

Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Кафедра мережних технологій факультету інформатики

ELASTICSEARCH ЯК ЯДРО ПОШУКОВОЇ СИСТЕМИ

**Текстова частина до курсової роботи
за спеціальністю „Прикладна математика” 113**

Керівник курсової роботи
доктор технічних наук, доцент
Глибовець А.М.

(підпис)

“ ____ ” _____ 2020 р.

Виконав студент прикладної
математики - 3
Федусов С.В.

“ ____ ” _____ 2020 р.

Київ 2020

Календарний план виконання курсової роботи

№ п/п	Назва етапу курсової роботи	Термін виконання етапу	Примітка
1.	Отримання завдання на дипломну роботу.	02.10.2019	
2.	Огляд технічної літератури за темою роботи.	13.01.2020	
3.	Огляд API індексації та пошуку	14.02.2020	
4.	Дослідження процесів індексації, пошуку та ранжування	29.02.2020	
5.	Огляд архітектури Elasticsearch	07.03.2020	
6.	Створення практичного застосунку для демонстрації описаних можливостей Elasticsearch	21.03.2020	

ЗМІСТ

Вступ	4
Розділ 1. Базовий функціонал пошукового двигуна Elasticsearch	6
1.1 Додавання даних	6
1.2 Аналіз та пошук	10
Розділ 2. Алгоритми індексації та пошуку	19
2.1 Індексація	19
2.2 Ранжування	21
Розділ 3. Архітектура	23
Розділ 4. Індексація та аналіз тестових даних	27
Висновки	34
Список використаної літератури	35

Вступ

Задача побудови пошукової системи вимагає чіткого розуміння аспектів сфери її використання, таких як: об'єм пошукових даних, кількість користувачів, кількість пошукових запитів в секунду (tps). Що робити, коли даних багато та навантаження на систему велике?

Сучасні пошукові системи мають наступні базові функції: швидкий повнотекстовий пошук, фільтрацію, сортування і ранжування документів, нечіткий пошук, швидке збереження та індексування структурованих даних. Необхідність отримувати результати пошукових запитів за мілісекунди, маючи десятки або сотні гігабайт інформації вимагає використання ефективних алгоритмів та структур даних. Окрім цього, сама система повинна гарантувати цілісність та відмовостійкість. Надає необхідний функціонал та задовольняє наведені вимоги пошуковий двигун Elasticsearch.

Для демонстрації можливостей індексації, пошуку та аналізу було використано статистичні дані поширення вірусу COVID-19, опубліковані університетом Джона Гопкінса [5]. Для ілюстрації індексованих даних поширення вірусу було використано Kibana - інструмент для візуалізації даних з Elasticsearch.

Мета роботи - розглянути можливості пошукового двигуна Elasticsearch, виконати індексацію та аналіз тестових даних.

Постановка задачі

1. Ознайомитися з базовим функціоналом пошукового двигуна Elasticsearch.

2. Дослідити алгоритми та структури даних, що використовуються для індексації.
3. Розглянути архітектуру пошукового двигуна.
4. Застосувати Elasticsearch для індексації та аналізу тестових даних.

Розділ 1. Базовий функціонал пошукового двигуна Elasticsearch.

Elasticsearch – це розподілена система пошуку і аналітики, що дозволяє ефективно зберігати та індексувати дані для швидкого пошуку в режимі реального часу.

1.1 Додавання даних

Ключові поняття:

Термін (term) - *“a word from the text and the basic unit for searching. In different contexts, this word can mean different things: it could be a name, for example, or it could be an IP address. If you want only exact matches on a field, the entire field should be treated as a word”* [1, 59].

Документ - найменша структурна одиниця інформації, що індексується. Як правило, документ є представленням певного об'єкту і містить в собі поля (атрибути) та їх значення.

Схема (mapping) - представлення структури документа, що надає інформацію про усі поля, їх типи та додаткові параметри для індексації.

Значення атрибутів документа можуть мати наступні основні типи:

- Текстові
- Keyword

Текстовий тип даних для індексування структурованих значень, наприклад e-mail адреса або номер телефону. Пошук документів, що містять атрибут типу keyword, здійснюється за його повним значенням.

- Text

Ще один текстовий тип даних, що, на відміну від `keyword`, призначений для індексування великих обсягів тексту, таких як вміст email листа або абзац книги. Цей тип даних використовується для пошуку документів, `text` поле яких містить шукане значення.

- Числові

- Цілі

Типи цілих чисел відрізняються лише розмірністю. `Byte` містить 1 байт, `short` - 2, `integer` - 4, `long` - 8.

- Дробові

Аналогічно до цілих, дробові типи `float` і `double` мають розмір 4 та 8 байт відповідно.

- Булевий

- Бінарний

- Дата

Дата може бути подана у вигляді “`yyyy-mm-dd`”, “`yyyy-mm-dd hh-mm-ss`” або навіть у мілісекундах. Такому представленню відповідає тип `date`. Схожим на нього є тип `date_nanos`, що відрізняється від попереднього можливістю задавати дату у наносекундах.

- Числовий проміжок

Тип `range` має ліву та праву границю, при чому вона може включатися або не включатися. Ліва границя задається параметрами `gte` (більше або

рівне) або `gt` (строго більше). Права границя проміжку аналогічно задається параметрами `lte` (менше або рівне) і `lt` (строго менше).

- **Об'єкт**

Застосовується для логічного представлення ієрархічних структур. Різновидами типу об'єкт є `object` і `nested`. На відміну від `object`, об'єкти типу `nested` індексуються і можуть бути знайдені незалежно один від одного.

Індекс - оптимізована для вставки і пошуку колекція, що містить множину документів. Одним з основних видів індексу, що використовується в Elasticsearch є інвертований індекс. Кожному індексованому терміну ставиться у відповідність множина документів, що його містять. Окрім цього, для індексації кожного окремого поля обирається спеціалізована, оптимізована структура даних: текстові поля зберігаються в інвертованому індексі, а числові і географічні - в BKD деревах [2].

Розглянемо структуру індексу для тестових даних. Для цього зобразимо статистику кожної країни у вигляді об'єкта з наступними атрибутами:

- `country` - назва країни
- `region` - регіон країни, для якого наявна окрема статистика. Дані більшості країн не мають розділу на регіони, але для деяких (наприклад Китай) такий розподіл показує більш детальну картину захворюваності.
- `location` - координати країни у форматі широта-довгота.

- daily - щоденна кількість хворих. За допомогою цієї інформації можна аналізувати поширення вірусу. Об'єкти daily представлені у вигляді nested типу, бо може виникнути необхідність додаткового пошуку по ним.

```
PUT /covid_19
{
  "settings": {
    "index": {
      "number_of_shards": 1,
      "number_of_replicas": 0
    }
  },
  "mappings": {
    "dynamic": "strict",
    "_source": {
      "enabled": "true"
    },
    "properties": {
      "region": {
        "type": "keyword"
      },
      "country": {
        "type": "keyword"
      },
      "location": {
        "type": "geo_point"
      },
      "daily": {
        "type": "nested",
        "properties": {
          "date": {
            "type": "date"
          },
          "cases": {
            "type": "integer"
          }
        }
      }
    }
  }
}
```

Рис. 1.1.1 Схema індексу даних розповсюдження COVID-19

```
POST /covid_19/_doc/

{
  "country": "Afghanistan",
  "location": {
    "lat": 33.0,
    "lon": 65.0
  },
  "daily": [
    {
      "date": "2020-01-22",
      "cases": 0
    },
    {
      "date": "2020-01-23",
      "cases": 0
    },
    {
      "date": "2020-01-24",
      "cases": 0
    }
  ]
}
```

Рис. 1.1.2 Приклад документа - країни з координатами та щоденною статистикою зареєстрованих випадків захворювання

1.2 Аналіз та пошук.

Elasticsearch надає широкий спектр можливостей для аналізу та пошуку: фільтрації, агрегації, ранжування. Для виконання пошукових запитів існує Elasticsearch Query DSL - декларативна мова, що дозволяє описувати очікувані результати.

Основні складові пошукового запиту:

- query - тіло запиту.

- `sort` - визначає порядок документів в пошуковій видачі. Параметр сортування задається списком поле-напрямок, розділеним комами. Можливі значення: `asc`, `desc` - за зростанням або спаданням відповідно.
- `size` - розмір результату, що має бути повернений.
- `from` - відступ від першого значення у видачі.

Query

Документи в індексі можуть бути знайдені за значеннями їх атрибутів. Секція `query` пошукового запиту описує документи, що повинні з'явитися у пошуковій видачі. Для цього Elasticsearch надає можливості булевого пошуку, фільтрації, агрегації.

Match_all

Отримати усі документи з індексу дозволяє `match_all` запит. Він не накладає жодних обмежень на результат, що має бути повернений. В цьому випадку, розмір пошукової видачі обмежується лише параметром `size` пошукового запиту.

```
POST /covid_19/_search
```

```
{
  "query": {
    "match_all": {}
  },
  "sort": "country",
  "from": 0,
  "size": 10
}
```

Рис 1.2.1 Пошук усіх документів індексу, відсортованих лексикографічно за назвою країни

Пошук по термінах

Щоб знайти документ, значенням певного атрибута якого є визначений термін, використовуються `term` та `match` пошукові запити.

- Term query

Результатом пошукового запиту будуть документи, значення шуканого атрибута яких чітко дорівнює шуканому терміну. Окрім того, запит `terms` дозволяє знайти документи, що містять деякі значення з переліку.

```
POST /covid_19/_search
{
  "query": {
    "terms": {
      "country": ["Ukraine"]
    }
  },
  "from": 0,
  "size": 10
}
```

Рис 1.2.2 Terms запит

- Match query

Надає більш широкий функціонал для пошуку по термінах. Для шуканої фрази є можливість вказати оператор, формуючи таким чином аналог булевого пошуку. Шукана фраза розділяється на токени і виконується булевий пошук за їх переліком. За замовчуванням,

оператором булевого пошуку між кожним токеном є OR. Існує також фразовий пошук з урахуванням відстані та префіксний пошук.

```
POST /covid_19/_search
{
  "query": {
    "match": {
      "country": {
        "query": "Ukraine"
      }
    }
  },
  "from": 0,
  "size": 10
}
```

Рис 1.2.3 Match запит з оператором

Term та match пошукові запити також відрізняються аналізом шуканого терміну: для першого термін не аналізується, тому результатом запиту будуть документи, значення шуканого атрибуту яких чітко дорівнює шуканому терміну, а для другого, навпаки, шуканий термін аналізується, що дозволяє виконувати нечутливий до регістру пошук.

Булевий пошук

Для того, щоб знайти документи, що повинні відповідати декільком вимогам, або якщо деякі вимоги не є обов'язковими, використовується булевий пошук. За допомогою операторів AND, OR, NOT будується предикат, якому повинен задовольняти кожен документ пошукової видачі.

Ukraine OR China AND NOT Italy

Рис 1.2.4 Приклад булевого пошуку

Elasticsearch query надає можливість вкладених булевих запитів, використовуючи оператори `must`, `should` і `must_not`. Кожен документ повинен задовільняти всі предикати `must`, може задовільняти предикати `should`, і не повинен задовільняти предикати `must_not`. Якщо пошуковий запит містить лише `should` частину, то документи мають задовільняти хоча б одному `should` предикату. У випадку наявності `must`, документи не обов'язково повинні задовільняти предикати `should`. Перевизначити стандартну поведінку дозволяє параметр `minimum_should_match`, що задає мінімальну кількість предикатів, яку повинен задовільняти кожен документ.

POST /covid_19/_search

```
{
  "query": {
    "bool": {
      "must": [
        {
          "term": {
            "country": {
              "value": "China"
            }
          }
        },
        {
          "geo_bounding_box": {
            "location": {
              "top_left": {
                "lat": 30,
                "lon": 100
              },
              "bottom_right": {
                "lat": 20,
                "lon": 110
              }
            }
          }
        }
      ]
    }
  },
  "from": 0,
  "size": 10
}
```

Рис 1.2.5 Пошук даних для країни Китай в обмеженому географічному сегменті використовуючи булевий запит

Range query

Щоб знайти усі документи, певний атрибут яких лежить у заданому проміжку, використовується range запит. Аналогічно з range типом даних, пошуковий запит має параметри gt, gte, lt, lte, що задають проміжок.

Range пошук можна застосовувати не лише для числових значень, але й для текстових і дати.

POST /index/_search

```
{
  "query": {
    "range" : {
      "start_time" : {
        "gte" : "2020-02-10T00:00:00",
        "lt" : "2020-02-15T00:00:00"
      }
    }
  }
}
```

Рис 1.2.6 Range запит

Фільтрація

На відміну від пошуку, фільтрація дозволяє лише накладати додаткові умови на результат пошукової видачі, при цьому не виконуючи ранжування документів. Така можливість дозволяє пришвидшити виконання запиту, зменшивши кількість виконуваних операцій.

“By default, Elasticsearch sorts matching search results by relevance score, which measures how well each document matches a query. The relevance score is a positive floating point number, returned in the `_score` meta-field of the search API. The higher the `_score`, the more relevant the document” [2].

Умови фільтрації вказуються в окремому блоці `filter[]`. Всі пошукові умови, що використовуються в пошуку також застосовні в блоці фільтрації.

POST /covid_19/_search

```
{
  "query": {
    "bool": {
      "must": [
        {
          "term": {
            "country": {
              "value": "China"
            }
          }
        }
      ]
    },
    "filter": [
      {
        "geo_bounding_box": {
          "location": {
            "top_left": {
              "lat": 30,
              "lon": 100
            },
            "bottom_right": {
              "lat": 20,
              "lon": 110
            }
          }
        }
      ]
    ]
  },
  "from": 0,
  "size": 10
}
```

Рис 1.2.7 Пошук даних для країни Китай з фільтром за географічним сегментом

Кешування

Ще однією перевагою фільтрації над пошуком є можливість зберігати результати часто виконуваних запитів для пришвидшення наступних пошуків.

“By default, the cache holds a maximum of 10000 queries in up to 10% of the total heap space. To determine if a query is eligible for caching, Elasticsearch maintains a query history to track occurrences” [2].

Розділ 2. Алгоритми індексації та пошуку.

Додаючи нові дані в індекс Elasticsearch, перед їх безпосереднім збереженням, вони проходять попередню обробку. Цей процес готує дані до індексації.

2.1 Індексція

Токенізація

Для того, щоб мати можливість пошуку документів, текстове поле яких містить певний термін, на етапі індексації відбувається процес токенизації.

Token - *“an instance of a sequence of characters in some particular document that are grouped together as a useful semantic unit for processing”* [3, 22].

Процес токенизації в Elasticsearch складається з декількох стадій:

- Трансформація і фільтрація символів
- Токенізація
- Нормалізація і фільтрація токенів
- Додавання токенів в індекс

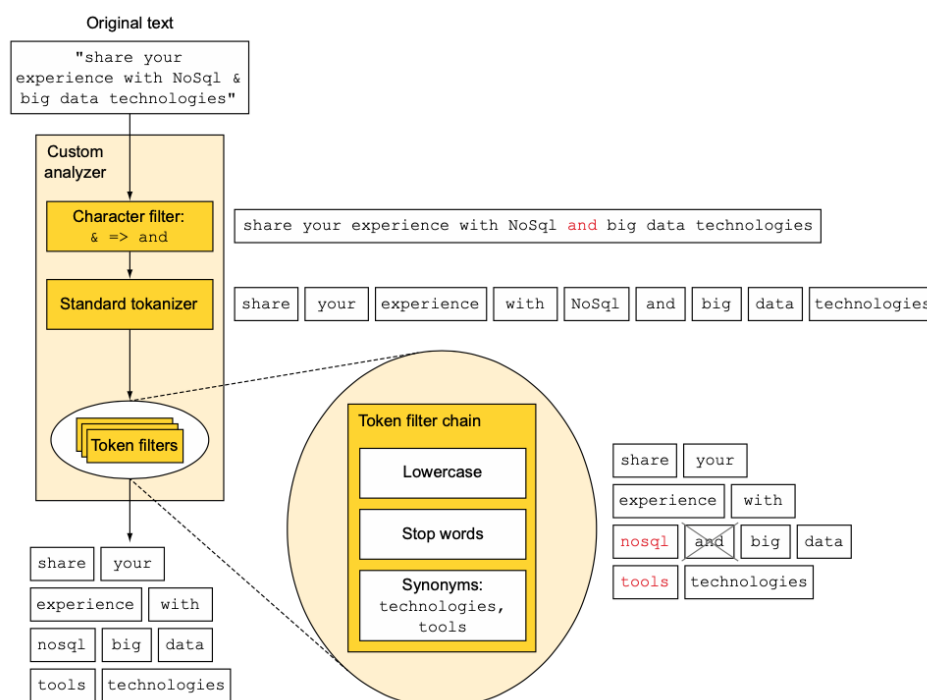


Рис 2.1.1 Токенізація в Elasticsearch [1, 119]

Type - *“the class of all tokens containing the same character sequence”* [3, 22].

Term - *“(perhaps normalized) type that is included in the IR system’s dictionary”* [3, 22].

Нормалізація токенів

Декілька токенів можуть мати однакове значення, але різне написання. В таких випадках, результати пошукового запиту одного з токенів не будуть містити релевантні документи, що містять еквівалентні токени.

Token normalization - *“the process of canonicalizing tokens so that matches occur despite superficial differences in the character sequences of the tokens”* [3, 28].

2.2 Ранжування

Пошуковий двигун Elasticsearch надає різноманітні можливості пошуку і ранжування документів. Для того, щоб отримати документи в порядку, що якнайкраще відповідає пошуковому запиту, для кожного з них обраховується `_score` - числове значення, що характеризує релевантність.

Ранжування за термінами

Для того, щоб оцінити важливість терміну t для кожного документа d обраховується TF-IDF показник.

$TF_{t,d}$ (term frequency - частота терміну) - кількість входжень терміну t в документ d .

DF_t (document frequency - частота документа) - загальна кількість документів, що містять певний термін.

IDF_t (inverse document frequency — обернена частота документа) - числовий показник, що обраховується за формулою:

$$IDF_t = 1 + \log((N + 1)/(DF_t + 1)). \quad [4]$$

В Elasticsearch `_score` кожного документу обчислюється за формулою:

$$score_{query, document} = \sum_t^q \sqrt{TF_{t,d}} * IDF_{t,d}^2 * norm(d, field) * boost(t)$$

Рис 2.2.1 `_score` формула [1, 151]

Формула 2.2.1 використовує додаткові множники: $\text{norm}(d, \text{field})$ і $\text{boost}(t)$. Параметр $\text{boost}(t)$ дозволяє вибірково збільшити вагу окремого терміну t в пошуковому запиті [4]. Це має сенс в тому випадку, якщо пошуковий запит складається з декількох частин, і деякі з них мають більший пріоритет, ніж інші. Параметр $\text{norm}(d, \text{field})$ - “an index-time boost factor that solely depends on the number of tokens of this field in the document, so that shorter fields contribute more to the score” [4].

Розділ 3. Архітектура

Оперуючи великими обсягами даних і маючи жорсткі вимоги до часу виконання пошукових запитів, фізичні обмеження однієї обчислювальної машини стають критичними. Elasticsearch є розподіленою системою, що надає можливість легкого масштабування відповідно до існуючих вимог.

Кластер - множина обчислювальних машин, що працюють разом і утворюють єдину систему.

Нода - обчислювальна машина в кластері.

Шард - частина індексу. За допомогою шардування, індекс можна розмістити на декількох нодах. Шарди поділяються на 2 типи: `primary` (основна) та `replica` (репліка). Реплікаційні шарди використовуються для відмовостійкості та покращення швидкодії. Вони розміщуються на нодах без дублювання, тобто на одній ноді не може розміщуватися два однакових шарди.

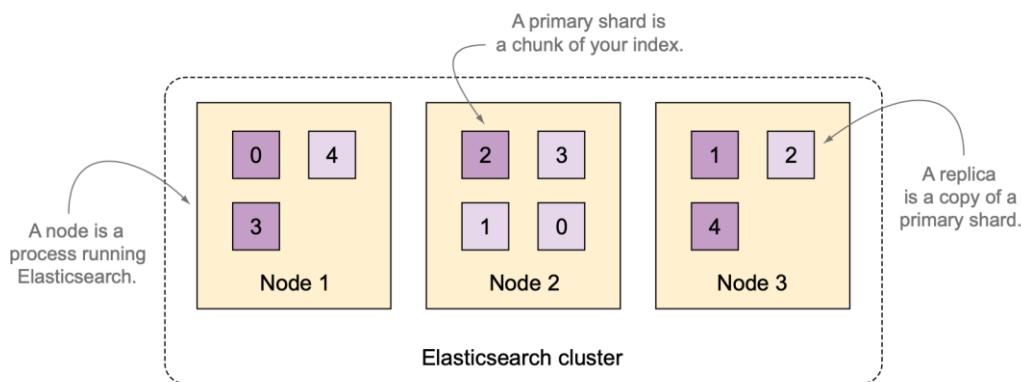


Рис 3.1 “A three-node cluster with an index divided into five shards with one replica per shard” [1, 26]

Така архітектура дозволяє Elasticsearch збільшувати кількість нод і обробляти більший об’єм даних.

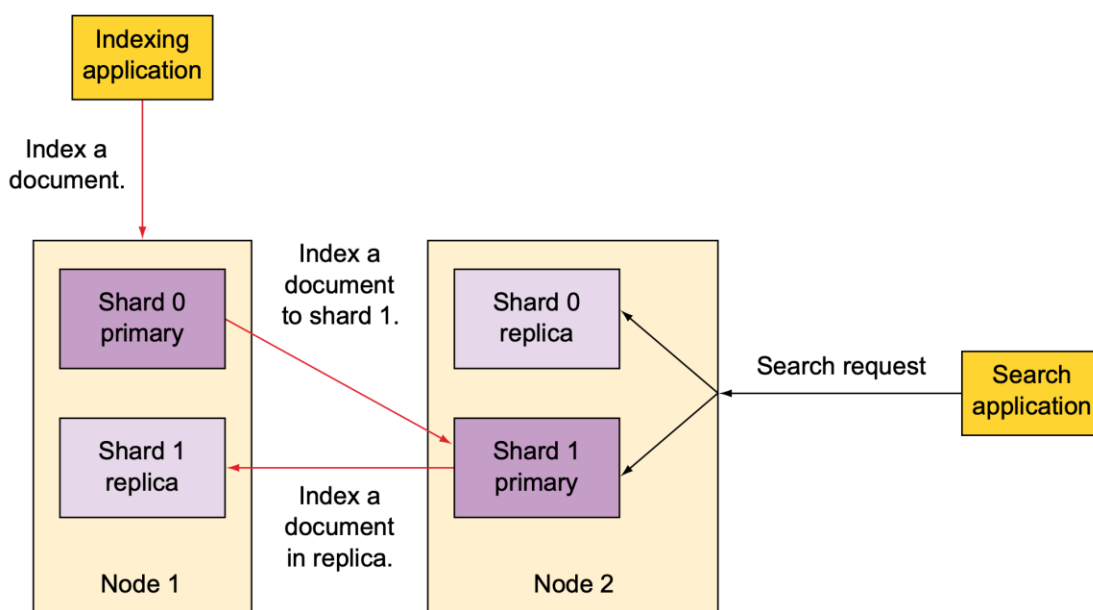


Рис 3.2 “Documents are indexed to random primary shards and their replicas. Searches run on complete sets of shards, regardless of their status as primaries or replicas” [1, 27]

Рис 3.2 ілюструє застосування реплікаційних шард для пришвидшення пошукових запитів в Elasticsearch кластері. Для того, щоб досягти розбиття Elasticsearch індексу на шарди, потрібно при його створенні вказати відповідні параметри в налаштуваннях.


```
PUT /covid_19
{
  "settings": {
    "index": {
      "number_of_shards": 5,
      "number_of_replicas": 1
    }
  }
}
```

Рис 3.3 Налаштування шардування індексу

Metrics Shards Aliases Mappings Administration

Shard	State	# Docs	Size	Primary	Node
0	STARTED	52	107.6kb	true	es01
0	STARTED	52	107.6kb	false	es02
1	STARTED	36	81.2kb	false	es01
1	STARTED	36	81.3kb	true	es02
2	STARTED	55	115.2kb	true	es01
2	STARTED	55	110.4kb	false	es02
3	STARTED	49	102.4kb	false	es01
3	STARTED	49	102.8kb	true	es02
4	STARTED	61	119.2kb	true	es01
4	STARTED	61	121kb	false	es02

Рис 3.4 Список створених шард

На рис 3.3 - 3.4 створюється індекс з п'ятьма шардами та однією реплікою. Створення індексу відбувається на кластері з двох, нод: es01 та es02. Таким чином, кожна нода буде мати кожен шард індексу, гарантуючи відмовостійкість та швидкодію: якщо одна нода кластеру потрапить в

аварійну ситуацію, то інша зможе повноцінно відпрацьовувати пошукові запити без втрати даних. Під час індексації, дані автоматично були приблизно рівномірно розподілені між кожним шардом.

Ноди

Кластер Elasticsearch зазвичай складається з декількох нод. Для того, щоб забезпечити злагоджене функціонування, існує розподіл нод на ролі:

Master node (ведуча нода) - *“responsible for lightweight cluster-wide actions such as creating or deleting an index, tracking which nodes are part of the cluster, and deciding which shards to allocate to which nodes.”* [2]

Data node - містить шарди з індексованими документами. [2]

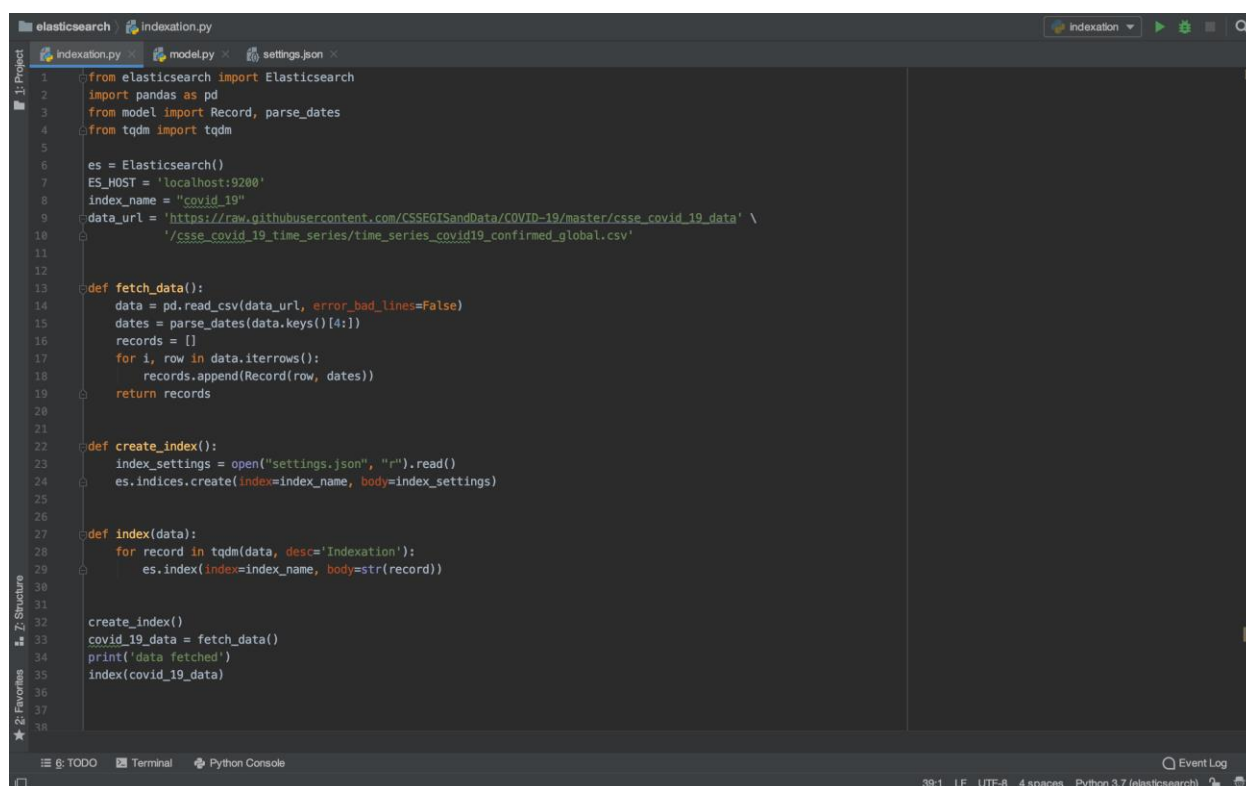
Ingest node - виконує попередню обробку документів, що індексуються. [2]

Щоб забезпечити функціонування кластера в аварійній ситуації за відмови master ноди, існує механізм голосування (voting): ноди, що сконфігуровані як master-eligible, домовляються, яка з них буде новим master.

Розділ 4. Індексція та аналіз тестових даних

За тестові дані для індексації та аналізу було обрано світову статистику поширення вірусу COVID-19. Для кожної країни надається щоденна інформація про кількість хворих, а також координати країни у вигляді широта-довгота. Для деяких країн окремо надана інформація, деталізована за регіонами.

Індексування даних здійснювалося за допомогою мови програмування Python, Elasticsearch Python API та бібліотеки Pandas.



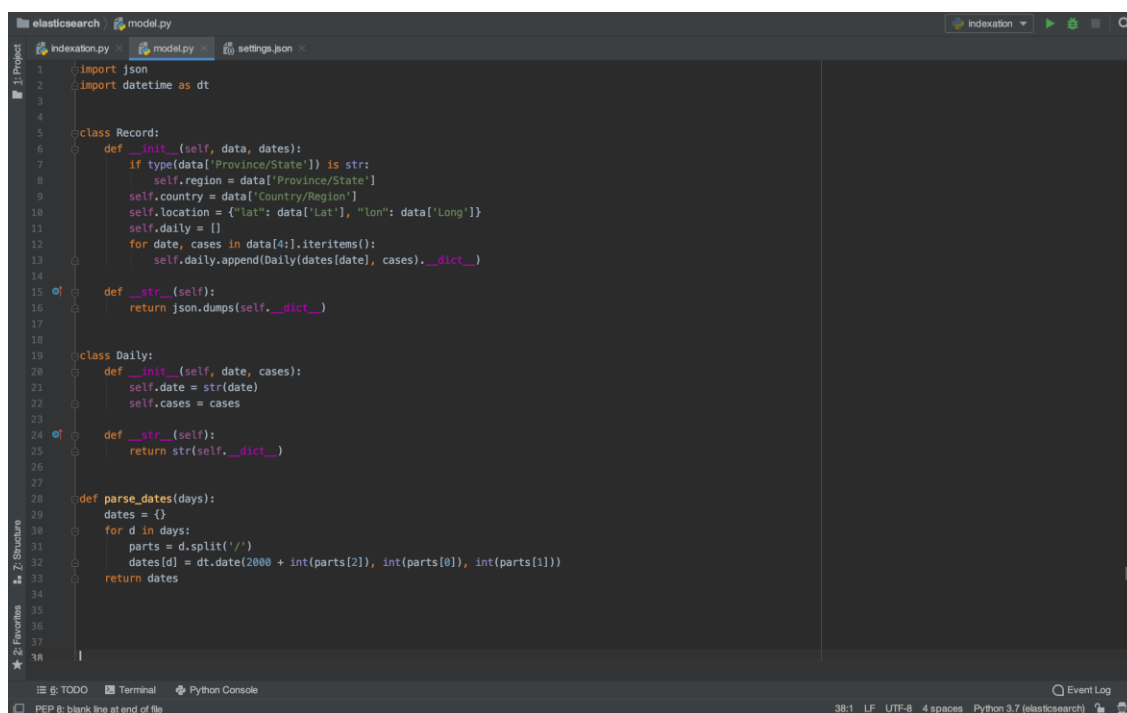
```

1 from elasticsearch import Elasticsearch
2 import pandas as pd
3 from model import Record, parse_dates
4 from tqdm import tqdm
5
6 es = Elasticsearch()
7 ES_HOST = 'localhost:9200'
8 index_name = "covid_19"
9 data_url = 'https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data' \
10           '/csse_covid_19_time_series/time_series_covid19_confirmed_global.csv'
11
12
13 def fetch_data():
14     data = pd.read_csv(data_url, error_bad_lines=False)
15     dates = parse_dates(data.keys()[4:])
16     records = []
17     for i, row in data.iterrows():
18         records.append(Record(row, dates))
19     return records
20
21
22 def create_index():
23     index_settings = open("settings.json", "r").read()
24     es.indices.create(index=index_name, body=index_settings)
25
26
27 def index(data):
28     for record in tqdm(data, desc='Indexation'):
29         es.index(index=index_name, body=record)
30
31
32 create_index()
33 covid_19_data = fetch_data()
34 print('data fetched')
35 index(covid_19_data)

```

Рис 4.1 Python скрипт для індексації

Для перетворення даних в JSON формат була використана бібліотека `Json`. Статистика з розділом по країнах, а також інформація про щоденні випадки захворювання була відображена за допомогою Python класів.



```

1 import json
2 import datetime as dt
3
4 class Record:
5     def __init__(self, data, dates):
6         if type(data['Province/State']) is str:
7             self.region = data['Province/State']
8             self.country = data['Country/Region']
9             self.location = {"lat": data['Lat'], "lon": data['Long']}
10            self.daily = []
11            for date, cases in data[4:].iteritems():
12                self.daily.append(Daily(dates[date], cases).__dict__)
13
14    def __str__(self):
15        return json.dumps(self.__dict__)
16
17 class Daily:
18     def __init__(self, date, cases):
19         self.date = str(date)
20         self.cases = cases
21
22    def __str__(self):
23        return str(self.__dict__)
24
25 def parse_dates(days):
26     dates = {}
27     for d in days:
28         parts = d.split('/')
29         dates[d] = dt.date(2000 + int(parts[2]), int(parts[0]), int(parts[1]))
30     return dates

```

Рис 4.2 Модель даних

Користуючись наведеними даними отримаємо інформацію про кількість захворювань в Україні у період з 1-го квітня. Для цього, сформуємо пошуковий запит, що знайде документи в індексі, значення поля `country` яких дорівнює `Ukraine`. Окрім цього, для пошуку щоденної кількості зареєстрованих випадків захворювання, додамо до запиту `nested` секцію, завдяки якій у результаті будуть отримані лише ті щоденні звіти, що були надані після 1-го квітня 2020 року.

POST /covid_19

```
{
  "query": {
    "bool": {
      "filter": [
        {
          "term": {
            "country": {
              "value": "Ukraine"
            }
          }
        },
        {
          "nested": {
            "path": "daily",
            "query": {
              "bool": {
                "filter": {
                  "range": {
                    "daily.date": {
                      "gte": "2020-04-01"
                    }
                  }
                }
              }
            }
          },
          "inner_hits": {
            "size": 10,
            "stored_fields": [
              "daily.date",
              "daily.cases"
            ]
          }
        }
      ]
    }
  },
  "stored_fields": [
    "region",
    "country",
    "location"
  ],
  "from": 0,
  "size": 10
}
```

Рис 4.3 Запит для пошуку щоденної статистики в Україні

```
{
  "took": 138,
  "timed_out": false,
  "_shards": {
    "total": 5,
    "successful": 5,
    "skipped": 0,
    "failed": 0
  },
  "hits": {
    "total": {
      "value": 1,
      "relation": "eq"
    },
    "max_score": 0.0,
    "hits": [
      {
        "_index": "covid_19",
        "_type": "_doc",
        "_id": "FBuscHEBLL0YHqeQf9IM",
        "_score": 0.0,
        "fields": {
          "country": [
            "Ukraine"
          ],
          "location": [
            "48.3794, 31.1656"
          ]
        }
      },

```

Рис 4.4 Результат поискового запроса ч.1

```

"inner_hits": {
  "daily": {
    "hits": {
      "total": {
        "value": 11,
        "relation": "eq"
      },
      "max_score": 0.0,
      "hits": [
        {
          "_index": "covid_19",
          "_type": "_doc",
          "_id": "FBuscHEB\LOYHqeQf9IM",
          "_nested": {
            "field": "daily",
            "offset": 70
          },
          "_score": 0.0,
          "fields": {
            "daily.cases": [
              794
            ],
            "daily.date": [
              "2020-04-01T00:00:00.000Z"
            ]
          }
        },
        {
          "_index": "covid_19",
          "_type": "_doc",
          "_id": "FBuscHEB\LOYHqeQf9IM",
          "_nested": {
            "field": "daily",
            "offset": 71
          },
          "_score": 0.0,
          "fields": {
            "daily.cases": [
              897
            ],
            "daily.date": [
              "2020-04-02T00:00:00.000Z"
            ]
          }
        }
      ]
    }
  }
},

```

Рис 4.5 Результат пошукового запиту ч.2

Використовуючи дані розташування країн, за допомогою Kibana побудуємо карту поширення вірусу. Кожна країна має відмітку у вигляді кола, що в інтерактивній формі ілюструє кількість випадків захворювання.

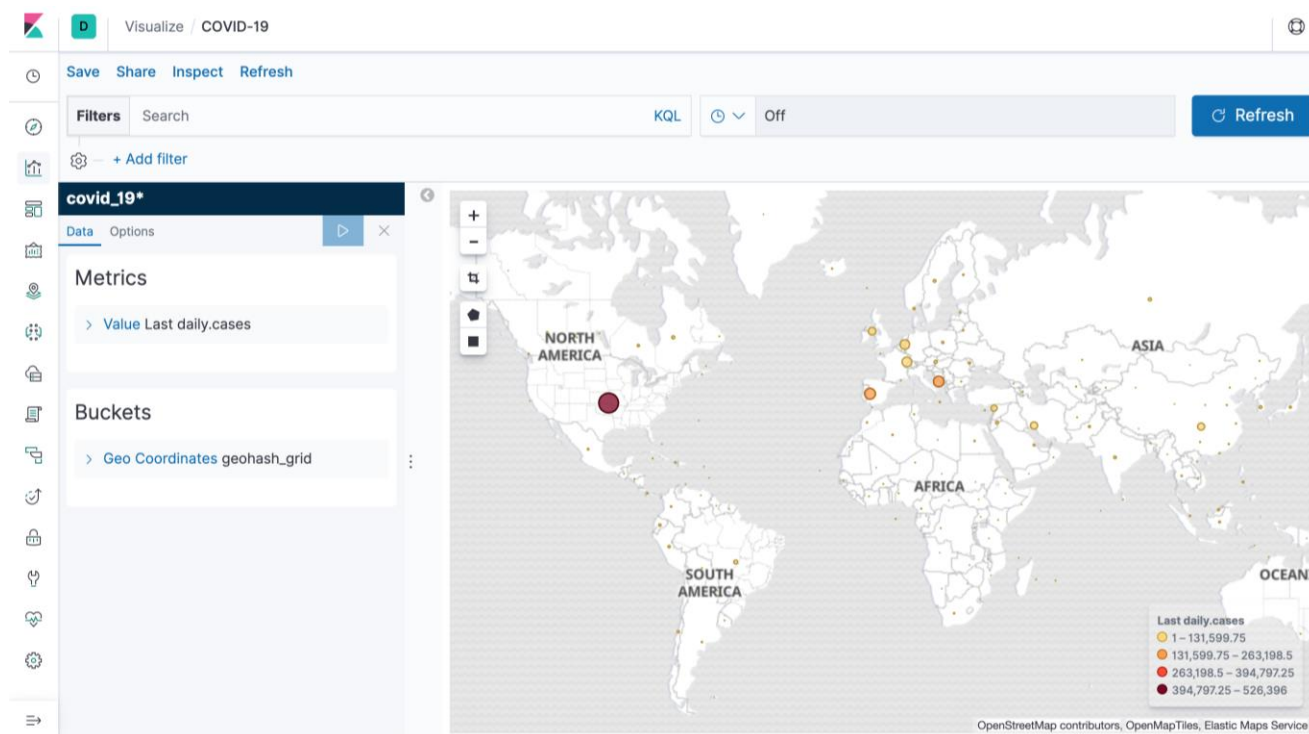


Рис 4.6 Географічна візуалізація статистичних даних

Окрім цього, Kibana дозволяє візуалізувати географічні дані у декількох формах, наприклад у вигляді теплової карти.

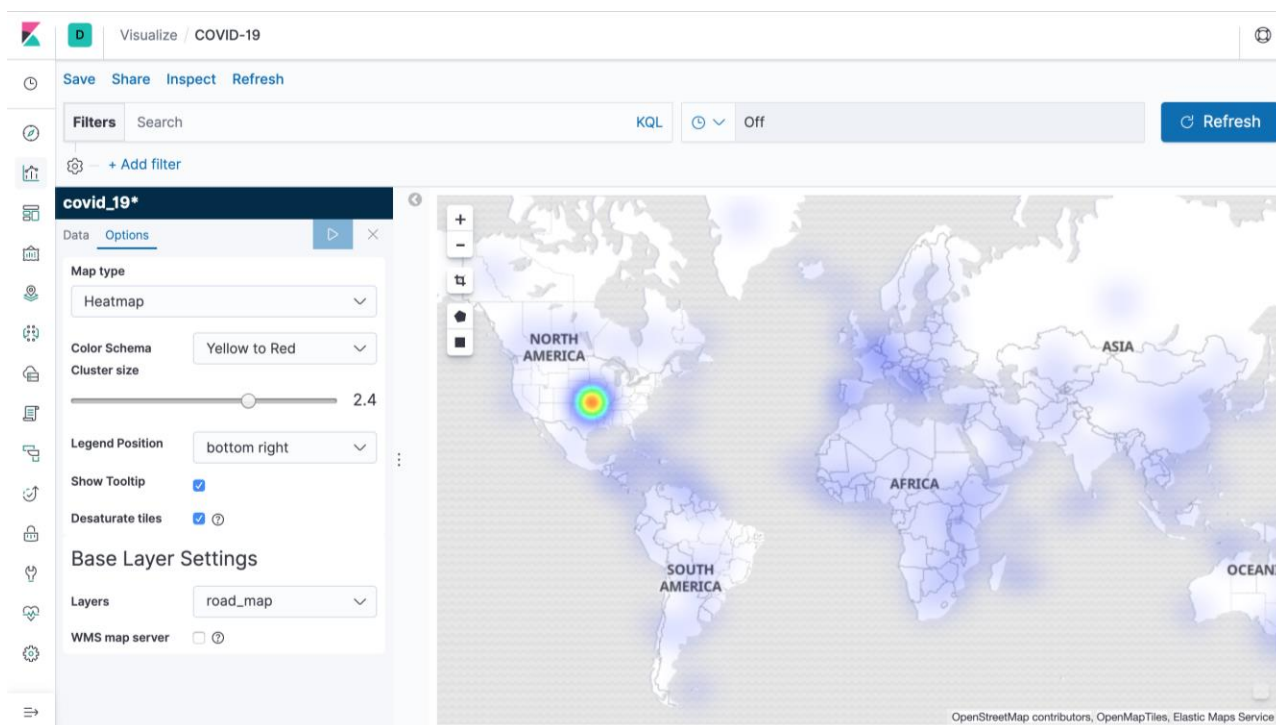


Рис 4.7 Географічна візуалізація даних у вигляді теплової карти

Висновки

Пошуковий двигун Elasticsearch надає широкі можливості індексації, пошуку та аналізу даних. Схема індексу дозволяє описати будь-яку структуру, а мова запитів задовольняє будь-яку інформаційну потребу. Завдяки розподіленій архітектурі нівелюються фізичні обмеження однієї обчислювальної машини: розмір дискової та оперативної пам'яті, обчислювальні ресурси. Ще однією перевагою такого підходу є забезпечення більшої надійності та відмовостійкості системи.

Таким чином, Elasticsearch є гарним рішенням для побудови пошукових систем.

Список літератури

1. Radu Gheorghe, Matthew Lee Hinman, and Roy Russo “Elasticsearch in Action”, November 2015.
2. Elasticsearch documentation.
<https://www.elastic.co/guide/en/elasticsearch/reference/current/index.html>
3. Christopher D. Manning, Prabhakar Raghavan and Hinrich Schütze
“Introduction to Information Retrieval”, Cambridge University Press. 2008.
4. Apache Lucene documentation.
https://lucene.apache.org/core/7_4_0/
5. <https://github.com/CSSEGISandData/COVID-19>