

user 和 name 域的映射 与 tweet 型的相同。事 上， type 映射只是一 特殊的 象 映射，我称之 根 象 。除了它有一些文 元数据的特殊 域，例如 _source 和 _all 域，它和其他 象一 。

内部 象是如何索引的

Lucene 不理解内部 象。 Lucene 文 是由一 列表 成的。 了能 Elasticsearch 有效地索引内部 ，它把我 的文 化成 ：

```
{
    "tweet":            [elasticsearch, flexible, very],
    "user.id":          [@johnsmith],
    "user.gender":      [male],
    "user.age":         [26],
    "user.name.full":   [john, smith],
    "user.name.first":  [john],
    "user.name.last":   [smith]
}
```

内部域 可以通 名称引用（例如， first ）。 了区分同名的 个域，我 可以使用全 路径 （例如，

user.name.first ） 或 type 名加路径（ tweet.user.name.first ）。

NOTE	在前面 扁平的文 中，没有 user 和 user.name 域。Lucene 索引只有 量和 ，没有 数据 。

内部 象数

最后，考 包含内部 象的数 是如何被索引的。假 我 有个 followers 数 ：

```
{
    "followers": [
        { "age": 35, "name": "Mary White"},
        { "age": 26, "name": "Alex Jones"},
        { "age": 19, "name": "Lisa Smith"}
    ]
}
```

个文 会像我 之前描述的那 被扁平化 理， 果如下所示：

```
{
    "followers.age":    [19, 26, 35],
    "followers.name":   [alex, jones, lisa, smith, mary, white]
}
```

{age: 35} 和 {name: Mary White} 之 的相 性已 失了，因 个多 域只是一包无序的 ，而不是有序数 。 足以 我 ，“有一个26 的追随者？”

但是我 不能得到一个准 的答案：“是否有一个26 名字叫 *Alex Jones* 的追随者？”

相 内部 象被称 *nested* 象，可以回答上面的 ，我 后会在[nested-objects]中介 它。