# 今日任务

1. 理解全文检索
2. Elasticsearch安装、配置
3. 掌握elasticsearch核心概念
4. 常用API练习

# 搜索引擎

## 问题？

思考电商站内如何检索商品？？使用传统的关系型数据库mysql可不可以呢？？

如果不考虑效率，当然是可以的，在小数据量的情况下，以前也是这么做的。不过随着数据量的增大，以及对搜索的要求提高，传统的关系型数据库肯定是不能胜任这个工作的，毕竟当初设计关系型数据库的初衷并不是用它们来做基于文本的“全文检索”的，正所谓“专业的事情让专业的工具去做”，这个工具就是搜索引擎。

## 概念

搜索引擎（Search Engine）是指根据一定的策略、运用特定的计算机程序从互联网上搜集信息，在对信息进行组织和处理后，为用户提供检索服务，将用户检索相关的信息展示给用户的系统。搜索引擎包括全文索引、目录索引、元搜索引擎、垂直搜索引擎、集合式搜索引擎、门户搜索引擎与免费链接列表等。

电商网站常用的就是其中的“全文检索”。

## 全文检索

毫无疑问，**Lucene**是目前最受欢迎的Java全文搜索框架，准确地说，它是一个全文检索引擎的架构，提供了完整的查询引擎和索引引擎，部分文本分析引擎。Lucene为开发人员提供了相当完整的工具包，可以非常方便地实现强大的全文检索功能。但是lucene还不能直接用于商业项目，需要二次开发才能更易于使用，目前有两个基于此的开源实现：**solr和elasticsearch**。

## Solr

Solr是一个高性能，基于Lucene的全文搜索服务器。同时对其进行了扩展，提供了比Lucene更为丰富的查询语言，同时实现了可配置、可扩展并对查询性能进行了优化，并且提供了一个完善的功能管理界面，是一款非常优秀的全文搜索引擎。

Solr也是一个独立的企业级搜索应用服务器，它对外提供类似于Web-service的API接口。用户可以通过http请求，向搜索引擎服务器提交一定格式的XML文件，生成索引；也可以通过Http Get操作提出查找请求，并得到XML格式的返回结果。

它默认是**集中式的**！！要分布式：需要**SolrCloud**。

## Elasticsearch

ElasticSearch是一个基于**Lucene**的搜索服务器。它提供了一个**分布式**多用户能力的全文搜索引擎，基于**RESTful web**接口。Elasticsearch是用Java开发的，并作为Apache许可条款下的开放源码发布，是当前**最流行的企业级搜索引擎**。设计用于**云计算中**，能够达到**实时搜索**，稳定，可靠，快速，安装使用方便。

我们建立一个网站或应用程序，并要添加搜索功能，但是想要完成搜索工作的创建是非常困难的。我们希望搜索解决方案要运行速度快，我们希望能有一个零配置和一个完全免费的搜索模式，我们希望能够简单地使用JSON通过HTTP来索引数据，我们希望我们的搜索服务器始终可用，我们希望能够从一台开始并扩展到数百台，我们要实时搜索，我们要简单的多租户，我们希望建立一个云的解决方案。因此我们利用Elasticsearch来解决所有这些问题及可能出现的更多其它问题。

## 选择

无论从目前的流行程度，使用的易用性、集群扩展的灵活性以及检索的性能，**elasticsearch**都完胜solr，所以我想这次选择并不难吧。

## 小故事

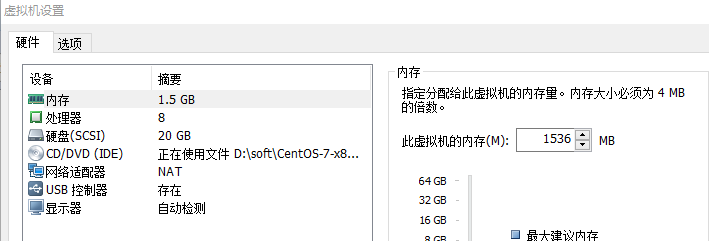
作者老婆要参加一个培训班，学烹饪的，有很多菜谱，每次查找都不方便，这个时候作者刚刚丢了饭碗，闲赋在家，他很爱自己的老婆（我的理解是反正闲着也是闲着C:\Users\ADMINI~1\AppData\Local\Temp\SGPicFaceTpBq\1604\1B6773AA.png C:\Users\ADMINI~1\AppData\Local\Temp\SGPicFaceTpBq\1604\1B679270.png），于是乎，他就用Lucene，写了一个搜索菜谱的软件并把它开源了出来，后来的事情大家都知道了，这个小软件发展到现在，就是强大的ES。

**重要提示：单身狗请尽快脱单，原来爱老婆，可以给自己带来很多好运气！**

# 入门

小提示：elasticsearch的版本更新是巨快的，这个是我见过的更新速度最快的开源软件了，真是疯狂啊。写这个文档时，当前版本是6.5.4

## 虚拟机



分配一个ip地址，安装java环境

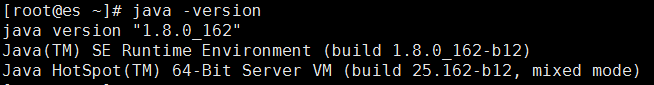
yum install lrzsz –y

上传命令：

rz



rpm -ivh jdk-8u162-linux-x64.rpm



## 安装

可以远程下载

**curl -L -O https://artifacts.elastic.co/downloads/elasticsearch/elasticsearch-6.5.4.tar.gz**

也可以直接使用资源包里提供的下载包（在相应的课件资源里）

**tar -xvf elasticsearch-6.2.4.tar.gz**

我们把解压目录移动到默认安装位置：/usr/local

**mv elasticsearch-6.2.4 /usr/local/**

**cd** **elasticsearch-6.2.4 /bin**

**./ elasticsearch**

默认启动报错：



解决办法：

最新版不允许用“root”启动，那只能创建其他用户！！

所以我们创建一个“es”用户：

useradd es

passwd es

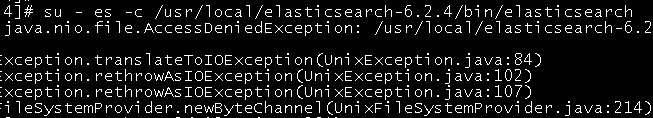
输入新密码：

（**如果想切换到其他用户中运行，教给大家一个简单的方法，利于新手操作**）

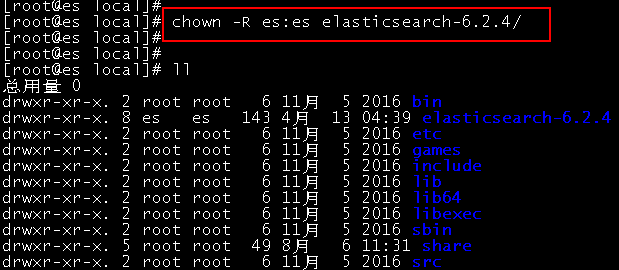
**简单的方法，推荐新手使用**

**su – es –c command**

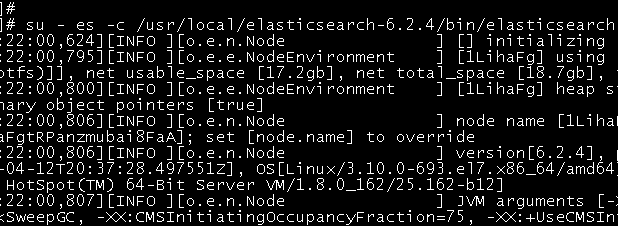
**报以下异常**



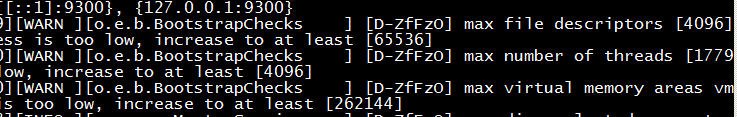
请修改文件权限：



执行命令：



会有警告信息：



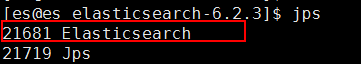
由于默认是内网访问，所以不会报错！！！！

默认启动会随机生成节点名称，如果想指定可以配置或命令行加参数：

**./elasticsearch -Ecluster.name=my\_cluster\_name -Enode.name=my\_node\_name**

## 配置

**su - es -c "/usr/local/elasticsearch-6.5.4/bin/elasticsearch -d"**

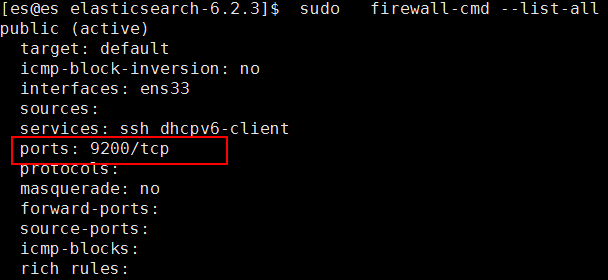


打开外部访问防火墙

**firewall-cmd --zone=public --add-port=9200/tcp --permanent**

**firewall-cmd --reload**

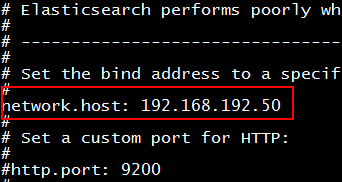
查看防火墙



修改网络配置：

**vi config/elasticsearch.yml**

**注意配置文件每条配置行前不能有空格！！切记！！**



重启服务：

**su - es -c "/usr/local/elasticsearch-6.2.4/bin/elasticsearch -d"**

启动是失败的！！！！

**注意：默认未修改网络之前启动是成功的，但只能本地访问，修改网络后，如需对外提供服务，这时就启动失败了！！**

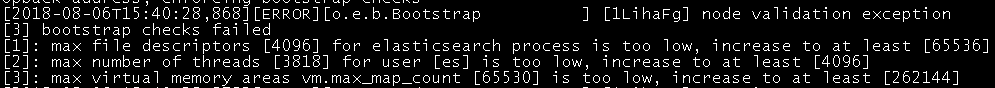
外网测试连接：



## 排错实战

查看服务器日志：

**more logs/elasticsearch.log**

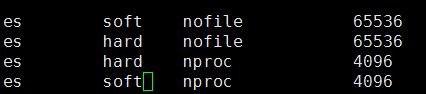


得出启动对外服务的es需要满足三个条件：

1. **最大文件数 65536**
2. **最大线程数 4096**
3. **最大虚拟内存 262144**

先修改文件上限、最大线程数

**vi /etc/security/limits.conf**



可以使用该命令查询“**ulimit -Hn**”新的配置结果！！



修改最大虚拟内存

**vi /etc/sysctl.conf**



使设置生效：**sysctl -p**



重启服务：





## 开机自启

(千万别复制，要一个字一个字敲，因为有隐藏的字符转换问题，导致错误！！)

**chmod u+x /etc/rc.d/rc.local**

**vi /etc/rc.d/rc.local**

**su - es -c “/usr/local/elasticsearch-6.5.4/bin/elasticsearch -d”**

## 安装插件

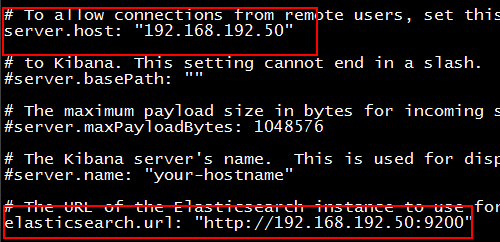
数据可视化工具

**Kibana**安装 ，同样可以在课件资源中获取安装文件。

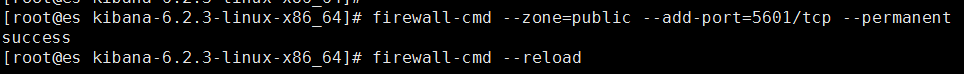
|  |
| --- |
| **wget https://artifacts.elastic.co/downloads/kibana/kibana-6.5.4-linux-x86\_64.tar.gz**  **tar -xzf kibana-6.5.4-linux-x86\_64.tar.gz**  **cd kibana-6.5.4-linux-x86\_64/** |

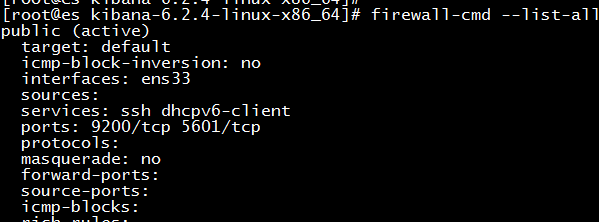
## 配置插件

**vi config/kibana.yml**



## 开防火墙

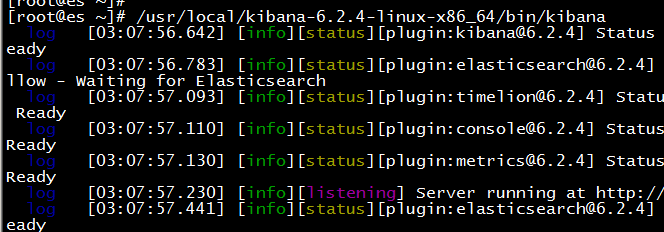




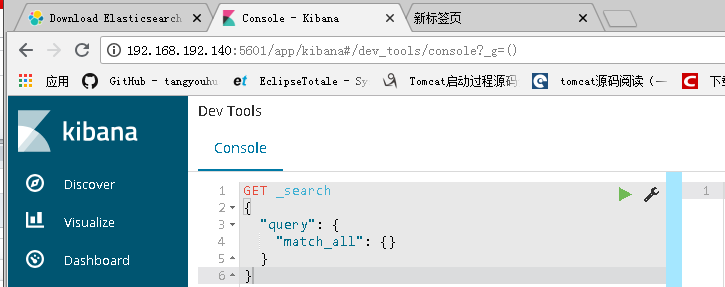
## 启动

安装到默认安装位置

**mv kibana-6.5.4-linux-x86\_64 /usr/local/**



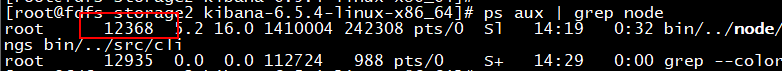
**浏览器访问**



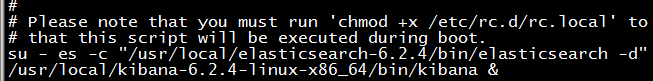
**退出：ctrl + c**

**后台进程：命令后加 &**

**注意：kibana是js写的，所以查不到进程，应该检查node**



## 开机自启



**练习环境准备完毕！！！**

# 理论概念

<https://www.elastic.co/guide/en/elasticsearch/reference/current/getting-started-concepts.html#_index>

## Near Realtime (NRT)

Es是一个**近实时**的检索平台。就是说从你添加文档到文档可以被检索的时间，正常情况下不到**一秒钟**。

## Cluster

**集群**是一个或多个节点（服务器）的集合，它们共同保存您的整个数据，并提供跨所有节点的**联合索引**和搜索功能。群集由唯一名称标识，默认情况下为“**elasticsearch**”。此名称很重要，因为如果节点设置为按名称加入群集，则该节点只能是群集的一部分。

确保不要在不同的环境中重用相同的群集名称，否则最终会导致节点加入错误的群集。例如，您可以使用logging-dev，logging-stage以及logging-prod 用于开发，测试和生产集群。

请注意，如果集群中只有一个节点，那么它是完全正常的。此外，您还可以拥有多个独立的集群，每个集群都有自己唯一的集群名称。

## Node

**节点**是作为群集一部分的单个服务器，存储数据并参与群集的索引和搜索功能。就像集群一样，节点由名称标识，默认情况下，该名称是在启动时分配给节点的随机通用唯一标识符（UUID）。如果不需要默认值，可以定义所需的任何节点名称。此名称对于管理目的非常重要，您可以在其中识别网络中哪些服务器与Elasticsearch集群中的哪些节点相对应。

可以将节点配置为按群集名称加入特定群集。默认情况下，每个节点都设置为加入一个名为cluster的集群elasticsearch，这意味着如果您在网络上启动了许多节点并且假设它们可以相互发现 - 它们将自动形成并加入一个名为elasticsearch的集群。

## Index

**索引**是具有某些类似特征的**文档**集合。例如，您可以拥有客户数据的索引，产品目录的另一个索引以及订单数据的另一个索引。索引由名称标识（必须全部小写），此名称用于在对其中的文档执行索引，搜索，更新和删除操作时引用索引。

在单个群集中，您可以根据需要定义任意数量的索引。

## Type

一种类型，曾经是索引的逻辑类别/分区，允许您在同一索引中存储不同类型的文档，例如，一种类型用于用户，另一种类型用于博客帖子。不再可能在索引中创建多个类型，并且将在更高版本中删除类型的整个概念。

**提示：类型，在6.0版本中是过渡概念了，在7.0中将彻底删除。**

## Document

文档是可以编制索引的基本信息单元。例如，您可以为单个客户提供文档，为单个产品提供另一个文档，为单个订单提供另一个文档。该文档以**JSON**（JavaScript Object Notation）表示，JSON是一种普遍存在的互联网数据交换格式。

在索引/类型中，您可以根据需要存储任意数量的文档。请注意，尽管文档实际上驻留在索引中，但实际上必须将文档编入索引/分配给索引中的类型。

## Shards & Replicas

**索引**可能存储大量可能超过单个节点的硬件限制的数据。例如，占用1TB磁盘空间的十亿个文档的单个索引可能不适合单个节点的磁盘，或者可能太慢而无法单独从单个节点提供搜索请求。

为了解决这个问题，Elasticsearch提供了将索引细分为多个称为**分片**的功能。创建索引时，只需定义所需的分片数即可。每个分片本身都是一个功能齐全且独立的“索引”，可以托管在集群中的任何节点上。

**分片很重要，主要有两个原因：**

它允许您水平拆分/缩放内容量

它允许您跨分片（可能在多个节点上）分布和**并行化**操作，从而提高性能/**吞吐量**

分片的分布方式以及如何将其文档聚合回搜索请求的机制完全由Elasticsearch管理，对用户而言是透明的。

在可以随时发生故障的网络/云环境中，非常有用，强烈建议使用故障转移机制，以防分片/节点以某种方式脱机或因任何原因消失。为此，Elasticsearch允许您将索引的分片的一个或多个副本制作成所谓的副本分片或简称副本。

**复制很重要，主要有两个原因：**

它在碎片/节点发生故障时提供**高可用性**。因此，请务必注意，副本分片永远不会在与从中复制的原始/主分片相同的节点上分配。

它允许您扩展搜索量/**吞吐量**，因为可以在所有副本上**并行**执行搜索。

总而言之，每个索引可以拆分为多个分片。索引也可以复制为零（表示没有副本）或更多次。复制后，每个索引都将具有主分片（从中复制的原始分片）和副本分片（主分片的副本）。

可以在创建索引时为每个索引定义分片和副本的数量。创建索引后，您还可以随时动态更改副本数。您可以使用\_shrink和\_split API 更改现有索引的分片数，但这不是一项简单的任务，并且预先计划正确数量的分片是最佳方法。

**默认情况下，Elasticsearch中的每个索引都分配了5个主分片和1个副本，这意味着如果群集中至少有两个节点，则索引将包含5个主分片和另外5个副本分片（1个完整副本），总计为每个索引10个分片。**

每个Elasticsearch分片都是Lucene索引。单个Lucene索引中可以包含最大数量的文档。截止LUCENE-5843，限制是2,147,483,519（= Integer.MAX\_VALUE - 128）文档。您可以使用\_cat/shards API 监控分片大小。

**提示：深刻理解这些概念是非常重要的，否则后面的操作你根本不知道自己在干嘛。**

**另外，ES的官方文档，是所有开源软件中应该是写的最详细，最贴心的了，比spring的强太多了，无论以后工作中遇到什么新的问题，建议都先去翻翻文档，一般都能解决的，其次再是其他的途径。最后一个，在面试时你能滔滔不绝地讲个没完。**

**让我们再通过一张图来总结这些概念**

## 核心概念（index）

它是一个逻辑上的概念，他是一个更大的抽象，它实际是由多个分片（shard）组成，每个shard实际上就是一个Lucene的index最小工作单元。

分片

副本

副本

分片

# Api练习

## 集群健康

GET /\_cat/health?v

绿色：集群完全可用

黄色：集群完全可用，但是没有副本备份

红色：集群部分可用，部分索引不可用

GET /\_cat/nodes?v

GET /\_cat/indices?v

## 创建index

PUT /customer?pretty

GET /\_cat/indices?v

## 添加文档

PUT /customer/\_doc/1?pretty

{

"name":"王栋"

}

## 查询文档

GET /customer/\_doc/1?pretty

## 删除index

DELETE /customer?pretty

GET /\_cat/indices?v

## 小结

命令模式：

**<REST Verb> /<Index>/<Type>/<ID>**

## 修改文档

修改之前先添加一个：

PUT /customer/\_doc/1?pretty

{

"name" : "cjl"

}

POST /customer/\_doc/1/**\_update**?pretty

{

"**doc**": { "name": "Jane Doe" }

}

修改添加字段，可以自动保留原来的结构

POST /customer/\_doc/1/\_update?pretty

{

"**doc**" : {"name":"xxx","age":100}

}

使用脚本修改

POST /customer/\_doc/1/\_update?pretty

{

"**script**" : "ctx.\_source.age += 5"

}

In the above example,**ctx.\_source** refers to the current source document that is about to be updated.

## 删除文档

DELETE /customer/\_doc/2?pretty

## 批处理

1次添加2个文档

POST /customer/\_doc/**\_bulk**?pretty

{"**index**":{"**\_id**":"1"}}

{"name": "John Doe" }

{"**index**":{"**\_id**":"2"}}

{"name": "Jane Doe" }

查看所有

GET /customer/\_search

{

"query": {

"match\_all": {}

}

}

修改、删除

POST /customer/\_doc/\_bulk?pretty

{"**update**":{"\_id":"1"}}

{"**doc**": { "name": "John Doe becomes Jane Doe" } }

{"**delete**":{"\_id":"2"}}

注意：以上操作，红色代表动作，“**index**”是**添加**的意思，也是索引一条数据的意思！！

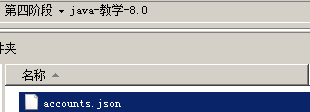
上面只是热身，下面的才是重点，电商项目中的大部分功能都是下面的查询来实现的。

# 准备数据

## 下载练习文档

<https://github.com/elastic/elasticsearch/blob/master/docs/src/test/resources/accounts.json?raw=true>

或者在课件资料里获取



## 添加文档集合

先上传文档到虚拟机



**注意文档的完整性，否则会出错！！！！！！！！！！！！！！！！！**

远程添加到es中

**curl -H "Content-Type: application/json" -XPOST "192.168.57.20:9200/bank/\_doc/\_bulk?pretty&refresh" --data-binary "@accounts.json"**

查询结果

**curl "192.168.192.50:9200/\_cat/indices?v"**

# 基本查询

## 请求参数

GET /bank/\_search?q=\*&sort=account\_number:desc&pretty

|  |
| --- |
| * **took – time** in milliseconds for Elasticsearch to execute the search * **timed\_out** – tells us if the search timed out or not * **\_shards** – tells us how many shards were searched, as well as a count of the successful/failed searched shards * **hits** – search results * **hits.total** – total number of documents matching our search criteria * **hits.hits** – actual array of search results (defaults to first 10 documents) * **hits.sort** - sort key for results (missing if sorting by score) * **hits.\_score** and **max\_score** - ignore these fields for now |

## 请求体

|  |
| --- |
| GET /bank/\_search  {  "query": { "match\_all": {} },  "**sort**": [  { "account\_number": "desc" }  ]  } |

**注意点: es查询结果返回后，不再需要和服务器保持任何的连接，不像传统的关系型数据库，在查询结果返回后，仍然需要保持连接！！**

# DSL

Elasticsearch提供基于JSON的完整查询DSL（**域特定语言**）来定义查询。是ES的主要查询语法，由两种类型的子句组成：

**叶子查询**

叶查询子句中寻找一个特定的值在某一特定领域，如 **match**，**term**或 **range**查询。这些查询可以单独使用。

**复合查询**

复合查询包装其他叶子或复合查询，用于以逻辑方式（例如bool或dis\_max查询）组合多个查询 ，或者更改其行为（例如 constant\_score查询）。

## 分页查询

默认第一页，10条，这里改成1条

GET /bank/\_search

{

"query": { "match\_all": {} },

"size": 1

}

查询第二页的数据：

GET /bank/\_search

{

"query": { "match\_all": {} },

"from": 10,

"size": 10

}

查本银行存款最多的前10位：

GET /bank/\_search

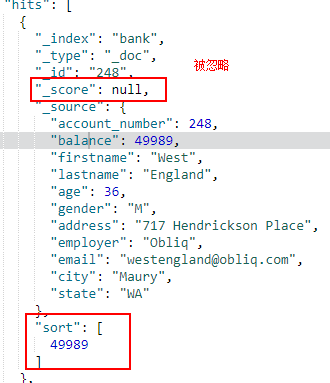
{

"query": { "match\_all": {} },

"sort": { "balance": { "order": "desc" } }

}

注意点：如果在查询中指定字段排序，那么默认的匹配度“\_score”将会被忽略！



## 结果过滤

只返回指定的字段值

GET /bank/\_search

{

"query": { "match\_all": {} },

"\_source": ["account\_number", "balance"]

}

## match\_all

查询所有记录，略，参考前面例子

## 匹配查询

查询指定账户：

GET /bank/\_search

{

"query": { "match": { "account\_number": 20 } }

}

模糊查询：

GET /bank/\_search

{

"query": { "match": { "address": "mill" } }

}

GET /bank/\_search

{

"query": { "match": { "address": "mill lane" } }

}

## 短语查询

按照短语为基本单位，进行模糊匹配

GET /bank/\_search

{

"query": { "match\_phrase": { "address": "mill lane" } }

}

## 逻辑查询

组合查询：可以将多个小的“match”组合一个更大的查询条件

逻辑“与“

GET /bank/\_search

{

"query": {

"bool": {

"**must**": [

{ "match": { "address": "mill" } },

{ "match": { "address": "lane" } }

]

}

}

}

逻辑“或“：should

GET /bank/\_search

{

"query": {

"bool": {

"**should**": [

{ "match": { "address": "mill" } },

{ "match": { "address": "lane" } }

]

}

}

}

逻辑“非”

GET /bank/\_search

{

"query": {

"bool": {

"**must\_not**": [

{ "match": { "address": "mill" } },

{ "match": { "address": "lane" } }

]

}

}

}

## 逻辑组合

查询所有40岁不在“ID”州的账户信息

GET /bank/\_search

{

"query": {

"bool": {

"**must**": [

{ "match": { "age": "40" } }

],

"**must\_not**": [

{ "match": { "state": "ID" } }

]

}

}

}

## 过滤filter

查询余额在20000到30000之间的，

关键是用范围过滤时，并不需要考虑匹配的问题“score”

GET /bank/\_search

{

"query": {

"bool": {

"must": { "match\_all": {} },

"filter": {

"range": {

"balance": {

"gte": 20000,

"lte": 30000

}

}

}

}

}

}

## 聚合查询

按照“州”来分组

GET /bank/\_search

{

"size": 0,

"aggs": {

"group\_by\_state": {

"terms": {

"field": "state.keyword"

}

}

}

}

按照“州”分组，然后按每个“州”的余额平均值再聚合

GET /bank/\_search

{

"size": 0,

"aggs": {

"group\_by\_state": {

"terms": {

"field": "state.keyword"

},

"aggs": {

"average\_balance": {

"avg": {

"field": "balance"

}

}

}

}

}

}

按“余额”降序来排序

GET /bank/\_search

{

"size": 0,

"aggs": {

"group\_by\_state": {

"terms": {

"field": "state.keyword",

**"order": {**

**"average\_balance": "desc"**

**}**

},

"aggs": {

"average\_balance": {

"avg": {

"field": "balance"

}

}

}

}

}

}

按年龄分组，再按性别分组，最后计算各组余额平均值

GET /bank/\_search

{

"size": 0,

"aggs": {

"group\_by\_age": {

"range": {

"field": "age",

"ranges": [

{

"from": 20,

"to": 30

},

{

"from": 30,

"to": 40

},

{

"from": 40,

"to": 50

}

]

},

"aggs": {

"group\_by\_gender": {

"terms": {

"field": "gender.keyword"

},

"aggs": {

"average\_balance": {

"avg": {

"field": "balance"

}

}

}

}

}

}

}

}

# 全文检索

根据提供的**关键字**来匹配符合条件的文档对象。是需要重点掌握的查询方式。

**match** ：用于执行全文查询的标准查询，包括模糊匹配和短语或邻近查询。

**match\_phrase** ：与match查询类似，但用于匹配精确短语或单词邻近匹配。

**match\_phrase\_prefix** ：像match\_phrase查询一样，但是在最后一个单词上进行通配符搜索。

**multi\_match**：match查询的多字段版本。

**common\_terms** ：通用词查询，可以将一些停用词加入查询，增加查询精度。

**query\_string** ：支持紧凑的Lucene 查询字符串语法，允许您在单个查询字符串中指定AND | OR | NOT条件和多字段搜索。仅限高级用户。

**simple\_query\_string**：一种更简单，更健壮的query\_string语法版本，适合直接向用户公开。

**掌握前2种即可，查看前面练习，后面的需要时查看文档学习。**

# Term查询

相较于模糊匹配的全文检索，term查询是一种**精确查询**。这些查询通常用于结构化数据，如数字，日期和枚举，而不是全文字段。

**term** ：查找包含指定字段中指定的确切术语的文档。

**terms** ：查找包含指定字段中指定的任何确切术语的文档。

**terms\_set** ：查找与一个或多个指定条款匹配的文档。必须匹配的术语数量取决于指定的最小值应匹配字段或脚本。

**range** ：查找指定字段包含指定范围内的值（日期，数字或字符串）的文档。

等等。。。。。。。。。。。。。。。。。。。。。。。。。。。。。。。。

例如：

GET /\_search

{

"query": {

"terms" : { "user" : ["cjl", "elasticsearch"]}

}

}