搜索——最基本的工具

在，我 已 学会了如何使用 Elasticsearch 作 一个 的 NoSQL 格的分布式文 存 系 。我可以将一个 JSON 文 到 Elasticsearch 里，然后根据 ID 索。但 Elasticsearch 真正 大之在于可以从无 律的数据中 出有意 的信息——从“大数据”到“大信息”。

Elasticsearch 不只会_存 （stores）_ 文 ， 了能被搜索到也会 文 添加_索引（indexes）_ ，也是 什 我 使用 化的 JSON 文 ，而不是无 的二 制数据。

文 中的 个字段都将被索引并且可以被 。不 如此，在 ，Elasticsearch 可以使用所有（*all*） 些索引字段，以 人的速度返回 果。 是 永 不会考 用 数据 去做的一些事情。

搜索（*search*） 可以做到：

- 在 似于 `gender` 或者 `age` 的字段上使用 化 ，`join_date` 的字段上使用排序，就像SQL的 化 一 。
- 全文 索， 出所有匹配 字的文 并按照_相 性（relevance）_ 排序后返回 果。
- 以上二者兼而有之。

很多搜索都是 箱即用的， 了充分 掘 Elasticsearch 的潜力， 需要理解以下三个概念：

映射（*Mapping*）
: 描述数据在 个字段内如何存

分析（*Analysis*）
: 全文是如何 理使之可以被搜索的

域特定 言（*Query DSL*）
: Elasticsearch 中 大 活的 言

以上提到的 个点都是一个大 ，我 将在 [search-in-depth] 一章 述它 。本章 我 将介 三点的一些基本概念—— 助 大致了解搜索是如何工作的。

我 将使用最 的形式 始介 `search` API。

> **数据**
>
> 本章 的 数据可以在 里 到：https://gist.github.com/clintongormley/8579281 。
>
> 可以把 些命令 制到 端中 行来 践本章的例子。
>
> 外，如果 的是在 版本，可以 点 个 接 感受下。

空搜索

搜索API的最基 的形式是没有指定任何 的空搜索，它 地返回集群中所有索引下的所有文 ：

```
GET /_search
```

返回的 果（ 了界面 的）像 ：

```
{
   "hits" : {
      "total" :       14,
      "hits" : [
        {
          "_index":   "us",
          "_type":    "tweet",
          "_id":      "7",
          "_score":   1,
          "_source": {
             "date":    "2014-09-17",
             "name":    "John Smith",
             "tweet":   "The Query DSL is really powerful and flexible",
             "user_id": 2
          }
       },
        ... 9 RESULTS REMOVED ...
      ],
      "max_score" :   1
   },
   "took" :           4,
   "_shards" : {
      "failed" :      0,
      "successful" :  10,
      "total" :       10
   },
   "timed_out" :      false
}
```

hits

返回 果中最重要的部分是 hits ，它包含 total 字段来表示匹配到的文 数，并且一个 hits 数包含所 果的前十个文 。

在 hits 数 中 个 果包含文 的 _index 、 _type 、 _id ，加上 _source 字段。 意味着我可以直接从返回的搜索 果中使用整个文 。 不像其他的搜索引 ， 返回文 的ID，需要 独去取文 。

个 果 有一个 _score ，它衡量了文 与 的匹配程度。 情况下，首先返回最相 的文果，就是 ，返回的文 是按照 _score 降序排列的。在 个例子中，我 没有指定任何 ，故所有的文 具有相同的相 性，因此 所有的 果而言 1 是中性的 _score 。

max_score 是与 所匹配文 的 _score 的最大 。

took

`took` 告 我 行整个搜索 求耗 了多少 秒。

shards

`_shards` 部分告 我 在 中参与分片的 数，以及 些分片成功了多少个失了多少个。正常情况下我 不希望分片失 ，但是分片失 是可能 生的。如果我 遭遇到一 的故障，在 个故障中 失了相同分片的原始数据和副本，那 个分片将没有可用副本来 搜索 求作出 。假若 ，Elasticsearch 将 告 个分片是失 的，但是会 返回剩余分片的 果。

timeout

`timed_out` 告 我 是否超 。 情况下，搜索 求不会超 。如果低 比完成 果更重要， 可以指定 `timeout` 10 或者 10ms（10 秒），或者 1s（1秒）：

```
GET /_search?timeout=10ms
```

在 求超 之前，Elasticsearch 将会返回已 成功从 个分片 取的 果。

WARNING	当注意的是 `timeout` 不是停止 行 ，它 是告知正在 的点返回到目前 止收集的 果并且 接。在后台，其他的分片可能 在 行即使是 果已 被 送了。 使用超 是因 SLA(服 等) 是很重要的，而不是因 想去中止 行的 。

多索引，多 型

有没有注意到之前的 empty search 的 果，不同 型的文 — <code>user</code> 和 <code>tweet</code> 来自不同的索引— <code>us</code> 和 <code>gb</code> ？

如果不 某一特殊的索引或者 型做限制，就会搜索集群中的所有文 。Elasticsearch 搜索 求到 一个主分片或者副本分片， 集 出的前10个 果，并且返回 我 。

然而， 常的情况下， 想在一个或多个特殊的索引并且在一个或者多个特殊的 型中 行搜索。我 可以通 在URL中指定特殊的索引和 型 到 效果，如下所示：

`/_search`
在所有的索引中搜索所有的 型

`/gb/_search`
在 `gb` 索引中搜索所有的 型

`/gb,us/_search`
在 `gb` 和 `us` 索引中搜索所有的文

`/g*,u*/_search`
　在任何以 g 或者 u　　的索引中搜索所有的　型

`/gb/user/_search`
　在 gb 索引中搜索 user　型

`/gb,us/user,tweet/_search`
　在 gb 和 us 索引中搜索 user 和 tweet　型

`/_all/user,tweet/_search`
　在所有的索引中搜索 user 和 tweet　型

当在　一的索引下　行搜索的　候，Elasticsearch　　求到索引的　个分片中，可以是主分片也可以是副本分片，然后从　个分片中收集　果。多索引搜索恰好也是用相同的方式工作的—只是会　及到更多的分片。

> **TIP**　搜索一个索引有五个主分片和搜索五个索引各有一个分片准　来所　是等　的。

接下来，　将明白　　的方式如何　活的根据需求的　化　　容　得　　。

分

在之前的 空搜索 中　明了集群中有 14 个文　匹配了（empty）query 。 但是在 hits 数　中只有 10 个文　。如何才能看到其他的文　？

和 SQL 使用 LIMIT　　字返回　个 page　果的方法相同，Elasticsearch 接受 from 和 size 参数：

`size`
　　示　　返回的　果数量，　　是 10

`from`
　　示　　跳　的初始　果数量，　　是 0

如果　　展示 5 条　果，可以用下面方式　求得到 1 到 3　的　果：

```
GET /_search?size=5
GET /_search?size=5&from=5
GET /_search?size=5&from=10
```

考　到分　　深以及一次　求太多　果的情况，　果集在返回之前先　行排序。　　但　　住一个　求　常跨越多个分片，　个分片都　生自己的排序　果，　些　果需要　行集中排序以保　整体　序是正　的。

在分布式系 中深度分

理解 什 深度分 是有 的，我 可以假 在一个有 5 个主分片的索引中搜索。 当我 求 果的第一 （ 果从 1 到 10 ）， 一个分片 生前 10 的 果，并且返回 点 ，点 50 个 果排序得到全部 果的前 10 个。

在假 我 求第 1000 — 果从 10001 到 10010 。所有都以相同的方式工作除了个分片不得不 生前10010个 果以外。 然后 点 全部 50050 个 果排序最后 掉 些果中的 50040 个 果。

可以看到，在分布式系 中， 果排序的成本随分 的深度成指数上升。 就是 web 搜索引 任何 都不要返回超 1000 个 果的原因。

TIP 在 [reindex] 中解 了如何 能 有效 取大量的文 。

量 搜索

有 形式的 搜索 API：一 是 `` 量的" 字符串 版本，要求在 字符串中 所有的参数，一 是更完整的 求体 版本，要求使用 JSON 格式和更 富的 表 式作 搜索 言。

字符串搜索非常 用于通 命令行做即席 。例如， 在 `tweet` 型中 `tweet` 字段包含 `elasticsearch` 的所有文 ：

```
GET /_all/tweet/_search?q=tweet:elasticsearch
```

下一个 在 `name` 字段中包含 `john` 并且在 `tweet` 字段中包含 `mary` 的文 。 的 就是

```
+name:john +tweet:mary
```

但是 字符串参数所需要的 百分比 （ 者注：URL ） 上更加 ：

```
GET /_search?q=%2Bname%3Ajohn+%2Btweet%3Amary
```

`+` 前 表示必 与 条件匹配。 似地， `-` 前 表示一定不与 条件匹配。没有 `+` 或者 `-` 的所有其他条件都是可 的——匹配的越多，文 就越相 。

_all 字段

个 搜索返回包含 `mary` 的所有文 ：

```
GET /_search?q=mary
```

之前的例子中，我 在 `tweet` 和 `name` 字段中搜索内容。然而， 个 的 果在三个地方提到了 `mary` ：

- 有一个用　叫做 Mary
- 6条微博　自 Mary
- 一条微博直接 @mary

Elasticsearch 是如何在三个不同的字段中　　到　果的　？

当索引一个文　的　候，Elasticsearch　　取出所有字段的　　接成一个大的字符串，作　　`_all`　　字段　行索引。例如，当索引　个文　　：

```
{
    "tweet":    "However did I manage before Elasticsearch?",
    "date":     "2014-09-14",
    "name":     "Mary Jones",
    "user_id":  1
}
```

　就好似　加了一个名叫 `_all` 的　外字段：

```
"However did I manage before Elasticsearch? 2014-09-14 Mary Jones 1"
```

除非　置特定字段，否　　字符串就使用 `_all` 字段　行搜索。

TIP	在　始　一个　用　，`_all`　　字段是一个很　用的特性。之后，　会　如果搜索　用指定字段来代替　`_all`　字段，将会更好控制搜索　果。当　`_all`　字段不再有用的候，可以将它置　失效，正如在 [all-field] 中所解　的。

更　的

下面的　　tweents　型，并使用以下的条件：

- `name` 字段中包含 `mary` 或者 `john`
- `date`　大于 `2014-09-10`
- `all` 字段包含 `aggregations` 或者 `geo`

```
+name:(mary john) +date:>2014-09-10 +(aggregations geo)
```

　字符串在做了　当的　后，可　性很差：

```
?q=%2Bname%3A(mary+john)+%2Bdate%3A%3E2014-09-10+%2B(aggregations+geo)
```

从之前的例子中可以看出，　　量　的　字符串搜索效果　是挺　人　喜的。　它的　　法在相　参考文　中有　解　，以便　的表　很　的　。　于通　命令做一次性　，或者是在　　段，都非常方便。

但同　也可以看到，　　精　　更加晦　和困　。而且很脆弱，一些　　字符串中很小的　法　　，像 `-`，`:`，`/` 或者 `"` 不匹配等，将会返回　　而不是搜索　果。

最后，　　字符串搜索允　任何用　在索引的任意字段上　行可能　慢且重量　的　　，　可能会暴露　私信息，甚至将集群　　。

TIP	因　　些原因，不推　直接向用　暴露　　字符串搜索功能，除非　于集群和数据来　非常信任他　。

相反，我　　常在生　　境中更多地使用功能全面的 *request　body*　　API，除了能完成以上所有功能，　有一些附加功能。但在到　那个　段之前，首先需要了解数据在 Elasticsearch 中是如何被索引的。