МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ОДЕСЬКА ЮРИДИЧНА АКАДЕМІЯ»

ФАКУЛЬТЕТ КІБЕРБЕЗПЕКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра інформаційних технологій

КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«СИСТЕМНИЙ АНАЛІЗ ПОБУДОВИ МУЛЬТИБАЗОВИХ СИСТЕМ У DATA SCIENCE ПРОЄКТАХ»

Виконав: студент 2 курсу

рівень: магістр

галузь знань 12 «Інформаційні технології»

спеціальність 124 «Системний аналіз»

Новосельський Дмитро Олексійович

Керівник: к.пед.н., доцент

Лобода Юлія Геннадіївна

Рецензент: к.т.н., доцент

Чикунов Павло Олександрович

Одеса – 2025

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ОДЕСЬКА ЮРИДИЧНА АКАДЕМІЯ»
Факультет **кібербезпеки та інформаційних технологій**
Кафедра **інформаційних технологій**
Галузь знань: **12 Інформаційні технології**
Спеціальність: **124 Системний аналіз**
Назва освітньої програми: **Аналіз та безпека даних**

ЗАТВЕРДЖУЮ
Завідувач кафедри інформаційних технологій
доцент _____ Н.І. Логінова
«____ »_____ 20__ року

ЗАВДАННЯ
на кваліфікаційну (магістерську) роботу
здобувачу Новосельському Дмитро Олексійовичу

1. Тема роботи: «Системний аналіз побудови мультибазових систем у Data Science проєктах»

Керівник роботи: к.пед.н, доцент Лобода Юлія Геннадіївна
затверджені наказом НУ ОЮА від «*10*» *жовтня 2025* року № *3183-06*

2. Строк подання здобувачем роботи «*25*» *листопада 2025 рік*

3. Дата видачі завдання «*02*» *червня 2025 рік*

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської роботи	Строк виконання етапів роботи	Примітка
1.	Вибір теми, вивчення літературних джерел та складання плану роботи	09.09.2025	
2.	Підготовка першого розділу роботи та подання його керівнику	01.10.2025	
3.	Підготовка другого розділу роботи та подання його керівнику	14.10.2025	
4.	Підготовка третього розділу роботи та подання його керівнику	07.11.2025	
5.	Підготовка вступу та висновків	10.11.2025	
6.	Доопрацювання роботи з урахуванням зауважень керівника	24.11.2025	
7.	Подання електронного варіанту роботи для перевірки на плагіат	25.11.2025	
8.	Допуск роботи до захисту завідувачем кафедри	28.11.2025	
9.	Захист роботи	02.12.2025	

Здобувач _____ Новосельский Д.О.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Лобода Ю.Г.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Кваліфікаційна робота на тему «Системний аналіз побудови мультибазових систем у Data Science проєктах» на здобуття другого (магістерського) рівня вищої освіти за спеціальністю 124 Системний аналіз, освітньою програмою: Аналіз та безпека даних, містить 10 рисунків, 34 літературних джерела за переліком посилань. Робота виконана на 60 сторінках загального тексту і 62 сторінках основного тексту.

Метою дослідження є системний аналіз підходів до побудови мультибазових архітектур у Data Science-проєктах та розробка методики проєктування таких систем на основі принципів Polyglot Persistence.

Об’єктом дослідження є процеси побудови мультибазових систем у Data Science-проєктах.

Предметом дослідження є методи та моделі системного аналізу, що застосовуються для побудови мультибазових систем у Data Science-проєктах.

У роботі обґрунтовано актуальність застосування підходу Polyglot Persistence для побудови масштабованих Data Science проєктів. Виконано класифікацію моделей баз даних та проаналізовано особливості організації структурованих, напівструктурованих і потокових джерел інформації. Досліджено архітектурні моделі зберігання даних та підходи інтеграції в мультибазових середовищах. Проведено порівняльний аналіз інструментів та платформ. На основі системного аналізу сформовано архітектуру мультибазової платформи електронної комерції, що охоплює транзакційні дані, каталоги товарів, графові моделі рекомендацій та поведінкові події користувачів. Виконано формалізацію вимог, проєктування архітектури, побудовано моделі даних і здійснено аналітичне оцінювання ефективності поліглотного підходу. Отримані результати підтверджують переваги використання спеціалізованих СКБД для забезпечення масштабованості, гнучкої моделі даних та незалежного масштабування підсистем.

МУЛЬТИБАЗОВІ СИСТЕМИ, POLYGLOT PERSISTENCE, СИСТЕМНИЙ АНАЛІЗ; DATA SCIENCE, АРХІТЕКТУРНІ МОДЕЛІ, ДАНІ

ABSTRACT

The qualification thesis “System analysis of multidatabase system development in Data Science projects”, submitted for obtaining the second (master’s) level of higher education in speciality 124 System Analysis under the educational program Data Analysis and Security, contains 10 figures and 34 bibliographic sources. The thesis comprises 60 pages of total text and 62 pages of main content.

The aim of the research is to perform a system analysis of approaches to constructing multidatabase architectures in Data Science projects and to develop a methodology for designing such systems based on the principles of Polyglot Persistence.

The object of the research is the processes of building multidatabase systems in Data Science projects.

The subject of the research is the methods and models of system analysis applied to the construction of multidatabase systems in Data Science projects.

The thesis substantiates the relevance of applying the Polyglot Persistence approach for building scalable Data Science projects. A classification of database models has been performed, and the characteristics of structured, semi-structured, and streaming data sources have been analyzed. Architectural models of data storage and integration approaches in multidatabase environments have been investigated. A comparative analysis of tools and platforms has been conducted. Based on system analysis, the architecture of a multidatabase e-commerce platform has been designed, covering transactional data, product catalogues, graph-based recommendation models, and user behavioural events. Requirements formalization, architectural design, data model construction, and analytical evaluation of the effectiveness of the polyglot approach have been carried out. The obtained results confirm the advantages of using specialized database systems to ensure scalability, flexible data modelling, and independent subsystem scaling.

MULTIDATABASE SYSTEMS, POLYGLOT PERSISTENCE, SYSTEM ANALYSIS, DATA SCIENCE, ARCHITECTURAL MODELS, DATA.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

BDI – Beliefs-Desires-Intentions

СКБД – системи управління базами даних

ACID – Atomicity, Consistency, Isolation, Durability

JOIN - об’єднання таблиць у SQL

IoT – Internet of Things

DevOps – Development and Operations

JSON - JavaScript Object Notation

CAP – Consistency, Availability, Partition Tolerance

ML - Machine Learning

ETL – Extract, Transform, Load

ELT – Extract, Load, Transform

S3 – Amazon Simple Storage Service

HDFS – Hadoop Distributed File System

ADLS – Azure Data Lake Storage

CDC - Change Data Capture

ВСТУП

У сучасних умовах розвитку цифрових технологій та інтенсивного використання методів Data Science ключовим викликом стає ефективна робота з великими, різнорідними та розподіленими масивами даних. Традиційні рішення часто не забезпечують необхідної масштабованості, гнучкості та швидкості обробки , що ускладнює реалізацію аналітичних і машинних задач. У цьому контексті **актуальним** є дослідження принципів побудови мультибазових систем, які дозволяють інтегрувати дані з різних джерел, оптимізувати їхнє зберігання та забезпечити узгоджений доступ у рамках складних Data Science-проєктів.

Метою дослідження є системний аналіз підходів до побудови мультибазових архітектур у Data Science-проєктах та розробка методики проєктування таких систем на основі принципів Polyglot Persistence.

Об’єктом дослідження є процеси побудови мультибазових систем у Data Science-проєктах.

Предметом дослідження є методи та моделі системного аналізу, що застосовуються для побудови мультибазових систем у Data Science-проєктах.

Для досягнення поставленої мети в кваліфікаційній роботі необхідно вирішити наступні **завдання**:

– дослідити предметну область з точки зору системного аналізу;

– проаналізувати сучасні підходи побудови мультибазових архітектур та принципи Polyglot Persistence;

– дослідити архітектурні моделі зберігання даних та систематизувати підходи до інтеграції даних;

– формалізувати вимоги до мультибазової системи у вибраній предметній області;

– сформувати рекомендації щодо вибору інструментів для побудови мультибазових систем;

– розробити архітектурну модель мультибазової системи;

– провести аналітичне дослідження ефективності запропонованої архітектури.

В кваліфікаційній роботі використовувалися загальнонаукові та спеціальні методи наукового дослідження таки, як системний аналіз, синтез, порівняння, статистичний аналіз, методи машинного навчання, візуалізації даних тощо.

Наукова новизна полягає у формуванні системного підходу до проєктування мультибазових архітектур на основі принципів Polyglot Persistence, що поєднує класифікацію моделей даних, критерії вибору систем управління базами даних та методи інтеграції гетерогенних джерел у інтегровану аналітичну інфраструктуру.

Теоретична значущість проведеного дослідження полягає в уточненні концептуальних засад мультибазових систем і систематизації архітектурних моделей зберігання та інтеграції даних у сучасних аналітичних середовищах. Отримані результати розширюють наукове підґрунтя для проєктування масштабованих та узгоджених Data Science-архітектур.

Практичне значення одержаних результатів полягає у формуванні рекомендацій та архітектурних рішень, які можуть бути безпосередньо застосовані під час побудови мультибазових систем у реальних Data Science-проєктах. Розроблена архітектурна концепція може слугувати практичним шаблоном при створенні систем аналітики, рекомендаційних сервісів та високонавантажених платформ обробки даних.

Результати кваліфікаційного дослідження **апробовані** на VIII Всеукр. наук.-практ. інтернет-конф. студентів, аспірантів та молодих вчених за тематикою «Сучасні комп’ютерні системи та мережі в управлінні» (24 листопада 2025 р., м. Херсон, м. Хмельницький) [34]

1 ТЕОРЕТИЧНІ ОСНОВИ МУЛЬТИБАЗОВИХ СИСТЕМ У DATA SCIENCE

1.1 Класифікація моделей баз даних у мультибазових системах

Сучасні інформаційні системи оперують даними, що відрізняються за структурою, формою подання, швидкістю оновлення та семантичними властивостями. У межах Data Science-проєктів ця різноманітність стає ще більш вираженою: поряд із транзакційними записами використовуються напівструктуровані документи, графи взаємодій, часові ряди, неструктуровані медіадані .[1] У таких умовах одна модель даних не може забезпечити оптимальну ефективність для всіх сценаріїв, що зумовлює необхідність поліглотних архітектур, де різні типи СКБД виконують спеціалізовані функції.

Модель даних є абстрактним способом організації та опису інформації й визначає операції над нею. Нижче наведено основні класи моделей, що застосовуються у мультибазових платформах.

1. Реляційна модель даних.

Реляційна модель ґрунтується на поданні даних у вигляді таблиць зі строго визначеними структурами атрибутів [1]. Чіткість схеми, референтна цілісність та підтримка транзакцій роблять її еталоном для систем, що вимагають високої точності та передбачуваності обробки. Ключовими характеристиками є:

– структуроване подання інформації, де кожен рядок відповідає сутності, а стовпці – її атрибутам;

– гарантії ACID, що забезпечують узгодженість навіть у багатокористувацьких сценаріях;

– нормалізація, яка зменшує дублювання та підвищує коректність даних [2].

Реляційна модель оптимальна для фінансових, облікових, транзакційних операцій, де помилки неприпустимі. Водночас її жорстка схема ускладнює роботу з динамічними або нерегулярними структурами даних.

2. Документна модель даних

Документноорієнтовані моделі призначені для роботи з гнучкими структурами – документами, що можуть містити вкладені об’єкти, масиви та довільні атрибути. На відміну від реляційної моделі, тут допускається варіативність схеми між окремими документами [3].

Характерні властивості:

– динамічна структура, що дозволяє зберігати неоднорідні записи;

– вкладеність об’єктів, яка зменшує потребу у складних зв’язках;

– природне відображення об’єктних моделей, що пришвидшує взаємодію з прикладною логікою.

Документні БД ефективні у випадках, коли атрибути сутностей різняться залежно від категорії, або коли структура інформації часто еволюціонує.

3. Колонкова модель даних

Колонкові моделі оптимізовано для аналітичних запитів, де переважає читання великих масивів даних із подальшими агрегаціями. Організація зберігання за стовпцями забезпечує:

– ефективне стиснення, оскільки значення одного атрибута розміщені компактно;

– швидке сканування стовпців, що суттєво прискорює аналітичні операції;

– високу продуктивність у запитах, де використовується частина атрибутів [4].

Колонкові моделі найкраще підходять для аналітики, побудови звітності, вітрин даних, обробки подій та часових рядів.

4. Графова модель даних

Графова модель представляє інформацію у вигляді вузлів і зв’язків між ними. На відміну від реляційних систем, де зв’язки потрібно відтворювати сумою JOIN-операцій, графи подають структуру зв’язків як перший клас об’єктів [3, 4].

Переваги:

– оптимальність для навігаційних запитів (пошук шляхів, виявлення спільнот);

– гнучка структура для складних реляційних патернів;

– можливість роботи з напівструктурованими сутностями.

Графові моделі незамінні для рекомендаційних систем, мережевої аналітики, систем виявлення шахрайства та аналізу взаємодій.

5. Модель "ключ–значення"

Key-value модель – це найпростіший і найшвидший спосіб зберігання даних, у якому кожному ключу відповідає одне значення. Основні властивості:

– мінімальна затримка доступу, що робить модель придатною для кешів;

– висока продуктивність при простих запитах;

– масштабованість за рахунок природного шардінгу за ключами [5].

Таке сховище використовується для тимчасових станів, сесій користувачів, проміжних результатів обчислень.

6. Модель часових рядів

Time-series моделі призначені для даних, що змінюються з часом: метрик, сигналів, показників системи. Вони підтримують:

– високошвидкісний запис потоків подій;

– ефективні віконні функції та агрегації;

– стиснення за часовими інтервалами [6].

Часові ряди застосовують у моніторингу, IoT, DevOps-аналітиці, прогнозуванні.

7. Порівняльний аналіз моделей даних

Кожна модель даних спеціалізується на власному наборі задач. Реляційні системи забезпечують максимальну консистентність, документні – гнучкість, колонкові – аналітичну продуктивність, графові – складні зв’язки, key-value – швидкість, time-series – потоки подій.

Саме це різноманіття ролей робить поліглотні архітектури фундаментом сучасних мультибазових рішень.

1.2 Особливості організації та обробки даних у Data Science-проєктах

Data Science-проєкти потребують обробки різнорідних даних із різними властивостями. Особливості організації таких даних визначають вибір моделей зберігання, архітектурну побудову систем та характер інтеграцій.

1. Структурованість та різноманітність даних

Дані у Data Science поділяють на три основні категорії:

– структуровані дані – мають стабільну схему (транзакції, табличні записи);

– напівструктуровані – містять гнучкі або змінні атрибути (JSON, журнали подій);

– неструктуровані – текст, зображення, аудіо, відео [7].

Чим більша різноманітність даних, тим складніше забезпечити уніфікований підхід до їх інтерпретації, зберігання та аналізу. Саме тому Data Science-системи майже завжди є мультибазовими.

2. Масштабованість та продуктивність

Data Science-проєкти обробляють значні обсяги інформації, що породжує специфічні вимоги:

– горизонтальне масштабування – необхідне для роботи із терабайтами даних;

– різні режими обробки: реального часу – для рекомендацій та моніторингу; інтерактивні – для аналітики; пакетні – для навчання моделей;

– висока пропускна здатність, особливо для поведінкових подій та логів.

Таким чином, проєкти Data Science вимагають архітектур, здатних обробляти і швидкі потоки, і великі історичні набори.

3. Вимоги до доступності та надійності

Оскільки дані є ключовим активом проєкту, їхня доступність та захищеність мають визначальне значення. Основні характеристики:

– доступність (availability) – визначає тривалість безвідмовної роботи;

– надійність – гарантує збереження даних навіть у випадку збоїв;

– узгодженість – баланс між точністю та доступністю відповідно до принципу CAP [8].

У Data Science-проєктах окремі компоненти можуть вимагати різних моделей узгодженості, що ускладнює архітектуру мультибазових систем.

4. Гетерогенність джерел та виклики інтеграції

Системи мають інтегрувати дані з різних джерел, які відрізняються [8]:

– моделями даних,

– форматами,

– частотою оновлення,

– семантикою,

– якістю.

Це створює потребу в спеціальних інтеграційних механізмах, що можуть забезпечити узгодженість інформації без втрати продуктивності.

5. Обробка даних для моделей машинного навчання

На відміну від класичних інформаційних систем, Data Science-проєкти потребують:

– централізованих сховищ ознак (feature stores),

– версіонування даних та моделей,

– контролю відтворюваності експериментів,

– розподіленого навчання для великих датасетів.

Ці вимоги ускладнюють архітектуру та ще більше підсилюють необхідність поліглотних рішень.

Особливості обробки даних у Data Science-проєктах зумовлюють використання різних типів сховищ, режимів обробки та моделей узгодженості. Гетерогенність даних, вимоги до масштабованості та потреба у відтворюваних ML-процесах формують архітектуру, де мультибазові системи є природним і необхідним рішенням.

1.3 Архітектурні моделі зберігання та інтеграції даних у мультибазових рішеннях

Data Science-проєкти потребують обробки різнорідних даних із різними властивостями. Особливості організації таких даних визначають вибір моделей зберігання, архітектурну побудову систем та характер інтеграцій.

1. Структурованість та різноманітність даних

Дані у Data Science поділяють на три основні категорії:

– структуровані дані – мають стабільну схему (транзакції, табличні записи);

– напівструктуровані – містять гнучкі або змінні атрибути (JSON, журнали подій);

– неструктуровані – текст, зображення, аудіо, відео.

Чим більша різноманітність даних, тим складніше забезпечити уніфікований підхід до їх інтерпретації, зберігання та аналізу. Саме тому Data Science-системи майже завжди є мультибазовими [7, 8].

2. Масштабованість та продуктивність

Data Science-проєкти обробляють значні обсяги інформації, що породжує специфічні вимоги:

– горизонтальне масштабування – необхідне для роботи із терабайтами даних;

– різні режими обробки:

– реального часу – для рекомендацій та моніторингу;

– інтерактивні – для аналітики;

– пакетні – для навчання моделей;

– висока пропускна здатність, особливо для поведінкових подій та логів [11].

Таким чином, проєкти Data Science вимагають архітектур, здатних обробляти і швидкі потоки, і великі історичні набори.

3. Вимоги до доступності та надійності

Оскільки дані є ключовим активом проєкту, їхня доступність та захищеність мають визначальне значення. Основні характеристики:

– доступність (availability) – визначає тривалість безвідмовної роботи;

– надійність – гарантує збереження даних навіть у випадку збоїв;

– узгодженість – баланс між точністю та доступністю відповідно до принципу CAP [12].

У Data Science-проєктах окремі компоненти можуть вимагати різних моделей узгодженості, що ускладнює архітектуру мультибазових систем.

4. Гетерогенність джерел та виклики інтеграції

Системи мають інтегрувати дані з різних джерел, які відрізняються: моделями даних, форматами, частотою оновлення, семантикою, якістю [10].

Це створює потребу в спеціальних інтеграційних механізмах, що можуть забезпечити узгодженість інформації без втрати продуктивності.

5. Обробка даних для моделей машинного навчання

На відміну від класичних інформаційних систем, Data Science-проєкти потребують:

– централізованих сховищ ознак (feature stores),

– версіонування даних та моделей,

– контролю відтворюваності експериментів,

– розподіленого навчання для великих датасетів.

Ці вимоги ускладнюють архітектуру та ще більше підсилюють необхідність поліглотних рішень.

Особливості обробки даних у Data Science-проєктах зумовлюють використання різних типів сховищ, режимів обробки та моделей узгодженості. Гетерогенність даних, вимоги до масштабованості та потреба у відтворюваних ML-процесах формують архітектуру, де мультибазові системи є природним і необхідним рішенням [12].

Висновки до розділу 1

У першому розділі систематизовано теоретичні засади мультибазових систем у контексті сучасних Data Science-проєктів та обґрунтовано необхідність використання поліглотних архітектур для роботи з різнорідними даними. Показано, що жодна окрема модель даних не може забезпечити оптимальну ефективність одночасно для транзакційних операцій, гнучкого зберігання напівструктурованої інформації, високопродуктивної аналітики, аналізу зв’язків та обробки потоків подій.

Розглянуто специфіку даних у Data Science-системах, що характеризуються різними рівнями структурованості, високою динамікою надходження, великими обсягами та значною семантичною варіативністю. Обґрунтовано, що вимоги до масштабованості, доступності та узгодженості даних формують комплекс обмежень, які не можуть бути задоволені єдиною моделлю або єдиною СКБД. Значну роль відіграють також особливості ML-процесів, які потребують централізованого управління ознаками, версіонування даних та підтримки відтворюваності експериментів.

Проаналізовано архітектурні моделі зберігання які визначають способи організації даних та інтеграції між компонентами мультибазових систем. Показано, що кожна з моделей має власне призначення та обмеження, а їх комбінування забезпечує баланс між гнучкістю, продуктивністю та керованістю даних.

Таким чином, перший розділ формує основу для подальшого аналізу технологій інтеграції та практичної реалізації мультибазових рішень у Data Science-проєктах, визначаючи теоретичні принципи, що лежать в основі поліглотних і гібридних архітектур.

2 ПІДХОДИ ТА ТЕХНОЛОГІЇ ПОБУДОВИ МУЛЬТИБАЗОВИХ СИСТЕМ

2.1 Концепція Polyglot Persistence та принципи розподілу функцій між СКБД

Мультибазові системи ґрунтуються на концепції Polyglot Persistence, яка передбачає використання кількох різних СКБД у межах однієї програмної платформи. Причина такого підходу полягає в тому, що сучасні Data Science-проєкти працюють з даними різних типів, різної структури та з різними вимогами до узгодженості, швидкодії та аналітичної складності. Одна універсальна СКБД неминуче створює компроміси, а сам те, що добре працює для транзакційних навантажень, майже завжди не підходить для аналітичних запитів, графових обходів або роботи з напівструктурованими документами [13].

Polyglot Persistence надає можливість поєднати сильні сторони різних СКБД і використовувати кожну там, де вона максимізує продуктивність та масштабованість.

У центрі концепції Polyglot Persistence лежать три фундаментальні положення, які визначають логіку застосування різних СКБД у межах однієї системи [14].

Принцип 1. Відсутність універсальної СКБД.

У сучасних цифрових системах різні частини архітектури виконують різні типи операцій: транзакції з високими вимогами до узгодженості (замовлення, платежі); аналітичні агрегати на великих обсягах даних (поведінкові події); обхід графів зв’язків (рекомендації); пошук за ключовими словами (фасетний пошук).

Жодна СКБД не може однаково ефективно виконувати всі ці задачі, оскільки вони потребують різних алгоритмів оптимізації, моделей зберігання, індексів та рівнів узгодженості. На практиці це працює, таким чином:

– PostgreSQL забезпечує ACID-транзакції, але не може конкурувати з ClickHouse у швидкості агрегацій по мільярдах рядків;

– MongoDB зберігає документи з різною структурою, але має значні обмеження для JOIN-операцій;

– Neo4j виконує графові обходи у сотні разів швидше за SQL, але повністю не підходить для аналітики .

Отже, вибір однієї СКБД для всіх цілей призводить до того, що критично важливі частини системи працюють неефективно.

Принцип 2. Спеціалізація забезпечує технологічну перевагу.

Сучасні СКБД розвиваються в напрямку глибокої оптимізації під конкретні класи задач: документоорієнтовані СКБД оптимізовані під вкладені JSON-структури, колонкові – під масові сканування і агрегації, графові – під пошук шляхів і зіставлення з шаблоном, key-value – під наднизьку латентність [15].

Лабораторні та промислові вимірювання показують виграш у продуктивності на порядок і більше [17]:

– ClickHouse: запит SELECT COUNT(*) FROM events → 0.2–1.0 сек;

– PostgreSQL на тих самих даних → 20–60 сек;

– MongoDB → суттєво повільніше або потребує денормалізації.

Це означає, що поєднання декількох спеціалізованих СКБД дає значно кращі результати, ніж спроби універсалізації.

Принцип 3. Ускладнена інтеграція компенсується виграшем у продуктивності.

Використання кількох СКБД вимагає різних механізмів реплікації, моніторингу, резервного копіювання та оновлення. Однак у складних системах ці витрати компенсуються: швидкодією, незалежним масштабуванням, зростанням пропускної здатності, підвищенням стійкості системи. Наприклад, аналітичні запити ClickHouse виконуються ізольовано від транзакцій PostgreSQL, тому аналітичні навантаження не впливають на покупки, платежі чи оновлення кошика.

Переваги застосування Polyglot Persistence:

1. Оптимальна продуктивність кожного компонента.

Система використовує найбільш ефективну СКБД для кожного класу задач:

– транзакції → PostgreSQL (ACID),

– каталог з варіативними атрибутами → MongoDB,

– граф рекомендацій → Neo4j,

– поведінкові події → ClickHouse,

– повнотекстовий пошук → Elasticsearch.

2. Незалежне масштабування компонентів.

Кожен блок масштабується відповідно до характеру навантаження:

– PostgreSQL → вертикальне масштабування;

– ClickHouse → горизонтальне;

– Redis → масштабування у пам’яті;

– MongoDB → шардінг за документами.

3. Ізоляція навантаження

Аналітика не впливає на транзакції, batch-процеси не заважають online-операціям, а ML-експерименти не знижують продуктивність production-сервісів.

4. Гнучкість розвитку системи.

Нові модулі можуть додавати нові СКБД, не змінюючи існуючих. Наприклад, додавання Elasticsearch не вимагає модифікації PostgreSQL або MongoDB.

Polyglot-архітектура має свої ризики і складності:

1. Операційна складність.

Необхідно підтримувати кілька систем резервного копіювання, моніторингу, адміністративних інструментів.

2. Узгодженість даних між СКБД.

Оскільки глобальних транзакцій між СКБД не існує, узгодженість забезпечують: асинхронна реплікація (CDC), подійно-орієнтована інтеграція (Kafka), компенсуючі транзакції (патерн Saga).

3. Складність інтеграції та управління даними.

Для коректної взаємодії між СКБД необхідні: інструменти контролю якості даних, механізми відповідність схем, засоби перетворення структур, федеративні запити.

4. Високі вимоги до компетенцій команди.

Команда має володіти щонайменше однією SQL-СКБД, однією NoSQL-СКБД, однією аналітичною (ClickHouse, BigQuery), потоковими технологіями (Kafka), принципами розподілених систем [17].

Порівняння монолітного та polyglot-підходів представлено у таблиці 2.1.

Таблиця 2.1 – Порівняння монолітного та polyglot підходів

Характеристика	Монолітна архітектура	Polyglot Persistence
Продуктивність	Компромісна	Максимальна для окремих задач
Масштабованість	Обмежена	Незалежна для кожного компонента
Узгодженість	Strong ACID	Eventual між СКБД
Час розробки	Швидший старт	Повільніша інтеграція
Операційна складність	Низька	Висока
Вимоги до команди	Невеликі	Високі
Застосування	Простий CRUD, MVP	Великі аналітичні системи, e-commerce, ML

Порівняльний аналіз підтверджує, що монолітна архітектура є зручною на початкових етапах, але швидко вичерпує свої можливості, коли система розширюється, збільшується кількість користувачів, зростає різноманітність даних та навантаження. Polyglot Persistence, навпаки, дозволяє масштабувати кожен компонент незалежно, підбираючи оптимальну СКБД для кожного типу даних і кожного класу операцій.

Таким чином, Polyglot Persistence є не просто альтернативою, а архітектурною необхідністю для сучасних e-commerce платформ, систем з високими аналітичними навантаженнями, ML-орієнтованих сервісів та комплексних інформаційних платформ. Переваги цього підходу суттєво переважають його операційну складність за умови правильного застосування механізмів синхронізації, управління даними та розподілу функцій між СКБД.

2.2 Підходи до інтеграції даних у мультибазових системах

Інтеграція даних у мультибазових архітектурах є фундаментальним компонентом, що забезпечує узгодженість між різнорідними СКБД, підтримує зв’язність даних та

створює можливість виконання комплексної аналітики. Сучасні системи використовують кілька підходів інтеграції, кожен з яких орієнтований на свій тип навантажень, обсяг даних та вимоги до латентності.

1. ETL (Extract–Transform–Load).

Традиційний підхід, у якому дані спочатку вилучаються, проходять трансформацію і лише після цього завантажуються у цільове сховище. ETL характерний для класичних аналітичних систем (Data Warehouse), де бізнес-логіка суворо визначена наперед, а узгоджена схема даних є обов'язковою умовою [17]. У мультибазових архітектурах ETL переважно застосовується для перенесення структурованих даних із реляційних систем у колонкові СКБД (наприклад, PostgreSQL → ClickHouse).

Архітектура ETL процесу показана на рисунку 2.1.

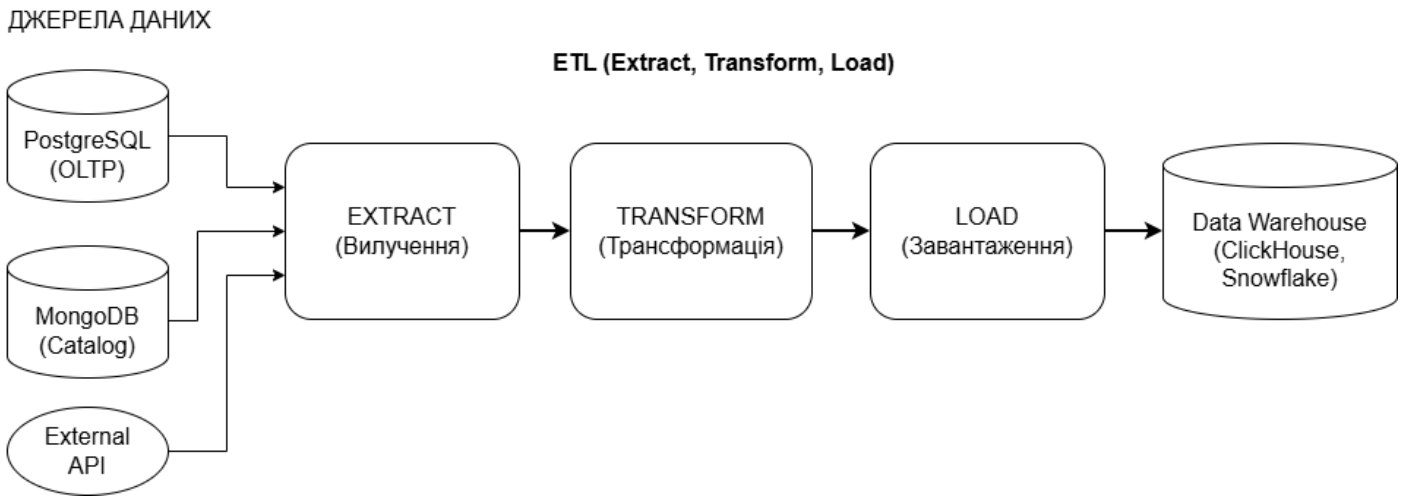


Рисунок 2.1 – Архітектура ETL (Extract, Transform, Load)

2. ELT (Extract–Load–Transform)

На відміну від ETL, підхід ELT передбачає спочатку завантаження «сирих» даних у масштабоване сховище (S3, HDFS, ADLS), а трансформації виконуються вже безпосередньо у середовищі зберігання. ELT широко застосовується у Data Lake та LakeHouse-платформах.

У мультибазових системах ELT є основним механізмом перенесення великих поведінкових потоків та напівструктурованих даних у колонкові системи аналітики.

Архітектура ELT процесу наведена на рисунку 2.2.

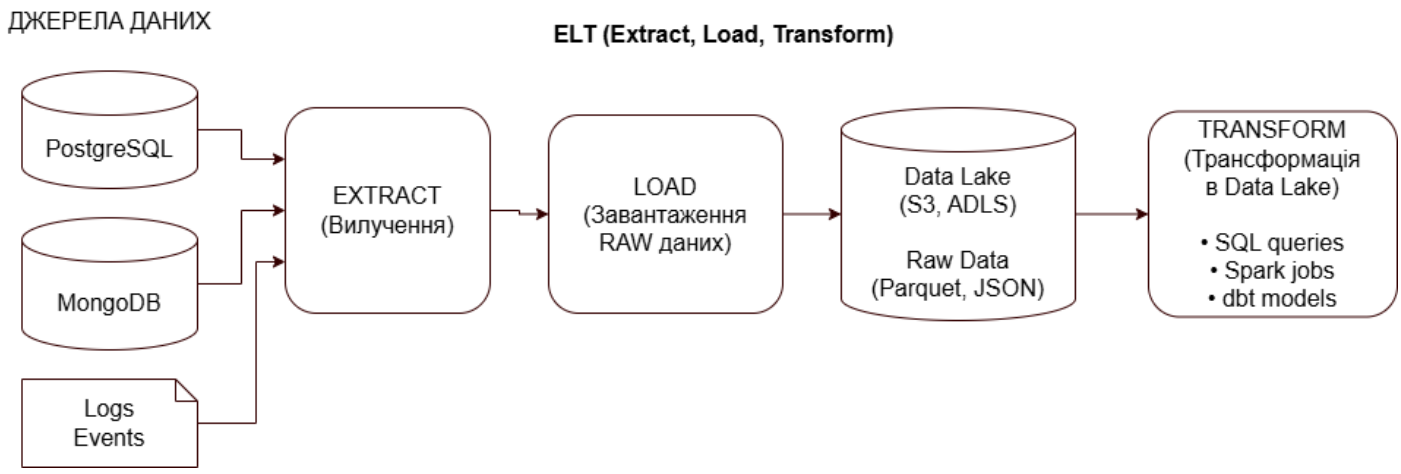


Рисунок 2.2 – Архітектура ELT (Extract, Load, Transform)

Різниця між ETL та ELT у мультибазових системах полягає не лише в порядку операцій, а й у роботи з даними: ETL забезпечує строгий контроль узгодженості та структури, тоді як ELT – гнучкість, масштабованість і швидкість обробки великих обсягів.

3. Change Data Capture (CDC)

CDC – підхід, який відстежує зміни у транзакційній СКБД через журнали змін та поширює їх у інші системи майже в реальному часі [18].

CDC забезпечує асинхронну узгодженість і дозволяє оновлювати множину СКБД без необхідності виконання прямих записів у кілька сховищ одночасно.

Приклад роботи, створення нового замовлення у PostgreSQL породжує подію OrderCreated, яка протягом мілісекунд оновлює:

– документ «профіль користувача» у MongoDB,
– графові зв’язки у Neo4j,
– подієві журнали у ClickHouse,

– кеш у Redis.

Таким чином CDC разом із Kafka формує транспортний шар мультибазової системи.

4. Федеративні запити (Federated Queries).

Модель, що дозволяє виконувати запити поверх кількох СКБД без фізичного переміщення даних. Федеративні запити використовуються у тих випадках, коли потрібні оперативні аналітичні зрізи або дослідження, але копіювання даних є недоцільним [19]. Наприклад, бізнес-аналітик може виконати запит JOIN між таблицею замовлень у PostgreSQL та каталогом товарів у MongoDB, не створюючи дубльованого сховища.

Однак цей підхід має суттєві обмеження: продуктивність залежить від повільнішого джерела, складні агрегації над великими обсягами значно повільніші, а гарантії узгодженості залежать від кожної СКБД окремо. Саме тому віртуалізація застосовується обмежено для оперативної аналітики, але не для важких навантажень.

5. Потокова інтеграція (Event-Streaming Integration)

Модель, у якій дані передаються у вигляді подій, що обробляються у режимі низької затримки. У мультибазових системах потокова інтеграція використовується для:

– відстеження поведінкових подій (page_view, click, add_to_cart),
– онлайн-аналітики у ClickHouse,
– оновлення графів схожості,
– детекції шахрайських транзакцій у Flink,
– формування моделей рекомендацій [19].

Перевага потокового підходу полягає у низькій латентності (<100 мс), scalability-by-design та можливості одночасно оновлювати десятки різних сховищ. На відміну від batch-процесів, event-driven інтеграція дозволяє системі реагувати на події майже миттєво, що є критично важливим для e-commerce або фінансових платформ.

6. Віртуалізація даних (Data Virtualization)

Метод, що дозволяє працювати з даними у різних СКБД як з єдиним логічним джерелом. На відміну від федеративних запитів, віртуалізація створює абстрактний рівень представлення даних, приховуючи від користувача фізичні джерела [20].

Застосовується тоді, коли потрібен єдиний логічний інтерфейс, небажано дублювати дані, аналітичне навантаження не є важким.

У мультибазових системах неможливо обрати один універсальний спосіб інтеграції, кожен має своє призначення. ETL забезпечує якість і стабільність даних, ELT – гнучкість для роботи з великими обсягами, CDC – оперативну синхронізацію транзакцій, федеративні запити – швидкий доступ для аналітиків, а потокові системи – найменшу затримку обробки. Всі вони доповнюють один одного, формуючи інтеграційний каркас, необхідний для узгодженого функціонування СКБД.

2.3 Інструменти та платформи реалізації мультибазових систем

Практична реалізація мультибазових систем вимагає цілісного технологічного стеку, здатного інтегрувати СКБД, забезпечувати потокову та пакетну обробку, підтримувати різні моделі узгодженості та гарантувати масштабованість у межах складних Data Science-архітектур. На відміну від систем, де всі дані зберігаються у межах однієї СКБД, мультибазові рішення потребують спеціалізованих інструментів для обміну, синхронізації, трансформації та аналітичного доступу.

1. Інструменти Change Data Capture (CDC).

Debezium – одна з найбільш стабільних платформ, побудованих за моделлю відкритого програмного коду CDC [21]. Вона працює поверх Kafka та зчитує журнали змін (PostgreSQL WAL, MySQL binlog, MongoDB oplog), публікуючи їх у вигляді подій. Події містять попередній і новий стан запису, час фіксації транзакції та технічні метадані, що забезпечує коректну обробку і можливість повторного відтворення.

Переваги Debezium:

– мінімальний вплив на вихідну СКБД;

– підтримка еволюції схем та інтеграція з Schema Registry;
– латентність 100–500 мс;
– можливість підключення десятків цільових СКБД через Kafka Connect.

Airbyte та Fivetran інструменти автоматизованої інтеграції даних, які реалізують передачу даних у форматі ELT [22, 23]. Airbyte забезпечує великий набір конекторів (>300), підтримує API-орієнтовані джерела та може працювати у batch- і micro-batch-режимах. Fivetran – кероване комерційне рішення з мінімальними вимогами до налаштування, але більш високою вартістю.

2. Платформи потокової обробки подій

Apache Kafka – домінуюча технологія потокової обробки завдяки: пропускній здатності до мільйонів подій за секунду, горизонтальному масштабуванню, високій відмовостійкості, можливості повторного відтворення подій, підтримці гарантованої одинарної доставки [24].

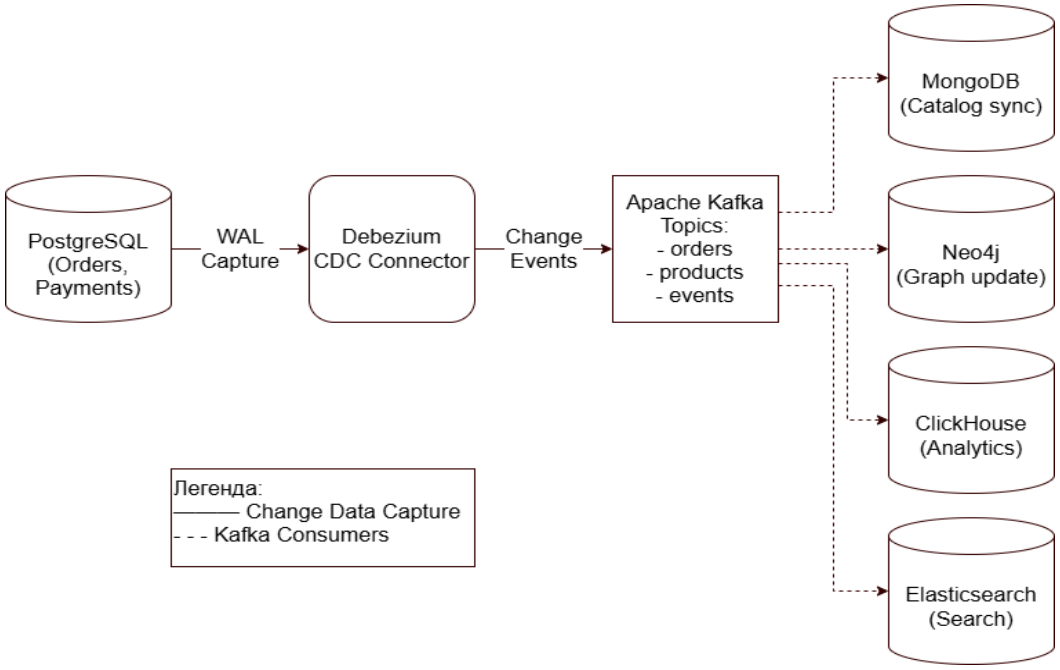
Kafka забезпечує незалежність між джерелом і споживачами, дозволяючи синхронізувати PostgreSQL → MongoDB → Neo4j → ClickHouse без дублювання логіки. На Kafka будується більшість сучасних e-commerce платформ, де сотні тисяч подій (page_view, click, add_to_cart) потрібно передавати в декілька сховищ одночасно.

Apache Pulsar та Redpanda є альтернативами Kafka. Pulsar [25] вирізняється багатокористувацькою архітектурою та розподіленим зберіганням, тоді як Redpanda орієнтована на мінімальну латентність (<10 мс) і спрощене адміністрування [26].

Архітектуру потокової інтеграції між PostgreSQL та іншими СКБД через CDC та Kafka наведено на рисунку 2.3.

На рисунку подано архітектуру потокової інтеграції даних у мультибазовій системі, що поєднує механізм Change Data Capture та подійно-орієнтовану шину Kafka. Зміни у транзакційній базі PostgreSQL фіксуються на рівні журналу WAL та передаються до конектора Debezium, який перетворює їх у структуровані події. Далі події публікуються у відповідні Kafka-топіки, звідки їх асинхронно споживають інші СКБД – MongoDB, Neo4j, ClickHouse та Elasticsearch. Такий підхід забезпечує низьку

латентність оновлень, ізоляцію навантаження між компонентами та незалежне масштабування кожного з контурів.



Рисунка 2.3 – Потокова інтеграція за допомогою CDC + Kafka

Схема демонструє ключову перевагу потокової інтеграції: усі споживачі отримують актуальні зміни без необхідності прямих записів у кілька СКБД. Це усуває ризики розсинхронізації та значно спрощує архітектуру порівняно з підходами прямого дублювання даних.

3. Інструменти для федеративних запитів

Федеративні запити дають змогу отримувати дані з різних СКБД без копіювання до єдиного сховища.

Найпоширенішими запитами є Trino (Presto), Dremio та Starburst.

Trino виконує SQL-запити поверх PostgreSQL, MongoDB, Elasticsearch, S3, ClickHouse та інших систем. Він розподіляє запит на підзапити, виконує їх безпосередньо на стороні джерела (push-down оптимізація) та поєднує результати.

Порівняння підходів інтеграції представлено у таблиці 2.2

Таблиця 2.2 – Порівняння підходів інтеграції

Підхід	Латентність	Узгодженість	Складність	Оптимальні сценарії
CDC (Debezium)	0.1–0.5 с	Eventual	Висока	Синхронізація транзакційних БД
Managed ELT (Fivetran)	1–5 хв	Eventual	Низька	BI, DWH
Batch ETL (Airflow)	Години	Strong в точці sync	Середня	Історичні агрегати
Federated queries (Trino)	миттєва	Strong (у момент запиту)	Середня	Ad-hoc аналітика
Event streaming (Kafka)	<100 мс	Eventual	Висока	Потокові системи, ML, рекомендації

4. Платформи для реалізації мультибазових архітектур.

На відміну від інструментів інтеграції, які працюють на операційному рівні, платформи формують технологічний фундамент для мультибазових архітектур.

Databricks Lakehouse Platform [28].

Databricks поєднує парадигми Data Lake та Data Warehouse та забезпечує ACID-гарантії для відкритих форматів (Parquet) завдяки Delta Lake. Платформа підтримує як batch-, так і streaming-обробку, забезпечує ML-інструментарій (MLflow) та централізоване управління метаданими (Unity Catalog).

Snowflake з Modern Data Stack.

Snowflake – cloud-native data warehouse з повним поділом зберігання та обчислень. Підтримує автоматичне масштабування, time travel, zero-copy cloning, оптимізований SQL-двигун.

Відкритий стек (Spark + Kafka + Airflow).

Забезпечує максимальну гнучкість і контроль. Apache Spark використовується для пакетної та потокової обробки даних, Kafka – для подій, Airflow – для оркестрації процесів, а Trino – для федеративних запитів. Потребує високої технічної експертизи та окремої команди підтримки.

5. Рекомендації щодо вибору технологічного стеку.

Вибір платформи залежить від характеру навантажень і організаційної складності:

– BI-орієнтовані компанії → Snowflake + Modern Data Stack.

– ML-орієнтовані проєкти з великими даними → Databricks Lakehouse.

– Компанії з сильною інженерною командою → Open-source стек (Spark + Kafka + Airflow).

Для систем електронної комерції оптимальним є гібридний підхід, який поєднує: PostgreSQL – транзакції, MongoDB – каталог, Neo4j – рекомендації, ClickHouse – аналітика, Kafka + Debezium – інтеграційний шар, Airflow – пакетні процеси, Trino – федеральная аналітика.

Інструменти та платформи мультибазових систем виконують взаємодоповнюючі функції. CDC-механізми забезпечують оперативну синхронізацію, потокові системи – мінімальну затримку, федеративні запити – доступ до різнорідних джерел без дублювання, а платформні рішення (Databricks, Snowflake, відкритий стек) формують інфраструктурну основу для комплексних Data Science-проєктів. Гібридний підхід, що поєднує спеціалізовані СКБД та подійно-орієнтовану інтеграцію, є оптимальним для складних доменів, таких як електронна комерція.

Висновки до розділу 2

У другому розділі проведено комплексний аналіз підходів, технологій та платформ, що визначають принципи побудови мультибазових архітектур у Data Science-проєктах. Обґрунтовано, що застосування концепції Polyglot Persistence є не опцією, а необхідністю для систем, які працюють із різними типами даних – структурованими, напівструктурованими, графовими та потоковими. Розглянуто принципи розподілу функцій між СКБД доводять, що сучасні системи потребують спеціалізації: транзакційні операції оптимально виконуються реляційними СКБД, аналітика – колонковими, робота з документами – документоорієнтованими, а персоналізація – графовими.

Показано, що інтеграція даних є центральним елементом мультибазових систем, а успішне поєднання різних СКБД неможливе без узгоджених механізмів передачі, трансформації та синхронізації. Проаналізовано ключові підходи інтеграції: ETL та ELT забезпечують пакетну обробку і формування історичних зрізів; Change Data Capture гарантує оперативну узгодженість між системами; федеративні запити надають можливість доступу до даних без дублювання; потокові системи забезпечують мінімальну затримку для наскрізних потоків подій.

Систематизовано інструменти та платформи, що реалізують зазначені підходи. Показано, що Debezium у поєднанні з Kafka забезпечує найнижчу латентність та високу надійність синхронізації транзакційних даних; Trino надає універсальний механізм доступу до гетерогенних джерел; Databricks та Snowflake демонструють різні стратегії організації аналітичних екосистем – lakehouse та cloud-warehouse відповідно. Відкритий стек (Spark + Kafka + Airflow) забезпечує максимальну гнучкість ціною підвищеної операційної складності.

3 СИСТЕМНИЙ АНАЛІЗ АРХІТЕКТУРИ МУЛЬТИБАЗОВОЇ СИСТЕМИ

3.1 Формалізація вимог та обґрунтування вибору предметної області

Для репрезентативної демонстрації переваг використання мультибазових систем у проєктах Data Science предметною областю обрано електронну комерцію, а саме аналітичну інтернет платформу з підтримкою персоналізованих рекомендацій. Така система охоплює повний цикл взаємодії користувачів: перегляд і фільтрацію товарів, пошук, роботу з кошиком, оформлення замовлень, отримання рекомендацій, аналіз поведінкових подій та формування аналітичних звітів.

Вибір предметної області обумовлений кількома чинниками.

По-перше, системи електронної комерції працюють із широким спектром даних, що відрізняються структурою, обсягом та характером використання. У межах однієї платформи паралельно обробляються транзакційні записи (замовлення та оплати), напівструктуровані каталоги товарів із варіативними атрибутами, графові зв’язки між користувачами й товарами та великі потоки поведінкових подій, які надходять у режимі реального часу. Кожен тип даних має специфічні вимоги до зберігання, швидкості доступу та можливостей масштабування, що зумовлює необхідність використання спеціалізованих СКБД.

По-друге, значна частина операцій у системах електронної комерції виконується у реальному часі, що висуває підвищені вимоги до продуктивності. Пошук, фільтрація каталогу, формування рекомендацій та обробка замовлень повинні відбуватися з мінімальною затримкою, особливо під час періодів пікового навантаження, коли кількість звернень може істотно зростати. У таких умовах система має забезпечувати стабільну роботу підсистеми обробки замовлень, швидкий доступ до каталогу та актуальність поведінкових даних без зниження продуктивності.

По-третє, електронна комерція є однією з галузей, де методи Data Science застосовуються найбільш активно. Персоналізовані рекомендації, сегментація

користувачів, прогнозування попиту, аналіз подібності товарів та виявлення аномальних транзакцій потребують інтеграції різнорідних даних і виконання складних аналітичних операцій, що підсилює вимоги до архітектури та робить мультибазовий підхід доцільним.

У таблиці 3.1 наведено основні функціональні блоки платформи електронної комерції та їх характеристики з погляду обробки даних.

Таблиця 3.1 – Функціональні блоки платформи електронної комерції

Функціональний блок	Основні операції	Профіль навантаження	Критичні вимоги
Управління каталогом товарів	Пошук, фільтрація, оновлення атрибутів	70% читання 30% запису	Гнучка схема, швидка фільтрація, мінімальна затримка
Обробка замовлень та платежів	Створення замовлення, оплата, резервування	40% читання 60% запису	ACID, ізоляція транзакцій
Рекомендаційна система	Формування персональних рекомендацій	95% читання 5% запису	Обхід графу, низька затримка
Аналітика поведінки користувачів	Збір подій, агрегація метрик, побудова звітів	Потокове додавання	Висока пропускна здатність
Управління користувачами	Аутентифікація, профілі, історія	80% читання 20% запису	Узгодженість, доступність, захист даних

Аналіз особливостей предметної області дозволяє зробити висновок, що електронна комерція є репрезентативним прикладом для дослідження мультибазових систем. Її структура поєднує декілька різних моделей зберігання та опрацювання даних, кожна з яких потребує окремого спеціалізованого підходу

У системах електронної комерції формується кілька категорій даних, що істотно відрізняються за структурою, частотою оновлення, обсягами та характером операцій доступу. Така різноманітність створює складні умови для побудови універсального сховища даних, оскільки кожний тип даних потребує окремого підходу до організації зберігання та обробки.

Наведемо класифікацію основних груп даних, характерних для аналітичних платформ інтернет-магазинів та їхні ключові властивості.

1. Транзакційні дані.

До транзакційних даних належать записи про замовлення, статуси оплати, операції з кошиком та інші події, що потребують гарантованої коректності. Такі дані мають жорстку структуру, високу частоту оновлень та критичну залежність від узгодженості.

Основні характеристики:

– структура: чітко визначені таблиці, фіксовані типи полів;

– операції: часті записи та оновлення;

– ключові вимоги: повна підтримка ACID, цілісність, ізоляція транзакцій;

– типові сценарії: створення замовлення, зміна статусу оплати, управління кошиком.

Транзакційні дані належать до найбільш критичних, оскільки помилки при їх обробці можуть призвести до втрати замовлень або фінансових неточностей.

2. Дані каталогу товарів.

Каталог товарів містить інформацію про назви, описи, характеристики, категорії, медіафайли та цінові параметри. На відміну від транзакційних даних, структура каталогів є напівструктурованою і може відрізнятися між різними групами товарів.

Основні характеристики:

– структура: документа-подібна, з варіативними атрибутами;

– операції: переважно читання, періодичні оновлення атрибутів;

– ключові вимоги: гнучкість схеми, підтримка повнотекстового пошуку, швидка фільтрація;

– типові сценарії: пошук товарів, відбір за характеристиками, оновлення описів.

Завдяки відсутності жорсткої схеми дані каталогу товарів зручно зберігати в документоорієнтованих сховищах.

3. Рекомендаційні та пов’язані дані.

Рекомендаційні механізми використовують зв’язки між користувачами та товарами: перегляди, покупки, взаємодії, переходи за категоріями. Природним способом моделювання таких залежностей є графи.

Основні характеристики:

– структура: вузли (товари, користувачі) та ребра (взаємодії, подібність);

– операції: інтенсивне читання, обходи графу, пошук патернів;

– ключові вимоги: швидкість обробки складних запитів, можливість виконання графових алгоритмів;

– типові сценарії: пошук подібних товарів, рекомендації на основі схожої поведінки.

Графові БД забезпечують високу продуктивність при роботі з такими моделями завдяки оптимізованим структурам даних.

4. Поведінкові дані.

До поведінкових даних належать події, пов’язані з активністю користувачів: перегляди сторінок, кліки, пошукові запити, додавання товарів у кошик. Такі дані надходять у великих обсягах і з високою швидкістю.

Основні характеристики:

– структура: подієві записи, часові мітки, часто колонковий формат;

– операції: потокове додавання (append-only), періодичні аналітичні агрегати;

– ключові вимоги: висока пропускна здатність, ефективне зберігання великих обсягів, швидка аналітика;

– типові сценарії: побудова воронок, аналіз поведінкових патернів, вимірювання активності.

Такі дані доцільно розміщувати в аналітичних або time-series базах, оптимізованих для швидкої обробки великих масивів подій.

У таблиці 3.2 подано класифікацію основних типів даних разом із ключовими вимогами до їх зберігання.

Таблиця 3.2 – Класифікація основних типів даних у системах електронної комерції

Тип даних	Приклади сутностей	Структура	Частота змін	Основні вимоги
Транзакційні	Замовлення, оплати, статуси	Жорстка схема (таблиці)	Висока	ACID, ізоляція
Каталог товарів	Опис товарів, атрибути, категорії	Напівструктуровані документи	Середня	Гнучка схема, пошук
Рекомендаційні	Перегляди, покупки, подібність	Граф (вузли, ребра)	Середня/ висока	Швидкі графові запити
Поведінкові	Кліки, перегляди, пошукові запити	Події, часові ряди	Дуже висока	Пропускна здатність, аналітика

Аналіз типів даних свідчить про істотні відмінності в їх структурі та вимогах. Жодна окрема СКБД не здатна забезпечити оптимальну роботу з усіма видами даних одночасно, це формує підґрунтя для застосування мультибазового підходу, який дає змогу використовувати спеціалізовані сховища для кожної категорії даних і таким чином досягати високої продуктивності, масштабованості та гнучкості системи.

Функціональні вимоги (FR) визначають перелік операцій, які повинна підтримувати аналітична платформа електронної комерції забезпечення повного циклу взаємодії користувачів та виконання основних бізнес-процесів [32].

На основі аналізу предметної області сформульовано такі ключові вимоги:

FR1. Підтримка управління каталогом товарів.

Система повинна забезпечувати можливість пошуку, фільтрації та перегляду товарів із врахуванням численних варіативних атрибутів. Структура каталогу має підтримувати динамічне додавання нових характеристик без необхідності зміни загальної схеми даних.

FR2. Обробка замовлень та операцій оплати.

Платформа повинна забезпечувати коректне створення замовлень, оновлення статусів, здійснення оплати та резервування товарів. Операції мають виконуватися атомарно, з гарантією відсутності суперечливих або неконсистентних станів.

FR3. Формування персоналізованих рекомендацій.

Система повинна підтримувати механізми побудови персоналізованих рекомендацій на основі історії взаємодій, поведінкових подій та асоціативних зв’язків між товарами. Рекомендаційний модуль повинен працювати в режимі, що забезпечує актуальність результатів.

FR4. Збір та зберігання поведінкових подій.

Платформа повинна реєструвати кліки, перегляди товарів, пошукові запити та інші дії користувачів у режимі реального часу. Інформація має зберігатися у форматі, придатному для подальшого аналізу та побудови моделей машинного навчання.

FR5. Підтримка виконання аналітичних запитів.

Система повинна забезпечувати можливість виконання агрегованих, вибіркових та часових аналітичних операцій, які необхідні для аналізу поведінкових патернів, оцінювання ефективності маркетингових активностей та оптимізації асортименту.

FR6. Управління користувачами та їхньою історією взаємодії.

Платформа повинна підтримувати створення та оновлення профілів користувачів, автентифікацію, зберігання історії взаємодій та надання доступу до персоналізованої інформації.

FR7. Інтеграція з зовнішніми сервісами.

Система повинна забезпечувати можливість інтеграції з платіжними системами, сервісами доставки та зовнішніми аналітичними платформами для обміну даними та розширення функціональності.

Сформульовані функціональні вимоги окреслюють ключові процеси та можливості, які повинна підтримувати аналітична система електронної комерції. Їх виконання передбачає роботу з різнорідними наборами даних і виконує важливу роль у

подальшому визначенні архітектурних рішень. Наявність різних моделей доступу та обробки підтверджує доцільність застосування мультибазового підходу.

Нефункціональні вимоги (NFR) встановлюють якісні критерії роботи аналітичної платформи електронної комерції та визначають обмеження, у межах яких повинні виконуватися функціональні можливості системи [32].

NFR1. Продуктивність.

Система повинна забезпечувати низьку латентність під час виконання ключових операцій:

– пошук та фільтрація каталогу – не більше 200 мс (95-й перцентиль);
– формування персоналізованих рекомендацій – не більше 150 мс;
– обробка транзакцій – не більше 500 мс.

Такі значення є критичними для втримання користувачів та забезпечення конкурентоспроможності платформи.

NFR2. Масштабованість.

Платформа повинна підтримувати горизонтальне масштабування для обробки зростаючих обсягів даних та навантаження. Система має коректно функціонувати в умовах збільшення кількості запитів на 300–500% під час сезонних піків активності (розпродажі, промоакції).

NFR3. Доступність.

Критичні компоненти – транзакційний блок, каталог товарів, рекомендаційна система, повинні забезпечувати доступність на рівні не менше 99.9%, що відповідає не більше ніж 8.76 години недоступності на рік.

NFR4. Узгодженість даних.

Транзакційні операції повинні виконуватися з дотриманням ACID-властивостей. Для даних, що не є критичними (поведінкові події, логи), допускається модель eventual consistency, за умови, що це не впливає на аналітичні результати.

NFR5. Гнучкість моделі даних.

Система повинна дозволяти зміну структури атрибутів товарів, категорій та поведінкових параметрів без необхідності модифікації всієї схеми даних. Це забезпечує зменшення часу впровадження нових функціональних можливостей.

NFR6. Пропускна здатність системи збору подій.

Модуль реєстрації поведінкових подій має підтримувати приймання та зберігання від 100 тис. до 1 млн подій на годину з мінімальною затримкою між їх надходженням та доступністю для аналітики.

NFR7. Безпека та захист даних.

Платформа повинна забезпечувати безпечне зберігання персональних даних користувачів, захист транзакційних операцій та відповідність вимогам конфіденційності згідно з чинними міжнародними нормами (наприклад, рекомендаціями OWASP) [33].

У таблиці 3.3 представлено кількісні метрики нефункціональних вимог.

Таблиця 3.3 – Кількісні метрики нефункціональних вимог

Вимога	Метрика	Цільове значення	Критичне значення
Латентність каталогу	Час відповіді	< 200 мс	< 500 мс
Латентність транзакцій	Створення замовлення	< 500 мс	< 1000 мс
Латентність рекомендацій	Генерація рекомендацій	< 150 мс	< 300 мс
Доступність	Час роботи	99.9%	99.5%
Пропускну здатність подій	Подій на годину	1 000 000	100 000
Масштабованість	Зростання навантаження	+500%	+200%

Нефункціональні вимоги формують рамки, у яких повинна працювати аналітична платформа електронної комерції, забезпечуючи її надійність, гнучкість та ефективність. Високі вимоги до продуктивності, масштабованості, узгодженості та пропускної здатності підкреслюють складність предметної області та обумовлюють необхідність застосування мультибазового підходу, що дозволяє оптимізувати роботу різних компонентів системи відповідно до їх специфічних характеристик.

3.2 Проєктування архітектури мультибазової системи

Архітектура мультибазової системи для аналітичної платформи електронної комерції формується на основі результатів аналізу предметної області, функціональних та нефункціональних вимог. Такий підхід передбачає розподіл даних між кількома спеціалізованими СКБД, кожна з яких оптимізована для певного типу даних і моделі доступу.

Метою архітектури є забезпечення узгодженої взаємодії між компонентами системи, мінімальної затримки критично важливих операцій, високої доступності та можливості масштабування.

Мультибазова архітектура побудована за принципом Polyglot Persistence [16], який передбачає використання спеціалізованих СКБД для різних типів даних. Такий підхід забезпечує оптимальну продуктивність, узгодженість транзакцій, гнучкість моделі даних та здатність системи масштабуватися під час пікових навантажень [2].

Загальна концепція архітектури.

У межах мультибазової архітектури кожен компонент системи відповідає за певний тип даних та модель доступу:

1. PostgreSQL – транзакційні дані: замовлення, платежі, профілі користувачів з вимогами до ACID.

2. MongoDB – напівструктурований каталог товарів із варіативними атрибутами.

3. Neo4j – графові зв’язки між товарами, користувачами та їх взаємодіями, необхідні для рекомендаційних моделей.

4. ClickHouse – поведінкові події та часові ряди для високошвидкісної аналітики.

Розміщення різних категорій даних у спеціалізованих СКБД дозволяє уникнути компромісів, характерних для монолітних баз, та забезпечує ефективну обробку всіх типів навантаження.

Таблиця 3.4 – Розподіл даних між СКБД у мультибазовій архітектурі

СКБД	Типи даних	Основні сутності	Модель зберігання	Ключові переваги	Типові операції
PostgreSQL	Транзакційні	User, Order, OrderItem, Payment, Cart, Address	Реляційні таблиці	ACID, referential integrity, SQL-стандарт	INSERT, UPDATE, SELECT з JOIN
MongoDB	Напів структуровані	Product, Category, ProductAttribute	JSON-документи	Гнучка схема, full-text search, sharding	find(), aggregate(), text search
Neo4j	Графові зв'язки	User, Product, взаємодії (VIEWED, PURCHASED)	Граф (nodes, relationships)	Швидкий обхід, Cypher, pattern matching	MATCH, graph algorithms
ClickHouse	Часові ряди	Event, Click, PageView, SearchQuery	Колонкове сховище	Швидка агрегація, стиснення, append-only	SELECT з GROUP BY, time-based aggregates

Як видно з таблиці 3.4, кожна СКБД має чітко визначену область відповідальності. PostgreSQL забезпечує транзакційну цілісність, MongoDB – гнучкість схеми, Neo4j – ефективність графових операцій, а ClickHouse – швидкість аналітики.

Логічна структура архітектури та моделі даних.

Архітектура системи організована у вигляді кількох взаємопов’язаних шарів:

1. Шар збору та первинної обробки даних.

Відповідає за реєстрацію поведінкових подій (перегляди, кліки, пошук, додавання до кошика). Події надходять через API Gateway та передаються у ClickHouse для подальшої аналітики.

Використання MergeTree та партиціонування за місяцями забезпечує швидке виконання аналітичних запитів та зручне управління історичними даними [28]. Колонкове зберігання забезпечує високий коефіцієнт стиснення (10-40x) та швидкість агрегації [14].

Структура таблиці подій у ClickHouse наведена на рис 3.1.

```
CREATE TABLE events (
    event_id UUID,
    user_id UInt64,
    product_id UInt64,
    event_type Enum('click', 'view', 'add_to_cart', 'search'),
    event_timestamp DateTime,
    session_id String,
    metadata String
) ENGINE = MergeTree()
PARTITION BY toYYYYMM(event_timestamp)
ORDER BY (event_timestamp, user_id);
```

Рисунок 3.1 – Структура таблиці подій у ClickHouse

2. Транзакційний шар (PostgreSQL).

Опрацьовує замовлення, платежі, кошики та облікові записи користувачів, гарантує повну транзакційну узгодженість. ER-модель транзакційного шару подана на рис. 3.2, детальні описи сутностей у Додатку А.

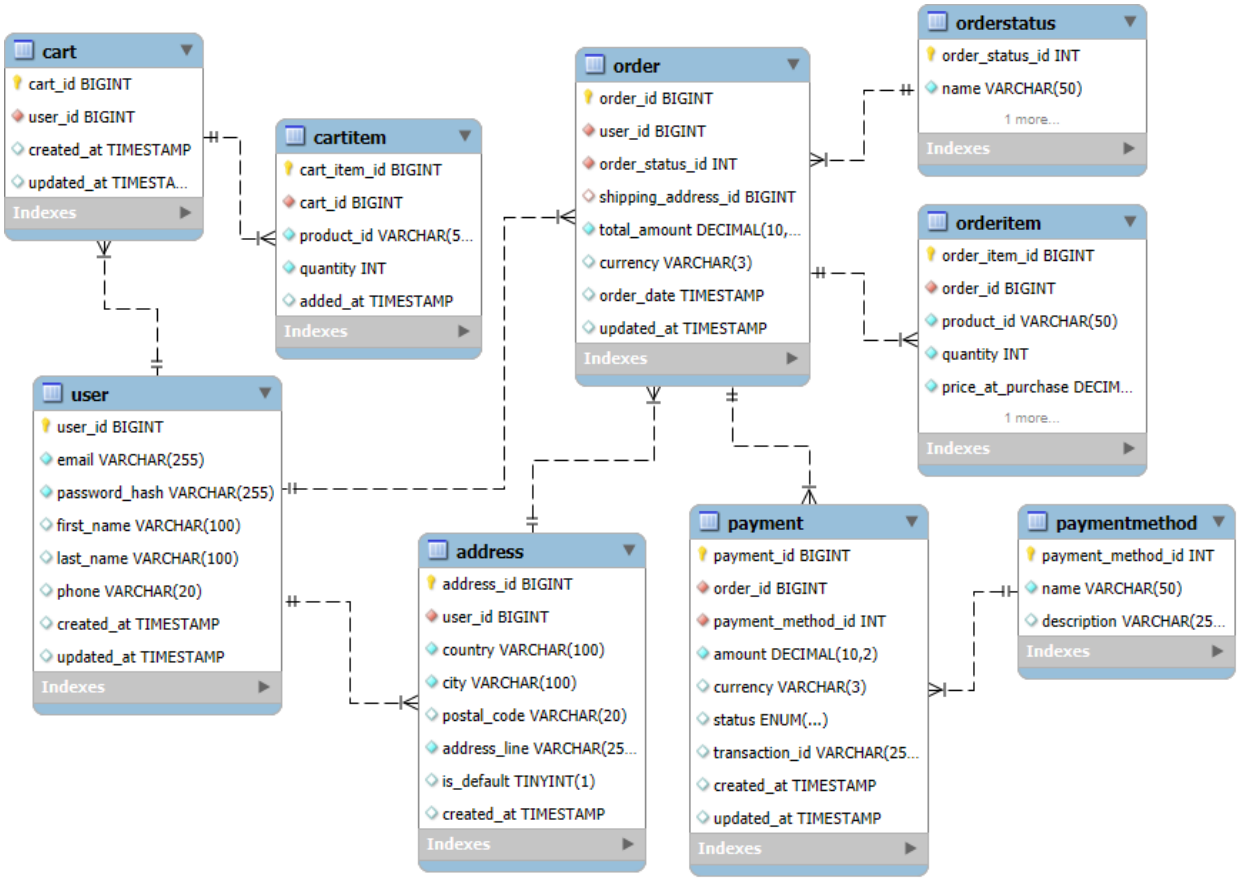


Рисунок 3.2 – ER-модель транзакційного шару (PostgreSQL)

Використання реляційної моделі для транзакційних даних забезпечує референтна цілісність через зовнішні ключі та дозволяє гарантувати узгодженість складних операцій за допомогою транзакцій [16]. Наприклад, створення замовлення вимагає виконання операцій: резервування товарів, списання коштів, оновлення статусу кошика, що забезпечується ACID-властивостями PostgreSQL [17].

3. Шар управління каталогом (MongoDB).

Зберігає дані про товари, категорії, атрибути, статуси та зображення. MongoDB забезпечує гнучку схему, необхідну для товарів з різною структурою атрибутів.

Приклад документа категорії «electronics» наведено на рис.3.3.

Рисунок 3.3 – Приклад документа товару категорії «electronics»

4. Рекомендаційний шар (Neo4j).

Містить графову модель взаємодій користувачів із товарами (таблиця 3.5) , включаючи зв’язки "переглянув", "придбав", "належить категорії", "подібний до".

Таблиця 3.5 - Графова модель взаємодій користувачів із товарами

Типи вузлів та зв’язків	Властивості
Основні типи вузлів (nodes):	
User (користувач):	Властивості: user_id, segment (new, regular, vip), registration_date
Product (товар):	Властивості: product_id, name, category, price
Category (категорія):	Властивості: category_id, name, parent_category_id
Основні типи зв'язків (relationships):	
VIEWED (переглянув): Від: User → Product	Властивості: viewed_at (timestamp), duration (seconds)
PURCHASED (придбав): Від: User → Product	Властивості: purchased_at, quantity, price_paid
SIMILAR_TO (подібний до): Від: Product → Product	Властивості: similarity_score (0.0-1.0), algorithm (collaborative, content-based)
IN_CATEGORY(належить категорії): Від: Product → Category	
ALSO_BOUGHT (також купували): Від: Product → Product	Властивості: co_occurrence_count, confidence_score

Типи вузлів: User, Product, Category

Типи зв’язків: VIEWED, PURCHASED, SIMILAR_TO, IN_CATEGORY, ALSO_BOUGHT.

Графова модель дозволяє виконувати складні запити з обходом графу на 2-3 порядки швидше, ніж еквівалентні JOIN-операції в реляційних базах [3]. За даними benchmarks Neo4j [5], запити типу "знайти рекомендації на основі друзів друзів" виконуються за 10-50 мс для графів з мільйонами вузлів, тоді як у PostgreSQL аналогічні запити потребують секунд.

5. Аналітичний шар (ClickHouse).

Призначений для: обробки потоків подій, часової аналітики,формування показників ефективності та звітів.

Приклади аналітичних запитів: обчислення конверсійної воронки (рис. 3.4) та аналіз популярності товарів за часовими інтервалами (рис. 3.5)

```
SELECT
    event_type,
    COUNT(DISTINCT user_id) AS unique_users,
    COUNT(*) AS total_events
FROM events
WHERE event_timestamp >= now() - INTERVAL 7 DAY
GROUP BY event_type
ORDER BY
    CASE event_type
        WHEN 'view' THEN 1
        WHEN 'click' THEN 2
        WHEN 'add_to_cart' THEN 3
        WHEN 'purchase' THEN 4
    END;
```

Рисунок 3.4 – Обчислення конверсійної воронки

```
SELECT
    product_id,
    toStartOfHour(event_timestamp) AS hour,
    COUNT(*) AS views
FROM events
WHERE event_type = 'view'
  AND event_timestamp >= now() - INTERVAL 24 HOUR
GROUP BY product_id, hour
ORDER BY hour DESC, views DESC
LIMIT 100;
```

Рисунок 3.5 – Аналіз популярності товарів за часовими інтервалами

ClickHouse забезпечує виконання таких запитів з латентністю 50-200 мс навіть для таблиць з мільярдами записів [25].

Система використовує комбінований підхід до інтеграції, комбінує синхронні та асинхронні механізми взаємодії, що забезпечує гнучкість взаємодії між компонентами та підтримку різних моделей узгодженості даних.

1. Синхронна взаємодія через API Gateway.

Забезпечує обробку CRUD-операцій і транзакцій у режимі реального часу [26].

Основні сценарії включають:

– оформлення замовлення з гарантіями ACID,

– перегляд каталогу товарів з фільтрацією,

– отримання персоналізованих рекомендацій.

API Gateway виконує функції маршрутизації запитів до відповідних мікросервісів, автентифікації, обмеження та агрегації відповідей від різних джерел [27].

2. Асинхронний обмін через подієвий стрім.

Для високочастотних подій використовується подійна шина (event bus), яка забезпечує:

– передачу подій у ClickHouse для аналітики;

– оновлення графових зв’язків у Neo4j;

– можливість масштабування потоків обробки подій.

Основні сценарії:

– Product Updates (оновлення каталогу): MongoDB → Kafka → Neo4j

– User Interactions (взаємодії користувачів): API → Kafka → ClickHouse + Neo4j

– Order Completed (завершення замовлення): PostgreSQL → Kafka → Neo4j

Асинхронна взаємодія дозволяє масштабувати навантаження та зберігати eventual consistency там, де це не критично.

3. Federated Queries (федеративні запити) для міжсистемної аналітики.

Для складних аналітичних звітів, що потребують даних з кількох СКБД, застосовується механізм федеративних запитів на основі Presto/Trino [], який дозволяє виконувати SQL-запити одночасно до різних СКБД (PostgreSQL, MongoDB, ClickHouse) без ETL-процесів.

Архітектура побудована так, щоб різні модулі не створювали конкуренції за ресурси. PostgreSQL обслуговує OLTP-операції, MongoDB працює з каталогом, Neo4j обробляє рекомендаційні запити, а ClickHouse виконує важкі аналітичні операції. Такий

розподіл дозволяє зберігати стабільну продуктивність системи навіть під час пікових навантажень [30].

Запропонована мультибазова архітектура забезпечує оптимальний розподіл даних між спеціалізованими СКБД відповідно до їх призначення, що дозволяє досягти високої продуктивності, масштабованості та гнучкості.

Архітектурні рішення підтримують як strong consistency для транзакцій, так і eventual consistency для аналітики та поведінкових даних.

3.3 Аналітичне дослідження ефективності мультибазової архітектури

Комплексне аналітичне оцінювання ефективності запропонованої мультибазової архітектури поєднує реляційні, документоорієнтовані, графові та колонкові СКБД, виконано з метою визначення відповідності архітектурного рішення функціональним і нефункціональним вимогам платформи електронної комерції. Аналіз охоплює методику дослідження, характеристику сценаріїв навантаження, порівняння монолітного та мультибазового підходів, аналіз альтернативних архітектур і виявлення потенційних вузьких місць. Оцінювання має аналітичний характер і базується на моделях даних, технічній документації СКБД, наукових публікаціях та промислових кейсах (Netflix, Uber, Alibaba, Shopify).

1. Методика дослідження.

Аналітичне оцінювання виконано у три взаємопов’язані етапи:

Етап 1. Аналіз відповідності СКБД типам даних та операцій.

Для кожної категорії даних оцінено:

- тип моделі даних (реляційна, документоорієнтована, графова, колонкова);
- характер оновлень і запитів;
- вимоги до узгодженості (ACID / eventual consistency);
- типові операції (OLTP, OLAP, graph traversal);
- підтримку індексів механізмів масштабування.

Мета етапу – обґрунтувати відповідність PostgreSQL, MongoDB, Neo4j та ClickHouse своїм підмножинам даних: транзакції, каталог, рекомендації, поведінкові події.

Етап 2. Аналіз продуктивності для різних профілів навантаження.

Оцінювання здійснено для ключових сценаріїв електронної комерції: пошук товарів, транзакції, рекомендаційні моделі, потокова аналітика.

Для кожного сценарію визначено:

– латентність операцій;
– пропускна здатність;
– масштабованість (горизонтальна/вертикальна);
– обчислювальну складність;
– вплив паралельності та обсягу даних.

Порівняння виконано для двох підходів: монолітний (усі дані в PostgreSQL) та мультибазовий (PostgreSQL + MongoDB + Neo4j + ClickHouse).

Етап 3. Аналіз узгодженості та ізоляції навантаження.

Оцінено, чи забезпечує архітектура :

– strong consistency для транзакційних операцій;
– eventual consistency для каталогу, подій та аналітики;
– відсутність конкуренції за ресурси між модулями;
– ізоляцію OLTP і OLAP навантаження;
– стійкість до пікових навантажень.

2. Сценарії навантаження та їх характеристика

Для оцінювання ефективності архітектури визначено сім характерних сценаріїв (S1–S7), що охоплюють усі основні моделі роботи платформи електронної комерції: транзакційні операції, пошук і фільтрацію, графові рекомендації, поведінкову аналітику та попередню агрегацію подій.

Сценарій S1 – Пошук і фільтрація товарів.

Опис: Користувач здійснює пошук товарів за ключовими словами та фільтрує результати за категорією, ціною, брендом та іншими атрибутами.

Задіяні СКБД: MongoDB (каталог товарів).

Типові операції: повнотекстовий пошук за назвою та описом, фільтрація за compound index, сортовані результатів вибірки, пагінація (limit/skip).

Складність: O(log n) для індексованих полів, де n – кількість товарів у категорії.

Обґрунтування очікуваної складності. MongoDB використовує B-tree індекси [1]

Пошук у B-tree має складність O(log n), де n – кількість документів у колекції, log n – кількість рівнів дерева (глибина).

Для 1 000 000 товарів $\log_2(1000000) \approx 20$ операцій порівняння.

Compound index (category, price) дозволяє:

1. Знайти category за O(log n) [30]

2. В межах category відфільтрувати за price за O(log m), де $m << n$

Сценарій S2 – Перегляд товару.

Опис: Користувач переглядає детальну інформацію про обраний товар, включаючи характеристики, відгуки та наявність на складі.

Задіяні СКБД: MongoDB (атрибути товару), PostgreSQL (наявність на складі).

Типові операції:

– MongoDB: `findOne() за product_id`

– PostgreSQL: `SELECT inventory WHERE product_id = ?`

Складність: O(1) для обох операцій при наявності індексів на product_id.

Сценарій S3 – Створення замовлення.

Опис: Користувач оформлює замовлення з кількома товарами, вказує адресу доставки та здійснює оплату.

Задіяні СКБД: PostgreSQL (транзакційні дані), Kafka (події).

Характеристика: високі вимоги до узгодженості.

Типові операції: створення замовлення, списання коштів, оновлення статусів, очищення кошика.

Складність: O(k), де k – кількість товарів у замовленні. Транзакція забезпечує ACID-властивості.

Сценарій S4 – Персоналізовані рекомендації.

Опис: Система генерує список рекомендованих товарів для користувача на основі його історії переглядів та покупок, а також поведінки схожих користувачів.

Задіяні СКБД: Neo4j (граф рекомендацій).

Вимоги: відповідь до 50–150 мс.

Типові операції: пошук схожих товарів (рис. 3.6), виявлення патернів поведінки, патерн-матчинг, обхід графу (алгоритми – PageRank, Community Detection).

```
MATCH (u:User {user_id: $userId})-[:PURCHASED]->(p:Product)
      <-[:PURCHASED]-(similar:User)
MATCH (similar)-[:PURCHASED]->(rec:Product)
WHERE NOT (u)-[:PURCHASED]->(rec)
RETURN rec.name, COUNT(similar) AS score
ORDER BY score DESC LIMIT 10
```

Рисунок 3.6 – Запит на пошук товарів (фільтрація)

Складність: O(d × m), де d – середній ступінь вузла (degree), m – кількість схожих користувачів. Для графів Neo4j типова латентність 10-50 мс [2].

Сценарій S5 – Аналіз конверсійної воронки.

Опис: Аналітик формує звіт про конверсію користувачів на різних етапах: перегляд → додавання до кошика → оформлення замовлення → оплата.

Задіяні СКБД: ClickHouse (поведінкові події)

Обсяг подій: 10–100 млн подій на добу.

Вимоги: латентність аналітичних запитів < 500 мс.

Типові операції: агрегації, часові зрізи, побудова метрик.

Складність: O(n/p), де n – кількість подій, p – кількість партицій. Колонкове зберігання забезпечує швидку агрегацію навіть для мільярдів записів [3].

Сценарій S6 – Визначення товарів, що купують разом.

Опис: Система визначає товари, які користувачі часто купують разом з обраним товаром (cross-sell).

Задіяні СКБД: Neo4j (зв'язки ALSO_BOUGHT)

Вимоги: обхід графу 5–20 мс.

Операції: попередньо обчислені ребра ALSO_BOUGHT (рис. 3.7).

```
MATCH (p:Product {product_id: $productId})-[r:ALSO_BOUGHT]->(related:Product)
RETURN related.name, r.co_occurrence_count, r.confidence_score
ORDER BY r.confidence_score DESC LIMIT 5
```

Рисунок 3.7 – Обчислення ребра ALSO_BOUGHT

Складність: O(d), де d – кількість вихідних ребер ALSO_BOUGHT з вузла товару. Типово $d < 20$ для електронної комерції [4].

Сценарій S7 – Аналіз популярності товарів за часовими періодами.

Опис: Аналітик формує звіт про топ-10 найпопулярніших товарів за останній тиждень з розбивкою по днях та годинах.

Задіяні СКБД: ClickHouse (часові ряди переглядів)

Типові операції: агрегація, часова фільтрація, сортування за кількістю переглядів.

Вимоги: часові агрегації < 0.8 с.

Складність: $O(n/p \times \log k)$, де n – кількість подій, p – кількість партицій, k – розмір топ-списку.

Узагальнена таблиця відповідності сценаріїв використання мультибазової системи наведена у таблиці 3.6.

Як видно з таблиці 3.6, різні сценарії мають різні профілі навантаження та вимоги до латентності. Сценарії S1, S2, S4, S6 є read-heavy, S3 – write-heavy, а S5, S7 – аналітичні read-only запити.

Таблиця 3.6 – Типові сценарії використання мультибазової системи

Сценарій	Опис	Задіяні СКБД	Тип навантаження	Критичні вимоги
S1	Пошук та фільтрація товарів	MongoDB	Read-heavy (95%)	Латентність < 200 ms, full-text search
S2	Перегляд деталей товару	MongoDB, PostgreSQL	Read-heavy (99%)	Латентність < 100 ms, узгодженість наявності
S3	Створення замовлення	PostgreSQL, Kafka	Write-heavy (80%)	ACID, латентність < 500 ms, ізоляція
S4	Персоналізовані рекомендації	Neo4j	Read-heavy (100%)	Латентність < 150 ms, точність рекомендацій
S5	Аналіз конверсійної воронки	ClickHouse	Read-only (100%)	Агрегація мільярдів записів, < 1 s
S6	Товари, що купують разом	Neo4j	Read-heavy (100%)	Латентність < 100 ms, релевантність
S7	Популярність товарів за періодами	ClickHouse	Read-only (100%)	Часова агрегація, < 500 ms

Застосовування різних сценаріїв підтверджує доцільність використання спеціалізованих СКБД для різних типів операцій.

3.3.3. Аналіз продуктивності за сценаріями

Порівняння проведено аналітичним шляхом, без експериментальної реалізації системи. Метрики отримано з наступних джерел:

– офіційна технічна документація СКБД (PostgreSQL, MongoDB, Neo4j, ClickHouse), яка містить типові характеристики продуктивності для різних типів операцій.

– опубліковані критерії від виробників (MongoDB Performance, Neo4j Benchmarks ClickHouse Performance) та незалежних організацій (TPC-H, YCSB) [30].

– наукові дослідження продуктивності баз даних (SIGMOD, VLDB, ICDE) [31, 17].

– кейси впровадження мультибазових архітектур у компаніях (Netflix, Uber, Alibaba).

Метрики наведено як діапазони, що відображає варіативність продуктивності залежно від розміру даних, конфігурації апаратного забезпечення та специфіки запитів. Для кожного сценарію наведено посилання на джерела, що підтверджують оцінки.

Оцінювання виконано за наступними метриками:

– латентність (Latency) – час відповіді на запит для 95-го перцентиля [5]

– пропускна здатність (Throughput) – кількість операцій на секунду (ops/sec або RPS)

– масштабованість (Scalability) – здатність підтримувати продуктивність при зростанні навантаження

– складність операцій – обчислювальна складність у нотації Big O

Основні результати за сценаріями наведено у таблиці 3.7.

Таблиця 3.7 – Порівняльний аналіз продуктивності за сценаріями

Сценарій	Монолітний підхід (PostgreSQL)	Мультибазовий підхід	Покращення
S1: Пошук товарів	300-800 мс	100-200 мс (MongoDB)	2-4x
S2: Деталі товару	50-100 мс	50-100 мс (MongoDB + PostgreSQL)	~1x
S3: Створення замовлення	50-80 мс	50-80 мс (PostgreSQL) + async events	~1x + async
S4: Рекомендації	5,000-30,000 мс	10-50 мс (Neo4j)	100-600x
S5: Конверсійна воронка	10,000-60,000 мс	100-500 мс (ClickHouse)	20-100x
S6: Cross-sell	500-3,000 мс	5-20 мс (Neo4j)	50-200x
S7: Популярність за періодами	5,000-30,000 мс	200-800 мс (ClickHouse)	10-50x

Основні висновки з аналізу продуктивності: MongoDB забезпечує 2–4× швидший пошук і фільтрацію порівняно з PostgreSQL (S1). PostgreSQL ефективний для транзакцій, мультибаза додає асинхронність без втрати продуктивності (S3). Neo4j забезпечує 100–600× прискорення рекомендацій порівняно з JOIN-операціями (S4). ClickHouse

прискорює аналітичні обчислення у 20–100× (S5, S7). Графові зв'язки також обчислюються на два порядки швидше (S6).

4. Порівняльний аналіз архітектурних альтернатив

Для обґрунтування вибору здійснено порівняльний аналіз чотирьох архітектурних рішень:

– монолітна реляційна архітектура (PostgreSQL)

– монолітна документоорієнтована архітектура (MongoDB)

– Data Lake архітектура (S3 + Apache Spark)

– мультибазова архітектура (Polyglot Persistence) – запропоноване рішення

Для кожного проведено оцінювання: функціональних можливостей, продуктивності, масштабованості, узгодженості, придатність для електронної комерції, операційних витрат.

Результати зведено у таблиці 3.8.

Таблиця 3.8 - Результати оцінювання

Критерій	Вага	PostgreSQL	MongoDB	Data Lake	Мультибазова
Функціональні критерії					
Транзакційна узгодженість (ACID)	0.15	5	3	2	5
Підтримка варіативних атрибутів	0.10	2	5	4	5
Ефективність графових запитів	0.15	1	2	2	5
Аналітична продуктивність (OLAP)	0.15	2	2	4	5
Повнотекстовий пошук	0.05	3	5	3	5
Нефункціональні критерії					
Продуктивність (low latency)	0.15	3	4	1	5
Масштабованість	0.10	2	4	5	5
Доступність	0.05	4	4	4	4
Складність впровадження	0.05	5	4	2	2
Операційні витрати	0.05	4	4	3	3
Зважена оцінка	1.00	2.85	3.50	2.95	4.75

Виконано багатокритеріальне оцінювання за функціональними та нефункціональними вимогами. За результатами зваженої моделі: мультибазова архітектура (4.75) отримала найвищу оцінку; найближчий конкурент – MongoDB-моноліт (3.50); Data Lake обмежений високою латентністю; монолітний PostgreSQL недостатньо ефективний для графових та аналітичних сценаріїв.

5. Виявлення потенційних вузьких місць та рекомендації

На основі проведеного аналізу виявлено п’ять критичних вузьких місць мультибазової архітектури:

1. Синхронізація даних між СКБД (Kafka, eventual consistency)

– ризики неузгодженості, затримки 100–500 мс.

– рекомендації: monitoring lag, збільшення партицій Kafka, hybrid consistency.

2. Масштабування Neo4j при зростанні графу

– складність O(d^k), ризик зростання латентності до 2 с.

– рекомендації: індекси, кешування, GDS-попередні обчислення, partitioning.

3. Масштабування ClickHouse при великих обсягах подій

– ризик зростання латентності до 10 секунд.

– рекомендації: TTL, tiered storage, MV, sampling.

4. API Gateway як Single Point of Failure (Єдина точка відмови)

– при падінні gateway зупиняється вся система.

– рекомендації: active-active, rate limiting, circuit breaker.

5. Взаємодія PostgreSQL з іншими СКБД

– ризик неузгодженості inventory/order.

– рекомендації: Saga pattern, idempotency, джерела подій.

Сформовано стратегії масштабування кожного компонента (таблиця 3.9), де визначено поточні межі продуктивності, рекомендовані механізми оптимізації та очікувані результати після покращень.

Таблиця 3.9 – Стратегії масштабування компонентів мультибазової системи

Компонент	Поточна межа масштабованості	Стратегія масштабування	Очікувана межа після оптимізації
PostgreSQL	10K TPS 500 GB	Read replicas (3-5) Connection pooling (PgBouncer) Partitioning за датою	50K TPS 5 TB
MongoDB	100K ops/s 1 TB	Sharding за category (10+ shards) Secondary reads	500K ops/s 50 TB
Neo4j	5K queries/s 10M nodes	Caching (Redis) Graph partitioning Fabric federation	20K queries/s 100M nodes
ClickHouse	1M rows/s insert 10B rows	Distributed tables (5+ nodes) Tiered storage Materialized Views	10M rows/s 100B rows
Kafka	100K events/s	Increase partitions (24+) Add brokers (5+) Compression	1M events/s
API Gateway	20K RPS	Horizontal scaling (5+ instances) ALB Caching	100K RPS

Проведений аналіз демонструє, що мультибазова архітектура є оптимальним рішенням для платформи електронної комерції з інтенсивною обробкою різнорідних даних. Вона забезпечує суттєве прискорення виконання критичних операцій (у 10–600 разів для графових і аналітичних сценаріїв), можливість масштабування кожного компонента незалежно та ізоляцію навантаження між транзакційними, графовими й аналітичними процесами. Порівняльна оцінка архітектурних альтернатив підтвердила перевагу мультибазового підходу (4.75 бала), тоді як виявлені вузькі місця можуть бути усунені за допомогою визначених стратегій масштабування та оптимізації. Таким чином, дослідження обґрунтовує доцільність впровадження Polyglot Persistence у системах електронної комерції середнього та великого масштабу.

Висновки до розділу 3

У третьому розділі проведено системний аналіз архітектури мультибазової системи для аналітичної платформи електронної комерції, який охоплює формалізацію

вимог, проєктування архітектури та аналітичне оцінювання її ефективності. На основі аналізу предметної області встановлено, що платформи електронної комерції працюють із різнорідними наборами даних, які істотно відрізняються за структурою, частотою оновлення та вимогами до узгодженості й продуктивності, що визначило доцільність використання підходу Polyglot Persistence та спеціалізованих СКБД для кожного типу даних.

Сформульовані функціональні та нефункціональні вимоги засвідчили високі потреби платформи щодо продуктивності, масштабованості, доступності, гнучкості схеми та пропускної здатності модулів потокової обробки подій. На основі цих вимог побудовано архітектуру мультибазової системи, у якій PostgreSQL відповідає за транзакційний контур, MongoDB за каталог товарів, Neo4j за рекомендаційні графи, а ClickHouse за високошвидкісну аналітику поведінкових подій.

Аналітичне дослідження показало, що мультибазовий підхід суттєво перевершує монолітний реляційний варіант у сценаріях пошуку, фільтрації, рекомендацій та аналітики. Порівняльний аналіз архітектурних альтернатив показав, що мультибазова архітектура отримала найвищу зважену оцінку, випередивши монолітні рішення на основі PostgreSQL чи MongoDB та Data Lake-підхід, які не здатні забезпечити низьку латентність для широкого спектра навантажень.

Загалом результати дослідження підтвердили, що мультибазова архітектура є найбільш обґрунтованим рішенням для аналітичної платформи електронної комерції. Вона забезпечує високу продуктивність, ізоляцію навантаження, гнучкість структури даних та масштабованість, необхідні для систем із великими обсягами різнорідної інформації та складними аналітичними процесами.

ВИСНОВКИ

Кваліфікаційна робота присвячена системному аналізу побудови мультибазових систем у Data Science проєктах.

В ході дослідження отримані наступні результати:

1. Доведено актуальність застосування мультибазових систем у Data Science. Показано, що сучасні аналітичні системи працюють із великими обсягами структурованих, напівструктурованих і неструктурованих даних, що мають різні вимоги до затримки, узгодженості та продуктивності. Жодна окрема СКБД не може забезпечити ефективну обробку всіх типів навантажень, що робить застосування концепції Polyglot Persistence обґрунтованою необхідністю.

2. Проаналізовано класи моделей баз даних та їх роль у мультибазових архітектурах. Науково обґрунтовано, що реляційні, документні, графові, колонкові, key-value та time-series СКБД виконують різні функції в єдиній системі: від транзакційної обробки до аналітики, роботи з гнучкими структурами та зберігання даних високої частотності. Це підтверджує їх взаємодоповнюваність, а не взаємозамінність.

3. Визначено ключові вимоги до організації та обробки даних у Data Science-проєктах. Проведений аналіз показав, що ефективність мультибазових систем визначають вимоги до масштабованості, латентності, пропускної здатності, гарантованої доступності та різних моделей узгодженості, що впливають на вибір СКБД і архітектурних рішень.

4. Порівняно архітектурні моделі зберігання даних (Data Warehouse, Data Lake, Lakehouse). Показано, що класичні DWH оптимальні для бізнес-аналітики, Data Lake – для масштабованої обробки raw-даних, а Lakehouse моделі поєднують їх переваги, забезпечуючи ACID-транзакції та масштабовану роботу з напівструктурованими даними. Це формує базис для мультибазових рішень у складних аналітичних системах.

5. Проаналізовано підходи до інтеграції даних і встановлено їхнє призначення. Показано, що ETL, ELT, CDC, потокова інтеграція, федеративні запити та віртуалізація

даних розв’язують різні задачі. Жоден підхід не є універсальним: ETL забезпечує стабільність даних, ELT – гнучкість, CDC – мінімальну латентність, streaming – реактивність, federated queries – оперативний доступ без дублювання.

6. Виконано системне порівняння інструментів і платформ реалізації мультибазових систем. Доведено, що Debezium і Kafka забезпечують найвищу продуктивність і надійність синхронізації; Trino – уніфікований доступ до гетерогенних СКБД; Databricks і Snowflake – фундамент для аналітичних і ML-орієнтованих рішень. Комбінація Spark, Kafka, Airflow та спеціалізованих СКБД забезпечує максимальну гнучкість.

7. Розроблено архітектурну модель мультибазової системи для вибраної предметної області. У роботі сформовано вимоги, обґрунтовано вибір СКБД, побудовано архітектурну модель взаємодії компонентів, визначено інтеграційний шар, структуру потоків даних та способи синхронізації. Це дозволяє практично застосовувати результати роботи для розробки подібних систем.

8. Проведено аналітичне дослідження ефективності мультибазової архітектури. Показано, що поєднання реляційних, документних, графових та колонкових СКБД забезпечує оптимальну продуктивність при значно нижчій латентності та кращому масштабуванні порівняно з монобазовими рішеннями. Сформовано висновки щодо вибору оптимального технологічного стеку.

Отримані результати підтверджують, що мультибазові системи є оптимальним рішенням для сучасних Data Science-проєктів, які працюють із різнорідними даними та потребують високу продуктивність як транзакційних, так і аналітичних операцій. Сформована методика проєктування та проведений системний аналіз можуть бути використані як основа для практичної реалізації масштабованих і надійних мультибазових архітектур.

ПЕРЕЛІК ПОСИЛАНЬ

1. Jiaheng Lu and Irena Holubová. Multi-model Databases: A New Journey to Handle the Variety of Data. *ACM Comput. Surv*. 2020, Vol. 52 (3), 38 p. https://doi.org/10.1145/3323214

2. Mihai, G. Multi-Model Database Systems: The State of Affairs. *Economics and Applied Informatics*, 2020, Vol. 26, pp. 211-215. https://doi.org/10.35219/eai15840409128.

3. Lu J. Holistic evaluation in multi-model databases benchmarking. *Distributed and Parallel Databases*, 2019, Vol. 39, pp. 1 - 33. https://doi.org/10.1007/s10619-019-07279-6.

4. Toorpu, A. R. Unified Data Access in Multi-Model Databases: Querying Relational, Graph, and Document Data Seamlessly. *International Journal of Global Innovations and Solutions*, 2025, Vol. *2*(3). https://doi.org/10.63412/4qcgbv22

5. Tankoano, J. Multi-model Database Design and Query Processing in NewSQL DBMSs. *American Journal of Computer Science and Technology*, 2025, Vol. 8(2), pp. 52-71. https://doi.org/10.11648/j.ajcst.20250802.12

6. Zhang C., Lu J. Holistic evaluation in multi-model databases benchmarking. *Distrib Parallel Databases*. 2021, Vol. 39, pp. 1–33. https://doi.org/10.1007/s10619-019-07279-6

7. George, G., Osinga, E., Lavie, D., & Scott, B. Big Data and Data Science Methods for Management Research. *Academy of Management Journal*, 2016, Vol. 59, pp. 1493-1507. https://doi.org/10.5465/amj.2016.4005.

8. Bilokon, P., Bilokon, O., & Amen, S., 2023. A compendium of data sources for data science, machine learning, and artificial intelligence. *ArXiv*, abs/2309.05682. https://doi.org/10.2139/ssrn.4567555.

9. Galkaduwa C., Ranasinghe N. Data Science and Its Importance. *Biomedical Science and Clinical Research*. 2024. h ttps://doi.org/10.33140/bscr.03.01.06.

10. Li, K., Griffin, M., Barker, T., Prickett, Z., Hodkiewicz, M., Kozman, J., & Chirgwin,P. Embedding data science innovations in organizations: a new workflow approach. *Data-Centric Engineering*, 2023, 4. https://doi.org/10.1017/dce.2023.22.

11. Grossi, V., Trasarti, R., & Dazzi, P. Data Science Workflows for the Cloud/Edge Computing Continuum. *Proceedings of the 1st Workshop on Flexible Resource and Application Management on the Edge*. 2020, https://doi.org/10.1145/3452369.3463820.

12. Cravero, A., Pardo, S., Galeas, P., López Fenner, J., & Caniupán, M. Data Type and Data Sources for Agricultural *Big Data and Machine Learning. Sustainability*, 2022, 14(23), 16131. https://doi.org/10.3390/su142316131

13. Omar Lajam and Salahadin Mohammed. Revisiting Polyglot Persistence: From Principles to Practice. *International Journal of Advanced Computer Science and Applications (IJACSA),* 2022 Vol(13.5). http://dx.doi.org/10.14569/IJACSA.2022.0130599

14. Groppe, S. and Groppe, J. (2020). Hybrid Multi-model Multi-platform (HM3P) Databases. In Proceedings of the 9th International Conference on Data Science, *Technology and Applications - DATA*; 2020, pp. 177-184. DOI: 10.5220/0009802401770184

15. Kazanavičius, J., Mažeika, D., & Kalibatienė, D.,. An Approach to Migrate a Monolith Database into Multi-Model Polyglot Persistence Based on Microservice Architecture: A Case Study for Mainframe Database. *Applied Sciences*. 2022. https://doi.org/10.3390/app12126189.

16. Lajam, O., & Mohammed, S., 2022. Revisiting Polyglot Persistence: From Principles to Practice. *International Journal of Advanced Computer Science and Applications*. https://doi.org/10.14569/ijacsa.2022.0130599.

17. ETL vs. ELT: Optimizing Data Integration for Retail and Insurance Analytics. Journal of Computational Intelligence and Robotics. https://www.thesciencebrigade.org/jcir/article/view/297

18. What is change data capture (CDC)? What is change data capture (CDC)? - SQL Server | Microsoft Learn

19. Sima, A., Farias, T., Zbinden, E., Anisimova, M., Gil, M., Stockinger, H., Stockinger, K., Robinson-Rechavi, M., & Dessimoz, C. Enabling semantic queries across federated bioinformatics databases. Database: *The Journal of Biological Databases and Curation*, 2019. https://doi.org/10.1101/686600.

20. Schwarte, A., Haase, P., Hose, K., Schenkel, R., & Schmidt, M. FedX: A Federation Layer for Distributed Query. *Processing on Linked Open Data*, 481-486. https://doi.org/10.1007/978-3-642-21064-8_39.

21. Debezium. https://debezium.io/

22. Airbyte. The Open Standard for Data Movement https://airbyte.com/

23. Fivetran. https://www.fivetran.com/

24. Apache Kafka . https://kafka.apache.org/

25. Apache Pulsar. https://pulsar.apache.org/

26. Redpanda. https://www.redpanda.com/

27. Databricks. https://www.databricks.com/product/data-lakehouse

28. Sarkar, S., Staratzis, D., Zhu, Z., & Athanassoulis, M.. Constructing and Analyzing the LSM Compaction Design Space. *Proc. VLDB Endow*., 2021, 14, 2216-2229. https://doi.org/10.14778/3476249.3476274.

29. MongoDB Documentation. Indexes. https://docs.mongodb.com/manual/indexes/.

30. M. Barata, J. Bernardino and P. Furtado, YCSB and TPC-H: Big Data and Decision Support Benchmarks, 2014 *IEEE International Congress on Big Data*, Anchorage, AK, USA, 2014, pp. 800-801, doi: 10.1109/BigData.Congress.2014.128.

31. SIGMOD. https://sigmod.org/

32. ISO/IEC/IEEE 29148:2018. Systemc and software engineering – Requirements engineering. https://www.iso.org/standard/72089.html

33. Open Web Application Security Project (OWASP) . https://owasp.org/

34. Лобода Ю. Г., Кричун О. С. Інтелектуальні агенти як засіб адаптивного керування конвеєрами даних у середовищах Big Data. *Інформаційні технології: моделі, алгоритми, системи (ITMAS – 2025).* VI Міжнародна науково-практична інтернет конференція (16-17 листопада 2025 р. Україна, Миколаїв). Миколаїв: НУК ім. адмірала Макарова. 2025. С. 456-458. https://itconf.nuos.edu.ua/2025/proceedings/

Додаток А.

Основні сутності та їх зв'язки:

User (користувач):

- user_id (PK)
- email, password_hash
- created_at, updated_at
- Зв'язки: 1:N з Order, 1:1 з Cart, 1:N з Address

Order (замовлення):

- order_id (PK)
- user_id (FK)
- order_status_id (FK)
- total_amount, currency
- created_at, updated_at
- Зв'язки: N:1 з User, 1:N з OrderItem, 1:N з Payment

OrderItem (позиція замовлення):

- order_item_id (PK)
- order_id (FK)
- product_id (FK до MongoDB)
- quantity, price_at_purchase
- Зв'язки: N:1 з Order

Payment (платіж):

- payment_id (PK)
- order_id (FK)
- payment_method_id (FK)
- amount, status
- transaction_id, created_at
- Зв'язки: N:1 з Order, N:1 з PaymentMethod

Cart (кошик):

- cart_id (PK)
- user_id (FK)
- created_at, updated_at
- Зв'язки: 1:1 з User, 1:N з CartItem

CartItem (елемент кошика):

- cart_item_id (PK)
- cart_id (FK)
- product_id (FK до MongoDB)
- quantity, added_at

Address (адреса):

- address_id (PK)
- user_id (FK)
- country, city, postal_code, address_line
- is_default

PaymentMethod (метод оплати):

- payment_method_id (PK)
- name (enum: credit_card, paypal, bank_transfer)

OrderStatus (статус замовлення):

- order_status_id (PK)
- name (enum: pending, confirmed, shipped, delivered, cancelled)

Рисунок А1. Опис сутностей