

# 一，Elasticsearch的介绍

Elasticsearch是一个基于Lucene的搜索引擎，业内简称ES。它提供了一个分布式多用户能力的全文搜索引擎，是用Java开发的，基于RESTful风格的api操作规范（即http请求接口）的分布式搜索引擎（或者说数据库）并且能够达到准实时搜索

## 1.1 什么是搜索引擎？

比如百度：就是在任何场景下，找寻你想要的信息，用户输入想要搜索的关键词，然后期望找到这个关键词相关的信息

## 1.2 用传统的数据库做搜索会怎么样？

MySQL作为传统的关系型数据库，是当下Web应用开发中最流行的关系型数据库，我能否直接用MySQL实现搜索引擎？

这里我们来举个比较实际的例子，看一下到底MySQL适不适合做搜索引擎：

职位 ID	职位名称	描述	所属公司
1	python 全栈开发		
2	java 开发		
3	爬虫工程师		
...			

查询语法： `select * from demo where name like "%python%"`

此时用户检索词：python 系全栈开发|

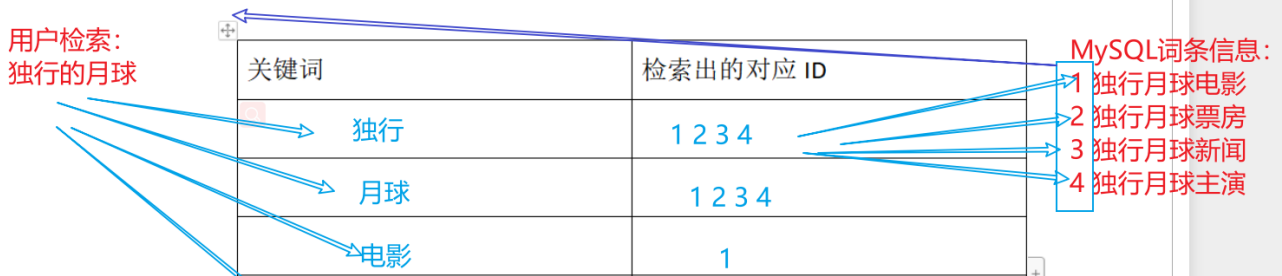
基于数据库去做搜索会有以下几个缺点：

- 1. 会进行全文扫描。并且——比对文本值，如果指定字符的文本值过长或者数据过多，扫描时间会特别长，无法实现高效率
- 2. 用户检索词为“Python系全栈开发”，由于中间插入其他词汇，导致检索结果受到干扰

所以数据库实现搜索，性能比较差，但是呢，我们一般使用ES实现自己的站内搜索，使用Mysql做原始数据的存储，然后在MySQL基础之前部署我们的ES的中间件来实现搜索引擎

## 1.3 什么是全文检索？

全文检索的核心在于倒排索引



以上过程就是ES的全文检索，如果数据有100万条，倒排索引中未必扫描100万次，就算倒排索引扫描100万次，跟数据库相对，ES性能更高，因为数据库中要匹配是否包含关键词，ES的倒排索引只要比对词条是否一样

## 1.4 什么叫Lunece?

Lucene是一个开源的信息检索工具包（类似于Java api），它包含了很多极优算法，索引结构、读写索引工具、排序索引等功能，而Elasticsearch底层是基于这些包，对其进行了扩展，提供了比Lucene更为丰富简洁的查询语言，可以非常方便的通过Elasticsearch的HTTP接口与底层Lucene进行逻辑交互。

一句话概括: Elasticsearch是Lucene面向企业搜索应用的扩展，极大的缩短研发周期，因为Elasticsearch是在它基础上扩展的应用程序,提供了简单易用的restful接口，可以直接拿来使用

## 二， ES和kibana的安装流程

### 安装过程中路径不能有任何中文及特殊字符

#### (1) ElasticSearch8.2.3 安装

- 软件包下载
  - <https://www.elastic.co/cn/downloads/elasticsearch>  
(<https://www.elastic.co/cn/downloads/elasticsearch>)
- 配置&使用
  - es&kibana都无需额外配置，下载解压后即可使用
- elasticSearch文件目录

■

> 下载 > elasticsearch-8.3.3 >

名称	修改日期	类型
bin	2022/07/23 19:37	文件夹
config	2022/08/09 17:01	文件夹
data	2022/08/09 17:40	文件夹
jdk	2022/07/23 19:37	文件夹
lib	2022/07/23 19:37	文件夹
logs	2022/08/09 17:15	文件夹
modules	2022/07/23 19:37	文件夹
plugins	2022/07/23 19:33	文件夹
LICENSE.txt	2022/07/23 19:29	文本文档
NOTICE.txt	2022/07/23 19:33	文本文档
README.asciidoc	2022/07/23 19:29	ASCIIDOC 文件

- elasticsearch启动方式 上述目录中，进入bin目录，双击 elasticsearch.bat 脚本，即可启动 es数据库

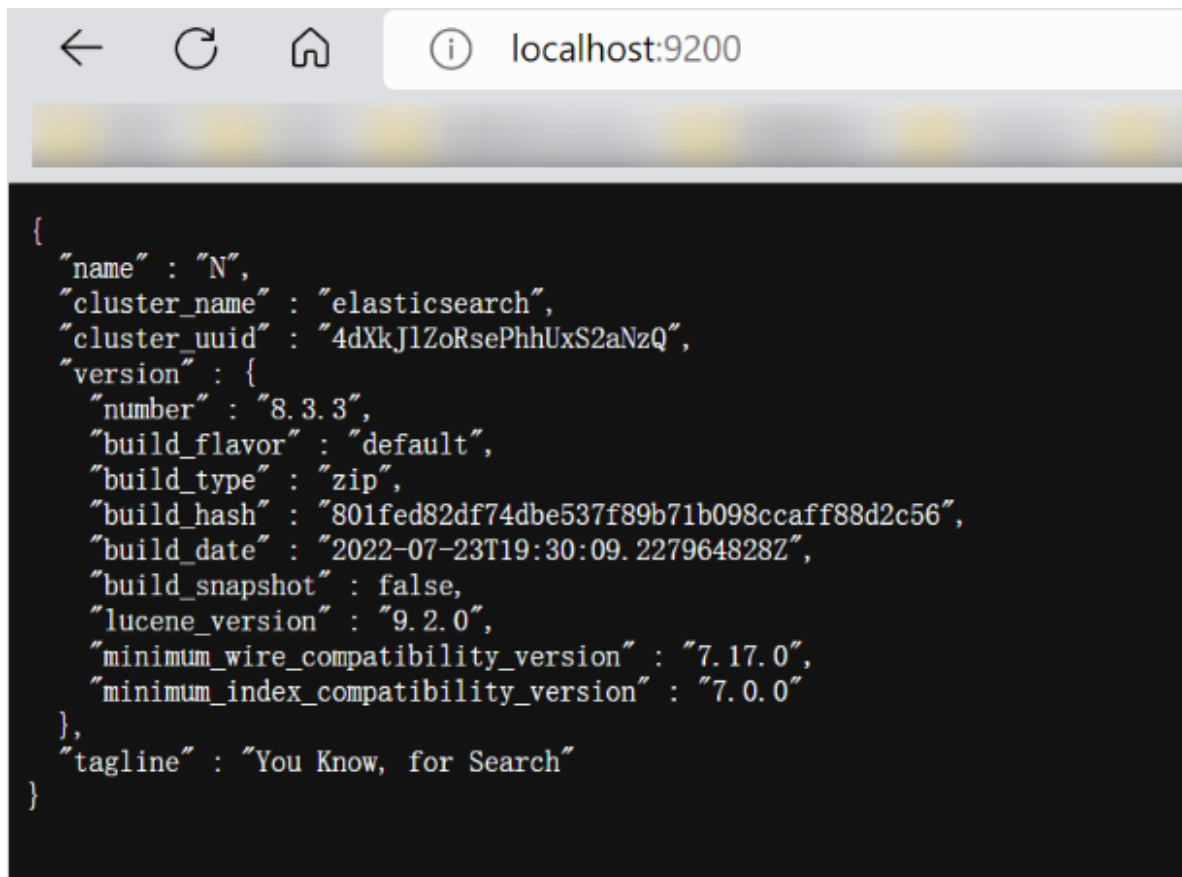
■

> 下载 > elasticsearch-8.3.3 > bin

名称	修改日期	类型	大小
elasticsearch	2022/07/23 19:29	.	1 KB
elasticsearch.bat	2022/07/23 19:29	Windows 批处理文件	1 KB
elasticsearch-certgen	2022/07/23 19:32	.	1 KB
elasticsearch-certgen.bat	2022/07/23 19:32	Windows 批处理文件	1 KB
elasticsearch-certutil	2022/07/23 19:32	.	1 KB
elasticsearch-certutil.bat	2022/07/23 19:32	Windows 批处理文件	1 KB

双击启动

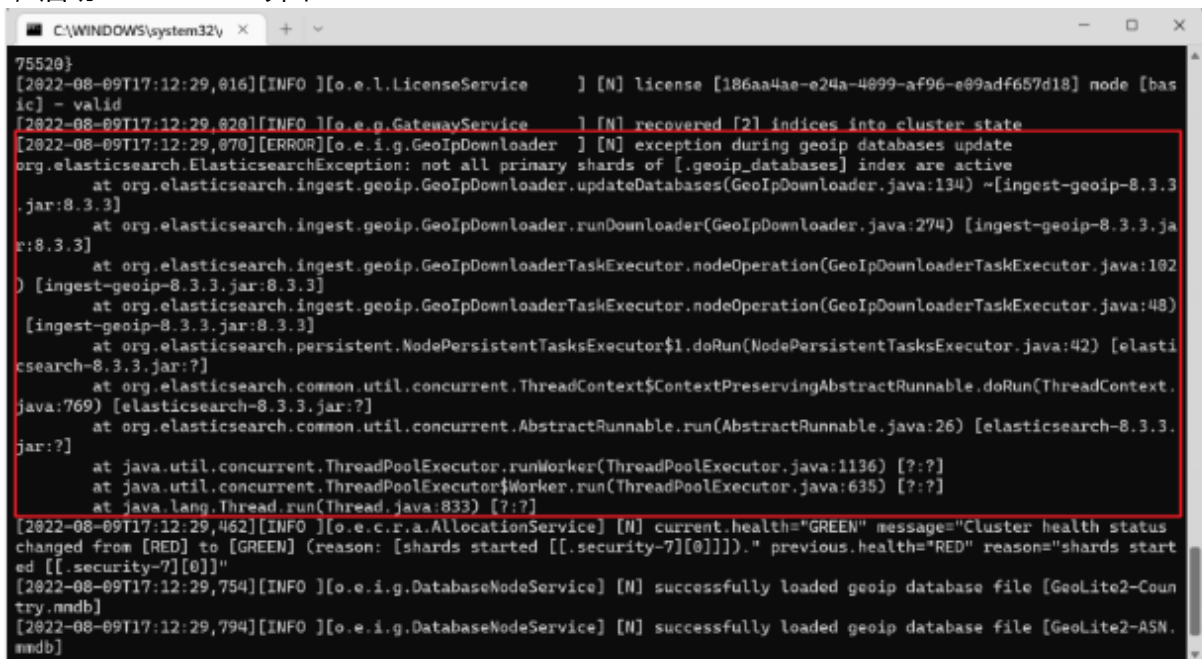
- 访问es数据库服务
  - 启动成功且无异常后，网页访问 127.0.0.1:9200，即可看到如下页面,如若无法正常访问，请查看后面步骤



```
{
  "name" : "N",
  "cluster_name" : "elasticsearch",
  "cluster_uuid" : "4dXkJlZoRsePhhUxS2aNzQ",
  "version" : {
    "number" : "8.3.3",
    "build_flavor" : "default",
    "build_type" : "zip",
    "build_hash" : "801fed82df74dbe537f89b71b098ccaff88d2c56",
    "build_date" : "2022-07-23T19:30:09.227964828Z",
    "build_snapshot" : false,
    "lucene_version" : "9.2.0",
    "minimum_wire_compatibility_version" : "7.17.0",
    "minimum_index_compatibility_version" : "7.0.0"
  },
  "tagline" : "You Know, for Search"
}
```

- elasticSearch异常&解决

- 1, 启动elasticSearch异常



```
75520}
[2022-08-09T17:12:29,016][INFO ][o.e.l.LicenseService ] [N] license [186aa4ae-e24a-4099-af96-e09adf657d18] node [bas
ic] - valid
[2022-08-09T17:12:29,020][INFO ][o.e.g.GatewayService ] [N] recovered [2] indices into cluster state
[2022-08-09T17:12:29,070][ERROR][o.e.i.g.GeoIpDownloader ] [N] exception during geoip databases update
org.elasticsearch.ElasticsearchException: not all primary shards of [.geoip_databases] index are active
    at org.elasticsearch.ingest.geoip.GeoIpDownloader.updateDatabases(GeoIpDownloader.java:134) ~[ingest-geoip-8.3.3
.jar:8.3.3]
    at org.elasticsearch.ingest.geoip.GeoIpDownloader.runDownloader(GeoIpDownloader.java:274) [ingest-geoip-8.3.3.ja
r:8.3.3]
    at org.elasticsearch.ingest.geoip.GeoIpDownloaderTaskExecutor.nodeOperation(GeoIpDownloaderTaskExecutor.java:102
) [ingest-geoip-8.3.3.jar:8.3.3]
    at org.elasticsearch.ingest.geoip.GeoIpDownloaderTaskExecutor.nodeOperation(GeoIpDownloaderTaskExecutor.java:48)
[ingest-geoip-8.3.3.jar:8.3.3]
    at org.elasticsearch.persistent.NodePersistentTasksExecutor$1.doRun(NodePersistentTasksExecutor.java:42) [elasti
csearch-8.3.3.jar:?]
    at org.elasticsearch.common.util.concurrent.ThreadContext$ContextPreservingAbstractRunnable.doRun(ThreadContext.
java:769) [elasticsearch-8.3.3.jar:?]
    at org.elasticsearch.common.util.concurrent.AbstractRunnable.run(AbstractRunnable.java:26) [elasticsearch-8.3.3.
jar:?]
    at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1136) [?:?]
    at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:635) [?:?]
    at java.lang.Thread.run(Thread.java:833) [?:?]
[2022-08-09T17:12:29,462][INFO ][o.e.c.r.a.AllocationService] [N] current.health="GREEN" message="Cluster health status
changed from [RED] to [GREEN] (reason: [shards started [[.security-7][0]])]" previous.health="RED" reason="shards start
ed [[.security-7][0]]"
[2022-08-09T17:12:29,754][INFO ][o.e.i.g.DatabaseNodeService] [N] successfully loaded geoip database file [GeoLite2-Coun
try.mmdb]
[2022-08-09T17:12:29,794][INFO ][o.e.i.g.DatabaseNodeService] [N] successfully loaded geoip database file [GeoLite2-ASN.
mmdb]
```

- 原因：启动时会去更新地图的一些数据库，这里直接禁掉即可
- 解决：修改配置文件 config/elasticsearch.yml 添加以下配置：
  - ingest.geoip.downloader.enabled: false

```
# Allow HTTP API connections from anywhere
# Connections are encrypted and require user authentication
http.host: 0.0.0.0

# Allow other nodes to join the cluster from anywhere
# Connections are encrypted and mutually authenticated
#transport.host: 0.0.0.0

ingest.geoip.downloader.enabled: false
```

## (2) Kibana[ES可视化工具] 安装

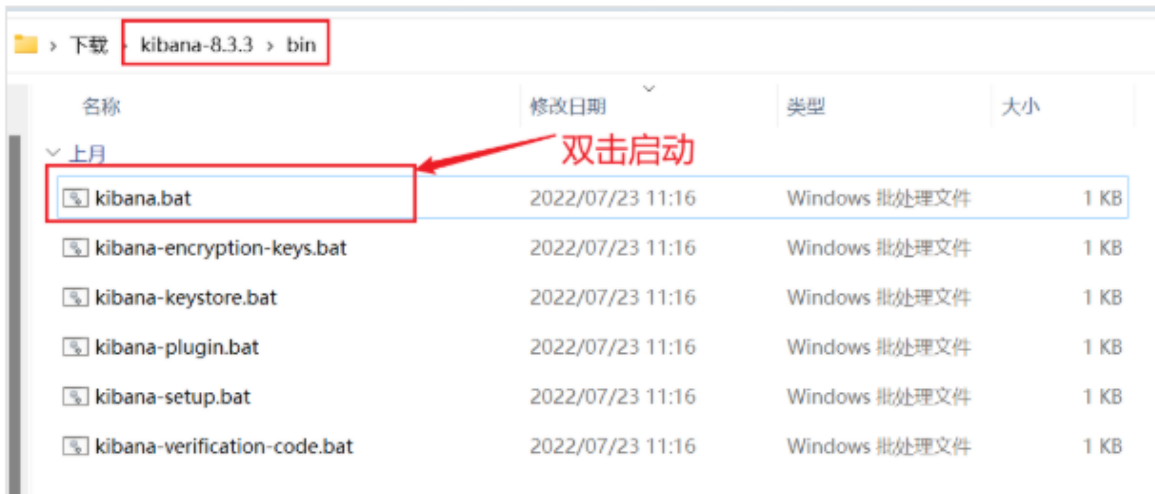
- 下载链接: <https://www.elastic.co/cn/downloads/kibana> (<https://www.elastic.co/cn/downloads/kibana>)
- kibana文件目录

■ 下载 > kibana-8.3.3 >

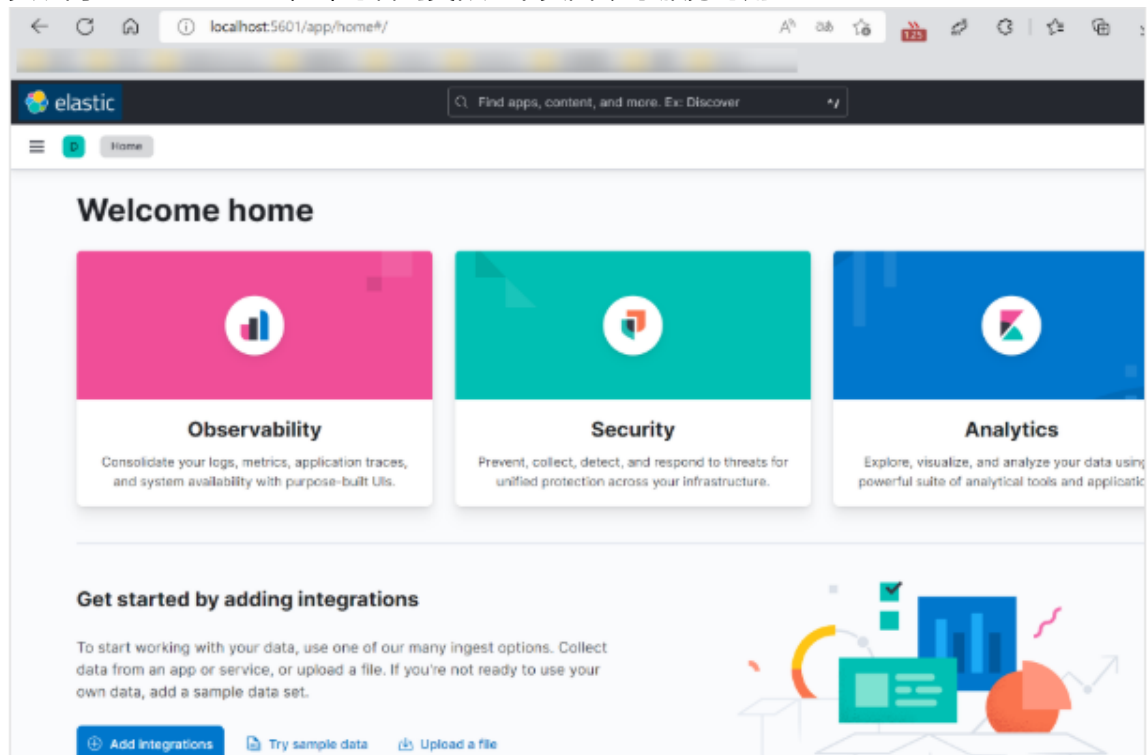
名称	修改日期	类型
bin	2022/07/23 11:17	文件夹
config	2022/07/23 11:16	文件夹
data	2022/08/09 17:27	文件夹
logs	2022/07/23 11:16	文件夹
node	2022/07/23 11:17	文件夹
node_modules	2022/07/23 11:16	文件夹
plugins	2022/07/23 11:16	文件夹
src	2022/07/23 11:16	文件夹
x-pack	2022/07/23 11:16	文件夹
.j18nrc.json	2022/07/23 11:16	JSON 文件
LICENSE.txt	2022/07/23 11:16	文本文档
NOTICE.txt	2022/07/23 11:16	文本文档
package.json	2022/07/23 11:16	JSON 文件
README.txt	2022/07/23 11:16	文本文档

- kibana启动方式:

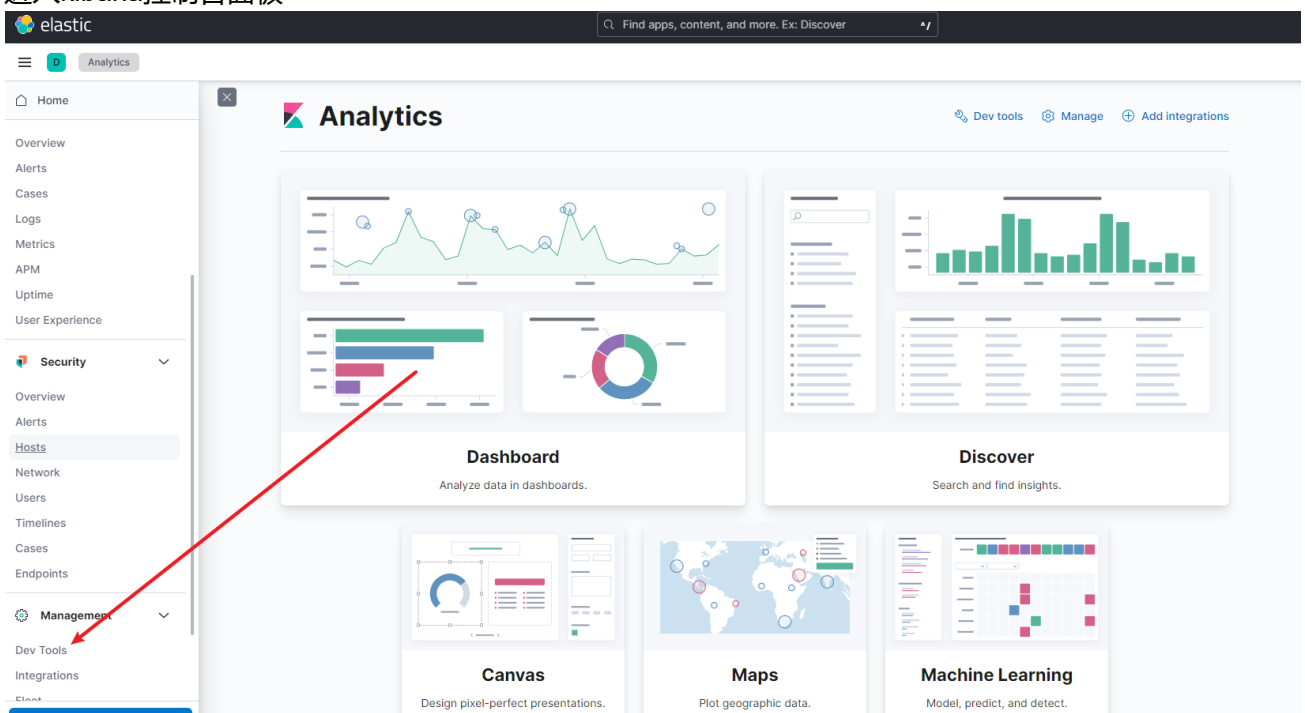
- 与es数据库启动方式一直, 进入对应的bin目录, 双击 kibana.bat 脚本, 即可启动



- 访问kibana服务
  - 网页访问 127.0.0.1:5601, 即可看到类似如下页面 表示服务可用



- 进入kibana控制台面板



## 三, Elasticsearch的基本结构

**3.1 Elasticsearch是基于Lucene的全文检索库, 本质也是存储数据, 很多概念与我们的MySQL类似**

与mysql的对比关系:

索引 (indices) -----Databases 数据库

类型 (type) -----Table 数据表

文档 (Document) -----Row 行 (数据记录)

字段 (Field) -----Columns 列 (字段名)

mappings ----- 数据结构

### 详细说明:

- 索引 (indices): indices是index的复数, 代表许多的索引,每个索引 (即数据库) 的名字必须小写
- 类型 (type) : ES是没有表的概念的, 在document中, 实际上将type作为一个document的field来存储, 所以一个index中的多个type是放在一起存储的, 因此才有了type是模拟mysql中的table概念
- 文档 (document) : 存入索引库原始的数据。比如每一条商品信息, 就是一个文档
- 字段 (field) : 文档中的属性(字段)

## 3.2, 语法

创建索引的请求格式:

- 请求方式: PUT
- 请求路径: /索引库名
- 请求参数: json格式:

以下语法为ES原生语法, 使用kibana操作:

In [1]:

```
# 2.1 创建索引
PUT /hotel # hotel索引名
{
  "mappings": { # mappings设置映射关系 相当于MySQL中创建表结构
    "properties": { # propertie 开始组建属性字段
      "name": {
        "type": "text" # type:给当前字段设为什么类型
      },
      "city": {
        "type": "keyword" # keyword 不会被分词
      },
      "price": {
        "type": "double" # 浮点型
      }
    }
  }
}
```

In [ ]:

```
# 2.2 写入文档数据
POST /hotel/_doc/001
{"name": "酒店",
 "city": "北京",
 "price": 67
}
```

In [7]:

```
# Get请求可以帮我们查看索引信息，格式：GET /索引库名
GET /hotel/_search

# 2.3 根据文档_id 001检索文档
GET /hotel/_doc/001
```

In [ ]:

```
# 2.4 根据普通字段搜索文档
GET /hotel/_search
{
  "query": {
    "term": {
      "name": {
        "value": "酒"
      }
    }
  }
}
```

In [ ]:

```
# 2.5 根据文本字段搜索文档
GET /hotel/_search
{
  "query": {
    "match": {
      "city": "上海"
    }
  }
}
```

In [ ]:

```
# 2.6 批量写入数据
PUT /hotel/_bulk
{"index": {"_id": 22}}
{"name": "酒店酒店", "city": "北京", "price": 32}
{"index": {"_id": 33}}
{"name": "酒店酒店", "city": "长沙", "price": 312}
```



In [ ]:

```
# 2.7根据条件删除文档（数据）
PUT /hotel/_delete_query
{
  "query": {
    "match": {
      "city": "上海"
    }
  }
}
```

In [ ]:

```
# 2.8 删除索引
DELETE hotel
```

### 三，Python API实践操作

以下代码的运行需要开启Elasticsearch服务

In [1]:

```
# 连接ES
from elasticsearch import Elasticsearch # pip install elasticsearch
es = Elasticsearch("http://localhost:9200")
```

In [17]:

```
# 删除索引
r = es.indices.delete(index="py_index01", ignore = 404)
print(r)
```

```
{'error': {'root_cause': [{'type': 'index_not_found_exception', 'reason': 'no such index [py_index01]', 'resource.type': 'index_or_alias', 'resource.id': 'py_index01', 'index_uuid': '_na_', 'index': 'py_index01'}], 'type': 'index_not_found_exception', 'reason': 'no such index [py_index01]', 'resource.type': 'index_or_alias', 'resource.id': 'py_index01', 'index_uuid': '_na_', 'index': 'py_index01'}, 'status': 404}
```

C:\Users\EDY\AppData\Local\Temp\ipykernel\_21284\714878680.py:2: DeprecationWarning: Passing transport options in the API method is deprecated. Use 'Elasticsearch.options()' instead.

```
r = es.indices.delete(index="py_index01", ignore = 404)
```

In [18]:

```
# 2 创建索引
r = es.indices.create(index='py_index01', ignore = 404)
print(r)
```

C:\Users\EDY\AppData\Local\Temp\ipykernel\_21284\3515760696.py:2: DeprecationWarning: Passing transport options in the API method is deprecated. Use 'Elasticsearch.options()' instead.

```
r = es.indices.create(index='py_index01', ignore = 404)
```

```
{'acknowledged': True, 'shards_acknowledged': True, 'index': 'py_index01'}
```

In [19]:

```
# 3, 插入数据
body = {
    "name": "lisi",
    "age": "13",
    "city": "深圳",
    "hobbies": "reading, singing, dancing"
}
es.index(index='py_index01', id=2, body=body)
```

C:\Users\EDY\AppData\Local\Temp\ipykernel\_21284\3309328189.py:8: DeprecationWarning: The 'body' parameter is deprecated and will be removed in a future version. Instead use the 'document' parameter. See <https://github.com/elastic/elasticsearch-py/issues/1698> (<https://github.com/elastic/elasticsearch-py/issues/1698>) for more information

```
es.index(index='py_index01', id=2, body=body)
```

Out[19]:

```
ObjectApiResponse({'_index': 'py_index01', '_id': '2', '_version': 1, 'result': 'created', '_shards': {'total': 2, 'successful': 1, 'failed': 0}, '_seq_no': 0, '_primary_term': 1})
```

In [20]:

```
# 3, 插入第二条数据
body = {
    "name": "wangwu",
    "age": "23",
    "city": "上海",
    "hobbies": "吃饭, 睡觉"
}
es.index(index='py_index01', body=body)
```

C:\Users\EDY\AppData\Local\Temp\ipykernel\_21284\1181194000.py:8: DeprecationWarning: The 'body' parameter is deprecated and will be removed in a future version. Instead use the 'document' parameter. See <https://github.com/elastic/elasticsearch-py/issues/1698> (<https://github.com/elastic/elasticsearch-py/issues/1698>) for more information

```
es.index(index='py_index01', body=body)
```

Out[20]:

```
ObjectApiResponse({'_index': 'py_index01', '_id': 'URcSYYcBxKJpPocP5Fab', '_version': 1, 'result': 'created', '_shards': {'total': 2, 'successful': 1, 'failed': 0}, '_seq_no': 1, '_primary_term': 1})
```

In [21]:

```
# 3, 插入第三条数据
body = {
    "name": "luce",
    "age": "23",
    "city": "北京",
    "hobbies": "篮球"
}
es.index(index='py_index01', body =body )
```

C:\Users\EDY\AppData\Local\Temp\ipykernel\_21284\108888719.py:8: DeprecationWarning: The 'body' parameter is deprecated and will be removed in a future version. Instead use the 'document' parameter. See <https://github.com/elastic/elasticsearch-py/issues/1698> (<https://github.com/elastic/elasticsearch-py/issues/1698>) for more information

```
es.index(index='py_index01', body =body )
```

Out[21]:

```
ObjectApiResponse({'_index': 'py_index01', '_id': 'UhcSYcBxKJpPocP6lbo', '_version': 1, 'result': 'created', '_shards': {'total': 2, 'successful': 1, 'failed': 0}, '_seq_no': 2, '_primary_term': 1})
```

In [22]:

```
# 4, match_all 查询所有
query = {
    "query": {
        "match_all": {}
    }
}
r = es.search(index='py_index01', body = query)
print(r)
```

```
{'took': 0, 'timed_out': False, '_shards': {'total': 1, 'successful': 1, 'skipped': 0, 'failed': 0}, 'hits': {'total': {'value': 3, 'relation': 'eq'}, 'max_score': 1.0, 'hits': [{'_index': 'py_index01', '_id': '2', '_score': 1.0, '_source': {'name': 'lisi', 'age': '13', 'city': '深圳', 'hobbies': 'reading, singing, dancing'}}, {'_index': 'py_index01', '_id': 'URcSYcBxKJpPocP5Fab', '_score': 1.0, '_source': {'name': 'wangwu', 'age': '23', 'city': '上海', 'hobbies': '吃饭, 睡觉'}}, {'_index': 'py_index01', '_id': 'UhcSYcBxKJpPocP6lbo', '_score': 1.0, '_source': {'name': 'luce', 'age': '23', 'city': '北京', 'hobbies': '篮球'}}]}}
```

C:\Users\EDY\AppData\Local\Temp\ipykernel\_21284\2217582170.py:7: DeprecationWarning: The 'body' parameter is deprecated and will be removed in a future version. Instead use individual parameters.

```
r = es.search(index='py_index01', body = query)
```

In [24]:

```
# 5, 根据term查询
# term主要用于精确匹配哪些值
query = {
    "query": {
        "term": {
            "age": "13"
        }
    }
}
es.search(index='py_index01', body = query)
```

C:\Users\EDY\AppData\Local\Temp\ipykernel\_21284\3048146489.py:10: DeprecationWarning: The 'body' parameter is deprecated and will be removed in a future version. Instead use individual parameters.

```
es.search(index='py_index01', body = query)
```

Out[24]:

```
ObjectApiResponse({'took': 0, 'timed_out': False, '_shards': {'total': 1, 'successful': 1, 'skipped': 0, 'failed': 0}, 'hits': {'total': {'value': 1, 'relation': 'eq'}, 'max_score': 0.9808291, 'hits': [{'_index': 'py_index01', '_id': '2', '_score': 0.9808291, '_source': {'name': 'lisi', 'age': '13', 'city': '深圳', 'hobbies': 'reading, singing, dancing'}}]})
```

In [27]:

```
# 6 根据terms查询 # 指定多个字段值查询
query = {
    "query": {
        "terms": {
            "city": ["深", "北"] # text
        }
    }
}
es.search(index='py_index01', body = query)
```

C:\Users\EDY\AppData\Local\Temp\ipykernel\_21284\260086827.py:9: DeprecationWarning: The 'body' parameter is deprecated and will be removed in a future version. Instead use individual parameters.

```
es.search(index='py_index01', body = query)
```

Out[27]:

```
ObjectApiResponse({'took': 1, 'timed_out': False, '_shards': {'total': 1, 'successful': 1, 'skipped': 0, 'failed': 0}, 'hits': {'total': {'value': 2, 'relation': 'eq'}, 'max_score': 1.0, 'hits': [{'_index': 'py_index01', '_id': '2', '_score': 1.0, '_source': {'name': 'lisi', 'age': '13', 'city': '深圳', 'hobbies': 'reading, singing, dancing'}}, {'_index': 'py_index01', '_id': 'UhcSYycBxKJpPocP6lbo', '_score': 1.0, '_source': {'name': 'luce', 'age': '23', 'city': '北京', 'hobbies': '篮球'}}]})
```

In [29]:

```
# 7, 根据name
query = {
    "query": {
        "terms": {
            "name": ["lisi", "luce"]
        }
    }
}
es.search(index='py_index01', body = query)
```

C:\Users\EDY\AppData\Local\Temp\ipykernel\_21284\2830547998.py:9: DeprecationWarning: The 'body' parameter is deprecated and will be removed in a future version. Instead use individual parameters.

```
es.search(index='py_index01', body = query)
```

Out[29]:

```
ObjectApiResponse({'took': 2, 'timed_out': False, '_shards': {'total': 1, 'successful': 1, 'skipped': 0, 'failed': 0}, 'hits': {'total': {'value': 2, 'relation': 'eq'}, 'max_score': 1.0, 'hits': [{'_index': 'py_index01', '_id': '2', '_score': 1.0, '_source': {'name': 'lisi', 'age': '13', 'city': '深圳', 'hobbies': 'reading, singing, dancing'}}, {'_index': 'py_index01', '_id': 'UhcSYcBxKJpPocP6lbo', '_score': 1.0, '_source': {'name': 'luce', 'age': '23', 'city': '北京', 'hobbies': '篮球'}}]})
```

In [31]:

```
# 8, multi_match    在match查询的基础上可同时搜索多个字段 在多个字段中同时查一个
query = {
    "query": {
        "multi_match": {
            "query": "reading",
            "fields": ["name", "hobbies"]
        }
    }
}
es.search(index='py_index01', body = query)
```

C:\Users\EDY\AppData\Local\Temp\ipykernel\_21284\284143556.py:10: DeprecationWarning: The 'body' parameter is deprecated and will be removed in a future version. Instead use individual parameters.

```
es.search(index='py_index01', body = query)
```

Out[31]:

```
ObjectApiResponse({'took': 9, 'timed_out': False, '_shards': {'total': 1, 'successful': 1, 'skipped': 0, 'failed': 0}, 'hits': {'total': {'value': 1, 'relation': 'eq'}, 'max_score': 0.9808291, 'hits': [{'_index': 'py_index01', '_id': '2', '_score': 0.9808291, '_source': {'name': 'lisi', 'age': '13', 'city': '深圳', 'hobbies': 'reading, singing, dancing'}}]})
```

In [33]:

```
# 9, wildcard 查询 模糊查询
query = {
    "query": {
        "wildcard": {
            "name": "lu*"
        }
    }
}
es.search(index='py_index01', body = query)
```

C:\Users\EDY\AppData\Local\Temp\ipykernel\_21284\783237115.py:9: DeprecationWarning: The 'body' parameter is deprecated and will be removed in a future version. Instead use individual parameters.

```
es.search(index='py_index01', body = query)
```

Out[33]:

```
ObjectApiResponse({'took': 0, 'timed_out': False, '_shards': {'total': 1, 'successful': 1, 'skipped': 0, 'failed': 0}, 'hits': {'total': {'value': 1, 'relation': 'eq'}, 'max_score': 1.0, 'hits': [{'_index': 'py_index01', '_id': 'UhcSYYcBxKJpPocP6lbo', '_score': 1.0, '_source': {'name': 'luce', 'age': '23', 'city': '北京', 'hobbies': '篮球'}}]})
```

In [37]:

```
# 10 regexp查询 正则查询
query = {
    "query": {
        "regexp": {
            "hobbies": ".*?rea.*"
        }
    }
}
es.search(index='py_index01', body = query)
```

C:\Users\EDY\AppData\Local\Temp\ipykernel\_21284\1320344072.py:9: DeprecationWarning: The 'body' parameter is deprecated and will be removed in a future version. Instead use individual parameters.

```
es.search(index='py_index01', body = query)
```

Out[37]:

```
ObjectApiResponse({'took': 991, 'timed_out': False, '_shards': {'total': 1, 'successful': 1, 'skipped': 0, 'failed': 0}, 'hits': {'total': {'value': 0, 'relation': 'eq'}, 'max_score': None, 'hits': []}})
```

In [36]:

```
# 11 根据id删除数据
es.delete(index='py_index01', id=2)
```

Out[36]:

```
ObjectApiResponse({'_index': 'py_index01', '_id': '2', '_version': 2, 'result': 'deleted', '_shards': {'total': 2, 'successful': 1, 'failed': 0}, '_seq_no': 3, '_primary_term': 1})
```

In [40]:

```
# 根据指定的字段 内容删除
dele = {
    "delete":{
        "match":{
            "name":"luce"
        }
    }
}
es.search(index='py_index01', body =dele)  # !!!!!
```

C:\Users\EDY\AppData\Local\Temp\ipykernel\_21284\2890723677.py:9: DeprecationWarning: The 'body' parameter is deprecated and will be removed in a future version. Instead use individual parameters.

```
es.search(index='py_index01', body =dele)
```

-----  
 -----  
**TypeError** Traceback (most recent call last)

Cell In [40], line 9

```
1 # 根据指定的字段 内容删除
2 dele = {
3     "delete":{
4         "match":{
5             (. . .)
6         }
7     }
8 }
```

```
----> 9 es.search(index='py_index01', body =dele)
```

File e:\python38\lib\site-packages\elasticsearch\\_sync\client\utils.py:414, in \_rewrite\_parameters.<locals>.wrapper.<locals>.wrapped(\*args, \*\*kwargs)

```
411     except KeyError:
412         pass
```

```
--> 414 return api(*args, **kwargs)
```

**TypeError:** search() got an unexpected keyword argument 'delete'

In [41]:

```
# 12 根据索引删除
r = es.indices.delete(index="py_index01", ignore = 404)
print(r)
```

```
{'acknowledged': True}
```

C:\Users\EDY\AppData\Local\Temp\ipykernel\_21284\2966514570.py:2: DeprecationWarning: Passing transport options in the API method is deprecated. Use 'Elasticsearch.options()' instead.

```
r = es.indices.delete(index="py_index01", ignore = 404)
```

In [42]:

```
# 13 检查索引是否存在
```

```
r = es.indices.delete(index="py_index01", ignore = 404)
print(r)
```

```
{'error': {'root_cause': [{'type': 'index_not_found_exception', 'reason': 'no such index [py_index01]', 'resource.type': 'index_or_alias', 'resource.id': 'py_index01', 'index_uuid': '_na_', 'index': 'py_index01'}], 'type': 'index_not_found_exception', 'reason': 'no such index [py_index01]', 'resource.type': 'index_or_alias', 'resource.id': 'py_index01', 'index_uuid': '_na_', 'index': 'py_index01'}, 'status': 404}
```

C:\Users\EDY\AppData\Local\Temp\ipykernel\_21284\3928655795.py:2: DeprecationWarning: Passing transport options in the API method is deprecated. Use 'Elasticsearch.options()' instead.

```
r = es.indices.delete(index="py_index01", ignore = 404)
```

## 四，IK分词器

### 4.1 IK分词是什么？

IK分词器是ES的一个插件，主要用于把我们要查询的数据拆分成一个个关键字，我们在搜索时，ElasticSearch会把数据进行分词，然后做匹配。默认的中文分词器会把每一个中文拆分，比如“李四在吃饭”，会拆分成“李”，“四”，“在”，“吃”，“饭”，显然，这并不符合我们的要求，会影响我们最终的搜索结果，所以ik分词器（中文分词器）能解决这个问题。

### 4.2 IK分词插件的安装

下载链接：<https://github.com/medcl/elasticsearch-analysis-ik/releases/tag/v8.2.3>  
(<https://github.com/medcl/elasticsearch-analysis-ik/releases/tag/v8.2.3>)

注意：IK插件的版本一定要与ES和 Kibana的版本一致

将下载好的插件解压并放入elasticsearch服务路径中的plugins文件夹中

此电脑 > app (F:) > elasticsearch-8.2.3-windows-x86_64 > elasticsearch-8.2.3 >				
名称	修改日期	类型	大小	
bin	2022/8/19 14:58	文件夹		
config	2022/8/19 14:59	文件夹		
data	2022/8/22 16:16	文件夹		
jdk	2022/8/19 14:59	文件夹		
lib	2022/8/19 14:59	文件夹		
logs	2022/8/22 16:01	文件夹		
modules	2022/8/19 14:59	文件夹		
plugins	2022/8/22 15:57	文件夹		
LICENSE.txt	2022/6/8 22:21	文本文档	4 KB	
NOTICE.txt	2022/6/8 22:25	文本文档	853 KB	
README.asciidoc	2022/6/8 22:21	ASCIIDOC 文件	3 KB	



此电脑 > app (F:) > elasticsearch-8.2.3-windows-x86\_64 > elasticsearch-8.2.3 > plugins >

名称	修改日期	类型	大小
----	------	----	----

### 4.3使用原生语法测试IK插件是否安装成功

In [11]:

```
GET _analyze
{
  "analyzer": "ik_smart", # 选择IK智能分词
  "text": "张三在吃饭， 李四在睡觉"
}

# analyzer不同程度的智能分词：https://blog.csdn.net/weixin_44062339/article/details/85006948
# https://blog.csdn.net/qq_32630565/article/details/84455572
```

Cell In [11], line 1  
GET analyze  
^  
SyntaxError: invalid syntax

In [ ]: