НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО”

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ АТОМНОЇ
ТА ТЕПЛОВОЇ ЕНЕРГЕТИКИ
КАФЕДРА ЦИФРОВИХ ТЕХНОЛОГІЙ В ЕНЕРГЕТИЦІ

“На правах рукопису”
УДК 004.451.5 (043.3)

“**До захисту допущено**”
Завідувач кафедри ЦТЕ

_____ Наталія АУШЕВА

“___”_____ 2022 р.

Магістерська дисертація

на здобуття ступеня магістра за освітньо-професійною програмою
“Комп’ютерний моніторинг та геометричне моделювання процесів і систем”
зі спеціальності 122 “Комп’ютерні науки”

на тему: Засоби підвищення ефективності зберігання великих обсягів даних

Виконав: студент 2-го курсу, групи ТР-13мп
Власюк Богдан Сергійович _____
(прізвище, ім’я, по батькові) (підпис)

Науковий керівник доц., доц., к.т.н. Кублій Лариса Іванівна _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Рецензент: проф. кафедри інформаційних технологій
та програмування Інституту комп’ютерних
технологій Університету “Україна”
с.н.с., к.т.н. Самарай Валерій Петрович _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ — 2022

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО”

Факультет (інститут) Навчально-науковий інститут атомної та теплової енергетики
(повна назва)

Кафедра Цифрові технології в енергетиці
(повна назва)

Рівень вищої освіти другий (магістерський)
Спеціальність 122. Комп’ютерні науки
(код і назва)

Освітньо-професійна програма Комп’ютерний моніторинг та геометричне моделювання процесів і систем
(код і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри

Н. М. Аушева
(підпис) (ініціали, прізвище)

“_____“ _____ 2022 р.

ЗАВДАННЯ
на магістерську дисертацію студенту
Власюку Богдану Сергійовичу
(прізвище, ім’я, по батькові)

1. Тема дисертації “Засоби підвищення ефективності зберігання великих обсягів даних”

Науковий керівник дисертації Кублій Лариса Іванівна, канд. техн. наук, доцент
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від “*07* “ листопада *2022* р. № 4067-с

2. Строк подання студентом дисертації 1 грудня 2022 року

3. Об’єкт дослідження Обробка даних великого обсягу.

4. Вихідні дані операційна система Linux Ubuntu, мова програмування Python, середовище розробки PyCharm IDE 2021, дані гоночних заїздів, Amazon Web Services

5. Перелік завдань, які потрібно розробити: дослідження предметної області, проектування та програмна реалізація системи, розробка системи для зберігання даних у реляційних базах даних, нереляційних базах даних і в файлових сховищах, одержання висновків щодо ефективності зберігання сховищах

6. Орієнтований перелік публікацій: VI Міжнародна науково-практична конференція “Інтеграція світових наукових процесів як основа суспільного прогресу”, 25-26 листопада 2022 року, м. Київ

7. Дата видачі завдання: 1 серпня 2022 року

Календарний план

№ з/ п	Назва етапів виконання магістерської дисертації	Строк виконання етапів магістерської дисертації	Примітка
1	Отримання завдання	01.08.2022	
2	Вивчення та аналіз задачі	01.08.2022–14.08.2022	
3	Збір інформації	15.08.2022–31.08.2022	
4	Аналіз вимог завдання, розробка методів і засобів розв’язання поставленої задачі	01.09.2022–30.09.2022	
5	Розробка та тестування програмного продукту	01.10.2022–14.10.2022	
6	Підготовка матеріалів магістерської роботи	15.10.2022–20.11.2022	
7	Захист програмного продукту	25.10.2022	
8	Передзахист	22.11.2022	
9	Захист	22.12.2022	

Студент _____ Б. С. Власюк
(підпис) (ініціали, прізвище)

Науковий керівник дисертації _____ Л. І. Кублій
(підпис) (ініціали, прізвище)

РЕФЕРАТ

Магістерська дисертація за темою “Засоби підвищення ефективності зберігання великих обсягів даних” виконана студентом кафедри цифрових технологій в енергетиці НН ІАТЕ Власюком Богданом Сергійовичем зі спеціальності 122 “Комп’ютерні науки” за освітньо-професійною програмою “Комп’ютерний моніторинг та геометричне моделювання процесів і систем” і складається зі: вступу; 5 розділів (“Поняття великих даних і підходів до їхнього зберігання”, “Хмарні технології для роботи з великими даними і сховищами даних”, “Розробка прототипів систем для дослідження ефективності зберігання даних”, “Тестування систем і збір статистики”, “Розробка стартап-проекту”), висновків до кожного з цих розділів; загальних висновків; списку використаних джерел, який налічує 43 джерела, 35 рисунків, 22 таблиці та одного додатку. Загальний обсяг роботи 99 сторінок.

Актуальність теми. На даний час в мережі Інтернет кількість інформації зростає експоненційно. Цей процес не зупиняється, також потрібно зберігати велику кількість даних з попередніх років. У більшості випадків ці дані зберігаються в хмарі. Але використання хмарних рішень не дешеве. Тому виникає проблема, як зберігати дані в компактному форматі, щоб зекономити гроші на інфраструктурі, і так, щоб клієнти могли ефективно отримувати ці дані. Ця проблема досліджується для того, щоб зробити зберігання даних більш ефективним і зрозумілим для інших користувачів.

Метою роботи є створення рішення, яке буде ефективно зберігати велику кількість даних у відповідному сховищі даних, що допоможе зекономити кошти на обслуговування даних в хмарі.

Завдання дослідження:

— аналіз існуючих рішень;

— дослідити кілька різних розв’язань проблеми;

— побудувати архітектуру та реалізувати кілька рішень для розв’язування проблеми:

– розробити систему для зберігання даних у реляційних базах даних;

– розробити систему для зберігання даних у нереляційних базах даних;

– розробити систему для зберігання даних у файлових сховищах даних.

— протестувати та проаналізувати, які з рішень є найоптимальнішими чи найефективнішими.

Об’єкт дослідження. Системи сховища даних.

Предмет дослідження. Ефективні методи обробки і збереження даних великого обсягу.

Практичне значення одержаних результатів. Метод може бути застосований для рішень, де потрібно зберігати великий обсяг даних, архівувати дані на 10 років згідно з GDPR. Зберігання даних повинно бути ефективним, тобто вартість зберігання даних великого обсягу повинно коштувати якнайменше грошей.

Публікації. За результатами виконаної роботи опубліковано 1 тези доповіді на конференції:

Власюк Б.С. Засоби підвищення ефективності зберігання великих обсягів даних */ Б.С. Власюк, Л.І. Кублій* // Інтеграція світових наукових процесів як основа суспільного прогресу: Матеріали VI Міжнародної науково-практичної конференції, м. Київ, 25-26 листопада 2022 року [Електронний ресурс]. — Київ-Запоріжжя : Інститут інноваційної освіти, 2022. — С. 99-102. — Режим доступу до ресурсу: https://novaosvita.com/wp-content/uploads/2022/12/IntWorldScProc-Kyiv-Nov2022.pdf. (тези).

Ключові слова. *Засоби підвищення ефективності зберігання великих обсягів даних, обробка даних, сховища даних, Big Data.*

ABSTRACT

The master's thesis on the topic "Means of increasing the efficiency of storing large volumes of data" was completed by the student of the Department of Digital Technologies in Energy of the National Institute of Energy of IATE Vlasiuk Bohdan Serhiiovych from the specialty 122 "Computer Science" under the educational and professional program "Digital Technologies in Energy" and consists of: introduction ; 5 chapters (general overview of the concept of big data and approaches to its storage; overview of cloud technologies for working with big data and data warehouses; development of system prototypes to study the efficiency of data storage. Data Lake concept; system testing and statistics collection: development of a startup project), conclusions to each of these sections; general conclusions; the list of used sources, which includes 43 sources, 35 figures, 22 tables and one appendix. The total volume of work is 99 pages.

Actuality of theme. In the era of great growth of users and devices on the Internet, the amount of data is growing exponentially. This process is non-stop and a large amount of data from previous years needs to be stored as well. In most cases, this data is stored in the cloud. But using cloud solutions is not cheap. Therefore, the question arises of how to store data in a compact format in order to save money on infrastructure and so that customers can efficiently receive this data. This issue is being investigated in order to make data storage more efficient and understandable for other users.

The purpose of the work is to create an example of a solution that will effectively store a large amount of data in a suitable data warehouse, which will help save money on data maintenance in the cloud.

Objectives of the study:

— analysis of existing solutions.

— explore several different solutions to the problem.

— build an architecture and implement several solutions to solve the problem:

– develop a system for storing data in relational databases;

– develop a system for storing data in non-relational databases;

– develop a system for data storage in file data stores.

— to test and analyze which of the solutions are the most optimal or the most effective.

Object of study. Data storage systems.

Subject of study. Effective methods of processing and saving large volumes of data.

Practical significance of the obtained results. This method can be used for solutions where you need to store a large amount of data, archive data for 10 years according to GDPR. Data storage should be efficient, that is, the cost of storing large amounts of data should cost as little money as possible.

Publications. Based on the results of the work performed, 1 abstract of the report at the conference was published:

Vlasyuk B.S. Management of data submission from heterogeneous sources / *B.S. Vlasyuk, L.I. Kublii* // Integration of world scientific processes as the basis of social progress: Materials of the VI International Scientific and Practical Conference, Kyiv, November 25-26, 2022 [Electronic resource]. — Kyiv-Zaporizhia: Institute of Innovative Education, 2022. — P. 99-102. — Mode of access to the resource: https://novaosvita.com/wp-content/uploads/2022/12/IntWorldScProc-Kyiv-Nov2022.pdf. (theses).

Keywords. Means of increasing the efficiency of storing large volumes of data, data processing, data storage, Big Data.

ЗМІСТ

СПИСОК УМОВНИХ СКОРОЧЕНЬ 10
ВСТУП 11
1. ПОНЯТТЯ ВЕЛИКИХ ДАНИХ І ПІДХОДИ ДО ЇХНЬОГО ЗБЕРІГАННЯ 14
1.1. Концепція великих даних 15
1.1.1. Проблеми, пов’язані з Big Data 18
1.1.2. Робота з великими даними 19
1.2. Концепція сховища даних 20
1.2.1. Еволюція сховищ даних 21
1.2.2. Поняття хмарного сховища даних 23
1.2.3. Проектування сховища даних 24
1.3. Взаємодія сховищ даних з даними великого обсягу 25
1.3.1. Реляційні бази даних 26
1.3.2. Нереляційні бази даних 28
1.3.3. Файлові сховища 32
Висновки до розділу 1 35
2. ХМАРНІ ТЕХНОЛОГІЇ ДЛЯ РОБОТИ З ВЕЛИКИМИ ДАНИМИ І СХОВИЩАМИ ДАНИХ 36
2.1. Хмарні технології на прикладі AWS 36
2.1.1. Історія AWS 37
2.1.2. Переваги хмарного провайдера AWS 37
2.1.3. Недоліки хмарного провайдера AWS 39
2.1.4. Головні веб-сервіси AWS 39

2.2. Розробка методології для порівняння ефективності підходів 41
2.3. Порядок побудови дослідження 42
Висновки до розділу 2 43
3. РОЗРОБКА ПРОТОТИПІВ СИСТЕМ ДЛЯ ДОСЛІДЖЕННЯ ЕФЕКТИВНОСТІ ЗБЕРІГАННЯ ДАНИХ 44
3.1. Архітектура системи 44
3.1.1. Сервіс Amazon RDS 49
3.1.2. Сервіс DynamoDB 51
3.1.3. Файлове сховище S3 Bucket 53
3.2. Концепція Data Lake 56
Висновки до розділу 3 61
4. ТЕСТУВАННЯ СИСТЕМ І ЗБІР СТАТИСТИКИ 62
4.1. Тестування системи, яка зберігає дані в DynamoDB 63
4.2. Тестування системи, яка зберігає дані в MySQL 65
4.3. Тестування системи, яка зберігає дані в S3 Bucket 67
Висновки до розділу 4 70
5. РОЗРОБКА СТАРТАП-ПРОЕКТУ 71
5.1. Опис ідеї проекту 71
5.2. Технологічний аудит ідеї проекту 73
5.3. Аналіз ринкових можливостей запуску стартап-проекту 74
5.4. Розробка ринкової стратегії проекту 79
5.5. Розробка маркетингової програми стартап-проекту 81
Висновки до розділу 5 84
ВИСНОВКИ 86
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 88
ДОДАТОК А 93

СПИСОК УМОВНИХ СКОРОЧЕНЬ

API — Application programming interface — Прикладний програмний інтерфейс.

SQL — Structured Query Language — Мова структурованих запитів.

AWS — Amazon Web Services — дочірня компанія Amazon.com, що надає платформу хмарних обчислень в оренду приватним особам, компаніям та урядам на основі платної підписки.

GCP — Google Cloud Platform — запропонований компанією Google набір хмарних служб, які виконуються на тій самій самій інфраструктурі.

EDW — Enterprise Data Warehouse — реляційне сховище даних, що містить бізнес-дані компанії, включаючи інформацію про її клієнтів.

DSS — Decision support system — комп'ютеризована система, яка через збирання та аналіз великого обсягу інформації може впливати на процес ухвалення керівничих рішень в бізнесі та підприємництві.

BI — Business intelligence — це збирання, зберігання і аналіз даних що утворюються при діяльності організації.

ШІ — Штучний інтелект — розділ комп'ютерної лінгвістики та інформатики, якмй опікується формалізацією проблем і завдань, які подібні до дій, що виконує людина.

ML — Machine Learning —підгалузь штучного інтелекту в галузі інформатики, яка часто застосовує статистичні прийоми для надання комп'ютерам здатності "навчатися" (тобто, поступово покращувати продуктивність у певній задачі) з даних, без того, що бути програмованими явно.

API — Application Programming Interface — набір визначень підпрограм, протоколів взаємодії та засобів для створення програмного забезпечення.

ВСТУП

Із середини XX століття починається нова епоха — інформаційна. З початком 1950-их років почали розробляти електронно-обчислювальні машини, які набагато ефективніше обробляють інформацію. Для подальшого використання інформації її потрібно зберігати на фізичних носіях. Спершу це були: перфокарти (листок цупкого паперу з відповідними отворами), перфострічка (стрічка з цупкого паперу або полімеру з відповідними отворами), магнітна стрічка (стрічка, виготовлена з тонкого пластичного матеріалу), жорсткий диск (магнітний диск, виготовлений з твердого матеріалу, такого як скло чи алюміній), гнучкий диск або дискета (магнітний диск, покритий феромагнітним матеріалом), оптичний диск (диск, виготовлений з пластику чи алюмінію).

Можна побачити тенденцію, що кожен носій стає меншим за розмірами і ефективнішим для читання і запису даних. Проте ці фізичні пристрої мають певні недоліки, а саме: носії займають певний фізичний простір (тому, якщо цих фізичних носіїв буде багато, відповідно буде тяжко класифікувати чи сортувати), безпечність використання (якщо людина знайшла фізичний пристрій, вона зможе легко зчитати конфіденційну інформацію) і саме виготовлення цих фізичних носіїв є доволі затратним і не екологічним. Тому фахівці почали шукати кращі варіанти зберігання даних у більш доступному і зручнішому варіанті, а саме варіант використання хмарних рішень.

На початку XXI століття почалось усвідомлення того, що IT-послуги можуть бути більш ефективними, якщо будуть надаватися якимось посередником, який має готові рішення і відповідну вартість за використання цих сервісів. З кожним роком накопичувалися знання і вимоги клієнтів. Так почався бум на використання хмарних рішень. Також з розвитком технологій зростає кількість

користувачів і відповідно пристроїв, які використовуються цими користувачами і які під’єднані до глобальної мережі Інтернет.

Саме після 2010 року також стався великий приріст пристроїв, які зчитують різну інформацію з різних сенсорів. Наприклад, датчики світла, пристрої для вимірювання температури чи вологості повітря, відеозаписи з відеокамер тощо. Проте бізнес хоче не тільки бачити інформацію в реальному часі, а й аналізувати її за попередні періоди. Також багато наукових інститутів і архівів оцифровують паперові документи для того, щоб забезпечити збереження інформації для подальшого використання. І ці всі дані з кожним роком додаються до попередніх, дані нагромаджуються і виникає проблема ефективного їхнього зберігання.

Найбільшими провайдерами хмарних рішень є AWS, GCP і Microsoft Azure. Ці провайдери і їхні розробники задають тренди в розробці і допомагають різними організаціям ефективно використовувати ресурси електронно-обчислювальних машин (серверів, комп’ютерів). Провайдери хмарних рішень надають великий перелік сервісів і рішень, за допомогою яких можна розв’язати багато бізнес-задач, але не всі з цих рішень є ефективними й дешевими. Тому виникає питання, як ефективно зберігати дані в компактному варіанті, щоб клієнти могли зекономити кошти на побудові і підтримці інфраструктури.

Підвищений інтерес до даної проблеми мають організації і компанії, які постійно збирають, обробляють і аналізують різноманітні дані для своїх потреб, щоб за допомогою аналізу зібраної інформації зрозуміти потреби споживачів. Наприклад, після повномасштабного вторгнення рф на територію України українські військові і командування отримують сотні терабайтів візуальних і аудіоданих і зберігати їх постійно на фізичних пристроях не є ефективним рішенням, оскільки ці дані можуть втратитися або передатися третій стороні. Тому потрібні рішення, які будуть ефективно зберігати дані за допомогою хмарних технологій. Ці дані потрібні будуть для документування військових злочинів на території України, вчинених російськими військовими і політичними злочинцями і подальшої передачі до Міжнародного кримінального суду. Також ці

матеріали можуть бути використанні для подальшого аналізу військових дій, щоб запобігти меншим жертвам і втратам.

З цього випливає, що виникає нагальна потреба в розв’язувані цієї проблеми. Отримані рішення повинні вміти працювати з великою кількістю даних, одночасно виконувати паралельні обчислення (до 1000 і більше) та вміти аналізувати запити споживачів.

Тому тема для магістерської роботи, а саме засоби підвищення ефективності збереження даних великого обсягу, є надзвичайно актуальною. Дана та схожі проблеми досліджуються, щоб збільшити ефективність зберігання даних для економії коштів за використання сховищ.

Метою магістерської дисертації є порівняння й аналіз основних способів збереження інформації в різних сховищах: реляційні бази даних, нереляційні бази даних і файлові сховища з точки зору ефективності зберігання та отримання інформації з даних платформ.

Для досягнення цієї мети буде розроблено й реалізовано три підсистеми для трьох різних видів сховищ, а саме: реляційних, нереляційних і файлових. Усі три підходи також будуть розгорнуті в хмарі і буде проведено порівняльний аналіз.

Отримані результати будуть корисними для фахівців, які цікавляться розробкою високонавантажених систем, працюють з великою кількістю даних у режимі реального часу, а також організаціями, які зберігають велику кількість інформації для подальшого аналізу чи використання.

1. ПОНЯТТЯ ВЕЛИКИХ ДАНИХ І ПІДХОДИ ДО ЇХНЬОГО ЗБЕРІГАННЯ

Обсяг даних, які продукуються кожного дня, зростає з дуже великою швидкістю. Зростає частка даних, яка продукується з різних цифрових пристроїв, із засобів масової інформації, соцмереж, сенсорів та “інтернету речей”. На рисунку 1.1 проілюстровано цей ріст. Високими темпами також накопичується інформація в сховищах і є розуміння того, що цей ресурс не є безлімітним. Також виникає важливе питання, як можна структурувати таку кількість інформації для подальшого використання іншими інформаційними системами, щоб можна було легко і швидко отримати дані, а збереження цих даних коштувало якомога менше.

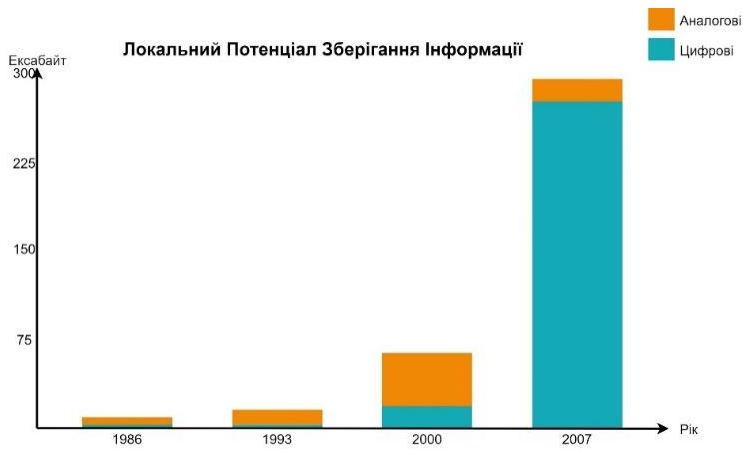


Рисунок 1.1 — Світовий технологічний потенціал для зберігання, передачі та обчислення інформаційних даних

Великі дані — це термін для позначення структурованих і неструктурованих даних дуже великих обсягів і будь-якої структури, які можна ефективно обробити за допомогою різних існуючих інструментів [1]. Ці інструменти повинні вміти горизонтально масштабуватися і з’явилися вони на зламі 2000-х років.

Проблеми аналізу великих даних включають збір, зберігання, аналіз, пошук, обмін, передачу, візуалізацію даних і конфіденційність інформації джерел даних. Великі дані спочатку асоціювалися за трьома ключовими значеннями: обсяг, різноманітність і швидкість обробки.

1.1. Концепція великих даних

Великі дані — це технології, покликані здійснювати три операції [2]:

— вміння обробляти великі обсяги даних порівняно з іншими випадками;

— вміння працювати з інформацією, яка швидко надходить у дуже великих обсягах. Тобто кількість з кожним разом стає все більшою і більшою;

— вміння обробляти структуровані і мало структуровані дані паралельно і в різних формах.

Ці три “вміння” надають можливість оптимізації і також виявити приховані закономірності, які людині складно знайти. Ці вміння можна використовувати в багатьох сферах повсякденного життя: керування проектами, державні реєстри, медичні інформаційні системи, телекомунікації, банкінг, фінанси, транспорт тощо.

Набір даних VVV — фізичний обсяг (volume), швидкість приросту даних і необхідність їхньої швидкої обробки (velocity), здатність обробляти дані різних типів (variety) [3]. Цей набір опубліковано організацією Meta Group у 2001 році для того, щоб вказати на однакову значимість керування трьома аспектами, які також подано на рисунку 1.2. Більш детальні характеристики з кожного аспекту продемонстровано в таблиці 1.1.

За останні кілька років з’явилося ще дві властивості V: цінність (viability) і правдивість (veracity) [4]. Дані мають внутрішню економічну доцільність обробки у відповідних умовах. Також важливим моментом є те, наскільки дані правдиві, чи можна їм довіряти і чи можна використовувати й аналізувіти їх у майбутньому в інших організаціях і компаніях.

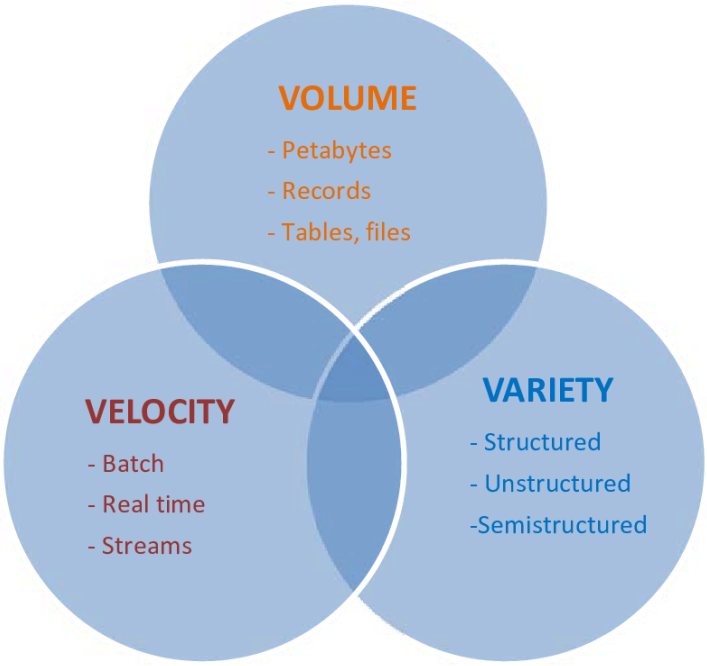


Рисунок 1.2 — Набір даних VVV

У таблиці 1.1 подано детальну характеристику властивостей набору VVV.

Таблиця 1.1

Набір даних VVV і їхній опис

Властивості	Характеристики
Volume (обсяг)	Обсяг даних має значення. З великими даними доводиться обробляти великі обсяги неструктурованих даних. Це можуть бути дані невідомої цінності, такі як дані з Twitter, потоки подій на веб-сторінці чи мобільному додатку або з обладнання з підтримкою датчиків. Для деяких організацій це можуть бути десятки терабайтів даних. Для інших це можуть бути сотні петабайтів.
Velocity (швидкість)	Швидкість — це властивість, яка показує з якою швидкістю дані надходять і обробляються. Деякі інтелектуальні продукти, які підключаються до Інтернету, працюють у режимі реального часу, які потребують оцінки та дій у реальному часі.

Таблиця 1.1 (продовження)

Variery (різноманітність)	Різноманітність стосується багатьох типів доступних даних. Традиційні типи даних були структуровані та чітко вписувалися в реляційну базу даних. Із зростанням обсягу великих даних дані надходять у вигляді нових неструктурованих типів даних. Неструктуровані та напівструктуровані типи даних, такі як текст, аудіо та відео, вимагають додаткової попередньої обробки для отримання значення та підтримки метаданих.

Нещодавні технологічні досягнення знизили вартість зберігання даних і обчислень, завдяки чому зберігання більшої кількості даних стало легшим і дешевшим, ніж будь-коли раніше. Завдяки збільшенню обсягу великих даних, які стали дешевшими і доступнішими, можна приймати значно точніші бізнес-рішення.

Виявлення цінності великих даних полягає не лише в їхньому аналізі (що є ще однією перевагою) [5]. Це цілий процес відкриття, який потребує проникливих аналітиків, бізнес-користувачів і керівників, які ставлять правильні запитання, розпізнають закономірності, роблять обґрунтовані припущення та прогнозують поведінку.

Виходячи з наведених вище визначень, основними принципами роботи з великими даними є:

— горизонтальна масштабованість. Це — основний принцип обробки даних. З часом даних стає більше й більше. Тому потрібно збільшувати кількість обчислювальних вузлів, за допомогою яких ці дані розподіляються. Також продуктивність обробки даних не повинна погіршуватися;

— відмовостійкість. Цей принцип сильно пов’язаний з попереднім. Вузли, які обробляють дані, у кластері можуть виходити з ладу. Відповідно для цього потрібно розробляти сценарії на випадок надзвичайних ситуацій;

— локальність даних. Якщо дані розподілені на великій кількості обчислювальних вузлів, то виникають втрати, коли дані передаються з одного вузла на інший. Тому рекомендується обробляти дані на тому вузлі, де вони й розміщуються.

1.1.1. Проблеми, пов’язані з Big Data

Хоч великі дані мають великі переваги, але вони не позбавлені проблем [6]. Насамперед, великі дані — це набори великих обсягів даних з різними структурами. Хоч розроблено нові технології для зберігання даних, обсяги даних подвоюються приблизно кожні два роки. Компаніям досить важко підлаштовуватися зі своїми даними і знаходити способи їхнього ефективного зберігання, а також тяжко знаходити ресурси на підтримку і зміну підходів до рішення.

Також недостатньо просто зберігати дані. Дані мають бути цінними. Чисті дані або дані, які мають відношення до клієнта та організовані таким чином, щоб можна було зробити змістовний аналіз, вимагають багато роботи. Люди, які працюють безпосередньо зі збором даних, витрачають від 50 до 80 відсотків свого часу на підготовку даних, перш ніж можна їх буде фактично використати [5].

Інші проблеми в керуванні системами великих даних включають забезпечення доступу до даних для науковців і аналітиків, особливо в розподілених середовищах, які об’єднують суміш різних платформ і сховищ даних. Щоб допомогти аналітикам знаходити релевантні дані, команди керування даними та аналітики все частіше створюють каталоги даних, які включають функції керування метаданими і походження даних. Процес інтеграції наборів великих даних також часто є складним, особливо коли впливають фактори різноманітності і швидкості даних.

Крім того, технологія великих даних змінюється швидкими темпами. Кілька років тому Apache Hadoop [7] була популярною технологією для обробки великих даних. Потім у 2014 році було представлено Apache Spark [8]. Сьогодні поєднання цих двох фреймворків є найкращим підходом для роботи з Big Data.

1.1.2. Робота з великими даними

Великі дані дають нові знання, які відкривають нові можливості та бізнес-моделі. Початок роботи з великими даними передбачає три основні дії:

— інтегрування. Великі дані об’єднують дані з багатьох різнорідних джерел і програм. Традиційні механізми інтеграції даних, такі як вилучення, перетворення і завантаження (ETL), як правило, не справляються із завданням. Це вимагає нових стратегій і технологій для аналізу великих наборів даних у масштабі терабайтів або навіть петабайтів. Під час інтеграції потрібно ввести дані, обробити їх і переконатися, що вони відформатовані й доступні у формі, з якою бізнес-аналітики можуть почати роботу;

— керування. Великі дані повинні коректно зберігатися. Можна зберігати дані в будь-якій формі і на вимогу додавати до цих наборів даних бажані вимоги до обробки та необхідні механізми обробки. Багато користувачів обирають своє рішення для зберігання відповідно до того, де зараз зберігаються їхні дані;

— аналіз. Інвестиції у великі дані окупляються, коли аналіз відбувається на основі зібраних даних. Також під час аналізу важливо візуалізувати різні набори даних. Дані після аналізу можна в подальшому використовувати для створення моделей даних для машинного навчання.

Розробка стратегії великих даних вимагає розуміння бізнес-цілей і даних, які на поточний момент доступні для використання, а також оцінки потреби в додаткових даних для досягнення поставлених цілей. Наступні кроки включають таке:

— визначення пріоритетів запланованих випадків використання і програм;

— визначення нових необхідних систем та інструментів;

— створення дорожньої карти і плану розгортання на середньо- і довгострокову перспективу;

— оцінка внутрішніх навичок, щоб побачити, чи потрібна перепідготовка або наймання.

1.2. Концепція сховища даних

Сховище даних — це тип системи керування даними, розробленої для забезпечення і підтримки діяльності бізнес-аналітики (BI). Сховища даних призначені виключно для виконання запитів і аналізу та часто містять великі обсяги історичних даних. Дані в сховищі даних, як правило, отримують із широкого діапазону джерел, таких як файли журналів додатків і програми транзакцій [9].

Архітектура сховища даних визначається конкретними потребами організації [10]. На рисунку 1.3 зображено загальну архітектуру сховища даних.

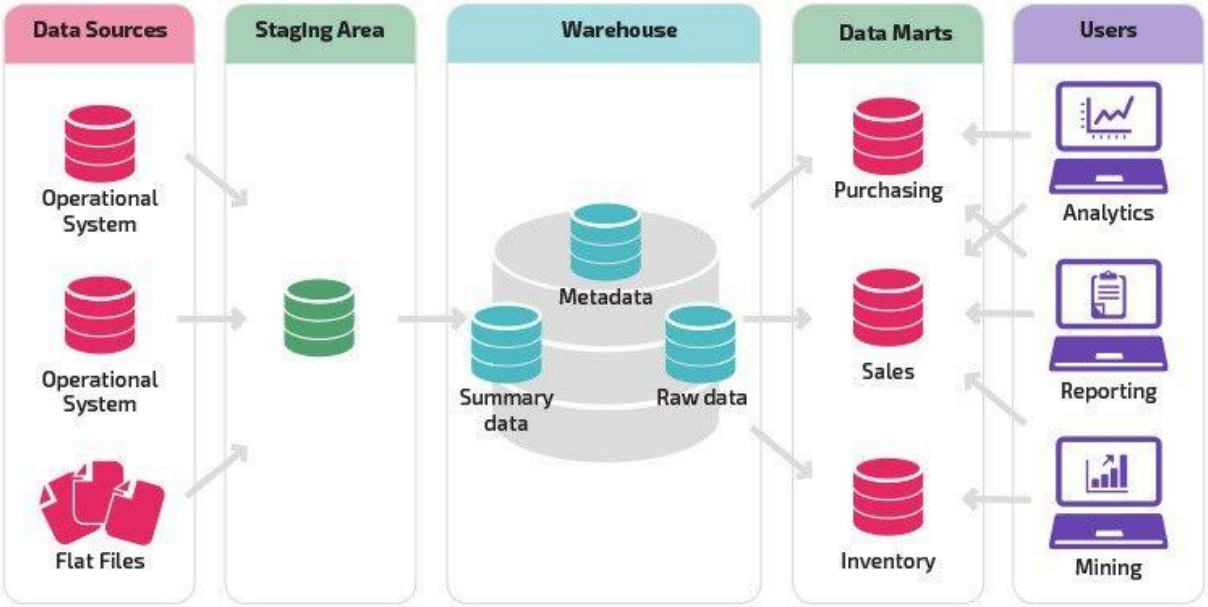


Рисунок 1.3 — Архітектура сховища даних

У загальному архітектура включає такі потреби:

— простота. Усі сховища даних мають спільний базовий дизайн, у якому метадані, зведені дані та необроблені дані зберігаються в центральному сховищі. Репозиторій живиться джерелами даних з одного боку, а кінцеві користувачі мають доступ до нього для аналізу, звітування та видобування даних з іншого боку;

— проста підготовка даних. Оперативні дані повинні бути очищені та оброблені перед зберіганням. Хоча це можна зробити програмно, багато сховищ даних додають проміжну область для даних перед тим, як вони потраплять у сховище, щоб спростити підготовку даних;

— концентратор. Додавання вітрин даних між центральним сховищем і кінцевими користувачами дає можливість організації налаштувати своє сховище даних для обслуговування різних напрямків бізнесу. Коли дані готові до використання, вони переміщуються до відповідної вітрини даних;

— ізольовані середовища. Ізольовані середовища — це приватні, захищені та безпечні зони, які дають можливість компаніям швидко й неформально досліджувати нові набори даних або способи аналізу даних, не дотримуючись формальних правил і протоколу сховища даних.

1.2.1. Еволюція сховищ даних

Коли сховища даних вперше з’явилися наприкінці 1980-х років, їхня мета полягала в тому, щоб допомогти потокові даних пройти від операційних систем до систем підтримки ухвалення рішень (DSS).

Ці ранні сховища даних вимагали величезної кількості резервування. Більшість організацій мали кілька середовищ DSS, які обслуговували різних користувачів. Хоч середовища DSS використовували ті ж самі набори даних, проте збір, очищення та інтеграція даних часто повторювалися для кожного середовища.

Оскільки сховища даних ставали ефективнішими, вони еволюціонували від сховищ інформації, які підтримували традиційні платформи, до широких аналітичних інфраструктур, які підтримують широкий спектр додатків, таких як операційна аналітика і керування ефективністю [11].

Ітерації сховищ даних з часом прогресували, щоб забезпечити додаткову цінність для клієнтів завдяки корпоративному сховищу даних (EDW) [12].

Підтримка кожного з п’яти кроків, поданих у таблиці 1.2, вимагала все більшої різноманітності наборів даних.

Таблиця 1.2

Можливості й переваги використання сховища даних

Крок	Можливість	Корпоративна цінність
1	Транзакційна звітність	Надає реляційну інформацію для створення знімків ефективності бізнесу
2	Спеціальні запити й інструменти BI	Розширює можливості для глибшого розуміння та надійнішого аналізу
3	Прогнозування майбутньої продуктивності	Розробляє візуалізацію та перспективну бізнес-аналітику
4	Тактичний аналіз (просторовий, статистика)	Пропонує сценарії “що-якщо” для прийняття практичних рішень на основі більш всебічного аналізу
5	Зберігає велику кількість історичних даних	Зберігає дані протягом тижнів або місяців

На рисунку 1.4 зображено рівні роботи основних сховищ даних. Усі рівні є важливими для роботи з даними. За допомогою ETL-інструментів можна спробувати аналізувати дані і будувати різні графіки використання [13].

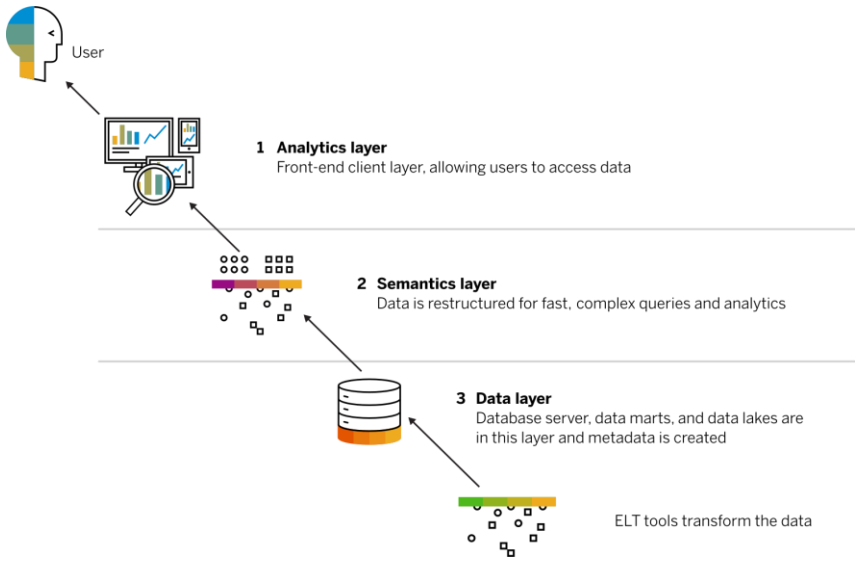


Рисунок 1.4 — Сховище даних, яке включає три окремі рівні

На даний час штучний інтелект і машинне навчання трансформують майже всі галузі, послуги і корпоративні активи — сховища даних не є винятком. Розширення великих даних і застосування нових цифрових технологій спричиняють зміни у вимогах і можливостях сховищ даних.

Автономне сховище даних є останнім кроком у цій еволюції, пропонуючи підприємствам можливість отримувати детальніший аналіз даних, одночасно знижуючи витрати та покращуючи надійність і продуктивність сховища даних.

1.2.2. Поняття хмарного сховища даних

Хмарне сховище даних використовує хмару для отримання і зберігання даних з різних джерел даних.

Оригінальні сховища даних було побудовано з локальними серверами. Ці локальні сховища даних і сьогодні мають багато переваг. У багатьох випадках вони можуть запропонувати покращене керування, безпеку, суверенітет даних і кращу затримку. Однак локальні сховища даних не такі еластичні, і вони вимагають комплексного прогнозування, щоб визначити, як масштабувати сховище даних для майбутніх потреб. Керування цими сховищами даних також може бути дуже складним.

З іншого боку, деякі з переваг хмарних сховищ даних є такими:

— еластична підтримка масштабування для великих або змінних вимог до обчислень або сховища;

— простота використання;

— простота керування;

— економія коштів.

Найкращі хмарні сховища даних є повністю керованими та автономними, що гарантує те, що навіть новачки зможуть створити і використовувати сховище даних лише за допомогою кількох клацань миші. На рисунку 1.5 подано один із прикладів схеми найсучаснішого сховища даних [13].

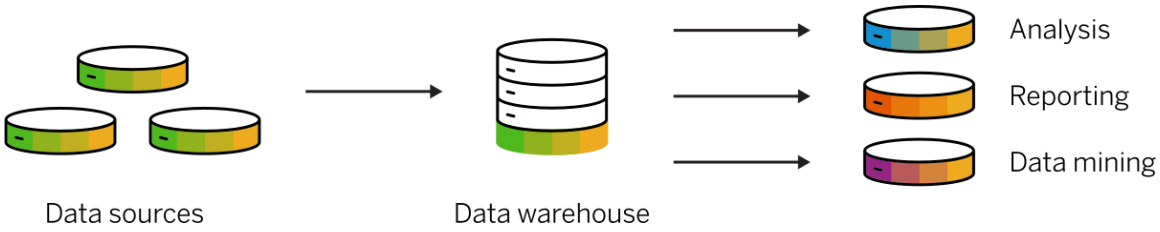


Рисунок 1.5 — Схеми сучасного сховища даних

Крім того, більшість хмарних сховищ даних дотримуються моделі оплати за використання, що забезпечує додаткову економію коштів для клієнтів.

Незалежно від того, чи є вони частиною команд IT, розробки даних, бізнес-аналітики чи наукових груп, різні користувачі в організації мають різні потреби в сховищі даних [14, 15].

Сучасне сховище даних включає:

— конвергентну базу даних, яка спрощує керування всіма типами даних і надає різні способи використання даних;

— сервіси самообслуговування прийому і перетворення даних;

— підтримку SQL, машинного навчання, графічної та просторової обробки;

— кілька параметрів аналітики, які спрощують використання даних без їхнього переміщення;

— автоматизоване керування для простого масштабування та адміністрування.

Сучасне сховище даних може ефективно оптимізувати робочі процеси даних. Це означає, що кожен, від аналітиків та інженерів обробки даних до фахівців із обробки даних та IT-команд, може виконувати свою роботу ефективніше та продовжувати інноваційну роботу без затримок і ускладнень.

1.2.3. Проектування сховища даних

Коли організація збирається розробити сховище даних, вона повинна почати з визначення конкретних бізнес-вимог, узгодження обсягу і розробки концептуального проекту. Тоді організація може створити як логічну, так і фізичну структуру для сховища даних. Логічний дизайн передбачає зв’язки між

об’єктами, а фізичний — найкращий спосіб зберігання і вилучення об’єктів. Фізичний дизайн також включає процеси транспортування, резервного копіювання і відновлення [16].

Будь-яка конструкція сховища даних повинна враховувати таке:

— конкретний вміст даних;

— зв’язки всередині та між групами даних;

— системне середовище, яке підтримуватиме сховище даних;

— типи необхідних перетворень даних;

— частота оновлення даних.

Основним фактором дизайну є потреби кінцевих користувачів. Більшість кінцевих користувачів зацікавлені в аналізі та перегляді даних у сукупності, а не як окремі транзакції. Проте часто кінцеві користувачі насправді не знають, чого вони хочуть, доки не виникає конкретна потреба. Таким чином, процес планування повинен включати достатньо досліджень, щоб передбачити потреби. Крім того, конструкція сховища даних повинна забезпечувати простір для розширення та еволюції, щоб йти в ногу з мінливими потребами кінцевих користувачів.

Сховища даних у хмарі пропонують ті самі характеристики та переваги, що й локальні сховища даних, але з додатковими перевагами хмарних обчислень, такими як гнучкість, масштабованість, безпека та зниження витрат. Хмарні сховища даних дають можливість підприємствам зосереджуватися виключно на отриманні своїх даних, а не на створенні та керуванні апаратною та програмною інфраструктурою для підтримки сховища даних.

1.3. Взаємодія сховищ даних з даними великого обсягу

Усі сховища даних мають можливість коректно працювати з великими даними, але все залежить від кількості даних, від того, з якою швидкістю потрібно обробляти дані, яка кількість даних записується чи зчитується за одиницю часу.

Умовно можна поділити сховища на такі типи: реляційні бази даних, нереляційні бази даних і файлові сховища даних.

1.3.1. Реляційні бази даних

Реляційна база даних — це тип бази даних, яка зберігає інформацію й надає доступ до точок даних, пов’язаних одна з одною. Реляційні бази даних базуються на реляційній моделі, інтуїтивно зрозумілому, простому способі подання даних у таблицях. У реляційній базі даних кожен рядок у таблиці є записом з унікальним ідентифікатором — ключем. Колонки таблиці містять атрибути даних, і кожен запис, як правило, має значення для кожного атрибута, що полегшує встановлення зв’язків між точками даних [17].

Реляційна модель означає, що логічні структури даних — таблиці даних, подання та індекси — відокремлені від фізичних структур зберігання. Це розділення означає, що адміністратори баз даних можуть керувати фізичним зберіганням даних, не впливаючи на доступ до цих даних як до логічної структури. Наприклад, перейменування файлу бази даних не перейменовує таблиці, яка зберігається в ньому.

Розрізнення між логічним і фізичним також стосується операцій бази даних, які є чітко визначеними діями і дають можливість програмам маніпулювати даними і структурами бази даних. Логічні операції дають можливість додатку вказувати потрібний вміст, а фізичні операції визначають спосіб доступу до даних і виконання завдання.

Щоб забезпечити постійну точність і доступність даних, реляційні бази даних дотримуються певних правил цілісності. Так, правило цілісності може вказувати, що повторювані рядки не допускаються в таблиці, щоб виключити можливість потрапляння помилкової інформації в базу даних.

На рисунку 1.6 зображено схему найпростішої реляційної бази даних. Видно, що база містить дві таблиці з інформацією про покупців і замовлення відповідно.

Рисунок 1.6 — Один з найпростіших прикладів схеми реляційної бази даних

Проста, але потужна реляційна модель використовується компаніями всіх типів і величини для широкого спектру інформаційних потреб. Реляційні бази даних використовуються для відстеження запасів, обробки транзакцій електронної комерції, керування дуже великими обсягами критично важливої інформації про клієнтів і багато іншого. Реляційну базу даних можна розглядати для будь-якої інформаційної потреби, у якій точки даних пов’язані одна з одною, і керувати ними потрібно безпечно, на основі правил, узгоджено.

Реляційна модель найкраще підтримує узгодженість даних у програмах і копіях бази даних (так звані примірники). Наприклад, коли клієнт вносить гроші в банкомат, а потім переглядає баланс рахунку на мобільному телефоні, клієнт очікує, що цей депозит буде негайно відображено в оновленому балансі рахунку. Реляційні бази даних перевершують такий тип узгодженості даних, гарантуючи, що кілька примірників бази даних постійно мають однакові дані.

1.3.2. Нереляційні бази даних

Термін “NoSQL” відноситься до нереляційних типів баз даних, і ці бази даних зберігають дані у форматі, який відрізняється від формату реляційних таблиць. Проте до баз даних NoSQL можна надсилати запити за допомогою ідіоматичних мовних API, декларативних структурованих мов запитів і мов запитів за прикладами, тому їх також називають базами даних “не лише SQL” [18].

Бази даних NoSQL широко використовують у веб-додатках реального часу та великих даних, оскільки їхніми головними перевагами є висока масштабованість і висока доступність.

Бази даних SQL є реляційними, а бази даних NoSQL — нереляційними. На рисунку 1.7 зображено основні відмінності між реляційними і нереляційними базами даних.

Рисунок 1.7 — Різниця між реляційними і нереляційними базами даних

Бази даних NoSQL також є кращим вибором розробників, оскільки вони природно піддаються гнучкій парадигмі розробки, швидко адаптуючись до мінливих вимог. Бази даних NoSQL дають можливість зберігати дані більш інтуїтивно зрозумілими та легшими для розуміння способами або ближчими до способів використання даних програмами — з меншою кількістю трансформацій, необхідних під час зберігання чи отримання за допомогою API NoSQL. Крім того, бази даних NoSQL можуть повною мірою використовувати переваги хмари для забезпечення кращого керування та масштабування.

Система керування реляційною базою даних (RDBMS) є основою для мови структурованих запитів (SQL), яка дає можливість користувачам отримувати доступ і маніпулювати даними у високоструктурованих таблицях. Це базова модель для систем баз даних, таких як MS SQL Server, IBM DB2, Oracle і MySQL. Але з базами даних NoSQL синтаксис доступу до даних може відрізнятися для різних конкретних баз даних.

У базах даних NoSQL дані можна зберігати без попереднього визначення схеми, що означає, що можна рухатися і швидко виконувати ітерації, визначаючи модель даних під час роботи.

До недавнього часу реляційні бази даних були найбільш широковикористовуваними моделями. Вони й досі ще надзвичайно корисні на багатьох підприємствах, проте різноманіття, швидкість та обсяг даних, іноді потребують зовсім іншої бази даних, щоб доповнити реляційну базу даних. Це стало поштовхом до впровадження в деяких галузях баз даних NoSQL — нереляційних. Завдяки своїй здатності до горизонтального та швидкого масштабування нереляційні бази даних можуть швидко і ефективно обробляти великі запити, які надходять від клієнтів.

Бази даних NoSQL пропонують гнучкі схеми, а також підтримують різноманітні моделі даних, які ідеально підходять для створення додатків, які вимагають великих обсягів даних і малої затримки або часу відповіді, наприклад, веб-додатків для онлайн-ігор і електронної комерції.

Бази даних NoSQL, як правило, базуються на ненормалізованих даних, підтримуючи типи програм, які використовують менше таблиць (або контейнерів) і чиї зв’язки даних моделюються не за допомогою посилань, а як вбудовані записи (або документи).

Іншою відмінністю баз даних NoSQL є складність запитів. Бази даних NoSQL добре працюють із запитами до однієї таблиці. Проте зі зростанням складності запитів кращим вибором є реляційні бази даних. База даних NoSQL, як правило, не пропонує складних об’єднань, підзапитів і вкладення запитів у речення WHERE [19].

Ще одна важлива відмінність полягає в тому, що бази даних NoSQL покладаються на процес, який називають “шардингом”, для горизонтального масштабування, що означає, що можна додати більше машин для обробки даних на кількох серверах. Вертикальне масштабування, яке спостерігається в інших базах даних SQL, вимагає додавання додаткової потужності й пам’яті до наявної машини, що може бути нестабільним, оскільки потрібно все більше і більше пам’яті.

Проте іноді не потрібно вибирати між реляційними та нереляційними базами даних. У багатьох випадках компанії обирали бази даних, які пропонують конвергентну модель, у якій вони можуть використовувати поєднання реляційної та нереляційної моделей даних. Цей гібридний підхід пропонує підвищену гнучкість при обробці різних типів даних, а також забезпечує послідовність читання й запису без зниження продуктивності.

Безпрецедентна швидкість і масштаби цифрової взаємодії та споживання даних, які спостерігаються за останні два десятиліття, вимагають від компаній прийняти більш сучасний підхід до того, як вони зберігають дані та як отримують до них доступ. Оскільки користувачі з усього світу вимагають безперервного потоку вмісту та функцій, базам даних довелося швидко адаптуватися.

Таким чином, маючи сказане вище на увазі, можна вказати деякі з ключових причин, чому розробники обирають нереляційні бази даних:

— висока продуктивність. Розширена архітектура бази даних NoSQL може бути особливо цінною, коли обсяг даних або трафік зростає. Як показано на рисунку 1.8, ця архітектура забезпечує швидкий і передбачуваний час відповіді в мілісекундах. Бази даних NoSQL також можуть отримувати дані та швидко й надійно передавати їх;

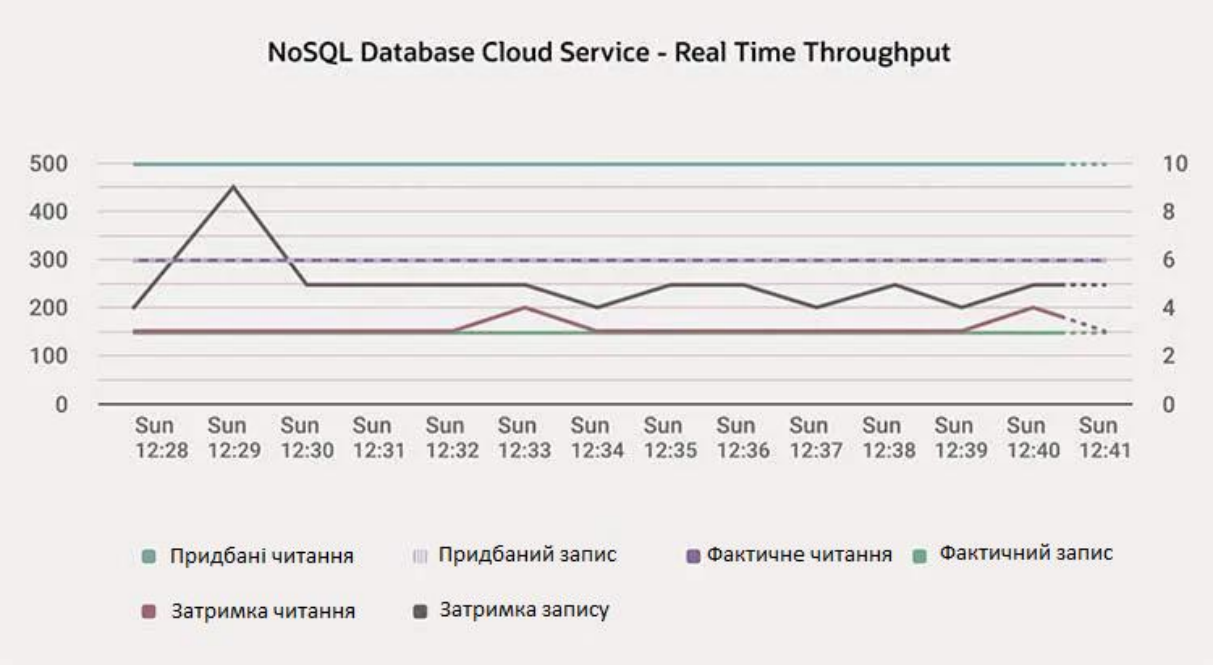


Рисунок 1.8 — Пропускна здатність баз NoSQL в режимі реального часу

— гнучкість. У базах даних SQL дані зберігаються в набагато більш жорсткій, заздалегідь визначеній структурі. Але за допомогою NoSQL дані можна зберігати в більш вільній формі без жорстких схем. Такий дизайн забезпечує інновації та швидку розробку програм. Розробники можуть зосередитися на створенні систем, щоб краще обслуговувати своїх клієнтів, не турбуючись про схеми. Бази даних NoSQL можуть легко обробляти будь-який формат даних, наприклад, структуровані, напівструктуровані та неструктуровані дані в одному сховищі даних;

— масштабованість. Замість масштабування шляхом додавання додаткових серверів бази даних NoSQL можна масштабувати за допомогою стандартного

обладнання. Це дає можливість підтримувати збільшений трафік, щоб задовольнити попит з нульовим простоєм;

— високофункціональність. Бази даних NoSQL розроблені для розподілених сховищ даних. Саме це робить NoSQL ідеальним вибором для великих даних, веб-додатків у реальному часі, онлайн-покупок, онлайн-ігор, Інтернету речей, соціальних мереж і онлайн-рекламних програм. На прикладі рисунка 1.9 видно, що нереляційні бази даних мають багато підвидів;

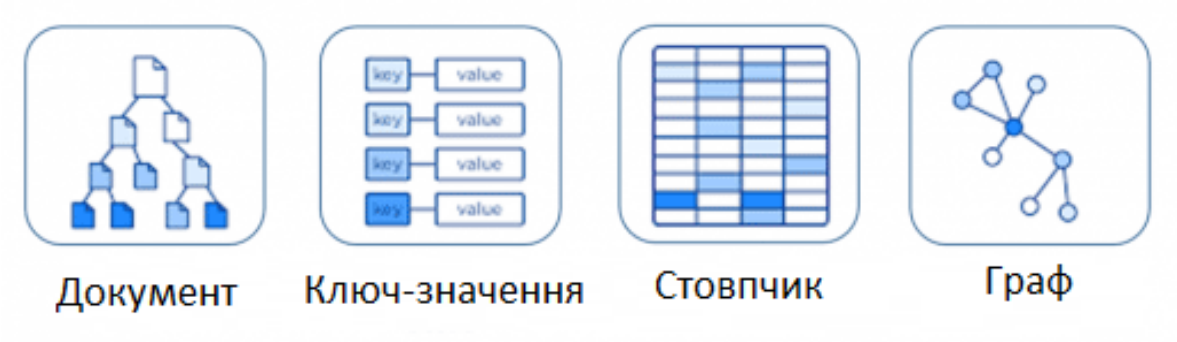


Рисунок 1.9 — Моделі даних для баз даних NoSQL

— доступність. Бази даних NoSQL автоматично копіюють дані на кількох серверах, центрах обробки даних або хмарних ресурсах. У свою чергу, це мінімізує затримку для користувачів, незалежно від того, де вони знаходяться. Ця функція також зменшує навантаження на керування базою даних, що звільняє час, щоб зосередитися на інших пріоритетах.

1.3.3. Файлові сховища

Сховище файлів — це система зберігання даних, яка розміщує повні файли в ряді вкладених папок для організаційних цілей. Будь-який жорсткий диск комп’ютера використовує модель зберігання файлів. Для хмарного сховища ідея майже така сама — єдина відмінність полягає в тому, що дані зберігаються на хмарному сервері, а не на локальному жорсткому диску [20].

Вибір файлового сховища для стратегії хмарного зберігання надає IT-відділам різноманітні послуги зберігання, орієнтовані на робоче навантаження. Цей підхід можна використати для забезпечення кінцевого користувача рішенням для

зберігання файлів, яке відповідає його індивідуальним потребам. Зберігання файлів ідеально підходить для даних, організованих з певною метою, а також для будь-яких файлів, які мають бути легко доступними та доступними для спільної роботи.

Хмарне сховище файлів працює так само, як і на локальному жорсткому диску. Під час запису файлу (встановлення програмного забезпечення, завантаження, вставлення) користувач вибирає папку призначення. Якщо передбачається пакетне встановлення, процес створює вкладені папки та розміщує туди відповідні файли.

Порівняно з іншими варіантами хмарного сховища, сховище файлів має переваги, які роблять його кращим вибором у кількох ситуаціях. Найбільш визнаними перевагами зберігання файлів є:

— простота. Зберігання файлів було популярним способом зберігання протягом десятиліть, оскільки воно відображає функціональність традиційного сховища на пристроях. Використовуючи стандартний процес ієрархічних папок і підпапок, зберігання файлів дає можливість легко сортувати дані та отримувати доступ до них без необхідності навчання;

— доступ. Інші методи зберігання, наприклад, блокове зберігання, розбивають файли на фрагменти даних. Завдяки сховищу файлів файли повністю зберігаються у встановленому місці, що робить їх легко доступними за запитом;

— співпраця. Проста організаційна структура хмарного сховища файлів робить його ідеальним для співпраці. Кілька користувачів можуть отримати доступ до однієї файлової структури, що полегшує сортування та пошук файлів. Конкретні папки можна створювати за проектом, статусом, рецензією, користувачем тощо, що забезпечує ефективний і спільний робочий процес;

— безпека. Блокування файлів — це підхід до безпеки, який блокує дозволи, пов’язані з файлом або папкою. Встановлює обмеження щодо того, які користувачі можуть змінювати або видаляти файл, гарантуючи, що випадкові оновлення під час спільного проекту не вплинуть на нього. Безпека зберігання файлів віддзеркалює безпеку традиційних пристроїв, що робить його простим при використанні без необхідності навчання.

Сховище файлів має переваги організації доступу та дозволів до об’єктів. Ці переваги компенсуються тим, як вони обмежують масштабованість. Оскільки до файлу прикріплено ієрархічне розташування і дозволи, сховище файлів має такий рівень деталізації, який може споживати ресурси. Під час масштабування ці властивості можуть вплинути на продуктивність [21].

Як і в інших методах хмарного зберігання, зберігання файлів має обмеження.

Затримка може вплинути на використання великих ресурсів, коли файлові системи стають громіздкими за розміром. Це може створити проблеми під час інтенсивних робочих навантажень, від відкриття файлів до пошуку та доступу.

Вибираючи стратегію хмарного зберігання, найкращим підходом є врахування розміру даних, використання, обсягу файлів і складності. Ці змінні допоможуть визначити найкращий варіант для будь-якої ситуації. Наприклад, сховище файлів є найкращим вибором для озера даних, але воно добре працює для домашнього каталогу користувача.

Нижче розглянуто випадки використання, коли сховище файлів є ідеальним вибором для потреб у хмарному сховищі:

— широке співробітництво. Проста ієрархічна структура зберігання файлів робить його кращим методом зберігання, коли організація співпрацює широко. Організація сховища файлів на основі папок забезпечує легкий доступ для користувачів, які можуть мати обмежену технічну підготовку;

— архіви. Оскільки архівовані дані, як правило, не потребують регулярного доступу, проблеми із затримкою та масштабованістю не викликають особливого занепокоєння під час використання файлового сховища;

— файли з низьким рівнем використання. Як і з архівами, сховище файлів добре працює для файлів, які мають бути доступними, але використовуються рідше. Приклади включають збережені мультимедійні зображення, аудіокліпи, документи та електронні таблиці. Зберігання файлів полегшує їхній пошук і, хоч можуть виникнути проблеми із затримкою, вони, ймовірно, не вплинуть на продуктивність.

Просте щодо використання для окремих осіб і організацій сховище файлів пропонує інтуїтивно зрозуміле хмарне сховище для спільної роботи, одночасно підтримуючи додатки, які потребують масштабованого доступу та ємності.

Висновки до розділу 1

У першому розділі розглянуто питання дослідження технологій збереження даних великого обсягу. Також в цьому розділі проаналізовано концепції великих даних і сховищах даних. Визначено базові атрибути великих обсягів даних: обсяг, швидкість, структурованість. Також визначено, що сховища даних можна поділити на три типи: реляційні бази даних, нереляційні бази даних і файлові сховища. Розглянуто загальні характеристики цих типів сховищах даних, їх переваги й недоліки, способи застосування разом з великими даними тощо.

Можна зробити висновок, що вивчення концепцій великих даних і сховищ даних є перспективним і актуальним у наш час. Отже, подальшою роботою буде аналіз існуючих технологій для роботи з великими даних і сховищами даних. Також потрібно буде розробити три окремі системи для кожного виду сховищ для порівняння.

2. ХМАРНІ ТЕХНОЛОГІЇ ДЛЯ РОБОТИ З ВЕЛИКИМИ ДАНИМИ І СХОВИЩАМИ ДАНИХ

Магістерську роботу виконано з використанням продуктів надавача хмарних послуг AWS через те, що містить майже готові інструменти для роботи з великими даними, хорошу документацію і також може швидко розгорнути будь-які рішення.

Одним із завдань, яке має розв'язувати вирішувати кожен IT-стартап чи бізнес, є інфраструктура.

Багато продуктів вибирають AWS як основного постачальника послуг. Є багато причин, чому можна вибрати цю компанію для своєї інфраструктури — через популярність Amazon, через їхню програму для стартапів, яка надає стартапам безкоштовні кредити на період від 6 до 12 місяців, через їхню сувору політику відповідності та безпеки, яка настільки хороша, що багато державних органів використовують для своїх послуг, — це рішення може вплинути на загальний бізнес у майбутньому.

2.1. Хмарні технології на прикладі AWS

Хмарний надавач послуг, або хмарний провайдер, Amazon Web Services (AWS) — це найбільш поширена в світі компанія з широкими можливостями, що надає понад 200 повнофункціональних сервісів для центрів обробки даних по всій планеті. Мільйони клієнтів, у тому числі стартап-компанії, найбільші корпорації та передові державні установи, використовують AWS для зниження витрат, підвищення гнучкості і прискореного впровадження інновацій [22].

2.1.1. Історія AWS

Ідея створення AWS виникла через потребу підвищити ефективність використання ресурсів Amazon і необхідність швидкого масштабування в міру зростання бізнесу.

Ще в 2003 році в Amazon зрозуміли, що вони мають значну перевагу порівняно з конкурентами. Цією перевагою були інфраструктурні послуги Amazon і їхня здатність надійно та ефективно керувати своєю інфраструктурою і масштабувати її. Саме тут спочатку народилася ідея Amazon Web Services.

Ідея почалася як “операційна система Інтернету”, де різні апаратні ресурси ізольовано (а саме обчислення, сховище, пам’ять тощо) як компоненти цієї операційної системи і вони подаються як керовані служби, які беруть на себе всю “недиференційовану важку роботу” з рук розробників. Це була революційна ідея, яка позбавляє необхідності керувати апаратним забезпеченням і фізичною інфраструктурою та дає можливість підприємствам зосередитися на своєму бізнесі.

2.1.2. Переваги хмарного провайдера AWS

Будучи одним з найпопулярніших постачальників хостингу та інфраструктури, AWS має високу стабільність і гнучкість щодо наданих послуг. Простої трапляються рідко і, як правило, спричинені винятковими обставинами. Крім того, якщо дотримуватися шаблону резервування N+1, можна отримати навіть кращі SLA, які пропонує AWS [23].

Крім того, AWS надає спеціальні інструменти, за допомогою яких продукти можуть автоматизувати процес відновлення у разі простою. Також можна інтегрувати вбудовану систему моніторингу, яка охоплює різні сфери, такі як технічні показники (велике використання ЦП, велике навантаження пам’яті тощо), фінансові (місячні витрати, розраховані до поточної дати, досягнення загального ліміту тощо). і доступність до інструменту керування інцидентами, який використовують в компанії. Отже, він охоплює багато випадків використання.

На рисунку 2.1 продемонстровано основні сервіси хмарного провайдера AWS.

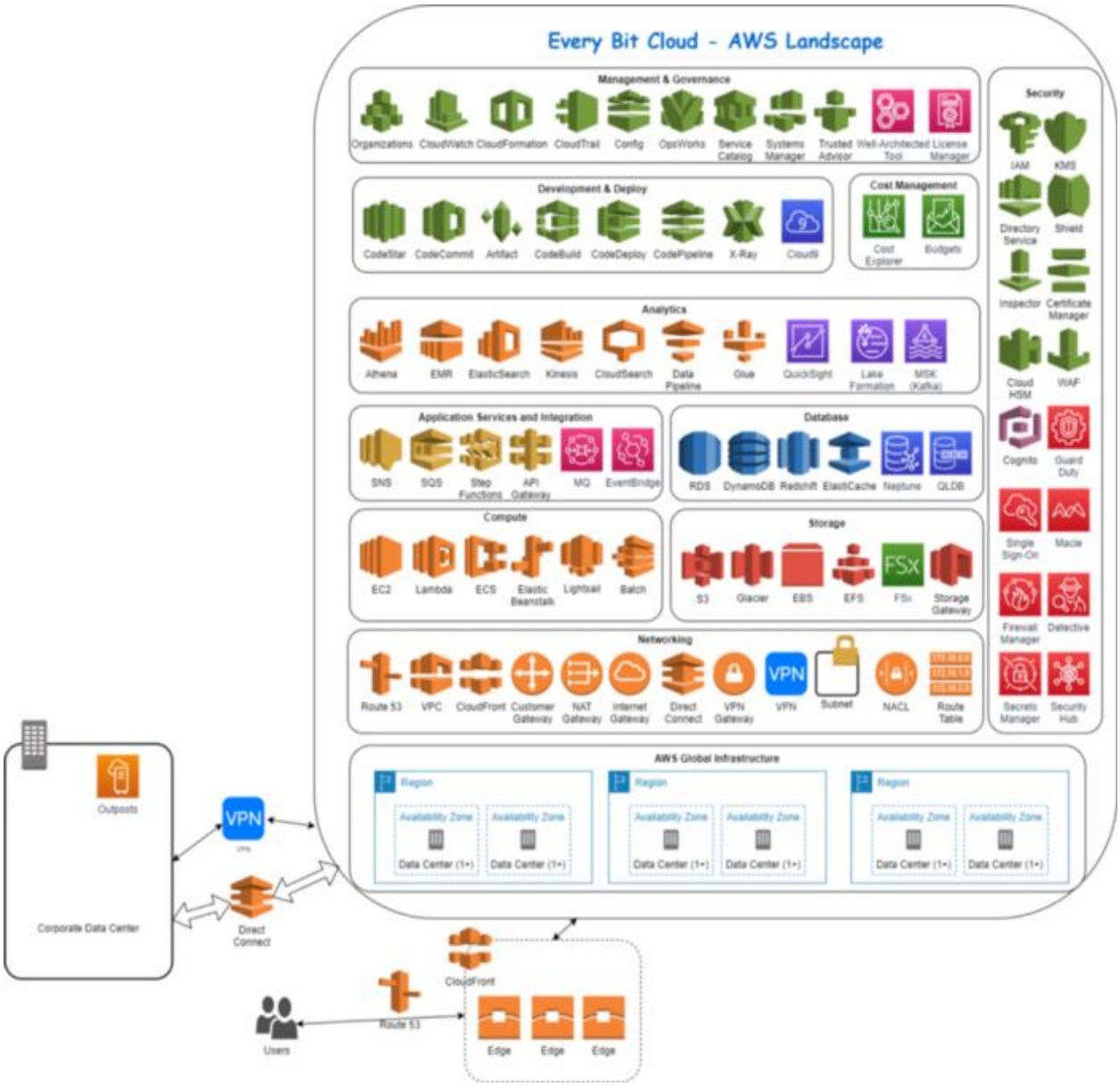


Рисунок 2.1 — Карта найбільш популярних сервісів в AWS

Хоч у деяких галузях можна знайти кращий і гнучкіший інструмент, ніж пропонує AWS, не надто багато проектів надають стільки послуг. Для DevOps складніше керувати і підтримувати кілька платформ, які не підключені одна до одної, замість однієї. Крім того, важко знайти інженера, який має глибокі знання і навички роботи з кількома платформами.

Таким чином, великою перевагою AWS є те, що вони пропонують майже всі послуги, які можуть знадобитися в більшості сучасних проектів.

Хмарний провайдер AWS запровадив одну з найкращих практик безпеки і відповідності. Це стосується не тільки фінансових даних, а й медичних. Проте аргумент, наведений вище щодо кількості наданих послуг, залишається актуальним.

2.1.3. Недоліки хмарного провайдера AWS

Великою перевагою використання відкритих протоколів і рішень з відкритим вихідним кодом є те, що ніхто не прив'язаний до конкретного інструменту чи конкретної компанії. Наприклад, використовується хмарна версія ElasticSearch, яка є інструментом, що надає широкі можливості пошуку. Навіть якщо компанія, яка надає цей інструмент, буде закрита або обліковий запис буде заблоковано, можна за 5–10 хвилин налаштувати власний примірник ElasticSearch у будь-якого постачальника VPS і продовжити роботу. Це можливо, тому що ElasticSearch є інструментом з відкритим кодом.

Проте, якщо використовується Amazon-клон ElasticSearch, який називається Amazon Open Search Service, не можна відразу використовувати його, оскільки є деякі відмінності. Потрібно розробити робочий процес міграції, що потребує певного часу та ресурсів. Така ж логіка стосується й інших сервісів AWS, тобто SQS.

Завдяки своїм перевагам AWS є дуже популярною платформою. Тому можна очікувати, що є багато людей, які шукають облікові дані, щоб вкрасти і заробити гроші. Платформа AWS має дуже хороший захист, і можна очікувати, що викрадення даних банківської картки є майже неможливим.

2.1.4. Головні веб-сервіси AWS

Провайдер AWS надає велику кількість сервісів і готових рішень для клієнта. Зараз кількість цих сервісів перевищує 500 одиниць, основні сервіси продемонстровано на рисунку 2.2.

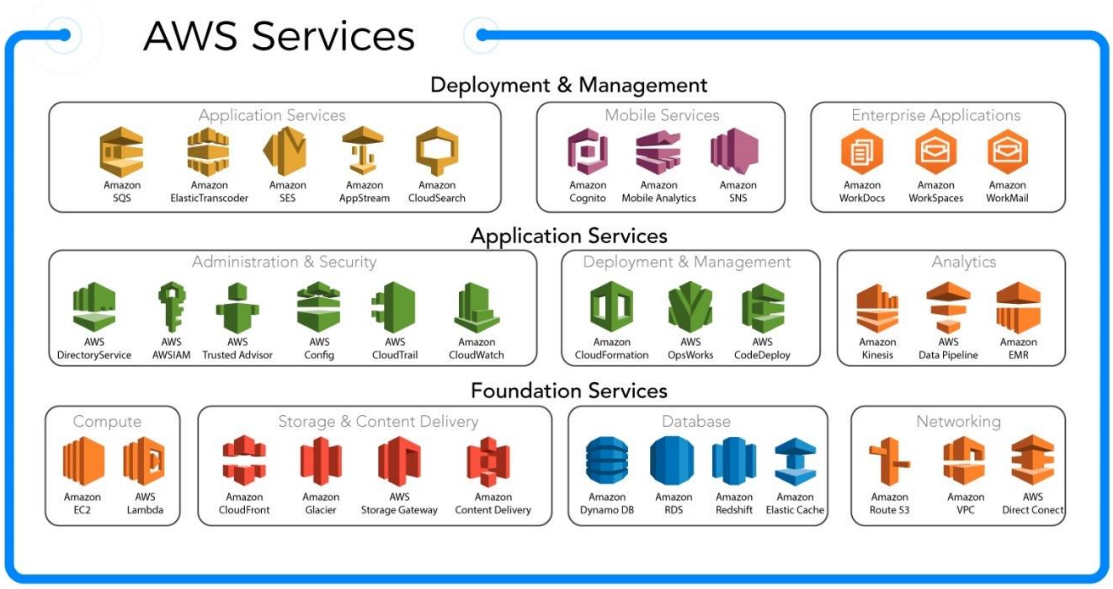


Рисунок 2.2 — Головні веб-сервіси AWS

Для розгортання розроблених систем, для аналізу і порівняння ефективності буде використано такі сервіси:

— сервіс для роботи з реляційними базами даних — RDS;

— файлове сховище даних для збереження об’єктів-файлів будь-якого типу і будь-якого розміру — S3;

— незалежне ізольоване середовище для виконання програмних інструкцій, яка може швидко масштабуватись — AWS Lambda;

— нереляційна таблиця, яка дозволяє зберігати дані в будь-якому форматі і не має чіткої схеми — DynamoDB;

— повністю керована черга, яка дозволяє отримувати, надсилати і зберігати повідомлення від інших сервіс — SQS;

— повністю керований сервіс для обміну повідомленнями між сервісами системи — SNS;

— інструмент для збору, обробка й аналіз потокових даних в режимі реального часу — Amazon Kinesis.

2.2. Розробка методології для порівняння ефективності підходів

Як було сказано вище, кількість даних на фізичних і цифрових носія нескінченно збільшується. Тому виникає задача ефективного і компактного зберігання даних у різних видах сховищ даних.

Ефективністю можна вважати значення, яке буде рівне обсягу даних поділене на вартість цих даних. Компактність даних буде залежати безпосередньо від формату зберігання і самого сховища даних.

Для роботи з цим ресурсом створено програмне забезпечення, яке дає можливість імпортувати дані з веб-ресурсу на локальну машину і експортувати дані з локальної машини далі до клієнтів. Це програмне забезпечення працює з консольним середовищем. Окремо було реалізовано програмне забезпечення, яке обробляє надіслані із консолі дані.

Для розробки системи та порівняння ефективності зберігання різних сховищ взято ресурс, який містить дані про гоночні заїзди машин. Кожен об’єкт містить загальну інформацію про стан машини, її положення в GPS-координатах, кількість палива в баку, тиск в шинах, час тощо. Найважливішим полем є саме час, оскільки за допомогою нього можна фільтрувати дані й перевіряти, в якому порядку клієнти отримують надіслані об’єкти.

Кожна машина може надіслати 5 записів даних у форматі JSON за одну секунду, тобто за одну хвилину одна машина надсилає 300 записів. Відповідно, кожен гоночний заїзд в середньому триває протягом однієї години і з 25 машинами. З цього можна зробити висновок, що один гоночний заїзд може містити в середньому 500000 записів даних.

На рисунку 2.3 подано приклад запису даних у форматі JSON. У цьому об’єктові міститься найголовніша інформація про стан машини в конкретний момент часу. Цей об’єкт може змінюватись в залежності від конфігурації машини.

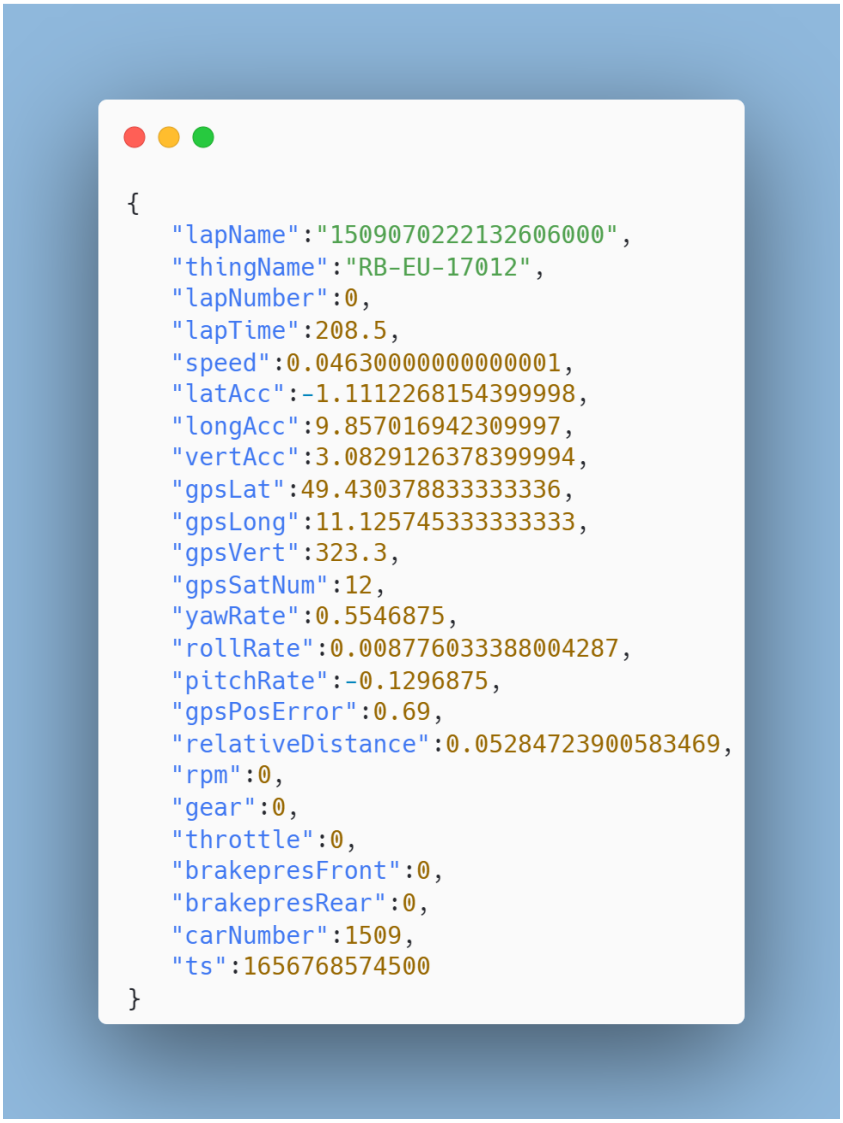


Рисунок 2.3 — Приклад даних для дослідження

За допомогою записів даних, які надсилаються до системи, можна дослідити ефективність зберігання даних у різних видах сховищ. Для того, щоб збільшити точність дослідження, варто надсилати дані в режимі реального часу. Кожен об’єкт містить поле з назвою “Timestamp”, яке містить час надсилання об’єкта. Цей старий час варто замінити на час комп’ютера в даний момент часу.

2.3. Порядок побудови дослідження

Найпершим кроком буде створення скриптів для роботи з веб-ресурсом, а саме для імпорту і експорту даних. Під час імпорту дані будуть записуватися в

локальну машину за допомогою SQLite, що є полегшеною реляційною системою керування базами даних. Її було обрано через те, що вона дає можливість структуровано зберігати дані безпосередньо в операційній системі. При експорті дані будуть зчитуватися саме з бази даних, побудованої на основі SQLite, а самі дані будуть надсилатися до наступних клієнтів для подальшої обробки.

Наступним кроком буде створення і побудова трьох рішень для трьох різних типів сховищ: реляційна база даних, нереляційна база даних і файлове сховище. Найефективнішим методом зберіганням даних буде саме файлове сховище, але воно більш складне для реалізації. Саме цьому типу буде приділено найбільше уваги під час цього дослідження.

Останнім кроком буде якраз запуск і порівняння цих всіх типів і їхньої ефективності.

Основною мовою програмування буде мова Python через її простоту, легкість вивчення, наявність дуже великої кількості готових бібліотек, які можуть розв’язувати безліч проблем, тощо. Ця мова також постійно оновлюється, виявляються різні помилки, пов’язані з безпекою і проблемами з використанням пам’яті комп’ютера.

Висновки до розділу 2

У даному розділі описано основні хмарні технології, на основі яких буде розроблено систему і буде проведено аналіз ефективності. Вибрано основну мову програмування Python для написання скриптів імпорту даних, експорту даних із зовнішнього ресурсу і для написання основної системи.

В якості тестових даних взято дані із гоночних заїздів. Кількість заїздів необмежена, а також має бути можливість симуляції заїзду в режимі реального часу.

Головна мета розділу — спланувати, які кроки потрібно виконати, щоб виміряти ефективність зберігання даних великого обсягу.

3. РОЗРОБКА ПРОТОТИПІВ СИСТЕМ ДЛЯ ДОСЛІДЖЕННЯ ЕФЕКТИВНОСТІ ЗБЕРІГАННЯ ДАНИХ

У даному розділі буде розглянуто основні компоненти, з яких складаються системи зберігання даних великого обсягу. Також буде продемонстровано реалізації систем, які зберігають дані великого обсягу в різні види сховищ. Кожна система має опис компонентів, які використовуються для реалізації. Буде розглянуто кожен вид сховища, використаний для побудови систем. Також буде обчислено вартість використання даних для кожного з видів сховища, які мають налаштування за замовчуванням. Буде розглянуто поняття озера даних і проаналізовано доцільність його використання.

3.1. Архітектура системи

У магістерському дослідженні реалізовано три варіанти зберігання даних для аналізу:

— система для зберігання даних у RDS;

— система для зберігання даних у DynamoDB;

— система для зберігання даних у S3 Bucket.

Кожна система має майже однакову архітектуру, крім цілі зберігання даних.

Спочатку проаналізуємо спільні компоненти. На рисунку 3.1 зображено загальну архітектуру системи.

Дана система буде складатись із трьох підсистем, які надсилатимуть, оброблюватимуть і зберігатимуть дані у різних видах сховищах. Система розраховує на необмежену кількість вхідних об’єктів, які будуть оброблюватись далі. Тобто систему можна вважати прикладом високонавантаженої і здатної до масштабування системи.

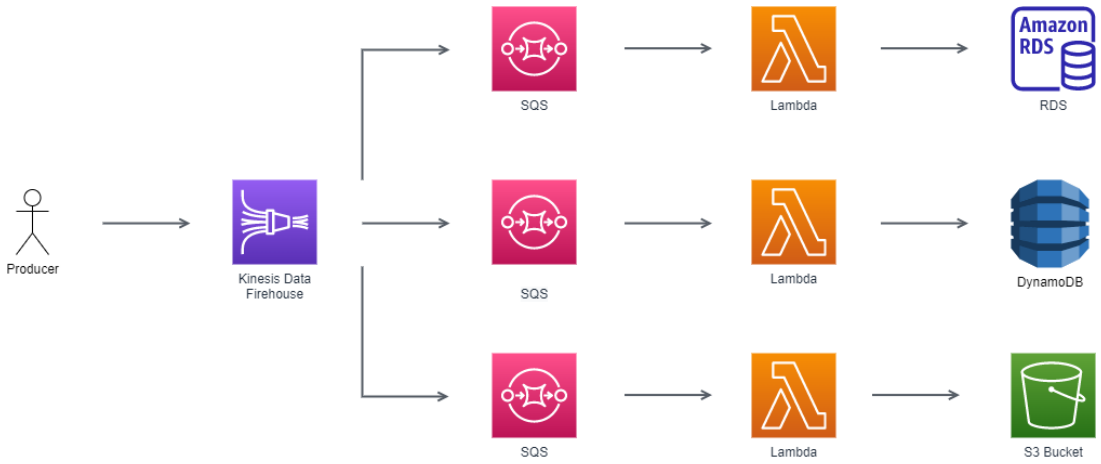


Рисунок 3.1 —Архітектура прототипів для зберігання даних

Кожна система використовує такі сервіси:

— сервіс Amazon Kinesis Data Firehose;

— сервіс Amazon Simple Queue Service;

— сервіс AWS Lambda.

Сервіс Amazon Kinesis — це веб-служба Amazon, призначена для обробки великомасштабних потоків даних з багатьох служб у режимі реального часу. Цей сервіс має аналог з відкритим початковим кодом — Apache Kafka, який також виступає брокером повідомлень. Це означає, що він працює як посередник між різними джерелами генерації даних. На рисунку 3.2 подано приклад роботи Amazon Kinesis [25].

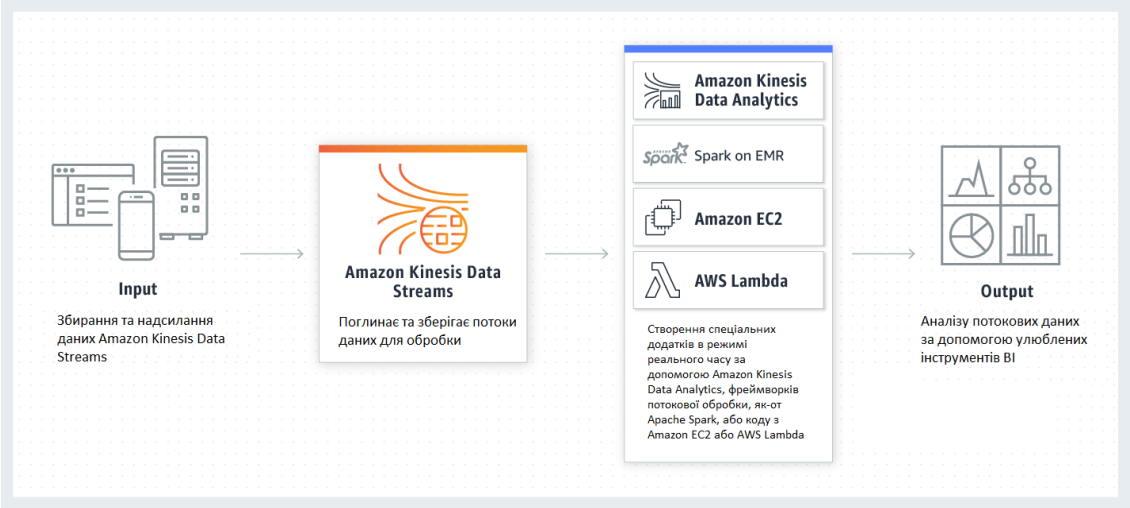


Рисунок 3.2 — Схема прикладу роботи Amazon Kinesis

Одна з ключових переваг сервісу Kinesis (і Kafka) полягає в тому, що він дає можливість обробляти й аналізувати дані майже миттєво, замість того, щоб чекати, поки надійде повний набір даних, потім обробляти його і надсилати на аналіз. Статистичні дані можна отримати за хвилини, а не за години, дні чи тижні. Сервіс Kinesis робить це можливим без тижнів складного налаштування, оскільки він поставляється як керована платформа, тобто не потрібно керувати жодною інфраструктурою. Сервіс Amazon Kinesis Data Firehose — це служба вилучення, перетворення та завантаження (ETL), яка забезпечує надійний збір, перетворення і доставку потокових даних в озера даних, сховища даних та аналітичні служби. В даному дисертаційному дослідження цей сервіс буде використано для розподілення подій між іншими споживачами даних [26].

Між Kinesis і Kafka є багато подібностей, а також відмінностей. Обидва призначені для прийому та обробки кількох великомасштабних потоків даних з великою гнучкістю щодо джерела. Обидва замінюють традиційні брокери повідомлень у середовищах, які приймають великі потоки даних, що потрібно обробити та доставити іншим програмам і службам [27].

Найбільша різниця між ними полягає в тому, що Amazon Kinesis — це керований сервіс, який вимагає мінімального налаштування. Сервіс Kafka — це рішення з відкритим початковим кодом, яке потребує значних інвестицій і знань для налаштування, часто вимагаючи тижнів, а не годин.

Сервіси Kafka і Kinesis виконують схожі функції та дають подібні результати, але працюють по-різному. Ключові поняття, які використовує Kinesis, включають виробників даних, споживачів даних, потоки даних, ключі тощо [28].

Сервіс AWS Lambda — це безсерверний обчислювальний сервіс, який дає можливість запускати код без керування або надання серверів [29]. Служба AWS Lambda використовує високодоступну, еластичну інфраструктуру для виконання коду. Вона виконує завдання з адміністрування обчислювальних ресурсів, такі як автоматичне масштабування та надання потужності, обслуговування операційної системи і сервера. Цей сервіс може запускати код майже для будь-якої форми

серверної програми або служби додатків. На рисунку 3.3 відображено схему роботи AWS Lambda.

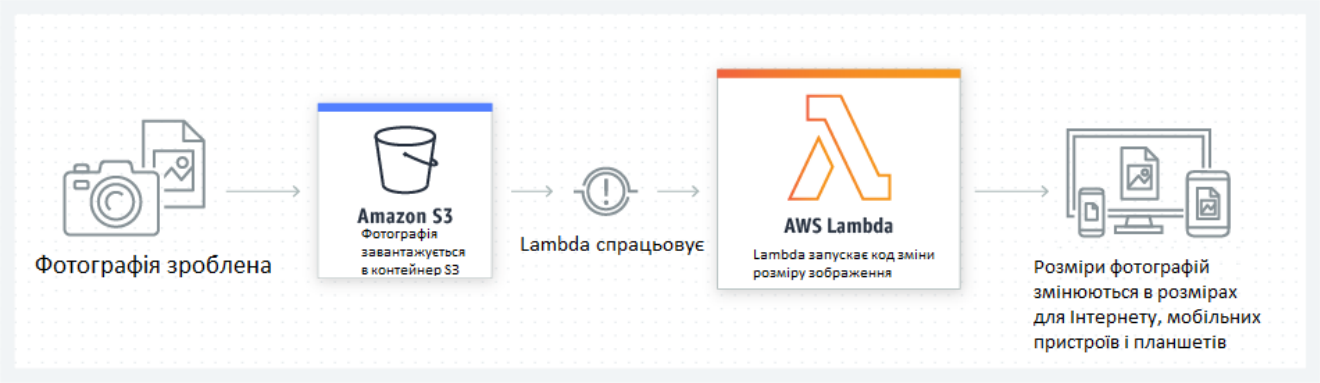


Рисунок 3.3 — Схема прикладу роботи сервісу AWS Lambda

У безсерверній моделі код має бути організований у функції. Сервіс Lambda запускає функції лише тоді, коли це необхідно. Служба автоматично масштабується, якщо отримує від кількох сотень запитів до тисячі запитів навіть протягом однієї секунди. Платити треба лише за використаний час обчислення, якщо код не виконується, оплата не стягуватиметься.

Сервіс AWS Lambda може запускати функції, коли вони запускаються подіями з різних служб AWS.

Компонентами AWS Lambda є:

— лямбда-функції — автономні функції, які викликаються рушієм AWS Lambda та завершуються після завершення роботи;

— функції упаковки — відповідають за стиснення лямбда-функцій, включаючи їхні залежності, і передачу до AWS. Функцію потрібно спочатку архівувати та завантажити в сегмент S3.

Модель виконання AWS Lambda:

— контейнер — операційна одиниця, яка містить функцію Lambda і використовує ресурси, керовані AWS, для запуску функції;

— функції без стану — кожна лямбда-функція оптимізована і контролюється AWS і, як правило, викликається один раз для кожного примірника контейнера;

— події — запити, які обслуговуються одним примірником функції Lambda. Події, як правило, надходять з інших частин екосистеми Amazon — наприклад, з таких баз даних, як DynamoDB, події CloudWatch або зміни файлів на S3 — і запускають функцію Lambda, яка обробляє події.

Сервіс Amazon SQS (Simple Queue Service) — це повністю керована служба, популярна завдяки своїй функції черги повідомлень. Розробники часто надсилають і отримують повідомлення різних типів і розмірів асинхронно за допомогою цієї служби. Amazon SQS дає змогу зберігати окремі мікросервіси і розподілені системи окремо одна від одної. Крім того, це дає можливість масштабувати ці системи та автоматизує процес створення і підтримки черг повідомлень. На рисунку 3.4 подано схему роботи AWS SQS [30].

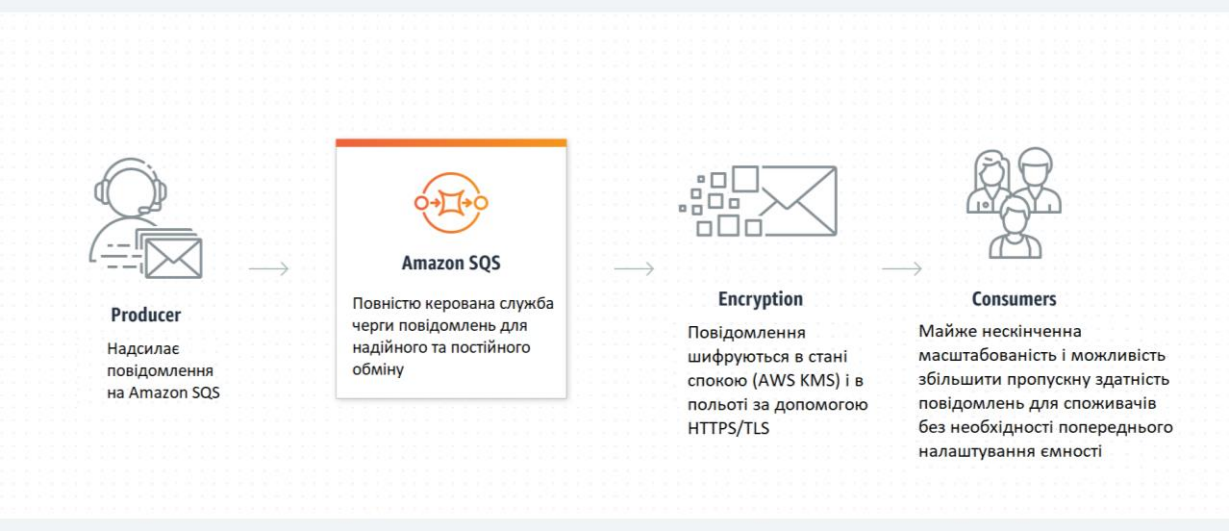


Рисунок 3.4 — Схема прикладу роботи AWS SQS

Сервіс Amazon SQS забезпечує безпечний перехід для встановлення зв’язку між програмними компонентами. Крім того, Amazon SQS розгортає програмний інтерфейс, який дає можливість отримувати доступ і використовувати різні веб-сервіси за допомогою популярних мов програмування. Сервіс Amazon SQS

пропонує свої засоби для завдань, які працюють асинхронно. Це означає, що Amazon SQS спрощує роботу програми, яка може просто надсилати повідомлення в чергу замість того, щоб безпосередньо намагатися викликати іншу програму окремо. Таким чином, кілька програм можуть легко переглядати це повідомлення.

Функціями, як роблять AWS SQS популярним рішенням для керування повідомленнями серед розробників, є такі:

— надійність: Amazon SQS покладається на кілька серверів для безпечного зберігання повідомлень. Крім того, його стандартні черги та черги FIFO працюють за системою доставки повідомлень принаймні один раз і рівно один раз;

— доступність: Amazon SQS використовує надлишкову інфраструктуру для створення високошвидкісного одночасного обміну повідомленнями;

— масштабованість: Amazon SQS пропонує високу масштабованість, оскільки може незалежно обробляти всі запити на буферизовані повідомлення. Крім того, сервіс може прозоро масштабуватися при раптових стрибках навантаження і не потребує жодних попередніх інструкцій;

Розглянемо безпосередньо сховища даних, їхні характеристики і використання.

3.1.1. Сервіс Amazon RDS

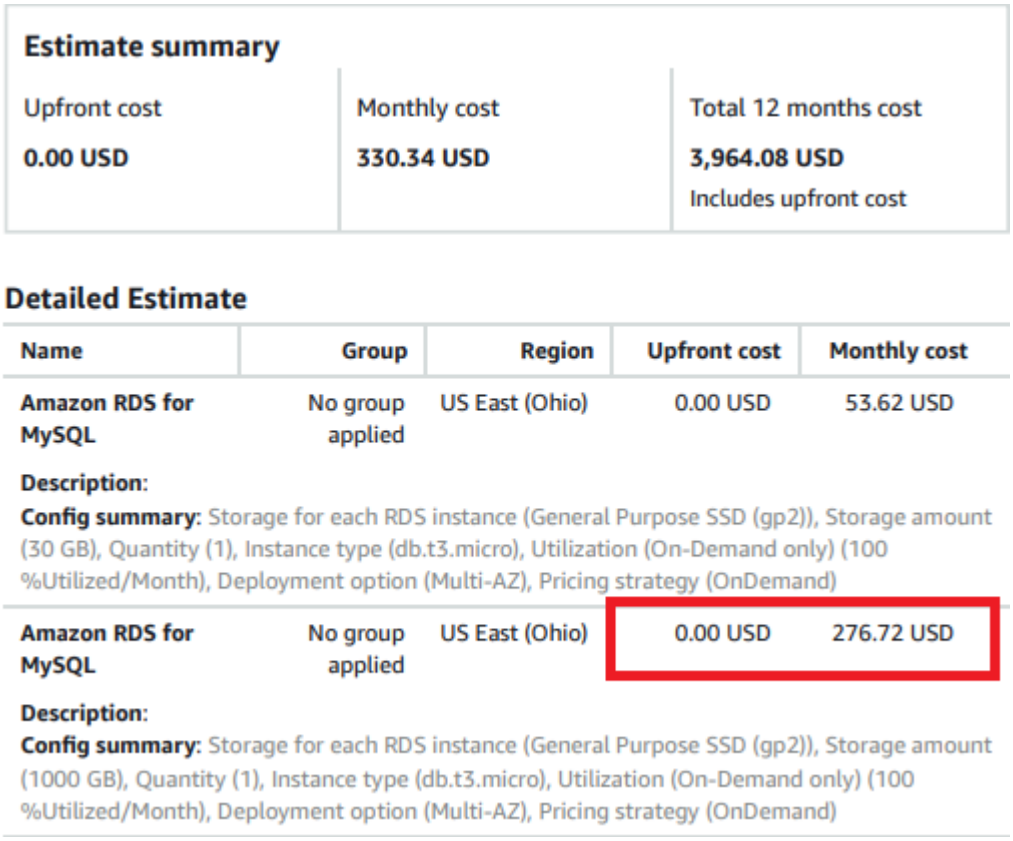
Сервіс RDS — одна з найпопулярніших служб в AWS, яка має широке коло клієнтів, що прагнуть зменшити залежність від своїх адміністраторів баз даних і надати можливість своєму штатові працювати з більшою кількістю баз даних, ніж вони могли раніше [31].

Процеси адміністрування, такі як виправлення бази даних RDS, резервне копіювання баз даних і відновлення на певний момент, автоматизовані.

Клієнти, які використовують примірник RDS, мають обмежений контроль над основною інфраструктурою та операційною системою. Запобігаючи кореневому доступу до вузла, AWS обмежує клієнтів щодо встановлення будь-якого стороннього програмного забезпечення на вузлі, включаючи спеціальне

програмне забезпечення для шифрування бази даних або програмне забезпечення для доставки журналів, які можуть потребувати кореневого доступу до вузла.

Для того, щоб почати записувати дані в це сховище, спершу треба створити сервер з базою даних. У даному випадку це буде сервер з типом micro (db.t3.micro) і для роботи з MySQL, хоч можна вибрати будь-яку СУБД. 1000 Гб цього сховища на місяць буде коштувати приблизно 277 доларів. На рисунку 3.5 подано розрахунки вартості використання.



Estimate summary

Upfront cost	Monthly cost	Total 12 months cost
0.00 USD	**330.34 USD**	**3,964.08 USD** Includes upfront cost

Detailed Estimate

Name	Group	Region	Upfront cost	Monthly cost
Amazon RDS for MySQL	No group applied	US East (Ohio)	0.00 USD	53.62 USD
Description: **Config summary:** Storage for each RDS instance (General Purpose SSD (gp2)), Storage amount (30 GB), Quantity (1), Instance type (db.t3.micro), Utilization (On-Demand only) (100 %Utilized/Month), Deployment option (Multi-AZ), Pricing strategy (OnDemand)				
Amazon RDS for MySQL	No group applied	US East (Ohio)	0.00 USD	276.72 USD
Description: **Config summary:** Storage for each RDS instance (General Purpose SSD (gp2)), Storage amount (1000 GB), Quantity (1), Instance type (db.t3.micro), Utilization (On-Demand only) (100 %Utilized/Month), Deployment option (Multi-AZ), Pricing strategy (OnDemand)				

Рисунок 3.5 — Обчислення вартості використання MySQL

Для того, щоб почати працювати з базою даних, потрібно також створити таблицю для збереження даних. Таблиця буде містити колонки “body” (тип Varchar; містить інформацію у форматі JSON), ”timestamp” (тип LongInt; epoch time в мілісекундах) і “car_number” (тип Int; унікальний номер машини).

Для взаємодії з MySQL використано бібліотеку PyMysql.

На рисунку 3.6 подано приклад роботи з бібліотекою PyMySQL для роботи з базою даних.

```python
def main(event: dict):
    messages = event.get("Records")
    if not messages:
        logger.error("Didn't receive any records from SQS")
        return

    conn = pymysql.connect(host=host, port=port, user=user, password=password, database=database)
    with conn.cursor() as cur:
        for message in messages:
            row_rundata = json.loads(message["body"])
            timestamp = int(row_rundata.pop("Timestamp"))
            car_number = row_rundata.pop("CarNumber")
            lap_number = row_rundata.pop("lapName")
            cur.execute(
                "INSERT INTO rundata (car_number, `timestamp`, lap_number, json) VALUES (%s, %s, %s, %s)",
                (car_number, timestamp, lap_number, json.dumps(row_rundata))
            )
        conn.commit()

    conn.close()
```

Рисунок 3.6— Приклад роботи програми з MySQL

З рисунка 3.6 видно, що використання драйвера для роботи з базою даних є не складним. Для цього потрібно тільки вказати основну інформацію про базу даних.

3.1.2. Сервіс DynamoDB

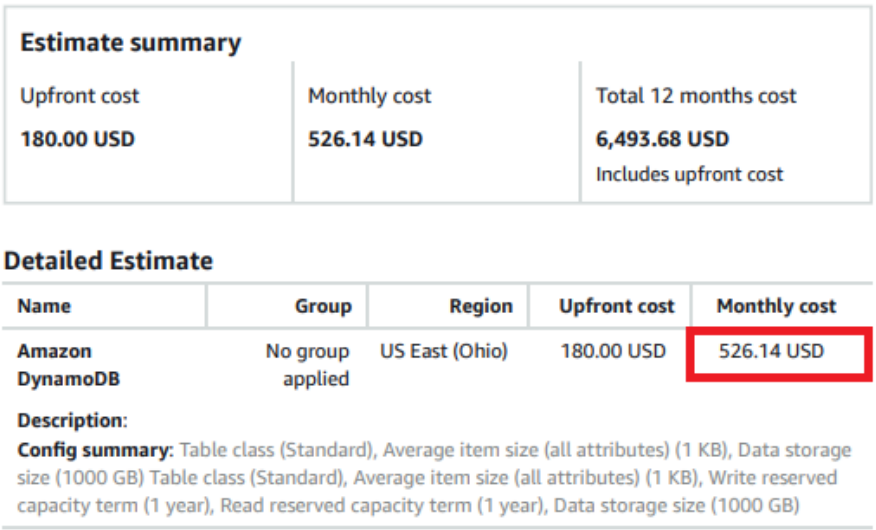
Сервіс Amazon DynamoDB — це повністю керована (“безсерверна”) служба баз даних NoSQL (нереляційна), доступна в Amazon Web Services. Сервіс DynamoDB має високу масштабованість, а це означає, що цей сервіс може динамічно розширюватися залежно від навантаження, не потребуючи повторного розгортання чи зміни архітектури. Служба також пропонує гнучку модель, яка використовує автоматичне масштабування пропускної здатності. Це означає, що сервіс масштабує обчислювальну потужність на основі попиту, заощаджуючи гроші та знижуючи початкові витрати. Це робить сервіс чудовим додатком для мобільних пристроїв, ігор, Інтернету речей та інших додатків, які швидко розвиваються і мають великий обсяг [32].

Служба Amazon DynamoDB має численні переваги порівняно з іншими системами керування базами даних NoSQL, такими як Apache Cassandra та MongoDB. Інтеграція між DynamoDB та іншими службами AWS особливо корисна.

Оскільки це безсерверна служба бази даних, налаштувати її легко. Треба відкрити консоль керування AWS і скористатися майстром для налаштування. І навпаки, щоб налаштувати локальний екземпляр MongoDB, потрібно виконати довгий список інструкцій і, можливо, доведеться виправити помилки автентифікації.

Безпека для DynamoDB регулюється сервісом для роботи з ролями AWS Identity and Access Management (IAM). Можна використовувати інші функції безпеки AWS, щоб покращити засоби керування. Незважаючи на те, що MongoDB безпечний, у минулому траплялися порушення безпеки через неправильне налаштування та керування.

Для того, щоб почати записувати дані в це сховище, спершу треба створити нову таблицю — це звичайна таблиця з налаштуваннями за замовчуванням. 1000 Гб цього сховища на місяць буде коштувати приблизно 526 доларів. На рисунку 3.7 подано розрахунки вартості використання бази даних.



Estimate summary

Upfront cost	Monthly cost	Total 12 months cost
180.00 USD	**526.14 USD**	**6,493.68 USD** Includes upfront cost

Detailed Estimate

Name	Group	Region	Upfront cost	Monthly cost
Amazon DynamoDB	No group applied	US East (Ohio)	180.00 USD	526.14 USD

Description:
Config summary: Table class (Standard), Average item size (all attributes) (1 KB), Data storage size (1000 GB) Table class (Standard), Average item size (all attributes) (1 KB), Write reserved capacity term (1 year), Read reserved capacity term (1 year), Data storage size (1000 GB)

Рисунок 3.7 — Обчислення вартості використання DynamoDB

Автоматичне масштабування за розкладом забезпечує високу доступність, відмовостійкість і економічність налаштування. Визначити політику масштабування та автоматичне масштабування в DynamoDB досить просто.

Для того, щоб почати працювати з DynamoDB, потрібно також створити таблицю для збереження даних. Таблиця буде містити колонки “Body” (тип String; містить інформацію у форматі JSON), “Timestamp” (тип Integer; epoch time в мілісекундах) і “CarNumber” (тип Integer; унікальний номер машини).

Для взаємодії з DynamoDB було використано бібліотеку boto3, застосування якої продемонстровано на рисунку 3.8.

```python
dynamodb = boto3.resource('dynamodb')
table = dynamodb.Table('rundata')


def main(event: dict):
    messages = event.get("Records")
    if not messages:
        logger.error("Didn't receive any records from SQS")
        return

    for message in messages:
        row_rundata = json.loads(message["body"])
        timestamp = int(row_rundata.pop("Timestamp"))
        car_number = row_rundata.pop("CarNumber")
        lap_number = row_rundata.pop("lapName")
        table.put_item(
            Item={
                "timestamp": timestamp,
                "carNumber": car_number,
                "lapNumber": lap_number,
                "json": json.dumps(row_rundata)
            }
        )
```

Рисунок 3.8 — Приклад роботи програми з DynamoDB

З рисунка 3.8 видно, що використання бібліотеки для роботи з DynamoDB є досить простим.

3.1.3. Файлове сховище S3 Bucket

Сервіс Amazon Simple Storage Service (S3) — це широкомасштабована служба зберігання, яка базується на технології зберігання об’єктів [33]. Сервіс забезпечує дуже високий рівень довговічності, високу доступність і високу

продуктивність. Доступ до даних можна отримати з будь-якого місця через Інтернет, консоль Amazon і потужний API S3.

Сховище S3 надає такі ключові функції:

— бакети — дані зберігаються у бакетах. Кожен бакет може зберігати необмежену кількість неструктурованих даних;

— класична масштабованість — S3 не має обмежень на пам'ять. Окремі об'єкти можуть мати обсяг до 5 ТБ;

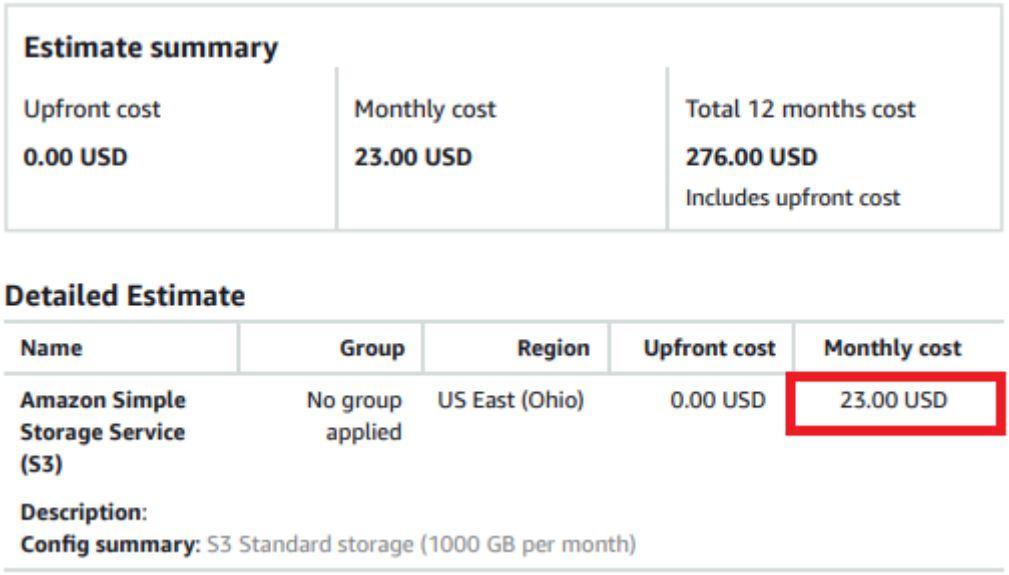
— гнучка структура даних — кожен об'єкт ідентифікується за допомогою унікального ключа, і можна використовувати його метадані для гнучкої організації даних;

— завантаження даних — легко обмінюватися даними з будь-ким в організації чи за її межами та дозволити їм завантажувати дані через Інтернет.

Дані Amazon S3 зберігаються як об'єкти. Такий підхід забезпечує високомасштабоване зберігання в хмарі. Об'єкти можна розміщувати на різноманітних фізичних дисках, розподілених по всьому центру обробки даних. Центри обробки даних Amazon використовують спеціалізоване обладнання, програмне забезпечення та розподілені файлові системи, щоб забезпечити справжню еластичну масштабованість.

Провайдер Amazon забезпечує резервування і контроль версій за допомогою методів блокового зберігання. Дані автоматично зберігаються в кількох місцях, розподіляються на кількох дисках, а в деяких випадках — у кількох зонах або регіонах доступності. Служба Amazon S3 періодично перевіряє цілісність даних, перевіряючи їхнє контрольне хеш-значення. Якщо виявлено пошкодження даних, для відновлення об'єкта використовуються надлишкові дані.

Для того, щоб почати записувати дані в S3, спершу треба створити новий бакет (набір елементів хеш-таблиці). У даному випадку це буде звичайний бакет із стандартними налаштуваннями. 1000 Гб цього сховища на місяць буде коштувати приблизно 26 доларів. На рисунку 3.9 подано розраховану вартість використання цього сховища даних. Порівняно з попередніми сховищами це дуже мало.



Estimate summary

Upfront cost	Monthly cost	Total 12 months cost
0.00 USD	**23.00 USD**	**276.00 USD** Includes upfront cost

Detailed Estimate

Name	Group	Region	Upfront cost	Monthly cost
Amazon Simple Storage Service (S3)	No group applied	US East (Ohio)	0.00 USD	23.00 USD

Description:
Config summary: S3 Standard storage (1000 GB per month)

Рисунок 3.9 — Обчислення вартості використання S3

Щоб почати працювати з S3 bucket, потрібно використати бібліотеку boto3, а саме потрібно взаємодіяти з клієнтом для S3. Це сховище приймає файли будь-якого типу: JSON, CSV, текстові файли тощо. Задача дисертації полягає в тому, щоб дані були структурованими і компактними. Тому буде використовуватися формати Parquet і будуть записуватися дані в цей файл з цим форматом за допомогою бібліотек Apache Arrow і pandas. На рисунку 3.10 подано приклад використання цих бібліотек.

```python
async def store_parquet_file(client, rows: Iterable, bucket_name: str, key: str) -> str:
    if not rows:
        logger.error("Don't have rows")
        return ''

    file_path = f"/tmp/{key[-16:]}"

    df = pd.DataFrame(rows)
    df.set_index(["timestamp"], inplace=True)
    table = pa.Table.from_pandas(df)
    pq.write_table(table, file_path)

    try:
        await client.upload_file(file_path, bucket_name, key)
    except Exception as error:
        logger.exception(f"Unable to s3 upload {key}: {error} ({type(error)})")
    finally:
        os.remove(file_path)

    return key[15:39]
```

Рисунок 3.10 — Приклад роботи програми з S3

З рисунка 3.10 видно, що, щоб зберегти об’єкт даних, потрібно перетворити його в об’єкт DataFrame і після цього за допомогою PyArrow [34] перетворити цей об’єкт у файл з форматом Apache Parquet.

3.2. Концепція Data Lake

Цей термін став популярним відносно недавно. Суть підходу з організацією озера даних полягає в тому, що всі дані з різних джерел “стікаються” в одне центральне сховище, ніби багато річок впадає в одне велике озеро [35]. Прийнято вважати, що дані туди потрапляють у своєму первісному вигляді, тому немає сенсу заздалегідь заготовляти структуровані таблиці і очищати дані перед укладанням, як це прийнято в традиційних DWH-системах. Тобто буде отримано велике сховище з файлами різних форматів: JSON, CSV, Parquet, Avro та інших. На рисунку 3.11 подано приклад архітектури для побудови Data Lake.

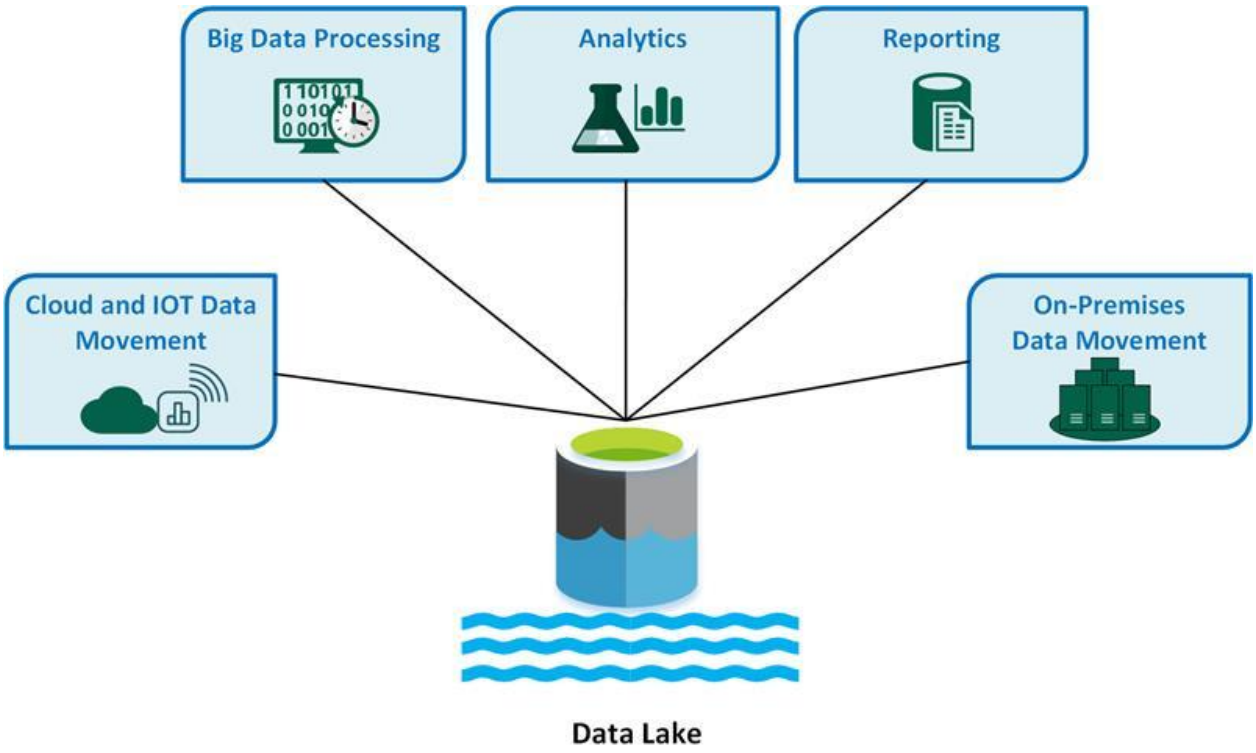


Рисунок 3.11 — Приклад архітектури Data Lake

Роботу над Hadoop було розпочато в 2006 році. Революційну технологію, яка змінила спосіб зберігання та обробки даних, яка раніше не містилися на

одному комп’ютері. Мільйони компаній по всьому світу отримали можливість зберігати й аналізувати терабайти даних. Платформа Hadoop є набором компонентів для зберігання та аналізу даних на кластері з сотень і навіть тисяч комп’ютерів. В основі лежить розподілена файлова система HDFS (Hadoop File System) — величезна файлова система розподілена на тисячі комп’ютерів, але клієнт працює з нею, як з одним фізичним диском на комп’ютері клієнта.

Пізніше розробники зрозуміли, що бажано навчитися отримувати користь з цих даних мінімальною кількістю зусиль. Безперечним лідером у змаганні Big Data був SQL. Так виник проект Apache Hive, який представляє собою ряд компонентів, включаючи СУБД над Hadoop і SQL-подібний двигун. Тепер у Big Data-спільноти з’явилася можливість виконувати SQL-запити, наприклад, на звичайних CSV-файлах.

Реалізація Data Lake засобами хмарних систем має ряд помітних переваг:

— швидкість розробки та отримання цінності від даних (вагомий плюс);

— відпадає необхідність супроводу систем, тому що це робить хмарний провайдер;

— надійність і безпека даних (тут необхідно розбиратися в налаштуваннях конкретного провайдера);

—стандарти безпеки (часто хмарні провайдери відповідають різним стандартам типу PCI DSS тощо).

Щодо недоліків, то можна вказати такі:

— дорожнеча (спірний момент, враховуючи вартість in-house фахівців та обладнання);

— законодавчі обмеження (наприклад, обмеження зберігання персональних даних);

— залежність від хмарного провайдера.

Основними сервісами, на основі яких будується озеро даних, є:

— сервіс Amazon S3;

— сервіс Amazon Glue;

— сервіс Amazon Athena.

Сервіс Amazon Glue — набір компонентів для задач типу ETL — Extract, Transform, Load [36]. Основними компонентами Glue є:

— сховище метаданих (Data Catalog);

— двигун ETL, що генерує код мовою Python і Scala для запуску завдань;

— планувальник завдань.

В екосистемі Hadoop Amazon Glue частково виконує функції Apache Hive. Наприклад, Amazon Glue Data Catalog — це аналог Hive Metastore.

Сервіс Amazon Athena — це SQL-двигун на основі Apache Presto [37]. Він може працювати з багатьма форматами файлів, включаючи Apache Parquet, Apache Avro, ORC, CSV, JSON. Оплата з боку Amazon йде лише за обсяг прочитаних даних. На момент написання роботи це $5 за 1 ТБ даних. Така цінова модель змушує розробників і дата-інженерів ефективно працювати з даними.

Крім того, розглянемо різні формати зберігання. По-перше, ціна за сервіс Amazon S3 безпосередньо залежить від обсягу даних, які зберігаються. Наприклад, текстові дані при зберіганні необхідно стиснути. По-друге, ціна за Amazon Athena прямопропорційна обсягові прочитаних даних. Той факт, що Athena вміло справляється з розпакуванням gzip-файлів, не зменшує обсяг інформації, що читається з архіву. Наприклад, якщо зберігається csv-файл розміром 1 Гб на S3 в стисненому вигляді, то після розпакування файлу Amazon Athena все одно потрібно прочитати той самий гігабайт для виконання аналітичного запиту.

З розвитком великих даних з’явилися ефективні засоби зберігання та читання даних. Одним із таких форматів є Apache Parquet [38].

Формат Apache Parquet — це формат файлів даних з відкритим початковим кодом, орієнтований на колонки, призначений для ефективного зберігання та пошуку даних. Він забезпечує ефективне стиснення даних і схеми кодування з підвищеною продуктивністю для масової обробки складних даних [39].

На рисунку 3.12 зображено схему роботи формату Apache Parquet.

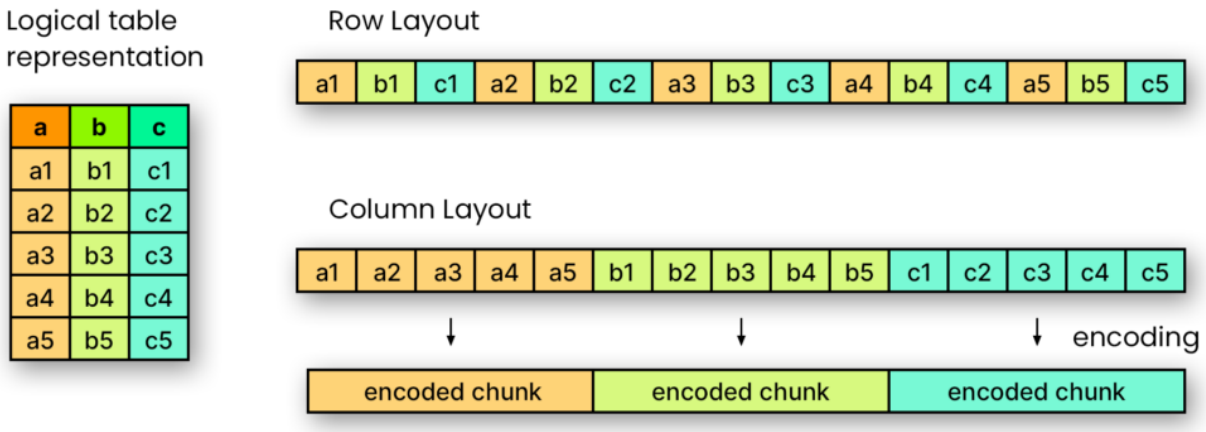


Рисунок 3.12 — Демонстрація роботи формату Apache Parquet

Формат Apache Parquet розроблено як загальний формат обміну як для пакетних, так і для інтерактивних робочих навантажень. Він схожий на інші формати файлів з колонками зберігання, доступні в Hadoop, а саме RCFile і ORC.

Характеристика формату Apache Parquet:

— безкоштовний формат файлу з відкритим кодом;

— формат на основі колонок — файли організовані за колонками, а не за рядками, що економить місце для зберігання і пришвидшує аналітичні запити;

— високоефективне стиснення та декомпресія даних;

— підтримує складні типи даних і розширені вкладені структури даних.

Перевагами використання цього формату є:

— добре підходить для зберігання великих даних будь-якого типу (таблиці структурованих даних, зображення, відео, документи);

— заощаджує простір у хмарному сховищі завдяки високоефективному стисненню колонок і гнучким схемам кодування для колонок з різними типами даних;

— збільшення пропускної здатності і продуктивності даних за допомогою таких методів, як пропуск даних.

Формат Apache Parquet реалізовано за допомогою алгоритму подрібнення записів і складання, який враховує складні структури даних, які можна

використовувати для зберігання даних. Формат Parquet оптимізовано для масової роботи зі складними даними, він пропонує різні способи ефективного стиснення даних і типів кодування. Цей підхід найкращий, особливо для тих запитів, які потребують читання певних колонок з великої таблиці. Формат Parquet може читати лише необхідні колонки, що значно мінімізує IO-пам’ять, яку використовує комп’ютер.

Формат CSV — це простий і поширений формат, який використовується багатьма інструментами, такими як Excel, Google-Таблиці та багатьма іншими. Незважаючи на те, що файли з форматом CSV за замовчуванням використовуються для конвеєрів обробки даних, формат має певні недоліки:

— сервіси Amazon Athena та Spectrum братимуть плату залежно від кількості сканованих даних за запит;

— провайдери Google і Amazon братимуть плату відповідно до обсягу даних, які зберігаються на GS/S3.

У таблиці 3.1 порівнюються економія, а також прискорення, отримані завдяки перетворенню даних в Apache Parquet із CSV-формату.

Таблиця 3.1

Результати ефективності зберігання даних великого обсягу

Набір даних	Розмір на Amazon S3	Час виконання запиту	Дані скановано	Вартість
Дані зберігаються у вигляді файлів CSV	1 ТБ	236 секунд	1,15 ТБ	5,75 доларів США
Дані зберігаються у форматі Apache Parquet	130 ГБ	6,78 секунди	2,51 ГБ	0,01 долара США
Економія	На 87% менше при використанні паркету	34х швидше	На 99% менше сканованих даних	99,7% економії

Формат Apache Parquet допоміг своїм користувачам зменшити вимоги до сховища принаймні на одну третину для великих наборів даних, крім того, він значно покращив час сканування та десеріалізації, отже, загальні витрати.

Також Athena підтримує розділення даних за будь-якою колонкою/колонками. Достатньо в S3 укласти дані в потрібній структурі і при створенні таблиць в Glue вказати колонку, за якою йде розділення даних.

При цьому розділення даних призводить до:

— скорочення GET-запитів до S3, отже, зменшення вартості;

— швидкість виконання запитів збільшується;

— зменшується обсяг даних, які читаються в Amazon Athena.

Висновки до розділу 3

В даному розділі розглянуто три підходи до збереження даних великого обсягу: реляційну базу даних, нереляційну базу даних і файлове сховище даних. Також побудовані системи для збереження даних із гоночних заїздів і розглянуто вартість використання сервісу.

Описано кожен вид сховища, визначено вартість користування цими сервісами. Після порівняння було визначено, що найефективнішим видом сховища для великих даних є файлові сховища. У даній роботі в ролі файлового сховища використано сервіс Amazon S3 Bucket.

Отже, можна зробити висновок, що використання файлового сховища буде найбільш вигідним. Це сховище є ефективним тому, що вартість 1 Гб коштує 0.026 долара. Порівняно з іншими сховищами це дуже вигідна вартість. Проте, робота з файловим сховищем є більш складною і вимагає більше досвіду для побудови.

Розглянуто поняття “озера даних”, яке використовують для систем, що зберігають дані у файловому сховищі S3, наприклад, у форматі Apache Parquet, а також зчитують інформацію за допомогою SQL-синтаксу, завдяки сервісу AWS Athena.

4. ТЕСТУВАННЯ СИСТЕМ І ЗБІР СТАТИСТИКИ

У даному розділі буде розглянуто, як відбувалося тестування коректності і правильності функціонування кожної із систем. Також буде подано графіки і записи роботи з відповідного сервісу AWS Cloudwatch [40]. Для тестування системи буде використано програму для експорту даних, які були відповідно імпортовані з Інтернет-ресурсу. Скрипт для експорту даних буде надсилати дані у сервіс Amazon Kinesis, який буде надсилати об’єкти у три різні черги SQS, які будуть передавати дані у середовища для виконання коду — AWS Lambda.

На рисунку 4.1 зображено використання бібліотек для надсилання даних. Після виконання програми дані буде збережено у сховищах, де можна буде побачити результат обробки даних

```python
async def process_message(executor: ThreadPoolExecutor, storage: Storage, client, offset, limit):
    coroutines = []
    entries = []
    number = 0

    for row in storage.get_rundata(FIELDS, offset, limit):
        if len(entries) == BATCH_MAX_SIZE:
            coroutines.append(send_messages_async(executor, client, entries))
            entries = []

        timestamp, car_id, json_string = row
        entries.append({
            "Id": str(number),
            "MessageBody": json_string
        })
        number += 1

    if len(entries) > 0:
        coroutines.append(send_messages_async(executor, client, entries))

    await asyncio.gather(*coroutines)
```

Рисунок 4.1 — Скрипт для надсилання і обробки даних в Amazon Kinesis

4.1. Тестування системи, яка зберігає дані в DynamoDB

Для тестування правильності функціювання роботи системи було запущено скрипт для експорту даних. У середньому було надіслано 500 тисяч об’єктів у форматі JSON, які будуть оброблені і збережені. Код програми подано в розділі 3 на рисунку 3.8.

Після запуску програми, AWS Lambda записує свою роботу безпосередньо у сервісі AWS Cloudwatch. За допомогою цього сервісу можна побачити, що програма завершила свою обробку успішно. На рисунку 4.2 показано результат роботи виконання запиту для отримання записів роботи програми. Кожен запуск програми створює нове ізольоване середовище, яке виконує відповідні програмні інструкції. Кожен цей запуск має унікальний ідентифікатор (значення генерується за стандартом UUID), який допомагає правильно збирати записи роботи програми.

▶	2022-10-24T01:26:36.921+03:00	START RequestId: 5c49badc-7e7e-5be2-9488-e113bfbe0ba7 Version: $LATEST
▶	2022-10-24T01:27:36.972+03:00	END RequestId: 5c49badc-7e7e-5be2-9488-e113bfbe0ba7
▶	2022-10-24T01:27:36.972+03:00	REPORT RequestId: 5c49badc-7e7e-5be2-9488-e113bfbe0ba7 Duration: 60050.59 ms Billed Duration: 60000 ms Memory Size: 1024 MB M…
▶	2022-10-24T01:27:36.972+03:00	2022-10-23T22:27:36.972Z 5c49badc-7e7e-5be2-9488-e113bfbe0ba7 Task timed out after 60.05 seconds
▶	2022-10-24T01:27:40.274+03:00	START RequestId: c38b3649-deb2-5f39-9c94-6d861f73cf7c Version: $LATEST
▶	2022-10-24T01:27:45.389+03:00	END RequestId: c38b3649-deb2-5f39-9c94-6d861f73cf7c
▶	2022-10-24T01:27:45.389+03:00	REPORT RequestId: c38b3649-deb2-5f39-9c94-6d861f73cf7c Duration: 5115.08 ms Billed Duration: 5116 ms Memory Size: 1024 MB Max…
▶	2022-10-24T01:27:45.786+03:00	START RequestId: 951e0f64-34ac-5d59-a3e3-54abf6b4f0e2 Version: $LATEST
▶	2022-10-24T01:28:45.850+03:00	END RequestId: 951e0f64-34ac-5d59-a3e3-54abf6b4f0e2
▶	2022-10-24T01:28:45.850+03:00	REPORT RequestId: 951e0f64-34ac-5d59-a3e3-54abf6b4f0e2 Duration: 60063.43 ms Billed Duration: 60000 ms Memory Size: 1024 MB M…
▶	2022-10-24T01:28:45.850+03:00	2022-10-23T22:28:45.850Z 951e0f64-34ac-5d59-a3e3-54abf6b4f0e2 Task timed out after 60.06 seconds
▶	2022-10-24T01:28:46.749+03:00	START RequestId: 1e42f444-303b-5052-acfb-af8bf27bf7e6 Version: $LATEST
▶	2022-10-24T01:28:48.539+03:00	END RequestId: 1e42f444-303b-5052-acfb-af8bf27bf7e6
▶	2022-10-24T01:28:48.539+03:00	REPORT RequestId: 1e42f444-303b-5052-acfb-af8bf27bf7e6 Duration: 1790.51 ms Billed Duration: 1791 ms Memory Size: 1024 MB Max…
▶	2022-10-24T01:28:57.802+03:00	START RequestId: 16f9f2b0-2f23-553a-b3c5-a97fed0afbd8 Version: $LATEST
▶	2022-10-24T01:29:57.866+03:00	END RequestId: 16f9f2b0-2f23-553a-b3c5-a97fed0afbd8
▶	2022-10-24T01:29:57.866+03:00	REPORT RequestId: 16f9f2b0-2f23-553a-b3c5-a97fed0afbd8 Duration: 60062.99 ms Billed Duration: 60000 ms Memory Size: 1024 MB M…
▶	2022-10-24T01:29:57.866+03:00	2022-10-23T22:29:57.866Z 16f9f2b0-2f23-553a-b3c5-a97fed0afbd8 Task timed out after 60.06 seconds

Рисунок 4.2 — Запис роботи програми для збереження даних у DynamoDB

Також, щоб побачити, чи працює програма коректно, треба використати сервіс DynamoDB для переглядання вмісту таблиці. Результат виконання подано на рисунку 4.3.

timestamp	carNumber	json	lapNumber
1656758467000	1510	{"id": null, "...	1510070222102544000
1656762008900	1506	{"id": null, "...	1506070222114000009
1656759417300	1496	{"id": null, "...	1496070222105450003
1656764296700	1486	{"id": null, "...	1486070222121745044
1656764535200	1505	{"id": null, "...	1505070222122147048
1656762169200	1514	{"id": null, "...	1514070222114241012
1656764419100	1496	{"id": null, "...	1496070222120507034
1656759815300	1511	{"id": null, "...	1511070222105443001
1656764807300	1500	{"id": null, "...	1500070222122559053

Рисунок 4.3 — Вигляд таблиці в DynamoDB

Також сервіс DynamoDB дає можливість завантажити результати у вигляді CSV-формату. Цей формат більш зручний для роботи, тому що файл можна відкрити за допомогою програми Microsoft Excel, результат якого подано на рисунку 4.4.

	A	B	C	D
1	timestamp	carNumber	json	lapNumber
2	1656763614600	1505	{"id": null, "speed": 140.64088, "...	1505070222120602031
3	1656766958200	1511	{"id": null, "speed": 0.087044000...	1511070222130122000
4	1656758314500	1492	{"id": null, "speed": 0.100008, "rp...	1492070222102855001
5	1656759992600	1496	{"id": null, "speed": 0.0370400000.	1496070222105450003
6	1656766913800	1511	{"id": null, "speed": 0.125936000...	1511070222130122000
7	1656759491400	1512	{"id": null, "speed": 0.120380000...	1512070222105445004
8	1656758467000	1510	{"id": null, "speed": 0.1389, "rpm...	1510070222102544000
9	1656762008900	1506	{"id": null, "speed": 170.352516,	1506070222114000009
10	1656759417300	1496	{"id": null, "speed": 0.161124, "rp...	1496070222105450003
11	1656764296700	1486	{"id": null, "speed": 184.4406800...	1486070222121745044
12	1656764535200	1505	{"id": null, "speed": 143.374432,	1505070222122147048
13	1656762169200	1514	{"id": null, "speed": 166.576288,	1514070222114241012
14	1656764419100	1496	{"id": null, "speed": 0.016668, "rp...	1496070222120507034
15	1656759815300	1511	{"id": null, "speed": 0.068524, "rp...	1511070222105443001
16	1656764807300	1500	{"id": null, "speed": 202.069868,	1500070222122559053
17	1656761867300	1508	{"id": null, "speed": 40.703256, "...	1508070222113730006
18	1656758249200	1487	{"id": null, "speed": 0.046300000...	1487070222101727001
19	1656759369600	1512	{"id": null, "speed": 0.0611160000..	1512070222105445004
20	1656768226300	1512	{"id": null, "speed": 0.198164, "rp...	1512070222131736000

Рисунок 4.4 — Результат роботи у програмі Microsoft Excel

Основним завданням даної роботи є вимір ефективності зберігання даних. У цьому випадку потрібно визначити обсяг даних, які були збереженні у сховищах. При виконанні програми для збереження даних у файлове сховище було збережено майже 100 Мб. На рисунку 4.5 можна побачити інформацію про обсяг даних, збережену після надсилання об’єктів. 1 Тб даних коштує 35 доларів. Відповідно 100 Мб даних буде коштувати 0.0026 долара.

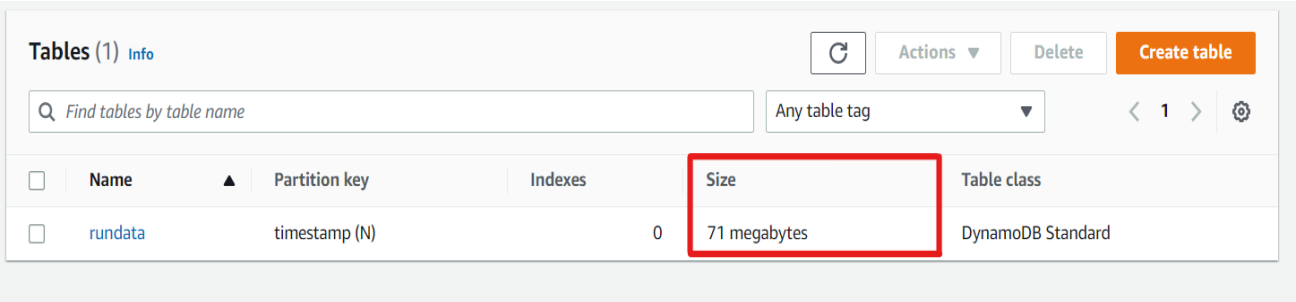


Рисунок 4.5 — Сервіс DynamoDB, який показує обсяг даних таблиці

Можна зробити висновок, що цей спосіб зберігання є ефективним, але є багато обмежень щодо аналізу, зчитування і оновлення цих даних. Використання сервісу DynamoDB вимагає додаткових бібліотек.

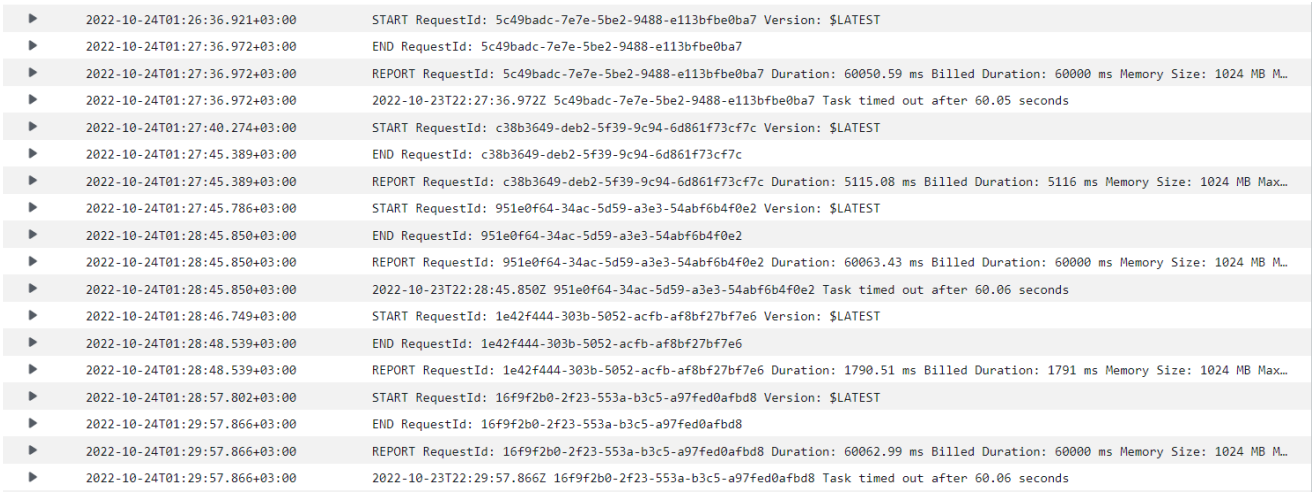
Також є варіант використання інших нереляційних баз даних, а саме MongoDB. Але цей варіант є набагато дорожчим, а також потрібно платити за саму підтримку сервісу, бо AWS поки не надає інструментів для роботи з MongoDB.

4.2. Тестування системи, яка зберігає дані в MySQL

При тестуванні роботи системи для зберігання даних в реляційну базу даних було запущено код для експорту даних, який в середньому надіслав 500 тисяч об’єктів в систему. Код скрипта подано в розділі 3 на рисунку 3.6.

Після запуску програми AWS Lambda записує роботу програми безпосередньо у сервісі AWS Cloudwatch. За допомогою цього сервісу можна

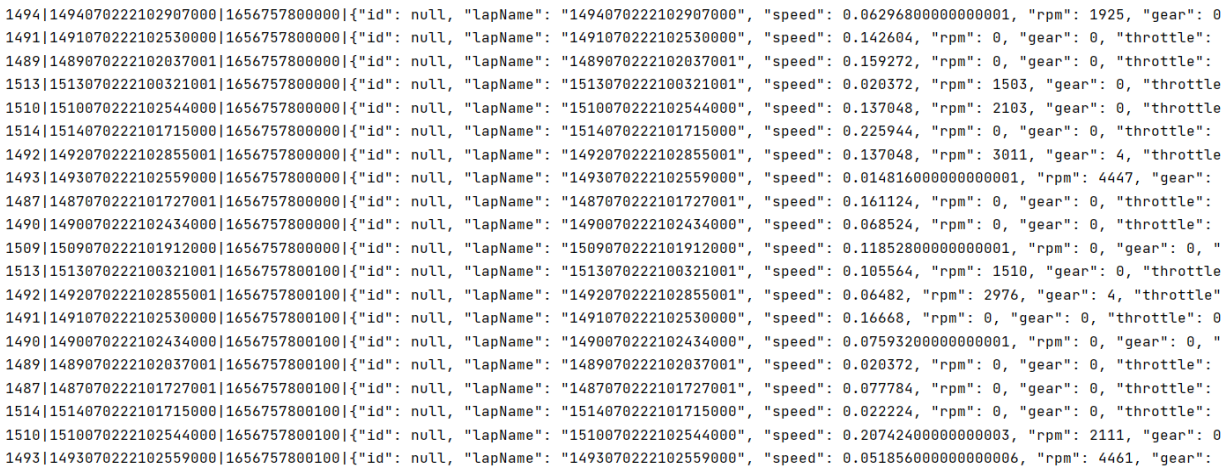
побачити, що програма завершила свою обробку успішно. На рисунку 4.6 продемонстровано роботу виконання запиту для отримання записів роботи програми збереження даних у MySQL.



2022-10-24T01:26:36.921+03:00	START RequestId: 5c49badc-7e7e-5be2-9488-e113bfbe0ba7 Version: $LATEST
2022-10-24T01:27:36.972+03:00	END RequestId: 5c49badc-7e7e-5be2-9488-e113bfbe0ba7
2022-10-24T01:27:36.972+03:00	REPORT RequestId: 5c49badc-7e7e-5be2-9488-e113bfbe0ba7 Duration: 60050.59 ms Billed Duration: 60000 ms Memory Size: 1024 MB M…
2022-10-24T01:27:36.972+03:00	2022-10-23T22:27:36.972Z 5c49badc-7e7e-5be2-9488-e113bfbe0ba7 Task timed out after 60.05 seconds
2022-10-24T01:27:40.274+03:00	START RequestId: c38b3649-deb2-5f39-9c94-6d861f73cf7c Version: $LATEST
2022-10-24T01:27:45.389+03:00	END RequestId: c38b3649-deb2-5f39-9c94-6d861f73cf7c
2022-10-24T01:27:45.389+03:00	REPORT RequestId: c38b3649-deb2-5f39-9c94-6d861f73cf7c Duration: 5115.08 ms Billed Duration: 5116 ms Memory Size: 1024 MB Max…
2022-10-24T01:27:45.786+03:00	START RequestId: 951e0f64-34ac-5d59-a3e3-54abf6b4f0e2 Version: $LATEST
2022-10-24T01:28:45.850+03:00	END RequestId: 951e0f64-34ac-5d59-a3e3-54abf6b4f0e2
2022-10-24T01:28:45.850+03:00	REPORT RequestId: 951e0f64-34ac-5d59-a3e3-54abf6b4f0e2 Duration: 60063.43 ms Billed Duration: 60000 ms Memory Size: 1024 MB M…
2022-10-24T01:28:45.850+03:00	2022-10-23T22:28:45.850Z 951e0f64-34ac-5d59-a3e3-54abf6b4f0e2 Task timed out after 60.06 seconds
2022-10-24T01:28:46.749+03:00	START RequestId: 1e42f444-303b-5052-acfb-af8bf27bf7e6 Version: $LATEST
2022-10-24T01:28:48.539+03:00	END RequestId: 1e42f444-303b-5052-acfb-af8bf27bf7e6
2022-10-24T01:28:48.539+03:00	REPORT RequestId: 1e42f444-303b-5052-acfb-af8bf27bf7e6 Duration: 1790.51 ms Billed Duration: 1791 ms Memory Size: 1024 MB Max…
2022-10-24T01:28:57.802+03:00	START RequestId: 16f9f2b0-2f23-553a-b3c5-a97fed0afbd8 Version: $LATEST
2022-10-24T01:29:57.866+03:00	END RequestId: 16f9f2b0-2f23-553a-b3c5-a97fed0afbd8
2022-10-24T01:29:57.866+03:00	REPORT RequestId: 16f9f2b0-2f23-553a-b3c5-a97fed0afbd8 Duration: 60062.99 ms Billed Duration: 60000 ms Memory Size: 1024 MB M…
2022-10-24T01:29:57.866+03:00	2022-10-23T22:29:57.866Z 16f9f2b0-2f23-553a-b3c5-a97fed0afbd8 Task timed out after 60.06 seconds

Рисунок 4.6 — Запис роботи програми для збереження даних у MySQL

Також, щоб побачити, чи працює програма коректно, треба використати сервіси для перегляду баз даних. У даному випадку використовуються консольні рішення, результат виконання яких можна побачити на рисунку 4.7.



```
1494|1494070222102907000|1656757800000|{"id": null, "lapName": "1494070222102907000", "speed": 0.06296800000000001, "rpm": 1925, "gear": 0
1491|1491070222102530000|1656757800000|{"id": null, "lapName": "1491070222102530000", "speed": 0.142604, "rpm": 0, "gear": 0, "throttle":
1489|1489070222102037001|1656757800000|{"id": null, "lapName": "1489070222102037001", "speed": 0.159272, "rpm": 0, "gear": 0, "throttle":
1513|1513070222100321001|1656757800000|{"id": null, "lapName": "1513070222100321001", "speed": 0.020372, "rpm": 1503, "gear": 0, "throttle
1510|1510070222102544000|1656757800000|{"id": null, "lapName": "1510070222102544000", "speed": 0.137048, "rpm": 2103, "gear": 0, "throttle
1514|1514070222101715000|1656757800000|{"id": null, "lapName": "1514070222101715000", "speed": 0.225944, "rpm": 0, "gear": 0, "throttle":
1492|1492070222102855001|1656757800000|{"id": null, "lapName": "1492070222102855001", "speed": 0.137048, "rpm": 3011, "gear": 4, "throttle
1493|1493070222102559000|1656757800000|{"id": null, "lapName": "1493070222102559000", "speed": 0.014816000000000001, "rpm": 4447, "gear":
1487|1487070222101727001|1656757800000|{"id": null, "lapName": "1487070222101727001", "speed": 0.161124, "rpm": 0, "gear": 0, "throttle":
1490|1490070222102434000|1656757800000|{"id": null, "lapName": "1490070222102434000", "speed": 0.068524, "rpm": 0, "gear": 0, "throttle":
1509|1509070222101912000|1656757800000|{"id": null, "lapName": "1509070222101912000", "speed": 0.11852800000000001, "rpm": 0, "gear": 0, "
1513|1513070222100321001|1656757800100|{"id": null, "lapName": "1513070222100321001", "speed": 0.105564, "rpm": 1510, "gear": 0, "throttle
1492|1492070222102855001|1656757800100|{"id": null, "lapName": "1492070222102855001", "speed": 0.06482, "rpm": 2976, "gear": 4, "throttle"
1491|1491070222102530000|1656757800100|{"id": null, "lapName": "1491070222102530000", "speed": 0.16668, "rpm": 0, "gear": 0, "throttle": 0
1490|1490070222102434000|1656757800100|{"id": null, "lapName": "1490070222102434000", "speed": 0.07593200000000001, "rpm": 0, "gear": 0, "
1489|1489070222102037001|1656757800100|{"id": null, "lapName": "1489070222102037001", "speed": 0.020372, "rpm": 0, "gear": 0, "throttle":
1487|1487070222101727001|1656757800100|{"id": null, "lapName": "1487070222101727001", "speed": 0.077784, "rpm": 0, "gear": 0, "throttle":
1514|1514070222101715000|1656757800100|{"id": null, "lapName": "1514070222101715000", "speed": 0.022224, "rpm": 0, "gear": 0, "throttle":
1510|1510070222102544000|1656757800100|{"id": null, "lapName": "1510070222102544000", "speed": 0.20742400000000003, "rpm": 2111, "gear": 0
1493|1493070222102559000|1656757800100|{"id": null, "lapName": "1493070222102559000", "speed": 0.051856000000000006, "rpm": 4461, "gear":
```

Рисунок 4.7 — Вигляд таблиці в MySQL

Основним завданням даної роботи є вимір ефективності зберігання даних. У даному випадку потрібно визначити обсяг даних, які були збереженні у сховищах.

При виконанні програми для збереження даних у файлове сховище було збережено майже 800 Мб. 1 Тб даних коштує 270 доларів. Відповідно 800 Мб даних буде коштувати 0.216 доларів.

Можна зробити висновок, що цей спосіб зберігання не є ефективним порівняно з попереднім типом сховища. Але перевагою реляційних баз даних є те, що вони дуже швидкі, існує багато рішень для роботи з ними, а також можна будувати зв’язки з іншими таблицями.

4.3. Тестування системи, яка зберігає дані в S3 Bucket

Щоб протестувати систему для зберігання даних у файловому сховищі даних, використано скрипт для експорту даних, поданий у розділі 3 на рисунку 3.10. В середньому код для експорту надіслав 500 тисяч об’єктів даних.

Після запуску програми AWS Lambda записує роботу програми (рисунок 4.8) безпосередньо у сервісі AWS Cloudwatch.

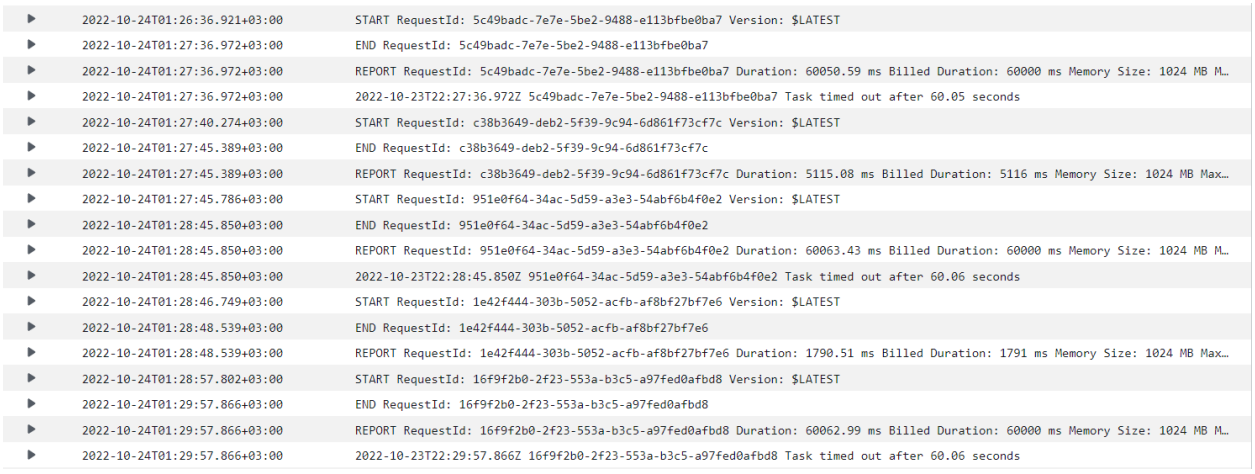


Рисунок 4.8 — Запис роботи програми для збереження даних у MySQL

За допомогою цього сервісу можна побачити, що програма завершила свою обробку успішно.

Також, щоб побачити чи працює програма коректно, треба використати S3 Bucket. На рисунку 4.9 продано результат виконання програми, а саме збережені файли у форматі Apache Parquet.

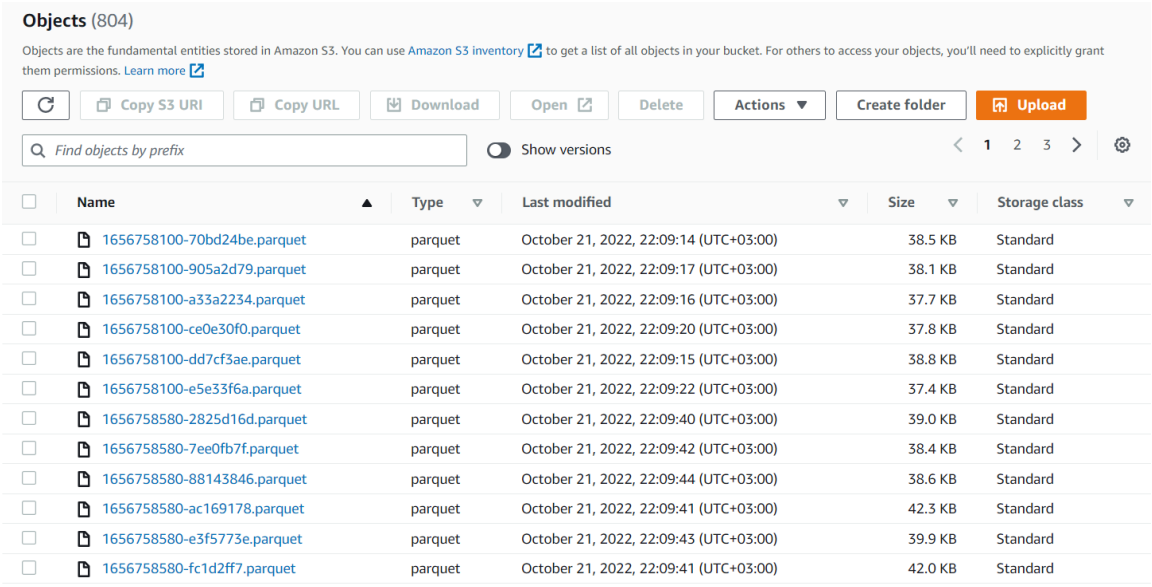


Рисунок 4.9 — Вигляд даних у S3 Bucket

Оскільки S3 Bucket — це один з типів файлового сховища і в цьому сховищі зберігаються саме файли і операції для отримання даних, як мова запитів SQL, тут не працює. Тому, щоб переглянути вміст файлу, потрібно завантажити його в онлайн-переглядач для формату Apache Parquet. На рисунку 4.10 продемонстровано дані, збережені у файлі.

Рисунок 4.10 — Вигляд вмісту файлів Parquet-формату

Основним завданням даної роботи є визначення ефективності зберігання даних. У даному випадку потрібно визначити обсяг даних, які були збережені у сховищах. При виконанні програми для збереження даних у файлове сховище було збережено майже 90 Мб. 1 Тб даних коштує 23 долари. Відповідно 85 Мб даних буде коштувати 0.00184 долара. На рисунку 4.11 подано результат виконання програми, а саме розмір сховища після обробки.

Можна зробити висновок, що цей спосіб зберігання є найефективнішим порівняно з попередніми випадками. Але також є певні недоліки, а саме: складність реалізації, складність читання файлів зі сховища і їхні модифікації, більше накладних витрат тощо. Тому варто використовувати це сховище разом з іншими службами, а саме: Amazon Athena і Amazon Glue. Тоді буде отримано озеро даних, де централізованим сховищем буде виступати бакет із вмістом файлів

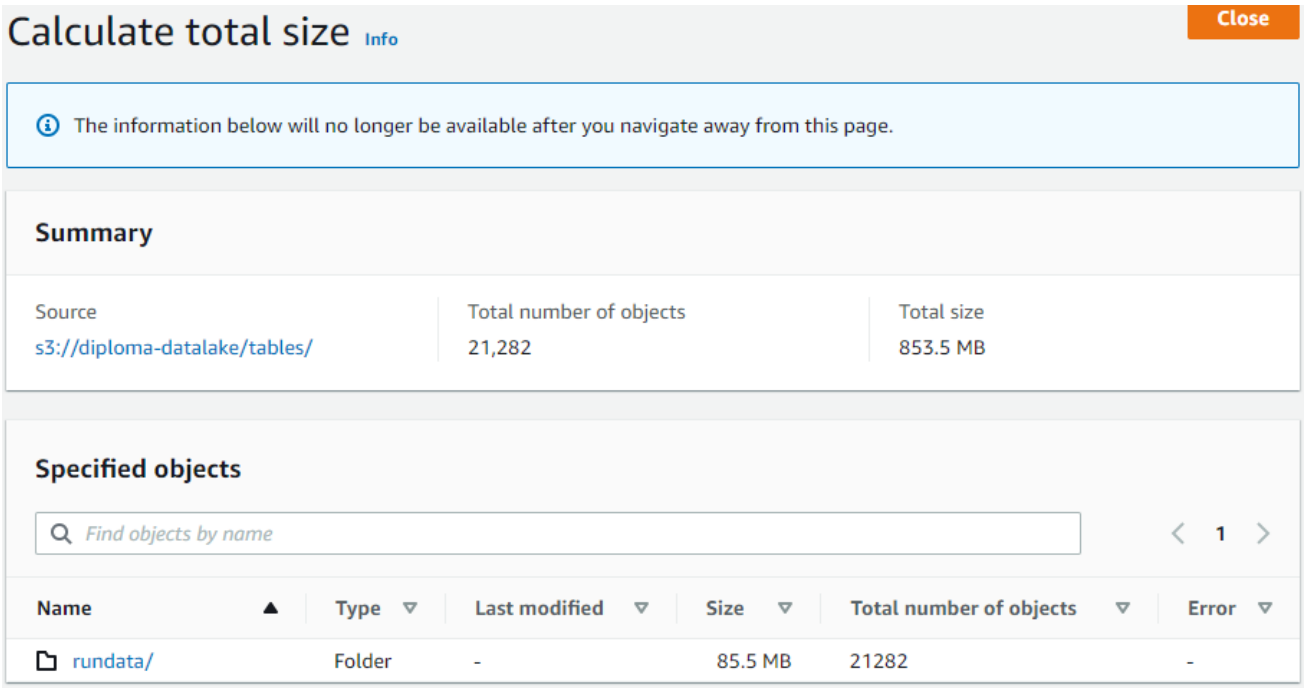


Рисунок 4.11 — Розмір сховища після виконання програми

Якщо сховище даних буде містити велику кількість файлів, навіть з форматом Parquet, то зчитування буде повільним, тому варто використовувати метод розбиття (partition), щоб швидше зчитувати дані. Розбиття файлів на окремі

підпапки, уникнення великих файлів і стиснення малих файлів є важливим. Оптимальне розміщення даних на диску залежить від шаблонів запитів.

Висновки до розділу 4

Метою даного розділу було зібрати дані про те, як системи виконують свою роботу і виміряти обсяг даних, збережених у різних сховищах.

Найменш ефективними в плані зберігання файлів виявилися реляційні бази даних. У дослідженні використано службу RDS MySQL. Але не варто відмовлятися від цього виду сховища. Цей вид сховища варто використовувати тоді, коли потрібно забезпечити швидке читання або запис даних в базу даних, якщо потрібно побудувати систему з кількома сутностями і якщо потрібна транзакційність даних.

Далі за ефективністю зберігання йде DynamoDB, який варто використовувати у випадках, коли потрібне швидке горизонтальне масштабування, і якщо схема таблиці часто змінюється.

Найефективнішим методом виявився метод зберігання даних у файлових сховищах завдяки своїй вартості зберігання даних. Але для того, щоб забезпечити цю ефективність, потрібно зберігати дані у файлах формату Apache Parquet. Також, щоб збільшувати ефективність зберігання даних, треба використати концепцію озера даних, щоб розробники могли зчитувати дані за допомогою SQL-запитів. Також варто розбивати дані на піддиректорії, щоб збільшувати ефективність читання файлів. Але цей метод варто використовувати у випадках, якщо треба архівувати дані на певний період і потім використовувати в інших дослідженнях. Цей метод підійде для роботи з машинним навчанням і штучним інтелектом.

5. РОЗРОБКА СТАРТАП-ПРОЕКТУ

У розділі описано проведений аналіз відповідного стартап-проекту, що має мету визначити принципові можливості для його можливого впровадження на ринок. Нижче описано кроки та дії, виконані для досягнення поставленої мети.

5.1. Опис ідеї проекту

В основному розробка і впровадження кожного стартапу має певну ціль. В планах не було створення стартап-проекту на основі дослідження, проте дана дисертація наглядно показує, коли варто використовувати один чи інший тип сховищ для збереження даних великого обсягу. Ця робота зможе допомогти користувачам, архітекторам застосунків чи бізнесу економити великі кошти на збереженні даних або допоможе скорегувати план дій щодо ефективної обробки і подальшого використання даних.

Багато з організацій мають велику кількість даних, які часто не використовуються. Наприклад, національні архіви, які містять історичні дані або бізнес організації, які кожного дня генерують значну кількість даних. Ці дані потрібні для використання в майбутньому, тому видалення тільки нашкодить для майбутніх досліджень чи аналізів.

В цій дисертації реалізовано декілька систем для зберігання даних великого обсягу. Також було проаналізовано ефективність зберігання кожного з сховищ. У даному розділі описано стартап-проект підходу до зберігання даних у файлових сховищах даних, а саме S3 Bucket. В таблиці 5.1 оглянуто зміст ідей, основні переваги та чим відрізняється цей метод від інших.

Даний проект має як переваги, так і значні недоліки. Порівняння буде з іншими сховищами даних. В таблиці 5.2 буде представлене порівняння цих

систем. За допомогою цього порівняння можна буде визначити, який тип сховища буде більш підходити для того чи іншого випадку.

Таблиця 5.1

Опис ідеї проекту

Зміст ідеї	**Напрямки застосування**	**Вигоди для клієнтів**
Система, яка зберігає дані у компактному форматі у файлових сховищах даних	1. Збереження даних, які нечасто використовуються і найчастіше використовуються для аналізу в майбутньому	Централізований репозиторій даних (Data Lake), який дозволяє ефективно зберігати і маніпулювати даними
	2. Архівація даних	Затрати на використання інфраструктури незначні.
	3. Допомога у збереженні даних для існуючих систем чи їхня часткова міграція	Зручні можливості для аналізу збережених даних

Таблиця 5.2

Визначення сильних, слабих та нейтральних характеристик проекту

		Потенційні концепції		
		Моя система	**Система 1**	**Система 2**
1	Слабка сторона	Дуже складна для побудови і імплементації система	Дуже дорога вартість використання	Незручне API і відсутність хороших інструментів для роботи з ним
2	Сильна сторона	Низькі витрати при великій кількості даних	Багато інструментів для роботи з базою	Можливість використання не структурованих даних

Таблиця 5.2 (продовження)

3	Нейтральна сторона	Можливість аналізу даних	Наявність транзакційності	Низька вартість використання

Як можна побачити з таблиці 5.2 основними перевагами даної системи є низькі витрати на інфраструктуру при великих кількостях даних. Основним недоліком є складність розробки та впровадження цього продукту, хоча ці витрати на людино-ресурси будуть виправдані через певний проміжок часу.

5.2. Технологічний аудит ідеї проекту

Обрано такі технології для ідеї проекту: найпопулярніша мова програмування Python, бібліотека для роботи з даними Pandas, бібліотека для створення компактних файлів Apache Arrow, AWS як основний хмарний провайдер, хмарне файлове сховище для зберігання даних великого обсягу S3 Bucket, AWS SAM утиліта для створення застосунків на основі шаблонів AWS Cloudformation.

Таблиця 5.3

Технологічна здійсненність ідеї проекту

№	**Технології її реалізації**	**Наявність технології**	**Доступність технології**
1	Мова програмування Python	Наявна	Доступна безкоштовна
2	Бібліотека Pandas	Наявна	Доступна безкоштовна
3	Бібліотека Apache Arrow	Наявна	Доступна безкоштовна
4	AWS	Наявна	Небезкоштовна
5	S3 Bucket,	Наявна	Небезкоштовна
6	AWS SAM	Наявна	Доступна безкоштовна

5.3. Аналіз ринкових можливостей запуску стартап-проекту

Виявлення ринкових можливостей, які можна використати, та ринкових загроз, які можуть перешкодити реалізації проекту, під час ринкової реалізації проекту дозволяє визначити та окреслити напрямок проекту з урахуванням стану ринку трафіку, потреб потенційних клієнтів та пропозиції конкуруючих проектів. У таблиці 5.4 наведено попередній опис потенційного ринку для стартап-проекту, що розробляється.

Таблиця 5.4

Попередня характеристика потенційного ринку стартапу-проекту

№	Показники стану ринку	Характеристика
1	Кількість головних гравців, од	>10
2	Динаміка ринку	Цей ринок росте швидко через те, що кількість даних збільшується
3	Наявність обмежень для входу	Велика складність для реалізації даних
4	Специфічні вимоги до стандартизації та сертифікації	GDPR

Лід — це термін, що позначає особу чи організацію, яка наразі не є клієнтом компанії, але є частиною цільової ринкової групи компанії.

Теоретично при якісній роботі всі потенційні клієнти можуть стати реальністю при дотриманні певних умов (більш інтенсивна або агресивна таргетована реклама, вища якість продукту, нижчі ціни, кваліфікована робота менеджерів з продажу, прозорість виконання і реалізації).

Надалі було складено приблизний список потенційних груп споживачів, їхні характеристики та потреби в продуктах для кожної групи, які можуть використовувати цей проект в майбутньому (таблиця 5.5).

Таблиця 5.5

Характеристика потенційних клієнтів стартап-проекту

№	Потреба що формує ринок	Цільова аудиторія	Відмінності у поведінці	Вимоги споживачів до товару
1	Зберігання даних у компактному варіанті	Бізнес, організації, які генерують велику кількість даних	Вигляд та структурованість даних	Можливість виведення статистики і аналізу
2	Невелика вартість даних	Архіви		Захищений доступ до даних

Після визначення груп потенційних споживачів було проведено аналіз ринкового середовища: розроблено таблиці факторів, що сприяють і перешкоджають реалізації проекту на ринку (таблиці 5.6 і 5.7).

Загрози інформаційній безпеці — це сукупність умов і факторів, що загрожують життєво важливим інтересам особи, суспільства та країни в інформаційному полі. Фактори загрози за типом класифікуються на політичні, економічні та організаційно-технічні

Таблиця 5.6

Фактори загроз

№	Фактор	Зміст загрози	Можлива реакція компанії
1	Невелика кількість даних при генерації	При зберіганні невеликої кількості даних, затрати на розробку будуть перевищувати чим витрати на інфраструктуру	Перехід на інший тип сховища даних

Таблиця 5.6 (продовження)

2	Занадто складна структура даних і відмовостійкість	При зберіганні даних постійно треба переглядати структуру сховищ і їхню функціональність	Створення окремих обробників для відмовостійкості. Написання міграцій для відслідковування змін в структурі
3	Обмеженість функцій	Багато з функцій для аналізу відсутні або частково присутні з обмеженнями	Додавання цих функцій і їхнє тестування

Таблиця 5.7

Фактори можливостей

№	**Фактор**	**Зміст можливості**	**Можлива реакція компанії**
1	Недоліки в попередніх і аналогічних системах	Наявні аналоги працюють не так як треба і з меншою ефективністю	Патентування методів. Доповнення слабких позицій
2	Клієнтська база	За допомогою доброго і прорахованого плану та стратегії можна розшити базу клієнтів і почати співпрацювати з провідними надавачами послуг	Співпрацюйте з відомими службами або тими, чиї ідеї популярні ще до того, як вони вийшли на ринок. Можна проводити демонстрації продукції

Для визначення та обґрунтування фактору конкурентоспроможності, необхідно на основі аналізу конкуренції, а також із урахуванням характеристик ідеї проекту (таблиця 5.2), вимог споживачів до товару та факторів маркетингового середовища (таблиця 5.6-5.7) визначити фактор.

Таблиця 5.8

Обґрунтування факторів конкурентоспроможності

№	Фактор конкуренто-спроможності	Обґрунтування
1	Збір статистики і різних даних для аналізу	Важливим моментом при роботі з даними є збір аналітики.
2	Невелика вартість зберігання	Під час обробки даних велику роль грає вартість зберігання даних. Тому даний метод є конкурентоспроможним
3	Зручна інтеграція і шаблонізація	Повинен бути створений адаптер для обробки об’єктів і шаблон для створення інфраструктури для подальшої роботи з даними

Таблиця 5.9

Порівняльний аналіз сильних та слабких сторін

№	Фактор конкурентоспроможності	Бали 1-20	Рейтинг товарів-конкурентів у порівнянні з розроблюваним продуктом						
			-3	-2	-1	0	1	2	3
1	Збір статистики і різних даних для аналізу	12					+		
2	Зручна інтеграція і шаблонізація	15		+					
3	Невелика вартість зберігання	18		+					

Заключним етапом аналізу ринку можливостей реалізації проекту є підготовка SWOT-аналізу (матриці аналізу сильних і слабких сторін, загроз і можливостей) на основі обраних ринкових загроз і можливостей, а також сильних і слабких сторін (таблиця 5.10). Традиційний метод SWOT — аналізу дозволяє провести детальне дослідження зовнішнього й внутрішнього середовища.

Таблиця 5.10

SWOT-аналіз стартап проекту

Сильні сторони: Невисока вартість витрат на інфраструктуру Можливість створювати статистичні дані для подальшого аналізу	**Слабкі сторони:** Потребує масштабної рекламної компанії Дуже складна для імплементації Складне тестування
Можливості: Розширення функціоналу для аналізу Створення шаблонізаторів для уніфікації створення	**Загрози:** Покупці генерують недостатню кількість даних Невелика кількість організацій

На основі SWOT-аналізу сформульовано варіанти ринкової поведінки (перелік заходів) для виведення стартап-проекту на ринок та приблизний оптимальний ринок часу його реалізації на ринку для потенційних проектів, які можуть залучити конкурентів.

На основі SWOT-аналізу з урахуванням потенційних проектів конкурентів, які можуть бути залучені, формуються варіанти ринкової поведінки (перелік заходів) для виведення стартап-проекту на ринок та орієнтовні оптимальні терміни його реалізації на ринку. на ринок.

Список ринкових небезпек та цілком ймовірних загроз, а також ймовірних можливостей складається за основою фактору загроз та за основую факторів можливостей середовища маркетингу. Загрози ринку та можливості ринку — це наслідки впливу факторів, які являються фактично реалізованими.

Маркетингова стратегія — це сукупність основних рішень, сформульованих для досягнення загальної мети підприємства на основі оцінки ринкової ситуації, власних можливостей та інших факторів і сил маркетингового середовища, і формує маркетингову стратегію підприємства в сучасний ринок. Під впливом багатьох факторів. Реалізація цієї стратегії сприяє перетворенню продукту на товар. Перший крок у розробці ринкової стратегії передбачає визначення стратегії охоплення ринку: опис цільової групи потенційних споживачів

Виявлені альтернативи були проаналізовані з точки зору умов і наявності ресурсів (таблиця 5.13).

Таблиця 5.11

Альтернативи ринкового впровадження стартап-проекту

№	Альтернатива ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
1	Розміщення ідеї в невеликих організаціях	Низька ймовірність так як малі організації можуть генерувати невелику кількість даних	Протягом трьох місяців
2	Розміщення ідеї в некомерційних державних організаціях	Середня ймовірність, проте є великі шанси того, що фінансування може припинитись	Протягом трьох місяців
	Розміщення в комерційних середніх чи великих організаціях	Більша за середню, але доволі тяжко вийти на ринок і найти потенційних клієнтів	Протягом трьох місяців

5.4. Розробка ринкової стратегії проекту

За результатами аналізу потенційних груп споживачів (ринкових сегментів) рекомендується вибрати конкретну цільову групу та розробити відповідно стратегію охоплення ринку. Оскільки цільовою аудиторією для розробки

продукту може бути організація, зацікавлена в отриманні найбільшого прибутку від проведення конкурсу, вибір цільової маркетингової стратегії є важливим.

Для роботи в обраних сегментах ринку необхідно сформувати базову стратегію розвитку (таблиця 5.12).

Таблиця 5.12

Визначення базової стратегії розвитку

Обрана альтернатива розвитку проекту	**Стратегія охоплення ринку**	**Ключові конкурентоспроможні позиції відповідно до обраної альтернативи**	**Базова стратегія розвитку**
Орієнтація поточної моделі на ринок середній або великий бізнес	Стратегія концентрованого маркетингу	Середній і великий бізнес потребує швидкості розробки та ефективності, яку надає програмний продукт.	Стратегія спеціалізації (спирається на диференціацію)

Таблиця 5.13

Визначення базової стратегії конкурентної поведінки

Чи є проект “першопрохідцем” на ринку?	**Чи буде компанія шукати нових споживачів**	**Чи буде компанія копіювати основні характеристики конкурента**	**Стратегія конкурентної поведінки**
Ні	Ні, це буде зроблено за допомогою спеціалізованої реклами	Ні	Стратегія заняття конкурентної ніші

Вибір стратегії має ґрунтуватися на чіткій концепції розвитку проекту, а саме: формулювання — чітке та однозначне, оскільки обрана довгострокова стратегія обмежує готовність керівництва до дій та впливає на всі рішення, які воно приймає. Вибрані альтернативи були ретельно досліджені та оцінені. При цьому слід враховувати багато факторів: ризик, досвід минулих стратегій, вплив акціонерів, фактор часу тощо.

Таблиця 5.14

Визначення стратегії позиціонування

№	Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкуренто-спроможні позиції	Асоціацій, які сформують позицію проекту
1	Легкість розуміння, зручний інтерфейс, надійний, швидкий, точний та достовірний ПП для аналізу даних.	Стратегія диференціації	Позиція на основі порівняння фірми з товарами конкурентів;	Економія коштів і часу; Ефективність результату

5.5. Розробка маркетингової програми стартап-проекту

Таблиця 5.15

Визначення ключових переваг концепції потенційного товару

Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами
Збільшення ефективності зберігання даних	Вартість зберігання даних зменшується	Конкуренти часто використовують не- оптимальні шляхи

Розроблено трирівневу модель товару: уточняється ідея продукту, частини, процес його впровадження (таблиця 5.16).

1-й рівень. Під час створення дизайну продукту розглядається питання про те, які потреби та/або проблеми буде використовуватися для вирішення продукту та які його основні переваги. Це питання безпосередньо пов’язане з формуванням технічного завдання при розробці конструкторської документації на виріб.

2-й рівень. Цей рівень являє собою рішення про те, як продукт буде реалізовано в реальному житті, включаючи якість, продуктивність, дизайн, упаковку, ціну і інші якісні характеристики.

3-й рівень. Покращені товари — додаткові послуги та переваги, які пропонуються споживачеві, які створюються на основі дизайну продукту та фактичного виконання продукту (забезпечення якості, доставка, умови оплати, документація до продукту тощо).

Таблиця 5.16

Опис трьох рівнів моделі товару

№	Рівні товару	Сутність та складова
1	Товар за задумом	Платформа, що допомагає ефективно зберігати дані і створювати централізоване файлове сховище даних
2	Товар у реальному виконані	Серверний застосунок, який ефективно пакує і зберігає дані у файлове сховище даних
		Клієнтський додаток, який допомагає аналізувати кількість і якість збережених даних
3	Товар з підкріпленням	До продажу: програмний код
		Після продажу: ліцензія з можливістю подальшої підтримки

Наступним кроком є визначення найкращої системи продажів для прийняття рішень (таблиця 5.17), за якою приймаються рішення (таблиця 6.20):

— проведення збуту власними силами чи залучення сторонніх посередників (тип системи збуту);

— канал збуту;

— вид посередників;

— проведення конкурентних тендерів на закупівлі.

Таблиця 5.17

Формування системи збуту

№	**Специфіка закупівельної поведінки цільових клієнтів**	**Функції збуту, які має виконувати постачальник товару**	**Глибина каналу збуту**	**Оптимальна система збуту**
1	Особливої специфіки не існує	Постачальник повинен надати кваліфікованих розробників, які допоможуть клієнту визначити домену область і побудувати систему	Одноріневий канал збуту через прямий контакт з клієнтом	Право власності залишається у розробника продукту

Таблиця 5.18

Визначення меж встановлення ціни

№	**Рівень цін на товари-замінники**	**Рівень цін на товари-аналоги**	**Рівень доходів цільової групи споживачів**	**Верхня та нижня межі становлення ціни на товар/послугу**
1	Дані не розповсюджуються	Дані не розповсюджуються	1000$ / місяць	200−350

Таблиця 5.19

Концепція маркетингових комунікацій

№	**Специфіка поведінки цільових клієнтів**	**Канали комунікації, якими користуються клієнти**	**Ключові позиц** обрані для **позиціонування**	**Завдання рекламного повідомлення**	**Концепція рекламного звернення**
1	Заключення контракту для співпраці перед запуском розробки сервісу	Зустрічі в онлайн зустрічах	Продаж програмного продукту та надання консультацій	Показ основних характеристик продукту і переваг над іншими аналогами	Безоплатні колаборації, щоб охопити потенційних клієнтів. Акцентування уваги на перевагах продукту

Висновки до розділу 5

Цей розділ містить інформацію про розробку стартового проекту для програмного продукту, що розробляється. Також дається опис концепції проекту, проводиться аналіз конкурентів на ринку та їхніх можливостей, розглядаються сильні та слабкі сторони, наводяться риси, які відрізняють розроблений проект від інших подібних проектів.

Наступні особливості проекту описані у відповідних підпунктах:

— дано основну ідею проекту, описано функції розробленого програмного забезпечення, наведено основні конкуруючі рішення та методи та порівняно їхні характеристики, а також визначено характеристики розробленого проекту, які відрізняються від інших проектів;

— проведено технічну перевірку розробленого програмного забезпечення та визначено деталі його впровадження;

— порівняно конкуруючих рішень щодо ринкових можливостей, перспектив запуску програмних продуктів. Також сформульовано стратегія виходу на ринок, визначається цільовий сегмент споживачів і аналізується їхній попит на продукт;

— проаналізовано канали збуту, описано основні принципи просування та методи продажу продукції, визначено шляхи та кроки, які компанії можуть зробити для розширення охоплення аудиторії та швидшого виходу на ринок.

Тому після аналізу виходить стартовий проект, який відображає шлях програмного продукту на ринок, його випуск. Також були отриманні знання зі створення стартап-проектів, аналізу ринку конкуруючих рішень, визначення маркетингових стратегій для виходу на ринок та аналізу характеристик проекту, щоб відрізнити продукт від інших.

ВИСНОВКИ

Головним завданням магістерської дисертації було розв’язання проблеми підвищення ефективності збереження даних великого обсягу. Для розв’язання цього завдання проведено докладний теоретичний аналіз та огляд різних типів сховищ даних.

Розроблено і реалізовано три різні варіанти зберігання даних в різних сховищах даних:

— система для зберігання даних у RDS;

— система для зберігання даних у DynamoDB;

— система для зберігання даних у S3 Bucket;

— інструменти для імпорту та експорту даних.

Кожна система має схожі складові, крім місця зберігання даних, тому можна не враховувати швидкість виконання самої програми, оскільки ефективність зберігання вимірюється вартістю пакету даних, які збережені за певний проміжок часу у сховищі. Для дослідження було вибрано Інтернет-ресурс, який зберігав дані про гоночні заїзди машин. Кожен об’єкт даних містить дані про стан машини в гоночному заїзді. Ці дані підходять для дослідження через те, що кожен об’єкт містить багато даних і структура часто змінюється.

Після створення систем проведено аналіз вартості даних і скільки пакетів даних займають місце в кожному сховищі. Якщо взяти до уваги всі результати, то можна викласти основні висновки:

— реляційні бази даних не є ефективними для зберігання даних великого обсягу. Проте цей тип сховища має переваги, а саме: швидкість виконання операцій, можливість створювати з’єднання з іншими таблицями тощо;

— нереляційні бази даних є більш ефективними для зберігання даних великого обсягу. Цей метод варто використовувати, якщо потрібно швидко

побудувати систему. Нереляційні бази даних є досить швидким і можуть швидко масштабуватися на відміну від реляційних. Проте вартість зберігання даних також є високою. Інші нереляційні бази даних ще більш дорогі, ніж DynamoDB;

— файлові сховища є найефективнішими для зберігання даних великого обсягу. Завдяки зберіганню файлів з даними у форматі Apache Parquet, вдалося зменшити обсяг даних, які зберігаються. Цей метод буде корисним, якщо дані будуть використовуватися не часто. Також варто використовувати цей метод разом із концептом озера даних Data Lake. Також цей метод вимагає більших затрат для реалізації і підтримки.

Отже, виходячи з усього сказаного вище, можна прийти до висновку, що зберігання даних великого обсягу є найефективнішим у файлових сховищах. Проте цей метод є варто застосовувати у випадках, коли дані використовуються не так часто, як в реляційних чи нереляційних базах даних. Також доцільно використовувати концепцію озера даних для побудови ще більш ефективного сховища. Інші сховища є менш ефективними, але також треба розглядати в контексті розв’язання задач, тому треба враховувати, як часто дані будуть використовуватися і як довго проект буде підтримуватися.

За результатами виконання магістерської роботи проаналізовано і розроблено план стартап-проекту, який містить: опис ідеї і технологічний аудит проекту, аналіз ринкових можливостей запуску стартап-проекту, розробку ринкової стратегії і маркетингової програми.

Результати виконання магістерської дисертації доповідалися на VI Міжнародній науково-практичній конференції “Інтеграція світових наукових процесів як основа суспільного прогресу”, 25-26 листопада 2022 року, м. Київ [43].

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Технології Big Data: ключові характеристики, особливості та переваги | AI CONFERENCE KYIV 2020. AI Conference Kyiv 2021. URL: https://aiconference.com.ua/uk/news/tehnologii-big-data-klyuchevie-harakteristiki-osobennosti-i-preimushchestva-97883 (дата звернення: 11.12.2022).
2. Алєксєєв Н. О. Система аналізу телекомунікаційних данних в Big Data середовищі : Магістерська дисертація : 122. Київ, 2018. 97 с.
3. Volume, velocity, and variety: Understanding the three V's of big data. ZDNET. URL: https://www.zdnet.com/article/volume-velocity-and-variety-understanding-the-three-vs-of-big-data/ (дата звернення: 11.12.2022).
4. Gutta S. The 5 v's of big data. Medium. URL: https://medium.com/analytics-vidhya/the-5-vs-of-big-data-2758bfcc51d (дата звернення: 11.12.2022).
5. Big data and business analytics / ред. J. Liebowitz. Auerbach Publications, 2016. URL: https://doi.org/10.1201/b14700 (дата звернення: 11.12.2022).
6. Jianqing Fan, Fang Han, Han Liu. // Challenges of big data analysis National Science Review. 2014. Pp. 293-314.
7. Apache Hadoop. Apache Hadoop. URL: https://hadoop.apache.org/ (дата звернення: 11.12.2022).
8. Apache Spark™ — Unified Engine for large-scale data analytics. Apache Spark™ — Unified Engine for large-scale data analytics. URL: https://spark.apache.org/ (дата звернення: 11.12.2022).
9. Mehta R. Big Data Analytics with Java: Data analysis, visualization & machine learning techniques. Packt Publishing, 2017. 418 с.
10. An Analysis of the Potential Applications of Big Data Analytics (BDA) in Supply Chain Management: Emerging Market Perspective | Semantic Scholar. Semantic

Scholar | AI-Powered Research Tool. URL: https://www.semanticscholar.org/paper/An-Analysis-of-the-Potential-Applications-of-Big-in-Rahman-Fahim/46f48d0ae0fccc2163a5a24bcaca1d2ad89b3ae7 (дата звернення: 11.12.2022).

11. What is data storage? | IBM. IBM — Deutschland | IBM. URL: https://www.ibm.com/topics/data-storage (дата звернення: 11.12.2022).

12. What Is an Enterprise Data Warehouse (EDW)?. Dremio. URL: https://www.dremio.com/resources/guides/intro-enterprise-data-warehouse/ (дата звернення: 11.12.2022).

13. What is a Data Warehouse?. Oracle | Cloud Applications and Cloud Platform. URL: https://www.oracle.com/database/what-is-a-data-warehouse/ (дата звернення: 11.12.2022).

14. Ohbyung Kwon, Nam Yeon Lee, Bongsik Shin. // Data quality management, data usage experience and acquisition intention of big data analytics // International Journal of Information Management. 2014. № 3. Pp. 387-394.

15. The Economist. Data, data everywhere. The Economist. URL: http://www.economist.com/node/15557443/ (дата звернення: 11.12.2022).

16. Dennis A. System Analysis Design. 2-ге вид. John Wiley & Sons Inc, 2002.

17. What is a relational database?. Oracle | Cloud Applications and Cloud Platform. URL: https://www.oracle.com/database/what-is-a-relational-database/ (дата звернення: 11.12.2022).

18. Why Do Developers Prefer NoSQL Databases?. Oracle | Cloud Applications and Cloud Platform. URL: https://www.oracle.com/database/nosql/what-is-nosql/ (дата звернення: 11.12.2022).

19. Relational vs. NoSQL data. Microsoft Learn: Build skills that open doors in your career. URL: https://learn.microsoft.com/uk-ua/dotnet/architecture/cloud-native/relational-vs-nosql-data (дата звернення: 11.12.2022).

20. What Is File Storage?. Oracle | Cloud Applications and Cloud Platform. URL: https://www.oracle.com/cloud/storage/file-storage/what-is-file-storage/ (дата звернення: 11.12.2022).

21. What is File Storage & How Does it Differ from Object Storage | DataCore. DataCore Software. URL: https://www.datacore.com/blog/file-object-storage-differences/ (дата звернення: 11.12.2022).

22. Wittig A., Wittig M. Amazon Web Services in Action. Manning Publications, 2015. 424 с.

23. Why are Cloud services so important in digital strategy?. Digital Strategy and IT Innovation. URL: https://leonardo-matsumota.com/2020/04/22/why-cloud-services-is-so-important-in-digital-strategy/ (дата звернення: 11.12.2022).

24. What is a data lake?. Amazon Web Services, Inc. URL: https://aws.amazon.com/ru/big-data/datalakes-and-analytics/what-is-a-data-lake/ (дата звернення: 11.12.2022).

25. Amazon Kinesis — Process & Analyze Streaming Data — Amazon Web Services. Amazon Web Services, Inc. URL: https://aws.amazon.com/kinesis/ (дата звернення: 11.12.2022).

26. Amazon Kinesis Data Firehose — Streaming Data Pipeline — Amazon Web Services. Amazon Web Services, Inc. URL: https://aws.amazon.com/kinesis/data-firehose/ (дата звернення: 11.12.2022).

27. Kafka vs. Kinesis: A Deep Dive Comparison | StreamSets. StreamSets. URL: https://streamsets.com/blog/kafka-vs-kinesis/ (дата звернення: 11.12.2022).

28. Abbasi A. AWS Certified Data Analytics Study Guide: Specialty Exam. Wiley & Sons, Limited, John, 2020. 408 с.

29. Serverless Computing — AWS Lambda — Amazon Web Services. Amazon Web Services, Inc. URL: https://aws.amazon.com/lambda/ (дата звернення: 11.12.2022).

30. Fully Managed Message Queuing — Amazon Simple Queue Service — Amazon Web Services. Amazon Web Services, Inc. URL: https://aws.amazon.com/sqs/ (дата звернення: 11.12.2022).

31. Fully Managed Relational Database — Amazon RDS — Amazon Web Services. Amazon Web Services, Inc. URL: https://aws.amazon.com/rds/ (дата звернення: 11.12.2022).

32. Fast NoSQL Key-Value Database — Amazon DynamoDB — Amazon Web Services. Amazon Web Services, Inc. URL: https://aws.amazon.com/dynamodb/ (дата звернення: 11.12.2022).

33. Cloud Object Storage — Amazon S3 — Amazon Web Services. Amazon Web Services, Inc. URL: https://aws.amazon.com/s3/ (дата звернення: 11.12.2022).

34. PyArrow — Apache Arrow Python bindings — Apache Arrow v10.0.1. Apache Arrow | Apache Arrow. URL: https://arrow.apache.org/docs/python/index.html (дата звернення: 11.12.2022).

35. Introduction to Data Lakes — Databricks. Databricks. URL: https://www.databricks.com/discover/data-lakes/introduction (дата звернення: 11.12.2022).

36. Serverless Data Integration — AWS Glue — Amazon Web Services. Amazon Web Services, Inc. URL: https://aws.amazon.com/glue/ (дата звернення: 11.12.2022).

37. Interactive Analytics — Amazon Athena — Amazon Web Services. Amazon Web Services, Inc. URL: https://aws.amazon.com/athena/ (дата звернення: 11.12.2022).

38. Hadoop: The Definitive Guide. 3-тє вид. O'Reilly Media, Inc, 2012. 688 с.

39. Apache Parquet. Apache Parquet. URL: https://parquet.apache.org/ (дата звернення: 11.12.2022).

40. Application and Infrastructure Monitoring — Amazon CloudWatch — Amazon Web Services. Amazon Web Services, Inc. URL: https://aws.amazon.com/cloudwatch/ (дата звернення: 11.12.2022).

41. Ignacio Rojas, Héctor Pomares, Olga Valenzuela. // A personal perspective on the origin(s) and development of "big data" [Електронний ресурс]: The phenomenon, the term, and the discipline (Scholarly Paper No. ID 2202843) Social Science Research Networkccc. URL: Режим доступу до ресурсу: http://papers.ssrn.com/sol3/papers.cfm?abstract_id=2202843/.

42. C.C. Aggarwal. // An introduction to social network data analytics // Social network data analytics, Springer, United States, -2011 pp. 1–15.

43. Власюк Б. С. Засоби підвищення ефективності зберігання даних великого обсягу / Б. С. Власюк, Л.І. Кублій // Інтеграція світових наукових процесів як основа суспільного прогресу: Матеріали VI Міжнародної науково-практичної конференції, м. Київ, 25-26 листопада 2022 року. Київ-Запоріжжя: Інститут інноваційної освіти, 2022. С. 99-102. URL: https://novaosvita.com/wp-content/uploads/2022/12/IntWorldScProc-Kyiv-Nov2022.pdf (дата звернення: 11.12.2022)

ДОДАТОК А

ЗАСОБИ ПІДВИЩЕННЯ ЕФЕКТИВНОСТІ ЗБЕРІГАННЯ ДАНИХ ВЕЛИКОГО ОБСЯГУ

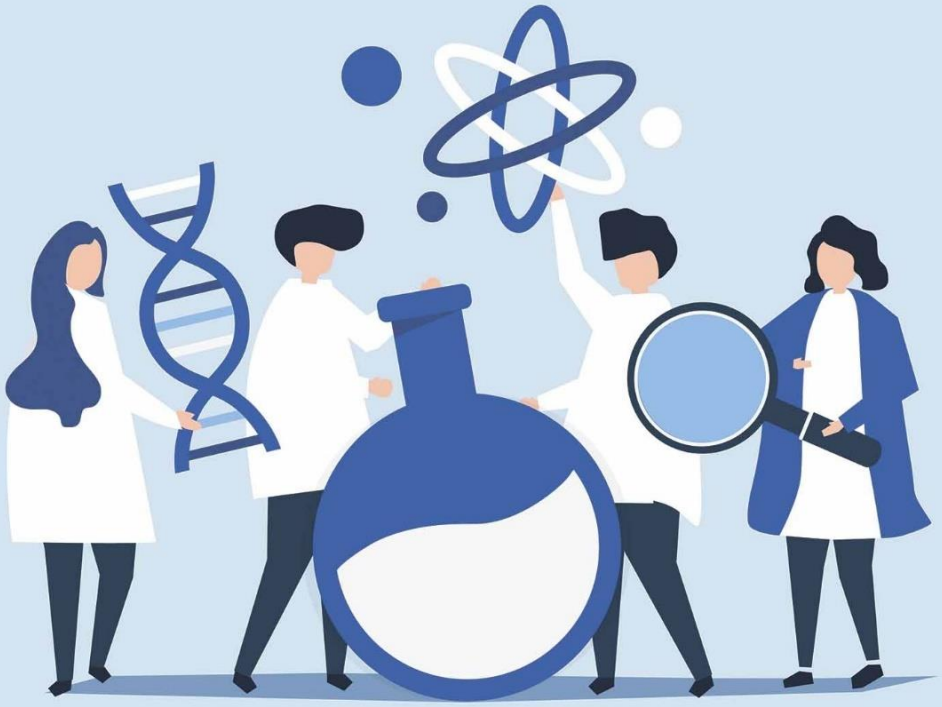
Тези на VI Міжнародній науково-практичну конференцію
“Інтеграція світових наукових процесів як основа суспільного прогресу”,
25-26 листопада 2022 року, м. Київ

УКР.НТУУ”КПІ ім. Ігоря Сікорського” _ІАТЕ_ЦТЕ_ ТР-13мп

Аркушів 7

Київ — 2022

Національна академія наук України
Науково-навчальний центр прикладної інформатики
Інститут інноваційної освіти



Інтеграція світових наукових процесів як основа суспільного прогресу

Матеріали
VI Міжнародної науково-практичної конференції
25-26 листопада 2022 р.

Інститут інноваційної освіти

Міжнародні та всеукраїнські науково-практичні конференції

www.novaosvita.com

Цирфа К.А., Спектор О.М.,
ПЕРЕХРЕСНИЙ ДОПИТ У МІЖНАРОДНОМУ
КОМЕРЦІЙНОМУ АРБІТРАЖІ: РОЛЬ ТА ЗНАЧЕННЯ 92

Розділ 7
ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ

Ahundov E., Lendiuk Taras V.,
PROJECT MANAGEMENT FOR THE DISTRIBUTION
NETWORK OF A START-UP COMPANY 96

Власюк Б.С., Кублій Л.І.,
ЗАСОБИ ПІДВИЩЕННЯ ЕФЕКТИВНОСТІ
ЗБЕРІГАННЯ ВЕЛИКИХ ОБСЯГІВ ДАНИХ 99

Ковшар Р.О., Кублій Л.І.,
КЕРУВАННЯ ПОДАННЯМ ДАНИХ З РІЗНОРІДНИХ ДЖЕРЕЛ 102

Смага О., Саченко А.О.,
УПРАВЛІННЯ МІКРОКЛІМАТОМ РОЗУМНОЇ ТЕПЛИЦІ
ДЛЯ СТВОРЕННЯ АВТОМАТИЗОВАНОЇ АГРОФЕРМИ 104

Худяков А.А., Сікора Я.Б.,
СПЕЦІАЛІЗОВАНЕ ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ
ДЛЯ ВИВЧЕННЯ ІНОЗЕМНОЇ МОВИ 107

Розділ 8
ЕЛЕКТРИЧНА ІНЖЕНЕРІЯ

Дороніна О.П.,
ОГЛЯД СУЧАСНИХ БАГАТОФУНКЦІОНАЛЬНИХ ЛІЧИЛЬНИКІВ
ЕЛЕКТРИЧНОЇ ЕНЕРГІЇ 111

Розділ 9
ВИРОБНИЦТВО І ТЕХНОЛОГІЇ

Бондарук С.Л.,
УДОСКОНАЛЕННЯ ТЕХНОЛОГІЙ БОРОШНЯНИХ
КОНДИТЕРСЬКИХ ВИРОБІВ НА ОСНОВІ ВИКОРИСТАННЯ
НАТУРАЛЬНИХ БІОЛОГІЧНО-АКТИВНИХ РЕЧОВИН 115

The anticipated model uses the evolutionary delivery model; each is connected to the data center. An additional advantage is in powering sensors by a server via informational network channel. Ensured that the client's necessities are met, the project is finished on time and within budget and that everyone else is doing their job effectively.

This was obtained by using techniques from project management to implement the above project in a systematic approach by making the best use of limited resources while achieving program goals.

Microsoft Project makes applying project-management principles to any nonprofit's endeavor easy and rewarding. Microsoft Project can hold the smallest projects that need resource distribution, scheduling, and development tracking, but it's also capable of managing extremely elaborate and large-scale projects.

The approach can be successfully used for planning and executing projects implementation of different kind of distribution networks and channels.

References

1. Bozarth, C. C., & Handfield, R. B. Introduction to Operations and Supply Chain Management, Upper Sadle River, New Jersey: Pearson Education, Inc., 2006.
2. Kotler, Ph.; Keller, K. L. Marketing Management, 13th Ed., Prentice-Hall, 2009.
3. Wright, R., Business to business marketing a step-by-step guide. Pearson Education, 2011.
4. Parker, D. and Mobey, A. Action Research to Explore Perceptions of Risk in Project Management. International Journal of Productivity and Performance Management, 53(1): 18–32, 2004.
5. Cleland D. I., Ireland L. R. Project Management Portable Hanbook. McGraw-Hill Professional, 2010.

Власюк Б.С.,
здобувач вищої освіти ступеня магістра
Національного технічного університету України
"Київський політехнічний інститут імені Ігоря Сікорського"
науковий керівник: ***Кублій Л.І.,***
кандидат технічних наук, доцент цифрових технологій в енергетиці
Навчально-наукового інституту атомної та теплової енергетики
Національного технічного університету України
"Київський політехнічний інститут імені Ігоря Сікорського"

ЗАСОБИ ПІДВИЩЕННЯ ЕФЕКТИВНОСТІ ЗБЕРІГАННЯ ВЕЛИКИХ ОБСЯГІВ ДАНИХ

В епоху масового зростання кількості користувачів і пристроїв Інтернету обсяги даних зростають експоненціально. Цей процес безперервний, і він також потребує зберігання великої кількості даних за попередні роки. У більшості випадків ці дані зберігаються в хмарі. Але

використання хмарних рішень коштує недешево. Тому постає питання, як зберігати дані в компактному форматі, щоб заощадити гроші на інфраструктурі та дозволити клієнтам ефективно отримувати ці дані.

Компанії та компанії, які постійно збирають, обробляють та аналізують різні дані для задоволення своїх потреб, все більше цікавляться цим питанням, щоб зрозуміти потреби споживачів шляхом аналізу зібраної інформації. Потреба в цьому призводить до швидкого зростання даних і труднощів їх обробки. Отримане рішення має бути здатним обробляти великі обсяги даних, бути в системі, яка може виконувати паралельні обчислення (до 1000 або більше паралельних обчислень), і мати можливість аналізувати запити споживачів.

Після 2001 року експерти зрозуміли, що найефективнішими є програмні послуги, які надає посередник із готовими рішеннями та доступними цінами. Оновлення потреб клієнтів і нова інформація надходили щороку. З появою нових технологій все більше людей підключаються до Інтернету та збільшують кількість пристроїв, якими вони користуються. Це викликало тенденцію використання хмарних даних.

Система для зберігання даних у DynamoDB є найпростішою для реалізації тому що не вимагає додаткових бібліотек для роботи з даними [3]. Якщо використовувати найпопулярніші мови програмування, то хватить стандартних бібліотек, які представляється разом з мовою. Створення бази даних доволі просте і це можна зробити за допомогою AWS CLI. Ризик виникнення помилки при зберіганні доволі низький. Вартість зберігання 100 Гб даних за місяць вартує приблизно 50 доларів. Тому цю систему варто використовувати, якщо система не буде зберігати багато даних.

Системи для зберігання даних у RDS вимагає наявності драйвера для роботи з реляційними базами даних [7]. Також схему таблиці потрібно редагувати, якщо клієнту потрібно змінити якісь дані чи їх типи. Плюсами можна вважати швидке читання даних, швидка реалізація, наявність транзакційності, можливість створення зв'язків між іншими таблицями. Вартість зберігання 100 Гб даних за місяць вартує приблизно 110 доларів. Тому цю систему варто використовувати для інших задач.

Система для зберігання даних у S3 Bucket є, мабуть, найскладнішою з реалізацій, тому що в цій реалізації є декілька шляхів вирішення задачі. Перш за все потрібно вибрати формат зберігання даних. Найпопулярнішими на даний момент форматами файлами є JSON і CSV, але мінусом є те, що ці формати займають також багато місцями. Для того щоб файл був компактнішим варто використати Apache Parquet.

Apache Parquet – це бінарний, колонково-орієнтований формат зберігання великих даних, що спочатку створений для екосистеми Hadoop, що дозволяє використовувати переваги стисненого та ефективного колонково-орієнтованого подання інформації [1]. Паркет дозволяє задавати

схеми стиснення на рівні стовпців і додавати нові кодування в міру їхньої появи. Разом з Apache Avro, Parquet є дуже популярним форматом зберігання файлів Big Data і часто використовується в Kafka, Spark та Hadoop. Для роботи з Apache Parquet було використано бібліотеку PyArrow, який є Python API для Apache Arrow. Apache Arrow — це платформа розробки для аналітики в пам'яті. Він містить набір технологій, які дозволяють системам великих даних швидко зберігати, обробляти та переміщувати дані. Розмір файлів з однаковою кількістю даних в форматі Apache Parquet є меншою у 2–3 рази, ніж у JSON і CSV.

Вартість зберігання 100 Гб даних за місяць вартує приблизно 3 долари, проте дані більш оптимізовані для комп'ютерів і читання даних варто робити через інші інструменти: AWS Athena, AWS S3 Glacier тощо [2].

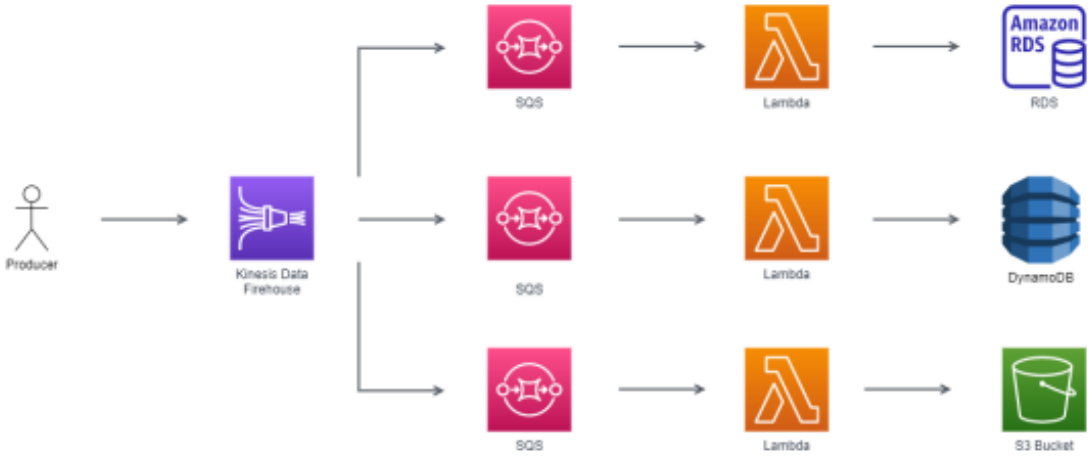


Рис. 1. Архітектура зберігання даних для дослідження

Підсумовуючи три системи, можна вважати, що для зберігання даних у великих обсягах найоптимальнішим і найефективнішим варіантом у довгостроковій реалізації буде система зберігання у S3 Bucket. Якщо клієнту потрібно швидкий і не довгостроковій реалізації, то варто використати систему зберігання у DynamoDB.

Список використаних джерел

1. Apache parquet. *Apache Parquet.* [Електронний ресурс]. – Режим доступу: https://parquet.apache.org/.
2. Cloud Object Storage – Amazon S3 – Amazon Web Services. *Amazon Web Services, Inc.* [Електронний ресурс]. – Режим доступу: https://aws.amazon.com/s3/.
3. Fast NoSQL Key-Value Database – Amazon DynamoDB – Amazon Web Services. *Amazon Web Services, Inc.* [Електронний ресурс]. – Режим доступу: https://aws.amazon.com/dynamodb/.
4. Serverless Computing – AWS Lambda – Amazon Web Services. *Amazon Web Services, Inc.* [Електронний ресурс]. – Режим доступу: https://aws.amazon.com/lambda.

5. Stedman C., Lutkevich B. What is a data lake? Definition from searchdatamanagement. *SearchDataManagement*. [Електронний ресурс]. – Режим доступу: https://www.techtarget.com/searchdatamanagement/definition/data-lake.
6. What is a data lake? *Amazon Web Services, Inc.* [Електронний ресурс]. – Режим доступу: https://aws.amazon.com/ru/big-data/datalakes-and-analytics/what-is-a-data-lake/.
7. Fully Managed Relational Database – Am azon RDS – Amazon Web Services. Amazon Web Services, Inc. [Електронний ресурс]. – Режим доступу: https://aws.amazon.com/rds (date of access: 22.11.2022).

Ковшар Р.О.,
здобувач вищої освіти ступеня магістра
Національного технічного університету України
"Київський політехнічний інститут імені Ігоря Сікорського"
науковий керівник: ***Кублій Л.І.,***
кандидат технічних наук, доцент цифрових технологій в енергетиці
Навчально-наукового інституту атомної та теплової енергетики
Національного технічного університету України
"Київський політехнічний інститут імені Ігоря Сікорського"

КЕРУВАННЯ ПОДАННЯМ ДАНИХ З РІЗНОРІДНИХ ДЖЕРЕЛ

Наразі у світі активно збільшується потік цифрових даних. Зі збільшенням цього потоку, виникає питання для зберігання та аналізу таких даних, оскільки аналіз може дати вектор для подальшого розвитку наукового дослідження, роботи бізнесу тощо.

Для виконання такого роду аналізу здебільшого використовуються реляційні бази даних. Вони забезпечують користувачів широким набором аналітичних можливостей. Але такий підхід накладає на роботу аналітиків певні обмеження:

- кожна база заточена під конкретну систему, тобто має свою схему;
- різні бази даних можуть мати різний синтаксис запитів та функціональність для роботи.

Такі обмеження значно ускладнюють можливість аналізу даних з різних реляційних баз даних, або ж використовувати для аналізу поєднання даних з таких баз.

З поміж багатьох рішень наведених проблем більш за всіх виділяється Apache Calcite [1].

Apache Calcite – це бібліотека, що використовується для побудови баз даних та систем управління базами даних. Бібліотека складається з SQL - парсеру, з засобів побудови виразу через реляційну алгебру і інструмент для планування виконання запиту. Calcite не зберігає в собі ні яких даних, проте може бути використаний для доступу до них.