**5搜索**

**搜索——基本的工具**

到目前为止，我们已经学会了如何使用elasticsearch作为一个简单的NoSQL风格的分布式文件存储器——我们可以将一个JSON文档扔给Elasticsearch，也可以根据ID检索它们。但Elasticsearch真正强大之处在于可以从混乱的数据中找出有意义的信息——从大数据到全面的信息。

这也是为什么我们使用结构化的JSON文档，而不是无结构的二进制数据。Elasticsearch不只会**存储(store)**文档，也会**索引(indexes)**文档内容来使之可以被搜索。

**每个文档里的字段都会被索引并被查询**。而且不仅如此。在简单查询时，Elasticsearch可以使用**所有**的索引，以非常快的速度返回结果。这让你永远不必考虑传统数据库的一些东西。

A *search* can be: **搜索(search)**可以：

* 在类似于gender或者age这样的字段上使用结构化查询，join\_date这样的字段上使用排序，就像SQL的结构化查询一样。
* 全文检索，可以使用所有字段来匹配关键字，然后按照**关联性(relevance)**排序返回结果。
* 或者结合以上两条。

很多搜索都是开箱即用的，为了充分挖掘Elasticsearch的潜力，你需要理解以下三个概念：

| **概念** | **解释** |
| --- | --- |
| **映射(Mapping)** | 数据在每个字段中的解释说明 |
| **分析(Analysis)** | 全文是如何处理的可以被搜索的 |
| **领域特定语言查询(Query DSL)** | Elasticsearch使用的灵活的、强大的查询语言 |

以上提到的每个点都是一个巨大的话题，我们将在《深入搜索》一章阐述它们。本章节我们将介绍这三点的一些基本概念——仅仅帮助你大致了解搜索是如何工作的。

我们将使用最简单的形式开始介绍search API.

**测试数据**

本章节测试用的数据可以在这里被找到<https://gist.github.com/clintongormley/8579281>

你可以把这些命令复制到终端中执行以便可以实践本章的例子。

# 5.1空搜索

最基本的搜索API表单是**空搜索(empty search)**，它没有指定任何的查询条件，只返回集群索引中的所有文档：

GET /\_search

响应内容（为了编辑简洁）类似于这样：

{

"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

响应中最重要的部分是hits，它包含了total字段来表示匹配到的文档总数，hits数组还包含了匹配到的前10条数据全文搜索自带了前10条匹配到的数据。

hits数组中的每个结果都包含\_index、\_type和文档的\_id字段，被加入到\_source字段中这意味着在搜索结果中我们将可以直接使用全部文档。这不像其他搜索引擎只返回文档ID，需要你单独去获取文档。

每个节点都有一个\_score字段，这是**相关性得分(relevance score)**，文档按照此字段进行降序排列，\_score不是文档自身具备的，它是搜索过程中根据相关性es计算得来的它衡量了文档与查询的匹配程度。默认的，返回的结果中关联性最大的文档排在首位；这意味着，它是按照\_score降序排列的。这种情况下，我们没有指定任何查询，所以所有文档的相关性是一样的，因此所有结果的\_score都是取得一个中间值1

max\_score指的是所有文档匹配查询中\_score的最大值。

## took

took告诉我们整个搜索请求花费的毫秒数。

## shards

\_shards节点告诉我们参与查询的分片数（total字段），有多少是成功的（successful字段），有多少的是失败的（failed字段）这里将主分片和复制分片看做是一个，都失败的话failed增加1，表明主分片和复制分片都没有命中数据。通常我们不希望分片失败，不过这个有可能发生。如果我们遭受一些重大的故障导致主分片和复制分片都故障，那这个分片的数据将无法响应给搜索请求。这种情况下，Elasticsearch将报告分片failed，但仍将继续返回剩余分片上的结果。

## timeout

time\_out值告诉我们查询超时与否。一般的，搜索请求不会超时。如果响应速度比完整的结果更重要，你可以定义timeout参数为10或者10ms（10毫秒），或者1s（1秒）

GET /\_search?timeout=10ms

Elasticsearch将返回在请求超时前收集到的结果。

超时不是一个断路器（circuit breaker）（译者注：关于断路器的理解请看警告）。timeout只是告诉查询请求的节点目前返回所有收集到的数据并关闭与会其他的节点之间的连接，其他的连接可能仍然在查询。

## 警告

需要注意的是timeout不会停止执行查询，它仅仅告诉你**目前**顺利返回结果的节点然后关闭连接。在后台，其他分片可能依旧执行查询，尽管结果已经被发送。

使用超时是因为对于你的业务需求（译者注：SLA，Service-Level Agreement服务等级协议，在此我翻译为业务需求）来说非常重要，而不是因为你想中断执行长时间运行的查询。

## 5.2多索引和多类别

你注意到空搜索的结果中不同类型的文档——user和tweet——来自于不同的索引——us和gb。

通过限制搜索的不同索引或类型，我们可以在集群中跨**所有**文档搜索。Elasticsearch转发搜索请求到集群中平行的主分片或每个分片的复制分片上，收集结果后选择顶部十个返回给我们。

通常，当然，你可能想搜索一个或几个自定的索引或类型，我们能通过定义URL中的索引或类型达到这个目的，像这样：

/\_search

在所有索引的所有类型中搜索

/gb/\_search

在索引gb的所有类型中搜索

/gb,us/\_search

在索引gb和us的所有类型中搜索

/g\*,u\*/\_search

在以g或u开头的索引的所有类型中搜索

/gb/user/\_search

在索引gb的类型user中搜索

/gb,us/user,tweet/\_search

在索引gb和us的类型为user和tweet中搜索

/\_all/user,tweet/\_search

在所有索引的user和tweet中搜索 search types user and tweet in all indices

当你搜索包含单一索引时，Elasticsearch转发搜索请求到这个索引的主分片或每个分片的复制分片上，然后聚集每个分片的结果。搜索包含多个索引也是同样的方式——只不过或有更多的分片被关联。

## 重要

搜索一个索引有5个主分片和5个索引各有一个分片**事实上是一样的**。

接下来，你将看到这些简单的情况如何灵活的扩展以适应你需求的变更。