ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: **«ДОСЛІДЖЕННЯ АРХІТЕКТУР ПОБУДОВИ ТА АЛГОРИТМІВ РОБОТИ СИСТЕМ ЗБЕРІГАННЯ ТА АНАЛІЗУ ВЕЛИКИХ ДАНИХ З МЕТОЮ ЇХ РЕАЛІЗАЦІЇ»**

на здобуття освітнього ступеня магістр
за спеціальності 123 Комп'ютерна інженерія
(код,найменування спеціальності)
освітньо-професійної програми Комп'ютерні системи та мережі
(назва)

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Євген ДЕГТЯРЬОВ
(підпис) *(ім'я, ПРІЗВИЩЕ здобувача)*

Виконав: здобувач вищої освіти гр.КСДМ-61
Євген ДЕГТЯРЬОВ
(ім'я, ПРІЗВИЩЕ)
Керівник: Андрій ЛЕМЕШКО
Доктор філософії, (PhD) *(ім'я, ПРІЗВИЩЕ)*
Рецензент:
(ім'я, ПРІЗВИЩЕ)

Київ 2023

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Комп'ютерної інженерії
Ступінь вищої освіти «Магістр»
Спеціальність 123 Комп'ютерна інженерія
Освітньо-професійна прорама Комп'ютерні системи та мережі

ЗАТВЕРДЖУЮ
Завідувач кафедрою Комп'ютерної інженерії
Наталія ЛАЩЕВСЬКА
(ім'я, ПРІЗВИЩЕ)
"____" _____2023 року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Дегтярьову Євгену Олександровичу
(прізвище, ім'я, по батькові здобувача)

1.Тема кваліфікаційної роботи: Дослідження архітектур побудови та алгоритмів роботи систем зберігання та аналізу великих даних з метою їх реалізації

керівник роботи Андрій ЛЕМЕШКО доктор філософії (PhD)
(ім'я, ПРІЗВИЩЕ, науковий ступінь, вчене звання)

затверджені наказом Державного університету інформаційно-комунікаційних технологій від "19" 10 2023 р. №145

2. Строк подання кваліфікаційної роботи 22.12.2023р.

3. Вихідні дані кваліфікаційної роботи:
3.1. Архітектурні рішення Big Data.
3.2. Систем зберігання та аналізу великих даних.
3.3. Алгоритми систем зберігання та аналізу Big Data.
3.4. Науково-технічна література.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити):
4.1. Аналіз особливостей та перспективи застосування технології Big Data.
4.2. Дослідження архітектурних та програмних рішень при побудові систем зберігання та аналізу великих даних.
4.3. Створення системи зберігання та аналізу великих даних.

5.Перелік ілюстраційного матеріалу: *презентація*

6. Дата видачі завдання " 20 " жовтня 2023р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Підбір технічної літератури	20.10.2023р. 26.10.2023р.	Виконано
2.	Аналіз загальних відомостей та понять Big Data	26.10.2023р. 07.11.2023р.	Виконано
3.	Дослідження архітектурних рішень при побудові систем зберігання та аналізу Big Data	07.11.2023р. 17.11.2023р.	Виконано
4.	Дослідження програмних рішень при побудові систем зберігання та аналізу Big Data	17.11.2023р. 25.11.2023р.	Виконано
5.	Практична реалізація процесу зберігання та аналізу Big Data	25.11.2023р. 04.12.2023р.	Виконано
6.	Розробка демонстраційних матеріалів, доповідь.	04.12.2023р. 16.12.2023р.	Виконано
7.	Оформлення магістерської роботи	16.12.2023р. 20.12.2023р.	Виконано

Здобувач вищої освіти _____ Євген ДЕГТЯРЬОВ
(підпис) *(ім'я, ПРІЗВИЩЕ)*

Керівник кваліфікаційної роботи _____ Андрій ЛЕМЕШКО
(підпис) *(ім'я, ПРІЗВИЩЕ)*

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття ступня магістр: 74 стор., 44 рис., 4 табл., 30 джерел.

Мета роботи - дослідження архітектур побудови, алгоритмів роботи систем зберігання та аналізу великих даних.

Об'єкт дослідження - процес зберігання та аналізу великих даних.

Предмет дослідження – системи зберігання та аналізу великих даних.

Короткий зміст роботи: У першому розділі досліджено особливості технології Big Data, виконано огляд сфер застосування та прогноз розвитку світового ринку великих даних.

Проаналізовано актуальність застосування технологій зберігання та аналізу великих даних. Описано і проаналізовано особливі властивості великих даних.

Виконано аналіз архітектурних та програмних рішень при побудові систем зберігання та аналізу великих даних. Зроблено огляд та аналіз застосування NoSQL баз даних.

У практичній частині, магістерської роботи, досліджено використання стандартних та сторонніх бібліотек мови Java при реалізації етапів доступу, очищення та завантаження до систем зберігання та аналізу великих даних.

КЛЮЧОВІ СЛОВА: РЕСУРС, АРХІТЕКТУРА, ТЕХНОЛОГІЯ, ПЛАТФОРМА, ДОДАТКИ, ІНФРАСТРУКТУРА, ТОПОЛОГІЯ, ЗБІР ДАНИХ, СТРАТЕГІЯ, МОНІТОРИНГ, BIG DATA.

ABSTRACT

The text part of the qualification work for obtaining a master's degree: 74 pages, xx figures, xx tables, 30 sources.

The purpose of the work is the study of construction architectures, algorithms of operation of storage systems and analysis of big data.

The object of research is the process of storing and analyzing Big Data.

The subject of research is systems of storage and analysis of Big Data.

Brief content of the work: In the first chapter, the features of Big Data technology were investigated, an overview of the areas of application and a forecast of the development of the world market of Big Data were performed.

The relevance of the application of big data storage and analysis technologies has been analyzed. The special properties of big data are described and analyzed.

The analysis of architectural and software solutions in the construction of storage systems and analysis of big data was performed. An overview and analysis of the application of NoSQL databases was made.

In the practical part of the master's thesis, the use of standard and third-party Java language libraries was investigated in the implementation of the stages of access, cleaning and uploading to big data storage and analysis systems.

KEYWORDS: RESOURCE, ARCHITECTURE, TECHNOLOGY, PLATFORM, APPLICATIONS, INFRASTRUCTURE, TOPOLOGY, DATA COLLECTION, STRATEGY, MONITORING, BIG DATA.

ЗМІСТ

ВСТУП

Актуальність теми. Великі дані виявилися однією з найбільш обговорюваних тем останнім часом, оскільки кожен аспект сучасного технологічного життя продовжує генерувати все більше даних.

Швидке зростання Інтернету та цифрової економіки сприяло експоненційному зростанню попиту на зберігання та аналітику даних, і IT-відділ стикається з величезною проблемою щодо захисту та аналізу цих збільшених обсягів інформації. Причина, чому організації збирають і зберігають більше даних, ніж будь-коли раніше, полягає в тому, що від цього залежить їхній бізнес. Тип інформації, що створюється, — це не традиційні дані, керовані базами даних, які називаються структурованими даними, а дані, які включають документи, зображення, аудіо, відео та вміст соціальних мереж, відомий як неструктуровані дані або великі дані. Big Data Analytics — це спосіб отримання цінності з цих величезних обсягів інформації, що відкриває нові ринкові можливості та максимізує утримання клієнтів.

Таким чином, актуальність кваліфікаційної магістерської роботи зумовлена необхідністю дослідженню технологій та систем зберігання і аналізу великих даних.

Мета роботи - дослідження архітектур побудови, алгоритмів роботи систем зберігання та аналізу великих даних.

Для виконання поставленої мети, у магістрській роботі розроблено та виконано наступні завдання:

- аналіз особливостей та перспективи застосування технології Big Data;
- дослідження архітектурних та програмних рішень при побудові систем зберігання та аналізу великих даних;
- створення системи зберігання та аналізу великих даних.

Об'єкт дослідження - процес зберігання та аналізу великих даних.

Предмет дослідження – системи зберігання та аналізу великих даних.

Методи дослідження. Під час виконання завдань магістерської роботи були використані методи теорії ймовірності, методи інтелектуального аналізу, а також методи системного аналізу, експертної оцінки та методи об'єктно-орієнтованого аналізу та проектування.

Джерела дослідження:

- https://rb.ua/howto/chto-takoe-big-data;
- http://statsoft.com/products/Enterprise/big-data.php;
- https://postnauka.com/faq/46974;
- https://www.calltouch.ua/glossary/big-data;
- https://web-creator.ua/articles/bigdata.

Наукова новизна одержаних результатів. Наукова новизна магістерської роботи, полягає у розробці практичних рекомендацій щодо реалізації етапів доступу, очищення та завантаження даних у систему зберігання та аналізу Big Data.

Практична значущість одержаих результатів. Практична значимість дослідження полягає у можливості практичного застосування досліджуваних технологій зберігання та аналізу великих даних.

Апробація результатів магістерської роботи. Основні положення і результати магістерської роботи доповідались і обговорювались на двох науково-практичних конференціях.

Публікації. За матеріалами роботи опубліковано одну статтю у науковому журналі.

РОЗДІЛ 1 АНАЛІЗ ЗАГАЛЬНИХ ВІДОМОСТЕЙ ТА ПОНЯТЬ BIG DATA

1.1 Огляд особливостей технології Big Data

Великі дані користуються великим ажіотажем і не без причини. Але розуміння суті великих даних і способів їх аналізу все ще розмиті. Правда в тому, що в цьому терміні є щось більше, ніж просто кількість отриманої інформації. Великі дані застосовуються не лише до величезних обсягів даних, що постійно зростають, які надходять у різних форматах, але й до ряду процесів, інструментів і підходів, які використовуються для отримання інформації з цих даних. І це найголовніше: аналітика Big Data допомагає компаніям справлятися з бізнес-проблемами, які не вдалося вирішити за допомогою традиційних підходів та інструментів [1].

Великі дані – це термін, що описує великі набори різноманітних даних – структурованих, неструктурованих і напівструктурованих – які постійно генеруються з високою швидкістю та у великих обсягах. Зараз зростає кількість компаній, які використовують ці дані, щоб отримати значущу інформацію та покращити процес прийняття рішень, але вони не можуть зберігати та обробляти їх за допомогою традиційних пристроїв зберігання та обробки даних.

Великі дані містять велику кількість даних, які не обробляються традиційним сховищем даних або процесором. Його використовують багато міжнародних компаній для обробки даних і бізнесу багатьох організацій . Потік даних перевищував би 150 екзабайт на день до тиражування.

Великі дані по суті поділяються на три типи: структуровані дані**,** неструктуровані дані**,** напівструктуровані дані.

Наведені вище три типи великих даних технічно застосовні на всіх рівнях аналітики. Під час роботи з великими обсягами великих даних дуже важливо розуміти джерело необроблених даних і їх обробку перед аналізом. Оскільки даних дуже багато, вилучення інформації має здійснюватися ефективно, щоб отримати

максимальну віддачу від даних. Процес ETL для кожної структури даних відрізняється.

Структуровані дані. Структуровані дані добре впорядковані, тому з ними найлегше працювати. Його розміри визначаються заданими параметрами. Кожна інформація згрупована в рядки та стовпці, як електронні таблиці. Структуровані дані містять кількісні дані, такі як вік, контакт, адреса, рахунок, витрати, номери дебетових або кредитних карток тощо.

Завдяки кількісній природі структурованих даних програмам легко сортувати та збирати дані. Для обробки структурованих даних не потрібно майже ніякої підготовки. Дані потрібно лише очистити та скоротити до відповідних точок. Для виконання належного запиту дані не потрібно перетворювати або інтерпретувати надто глибоко [2].

Структуровані дані слідують за дорожніми картами до конкретних точок даних або схем для окреслення розташування кожного даного та його значення.

Спрощений процес об’єднання корпоративних даних із реляційними є однією з переваг структурованих даних. Через те, що відповідні розміри даних визначені та знаходяться в єдиному форматі, потрібна дуже незначна підготовка, щоб усі джерела були сумісними.

Процес ETL для структурованих даних зберігає готовий продукт у сховищі даних . Початкові дані збираються для певних цілей аналітики, і для цього бази даних добре структуровані та відфільтровані. Однак існує лише обмежена кількість доступних структурованих даних, і вони підпадають під невелику меншість усіх існуючих даних. Консенсус говорить, що структуровані дані складають лише 20 відсотків або менше всіх даних.

Неструктуровані дані. Не всі дані структуровані та добре відсортовані з інструкціями щодо їх використання. Усі неорганізовані дані називаються неструктурованими.

Майже все, що створюється комп’ютером, є неструктурованими даними. Час і зусилля, необхідні для того, щоб зробити неструктуровані дані читабельними, можуть бути громіздкими. Щоб отримати реальну цінність даних, набори даних

мають бути інтерпретованими. Але процес досягнення цього може бути набагато більш корисним.

Складна частина аналізу неструктурованих даних полягає в тому, щоб навчити програму розуміти інформацію, яку вона витягує. Часто потрібен переклад у структуровану форму, що нелегко і залежить від різних форматів і кінцевих цілей. Деякі методи досягнення перекладу полягають у використанні синтаксичного аналізу тексту, NLP та розробці ієрархії вмісту за допомогою таксономії. Для поєднання процесів сканування, інтерпретації та контекстуалізації використовуються складні алгоритми.

На відміну від структурованих даних, які зберігаються в сховищах даних , неструктуровані дані зберігаються в озерах даних. Озера даних зберігають необроблений формат даних, а також всю їхню інформацію. Озера даних роблять дані більш гнучкими, на відміну від сховищ даних, де дані обмежені визначеною схемою.

Напівструктуровані дані. Напівструктуровані дані є чимось середнім між структурованими та неструктурованими даними. Здебільшого це перекладається на неструктуровані дані, до яких прикріплено метадані. Можна успадкувати напівструктуровані дані, наприклад місцезнаходження, час, адресу електронної пошти або штамп ідентифікатора пристрою. Це може бути навіть семантичний тег, доданий до даних пізніше [3].

Розглянемо приклад електронного листа. Час надсилання електронного листа, адреси електронної пошти відправника та одержувача, IP-адреса пристрою, з якого було надіслано електронний лист, та інша відповідна інформація пов’язані зі змістом електронного листа. Хоча фактичний вміст сам по собі не структурований, ці компоненти дозволяють групувати дані структурованим чином.

Використання правильних наборів даних може зробити напівструктуровані дані значним активом. Він може допомогти машинному навчанню та навчанню ШІ, зв’язуючи шаблони з метаданими.

Напівструктуровані дані без встановленої схеми можуть бути як перевагою, так і проблемою. Докласти всіх зусиль, щоб повідомити програмі значення кожної точки даних, може бути складно. Але в той же час, ETL структурованих даних не має обмежень щодо визначення.

Підтипи даних. Окрім трьох вищезгаданих типів, існують підтипи даних, які офіційно не вважаються великими даними, але певною мірою стосуються аналітики. У більшості випадків це джерело таких даних, як соціальні мережі, машинне (оперативне журналювання), викликане подіями або геопросторове. Він також може включати рівні доступу: відкритий (з відкритим вихідним кодом), зв'язаний (веб-дані, що передаються через API та інші методи з'єднання), або темний або втрачений (розташований у системах для недоступності сторонніх осіб, таких як системи відеоспостереження).

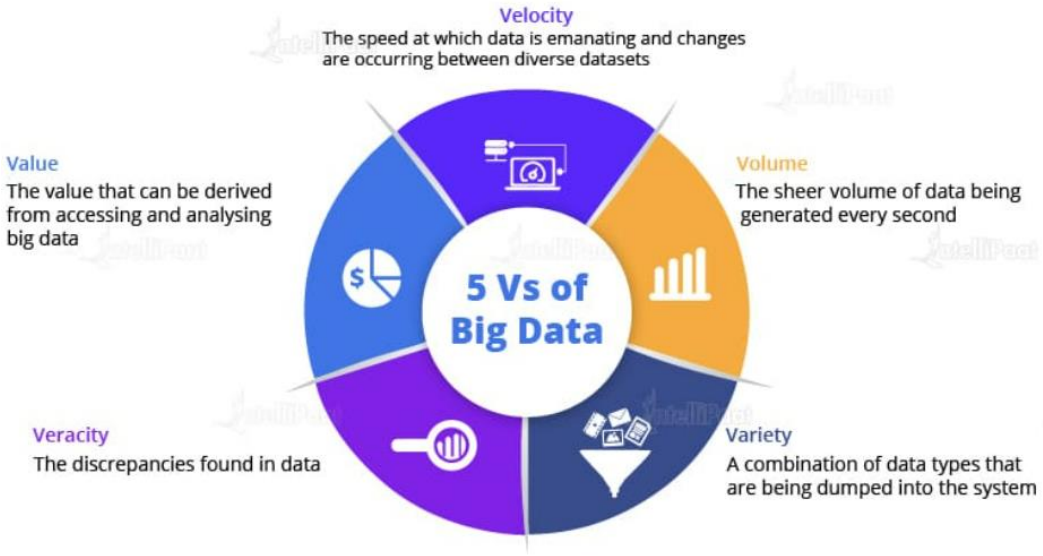


Рисунок 1.1- Складові Big Data, які пояснюють характеристики

1.Обсяг. Сама назва Big Data пов'язана з величезним розміром. Великі дані — це величезні «об'єми» даних, які щодня генеруються з багатьох джерел, наприклад бізнес-процесів, машин, платформ соціальних медіа, мереж, взаємодії людей тощо [4].

Facebook може генерувати приблизно мільярд повідомлень, 4,5 мільярда разів записує кнопку « Подобається », і щодня завантажується понад 350 мільйонів нових публікацій. Технології великих даних можуть обробляти великі обсяги даних.

2. Різноманітність. Причиною такого швидкого зростання обсягу даних є те, що дані надходять із різних джерел у різних форматах. Ми вже обговорювали, як дані класифікуються на різні типи. Давайте ще раз поглянемо на це з іншими прикладами.

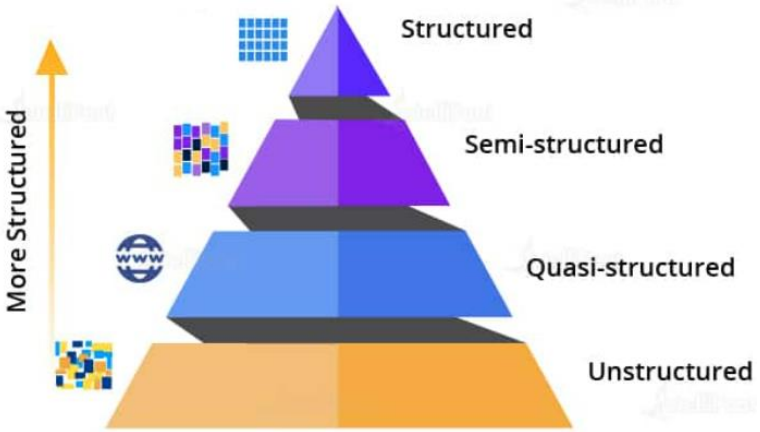


Рисунок 1.2 - Класифікація даних

а). Структуровані дані: тут дані присутні у структурованій схемі разом із усіма необхідними стовпцями. Він має структурований або табличний формат. Дані, які зберігаються в системі керування реляційною базою даних, є прикладом структурованих даних. Наприклад, у наведеній нижче таблиці співробітників, яка присутня в базі даних, дані представлені в структурованому форматі [5].

б). Напівструктуровані дані: у цій формі даних схема не визначена належним чином, тобто присутні обидві форми даних. Отже, напівструктуровані дані мають структуровану форму, але вона не визначена; наприклад, JSON, XML, CSV, TSV та електронна пошта. Неструктуровані дані веб-додатків містять файли історії транзакцій, файли журналів тощо. Системи онлайнової обробки транзакцій (OLTP)

створені для роботи зі структурованими даними, і ці дані зберігаються у зв'язках, тобто таблицях.

в). Неструктуровані дані: цей формат даних включає всі неструктуровані файли, такі як відеофайли, файли журналу, аудіофайли та файли зображень. Будь-які дані, які мають незнайому модель або структуру, класифікуються як неструктуровані дані. Оскільки розмір неструктурованих даних є великим, вони мають різні проблеми з обробкою для отримання з них цінності. Прикладом цього є складне джерело даних, яке містить суміш текстових файлів, відео та зображень. Кілька організацій мають у своєму розпорядженні багато даних, але вони не знають, як отримати з них цінність, оскільки дані знаходяться в необробленому вигляді.

г). Квазіструктуровані дані: цей формат даних складається з текстових даних із неузгодженими форматами даних, які можна відформатувати зусиль, часу та за допомогою кількох інструментів. Наприклад, журнали веб-сервера, тобто файл журналу, який автоматично створюється та підтримується сервером і містить список дій.

3. Швидкість: швидкість накопичення даних також відіграє важливу роль у визначенні того, чи є дані великими чи звичайними даними [6].

Як видно на зображенні нижче, мейнфрейми спочатку використовувалися, коли менше людей використовувало комп'ютери. У міру розвитку комп'ютерів з'явилася модель клієнт/сервер. Пізніше з'явилися веб-програми, і їхня популярність поширювалася на все більше пристроїв, таких як мобільні телефони, що призвело до створення великої кількості даних!

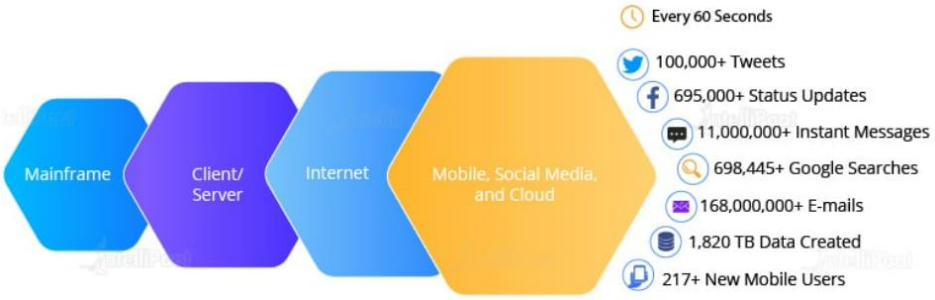


Рисунок 1.3 – Швидкість накопичення даних

4. Значення: як буде працювати вилучення даних? Тут з’являється наш четвертий V; він має справу з механізмом виявлення правильного значення даних. Перш за все, вам потрібно видобути дані, тобто процес перетворення необроблених даних у корисні дані. Потім виконується аналіз даних, які ви очистили або отримали з необроблених даних. Потім вам потрібно переконатися, що будь-який аналіз, який ви провели, принесе користь вашому бізнесу, як-от отримання інформації, результатів тощо, у спосіб, який раніше не був можливим [7].

Вам потрібно обов’язково очистити будь-які необроблені дані, які вам надають для отримання ділової інформації. Після того, як ви очистили дані, з’являється проблема, тобто під час скидання великої кількості даних деякі пакети можуть бути втрачені.

Отже, щоб вирішити цю проблему, з’являється наш наступний V.

5. Достовірність: оскільки пакети втрачаються під час виконання, нам потрібно почати знову зі стадії видобутку необроблених даних, щоб перетворити їх на цінні дані. І цей процес триває. У даних також будуть невизначеності та суперечності, які можна подолати правдивістю. Правдивість означає достовірність і якість даних. Необхідно підтримувати правдивість даних. Наприклад, подумайте про публікації у Facebook, хештеги, абревіатури, зображення, відео тощо, які роблять публікації ненадійними та погіршують якість їхнього вмісту. Збирати величезну кількість даних марно, якщо якість і достовірність даних не на належному рівні.

1.2 Огляд сфер застосування

Застосування великих даних надає рішення для всіх секторів, таких як банківська справа, уряд, освіта, охорона здоров’я тощо. Найбільш поширені наведені нижче:

1.Банківська справа. Оскільки існує величезна кількість даних, які надходять із незліченних джерел, банкам потрібно знайти незвичайні та нетрадиційні способи керування великими даними. Важливо також вивчати вимоги клієнтів, надавати

послуги відповідно до їхніх специфікацій і зменшувати ризики, дотримуючись нормативних вимог. Щоб вирішити цю проблему, фінансовим установам доводиться мати справу з Big Data Analytics. Як приклад щодня NYSE генерує близько одного терабайта нових торгових даних. Тож уявіть собі, якщо один терабайт даних генерується щодня, скільки даних потрібно було б обробити за цілий рік. Ось для чого використовуються великі дані [8].

Рисунок 1.4- Застосування Big Data у банківській справі

2. Уряд. Державні установи використовують великі дані та створили багато керуючих установ, які керують комунальними послугами, борються з пробками чи обмежують наслідки злочинності. Однак, окрім переваг великих даних, уряд також звертає увагу на питання прозорості та конфіденційності. Для прикладу, індійський уряд має рекорд із усіх 1,21 мільярда громадян. Ці величезні дані зберігаються та аналізуються, щоб дізнатися кілька речей, наприклад кількість молоді в країні. За якими зроблено кілька схем для максимального охоплення населення. Усі ці великі дані не можна зберігати в традиційній базі даних, тому їх залишають для зберігання та аналізу за допомогою кількох інструментів Big Data Analytics .

3.Освіта. Освіта щодо великих даних має життєво важливий вплив на учнів, шкільні системи та навчальні програми. Інтерпретуючи великі дані, люди можуть забезпечити розвиток учнів, виявити учнів групи ризику та створити імпровізовану систему для оцінювання та допомоги директорам і вчителям.

Сектор освіти містить багато інформації щодо навчальної програми, студентів і викладачів. Інформація аналізується, щоб отримати інформацію, яка може підвищити операційну адекватність освітньої організації. Збір і аналіз інформації про студента, як-от відвідуваність, тестові бали, оцінки та інші питання, займають багато даних. Таким чином, великі дані наближаються до прогресивної структури, в якій ці дані можна зберігати та аналізувати, що полегшує роботу для інститутів.

Рисунок 1.5 - Застосування Big Data в освіті

4.Великі дані в охороні здоров'я. Коли справа доходить до того, що таке великі дані в охороні здоров’я, ми бачимо, що вони надзвичайно використовуються. Це включає збір даних, їх аналіз, використання для клієнтів. Крім того, клінічні дані пацієнтів занадто складні, щоб їх можна було розгадати або зрозуміти традиційними системами. Оскільки великі дані обробляються алгоритмами машинного навчання та Data Scientists, робота з такими величезними даними стає легкою. сьогодні лікарі здебільшого покладаються на клінічні історії пацієнтів, а це означає, що потрібно зібрати багато даних, також щодо різних пацієнтів. Старі або традиційні методи зберігання даних не можуть зберігати ці дані. Оскільки існує велика кількість даних, що надходять із різних джерел у різних форматах, потреба в обробці цієї великої кількості даних зростає, і саме тому потрібен підхід Big Data.

5. Електронна комерція. Підтримка відносин з клієнтами є найважливішою в галузі електронної комерції. Веб-сайти електронної комерції мають різні

маркетингові ідеї для роздрібної торгівлі своїми товарами своїм клієнтам, керування транзакціями та впровадження кращої тактики використання інноваційних ідей із великими даними для покращення бізнесу. Прикладом є Flipkart. Flipkart — це величезний веб-сайт електронної комерції, який щодня має великий трафік. Але коли на Flipkart відбувається заздалегідь оголошений розпродаж, трафік зростає експоненціально, що призведе до збою веб-сайту. Отже, для обробки такого типу трафіку та даних Flipkart використовує Big Data. Великі дані можуть допомогти в організації та аналізі даних для подальшого використання [9,10].

6. Соціальні медіа. Соціальні медіа в поточному сценарії вважаються найбільшим генератором даних. Статистика показала, що близько 500+ терабайт нових даних генерується в базах даних соціальних мереж щодня, особливо у випадку Facebook. Згенеровані дані в основному складаються з відео, фотографій, обміну повідомленнями тощо. Одна активність на будь-якому сайті соціальних мереж створює багато даних, які знову зберігаються та обробляються, коли це необхідно. Оскільки дані зберігаються в терабайтах, обробка займе багато часу, якщо це буде виконано нашими застарілими системами. Великі дані є вирішенням цієї проблеми.

Рисунок 1.6 - Застосування Big Data в соціальних медіа

1.3 Прогноз розвитку світового ринку великих даних

Очікується, що розмір глобального ринку великих даних досягне 300,7 мільярдів доларів США до 2027 року, зростаючи при середньорічному зростанні ринку на 10,9% протягом прогнозованого періоду.

Великі дані описуються як великі обсяги неструктурованих або напівструктурованих даних, які були накопичені з часом. Великі дані часто неструктуровані та розпорошені. Термін «великі дані» також відноситься до величезної кількості накопичених даних, а також до їх швидкості, швидкості, різноманітності та складності. Традиційним системам керування базами даних важко обробляти та/або обробляти ці масивні колекції даних, які вимірюються в петабайтах/екзабайтах даних. Великі дані визначаються в цьому дослідженні як величезний набір даних, який не вписується в стандартний дизайн бізнес-баз даних. Зчитувачі RFID, соціальні мережі, сенсорні мережі, Інтернет-текст і документи, телефонні реєстри, індексація Інтернет-пошуку, наукові дослідження, медичні записи, військове спостереження та електронна комерція, серед багатьох інших джерел, створюють великі дані [11].

Великі дані поєднують інформацію з різноманітних джерел і програм. Видобуток, перетворення та завантаження (ETL) — це звичайні процедури інтеграції даних, які не витримують виклику. Щоб оцінити великі масиви даних у масштабі терабайт або навіть петабайт, потрібні нові методології та технології. Під час процесу інтеграції необхідно ввести дані, обробити їх і переконатися, що вони підготовлені та доступні таким чином, щоб бізнес-аналітики могли їх використовувати.

Для великих даних потрібен простір для зберігання. Рішення для зберігання може бути хмарним, локальним або поєднувати обидва. Користувачі можуть зберігати свої дані в будь-якому бажаному форматі, а потім за потреби застосовувати до цих наборів даних свої потреби в обробці та механізми обробки. Рішення людей щодо того, де зберігати свої дані, залежить від того, де вони зараз зберігаються. Хмара швидко набуває популярності, оскільки вона задовольняє

поточні вимоги компаній до обчислень і дозволяє їм за потреби збільшувати ресурси.

Користувачі можуть робити нові висновки з більшими наборами даних. З цією метою нові інвестиції в навички, організацію чи інфраструктуру мають здійснюватися в контексті сильного бізнес-середовища, щоб забезпечити постійні інвестиції та фінансування проектів.

Таблиця 1.1 – Стан та прогноз ринку великих даних

Атрибут звіту	Подробиці
Вартість ринку в 2020 році	149 мільярдів доларів США
Прогноз розміру ринку в 2027 році	300,7 млрд дол
Базовий рік	2020 рік
Історичний період	2017 по 2019 рік
Прогнозний період	2021-2027
Темп зростання доходу	CAGR 10,9% з 2021 по 2027 рік
Кількість сторінок	432
Кількість столів	773
Звіт про покриття	Ринкові тенденції, оцінка та прогноз доходу, аналіз сегментації, розподіл по регіонах і країнах, конкурентний ландшафт, стратегічний розвиток компаній, профіль компанії
Покриті сегменти	Компонент, бізнес-функція, режим розгортання, розмір організації, вертикаль, регіон
Сфера дії країни	США, Канада, Мексика, Німеччина, Великобританія, Франція, Іспанія, Італія, Китай, Японія, Індія, Південна Корея, Сінгапур, Малайзія, Бразилія, Аргентина, ОАЕ, Саудівська Аравія, Південна Африка, Нігерія
Драйвери зростання	- Експоненціальне зростання доступності даних - Розвиток нових тенденцій, таких як медіааналітика
Обмеження	- Занепокоєння конфіденційністю та суворі правила безпеки даних, запроваджені урядом

Розуміння того, як фільтрувати журнали сайтів для аналізу активності електронної комерції, вилучення настроїв із соціальних мереж і контактів служби

підтримки клієнтів, а також підходи до статистичної кореляції та їх значення для клієнта, продукту, виробництва та інженерних даних – це лише кілька прикладів [12].

За вертикаллю ринок сегментований на BFSI, роздрібну торгівлю та споживчі товари, телекомунікації та інформаційні технології, уряд та оборону, охорону здоров'я та науки про життя, виробництво, медіа та розваги, транспорт та логістику та інші. Сегмент роздрібної торгівлі та споживчих товарів отримав значну частку доходу на ринку великих даних у 2020 році. У секторі роздрібної торгівлі аналітика великих даних дозволяє компаніям давати індивідуальні рекомендації на основі їх історії покупок, що призводить до більш персоналізованого досвіду покупок і кращого обслуговування клієнтів. У галузі роздрібної торгівлі великі дані мають вирішальне значення для розробки відповідної стратегії продажів. Щоб зрозуміти поведінку клієнтів, передбачити попит і підвищити ціну, великі дані використовуються на кожному рівні процесу роздрібної торгівлі.

На основі регіонів ринок поділений на наступні сегменти: Північну Америку, Європу, Азіатсько-Тихоокеанський регіон, Латинську Америку, Близький Схід і Африку. Північна Америка стала провідним регіоном із найвищою часткою доходу на ринку великих даних у 2020 році. Більшість підприємств і вертикалей у Північній Америці вважають виявлення даних і аналітику великих даних досить продуктивними. Високий рівень впровадження передових рішень і послуг у різних країнах регіону сприяє зростанню регіонального ринку протягом прогнозованого періоду [13].

Основними стратегіями, яких дотримуються учасники ринку, є партнерства. На основі аналізу, представленого в кардинальній матриці; Корпорація Microsoft і Google LLC є попередниками на ринку великих даних. Такі компанії, як Oracle Corporation, SAP SE та IBM Corporation, є одними з ключових новаторів на ринку.

Звіт про дослідження ринку містить аналіз ключових учасників ринку. Ключові компанії, описані у звіті, включають IBM Corporation, Google LLC, Oracle Corporation, Microsoft Corporation, SAS Institute, Inc., SAP SE, Alteryx, Inc., Teradata Corporation, Salesforce.com, Inc., і TIBCO Software, Inc.

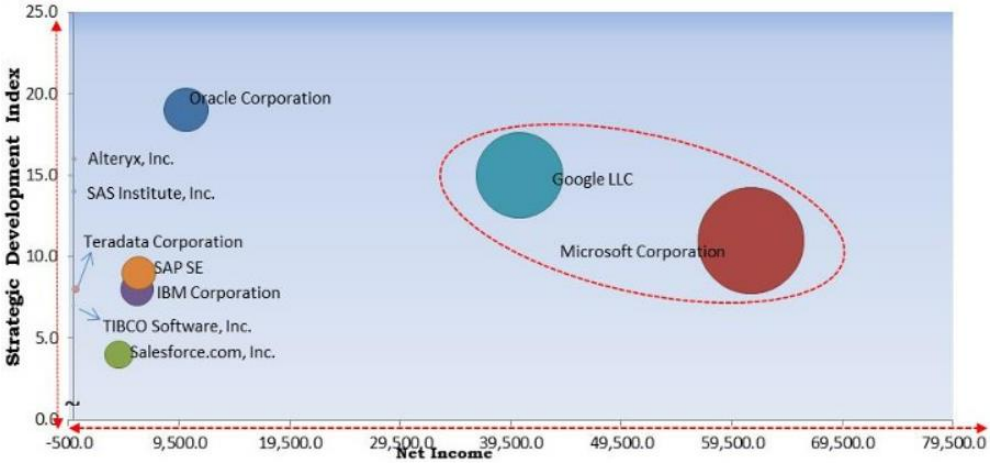


Рисунок 1.7 - Кардинальна матриця KBV - аналіз конкуренції на ринку великих даних

У свою чергу, за прогнозами, наведеними в [14,15], висунення від продажу ринку програмного забезпечення та послуг на світовому ринку більших даних зросте з 42 млрд доларів у 2018 році до 103 млрд доларів у 2027 році, досягнувши совокупного річного темпу зростання (CAGR) в 10,48% рис. 1.8.

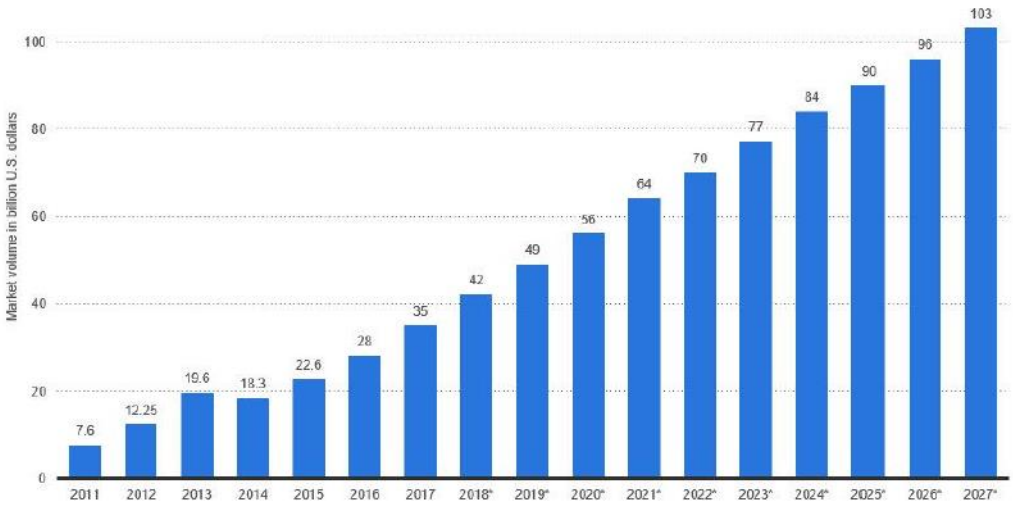


Рисунок 1.8 — Прогноз ринку Великих даних

Згідно з дослідженням Accenture [5], 79% керівників підприємств погоджуються з тим, що компанії, які не використовують більші дані, втрачають свої конкурентні позиції та можуть виявитися на межі випробувань. Більш того,

83% керівників реалізували проекти Big Data, щоб збільшити конкурентну перевагу підприємства на ринку.

Технології великих даних активно використовують перевагу інтелектуальних технологій. Так 59% керівників говорять, що великі дані в їх компанії будуть покращені за рахунок використання ШІ.

Продажі та маркетинг, дослідження та розробки (R&D), управління цепочками поставок (SCM), включаючи дистрибуцію, управління робочими місцями та операціями, - це те, де передова аналітика, включаючи більші дані, вносить найбільший вклад у зростання виручки сьогодні. Дослідження McKinsey [7] являє собою вичерпний огляд того, як аналітичні технології та більші дані дозволяють створити абсолютно нові екосистеми, що служать фундаментальною технологією для штучного інтелекту (ШІ). McKinsey вважає, що аналітика і великі дані вносять найбільш цінний вклад у галузь базових матеріалів і високих технологій.

Майже 50% респондентів, які взяли участь в опитуванні McKinsey Analytics, вважають, що аналітика і більші дані коренним чином змінили методи ведення бізнесу в сфері продажів і маркетингу.

За даними NewVantage Venture Partners [15], більші дані приносять найбільшу користування підприємствам за рахунок скорочення витрат (49,2%) і створення нових можливостей для інновацій (44,3%). 69,4% підприємств почали використовувати більші дані для створення бізнес-процесів, керованими даними

Прогнозується, що ринок Hadoop і Big Data зріс з 17,1 млрд доларів у 2017 році до 99,31 млрд доларів у 2022 році, досягнувши 28,5% CAGR рис.1.9. Найбільший період прогнозованого зростання прийде на 2021 і 2022 роки, коли, згідно з прогнозами, ринок виросте на 30 млрд доларів за один рік [16].

Відповідно до прогнозу [32], що додатки та аналітика для більшої кількості даних зросли з 5,3 млрд доларів у 2018 році до 19,4 млрд доларів у 2026 році, досягнувши показника CAGR 15,49%. Світовий ринок великих даних, що включає в себе професійні послуги, виріс з 16,5 млрд доларів у 2018 році до 21,3 млрд доларів у 2026 році.

Порівняння світового запиту на сучасні технології аналітики даних і пов'язане з великими даними обладнання, послуги та програмне забезпечення, стає очевидним домінуванням останньої категорії. Прогнозується, що сегмент програмного забезпечення збільшиться всього відносно інших категорій з 14 мільярдів доларів у 2018 році до 46 мільярдів доларів у 2027 році рис. 1.10, досягнувши показника CAGR у 12,6% [16].

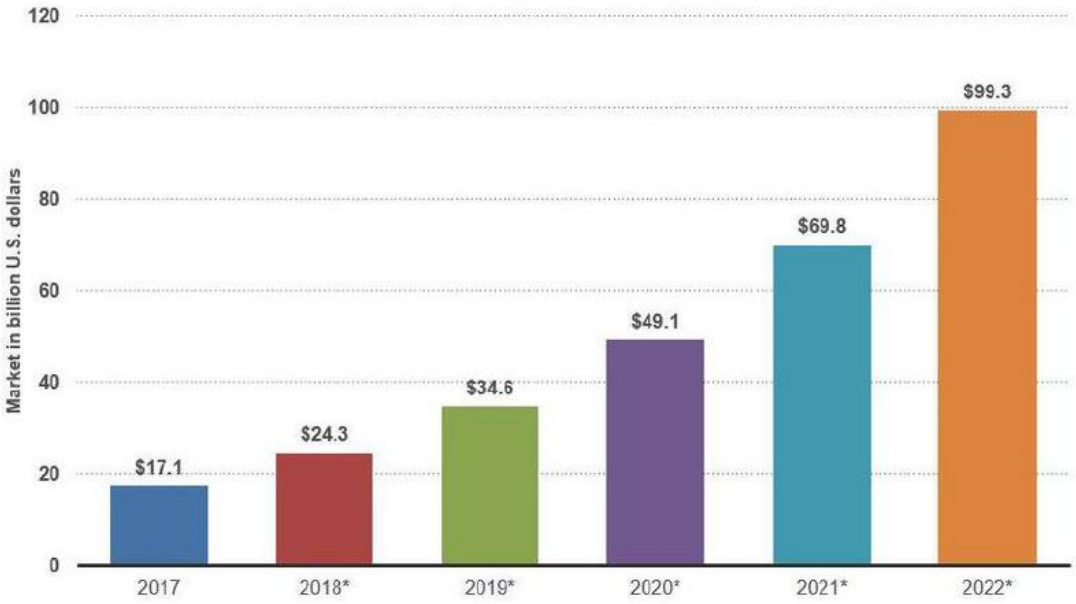


Рисунок 1.9— Прогноз обсягу ринків Hadoop і Big Data

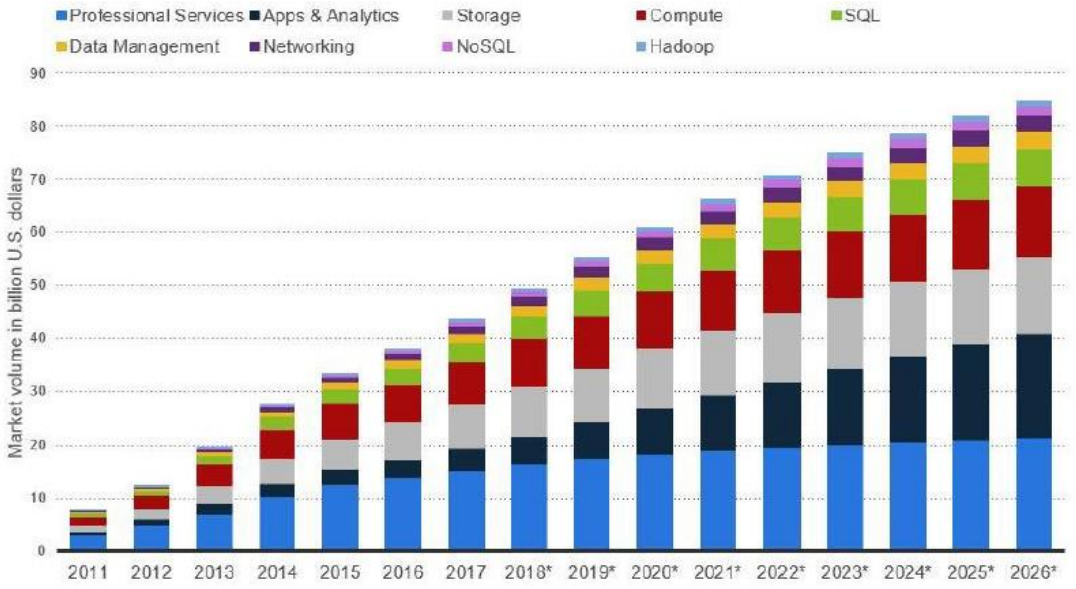


Рисунок 1.10— Прогноз ринку додатків Big Data за сегментами програмного забезпечення

Наведені прогнозні дані говорять про безумовну актуальність теми наукових та інженерних досліджень в області великих даних.

1.4 Аналіз рішень для Big Data

Apache Hadoop. Apache Hadoop — це безкоштовна розподілена файлова система з відкритим вихідним кодом, розроблена для надшвидкої обробки масивних даних, що зберігаються в кластерах, і плавного зростання відповідно до потреб будь-якої організації. Це одне з відомих рішень для зберігання великих даних. Підтримуються розподілені бази даних NoSQL (наприклад, HBase), що дозволяє розподіляти дані по тисячах серверів без впливу на продуктивність. Його можна розгорнути як у хмарі, так і локально. Hadoop YARN керує ресурсами на ПК в кластерах. Його компоненти включають систему розподілу файлів Hadoop (HDFS) для зберігання, MapReduce для обробки даних тощо.

Переваги. Реплікація даних забезпечує послідовний доступ до конфіденційних даних, навіть якщо вони розподілені між численними серверами та пристроями зберігання. Щоб полегшити пошук даних із низькою затримкою, загальнокластерний балансир навантаження рівномірно розподіляє дані між дисками.

Hadoop передає пакетний код багатьом вузлам у кластері, а потім розподіляє файли, забезпечуючи паралельну локальну обробку даних.

Власники бізнесу отримують вигоду від його високого рівня масштабованості та доступності; виявляються та виправляються помилки на рівні програми. Легко додати нові вузли YARN до керування ресурсами, щоб вони могли виконувати завдання, і так само легко видалити їх, щоб можна було зменшити масштаб кластера.

Під керуванням із централізованого розташування користувачі можуть керувати програмою для зберігання блоків даних за власним вибором у локальних кешах, розташованих на кількох вузлах. Користувачі можуть зберігати лише певну

кількість реплік прочитаних блоків у буферному кеші під час використання явного закріплення, звільняючи цінний простір пам’яті для інших цілей.

Hadoop гарантує цілісність даних, не копіюючи фактичні дані, а натомість покладаючись на знімки файлової системи на певний момент часу, щоб зберегти список блоків і розмір файлу. Для швидкого отримання актуальної інформації він реєструє зміни файлової системи у зворотному хронологічному порядку.

Особливості. Ця структура дозволяє програмістам створювати програми обробки даних для обчислення операцій на численних вузлах у кластері. Користувачі можуть запускати окрему версію фреймворку MapReduce, використовуючи розгортання розподіленого кешу, щоб виконувати поточне оновлення [17].

Кодеки стиснення, власні утиліти вводу-виводу для таких цілей, як централізоване керування кеш-пам’яттю, і реалізації контрольної суми – це лише кілька прикладів нативних компонентів, включених до бібліотеки Hadoop.

HDFS NFS Gateway: коли HDFS змонтовано у файловій системі клієнта, користувач може переглядати файли HDFS локально, завантажувати та завантажувати їх.

Оскільки HDFS дозволяє записувати в пам’ять поза купою, дані з пам’яті можна скидати на диск без втручання в конвеєр вводу-виводу, покращуючи швидкість. Lazy Persist Write — це розвантаження даних, яке допомагає пришвидшити час, потрібний запитам для повернення результатів.

Додаткова інформація про inodes може зберігатися в розширених атрибутах, які програми користувача можуть використовувати для асоціації метаданих з файлом або каталогом.

Обмеження. Немає підтримки потокових даних; дозволена лише пакетна обробка. Через це він загалом працює повільніше.

Він неефективний при ітераційній обробці, оскільки не допускає циклічного потоку даних.

Ні сховище, ні мережевий рівень шифрування не застосовуються. Для безпеки використовується автентифікація Kerberos, яку важко дотримуватися.

Apache Spark. Apache Spark, обчислювальний механізм з відкритим вихідним кодом, перевершує Hadoop, оскільки він може обробляти дані як у пакетному, так і в режимі реального часу. Блискавична швидкість обробки Spark стала можливою завдяки його обчислювальній архітектурі «in-memory», яка зберігає проміжні дані в оперативній пам’яті та мінімізує дисковий ввід/вивід. Він був розроблений, щоб доповнити стек Hadoop і забезпечує сумісність з мовою програмування такою як: Java, Python, R, SQL і Scala. Spark — це розширення архітектури MapReduce, яке може обробляти потоки даних а також інтерактивний запит зі швидкістю думки.

Переваги. Під час розгортання Spark має змогу функціонувати на кластерах Apache Mesos, YARN і Kubernetes, а також його можна запускати незалежно та запускати вручну або за допомогою сценаріїв запуску. Користувачі можуть запускати всі демони на одному хості для розробки та тестування.

Spark SQL: Spark SQL забезпечує запити даних через SQL або API DataFrame з підтримкою багатьох джерел даних, зокрема Hive, Parquet, JSON, JDBC тощо. Він надає доступ до вже існуючих сховищ Hive і підключення до інструментів бізнес-аналітики, підтримуючи синтаксис HiveQL, Hive SerDes і UDF.

Потокова аналітика: вона зчитує дані з HDFS, Flume, Kafka, Twitter, ZeroMQ і користувацьких джерел даних, забезпечуючи ефективну пакетну та потокову обробку, об’єднуючи потоки з історичними даними та виконуючи спеціальні запити до даних, що надходять у режимі реального часу. .

Підключення програм R до кластера Spark: SparkR — це пакет, який полегшує цей процес у RStudio, SHELL, Rscript та інших інтегрованих середовищах розробки R. Включення розподіленого фрейму даних для виконання таких операцій, як відбір, фільтрація та агрегація великих наборів даних, а також доступність MLlib роблять можливим машинне навчання.

Особливості. Дизайн: екосистема Spark включає не лише RDD, але й Spark SQL, Scala, MLlib і основне програмне забезпечення Spark. Він використовує архітектуру головний-підлеглий, де програма-драйвер (яка може бути розміщена

на головному або клієнтському вузлі) контролює групу виконавців (розміщених на робочих вузлах) для паралельного виконання завдань.

Основний механізм обробки даних Spark під назвою «Spark Core» полегшує керування пам’яттю на рівні кластера, відновлення після помилок, планування, розподіл і моніторинг діяльності.

Абстракція: відмовостійкі розподілені набори даних (RDD) Spark, колекція елементів, розділених між вузлами для паралельної обробки, дають змогу інтелектуально повторно використовувати дані та змінні. Клієнти також можуть вимагати кешування RDD у пам’яті для подальшого використання. Іншою абстракцією, доступною Spark, є можливість повторно використовувати раніше збережені дані в змінних пам’яті або виконувати арифметичні операції за допомогою лічильників [16].

Використовуючи методи кластеризації, класифікації, моделювання та рекомендацій, Spark дає змогу виконувати такі процеси ML, як трансформація функцій, оцінка моделі та побудова конвеєра ML.

Обмеження. Оскільки безпека вимкнена за замовчуванням, розгортання можуть бути відкритими для атак, якщо їх не налаштовано належним чином.

Схоже, що їх основні версії не сумісні.

Наявність механізму обробки в пам’яті означає, що він використовує багато оперативної пам’яті.

Платформа даних Hortonworks – Cloudera. Yahoo розробила Hortonworks у 2011 році, щоб полегшити перехід на Hadoop для великих компаній. У 2019 році Hortonworks об’єдналася з Cloudera. Hortonworks Data Platform (HDP) — це дистрибутив Hadoop із відкритим кодом і безкоштовний. Він також забезпечує конкурентоспроможну власну експертизу, що робить його привабливим варіантом для компаній, які бажають прийняти Hadoop. HDFS, MapReduce, Pig, Hive і Zookeeper – лише деякі з включених проектів Hadoop. Ambari для адміністрування, Stinger для обробки запитів і Apache Solr для пошуку даних є відкритими в HDP, який відомий своєю безкомпромісною прихильністю до відкритого коду та постачається з нульовим власним програмним забезпеченням. HCatalog є частиною

HDP, яка полегшує зв’язок між Hadoop та іншими бізнес-програмами. Це стало основним корпоративним рішенням для великих даних.

Переваги. Розгортання будь-де: це рішення можна розгортати локально, у хмарі (як компонент Microsoft Azure HDInsight) або як гібридне рішення, відоме як Cloudbreak. Cloudbreak пропонує еластичну масштабованість для ефективного використання ресурсів і розроблено спеціально для компаній, які вже мають локальні центри обробки даних та IT-інфраструктуру.

Масштабованість і висока доступність: за допомогою об’єднання NameNode інфраструктуру компанії можна розширити, щоб розмістити тисячі вузлів і мільярди файлів. NameNodes відповідають за керування шляхом до файлу та інформацією, пов’язаною з відображенням, і об’єднання гарантує, що вони незалежні один від одного. Це призводить до підвищення доступності при зниженні загальної вартості володіння. Крім того, кодування стирання значно підвищує ефективність зберігання даних, забезпечуючи ефективнішу реплікацію даних.

Безпека та управління: Apache Ranger і Apache Atlas забезпечують відстеження походження даних від точки їх походження до озера даних. Це дає змогу створювати ретельні журнали аудиту для керування конфіденційною чи секретною інформацією.

Скорочений час виходу на ринок: це дає організаціям можливість розгортати програми за лічені хвилини, скорочуючи час, необхідний для виведення продуктів на ринок. Використання графічних процесорів дозволяє впроваджувати машинне та глибоке навчання в програми (графічні процесори). Гібридна архітектура даних цієї компанії забезпечує хмарне сховище для необмеженої кількості даних, які зберігаються у вихідному форматі. Це хмарне сховище можна знайти в ADL, WASB, S3 і GCP.

Особливості. Централізована архітектура: оператори Hadoop можуть за потреби розширювати свої ресурси великих даних завдяки Apache YARN на серверній частині. Для операцій, безпеки та управління YARN без особливих зусиль надає ресурси та послуги для програм, розподілених по кластерах. Це

допомагає компаніям перевіряти дані, отримані з широкого діапазону джерел і форматів [9].

Програми сторонніх розробників швидше розгортаються в Apache Hadoop завдяки вбудованій підтримці YARN для контейнерів Docker. Користувачі можуть тестувати різні версії однієї програми, не впливаючи на поточну. Якщо ви поєднаєте це з природними перевагами контейнерів — ресурсоефективністю та підвищеною пропускною спроможністю завдань — ви отримаєте конкурентоспроможне рішення.

Доступ до даних: за допомогою YARN різні методи доступу до даних можуть співіснувати в одному кластері проти загальних наборів даних. HDP використовує цю здатність, щоб дозволити користувачам взаємодіяти з кількома наборами даних одночасно кількома способами. У результаті бізнес-користувачі можуть керувати та аналізувати дані в одному кластері, використовуючи інтерактивний SQL, потокову передачу в реальному часі та пакетну обробку, таким чином усуваючи накопичені дані.

Інтероперабельність. Розроблений «знизу вгору», щоб надати організаціям рішення Hadoop із повністю відкритим вихідним кодом, HDP бездоганно взаємодіє з широкою різноманітністю центрів обробки даних і програм BI. Компанії можуть легко підключити свою поточну IT-інфраструктуру до HDP, заощаджуючи гроші, час і зусилля.

Обмеження. Реалізація SSL під час використання кластера Kerberized є серйозною проблемою.

Hive є частиною HDP, однак до даних не можуть застосовуватися додаткові заходи безпеки.

Платформа розширеної аналітики Vertica. Після злиття MicroFocus і HPE у 2017 році компанія Vertica, яка з 2011 року належить Hewlett Packard Enterprises (HPE), увійшла до складу Microfocus. Платформа Vertica Analytics, як і Hadoop, є масштабованим рішенням для великих даних, яке використовує масові паралельні обчислення, але також має реляційну базу даних нового покоління, звичайний SQL і транзакції ACID. Hadoop чудово підходить для пакетної обробки, а платформа

Vertica Analytics також дозволяє проводити аналітику в реальному часі. Вони співпрацюють за допомогою кількох з'єднань, таких як з'єднувач HDFS, який дозволяє завантажувати дані на платформу Vertica Advanced Analytics.

Переваги. Управління ресурсами: за допомогою диспетчера ресурсів користувачі можуть дозволити ефективне виконання одночасного робочого навантаження. Це зменшує використання процесора та пам'яті, а також час обробки дискового вводу-виводу та стискає дані до 90% без шкоди для інформації. Його механізм SQL підтримує масивну паралельну обробку (MPP) і пропонує активне резервування, автоматичну реплікацію, відновлення після відмови та відновлення.

Це високопродуктивна аналітична база даних, яку можна встановити локально, у хмарі або як гібридну систему. Він розроблений для роботи в хмарах Amazon, Azure, Google і VMware.

Управління даними: завдяки зберіганню даних у вигляді стовпчиків, він підходить для завдань, які інтенсивно читають. Vertica приймає широкий діапазон форматів вхідних файлів і має швидкість завантаження кілька гігабайт на секунду на машину на один потік завантаження. Коли багато користувачів отримують доступ до тих самих даних одночасно, блокування даних використовується для контролю якості даних.

Інтеграція: допомагає аналізувати дані з Apache Hadoop, Hive, Kafka та інших систем озера даних за допомогою вбудованих конекторів і стандартних клієнтських бібліотек, таких як JDBC і ODBC. Він підключається до таких продуктів BI, як Cognos, Microstrategy і Tableau, а також до таких систем ETL, як Informatica, Talend і Pentaho [18].

Vertica об'єднує функції бази даних із аналітичними можливостями, такими як машинне навчання та методи регресії, класифікації та кластеризації. Підприємства можуть використовувати його готові можливості геопросторового аналізу та аналізу часових рядів, щоб отримати швидкі результати щодо вхідних даних без придбання додаткових аналітичних рішень.

Особливості. З точки зору підготовки даних, гнучкі таблиці дозволяють користувачам імпортувати та перевіряти як структуровані, так і напівструктуровані набори даних.

Про Hadoop: надійні запити та аналітика Vertica для SQL стали можливими завдяки його прямій інсталяції на Apache Hadoop. Він може читати файли Parquet і ORC, обидва з яких є рідними для Hadoop, а також записувати їх як Parquet.

Використовуючи зведені таблиці, аналітики можуть швидко складати запити та виконувати складні операції JOIN. Вони не залежать від оригінальних баз даних, тому зміна однієї не вплине на іншу. Через це складні структури бази даних можуть підтримувати обробку великих даних у більш швидкому темпі.

Завдяки розробнику баз даних можливі дизайни з оптимізованою продуктивністю для спеціальних запитів і оперативних звітів за допомогою автоматично або вручну встановлених сценаріїв SQL.

Аналізатор робочого навантаження перевіряє системні таблиці, щоб надати пропозиції щодо оптимізації та рекомендації для об’єктів бази даних. Використовуючи робоче навантаження та історію виконання запитів, а також доступні ресурси та специфікації системи, можна виконати аналіз першопричини.

Обмеження. Перевірка зовнішнього ключа або посилальної цілісності не підтримується.

У разі використання із зовнішніми таблицями автоматичні обмеження не підтримуються.

На видалення потрібен час, що може затримати виконання інших завдань.

РОЗДІЛ 2 ДОСЛІДЖЕННЯ АРХІТЕКТУРНИХ ТА ПРОГРАМНИХ РІШЕНЬ ПРИ ПОБУДОВІ СИСТЕМ ЗБЕРІГАННЯ ТА АНАЛІЗУ BIG DATA

2.1 Огляд архітектурних рішень великих даних

На сьогоднішній день існує безліч архітектурних рішень та інструментів, які використовуються для Big Data. Широковідомою основою для побудови систем великих даних є система Hadoop що є проектом Apache Software Foundation, що є набір утиліт, бібліотек і фреймворків, що вільно розповсюджуються, для реалізації розподілених програм і їх застосування у кластерах [19]. Автор вважає різноманітність архітектурних прикладів для додатків Big Data можливо сортувати за такими характеристиками: типи Big Data та застосування хмарних обчислень. За типом даних можливо виділити два класи додатків:

- системи обробки пакетів великих даних (Batch Processing/Data Processing);
- системи потокової обробки великих даних (Stream Processing).

Системи обробки пакетів обробляє вже накопичені за певний період дані. Якприклад, дані за період можутьати мільйон записів і зберігатися як файл. Цей файл може оброблятися в кінці дня або наступного дня. Тоді є можливість у тому, що для обробки денних даних необхідно чимало часу. Hadoop MapReduce, за оцінкою чималої кількості фахівців, платформа що є більш підходящою для обробки даних у пакетному режимі. На рис. 2.1 зображено схему обробки даних за допомогою MapReduce. [20].

Системи обробки пакетів можна застосовувати у тому випадку, коли відсутня необхідність обробки в реальному часі, а загальною метою є отримання більш детальних даних.

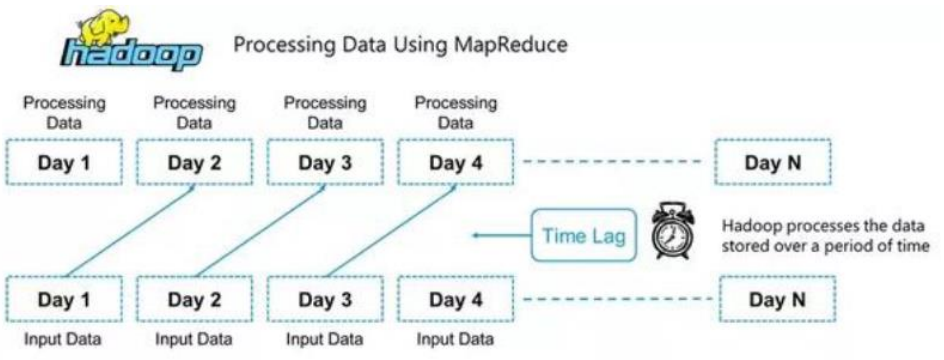


Рисунок 2.1 – Схема пакетної обробки даних у Hadoop MapReduce

Ситуація у якій використовують систему потокової обробки, коли потрібно одержати аналітичні результати в часі. При потоковій обробці є можливість обробляти дані в реальному часі в міру їх надходження та швидко виявляти деякі ситуації напротязі нетривалого періоду з часу отримання інформації. Загальною причиною більшої швидкості потокової обробки є особливістю системи, яка полягає в можливості аналізувати дані та запису у файлову систему.

Потокова обробка є корисною для наступних завдань, а саме моніторинг активностей сайтів, запобігання шахрайства під час проведення транзакцій у різних сферах бізнесу або навчанні в режимі ревльного часу, а саме при обробленні інформації транзакцій у потоковому режимі, виявлення різних аномалій, які дають сигнал про шахрайство в режимі реального часу для їхнього блокування. На рис. 2.2 наведено схему обробки даних у реальному часі з використанням Apache Spark.

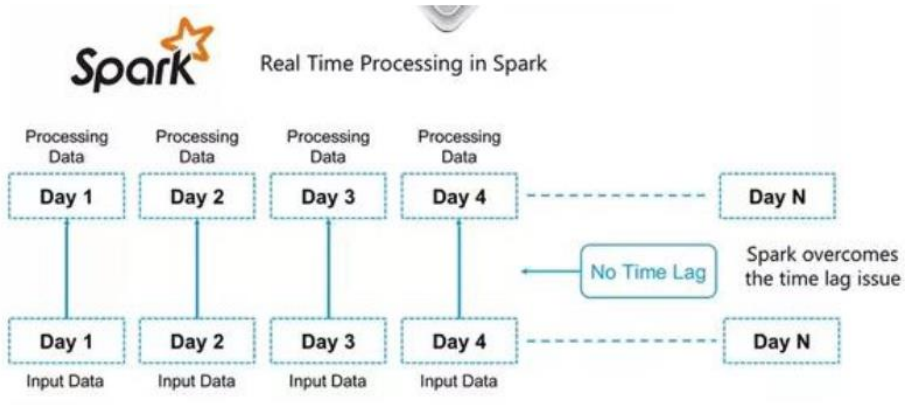


Рисунок 2.2 – Обробка великих даних у реальному часі за допомогою Apache Spark

Деякий інтерес реалізації систем потокової обробки даних становлять системи, які побудуються з урахуванням лямбда архітектури [21]. Однак, у цій роботі це питання не розглядається.

З погляду застосування хмарних обчислень системи великих даних можуть класифікуватися такі види рис. 2.3 [24]:

- інфраструктура як сервіс – Infrastructure as a Service (IaaS); - платформа як сервіс – Platform as a Service (PaaS);

- Дані як сервіс - Data as a Service (DaaS);

- бізнес-рішення для великих даних – Big Data Business Functions as a Service (BFaaS).

Системи класу IaaS включають сховища, сервери та мережеві рішення як базові, відносно недорогі рішення для створення стека великих даних. Розподілені файлові системи є частиною цього рівня.

Системи класу PaaS надають сховища даних NoSQL та розподілені кеші, які можуть бути логічно запрошені з використанням мов запитів для створення платформного рівня великих даних. Цей рівень забезпечує логічну модель для необроблених, неструктурованих даних, які у файлах.

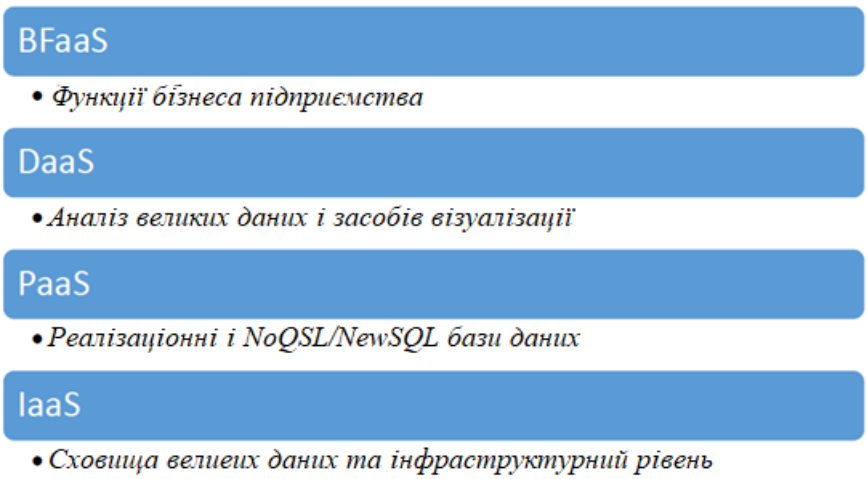


Рисунок 2.3 – Хмарні рівні архітектури для побудови систем великих даних

DaaS системи включають весь набір інструментів, доступних для інтеграції з шаром PaaS з використанням пошукових систем, інтеграційних адаптерів,

пакетних програм і т.д. API-інтерфейси, доступні цьому рівні, можуть використовуватися всіма кінцевими системами як обчислювальної еластичності (elastic-computing mode) [23].

Системи BFaaS можуть створюватись як наскрізні рішення над рівнем DaaS. BFaaS рішення створюються для певних галузей, таких як охорона здоров'я, енергетика, банківська справа, роздрібна та електронна торгівля.

Як зазначалося вище, для побудови систем великих даних може бути використаний великий набір фреймворків, продуктів, бібліотек, які називають екосистемою великих даних.

2.2 Аналіз екосистеми великих даних

Екосистема великих даних включає наступні групи інструментів [25]:

- розподілені файлові системи (Distributed File Systems – DFS);
- інструменти розгортання; бази даних NoSQL та NewSQL; інструменти інтеграції даних; інструменти машинного навчання;
- інструменти програмування служб;
- інструменти планування;
- інструменти порівняльного аналізу; інструменти безпеки.

Для зберігання великих обсягів даних використовуються розподілені файлові системи, які мають такі відмінності:

- здатність зберігати файли, розмір яких перевищує розмір окремого диска сервера зберігання;
- файли, що зберігаються, автоматично реплікуються на серверах зберігання, що дозволяє здійснювати паралельну обробку і створювати надмірність для забезпечення надійної роботи кластера;
- система може легко масштабуватись, використовуючи принцип горизонтального масштабування.

Найбільш поширеною розподіленою файловою системою є Hadoop File System. Прикладами DFS також є: Red Hat ClasterFS, QuantCast File System, Ceph File System.

Для зберігання великих даних потрібні системи баз даних та засоби доступу до даних. Через відомі обмеження використання класичних реляційних бfз даних, таких як Oracle SQL, DB2, неможливо. У системах аналізу та зберігання великих даних застосовуються системи, що отримали назву NoSQL та їх подальший розвиток NewSQL. Більш детально вказані системи баз даних будуть розглянуті нижче. Тут зазначимо, що крім самих систем баз даних, необхідні системи збирання, перетворення пакетних та потокових даних.

Інструменти інтеграції даних, як це випливає з назви, служить для злиття даних за допомогою переміщення даних з одного джерела до іншого. Як зазначалося вище, існують два основних класу рішень великих даних обробки пакетних і потокових даних. Прикладами рішень пакетної обробки є Apache Sqoop. Прикладами рішення для використання в обробці потокових даних є неординарність задачі, що призвела до появи таких технологій, як Apache Flume, Apache Storm і Apache Kafka.

Після переміщення даних у розподілену файлову систему необхідно перейти до вилучення інформації з даних. Для цього в процесі аналізу великих даних використовуються методи прикладної математики, математичної статистики, машинного навчання. Однак важливою відмінністю алгоритмів, що використовуються при аналізі великих даних, є те, що алгоритми повинні бути розподіленими. На сьогоднішній день існує багато бібліотек та фреймворків, які реалізують зазначені алгоритми. Прикладами мов програмування, що активно використовуються в даній задачі є Python, Java, R. Наприклад, для Python є такі бібліотеки, як:

- NLTK – Natural Language Toolkit – бібліотека обробки даних природною мовою;

- Scikit-learn – одна з найвідоміших бібліотек машинного навчання;

– TenzorFlow – бібліотека глибокого машинного навчання від Google.

Прикладом системи машинного навчання у реальному часі може бути Apache Spark.

Інфраструктури розподіленого програмування спрощують реалізацію розподілених алгоритмів, оскільки реалізують «низькорівневі розподілені» завдання, приховуючи їх від програміста. До таких завдань можна віднести: перерозподіл завдань у разі їхнього збою на вузлі обчислень, комунікація між підпроцесами. Прикладами інфраструктури розподіленого програмування є Apache Thrift, Zookeeper.

Інструменти планування служать для автоматизації завдань, що повторюються. Наприклад, запуск задач MapReduce з появою нового набору даних. Представниками цієї групи є Hadoop YARN.

Інструменти порівняльного аналізу служать оптимізації інфраструктур великих даних з допомогою використання стандартизованих профілів. Кожен профіль будується на основі певного набору інструментів для зберігання та обробки великих даних.

2.3 Приклад архітектури систем зберігання та аналізу великих даних

Будь-яка інформаційна система зберігання та аналізу великих даних повинна будуватися на певній апаратно-програмній архітектурі. Існує узагальнена архітектурна схема додатків великих даних, що визначає технологічний стек великих даних (big data tech stack). У випадку архітектура великих даних то, можливо представлена як, зображеному рис. 2.4 [24].

Опишемо рівні представленої архітектури докладніше.

Рівень джерел даних (Data Sources). Для підприємств зазвичай доступні кілька внутрішніх та зовнішніх джерел даних. У цьому є вимоги, щоб перед записом дані мають бути очищені, верифіковані, масштабовані.

Дані можуть постачатися в різних форматах: результати запитів до реляційних баз даних та сховищ даних, повідомлення електронної пошти, XML,

JSON, HTML, миттєві повідомлення, відео та аудіо дані, формати документів офісних програм (Word, Excel, pdf), а також потокові дані.

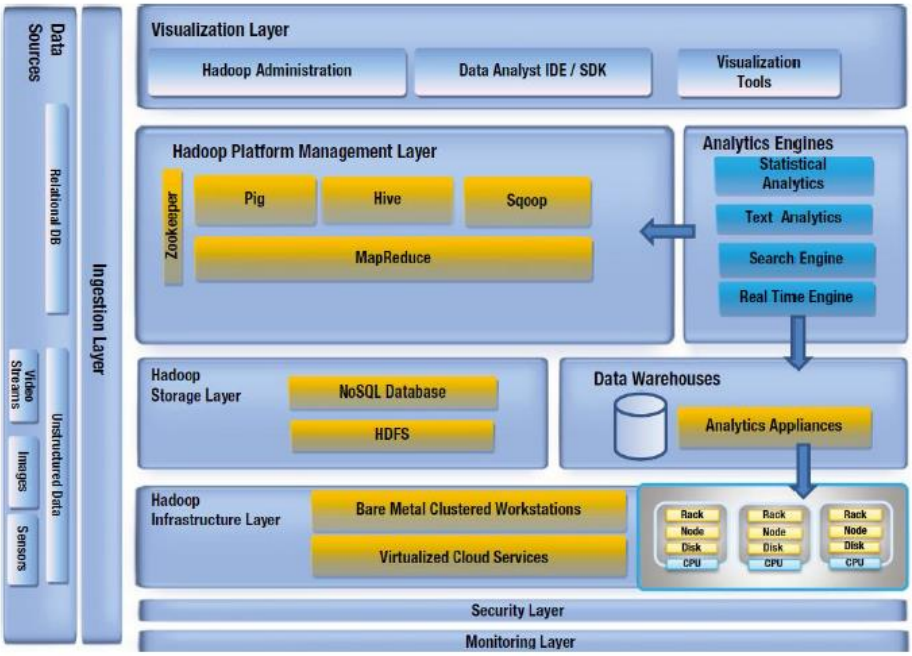


Рисунок 2.4 - Архітектура системи зберігання та аналізу великих даних

Як уже зазначалося вище, дані джерела характеризуються великою швидкістю видачі даних, великою різноманітністю форматів, великим обсягом даних.

Шар завантаження даних (Ingestion Layer). Шар завантаження рис. 2.5 – новий шар обробки даних підприємства. Цей шар відповідає за відокремлення шуму від відповідної інформації. Алгоритми цього шару повинні мати можливість перевіряти, очищати, перетворювати, скорочувати та агрегувати дані в технічний стек великих даних для подальшої обробки. Це нове проміжне програмне забезпечення, яке має бути масштабованим, стійким до відмови, гнучким і регулюючим в архітектурі великих даних. Відповідно до процесу Data Science помилки в даному шарі можуть звести нанівець всю подальшу роботу.

Рівень завантаження завантажує остаточну релевантну інформацію без шуму на розподілений рівень зберігання Hadoop. Алгоритми цього рівня повинні

перевіряти, очищати, перетворювати, скорочувати та інтегрувати дані в технологічний стек великих даних для подальшої обробки.

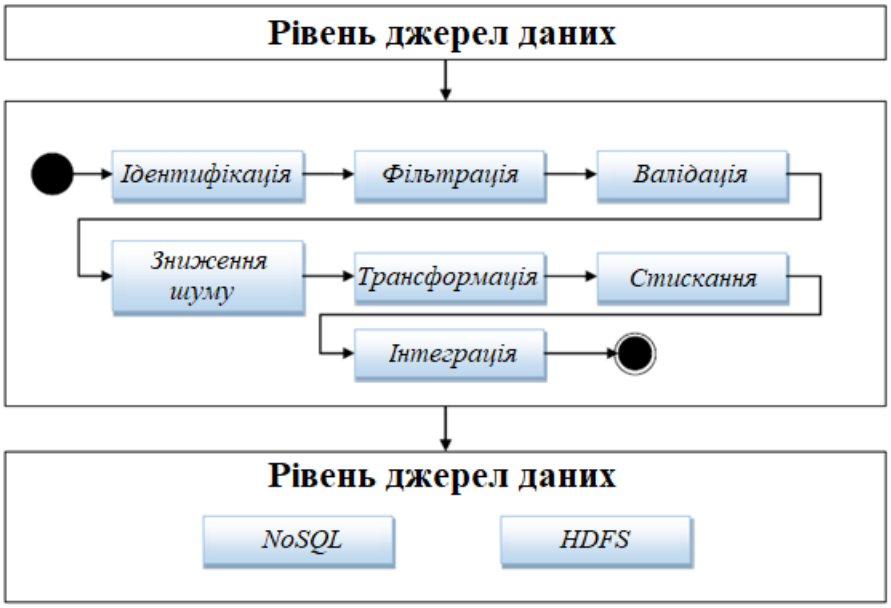


Рисунок 2.5 – Роль рівня завантаження даних в архітектурі додатків зберігання та аналізу великих даних

Архітектурні шаблони шару завантаження описують рішення часто зустрічаються проблем джерел даних з погляду впливу рівень завантаження. Ці рішення можуть бути обрані на основі вимог до продуктивності, масштабованості та доступності. Ми розглянемо ці шаблони (які показані малюнку 3-1) у наступних розділах. У цьому розділі ми розглянемо такі поширені шаблони завантаження пакетних і потокових даних [24]:

- шаблон отримання даних з безлічі джерел (Multisource Extractor Pattern) є підходом для ефективного використання декількох типів джерел даних; шаблон перетворювача протоколу (Protocol Converter Pattern). У цьому шаблоні використовується посередник протоколу забезпечення абстракції для вхідних даних із різних рівнів протоколу;

- шаблон багатоцільового призначення (Multidestination Pattern): цей шаблон використовується в сценарії, коли шар завантаження повинен переносити дані до

декількох компонентів зберігання, таких як розподілена файлова система Hadoop, вітрини даних або аналітичні механізми реального часу;

- шаблон перетворення «точно вчасно» (Just-in-Time Transformation Pattern). Великі обсяги неструктурованих даних можуть бути завантажені в пакетному режимі за допомогою традиційних інструментів і методів ETL (вилучення, передачі та завантаження). Однак дані перетворюються лише тоді, коли це необхідно для економії часу обчислень;

- Шаблони потокової передачі в реальному часі (Real-Time Streaming patterns). Деякі бізнес-проблеми вимагають миттєвого аналізу даних, що надходять на підприємство. У цих умовах необхідні завантаження та аналіз даних у режимі реального часу.

Рівень розподіленого зберігання (Distributed (Hadoop) Storage Layer) забезпечує надійне, масштабоване оточення для обчислення паралельних алгоритмів обробки великих даних Розподілена файлова система Hadoop є основним елементом рівня. Безпосереднє керування доступом у розподілених даних здійснюють NoSQL бази даних, що розглядаються нижче.

Інфраструктурний рівень (Hadoop Infrastructure Layer) – рівень, який підтримує рівень зберігання, тобто фізичну інфраструктуру. Інфраструктурний рівень є основним для роботи та масштабованості архітектури великих даних. Для підтримки несподіваного чи непередбачуваного обсягу, швидкості чи різноманітності даних фізична інфраструктура для великих даних має відрізнятися від інфраструктури для традиційних реляційних даних.

Рівень фізичної інфраструктури Hadoop (Hadoop physical infrastructure layer – HPIL) ґрунтується на моделі розподілених обчислень. Це означає, що дані фізично зберігаються на багатьох місцях і зв'язуються разом через мережі та розподілену файлову систему. Це архітектура «без поділу ресурсів», у якій дані та функції, необхідні управління ними, перебувають разом одному вузлі. На відміну від традиційної моделі клієнт-сервер дані більше не передаються на монолітний сервер, де для їх обробки застосовуються функції SQL. В інфраструктуру цього рівня вбудована надмірність.

Рівень безпеки (Security Layer). Оскільки аналіз великих даних стає одним із головних завдань для організацій, безпека цих даних стає також головним завданням. Збережені дані та результати їх обробки повинні бути захищені як для дотримання відповідних вимог, так і для захисту приватного життя людини. Тому з самого початку мають плануватись засоби авторизації та аутентифікації.

Щоб реалізувати базову основу безпеки, необхідно спроектувати певний стек технологій, який мінімально виконував такі дії:

- аутентифікація вузлів, використовуючи протоколи, такі як Kerberos;

- шифрування на рівні файлів;

- підписка на службу управління ключами для довірених ключів та сертифікатів; використання таких інструментів, як Chef або Puppet, для перевірки під час розгортання наборів даних або застосування виправлень на віртуальних вузлах;

- реєстрація зв'язку між вузлами та використання розподілених механізмів реєстрації для відстеження аномалій за рівнями;

- гарантія того, що зв'язок між вузлами є безпечним. Наприклад, використання SSL, TLS тощо.

Моніторинговий рівень (Monitoring Layer). Системи моніторингу застосовуються для відстеження стану розподілених кластерів і збирають інформацію про операційні системи, обладнання, що використовуються і т.п. Для виконання цієї задачі машини повинні взаємодіяти з інструментом моніторингу через протоколи високого рівня, такі як XML замість двійкових форматів, які залежать від машини. Моніторингові системи також повинні надавати інструменти для зберігання та візуалізації даних.

Для моніторингу стеків великих даних широко використовуються такі інструменти з відкритим кодом, як Ganglia і Nagios.

Додатки аналітичного рівня (Analytics Engine) служать до виконання пошукових запитів над розподіленими даними, виконують аналітичну обробку тексту, статистичну аналітику і виконують інтелектуальні алгоритми над даними.

Більш детально алгоритми аналізу великих даних буде розглянуто у наступному розділі.

Засоби візуалізації (Analytics Engine). Як правило, «сирі» вихідні аналітичні дані не можуть використовуватися для вирішення бізнес-завдань. Необхідний переведення аналітичних даних у табличну або графічну форму, а також можливість погляду на дані під різними кутами. Тому засоби візуалізації є невід'ємною частиною систем зберігання та аналізу великих даних.

Засоби візуалізації працюють поверх консолідованих та агрегованих вихідних даних. У тому випадку, коли потрібне вивчення результатів аналітики в режимі реального часу, можуть використовуватися механізми реального часу, що працюють на механізмах комплексної обробки подій (Complex Event Processing – CEP) та архітектурах (Event-driven Architectures – EDA). На рис. 2.6 показано взаємодію між різними рівнями стека великих даних та традиційними засобами BI.

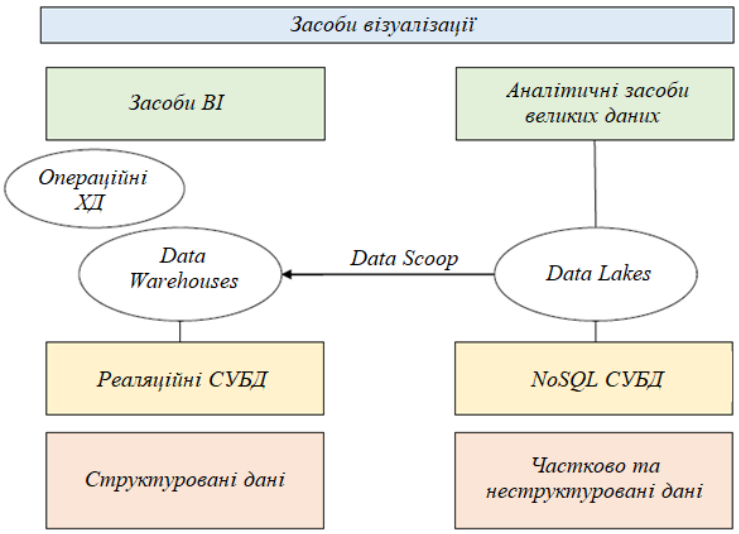


Рисунок 2.6 – Концептуальна архітектура шару візуалізації

2.4 Огляд видів NoSQL баз даних та їх області застосування

Основу сучасних трансакційних систем становлять реляційні бази даних. Однак, незважаючи на їх поширення, реляційні бази мають усім відоме обмеження, пов'язане з проблемами продуктивності при виконанні запитів, що агрегують, до

даних. Для вирішення цієї проблеми в BI системах використовуються рішення, що базуються на використанні сховищ даних (ХД). Однак ХД через свої обмеження не можуть бути безпосередньо використані при побудові систем зберігання та аналізу великих даних.

Другою проблемою традиційних систем керування даних є те, що в основному вони покликані для розгортання на одній машині. Проте, вже досить давно було розроблено рішення для розпаралелювання роботи на розподілених кластерах. Такі рішення отримали узагальнену назву NewSQL [24].

У системах зберігання та аналізу великих даних найширше застосування отримали системи управління базами даних під узагальненою назвою NoSQL (Not Only SQL – не лише SQL). Особливістю NoSQL БД є їхня здатність працювати не тільки з великими даними (Volume), але і з рештою «V».

На сьогоднішній день загальноприйнятою є класифікація NoSQL баз даних за чотирма видами:

- стовпцеві;
- «ключ-значення»;
- сховища документів;
- графові.

Ці різновиди NoSQL баз даних дозволяють по-різному організовувати дані залежно від завдання. Крім того, зазначені різновиди застосовуються до різних ситуацій з точки зору складності організації та розміру даних рис. 2.7.

Крім різних форматів і підходів до організації даних, що зберігаються, реляційні та NoSQL бази даних мають різні базові принципи, що визначають стан системи зберігання даних.

Так для реляційних та графових баз даних повинен виконуватися принцип ACID, що описує властивості класичних транзакцій у реляційних базах. Принцип ACID має на увазі чотири складові:

- Atomicity, Consistency, Isolation, Durability – ACID: Atomicity – атомарність. Якщо відбувається CRUD операція над блоком даних, що складається з кількох пов'язаних дій, операція вважається успішною, якщо всі елементи послідовності

успішні. Інакше операція скасовується. Цей принцип відомий також під назвою "все чи нічого";

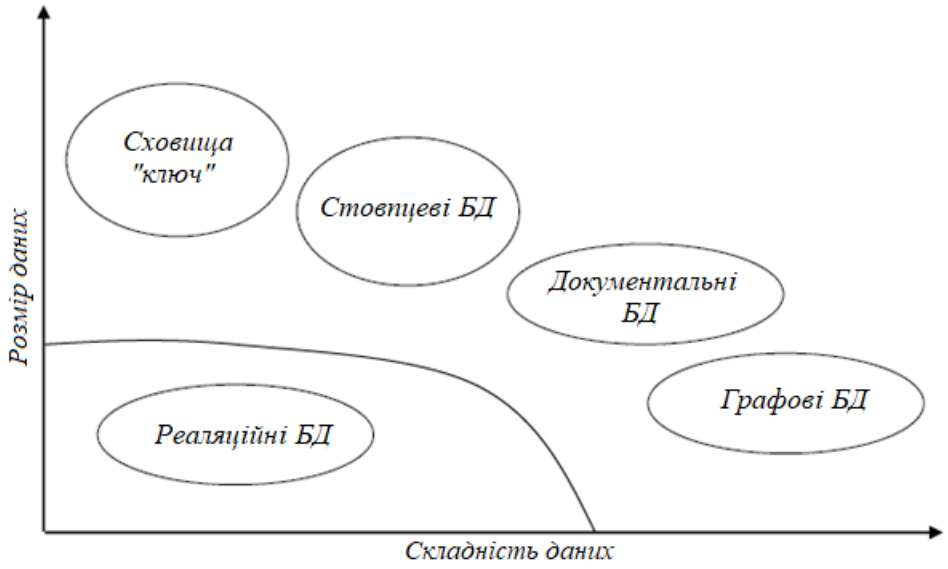


Рисунок 2.7 – Області застосування NoSQL баз даних у системі «Складність-розмір» даних

- Consistency – узгодженість. Якщо стан системи є у несуперечливому стані до виконання транзакції, то після виконання транзакції система також повинна перебувати у несуперечливому стані;

- Isolation – ізольованість. Якщо над даними відбувається кілька паралельних транзакцій, їх результат має бути такий самий, якби ці транзакції виконувались послідовно;

- Durability – сталість. Якщо дані внесені до бази даних, вони повинні там знаходитися постійно. Жодні збої, мабуть, крім фізичного руйнації носія, що неспроможні призвести до втрати даних.

Дотримання принципів ACID призводить у високонавантажених реляційних системах до великих проблем, пов'язаних з забезпеченням принципу ізоляції паралельних транзакцій. Дійсно, при здійсненні паралельних транзакцій можуть відбуватися такі помилки [22]:

- втрати оновлення (Lost Update);

- не чисте читання (Dirty Read);

- неповторне (розмите) читання (Fuzzy Read); - фантом (фіктивні елементи) (Phantom).

Для забезпечення прийнятної продуктивності системи доводиться йти на деякі компроміси, визначаючи механізми блокувань та рівні ізоляції:

- читання даних які не зафвіксовані (READ UNCOMMITTED). Найнижчий рівень, коли транзакція може читати незафіксованої інформації;

- читання зафіксованої інформації (READ COMMITED). У цьому рівні транзакція неспроможна читати незафіксовані дані; - повторюване читання. Транзакція не може змінити дані, що зчитуються іншими транзакціями;

- серіалізоване читання (SERIALIZABLE). Найвищий рівень ізоляції. Транзакція має ексклюзивне право читання. Інші транзакції що неспроможні ні читати, ні записувати ці дані.

Однак, навіть незважаючи на наявність різних рівнів ізоляції, застосування реляційних систем управління базами даних у розподілених системах не є прийнятним. Для розподілених систем зберігання, коли дані розподілені по різних серверах, дотримання принципів ACID неможливе через теорему CAP [25]. Відповідно до теореми CAP розподілена база даних може мати будь-які дві з трьох властивостей, але ніколи всіма трьома. До зазначених властивостей теорема CAP відносить:

- узгодженість (Consitency). Запитуючий вузол отримає одні й самі дані, незважаючи на те від якого вузла-джерела ці дані прийшли;

- доступність (Availability). Якщо вдалося зв'язатися з вузлом і це вузол знаходиться у працездатному стані, то вузол обов'язково відповість, навіть якщо зникне зв'язок з іншими вузлами бази;

- стійкість до поділу (Partition tolerance). Дані в базі не перестануть бути доступними через мережну сегментацію або мережний збій.

Таким чином, якщо система зберігання даних секційована і має стійкість до поділу, то вона або доступна, або узгоджена. Виходячи зі сказаного, NoSQL бази даних забезпечують виконання принципів BASE, а не ACID:

- базова доступність (Basically Available). Гарантується доступність у сенсі CAP. На одному вузлі зберігаються кілька сегментів даних з інших вузлів, що дозволяє вирішувати проблему доступу до інформації під час мережного збою;

- нестійкий стан (Soft state). Стан системи змінюється з часом і може переходити в неузгоджений стан, але в результаті діє принцип «узгодженості зрештою»;

- узгодженість зрештою (Eventual consistency). Цей принцип свідчить, що система рано чи пізно прийде у узгоджений стан. Наприклад, якщо на вузлі А були змінені дані, репліковані на вузлі, то через деякий час дані будуть оновлені і на вузлі В. При цьому ніяких додаткових дій від самих вузлів (користувача) не потрібно.

Стовпцеві БД. Традиційні реляційні бази даних є рядково орієнтованими, коли кожному рядку таблиці відповідає унікальний ідентифікатор – первинний ключ (Primary Key – PK). У цих базах для пошуку інформації доводиться сканувати дані в таблиці за певними стовпцями, виконуючи відповідні запити. Незважаючи на те, що для прискорення виконання запитів застосовується індексування, у розріджених таблицях пошук може бути малоефективним у принципі. Крім того, певні проблеми може скласти додавання одного або кількох атрибутів (стовпців).

Тому в стовпцевих базах даних, як правило, стовпці зберігаються окремо, що призводить до скорочення часу сканування по окремих стовпцях.

На відміну від реляційних баз даних добре зарекомендували себе при побудові OLTP систем, коли часто виконується CRUD операція над усім записом (рядком), стовпцеві бази даних застосовні при побудові аналітичних систем. Часто бази даних аналітичних систем оновлюються за розкладом (наприклад, ночами), переносячи дані з реляційних сховищ OLTP систем. Надалі виконання аналітичних запитів (пошук та агрегація) відбувається за дуже короткий проміжок часу з використанням алгоритмів MapReduce.

Прикладами стовпцевих баз даних є Apach HBase [10], HyperTable [33], Apache Cassandra [8], Google BigTable [16], Yandex ClickHouse [17].

NoSQL бази даних "ключ-значення". Бази даних «ключ-значення» є набором записів, які мають унікальне поле «ключ» і відповідне поле «значення». Таким чином, дані тип сховищ реалізує відому структуру даних map. Завдяки своїй простоті сховища «ключ-значення» мають найкращу масштабованість і, відповідно, дозволяють зберігати величезні обсяги інформації.

З урахуванням відомого прийому організації даних у колекціях map, коли як «значення» може бути інша структура даних (рядок, список, безліч і навіть відображення (map)), структура даних може бути дуже різноманітною.

Прикладами сховищ ключ-значення є Reddis [21], MemCache [11], Amazon DynamoDB [6], Microsoft Azure Cosmos DB [12].

Документальні NoSQL бази даних. Документальні NoSQL бази даних передусім передбачають певну структуру документів. Структура документів визначається схемою документів. Найбільш часто зустрічаються структури, що базуються на мовах розмітки таких як XML і JSON. При використанні як одиниці зберігання документів можна уникнути великої структури зв'язаних таблиць реляційних сховищ, що дозволяє знизити до мінімуму когнітивне навантаження розробки алгоритмів роботи з документальними NoSQL базами даних.

Прикладами документальних NoSQL баз даних є односекційна база MongoDB, Elasticsearch, Apache CouchDB [13].

Графові бази даних NoSQL. Графові бази даних орієнтовані побудова різних відносин між сутностями і з цієї точки зору є найскладнішою з організації даних. Якщо дані мають високу пов'язаність, наприклад, соціальні мережі, то графові бази даних є найкращим рішенням. У даних базах безпосередньо дані складаються з двох видів компонентів:

- Вузол - компонента, що представляє безпосередньо вузол і пов'язану з ним інформацію;

- ребро – компонента, що визначає відношення між двома ребрами. Ребро може мати напрямок та додаткові атрибути

На відміну від інших видів NoSQL баз даних підтримують дотримання принципів ACID.

Області застосування графових баз даних величезні. Це можуть бути рекомендаційні системи [63], програми в області онлайн та додаткової освіти тощо.

Прикладами графових баз даних є Neo4j, Microsoft Azure Cosmos DB, OrientDB, ArangoDB, Virtuoso [15].

Сегмент баз даних загалом та NoSQL баз даних зокрема є досить динамічним ринком програмних продуктів. Тому з'являються нові пропозиції, які займають лідируючі місця у певних сегментах. На інтернет ресурсі DB-Engines [22] можна ознайомитись із актуальним рейтингом баз даних.

Таким чином, провівши огляд NoSQL баз даних можна зробити такі висновки:

- максимальна швидкодія при виконанні пошукових і агрегуючих запитів мають стовпцеві NoSQL бази даних;

- найкращі показники сегментації та обсягу зберігання інформації притаманні сховищам «ключ-значення»;

- документальні NoSQL бази даних дозволяють створювати чіткі структури різнорідних даних, що забезпечує мінімум когнітивного навантаження під час розробки алгоритмів обробки структур даних;

- графові бази даних мають здатність зберігання даних з високим рівнем зв'язаності.

2.5 Приклад розгортання систем зберігання та аналізу великих даних

Як показує виконаний огляд архітектур та інструментів систем зберігання та аналізу великих даних, існує величезний набір можливих варіантів побудови. Кількість можливих рішень може бути дуже великою, якщо взяти до уваги переліки інструментів, що розробляються (ландшафтів - Landscape) для створення систем зберігання та аналізу великих даних. Прикладом може бути інтернет ресурс, присвячений вивченню рішень у сфері великих даних [14]. Наприклад, рис. 2.8 наведено ландшафт додатків рівня підприємства великих даних. Основна мета додатків Big Data – допомогти компаніям приймати більш інформативні бізнес-

рішення, аналізуючи великі обсяги даних. Ці дані можуть включати журнали веб-серверів, дані про потік інтернет-кліків, контенти в соціальних мережах і звіти про активність, тексти з електронних листів клієнтів, відомості про дзвінки з мобільних телефонів і технічні дані, що знімаються датчиками за технологією IoT. Як зазначалося вище, ці можуть бути структурованими, частково структурованими та неструктурованими.

Рисунок 2.8 – Ландшафт додатків Big Data рівня підприємства

Як бачимо, існує велика кількість інструментів. Тому гостро виникає питання про їх інтеграцію та розгортання на обчислювальних кластерах. Цілком очевидно, що тільки розгортання такої системи може стати найскладнішим завданням. Для

швидкого створення систем зберігання та аналізу великих даних можуть використовуватись спеціальні дистрибутиви — програмні комплекси для розгортання кластера, його моніторингу та управління. На сьогоднішній день існує низка таких дистрибутивів:

1. Платформи розгортання від фірми Hortonworks [16]:

- Hortonworks Data Platform (HDP) – віртуальна машина з повним набором інструментів для пакетної обробки даних;

- Hortonworks DataFlow (HDF) - віртуальна машина з повним набором інструментів для потокової обробки даних;

2. Cloudera Distribution including Apache Hadoop (CDH) [18] – дистрибутив програмного відкритого забезпечення, що містить Apache Hadoop та ключові компоненти, такі як Apache Flume, Apache Hive, Apache Kafka і т.д.

3. MapR Converged Data Platform [26] - єдина платформа, виконана на одній кодовій базі, що поєднує ключові технології: розподілену файлову систему, багатомодельну базу даних NoSQL, механізм публікації/підписки потокових подій, ANSI SQL та широкий набір технологій аналізу даних з відкритим вихідним кодом.

4. Microsoft HDInsight – служба, що працює у хмарі Windows Azure, що дозволяє здійснювати швидкий запуск таких популярних платформ з відкритим кодом як Apache Hadoop, Spark та Kafka.

5. Arenadata Hadoop (ADH) – також дистрибутив, до складу якого входять актуальні стабільні версії найпопулярніших інструментів, такі як Apache Hive, Apache Spark і Apache Atlas.

Так як розглянуті дистрибутиви мають приблизно однакові характеристики в подальших дослідженнях будуть використані віртуальні машини від Hortonworks.

РОЗДІЛ 3 ПРАКТИЧНА РЕАЛІЗАЦІЯ ПРОЦЕСУ ЗБЕРІГАННЯ ТА АНАЛІЗУ BIG DATA

Процес Data Science стосовно великих даних, описаний у першому розділі, повинен спиратися на набір алгоритмів та мов програмування, які підтримують всі етапи, починаючи з підключення до джерел даних до візуалізації результатів досліджень.

У цьому розділі, згідно з метою дослідження, докладно будуть розглянуті алгоритми перших етапів процесу зберігання та аналізу Big Data.

Відомо, що для програмування систем зберігання та аналізу Big Data може використовуватись чимала кількість наборів мов програмування, такі як: Python, Java тощо. У роботі для реалізації елементів систем зберігання та аналізу Big Data буде використана Java з наступних причин:

1. Платформи Java SE і Java EE добре підходять у реалізації масштабованих, навантажених додатків для зберігання та обробки Big Data.

2. Мова Java нараховує ряд загальних можливостей для побудови систем обробки Big Data, найбільш цікавими з яких є:

- базові методи обробки рядків, що включають методи класу String, змінювані клас StringBuilder, потокобезпечний клас StringBuffer, парсер рядків – клас StringTokenizer, а також потужний механізм обробки тексту, заснований на регулярних виразах – Regular Expressions;

- методи функціонного програмування із застосуванням лямбда-виразів, які вводяться у Java, що надаютьзмогу застосовувати сильні та чіткі засоби реалізації додатків;

- потокові можливості колекцій (collections streaming), що полегшує роботу з колекціями даних;

3. Сторонні Java бібліотеки мають можливість надавати можливості більш просту та надійну реалізацію алгоритмів для зберігання та аналізу Big Data:

- Google Guava (https://github.com/google/guava) та Apache Common Collections (https://commons.apache.org/collections), що розширюють технологію роботи з колекціями;

- Apache Commons IO (https://commons.apache.org/io) для реалізації процесу завантаження/вивантаження даних;

- AOL Cyclops-React (https://github.com/aol/cyclops-react) для більш функціонального паралельного потокового обміну;

4. Підтримка онлайнового обчислення Read-Evaluate-Print Loop (REPL) у Java 9. Постачання Java 9 передбачає онлайн термінал jshell, який робить мову Java повністю придатною для використання на етапі дослідницького аналізу даних - Exploratory Data Analysis (EDA).

3.1 Методи отримання, обробки та аналізу великих даних

Наука даних пов'язана з обробкою та аналізом великих обсягів даних і ставить за мету побудови моделей даних, які можна використовуватиме прогнозування чи досягнення конкретної наукової чи бізнес мети [28]. Вибір чи розробка підходу до вирішення конкретного завдання проблеми залежить від характеру самої задачі. Проте можна виділити такі групи високорівневих методів, які у процесі роботи з великими даними: - отримання даних – Acquiring the data;

- очищення даних – Cleaning the data;

- аналіз даних – Analyzing the data.

Розглянемо докладніше високорівневі методи, що входять до зазначених груп.

Отримання даних. Перш ніж дані можуть бути опрацьовані, вони повинні бути отримані. Дані часто зберігаються в різних форматах і надходять з різних джерел даних, які можна розділити на зовнішні і внутрішні.

Очищення даних. Після того, як дані були отримані, їх часто необхідно перетворити на інший формат, перш ніж їх можна буде використовувати. Крім

того, дані повинні бути попередньо оброблені, очищені та наведені у форму, придатну для подальшого аналізу.

Аналіз даних. Етап аналізу даних може бути виконаний з використанням низки загальних методів, таких як:

1. Статистичний аналіз даних. На цьому кроці використовується безліч статистичних методів, що дозволяють отримати інформацію про характер розподілу динних та їх зв'язків. p align="justify"> Серед статистичних методів можна виділити елементарні, такі як розрахунок характеристик випадкових величин - середнього, моди, медіани, квартилів, дисперсії, середнього квадратичного відхилення і т.д. Ряд методів можна віднести до просунутих: перевірка статистичних гіпотез, методи кореляційного та регресійного аналізу.

2. Методи аналізу з урахуванням штучного інтелекту (ШІ). Ці методи можна згрупувати в методи машинного навчання, нейронні мережі та методи глибокого навчання:

- методи машинного навчання дозволяють створювати алгоритми, які можуть навчатися без спеціального програмування для виконання конкретного завдання;

- нейронні мережі побудовані навколо моделей, створених на зразок нейронного зв'язку мозку;

- глибоке навчання намагається виявити вищі рівні абстракції в наборі даних

3. аналіз тексту. Це поширена форма аналізу, яка працює з природними мовами для визначення таких характеристик, як імена людей і місць, взаємозв'язок між частинами тексту та значення тексту.

4. візуалізація даних. Візуалізація є важливим інструментом аналізу. Саме візуалізація даних дозволяє особі приймає рішення скористатися результатами аналізу, що представляється у числовій формі, практично непридатною для сприйняття людиною;

5. Обробка та аналіз відео, зображень та аудіо. Це більш спеціалізована форма аналізу, яка стає все більш поширеною з виявлення більш досконалих методів аналізу та появи більш швидких процесорів.

На додаток необхідно сказати про методи, що безпосередньо не належать до зберігання та аналізу великих даних, але дозволяють створювати високоефективні програми. До таких методів слід зарахувати методи паралельного, розподіленого програмування, а також програмування з використанням графічних процесорів (GPU) [29].

На додаток до сказаного, можна додати, що методи машинного навчання можна розділити такі групи:

- методи контрольованого навчання. Дані методи навчаються на наборах зазначених даних. Для позначення даних потрібна участь людини. Після навчання алгоритми здатні розрізняти результати. Наївний байєсівський класифікатор є представником цієї групи;

- методи неконтрольованого навчання. Ці способи не вимагають позначки даних і намагаються знайти закономірності даних без участі людини. Метод основних компонентів відноситься до методів неконтрольованого навчання;

- методи з частковим контролем можуть працювати як із використанням помічених даних, так і без позначки.

Відповідно до теми дослідження необхідно провести аналіз способів та технологій, що використовуються на перших стадіях процесу зберігання та аналізу великих даних: отримання, очищення та завантаження даних у розподілену файлову систему засобами мови Java.

3.2 Аналіз способів та технології отримання великих даних

Як зазначалося вище, для подальшого аналізу великих даних необхідно використовувати як внутрішні дані компанії, а й зовнішні дані.

Як правило, якщо компанія усвідомлює використання систем зберігання та аналізу великих даних, то внутрішні дані вже зберігаються у різних корпоративних сховищах у узгодженому вигляді. Внутрішні дані компанії зберігаються у базах даних, вітринах даних, сховищах (складах) даних та озерах даних. Як правило, технічний доступ до внутрішніх даних підприємства не становить труднощів,

оскільки здійснюється за допомогою добре відомих технологій, таких як JDBC, JPA (Java Persistence Architecture), Java Connectors або технологій, заснованих на web сервісах (наприклад, Restful), що реалізують з'єднання типу B2B .

Важливість завантаження даних із зовнішніх джерел, а отже, і наявність таких джерел давно усвідомлено у світі. Більше того, дані з цього погляду стають дуже дорогим продуктом.

У нашій країні ведеться планомірна робота зі створення системи ресурсів відкритих даних, особливого імпульсу ця робота набула у зв'язку з виконанням національних проектів загалом та проекту «Цифрова економіка» [8], зокрема.

Серед найважливіших офіційних ресурсів можна зазначити такі:

- портал відкриті дані;

- портал відкритих даних уряду;

- відкриті дані служби державної статистики [17];

Як зарубіжні ресурси відкритих даних можна навести такі приклади:

- офіційний ресурс уряду США, що надає понад 100 000 наборів даних із різних категорій [20]; відкриті дані світового банку (World Bank Group), що охоплюють демографію та достатню кількість економічних показників та показників розвитку з усього світу [11];

- відкриті дані Міжнародного валютного фонду про міжнародні фінанси, ставки боргу, валютні резерви, ціни на сировинні товари та інвестиції [14];

- проект European Data Portal, що містить понад 433800 наборів відкритих даних з усього континенту [14].

Перелік із тридцяти ресурсів відкритих даних наведено в огляді [37].

Крім того, традиційними зовнішніми джерелами зовнішніх даних є соціальні мережі та різні сервіси, такі як YouTube, Wikipedia і т.д.

Для цілей навчання та тестування алгоритмів, що застосовуються для зберігання та аналізу великих даних, можуть бути рекомендовані ресурси, що дозволяють генерувати набори даних. Наприклад, сервіс generatedata.com [25] дозволяє згенерувати набір та зберегти його у різних форматах рис. 3.1.

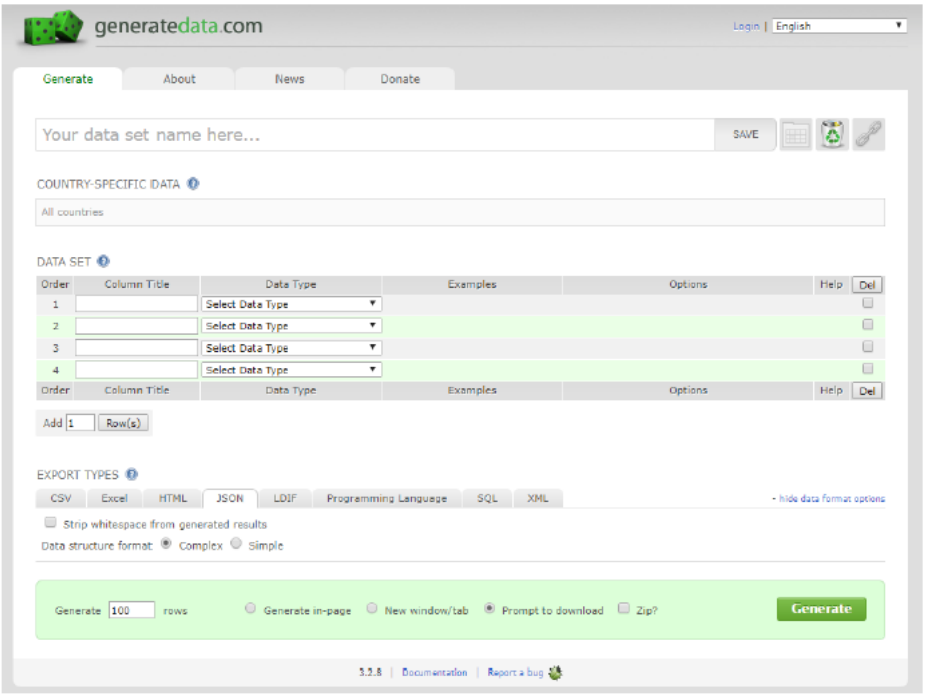


Рисунок 3.1 – Сервіс generatedata.com

Розглянемо способи отримання даних із зовнішніх джерел докладніше. На думку автора можна розділити способи отримання зовнішніх даних на два типи:

1. Методи, що ґрунтуються на аналізі контенту сайтів.

2. Методи, що ґрунтуються на використанні спеціальних API інтернет ресурсів.

До методів першого типу можна віднести веб-скріпінг (зіскоблювання, scraping) і веб-сканування (crawling). Ми можемо отримати доступ до даних з Інтернету, завантаживши певні файли або за допомогою процесу, відомого як веб-скріпінг, який включає вилучення вмісту веб-сторінки. Другий метод - веб-сканування, заснований на розробці програми, яка досліджує веб-сайт на можливість застосування його контенту в дослідженні та слідують за посиланнями для аналізу інших потенційно релевантних сторінок.

Розробники великих інтернет ресурсів, таких як соціальні мережі та YouTube, враховуючи негативні ефекти від масового використання на їх сайтах алгоритмів скріпінгу та сканування, розробили спеціальне API, через яке можна отримувати необхідний для дослідження контент. Що й зумовило появу методів другого типу.

Веб-скріпінг. Веб-скріпінг – це процес отримання даних і метаданих з веб сторінки [29]. За своєю суттю для здійснення веб-скріпінгу необхідно реалізувати HTML парсер (веб-скебок), оскільки результат http запиту представляється зазвичай як HTML коду.

Аналіз технологій веб-скріпінгу стосовно мови Java показав, що найкращим рішенням є використання відкритої Java бібліотеки Soup, що є HTML парсером. Парсер Súp дозволяє також проводити пошук певних елементів HTML розмітки і їх використовувати для подальшого аналізу, або, навпаки, фільтрувати ці елементи, проводячи очищення вмісту HTML сторінки.

Розглянемо приклад JSoup парсингу рис. 3.2 статті вікіпедії, присвяченої великим даним [30]. Спочатку через súp з'єднання створюється документ, у який завантажується вміст HTML сторінки за вказаною адресою. Далі, використовуючи різні методи об'єкта document, отримує такі елементи, як назва, тіло, список гіперпосилань і т.п. Далі можна, наприклад, обробляти тіло статті, що є стоком, використовуючи розбивку на лексеми за допомогою стандартних класів (String, StringTokenizer), технології регулярних виразів (Regular Expressions) або сторонні бібліотеки Java.

```
import java.io.IOException;
import java.util.List;
import org.jsoup.Jsoup;
import org.jsoup.nodes.Document;
import org.jsoup.select.Elements;

public class JsoupParser {

    //https://ru.wikipedia.org/wiki
    private static String url =
"https://ru.wikipedia.org/wiki/%D0%91%D0%BE%D0%BB%D1%8C%D1%88%D0%B8%D0%B5
_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D0%B5";

    public static void main(String[] args) {
        try {
            Document document = Jsoup.connect(url).get();
            //get a title
            String title = document.title();
            System.out.println("Назва статті: " + title);
            //get a body
            Elements body = document.select("body");
            System.out.println("Зміст: " + body.text());
            //get links
            Elements links = document.select("a[href]");
            //System.out.println(links.text());
            List<String> linksList = links.eachText();
            int linksNum = linksList.size();
            System.out.println("Число гіперпосилань: " + linksNum + "\n
Перші п'ять:");
            for(int i=0; (I < 5) & (I < linksNum); i++) {
                System.out.println(linksList.get(i));
            }
        } catch (IOException ex) {
            ex.printStackTrace();
```

Рисунок 3.2 – Код JSoup парсера, що реалізує веб-скріпінг сторінки вікіпедії

Приклад роботи JSoup парсера наведено рис. 3.3

Веб-сканування. Веб-сканування (Web crawling) – це процес обходу безлічі взаємопов'язаних веб-сторінок та отримання відповідної інформації з цих сторінок.

Великі дані — Вікіпедія
— big data, [ˈbɪg ˈdeɪtə]) — позначення структурованих та неструктурованих даних величезних об'ємів та значного різноманіття, що ефективно обробляються горизонтально масштабованими програмними інструментами, що з'явилися наприкінці 2000-х років та альтернативних традиційним системам управління базами даних та рішенням класу Business Intelligence [2] [3]. У широкому значенні про «великі дані» говорять як про соціально-економічний феномен, пов'язаний з появою технологічних можливостей аналізувати величезні масиви даних, в деяких проблемних областях — весь світовий обсяг даних, і з цього трансформаційних наслідків [4]. Як визначальних характеристик великих даних традиційно виділяють «три V»: обсяг (англ. volume, у сенсі величини фізичного обсягу), швидкість (velocity у сенсах як швидкості приросту, і необхідності високошвидкісної обробки та отримання результатів), різноманіття (variety, у сенсі можливості одночасної обробки різних типів структурованих та напівструктурованих даних) [5] [6]; надалі виникли різні варіації та інтерпретації цієї ознаки [⇨].
…
некомерційної організації Wikimedia Foundation, Inc. Політика конфіденційності Опис Вікіпедії Відмова від відповідальності Зв'яжіться з нами Розробники Угода про cookie Мобільна версія
Число гіперпосилань: 435
Перші п'ять:
Skip to navigation
Перейти до пошуку
англ.
ˈbɪg ˈdeɪtə
неструктурованих даних

Рисунок 3.3 – Парсинг сторінки статті вікіпедії, присвяченої великим даним

Процес веб-сканування здійснюється шляхом обробки поточної сторінки та наступного переходу за посиланнями на сторінці. Веб-сканування корисне, коли незважаючи на наявність доступних наборів даних, все ж таки може знадобитися збір даних безпосередньо з Інтернету. Наприклад, деякі джерела, такі як сайти новин, стрічки повідомлень у соціальних мережах постійно оновлюються, і час від часу їх необхідно переглядати.

Веб-сканер – це програма, яка відвідує різні сайти та збирає інформацію. Процес веб-сканування складається із серії кроків:

1. Визначення URL для відвідування.

2. Отримання сторінки.

3. Розбір сторінки.

4. Вилучення відповідного контенту.

5. Вилучення відповідних URL для подальшого відвідування.

Вказана послідовність кроків повторюється для кожного відвіданого URL-адреси.

Однак, існує кілька проблем, які необхідно враховувати під час реалізації веб-сканування:

- важливість сторінки: немає потреби обробляти нерелевантні сторінки;

- обробка конкретного контенту: наприклад, оброблятиметься лише HTML контент, а перехід за посиланнями на зображення не здійснюватиметься;

- виключення нескінченного аналізу (Spider traps): необхідно оминати сайти, які можуть призвести до нескінченної кількості запитів. Вказана ситуація може відбуватися з сторінками, що динамічно генеруються, коли один запит веде до іншого;

- виняток повторення: необхідно виключити сканування однієї і тієї ж сторінки більше одного разу;

- "ввічливість" сканера: не слід робити надмірної кількості запитів на сайт. Для цього потрібно переглядати файли robot.txt, які вказують, які частини сайту не слід сканувати.

Як випливає з наведеної інформації, створення веб-сканера з нуля є складним завданням. Тому можна використовувати один веб-сканер з відкритим вихідним кодом, серед яких можна вказати:

- crawler4j (https://github.com/yasserg/crawler4j);

- Nutch (http://nutch.apache.org);

- WebSPHINX (http://www.cs.cmu.edu/~rcm/websphinx);

- JSpider (http://j-spider.sourceforge.net);

- Heritrix (https://webarchive.jira.com/wiki/display/Heritrix).

Розглянемо побудову веб-сканера з урахуванням бібліотеки crawler4j. Для цього необхідно розробити два класи. Перший клас TltsuCrawler реалізовуватиме

логіку сканера, який обстежує контент сайту університету [23]. Другий клас являє собою контролер або додаток. Діаграма зазначених класів представлена рис. 3.4.

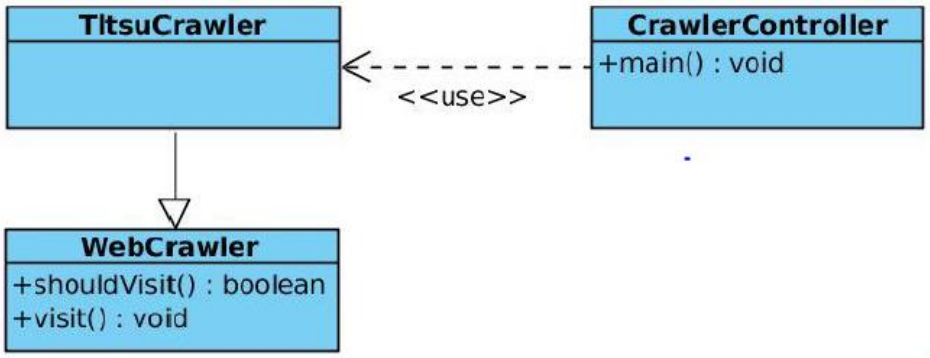


Рисунок 3.4 – Діаграма класів реалізації веб-сканера

Клас веб-сканера перевантажує два методи бібліотечного класу WenCrawler:shoudVisit() (визначає фільтр відвідування веб-сторінок) та visit() (визначає алгоритм обробки сторінки). Код методу visit(), зображений рис. 3.5, використовує регулярне вираз для відсіву посилань, які ведуть ресурсні (css), мультимедійні (gif, png, …) і архівні (zip, gz) файли, які у цьому прикладі пропускаються для відбору.

```
public class TltsuCrawler extends WebCrawler {
    private final static Pattern FILTERS =
Pattern.compile(".*(\\.(css|js|gif|jpg|png|mp3|mp4|zip|gz))$");

    /**
     * Specify whether the given url should be crawled
     */

    @Override
    public boolean shouldVisit(Page referringPage, WebURL url) {
        System.out.println("shouldVisit: " + url.getURL().toLowerCase());

        String href = url.getURL().toLowerCase();
        boolean result = !FILTERS.matcher(href).matches();
        if(result) {
         System.out.println("Visited " + url.getURL().toLowerCase());
        } else {
         System.out.println("Not visited " +
                url.getURL().toLowerCase());
        }
        return result;
    }
```

Рисунок 3.5 – Фрагмент коду методу фільтрації сторінок веб-сканера

Код методу visit() веб-сканера представлений рис.3.6. Цей метод спершу перевіряє, що поточна сторінка має розмітку HTML і може бути розібрана HTML парсером. Далі сторінка аналізується за низкою параметрів, які можна використовуватиме подальшого аналізу. У прикладі запис даних йде на консоль, хоча можна організувати виведення даних у файл, для подальшої попередньої обробки і завантаження в систему зберігання великих даних.

Фрагменти роботи веб-сканера двома обчислювальними потоками, на глибину трьох рівнів і максимальною кількістю сторінок, що дорівнює 50, наведено на рис. 3.7.

```
    /**
     * This function is called when a page is fetched and ready
     * to be processed by the program.
     */
    @Override
    public void visit(Page page) {
        String url = page.getWebURL().getURL();
        System.out.println("URL: " + url);

        if (page.getParseData() instanceof HtmlParseData) {
            HtmlParseData htmlParseData = (HtmlParseData)
page.getParseData();
            String text = htmlParseData.getText(); //extract text
from page
            String html = htmlParseData.getHtml(); //extract html
from page
            Set<WebURL> links = htmlParseData.getOutgoingUrls();

            System.out.println("----------------------------------
----------------------");
            System.out.println("Page URL: " + url);
            System.out.println("Text length: " + text.length());
            System.out.println("Html length: " + html.length());
            System.out.println("Number of outgoing links: " +
links.size());
            System.out.println("----------------------------------
----------------------");
        }
    }
```

Рисунок 3.6 – Фрагмент коду методу обробки веб-сканером HTML сторінки

Відповідно до наведеного лога роботи веб-сканера, було пройдено ряд HTML сторінок і по них сканер виділив необхідну інформацію. Лог також показує, що ряд ресурсів не був просканований (на рисунку файл extension.min.js), як той, що не

пройшов фільтрацію. Після завершення роботи всіх обчислювальних потоків сканера та очікування в 10 секунд програма завершила свою роботу.

Доступ до даних через API Інтернет ресурсів. Як зазначалося вище, ряд інтернет ресурсів є сховищами величезної та різноманітної інформації, яка цікава для аналізу даних великій кількості сторін.

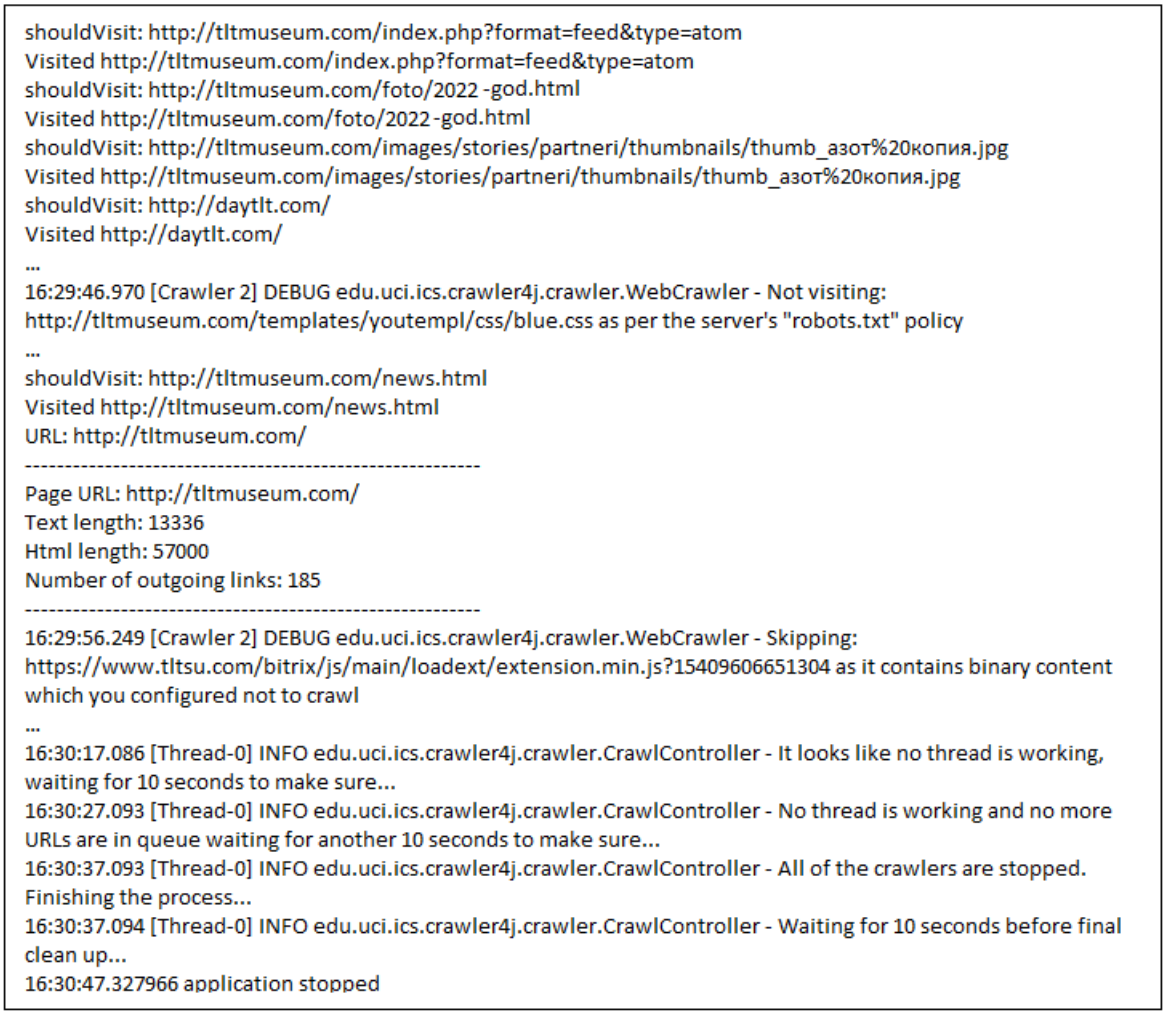


Рисунок 3.7 – Результат роботи веб-сканера

Якщо для збирання та подальшого аналізу інформації використовувати раніше розглянуті способи веб-скріпінгу та веб-сканування, то зазначені ресурси можуть бути заблоковані на тривалий час такими запитами на отримання даних. Тому ресурси соціальних мереж, сховища інформації та інші ресурси, що спеціалізуються на наданні даних у тому числі і для їх аналізу, розробляють

спеціалізоване API за допомогою якого можливо не тільки шукати і завантажувати необхідні дані, але і працювати з метаінформацією.

Розглянемо доступом до відкритих даних на порталі відкритих даних країн. Доступ до відкритих даних на порталі окрім безпосереднього завантаження з веб-сторінок можливий двома способами: використання спеціального API та через інтерактивні SPARQL-запити. Розглянемо доступом до даних через спеціалізований API [30]. Для роботи з API потрібно отримати особистий ключ, доступний після реєстрації на Порталі. Отримати дані можна у двох форматах JSON та XML. Оскільки формат JSON найбільше підходить для використання в системах зберігання та аналізу великих даних, то ми будемо використовувати саме цей формат даних.

На рис. 3.8 наведено фрагмент коду доступу до порталу відкритих даних України та запит на перелік всіх наявних наборів. Спочатку формується запит до набору даних відповідно до правил порталу (у коді прихований реальний ключ доступу). Потім за допомогою бібліотеки IOUtils apache.commons.io формується масив JSON. Після чого визначається його довжина і відбувається виведення трохи більше п'яти елементів.

```
public class Main {

    public static void main(String[] args) {
        try {
            URL url = new URL ("https://data.gov/api/dataset/?" +
            "access_token=XXX");

            JSONArray datasets = new JSONArray(
                IOUtils.toString(url, Charset.forName("UTF-8")));
            int numRec = datasets.length();
            System.out.println(numRec);
            JSONObject e;
            for (int i=0; (i < 5) && (i < numRec); i++) {
                e = (JSONObject) datasets.get(i);
                System.out.println(e.toString());
            }
        } catch (MalformedURLException ex) {
            ex.printStackTrace();
        } catch (IOException ex) {
            ex.printStackTrace();
        }
```

Рисунок 3.8 – Отримання доступних наборів даних на порталі відкритих даних України

Приклад виведення наведено на рис. 3.9. Отримано лише 10 000 джерел. Ця кількість джерел є максимальною для простих запитів.

Модифікуючи запити до порталу, можна звузити діапазон пошуку, наприклад знайти набори даних, що відносяться до розділу «Освіта/Education» рис. 3.10.

10000
{"identifier":"8912001599-muzikalka","organization":"8912001599","topic":"Education","organization_name":" Заклади естетичного виховання. Центри культури\"Київська дитяча школа мистецтв №8\":"школа мистецтв м. Київ"}

{"identifier":"8901012609-regprojects","organization":"8901012609","topic":"Government","organization_name":"Апарат мера міста Києва","title":"Регіональні проекти, апаратом мера столиці}

{"identifier":"8901017727-reestrteplo","organization":"8901017727","topic":"Economics","organization_name":"Департамент тарифної політики, енергетики та житлово-комунального комплексу title":"Реєстр організацій, яким встановлено плату за технологічне приєднання до системи теплопостачання"}

{"identifier":"8901017283-poiskoviki","organization":"8901017283","topic":"Government","organization_name":"Департамент молодіжної політики та туризму міста Києва","title":"Перелік пошукових загонів міста Києва"}

{"identifier":"8901003315-culturalheritage","organization":"8901003315","topic":"Culture","organization_name":"Адміністрація муніципальної освіти місто Києва","title":" Об'єкти культурної спадщини міста Києва"}

Рисунок 3.9 – Перші п'ять доступних наборів даних на порталі відкритих даних

```
URL url = new URL ("https://data.gov.ru/api/dataset/?" +
                   "topic=Education&" +
                   "access_token=XXX");
```

Рисунок 3.10 – Запит на доступні набори даних на порталі відкритих даних України з розділу «Освіта»

На рис.3.11 наведено приклад відгуку порталу. У розділі «Освіта» було знайдено 1201 набір та показано перші п'ять наборів.

1201
{"identifier":"7710539135-DO","organization":"7710539135","topic":"Education","organization_name":"Міністерство освіти і науки України","title":"Відомості про функціонування системи дошкільної освіти"}

{"identifier":"7710539135-trud","organization":"7710539135","topic":"Education","organization_name":"Міністерство освіти та науки України","title":"Працевлаштування випускників закладів професійної освіти "}

{"identifier":"7710539135-ob_obr","organization":"7710539135","topic":"Education","organization_name":"Міністерство освіти і науки України","title":"Відомості про функціонування системи загального освіти"}

{"identifier":"7710539135-SPO","organization":"7710539135","topic":"Education","organization_name":"Міністерство освіти та науки України","title":"Відомості про функціонування системи середнього професійної освіти"}

{"identifier":"7710539135-podved","organization":"7710539135","topic":"Education","organization_name":"Міністерство освіти і науки України","title":"Перелік підвідомчих організацій Міносвіти України"}

Рисунок 3.11 – Перші п'ять доступних наборів даних з розділу освіти на порталі відкритих даних України

Аналізуючи отримані списки, можна вибрати набір для завантаження. Код, наведений на рис. 3.8-3.11, показує алгоритм скачування набору №7710539135-trud, що має назву «Працевлаштування випускників закладів професійної освіти». Відповідно до опису API порталу необхідно визначити версію набору. Ми будемо використовувати останню версію. Алгоритм знаходження останньої версії шуканого набору наведено рис. 3.12. Спочатку формується url запиту, яким отримуємо набір JSON об'єктів, відповідальних за версії. Кожна версія набору відкритих даних має єдину властивість created – унікальну дату версії набору відкритих даних. При зверненні до першого елемента ми отримуємо останню версію. Так у нашому випадку було отримано результат у форматі:

```
public class Main {

    static final String ACCESS_TOKEN = "/?access_token=XXX";
    static final String ACCESS_URL =
            "https://data.gov.ru/api/json/dataset" +
            "/7710539135-trud" +
            "/version";

    public static void main(String[] args) {
        try {
            // отримуємо набір версій
            //url для отримання набору версій
            URL url = new URL (ACCESS_URL + ACCESS_TOKEN);
            JSONArray versions = new JSONArray(
                IOUtils.toString(url, Charset.forName("UTF-8")));
            int numRec = versions.length();
            JSONObject lastVersion = (JSONObject) versions.get(0);
            String created = lastVersion.getString("created");
…
            } catch (MalformedURLException ex) {
            ex.printStackTrace();
        } catch (IOException ex) {
            ex.printStackTrace();
        }
```

Рисунок 3.12 – Визначення останньої версії набору даних, що запитується.

Далі необхідно отримати інформацію за версією набору. Алгоритм отримання інформації представлено рис. 3.13. Спочатку модифікується URL під версію набору. Після цього програма звертається до порталу та отримує список JSON об'єктів, які відповідають за інформацію про набір, яка, у нашому випадку, подається у такому вигляді:

{"provenance":"Внесено зміни до опису набору даних. Опубліковано додаткову інформацію",
"created":"20150609T121501",
"format":"csv",
"source":"https://data.gov.ru/sites/default/files/opendata/7710539135-trud/data-2015-06-09T12-15-01-structure-2013-07-03T00-00-00 .csv",
"source_id":"832895","updated":"20150609T121501"}

```
//отримуємо адресу останньої версії набору даних
////url для отримання інформації про набір даних
    url = new URL (ACCESS_URL +
            "/" + created +
            ACCESS_TOKEN);
    JSONArray dataset = new JSONArray(
            IOUtils.toString(url, Charset.forName("UTF-8")));
    JSONObject data = (JSONObject) dataset.get(0);
    String dataURI = data.getString("source");
```

Рисунок 3.13 – Отримання системної інформації щодо версії набору даних

Далі відбувається виділення імені файлу і, використовуючи канали обміну, відбувається запис даних з порталу в файл на диску рис. 3.14. У нашому випадку файл має ім'я:

data-2021-06-09T12-15-01-structure-2019-07-03T00-00-00.csv

```
//виділення ім'я файла з url
String fname = dataURI.substring(
           dataURI.lastIndexOf('/')+1, dataURI.length());
//transfer bytes between 2 Channels
//without buffering them into the application memory
//read the file from our URL
ReadableByteChannel readableByteChannel =
        Channels.newChannel(new URL(dataURI).openStream());
//The bytes read from the ReadableByteChannel
//will be transferred to a FileChannel corresponding
//to the file that will be downloaded
try (FileOutputStream fileOutputStream = new
        FileOutputStream(fname);
    FileChannel fileChannel = fileOutputStream.getChannel()) {
    /use the transferFrom() method from the
    //ReadableByteChannel class
    //to download the bytes from the given URL to our FileChannel
    fileOutputStream.getChannel()
        .transferFrom(readableByteChannel, 0, Long.MAX_VALUE);
}
```

Рисунок 3.14 – Використання каналів для збереження набору даних на диск

Після виконання коду файл може бути завантажений у систему зберігання та аналізу великих даних для подальшої обробки. Фрагмент вмісту файлу наведено на в табл.3.1.

Таблиця 3.1 - Фрагмент вмісту набору даних порталу відкритих даних

region	npo	ntu	our	ppa	tu	all_vyp
Київська область	4738	720	1050	5006	18292	29806
Київська область	60	10	6	8	114	198
Київська область	122	4	4	26	120	276
Київська область	428	200	78	180	2064	2950
Київська область	58	2	14	6	38	118
Київська область	378	24	130	136	1270	1938
Київська область	436	62	112	48	1026	1684
Житомирська область	3644	922	674	6748	11884	23872
Житомирська область	8	14	8	32	62	124
Житомирська область	98	18	8	32	110	266
Житомирська область	586	174	112	556	2422	3850
Житомирська область	20	6	4	14	18	62
Житомирська область	186	44	126	98	824	1278
Житомирська область	216	52	56	768	732	1824
Житомирська область	4	6	6	24	30	70
Житомирська область	0	0	0	0	0	0
Житомирська область	68	20	0	278	216	582
Житомирська область	334	58	2	870	604	1868
Житомирська область	0	0	0	0	0	0
Житомирська область	0	0	0	0	0	0
Житомирська область	0	0	0	0	0	0
Житомирська область	210	74	4	1304	890	2482
Житомирська область	14	14	0	10	24	62
Житомирська область	22	0	0	56	52	130

3.3 Дослідження прикладу очищення даних

Реальні дані часто є брудними – містять різного роду помилки – і неструктурованими, тому перед використанням вони мають бути перероблені. Дані можуть містити помилки, мати записи, що повторюються, існувати в неправильному форматі або бути суперечливими. Процес вирішення цих типів проблем називається очищенням даних. Очищення даних також називається обробкою, масуванням, зміною форми або обробкою даних [28]. Злиття даних, коли дані з кількох джерел об'єднуються, часто вважається очищенням даних.

Очищення даних є невід'ємним етапом роботи з великими даними, оскільки будь-який аналіз, що базується на неточних даних, може призвести до помилкових

результатів. Таким чином, необхідно переконатись, що дані, з якими ми працюємо, є якісними даними. Якість даних включає:

- придатність (Validity) вимога забезпечення правильної форми чи структури даних;

- точність (Accuracy): значення даних дійсно є репрезентативними для набору даних;

- повнота (Completeness): немає відсутніх елементів;

- узгодженість (Consistency): зміни у даних синхронізовані;

- однорідність (Uniformity): використовуються самі одиниці виміру.

Можна вказати кілька методів та інструментів, які використовуються для очищення даних:

- обробка різних типів даних;

- очищення та маніпулювання текстовими даними; заповнення даних, що відсутні;

- перевірка даних.

Для очищення даних з використанням мови Java, як і раніше, можуть застосовуватися як стандартні технології, так і сторонні бібліотеки.

Одним із перших завдань, як правило, є перевірка коректності форматів вхідних даних, які часто можуть надходити у вигляді:

- CSV наборів;

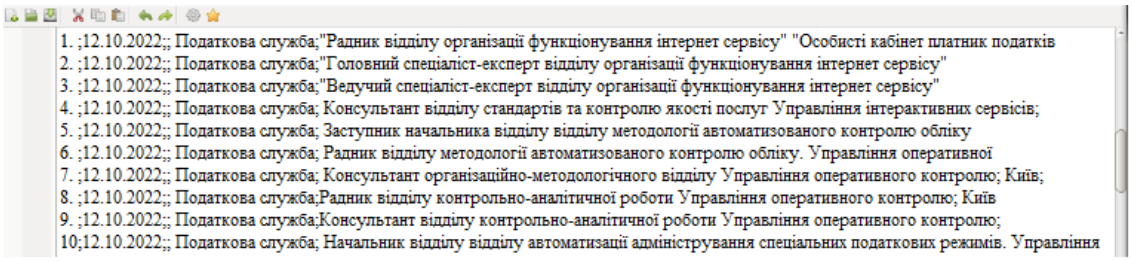
- таблиць різного виду;

- файлів у форматі PDF;

- файли у форматі JSON.

Окремим завданням є перевірка файлів мультимедіа.

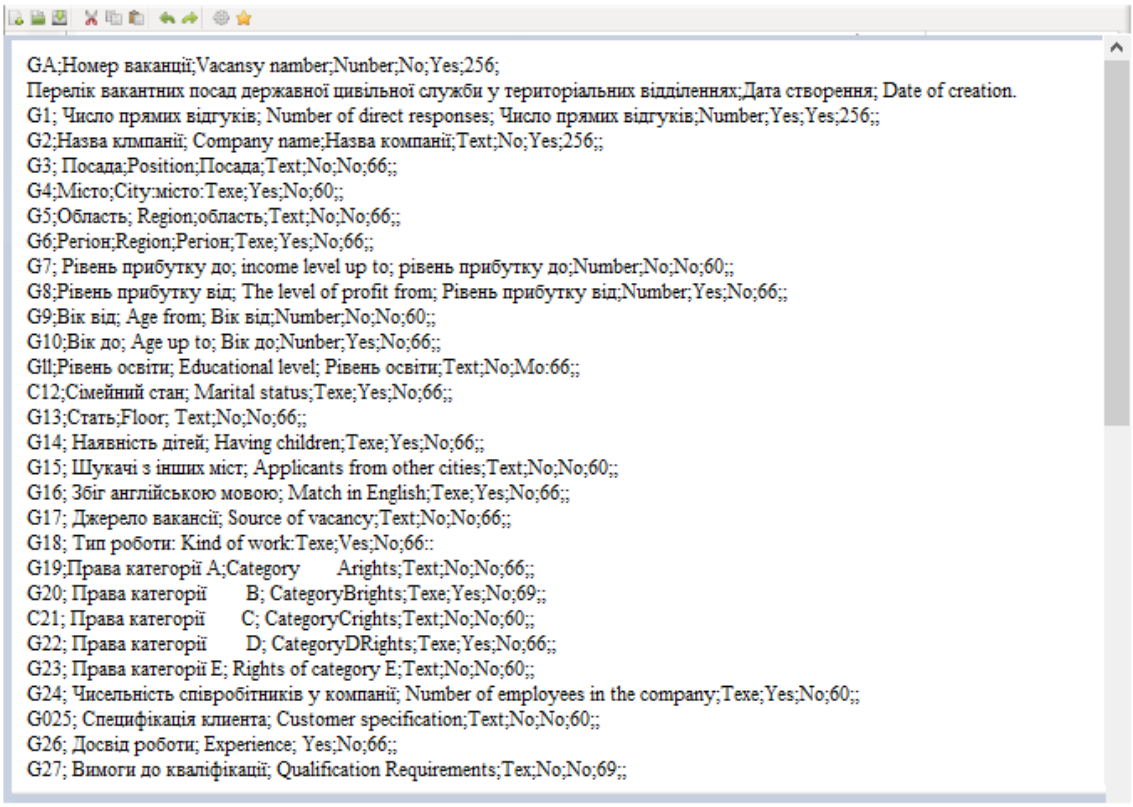
При використанні порталів відкритих даних зазвичай доводиться мати справу з форматом CSV. Ці файли часто отримані шляхом автоматичного вивантаження з інформаційних систем, однак вони потребують очищення. Розглянемо аналіз структури відкритих даних із сайту університету [26]. Як приклад розглянемо «Перелік вакантних посад університету. Фрагмент даних із зазначеного набору представлений на рис. 3.15. а фрагмент опису структури – на рис. 3.16.



```
1. ;12.10.2022;; Податкова служба;"Радник відділу організації функціонування інтернет сервісу" "Особисті кабінет платник податків
2. ;12.10.2022;; Податкова служба;"Головний спеціаліст-експерт відділу організації функціонування інтернет сервісу"
3. ;12.10.2022;; Податкова служба;"Ведучий спеціаліст-експерт відділу організації функціонування інтернет сервісу"
4. ;12.10.2022;; Податкова служба; Консультант відділу стандартів та контролю якості послуг Управління інтерактивних сервісів;
5. ;12.10.2022;; Податкова служба; Заступник начальника відділу відділу методології автоматизованого контролю обліку
6. ;12.10.2022;; Податкова служба; Радник відділу методології автоматизованого контролю обліку. Управління оперативної
7. ;12.10.2022;; Податкова служба; Консультант організаційно-методологічного відділу Управління оперативного контролю; Київ;
8. ;12.10.2022;; Податкова служба;Радник відділу контрольно-аналітичної роботи Управління оперативного контролю; Київ
9. ;12.10.2022;; Податкова служба;Консультант відділу контрольно-аналітичної роботи Управління оперативного контролю;
10;12.10.2022;; Податкова служба; Начальник відділу відділу автоматизації адміністрування спеціальних податкових режимів. Управління
```

Рисунок 3.15 – Фрагмент даних із сайту

Потрібно перевірити набір на придатність його завантаження в систему зберігання та аналізу великих даних, а за наявності помилок вивести звіт про помилки.



```
GA;Номер ваканцій;Vacansy namber;Nunber;No;Yes;256;
Перелік вакантних посад державної цивільної служби у територіальних відділеннях;Дата створення; Date of creation.
G1; Число прямих відгуків; Number of direct responses; Число прямих відгуків;Number;Yes;Yes;256;;
G2;Назва клмпанії; Company name;Назва компанії;Text;No;Yes;256;;
G3; Посада;Position;Посада;Text;No;No;66;;
G4;Місто;City:місто:Texe;Yes;No;60;;
G5;Область; Region;область;Text;No;No;66;;
G6;Регіон;Region;Регіон;Texe;Yes;No;66;;
G7; Рівень прибутку до; income level up to; рівень прибутку до;Number;No;No;60;;
G8;Рівень прибутку від; The level of profit from; Рівень прибутку від;Number;Yes;No;66;;
G9;Вік від; Age from; Вік від;Number;No;No;60;;
G10;Вік до; Age up to; Вік до;Nunber;Yes;No;66;;
Gll;Рівень освіти; Educational level; Рівень освіти;Text;No;Mo:66;;
C12;Сімейний стан; Marital status;Texe;Yes;No;66;;
G13;Стать;Floor; Text;No;No;66;;
G14; Наявність дітей; Having children;Texe;Yes;No;66;;
G15; Шукачі з інших міст; Applicants from other cities;Text;No;No;60;;
G16; Збіг англійською мовою; Match in English;Texe;Yes;No;66;;
G17; Джерело вакансії; Source of vacancy;Text;No;No;66;;
G18; Тип роботи: Kind of work:Texe;Ves;No;66::
G19;Права категорії A;Category      Arights;Text;No;No;66;;
G20; Права категорії      B; CategoryBrights;Texe;Yes;No;69;;
C21; Права категорії      C; CategoryCrights;Text;No;No;60;;
G22; Права категорії      D; CategoryDRights;Texe;Yes;No;66;;
G23; Права категорії E; Rights of category E;Text;No;No;60;;
G24; Чисельність співробітників у компанії; Number of employees in the company;Texe;Yes;No;60;;
G025; Специфікація клиента; Customer specification;Text;No;No;60;;
G26; Досвід роботи; Experience; Yes;No;66;;
G27; Вимоги до кваліфікації; Qualification Requirements;Tex;No;No;69;;
```

Рисунок 3.16 – Опис структури даних у форматі CSV

Незважаючи на те, що парсинг CSV файлів здійснимо з використанням стандартних бібліотек, таких як бібліотеки введення-виведення, включаючи NIO2, а також засоби розбивки рядків на лексеми та стандартні перетворення типів та використанням класів-оберток, на думку автора, найбільш прийнятним способом є використання сторонніх бібліотек. У розглянутій задачі найбільш популярною

бібліотекою є OpenCSV [19]. Наведемо приклад коду, що аналізує алгоритм перевірки набору на коректність даних. На рис. 3.17 наведено фрагмент коду програми. Спочатку на основі файлового буферизованого потоку створюється екземпляр класу CsvToBean, що є реалізацією відображення даних з CSV файлу на об'єкти класу Vacancy. Для цього використовується екземпляр класу-побудовника CsvToBeanBuilder, який, використовуючи потокові методи, проводить додаткові налаштування, такі як вибір роздільника, вказівку видаляти кінцеві прогалини тощо.

```
public class VacancyReader {
    static final String FILE_NAME = "data.csv";

    public static void main(String[] args) {
            try (   Reader reader =
Files.newBufferedReader(Paths.get(FILE_NAME))   ) {
                CsvToBean<Vacancy> csvToBean =
                        new CsvToBeanBuilder<Vacancy> (reader).
                            withType(Vacancy.class).
                            withSeparator(';').
                            withIgnoreLeadingWhiteSpace(true).
                            build();
                Iterator<Vacancy> it = csvToBean.iterator();
                while (it.hasNext()) {
                    Vacancy vacancy = it.next();
                    System.out.println(vacancy);
                }
            } catch (FileNotFoundException e) {
                e.printStackTrace();
            } catch (IOException e) {
                e.printStackTrace();
            }
```

Рисунок 3.17 – Фрагмент коду програми для читання CSV файлу та відображення даних в об'єктах Java

Після ініціалізації потоку ініціалізується ітератор, який проходить у всьому наборі. При цьому кожен рядок CSV файлу відображається відповідно до заголовків стовпців (перший рядок) в екземплярах класу Vacancy рис. 3.18.

```
public class Vacancy {
    // номер ваканції
    @CsvBindByName (column = "GA", required = true)
    @CsvNumber("##0")
    private int number;

    // місто
    @CsvBindByName (column = "G4", required = true)
    @Size(max = 60, message = "Довжина назви міста не може бути
 більше 60")

    private String city;

    // Рівень доходу від
    @CsvBindByName (column = "G7", required = true)
    @CsvNumber("####0")
    private int incomeLevelMin;

    //Рівень доходу до
    @CsvBindByName (column = "G8", required = true)
    @CsvNumber("####0")
    private int incomeLevelMax;
```

Рисунок 3.18 – Клас-прототип CSV даних набору

Клас Vacancy є компонентом Java, поля якого анотовані відповідно до бібліотеки OpenCSV і стандартної технології Bean Validation. Основу відображення створюють анотації @CsvBindByName, в яких можна вказати дані якого стовпця зв'язуються з конкретним полем, а також вказати на обов'язковість поля, правила форматування CSV для відображення і т.д.

Результат запуску програми, наведений рис. 3.19, дозволяє зробити такі выводы:

- перші 60 рядків є коректними;

- на 61-му рядку сталася помилка відображення в об'єкт класу Vacancy.

Для подальшого аналізу необхідно просканувати вест набір даних та вивести звіт про помилки. Для цього було розроблено додатковий додаток, код якого представлений на рис. 3.20. Аналогічно наведеному прикладу, на основі фалового потоку і парсера CSV створюється вхідний потік для читання CSV у форматі масиву рядків. Звертаючись до конкретного елементу масиву (змінна rec) можна отримати рядкове представлення даних конкретного стовпця. Знаючи необхідну структуру даних, можна проконтролювати помилки і вивести їх опис для подальшого аналізу.

```
number: 1| city :Київ| LevelMin :27000 | LevelMax :30000
number: 2| city :Київ| LevelMin :21000 | LevelMax :23000
number: 3| city :Київ| LevelMin :27000 | LevelMax :30000
number: 4| city :Київ| LevelMin :27000 | LevelMax :30000
number: 5| city :Київ| LevelMin :28000 | LevelMax :31000
number: 6| city :Київ| LevelMin :27000 | LevelMax :30000
number: 7| city :Київ| LevelMin :27000 | LevelMax :30000
number: 8| city :Київ| LevelMin :27000 | LevelMax :30000
number: 9| city :Київ| LevelMin :28000 | LevelMax :31000
…
number: 58| city :Київ| LevelMin :30000 | LevelMax :34000
number: 59| city :Київ| LevelMin :29000 | LevelMax :32000
number: 60| city :Київ| LevelMin :28000 | LevelMax :31000
Exception в thread "main" java.lang.RuntimeException: com.opencsv.exceptions.CsvRequiredFieldEmptyException:
Field 'number' is mandatory but no value was provided.
   at com.opencsv.bean.concurrent.ProcessCsvLine.run(ProcessCsvLine.java:101)
   at
com.opencsv.bean.CsvToBean$CsvToBeanIterator.readLineWithPossibleError(CsvToBean.java:551)
   at com.opencsv.bean.CsvToBean$CsvToBeanIterator.readSingleLine(CsvToBean.java:571)
   at com.opencsv.bean.CsvToBean$CsvToBeanIterator.next(CsvToBean.java:591)
   at edu.bigdata.VacancyReader.main(VacancyReader.java:25)
Caused by:  com.opencsv.exceptions.CsvRequiredFieldEmptyException: Field 'number' is mandatory but no value was
provided.
   at com.opencsv.bean.AbstractBeanField.setFieldValue(AbstractBeanField.java:163)
   at com.opencsv.bean.AbstractMappingStrategy.setFieldValue(AbstractMappingStrategy.java:449)
   at
com.opencsv.bean.AbstractMappingStrategy.populateNewBean(AbstractMappingStrategy.java:317)
   at com.opencsv.bean.concurrent.ProcessCsvLine.processLine(ProcessCsvLine.java:132)
   at com.opencsv.bean.concurrent.ProcessCsvLine.run(ProcessCsvLine.java:85)
... 4 більше
```

Рисунок 3.19 – Результат перетворення набору даних

Результат роботи програми, що виводить звіт про помилки в CSV файлі наведено на рис.3.22, з якого випливає, що з 61 рядка йдуть рядки, в яких відсутні дані рис. 3.23. Таким чином, необхідне ручне або автоматичне коригування набору.

```
try (Reader in = Files.newBufferedReader(Paths.get(FILE_NAME))) {
    CSVParser csvParser =
            new CSVParserBuilder().
                withSeparator(';').
                withIgnoreLeadingWhiteSpace(true).
                build();
    CSVReader reader =
            new CSVReaderBuilder(in).
                withCSVParser(csvParser).
                withSkipLines(1). //pass headers line
                build() ;

    int linesReaded = 1;
    int vacancyNum;
    String vacancyCity;
    String[] rec = reader.readNext();
    while (rec != null) {
        linesReaded++;
        try {
            vacancyNum = Integer.parseInt(rec[0]);
            System.out.println("line " +linesReaded +
                               "; vacancyNum " + vacancyNum);
        } catch (NumberFormatException e) {
            System.out.println("line " +linesReaded +
                               ": vacancyNum err");
        }
        vacancyCity = rec[5];
        if ((vacancyCity == null) |
            (vacancyCity.isEmpty()) |
            (vacancyCity.length() > 60)) {
                System.out.println("line " +linesReaded +
                                   "; vacancyCity err");
        } else {
            System.out.println("line " +linesReaded +
                               "; vacancyCity " + vacancyCity);
        }
        rec = reader.readNext();
    }
} catch (IOException e) {
    e.printStackTrace();
}
```

Рисунок 3.20 – Фрагмент коду для логування помилок CSV файлу

```
line 2; vacancyNum 1
line 2; vacancyCity Москва
line 3; vacancyNum 2
line 3; vacancyCity Москва
…
line 60; vacancyNum 59
line 60; vacancyCity Москва
line 61; vacancyNum 60
line 61; vacancyCity Москва
line 62: vacancyNum err
line 62; vacancyCity err
…
line 314: vacancyNum err
line 314; vacancyCity err
line 315: vacancyNum err
line 315; vacancyCity err
```

Рисунок 3.21 – Звіт про помилки в наборі CSV

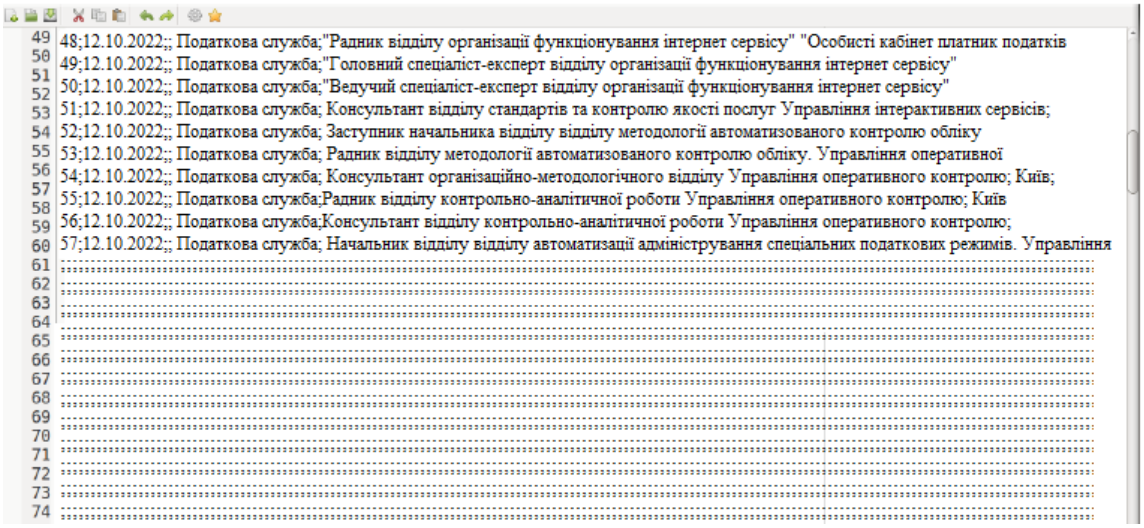


Рисунок 3.22 – Некоректні дані у наборі

Необхідно відзначити, що очищення та консолідація даних може виконуватися і тоді, коли дані завантажені у розподілену файлову систему. Наприклад, ці процеси можуть виконуватися за допомогою команд Spark. Однак дані технології виходять за рамки дослідження.

3.4 Завантаження даних у систему зберігання та аналізу великих даних

Завантаження даних у розподілену файлову систему або базу даних NoSQL також може бути здійснена різними способами, наприклад: з використанням утиліт, наприклад hdfs, hdfs shell [30]; з використанням GUI інтерфейсу системи зберігання

та аналізу великих даних, наприклад Ambari; із використанням спеціально розроблених додатків; за допомогою Yarn, Spark.

У ході дослідження було досліджено перші два способи завантаження даних.

Завантаження даних з використанням команди hdfs аналогічне роботі з файлами в операційній системі класу Unix. Створення каталогу завантаження в hdfs виконується командою

hdfs dfs -mkdir /usr/data,

яка створює у розподіленій файловій системі каталог data всередині каталогу usr.

Копіювання файлу data.csv з локального хоста до розподіленої файлової системи Hadoop здійснюється командою

hdfs dfs -put data.csv /usr/data.

На додаток можна використовувати такі команди hdfs:

- ls – читання вмісту каталогу;
- rm – видалення файлу;
- du -h - Висновок розміру файлу;
- cmod – зміна атрибутів доступу;
- cat – читання вмісту файлу;
- mv – переміщення файлу на місце.

Приклад виконання завантаження файлу до системи зберігання та аналізу великих даних представлений на рис. 3.23.

```
[maria_dev@sandbox-hdp ~]$ hdfs dfs -mkdir data
[maria_dev@sandbox-hdp ~]$ hdfs dfs -ls
Found 1 items
drwxr-xr-x   - maria_dev hdfs          0 2022-06-13 06:35 data
[maria_dev@sandbox-hdp ~]$ hdfs dfs -put "/home/user/data-20181205-
structure-20181205.csv" data
```

Рисунок 3.23 – Завантаження файлу в систему зберігання та аналізу великих даних утилітою hdfs

Другий спосіб завантаження даних – використання веб-інтерфейсу системи Ambari складання Hortonworks системи зберігання та аналізу великих даних. Після входу в систему рис. 3.24 можна здійснити навігацію HDFS і вибрати потрібний каталог для завантаження файлу рис. 3.25.

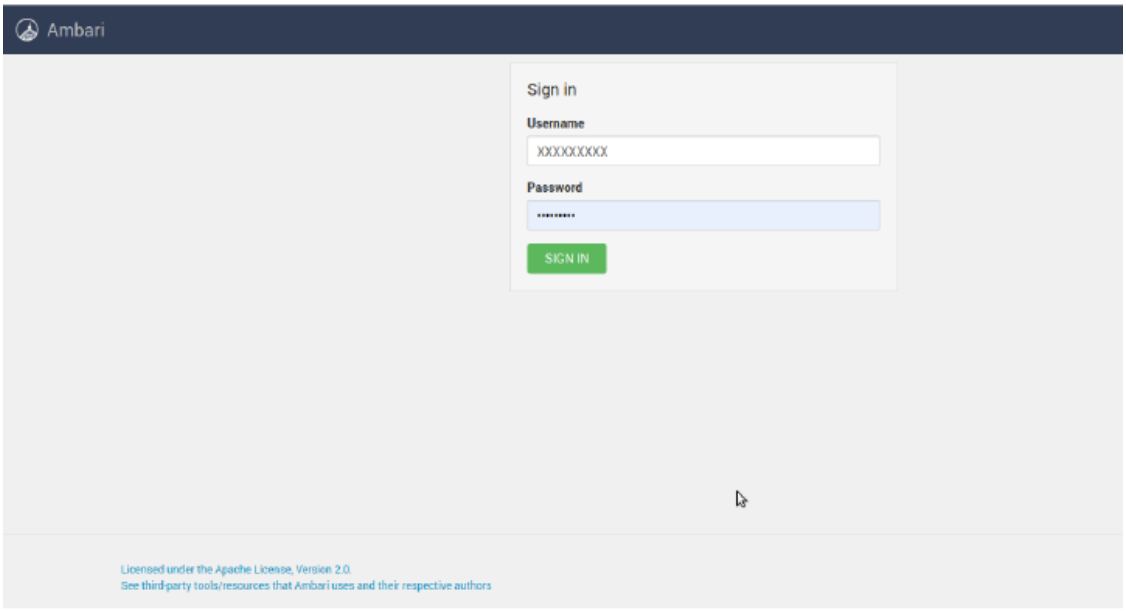


Рисунок 3.24 – Вхід до системи зберігання та аналізу даних від Hortinworks

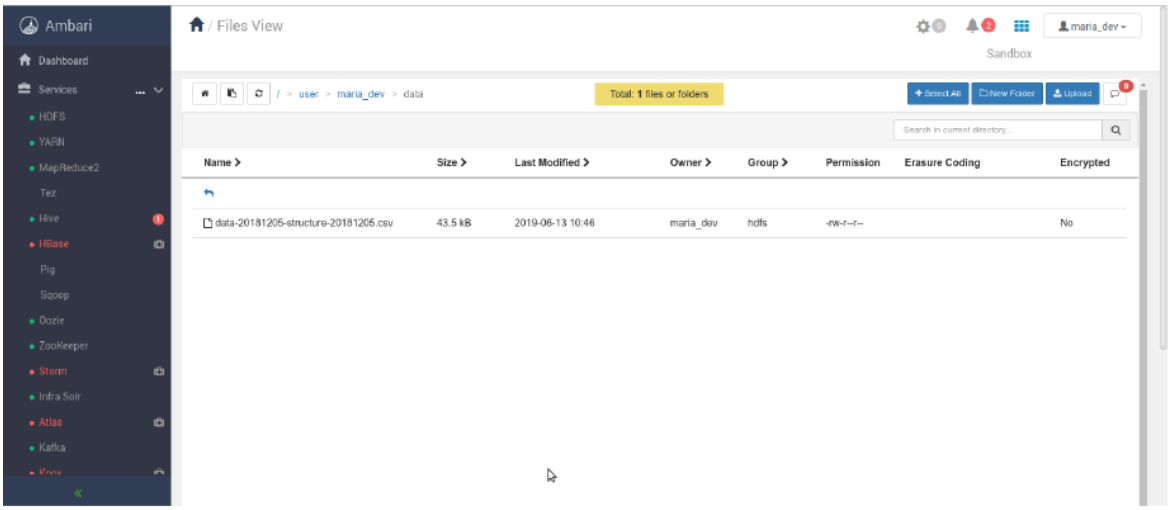


Рисунок 3.25 – Навігація по HDFS та завантаження файлу

Після завантаження файлу можна змінити права доступу до файлів і каталогів. Наприклад, на рис. 3.26 показано зміну атрибутів каталогу для можливості використання завантажених файлів у системі Hive.

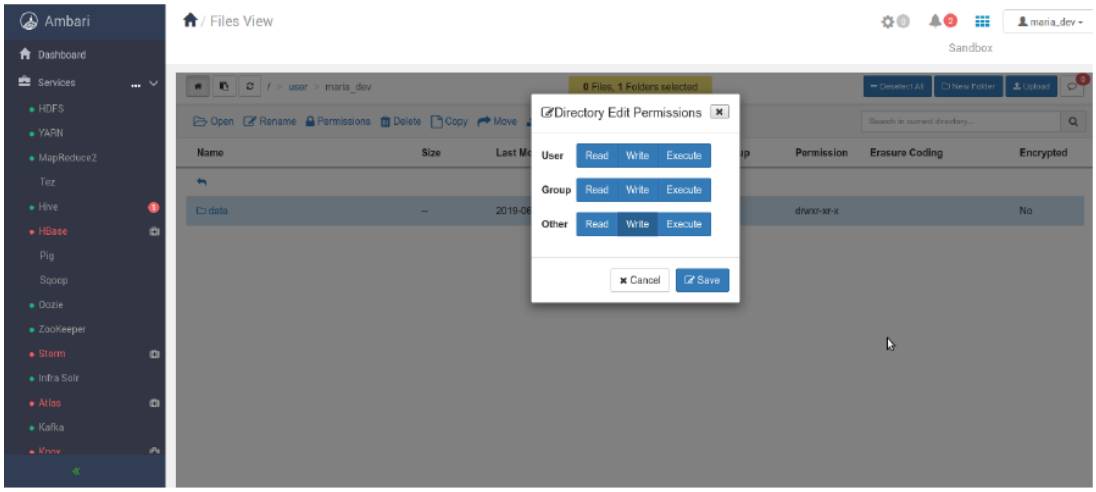


Рисунок 3.26 – Зміна атрибутів доступу до каталогу для можливості обробки Hive

ВИСНОВКИ

Отже, виконуючи поставлені задачі, у магістерській роботі досліджено особливості технології Big Data, виконано огляд сфер застосування та прогноз розвитку світового ринку великих даних. Проаналізовано актуальність застосування технологій зберігання та аналізу великих даних. Описано і проаналізовано особливі властивості великих даних.

Зроблено висновок про актуальність магістерської роботи, яка зумовлена необхідністю дослідженню технологій та систем зберігання і аналізу великих даних.

Виконані у роботі наукові дослідження представлені такими основними результатами:

1. Проведено аналіз архітектур побудови систем зберігання та аналізу великих даних. Показано, що на початкових етапах доцільно використовувати дистрибутиви для розгортання систем, що розглядаються, з урахуванням пакетної або потокової обробки даних.

2. Досліджено застосування чотирьох видів NoSQL баз даних. Розглянуто сфери їх застосування.

Системи зберігання та аналізу великих даних будуються за двома архітектурними типами: системи пакетної та/або потокової обробки великих даних.

Існує велика кількість інструментів, що застосовуються для створення систем зберігання та аналізу великих даних і утворюють екосистему великих даних, яка включає групи інструментів: розподілені файлові системи, інструменти розгортання, бази даних NoSQL і т.д.

Існує чотири види баз даних NoSQL: стовпцеві, документальні, ключ-значення, графові. Кожен з видів NoSQL баз даних має свої переваги та найкращу застосовність при вирішенні певного кола завдань.

Розгортання систем зберігання та аналізу великих даних представляє складне технічне та інженерне завдання. Для полегшення процесу розгортання систем

зберігання та аналізу великих даних можуть використовуватись спеціальні дистрибутиви, виконані у вигляді віртуальних машин або хмарних служб.

У третьому розділі проаналізовано способи доступу до зовнішніх даних, що надаються інтернет ресурсами. Апробовано технології веб-сканування, веб-скріпінгу та доступу до зовнішніх API інтернет ресурсів, використовуючи стандартні та сторонні бібліотеки Java; показано використання стандартних бібліотек Java та відкритої бібліотеки OpenCSV для очищення даних засобами мови Java. Відзначено, що навіть при використанні автоматично згенерованих даних може знадобитися очищення; апробовано способи завантаження даних у систему зберігання даних. Як реалізацію використана збірка від Hortonworks. Найбільш простими способами завантаження є використання веб-інтерфейсу системи Ambari або використання команд hdfs для завантаження даних з локальної системи HDFS Hadoop;

Таким чином, досліджено технологію створення системи зберігання та аналізу великих даних, а також апробовано та описано рішення щодо виконання перших етапів процесу Data Science: отримання, очищення та завантаження даних.

ПЕРЕЛІК ПОСИЛАНЬ

1. Shmueli, G. & Koppius, O.Predictive Analytics in Information Systems Research. MIS Quarterly, 2020, pp. 553-72.

2. Chae, B., Sheu, C., Yang, C. and Olson, D. The impact of advanced analytics and data accuracy on operational performance: A contingent resource based theory (RBT) perspective, Decision Support Systems, 2019, pp.119-126.

3. Alhomdy, S.; Thabit, F.; Abdulrazzak, F.A.H.; Haldorai, A.; Jagtap, S. The role of cloud computing technology: A savior to fight the lockdown in COVID-19 crisis, the benefits, characteristics and applications. Int. J. Intell. Netw. 2021, pp. 166–174.

4. Alsunaidi, S.J.; Almuhaideb, A.M.; Ibrahim, N.M.; Shaikh, F.S.; Alqudaihi, K.S.; Alhaidari, F.A.; Khan, I.U.; Aslam, N.; Alshahrani, M.S. Applications of Big Data Analytics to Control COVID-19 Pandemic. Sensors 2021.

5. Adrian, C.; Adrian, C.; Abdullah, R.; Atan, R.; Jusoh, Y.Y. Conceptual Model Development of Big Data Analytics Implementation Assessment Effect on Article in Press Conceptual Model Development of Big Data Analytics Implementation Assessment Effect on Decision-Making. Int. J. Interact. Multimed. Artif. Intell. 2018, pp. 101–106.

6. Gupta, M.; George, J.F. Toward the development of a big data analytics capability. Inf. Manag. 2016, pp.1049–1064.

7. Kalema, B.M.; Mokgadi, M. Developing countries organizations' readiness for Big Data analytics. Probl. Perspect. Manag. 2017,pp. 260–270.

8. Rialti, R.; Marzi, G.; Ciappei, C.; Busso, D. Big data and dynamic capabilities: A bibliometric analysis and systematic literature review. Manag. Decis. 2019, pp. 2052–2068.

9. Al-Sai, Z.A.; Abdullah, R.; Husin, M.H. Big Data Impacts and Challenges: A Review. In Proceedings of the 2019 IEEE Jordan International Joint Conference on Electrical Engineering and Information Technology, Amman, Jordan, 9–11 April 2019; pp. 150–155

10. Davenport, T.; Dyché, J. Big Data in Big Companies. Baylor Bus. Rev. 2013, pp. 20–21.

11. Comuzzi, M.; Patel, A. How organisations leverage Big Data: A maturity model. Ind. Manag. Data Syst. 2016, pp. 1468–1492.

12. Kanan, T.; Mughaid, A.; Al-Shalabi, R.; Al-Ayyoub, M.; Elbe, M.; Sadaqa, O. Business intelligence using deep learning techniques for social media contents. Cluster Comput. 2022, pp.1–12.

13. Singh, S.; Singh, N. Big Data analytics. In Proceedings of the 2012 International Conference on Communication, Information & Computing Technology, Mumbai, India, 19–20 October 2012; pp. 1–4.

14. Soon, K.W.K.; Lee, C.A.; Boursier, P. A study of the determinants affecting adoption of big data using integrated Technology Acceptance Model (TAM) and diffusion of innovation (DOI) in Malaysia. Int. J. Appl. Bus. Econ. Res. 2016, 14, 17–47.

15. Delen, D.; Zolbanin, H.M. The analytics paradigm in business research. J. Bus. Res. 2018, pp. 186–195.

16. Lepenioti, K.; Bousdekis, A.; Apostolou, D.; Mentzas, G. Prescriptive analytics: Literature review and research challenges. Int. J. Inf. Manag. 2020, pp. 57–70.

17. Delen, D.; Ram, S. Research challenges and opportunities in business analytics. J. Bus. Anal. 2018, 1, 2–12.

18. Gherhes, , V.; Stoian, C.E.; Fărcas,iu, M.A.; Stanici, M. E-learning vs. Face-to-face learning: Analyzing students' preferences and behaviors. Sustainability 2021.

19. Ruiz-Palmero, J.; Colomo-Magaña, E.; Ríos-Ariza, J.M.; Gómez-García, M. Big data in education: Perception of training advisors on its use in the educational system. Soc. Sci. 2020.

20. Fischer, C.; Pardos, Z.A.; Baker, R.S.; Williams, J.J.; Smyth, P.; Yu, R.; Slater, S.; Baker, R.; Warschauer, M. Mining Big Data in Education: Affordances and Challenges. Rev. Res. Educ. 2020, pp. 130–160.

21. Blazquez, D.; Domenech, J. Big Data sources and methods for social and economic analyses. Technol. Forecast. Soc. Chang. 2018, pp. 99–113.

22. Bhasin, M.L. Combatting Bank Frauds by Integration of Technology: Experience of a Developing Country. Br. J. Res. 2016, pp. 64–92.

23. Mihalis, K. Ten technologies to fight coronavirus. Eur. Parliam. Res. Serv. 2020, pp. 1–20.

24. Sawant N., Shah H. Big Data Application Architecture //Big data Application Architecture Q & A. – Apress, Berkeley, CA, 2013. – C. 9-28.

25. Cielen D., Meysman A., Ali M. Introducing data science: big data, machine learning, and more, using Python tools. – Manning Publications Co., 2016.

26. Erik Meijer. The world according to linq. Communications of the ACM, 54(10):45–51, 2011.

27. Larman C. “Applying UML and patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development”, (New Jersey: Prentice Hall), 2004, p. 736.

28. Mehta R. Big Data Analytics with Java / R. Mehta. Packt Publishing, 2017. – 418 C.

29. Olive A. “Conceptual Modeling of Information Systems”, 2007.

30. Paul Zikopoulos, Chris Eaton, et al. Understanding big data: Analytics for enterprise class hadoop and streaming data. McGraw-Hill Osborne Media, 2011.

ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ (Презентація)

Слайд №2 ЗАВДАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Мета роботи - дослідження архітектур побудови, алгоритмів роботи систем зберігання та аналізу великих даних.

Об'єкт дослідження - процес зберігання та аналізу великих даних.

Предмет дослідження – системи зберігання та аналізу великих даних.

Для виконання поставленої мети, у магістерській роботі розроблено та виконано наступні завдання:

- аналіз особливостей та перспективи застосування технології Big Data;
- дослідження архітектурних та програмних рішень при побудові систем зберігання та аналізу великих даних;
- створення системи зберігання та аналізу великих даних.

Слайд №3 ОГЛЯД ОСОБЛИВОСТЕЙ ТЕХНОЛОГІЇ BIG DATA

Рисунок 1 - Класифікація даних

Рисунок 2- Складові Big Data, які пояснюють характеристики

Рисунок 1.3 – Швидкість накопичення даних

Слайд №4 ОГЛЯД СФЕР ЗАСТОСУВАННЯ

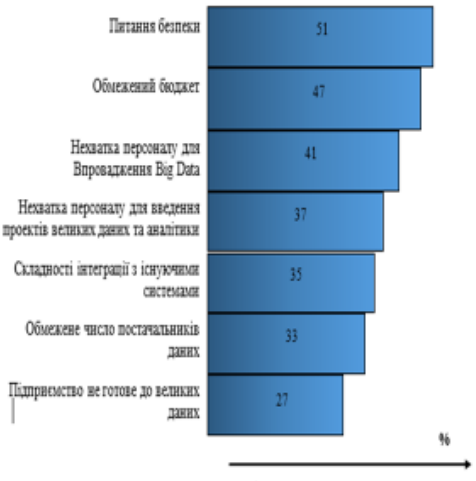


Рисунок 1.4 - Основні проблеми при впровадженні проектів Big Data

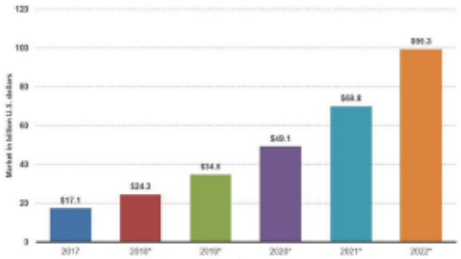


Рисунок 1.5— Огляд обсягу ринків Big Data

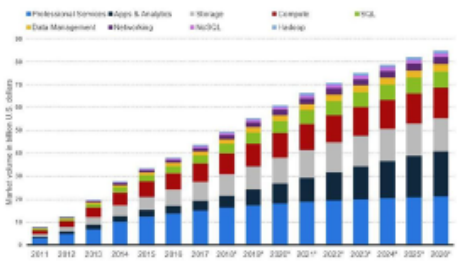


Рисунок 1.6— Прогноз ринку додатків Big Data за сегментами програмного забезпечення

Слайд № 5 Огляд архітектурних рішень великих даних

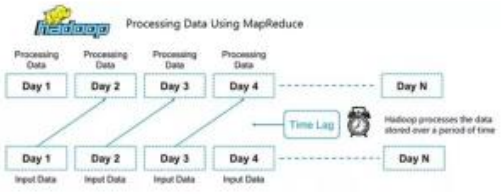


Рисунок 1.7 – Схема пакетної обробки даних у Hadoop MapReduce

Рисунок 1.9 – Хмарні рівні архітектури для побудови систем великих даних

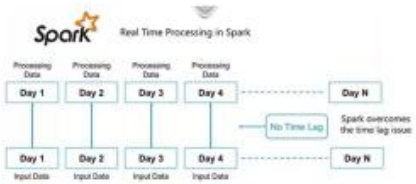


Рисунок 1.8 – Обробка великих даних у реальному часі за допомогою Apache Spark

Слайд № 6 Приклад архітектури систем зберігання та аналізу великих даних

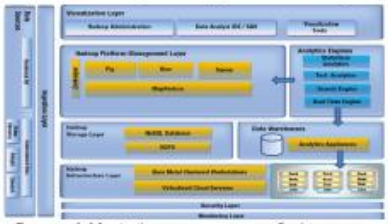


Рисунок 1.10 - Архітектура системи зберігання та аналізу великих даних

Рисунок 1.11 – Роль рівня завантаження даних в архітектурі додатків зберігання та аналізу великих даних

Рисунок 1.12 – Ландшафт додатків Big Data рівня підприємства

Слайд №7 Аналіз способів та технології отримання великих даних

Рисунок 1.13 – Сервіс generatedata.com

```
import java.io.IOException;
import java.util.List;
import org.jsoup.Jsoup;
import org.jsoup.nodes.Document;
import org.jsoup.select.Elements;

public class JavaParsers {

    //https://ru.wikipedia.org/wiki
    private static String url =
"https://ru.wikipedia.org/wiki/%D0%91%D0%BE%D0%BB%D1%8C%D1%88%D0%B8%D0%B5_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D0%B5";

    public static void main(String[] args) {
        try {
            Document document = Jsoup.connect(url).get();
            //get a title
            String title = document.title();
            System.out.println("Name Web: " + title);
            //get a body
            Elements body = document.select("body");
            System.out.println("Name: " + body.text());
            //get links
            Elements links = document.select("a[href]");
            //System.out.println(links.text());
            StringString linkValue = links.attr("href");
            int linkMap = links.size();
            System.out.println("Name Hyperlinks: " + linkMap + "\n
Text this:");
            for(int i=0; (i < 0) & (i < linkMap); i++) {
                System.out.println(links.get(i));
            }
        } catch (IOException ex) {
            ex.printStackTrace();
        }
```

Рисунок 1.14 – Код JSoup парсера, що реалізує веб-скріпінг сторінки вікіпедії

Великі дані — Вікіпедія
— big data, [ˈbɪg ˈdeɪtə]) — позначення структурованих та неструктурованих даних величезних об'єми та значного різноманіття, що ефективно обробляються горизонтально масштабованими програмними інструментами, що з'явилися наприкінці 2000-х років та альтернативних традиційним системам управління базами даних та рішенням класу Business Intelligence [2] [3]. У широкому значенні про «великі дані» говорять як про соціально-економічний феномен, пов'язаний з появою технологічних можливостей аналізувати величезні масиви даних, в деяких проблемних областях — весь світовий обсяг даних, і з цього трансформаційних наслідків [4]. Як визначальних характеристик великих даних традиційно виділяють «три V»: обсяг (англ. volume, у сенсі величини фізичного обсягу), швидкість (velocity у сенсах як швидкості приросту, і необхідності високошвидкісної обробки та отримання результатів), різноманіття (variety, у сенсі можливості одночасної обробки різних типів структурованих та неструктурованих даних) [5] [6]; надалі виявили різні варіації та інтерпретації цієї основи [⇔].
…
некомерційної організації Wikimedia Foundation, Inc. Політика конфіденційності Опис Вікіпедії Відмова від відповідальності Зв'яжіться з нами Розробники Угода про cookie Мобільна версія
Число гіперпосилань: 435
Перші п'ять:
Skip to navigation
Перейти до пошуку
англ.
ˈbɪg ˈdeɪtə
неструктурованих даних

Рисунок 1.15 – Парсинг сторінки статті вікіпедії, присвяченої великим даним

Слайд №8 Веб сканування та доступ до даних через API

```java
    /**
     * This function is called when a page is fetched and ready
     * to be processed by the program.
     */
    @Override
    public void visit(Page page) {
        String url = page.getWebURL().getURL();
        System.out.println("URL: " + url);

        if (page.getParseData() instanceof HtmlParseData) {
            HtmlParseData htmlParseData = (HtmlParseData)
page.getParseData();
            String text = htmlParseData.getText(); //extract text
from page
            String html = htmlParseData.getHtml(); //extract html
from page
            Set<WebURL> links = htmlParseData.getOutgoingUrls();

            System.out.println("=========================================
========================");
            System.out.println("Page URL: " + url);
            System.out.println("Text length: " + text.length());
            System.out.println("Html length: " + html.length());
            System.out.println("Number of outgoing links: " +
links.size());
            System.out.println("=========================================
========================");
        }
    }
```

Рисунок 1.16 – Фрагмент коду методу обробки веб-сканером HTML сторінки

```java
public class Main {

    public static void main(String[] args) {
        try {
            URL url = new URL ("https://data.gov/api/dataset/?" +
            "access_token=XXX");

            JSONArray datasets = new JSONArray(
                IOUtils.toString(url, Charset.forName("UTF-8")));
            int numRec = datasets.length();
            System.out.println(numRec);
            JSONObject e;
            for (int i=0; (i < 5) && (i < numRec); i++) {
                e = (JSONObject) datasets.get(i);
                System.out.println(e.toString());
            }
        } catch (MalformedURLException ex) {
            ex.printStackTrace();
        } catch (IOException ex) {
            ex.printStackTrace();
        }
```

Рисунок 1.17– Отримання доступних наборів даних на порталі відкритих даних України

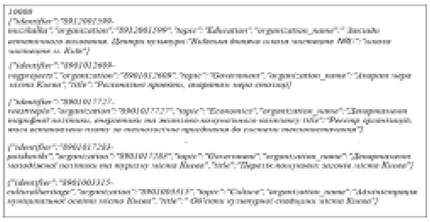


Рисунок 1.18 – Перші п'ять доступних наборів даних на порталі відкритих даних

Таблиця 1.1 - Фрагмент вмісту набору даних порталу відкритих даних

Київська область	4758	720	1050	5006	18292	29806
Київська область	60	10	6	8	114	198
Київська область	122	4	4	28	120	278
Київська область	428	200	78	180	2064	2950
Київська область	58	2	14	6	38	118
Київська область	378	24	130	136	1270	1938
Київська область	436	62	112	48	1026	1684
Житомирська область	5644	922	674	6748	11884	23872
Житомирська область	8	14	8	32	62	124
Житомирська область	98	18	8	32	110	266
Житомирська область	586	174	112	556	2422	3850
Житомирська область	20	6	4	14	18	62
Житомирська область	188	44	128	98	824	1278
Житомирська область	216	52	56	768	732	1824
Житомирська область	4	6	6	24	30	70
Житомирська область	0	0	0	0	0	0
Житомирська область	68	20	0	278	216	582
Житомирська область	334	58	2	870	604	1868
Житомирська область	0	0	0	0	0	0
Житомирська область	0	0	0	0	0	0
Житомирська область	0	0	0	0	0	0
Житомирська область	210	74	4	1304	890	2482
Житомирська область	14	14	0	10	24	62
Житомирська область	22	0	0	56	52	130

Слайд №9 Дослідження прикладу очищення даних

Рисунок 1.19 – Фрагмент даних із сайту

Рисунок 1.20 – Опис структури даних у форматі CSV

```java
public class VacancyReader {
    static final String FILE_NAME = "data.csv";

    public static void main(String[] args) {
            try (   Reader reader =
Files.newBufferedReader(Paths.get(FILE_NAME))   ) {
                CsvToBean<Vacancy> csvToBean =
                        new CsvToBeanBuilder<Vacancy> (reader).
                            withType(Vacancy.class).
                            withSeparator(';').
                            withIgnoreLeadingWhiteSpace(true).
                            build();
                Iterator<Vacancy> it = csvToBean.iterator();
                while (it.hasNext()) {
                    Vacancy vacancy = it.next();
                    System.out.println(vacancy);
                }
            } catch (FileNotFoundException e) {
                e.printStackTrace();
            } catch (IOException e) {
                e.printStackTrace();
            }
```

Рисунок 1.21 – Фрагмент коду програми для читання CSV файлу та відображення даних в об'єктах Java

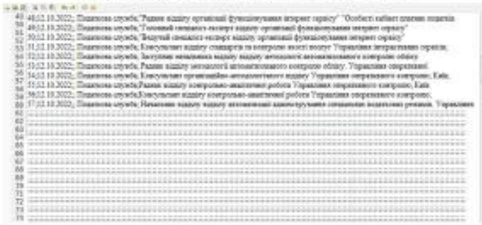


Рисунок 1.22 – Некоректні дані у наборі

Слайд №10 Завантаження даних у систему зберігання та аналізу великих даних

```
[maria_dev@sandbox-hdp ~]$ hdfs dfs -mkdir data
[maria_dev@sandbox-hdp ~]$ hdfs dfs -ls
Found 1 items
drwxr-xr-x   - maria_dev hdfs          0 2022-06-13 06:35 data
[maria_dev@sandbox-hdp ~]$ hdfs dfs -put "/home/user/data-20181205-
structure-20181205.csv" data
```

Рисунок 1.23 – Завантаження файлу в систему зберігання та аналізу великих даних утилітою hdfs

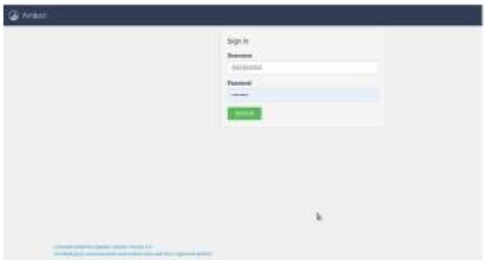


Рисунок 1.24 – Вхід до системи зберігання та аналізу даних від Hortonworks

Рисунок 1.25 – Навігація по HDFS та завантаження файлу

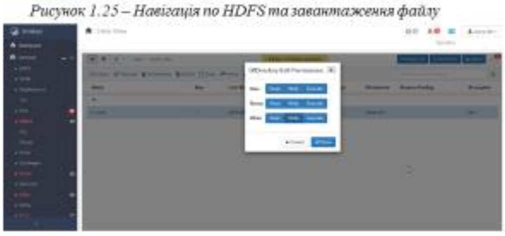


Рисунок 1.26 – Зміна атрибутів доступу до каталогу для можливості обробки Hive