



Д.В. Ланде, І.Ю. Субач, А.Я. Гладун

ОБРОБЛЕННЯ НАДВЕЛИКИХ МАСИВІВ ДАНИХ (BIG DATA)

Навчальний посібник

2021

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
ІНСТИТУТ СПЕЦІАЛЬНОГО ЗВ'ЯЗКУ
ТА ЗАХИСТУ ІНФОРМАЦІЇ**

Д.В. Ланде, І.Ю. Субач, А.Я. Гладун

**ОБРОБЛЕННЯ НАДВЕЛИКИХ МАСИВІВ ДАНИХ
(BIG DATA)**

Навчальний посібник

*Розглянуто Вченою радою ІСЗЗІ КПІ ім. Ігоря Сікорського
для використання у навчальному процесі з підготовки
фахівців другого (магістерського) рівня вищої освіти
зі спеціальності 122 «Комп'ютерні науки»*

Київ
ІСЗЗІ КПІ ім. Ігоря Сікорського
2021

УДК 004.942 (075.8)
Л 12

*Розглянуто Вченою радою
ІСЗІ КПІ ім. Ігоря Сікорського
(Протокол № 6 від 30.12.2021 р.)*

Рецензенти: В.В. Вишнівський, д.т.н., проф.
С.В. Толюпа, д.т.н., проф.

Л12 Д.В. Ланде, І.Ю. Субач, А.Я. Гладун. Оброблення надвеликих масивів даних (Big Data) : навчальний посібник. Київ 2021. – 168 с.

ISBN 978-966-2344-83-7

Навчальний посібник “Оброблення надвеликих масивів даних (Big Data)” включає основні положення концепції “великих даних”, технології оброблення масивів даних надвеликих обсягів, методи створення інформаційних систем, що працюють з “великими даними”, приклади практичного застосування результатів обробки “великих даних”, а також широке коло питань, пов’язаних з практикою обробки та аналізу “великих даних”.

У посібнику розглянуто питання застосування методів інтелектуального аналізу даних до великих обсягів даних (Big Data). Проаналізовані відповідні програмно-інструментальні засоби, у тому числі, потужна система обробки великих даних Hadoop, пошукова система Elasticsearch, система обробки графових баз даних Neo4j, сучасна NoSQL система управління базами даних MongoDB. Для закріплення навчального матеріалу в посібнику наведені приклади програм мовою Python.

Видання призначене курсантам інституту, що здобувають освіту за другим (магістерським) рівнем за спеціальністю 122 “Комп’ютерні науки”. Може бути також корисним всім, хто самостійно хоче засвоїти методи та технології оброблення “великих даних” – курсантам, студентам, аспірантам, спеціалістам технічних спеціальностей та програмістам.

УДК 004.942 (075.8)

ISBN 978-966-2344-83-7

© ІСЗІ КПІ ім. Ігоря Сікорського, 2021

ЗМІСТ

Вступ.....	8
1. Формування масиву даних для аналітичної обробки.....	11
Формат даних.....	12
RSS-фіди.....	15
Процедури завантаження даних	15
Питання до практичної роботи:.....	18
Завдання на самостійну роботу	19
Література	19
2. Формування JSON-файлу для завантаження в Elasticsearch	20
Формат JSON	20
Програма перетворення RSS в JSON	24
Питання до практичної роботи:.....	26
Завдання на самостійну роботу	26
Література	26
3. Встановлення системи Elasticsearch. Завантаження інформації в БД.....	27
Стек Elastic	27
Основні властивості Elasticsearch.....	28
Встановлення Elasticsearch	31
Завантаження бази даних Elasticsearch	32
Перевірка результатів завантаження даних.....	34
Операції CRUD	36
Питання до практичної роботи.....	36
Завдання на самостійну роботу	36
Література	37
4. Запити до Elasticsearch і веб-інтерфейс	38
Запити до Elasticsearch	38

"title" : "Nearly half of Canadians lack confidence in cybersecurity of CRA, Elections Canada: survey",	40
Модель пошукової системи на базі веб-інтерфейсу	42
Програма форматування JSON-файлу	44
Питання до практичної роботи:	46
Завдання на самостійну роботу	46
Література	46
5. Робота із системою Kibana	47
Встановлення системи Kibana	47
Робота із інструментом Console	48
Робота із інструментом Discover	52
Робота із інструментом Visualizations	53
Питання до практичної роботи	55
Завдання на самостійну роботу	55
Література	55
6. Агрегація в Elasticsearch	56
Реалізація агрегації у запитах	56
Динаміка публікацій за запитом	57
Переведення даних в формат CSV	59
Обробка даних у середовище Excel	60
Питання до практичної роботи	61
Завдання на самостійну роботу	61
Література	62
7. Статистична обробка агрегованих даних	63
Середовище розробки	63
Віконне згладжування	64
Експоненціальне згладжування	66
Побудова вейвлет-скейлограми	68
Питання до практичної роботи	72

Завдання на самостійну роботу	73
Література	73
8. Екстрагування ключових слів з відібраних за запитом документів	74
Отримання масиву документів	74
Формування словника	75
Вибір найбільш вагомих слів.....	77
Питання до практичної роботи.....	78
Завдання на самостійну роботу	79
Література	79
9. Формування мережі концептів. Візуалізація в Gephi.....	80
Формування матриці суміжності концептів	80
Основні відомості щодо системи Gephi.....	82
Встановлення системи Gephi.....	84
Візуалізація мережі у середовищі Gephi.....	85
Питання до практичної роботи.....	86
Завдання на самостійну роботу	87
Література	87
10. Apache Hadoop	88
Основні компоненти Hadoop	89
Apache Hadoop та екосистема Hadoop	90
Встановлення Hadoop на кластер за допомогою Cloudera Manager	90
Розгортання розподіленого середовища на базі Apache Hadoop	91
Порядок виконання роботи.....	93
Встановлення Hadoop в автономному режимі	96
Питання до практичної роботи.....	97
Завдання для самостійної роботи	97
Література	98
11. Програмування MapReduce.....	99
Програмування задачі WordCount.....	99

Map only job	104
Ланцюжки MapReduce-задач.....	105
Distributed cache	106
Reduce Join	106
MapJoin	108
Питання до практичної роботи:.....	110
Завдання на самостійну роботу	110
Література	111
12. СУБД MongoDB	112
Встановлення сервера і рядкового клієнту.....	113
Деякі команди рядкового клієнта.....	113
Імпорт бази даних із таблиці у зовнішньому форматі.....	116
Експорт бази даних в таблицю у зовнішньому форматі	117
Питання до практичної роботи.....	117
Завдання на самостійну роботу	118
Література	118
13. Клієнт Compass СУБД MongoDB	119
Встановлення графічного інтерфейсу користувача Compass	119
Запуск GUI MongoDB Compass	120
Реалізація команд CRUD	121
Питання до практичної роботи.....	124
Завдання для самостійної роботи.....	124
Література	124
14. Основи роботи з Neo4j.....	125
Встановлення Neo4j.....	126
Запуск системи.....	129
Створення графа	131
Імпорт даних із формату .csv в граф neo4j	134
Додаток. Необхідний мінімум з мови Python.....	137

Типи даних	137
Модуль datetime	152
Модуль sys	154
Модуль NumPy.....	158
Модуль string.....	160
Matplotlib: Наукова графіка в Python	161
PyWavelets : вейвлет-перетворення в Python	166
Предметний покажчик	167

Вступ

На цей час в галузі кібернетичної безпеки все більшу роль відіграє поняття «великих даних» (Big Data). Фахівці, що займаються обробкою, агрегацією великих обсягів даних, вирішенням проблем, обумовлених їх зростанням, динамікою, варіативністю на цей час називають «вченими з обробки даних» (Data Scientists), відповідно, наука – Data Science.

Великі дані – це термін, що характеризує множину наборів даних настільки об'ємних і складних, що унеможливорює застосування наявних традиційних інструментів управління базами даних і додатків для їх обробки. Проблема представляють збирання, очищення, зберігання, пошук, доступ, передача, аналіз і візуалізація таких наборів як цілісної сутності, а не локальних фрагментів. В якості визначальних характеристик для великих даних відзначають «три V»: обсяг (Volume, в сенсі величини фізичного обсягу), швидкість (Velocity, що означає в даному контексті швидкість приросту і необхідність високошвидкісної обробки і отримання результатів), різноманіття (Variety, в сенсі можливості одночасної обробки різних типів структурованих і напівструктурованих даних).

Термін Big Data вперше з'явився в редакційній статті Кліффорда Лінча, редактора журналу Nature, 3 вересня 2008 року, який присвятив цілий спеціальний випуск того журналу темі "що можуть значати для сучасної науки набори великих даних".

Розвиток напрямку Big Data пов'язано з розвитком соціальних мереж, незважаючи на те, що великі обсяги даних притаманні таким також галузям, як телекомунікаційна, енергетична, транспортна тощо. І однією з перших інформаційних технологій в сфері безпеки і оборони, пов'язаних з цією проблемою є OSINT (Open Source Intelligence, розвідка у відкритих джерелах), яка є складовою частиною кібербезпеки.

На цей час існує ряд публікацій, пов'язаних із роллю Data Science, необхідністю підготовки фахівців з цього напрямку, разом з цим комплексного навчального курсу, пов'язаного з реальною практичною базою, яку побудовано на сучасному вільному програмному забезпеченні, досі не існувало.

В рамках курсу «Оброблення надвеликих масивів даних (Big Data)» розглядаються базові, найпоширеніші сьогодні технології і інструменти в області кібербезпеки, перелік яких дозволяє отримати цілісне уявлення про те, що використовують сьогодні фахівці в області Data Science та про інструменти, якими необхідно володіти, щоб вести проекти з використанням «великих даних».

В рамках навчальної дисципліни розглядаються:

- можливості технологій аналізу надвеликих масивів даних для вирішення проблем кібербезпеки, а також можливостей застосування наукових методів, у тому числі методів інтелектуального аналізу даних, до «великих даних»;
- особливості архітектурних рішень при створенні та розгортанні систем обробки «великих даних», а також вибору технології зберігання й обробки великих даних, використання сучасних високопродуктивних систем зберігання й обробки великих даних;
- основні технології і інструменти роботи з великими даними: Elastic Stack, Elasticsearch, Kibana, Hadoop, MapReduce, Neo4j, MongoDB;
- компоненти програмного забезпечення, необхідні для роботи в розподілених інформаційних системах обробки «великих даних».

Як основний засіб агрегації великих обсягів неструктурованих даних, їх пошуку, обробки візуалізації в рамках цього курсу розглядається екосистема компонентів Elastic Stack. Elasticsearch – це інформаційно-пошукова система, ядро Elastic Stack, яка дозволяє здійснювати обробку неструктурованих

даних, інформаційний пошук, аналіз даних, забезпечує підтримку призначених для користувача бібліотек і REST API; легке управління і масштабування. Утиліта Kibana – це вікно в Elastic Stack, засіб маніпуляції, аналізу і візуалізації інформації, що реалізує такі види відображення даних із Elasticsearch, як гістограми, карти, лінійні графіки, часові ряди.

Посібник також містить відомості щодо технологічних платформ оброблення «великих даних», серед яких система Apache Hadoop, призначена для організації розподіленого оброблення великих об'ємів даних, MapReduce – технологія розподіленого паралельного оброблення великих масивів даних з використанням великого числа обчислювальних кластерів.

Окремо розглядаються засоби аналізу великих мереж, графових СКБД. Розглядаються можливості двох основних систем – програм аналізу та візуалізації графів Gephi і графова система керування базами даних Neo4j. Серед особливостей програми Gephi вивчаються інтерфейс користувача, можливості компонування графів, фільтрація, дослідження даних, візуалізація, підтримка графічних форматів даних. Графова СУБД Neo4j забезпечує збереження і обробку мережових даних великих обсягів, містить декларативну мову запитів до графів Cypher.

Таким чином, у цьому посібнику обґрунтовано і представлено основні положення навчального курсу «Оброблення надвеликих масивів даних» як введення в спеціальність Data Science в сфері кібербезпеки, на основі вивчення теоретичних і технологічних основ цієї спеціальності, практичного застосування відповідних інформаційних технологій агрегації великих обсягів даних.

1. Формування масиву даних для аналітичної обробки

Мета практичної роботи, що відповідає розділу, – дослідження та збір «великих даних» у визначених форматах із соціальних мереж (RSS-фідів), збереження та підготовка їх для подальшої аналітичної обробки.

Розвиток напрямку обробки надвеликих масивів даних (Big Data) насамперед пов'язаний з розвитком інформаційної складової мережі Інтернет, а саме, веб-ресурсів і соціальних мереж. Саме такий контент розглядається як своєрідний полігон для проведення практичних робіт в рамках курсу, що вивчається. Для подальшої інтелектуальної обробки таких інформаційних ресурсів необхідно на першому етапі створити систему її збирання, але вирішення цієї цікавої задачі виходить за межі навчального курсу.

Саме тому, для первинного збирання даних для її подальшої інтелектуальної обробки обмежимося лише одним видом неструктурованої текстової інформації, яка вільно доступна в мережі Інтернет і є, мабуть, однією із найбільш стандартизованих, а саме інформацією, наведеною в форматі RSS.

У кінці XX століття для вирішення завдання уніфікації контенту мережі Інтернет з метою її подальшого оброблення програмними застосунками, узагальнення і подальшого розподілу (синдикації) було створено декілька форматів опису даних на основі стандарту XML. Найпоширеніший формат отримав назву RSS, що означає Really Simple Syndication, Rich Site Summary, хоча спочатку він називався RDF Site Summary. Зміст всіх цих аббревіатур полягає в простому способі узагальнення та розподілу інформаційного наповнення веб-сайтів – синдикації контенту.

Розробка RSS, почалася 1997 року. Визнання ця технологія здобула, коли компанія Netscape використовувала її для наповнення каналів свого порталу Netcenter. Незабаром технологія RSS стала використовуватися для

трансляції контенту на багатьох новинних сайтах – у тому числі таких як BBC, CNET, CNN, Disney, Forbes, Wired, Red Herring, Slashdot, ZDNet та багатьох інших. Першою відкритою офіційною версією RSS стала версія 0.90. Формат був заснований на RDF (Resource Description Framework – стандарт схеми опису ресурсів).

Сьогодні практично всі провідні веб-сайти, блоги, деякі соціальні мережі, що працюють в Інтернет, використовують RSS як інструмент оперативного представлення своїх оновлень. Специфікації окремих версій формату RSS наведені на таких веб-сторінках:

RSS 0.90: <http://www.purplepages.ie/RSS/netscape/rss0.90.html>

RSS 0.91: <http://my.netscape.com/publish/formats/rss-spec-0.91.html>

RSS 1.0: <http://web.resource.org/rss/1.0/>

RSS 2.0: <http://backend.userland.com/rss/>

Термін «RSS» (Really Simple Syndication) часто використовується як загальна назва усіх веб-стрічок, включаючи ті, що мають формат, відмінний від RSS (наприклад, для стрічок у форматі Atom).

Формат даних

RSS базується на стандарті XML. Нижче наведено основні відомості щодо RSS 2.0. Перший тег в RSS-документі обов'язково вказує на формат XML, що застосовується. Після нього йде тег <rss> з обов'язковим атрибутом version, який вказує на версію документа.

У обов'язковому порядку RSS-документ містить 2 теги: <channel> і <item>.

Основну інформацію про цей RSS-канал містить тег <channel>. Він зустрічається лише 1 раз.

У обов'язковому порядку він містить такі три теги:

<title> – ім'я каналу. Може збігатися із ім'ям сайту.

<link> – Посилання на сайт, з яким пов'язаний канал.

<description> – опис каналу.

Опціональні теги:

<language> – мова стрічки.

<copyright> – авторські права.

<managingEditor> – електронна пошта редактора вмісту каналу

<webMaster> – електронна пошта веб-майстра каналу

<pubDate> – дата публікації каналу.

<lastBuildDate> – дата останньої зміни вмісту каналу.

<category> – категорії контенту каналу.

<generator> – програма, за допомогою якої було згенеровано канал.

<docs> – посилання на документацію використовуваного формату RSS.

<ttl> – час актуальності каналу в хвилинах.

<image> – зображення, яке відображається з каналом. У свою чергу, тег має такі параметри:

<Title> – заголовок.

<Description> – опис (аналог тега ALT в HTML).

<Link> – посилання на сайт, з яким пов'язаний канал.

<URL> – адреса зображення.

<Width> – ширина зображення.

<Height> – висота зображення.

<skipHours> – скільки годин не вимагати оновлення з каналу

<skipDays> – скільки днів не вимагати оновлення з каналу

Тег <item> містить інформацію про публікацію.

Обов'язкові вкладені теги:

<title> – назва публікації.

<link> – посилання на сторінку з повним текстом публікації.

<description> – короткий текст публікації.

Необов'язкові вкладені теги:

<Author> – електронна пошта автора

<Category> – категорія повідомлення

<Comments> – посилання на сторінку з коментарями до повідомлення

<Enclosure> – приєднаний мультимедійний об'єкт. Його параметри:

<URL> – адреса об'єкта

<Length> – розмір об'єкта в байтах

<Type> – MIME-тип файлу


<Guid> – ідентифікатор повідомлення

<PubDate> – дата публікації.

Приклад:

```
<?xml version="1.0" encoding="windows-1251" ?>
<rss version="2.0">
<channel>
<title>Мої новини</title>
<link>http://site.ua</link>
<description>Опис моїх новин</description>
<image>
<url>http://site.ua /moiLogotip.gif</url>
<link>http://site.ua </link>
<title>Мої новини</title>
</image>
<lastBuildDate>26 feb 2021 11:11:11 +0300</lastBuildDate>
<item>
<title>З'явилась нова публікація!</title>
<link>http://site.ua /?str=news</link>
<description>Сьогодні з'явилась нова стаття щодо «великих
даних»</description>
<pubDate>26 feb 2021 11:11:11 +0300</pubDate>
<guid> http://site.ua /?str=news</guid>
</item>
</channel>
</rss>
```

RSS-фід

Масиви документів, які надаються на ресурсах веб-серверів у форматі RSS мають назви «RSS-фіди» (від англ. Feed – годувати, постачати), канали інформації. Адреси цих фідів задаються у явному вигляді на сторінках веб-сайтів (стандартне позначення – ) , або їх можна знайти у вихідному коді HTML сторінок, наприклад, як у фрагменті першої сторінки веб-сайту агентства новин УНІАН:

```
<a class="footer-menu__icon " target="_blank" rel="noopener"
aria-label="rss" href="https://rss.unian.net/site/news_rus.rss">
<i class="unian-rss"></i></a>
```

Процедури завантаження даних

Для автоматичного завантаження RSS-фіди зазвичай можна використовувати стандартні утиліти завантаження інтернет-контенту, яких існує дуже багато, наприклад програми cURL або wget.

cURL — це утиліта для організації вибірки даних з веб-сайтів, яка надає можливість оперувати з файлами на боці веб-серверу за допомогою параметрів, що можуть бути переданими в рядку URL. За допомогою cURL можна отримувати веб-сторінки, не використовуючи для цього браузер. Крім HTTP-запитів, cURL підтримує SMTP, IMAP, Telnet, FTP та інші мережні протоколи. Базове використання cURL полягає у простому наборі у командній консолі команди cURL, за якою іде URL для завантаження. cURL за замовчуванням відображає вивід отриманого у стандартний потік виводу системи, тобто виклик cURL покаже програмний код сторінки в вікні терміналу.

Програма cURL може записати вивід до файлу при завданні опції «-o»:

```
curl -o example.html www.example.com
```


Такий виклик забезпечує збереження коду титульної сторінки `www.example.com` у файлі `example.html`.

Wget – це консольна утиліта для завантаження файлів за протоколами HTTP, HTTPS та FTP. wget дає змогу рекурсивно завантажувати мережеві файли, копіювати як окремі сторінки, так і цілі веб-сайти, конвертувати посилання тощо. Ця утиліта портована й запускається на багатьох UNIX-подібних системах, Microsoft Windows, MacOS X тощо. wget — не інтерактивна програма, тобто, після того, як її запущено з певними параметрами, вона виконує всі необхідні дії і не потребує додаткового втручання у свою роботу. Wget може працювати як пошуковий робот, тобто отримувати ресурси, на які посилаються елементи HTML сторінки, й рекурсивно просуватися web-деревом, доки всі необхідні файли не будуть завантажені.

Процедури скачування RSS-фідів (каналів інформації) мають вигляд:

```
curl -o habr https://habr.com/ru/rss/all/all/
```

або

```
wget -O habr https://habr.com/ru/rss/all/all/
```

Нижче наведено список RSS-фідів, які можна використовувати для формування масиву даних для подальших практичних робіт.

```
http://economictimes.indiatimes.com/tech/software/rssfeeds/13357555.cms  
http://nypost.com/tech/feed  
http://www.smh.com.au/rssheadlines/technology-news/article/rss.xml  
http://zeenews.india.com/rss/science-technology-news.xml  
http://www.washingtontimes.com/rss/headlines/culture/technology/  
https://www.thehindu.com/sci-tech/technology/?service=rss  
https://www.economist.com/science-and-technology/rss.xml  
https://feeds.skynews.com/feeds/rss/technology.xml  
http://www.innertemplelibrary.com/category/secure-hospitals/feed/  
http://e-news.com.ua/export/news.xml  
http://fakty.ua/rss_feed/science
```

Приклад фіду:

```
<rss version="2.0">
<channel>
<title>Наука</title>
<link>https://fakty.ua/rss_feed/science</link>
<description>Новини науки та технологій</description>
<language>uk</language>
<image>
<link>https://fakty.ua/images/</link>
<url>https://fakty.ua/images/logo_uk.png</url>
<title>Наука</title>
</image>
<item>
<title>
<![CDATA[
У додатку «Дія» з'явилася нова функція, яка допоможе українцям
у боротьбі із шахраями
]]>
</title>
<link>
<![CDATA[
https://science.fakty.ua/393252-v-prilozhenii-diya-poyavilas-novaya-
funkciya-kotoraya-pomozhet-ukraincam-v-borbe-s-moshennikami
]]>
</link>
<description>
<![CDATA[
Вона працює за замовчуванням, оновлювати «Дію» не потрібно
]]>
</description>
<category>
<![CDATA[ Наука та технології ]]>
</category>
<guid>
https://science.fakty.ua/393252-v-prilozhenii-diya-poyavilas-novaya-
funkciya-kotoraya-pomozhet-ukraincam-v-borbe-s-moshennikami
</guid>
<pubDate>Tue, 04 Jan 2022 22:30:00 +0200</pubDate>
</item>
<item>
<title>
<![CDATA[
Спойлери, реакції, QR-коди: Telegram випустив передноворічне оновлення
]]>
</title>
<link>
<![CDATA[
https://science.fakty.ua/392950-spojleri-reakcii-qr-kody-telegram-
vypustil-prednovogodnee-obnovlenie
]]>
</link>
<description>
<![CDATA[ З'явилися нові корисні функції ]]>
</description>
```

```

<category>
<![CDATA[ Наука та технології ]]>
</category>
<guid>
https://science.fakty.ua/392950-spojlery-reakcii-qr-kody-telegram-
vypustil-prednovogodnee-obnovlenie
</guid>
<pubDate>Thu, 30 Dec 2021 16:58:00 +0200</pubDate>
</item>
>Tue, 28 Dec 2021 16:47:00 +0200</pubDate>
</item>
<item>
<title>
<![CDATA[
Недоступні повідомлення та коментарі: у Telegram стався масштабний
збій
]]>
</title>
<link>
<![CDATA[
https://science.fakty.ua/392723-nedostupny-soobcscheniya-i-kommentarii-
v-telegram-proizoshel-masshtabnyj-sboj
]]>
</link>
<description>
<![CDATA[ Користувачі помітили збій увечері 27 грудня ]]>
</description>
<category>
<![CDATA[ Наука та технології ]]>
</category>
<guid>
https://science.fakty.ua/392723-nedostupny-soobcscheniya-i-kommentarii-
v-telegram-proizoshel-masshtabnyj-sboj
</guid>
<pubDate>Mon, 27 Dec 2021 20:10:00 +0200</pubDate>
</item>
<item>
...
</channel>
</rss>

```

Утиліти cURL або wget можна викликати за розкладом, наприклад, використовуючи засоби crontab. При цьому у подальшому на етапі обробки скачаних фалів треба передбачити заборону урахування документів, що мають в мережі однакові адреси (тег <link> у RSS-фіді).

Питання до практичної роботи:

1. Що таке RSS і як він використовується?
2. Що таке RSS канал?

3. З чого складається RSS?
4. У чому різниця між веб-стрічками та RSS?

Завдання на самостійну роботу

1. Розширити список RSS-фідів каналами комп'ютерної і телекомунікаційної спрямованості (але не торговельними майданчиками).
2. Створити процедуру (скрипт) для періодичного скачування інформації із створеного переліку RSS-фідів.
3. Реалізувати процедуру створення файлу, в якому об'єднуються всі скачані RSS-фіди, і підключити її до скрипту скачування.

Література

1. Dave Johnson. RSS and Atom in action: web 2.0 building blocks. – Manning Publications, 2006. – 398 pp. ISBN 9781932394498, 1932394494.
2. Michael Schrenk. Webbots, Spiders, and Screen Scrapers: A Guide to Developing Internet Agents with PHP/CURL. – No Starch Press, 2007. – 328 pp. ISBN 9781593271206, 1593271204.
3. <https://www.gnu.org/software/wget/>

2. Формування JSON-файлу для завантаження в Elasticsearch

Мета практичної роботи, що відповідає розділу, – дослідження та трансформація RSS-форматів «великих даних» у стандартні текстові JSON-файли для подальшого їх завантаження у пошукову систему Elasticsearch для індексування та пошуку будь-яких типів документів.

У попередньому розділі було сформовано інформаційну основу наступних практичних занять, файли за тематикою, що визначається інформаційними джерелами, завантажені в форматі RSS, який можна вважати обмінним, комунікаційним форматом.

Разом з цим, для подальшої роботи із сучасними пошуковими системами отриманні дані мають бути перетворені до вхідних форматів таких систем. Зокрема, для подальшої роботи із системою Elasticsearch, дані, що мають завантажуватись, повинні бути надані у форматі JSON.

Формат JSON

JSON (англ. JavaScript Object Notation – запис об'єктів JavaScript) – це текстовий формат обміну даними між комп'ютерами. JSON базується на тексті, може бути прочитаним людиною. Формат дає змогу описувати об'єкти та інші структури даних. Цей формат використовується переважно для передачі структурованої інформації через мережу.

JSON з'явився через необхідність обміну даними із сервером у реальному часі без використання плагінів для браузерів, flash-застосунків або Java-апплетів. За рахунок своєї лаконічності в порівнянні з XML, формат JSON є більш придатним для представлення складних структур.

JSON будується на двох структурах:

- Набір пар назва/значення. У різних мовах програмування це реалізовано як об'єкт, запис, структура, словник, хеш-таблиця, список із ключем або асоціативним масивом.

- Впорядкований список значень. У багатьох мовах це реалізовано як масив, вектор, список або послідовність.

Структури даних, що використовуються JSON, підтримуються будь-якою сучасною мовою програмування, що дозволяє застосовувати JSON для обміну даними між різними мовами програмування і програмними системами.

Як значення в JSON можуть бути використані:

- запис — це неупорядкована множина пар «ключ-значення», укладене у фігурні дужки «{ }». Ключ описується рядком, між ним та значенням стоїть символ «:». Пари ключ-значення відокремлюються один від одного комами.
- масив (одномірний) — це впорядкована множина значень. Масив записується у квадратних дужках «[]». Значення поділяються комами. Масив може бути порожнім, тобто не містити жодного значення. Значення не більше одного масиву можуть мати різний тип.
- число (ціле або дійсне).
- літерали true (логічне значення «істина»), false (логічне значення «хибне») та null.
- рядок — це впорядкована множина з нуля або більше символів юнікоду, укладена в подвійні лапки. Символи можуть бути вказані з використанням escape-послідовностей, що починаються зі зворотної косої риси "\" (підтримуються варіанти \\", \/, \t, \n, \r, \f і \b), або записані шістнадцятковим кодом у кодуванні Unicode у вигляді \uFFFF.

«Рядок» дуже схожий на літерал однойменного типу даних у мові JavaScript. «Число» теж дуже схоже на JavaScript-число, за винятком того, що використовується лише десятковий формат (з точкою як роздільник).

Пробіли можуть бути вставлені між будь-якими двома синтаксичними елементами.

Наступний приклад показує JSON-подання даних про об'єкт, що описує людину. У даних присутні рядкові поля імені та прізвища, інформація про адресу та масив, що містить список телефонів. Як видно з прикладу, значення може бути вкладеною структурою.

```
{
  "firstName": "Іван",
  "lastName": "Іванов",
  "address": {
    "streetAddress": "пров. П.Мирного, 10, кв.101",
    "city": "Суми",
    "postalCode": 10101
  },
  "phoneNumbers": [
    "067 123-1234",
    "050 123-4567"
  ]
}
```

Треба звернути увагу на пару "postalCode": 10101. В якості значень JSON можуть бути використані як числа, так і рядки. Тому запис "postalCode": "10101" містить рядок, а "postalCode": 10101 – числове значення. Через слабку типізацію в Javascript і PHP рядок може бути приведений до числа і не впливати на логіку програми. Тим не менш, рекомендується акуратно поводитися з типом значення, оскільки JSON служить для міжсистемного обміну.

Система Elasticsearch не нав'язує чітку структуру даних; можна зберігати будь-які документи JSON, які, зокрема, підходять для Elasticsearch на відміну від рядків та стовпців у реляційних базах даних. Такий документ можна порівняти із записом у таблиці БД. У традиційних реляційних базах даних таблиці структуровані: мають фіксовану кількість стовпців, кожен має свій тип і розмір. Але дані можуть динамічно змінюватися, що вимагатиме підтримки нових або динамічних стовпців. Документи JSON спочатку підтримують такий тип даних.

Наведений нижче ще один приклад показує JSON представлення об'єкта, що описує клієнта:

```
{
  "name": "John Smith",
  "address": "121 John Street, NY, 10010",
  "age": 40
}
```

Так може виглядати запис клієнта. У ній вказано ім'я, адресу та вік клієнта. А інший запис може виглядати так:

```
{
  "name": "John Doe",
  "age": 38,
  "email": "john.doe@company.org"
}
```

Треба звернути увагу, що другий клієнт не має поля для адреси, але замість нього є поле для електронної пошти. А в інших клієнтів може бути зовсім інший набір полів. Отже, реалізована гнучкість щодо того, що може зберігатися в таблиці.

Для завантаження в систему Elasticsearch необхідно отриманий на попередній практичній роботі файл перетворити з формату RSS, а саме створити формат JSON. Пропонується такий спрощений перелік полів документа, які надалі мають завантажуватися до системи Elasticsearch:

"title" – назва повідомлення;

"textBody" – текст повідомлення;

"source" – джерело повідомлення;

"pubDate" – дата і час у форматі YYYYMMDD HH:MM

"url" – адреса повідомлення в мережі Інтернет.

Нижче наведено фрагмент файлу в форматі JSON, який має бути отримано:

```
{
  "title": " У додатку «Дія» з'явилася нова функція, яка
допоможе українцям у боротьбі із шахраями",
  "textBody": " Вона працює за замовчуванням, оновлювати «Дію»
не потрібно",
```



```

"source": "Факти",
"PubDate": "2021-12-28T10:04:00Z",
"URL": "https://science.fakty.ua/393252-v-prilozhenii-diya-poyavilas-
novaya-funkciya-kotoraya-pomozhet-ukraincam-v-borbe-s-moshennikami"
}
,
{
"title": "Суд у смартфоні: «Дія» тестує новий сервіс",
"textBody": "Через додаток хочуть надсилати повідомлення про
судові засідання",
"source": "Факти",
"PubDate": "2021-12-28T10:54:00Z",
"URL": "https://science.fakty.ua/392788-sud-v-smartfone-diya-
testiruet-novyyj-servis"
}
{
"title": "James Webb може переписати космічну історію: відео
запуску новітнього телескопу від NASA",
"textBody": "Роботу апарат розпочне влітку 2022 року",
"source": "Факти",
"PubDate": "2021-12-25T10:14:00Z",
"URL": "https://science.fakty.ua/392589-james-webb-mozhet-
perepisat-kosmicheskuyu-istoriyu-video-zapuska-novogo-teleskopa-
ot-nasa"
}

```

Програма перетворення RSS в JSON

Практичним завданням, що вирішується в рамках цього розділу, є розробка програми мовою Python, яка перетворює (конвертує) інформацію, надану у форматі RSS, до формату JSON, не змінюючи зміст цієї інформації. Нижче наведено приклад такої програми (пояснення надані у вигляді коментарів у тексті програми, після знаку #):

```

#Програма конвертації даних із формату RSS до формату JSON
#Підключення модулів для роботи із регулярними виразами і часом
import re
import datetime

#Відкриття файлу rss.xml у режимі «читання»
f = open("rss.xml", "r")

#Зчитування вмісту файлу rss.xml у змінну t
t = f.read()

```

```

#Закриття файлу rss.xml
f.close()

#Розбиття файлу по рядкам і склеювання рядків через пропуск
rss = t.split('\n')
t=""
for i in range(len(rss)):
    t=t+" "+rss[i]

#Видалення перших пропусків
t=re.sub('^\s+', '', t)

#Формування масиву заголовків
title = re.findall('<title>(.*?)</title>', t)

#Перший заголовок - назва фіду, далі - його специфічна обробка
source=title[0]
source=re.sub('[\s-]*$', '', source)
source=re.sub("'", '\'', source)

#Розмірність масиву заголовків
x=range(len(title))

#Формування масиву описів
text = re.findall('<description>(.*?)</description>', t)

#Формування масиву гіперпосилань
link = re.findall('<link>(.*?)</link>', t)

#Формування дати і часу в форматі "YY-MM-MMTTHH:MM:00Z"
now = datetime.datetime.now()
tim=now.strftime("%Y-%m-%dT%H:%M:00Z")

#Виведення результатів
for i in range(1, len(title)):
    print "{\n"title\":"+" "+title[i]+"\", "

    #Специфічна обробка тексту
    text[i]=re.sub('[\s-]*$', '', text[i])
    text[i]=re.sub("'", '\'', text[i])
    text[i]=re.sub('\'', '&quot;', text[i])

    #Подальша виведення результатів
    print "\"textBody\":"+" "+text[i]+"\", "
    print "\"source\":"+" "+source+"\", "
    print "\"PubDate\":"+" "+tim+"\", "
    print "\"URL\":"+" "+link[i]+"\""}

```

```
if i<len(title)-1:  
    print ", "
```

У результаті виконання наведеної програми має сформуватися JSON-файл, придатний для завантаження у середовище інформаційно-пошукової системи Elasticsearch.

Питання до практичної роботи:

1. Що таке JSON-формат?
2. Які основні переваги JSON-формату?
3. Які ви знаєте правила створення структури JSON-файлу в об'єкті, масиві і при присвоєнні значення?
4. Що таке пошукова система Elasticsearch та її призначення?

Завдання на самостійну роботу

1. Написати програму формування пакетного файлу в форматі JSON.
2. Встановити на комп'ютері бібліотеку для роботи із регулярними виразами у середовищі мови програмування (Python).
3. Ознайомитися із основними можливостями мови програмування Python щодо роботи із строковими даними.

Література

1. Основы Data Science и Big Data. Python и наука о данных / Дэви Силен, Арно Мейсман. – Питер, 2017. – 336 с.
2. Пранав Шукла, Шарат Кумар. Elasticsearch, Kibana, Logstash и поисковые системы нового поколения. – Питер, 2019. – 363 с.
3. Бонцанини М. Анализ социальных медиа на Python / пер. с англ. А. В. Логунова. – М.: ДМК Пресс, 2018. – 288 с.
4. С. Шапошникова. Основы программирования на Python. Вводный курс. – Лаборатория юного линуксоида, 2011. – 44 с.

3. Встановлення системи Elasticsearch. Завантаження інформації в БД

Мета практичної роботи, що відповідає розділу, – навчитися встановлювати систему Elasticsearch та завантажувати інформацію отриману в попередній роботі у форматі JSON в базу даних пошукової системи Elasticsearch для індексування та пошуку документів різних типів.

Цей розділ присвячено встановлення і вивченню можливостей системи Elasticsearch, інформаційно-пошукової системи, що забезпечує накопичення і пошук в великих обсягів даних. Ця система входить до стеку технологій Elastic.

Стек Elastic

За останні кілька років з'явилися різні системи зберігання та обробки великих масивів даних. Серед них можна виділити компоненти екосистеми Elastic, які служать для пошуку та обробки даних. Основні компоненти Elastic Stack – це Kibana, Logstash, Beats, X-Pack та Elasticsearch. Ядром Elastic Stack є пошукова система Elasticsearch, яка надає можливості для зберігання, пошуку та обробки даних. Утиліта Kibana, яку також називають вікном в Elastic Stack, є відмінним засобом візуалізації та інтерфейсом користувача Elastic Stack. Компоненти Logstash та Beats дозволяють передавати дані до Elastic Stack. X-Pack надає потужний функціонал: можна налаштовувати моніторинг, додавати різні повідомлення, встановлювати параметри безпеки для підготовки вашої системи до експлуатації. Оскільки Elasticsearch є ядром Elastic Stack.

Elasticsearch – високмасштабована розподілена пошукова система повнотекстового пошуку та аналізу даних, що працює в режимі реального часу. Утиліта дозволяє зберігати, шукати та аналізувати великі обсяги даних.

Зазвичай використовується як базовий механізм/технологія, допомагаючи програмам зі складними функціями пошуку. Elasticsearch є основним компонентом Elastic Stack.

Elasticsearch як серце Elastic Stack відіграє основну роль у пошуку та аналізі даних. Вона побудована на унікальній технології – Apache Lucene. Завдяки цьому Elasticsearch докорінно відрізняється від традиційних рішень для реляційних баз даних або NoSQL. Нижче перераховані основні переваги використання Elasticsearch як сховища даних:

- неструктурованість, документоорієнтованість;
- можливість пошуку;
- можливість аналізу даних;
- підтримка користувацьких бібліотек та REST API;
- легке управління та масштабування;
- робота у псевдореальному часі;
- висока швидкість роботи;
- стійкість до помилок та збоїв.

Основні властивості Elasticsearch

Elasticsearch забезпечує зберігання даних, надає можливості пошуку та аналізу у масштабованому вигляді.

Система Elasticsearch має чітку орієнтацію на документи. Найкраще для неї підходять JSON-документи. Вони організовані за допомогою різних типів та індексів. Ключові поняття Elasticsearch:

- індекс;
- тип;
- документ;
- кластер;
- вузол;

- шарди та копії;
- розмітку та типи даних;
- зворотний індекс.

Індекс – це контейнер, який в Elasticsearch зберігає документи одного типу та керує ними. Індекс може містити документи одного типу.

Індекси в Elasticsearch приблизно аналогічні за структурою базам даних у реляційних базах даних. Тип Elasticsearch відповідає таблиці, а документ – запису в ній.

Типи допомагають логічно групувати чи організовувати однотипні документи за індексами.

Зазвичай, документи з найбільш поширеним набором полів, що групуються під одним типом. Elasticsearch не вимагає наявності структури, дозволяючи зберігати будь-які документи JSON з будь-яким набором полів під одним типом.

JSON-документи найкраще підходять для використання в Elasticsearch. Документ JSON складається з декількох полів і є базовою одиницею інформації, що зберігається в Elasticsearch.

Elasticsearch – неструктурована система, завдяки чому можна зберігати документи з будь-якою кількістю і типами полів.

Elasticsearch підтримує широкий набір типів даних для різних сценаріїв зберігання текстових даних, чисел, булевих, бінарних об'єктів, масивів, геоточок, геоформ та багатьох інших спеціалізованих типів даних, наприклад адрес IPv4 і IPv6.

Документи JSON зазвичай містять кілька полів. У документах JSON кожне поле має певний тип. Кожне поле та його значення можна побачити як пару «ключ – значення» у документі, де ключ – це ім'я поля, а значення – значення поля.

Elasticsearch – розподілена система. Вона охоплює множини процесів, запущених на різних пристроях у мережі та взаємодіючих з іншими процесами.

Вузол Elasticsearch – це одиничний сервер системи, який може бути частиною великого кластера вузлів. Він бере участь у індексуванні, пошуку та виконанні інших операцій, що підтримуються Elasticsearch. Кожному вузлу Elasticsearch у момент запуску надаються унікальний ідентифікатор та ім'я.

Кожному вузлу Elasticsearch відповідає основний конфігураційний файл. Формат файлу YML (повна назва - YAML Ain't Markup Language). Цей файл можна використовувати для зміни значень, таких як ім'я вузла, порти, кластер. На базовому рівні вузол відповідає одному запущеному процесу Elasticsearch. Він відповідає за управління відповідною частиною даних. Кластер містить один або кілька індексів і відповідає за виконання таких операцій, як пошук, індексування та агрегація. Кластер формується одним або декількома вузлами, кожен з яких відповідає за зберігання своєї частини даних та управління нею. Один кластер може зберігати один або кілька індексів. Індекс логічно групує різні типи документів.

Будь-який вузол Elasticsearch завжди є частиною кластера, навіть якщо це кластер одиничного вузла. За замовчуванням кожен вузол намагається приєднатися до кластера з ім'ям Elasticsearch. Якщо кілька вузлів запускаються всередині однієї мережі без зміни параметра `cluster.name` у файлі `config/elasticsearch.yml`, вони автоматично об'єднуються в кластер.

Шарди допомагають розподілити індекс за кластером. Вони розподіляють документи з одного індексу з різних вузлів. Обсяг інформації, який може зберігатися в одному вузлі, обмежується дисковим простором, оперативною пам'яттю та обчислювальними можливостями цього вузла. Шарди допомагають розподіляти дані одного індексу по всьому кластеру і

чим оптимізувати ресурси кластера. Процес поділу даних по шардах називається шардуванням. Це невід'ємна частина Elasticsearch, необхідна для оптимізації дискового простору з різних вузлів кластера, а також обчислювальної потужності з різних вузлів кластера.

Кожен шард індексу може бути налаштований таким чином, щоб він мав деяку кількість додаткових копій оригінального або первинного шарду для забезпечення високого рівня доступності даних.

Встановлення Elasticsearch

Для встановлення компонентів Elastic Stack – Elasticsearch та надалі системи Kibana, необхідно зайти на стартову веб-сторінку в мережі Інтернет (<https://www.elastic.co/start>, Рис. 3.1) та завантажити відповідний архів. Починаючи з версії Elastic Stack 5.x всі компоненти оновлюються разом і мають один номер версії. Це стосується компонентів Elastic Stack версії 7.x.

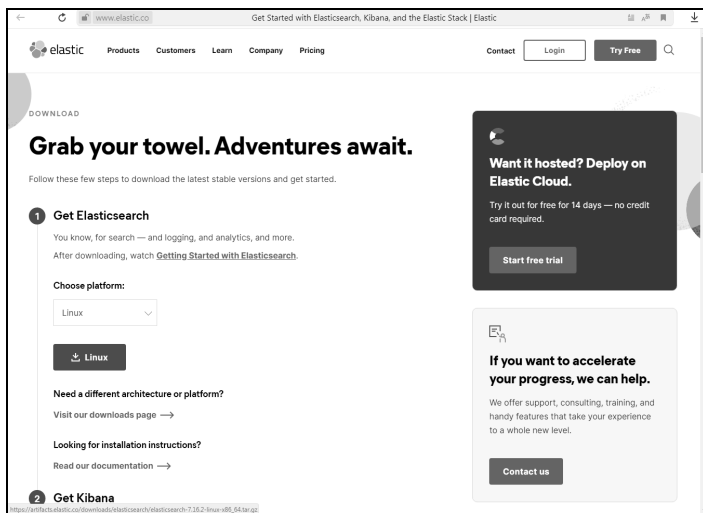


Рис. 3.1 – Стартова сторінка системи в Elasticsearch Інтернеті

Після скачування дистрибутиву у форматі ZIP (кнопка Linux), можна переглянути його вміст (Рис. 3.2).

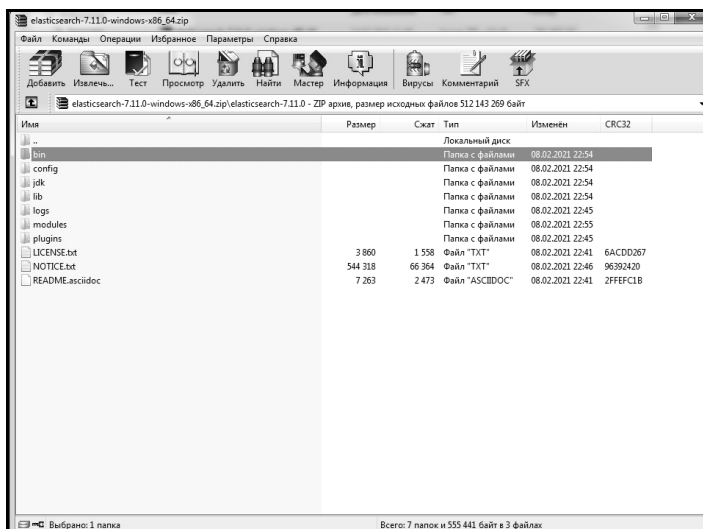


Рис. 3.2 – Вміст дистрибутиву системи в Elasticsearch

Для встановлення системи Elasticsearch на комп'ютері (сервері) достатньо розпакувати файли та перейти до створеної папки. Після цього можна запустити (в залежності від операційної системи) файли bin/elasticsearch або bin/elasticsearch.bat і система буде встановлена.

Після звернення через браузер або утиліту типу cURL за адресою curl <http://localhost:9200> можна побачити відгук системи Elasticsearch (Рис. 3.3).

Першою задачею, що виникає після встановлення системи Elasticsearch, це задача створення і завантаження бази даних JSON-документами, отримання яких описано в розділах 1 і 2.

Завантаження бази даних Elasticsearch

Для завантаження JSON-файлу можна застосувати утиліту cURL в режимі запису даних XPOST, шляхом вказівки адреси системи Elasticsearch (<http://localhost:9200>), назви індексу (test, задається довільно) та імені типу,

відповідно до якого документ має індексуватися (_doc, також задається довільно).

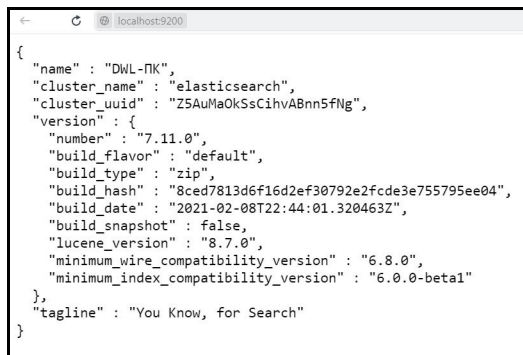


Рис. 3.3 – Звернення до системи за локальною адресою `http://localhost:9200`

Файл для завантаження документів (відрізняється від JSON-файлу наявністю команди `curl` з параметром `XPOST`, вказівки адреси системи) має вигляд:

```
curl -XPOST 'http://localhost:9200/test/_doc/' -H 'Content-Type: application/json' -d '{
  "title": "В Україні зазбоїв месенджер Telegram, проблеми також у користувачів Instagram",
  "textBody": "Проблеми з доступом до соцмереж відчують у багатьох країнах",
  "source": "Факти",
  "PubDate": "2021-12-04T18:51:00Z",
  "URL": "https://science.fakty.ua/390973-v-ukraine-zasboil-messendzher-telegram-problemy-takzhe-u-polzovatelej-instagram"
}'

curl -XPOST 'http://localhost:9200/test/_doc/' -H 'Content-Type: application/json' -d '{
  "title": "Зміна прописки через «Дію»: як взяти участь у тестуванні і що важливо знати",
  "textBody": "Участь у проекті беруть 3,5 тисяч добровольців",
  "source": "Факти",
  "PubDate": "2021-12-02T10:02:00Z",
  "URL": "https://science.fakty.ua/390775-smena-propiski-cherez-diyu-kak-prinyat-uchastie-v-nachavshemsya-testirovanii-i-hto-vazhno-znat"
}'
```

```
curl -XPOST 'http://localhost:9200/test/_doc/' -H 'Content-Type:
application/json' -d '
{
"title":"В Україні став доступний мобільний додаток, який «рятує» жінок від
насилства",
"textBody":"Додаток розробили в Ізраїлі",
"source":"Факти",
"PubDate":"2021-11-11T14:59:03Z",
"URL":"https://science.fakty.ua/389234-v-ukraine-stalo-dostupno-mobilnoe-
prilozhenie-spasayucshee-zhencshin-ot-nasiliya"
}'
```

Під час завантаження JSON-документів система Elasticsearch видає повідомлення (протокол завантаження) наступного вигляду:

```
{"_index":"test","_type":"_doc","_id":"JF37oXcButXogwwaYkt_","_version":
":1","result":"created","_shards":{"total":2,"successful":1,"failed":0}
,"_seq_no":0,"_primary_term":1}
{"_index":"test","_type":"_doc","_id":"JV37oXcButXogwwaYkvi","_version":
":1","result":"created","_shards":{"total":2,"successful":1,"failed":0}
,"_seq_no":1,"_primary_term":1}
. . .
{"_index":"test","_type":"_doc","_id":"Jl37oXcButXogwwaYkv6","_version":
":1","result":"created","_shards":{"total":2,"successful":1,"failed":0}
,"_seq_no":9,"_primary_term":1}
```

Перевірка результатів завантаження даних

Для перевірки кількості завантажених документів достатньо звернутися до системи Elasticsearch наступним чином:

```
curl -XGET 'http://localhost:9200/test/_doc/_count'
```

Результат виконання команди перевірки кількості завантажених документів має приблизно такий вигляд:

```
{"count":10,"_shards":{"total":1,"successful":1,
"skipped":0,"failed":0}}
```

Для перевірки коректності створення індексу вводиться запит (зписаний у JSON-форматі), згідно з яким необхідно знайти документ, у тілі якого (поле "textBody") присутнє слово "сервіс":

```
curl -XGET 'http:// localhost:9200/test/_doc/_search?pretty=true' -H
'Content-Type: application/json' -d '
{
  "query" : {
    "match" : { "textBody": "сервіс" }
  }
}'
```

При коректному створенні і заповненні бази даних буде отримано результат:

```
{
  "took" : 0,
  "timed_out" : false,
  "_shards" : {
    "total" : 1,
    "successful" : 1,
    "skipped" : 0,
    "failed" : 0
  },
  "hits" : {
    "total" : {
      "value" : 1,
      "relation" : "eq"
    },
    "max_score" : 2.3004205,
    "hits" : [
      {
        "_index" : "test",
        "_type" : "_doc",
        "_id" : "LF37oXcButXogwwaY0uW",
        "_score" : 2.3004205,
        "_source" : {
          "title" : "У додатку «Київ цифровий» з'явилися нові функції:
що важливо знати користувачам",
          "textBody" : "Також було вдосконалено деякі існуючі сервіси.",
          "source" : "факти",
          "PubDate" : "2021-11-08T15:40:00Z",
          "URL" : "https://science.fakty.ua/388979-v-prilozhenii-kiev-
cifrovoj-poyavilis-novye-funkcii-hto-vazhno-znat-polzovatelyam"
        }
      }
    ]
  }
}
```

Операції CRUD

CRUD – англ. Create, Read, Update, Delete – 4 основні функції керування даними «створення, читання, оновлення і вилучення».

У системі Elasticsearch операції CRUD націлені на документи. Для операцій CRUD реалізовано такі API:

- Index API;
- Get API;
- Update API;
- Delete API.

Питання до практичної роботи

1. У чому полягають особливості пошукового рушія Elasticsearch (на відміну від веб-механізмів пошукових систем в Інтернеті)?
2. Що таке повнотекстовий пошук та принцип його роботи?
3. Чим відрізняється повнотекстовий пошук від пошуку документів на основі метаданих?
4. Що таке CRUD-операції в системі Elasticsearch?
5. Які методи CRUD в системі Elasticsearch ви знаєте?

Завдання на самостійну роботу

1. Написати програму формування пакетного файлу для завантаження в Elasticsearch на основі сформованого на попередньому занятті файлу в форматі JSON.
2. Встановити на комп'ютері утиліту curl (<http://curl.se>) і вивчити склад і значення її основних параметрів.
3. Ознайомитися зі складом і змістом CRUD-операцій в Elasticsearch.

Литература

1. Пранав Шукла, Шарат Кумар. Elasticsearch, Kibana, Logstash и поисковые системы нового поколения. – Питер, 2019. – 363 с.
2. Elasticsearch: The Definitive Guide / Clinton Gormley and Zachary Tong. - O'Reilly Media, Inc., 2015. – 719 p.
3. Elasticsearch Blueprints. A practical project-based guide to generating compelling search solutions using the dynamic and powerful features of Elasticsearch / Vineeth Mohan. – Packt Publishing, 2015. – 192 p.

4. Запити до Elasticsearch і веб-інтерфейс

Мета практичної роботи, що відповідає розділу, – навчитися формувати пошукові запити в системі Elasticsearch та створювати веб-інтерфейс (CGI-процедури на мові Python) до бази даних Elasticsearch.

Запити до Elasticsearch

В попередньому розділі вже розглядався запит до бази даних за терміном, що входить до неструктурованого текстового поля.

У JSON-запиті у цьому випадку використовувалася опція відповідності (match query). Запит відповідності – це запит, який за умовчанням застосовується у режимі повнотекстового пошуку. Це один із високорівневих запитів, який передбачає використання аналізатора для вихідного поля.

Коли ми виконуємо запит на відповідність, очікуються наступні дії:

1. Пошук термінів по всіх документах у межах певного поля (наприклад, "textBody").

2. Знаходження найкращих відповідей, відсортованих за оцінкою в порядку зменшення частоти народження.

Коли кінцевий користувач шукає будь-які терміни, іноді потрібно виконувати запити по двох полях "textBody" (тіло документа) та "title" (назва). Це можливо за допомогою запиту на пошук у декількох полях.

Наступний запит знайде всі документи, які мають слово "cybersecurity" у полях назви або опису:

```
curl -XGET
'http://172.16.33.11:9200/_all/_search?pretty=true&size=100' -H
'Content-Type: application/json' -d '
{
  "query": {
    "bool": {
      "must": [
```

```

    {
        "multi_match": { "query": "cybersecurity",
        "fields": ["textBody", "title"]
        }
    }
]
}
}
}'

```

Нижче наведено ще один запит, який звертається до декількох полів за словосполученням «cyber security». Крім того у запиті на пошук здійснюється фільтрація за джерелом ("CTVNews.ca"):

```

curl -XGET
'http://172.16.33.11:9200/_all/_search?pretty=true&size=100' -H
'Content-Type: application/json' -d '
{
  "query": {
    "bool": {
      "must": [
        {
          "multi_match": { "query": "cyber security",
          "fields": ["textBody", "title"]
          }
        }
      ],
      "filter": [
        {
          "match": {
            "source": "CTVNews.ca"
          }
        }
      ]
    }
  }
}'

```

У результаті обробки наведеного запиту отримуємо:

```

agregator@cyber_agregator_search:~/elastic$ ./query45_test_sm
{
  "took" : 13,
  "timed_out" : false,
  "_shards" : {
    "total" : 113,
    "successful" : 113,

```



```

    "skipped" : 0,
    "failed" : 0
  },
  "hits" : {
    "total" : {
      "value" : 2413,
      "relation" : "eq"
    },
    "max_score" : 21.18586,
    "hits" : [
      {
        "_index" : "hb",
        "_type" : "_doc",
        "_id" : "6piHNnUBmqEEOXSaGYZ0",
        "_score" : 21.18586,
        "_source" : {
          "id" : "20191111102600.62_hb",

          "title" : "Nearly half of Canadians lack confidence
in cybersecurity of CRA, Elections Canada: survey",

          "textBody" : "Thursday, December 9, 2021 10:20PM EST
(Soumil Kumar / Pexels.com) Share: Reddit Share Text: According
to a new survey, nearly half of Canadians lack confidence in
the cybersecurity of Elections Canada and federal government
services such as the Canada Revenue Agency. More than 100
ransomware attacks targeted notable Canadian sites in 2021,
including hospitals and Rideau Hall, and in the wake of this,
Canadians are feeling a lack of confidence in institutions'
ability to protect from cyber threats, according to a report
released by Angus Reid Institute on Thursday. The report detailed
the results of a survey that found that more than half of
Canadians said they were "not confident" that their municipal
government had good cybersecurity, and half were not confident
that their local health authority had good cybersecurity.
. . .
This survey comes after federal ministers urged Canadians to
bolster their cybersecurity earlier this week, stating in an open
letter that Canadians should build a response plan. METHODOLOGY
The Angus Reid Institute conducted an online survey from Nov. 3-
7, 2021, among a representative randomized sample of 1,611
Canadian adults who are members of Angus Reid Forum. For
comparison purposes only, a probability sample of this size would
carry a margin of error of plus or minus 2.5 percentage points,
19 times out of 20.."
```

```

          "source" : "CTVNews.ca",
          "user" : "Cloud4Y",
          "pubDate" : "2021-12-10T07:09:00Z",

```

```
"URL" : " https://www.ctvnews.ca/canada/nearly-half-  
of-canadians-lack-confidence-in-cybersecurity-of-cra-elections-  
canada-survey-1.5701735"
```

```
    }  
  },  
  {  
    "_index" : "hb",  
    "_type" : "_doc",  
    "_id" : "oJmYNnUBmqEEOXSaKzGy",  
    "_score" : 17.96392,  
    "_source" : {  
      "id" : "20200507122600.104_hb",  
      "title" : "N.L. health officials confirm cyberattack  
causing health system interruptions",  
      "textBody" : "Newfoundland and Labrador's minister of  
health confirmed Wednesday the IT problems causing disruptions to  
the province's health-care system are a result of a cyberattack.  
"At this stage of our investigation, we can confirm we've been  
the victim of a cyberattack that has impacted our health-care  
systems," John Haggie said during a news conference. "We have  
engaged cyber security experts to help us investigate and resolve  
and we've informed the appropriate authorities." Haggie said  
the cyberattack was detected on Saturday, and teams are working  
around the clock to deal with the situation, but he said  
information was limited at this time. "Those involved in the  
attack may actually be monitoring what we are saying in media and  
on the floor of the house. It is very important that we don't do  
or say anything that compromised the efforts to investigate and  
resolve this matter," said Haggie. The province has activated  
it's provincial emergency operations centre, and each of the  
Regional Health Authorities is in a 'Code Grey' situation. Haggie  
said the worst system impacts are in the Western Health zone, and  
that system has been taken offline as of Wednesday morning.  
Eastern Health and Central Health also say they expect to  
continue cancelling thousands of non-emergency procedures  
Wednesday and Thursday. With files from the Canadian Press.  
Atlantic Top Stories Nova Scotia issues eight-week grace period  
for its vaccine mandate in public sector Thousands attend rally  
outside N.B. legislature in support of striking CUPE workers N.L.  
health officials confirm cyberattack causing health system  
interruptions Nova Scotia reports 11 new COVID-19 cases Tuesday,  
active cases drop to 161 N.B. reports one death, 40 new cases,  
and 75 recoveries Tuesday as active cases drop to 470 N.S.",  
      "source" : "CTVNews.ca",  
      "user" : "PhoenixContact",  
      "pubDate" : "2020-05-07T12:26:00Z",
```

```

        "URL" :
        "https://habr.com/ru/post/500864/?utm_source=habrahabr&utm_medium=rss&utm_campaign=500864"
    },
    . . .
}

```

Модель пошукової системи на базі веб-інтерфейсу

Систему Elasticsearch можна використовувати як пошуковий механізм, для пошукових систем, що встановлюються на веб-серверах, на яких функціонує можливість CGI.

CGI (від англ. Common Gateway Interface – Загальний інтерфейс шлюзу) – стандарт інтерфейсу, що використовується для зв'язку зовнішньої програми з веб-сервером. Програму, яка працює за таким інтерфейсом спільно з веб-сервером, прийнято називати шлюзом або «CGI-скриптом».

Сам інтерфейс розроблений таким чином, щоб можна було використовувати будь-яку мову програмування, яка може працювати зі стандартними пристроями вводу/виводу. Такі можливості мають навіть скрипти для вбудованих командних інтерпретаторів операційних систем.

Припустимо, що із застосуванням механізму CGI необхідно реалізувати веб-інтерфейс, наведений на Рис. 4.1.

Запит:

Джерело:

Рис. 4.1 – Веб-інтерфейс для введення запиту

Нижче наведено форму мовою HTML (CGI-форму) моделі пошукової системи:

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">

<title>RSS Aggregator</title>
</head>

<body bgcolor="white">

<table border=0 width=100%><tr>
<td width=170>

<form action="/cgi-bin/test_json.py" target="content">

Запит: <input type="text" name="query" value="">
<br>Джерело: <input type="text" name="src" value="">
<br><input type="submit" name="act" value="Старт">
</form>
</body>
</html>
```

Наведена CGI-форма містить виклик CGI-процедури на мові Python (test_json.py) із приймання параметрів і виклику запиту до системи Elasticsearch:

```
#!/usr/local/bin/python2.7
import os
import re
import cgi
import subprocess

form = cgi.FieldStorage()
query = form.getfirst("query", "Query not defined")
src = form.getfirst("src", "Source not defined")
os.system('"/usr/local/www/apache24/cgi-bin/test_json.sh" %s %s'
% (query, src))
```

Shell-скрипт, що забезпечує передачу запиту системі Elasticsearch (test_json.sh) і подальший виклик програми форматування відповіді (prog2_html):

```
#!/bin/sh
query=$1
src=$2
/usr/local/bin/curl -XGET
'http://172.16.33.11:9200/_all/_search?pretty=true&size=20' -H
'Content-Type: application/json' -d '
{
  "query": {
    "bool": {
      "must": [
        {
          "multi_match": { "query": "'$query'",
            "fields": ["textBody", "title"]
          }
        }
      ],
      "filter": [
        {
          "match": {
            "source": "'$src'"
          }
        }
      ]
    }
  }
}' | /usr/local/www/apache24/cgi-bin/prog2_html
```

Програма форматування JSON-файлу

В результаті опрацювання запиту користувача система Elasticsearch надає видачу результатів у вигляді JSON-файлу. Нижче наведено код програмного модулю мовою Python (prog2_html), який на основі отриманого JSON-файлу формує HTML-файл, що забезпечує вивід результатів у вигляді, представленою на Рис. 4.2.

3. : Українські сенсації. Увага! Вас слухають

Українські сенсації. Увага! Вас слухають українські сенсації, прослуховування, інформаційна безпека, інформаційна гігієна, україна політика, соцмережі безпека, особиста інформація соцмережі безпека,...

https://www.youtube.com/watch?v=7r_oqoirV38

4. : Як убезпечити себе від атаки хакерів – поради експерта з інформаційної безпеки

Як убезпечити себе від атаки хакерів – поради експерта з інформаційної безпеки українські сенсації, прослуховування, інформаційна безпека, інформаційна гігієна, україна політика, соцмережі безпека, особиста інформація соцмережі безпека,...

<https://www.youtube.com/watch?v=DWwOZq4aJ9U>

5. : У Києві стартували «Зброя та Безпека – 2021» і «Авіасвіт – XXI»

15 червня 2021 року у киеві розпочалися XVII міжнародна спеціалізована виставка «зброя та безпека – 2021» та ХІІ міжнародний авіакосмічний салон «авіасвіт – ххі», які традиційно проводяться в міжнародному виставковому центрі на броварському проспекті, 15. нові розробки у сфері оборони та авіакосмічної галузі представляють 332 підприємства, з яких – 35 закордонних компаній із 13 країн. Українську експозицію представляє: дк «укроборонпром» – 36 підприємств; ге «ліга оборонних підприємств України» – 23 підприємства; державне космічне агентство України – 2 підприємства, а також понад 130 інших державних і приватних підприємств

Рис. 4.2 – Фрагмент форми виведення результатів пошуку

Програмний модуль форматування виводу prog2_html має вигляд:

```
#!/usr/bin/python2.7
import sys
import re
import datetime

t = sys.stdin.read()
json = t.split('\n')
t=""
for i in range(len(json)):
    t=t+" "+json[i]
t=re.sub('^\s', '', t)
total = re.findall('"total" : (\d+)', t)
print "Content-type: text/html; charset=utf-8\n<html><body
bgcolor=yellow>"
print "<b>Found: "+total[0]+"</b><hr><ol>"
title = re.findall('"title" : "(.*)"', t)
text = re.findall('"textBody" : "(.*)"', t)
url = re.findall('"URL" : "(.*)"', t)
for i in range(len(title)):
    doc_ind=i+1
    print "<li><b>"+": "+title[i]+"</b>"
    print "<br>"+text[i]
    print "<br><i>"+url[i]+"</i><br /><hr>"
print "</ol></body></html>"
```

Питання до практичної роботи:

1. Які типи запитів формуються в Elasticsearch?
2. Як працює мова запитів DSL (Domain Specific Language) в системі Elasticsearch?
3. Як сортуються результати пошуку в Elasticsearch згідно з релевантністю кожного документа до запиту?
4. Яка роль JSON REST API в Elasticsearch?

Завдання на самостійну роботу

1. Забезпечити функціонування на комп'ютері (сервері) засобів інтерактивної взаємодії з користувачем (наприклад, CGI).
2. Ознайомитися із основними можливостями пошуку і фільтрації в Elasticsearch.
3. Ознайомитись з основами HTML-розмітки.

Література

1. Пранав Шукла, Шарат Кумар. Elasticsearch, Kibana, Logstash и поисковые системы нового поколения. – СПб: Питер, 2019. - 363 с.
2. Гончаров А. Самоучитель HTML. – СПб: Питер, 2002. – 242 с.
3. М. Мокс. Python for web application, 2016. – 263 с.

5. Робота із системою Kibana

Мета практичної роботи, що відповідає розділу, – навчитися встановлювати і запускати систему візуалізації Kibana та отримати практичні навички із інструментами Console при роботі з REST API для будь-яких можливих операцій Elasticsearch та інструментами Discover і Visualizations для інтерактивного аналізу даних.

Kibana – інструмент візуалізації для, що входить до Elastic Stack, який, зокрема, допомагає наочно представити дані з системи Elasticsearch. Цей інструмент часто називають вікном в Elastic Stack. В рамках системи Kibana пропонується декілька варіантів візуалізацій, таких як гістограма, карта, лінійні графіки, часові ряди і ін.

Система не тільки дозволяє візуалізувати дані, а й містить інструменти для розробників, такі як Console (Консоль).

У системі Kibana також передбачені інструменти для розробки. Можливо керувати налаштуваннями X-Pack для забезпечення безпеки в Elastic Stack, а за допомогою інструментів розробника створювати та тестувати запити REST API.

Встановлення системи Kibana

Для завантаження системи Kibana на комп'ютер (сервер) достатньо зайти на сторінку завантаження в Інтернеті за адресою <https://www.elastic.co/downloads/kibana>, вибрати необхідну операційну систему і активізувати відповідну кнопку (Рис. 5.1).

Система Kibana розповсюджується у вигляді ZIP-файлу, який після розгортання містить сукупність файлів, представлену на Рис. 5.2.

Для запуску системи Kibana достатньо із директорії bin запустити kibana або kibana.bin в залежності від операційної системи.

Після цього в браузері на локальному комп'ютері достатньо ввести адресу <http://localhost:5601>, щоб в штатному режимі отримати відповідь, наведеному на Рис. 5.3.

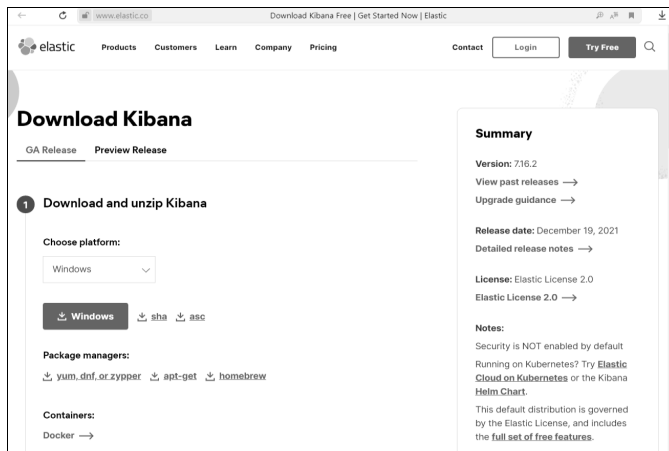


Рис. 5.1 – Сторінка завантаження в Інтернеті:
<https://www.elastic.co/downloads/kibana>

Ім'я	Размер	Сжат	Тип	Изменен	CRC32
bin			Папка с файлами	15.02.2021 13:11	
config			Папка с файлами	15.02.2021 13:10	
data			Папка с файлами	15.02.2021 13:10	
node			Папка с файлами	15.02.2021 13:10	
node_modules			Папка с файлами	15.02.2021 13:10	
plugins			Папка с файлами	15.02.2021 13:10	
src			Папка с файлами	15.02.2021 13:10	
x-pack			Папка с файлами	15.02.2021 13:10	
.j8nrc.json	3 106	830	Файл "JSON"	15.02.2021 13:10	12ED3E24
LICENSE.txt	3 860	1 557	Файл "TXT"	15.02.2021 13:10	6ACDD267
NOTICE.txt	1 489 914	147 238	Файл "TXT"	15.02.2021 13:10	41099EDA
package.json	740	390	Файл "JSON"	15.02.2021 13:10	AADFEA8B
README.txt	3 968	1 483	Файл "TXT"	15.02.2021 13:10	FCF99B46

Рис. 5.2 – Вміст архіву системи Kibana

Робота із інструментом Console

При переході за гіперпосиланням отримуємо панель вибору режимів роботи (<http://localhost:5601/app/home#/3>). З цієї панелі спочатку вибираємо роботу із інструментом Console (http://localhost:5601/app/dev_tools#/console).

Інструмент Console стане в нагоді при роботі з REST API для будь-яких можливих операцій Elasticsearch. Тобто система Kibana потрібна не тільки для візуалізацій, але й у якості інтерфейсу користувача для взаємодії із системою Elasticsearch.

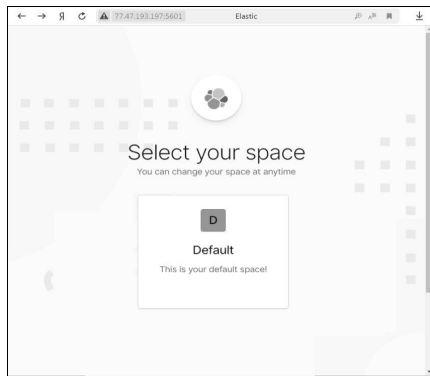


Рис. 5.3 – Перша веб-сторінка доступу до системи Kibana

Kibana Console представляє собою зручний редактор, який підтримує функцію автозаповнення та форматування запитів під час їх написання.

REST означає Representational State Transfer – “передача репрезентативного стану”. Це архітектурний стиль для взаємодії систем між собою. REST розвивався разом з протоколом HTTP, і майже всі системи, засновані на REST, використовують HTTP як свій протокол. HTTP підтримує різні методи: GET, POST, PUT, DELETE, HEAD і ін. Наприклад, GET призначений для отримання або пошуку чогось, POST використовується для створення нового ресурсу, PUT може застосовуватися для створення або відновлення існуючого ресурсу, а DELETE – для безповоротного видалення.

Після запуску Kibana слід вибрати посилання Dev Tools (Інструменти для розробників) на навігаційній панелі зліва. Консоль розділена на дві частини: поле редактора та поле результатів. Можна вводити команди REST

API, і після клацання на зеленому трикутнику запит буде надіслано до Elasticsearch (або кластер).

Після вибору інструменту Console здійснюється відображення відповідного інтерфейсу Console, який представлено на Рис. 5.4

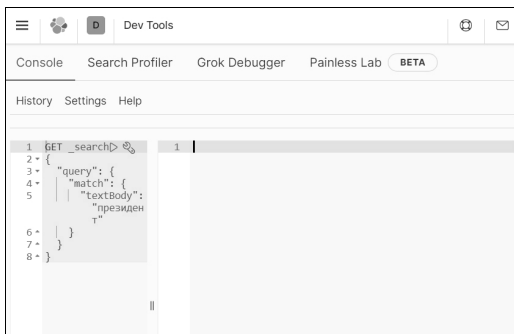


Рис. 5.4 – Інтерфейс інструменту Console

На прикладі, наведеному на Рис. 5.5, відправлений запит GET /. Це аналог команди curl, тільки набагато коротший. У цьому випадку не потрібно набирати http, хост і порт вузла Elasticsearch, а саме http://localhost:9200.

Як тільки ми починаємо набирати текст у редакторі консолі, ми отримуємо список можливих команд.

На наступному прикладі показано, як отримати інформацію щодо кількості записів в індексі hb (команда GET /hb/_count).

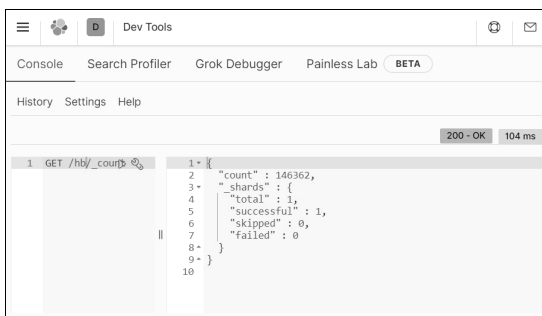


Рис. 5.5 – Інформація щодо кількості записів в індексі

На Рис. 5.6 наведено приклад опрацювання запитів вже відомої (див. розділ 4) структури:

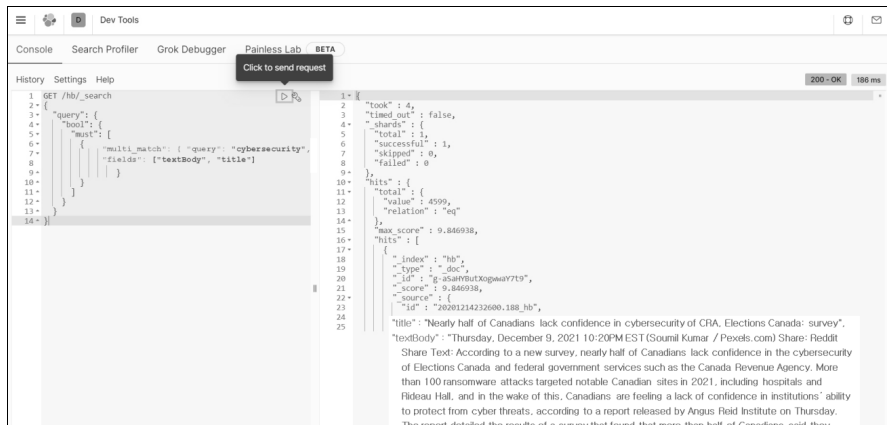


Рис. 5.6 – Результат опрацювання запиту за словом у двох полях

Результати опрацювання запиту на пошук у декількох полях з фільтрацією по джерелу наведено на Рис. 5.7.

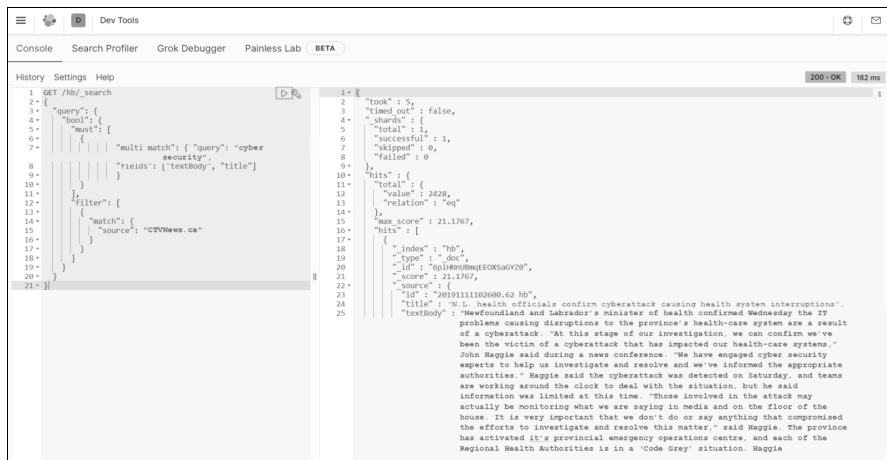


Рис. 5.7 – Результат опрацювання запиту із фільтрацією за полем “source”

Робота із інструментом *Discover*

Інструмент Kibana Discover (Дослідження) призначено для інтерактивного аналізу даних. Користувач за допомогою цього інструменту може виконувати запити пошуку, фільтрації, переглядати дані документів. Можна також зберігати запити на пошук або критерії фільтрації для повторного використання або створення візуалізацій на базі відфільтрованих результатів.

На Рис. 5.8. наведено результати пошуку за словом Huawei в індексі hb за проміжком часу у 200 діб.

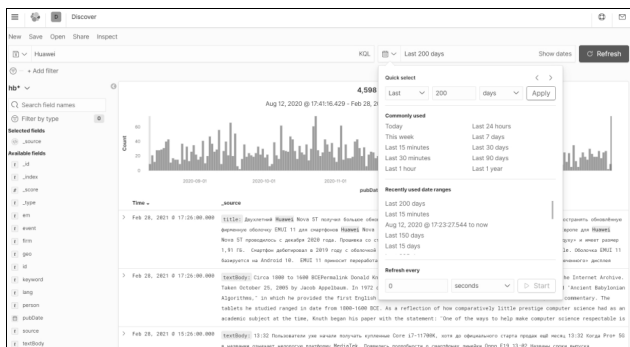


Рис. 5.8 – Результати аналізу результатів пошуку

Інструмент Discover дозволяє виводити для дослідження вибрані поля, додавати додаткову фільтрацію, за змістом окремих полів, тощо. На Рис. 5.9 наведено результати пошуку за словом Huawei в індексі hb (дати, джерело і заголовки) за діапазон часу у 200 діб, відфільтрований за словом Huawei. Як можна бачити, змінилась і діаграма кількості документів за добу.



Рис. 5.9 – Результати аналізу результатів пошуку з фільтрацією

Робота із інструментом *Visualizations*

Інструмент Kibana Visualizations призначений для створення візуальних образів на основі даних (візуалізацій). Існують можливості сформувати різні варіанти оформлення: гістограми, лінійні діаграми, карти, хмари тегів тощо. Користувач може підібрати необхідні візуалізації для полегшення аналізу даних.

Для створення нової візуалізації необхідно виконати кроки:

1. Перейти на сторінку Visualize і натисніть кнопку Create a new Visualization (Створити нову візуалізацію) або кнопку “+”.
2. Вибрати тип візуалізації.
3. Вибрати джерело даних.

На першому прикладі (Рис. 5.10) створено візуалізацію типу Агеа, в якій як критерій пошуку використовувалось аббревіатура HDD, як умова фільтрації вибиралось приналежність документів до джерела *НавгаНавг*, вісь абсцис відповідала часовому періоду у 200 діб. Як джерело інформації розглядався індекс *hb*. Інтерфейс візуалізації виглядає наступним чином.



Рис. 5.10 – Візуалізація типу Ageo для результатів пошуку

На другому прикладі створено візуалізацію типу Goal, в якій як критерій пошуку використовувалось слово «безпе́ность», як параметр, що досліджується, використовувались назви джерел (source.keywords). Як джерело інформації розглядався індекс hb. Інтерфейс візуалізації виглядає наступним чином (Рис. 5.11).

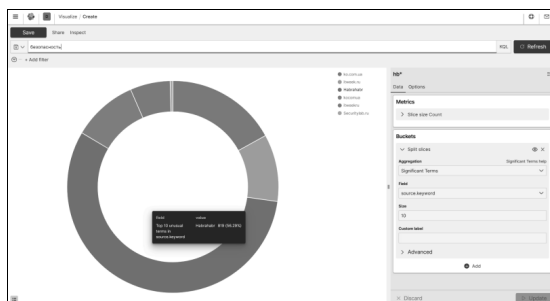


Рис. 5.11 – Візуалізація типу Goal для результатів пошуку

У цьому розділі ми ознайомились із системою Kibana як інструментом візуалізації та вивчення даних, який застосовується для таких завдань, як аналіз журналів та часових рядів, моніторинг програмних застосунків та поточних процесів. Він пропонує потужні та прості у використанні можливості: гістограми, лінійні графіки, кругові діаграми, теплові карти та вбудовану геопросторову підтримку. Крім того, Kibana забезпечує тісну

взаємодію з Elasticsearch, де вона за замовчуванням виступає як інструмент візуалізації даних, що зберігаються в Elasticsearch.

Питання до практичної роботи

1. Що таке система Kibana та яка її роль в обробці даних системою Elasticsearch?
2. Що таке інструмент Console та її застосування при роботі з Elasticsearch?
3. Що таке інструмент Discover і його можливості при візуалізації результатів роботи Elasticsearch?
4. Що таке інструмент Visualizations і його можливості при візуалізації результатів роботи Elasticsearch?

Завдання на самостійну роботу

1. Забезпечити функціонування на комп'ютері системи Kibana.
2. Детально ознайомитись із можливостями візуалізації в системі Kibana.
3. Побудувати часові діаграми кількості публікацій із різних RSS-джерел.

Література

1. Пранав Шукла, Шарат Кумар. Elasticsearch, Kibana, Logstash и поисковые системы нового поколения. – СПб: Питер, 2019. – 363 с.
2. Learning Kibana 5.0. Exploit the visualization capabilities of Kibana and build powerful interactive dashboards / Bahaaldine Azarmi. – Packt Publishing, 2017. – 275 p.

6. Агрегація в Elasticsearch

Мета практичної роботи, що відповідає розділу, – навчитися програмувати та правильно реалізувати узагальнені (агреговані) за деякою ознакою дані, отримані після аналітичної обробки результатів в Elasticsearch.

Аналітика в Elasticsearch призначена для отримання повної картини даних. При пошуку детально розглядається декілька записів, аналітика ж дозволяє поглянути на дані ширше і згрупувати їх різними способами.

Реалізація агрегації у запитах

Агрегація в Elasticsearch, яка визначається елементом aggregations, дозволяє отримувати узагальнені за деякою ознакою дані. Всі запити на агрегацію мають вигляд:

```
GET /<index_name>/<type_name>/_search
{
  "query": { ... тип запиту ... },
  "aggregations": {
    ... тип агрегації ...
  },
  "size": 0
}
```

Елемент aggregations повинен містити фактичний запит агрегації. Тіло запиту залежить від бажаного типу агрегації. Необов'язковий елемент query вказує контекст агрегації, тобто якщо необхідно обмежити контекст агрегації, необхідно вказати елемент query. Агрегація буде враховувати всі документи даного індексу і типу, якщо не вказано елемент query (ми можемо вважати його рівним запитом match_all, якщо немає іншого запиту). Наприклад, цей параметр вказується, якщо необхідно, щоб агрегація

працювала не за всіма даними, а тільки по певним документам, які відповідають конкретним умовам.

Запит фільтрує документи, які будуть оброблені елементом aggregations. Елемент size вказує, скільки відповідних пошуку документів повинно бути повернуто у відповіді. Значення за замовчуванням становить 10. Якщо значення size не вказано, відповідь буде містити не більше десяти релевантних документів. Зазвичай, якщо необхідно отримати тільки перші тестові результати агрегації, необхідно присвоювати елементу size значення 0, щоб не отримувати інших результатів.

Динаміка публікацій за запитом

Для отримання динаміки кількості публікацій за запитом в Elasticsearch, необхідно здійснити агрегацію даних, які відповідають тематиці, визначеній первинним запитом за полем, що відповідає значенню дати і часу. Конкретно, щоби здійснити агрегацію за датами всіх документів із індексу hb, що відповідають запиту Samsung, за полем часу, вводимо запит у форматі JSON:

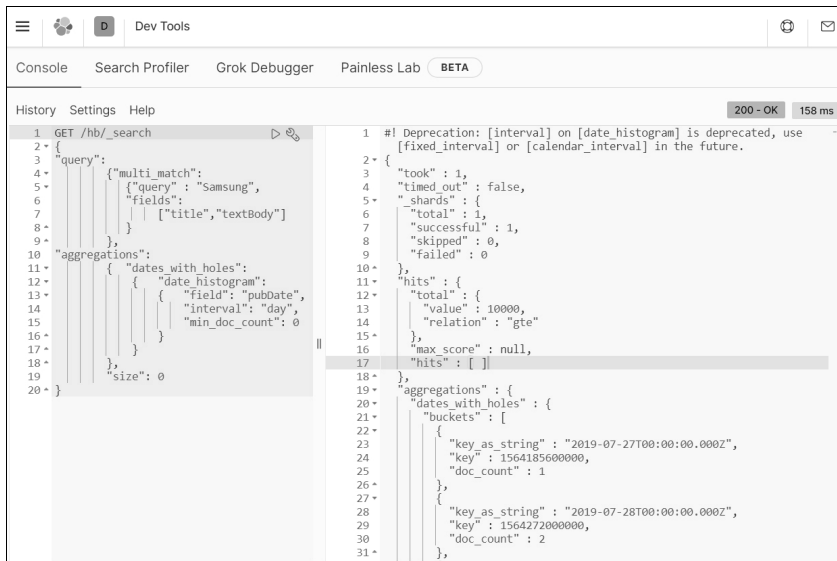
```
curl -XGET 'http://localhost:9200/hb/_search?pretty=true' -H
'Content-Type: application/json' -d '{
  "query": {
    "multi_match": {
      "query": "Samsung",
      "fields": ["title", "textBody"]
    }
  },
  "aggregations": {
    "dates_with_holes": {
      "date_histogram": {
        "field": "pubDate",
        "interval": "day",
        "min_doc_count": 0
      }
    }
  },
  "size": 0
}'
```

У результаті виконання цього запиту отримуємо вихідні дані для побудови діаграми – відповідь такого вигляду:

```
{
  "took" : 2,
  "timed_out" : false,
  "_shards" : {
    "total" : 1,
    "successful" : 1,
    "skipped" : 0,
    "failed" : 0
  },
  "hits" : {
    "total" : {
      "value" : 10000,
      "relation" : "gte"
    },
    "max_score" : null,
    "hits" : [ ]
  },
  "aggregations" : {
    "dates_with_holes" : {
      "buckets" : [
        {
          "key_as_string" : "2019-07-27T00:00:00.000Z",
          "key" : 1564185600000,
          "doc_count" : 1
        },
        {
          "key_as_string" : "2019-07-28T00:00:00.000Z",
          "key" : 1564272000000,
          "doc_count" : 2
        },
        . . .
        {
          "key_as_string" : "2021-03-04T00:00:00.000Z",
          "key" : 1614816000000,
          "doc_count" : 35
        },
        {
          "key_as_string" : "2021-03-05T00:00:00.000Z",
          "key" : 1614902400000,
          "doc_count" : 40
        }
      ]
    }
  }
}
```

Ці дані можна отримати із інструмента Console системи Kibana (Рис. 6.1).

Для подальшої обробки достатньо застосовувати дані, що відповідають полям `key_as_string` і `doc_count`.



The screenshot shows the Kibana Dev Tools console. On the left, a REST API call is shown: `GET /_search` with a query for 'Samsung' and an aggregation on 'pubDate' using a date histogram. The response on the right shows the search results, including a 'hits' section with two documents and an 'aggregations' section with a 'dates_with_holes' bucket containing two sub-buckets for 'key_as_string' and 'doc_count'.

```
1 GET /_search
2 {
3   "query": {
4     "multi_match": {
5       "query": "Samsung",
6       "fields": ["title", "textBody"]
7     }
8   },
9   "aggregations": {
10    "dates_with_holes": {
11      "date_histogram": {
12        "field": "pubDate",
13        "interval": "day",
14        "min_doc_count": 0
15      }
16    }
17  },
18  "size": 0
19 }
20 }
```

```
1 #! Deprecation: [interval] on [date_histogram] is deprecated, use
2   [fixed_interval] or [calendar_interval] in the future.
3 {
4   "took": 1,
5   "timed_out": false,
6   "shards": {
7     "total": 1,
8     "successful": 1,
9     "skipped": 0,
10    "failed": 0
11  },
12  "hits": {
13    "total": {
14      "value": 10000,
15      "relation": "gte"
16    },
17    "max_score": null,
18    "hits": [ ]
19  },
20  "aggregations": {
21    "dates_with_holes": {
22      "buckets": [
23        {
24          "key_as_string": "2019-07-27T00:00:00.000Z",
25          "key": "1564185600000",
26          "doc_count": 1
27        },
28        {
29          "key_as_string": "2019-07-28T00:00:00.000Z",
30          "key": "1564272000000",
31          "doc_count": 2
32        }
33      ]
34    }
35  }
36 }
```

Рис. 6.1 – Результати агрегації в консолі системи Kibana

Переведення даних в формат CSV

Для застосування отриманих результатів у середовищі спеціалізованих систем обробки цифрових даних, можна перетворити їх до формату CSV за допомогою програми мовою Python, код якої наведено нижче:

```
#!/usr/bin/python2.7
import sys
import re

t = sys.stdin.read()
json = t.split('\n')
t=""
for i in range(len(json)):
    t=t+" "+json[i]

t=re.sub('^\\s+', '', t)
days = re.findall('"key_as_string" : "(.+?)T', t)
count = re.findall('"doc_count" : (\\d+)', t)
```

```
for i in range(len(days)):
    print days[i]+"-"+count[i]
```

У результаті виконання програми можна отримати дані у форматі CSV такого вигляду:

```
2019-07-27;1
2019-07-28;2
2019-07-29;0
2019-07-30;1
2019-07-31;2
2019-08-01;0
2019-08-02;0
. . .
2021-03-02;12
2021-03-03;32
2021-03-04;35
2021-03-05;40
```

Обробка даних у середовищі Excel

Для подальшої обробки даних в форматі завантажимо отриманий CSV-файл у середовище Excel і побудуємо графік динаміки повідомлень (Рис. 6.2).

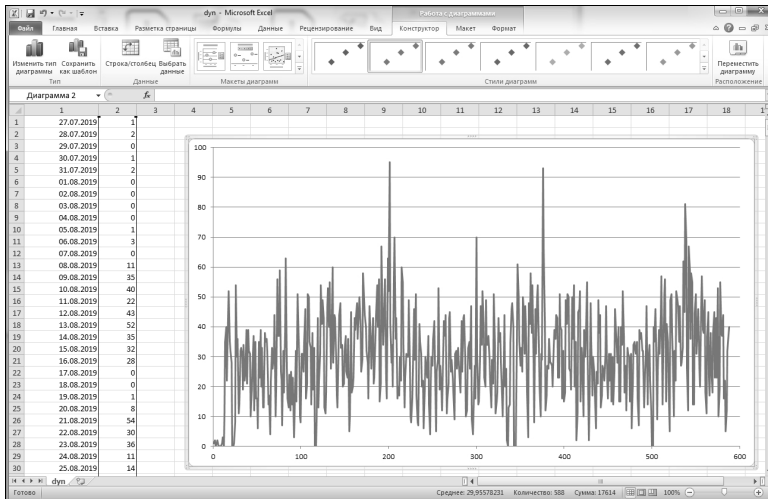


Рис. 6.2 – Завантажені в систему Excel дані та побудований графік

Після завантаження CSV-файлу в систему цифрової обробки даних виникають прості можливості його статистичного оброблення. На Рис. 6.3. наведено приклад знаходження поліноміального тренду цього ряду.

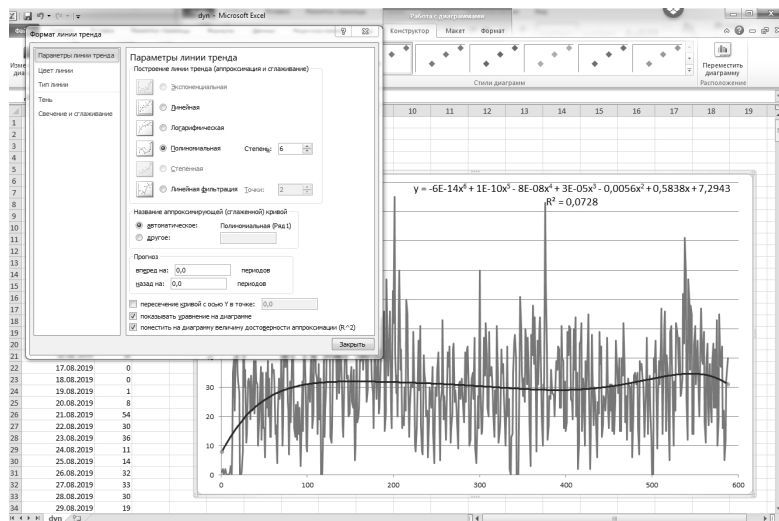


Рис. 6.3 – Поліноміальний тренд часового ряду, що досліджується

Питання до практичної роботи

1. Що таке агрегація даних?
2. Які існують типи агрегації в Elasticsearch?
3. Що таке CSV-формат файлу?
4. Як побудувати графік завантажених даних і визначити тренд часового ряду

Завдання на самостійну роботу

1. Завантажити в базу даних системи Elasticsearch тестову базу даних із накопиченими даними з RSS-фідів з мережі Інтернет.
2. Самостійно здійснити агрегацію цих даних за полем дата-час.

3. Завантажити отримані результати у доступну систему типу DSP (Digital Signal Processing).

Література

1. Пранав Шукла, Шарат Кумар. Elasticsearch, Kibana, Logstash и поисковые системы нового поколения. – СПб: Питер, 2019. – 363 с.
2. Кузьмичов А.І. Економетричне моделювання та прогнозування в Excel. Навч. пос. / Бишовець Н.Г., Кузьмичов А.І., Медведєв М.Г., Омецинська Н.В. – К.: ВПЦ АМУ, 2010. – 324 с.

7. Статистична обробка агрегованих даних

Мета практичної роботи, що відповідає розділу, – набуття практичних навичок при обробці текстових документів отриманих із мережі Інтернет, здатностей застосовувати загальні методи і засоби підготовки, статистичної обробки, візуалізації та аналізу агрегованих за деякою ознакою даних, використовуючи основні бібліотеки мови Python для роботи з «великими даними».

Середовище розробки

Як середовище програмування задач статистичної обробки даних передбачається застосування засобів WinPython (<https://winpython.github.io/>). Для цього встановлюємо програмне забезпечення, яке, зокрема, розміщено на веб-сайті [sourceforge.net](https://sourceforge.net/projects/winpython/files/) (https://sourceforge.net/projects/winpython/files/WinPython_3.8/3.8.8.0/). На Рис. 7.1 наведено фрагмент веб-сторінки WinPython.

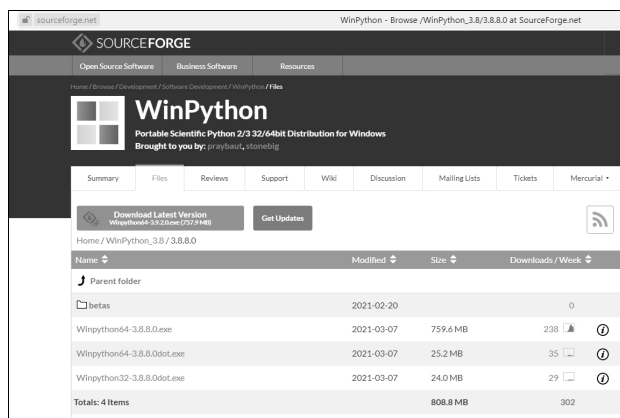


Рис. 7.1 – Фрагмент веб-сторінки системи програмування WinPython

Після розгортання дистрибутиву WinPython, можна активізувати веб-середовище розробки Jupyter Lab, яке стає доступним за адресою

http://localhost:8888/lab (Рис. 7.2). Саме у цьому програмному середовищі розроблюється програмне забезпечення на основі Python 3.8, яке відображує, серед іншого, графічні образи.

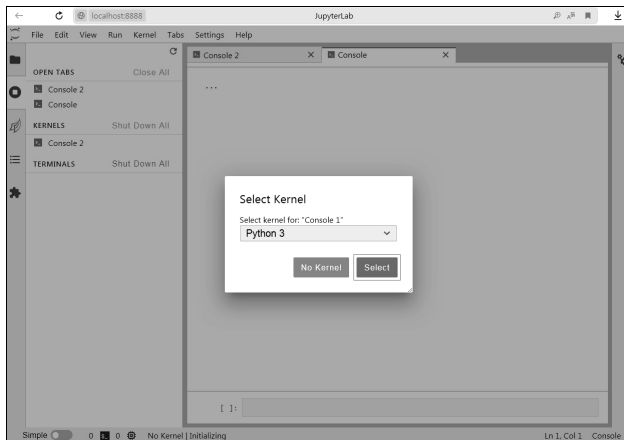


Рис. 7.2 – Завантаження середовища розробки Jupyter Lab

Віконне згладжування

В попередньому розділі було показано, як у результаті агрегації отримується часовий ряд, поведінку якого можна визначити шляхом статистичної обробки. У деяких випадках корисно розглядати більш гладку версію вихідного часового ряду $D = \{d_t\}$, $t = 1, \dots, N$. Згладжування допомагає виявити суттєві тенденції в динаміці ряду, приховавши при цьому шум і різні особливості, які проявляються при невеликих масштабах. Існують різноманітні методи згладжування. Найбільш простий спосіб згладжування - це обчислення віконного ковзного середнього. Просте ковзне середнє дорівнює середньому арифметичному значенню елементів ряду з інтервалу заданої довжини, а саме:

$$S = \{s_t\}, \quad s_t = \frac{1}{w} \sum_{k=-\lfloor w/2 \rfloor}^{\lfloor w/2 \rfloor} d_{t+k}$$

де w – ширина інтервалу згладжування (кількість елементів, за якими розраховується середнє), s_t – значення віконного ковзного середнього в точці $(- \lfloor w/2 \rfloor \leq t \leq \lfloor w/2 \rfloor)$.

При використанні віконного згладжування, чим більше ширина інтервалу згладжування, тим більш гладкою вийде результуюча функція. На Рис. 7.3 показано як виглядає згладжений ряд D при збільшенні значення w до 8.

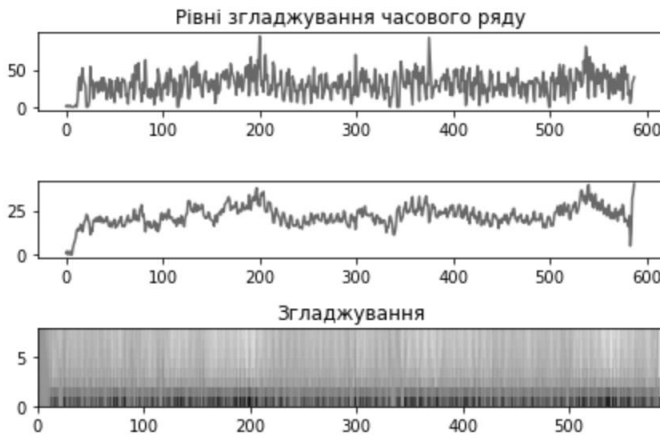


Рис. 7.3 – Результати віконного згладжування часового ряду

Результати згладжування ряду можна продемонструвати на графіку, у якого вісь абсцис відповідає часовій осі, а вздовж осі ординат відкладена ширина згладжує інтервал. На графіку показані значення - тобто елементи згладженого ряду в точці при використанні інтервалу ширини w .

Нижче наведено код програми мовою Python 3.8, яка візуалізує діаграму змін значень часового ряду при різних вікнах згладжування (вісь ординат).

```
import matplotlib.pyplot as plt
import numpy as np
from matplotlib.colors import LogNorm

D=[1, 2, 0, 1,2, 0, 0, 0, 0, 1, 3,0, 11, 35,40,22,43,52,35,32,
. . .
38,45,23,43,23,53,10,19,55,38,37,44,16,22,5,12,32,35,40]
```

```

M=8
N=len(D)
Z=np.random.rand(M,N)
C=D
for i in range(N):
    for j in range(M):
        Z[j][i]=D[i]
for j in range(M):
    for k in range(j,N-j):
        Z[j][k]=0;
        for l in range(1,j):
            Z[j][k]=Z[j][k]+D[k-l]
            Z[j][k]=Z[j][k]+D[k+l]

        Z[j][k]=Z[j][k]-D[k]
        Z[j][k]=Z[j][k]/(2*j+1)
fig, (ax0,ax1,ax2)=plt.subplots(3,1)
ax0.plot(D)
ax0.set_title('Півні згладжування часового ряду')
fig.tight_layout()

for i in range(N):
    C[i]=Z[7][i]

ax1.plot(C)
fig.tight_layout()
ax2.pcolor(Z,cmap='plasma')
ax2.set_title('Згладжування')
fig.tight_layout()
plt.show()

```

Експоненціальне згладжування

Інший часто використовуваний метод згладжування рядів - це експоненційне згладжування (Рис. 7.3). Попередні значення ряду враховуються з ваговими значеннями, що зменшуються експоненціально. Будемо позначати елементи згладженого ряду s_t , і відразу визначимо $s_0 = d_0$. Наступні елементи ряду отримують по рекурсивної формулою:

$$s_t = \alpha d_t + (1 - \alpha)s_{t-1}$$

де $0 \leq \alpha \leq 1$ – коефіцієнт згладжування. Очевидно, що при $\alpha = 1$ одержуваний ряд $S = \{s_t\}$, $t = 1, \dots, N$ збігається з вихідним. Таким чином, якщо значення α близьке до 1, то найбільшу вагу при визначенні s_t присвоюється відповідному d_t , а передісторія ряду має менше значення. З іншого боку,

якби α дорівнювало 0, то весь ряд згладився б до одного значення $s_t = d_0$. Тобто при α близькому до 0 передісторія ряду враховується з більш великою вагою, ніж поточне значення.

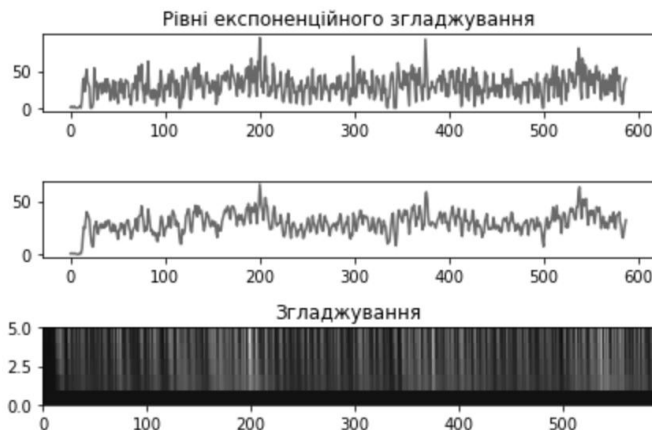


Рис. 7.3 – Результати експоненціального згладжування часового ряду

Нижче наведено код програми мовою Python 3.8, яка візуалізує діаграму змін значень часового ряду при різних коефіцієнтах експоненціального згладжування (вісь ординат).

```
M=5
N=len(D)
Z=np.random.rand(M,N)
C=D
for i in range(N):
    for j in range(M):
        Z[j][i]=D[i]

for j in range(M):
    alf=j/M
    for k in range(1,N):
        Z[j][k]=D[k]*alf+Z[j][k-1]*(1-alf);

fig, (ax0,ax1,ax2)=plt.subplots(3,1)
ax0.plot(D)
ax0.set_title('Рівні експоненційного згладжування')
fig.tight_layout()
```

```

for i in range(N):
    C[i]=Z[2][i]
ax1.plot(C)
fig.tight_layout()
ax2.pcolor(Z, cmap='plasma')
ax2.set_title('Згладжування')
fig.tight_layout()
plt.show()

```

Побудова вейвлет-скейлограм

До кола сучасних інструментальних засобів оцінки рядів спостережень відноситься також вейвлет-аналіз (Wavelet Analysis). Він особливо ефективний у тих випадках, коли крім загальних спектральних характеристик потрібно виявляти локальні в часі особливості поведінки процесу, що досліджується. Основою вейвлет-аналізу є вейвлет-перетворення, яке є особливим типом лінійного перетворення, базисні функції якого (вейвлети) мають специфічні властивості. Аналіз даних з використанням вейвлет-перетворень є зручним, надійним і потужним інструментом дослідження часових рядів і дозволяє представити результати у наочному вигляді, зручному для інтерпретації.

Вейвлетом (малою хвилею) називається деяка функція, зосереджена в невеликій околиці деякої точки та різко убутна до нуля по мірі видалення від неї як у часовий, так і в частотній області. Існують найрізноманітніші вейвлети, що мають різні властивості. Разом з тим, усі вейвлети мають вигляд коротких хвильових пакетів з нульовим інтегральним значенням, локалізованих на часовій осі, які є інваріантними до зсуву і до масштабування.

До будь-якому вейвлету можна застосувати дві операції:

- зрушення, тобто переміщення області його локалізації в часі;
- масштабування (розтягання або стиск).

Головна ідея вейвлет-перетворення полягає в тому, що нестационарний часовий ряд розподіляється на окремі проміжки (так звані «вікна

спостереження»), і на кожному з них виконується обчислення скалярного добутку (величини, що показує ступінь близькості двох закономірностей) досліджуваних даних із різними зрушеннями деякого вейвлету на різних масштабах.

На підставі базового вейвлету будують сімейство функцій за допомогою розтягування/стиснення та паралельного перенесення. Це необхідно, щоб досліджувати різні області вихідного сигналу та з різним ступенем детальності.

За допомогою безперервного вейвлет-перетворення виявляються ділянки досліджуваного ряду, які формою найбільш схожі на вейвлет. Ідея полягає в тому, щоб порівняти частини ряду з деяким шаблоном на різних масштабах.

Вейвлет перетворення – це кореляція між вихідним часовим рядом та деяким базовим вейвлетом. Вейвлет-перетворення генерує набір коефіцієнтів, за допомогою яких представляється початковий ряд. Вони є функціями двох змінних: часу і частоти, і тому утворюють поверхню у трьохвимірному просторі. Таким чином, вейвлет перетворення залежить від положення вейвлету на часовій осі та його масштабу. Процеси, що розглядаються, чітко проглядаються як на вейвлет-скейлограмах, так і на відповідних їм скелетонах (графіках ліній екстремумів). Ці коефіцієнти показують наскільки поведінка процесу в даній точці аналогічна вейвлету в даному масштабі. Чим ближче від аналізованої залежності в межах даної точки до виду вейвлету, тим більшу абсолютну величину має відповідний коефіцієнт. Застосування цих операцій, з урахуванням властивості локальності вейвлету в частотно-часовій області, дозволяє аналізувати дані на різних масштабах і точно визначати положення їхніх характерних особливостей в часі.

На скейлограмах можна побачити всі характерні особливості вихідного ряду: масштаб і інтенсивність періодичних змін, напрямки і значення трендів, наявність, розташування і тривалість локальних особливостей.

Найбільш поширені дійсні базисні вейвлет-функції конструюються на основі похідних функції Гауса ($g_0(t) = \exp(-t^2 / 2)$) (Рис. 7.4). Це обумовлено тією обставиною, що функція Гауса має найкращі показники локалізації як у часовій, так і у частотній областях. При $n = 1$ отримуємо вейвлет першого порядку, так званий WAVE-вейвлет з рівним нулю нульовим моментом. При $n = 2$ отримуємо МНАТ-вейвлет, «мексиканський капелюх» (Mexican Hat), при $n = 3$ – вейвлет Морле (Morle).

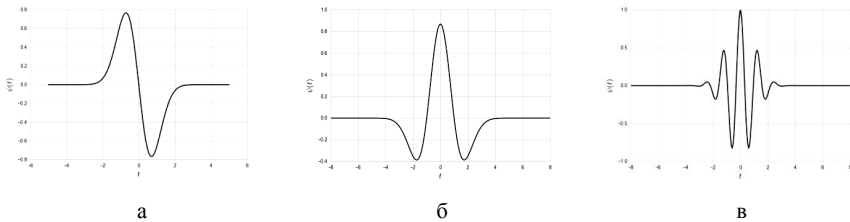


Рис. 7.4 – Приклади вейвлетів: (а) гауссова хвиля (перша похідна гауссової функції), (б) мексиканська капелюх, (в) вейвлет Морле (дійсна частина)

Отримані вейвлет-коефіцієнти можна представити в графічному вигляді. Якщо по одній осі відкласти зрушення вейвлету (вісь часу), а за іншою – масштаби (вісь масштабів), і розфарбувати точки отриманої схеми залежно від величини відповідних коефіцієнтів (чим більше коефіцієнт, тим яскравіше кольори).

Вейвлет-спектр $W_f(a, b)$ (wavelet spectrum, або time-scale-spectrum – масштабно-часовий спектр) є функцією двох аргументів: перший аргумент a (часової масштаб) аналогічний періоду осциляцій, тобто обернений частоті, а другий b – аналогічний зміщенню сигналу по осі часу.

Слід зазначити, що $W_f(a_0, b)$ характеризує часову залежність (при $a = a_0$), тоді як залежності $W_f(a, b_0)$ можна поставити у відповідність частотну залежність (при $b = b_0$).

Якщо досліджуваний сигнал $f(t)$ являє собою одиночний імпульс тривалістю τ_u , зосереджений в околі $t = t_0$, то його вейвлет-спектр матиме найбільше значення в околі точки з координатами $a = \tau_u$, $b = t_0$.

Отримані коефіцієнти представляються у графічному вигляді картою коефіцієнтів перетворення, або скейлограмою (Рис. 7.5). На скейлограмі по одній осі відкладаються зрушення вейвлету (вісь часу), а по іншій – масштаби (вісь масштабів), після чого точки схеми, що отримується, офарбовуються залежно від величини відповідних коефіцієнтів (чим більше коефіцієнт, тим яскравіше кольори зображення). На скейлограмі видні всі характерні риси вихідного ряду: масштаб та інтенсивність періодичних змін, напрямок і величина трендів, наявність, розташування та тривалість локальних особливостей.

```
import matplotlib.pyplot as plt

import pywt

f_s = 100                # Sampling rate

x=[1, 2, 0, 1,2, 0, 0, 0, 0, 1, 3,0, 11, 35,40,22,43,52,35,32,
  . . .
  45,23,43,23,53,10,19,55,38,37,44,16,22,5,12,32,35,40]

N=len(x)
t= range(N)

##### Visualization

fig, (ax1, ax4) = plt.subplots(2,1, sharex = True, figsize = (10,8))

# Signal
ax1.plot(t, x)
ax1.grid(True)
ax1.set_ylabel("Кількість публікацій")
ax1.set_title("Динаміка публікацій")
```



```
# Wavelet transform, i.e. scaleogram

cwtmatr, freqs = pywt.cwt(x, range(1, N), "mexh", sampling_period = 1
/ f_s)
ax4.pcolormesh(t, freqs, cwtmatr, vmin=-100, cmap = "inferno" )
ax4.set_ylim(0,10)
ax4.set_ylabel("Масштаб")
ax4.set_xlabel("Час")
ax4.set_title("Скейлограма на бази вейвлету MexH")

# plt.savefig("./fourplot.pdf")

plt.show()
```

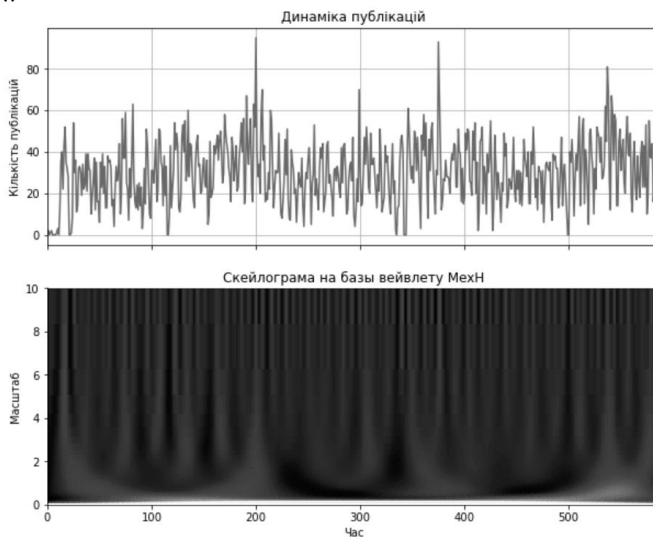


Рис. 7.5 – Приклад вейвлет-скейлогами вихідного часового ряду

Питання до практичної роботи

1. У чому полягає суть віконного згладжування у статистичній обробці даних?
2. Що таке експоненціальне згладжування у статистичній обробці даних?
3. Які переваги має вейвлет-аналіз (вейвлет-перетворення) при дослідженні часових рядів і інтерпретації результатів аналізу «великих даних»?
4. Назвати найбільш поширені дійсні базисні вейвлет-функції?

Завдання на самостійну роботу

1. Встановити на комп'ютері систему Python з підтримкою графічної бібліотеки matplotlib.
2. Детально ознайомитись із можливостями бібліотек matplotlib і pyplot.
3. Побудувати вейвлет-скейлограму з іншою базовою вейвлет-функцією. Дослідити різницю вейвлет-скейлограм при різних базових вейвлет-функціях.

Література

1. Sandro Tosi. Matplotlib for Python Developers. – Packt Publishing, 2009. – 308 p.
2. Fabio Nelli. Python Data Analytics: Data Analysis and Science Using Pandas, matplotlib, and the Python Programming Language. – Apress, 2015. – 350 c.
3. Поликар Р. Введение в вейвлет-преобразование. – СПб: АВТЭКС, 2001. – 59 с.
4. Распознавание информационных операций. А.Г. Додонов, Д.В. Ландэ, В.В. Цыганок и др. – Киев: Инжиниринг, 2017. – 282 с.
5. Основи теорії і практики інтелектуального аналізу даних у сфері кібербезпеки: навчальний посібник / Д.В. Ланде, І.Ю. Субач, Ю.Є. Бояринова. –Київ: ІСЗІ КПІ ім. Ігоря Сікорського, 2018. – 300 с.

8. Екстрагування ключових слів з відібраних за запитом документів

Мета практичної роботи, що відповідає розділу, – набуття практичних навичок для аналітичної обробки текстових документів, отриманих за допомогою пошукової системи Elasticsearch. Результатом обробки масиву документів є словник, сортований за частотою появи слів, після обробки якого отримуємо найбільш вагомі слова, що відносяться до заданої тематики.

Отримання масиву документів

Для подальшої обробки текстових документів, зокрема екстрагування найбільш вагомих, ключових слів, необхідно отримати тестові документи із бази даних Elasticsearch. Отримання таких документів, релевантних деякому тематичному запиту (наприклад, ключове слово Samsung), здійснюється за допомогою запиту до системи Elasticsearch, наприклад, до інформації, що зберігається за індексом hb, який можна ввести через систему Kibana (Рис. 8.1):

```
{
  "query": {
    "multi_match": {
      "query": "Samsung",
      "fields": ["title", "textBody"]
    }
  },
  "size": 1000
}
```

Відібрана інформація надається у правому фреймі інтерфейсу Kibana (Console), після чого зберігається у файлі samsung.txt у форматі JSON (Рис. 8.1).

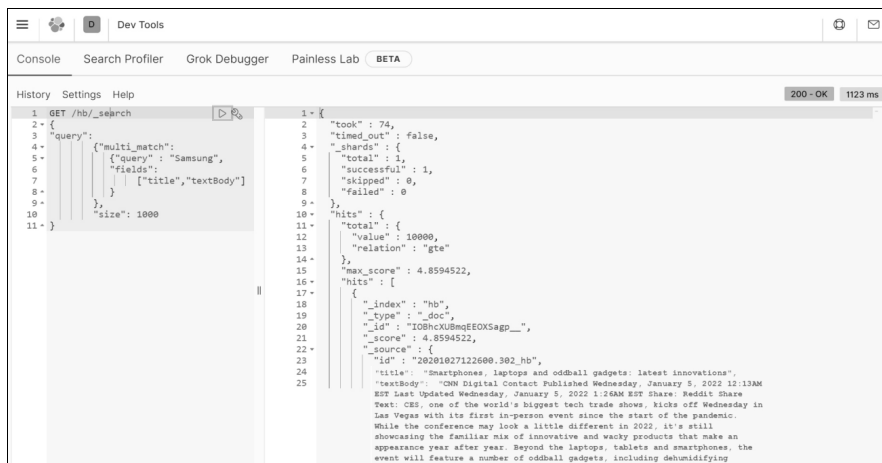


Рис. 8.1 – Відібрані документи у форматі JSON

Формування словника

Для формування словника слів із отриманого масиву документів D , що розміщуються в файлі `samsung.json`, у середовищі Python об'єднаємо вміст всіх полів "title" і "textBody", замінимо всі розподільні символи пропусками, і за допомогою регулярних виразів визначимо всі слова. Після сортування словника слів визначимо лише унікальні слова і абсолютну частоту їх появ у масиві D . Для кожного слова t , це значення $tf(t)$. Фрагмент вихідного коду мовою Python наведено нижче:

```
import re
import string
f = open("G:/samsung.json", "r")
t = f.read()
f.close()

json=t.split('\n')
t=""
for i in range(len(json)):
    t=t+" "+json[i]
#print(t)
title = re.findall('"title" : "(.+)"source"', t)
```

```

t=""
for i in range(len(title)):
    t=t+" "+title[i]
t=re.sub('"textBody" :',' ',t)
t=re.sub('["--"\[\]\./?0-9",.()$+»«-:;_...'],' ',t)
t=t.upper()
t=re.sub('\s\w\s',' ',t)
t=re.sub('\s\w\w\s',' ',t)
t=re.sub('\s\s',' ',t)
word =t.split(' ')
word.sort()

# Dictionary building
d={}
old=""
n=0;
for i in range(len(word)):
    if (word[i] == old):
        n=n+1;
    else:
        #print(old,n)
        d[old]=n
        old=word[i]
        n=1
d[old]=n
#print(d)

sorted_dict = {}
sorted_keys = sorted(d, key=d.get, reverse=True) # [1, 3, 2]

for w in sorted_keys:
    sorted_dict[w] = d[w]

print(sorted_dict)

```

У результаті роботи програми отримуємо сортований за частотою список слів (Рис. 8.2). Як можна бачити, до найбільш частотних слів відносяться також слова, що не несуть змістовного навантаження, так звані «стоп-слова». Списки стоп-слів для різних мов, що складені вченими-лінгвістами можна знайти в Інтернеті, наприклад, за адресою: <https://code.google.com/archive/p/stop-words/>.

Вибір найбільш вагомих слів

Для визначення ваги w слова t , будемо застосовувати частотний метод, а саме із сформованого на базі аналізу масиву словника відберемо найбільш частотні слова:

$$w(t) = \sum_{\{d: t \in d\}} tf(t, d),$$

де $f(t, d)$ – це частота появи слова t в документі d .

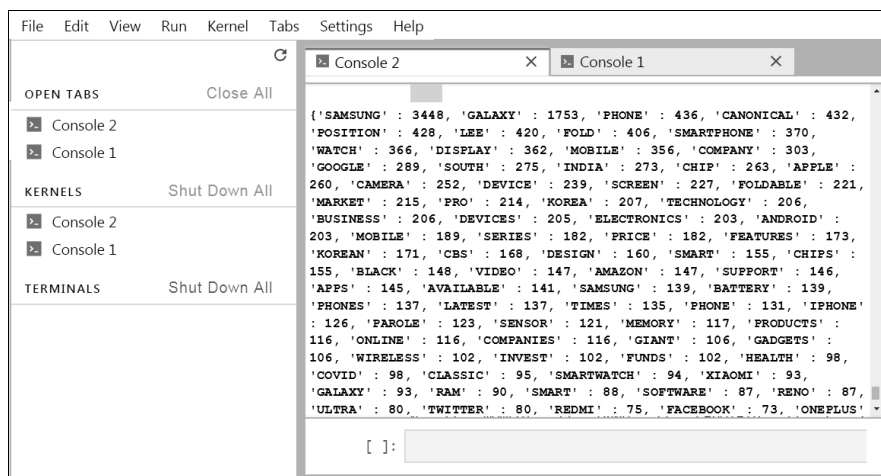


Рис. 8.2 – Відібрані слова і частоти їх появи

Як можна бачити, до найбільш частотних слів відносяться також слова, що не несуть змістовного навантаження, так званні «стоп-слова». Списки стоп-слів для різних мов, що складені компанією Google можна знайти в Інтернеті, наприклад, за адресою <https://code.google.com/archive/p/stop-words/>

Для подальшого застосування вибирається M найвагоміших слів, наприклад, $M = 50$, таких що не входять до стоп-словника. Фрагмент програмного коду мовою Python наведено нижче:

```
f = open("G:/stop.txt", "r")
t = f.read()
f.close()
```

```

t=t.upper()
stop =t.split('\n')
M=50
j=1
sorted_dict = {}
sorted_keys = sorted(d, key=d.get, reverse=True) # [1, 3, 2]

for w in sorted_keys:
    sorted_dict[w] = d[w]
    pr=0
    for i in range(len(stop)):
        if (stop[i] == w):
            pr=1
    if (pr == 0):
        print(j,w,sorted_dict[w])
        j=j+1
    if (j > M):
        break

```

Нижче наведено перелік 20 найвагоміших слів із документів бази даних системи Elasticsearch за тематикою, визначеною запитом Samsung:

```

3448 SAMSUNG
1753 GALAXY
436 PHONE
432 CANONICAL
428 POSITION
420 LEE
406 FOLD
370 SMARTPHONE
366 WATCH
362 DISPLAY
356 MOBILE
303 COMPANY
289 GOOGLE
273 INDIA
263 CHIP
260 APPLE
252 CAMERA
239 DEVICE
227 SCREEN
215 MARKET

```

Отримані результати записуються на диск (файл G:/words.txt).

Питання до практичної роботи

1. Як відбувається вибір найбільш вагомих слів за заданою тематикою?
2. Яка подальша мета використання отриманих слів по заданій тематиці?

3. У чому полягає суть іншого методу розрахунку ваги слів, зокрема методу TF-IDF?
4. У чому полягає суть іншого методу розрахунку ваги слів, зокрема дисперсійного методу?

Завдання на самостійну роботу

1. Детально ознайомитись із засобами обробки тестових рядків в мові Python.
2. Мовою Python самостійно розробити програмний модуль підрахунку ваги слів.
3. Ознайомитись з іншими методами розрахунку ваги слів, зокрема методом TF-IDF, дисперсійним методом і методом горизонтальної видимості (HVG).

Література

1. Ландэ Д.В., Снарский А.А., Безсуднов И.В. Интернетика: Навигация в сложных сетях: модели и алгоритмы – М.: Либроком (Editorial URSS), 2009. – 264 с.
2. Основи теорії і практики інтелектуального аналізу даних у сфері кібербезпеки: навчальний посібник / Д.В. Ланде, І.Ю. Субач, Ю.Є. Бояринова. – Київ: ІСЗЗІ КПІ ім. Ігоря Сікорського, 2018. – 300 с.

9. Формування мережі концептів. Візуалізація в Gephi

Мета практичної роботи, що відповідає розділу, – отримати практичні навички у формуванні мережі ключових слів, що відповідають найважливішим поняттям із текстового масиву, концептам, що отримуються при аналітичній обробці текстових документів та засвоїти принципи роботи системи Gephi, призначеної для відображення графових структур, в яких як вузли будуть розглядатися концепти, а як ребра – зв'язки між ними.

Формування матриці суміжності концептів

Відомо, що графова структура або мережа може визначатися матрицею суміжності. Для формування мережі концептів, вузлами якої будуть концепти, а як ребра – зв'язки між ними, аналізуються документи і масиви ключових слів. При формуванні матриці суміжності концептів застосовується такий підхід: два концепти вважаються зв'язаними, якщо вони входять до того ж самого речення. Речення визначається як частина тексту, що розділена відповідними знаками розподілу. При цьому сила зв'язку концептів відповідає кількості речень, де одночасно з'являються відповідні їм слова. За замовченням вважається, що вага зв'язку концепту самого із собою дорівнює нулю.

Матриця суміжності концептів $A = \{a_{i,j}\}$, де $a_{i,j}$ – сила (вага) зв'язку між концептами i та j .

Для відображення мережі в системі аналізу і візуалізації графів Gephi необхідно завантажити до неї матриці суміжності, яка має бути наведена у форматі:

$Concept_1; Concept_2; Concept_3; \dots; Concept_M$
 $Concept_1; a_{1,1}; a_{1,2}; a_{1,3}; \dots; a_{1,M}$
 $Concept_2; a_{2,1}; a_{2,2}; a_{2,3}; \dots; a_{2,M}$
 $Concept_3; a_{3,1}; a_{3,2}; a_{3,3}; \dots; a_{3,M}$
 \dots
 $Concept_M; a_{M,1}; a_{M,2}; a_{M,3}; \dots; a_{M,M}$

Нижче наводимо фрагмент коду мовою програмування Python, за допомогою якого формується файл words.csv, використовуючи в якості слів, що відповідають концептам, слова, отримані за допомогою засобів, описаних у попередньому розділі (файл G:/words.txt):

```

import re
import string
import numpy as np

f = open("G:/samsung.json", "r")
t = f.read()
f.close()
json = t.split('\n')
t=""
for i in range(len(json)):
    t=t+" "+json[i]
title = re.findall('"title" : "(.+)"source"', t)
t=""
for i in range(len(title)):
    t=t+" "+title[i]
t=re.sub('"textBody" : ','.',t)
t=re.sub('[-%""[\]\0-9', '()$+><=:;_...') , ' ', t)
t=t.upper()
sent = t.split('.')
print(sent)
f = open("G:/word.txt", "r")
t = f.read()
f.close()
t=t.upper()
w = t.split('\n')
for i in range(len(w)):
    s = w[i].split(' ')
    w[i]=s[1]

mtr = np.eye(len(w))
for i in range(len(w)):
    mtr[i][i]=0
stroka=""
for i in range(len(w)):
    stroka=stroka+" "+w[i]

```

```

print(stroka)

for i in range(len(sent)):
    for j in range(len(w)):
        for k in range(len(w)):
            if(j!=k):
                if (re.search(w[j],sent[i])):
                    if (re.search(w[k],sent[i])):
                        mtr[k][j]=mtr[k][j]+1

for i in range(len(w)):
    stroka=w[i]+";"
    for j in range(len(w)):
        a=int(mtr[i][j])
        b=a.__str__()
        if (j<len(w)-1):
            stroka=stroka+b+";"
        else:
            stroka=stroka+b
    print(stroka)

```

Результат виконання програми – файл в указаному вище форматі (Рис. 9.1).

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1		SAMSUNG	GALAXY	NOTE	ULTRA	COMPANY	IXBT	SMARTPHI	FOLD	ELECTRON	LINE	FLIP	IMAGES	MODEL
2	SAMSUNG	0	2924	570	510	1008	570	1570	309	299	201	196	195	338
3	GALAXY	2924	0	684	571	379	327	1054	370	78	216	219	136	287
4	NOTE	570	684	0	183	65	68	249	61	22	59	27	31	67
5	ULTRA	510	571	183	0	47	56	148	21	8	20	17	37	66
6	COMPANY	1008	379	65	47	0	10	420	51	151	48	38	22	69
7	IXBT	570	327	68	56	10	0	115	36	8	4	23	9	10
8	SMARTPHI	1570	1054	249	148	420	115	0	163	58	133	93	73	132
9	FOLD	309	370	61	21	51	36	163	0	6	16	60	5	34
10	ELECTRON	299	78	22	8	151	8	58	6	0	9	2	8	5
11	LINE	201	216	59	20	48	4	133	16	9	0	9	8	38
12	FLIP	196	219	27	17	38	23	93	60	2	9	0	1	19
13	IMAGE	195	136	31	37	22	9	73	5	8	8	1	0	20
14	MODEL	338	287	67	66	69	10	132	34	5	38	19	20	0
15	SCREEN	430	277	52	55	97	27	216	49	9	20	33	50	54
16	DEVICE	335	175	45	12	115	8	128	23	24	17	6	9	21
17	BUDS	110	139	21	16	15	16	30	14	4	6	7	5	8
18	CAMERA	360	310	49	85	38	29	170	10	4	7	4	66	28
19	MARKET	160	44	10	6	80	10	88	13	5	6	5	8	11
20	ANDROID	137	86	21	7	18	20	87	1	4	4	4	1	6
21	PRO	158	128	55	39	19	19	54	6	3	4	5	11	23

Рис. 9.1 – Результат виконання програми – файл у форматі CSV

Основні відомості щодо системи Gephi

Gephi (<https://gephi.org/>) – це в даний час найпопулярніша програма візуалізації і аналізу мереж та графів («мережових графів»). Gephi забезпечує швидку компоновку, ефективну фільтрацію та інтерактивне дослідження

даних, а також є одним з кращих варіантів для візуалізації великомасштабних мереж.

Gephi – це мультиплатформне програмне забезпечення яке розповсюджується з відкритим кодом згідно з ліцензіями CDDL 1.0 і GNU General Public License v3. За адресою <https://gephi.org/> (Рис. 9.2) доступні версії для Mac OS X, Windows і Linux вихідних кодів. Для роботи програми потрібна Java версія 7 і вище. У теперішній час програма локалізується для таких мов: англійська, французька, португальська, російська, китайська, чеська та німецька.

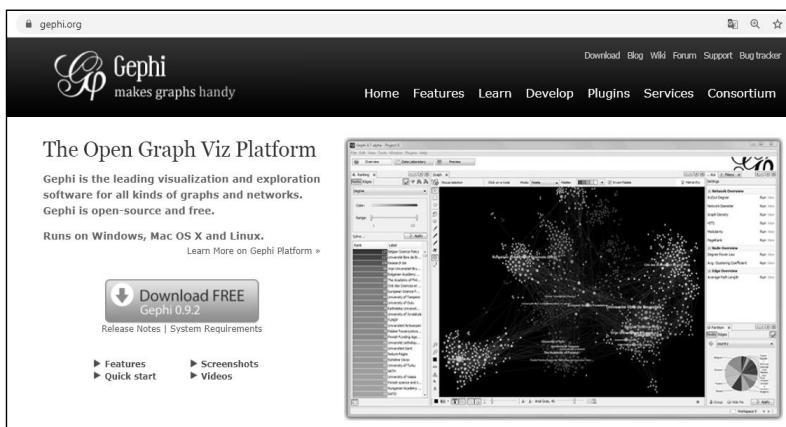


Рис. 9.2 – Стартове вікно *Gephi* – кнопка завантаження останньої версії

Розробники *Gephi* описують цю програму як "*Photoshop*, але для даних".

Gephi дозволяє завантажувати дані мереж в форматах *GEXF*, *GDF*, *GML*, *GraphML*, *Pajek* (*NET*), *GraphViz* (*DOT*), *CSV*, *UCINET* (*DL*), *Tulip* (*TPL*), *Netdraw* (*VNA*) і таблиць *Excel*. Крім того, *Gephi* дозволяє експортувати дані мереж в форматах *JSON*, *CSV*, *Pajek* (*NET*), *GUESS* (*GDF*), *Gephi* (*GEFX*), *GML* та *GraphML*. Завдяки цьому *Gephi* може взаємодіяти з іншими системами аналізу і візуалізації графів.

Програма включає в себе множину різних алгоритмів компонування (укладання графіків на площині) і дозволяє налаштувати кольори, розміри і мітки в графах. *Gephi* є інтерактивним програмним забезпеченням і надає засоби для виявлення спільнот, а також надається можливість розрахунку найкоротших шляхів або відносної відстані від будь-якого вузла до цього вузла. Плагіни від *Gephi* дозволяють розширювати її функціональність і додавати нові алгоритми, макети та інструменти вимірювань. *Gephi* має багатопотокову схему обробки даних, і таким чином, дозволяє виконувати кілька видів аналізу одночасно.

Інтерфейс користувача системи *Gephi* включає три основні розділи (вікна):

- «Лабораторія даних»: тут зберігаються всі вихідні дані про мережі, а також додаткові розрахункові значення;
- «Обробка даних»: тут відбувається велика частина операцій користувача, зокрема, ручне редагування мереж, тестування макетів, встановлення фільтрів;
- «Попередній перегляд»: тут уточнюється форма виводу графу, як правило, за допомогою набору інструментів граф допрацьовується, в тому числі, і з естетичної точки зору. У цьому ж вікні реалізований виклик експорту графа в формати *PDF*, *PNG* і *SVG*.

Ці три основні розділи охоплюють множину вкладок, які дозволяють користувачу реалізовувати окремі функції.

Встановлення системи Gephi

Для встановлення системи візуалізації і аналізу графів *Gephi* необхідно перейти на сайт системи <http://gephi.org> і завантажити відповідне програмне забезпечення (кнопка Download FREE), після чого здійснити стандартні кроки з його інсталяції.

Візуалізація мережі у середовищі Gephi

При завантаженні матриці суміжності концептів в систему Gephi (з формату CSV), відображається інформація щодо кількості вузлів (акторів) і зв'язків між ними (Рис. 9.3).

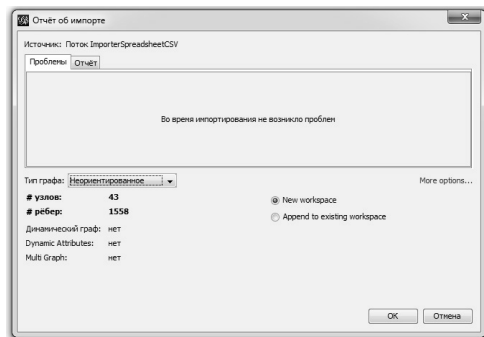


Рис. 9.3 – інформація щодо імпорту даних

Після цього відкривається файл, який в режимі «Обработка» відображається як мережа в неупорядкованому вигляді. Первинне відображення мережі наведено на Рис. 9.4.

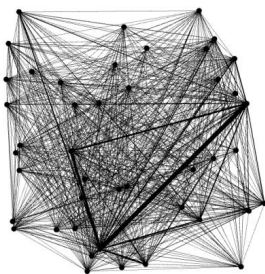


Рис. 9.4 – Первинне відображення мережі

Далі мережа обробляється шляхом укладання вузлів за алгоритмом OpenOrd, розмір вузлів – за ступенем, розфарбування – за кластерами по

класу модульності (Рис. 9.5). У режимі «Перегляд» мережа концептів має наступний вигляд, представлений на Рис. 9.6.

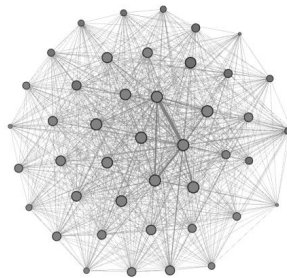


Рис. 9.5 – Візуалізація у режимі «Обробка»

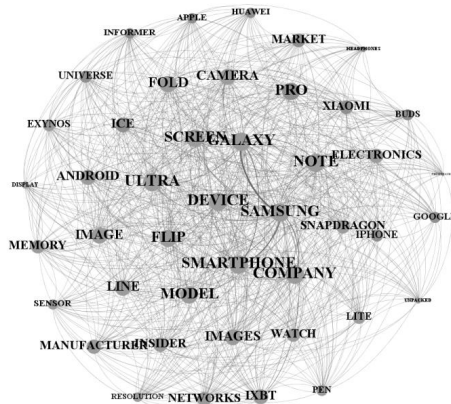


Рис. 9.6 – Візуалізація у режимі «Перегляд»

Питання до практичної роботи

1. Що таке матриця суміжності концептів?
2. Які ви знаєте засоби перетворення типів даних в мові Python?
3. У яких дослідницьких проектах найбільш широко використовується пакет Gephi?
4. Назвіть основні режими роботи системи Gephi.

Завдання на самостійну роботу

1. Детально ознайомитись із засобами перетворення типів даних в мові Python.
2. Самостійно встановити на комп'ютері систему Gephi.
3. Ознайомитись з основними режимами і можливостями системи Gephi.

Література

1. Ken Cherven. Mastering Gephi Network Visualization. – Packt Publishing, 2015. ISBN 78-1-78398-734-4.
2. John W. Foreman. Data Smart. Using Data Science to Transform Information into Insight. – Wiley, 2013. ISBN 111-8-66146-X, 978-1-11866-146-8.
3. Основи теорії і практики інтелектуального аналізу даних у сфері кібербезпеки: навчальний посібник / Д.В. Ланде, І.Ю. Субач, Ю.Є. Бояринова. – Київ: ІСЗЗІ КПП ім. Ігоря Сікорського, 2018. – 300 с.
4. Ландэ Д.В., Субач І.Ю. Візуалізація та аналіз мережевих структур : навчальний посібник. – Київ : КПП ім. Ігоря Сікорського, Вид-во "Політехніка", 2021. – 80 с. ISBN 978-966-2577-14-3

10. Apache Hadoop

Мета практичної роботи, що відповідає розділу, – вивчити принципи роботи Hadoop – потужного інструментарію для роботи з великими даними від Apache foundation, отриманих з Інтернет (оброблення логічних записів віддалених серверів). Розглянути приклади розробки MapReduce-програм (компанії Google, 2004 рік) під Hadoop.

Apache Hadoop – програмна платформа і основа для організації розподіленого зберігання і обробки наборів надвеликих даних з використанням моделі MapReduce, при якій завдання ділиться на дрібніші відособлені фрагменти, кожен з яких може бути запущений на окремому вузлі кластера, що складається з окремих серверів. Всі модулі в Hadoop спроектовані з врахуванням того, що апаратне забезпечення може виходити з ладу і це має автоматично опрацьовуватись фреймворком.

Ядро системи Apache Hadoop складається з розподіленої файлової системи Hadoop Distributed Filesystem (HDFS), та системи обчислень на основі моделі програмування MapReduce.

Кластер Hadoop складається з багатьох машин, які зберігають та паралельно обробляють великі набори даних. Клієнтські комп'ютери посилають до цього обчислювального хмара завдання та отримують результати.

Hadoop – це каркас із відкритим вихідним кодом, призначений для створення та запуску розподілених додатків, що обробляють великі обсяги даних. Розподілені обчислення – це широка та багатогранна область, але Hadoop має ряд важливих відмінних рис, а саме:

- Доступність – Hadoop працює на великих кластерах, зібраних зі стандартних комп'ютерів, або у обчислювальному хмарі.

- Надійність – оскільки Hadoop має працювати на стандартному обладнанні, його архітектура розроблена з урахуванням можливості частих відмов.
- Масштабованість – Hadoop масштабується лінійно, тобто зі збільшенням обсягу даних досить додати нові вузли кластеру.
- Простота – Hadoop дозволяє користувачеві швидко створювати ефективний паралельний код.

Доступність та простота Hadoop дають йому конкурентну перевагу у справі написання та запуску великих розподілених програм. З іншого боку, надійність і масштабованість роблять Hadoop відповідним засобом навіть для відповідальних завдань.

Основні компоненти Hadoop

Фреймворк Apache Hadoop складається з наступних модулів:

- Hadoop Distributed File System (HDFS) – розподілена файлова система, що дозволяє зберігати інформацію практично необмеженого обсягу.
- Hadoop YARN – фреймворк для керування ресурсами кластера і менеджменту завдань, в тому числі включає фреймворк MapReduce.
- Hadoop Common – пов'язуюче програмне забезпечення — набір інфраструктурних програмних бібліотек та утиліт, що використовуються для інших модулів та родинних проектів

Також існує велика кількість проектів безпосередньо пов'язаних з Hadoop, але не входять в Hadoop core:

- Hive – інструмент для SQL-like запитів над великими даними (перетворює SQL-запити в серію MapReduce-завдань);
- Pig – мова програмування для аналізу даних на високому рівні. Один рядок коду на цій мові може перетворитися в послідовність MapReduce-завдань;
- Hbase – колонкова база даних, яка реалізує парадигму BigTable;

- Cassandra – високопродуктивна розподілена key-value база даних;
- ZooKeeper – сервіс для розподіленого зберігання конфігурації і синхронізації змін цієї конфігурації;
- Mahout – бібліотека і рушій машинного навчання на великих даних.

Проект Apache Spark представляє собою засіб для розподіленої обробки даних. Apache Spark зазвичай використовує компоненти Hadoop, такі як HDFS і YARN для своєї роботи.

Apache Hadoop та екосистема Hadoop

Хоча Hadoop найчастіше асоціюється з MapReduce та розподіленою файловою системою (HDFS, що раніше називалася NDFS), цим терміном часто позначають ціле сімейство взаємопов'язаних проектів, об'єднаних інфраструктурою розподілених обчислень та великомасштабної обробки даних.

Усі базові проекти ведуться фондом Apache Software Foundation, що надає підтримку спільноті проектів з відкритим кодом – включаючи вихідний HTTP-сервер, від якого походить назва.

З розширенням екосистеми Hadoop з'являються нові проекти, що необов'язково знаходяться під керуванням Apache, але надають додаткові функції Hadoop або утворюють абстракції вищого рівня на основі базової функціональності.

Встановлення Hadoop на кластер за допомогою Cloudera Manager

Раніше установка Hadoop представляла собою досить важке заняття – потрібно було окремо конфігурувати кожну машину в кластері, стежити за тим, щоб нічого не було забуто, акуратно налаштовувати моніторинги. З ростом популярності Hadoop з'явилися компанії (такі як Cloudera, Hortonworks, MapR), які надають власні збірки Hadoop і потужні засоби для керування Hadoop-кластером. В рамках цієї роботи ми будемо користуватися збіркою Hadoop від компанії Cloudera.

Cloudera Hadoop складається з:

- Cloudera Hadoop (CDH) – власне дистрибутив Hadoop;
- Cloudera Manager – інструмент для розгортання, моніторингу та управління кластером Hadoop.

Для того щоб встановити Hadoop на свій кластер, потрібно виконати кілька простих кроків:

1. Завантажити Cloudera Manager Express на одну з машин свого кластера звідси – <https://www.cloudera.com/downloads/cdp-private-cloud-trial.html>;
2. Присвоїти права на виконання і запустити;
3. Слідувати інструкціям установки.

Кластер повинен працювати на одній з підтримуваних операційних систем сімейства Linux: RHEL, Oracle Enterprise Linux, SLES, Debian, Ubuntu.

Після установки надається консоль керування кластером, де можна переглядати встановлені сервіси, додавати/видаляти сервіси, стежити за станом кластера, редагувати конфігурацію кластера (Рис. 10.1).

Більш детально з процесом установки Hadoop на кластер за допомогою Cloudera Manager можна ознайомитися за посиланням в розділі Quick Start (https://docs.cloudera.com/documentation/enterprise/5-2-x/topics/cloudera_quickstart_vm.html).

У разі ознайомлення і виконання практичної роботи достатньо просто завантажити попередньо налаштовану віртуальну машину користуватися налаштованим програмним забезпеченням.

Розгортання розподіленого середовища на базі Apache Hadoop

Перед початком роботи необхідно ознайомитись з існуючими реалізаціями розподілених середовищ (Apache Hadoop). Навчитись розгортати та використовувати розподілені середовища для вирішення ресурсномістких задач.

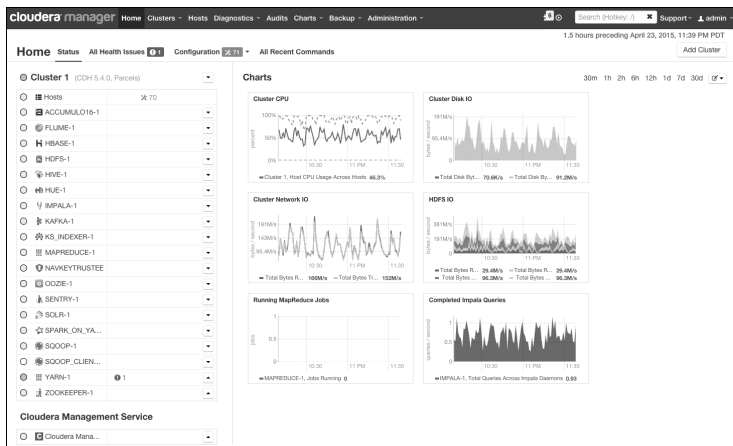


Рис. 10.1 – Консоль керування кластером

Apache Hadoop – проект, який колись був частиною відкритої пошукової системи Nutch, але пізніше відокремився від неї як реалізація розподіленої файлової системи (Distributed File System) і парадигми MapReduce. Hadoop складається з двох модулів. Це реалізація парадигми розподіленої файлової системи (яка називається HDFS) і MapReduce, тобто реалізація MapReduce-фреймворку.

Distributed File System (DFS). Основні властивості: зберігання великих обсягів даних; прозорість, робота як зі звичайною файловою системою – з можливістю відкривати файли і не думати про реалізацію розподіленого зберігання; масштабованість – можливість додавати нові ноди для зберігання даних; надійність – вихід з ладу деякого відсотка вузлів не повинно зашкодити цілісності всієї системи. DFS використовує кластер, що складається з керуючого master-вузла, і slaves-вузлів, на яких зберігаються дані. На master-вузлі зберігається таблиця файлів. Кожен файл розбитий на блоки. На master-вузлі зберігається інформація, на яких конкретно кластерних машинах знаходяться блоки файлів. При читанні/запису даних майстер надає інформацію про розташування блоків файлу: на якій машині

зберігаються конкретні блоки та їх порядковий номер. Для забезпечення надійності, кожен блок зберігається в декількох примірниках, на декількох машинах.

Порядок виконання роботи

Налаштування кластеру з одним вузлом.

1. Оновлення індексу репозиторіїв:

```
$ sudo apt-get update
```

2. Встановлення Java:

```
$ sudo apt-get install default-jdk $ update-alternatives --config java
```

Версія Java повинна бути не менше 1.7.

Перевірка:

```
$ java -version
```

3. Встановлення SSH:

```
$ sudo apt-get install ssh
```

4. Встановлення RSYNC:

```
$ sudo apt-get install rsync
```

5. Генерація ключів для SSH:

```
$ ssh-keygen -t dsa -P ' ' -f ~/.ssh/id_dsa $ cat ~/.ssh/id_dsa.pub >> ~/.ssh/authorized_keys
```

6. Завантажуємо Hadoop (TAR.GZ архів) з офіційного сайту.

7. Розпаковування Hadoop:

```
$ sudo tar -zxvf hadoop-.tar.gz
```

8. Додавання змінних середовища в ~/.bashrc

```
#Hadoop Variables
```

```
export JAVA_HOME=<шлях до каталогу JAVA>
```

```
export HADOOP_HOME=<шлях до розпакованого каталогу Hadoop>
```

```
export PATH=$PATH:$HADOOP_HOME/bin
```

```
export PATH=$PATH:$HADOOP_HOME/sbin
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export YARN_HOME=$HADOOP_HOME
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
export HADOOP_OPTS="-Djava.library.path=$HADOOP_HOME/lib"
```

9. Оновлення конфігурації після внесення змін:

```
$ source ~/.bashrc
```

10. В каталозі hadoop відредагувати hadoop-env.sh

```
export JAVA_HOME=<шлях до каталогу Java>
```

11. Редагування core-site.xml

```
<configuration>
<property>
<name>fs.defaultFS</name>
<value>hdfs://localhost:9000</value>
</property>
</configuration>
```

12. Редагування yarn-site.xml

```
<configuration>
<property>
<name>yarn.nodemanager.aux-services</name>
<value>mapreduce_shuffle</value>
</property>
<property>
<name>yarn.nodemanager.auxservices.mapreduce.shuffle.class</name>
<value>org.apache.hadoop.mapred.ShuffleHandler</value>
</property>
```

```
</configuration>
```

13. Копіювання шаблону:

```
$ sudo cp mapred.site.xml.template mapred-site.xml
```

14. Редагування шаблону (mapred-site.xml):

```
<configuration>
<property>
<name>mapreduce.framework.name</name>
<value>yarn</value>
</property>
</configuration>
```

15. Редагування hdfs-site.xml

```
<configuration>
<property>
<name>dfs.replication</name>
<value>1</value>
</property>
<property>
<name>dfs.namenode.name.dir</name>
<value>
file:<каталог Hadoop>/hadoop_data/hdfs/namenode
</value>
</property>
<property>
<name>dfs.datanode.data.dir</name>
<value>
file: <каталог Hadoop>/hadoop_store/hdfs/datanode
</value>
</property>
```


</configuration>

16. Створення файлів вузлу (нод):

```
$ mkdir -p <каталог Hadoop>/hadoop_data/hdfs/namenode
```

```
$ mkdir -p <каталог Hadoop>/hadoop_data/hdfs/datanode
```

17. Зміна поточного користувача та групи каталогу з Hadoop:

```
$ sudo chown <користувач>:<група> -R <каталог Hadoop>
```

18. Форматування та запуск

```
$ hdfs namenode -format
```

```
$ start-all.sh
```

19. Перевірка статусу:

```
$ jps
```

Завантаживши Hadoop, можна керувати кластером Hadoop в одному з трьох підтримуваних режимів:

Локальний/автономний режим — після завантаження Hadoop за замовчуванням він налаштовується в автономному режимі та може працювати як окремий процес Java.

Псевдорозподілений режим — це розподілене моделювання на одній машині. Кожен демон Hadoop, такий як hdfs, yarn, MapReduce і т. д., працюватиме як окремий процес Java. Цей режим є корисним для розробки.

Повністю розподілений режим — цей режим повністю розподілений з мінімумом двох або більше комп'ютерів як кластер.

Встановлення Hadoop в автономному режимі

При встановленні Hadoop 2.4.1 в автономному режимі демони не запускаються і все працює в одній віртуальній машині Java. Автономний режим підходить для запуску програм MapReduce під час розробки, оскільки їх легко тестувати та налагоджувати.

Для налаштування Hadoop можна встановити змінні середовища Hadoop, додавши наступні команди у файл ~/.bashrc:

```
export HADOOP_HOME=/usr/local/hadoop
```

Можна переконаватися, що Hadoop працює нормально. Просто введіть наступну команду:

```
$ hadoop version
```

Якщо з налаштуваннями все гаразд, ви повинні побачити результат (з точністю до версії):

```
Hadoop 2.4.1
```

```
Subversion https://svn.apache.org/repos/asf/hadoop/common -r 1529768
```

```
Compiled by hortonmu на 2013-10-07T06:28Z
```

```
Compiled with protoc 2.5.0
```

```
From source with checksum 79e53ce7994d1628b240f09af91e1af4
```

Це означає, що налаштування автономного режиму Hadoop працює нормально. За замовчуванням Hadoop налаштовано для роботи в нерозподіленому режимі на одній машині.

Питання до практичної роботи.

1. Що таке платформа для обробки масиву надвеликих даних Apache Hadoop?
2. Які основні компоненти (протоколи та специфікації) формують стек Apache Hadoop?
3. Чому надвеликі масиви даних можна обробляти розділеними на блоки алгоритмом обробки MapReduce?

Завдання для самостійної роботи

1. Встановити на комп'ютері платформу Apache Hadoop і вивчити склад її компонентів та значення її основних параметрів.
2. Перевірити працездатність Hadoop.
3. Ознайомитись з процедурою роботи механізму обробки великих даних MapReduce.

Литература

1. Andrew S. Tanenbaum, Maarten van Steen. Distributed Systems: Principles and Paradigms. – Upper Saddle River, NJ: Pearson Prentice Hall, 2nd edition, 2006. – 686 p.
2. Nancy A. Lynch. Distributed Algorithms. – San Francisco, Calif.: Morgan Kaufmann Publishers, 1996. – 872 p.
3. Ajay D. Kshemkalyani, Mukesh Singhal. Distributed Computing: Principles, Algorithms and Systems. – Cambridge University Press, 2008. – 736 p.
4. Mikito Takada. Distributed Systems for Fun and Profit (Электронный ресурс <http://book.mixu.net/distsys/ebook.html>).
5. Apache Hadoop (Электронный ресурс <http://hadoop.apache.org/>).
6. Torque Resource Manager (Электронный ресурс <http://www.adaptivecomputing.com/products/open-source/torque/>).

11. Програмування MapReduce

Мета практичної роботи, пов'язаної із розділом – навчитися програмувати прості MapReduce-застосунки для практичних задач та об'єднання простих задач у ланцюжки з використанням команд мови Python та просту структуру сховищ даних «ключ-значення», вивчити принципи обробки текстових документів (корпусів мови, csv серверних логів) з використанням інструмента Hadoop та команд мови Python для написання коротких програм обробки.

MapReduce – це концепція розподілу обчислень. Робота MapReduce складається з двох основних кроків: Map і Reduce. На першому кроці (Map) відбувається попередня обробка вхідних даних. Для цього один з комп'ютерів (master-вузол) отримує вхідні дані задачі, розділяє їх на частини і передає іншим комп'ютерам (worker-вузлам). На другому кроці (Reduce) відбувається агрегування попередньо оброблених даних на кроці Map. Головний вузол отримує відповіді від робочих вузлів і на їх основі формує результат – рішення задачі.

Програмування задачі WordCount

Розглянемо як запустити MapReduce-завдання мовою Python на Hadoop. Як завдання скористаємося класичним прикладом WordCount. Для того, щоб експериментувати на реальних даних треба використати підготовлений архів новин.

Формулювання завдання: є набір документів. Необхідно для кожного слова, що зустрічається в наборі документів, порахувати, скільки разів зустрічається це слово в архіві на сервері (lenta_articles.tar.gz).

Розглянемо рішення задачі. Якщо маємо великий корпус документів - нехай один документ буде одним вхідним записом для MapReduce-завдання. У MapReduce ми можемо тільки задавати функції користувача, що ми і

зробимо (використовуватимемо python-like псевдокод). Map розбиває документ на слова і повертає множину пар вигляду (word, 1).

```
def map(doc):  
    for word in doc:  
        yield word, 1
```

Reduce підсумовує входження кожного слова:

```
def reduce(word, values):  
    yield word, sum(values)
```

Тепер завдання запрограмувати це рішення у вигляді коду, який можна буде виконати на Hadoop і запустити.

Найпростіший спосіб запустити MapReduce-програму на Hadoop - скористатися streaming-інтерфейсом Hadoop. Streaming-інтерфейс передбачає, що Map і Reduce реалізовані у вигляді програм, які приймають дані з stdin і видають результат на stdout.

Програма, яка виконує функцію Map називається Mapper. Програма, яка виконує Reduce, називається, відповідно, Reducer.

Streaming інтерфейс передбачає за замовчуванням, що один рядок, що входить в Mapper або Reducer відповідає одному запису, що входить для Map.

Висновок Mapper'а потрапляє на вхід Reducer'у у вигляді пар (ключ, значення), при цьому всі пари, що відповідають одному ключу:

- Гарантовано будуть оброблені одним запуском Reducer'a;
- Будуть подані на вхід поспіль (тобто якщо один Reducer обробляє кілька різних ключів – вхід буде згруповано по ключу).

Отже, реалізуємо Mapper і Reducer на мові програмування Python:

```
#mapper.py  
import sys  
  
def do_map(doc):  
    for word in doc.split():
```

```

        yield word.lower(), 1

for line in sys.stdin:

    for key, value in do_map(line):

        print(key + "\t" + str(value))

#reducer.py

import sys

def do_reduce(word, values):

    return word, sum(values)

prev_key = None

values = []

for line in sys.stdin:

    key, value = line.split("\t")

    if key != prev_key and prev_key is not None:

        result_key, result_value = do_reduce(prev_key, values)

        print(result_key + "\t" + str(result_value))

        values = []

    prev_key = key

    values.append(int(value))

if prev_key is not None:

    result_key, result_value = do_reduce(prev_key, values)

    print(result_key + "\t" + str(result_value))

```

Дані, які буде обробляти Hadoop повинні зберігатися на HDFS. Завантажимо опис роботи і покладемо її в HDFS. Для цього потрібно скористатися командою `hadoop fs`:

```

wget https://www.dropbox.com/s/opp5psid1x3jt41/lenta_articles.tar.gz

tar xzvf lenta_articles.tar.gz

hadoop fs -put lenta_articles

```

Утиліта Hadoop fs підтримує велику кількість методів для маніпуляцій з файлової системою, багато з яких один в один повторюють стандартні утиліти linux. Детальніше з її можливостями можна ознайомитися за посиланням - <https://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-common/FileSystemShell.html>.

Тепер запустимо streaming-завдання:

```
yarn jar /usr/lib/hadoop-mapreduce/hadoop-streaming.jar\  
-input lenta_articles\  
-output lenta_wordcount\  
-file mapper.py\  
-file reducer.py\  
-mapper "python mapper.py"\  
-reducer "python reducer.py"
```

Утиліта yarn служить для запуску і керування різними застосунками (в тому числі map-reduce based) на кластері. Hadoop-streaming.jar - це якраз один з прикладів такого yarn-застосунку.

Далі йдуть параметри запуску:

- input – папка з вихідними даними на hdfs;
- output – папка на hdfs, куди потрібно покласти результат;
- file – файли, які потрібні в процесі роботи map-reduce завдання;
- mapper – консольна команда, яка буде використовуватися для map-стадії;
- reduce - консольна команда яка буде використовуватися для Reduce-стадії.

Після запуску в консолі можна буде побачити прогрес виконання завдання і URL для перегляду більш детальної інформації про завдання (Рис. 11.1).

В інтерфейсі доступному з цього URL можна отримати більш детальний статус виконання завдання, подивитися логи кожного Mapper'а і Reducer'а (Рис. 11.2).

```

cloudera@quickstart: ~/Downloads
File Edit View Search Terminal Help
[cloudera@quickstart Downloads]$ yarn jar /usr/lib/hadoop-mapreduce/hadoop-streaming.jar -input lenta_articles -output lenta_wordcount -file mapper.py -file reducer.py -mapper 'python mapper.py' -reducer 'python reducer.py'
15/10/03 01:28:11 WARN streaming.StreamJob: -file option is deprecated, please use generic option -files instead.
packageJobJar: [mapper.py, reducer.py] [/usr/jars/hadoop-streaming-2.6.0-cdh5.4.2.jar] /tmp/streamjob4927104235339879050.jar tmpDir=null
15/10/03 01:28:14 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
15/10/03 01:28:14 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
15/10/03 01:28:16 INFO mapred.FileInputFormat: Total input paths to process : 20
15/10/03 01:28:16 INFO mapreduce.JobSubmitter: number of splits:20
15/10/03 01:28:17 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1443859552117_0002
15/10/03 01:28:17 INFO impl.YarnClientImpl: Submitted application application_1443859552117_0002
15/10/03 01:28:17 INFO mapreduce.Job: The url to track the job: http://quickstart.cloudera:8080/proxy/application_1443859552117_0002/
15/10/03 01:28:17 INFO mapreduce.Job: Running job: job_1443859552117_0002

```

Рис. 11.1 – Протокол виконання завдання

Job Overview			
Job Name:	streamjob4927104235339879050.jar		
State:	RUNNING		
Uberized:	false		
Started:	Sat Oct 03 01:28:28 PDT 2015		
Elapsed:	1mins, 18sec		

ApplicationMaster			
Attempt Number	Start Time	Node	Logs
1	Sat Oct 03 01:28:20 PDT 2015	quickstart.cloudera:8042	logs

Task Type	Progress	Total	Pending	Running	Complete
Map	<div><div></div></div>	20	9	5	6
Reduce	<div><div></div></div>	1	0	1	0
Attempt Type	New	Running	Failed	Killed	Successful
Maps	9	5	0	0	6
Reduces	0	1	0	0	0

Рис. 11.2 – Статус завдання

Результат роботи після успішного виконання складається в HDFS в папку, яку ми вказали в полі output. Переглянути її вміст можна за допомогою команди «hadoop fs -ls lenta_wordcount».

Сам результат можна отримати наступним чином:

```

hadoop fs -text lenta_wordcount/* | sort -n -k2,2 | tail -n5
с      41
что    43
на     82
и      111
в      194

```

Команда «hadoop fs -text» видає вміст папки в текстовому вигляді. Ми відсортували результат за кількістю входжень слів. Як і очікувалося, найчастіші слова в мові - прийменники.

Map only job

Насправді існує ряд завдань, в яких можна обійтися тільки стадією Map. Ось приклади таких завдань:

1. Фільтрація даних (наприклад, «Знайти всі записи з IP-адреси 123.123.123.123» в логах web-сервера);
2. Перетворення даних («Видалити колонку в csv-логах»);
3. Завантаження і вивантаження даних з зовнішнього джерела («Вставити всі записи з логу в базу даних»).

Такі завдання вирішуються за допомогою методу Map-Only. При створенні Map-Only завдання в Hadoop потрібно вказати нульову кількість Reducer'ів:

У Hadoop для цього можна визначити функцію Combine (комбінування), тобто буде обробляти вихід частини Mapper'ів. Функція, що комбінує дуже схожа на функцію Reduce – вона приймає на вхід вихід частини Mapper'ів і видає агрегований результат для цих Mapper'ів, тому дуже часто Reducer використовують і як combiner. Важлива її відмінність від Reduce – на функцію Combine потрапляють в повному обсязі значення, відповідні одному ключу.

Більш того, Hadoop не гарантує того, що функція комбінування взагалі буде виконана для виходу Mapper'a. Тому що функція комбінування не завжди може бути застосована, наприклад, у разі пошуку медіанного значення по ключу. Проте, у тих завданнях, де функція комбінування може бути застосованою, її використання дозволяє домогтися істотного приросту швидкості виконання MapReduce-завдання.

Найскладніша стадія при виконанні Map-Reduce завдання - це стадія Shuffle. Відбувається це тому, що проміжні результати (виходи Mapper'a) записуються на диск, упорядковано і передаються по мережі. Однак існують завдання, в яких така поведінка видається не дуже розумною. Наприклад, в задачі підрахунку слів у документах можна попередньо агрегувати

результати виходів кількох Mapper'ів на одному вузлі map-reduce завдання, і передавати на Reducer вже підсумовані значення по кожній машині.

Ланцюжки MapReduce-задач

Існують ситуації, коли для вирішення задачі одним MapReduce не обійтися. Наприклад, розглянемо трохи видозмінене завдання WordCount: є набір текстових документів, необхідно порахувати, скільки слів зустрілося від 1 до 1000 разів в наборі, скільки слів від 1001 до 2000, скільки від 2001 до 3000 і так далі. Для вирішення цього завдання нам буде потрібно 2 завдання MapReduce:

1. Видозмінений WordCount, який для кожного слова розрахує, в який з інтервалів воно потрапило;
2. MapReduce, що підраховує, скільки разів у виході першого MapReduce зустрівся кожен з інтервалів.

Рішення на псевдокодi:

```
#map1
def map(doc):
    for word in doc:
        yield word, 1

#reduce1
def reduce(word, values):
    yield int(sum(values)/1000), 1

#map2
def map(doc):
    interval, cnt = doc.split()
    yield interval, cnt

#reduce2
def reduce(interval, values):
    yield interval*1000, sum(values)
```

Для того, щоб виконати послідовність MapReduce-завдань на hadoop, досить просто в якості вхідних даних для другого завдання вказати папку, яка була вказана в якості output для першої і запустити їх по черзі.

На практиці ланцюжки MapReduce-завдань можуть являти собою досить складні послідовності, в яких MapReduce-завдання можуть бути підключені як послідовно, так і паралельно один до одного (Рис. 11.3). Для спрощення керування такими планами виконання завдань існують окремі інструменти типу планувальник oozie (<http://oozie.apache.org/>) і luigi (це пакет Python, який допомагає створювати складні конвеєри пакетних завдань).

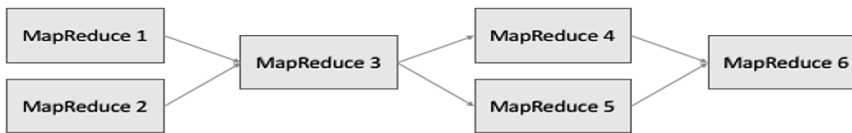


Рис. 11.3 – Приклад ланцюжка MapReduce-завдань.

Distributed cache

Важливим механізмом в Hadoop є Distributed Cache. Distributed Cache дозволяє додавати файли (наприклад, текстові файли, архіви, jar-файли) до оточення, в якому виконується MapReduce-завдання.

Можна додавати файли, що зберігаються на HDFS, локальні файли (локальні для тієї машини, з якої виконується запуск завдання). Distributed Cache можна використовувати разом з Hadoop streaming: додаючи через опцію `-file` файли `mapper.py` і `reducer.py`. Можна додавати не тільки `mapper.py` і `reducer.py`, а взагалі довільні файли, і потім користуватися ними як ніби вони знаходяться в локальній папці.

Reduce Join

При роботі з реляційними базами часто використовується операція Join, яка дозволяє спільно обробити зміст деяких таблиць, об'єднавши їх по

деякому ключу. При роботі з великими даними таке завдання теж іноді виникає. Розглянемо наступний приклад:

Розглянемо приклад, коли є логи двох веб-серверів, кожен з яких має наступний вигляд:

```
1446792139 178.78.82.1    /sphingosine/unhurrying.css
1446792139 126.31.163.222 /accentually.js
1446792139 154.164.149.83 /pyroacid/unkemptly.jpg
1446792139 202.27.13.181  /Chawia.js
1446792139 67.123.248.174  /morphographical/dismain.css
1446792139 226.74.123.135  /phanerite.php
1446792139 157.109.106.104 /bisonant.css
```

Необхідно порахувати для кожного IP-адреси на який з 2-х серверів він частіше заходив. Результат повинен бути представлений у вигляді:

```
178.78.82.1    first
126.31.163.222 second
154.164.149.83 second
226.74.123.135 first
```

На відміну від реляційних баз даних, у загальному випадку об'єднання двох логів по ключу (за IP-адресою) являє собою досить складну операцію і вирішується за допомогою 3-х MapReduce і патерну Reduce Join (Рис. 11.4).

ReduceJoin працює наступним чином:

На кожен з вхідних логів запускається окремий MapReduce (Map only), що перетворює вхідні дані до наступного вигляду:

```
key -> (type, value),
```

де key – ключ, за яким потрібно об'єднувати таблиці, type – тип таблиці (first або second в нашому випадку), а value – це будь-які додаткові дані, прив'язані до ключа.

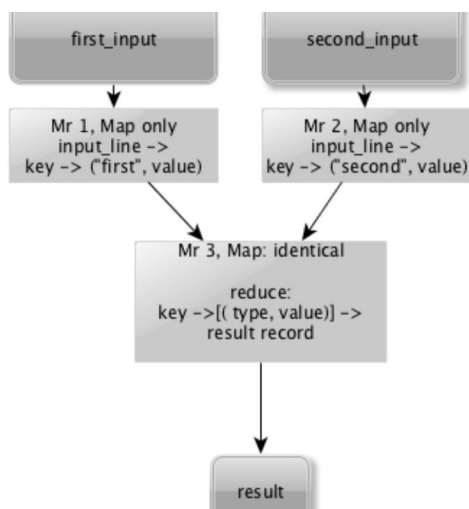


Рис. 11.4 – Загальна схема ReduceJoin

Виходи обох MapReduce подаються на вхід 3-му MapReduce, який, власне, і виконує об'єднання. Цей MapReduce містить порожній Mapper, який просто копіює вхідні дані. Далі Shuffle розкладає дані по ключах і подає на вхід Reducer'у у вигляді:

key -> [(type, value)]

Важливо, що в цей момент на Reducer потрапляють записи з обох логів і при цьому за полем type можна ідентифікувати, з якого з двох логів потрапило конкретне значення. Значить даних достатньо, щоб вирішити вихідну задачу. У нашому випадку Reducer просто повинен порахувати для кожного ключа записів, з яким type зустрілося більше і вивести цей type.

MapJoin

Патерн ReduceJoin описує загальний випадок об'єднання двох логів за ключем. Однак є окремий випадок, при якому завдання можна істотно спростити і прискорити. Це випадок, при якому один з логів має розмір істотно менший, ніж інший.

Розглянемо наступну задачу:

Існує 2 логи. Перший лог веб-серверу (такий же як в попередній задачі), другий файл (розміром в 100 Кб) містить відповідність URL → Тематика.

Приклад 2-го файлу:

```
/toyota.php auto
/football/spartak.html sport
/Cars auto
/Finances/money business
```

Для кожної IP-адреси необхідно розрахувати сторінки якої категорії з даної IP-адреси завантажувалися найчастіше.

У цьому випадку нам теж необхідно виконати Join 2-х логів за URL. Однак у цьому випадку нам не обов'язково запускати 3 MapReduce, оскільки другий лог повністю поміститься в пам'яті. Для того, щоб вирішити задачу за допомогою 1-го MapReduce, ми можемо завантажити другий лог в Distributed Cache, а при ініціалізації Mapper'а просто прочитати його в пам'ять, поклавши його в словник → topic (Рис. 11.5).

Далі завдання вирішується таким чином:

Map:

```
# Знаходимо тематику кожної зі сторінок першого логу
input_line -> [ip, topic]
```

Reduce:

```
ip -> [topics] -> [ip, most_popular_topic]
```

Reduce отримує на вхід ip і список всіх тематик, просто обчислює, яка з тематик зустрілася найчастіше. Таким чином задача вирішена за допомогою 1-го MapReduce, а власне Join взагалі відбувається всередині Map (тому якби не потрібна була додаткова агрегація за ключем – можна було б обійтися MapOnly завданням).

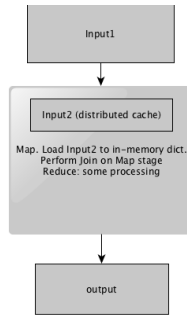


Рис. 11.5 – Схема роботи MapJoin

У цій роботі ми розглянули Hadoop – програмний стек для роботи з великими даними, описали процес установки Hadoop на прикладі дистрибутива Cloudera, показали, як писати mapreduce-програми, використовуючи streaming-інтерфейс і API Hadoop'a.

Питання до практичної роботи:

1. У чому полягає суть завдання Map в алгоритмі MapReduce?
2. У чому полягає суть завдання Reduce в алгоритмі MapReduce?
3. Скільки етапів виконує алгоритм MapReduce при обробці Великих Даних?
4. Яке основне завдання функції спліт у Hadoop при організації обробки Великих Даних для завдань MapReduce?
5. Які найрозповсюдженіші сфери застосування MapReduce для обробки «великих даних» ви знаєте?

Завдання на самостійну роботу

1. Використовуючи паттерни програмування в MapReduce, створити програму обробки текстових файлів.
2. Розробити програму обробки логів, використовуючи паттерни програмування в MapReduce.

3. Ознакомиться с принципами работы алгоритму обработки великих данных MapReduce.

4. Створити на власному комп'ютері середовище обробки великих даних Hadoop, використовуючи технологію віртуалізації.

Література

1. Томас Эрл, Ваджид Хаттак, Пол Булер Основы Big Bata: концепции, алгоритмы и технологии/ Пер. с англ. , К.: Издательство Бизнес, Баланс, Букс - BBB, 2016.-342с.
2. Натан Марц, Джеймс Уоррен Большие данные: принципы и практика построения масштабируемых систем обработки данных в реальном времени: Пер. с англ. – Киев. «Вильямс». 2016.-308с.
3. Том Уайт Hadoop. Подробное руководство. Пер. с англ. СПб «Питер». 2013.-672с.
4. Технология Hadoop. Практический опыт и экспертиза DISgroup, 2012. URL: <http://www.dis-group.ru>)
5. Чак Лэм. Hadoop в действии. – М.: ДМК Пресс, 2012. – 424 с. ISBN 978-5-94074-785-7
6. <https://habr.com/ru/company/dca/blog/268277/>
7. <https://habr.com/ru/post/270453/>
8. <https://habr.com/ru/post/345672/>

12. СУБД MongoDB

***Мета практичної роботи,** що відповідає розділу, – отримати досвід роботи з NoSQL СКБД MongoDB, встановлення сервера і рядкового клієнту цієї системи, роботи у режимі командного рядку, завантаження зовнішніх файлів, зокрема, у форматі CSV до баз даних цієї СКБД, а також експортування даних у зовнішні файли.*

NoSQL-технології приходять на заміну давно відомим реляційним базам даних. На відміну від реляційних баз даних NoSQL СКБД, пропонують документоорієнтовані моделі даних, завдяки чому багато з таких СКБД працюють швидше, мають кращу масштабованість, їх легше використовувати.

MongoDB, одна з NoSQL-СКБД, реалізує новий підхід до побудови баз даних, де немає таблиць, схем, запитів SQL, зовнішніх ключів та багатьох інших речей, які притаманні об'єктно-реляційним базам даних.

MongoDB – це не реляційна, а документоорієнтована система керування базами даних. Основною причиною відмови від реляційної моделі є спрощення горизонтального масштабування, але є інші переваги. MongoDB написана на C++, тому її легко портувати на різні платформи.

MongoDB можна розгорнути на платформах Windows, Linux, MacOS, Solaris.

Шість основних концепцій MongoDB:

1. MongoDB - концептуально те саме, що звичайна, звична нам база даних. Всередині MongoDB може бути нуль або більше бази даних, кожна з яких є контейнером для інших сутностей.

2. База даних може мати нуль або більше колекцій. Колекція настільки схожа на традиційну «таблицю», що можна сміливо вважати їх одним і тим самим.

3. Колекції складаються з нуля чи більше «документів». Знову ж таки, документ можна розглядати як «рядок».

4. Документ складається з одного або більше «полів», які подібні до «стовпця».

5. Індеси в MongoDB майже ідентичні до індесів в реляційних базах даних.

6. "Курсори" відрізняються від попередніх п'яти концепцій, але вони дуже важливі (хоча часом їх обходять увагою) і заслуговують на окреме обговорення.

Встановлення сервера і рядкового клієнту

Для інсталяції сервера MongoDB необхідно:

1. Зайти на офіційну веб-сторінку скачування (Рис. 12.1) і завантажити бінарний файл (рекомендовану стабільну версію). Можна використовувати як 32, так і 64-розрядну версію.

2. Розпакувати (куди завгодно) і перейдіть в папку bin. Два виконуваних файлу, з якими належить працювати: `mongod` – це сервер, а `mongo` – клієнтська консоль.

3. Створити новий файл в папці bin і назвати його `mongodb.config`

4. Додати в `mongodb.config` один рядок: `dbpath = <Шлях до файлів бази даних>`.

Наприклад, в Windows можна написати `dbpath=c:\mongodb\data` а в Linux – `dbpath=/etc/mongodb/data`.

5. Запустити `mongod` з параметром `--config /path/to/your/mongodb.config`.

Деякі команди рядкового клієнта

Для запуску рядкового клієнту MongoDB досить запустити команду:

```
$ mongo
```

Після цього СКБД MongoDB надасть відповідь з інформацією щодо версії, адреси сервера, ідентифікаційного номера, дати, часу тощо:

```
MongoDB shell version v4.4.6
connecting to:
mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("f2faf2a9-87ac-4d71-91cc-a77d00275e4a") }
MongoDB server version: 4.4.5
```

The server generated these startup warnings when booting:
2021-05-28T12:14:01.468+00:00: Using the XFS filesystem is strongly recommended with the WiredTiger storage engine.

>

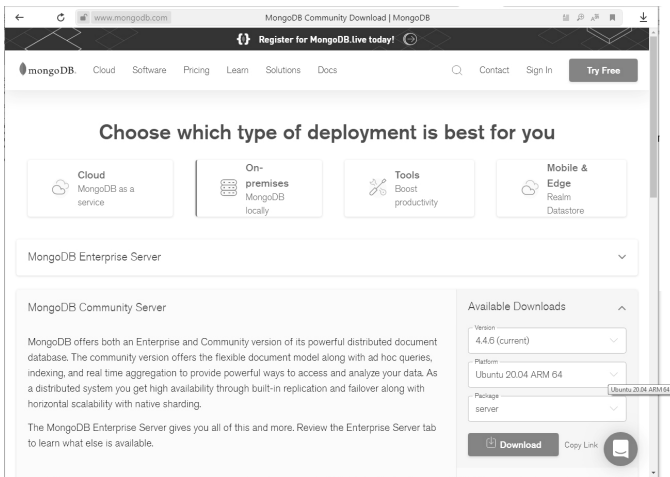


Рис. 12.1 – Стартова веб-сторінка MongoDB (<https://www.mongodb.com>)

Для отримання переліку активних баз даних необхідно ввести команду:

```
>db.getMongo().getDBNames()
```

або

```
>db.adminCommand('listDatabases')
```

Для підключення до бази даних необхідно ввести команду:

```
>use <ім'я бази даних>
```

Наприклад, коли ім'я бази даних – «sources» (джерела, наприклад, аккаунти соціальних мереж для подальшого моніторингу):

```
>use sources
```

Для отримання переліку колекцій в базі даних треба ввести команду:

```
>db.getCollectionNames()
```

Результат буде мати такий вигляд (приклад):

```
[ "tg", "ytc" ]
```

Для отримання кількості документів в колекції (на прикладі ім'я колекції ytc) необхідно ввести команду:

```
>db.ytc.find().count()
```

Для перегляду контенту всіх документів у колекції (на прикладі ім'я колекції ytc) достатньо ввести команду пошуку без параметра – запиту:

```
>db.ytc.find()
```

Результат буде мати такий вигляд (приклад):

```
{ "_id" : ObjectId("60cb88b9b8ca2f6093347e79"), "saddr" : "UCopQibF6yqBPSNFz9x4OYw", "name" : "Klimenko Time" }
{ "_id" : ObjectId("60cb88b9b8ca2f6093347e7a"), "saddr" : "UCQwVj4PyS5leCgEJY4I2t1Q", "name" : "DW Ukr" }
{ "_id" : ObjectId("60cb88b9b8ca2f6093347e7b"), "saddr" : "UCXoAjrdrHfa2hEL3Ug8REC1w", "name" : "DW Rus" }
{ "_id" : ObjectId("60cb88b9b8ca2f6093347e7c"), "saddr" : "UCD7dXP0dxuf2b341Bcfb4IA", "name" : "TMC TB" }
{ "_id" : ObjectId("60cb88b9b8ca2f6093347e7d"), "saddr" : "UCsxd-Drk7iYG50Ly-LVNWaw", "name" : "PEH TB" }
{ "_id" : ObjectId("60cb88b9b8ca2f6093347e81"), "saddr" : "UCdRzHJ8wUmDCr0wYM2zi07Q", "name" : "Криминформ" }
{ "_id" : ObjectId("60cb88b9b8ca2f6093347e82"), "saddr" : "UCw5pE2GE1cHeErKVyF76jjg", "name" : "TV Center" }
{ "_id" : ObjectId("60cb88b9b8ca2f6093347e83"), "saddr" : "UCUjpfpU1cVwbWD8yyCe5Zew", "name" : "Factorium Room" }
{ "_id" : ObjectId("60cb88b9b8ca2f6093347e84"), "saddr" : "UCJ9e_x5_ZuR9TwZwzvqRsIA", "name" : "Rogandar News" }
{ "_id" : ObjectId("60cb88b9b8ca2f6093347e85"), "saddr" : "UCPMf85Kr0Xtlr3bIYXdsZ3w", "name" : "RGD News" }
{ "_id" : ObjectId("60cb88b9b8ca2f6093347e86"), "saddr" : "UCgS19QtJ5nQki_ErxYVoIpA", "name" : "News 7" }
{ "_id" : ObjectId("60cb88b9b8ca2f6093347e87"), "saddr" : "UCbJYQHINGGcUoDYKIQZ816A", "name" : "Глеб Волин" }
{ "_id" : ObjectId("60cb88b9b8ca2f6093347e88"), "saddr" : "UCBi2mrWuNuyY4qbM6FU18Q", "name" : "ABC News" }
{ "_id" : ObjectId("60cb88b9b8ca2f6093347e89"), "saddr" : "UC7aFhtshxOnaqioFAJ0ZSvA", "name" : "Vovan222prank" }
{ "_id" : ObjectId("60cb88b9b8ca2f6093347e8a"), "saddr" : "UC6ZF9N9Tx6xh-skXCuRHCDpQ", "name" : "PBS NewsHour" }
{ "_id" : ObjectId("60cb88b9b8ca2f6093347e8b"), "saddr" : "UCupvZG-5ko_eiXaupDfxWw", "name" : "CNN" }
{ "_id" : ObjectId("60cb88b9b8ca2f6093347e8c"), "saddr" : "UCv1qjOZN3ZvDemZfHuKICIQ", "name" : "CGTN Live" }
```

Type "it" for more

Якщо ми просто хочемо побачити один документ з колекції, можна використовувати функцію findOne:

```
> db.movies.findOne()
```

```
{
  "_id" : ObjectId("5721794b349c32b32a012b11"),
  "title" : "Star Wars: Episode IV – A New Hope",
  "director" : "George Lucas",
  "year" : 1977
}
```

Для пошуку документів в колекції (на прикладі ім'я колекції tg) необхідно ввести команду з параметром – запитом (на прикладі, за полем name):

```
>db.tg.find(name='@Aavst')
```

Результат буде мати приблизно такий вигляд:

```
{ "_id" : ObjectId("60cb8af025ad9284143be469"), "saddr" : "@Aavst" }
```

Для пошуку, наприклад, за іншим полем (поле _id), необхідно ввести запит:

```
> db.tg.find({"_id": ObjectId("60cb8af025ad9284143be46c")})
```

Для виходу із рядкового редактора треба ввести команду:

```
>exit
```

Для видалення колекції необхідно ввести команду:

```
>db.<ім'я колекції>.drop()
```

Для видалення бази даних необхідно ввести команду:

```
>db.dropDatabase()
```

Імпорт бази даних із таблиці у зовнішньому форматі

Якщо користувач має у розпорядженні масив даних (таблицю), представлений, зокрема, у форматі CSV, то він може створити і завантажити базу даних MongoDB цими даними. Він може скористатися утилітою mongoimport, для імпортування відповідної таблиці. Нижче наведено приклад створення і завантаження бази даних із таблиці у форматі CSV. Для цього достатньо застосувати команду, що задається із командного рядку операційної системи:

```
$mongoimport -d <ім'я бази> -c <ім'я колекції>  
--type <тип файлу> -file <ім'я файлу> --fields <поля через кому>
```

Приклад команди:

```
$mongoimport -d sources -c ytc --type csv -file ytc.csv --fields  
saddr,name
```

Приклад результату:

```
2021-06-17T20:39:05.040+0300    connected to: mongodb://localhost/
2021-06-17T20:39:05.221+0300    104 document(s) imported successfully.
0 document(s) failed to import.
```

Експорт бази даних в таблицю у зовнішньому форматі

Якщо користувач має у розпорядженні базу даних у середовищі СКБД MongoDB, то він має можливість експорту (вивантаження) інформації із цієї бази даних у зовнішню таблицю, представлену, зокрема, у форматі CSV, яку можна використовувати як вхідні дані для інших програм. Він може скористатися утилітою `mongoexport`, для імпортування відповідної таблиці. Нижче наведено приклад експорту інформації в таблицю у форматі CSV. Для цього достатньо застосувати команду, що задається із командного рядку операційної системи:

```
$mongoexport --<тип файлу> -o <ім'я файлу> -d <ім'я бази> -c <ім'я
колекції> -f =<поля через кому>
```

Приклад команди:

```
$mongoexport --csv -o /tmp/ytcl.csv -d sources -c ytc -f saddr,name
```

Приклад звіту щодо результату:

```
2021-06-17T20:43:13.164+0300    csv flag is deprecated; please use --
type=csv instead
2021-06-17T20:43:13.197+0300    connected to: mongodb://localhost/
2021-06-17T20:43:13.247+0300    exported 104 records
```

Питання до практичної роботи

1. Що таке NoSQL СКБД, які їх основні властивості?
2. Як встановлюється система MongoDB?
3. Які основні команди взаємодії із MongoDB задаються із командного рядка?
4. В яких форматах можна імпортувати або експортувати інформацію до/із баз даних MongoDB.

Завдання на самостійну роботу

1. Встановити систему MongoDB
2. Ознайомитись з основними поняттями СУБД MongoDB. У чому відмінність від SQL-СУБД?
3. Ознайомитись із особливостями Імпорту/експорту даних із формату JSON.

Література

1. Karl Seguin. “The Little MongoDB Book”. –34 p. URL: <http://github.com/karlseguin/the-little-mongodb-book>
2. Шеннон Брэдшоу, Йон Брзіл, Кристина Ходоров. “MongoDB: полное руководство. Мощная и масштабируемая система управления базами данных” – М.: ДМК Пресс, 2020. – 540 с.
3. Маркин А.В. Постреляционные базы данных. MongoDB : учебное пособие / Маркин А.В.. — Москва : Ай Пи Ар Медиа, 2020. — 383 с. — ISBN 978-5-4497-0632-4.
4. Import и Export База данных MongoDB. URL: <https://betacode.net/10279/importing-and-exporting-mongodb-database>

13. Клієнт Compass СУБД MongoDB

Мета практичної роботи, що відповідає розділу, – отримати досвід роботи з системою Compass – графічним інтерфейсом користувача (GUI) СКБД MongoDB, встановленням цього GUI, роботи з даними, наведені ними у форматі JSON, командами CURD (Create – створення, Read – читання, Update – модифікація, Delete – видалення).

З версії MongoDB 3.2 як графічна оболонка поставляється MongoDB Compass.

Встановлення графічного інтерфейсу користувача Compass

Для завантаження MongoDB Compass необхідно перейти на офіційну адресу цього GUI <https://www.mongodb.com/try/download/compass>. На цій сторінці можна вибрати опції для завантаження – версію Compass та цільову операційну систему (Рис. 13.1).

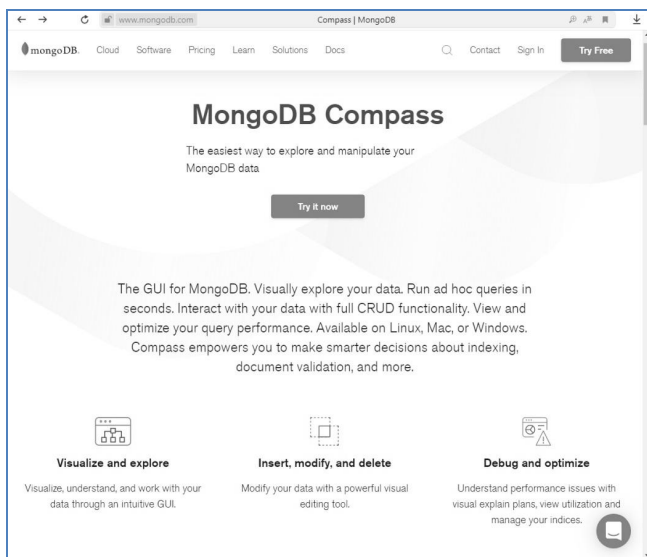


Рис. 13.1 – Стартова сторінка MongoDB Compass

Після завантаження і розпакування архіву MongoDB Compass отримуємо каталог (Рис. 13.2).

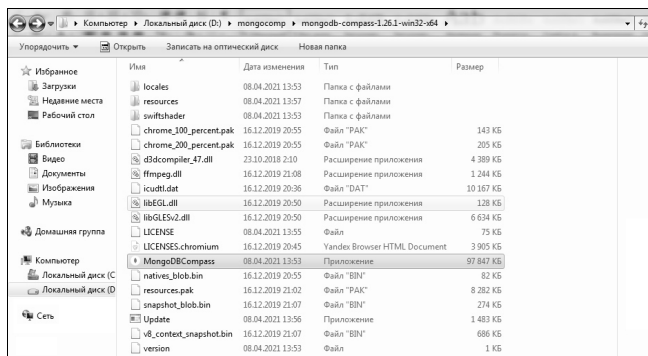


Рис. 13.2 – Директорія MongoDB Compass на клієнтському комп'ютері

Запуск GUI MongoDB Compass

Тут можна знайти файл MongoDBCompass.exe. Саме він і представляє модуль програми. Активізувавши цей модуль отримаємо інтерфейс системи, куди треба ввести адресу сервера і порта, де вже активізований сервіс mongod (Рис. 13.3).

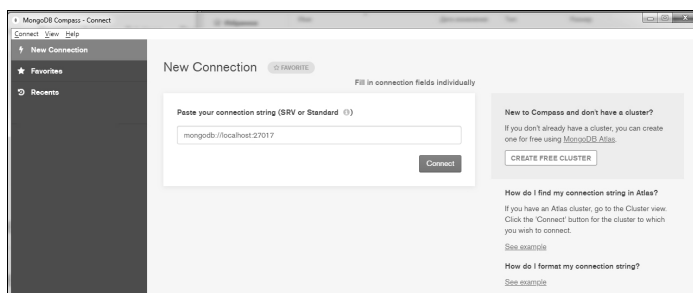


Рис. 13.3 – Налаштування клієнту MongoDB Compass

Після переходу за вказаною адресою (кнопка Connect) відображується перехід до переліку активних баз даних (Рис. 13.4).

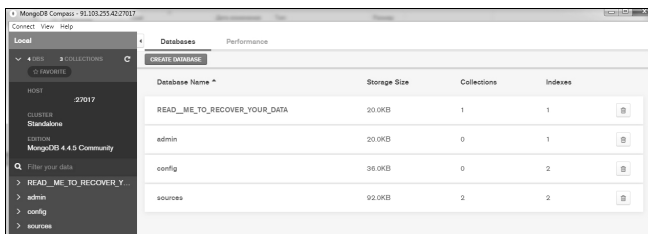


Рис. 13.4 – Перелік баз даних MongoDB Compass

У попередньому розділі було надано відомості щодо створення бази даних sources шляхом імпорту з таблиці. Перейдемо до детального розгляду цієї бази даних шляхом активізації відповідного гіперпосилання (Рис. 13.5).

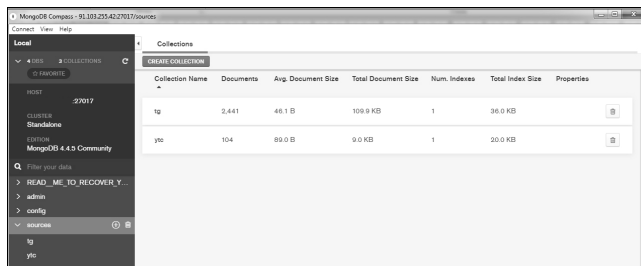


Рис. 13.5 – Перелік колекцій бази даних sources

На наведеному прикладі можна бачити, що база даних sources містить дві колекції – tg і ytc (колекції адрес каналів Telegram і Youtube, відповідно). В наведеній таблиці можна бачити кількість документів у кожній колекції (2441 і 104), середню довжину документа, кількість місця на диску, зайнятого колекціями, кількість індексів (допоміжних пошукових файлів), обсяг індексів.

Реалізація команд CRUD

Перехід до перегляду конкретної колекції (наприклад, ytc, канали Youtube) здійснюється шляхом активізації відповідного гіперпосилання.

Як можна бачити на Рис. 13.6, MongoDB Compass реалізує доступ до перегляду окремих документів колекції, наведених у форматі JSON.

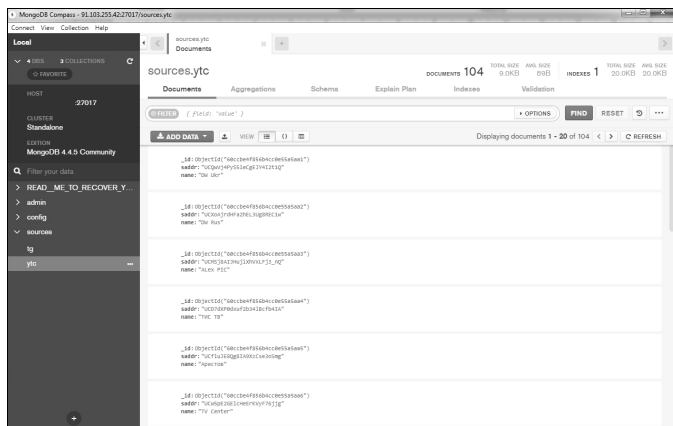


Рис. 13.6 – Вміст колекції ytc

Крім того, реалізація функції CRUD – R (Read, читання) можлива також у режимі пошуку. Для цього у полі Filter достатньо ввести запит у форматі JSON, наприклад, {name: 'Балтия недели'}. Результат наведено на Рис. 13.7.

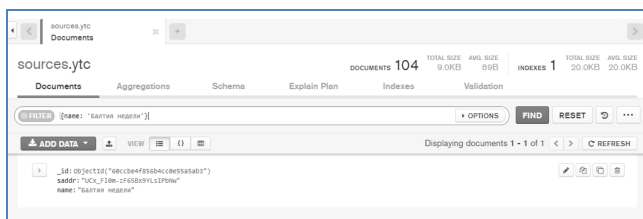


Рис. 13.7 – Результат пошуку

Реалізація D (Delete): Як бачимо на операція видалення для всієї колекції (аналог db.<ім'я колекції>.drop()) реалізовано на інтерфейсі (Рис. 13.5) у вигляді іконки «Корзина для сміття».

Таку ж операцію для всієї бази даних (аналог `db.drop.Database()`) можна побачити на Рис. 13.5 у вигляді тієї ж самої іконки.

Для окремого запису досить активізувати аналогічну іконку, наведену на Рис. 13.7.

Для реалізації функції U (Update) для окремого документа необхідно активізувати іконку Edit, після чого можна редагувати окремі поля документа і додавати нові (Рис. 13.8).

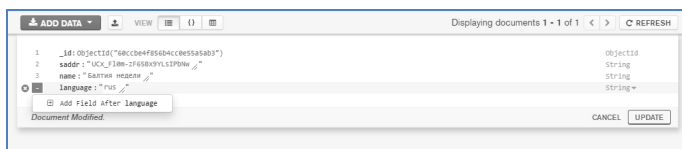


Рис. 13.8 – Результат пошуку

C(Create) – створення нових документів (режим ADD DATA) можливо у двох варіантах (ручного введення або імпорту файлів – Рис. 13.9, 13.10).

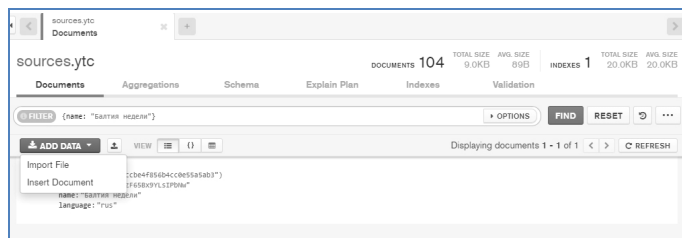


Рис. 13.9 – Режим ADD DATA

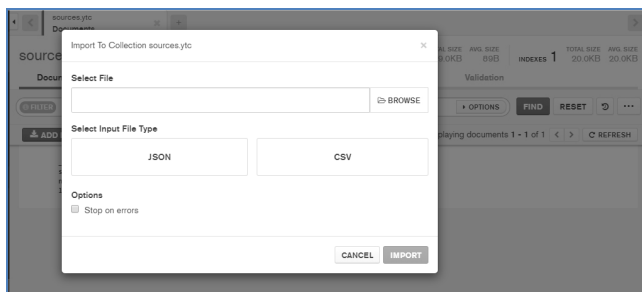


Рис.13.10 – Створення нового документу

При імпорті даних завантаження здійснюється шляхом фонового виконання команди `mongoimport` після активізації клавіші `IMPORT`.

Питання до практичної роботи

1. Що таке CRUD?
2. Як встановлюється система MongoDB Compass?
3. Як реалізуються основні команди взаємодії з MongoDB через MongoDB Compass?
4. В яких форматах можна імпортувати або експортувати інформацію до/із баз даних MongoDB

Завдання для самостійної роботи

1. Встановити GUI MongoDB Compass
2. Ознайомитись з основними можливостями інтерфейсу GUI MongoDB Compass.
3. Можливості CRUD у середовищі MongoDB Compass.

Література

1. Karl Seguin. “The Little MongoDB Book”. –34 p. URL: <http://github.com/karlseguin/the-little-mongodb-book>
2. Шеннон Бредшоу, Йон Брзил, Кристина Ходоров. “MongoDB: полное руководство. Мощная и масштабируемая система управления базами данных”. – М.: ДМК Пресс, 2020. – 540 с.
3. А.Н. Филиппов, Я.А. Поволоцкий. Применение нереляционной СУБД MongoDB в САПР ТП / Методическое пособие // Описание применения. СПб: Университет ИТМО, 2018. – 46 с. URL: <https://books.ifmo.ru/file/pdf/2441.pdf>
4. MongoDB for Giant Ideas (Електронний ресурс <https://www.mongodb.com/>).

14. Основи роботи з Neo4j

Мета практичної роботи, що відповідає розділу, – отримати досвід роботи з графовою СКБД Neo4j використовуючи Neo4j Browser та використовуючи мову запитів Cypher, яка надає людино-читану мову запитів до графових баз даних, що схожа на SQL.

Безумовно, концепція «великих даних» безпосередньо пов'язана з мережними задачами, оскільки кількість можливих зв'язків у мережах значно перевищує кількість вузлів. Для роботи з такими задачами створюються спеціальні програмні системи. У тому числі Neo4j — графова система керування базами даних із відкритим вихідним кодом, реалізована на Java. Вважається найпоширенішою графовою СКБД.

Дані в Neo4j зберігаються у власному форматі, спеціально пристосованому для представлення графової інформації, такий підхід у порівнянні з моделюванням графової бази даних засобами реляційної СКБД дозволяє застосовувати додаткову оптимізацію у разі даних із складнішою структурою.

Для обробки графа не потрібне його розміщення цілком в оперативну пам'ять обчислювального вузла, таким чином, можливе оброблення досить великих графів.

Інтерфейс прикладного програмування для СКБД реалізований для багатьох мов програмування, включаючи Java, Python, Clojure, Ruby, PHP, також реалізовано API у стилі REST.

У СКБД використовується власна мова запитів – Cypher, який є не тільки мовою запитів, але і мовою маніпулювання даними, так як надає функції CRUD для графового сховища.

Цей розділ присвячено основам роботи з графовою СКБД Neo4j, використовуючи її найважливішу частину – Neo4j Browser.

Стартову сторінку веб-серверу (<https://neo4j.com>) наведено на Рис. 14.1.

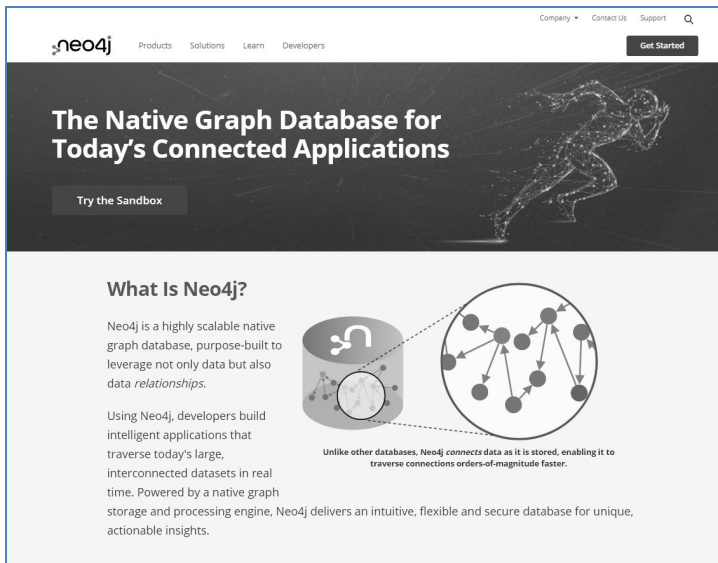


Рис. 14.1 – Стартова веб-сторінка Neo4j

Встановлення Neo4j

Для встановлення Neo4j необхідно перейти до центру завантаження за адресою: <https://neo4j.com/download-center/> (Рис. 14.2).

Для подальшої роботи Neo4j мають виконуватись такі початкові вимоги:

- Для роботи СКБД Neo4j необхідно мінімум 2Gb оперативної пам'яті, а для стабільної роботи рекомендується 16Gb.
- У якості дискового масиву рекомендується застосовувати SSD-диски.
- Neo4j реалізована на Java, тому необхідне встановлення JVM8.

Перед встановленням самої системи, спочатку треба скачати зашифрований GPG-ключ і додати репозиторій до локального списку:

```
wget --no-check-certificate -O -  
https://debian.neo4j.org/neotechnology.gpg.key | sudo apt-key add  
-
```

```
echo 'deb http://debian.neo4j.org/repo stable/' >  
/etc/apt/sources.list.d/neo4j.list
```

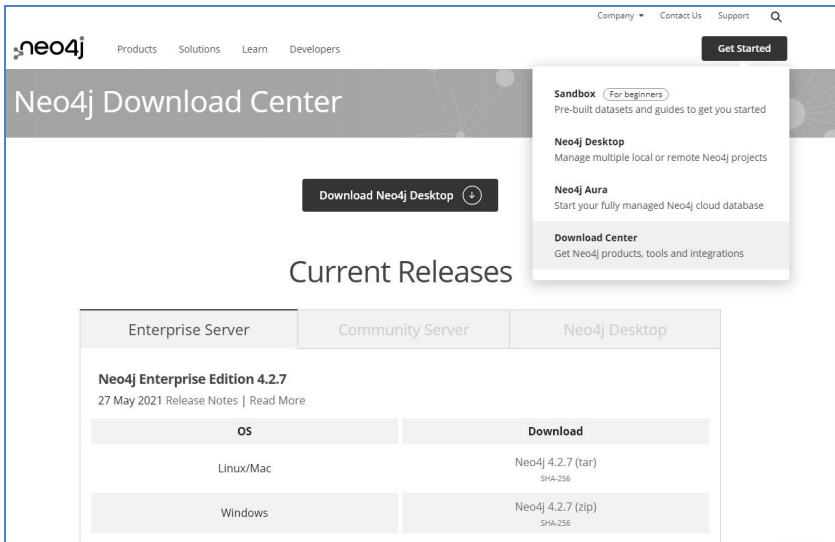


Рис. 14.2. Центр завантаження Neo4j

Після цього треба оновити локальну базу пакетів і встановити СКБД:

```
apt update apt install neo4j
```

Для того, щоб підключитися до бази даних з будь-якого IP-адреси, а не тільки на локальному хості, в файлі конфігурації необхідно змінити деякі рядки. Для цього за допомогою текстового редактора необхідно відкритий файл налаштувань:

```
/etc/neo4j/neo4j.conf
```

Необхідно знайти рядки, вони знаходяться у різних частинах файлу:

```
#dbms.connector.http.listen_address=:7474  
#dbms.connector.bolt.listen_address=:7687
```

Необхідно зняти коментарі, як показано нижче:

```
dbms.connector.http.listen_address=0.0.0.0:7474  
dbms.connector.bolt.listen_address=0.0.0.0:7687
```

Після збереження змін треба перезавантажити Neo4j:

```
service neo4j restart
```


При потребі здійснюється налаштувати firewall для віддаленого доступу:

```
iptables -A INPUT -p tcp --dport 7474 -j ACCEPT  
iptables -A INPUT -p tcp --dport 7687 -j ACCEPT
```

Підключення в браузері

Необхідно відкрити браузер і підключитися до СКБД за наступною адресою:

<IP-адреса Neo4J>: 7474

Для роботи з базою даних застосовується мережний протокол bolt – високоефективний і легкий клієнт-серверний протокол, призначений для застосунків баз даних.

При першому підключенні замініть значення localhost на ваш IP-адресу для протоколу bolt, а в якості пароля використовуйте назву СКБД, далі вкажіть і підтвердить пароль для користувача Neo4J (Рис. 14.3).

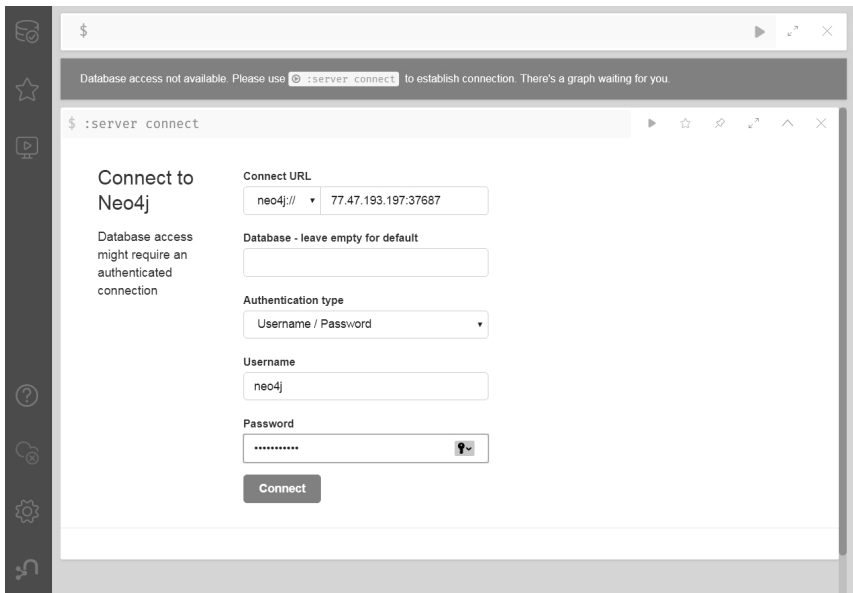


Рис. 14.3 – З'єднання з Neo4J

Після цього можна почати роботу з графовою СКБД Neo4J.

Запуск системи

Для початку роботи з Neo4J необхідно перевірити, що СКБД запущена (приклад для Linux):

```
service -status-all | grep neo4j  
[ + ] neo4j
```

Знак «плюс» означає, що СКБД вже запущена, «мінус» - ще немає. Для запуску Neo4J виконайте команду:

```
sudo service neo4j start
```

Після запуску за допомогою звичайного браузера можна перейти за посиланням <http://localhost:7474/browser/>, після чого повинен з'явитися інтерфейс Neo4j Browser (Рис. 14.4).

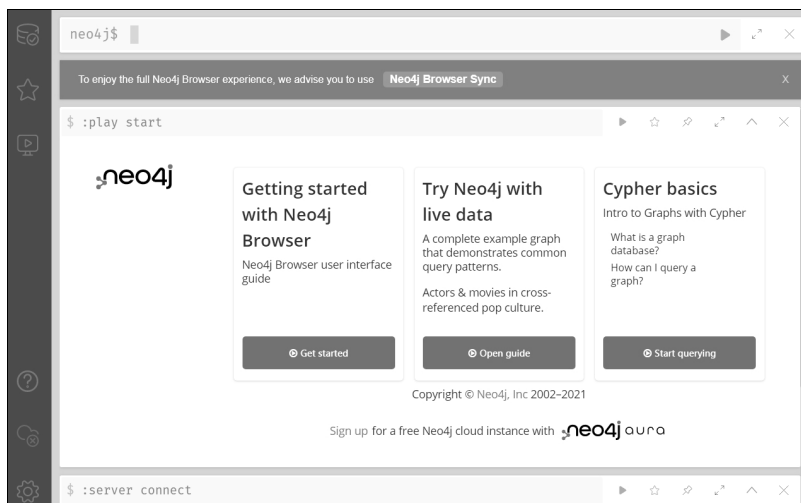


Рис. 14.4 – Інтерфейс Neo4j Browser

У верхній частині вікна Neo4j Browser розташовується рядок так званого редактора. Починаючи набір команд з двокрапки, побачимо список всіх доступних команд з коротким описом (Рис. 14.5).

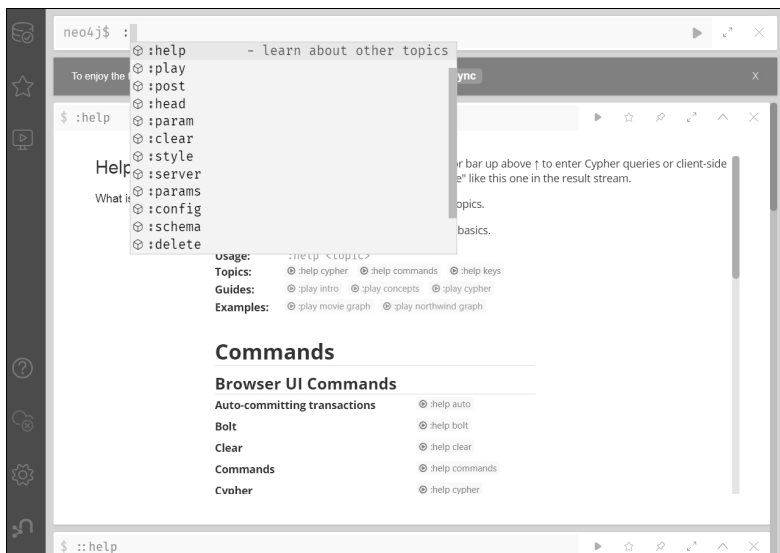


Рис. 14.5 – Список команд Neo4j Browser

Детальний опис команд можна отримати через виклик команди :help (Рис. 14.6).

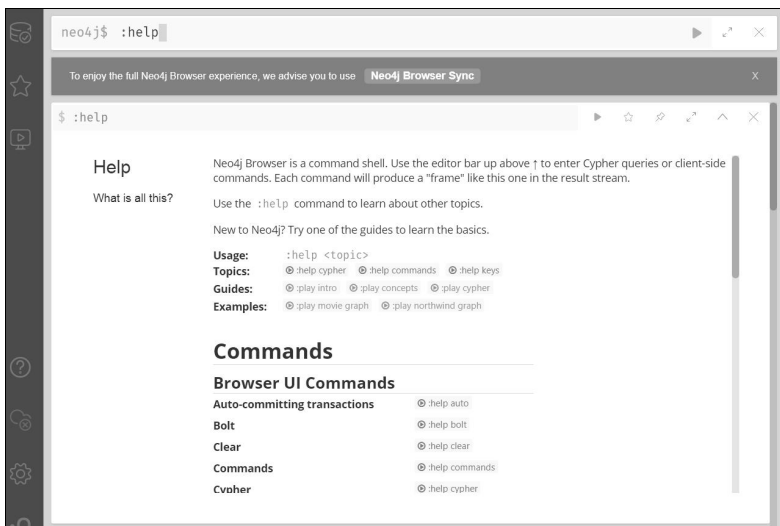


Рис. 14.6 – Команда :help

Створення графа

Для створення і подальшої обробки графів будемо застосовувати мову Cypher, яка є мовою маніпулювання даними, а також надає функції CRUD для графового сховища.

Для короткого ознайомлення з мовою Cypher можна викликати команду (Рис. 14.7):

```
:play cypher
```

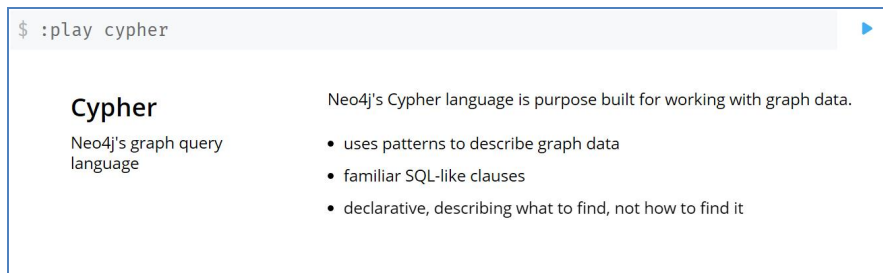


Рис. 14.7 – Опис мови Cypher

Для початку створення невеликого графу соціальної спрямованості перейдемо в редактор і наберемо першу команду на мові Cypher:

```
CREATE (u1:Person {name: "Василь", from: "Житомир"})
```

Після виконання команди Browser видається результат (Рис. 14.8).

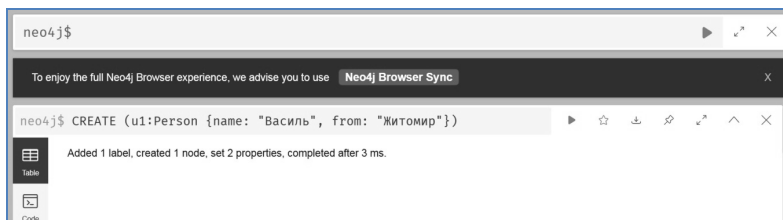


Рис. 14.8 – Створення вузла

Додамо ще один вузол:

```
CREATE (u2:Person {name: "Дмитро", from: "Київ"})
```

Тепер запитасмо всі вузли типу Person і винесемо значення властивості name (Рис. 14.9):

```
MATCH (ee:Person) RETURN ee.name
```

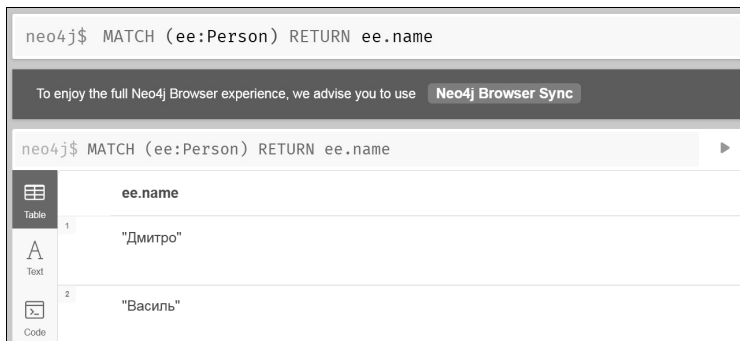


Рис. 14.9 – Відображення переліку вузлів

Існує можливість упорядкувати відобуті дані за якимось полем:

```
MATCH (ee:Person) RETURN ee.name ORDER BY ee.name
```

Далі можемо запитати усі вузли цього типу (Рис. 14.10):

```
MATCH (ee:Person) RETURN ee
```

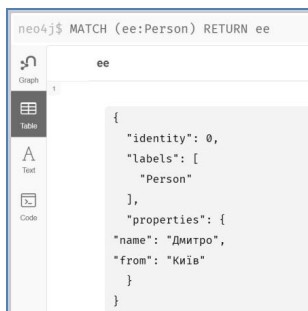


Рис. 14.10 – Повний опис вузла

Натиснувши на кнопку Graph можна побачити наші вузли в графічному вигляді (14.11).



Рис. 14.11 – Графічне представлення вузлів

Зв'язки між вузлами в Cypher можна додавати за допомогою команди `CREATE` (Рис. 14.12, 14.13).

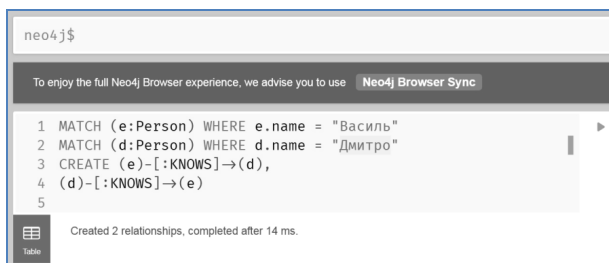


Рис. 14.12 – Графічне представлення вузлів

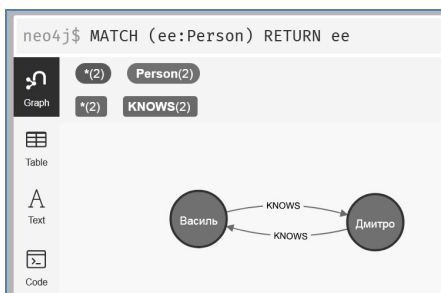


Рис. 14.13 – Відображення графу із зв'язками

За допомогою Cypher можна також виконувати різні операції над графами, наприклад, запитувати суміжні вершини, друзів в соціальному графі, видаляти ребра і вершини і багато іншого, але це тема для окремої розмови.

Також можна налаштовувати Neo4j Browser на різний стиль відображення вузлів і зв'язків в залежності від заданих їм міток.

Видалення графа (Рис. 14.14):

```
Match (n)-[r]-()
Delete n,r;
```

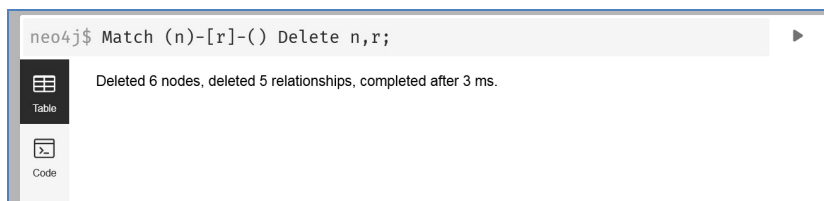


Рис. 14.14 – Видалення графа

Імпорт даних із формату .csv в граф neo4j

Представляємо зв'язок між іменованими вузлами (на прикладі будемо позначати їх цифрами). Розглянемо приклад, сукупність рядків, в яких кожен рядок виражає відношення суміжності між двома вузлами, наприклад, 1->2, 2->3, 1->3, і т. д.

```
1,2
2,3
1,3
1,4
2,5
3,4
3,5
4,5
```

Цей файл, в якому записані ці рядки, слід розміщувати на локальній машині (для стандартної установки neo4j)

```
/var/lib/neo4j/import/
```

Ім'я файлу, наприклад, csv

Тоді команди завантаження файлу, формування і відображення графу будуть мати вигляд:

```
LOAD CSV FROM 'file:///csv' AS line
MERGE (a:node {name:line[0]})
MERGE (b:node {name:line[1]})
MERGE (a)-[:connects]->(b);
```

Після чого мережу можна відобразити (Рис. 14.15) шляхом введення команди:

```
MATCH (ee:node) RETURN ee
```

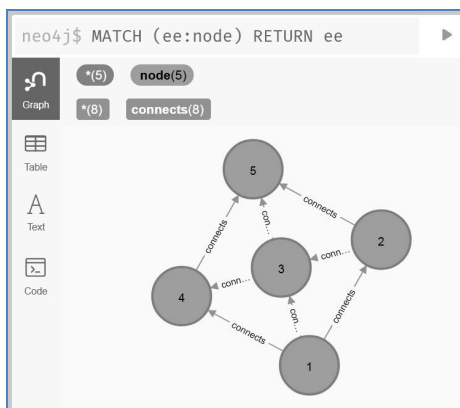


Рис. 14.15 – Відображення завантаженого графа

Питання до практичної роботи

1. Що таке графові СКБД та які їх основні особливості?
2. Які інші СКБД, крім Neo4J, підтримує мова запитів Cypher?
3. Які CRUD- функції можна використати для роботи з графовим сховищем?

Завдання на самостійну роботу

1. Детально ознайомитись із основними можливостями графової СУБД . Neo4J і мови Cypher
2. Ознайомитись із можливостями завантаження графів із формату JSON

3. Ознайомитись із можливостями оформлення візуалізації графів за допомогою команди `:style`.

Література

1. Робинсон Ян, Вебер Джим, Эифрем Эмиль. Графовые базы данных: новые возможности для работы со связанными данными / пер. с англ. Р. Н. Рагимова; науч. ред. А. Н. Кисилев. – 2-е изд. – М.: ДМК Пресс, 2016. – 256 с. ISBN 978-5-97060-201-0
2. Эрик Редмонд, Джим. Р. Уилсон. Семь баз данных за семь недель. Введение в современные базы данных и идеологию NoSQL. Под редакцией Жаклин Картер / Пер. с англ. Слинкин А. А. – М.: ДМК Пресс, 2013. – 384с. ISBN 978-5-94074-866-3
3. Holzschuher, Florian and Peinl, Rene (2013). "Performance of Graph Query Languages: Comparison of Cypher, Gremlin and Native Access in Neo4J" in EDBT '13. Proceedings of the Joint EDBT/ICDT 2013 Workshops: 195-204, Genoa, Italy: ACM. DOI:10.1145/2457317.2457351

Додаток. Необхідний мінімум з мови Python

Python – це універсальна мова програмування, що підтримує об'єктно-орієнтоване програмування, створена в 1991 Гвідо ван Россумом. Офіційний сайт мови програмування <http://python.org>. Python розповсюджується вільно на підставі ліцензії подібної до GNU General Public License.

Python – мова, що інтерпретується: вихідний код перетворюється на машинний в процесі інтерпретації.

Типи даних

В Python використовуються типи даних:

1. None (невизначене значення змінної)
2. Логічні змінні (Boolean Type)
3. Числа (Numeric Type)
 - int – ціле число
 - float – число з плаваючою точкою
 - complex – комплексне число
4. Списки (Sequence Type)
 - list – список
 - tuple – кортеж
 - range – діапазон
5. Строки (Text Sequence Type)
 - str

У деяких випадках програма отримує дані у вигляді рядків, а оперувати має числами (або навпаки). У разі використовуються спеціальні функції (особливі оператори), дозволяють перетворити один тип даних на інший. Так функція `int()` перетворює передану їй рядок (або число з плаваючою точкою) у ціле, функція `str()` перетворює переданий їй аргумент на рядок, `float()` – на дробове число.

Рядки як послідовності символів

Рядок – це складний тип даних, що є послідовністю символів. Рядки в мові Python можуть укладатися як в одиначні, так і в подвійні лапки. Однак, початок та кінець рядка повинні обрамлятися однаковим типом лапок.

Рядок – базовий тип, що представляє собою незмінну послідовність символів (str від англ. «string» — «рядок»).

Функції та методи роботи із рядками

Функція або метод	Призначення
S1 + S2	Конкатенація (складання рядків)
S1 * 3	Повторення рядку
S[i]	Звернення за індексом
S[i:j:step]	Витяг зрізу
len(S)	Довжина рядка
S.join(список)	З'єднання рядків із послідовності str через роздільник, заданий рядком
S1.count(S[, i, j])	Кількість входжень підрядка s рядок s1. Результатом є кількість. Можна вказати позицію початку пошуку i та закінчення пошуку j
S.find(str, [start],[end])	Пошук підрядка у рядку. Повертає номер першого входження або -1
S.index(str, [start],[end])	Пошук підрядка у рядку. Повертає номер першого входження або викликає ValueError
S.rindex(str, [start],[end])	Пошук підрядка в рядку. Повертає номер останнього входження або викликає ValueError
S.replace(шаблон, заміна)	Заміна шаблону
S.split(символ)	Розбиття рядка по роздільнику
S.upper()	Перетворення рядка до верхнього регістру
S.lower()	Перетворення рядка до нижнього регістру

Існує функція `len()`, що дозволяє виміряти довжину рядка. Результатом її виконання є число, яке показує кількість символів у рядку.

Також для рядків існують операції конкатенації (+) та дублювання (*).

Приклади:

```
>> len('It is a long string')
```

Результат:

```
19
```

```
>> '!!!' + ' Hello World ' + '!!!'
```

Результат:

```
'!!! Hello World !!!'
```

```
>> '-' * 20
```

Результат:

```
'-----'
```

У послідовностях важливий порядок символів, кожен символ у рядку має унікальний порядковий номер — індекс. Можна звертатися до конкретного символу в рядку та витягувати його за допомогою оператора індексування, який є квадратними дужками з номером символу в них.

```
>>> 'morning, afternoon, night'[1]
```

Результат:

```
'o'
```

```
>>> tday = 'morning, afternoon, night'
```

```
>>> tday[4]
```

Результат:

```
'i'
```

У прикладі, вираз `'morning, afternoon, night'[1]` призвело до вилучення другого символу. Справа в тому, що індексація починається не з одиниці, а з нуля. Також можна витягувати символи, починаючи відлік з кінця. І тут відлік починається з -1 (останній символ).

Можна витягувати з рядка не один символ, а кілька, тобто. отримувати зріз (підрядок). Оператор вилучення зрізу з рядка має такий вигляд: [X:Y]. X – це індекс початку зрізу, а Y – його закінчення; причому символ із номером Y до зрізу вже не входить. Якщо немає першого індексу, то зріз береться від початку до другого індексу; за відсутності другого індексу, зріз береться від першого індексу до кінця рядка.

```
>>> tday = 'morning, afternoon, night'
>>> tday[0:7]
'morning'
>>> tday[9:-7]
'afternoon'
>>> tday[-5:]
'night'
```

Списки – послідовності, що змінюються

Списки Python, як і рядки, є впорядкованими послідовностями. Однак, на відміну від рядків, списки складаються не з символів, а з різних об'єктів (значень, даних), і полягають не в лапки, а в квадратних дужках []. Об'єкти відокремлюються один від одного за допомогою коми.

Списки можуть складатися з різних об'єктів: чисел, рядків та навіть інших списків. В останньому випадку списки називають вкладеними.

```
[23, 656, -20, 67, -45] # список цілих чисел
[4.15, 5.93, 6.45, 9.3, 10.0, 11.6] # список із дробових чисел
["Katy", "Sergei", "Oleg", "Dasha"] # список із рядків
["Москва", "Титова", 12, 148] # змішаний список
[[0, 0, 0], [0, 0, 1], [0, 1, 0]] # список, що складається зі
списків
```

Як і над рядками над списками можна виконувати операції з'єднання та повторення:

```
>>> [45, -12, 'april'] + [21, 48.5, 33]
[45, -12, 'april', 21, 48.5, 33]
>>> [[0,0],[0,1],[1,1]] * 2
[[0, 0], [0, 1], [1, 1], [0, 0], [0, 1], [1, 1]]
```

Матрицями називаються масиви елементів, які у вигляді прямокутних таблиць, котрим визначено правила математичних процесів. Елементами матриці можуть бути числа, символи алгебри або математичні функції.

Для роботи з матрицями Python також використовуються списки. Кожен елемент списку-матриці містить вкладений список.

Таким чином, формується структура вкладених списків, кількість яких визначає кількість стовпців матриці, а число елементів всередині кожного вкладеного списку вказує на кількість рядків у вихідній матриці.

Двовимірний список не можна створювати за допомогою операції повторення одного рядка. Для створення списку слід спочатку створити список із n елементів (для початку просто з n нулів), а потім зробити кожен елемент списку посиланням на інший одновимірний список із m елементів.

Приклад:

```
n=3
m=3
A=[0]*n
for i in range(n):
    a[i]=[0]*m
print('A:',A)
```

Результат:

```
A: [[0, 0, 0], [0, 0, 0], [0, 0, 0]]
```

Виведення вкладеного списку (двовимірного масиву)

Для обробки та виведення списку зазвичай використовується два вкладені цикли. Перший цикл за номером рядка, другий цикл за елементами всередині рядка. Наприклад, вивести двовимірний числовий список на екран рядково, розділяючи числа пробілами всередині одного рядка, можна так:

```
for i in range(n):
    for j in range(n):
        print(A[i][j], end = ' ')
    print()
```

Словники

Одним із складних типів даних (поряд із рядками та списками) у мові програмування Python є словники. Словник - це змінний (як список)

невпорядкований (на відміну від рядків та списків) набір пар "ключ: значення".

Синтаксис словника на Пітоні можна описати такою схемою:

```
{ ключ: значення, ключ: значення, ключ: значення, ...}
```

У словнику доступ до значень здійснюється за ключами, які полягають у квадратних дужках.

```
>>> dic = {'cat': 'кішка',  
'dog': 'пес', 'bird': 'птиця', 'mouse': 'миша'}  
>>> dic['cat']  
'кішка'  
>>> dic['bird']  
'птиця'
```

Словники, як і списки, є типом даних, що змінюється: можна змінювати, додавати і видаляти елементи (пари "ключ: значення"). Спочатку словник можна створити порожнім (наприклад, `d = {}`) і потім заповнити його елементами. Додавання та зміна має однаковий синтаксис: словник `[ключ] = значення`.

Ключ може бути як існуючим (тоді відбувається зміна значення), так і новим (відбувається додавання елемента словника). Видалення елемента словника здійснюється за допомогою функції `del ()`.

```
>>> dic  
={'cat': 'кішка', 'dog': 'пес', 'bird': 'птиця', 'mouse': 'миша'}  
>>> dic['elephant'] = 'слон'  
>>> dic['fox'] = 'лисиця'  
>>> dic
```

Оператори

Умовний оператор. Інструкція `if`

Іноді частина коду повинна виконуватись лише за певного значення конкретної змінної. Зазвичай у мовах програмування використовується приблизно така конструкція:

```
if a логічний оператор b:  
    вираз, що виконується у випадку результату True
```

Приклад реалізації:

```
if numbig < 100: # якщо значення змінної numbig менше 100, то...
    c = a**b # звести значення змінної a до ступеня b,
    # результат присвоїти c.
```

(# – ознака коментаря, все що слідує ним не інтерпретується програмно)

Зустрічається і складніша форма розгалуження: if–else. Якщо умова при інструкції if виявляється помилковою, то виконується блок коду при інструкції else.

Приклад коду з гілкою else:

```
print "Привіт"
товар1 = 50
товар2 = 32
if товар1+ товар2 > 99 :
    print "Сума не достатня"
else:
    print "Чек сплачено"
print "До побачення"
```

У Python передбачено спеціальне розширення інструкції if, що дозволяє направити потік виконання програми по одній з багатьох гілок. Ця розширена інструкція, крім необов'язкової частини else, містить низку гілок elif (скорочення від "else if" - "ще якщо").

Синтаксис оператора if – elif – else виглядає так:

```
if логічний вираз_1:
    команда_1
    команда_2
    ...
    команда_n
elif логічний вираз_2:
    команда_1
    команда_2
    ...
    команда_n
elif логічний вираз_3:
    команда_1
    команда_2
    ...
    команда_n
else:
    команда_1
    команда_2
```



```
...  
команда_n
```

Приклади скрипту з використанням інструкції if-elif-else:

```
x = -10  
if x > 0:  
    print 1  
elif x < 0:  
    print -1  
else:  
    print 0
```

Цикли

Цикл while

Цикли - це інструкції, що виконують одну і ту ж послідовність дій, поки діє задана умова.

Універсальний організатор циклу в мові програмування Python (як і в багатьох інших мовах) є конструкція while.

Конструкцію while мовою Python можна описати наступною схемою:

```
while a логічний оператор b:  
    дія  
    зміна a
```

Логічне вираз у заголовку циклу while може бути складним, а може змінюватися змінна (або вираз) b.

Коли виконання програмного коду доходить до циклу while, виконується логічний вираз у заголовку, і якщо було отримано True (істина), виконуються вкладені вирази. Після потік виконання програми знову повертається в заголовок циклу while і знову перевіряється умова.

Найпростіший цикл може виглядати так:

```
str1 = "+"  
i = 0  
while i < 10:  
    print (str1)  
    i = i + 1
```

Цикл for

У програмах, написаних на Пітоні, широко застосовується цикл for, який є циклом обходу заданої множини елементів і виконання у своєму тілі різних

операцій над ними. Наприклад, якщо є список чисел, і необхідно збільшити значення кожного елемента на дві одиниці, можна перебрати список за допомогою циклу `for`, виконавши над кожним його елементом відповідну дію.

```
>>> spisok = [0,10,20,30,40,50,60,70,80,90]
>>> i = 0
>>> for element in spisok:
    spisok[i] = element + 2
    i = i + 1
>>> spisok
[2, 12, 22, 32, 42, 52, 62, 72, 82, 92]
```

Коли всі елементи оброблені цикл `for` закінчує свою роботу. Відсутність чергового елемента є умовою завершення роботи циклу `for` (для порівняння: у циклі `while` умовою завершення служить результат `false` логічного вираження у заголовку).

Цикл `for` використовується для роботи зі словниками:

```
>>> d = {1: 'one', 2: 'two', 3: 'three', 4: 'four'}
>>> for key in d:
    d[key] = d[key] + '!'
>>> d
{1: 'one!', 2: 'two!', 3: 'three!', 4: 'four!'}
```

Варто запам'ятати, що цикл `for` Python не є аналогом циклів `for` в багатьох інших мовах програмування, так як не являє собою циклом з лічильником.

```
{'fox': 'лисиця', 'dog': 'пес', 'cat': 'кішка', 'elephant':
'слон', 'mouse': 'миша', 'bird': 'птиця'}
>>> dic['elephant'] = 'слон'
>>> del(dic['bird'])
>>> dic
{'fox': 'лисиця', 'dog': 'пес', 'cat': 'кішка', 'elephant':
'слон', 'mouse': 'миша'}
```

Введення та виведення даних

Введення даних здійснюється за допомогою функції `input` (список введення):

```
a = input()
print(a)
```

У дужках функції можна вказати повідомлення - коментар до даних, що вводяться:

```
a = input ("Введіть кількість: ")
```

Команда input() за замовчанням сприймає вхідні дані як рядок символів. Тому, щоб ввести ціле значення, слід вказати функцію int():

```
a = int (input())
```

Для введення дійсних чисел застосовується функція float()

```
a=float(input())
```

Виведення даних здійснюється за допомогою команди print (список виведення):

```
a = 1
b = 2
print(a)
print(a + b)
print('сума = ', a + b)
```

Існує можливість запису команд в один рядок, поділяючи їх через ;. Однак не слід часто використовувати такий спосіб, це знижує зручність читання:

```
a = 1; b = 2; print(a)
print (a + b)
print ('сума = ', a + b)
```

Математичні операції в Python

Цілі числа (int)

В Python підтримується набір найпростіших математичних операцій:

$x + y$	Додавання
$x - y$	Віднімання
$x * y$	Множення
x / y	Ділення

<code>x // y</code>	Отримання цілої частини від ділення
<code>x % y</code>	Залишок від ділення
<code>-x</code>	Зміна знаку числа
<code>abs(x)</code>	Модуль числа
<code>divmod(x, y)</code>	Пара (<code>x // y</code> , <code>x % y</code>)
<code>x ** y</code>	Зведення в ступінь
<code>pow(x, y[, z])</code>	<p><code>x</code> : Число, яке потрібно звести в ступінь.</p> <p><code>y</code> : Число, що є мірою, в яку потрібно звести перший аргумент. Якщо число негативне або одне з чисел "<code>x</code>" або "<code>y</code>" не цілі, то аргумент "<code>z</code>" не приймається.</p> <p><code>z</code> : Число, на яке потрібно зробити поділ за модулем. Якщо число вказано, очікується, що "<code>x</code>" та "<code>y</code>" позитивні та мають тип <code>int</code>.</p>

Бібліотека (модуль) `math`

У стандартне постачання Python входить бібліотека `math`, в якій міститься велика кількість математичних функцій, що часто використовуються.

Для роботи з даним модулем його потрібно імпортувати:

```
Import math
```

Найчастіше використовувані функції модуля `math`

<code>math.ceil(x)</code>	Повертає найближче ціле число більше, ніж <code>x</code>
<code>math.fabs(x)</code>	Повертає абсолютне значення числа <code>x</code>
<code>math.factorial(x)</code>	Обчислює факторіал <code>x</code>
<code>math.floor(x)</code>	Повертає найближче ціле число менше, ніж <code>x</code>
<code>math.exp(x)</code>	Обчислює <code>e**x</code>
<code>math.log2(x)</code>	Логарифм за основою 2
<code>math.log10(x)</code>	Логарифм за основою 10
<code>math.log(x[, base])</code>	За замовчуванням обчислює логарифм за основою <code>e</code> , Додатково можна вказати основу логарифма
<code>math.pow(x, y)</code>	Обчислює значення <code>x</code> у ступені <code>y</code>
<code>math.sqrt(x)</code>	Корінь квадратний від <code>x</code>

Тригонометричні функції модуля math

math.cos(x)	Повертає cos числа X
math.sin(x)	Повертає sin числа X
math.tan(x)	Повертає tan числа X
math.acos(x)	Повертає acos числа X
math.asin(x)	Повертає asin числа X
math.atan(x)	Повертає atan числа X

Константи:

- math.pi - число π .
- math.e - число e (експонента).

Регулярні вирази

Регулярні висловлювання, також називається regex, використовуються практично у всіх мовах програмування. У Python вони реалізовані в стандартному модулі re.

Шаблон регулярного виразу є спеціальною мовою, яка використовується для представлення загального тексту, цифр або символів, вилучення текстів, які відповідають цьому шаблону.

Регулярними виразами називаються шаблони, які використовуються для пошуку відповідного фрагмента тексту та зіставлення символів.

Для створення складних патернів регулярні вирази містять спеціальні символи/оператори.

Оператор "[]"

Будь-який із символів у квадратних дужках.

[abc] – знайде a, b і c.

[a-z] – знайде всі значення від a до z.

[a-z0-9A-Z] – знайде всі значення від a до z, від A до Z і від 0 до 9.

Оператор крапки

Оператор крапки (.) використовується для пошуку відповідності будь-якому одиничному символу, крім символу нового рядка.

Цей оператор можна використовувати у поєднанні з іншими.

Метапослідовності

Деякі патерни використовуються постійно. Для них є шорткати. Ось найчастіше використовувані:

`\w` — відповідність будь-якій літері, цифрі або підкресленню, еквівалентна `[a-zA-Z0-9_]`.

`\W` — відповідність будь-якому символу, крім буквеного та цифрового символу та символу підкреслення.

`\d` — відповідність будь-якому цифровому символу, еквівалентному `[0-9]`.

`\D` — відповідність будь-якому нецифровому символу.

Оператори “+” і “*”

Символ `+` використовується для 1 або більше значень лівого символу.

Символ `*` використовується для 0 або більше значень лівого символу.

Наприклад, якщо потрібно знайти всі підрядки, що починаються з “d” і закінчуються “e”, нам може зустрітися нуль або більше символів між “d” та “e”. Використовується: `d\w*e`.

Якщо потрібно знайти всі підрядки, що починаються з “d” і закінчуються на “e” з як мінімум одним символом між “d” та “e”, використовується: `d\w+e`.

Повторення:

`\w{n}` — повторює `\w` рівно `n` разів.

`\w{n,}` — повторює хоча б `n` разів або більше.

`\w{n1, n2}` - повторює `\w` хоча б `n1` разів, але не більше `n2` разів.

Оператори “^” і “\$”

Оператор “^” виділяє початок рядка, а “\$” виділяє кінець рядка.

Коди пошуку

Короткий список з коротким поясненням кожного коду:

\d відповідає цифрі

\D відповідає не цифрі

\s відповідає порожньому полю (пробіл)

\S відповідає заповненому полю

\w відповідає алфавітно-цифровому значенню

\W відповідає не алфавітно-цифровому значенню

В Python для роботи з регулярними виразами використовується модуль `re`.

Основні функції модуля `re`:

Функція `compile()` модуля `re`

Функція `compile()` модуля `re` компілює шаблон регулярного вираження `pattern` в об'єкт регулярного вираження, який може бути використаний для пошуку збігів.

Функція `search()` модуля `re`

Функція `search()` модуля `re` сканує рядок `string` у пошуках першого збігу з шаблоном `pattern` регулярного вираження та повертає відповідний об'єкт відповідності.

Функція `match()` модуля `re`

Функція `match()` модуля `re` повернути відповідний об'єкт зіставлення, якщо нуль або більше символів на початку рядка `string` відповідають шаблону регулярного виразу `pattern`.

Функція `fullmatch()` модуля `re`

Функція `fullmatch()` модуля `re` поверне об'єкт зіставлення, якщо весь рядок `string` відповідає шаблону регулярного вираження `pattern`.

Функція `finditer()` модуля `re`

Функція `finditer()` модуля `re` повертає ітератор об'єктів зіставлення по всіх збігах, що не перекриваються, для шаблону регулярного вираження в рядку.

Функція `split()` модуля `re`

Функція `split()` модуля `re` ділить рядок по появі шаблону регулярного виразу `pattern` і повертає список підстрок, що вийшли.

Функція `findall()` модуля `re`

Функція `findall()` модуля `re` повертає всі збіги шаблону `pattern` у рядку `string` у вигляді списку рядків. Рядок сканується зліва направо, і збіги повертаються у знайденому порядку.

Функція `sub()` модуля `re`

Функція `sub()` модуля `re` повертає рядок, отриманий шляхом заміни крайнього лівого входження шаблону регулярного вираження `pattern` у рядку `string` на рядок заміни `repl`. Якщо шаблон регулярного виразу не знайдено, рядок повертається без змін.

Функція `subn()` модуля `re`

Функція `subn()` модуля `re` виконує ту саму операцію, що й функція `sub()`, але повертає кортеж (`new_string`, `number_of_subs_made`)

Функція `escape()` модуля `re`

Функція `escape()` модуля `re` виконує екранування спеціальних символів у шаблоні. Це корисно, якщо потрібно зіставити довільний рядок літералу, який може містити метасимволи регулярних виразів.

Функція `purge()` модуля `re`

Функція `purge()` модуля `re` очищує кеш від регулярних виразів.

Виняток error() модуля re

Виняток error() модуля re виникає, коли рядок, переданий однією з функцій модуля, не є допустимим регулярним виразом, наприклад шаблон може містити невідповідні дужки або коли виникає будь-яка інша помилка під час компіляції шаблону або зіставлення з рядком.

Об'єкт регулярного вираження Pattern модуля re

Об'єкт регулярного виразу Pattern виходить у результаті компіляції шаблону регулярного виразу. Скомпільовані об'єкти регулярних виразів підтримують розглянуті нижче методи та атрибути.

Об'єкт зіставлення із шаблоном Match модуля re

Об'єкт зіставлення регулярного виразу зі рядком завжди має логічне значення True. Можна перевірити, чи був збіг за допомогою простого затвердження if...else. Об'єкти зіставлення підтримують методи та атрибути.

Модуль datetime

Модуль datetime надає класи для обробки часу та дати різними способами. Підтримується і стандартний спосіб уявлення часу, проте більший акцент робиться на простоті маніпулювання датою, часом та їх частинами.

Класи, що надаються модулем datetime:

Клас datetime.date(year, month, day) – стандартна дата. Атрибути: year, month, day. Незмінний об'єкт.

Клас datetime.time(hour=0, minute=0, second=0, microsecond=0, tzinfo=None) – стандартний час, не залежить від дати. Атрибути: hour, minute, second, microsecond, tzinfo.

Клас `datetime.timedelta` – різниця між двома моментами часу, з точністю до мікросекунд.

Клас `datetime.tzinfo` - абстрактний базовий клас для інформації про тимчасову зону (наприклад, для обліку часового поясу та/або літнього часу).

Клас `datetime.datetime(year, month, day, hour=0, minute=0, second=0, microsecond=0, tzinfo=None)` - комбінація дати та часу.

Обов'язкові аргументи:

- `datetime.MINYEAR (1) ≤ year ≤ datetime.MAXYEAR (9999)`
- `1 ≤ month ≤ 12`
- `1 ≤ day ≤ кількість днів у цьому місяці та році`

Необов'язкові:

- `0 ≤ minute < 60`
- `0 ≤ second < 60`
- `0 ≤ microsecond < 1000000`

Методи класу `datetime`:

`datetime.today()` – об'єкт `datetime` з поточної дати та часу. Працює як і `datetime.now()` зі значенням `tz=None`.

`datetime.fromtimestamp(timestamp)` – дата із стандартного уявлення часу.

`datetime.fromordinal(ordinal)` – дата з числа, що становить кількість днів, що минули з 01.01.1970.

`datetime.now(tz=None)` – об'єкт `datetime` з поточної дати та часу.

`datetime.combine(date, time)` – об'єкт `datetime` з комбінації об'єктів `date` та `time`.

`datetime.strptime(date_string, format)` – перетворює рядок на `datetime` (так само, як і функція `strptime` з модуля `time`).

`datetime.date()` – об'єкт дати (з відсіканням часу).

`datetime.time()` – об'єкт часу (з відсіканням дати).

`datetime.replace([year[, month[, day[, hour[, minute[, second[, microsecond[, tzinfo]]]]]]])` – повертає новий об'єкт `datetime` зі зміненими атрибутами.

`datetime.timetuple()` – повертає `struct_time` із `datetime`.

`datetime.toordinal()` – кількість днів, що минули з 01.01.1970.

`datetime.timestamp()` – повертає час у секундах з початку епохи.

`datetime.weekday()` – день тижня у вигляді числа, понеділок – 0, неділя – 6.

`datetime.isoweekday()` – день тижня у вигляді числа, понеділок – 1, неділя – 7.

`datetime.isocalendar()` – кортеж (рік у форматі ISO, ISO номер тижня, ISO день тижня).

`datetime.isoformat(sep='T')` – гарний рядок виду "YYYY-MM-DDTHH:MM:SS.mmmmmm" або, якщо `microsecond == 0`, "YYYY-MM-DDTHH:MM:SS".

Модуль sys

Модуль `sys` забезпечує доступ до деяких змінних та функцій, що взаємодіють з інтерпретатором `python`.

`sys.argv` – список аргументів командного рядка, що передаються сценарієм `Python`. `sys.argv[0]` є ім'ям скрипта (порожнім рядком в інтерактивній оболонці).

`sys.byteorder` – порядок байтів. Матиме значення 'big' при порядку проходження бітів від старшого до молодшого, і 'little', якщо навпаки (молодший байт перший).

`sys.builtin_module_names` – кортеж рядків, що містить імена всіх доступних модулів.

`sys.call_tracing`(функція, аргументи) - викликає функцію з аргументами та включеним трасуванням, у той час як трасування вклучене.

`sys.copyright` – рядок, що містить авторські права, що стосуються інтерпретатора Python.

`sys.clear_type_cache()` - очищає внутрішній кеш типу.

`sys.current_frames()` - повертає словник-відображення ідентифікатора для кожного потоку у верхньому кадрі стеку в даний час у цьому потоці на момент виклику функції.

`sys.dllhandle` – ціле число, що визначає дескриптор DLL Python (Windows).

`sys.exc_info()` – повертає кортеж із трьох значень, які дають інформацію про винятки, що обробляються на даний момент.

`sys.exec_prefix` – каталог установки Python.

`sys.executable` – шлях до інтерпретатора Python.

`sys.exit([arg])` – вихід із Python. Порушує виняток `SystemExit`, який може бути перехоплений.

`sys.flags` – прапори командного рядка. Атрибути лише для читання.

`sys.float_info` – інформація про тип даних `float`.

`sys.float_repr_style` – інформація про застосування вбудованої функції `repr()` типу `float`.

`sys.getdefaultencoding()` – повертає використовуване кодування.

`sys.getdlopenflags()` – значення прапорів для викликів `dlopen()`.

`sys.getfilesystemencoding()` – повертає кодування файлової системи.

`sys.getrefcount(object)` – повертає кількість посилань на об'єкт. Аргумент функції `getrefcount` - ще одне посилання на об'єкт.

`sys.getrecursionlimit()` – повертає ліміт рекурсії.

`sys.getsizeof(object[, default])` – повертає розмір об'єкта (у байтах).

`sys.getswitchinterval()` - інтервал перемикання потоків.

`sys.getwindowsversion()` – повертає кортеж, що описує версію Windows.

`sys.hash_info` – інформація про параметри хешування.

`sys.hexversion` – версія python як шістнадцяткове число (для 3.2.2 final це буде 30202f0).

`sys.implementation` - об'єкт, що містить інформацію про запущений python інтерпретатор.

`sys.int_info` – інформація про тип `int`.

`sys.intern(рядок)` - повертає інтернований рядок.

`sys.last_type`, `sys.last_value`, `sys.last_traceback` - інформація про оброблювані винятки. За змістом нагадує `sys.exc_info()`.

`sys.maxsize` – максимальне значення числа типу `Py_ssize_t` (231 на 32-бітних та 263 на 64-бітних платформах).

`sys.maxunicode` – максимальна кількість біт для зберігання символу Unicode.

`sys.modules` – словник імен завантажених модулів. Змінюємо, тому можна потішитися :)

`sys.path` - список шляхів пошуку модулів.

`sys.path_importer_cache` – словник-кеш для пошуку об'єктів.

`sys.platform` – інформація про операційну систему.

`sys.prefix` – папка установки інтерпретатора python.

`sys.ps1`, `sys.ps2` - первинне та вторинне запрошення інтерпретатора (визначено лише якщо інтерпретатор перебуває в інтерактивному режимі). За замовчуванням `sys.ps1 == ">>>"`, а `sys.ps2 == "..."`.

`sys.dont_write_bytecode` - якщо true, python не буде писати .рус файли.

`sys.setdlopenflags(flags)` – встановити значення прапорів для викликів `dlopen()`.

`sys.setrecursionlimit(межа)` – встановити максимальну глибину рекурсії.

`sys.setswitchinterval(інтервал)` - Встановити інтервал перемикання потоків.

`sys.settrace(tracefunc)` – встановити "слід" функції.

`sys.stdin` – стандартне введення.

`sys.stdout` – стандартний висновок.

`sys.stderr` – стандартний потік помилок.

`sys.__stdin__`, `sys.__stdout__`, `sys.__stderr__` - вихідні значення потоків введення, виведення та помилок.

`sys.tracebacklimit` – максимальна кількість рівнів відстеження.

`sys.version` – версія python.

`sys.api_version` – версія C API.

`sys.version_info` – Кортеж, що містить п'ять компонентів номера версії.

`sys.warnoptions` – реалізація попереджень.

`sys.winver` – номер версії python, що використовується для формування реєстру Windows.

Модуль NumPy

NumPy – це розширення мови Python, що додає підтримку великих багатовимірних масивів та матриць, разом із великою бібліотекою високорівневих математичних функцій для операцій із цими масивами.

Основним об'єктом NumPy є однорідний багатовимірний масив (`numpy` називається `numpy.ndarray`). Це багатовимірний масив елементів (зазвичай чисел), одного типу.

Найважливіші атрибути об'єктів `ndarray`:

`ndarray.ndim` - число вимірів (частіше їх називають "осі") масиву.

`ndarray.shape` – розміри масиву, його форма. Це кортеж натуральних чисел, що показує довжину масиву кожної осі. Для матриці з `n` рядків та `m` стовпів, `shape` буде `(n,m)`. Число елементів кортежу `shape` дорівнює `ndim`.

`ndarray.size` – кількість елементів масиву. Вочевидь, дорівнює добутку всіх елементів атрибута `shape`.

`ndarray.dtype` – об'єкт, що описує тип елементів масиву. Можна визначити `dtype` за допомогою стандартних типів даних Python. NumPy тут надає цілий букет можливостей як вбудованих, наприклад: `bool_`, `character`, `int8`, `int16`, `int32`, `int64`, `float8`, `float16`, `float32`, `float64`, `complex64`, `object_`, так і можливість визначити власні типи даних, у тому числі і склад.

`ndarray.itemsize` – розмір кожного елемента масиву в байтах.

`ndarray.data` – буфер, що містить фактичні елементи масиву. Зазвичай не потрібно використовувати цей атрибут, тому що звертатися до елементів масиву найпростіше за допомогою індексів.

У NumPy існує багато способів створення масиву. Один з найпростіших - створити масив зі звичайних списків або кортежів Python, використовуючи функцію `numpy.array()` (`array` - функція, що створює об'єкт типу `ndarray`):

```
>>> import numpy as np
>>> a = np.array([1, 2, 3])
>>> a
array([1, 2, 3])
>>> type(a)
<class 'numpy.ndarray'>
```

Функція `array()` трансформує вкладені послідовності багатомірні масиви. Тип елементів масиву залежить від типу елементів вихідної послідовності (можна перевизначити у момент створення).

```
>>> b = np.array([[1.5, 2, 3], [4, 5, 6]])
>>> b
array([[ 1.5,  2. ,  3. ],
       [ 4. ,  5. ,  6. ]])
```

Можна також перевизначити тип у момент створення:

```
>>> b = np.array([[1.5, 2, 3], [4, 5, 6]], dtype=np.complex)
>>> b
array([[ 1.5+0.j,  2.0+0.j,  3.0+0.j],
       [ 4.0+0.j,  5.0+0.j,  6.0+0.j]])
```

Для створення масивів із випадковими елементами використовується, зокрема, модуль `numpy.random`.

Найпростіший спосіб задати масив з випадковими елементами – використовувати функцію `sample` (або `random`, або `random_sample`, або `rand` – це та сама функція).

```
>>> np.random.sample()
0.6336371838734877

>>> np.random.sample(3)
array([ 0.53478558,  0.1441317 ,  0.15711313])

>>> np.random.sample((2, 3))
array([[ 0.12915769,  0.09448946,  0.58778985],
       [ 0.45488207,  0.19335243,  0.22129977]])
```

Без аргументів повертає просто число у проміжку `[0, 1)`, з одним цілим числом – одномірний масив, з кортежем – масив із розмірами, зазначеними у кортежі (всі числа – з проміжку `[0, 1)`).

За допомогою функції `randint` або `random_integers` можна створити масив цілих чисел. Аргументи: `low`, `high`, `size`: від якого, до якого числа (`randint` не включає це число, а `random_integers` включає), і `size` - розміри масиву.>>>
`np.random.randint(0, 3, 10)`

```
array([0, 2, 0, 1, 1, 0, 2, 2, 2, 0])
>>> np.random.random_integers(0, 3, 10)
array([2, 2, 3, 3, 1, 1, 0, 2, 3, 2])
>>> np.random.randint(0, 3, (2, 10))
array([[0, 1, 2, 0, 0, 0, 1, 1, 1, 2],
       [0, 0, 2, 2, 2, 0, 1, 2, 2, 1]])
```

Модуль *string*

Модуль `string` містить різні інструменти для роботи зі рядками, багато з яких дублюють можливості стандартного типу `str`. Модуль `string` також містить набір констант:

```
string.ascii_lowercase # 'abcdefghijklmnopqrstuvwxyz'
string.ascii_uppercase # 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
string.ascii_letters
# 'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ'
string.digits          # '0123456789'
string.punctuation    # '!"#$%&'()*+,-./:;<=>?@[\\]^_`{|}~'
```

У модулі **string** **перед інших** визначена функція `capwords()`, яка переводить у верхній регістр першу букву кожного слова в рядку.

```
import string
s = 'It was nice talking to you! Thank you!'
print(s)
print(string.capwords(s))
```

Результат роботи функції буде таким:

```
It was nice talking to you! Thank you!
It Was Nice Talking To You! Thank You!
```

У модулі **string** визначено ряд констант, що мають відношення до таблиці ASCII і символьних послідовностей.

Модуль `string` містить кілька типів інструментів, таких як функції, методи та класи. Він також містить найзагальніші строкові константи.

Функції

`index (s, sub, i=0)`

`lower (s)`

`splitfields (s, sep)`

`joinfields (<слова>,
<розподільник>)`

`strip (s)`

`upper (s)`

Результат

повертає індекс першого входження підрядка sub у рядок s, починаючи з позиції i.

повертає рядок s у нижньому регістрі літер.

повертає список підрядків рядків s розділених символом sep.

зчіплює список <слова> використовуючи <розділювач>.

повертає рядок, отриманий з s шляхом виключення пропусків.

повертає рядок s у верхньому регістрі літер.

Matplotlib: Наукова графіка в Python

Matplotlib це бібліотека для Python, призначена для візуалізації даних, для побудови графіків функцій та рівнянь. Matplotlib використовується в наукових дослідженнях та конференціях для демонстрації отриманих даних.

Для побудови графіків необхідно імпортувати модуль Pyplot. Pyplot це модуль для роботи з графіками у Python.

Встановлення:

```
import numpy as np
import matplotlib.pyplot as plt
```

Для побудови графіка функції Python потрібно задати саму функцію. Її можна поставити за допомогою лямбда-функції. Лямбда-функція – це короткий спосіб запису звичайної функції в один рядок.

Розглянемо побудову синусоїди на Python $f(x) = \sin(x)$:

```
y = lambda x: np.sin(x)
```

y – це позначення функції, lambda - це ключове слово, що означає початок завдання лямбда-функції, x це – аргумент, що використовується у функції, після двокрапки задається функція. Так як у стандартному Python немає функції, що повертає синус x, вона задається м за допомогою NumPy.

Для побудови графіка функції Python потрібно спочатку задати сітку координат за допомогою команди `plt.subplots()`.

```
fig = plt.subplots()
```

Визначення області значень, на якій будується графік функції в Пітоні, використовується `linspace`:

```
x = np.linspace(-3, 3, 100)
```

У даному прикладі `linspace` створює масив з нижньою межею -3 і верхньою межею 3, у створеному масиві буде 100 елементів. Чим більше буде останнє число, тим більше значень функції буде розраховуватись, тим точніше відобразатиметься графік у Python.

Для безпосередньої побудови графіка функції Python потрібно використовувати команду `plt.plot(x, y(x))`, де x – це аргумент, $y(x)$ – це функція від x , що задається за допомогою лямбда-виразу. Після того, як графік побудований, його потрібно відобразити за допомогою команди `plt.show()`.

Повний код програми на Python для відображення графіка функції:

```
import numpy as np
import matplotlib.pyplot as plt

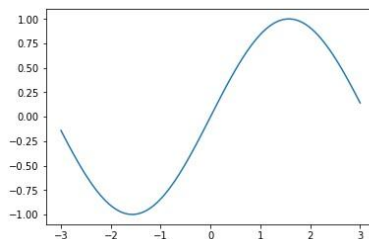
y = lambda x: np.sin(x)

fig = plt.subplots()

x = np.linspace(-3, 3, 100)

plt.plot(x, y(x))
plt.show()
```

В результаті в окремому вікні формується графік синусоїди.



Відображення декількох графіків у одній області в Python

В одній області Python можна відобразити графіки декількох функцій.

Додамо `aeugwb`. $y=x$ і відобразимо її разом із синусоїдою.

Для цього введемо ще одну функцію за допомогою `lambda`.

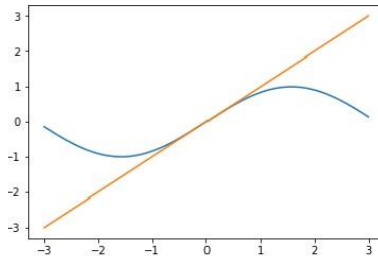
```
y1=lambda x: x
```

Побудуємо графік цієї функції

```
plt.plot(x,y1(x))
```

Програма побудови графіків двох функцій в одному вікні:

```
import numpy as np
import matplotlib.pyplot as plt
y = lambda x: np.sin(x)
y1=lambda x: x
fig = plt.subplots()
x = np.linspace(-3, 3,100)
plt.plot(x, y(x))
plt.plot(x,y1(x))
plt.show()
```



Тривимірні поверхні в Python

У тривимірному просторі потрібні два аргументи для завдання функції.

Задамо, наприклад, функцію від двох аргументів:

$$f(x,y)=x^2 - y^2.$$

Відповідно:

```
f = lambda x, y: x ** 2 - y ** 2
```

Спочатку визначається область побудови за допомогою функції `plt.figure`, в якій параметр `figsize(x, y)` визначає ширину і висоту малюнка в дюймах.

Визначення області:

```
fig = plt.figure(figsize = (12, 6))
```

Створимо малюнок, в якому буде відображено тривимірний простір з координатними осями та сама поверхня. Для цього використовується `fig.add_subplot()`.

```
ax = fig.add_subplot(1, 1, 1, projection = '3d')
```

Функція Python `fig.add_subplot()` розбиває область побудови на клітини і задає в якій клітині малювати тривимірний графік.

Команда `ax = fig.add_subplot(1, 1, 1, projection = '3d')` розбиває область побудови на дві клітини і в першій клітині відображатиметься тривимірний графік завдяки аргументу `projection = '3d'`.

Введемо області відображення функції для кожного аргументу:

```
xval = np.linspace(-5, 5, 100)
```

```
yval = np.linspace(-5, 5, 100)
```

Для створення поверхні використовується:

```
surf = ax.plot_surface(x, y, z, rstride = 4, cstride = 4, cmap =  
cm.plasma)
```

Тут `x` та `y` – аргументи, а `z` це цільова функція, `rstride` та `cstride` відповідає за крок промальовування. Чим меншими будуть ці значення, тим плавніше виглядатиме градієнт на поверхні. Завдяки величині `cmap.plasma` поверхня буде відображатися в колірній схемі `plasma`. Існують і інші колірні схеми, такі як `viridis` та `magma`.

Приклад програми побудови поверхні у тривимірному просторі:

```
from mpl_toolkits.mplot3d import Axes3D  
import numpy as np  
from matplotlib import cm  
import matplotlib.pyplot as plt  
  
f = lambda x, y: x ** 2 - y ** 2
```

```

fig = plt.figure(figsize = (10, 10))

ax = fig.add_subplot(1, 1, 1, projection = '3d')

xval = np.linspace(-4, 4, 100)
yval = np.linspace(-4, 4, 100)

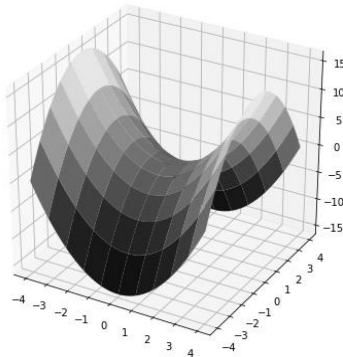
x, y = np.meshgrid(xval, yval)

z = f(x, y)

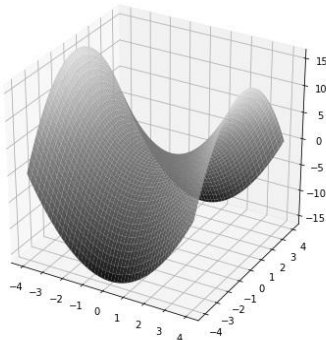
surf = ax.plot_surface(x, y, z, rstride = 10, cstride = 10, cmap =
cm.plasma)

```

Отримуємо графік тривимірної поверхні у колірній гамі plasma у спеціальному вікні.



При зміні параметрів $rstride = 2$, $cstride = 2$, $cmap = cm.viridis$, формується графік тривимірної поверхні в Python більш точний і в іншій колірній гамі.



PyWavelets : вейвлет-перетворення в Python

PyWavelets — це програмне забезпечення для вейвлет-перетворення з відкритим вихідним кодом для Python. Основна ідея вейвлет-перетворення відповідає специфіці багатьох часових рядів, що демонструють еволюцію у часі своїх основних характеристик – середнього значення, дисперсії, періодів, амплітуд та фаз гармонійних компонентів. Переважна більшість процесів, що вивчаються у різних галузях знань, мають перелічені вище особливості.

Безперервне вейвлет-перетворення (CWT) здійснюється викликом функції:

```
pywt.cwt(data, scales, wavelet)
```

Параметри:

data : вхідний сигнал;

scales : шкала вейвлета;

wavelet : найменування вейвлету

Предметный показчик

Apache Hadoop 86, 88, 89
Apache Spark 88
API 10, 27, 35, 47, 123
Atom 12
Big Data 8, 9, 11
CGI (Common Gateway Interface) 40
CDH (Cloudera Hadoop) 89
Cloudera 88, 89
Cloudera Manager 88,89
CRUD 35, 119, 120
CSV 57, 58, 79, 83, 114, 115
cURL 15, 16, 31, 32
Cypher 129, 132
Data Science 9, 10
DFS (Distributed file system) 87, 90
Elastic 26, 30
Elastic Stack 9, 10, 26, 27
Elasticsearch 21, 26, 27, 30, 37
Excel 58
Gephi 78, 80, 81
GUI 117, 118
Hadoop 78, 80, 81
Hadoop Common 87
Hadoop YARN 87
HDFS (Hadoop Distributed Filesystem) 87
JSON (JavaScript Object Notation JavaScript) 19
Jupyter Lab 61, 62
Kibana 26, 45
Kibana Console 47
Kibana Discover 49
Kibana Visualizations 51
Mapper 98, 100
MapReduce 86, 87, 88
MHAT Wavelet 68
MongoDB 110, 117
MongoDB Compass 117, 118
Morle Wavelet 68
Neo4j 123, 124, 127
Neo4j Browser 123, 127, 132
NoSQL 111
OSINT (Open Source Intelligence) 8
Python 23, 37, 135
RDF (Resource Description Framework) 12
Reducer 98
REST API 47
RSS 11, 12
Wavelet 66, 67, 68, 69
WAVE Wavelet 68
Wget 16
WinPython 61
WordCount 97, 100, 103
XML 11, 12
XGET 33, 55
X-Pack 26, 45
XPOST 31, 32
YAML Ain't Markup Language 29

Навчальне видання

Ланде Дмитро Володимирович

Субач Ігор Юрійович

Гладун Анатолій Ясонович

**ОБРОБЛЕННЯ НАДВЕЛИКИХ
МАСИВІВ ДАНИХ (BIG DATA)**

Навчальний посібник

В авторській редакції

Інститут спеціального зв'язку та захисту інформації
Національного технічного університету України
«Київський політехнічний інститут імені Ігоря Сікорського»

Підп. до друку 30.12.2021. Формат 60х84¹/16. Папір офс. Гарнітура Times.
Спосіб друку – ризографія. Ум. друк. арк. 10,5. Обл.-вид. арк. 24,36. Наклад 100 пр.
Зам. № 15-200.

ТОВ "Інжиніринг" ISBN 978-966-2344-83-7



ISBN 978-966-2344-83-7



9 789662 344837

ТОВ «Інжиніринг»