# 06-如何用Elasticsearch构建商品搜索系统?

你好，我是李玥。

搜索这个特性可以说是无处不在，现在很少有网站或者系统不提供搜索功能了，所以，即使你不是一个专业做搜索的程序员，也难免会遇到一些搜索相关的需求。搜索这个东西，表面上看功能很简单，就是一个搜索框，输入关键字，然后搜出来想要的内容就好了。

搜索背后的实现，可以非常简单，简单到什么程度呢？我们就用一个SQL，LIKE一下就能实现；也可以很复杂，复杂到什么程度呢？不说百度谷歌这种专业做搜索的公司，其他非专业做搜索的互联网大厂，搜索团队大多是千人规模，这里面不仅有程序员，还有算法工程师、业务专家等等。二者的区别也仅仅是，搜索速度的快慢以及搜出来的内容好坏而已。

今天这节课，我们就以电商中的商品搜索作为例子，来讲一下，如何用ES(Elasticsearch)来快速、低成本地构建一个体验还不错的搜索系统。

## 理解倒排索引机制

刚刚我们说了，既然我们的数据大多都是存在数据库里，用SQL的LIKE也能实现匹配，也能搜出结果，为什么还要专门做一套搜索系统呢？我先来和你分析一下，为什么数据库不适合做搜索。

搜索的核心需求是全文匹配，对于全文匹配，数据库的索引是根本派不上用场的，那只能全表扫描。全表扫描已经非常慢了，这还不算，还需要在每条记录上做全文匹配，也就是一个字一个字的比对，这个速度就更慢了。所以，使用数据来做搜索，性能上完全没法满足要求。

那ES是怎么来解决搜索问题的呢？我们来举个例子说明一下，假设我们有这样两个商品，一个是烟台红富士苹果，一个是苹果手机iPhone XS Max。

| DOCID | SKUID | 标题 |
| --- | --- | --- |
| 666 | 100002860826 | 烟台红富士苹果 5kg 一级铂金大果 单果230g以上 新鲜水果 |
| 888 | 100000177760 | 苹果 Apple iPhone XS Max (A2104) 256GB 金色 移动联通电信4G手机 双卡双待 |

这个表里面的DOCID就是唯一标识一条记录的ID，和数据库里面的主键是类似的。

为了能够支持快速地全文搜索，ES中对于文本采用了一种特殊的索引：倒排索引（Inverted Index）。那我们看一下在ES中，这两条商品数据倒排索引长什么样？请看下面这个表。

| TERM | DOCID |
| --- | --- |
| 烟台 | 666 |
| 红富士 | 666 |
| 苹果 | 666,888 |
| 5kg | 666 |
| 一级 | 666 |
| 铂金 | 666 |
| 大果 | 666 |
| Apple | 888 |
| iphone | 888 |
| XS | 888 |
| Max | 888 |
| 手机 | 888 |
| … | … |

可以看到，这个倒排索引的表，它是以单词作为索引的Key，然后每个单词的倒排索引的值是一个列表，这个列表的元素就是含有这个单词的商品记录的DOCID。

这个倒排索引怎么构建的呢？当我们往ES写入商品记录的时候，ES会先对需要搜索的字段，也就是商品标题进行**分词**。分词就是把一段连续的文本按照语义拆分成多个单词。然后ES按照单词来给商品记录做索引，就形成了上面那个表一样的倒排索引。

当我们搜索关键字“苹果手机”的时候，ES会对关键字也进行分词，比如说，“苹果手机”被分为“苹果”和“手机”。然后，ES会在倒排索引中去搜索我们输入的每个关键字分词，搜索结果应该是：

| TERM | DOCID |
| --- | --- |
| 苹果 | 666,888 |
| 手机 | 888 |

666和888这两条记录都能匹配上搜索的关键词，但是888这个商品比666这个商品匹配度更高，因为它两个单词都能匹配上，所以按照匹配度把结果做一个排序，最终返回的搜索结果就是：

> **苹果**Apple iPhone XS Max (A2104) 256GB 金色 移动联通电信4G**手机**双卡双待

> 烟台红富士**苹果**5kg 一级铂金大果 单果230g以上 新鲜水果

看起来搜索的效果还是不错的。

为什么倒排索引可以做到快速搜索？我和你一起来分析一下上面这个例子的查找性能。

这个搜索过程，其实就是对上面的倒排索引做了二次查找，一次找“苹果”，一次找“手机”。**注意，整个**

**搜索过程中，我们没有做过任何文本的模糊匹配**。ES的存储引擎存储倒排索引时，肯定不是像我们上面表格中展示那样存成一个二维表，实际上它的物理存储结构和MySQL的InnoDB的索引是差不多的，都是一颗查找树。

对倒排索引做两次查找，也就是对树进行二次查找，它的时间复杂度，类似于MySQL中的二次命中索引的查找。显然，这个查找速度，比用MySQL全表扫描加上模糊匹配的方式，要快好几个数量级。

## 如何在ES中构建商品的索引?

理解了倒排索引的原理之后，我们一起用ES构建一个商品索引，简单实现一个商品搜索系统。虽然ES是为搜索而生的，但本质上，它仍然是一个存储系统。ES里面的一些概念，基本上都可以在关系数据库中找到对应的名词，为了便于你快速理解这些概念，我把这些概念的对应关系列出来，你可以对照理解。

| ElasticSearch | RDBMS |
| --- | --- |
| INDEX | 表 |
| DOCUMENT | 行 |
| FIELD | 列 |
| MAPPING | 表结构 |

在ES里面，数据的逻辑结构类似于MongoDB，每条数据称为一个**DOCUMENT**，简称DOC。DOC就是一个JSON对象，DOC中的每个JSON字段，在ES中称为**FIELD**，把一组具有相同字段的DOC存放在一起，存放它们的逻辑容器叫**INDEX**，这些DOC的JSON结构称为**MAPPING**。这里面最不好理解的就是这个INDEX，它实际上类似于MySQL中表的概念，而不是我们通常理解的用于查找数据的索引。

ES是一个用Java开发的服务端程序，除了Java以外就没有什么外部依赖了，安装部署都非常简单，具体你可以参照它的官方文档先把ES安装好。我们这个示例中，使用的ES版本是目前的最新版本7.6。

另外，为了能让ES支持中文分词，需要给ES安装一个中文的分词插件IK Analysis for Elasticsearch，这个插件的作用就是告诉ES怎么对中文文本进行分词。

你可以直接执行下面的命令自动下载并安装：

```
$elasticsearch-plugin install https://github.com/medcl/elasticsearch-analysis-ik/releases/download/v7.6.0/e
```

安装完成后，需要重启ES，验证一下是否安装成功：

```
curl -X POST "localhost:9200/_analyze?pretty" -H 'Content-Type: application/json' -d '{ "analyzer": "ik_sma
{
  "tokens" : [
    {
      "token" : "极",
      "start_offset" : 0,
      "end_offset" : 1,
      "type" : "CN_CHAR",
```

```
      "position" : 0
    },
    {
      "token" : "客",
      "start_offset" : 1,
      "end_offset" : 2,
      "type" : "CN_CHAR",
      "position" : 1
    },
    {
      "token" : "时间",
      "start_offset" : 2,
      "end_offset" : 4,
      "type" : "CN_WORD",
      "position" : 2
    }
  ]
}
```

可以看到，这个分词器把“极客时间”分成了“极”、“客”和“时间”，没认出来“极客”这个词，还是有改进空间的。

为了能实现商品搜索，我们需要先把商品信息存放到ES中，首先我们先定义存放在ES中商品的数据结构，也就是MAPPING。

| Field | Datatype | 说明 |
| --- | --- | --- |
| sku_id | long | 商品ID |
| title | text | 商品标题 |

我们这个MAPPING只要两个字段就够了，sku_id就是商品ID，title保存商品的标题，当用户在搜索商品的时候，我们在ES中来匹配商品标题，返回符合条件商品的sku_id列表。ES默认提供了标准的RESTful接口，不需要客户端，直接使用HTTP协议就可以访问，这里我们使用curl通过命令行来操作ES。

接下来我们使用上面这个MAPPING创建INDEX，类似于MySQL中创建一个表。

```
curl -X PUT "localhost:9200/sku" -H 'Content-Type: application/json' -d '{
        "mappings": {
                "properties": {
                        "sku_id": {
                                "type": "long"
                        },
                        "title": {
                                "type": "text",
                                "analyzer": "ik_max_word",
                                "search_analyzer": "ik_max_word"
                        }
                }
        }
}'
{"acknowledged":true,"shards_acknowledged":true,"index":"sku"}
```

这里面，使用PUT方法创建一个INDEX，INDEX的名称是“sku”，直接写在请求的URL中。请求的BODY是一个JSON对象，内容就是我们上面定义的MAPPING，也就是数据结构。这里面需要注意一下，由于我们要在title这个字段上进行全文搜索，所以我们把数据类型定义为text，并指定使用我们刚刚安装的中文分词插件IK作为这个字段的分词器。

创建好INDEX之后，就可以往INDEX中写入商品数据，插入数据需要使用HTTP POST方法：

```
curl -X POST "localhost:9200/sku/_doc/" -H 'Content-Type: application/json' -d '{
        "sku_id": 100002860826,
        "title": "烟台红富士苹果 5kg 一级铂金大果 单果230g以上 新鲜水果"
}'
{"_index":"sku","_type":"_doc","_id":"yxQVSHABiy2kuAJG8ilW","_version":1,"result":"created","_shards":{"tot

curl -X POST "localhost:9200/sku/_doc/" -H 'Content-Type: application/json' -d '{
        "sku_id": 100000177760,
        "title": "苹果 Apple iPhone XS Max (A2104) 256GB 金色 移动联通电信4G手机 双卡双待"
}'
{"_index":"sku","_type":"_doc","_id":"zBQWSHABiy2kuAJGgim1","_version":1,"result":"created","_shards":{"tot
```

这里面我们插入了两条商品数据，一个烟台红富士，一个iPhone手机。然后就可以直接进行商品搜索了，搜索使用HTTP GET方法。

```
curl -X GET 'localhost:9200/sku/_search?pretty' -H 'Content-Type: application/json' -d '{
  "query" : { "match" : { "title" : "苹果手机" }}
}'
{
  "took" : 23,
  "timed_out" : false,
  "_shards" : {
    "total" : 1,
    "successful" : 1,
    "skipped" : 0,
    "failed" : 0
  },
  "hits" : {
    "total" : {
      "value" : 2,
      "relation" : "eq"
    },
    "max_score" : 0.8594865,
    "hits" : [
      {
        "_index" : "sku",
        "_type" : "_doc",
        "_id" : "zBQWSHABiy2kuAJGgim1",
        "_score" : 0.8594865,
        "_source" : {
          "sku_id" : 100000177760,
          "title" : "苹果 Apple iPhone XS Max (A2104) 256GB 金色 移动联通电信4G手机 双卡双待"
        }
      },
      {
        "_index" : "sku",
        "_type" : "_doc",
        "_id" : "yxQVSHABiy2kuAJG8ilW",
```

```
        "_score" : 0.18577608,
        "_source" : {
          "sku_id" : 100002860826,
          "title" : "烟台红富士苹果 5kg 一级铂金大果 单果230g以上 新鲜水果"
        }
      }
    ]
  }
}
```

我们先看一下请求中的URL，其中的“sku”代表要在sku这个INDEX内进行查找，“_search”是一个关键字，表示要进行搜索，参数pretty表示格式化返回的JSON，这样方便阅读。再看一下请求BODY的JSON，query中的match表示要进行全文匹配，匹配的字段就是title，关键字是“苹果手机”。

可以看到，在返回结果中，匹配到了2条商品记录，和我们在前面讲解倒排索引时，预期返回的结果是一致的。

我们来回顾一下使用ES构建商品搜索服务的这个过程：首先安装ES并启动服务，然后创建一个INDEX，定义MAPPING，写入数据后，执行查询并返回查询结果，其实，这个过程和我们使用数据库时，先建表、插入数据然后查询的过程，就是一样的。所以，你就把ES当做一个支持全文搜索的数据库来使用就行了。

## 小结

ES本质上是一个支持全文搜索的分布式内存数据库，特别适合用于构建搜索系统。ES之所以能有非常好的全文搜索性能，最重要的原因就是采用了倒排索引。倒排索引是一种特别为搜索而设计的索引结构，倒排索引先对需要索引的字段进行分词，然后以分词为索引组成一个查找树，这样就把一个全文匹配的查找转换成了对树的查找，这是倒排索引能够快速进行搜索的根本原因。

但是，倒排索引相比于一般数据库采用的B树索引，它的写入和更新性能都比较差，因此倒排索引也只是适合全文搜索，不适合更新频繁的交易类数据。

## 思考题

我们在电商的搜索框中搜索商品时，它都有一个搜索提示的功能，比如我输入“苹果”还没有点击搜索按钮的时候，搜索框下面会提示“苹果手机”、“苹果11、苹果电脑”这些建议的搜索关键字，请你课后看一下ES的文档，想一下，如何用ES快速地实现这个搜索提示功能？

欢迎你在留言区与我讨论，如果你觉得今天的内容对你有帮助，也欢迎把它分享给你的朋友。

## 精选留言：

- 李玥 2020-03-10 10:09:41
  Hi，我是李玥。

  还是在这里回顾一下上节课的思考题：

  2PC也有一些改进版本，比如3PC、TCC这些，它们大体的思想和2PC是差不多的，解决了2PC的一些问题，但是也会带来新的问题，实现起来也更复杂，限于篇幅我们没法每个都详细地去讲解。在理解了2PC的基础上，课后请你自行去学习一下3PC和TCC，然后对比一下，2PC、3PC和TCC分别适用于什么样的业务

场景？

谈一下我的理解：

3PC相比于2PC做了两个改进，一是事务执行器也增加了超时机制，避免我们课程中提到的因为协调者宕机，导致执行器长时间卡死的问题，另外，3PC在2PC之前增加一个询问阶段，这个阶段事务执行器可以去尝试锁定资源（但不等待），这样避免像2PC那样直接去锁定资源，而资源不可用的情况下，一直等待资源而卡住事务的情况。

TCC可以理解为业务层面的2PC（也有观点主张TCC和2PC是完全不同的，我个人建议没必要在这些概念上较真，理解并正确使用才是关键），TCC同样分为Try和Confirm/Cancel 两个阶段，在Try阶段锁定资源，但不执行任何更新操作，Confirm阶段来执行所有更新操作并提交，如果失败进入Cancel阶段。Cancel阶段就是收拾烂摊子，把Confirm阶段做的数据更新都改回去，把Try阶段锁定的资源都释放。相比于2PC，TCC可以不依赖于本地事务，但是Cancel阶段的业务逻辑比较难实现。

- Geek_c76e2d 2020-03-10 01:19:16
  因为用户每输入一个字都可能会发请求查询搜索框中的搜索推荐。所以搜索推荐的请求量远高于搜索框中的搜索。es针对这种情况提供了suggestion api，并提供的专门的数据结构应对搜索推荐，性能高于match，但它应用起来也有局限性，就是只能做前缀匹配。再结合pinyin分词器可以做到输入拼音字母就提示中文。如果想做非前缀匹配，可以考虑Ngram。不过Ngram有些复杂，需要开发者自定义分析器。比如有个网址www.geekbang.com，用户可能记不清具体网址了，只记得网址中有2个e，此时用户输入ee两个字母也是可以在搜索框提示出这个网址的。以上是我在工作中针对前缀搜索推荐和非前缀搜索推荐的实现方案。 [7赞]

- 每天晒白牙 2020-03-10 10:04:35
  老师，请教个问题，比如一个商品搜索系统，给一些商品打标签，然后支持根据商品信息和标签搜索商品，有啥方案吗？

- 每天晒白牙 2020-03-10 10:02:38
  应该输入的时候就会实时去 ES 中检索吧

- hello 2020-03-10 09:50:40
  老师，能否来一篇加餐，讲讲ES、MySQL、MongoDB、RocketMQ/Kafka、newSQL这些存储的对比，底层是基于什么原理擅长干哪些事，不擅长干哪些事？

- 墨雨 2020-03-10 08:43:59
  那一般什么时候来更新索引呢？是建立一个定时任务来更新么

- ELLIOT 2020-03-10 03:17:23
  使用match query对商品名进行匹配，然后按score进行sort排序，选择前n项