

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет кораблебудування імені адмірала Макарова
Навчально-науковий інститут комп'ютерних наук та управління проектами
Кафедра інформаційних управляючих систем та технологій

"Допущений до захисту"

Завідувач кафедри ІУСТ



к.т.н, доц. Михелєв І.Л.

" ____ " _____ 2025 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня вищої освіти "бакалавр"

на тему:

Аналіз технологій і розроблення підсистеми збереження даних корпоративної
інформаційної системи

Виконав: студент групи 4145

Литвинчук А.М.



(підпис)

Керівник роботи:

Доцент, к.т.н., Гайда А.Ю.

(посада, науковий ступінь вчене звання)



(підпис)

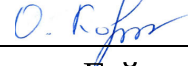
м. Миколаїв – 2025 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет кораблебудування імені адмірала Макарова
Навчально-науковий інститут комп'ютерних наук та управління проектами
Кафедра інформаційних управляючих систем та технологій
Спеціальність 124 "Системний аналіз"
Освітня програма "Системний аналіз"

"ЗАТВЕРДЖУЮ"

Гарант освітньої програми



к.т.н, доц. Гайдаєнко О. В.

" ____ " _____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ
на здобуття ступеня вищої освіти "бакалавр"

Студенту Литвинчуку Анатолію Миколайовичу

(прізвище, ім'я, по батькові)

1. Тема роботи: "Аналіз технологій і розроблення підсистеми збереження даних корпоративної інформаційної системи"

Керівник роботи к.т.н. Гайда А. Ю.

Затверджені наказом ректора № 295-уч від "27" березня 2025 року.

2. Термін подання роботи: 20 червня 2025 р.

3. Вихідні дані до проекту (роботи) (ДСТУ щодо обробки інформації, літературні джерела, технічна документація на існуючі аналоги систем): ДСТУ 3008:2015 – "Інформація та документація. Звіти у сфері науки і техніки"; ДСТУ ISO/IEC 15910:2012 "Інформаційні технології. Документація для програмного забезпечення"; ODOO: Управління базами даних – ODOO 15.0; ODOO: Как создать базу данных - ODOO development

4. Перелік питань, що належать до розробки (найменування розділів): Аналіз методів та засобів збереження та доступу до даних в корпоративних інформаційних системах; Розробка моделей підсистеми збереження даних корпоративної інформаційної системи; Впровадження підсистеми збереження даних

5. Перелік презентаційних матеріалів: Актуальність, мета, задачі роботи; Модель процесів збору, накопичення та обробки даних в КІС; Механізми доступу до даних КІС для реляційних та об'єктних СКБД; Архітектура розподілених СКБД для збереження і обробки даних КІС; Механізми доступу до даних КІС для розподілених СКБД; Математична модель мінімізації вартості організації і збереження даних КІС; Модель розподілу даних на основі аналізу використання даних; Реалізація модуля візуалізації результатів аналізу використання даних (фрагмент); Висновки

6. Консультанти розділів роботи

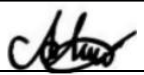
Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1	Доцент, к.т.н., Гайда А.Ю.	03.02.2025	22.02.2025
2	Доцент, к.т.н., Гайда А.Ю.	22.02.2025	21.03.2025
3	Доцент, к.т.н., Гайда А.Ю.	21.03.2025	25.04.2025
4	Доцент, к.т.н., Гайда А.Ю.	25.04.2025	07.06.2025

7. Дата видачі завдання 03 лютого 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів роботи	Примітка
1	Аналіз предметної області	13.02.2025	
2	Постановка задачі	15.02.2025	
3	Розробка концепції	22.02.2025	
4	Розробка проекту ІС	21.03.2025	
5	Реалізація проекту	25.04.2025	
6	Розробка питань з охорони праці	07.06.2025	
7	Розробка графічних матеріалів	12.06.2025	
8	Подання роботи на перевірку на плагіат	10.06.2025	
9	Подання роботи рецензенту	15.06.2025	
10	Подання роботи на захист	20.06.2025	

Студент


(підпис)

Литвинчук А.М.

(ПІБ)

Керівник роботи


(підпис)

Доцент, к.т.н., Гайда А.Ю.

(ПІБ)

Анотація

Робота присвячена аналізу технологій збереження даних і розробленню підсистеми організації і збереження даних корпоративної автоматизованої інформаційної системи.

В роботі представлені результати аналізу технологій збереження, організації та доступу до даних в корпоративних автоматизованих інформаційних системах, розглянуті та визначені особливості застосування інструментальних засобів збереження і організації даних в базах даних, визначені критерії використання локальних та розподілених баз даних, поділу даних між локальними та зовнішніми сховищами даних відповідно до їх призначення та використання в інформаційній системі, визначені підходи до оцінки відповідності обраних технологій та інструментів збереження і доступу до даних вимогам до інформаційної системи.

В роботі розроблені моделі і на їх основі запропонований підхід до оцінки засобів збереження даних відповідно до їх використання в окремих задачах автоматизованої інформаційної системи, що може застосовуватись в процесі проектування розподілених інформаційних систем, наведені результати застосування розроблених моделей та механізмів у реальних проєктах.

Робота викладена на ____ сторінках друкованого тексту, містить ____ рисунків, ____ таблиць, перелік джерел з ____ найменувань та ____ додатків.

Ключові слова: корпоративна інформаційна система; сховище даних; розподілена база даних; структура бази даних.

Abstract

The work is devoted to the robotic analysis of data technologies and the development of a subsystem for the preservation of the organization and data preservation of a corporate automated information system.

The work presents the results of the analysis of the preservation of technologies, organization and access to data in corporate automated information systems, presents and defines the features of the application of the preservation of organizational means and data in databases, defines the criteria for using local and distributed databases, divides data between local and external data storage tools according to their purpose and use in the information system, defines approaches to assessing the compliance of selected technologies and tools and preserving access to data with the requirements for the information system.

The work develops models and, based on them, proposes an approach to assessing the preservation of data from tools according to their use in individual tasks of an automated information system, which can be used in the design of distributed information systems, presents the results of applying the developed models and mechanisms in real projects.

The work is presented on ____ pages of printed text, contains ____ figures, ____ tables, a list of ____ names from sources and ____ appendices.

Keywords: corporate information system; data warehouse; distributed database; database structure.

Зміст

Вступ

1. АНАЛІЗ МЕТОДІВ ТА ЗАСОБІВ ЗБЕРЕЖЕННЯ ТА ДОСТУПУ ДО ДАНИХ В КОРПОРАТИВНИХ ІНФОРМАЦІЙНИХ СИСТЕМАХ

1.1. Огляд методів та засобів збереження та доступу до даних в корпоративних інформаційних системах

1.1.1. Великі дані і бізнес-процеси в корпоративних інформаційних системах

1.1.2. Проблеми використання даних на прикладі корпоративних інформаційних систем в інтернет-торгівлі

1.1.3. Проблеми використання даних на прикладі корпоративних інформаційних систем в дистанційному навчанні

1.1.4. Навантажені корпоративні інформаційні системи і великі дані

1.2. Аналіз технологій, методів та інструментальних засобів організації сховищ даних корпоративних інформаційних систем

1.2.1. Приклади проєктів

1.2.2. Розподілені системи зберігання даних та обробки запитів

1.2.3. Технології збереження даних NoSQL

1.2.4. Проблеми застосування класичних рішень збереження даних

1.3. Огляд інших рішень, які можуть бути застосовані для збереження даних

1.3.1. ОО СКБД

1.3.2. Документо-орієнтовані бази даних

1.3.3. Розподілені бази даних

1.3.4. Реляційні СКБД

1.4. Висновки щодо методів та засобів збереження даних

1.5. Огляд сервісів передачі та протоколів доступу до даних

1.5.1. Механізми доступу та сервіси

1.5.2. Протоколи HTTP та HTTPS

1.5.3. Служба FTP та SFTP

1.5.4. Висновки щодо методів та засобів передачі даних

1.6. Визначення проблем доступу до даних в корпоративних системах

- 1.6.1. Проблеми захисту даних та каналів передачі даних
- 1.6.2. Проблеми, викликані модифікацією структур даних
- 1.6.3. Проблеми синхронізації даних
- 1.6.4. Проблеми, що визначаються інструментальним забезпеченням

системи

1.7. Визначення критеріїв оцінки методів та засобів розробки сховищ даних корпоративних систем

1.8. Висновки до розділу 1

2. РОЗРОБКА МОДЕЛЕЙ ПІДСИСТЕМИ ЗБЕРЕЖЕННЯ ДАНИХ КОРПОРАТИВНОЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1. Моделі зберігання даних та обробки запитів

2.1.1. Моделі зберігання даних

2.1.2. Механізми забезпечення якості збереження даних

2.1.3. Механізми забезпечення якості доступу до даних

2.2. Розробка моделей вибору методів та засобів збереження та доступу до даних

2.3. Вибір інструментального забезпечення розробки проєктів корпоративних систем

2.4. Розробка моделей та механізмів інтелектуального аналізу даних

2.5. Висновки до розділу

3. ВПРОВАДЖЕННЯ ПІДСИСТЕМИ ЗБЕРЕЖЕННЯ ДАНИХ

3.1. Реалізація механізмів оцінки якості інструментального забезпечення підсистеми

3.2. Висновки до розділу

Загальні висновки по роботі

Список використаних джерел

Додаток А. Технічне завдання на розробку проєкту підсистеми збереження даних корпоративної інформаційної системи

Додаток Б. Статистика використання мов програмування в проєктах в Україні

Додаток В. Реалізація алгоритму обчислення вагових коефіцієнтів та вартості відповідно до критеріїв оцінки вимог до корпоративної ІС

Вступ

Зважаючи на вимоги корпоративних інформаційних систем до отримання і обробки значних обсягів даних сучасні системи збереження корпоративних даних повинні відповідати цим вимогам. Крім того, Інтернет-доступ до СКБД висуває підвищені вимоги до сховищ даних з точки зору масштабованості, яка характеризується наявністю можливостей користувачів корпоративних інформаційних систем щодо оперативного доступу до даних, розширення обсягів дискового простору, представлення результатів тощо. Все це разом визначає вимоги до методів доступу до даних, механізмів організації сховищ даних, вибору технологій зберігання даних, вдосконалення засобів адміністрування корпоративних сховищ даних та забезпечення відповідності їх структури вимогам бізнес-процесів в частині аналізу даних та представлення результатів аналізу.

Значні обсяги даних корпоративних інформаційних систем та їх стрімке зростання вимагають високої потужності засобів їх обробки, яка насамперед досягається за рахунок зростання пропускнуєї спроможності мереж, серверів додатків та серверів баз даних. Причому, якщо недостатня продуктивність або надійність мереж та серверів додатків призводить до збільшення часу на обробку транзакцій, то проблеми з серверами баз даних можуть призвести до втрати даних, наприклад, про замовлення або розрахунки за товари, що призведе до відмови покупців від користування такими системами.

Таким чином, істотною проблемою розробки та експлуатації (у тому числі – розширення) корпоративних інформаційних систем з великими наборами даних є обмеження, пов'язані з механізмами доступу до даних в мережі Інтернет в частині вибірки, передачі та захисту даних; обмеження, пов'язані зі складністю моделювання бізнес процесів, що постійно змінюються та вимагають модифікації алгоритмів обробки даних та самих структур даних; відповідність структур та способів представлення даних методам обробки даних. Як наслідок, розширення функціональності таких систем та нові вимоги щодо обробки даних часто супроводжуються змінами в структурах даних бази даних, які складаються в результаті розвитку таких систем, що для складних систем є нетривіальним завданням і часто вимагає суттєвої переробки модулів, інтерфейсів і архітектури

систем, які використовують такі дані. Також часто це вимагає змін в алгоритмах обробки даних, оскільки конвертація великих наборів даних з одного формату в інший сама є нетривіальною задачею.

Зважаючи на значну кількість технологій організації сховищ даних корпоративних інформаційних систем та використання даних в інформаційних системах постає задача науково обгрунтованого вибору технологій механізмів організації сховищ даних, збереження даних в сховищах даних навантажених систем.

Актуальність роботи визначається поширенням корпоративних технологій, які базуються на великих даних, та наявними проблемами, пов'язаними зі збереженням, обробкою та супроводом великих наборів даних, що створює серйозні перепони щодо забезпечення ефективного використання таких даних в корпоративних інформаційних системах, у тому числі - в задачах аналізу та добування нових знань.

Метою роботи є підвищення ефективності процесів збереження і обробки великих наборів даних в корпоративних інформаційних системах шляхом впровадження науково обгрунтованих моделей та механізмів організації і збереження даних.

Для досягнення мети роботи необхідно вирішити наступний перелік задач: виконати аналіз наявних технологій збереження та обробки даних в локальних та розподілених базах даних корпоративних інформаційних систем; визначити моделі організації даних в базах даних; розробити засоби організації і обслуговування великих наборів даних.

1. АНАЛІЗ МЕТОДІВ ТА ЗАСОБІВ ЗБЕРЕЖЕННЯ ТА ДОСТУПУ ДО ДАНИХ В КОРПОРАТИВНИХ ІНФОРМАЦІЙНИХ СИСТЕМАХ

1.1. Огляд методів та засобів збереження та доступу до даних в корпоративних інформаційних системах

1.1.1. Великі дані і бізнес-процеси в корпоративних інформаційних системах

Навантаженими є корпоративні інформаційні системи, що використовуються в електронній торгівлі, популярних пошукових системах [10], інформаційних та ігрових порталах, дистанційному навчанні тощо [9]. Окрім власне накопичення, обробки і надання інформаційних ресурсів такі сервіси створюють інформаційні моделі (профілі) користувачів, забезпечують управління обліковими записами користувачів та захист контенту.

Стрімке зростання обсягів даних в корпоративних інформаційних системах та, зокрема, в хмарних сховищах даних створює серйозні перепони щодо можливостей використання та обробки цих даних. Так, аналітики стверджують, що якщо зібрати всю інформацію, накопичену людством з початку часів і включно до 2000-го року, то виявиться, що її менше, ніж нині сучасне суспільство продукує протягом одної хвилини. Це змушує розробників корпоративних систем шукати підходи, які б дозволили ефективно зберігати, отримувати та обробляти дані в умовах швидкого зростання обсягів таких даних, які перевищують можливості сучасних комп'ютерів щодо їх збереження та обробки.

Інформаційні системи, навантаження на які перевищують можливості щодо ефективної, з точки зору деякої задачі, виконання запитів на збереження і обробку даних, називають “навантаженими сервісами”.

Спеціальні методи і алгоритми традиційних СКБД не здатні забезпечити прийнятний час відгуку на складних запитах даних в задачах аналізу та побудови корпоративної звітності. Крім того, зважаючи на великі обсяги даних та їх можливу безперервну зміну такі задачі часто вирішують з використанням розподілених сховищ даних і на спеціальних обчислювальних системах з масивно-паралельною архітектурою, які, у свою чергу, визначають жорсткі вимоги до систем збереження даних [10].

Розподілення даних між окремими вузлами в розподіленій інформаційній системі та паралельні обчислення дозволяють розпаралелити обробку даних і винести частину обробки з рівня СКБД на рівень окремих серверів зберігання даних. Як наслідок, останнім часом швидко розвивається напрямок розробки та впровадження інтелектуальних сховищ даних.

У свою чергу, задачі обробки великих наборів даних часто вимагають застосування спеціальних методів обробки даних, що базуються на методах математичної статистики, класифікації і кластеризації, нечіткої логіки тощо. Це пов'язане з тим, що великі набори даних без попередньої обробки не можуть бути сприйняті людиною, а тим більше не можуть безпосередньо, без попередньої обробки, використовуватись для прийняття рішень.

Також останнім часом активно розвиваються так звані хмарні обчислення (Cloud computing). Ця технологія стала привабливою для розробників корпоративних інформаційних систем і для окремих користувачів, оскільки такі сховища можуть бути отримані з невеликими витратами разом із сервісами обробки даних. Останнім часом в інтернет набули популярності служби, через які можна замовити необхідні для збереження і обробки великих наборів даних обчислювальні ресурси. Прикладами Cloud Computing є сервіси зберігання (Amazon S3), обчислювальні сервіси (Google App Engine, Amazon EC2) і сервіси даних (Amazon SimpleDB, Microsoft SQL Server Data Services, Google's Datastore) тощо.

1.1.2. Проблеми використання даних на прикладі корпоративних інформаційних систем в інтернет-торгівлі

Останніми роками в усьому світі все більшого поширення набувають методи та технології ведення бізнесу в Інтернет. Так, вже станом на 2018 рік обсяги електронної комерції сягнути близько 2.86 трлн доларів США. Значна частина цих обсягів – близько 2 трлн доларів США) припала на електронну або інтернет-торгівлю і надалі ці обсяги швидко зростали (Рисунок 1.1).

Застосування Інтернет дозволяє швидко і з незначними витратами виводити і просувати продукцію на національний і міжнародні ринки при відносно невеликих витратах на утримання торгових площ, придбання

торговельного обладнання, підготовку торговельного персоналу тощо. Як наслідок, торгівля через Інтернет дозволяє істотно знизити вартість продукції та доступність її широкого асортименту зі всього світу.

Крім того, споживач у реальному масштабі часу може переглянути номенклатуру продукції, що реалізується, швидко знайти потрібний товар, визначити його характеристики, ознайомитися з відгуками інших споживачів, обрати зручний спосіб і час доставки товару, відкласти платежі на час отримання товару або достроково провести платежі через Інтернеті тощо.

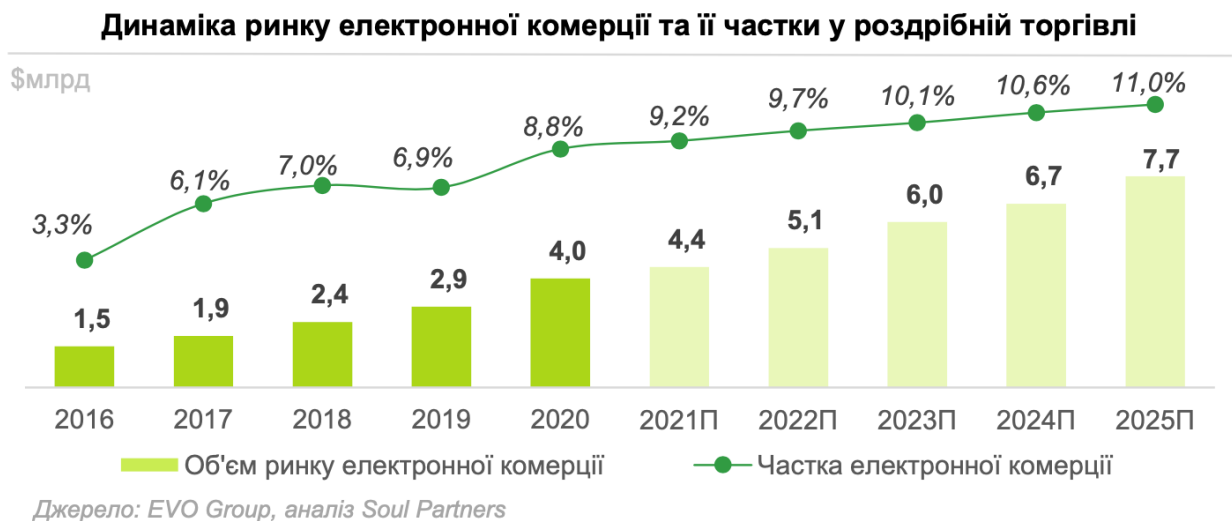


Рисунок 1.1. Світові обсяги електронної торгівлі за роками, млрд. доларів США

Значний інтерес інтернет-торгівля становить і для України, оскільки дозволяє в наявних складних економічних умовах швидко і з відносно невеликими витратами організувати збут продукції вітчизняних товаровиробників, забезпечуючи тим самим їх сталий розвиток. Так, в Україні довгий час використовуються і набули популярності загальнонаціональні мережі і портали інтернет-торгівлі, такі як “Prom.UA”, “Olx”, “Rozetka” тощо. На такий майданчик поступово перетворюються великі інтернет-магазини, надаючи свої сервіси (у тому числі для проведення розрахунків) іншим, відносно невеликим інтернет-магазинам. Прикладом таких сервісів є “Розетка”. Проте практика свідчить, що незважаючи на швидке зростання обсягів електронної торгівлі і в Україні і світі загалом наявний потенціал розвитку цього виду бізнесу все ще використовується недостатньою мірою, зокрема внаслідок того, що вибір

організаційних форм і технологічних засобів інтернет-торгівлі часто здійснюється нераціонально, без урахування специфіки товару та споживача, особливостей обраних цільових ринків тощо.

1.1.3. Проблеми використання даних на прикладі корпоративних інформаційних систем в дистанційному навчанні

Наслідком заходів щодо запобігання епідемії ковіду у світі в цілому і Україні зокрема стало істотне зростання інтересу до систем дистанційного навчання. Починаючи з 2022-го року в Україні цей інтерес також обумовлений необхідністю розосередження навчальних закладів в наслідок ударів і повітряних атак російських балістичних ракет та безпілотних систем на заклади цивільної інфраструктури.

В дистанційному навчанні сервіси використовуються для розміщення навчальних лекцій у форматі відео, аудіо та PDF. Популярні сервіси використовуються по всьому світу, що створює значні навантаження на веб-сервери та файлові сховища таких систем. Застосування тих чи інших засобів для організації занять визначається моделлю дистанційного навчання, яку використовує навчальний заклад або окремих викладач. Так, при організації навчального процесу за дистанційною формою за онлайн-моделі найчастіше використовуються загальнодоступні програми для спілкування в мережі Інтернет, наприклад, Zoom, Discord та подібних. Для організації офлайн-занять і при використанні дистанційних курсів, лектор також може використовувати різні системи управління навчанням, наприклад, Moodle та інші подібні.

Серед таких систем активно використовуються і розвиваються сервіси веб-конференцій, які дозволяють учням спілкуватися з викладачем безпосередньо через браузер. Вони використовуються багатьма навчальними закладами, що надають послуги дистанційного навчання. Зручність веб-конференцій полягає в тому, що учневі і викладачеві не потрібно встановлювати на персональний комп'ютер сторонні додатки. Відмінною рисою цих сервісів є використання спеціальної дошки (whiteboard) для написання формул, малювання графіків і тому подібні, а також спільний перегляд презентацій, навчальних відеороликів та інші.

Ще один варіант віддаленого спілкування між викладачем і учнем – використання спеціалізованих інтернет-сервісів для дистанційних репетиторів.

Загалом, подібні системи не визначаються надмірними обсягами даних, але, натомість, визначаються значною кількістю одночасно під'єднаних користувачів та значним навантаженням на мережу передачі даних через необхідність оперативної передачі аудіо- та відео- контенту.

1.1.4. Навантажені корпоративні інформаційні системи і великі дані

Поняття Big Data ("великі дані") окреслюють групу технологій та методів для отримання якісно нових знань, за допомогою яких аналізують та обробляють великі набори даних, як структурованих так і неструктурованих.

Big Data працює за принципом: чим більшу кількість інформації зберігає і може обробити система, тим значнішим і точнішим буде ефект від використання таких даних. Можливість порівняння певних даних та взаємозв'язків між ними дозволяє знайти закономірності, які були приховані до цього. Все це разом дозволяє знайти корисні рішення та можливості керування потрібними процесами.

Технології великих даних застосовуються при вирішенні різних задач, які можуть бути згруповані в декілька “кластерів”.

Найчастіше процес обробки великих об'ємів даних включає в себе побудову моделей та запуск симуляцій, під час яких постійно змінюються ключові налаштування, при цьому система постійно відслідковує, як ці зміни впливають на можливий результат. Це все відбувається в автоматичному режимі, допоки не буде знайдено деяку оптимальну з точки зору окремої задачі або комплексу задач організацію даних, який допоможе ефективно вирішити поставлену задачу або задачі.

Оскільки переважна більшість даних є неструктурована, то для перетворення їх у такі, що сприйматимуться людьми, використовуються сучасні технології аналізу [2]. Загалом, технології аналізу даних досить різноманітні. Зважаючи на великий обсяг даних та задачі, для яких проводиться аналіз, важливим є застосування саме тих способів організації даних в корпоративному сховищі даних, які дозволять з найменшими витратами часу та ресурсів отримати

корисні результати. Таким чином, організація даних в корпоративному сховищі виявляється напряду пов'язаними з бізнес процесами, в яких такі дані або результати їх обробки будуть використовуватись.

Значне різноманіття способів організації даних призвело до розроблення і поширення технологій, альтернативних традиційним системам керування базами даних і рішенням класу Business Intelligence. У дану серію включають засоби масово-паралельної обробки розподілених даних та невизначено структурованих даних, насамперед, рішеннями категорії NoSQL, алгоритмами MapReduce, програмними каркасами і бібліотеками проекту Hadoop тощо.

1.2. Аналіз технологій, методів та інструментальних засобів організації сховищ даних корпоративних інформаційних систем

1.2.1. Приклади проєктів

Останнім часом значне поширення в області корпоративних інформаційних систем з віддаленим доступом отримали технології, що базуються на PHP – інтерпретованій мові програмування високого рівня. PHP (від Hypertext Preprocessor – гіпертекстовий препроцесор, попередня назва Personal Home Page Tools) був створений для генерації HTML-сторінок на стороні веб-сервера. PHP є однією з найпоширеніших мов, що використовуються у сфері веб-розробок (разом із Java, Python, Ruby, .NET, Perl). PHP підтримується переважною більшістю хостинг-провайдерів і поширюється як відкрите програмне забезпечення. PHP інтерпретується веб-сервером у HTML-код, який передається на сторону клієнта. На відміну від скриптової мови JavaScript, користувач не бачить PHP-коду, бо браузер отримує вже згенерований html-код. Це є перевагою з точки зору безпеки, але погіршує інтерактивність сторінок, тому PHP часто використовують також для динамічного генерування JavaScript-кодів, які передаються клієнтам і виконуються вже на стороні клієнта.

Наприклад, русій сервісу “Facebook” був написаний на PHP, але згодом був перероблений, щоб знизити навантаження на сервери. Результатом переробки став транслятор HipHop for PHP, який попередньо одноразово транслював код PHP в компільовану мову C++ і компілював отриманий код. HipHop for PHP має обмеження, оскільки підтримуються не всі можливості PHP, отже Facebook

переведений на C++ лише частково. Із прогресом у розробці HipHop стало зрозуміло, що ці напрацювання можуть покращити швидкодію PHP застосунків загалом, а не тільки конкретних скриптів. Застосування HipHop давало 6-ти кратний приріст продуктивності при генерації web-сторінок у порівнянні з ядром PHP (Zend PHP).

Так само розробники корпоративних інформаційних систем, що використовують мову програмування Python, часто поєднують модулі Python з модулями на C++, щоб компенсувати відносно низьку швидкодію інтерпретатора Python. Одним з рішень для цього є так звані модулі-акселератори C++. Щоб вбудувати C++ у програми Python, потрібно створити динамічну бібліотеку на C++, скомпілювати її та зв'язати з основною програмою. Для вбудовування Python зазвичай застосовують CPython.

За наведеними прикладами можна визначити, що:

- Початковий вибір технологій та інструментальних засобів, що використовуються при створенні навантажених серверів (у тому числі корпоративних інформаційних систем), визначається амбіційністю задач, які намагаються вирішити розробники. Сервіси, що започатковувались як невеликі волонтерські проєкти, звичайно розроблялись мовами, що не відповідають вимогам до навантажених сервісів. В подальшому розробники таких сервісів змушені переходити на використання компільованих мов програмування, зазвичай C/C++.
- Використання мов, подібних до C/C++ на ранніх етапах реалізації таких проєктів істотно збільшує терміни розробки. C++ та подібні компільовані мови є потужним інструментом, і це вимагає високої дисципліни при проєктуванні та розробленні програмної системи.

Програміст, який пише на C++, повинен мати високу кваліфікацію, щоб код був ясним і зрозумілим для тих, хто буде підтримувати проєкт. Використання C++ також пов'язане з нестачею високорівневих готових рішень, особливо в області розробки рішень корпоративного рівня. Як приклад, для PHP та Python існує багато фреймворків та бібліотек, для C++ таких розробок практично немає.

Разом зі складностями, використання C++ надає істотні переваги, в першу чергу – можливість отримання готового ефективного коду, що практично

неможливо або вкрай важко досягти на інших платформах. Отже, отримання ефективного компільованого коду є вкрай важливим для навантажених корпоративних інформаційних систем та засобів обробки даних.

Основна перевага НірНор та подібних проєктів – підтримка з боку авторитетних розробників інструментального забезпечення (для НірНор — компанія Facebook). Також перевагою є те, що генерується нативний код. Такі продукти є також досить гнучкими в конфігурації і в них передбачено досить багато параметрів для налаштування.

1.2.2. Розподілені системи зберігання даних та обробки запитів

Для безпечного зберігання даних в корпоративних інформаційних системах використовують відповідні програмні і апаратні рішення [1]. Одним з таких рішень є розподілені дата-центри з дублюванням інформації на серверах. Це дозволяє залежно від місця клієнта надавати дані з сервера, який знаходиться ближче (тобто, з меншими затримками на обробку запиту і передачу даних, Рисунок 1.2). Також існують спеціалізовані компанії, які надають платні послуги CDN.

Для надання файлових даних використовують спеціалізовані асинхронні файлові сервери, розроблені компільованими мовами (наприклад С/С++). Завдяки файловим хостингам вдається вирішити проблему перевантаження для окремих серверів або інших ресурсів, а також надати для звичайного користувача можливість безкоштовного передавання файлу, наприклад, розміром від десятків мегабайт до 1 Гігабайта. Природно, що за безкоштовність доводиться платити (банери на сторінці завантаження, відтерміноване викачування користувачем, примусовий перегляд веб-сторінок), бо витрати власників файлових хостингів необхідно якось окупати.

Як приклад, на рис 1.2. представлені два центри обробки даних навантаженої інформаційної системи, до складу яких входять:

- Cluster Web – кластер веб-серверів для прийому і обробки замовлень від користувачів;
- Cluster FS – кластер серверів доступу до файлових ресурсів проєкту (download і upload контенту);

- GW – шлюз, який утворює захищений канал для об'єднання внутрішніх мереж різних дата-центрів в єдину мережу (використовується для створення розподіленої мережі зберігання даних і їх реплікації).

1.2.3. Технології збереження даних NoSQL

NoSQL – база даних, яка забезпечує механізм зберігання та видобування даних відмінний від підходу таблиць-відношень в реляційних базах даних. Подібні бази даних існували вже в другій половині 1960-х років. NoSQL бази даних все більше і більше використовуються в задачах із застосуванням даних та real-time web-застосунках. NoSQL системи також називають «Not only SQL» для підкреслення того, що вони можуть підтримувати SQL-подібну структуру, мову запитів та окремі структури.

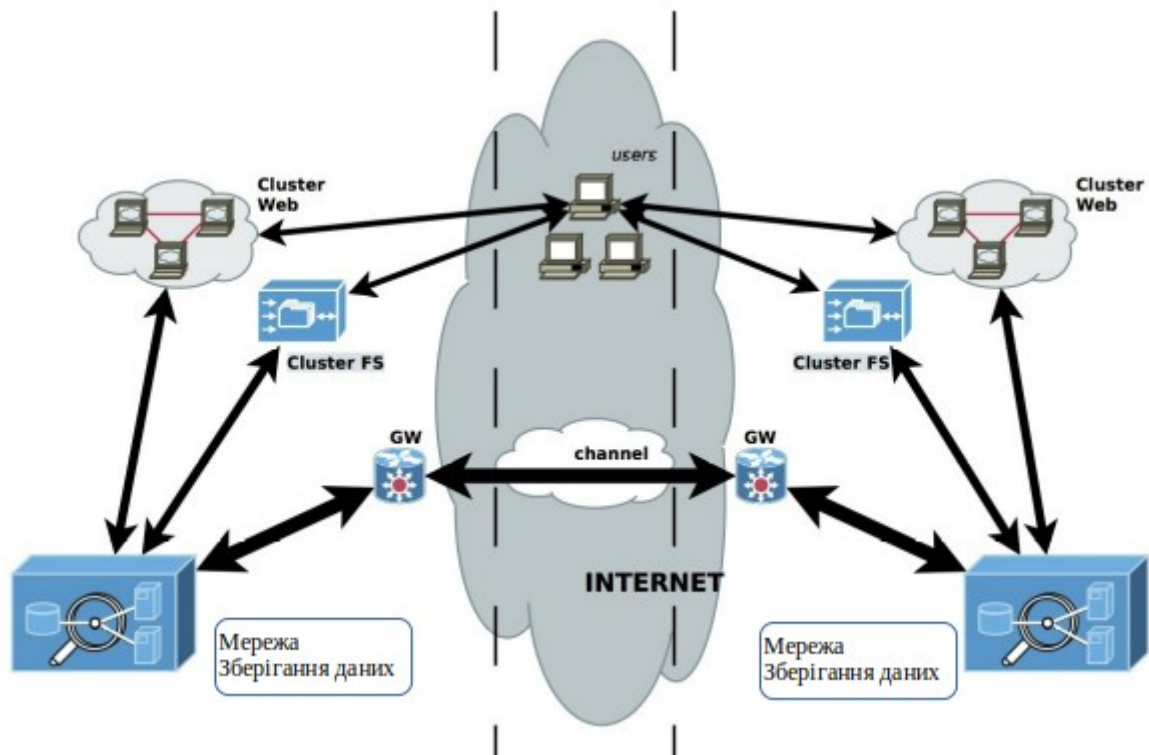


Рисунок 1.2. Приклад архітектури з розподіленою обробкою запитів і зберігання даних

NoSQL (Рисунок 1.3) забезпечують простоту та гнучкість дизайну схеми БД, значно спрощене горизонтальне масштабування у кластери (що є проблемою для реляційних баз даних [7]), і тонкий контроль над доступністю. Структури даних, що використовуються в NoSQL (такі як ключ-значення, Wide column store,

граф, документ) є відмінними від тих, що використовуються за замовчуванням в реляційних базах, що робить тим самим деякі операції над даними NoSQL значно швидшими.

Серед NoSQL-рішень виділяються:

- MongoDB – крос-платформова документо-орієнтована система керування базами даних з підтримкою JSON та динамічних схем;
- Apache Cassandra – масштабована база даних, орієнтована на стійкість до відмов;
- Hbase – масштабована розподілена база даних з підтримкою структурованого зберігання даних великого обсягу та інші.
- Серед NoSQL-рішень виділяються:
- MongoDB – крос-платформова документо-орієнтована система керування базами даних з підтримкою JSON та динамічних схем;
- Apache Cassandra – масштабована база даних, орієнтована на стійкість до відмов;
- Hbase – масштабована розподілена база даних з підтримкою структурованого зберігання даних великого обсягу та інші.

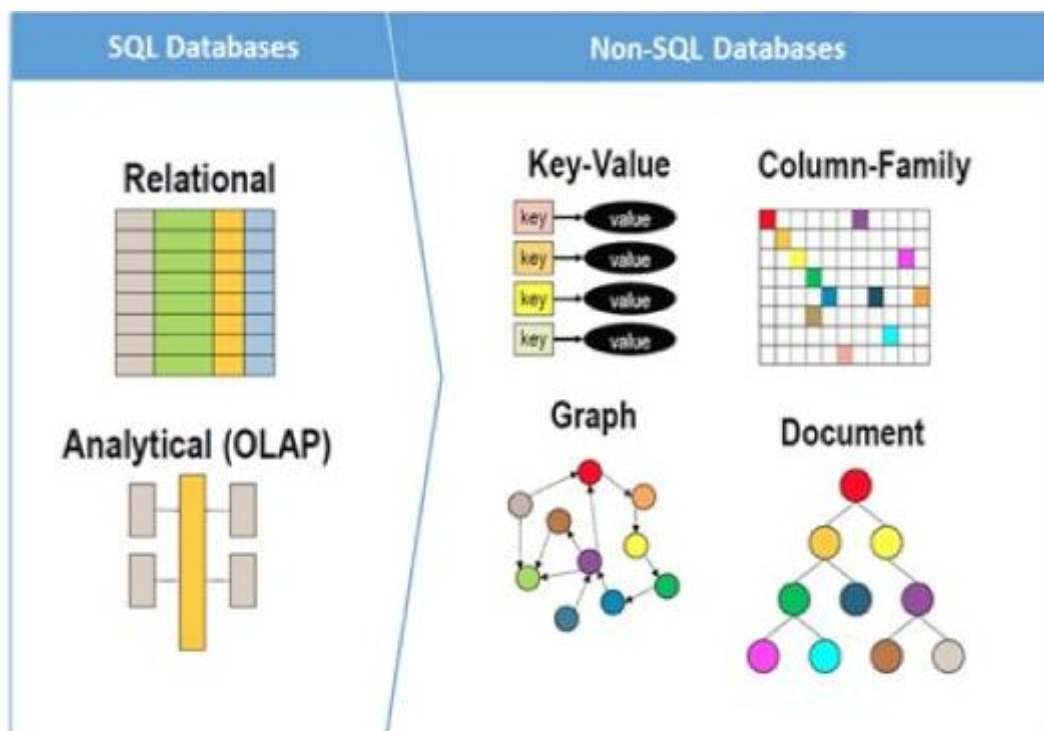


Рисунок 1.3. Відмінності SQL та NoSQL СКБД

MongoDB – це NoSQL сховище даних, вкрай зручне для зберігання інформації, яка не може бути у простий спосіб нормально структурована в рамках реляційних баз даних.

MongoDB – СКБД з відкритим вихідним кодом, що не вимагає опису схеми таблиць. СКБД написана на мові C++. Завдяки спеціалізації бази даних вдалося відійти від принципу «один розмір під все», а за рахунок мінімізації методів роботи з транзакціями з'явилася можливість вирішення цілого ряду проблем, пов'язаних з нестачею продуктивності, що також сприяло поліпшенню горизонтального масштабування.

Документи (дані) в MongoDB зберігаються в JSON або BSON форматах, робота з такою моделлю простіше кодується і простіше управляється, а внутрішня угруповання релевантних даних забезпечує додатковий виграш у швидкодії. MongoDB, за даними розробників, повинна заповнити розрив між найпростішими NoSQL-СКБД, що зберігають дані у вигляді «ключ – значення» (простими і легко масштабованими, але володіють мінімальними функціональними можливостями, такими, як наприклад, Memcached) і великими реляційними СКБД (зі складною організацій даних (структурними схемами даних) і системами генерації складних запитами з вбудованими оптимізаторами).

Переваги NoSQL:

- гнучка мова для формування запитів;
- динамічні запити;
- індексація колекцій;
- профілювання запитів;
- журналювання операцій запису;
- відмовостійкість і масштабованість;
- асинхронна реплікація і шардінг;
- повнотекстовий пошук з підтримкою морфології.

Робота з MongoDB реалізована в мовах програмування Java, C++, C#, PHP, Python, Perl, Ruby та інших, а відповідні компоненти є в багатьох фреймворках (в Ruby on Rails і в Yii, наприклад).

1.2.4. Проблеми застосування класичних рішень збереження даних

NoSQL. З метою отримання переваг NoSQL часто нехтують узгодженістю даних на противагу доступності, партиціюванню та швидкості. При цьому виникають складності створення та підтримки NoSQL сховищ – використання низькорівневої мови запитів замість добре розвиненого та стандартизованого SQL, нестандартизованість інтерфейсів. Як наслідок, більшість NoSQL СКБД пропонують концепцію випадкового узгодження даних, в якому зміни в базі дублюють на всі вузли «випадковим чином» (зазвичай така дія займає мілісекунди), тому запити даних можуть не повернути оновлені дані моментально, або ж прочитані дані будуть не точними. На додачу, деякі NoSQL системи можуть призводити до втрати даних, хоча окремі NoSQL забезпечують принцип WAL-журналювання для уникнення таких втрат даних.

Для розподіленої обробки транзакцій поверх множинних баз даних, узгодження даних є більш складним завданням, ніж воно постає при реляційному підході [6, 7].

Hadoop. Використання Hadoop також пов'язане з певними проблемами. Головне – висока вартість підтримки Hadoop-кластерів з багатьох машин. Також, на відміну від промислових сховищ даних на базі РСКБД, висока вартість отримання даних і значно перевищує вартість розробки SQL-запитів, що є наслідком застосування кластерної архітектури, використання map-reduce та job-ів. На додачу існує проблема з real-time. Hadoop – не real-time система. Для підтримки узгодженості розподілених даних потрібно постійно оновлювати дані.

Наявність зазначених проблем змушує розробників корпоративних інформаційних систем шукати і інші рішення, які були б здатні забезпечити (можливо з обмеженнями) збереження та обробку даних, у тому числі, при навантаженнях на систему, притаманних навантаженим веб-сервісам.

1.3. Огляд інших рішень, які можуть бути застосовані для збереження даних корпоративних інформаційних систем

1.3.1. ОО СКБД

Зазначені вище проблеми частково можуть бути подолані за рахунок використання об'єктно-орієнтованих систем управління базами даних (ОО СКБД), що дозволяють повною мірою задовольнити потреби в організації даних

з боку об'єктів програмного забезпечення. На жаль, об'єктно-орієнтовані СКБД на сьогодні не набули широкого поширення, насамперед – через свою складність і надмірну вартість [5, 6].

З метою подолання зазначених проблем останнім часом широко використовують об'єктно-орієнтовані надбудови над реляційними базами даних (ORM, Object Relational Mapping), які забезпечують підтримку відповідності організації даних на рівні додатку і на рівні структур даних бази даних [12, 13] (Рисунок 1.5).

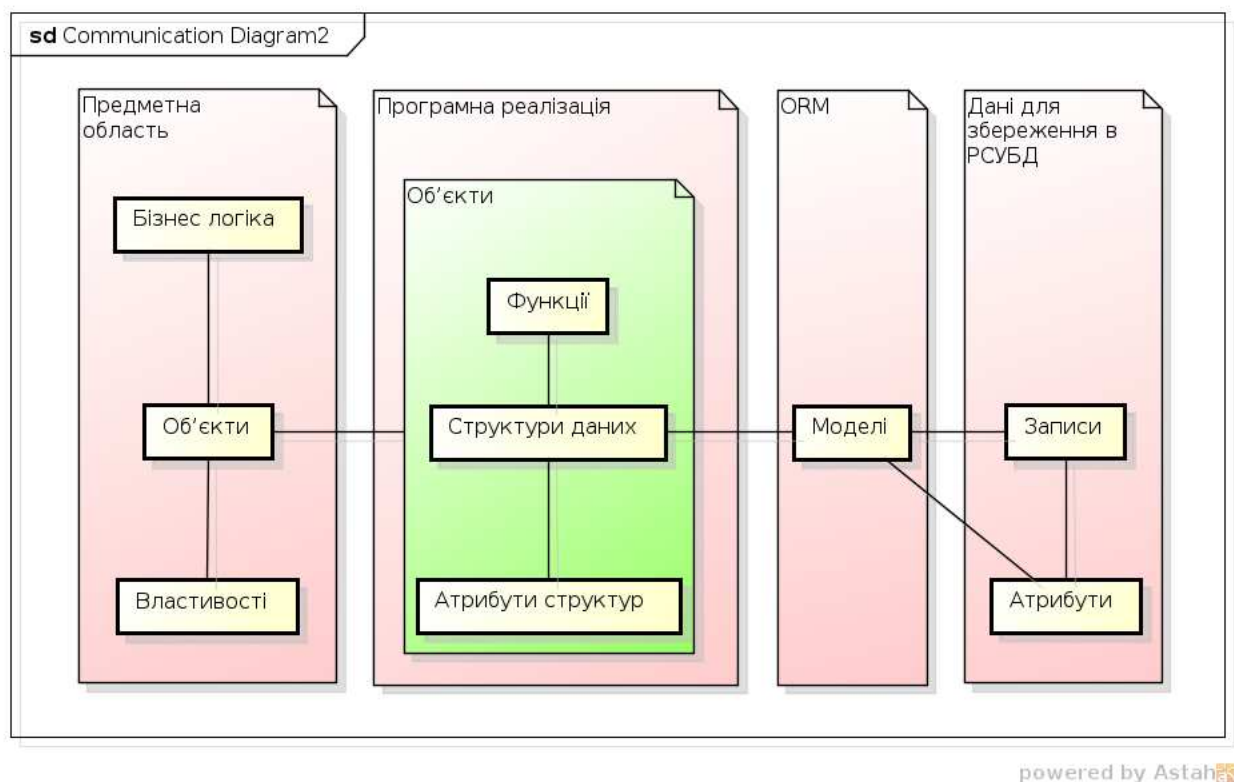


Рисунок 1.5. Відображення предметної області в реалізації інформаційної системи при об'єктному підході з ORM та РСКБД

Таким чином, використання ORM надає розробнику можливість відносно вільно маніпулювати даними на рівні об'єктів [12] і абстрагуватися від особливостей реалізації структур даних на конкретному сервері баз даних або на декількох вузлах кластера. Для цього в ORM можуть бути вбудовані адаптери даних, які надають користувачеві узагальнені механізми доступу до даних, приховуючи їх дійсне розміщення, формати та способи організації [13]. Як приклад, ORM може реалізувати SQL-доступ до NoSQL баз даних (Рисунок 1.6).

Це також дозволяє відносно простими засобами отримувати зрозумілий і якісний код системи обробки даних корпоративної інформаційної системи, спрощує його тестування, подальшу модифікацію і супровід.

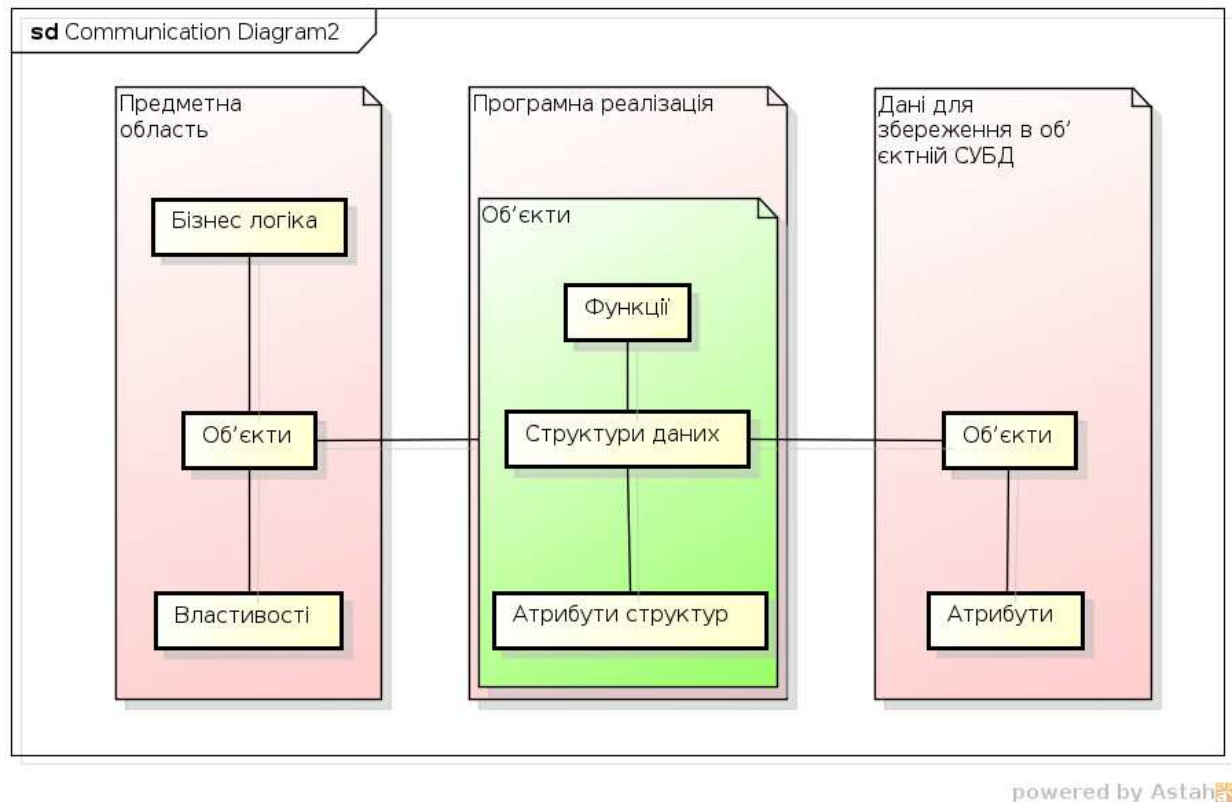


Рисунок 1.6. Відображення предметної області в реалізації інформаційної системи при об'єктному підході з використанням об'єктної СКБД

Деякі програмні пакети вирішують цю проблему, надаючи бібліотеки класів, здатних виконувати перетворення даних автоматично. Маючи список таблиць в базі даних і об'єктів в програмі, вони автоматично перетворюють об'єктні запити в реляційні та забезпечують конвертацію даних з внутрішнього подання бази даних в об'єктне уявлення, придатний для використання в об'єктно-орієнтованій програмі. При цьому в результаті виконання ORM-запиту генеруються і виконуються відповідні запити до таблиць сервера баз даних, а результати перетворюються в об'єкти відповідного типу.

Застосування ORM-технологій має і свої недоліки, насамперед, через деяке зменшення пропускну здатності сервера баз даних. Більше того, шар транзакцій при цьому виявляється повільнішим, а часто ще й неефективним (особливо в

термінах згенерованого запиту, записаного мовою запитів – SQL). Таким чином, використання ORM може призвести до того, що програми будуть працювати повільніше і використовувати більше пам'яті, ніж програми, що містять «вручну» написані запити, але ORM позбавляє розробника необхідності написання великої кількості коду, часто одноманітного і схильного до помилок, тим самим значно підвищуючи швидкість розробки. Більшість сучасних реалізацій ORM дозволяють жорстко визначати код SQL запитів, який буде використовуватися при тих чи інших діях (збереження в базу даних, завантаження, пошук тощо) з постійним об'єктом.

Взагалі, для проєктів, які характеризуються високим навантаженням на сховища даних, питання отримання необхідної продуктивності гостро стоїть і без застосування ORM. Але для таких систем також мають велике значення та можливості отримання відносно простих інструментів розширення системи, а саме такі можливості і забезпечуються використанням ORM. Впровадження ORM зазвичай призводить до деградації продуктивності системи на близько 30%. Маючи на увазі швидкий прогрес у підвищенні продуктивності обчислювальних систем (близько 30% – 50% на рік) і тенденції до зменшення вартості обладнання, можна розраховувати на те, що протягом року недоліки системи, пов'язані з деградацією обчислень внаслідок використання ORM будуть повністю нівельовані за рахунок підвищення продуктивності апаратного забезпечення. При цьому переваги від впровадження ORM, пов'язані з можливостями більш простого розширення структур даних, повною мірою будуть збережені і надалі [12].

1.3.2. Документо-орієнтовані бази даних

Документо-орієнтовані бази даних – спеціально призначені для зберігання ієрархічних структур даних (документів) і зазвичай реалізовані за допомогою підходу NoSQL. В основі документо-орієнтованих СКБД лежать документні сховища, які мають структуру дерева або лісу. Структура дерева починається з кореневого вузла і може містити кілька внутрішніх і листових вузлів. Листові вузли містять дані, які при додаванні документа заносяться в індекси, що дозволяє навіть при досить складній структурі знаходити місце (шлях) шуканих

даних.

API для пошуку дозволяє знаходити документо-орієнтованих базах даних за запитом документи або окремі частини документів. На відміну від сховищ типу ключ-значення, вибірка за запитом до документного сховища може містити частини великої кількості документів без повного завантаження цих документів на клієнтські машини.

1.3.3. Розподілені бази даних

Розподілені БД складають ще один напрям в просторі досліджень і розробок систем керування базами даних. У цих системах доводиться вирішувати всі завдання, властиві централізованим СКБД, але, як правило, в більш складних постановках. У розподілених системах виникають і специфічні проблеми, від вирішення яких багато в чому залежить ефективність, надійність і доступність систем БД. В даний час більшість розподілених СКБД базується на реляційній моделі даних і розрахована на використання в локальних обчислювальних мережах. Багато проблем поширюються і на розподілені СКБД в територіально рознесених мережах, і майже всі проблеми зберігаються для розподілених СКБД, заснованих на інших моделях даних.

Розподілена база даних (РБД) – це множина логічно взаємозалежних баз даних, розподілених у комп'ютерній мережі.

Серед особливостей використання таких баз даних для навантажених сервісів можна відзначити:

- великий обсяг транзакцій та обмінів даними (напружений мережевий трафік);
- низьку надійність системи через відмову каналів зв'язку та центральної СКБД;
- низьку загальну продуктивність (вузькі місця – мережа і центральний сервер);
- великі витрати на розробку і експлуатацію.

1.3.4. Реляційні СКБД

Рішення на базі реляційних СКБД є традиційними для багатьох проєктів

навантажених інформаційних систем через поширення цих СКБД та високий рівень формалізації реляційної алгебри. Вони також дозволяють забезпечити відображення даних предметної області в структури даних сховища у найбільш простий спосіб.

Найбільш поширеними базами даних є реляційні – MySQL, PostgreSQL, MSSQL і Oracle. Перші дві – безкоштовні і для сайтів використовуються найчастіше. Другі дві – платні і здебільшого застосовуються в корпоративних системах. По суті, для звичайних проєктів в технічному плані немає істотної різниці, яку базу використовувати, але в економічному плані вигідніше використовувати найпоширенішу MySQL (та його відгалудження MariaDB) або трохи менш поширену PostgreSQL (більше розробників, нижче вартість підтримки і розробки).

MariaDB – відгалуження реляційної СКБД MySQL, розроблювальне сообществом під ліцензією GPL. MariaDB повністю сумісна з додатками, що використовують MySQL, а перехід на цю СКБД виправданий тим, що MySQL вже не так активно розвивається.

Крім зрозумілої політики поширення продукту, дана СКБД відрізняється від MySQL більш високою продуктивністю, новими можливостями і меншою кількістю помилок в коді (зокрема, за рахунок кращого покриття тестами і більш активного розвитку).

У MariaDB вбудовані покращений оптимізатор запитів, безпечна і швидка реплікація, більш швидкі індекси для механізму зберігання даних MEMORY, більш висока продуктивність перекодування символів, використання пулу потоків, а також багато інших покращань, позитивно впливають на продуктивність.

У MariaDB відбулася відмова від підсистеми зберігання даних InnoDB і його заміна на XtraDB. Також на додаток до MyISAM, Blackhole, CSV, Memory і Archive включені підсистеми зберігання Aria, PBXT, FederatedX, OQGRAPH, SphinxSE, IBMDB2I, TokuDB, а також з версії 10.0 – Cassandra, CONNECT, SEQUENCE, Spider.

1.4. Висновки щодо методів та засобів збереження даних корпоративних

інформаційних систем

Для відносно невеликих проєктів простіше використовувати MySQL для більш складних — PostgreSQL за умов, що обсяги даних не перевищують їх можливості, оскільки на простих операціях не надто помітна різниця між різними СКБД. Якщо проєкт передбачає складну логіку обробки даних (Рисунок 1.7), що є характерним для корпоративних інформаційних систем, то вибір СКБД необхідно здійснити виходячи з технічних характеристик та області використання кожної окремої інформаційної системи та її сховища даних (Таблиця 1.1).

Таблиця 1.1. Типи сховищ баз даних за областями призначення

№	Тип сховища	Область використання сховища
	Реляційні СКБД	Класичні проєкти з відносно простою бізнес логікою
	NoSQL СКБД	Проєкти зі складною бізнес логікою та відносно невеликим обсягом даних
	ОО СКБД	Проєкти зі складною бізнес логікою та відносно великим обсягом даних
	Документо-орієнтовані СКБД	Проєкти з ієрархічно-впорядкованими даними
	Розподілені СКБД	Проєкти систем обробки даних з низькими вимогами до надійності

При зростанні обсягу даних у БД вузьким місцем стають канали передачі даних між сервером БД і пристроями з сервісами обробки даних. Можливість збільшити кількість і пропускну здатність таких каналів, а також знизити обсяг переданих даних за рахунок виконання частини операцій з перегляду та вибірці даних на рівні осередків зберігання, дозволяє зняти проблему каналів передачі. Така зв'язка обладнання та спеціального ПЗ, звана машина баз даних, описана в літературі. Приклади – Netezza, Datallegro, Teradata.

Враховуючи переваги та недоліки кожної з технологій було б бажано поєднати технології обробки даних, які використовуються в реляційних [7] та нереляційних системах управління даними [5]. Для цього необхідно також розглянути механізми передачі даних між вузлами системи.

1.5 Огляд сервісів передачі та протоколів доступу до даних навантажених систем

1.5.1. Механізми та сервіси доступу до даних

Окрім структури організації даних бази даних також істотно відрізняються механізмами доступу, які можуть бути реалізовані на різних рівнях реалізації інформаційної системи [8]: на рівні файлової системи ОС, СКБД або навіть на рівні додатку (Рисунок 1.8).

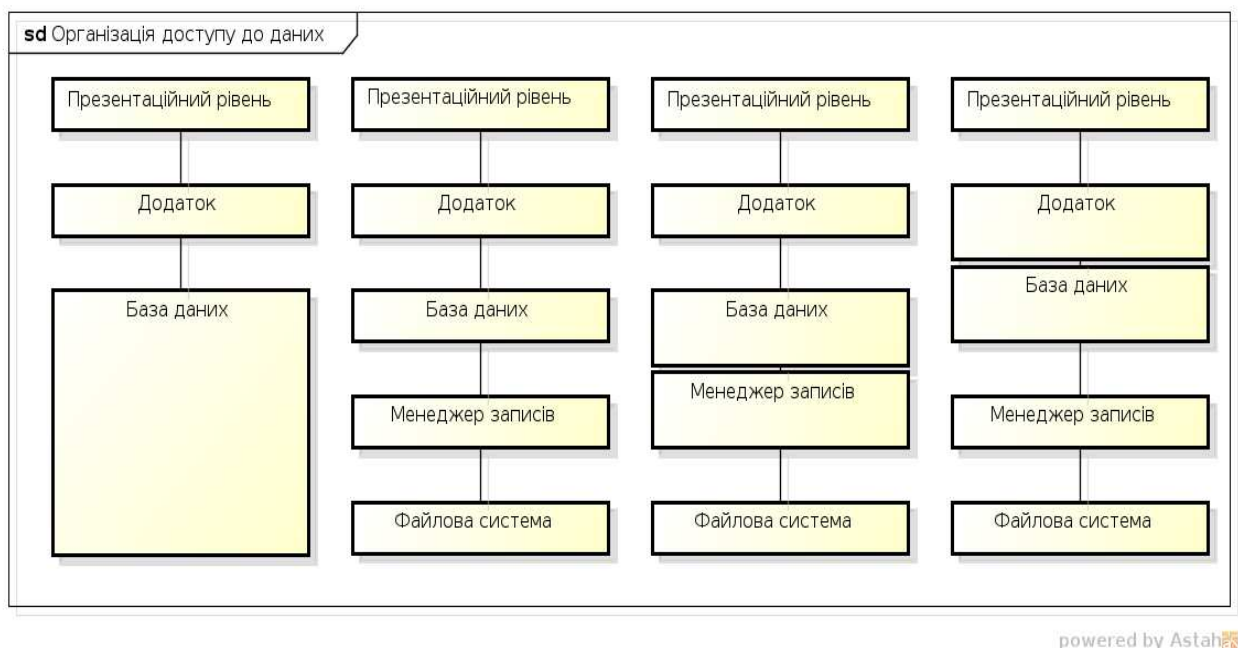


Рисунок 1.8. Реалізація механізмів доступу до даних за рівнями

Залежно від рівня реалізації механізмів доступу також істотно відрізняються можливості щодо побудови складних запитів (чим нижче рівень – тим складніше) та швидкості їх виконання (чим вище рівень – тим повільніше).

1.5.2. Протоколи HTTP та HTTPS

HTTPS (HyperText Transfer Protocol Secure) – захищений протокол передачі гіпертексту. HTTPS – розширення протоколу HTTP, що підтримує шифрування. Передані по протоколу HTTPS дані зашифровані, що забезпечує захист від прослуховування. HTTPS широко використовується в світі і підтримується всіма популярними браузерами.

HTTPS – це не окремий протокол, а розширення звичайного HTTP, що

працює через шифровані транспортні механізми SSL або TLS. HTTPS забезпечує захист від атак, заснованих на прослуховуванні мережевого з'єднання, тобто від сніфферських атак і атак типу man-in-the-middle.

На відміну від HTTP, який використовує TCP-порт 80, для HTTPS за замовчуванням використовується TCP-порт 443. Щоб підготувати веб-сервер для обробки https-з'єднань, адміністратор повинен отримати і встановити в систему SSL-сертифікат для цього веб-сервера.

SSL-сертифікат – це свого роду «електронний паспорт» сайту, в якому він підтверджує, що він дійсно той сайт, за який видає себе. Сертифікат сайту покликаний додатково підтверджувати, що цей сайт – дійсно сайт банку, платіжної системи і т.д. SSL-сертифікат, як і всякий паспорт, зберігає різні дані: найменування організації та підрозділи, країну, регіон, місто і ім'я сервера, для якого даний сертифікат був створений.

Також, для нормальної роботи HTTPS, SSL-сертифікат повинен бути підписаний організацією, якій довіряють все. У разі SSL-сертифікатів такими організаціями є центри сертифікації. І якщо сертифікат підписаний надійним центром сертифікації, то вважається, що він містить правильну і відповідну дійсності інформацію.

Вартість сертифікатів сильно різниться залежно від центру сертифікації та для EV-сертифікатів (Extended Validation Certificates) може становити більше 1 000 євро на рік, процес оформлення таких сертифікатів вимагає значного обсягу документів. Також існують і сертифікати, створення без участі центру сертифікації – самопідписні. Технічно, вони забезпечують абсолютно ідентичне шифрування, але для використання на публічних сайтах сертифікати непридатні, оскільки для їх нормальної роботи потрібні налаштування на стороні користувача сайту, без них браузер буде повідомляти користувачеві, що дане з'єднання не є довіреним.

Широкий спектр технологій доступу до даних породжує проблеми вибору цих технологій в окремих проєктах (Таблиця 1.2).

Вибір конкретних технологій здійснюється, як правило, за результатами вибору бази даних для проєкту.

Таблиця 1.2. Типи технологій доступу залежно від баз даних

№	Тип доступу	Область призначення сховища
1	HTTP-доступ	Незахищені файлові сховища, реляційні СКБД
2	HTTPS-доступ	Захищені файлові сховища, реляційні СКБД

1.6. Визначення проблем доступу до даних в корпоративних системах

1.6.1. Проблеми захисту даних та каналів передачі даних

Очевидно, що засоби захисту даних від несанкціонованого доступу дуже важливі і будуть продовжувати вдосконалюватися. Будуть ускладнюватися алгоритми кодування даних на всіх рівнях (в БД, архівах, при передачі по мережі), способи авторизації і аутентифікації і т.д.

Останнім часом з'явилася тенденція виносу механізму управління користувачами (обліковими записами) з окремих СКБД і додатків в єдину централізовану систему організації (Identity & Access management). Що спрощує управління користувачами в масштабах організації та дозволяє керувати ними з урахуванням вимог бізнесу, в автоматичному режимі. Удосконалюються методи аудиту, з'являтиметься інтегрована середовище збору та аналізу аудит-інформації всієї організації (зі безлічі додатків і БД).

Крім того, інтенсивно вдосконалюються засоби захисту даних усередині БД. Наприклад, швидко розвивається механізм завдання політик перевизначення запитів на льоту, тобто запит автоматично модифікується залежно від самих різних параметрів (назва програми, час запуску на виконання, ім'я користувача, місце надходження запиту – Інтернет / локальна мережа, номер терміналу і т.д.).

Наступним аспектом, що визначає захищеність даних корпоративних систем, є обсяг даних. Цей аспект є ключовим при розробці навантажених корпоративних систем (через число користувачів та значний обсяг даних) і застосовуються для зменшення обсягу даних механізмами стиснення даних. По-перше, для різних даних повинні автоматично застосовуватися різні алгоритми стиснення і лише тоді, коли стискання дає вигоду. По-друге, адміністратор повинен мати можливість вибирати різні рівні стиснення, розуміючи, що за економію місця доведеться платити процесорним часом. По-третє, стиснення має

бути прозоре для програм. При цьому СКБД повинна працювати зі стиснутими даними так само ефективно, як і з нестиснутими даними, не витрачаючи час на перебудову таблиць, на розтискання непотрібних блоків.

Традиційні алгоритми стиснення дозволяють стиснути дані в 2-3 рази. Деякі специфічні види стиснення дозволяють збільшити цю ступінь стиснення на порядок, але при цьому сильно сповільнюється виконання операторів мови SQL, які використовуються для роботи з цими даними. Наприклад, відомо, що стиснення за полями збережених і заздалегідь відсортованих даних забезпечить дуже високий рівень стиснення (іноді в десятки разів). Проте спроби зміни таких таблиць в Oracle 11.2 показали, що час оновлення даних значно збільшується.

Адміністратор бази даних повинен мати можливість обирати ті варіанти стискання та зберігання (за рядками, стовпцями або окремими таблицями) для різних даних, різних таблиць і навіть частин таблиць, які найбільш підходять для вирішення поточної задачі з точки зору бізнес-використання цих даних. Причому, оскільки для додатків способи зберігання і стиснення прозорий, адміністратор бази даних може періодично міняти способи зберігання і стиснення.

Загалом, такий підхід буде дуже корисний користувачам великих БД (наприклад, SAP-користувачам), які сьогодні змушені постійно збільшувати системи зберігання великих даних. Розумний механізм стиснення дозволяє залишати найсвіжіші та найбільш активно використовувані дані деякий час не стиснутими. Пізніше вони стискаються автоматично у фоновому режимі [8].

Ще один перспективний напрям розвитку – ізоляція адміністратора бази даних від даних. Зазвичай адміністратор може бачити і змінювати всі дані в БД, але вже розроблені засоби (наприклад, Oracle Data Vault option та подібні), які дозволять адміністратору бази даних виконувати всі операції по адмініструванню БД без можливості перегляду і, тим більше, зміни даних. Також адміністратору обмежують набір дозволених до виконання операцій. Обмеження може бути прив'язане до імені адміністратора бази даних, часу, точки підключення до системи тощо.

Таким чином, розглянуті механізми вказують, що динамічна реконфігурація бази даних, включаючи її структуру, не тільки можлива, але і виявляється корисною.

1.6.2. Проблеми, викликані модифікацією структур даних

Як показано на рисунку 1.2 структура баз даних може істотно відрізнятись. Це значить, що перехід з однієї структури на іншу не може бути здійснений шляхом заміни операційної системи або сервера СКБД – такий перехід потребуватиме повного перепроєктування бази даних та переробки алгоритмів доступу до даних в програмному забезпеченні додатків.

Отже, фактично, перехід на нову структуру бази даних вимагає повної переробки системи. Причому, витрати на таку переробку можуть бути навіть більшими, ніж на первинну розробку, оскільки ця переробка має здійснюватись в умовах, коли в первинній системі вже були накопичені дані і їх перенесення та забезпечення, повноти та внутрішньої узгодженості може стати досить складною та відповідальною задачею.

1.6.3. Проблеми синхронізації даних

Проблеми синхронізації даних розглянемо на прикладі однорангової мережі, в якій може ефективно виконуватись передача великих файлів між декількома вузлами (системами збереження/обробки даних).

Синхронізація робочих файлів між пристроями – найбільш очевидний спосіб, який дозволяє синхронізувати дані між декількома БД або серверами, забезпечуючи наявність на кожному з пристроїв ідентичною інформації. Даний спосіб корисний в організації зберігання особистих і корпоративних даних [9].

Резервне копіювання може використовуватися в якості транспортного механізму між пристроєм, дані з якого вимагають резервування, і місця зберігання резервних копій. По суті, на резервується пристрої можна створити дзеркала потрібних директорій і реалізувати на їх базі інкрементне резервне копіювання або версійність.

Віддзеркалення і реплікація не самих критичних даних в хмарних або кластерних системах забезпечує зберігання ідентичного набору даних на декількох серверах, що дозволяє розподілити між ними навантаження. При цьому треба враховувати, що синхронізація потребує часу та обчислювальних ресурсів, тому в реальній роботі не варто спиратися на те, що всі вузли володіють

актуальним і ідентичним набором даних.

Також варто врахувати те, що репліки не усувають конфлікти інтелектуально – якщо в наборі даних є файли з ідентичними ім'ям і шляхом, то буде збережена тільки найбільш пізня версія. Таким чином, якщо файли створюються програмним забезпеченням відразу на декількох вузлах (master-master реплікація), то використовують механізми іменування, які дозволять уникнути конфліктів імен.

1.6.4. Проблеми, що визначаються інструментальним забезпеченням системи

За окремими виключеннями будь який проєкт може з початкового рівня перерости в значний за обсягом. Таким чином, з точки зору предметної області вимоги до проєктів не можуть бути досить просто визначені відповідно до особливостей предметної області, але ці особливості напряму впливають на вибір інструментального забезпечення, що використовується при реалізації проєкту. Отже, при цьому виникають істотні проблеми, пов'язані з специфікою стосовно предметної області та рівнем стосовно операційної системи задач, що мають бути вирішені в проєкті.

Для написання програм на низькому рівні абстрагування, докладного і детального опису операцій, що враховують архітектура комп'ютера та операційної системи, використовують низькорівневі мови програмування, такі як асемблер та С. У розробці корпоративних інформаційних систем (у тому числі, веб програмуванні) асемблер ніколи не використовується, а мова С звичайно застосовується для розробки самого інструментального забезпечення, а не користувацьких додатків. В порівнянні з більш високорівневими мовами програмування програми мовою С звичайно вимагають більших витрат часу та високого рівня кваліфікації програмістів, отже збільшується і ймовірність появи помилок. Це можна вважати недоліком програмування на мовах відносно низького рівня взагалі, та програмування додатків навантажених сервісів корпоративних інформаційних систем зокрема.

Мови програмування високого рівня дозволяють подолати недоліки низькорівневого програмування – використовувати більш розвинуті операції

обробки агрегатних даних (списки, вектори, масиви, черги, стеки, сервіси операційної системи, комплексні числа тощо), не зважаючи на деталі їх реалізації на конкретній архітектурі. Програми при цьому виявляються більш короткими, надійними і універсальними, а процес їх складання скорочується. Програми на мовах високого рівня також легше читати та супроводжувати, в них простіше розібратися. Мови високого рівня бувають чотирьох видів: імперативні (процедурні), функціональні, логічні, об'єктно-орієнтовані. Основними об'єктами в імперативних мовах є змінні, оператори присвоювання, стандартні алгоритмічні конструкції.

Об'єктно-орієнтовані мови також значно спрощують процеси програмування за рахунок використання технологій об'єктно-орієнтованого програмування (Python, Java тощо), що, з точки зору можливого документально-орієнтованого представлення інформації корпоративних систем, є корисним.

Враховуючи вищезазначене визначимо, що однією з основних проблем вибору інструментального забезпечення для реалізації проєктів корпоративних систем слід вважати специфіку задач кожного конкретного проєкту з точки зору вимог особливостей методів обробки даних, що визначаються бізнес-процесами та реалізуються інструментальними засобами розробки.

1.7 Визначення критеріїв оцінки методів та засобів розробки проєктів корпоративних систем та постановка задачі дослідження

На основі результатів аналізу сучасних систем управління базами даних, їх переваг та недоліків можна визначити декілька критеріїв щодо можливостей застосування СКБД кожного з типів для вирішення задач збереження даних корпоративних інформаційних систем.

Враховуючи, що однією з основних проблем вибору інструментального забезпечення для реалізації корпоративної системи обробки даних слід вважати специфіку задач кожного конкретного проєкту з точки зору вимог особливостей методів обробки даних, що визначаються бізнес-процесами організації. Зважаючи на вищенаведене, визначимо наступні ключові критерії щодо застосування інструментальних засобів у проєктах обробки даних .

Залежно від вимог щодо інструментального та методичного забезпечення

кожної з цих областей наявні інтернет-проекти можуть бути поділені на категорії з різними вимогами щодо навантаження на систему. Нижче наведені окремі приклади областей застосування таких систем (таблиця 1.3).

Наведений перелік звичайно не є повним, але він дає можливість виконати оцінку адекватності зазначених критеріїв стосовно предметних областей. Як наслідок, для забезпечення цих критеріїв можуть застосовуватись різні інструментальні засоби, кожен з яких тим чи іншим чином відповідає вимогам щодо критеріїв.

Очевидно, що для забезпечення відповідності властивостей систем зазначеним вище критеріям якості інформаційних систем знадобляться і відповідні інструментальні засоби, методи проектування, механізми збереження, обробки та передачі даних.

Виходячи з результатів аналізу виконаємо постановку задачі дослідження у наступному формулюванні: розробити моделі вибору технологій збереження і обробки даних, що дозволять обрати з наявного спектру інструментального забезпечення, методів та засобів збереження, транспортування та обробки даних обрати ефективні засоби.

Таблиця 1.3. Приклади ранжування проектів за категоріями вимог до навантажених інформаційних систем

Предметна область №	Збереження та надання ресурсів	Комунікації, захист та обмін інформацією	Масові обчислення
1 Розвиток мереж	середні	низькі	середні
2 Управління якістю	високі	середні	низькі
3 Цільовий маркетинг	високі	середні	низькі
4 Кібербезпека та антиспам	середні	середні	високі
5 Соціальні мережі	низькі	високі	низькі
6 Контроль фінансових потоків	високі	середні	низькі
7 Call-центри	високі	високі	низькі
8 Геоаналітика	високі	середні	високі

1.8. Висновки до розділу 1

В розділі були розглянуті технології організації і доступу до даних у проєктах корпоративних систем, що визначаються інструментальним забезпеченням, методами та засобами збереження, транспортування, захисту та обробки даних проєктів корпоративних систем.

Визначено, що широке представлення та популярність технологій збереження даних у файлових сховищах навантажених сервісів та інтеграція корпоративних систем у загальний інформаційний простір вказують на те, що при збереженні значних обсягів інформації корпоративні інтернет-системи можуть використовувати такі сховища та надавані ними інструменти доступу до даних як одну з технологій збереження та доступу до даних.

В розділі також виконано постановку задачі.

2. РОЗРОБКА МОДЕЛЕЙ ПІДСИСТЕМИ ЗБЕРЕЖЕННЯ ДАНИХ КОРПОРАТИВНОЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1. Моделі зберігання даних та обробки запитів

2.1.1. Моделі зберігання даних

Складність вирішення задач роботи з даними починається з моделювання бізнес процесів загалом та необхідної час від часу реорганізації структур даних в базах даних як одного зі способів зробити такі бази даних більш ефективними при роботі з надвеликими наборами даних. При використанні реляційних СКБД (які набули найбільшого поширення і продовжують застосовуватись при роботі з великими даними) ускладнення структур даних також відбувається внаслідок невідповідності методів організації даних в програмах, що використовують базу даних (вони, зазвичай, розробляються відповідно до методологій об'єктно-орієнтованого програмування) і методів організації даних в необ'єктних сховищах даних. Така невідповідність призводить до необхідності побудови і виконання над базами даних вкрай складних запитів, що в умовах даних істотно збільшує час обробки запитів або взагалі унеможливорює їх виконання через нестачу оперативної пам'яті. При цьому розробка запитів на складних структурах даних є невиправдано трудомістким завданням і, в окремих випадках, вимагає створення допоміжних таблиць даних (Temporary tables) або видів (Views), що, в свою чергу, також ускладнює роботу з базою даних. За таких умов будь-які зміни структури бази даних внаслідок змін у бізнес логіці або розвитку систем змушує розробників виконувати переробку раніше створених запитів.

Ці задачі виявляються тим більше складними, що помилки вибору структури бази даних важко виправляти через істотні розбіжності в механізмах організації даних в базах даних, а самі дані, як правило, виявляються розподіленими між декількома вузлами.

Враховуючи особливості даних навантажених сервісів в частині структуризації (а окремі дані часто є неструктурованими) для збереження даних намагаються використовувати, насамперед, системи баз даних, що підтримують неструктуровані дані. Серед таких систем найбільше поширення отримали NoSQL, MapReduce, Hadoop.

На основі вищезазначеного може бути запропонована архітектура, яка

дозволять здійснювати розподіл даних між окремими СКБД, за якої на користувацьку машину можуть бути передані дані, необхідні для їх оперативної обробки (Рисунок 2.1).



Рисунок 2.1. Архітектура розподіленої системи з центральним сервером БД

З метою визначення особливостей реалізації бізнес логіки в корпоративних інформаційних системах, пов'язаних з обробкою даних навантажених сервісів, та її відображення в моделі та механізми збереження, доступу та обробки даних розглянемо особливості обробки запитів сучасних СКБД, принципи розробки таких систем та визначимо їх переваги та недоліки.

2.1.2. Моделі обробки запитів

Найпростіше, дешеве і добре випробуване рішення підвищення продуктивності навантажених систем в частині обробки запитів – кешування байткоду замість його багаторазового завантаження, парсингу та синтаксичного аналізу. Для цього можуть бути застосовані спеціальні розширення, такі як APC, eAccelerator, Xcache та інші. Zend Engine просто виконує байткод.

Другий варіант – це профілювання коду, виявлення вузьких місць. При цьому підході розробники визначають модулі, що створюють найбільші затримки або навантаження на ресурси і переписують їх компільованими мовами (звичайно C/C++).

Третій варіант більш глобальний – переписати весь код компільованими мовами (звичайно C/C++). Такий варіант застосовують тільки в тому випадку,

коли php-коду відносно мало. У великих проєктах (або тих, що існують досить довгий час) обсяг коду зазвичай значний, отже виконати його переписування складно. Є також альтернативний варіант ручному переписуванню – компілювати код, написаний інтерпретованою мовою, у щось більш швидке. Такий варіант можливий для мов, для яких підтримується такий інструмент.

2.1.3. Принципи розробки навантажених сервісів

Як було показано вище, розробка підсистеми керування даними корпоративної інформаційної системи не можлива без визначення ключових принципів побудови великих систем. Насамперед, це викликано тим, що чим більшою стає система, тим більш складним і менш ефективним буде її довгострокове планування. Але успішність роботи над великим проєктом такої системи має на увазі зовсім не детальне планування всіх аспектів, основні зусилля мають бути спрямовані на забезпечення гнучкості системи, що дозволить забезпечити прийнятний фактор розвитку такої системи.

При розробці бажано дотримуватись відомого розподілу Парето: слід зосередитись на розробці 80% функціоналу, а решту 20% відкинути – це окремі випадки, які ведуть до значного ускладнення системи, але істотно не впливають на її функціональність. Наприклад:

- Деякі користувачі в соціальній мережі можуть мати десятки тисяч зв'язків. Якщо їх менше 20%, слід поставити обмеження на число зв'язків і відкласти вирішення цієї задачі, поки є важливіші проблеми.
- Деякі користувачі у файлообмінній мережі можуть завантажувати відео-файли, які мають нестандартну систему кодування. Якщо їх менше 20% краще згенерувати помилку замість того, щоб витратити час на доопрацювання конвертера.
- Деякі користувачі використовують браузер з відключеними куками, обмеженнями на виконання Javascript тощо. Згідно з принципом Парето, якщо таких користувачів менше 20%, то краще вимкнути для них можливість використання сайту і розмістити докладні інструкції по оновленню браузера замість адаптації програмного забезпечення сайту під таких користувачів.

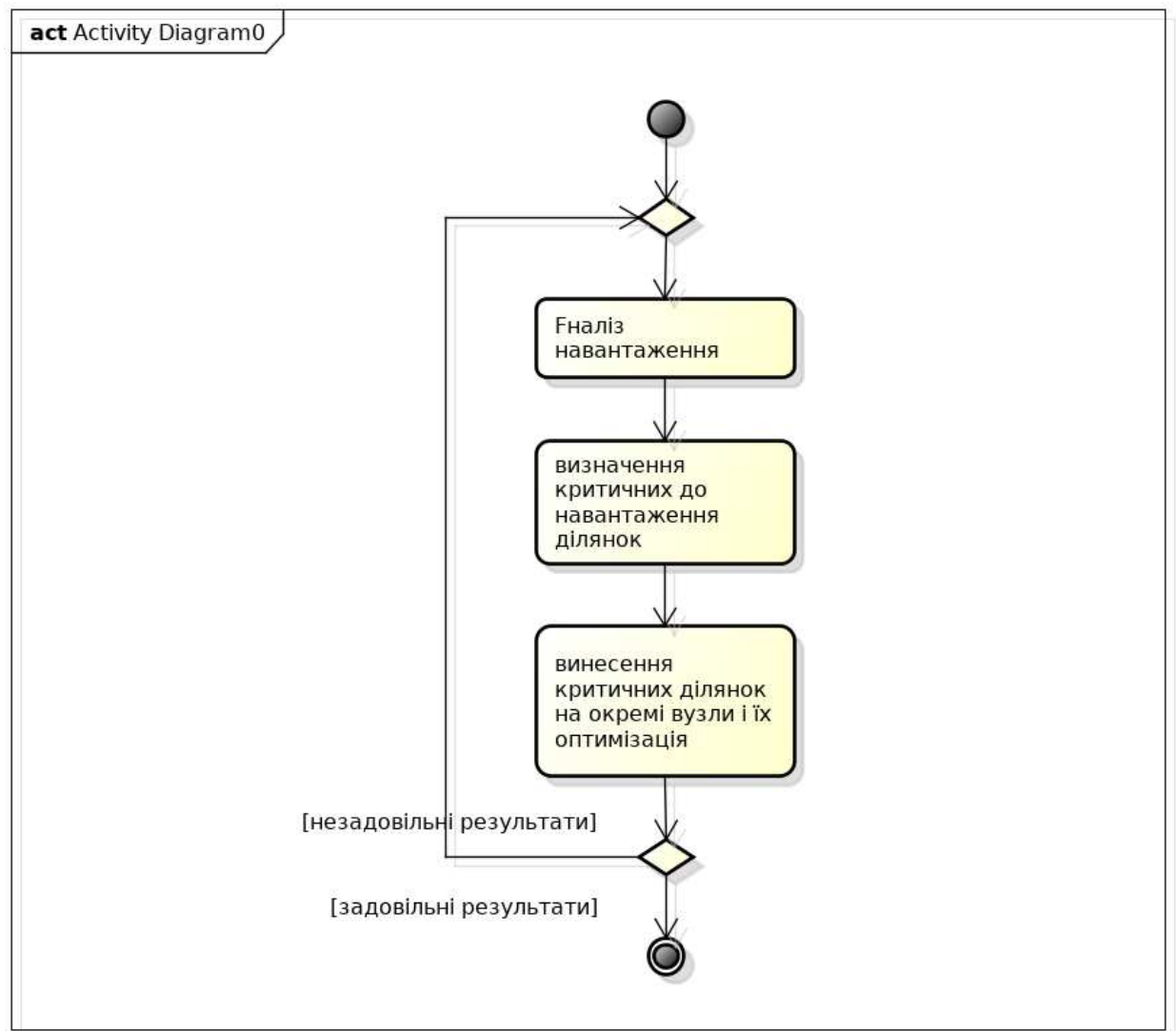
Такий підхід дозволить акцентувати зусилля команди розробників корпоративних систем на важливих аспектах системи, забезпечувати зручності і вирішувати ті проблеми, які виникають у переважної більшості користувачів.

2.1.4. Архітектурні рішення розробки

Масштабування будь-якого додатку з доступом до великих даних може бути реалізована як поступовий ітеративний процес, що включає задачі аналізу навантаження, визначення, ізоляції та переробки найбільш критичних або схильних до навантаження ділянок (Рисунок 2.2). Загалом, масштабування баз даних — одна з найскладніших задач розвитку проєкту [8]. При цьому застосовують такі механізми, як денормалізація, реплікації, шардінг і багато інших.

Проста архітектура такого додатку звичайно така, що кожен окремий новий додаток запускається на одному сервері, на якому працюють і сервер додатків (наприклад, на базі і Web-серверу), і сервер баз даних, і сам додаток (Рисунок 2.3).

Так, у разі використання сервера додатків на базі і Web-серверу, на початковій фазі реалізації проєкту такий підхід є виправданим, оскільки економить час і гроші на запуск. Найчастіше, першим вузлом, який опиняється під навантаженням, є сховище даних. Згодом, при збільшенні навантаження краще виокремити сервери баз даних, що дозволить легко нарощувати потужність підсистеми збереження даних та обробки запитів, збільшити її продуктивність і знизити негативний вплив на інші компоненти (наприклад, Python або PHP-модулі додатків навантажених серверів, Nginx і тому подібних; Рисунок 2.4).



powered by Astah

Рисунок 2.2. Модель масштабування сховища даних корпоративної інформаційної системи

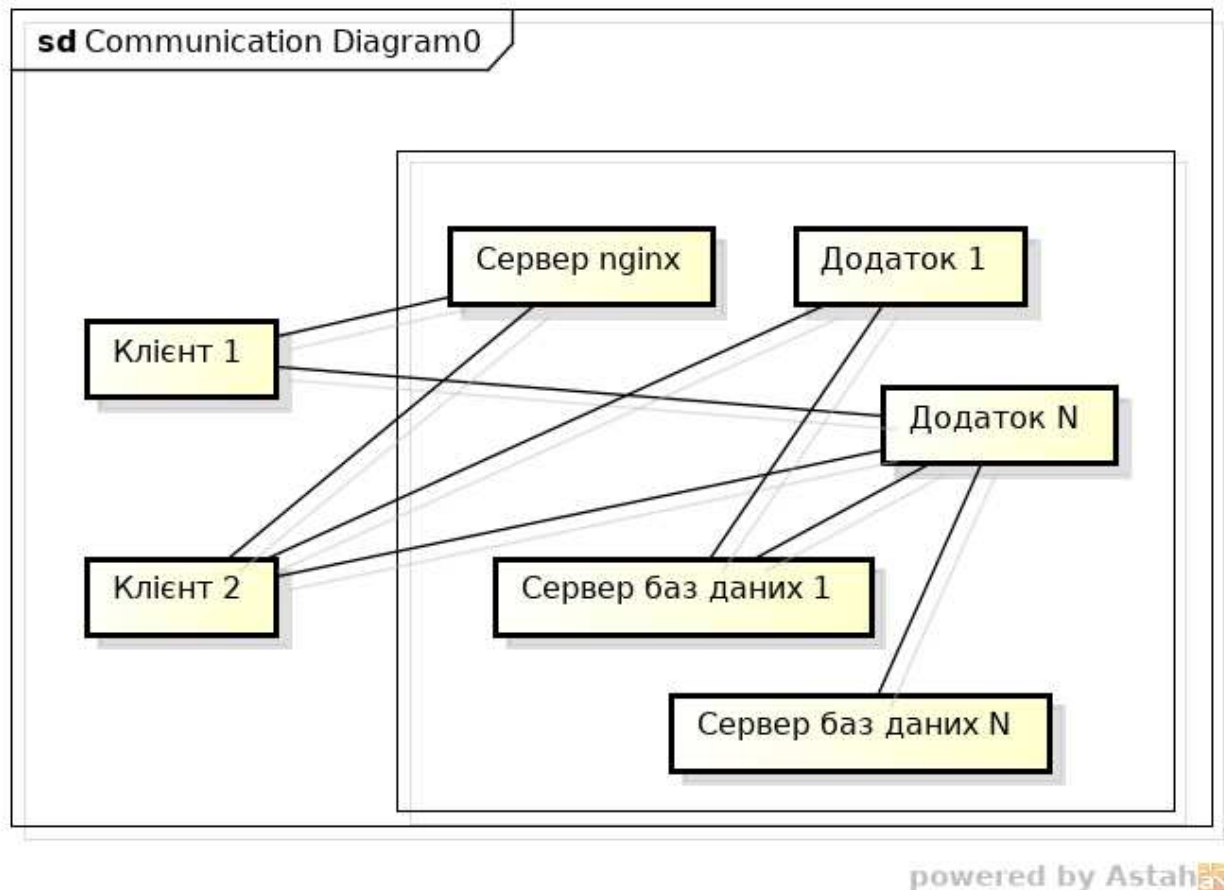


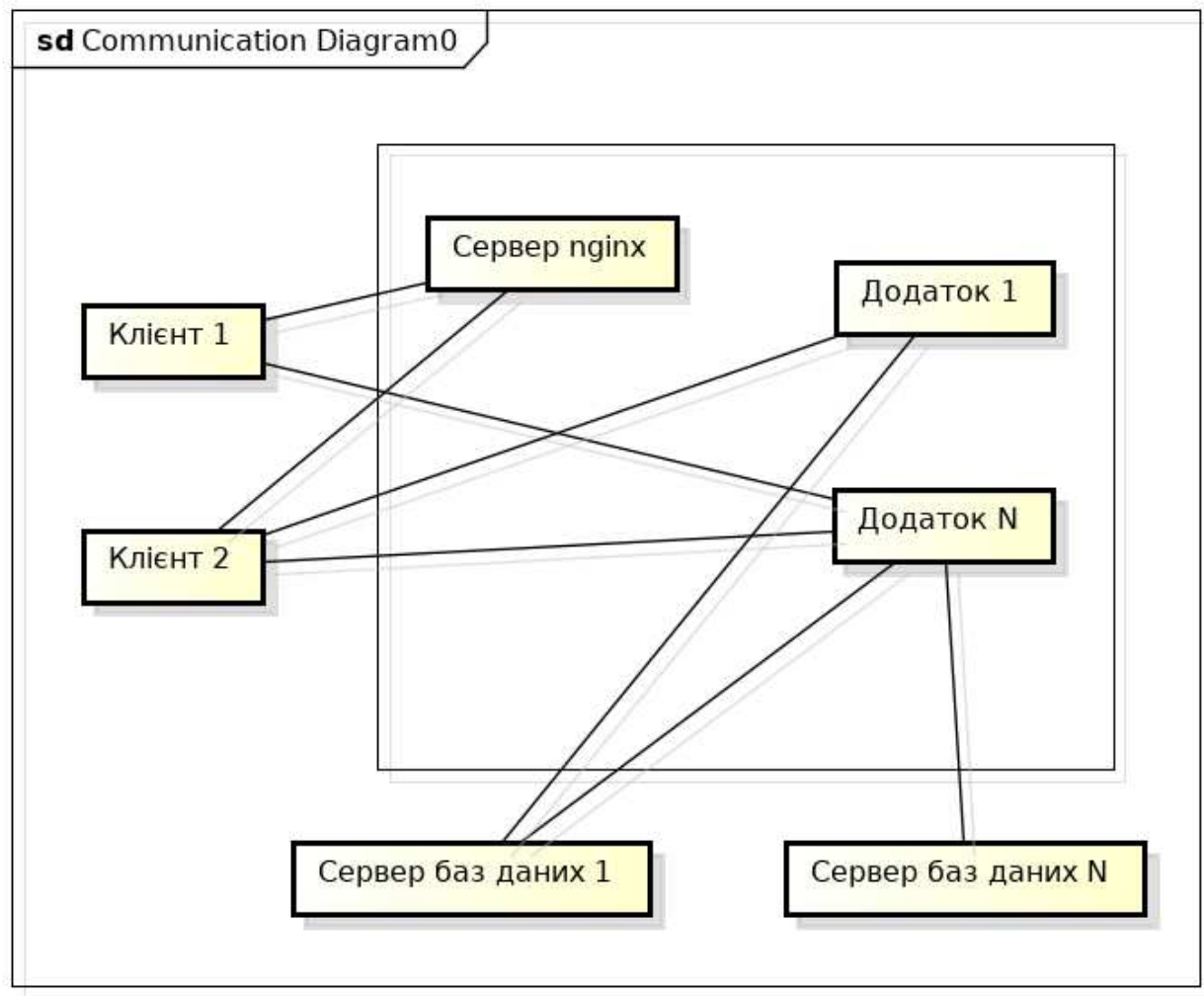
Рисунок 2.3. Класична архітектура ненавантаженого Веб-сервісу

2.1.5. Відділення сервера додатків та пропонована архітектура обробки запитів

Відділення сервера додатків на окремий вузол дозволить залишити більше ресурсів для програм (наприклад, Python або PHP).

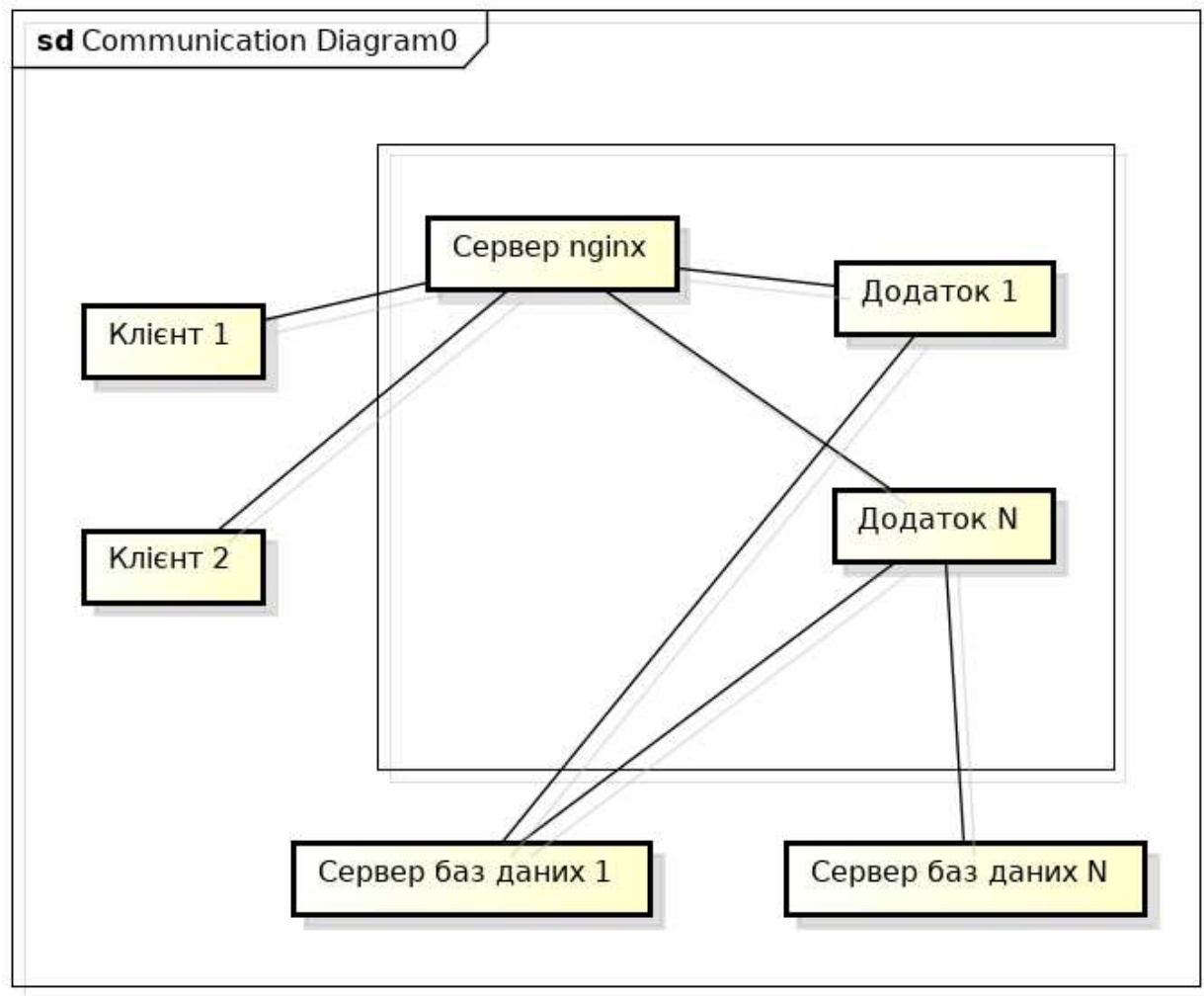
В цьому випадку доведеться налаштувати deployment додатки і на сервер Nginx, і на сервер з Python або PHP. Сервер додатків на основі Python або PHP зазвичай називають бекендом. Тоді Nginx буде віддавати файли статички самостійно, а PHP сервер буде зайнятий лише обробкою скриптів (Рисунок 2.5).

З урахуванням вищенаведеного щодо зберігання та передачі даних може бути запропонована архітектура системи збереження і обробки даних сервера додатків як навантаженого веб-сервісу (Рис 2.6).



powered by Astah

Рисунок 2.4. Архітектура навантаженого сервера додатків на основі веб-сервісу з приєднаним веб-сервісом



powered by Astah

Рисунок 2.5. Архітектура навантаженого сервера додатків веб-сервісу з відділеним веб-сервісом

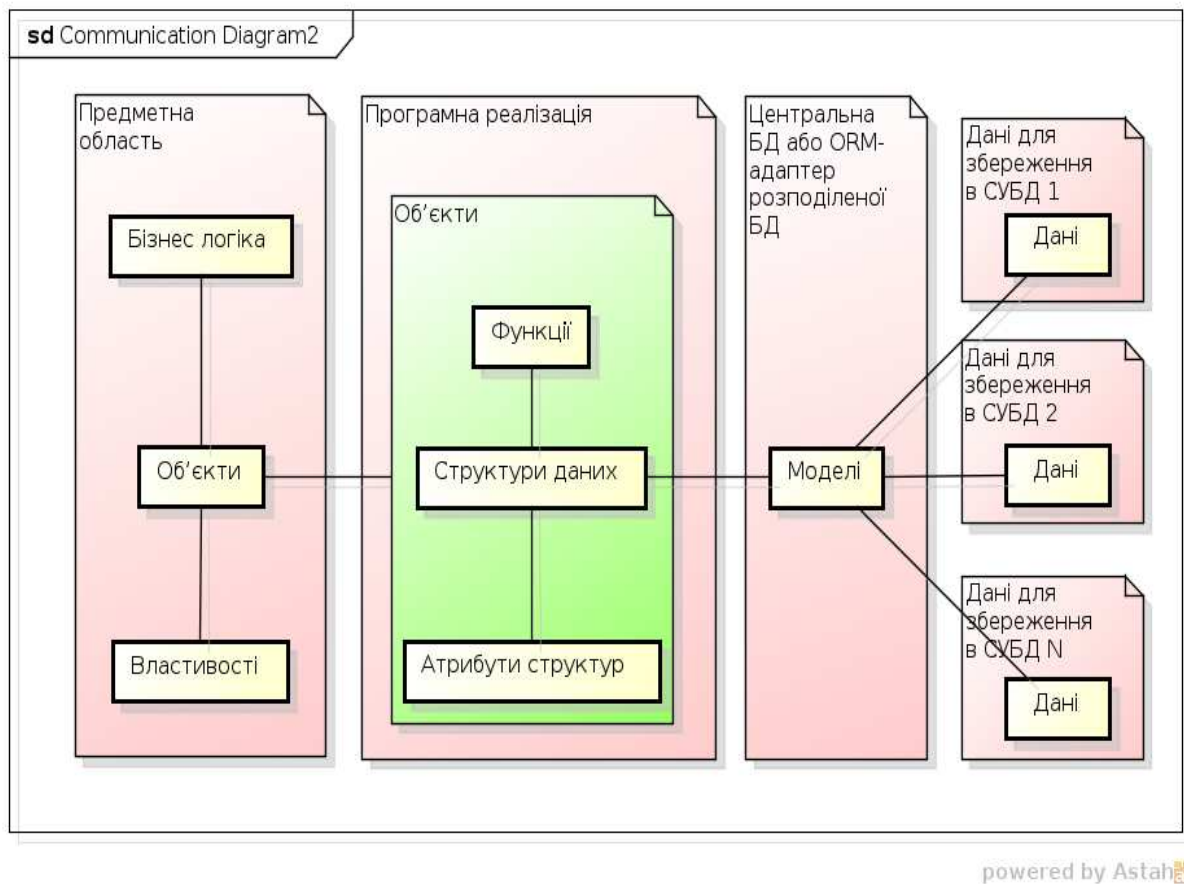


Рис 2.6. Пропонована архітектура системи збереження і обробки даних

2.2. Розробка моделей оцінки якості механізмів збереження та доступу до даних

2.2.1. Критерії оцінки якості технологій збереження та доступу до даних

Враховуючи встановлені в попередньому розділі критерії, що визначають якість механізмів збереження та доступу до даних, визначимо підходи до розробки моделей оцінки якості механізмів збереження та доступу до даних.

Критерій відповідності полягає в тому, що сервіси та протоколи за своїми функціональними можливостями повинні відповідати призначенню проєкту корпоративних систем, наприклад, в комерційних проєктах повинні застосовуватися захищені канали передачі даних, сертифікація, стиснення даних і т.д..

Критерій простоти реорганізації бази даних полягає в тому, щоб кожен окремий запит звертався тільки до тих таблиць, які містять дані, що відносяться до цього запиту. Як наслідок, великі таблиці повинні бути розбиті на кілька

більш простих, можливо однотипних, але маленьких.

Визначимо загальну вартість доступу до даних, що зберігаються на локальному, віддалених локальних та глобальних вузлах збереження даних, а також їх передачі та обробки як:

$$C = k_s * C_s + k_t * C_t + c_c * C_c, \quad (2.1)$$

де: C_s – вартість збереження; C_t – вартість доступу; C_c – вартість обчислень відповідно; k – відповідні коефіцієнти, що визначаються для кожної окремої області використання технологій даних.

Вартість збереження даних:

$$C_s = C_{s,l} + C_{s,u} + C_{s,g}, \quad (2.2)$$

де: $C_{s,l}$ – вартість збереження локальних; $C_{s,u}$ – вартість збереження віддалених локальних; $C_{s,g}$ – вартість збереження глобальних даних відповідно.

Вартість доступу до даних:

$$C_t = C_{t,l} + C_{t,u} + C_{t,g}, \quad (2.3)$$

де: $C_{t,l}$ – вартість збереження локальних; $C_{t,u}$ – вартість збереження віддалених локальних; $C_{t,g}$ – вартість збереження глобальних даних відповідно.

Тоді задача роботи може бути сформульована наступним чином: знайти таке розміщення даних в системі локального, віддаленого локального та глобального збереження даних, для якого значення функції вартості збереження, доступу та обчислення сягне мінімуму.

$$C = k_s * C_s + k_t * C_t + c_c * C_c \rightarrow \min \quad (2.4)$$

Зважаючи на вищенаведене, визначаємо, що відповідність механізмів збереження та доступу до даних корпоративної інформаційної системи залежно від предметної області задач такої системи та інструментальних засобів її розробки (в тому числі мови програмування) щодо можливостей забезпечення збереження та доступу до даних.

2.2.2. Механізми забезпечення якості збереження даних

Серед основних механізмів забезпечення якості в системах збереження даних найбільш цікавим, з точки зору технологій збереження та обробки даних, є масштабування баз даних – партиціонування, реплікація і шардинг.

СКБД часто є «вузьким місцем» в продуктивності додатків корпоративних

систем. У момент, коли сервер баз даних не може впоратися з навантаженнями, здійснюється масштабування. Розглянемо основні способи збільшення продуктивності СКБД.

Розглянуті механізми масштабування, що можуть бути застосовні як для реляційних баз даних, так і для NoSQL-сховищ. Зрозуміло, що у всіх баз даних і сховищ є своя специфіка, тому ми розглянемо тільки основні напрямки і в деталі реалізації вдаватися не будемо.

Партіціонування (partitioning) – це розбиття таблиць, що містять велику кількість записів, на логічні частини за деякими обраним адміністратором критерієм. Партіціонування таблиць ділить весь обсяг операцій з обробки даних на кілька незалежних і паралельно виконуються потоків, що істотно прискорює роботу СКБД. Для правильного конфігурування параметрів партіціонування необхідно, щоб в кожному потоці було приблизно однакову кількість записів.

Наприклад, на новинних сайтах має сенс партіціонувати записи за датою публікації, так як свіжі новини на кілька порядків більш затребувані і частіше потрібна робота саме з ними, а не з усіх архівом за роки існування новинного ресурсу.

Реплікація (replication) – це синхронне або асинхронне копіювання даних між декількома серверами. Провідні сервера називають майстрами (master), а ведені сервера – слейвами (slave). Майстри використовуються для зміни даних, а слейви – для зчитування. У класичній схемі реплікації зазвичай жоден майстер і кілька слейвів, оскільки в більшій частині проєктів корпоративних систем операцій читання на кілька порядків більше, ніж операцій запису. Однак у більш складній схемі реплікації може бути і кілька майстрів.

Наприклад, створення декількох додаткових slave-серверів дозволяє зняти з основного сервера навантаження і підвищити загальну продуктивність системи, а також можна організувати слейви під конкретні ресурсомісткі завдання і таким чином спростити складання серйозних аналітичних звітів – використовуваний для цих цілей slave може бути навантажений на 100%, але на роботу інших користувачів додатку це не вплине.

Шардінг (sharding) – це прийом, який дозволяє розподіляти дані між різними фізичними серверами. Процес шардінга припускає рознесення даних

між окремими Шарден на основі якогось ключа шардінга. Пов'язані однаковим значенням ключа шардінга суті групуються в набір даних по заданому ключу, а цей набір зберігається в межах однієї фізичної Шарда. Це істотно полегшує обробку даних. Наприклад, в системах типу соціальних мереж ключем для шардінга може бути ID користувача, таким чином всі дані користувача будуть зберігатися та оброблятися на одному сервері, а не збиратися по частинах з декількох джерел.

Крім цього, істотним з точки зору задоволення критеріям оцінки якості, є механізми зберігання та індексування даних в БД.

Серед популярних механізмів зберігання даних найбільш цікавим для реалізації вузькоспеціалізованих і високонавантажених серверів БД, до яких відносяться БД для збереження даних , представляються In-memory СКБД реального часу як кеш для комерційних СКБД.

Практично всі універсальні комерційні СКБД сьогодні – це СКБД, орієнтовані на роботу з дисками. Така архітектура, навіть при добре налаштованих додатках, може дати час відгуку порядку декількох мілісекунд. Однак для багатьох додатків (наприклад, для систем, що працюють в реальному часі) потрібна швидкодія на порядок вище (час відгуку – мікросекунди). Для отримання такої швидкодії використовуються спеціальні СКБД, що працюють в оперативній пам'яті (In memory DB) і мають спеціальну архітектуру, спеціальні способи зберігання, адресації і індексації даних, спеціальні механізми оптимізації і буферизації і т.д. Дані на диск вони не скидають або роблять це у фоновому асинхронному режимі. Надійність роботи та обробка збоїв реалізується спеціальними методами (наприклад, за рахунок реплікації пам'ять – пам'ять). Окремі функції таких СКБД можна відключати (наприклад, журналювання) на догоду швидкодії.

Зараз намітилася тенденція використання таких швидких in memory СКБД в якості високошвидкісних кешів до комерційних дискових СКБД. Oracle використовує в такій якості СКБД Times Ten, IBM придбала СКБД SolidDB і почала її інтеграцію з DB2. Спеціальне ПЗ дозволяє підняти або підкачувати дані в такий швидкісний кеш і синхронізувати зміни в дисковій СКБД і в кеші. Додатки, що вимагають високої швидкодії і малого гарантованого часу відгуку,

практично працюють з таким кешем в пам'яті, який вже сам синхронізується з дисковою СКБД. Таких кешів над дисковою СКБД можна змонтувати кілька, даючи можливість багатьом додаткам отримувати час відгуку (особливо, при читанні і коротких оновленнях) в кілька мікросекунд. При цьому розміри основної БД можуть бути дуже великими (десятки і сотні терабайт), що відповідає вимогам даних, але кешується за певною політикою лише частина даних. Важливо, що для роботи з використанням такого кеша не потрібне спеціальне програмування – можна використовувати стандартні протоколи (ODBC, JDBC) та звичну мову доступу до дискової СКБД.

2.2.3. Механізми забезпечення якості доступу до даних

З точки зору оцінки якості доступу до даних в складних корпоративних системах перевагу слід віддавати тим рішенням, які, насамперед, дозволяють реалізувати необхідну бізнес логіку (так як в іншому випадку вже не важливі такі характеристики, як швидкість, захищеність і т.д.). Виходячи зі складності таких систем, для їх розробки застосовують, як правило, об'єктно-орієнтовані технології проектування та програмування. При цьому сховища даних також повинні бути об'єктними. Враховуючи, що реляційні СКБД все ще популярні і широко використовуються рішенням задачі об'єктного доступу до даних в реляційному сховищі є ORM-системи (Рисунок 2.7).

Для забезпечення якості програміста ORM-система повинна виглядати як постійне сховище об'єктів. На практиці все ORM-системи, як правило, виявляють себе саме таким чином, забезпечуючи можливість ігнорування реляційної природи бази даних як набору таблиць і відносин між ними.

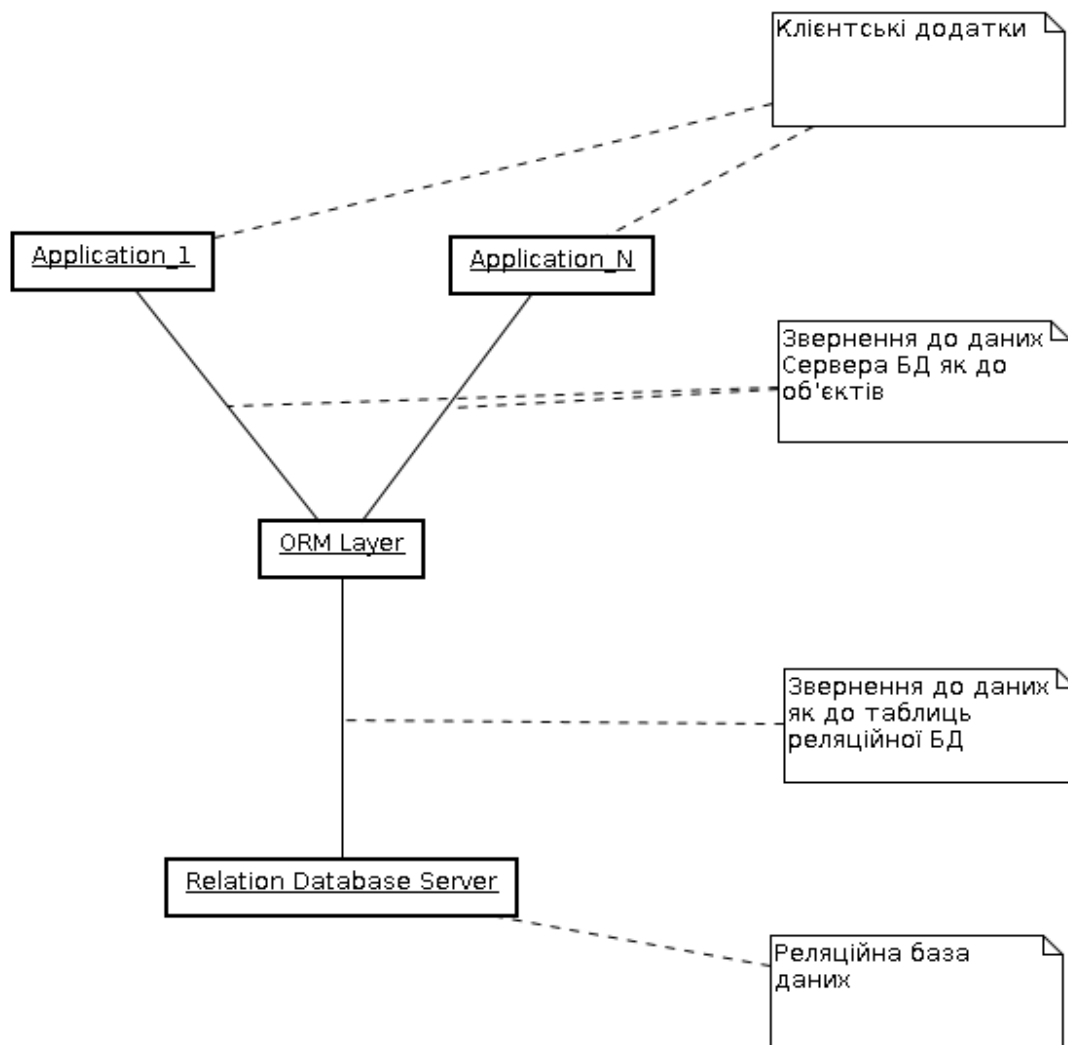


Рисунок 2.7. Компонентна модель доступу до реляційної бази даних з Object Relational Mapping (ORM)

Враховуючи, що програмне забезпечення для складних систем зазвичай розробляється відповідно до принципів об'єктно-орієнтованого програмування, необхідність використання ORM-систем так само виникає як результат невідповідності способів організації структурованих даних систем управління реляційними базами даних способам організації об'єктів як структурованих даних і механізмів їх обробки в програмах, які використовують такі бази даних. Ця невідповідність полягає в тому, що об'єкти програми можуть перебувати у відносинах агрегації, асоціації та використання, а реляційні бази даних підтримують тільки відносини асоціації, причому, відносини асоціації "багато-до-багатьох" більшістю реляційних баз даних взагалі не підтримується.

Захист клієнта від можливості отримання недостовірних даних або втрати

власних базуються на наступних механізмах:

- захисті каналів передачі даних та захисті від потрапляння на клієнтську машину недостовірних даних, що має місце при підміні сервера, коли за тією самою адресою працює інший, недостовірний сервер (який може видавати себе за достовірний шляхом надання очікуваного контенту) або в результаті автоматичного перенаправлення запиту;
- впровадження в сторінку шкідливого коду, який буде виконаний на комп'ютері користувача при відкритті ним цієї сторінки (XSS-атака).

Найбільш часто експлуатуються такі уразливості в корпоративних системах, де користувачі можуть публікувати свої дані, роблячи їх доступними для інших. Зловмисник, при відсутності належної фільтрації, може або безпосередньо використовувати уразливість, або заражати системи користувачів. Розробники корпоративних проєктів для запобігання атак цього типу змушені або просто обмежувати можливості користувачів публікувати свої дані, або ретельно фільтрувати дані, що надходять від користувачів.

2.3. Розробка моделей вибору методів та засобів збереження та доступу до даних

Найпростіший спосіб введення таких механізмів полягає у використанні системи вагових коефіцієнтів з можливістю приведення оцінок за критеріями до скаляра для подальшого співставлення з інструментами, які дозволять найкращим чином досягти визначених критеріїв.

Очевидно, що для кожної з предметних областей значення цих вагових коефіцієнтів будуть відрізнятись (що було показано в п. 1.4.2). В якості прикладу розглянемо проєкти з двох предметних областей (табл. 2.3).

Таблиця 2.3. Основні параметри ІС

Критерій	Навчання	Соціальні мережі
Збереження та надання інформаційних ресурсів	0.5	0.2
Комунікації та обмін інформацією	0.6	0.8
Масові обчислення	0.4	0.2

Також визначимо оцінки для інструментальних засобів розробки різними мовами програмування щодо можливостей забезпечення зазначених вище критеріїв (табл. 2.4, для прикладу для мов C/C++ та Python).

Таблиця 2.4. Основні параметри інструментальних засобів щодо забезпечення властивостей системи

Критерій	C/C++	Python
Збереження та надання інформаційних ресурсів	0.5	0.5
Комунікації та обмін інформацією	0.6	0.6
Масові обчислення	0.9	0.2

Також визначимо якості кожного з інструментальних засобів розробки стосовно розробників системи (табл. 2.5, для прикладу для мов C/C++ та Python).

2.4. Вибір інструментального забезпечення розробки проєктів корпоративних систем

Враховуючи результати аналізу, наведені в розділі 1, можна стверджувати, що розробку проєктів корпоративних систем на поточний момент доцільно виконувати на одній з наступних мов: PHP, Python, Perl, Ruby та Java. Кожна з цих мов має розвинуті інструментальні засоби (фреймворки), написані цими ж мовами.

Таблиця 2.5. Основні параметри інструментальних засобів щодо складності розробки

Критерій	C/C++	Python
Захищеність програмного забезпечення	0.9	0.1
Швидкість розробки ПЗ проєкту	0.3	0.8
Наочність коду	0.4	0.9
Швидкість обробки запитів	0.8	0.5
Валідність коду	0.6	0.7

Хоча кожна з зазначених мов програмування має свої особливості, їх спільні переваги з погляду наявності фреймворків з підтримкою технологій програмування проєктів корпоративних систем наступні [14]:

- Висока швидкість розробки. Це обумовлено як технічними особливостями наявних фреймворків (наприклад, продумані контракти між компонентами фреймворку спрощують конфігурацію програмного продукту, що базується на фреймворку), так і інструментами розробки (консольні утиліти валідації, генератори коду, наявні модулі і бібліотеки, протоколи тощо).
- Простота реалізації складної бізнес-логіки – представлені у фреймворках конвенції оформлення програмного коду дозволяють створювати одноманітний, а отже і зрозумілий програмний код, який згодом може бути легко модифікований, а дотримання цих конвенцій робить програмний код супроводжуваним не тільки початковим розробником, але і іншим спеціалістом або іншою командою.
- Висока надійність і можливості супроводу рішень – в таких мовах і базованих на них фреймворках зазвичай використовується TDD-підхід (TDD – робити речі правильно, на відміну від BDD – робити правильні речі), що робить створювані продукти більш стабільними і супроводжуваними. Функціональність фреймворків також звичайно покрита автоматичними тестами, що робить їх використання дійсно надійним – є впевненість, що функціональ. Для бізнес-проєктів ця складова вкрай важлива – від стабільності роботи додатка часто залежить ефективність роботи бізнесу в цілому.
- Масштабованість, продуктивність і високі навантаження – фреймворки звичайно «заточені» під розробку додатків, до яких пред'являються високі вимоги щодо доступності: окремі фреймворки відмінно розгортаються і працюють в кластерах серверів або в «хмарах». Для сервісів корпоративних систем це дуже важливий критерій, оскільки для власної ефективності і для привабливості в очах користувача вони повинні мати високі швидкості розгортання і роботи.
- Компетентні розробники. Частка хороших програмістів на цих мовах набагато вища, ніж на інших мовах. Це є наслідком їх високого рівня,

значного поширення та великої кількості проєктів корпоративних систем. Чим компетентніше розробники, тим швидше (дешевше) розробляється проєкт і тим він більш продуманий, масштабується і надійний [15, 16].

2.5. Розробка моделей та механізмів інтелектуального розподілу даних між серверами даних

З метою забезпечення можливостей збереження та обробки даних за умов та обмежень, що були викладені вище, розробимо моделі та механізми підтримки розподілу даних між локальними, зовнішніми локальними та глобальними базами даних шляхом динамічного перерозподілу даних між сховищами даних та, можливо, динамічної реконфігурації баз даних, для мінімізації загальної вартості збереження, отримання та обробки даних в системі.

Загалом, такі операції можуть бути запрограмовані жорстко або виконуватись на основі результатів аналізу даних. Зважаючи, що обсяги даних значні і постійно змінюються, а задачі обробки даних ніколи до кінця не визначені (через що, власне і з'явилися технології даних) рішення про реконфігурацію сховищ даних при збереженні і обробці даних має прийматись виключно на основі результатів інтелектуального аналізу. При цьому система, яка в процесі свого функціонування самостійно змінює свою структуру та алгоритми функціонування є адаптивною.

Інтелектуальний аналіз даних являє собою процес виявлення придатних до використання відомостей в великих наборах даних.

Адаптивна система (система, що сама пристосовується) – система, що автоматично змінює алгоритми свого функціонування, а іноді і свою структуру з метою збереження або досягнення оптимального стану при зміні зовнішніх умов.

Механізм реконфігурації може бути реалізований шляхом обчислення числа операцій по доступу до локальних та віддалених даних або їх атрибутів та їх вартості для системи (наприклад, визначеній через час виконання операції). В такому разі на основі математичних моделей (2.1 – 2.4) може бути здійснена реконфігурація структур даних та самих розподілених баз даних шляхом переміщення окремих таблиць або атрибутів між локальними та віддаленими сховищами. Спрощений алгоритм реконфігурації представлений на Рисунок 2.8.

Спрощений алгоритм реконфігурації окремої бази даних представлений на Рисунок 2.9.

Обчислення та нормалізація вагових коефіцієнтів (рішень) можуть бути виконані за алгоритмом, представленим на рисунку 2.8.

На основі вищенаведеного може бути розроблено спрощену систему визначення вагових коефіцієнтів відповідно до властивостей інструментального забезпечення проєкту (Рисунок 2.10).

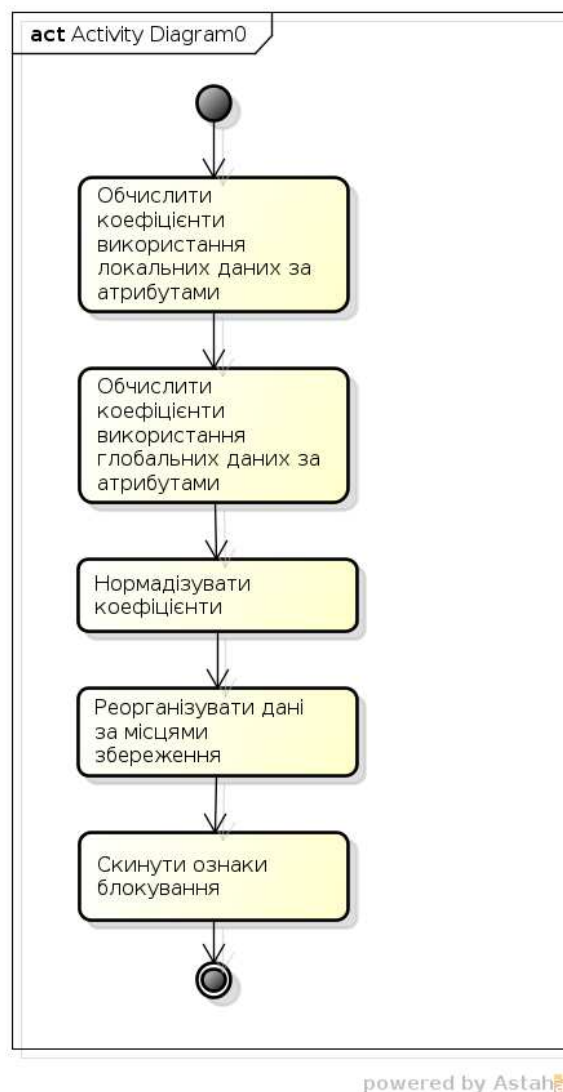


Рисунок 2.8. Алгоритм реконфігурації моделей даних в розподіленій базі даних корпоративної системи

2.5. Висновки до розділу

В розділі були розглянуті технології організації і доступу до даних у проєктах корпоративних систем, що визначаються інструментальним забезпеченням, методами та засобами збереження, транспортування, захисту та обробки даних проєктів корпоративних систем.

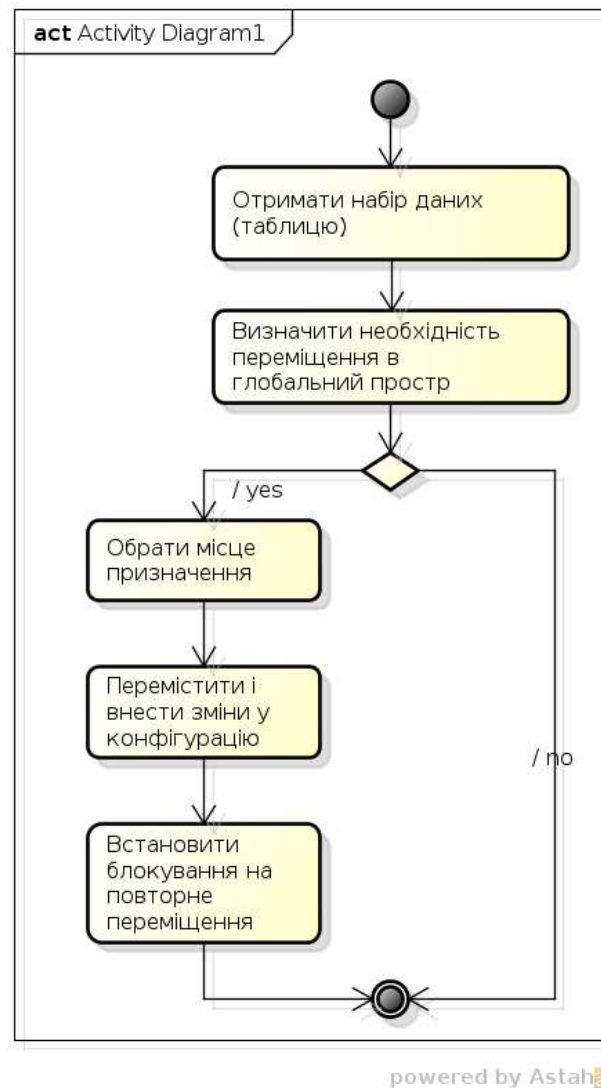
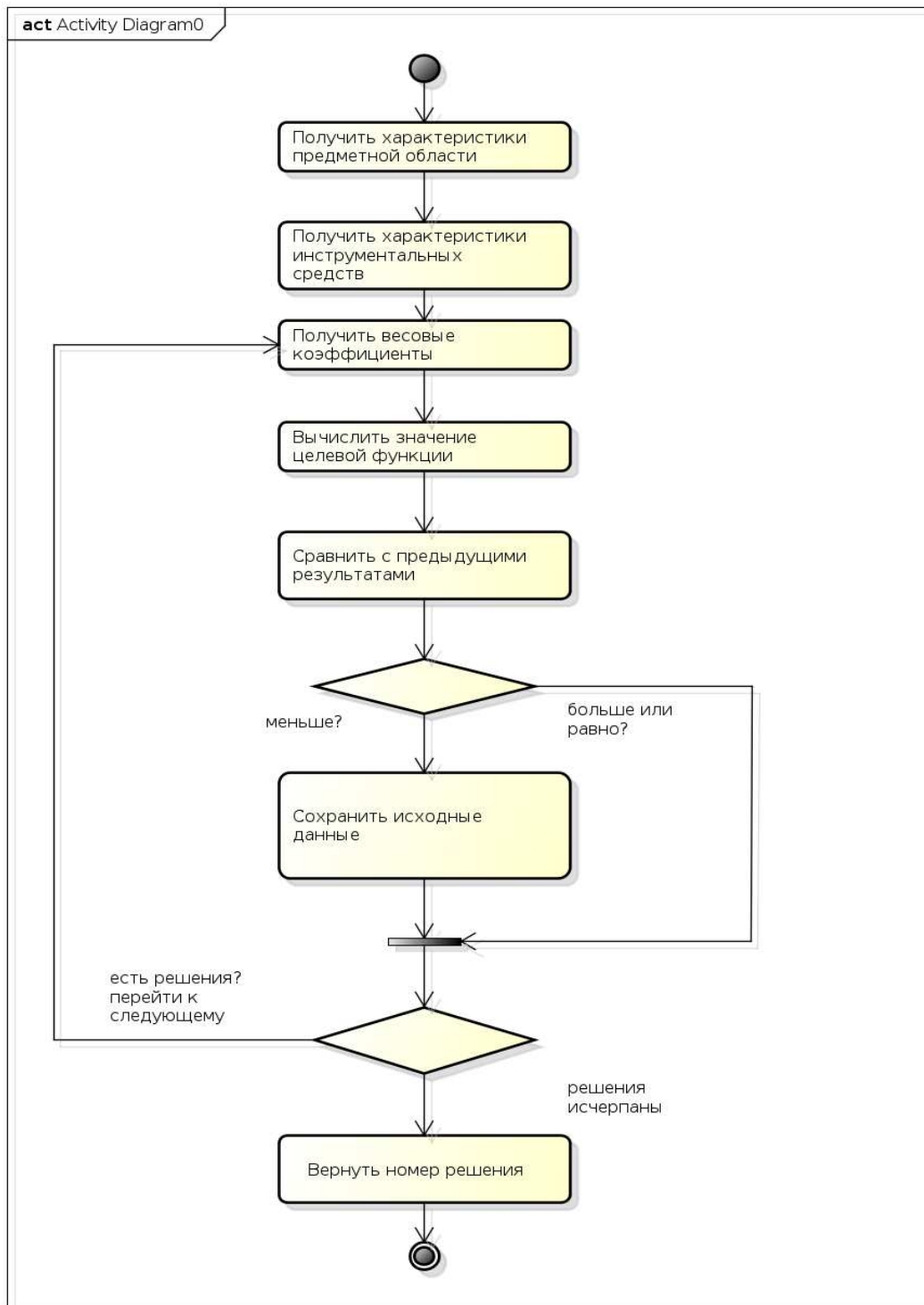


Рисунок 2.9. Алгоритм реконфігурації локальної бази даних в розподіленій базі даних корпоративної системи

Виконано розробку математичної моделі обчислення відповідності технологій проєкту щодо можливостей його реалізації в даній предметній області та можливостей забезпечення необхідних користувацьких якостей.

Розроблено спрощену систему визначення вагових коефіцієнтів відповідно до властивостей інструментального забезпечення проєкту.



powered by Astah

Рисунок 2.10. Алгоритм обчислення вагових коефіцієнтів функції реконфігурації методом перебору

3. ВПРОВАДЖЕННЯ ПІДСИСТЕМИ ЗБЕРЕЖЕННЯ ДАНИХ

3.1. Реалізація механізмів оцінки якості інструментального забезпечення створення підсистеми

З метою впровадження технологій управління даними в корпоративних інформаційних системах та розробка були реалізовані модулі обчислення значення цільової функції (Рисунок 2.5), яке виконується методом перебору. Реалізація виконана мовою програмування Python (Додаток В).

В більш складному випадку можуть бути застосовані методи вирішення оптимізаційних задач (лінійного програмування, сімплекс методи, еволюційні моделі тощо).

На основі розрахунків, проведених на основі розробленого алгоритму (Додаток В) визначено, що корпоративних системах з даними доцільно використовувати в якості глобальних сховищ хмаро-орієнтовані сховища, зокрема Google Drive, Microsoft OneDrive, Dropbox та подібні.

Результати досліджень були застосовані в окремих модулях проєкту автоматизованої системи управління комунальним господарством. З метою забезпечення можливостей горизонтальної інтеграції даних для декількох підсистем автоматизованої системи між собою і вертикальної інтеграції системи в інформаційну систему комунального господарства, у проєкті автоматизованої системи повинні бути передбачені відповідні складності і особливостям вирішуваних завдань механізми та засоби інтеграції підсистем інформаційної системи в єдину систему.

Забезпечення можливостей горизонтальної інтеграції підсистем між собою повинна передбачати такі інструменти реалізації системи, які дозволять досить просто нарощувати функціональність системи шляхом розширення інформаційної бази системи і введення в систему нових модулів. Вертикальна інтеграція системи в інформаційну систему міста повинна передбачати забезпечення таких можливостей доступу до системи з боку зацікавлених осіб і не повинні вимагати істотної модифікації їх інформаційних систем. Це завдання може бути вирішена шляхом надання віддаленого доступу до ресурсів системи для інших клієнтів системи через локальну мережу або Інтернет із застосуванням поширених і загальнодоступних комунікаційних

протоколів для передачі даних і управління між підсистемами.

Локальний доступ може забезпечуватися через автоматизовані робочі місця (АРМ), встановлені на клієнтських робочих місцях персоналу підприємства. Для цього на клієнтські робочі місця повинно бути встановлено відповідне функціям клієнтів клієнтське програмне забезпечення, а клієнтам повинні бути надані права і повноваження по доступу до інформаційних ресурсів сервера інформаційної системи відповідно до функціональних обов'язків кожного клієнта системи.

Враховуючи, що в кожній інформаційній системі накопичується інформація про параметри інформаційної системи [14, 15], яка може використовуватися як в поточній, так і в інших системах, необхідно розглянути механізми доступу до інформаційних ресурсів системи з боку інших систем. Для цього може використовуватися глобальний (віддалений) доступ через мережу Інтернет, у тому числі – із застосуванням протоколів TCP / IP і HTTP.

У результаті забезпечення зазначених механізмів доступу, доступ до інформаційних ресурсів системи може бути наданий для наступних видів користувачів системи:

- авторизованих користувачів – менеджерів підприємства, команд управління проектами (надання прав і повноважень доступу до інформаційних ресурсів системи повинне буде забезпечуватися як для режиму локального доступу);
- авторизованих користувачів – співробітників розрахункових центрів комунальних служб водопостачання (авторизація виконується на основі відомої користувачам приватної облікової інформації: адреса, номер розрахункової книжки тощо);
- інших користувачів – інформація надається без авторизації (оголошення, таблиці-графіки ремонтних робіт, параметри води і т.п.).

3.2. Висновки до розділу 3

В розділі було запропоновано спрощений алгоритм пошуку рішення проектування навантаженого веб-сервісу на основі цільової функції оцінки якості, розглянуті можливості її застосування щодо проектів з декількох

предметних областей.

Розроблені пропозиції щодо використання інструментальних засобів реалізації підсистеми сховища даних корпоративної інформаційної системи (у тому числі з доступом через інтернет).

Загальні висновки по роботі

В роботі в цілому вирішені поставлені задачі, що дозволило досягти основної мети роботи: розроблені моделі вибору інформаційних технологій організації і збереження даних у підсистемі сховища даних корпоративної інформаційної системи, що дозволило підвищити ефективність процесів збереження, доступу та обробки даних.

Запропоновані рішення можуть бути застосовані при вирішенні задач реорганізації розподілених сховищ даних в системах збору і обробки великих наборів даних.

Список використаних джерел

1. Вайсман, В.О. Моделі, методи та механізми створення і функціонування проєктно-керованої організації: Монографія [Текст] / В.О. Вайсман – К. : Наук. світ, 2009. – 146 с.
2. Гайна, Г.А. Основи проєктування баз даних / Навчальний посібник – К. : Кондор, 2024. – 280 с.
3. ДСТУ 3008:2015 – “Інформація та документація. Звіти у сфері науки і техніки. Структура та правила оформлювання”
4. ДСТУ ISO/IEC 15910:2012 "Інформаційні технології. Документація для програмного забезпечення, облаштованого інтерфейсом користувача"
5. Заверач, М.М. Базы даних. Інформаційні системи [Текст]: навч. посіб. / М. М. Заверач,, В. В Третько. – Хмельницький : ХНУ, 2007, – 303 с.
6. Зарицька О. Л. Базы даних та інформаційні системи: методичний посібник. – Житомир: Вид-во ЖДУ ім. І. Франка, 2010. – 132 с.
7. Організація реляційних баз даних : навч. посіб. / Я.Г. Куваєв, О.А. Жукова, І.А. Сечкін – 2-ге вид., допов. та переробл. – Дніпро : НГУ, 2017. – 157 с.
8. Пасічник, В.В., Литвин, В.В., Шаховська, Н.Б. Проєктування інформаційних систем. Навчальний посібник (затв. МОН України) – Львів: 2013. – 380 с.
9. Розвиток ринку e-commerce у світі та Україні. Теплосила & e-com / Електронний ресурс, URL: <https://teplosila.ua/e-commerce-blog/>
- 10.Томашевський О.М. Інформаційні технології та моделювання бізнес-процесів / Навчальний посібник / О.М.Томашевський, Цегелик М.Б., Вітер Г.Г., В.І.Дубук. К.: Центр учбової літератури, 2005. – 296 с.
- 11.Цибко, Г. Ю., Рамський, Ю. С. Проєктування й опрацювання баз даних : Посібник для вчителів. 2-ге вид., перероб. і доп. Тернопіль : Навчальна книга. Богдан, 2005.
- 12.ODOO: Управління базами даних – ODOO 15.0 documentation / Електронний ресурс, URL: <https://www.odoo.com/documentation/administration>
- 13.ODOO: Как создать базу данных - ODOO development - Read the Docs /

Электронный ресурс, URL:

<https://odoo-development.readthedocs.io/ru/latest/odoo/usage/create-database.html>

14. Agile Web Development with Rails Third Edition Sam Ruby, Dave Thomas, David Heinemeier Hansson, The Pragmatic Programmers LLC, Printed in the United States of America printing, June 4, 2008 printing, June 4, 2008.
15. Bart Voorn, Marieke L. van Genugten, Sandra van Thiel. The efficiency and effectiveness of municipally owned corporations: a systematic review // Электронный ресурс. URL:
<https://www.tandfonline.com/doi/full/10.1080/03003930.2017.1319360>
16. Martin Fowler. Patterns of Enterprise Application Architecture. Addison Wesley Longman, Reading, MA, 2003.

ТЕХНІЧНЕ ЗАВДАННЯ НА РОЗРОБКУ ПРОЄКТУ ПІДСИСТЕМИ ЗБЕРЕЖЕННЯ ДАНИХ КОРПОРАТИВНОЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ

А.1. Призначення розробки

Назва – «Розроблення підсистеми збереження даних корпоративної інформаційної системи».

Розроблювана автоматизована інформаційна система призначена для збереження і надання інформації в корпоративній інформаційній системі. Функціональне призначення системи – підвищити швидкість та якість роботи з корпоративними даними шляхом забезпечення відповідної задачі структури сховища даних.

А.2. Вимоги до підсистеми автоматизованої інформаційної системи

Для забезпечення ефективної роботи підсистеми корпоративної інформаційної системи програмне забезпечення повинне мати зрозумілий користувачеві інтерфейс, зрозуміле призначення функціональних частин інтерфейсу і наочний результат роботи підсистеми; надійно працювати в умовах експлуатації, володіти можливостями захисту від збоїв і ушкодження даних.

Підсистема повинна бути відкритою для подальшого вдосконалення й збільшення функціональних можливостей.

А.3. Вимоги до функціональності системи

Інформаційна система повинна бути побудована у вигляді взаємозалежних модулів, де кожний модуль відповідає за своє обмежене завдання, що полегшить роботу з даними і алгоритмами.

Вихідною інформацією підсистеми має бути:

- Інформація про структуру даних у сховищі даних;
- Статистичні дані щодо виконання операцій доступу до даних: місце розташування, обсяг, задача, час виконання операції доступу до даних, час

обробки даних.

А.4. Вимоги до надійності й безпеки

ПЗ підсистеми повинне забезпечувати надійну роботу з даними. Необхідна можливість збереження резервних копій даних на випадки непередбачених збоїв в апаратному або програмному забезпеченні комп'ютерної системи, на базі якої буде працювати програма.

Надійність технічних засобів системи має підтримуватись технічним обслуговуванням (виконання профілактичних робіт і ремонту) технічних засобів.

А.5. Умови експлуатації

Перед початком роботи з даним програмним продуктом необхідно ознайомитись з інструкцією використання програмного продукту.

Умови експлуатації підсистеми повинні відповідати умовам експлуатації апаратного забезпечення. Умови експлуатації носіїв інформації повинні відповідати занесеним у їхню супровідну документацію вимогам (споживана потужність, допуски по напрузі живлення, інтерфейси підключення, температура, вологість).

А.6. Вимоги до технічних засобів

Для установки й експлуатації програмного забезпечення підсистеми необхідний персональний комп'ютер, що володіє системними характеристиками не гіршими, ніж:

Процесор: не менше 6 ядер 4 ГГц.

Пам'ять: не менше 8 Гб;

Відеокарта: не менше 64 Mb;

Маніпулятори: миша, клавіатура.

А.7. Вимоги до інформаційної й програмної сумісності

Для роботи програмного забезпечення системи необхідно:

- Інтерпретатор Python версії 3.10 або вище;

- СУБД PostgreSQL версії 11 або вище або MariaDB версії 5 або вище.

A.8. Вимоги до програмної документації

Програмна документація повинна бути розроблена в наступному складі:

- технічне завдання;
- інструкція користувача.

A.9. Стадії та етапи розробки

Стадії та етапи розробки з потрібними строками вказані в таблиці А.1.

Таблиця А.1 – Стадії та етапи розробки

Стадії	Етапи роботи	Початок робіт	Закінчення робіт
1. Ескізний проект	Розробка ескізного проекту	13.03.2025	04.03.2025
	Затвердження ескізного проекту	04.04.2025	28.04.2025
2. Технічний проект	Розробка технічного проекту	28.04.2025	12.05.2025
	Затвердження технічного проекту	12.05.2025	17.05.2025
3. Робочий проект	Розробка програмного забезпечення підсистеми	17.05.2025	23.05.2025
	Розробка програмної документації	23.05.2025	01.06.2025
	Випробування програмного забезпечення підсистеми	01.06.2025	14.06.2025

A.10. Порядок контролю та приймання

Для контролю і приймання повинен бути наданий опис програмного забезпечення підсистеми, а також програма і методика випробування.

Порядок контролю і прийому даної розробки здійснюється представником замовника у присутності представника розробника згідно з програмою та

методикою випробувань.

Якщо програмне забезпечення підсистеми не пройшло випробування, виконавець зобов'язаний виправити помилки та недоліки у строк не більш ніж 1 місяць зі дня випробування.

За результатами приймання складається акт, який підписується представником замовника та представником розробника та утверджується керівниками організації-замовника і організації-розробника.

У випадку виявлення помилок під час прийому програмного виробу складається акт про виявлені помилки, який підписується представниками замовника та розробника й стверджується керівниками організації–замовника й організації–розробника. Розробник повинен впродовж не більш як 1 місяця видалити вказані зауваження і повідомити Замовника про повторне проведення перевірки не пізніше ніж за 2 тижні до початку приймання підсистеми.

Додаток Б. Статистика використання мов програмування в проектах в Україні (дані сервера ІТ-вакансій на початок 2014 року)



Рисунок Б.1. Статистика використання мов програмування за корпоративними проектами

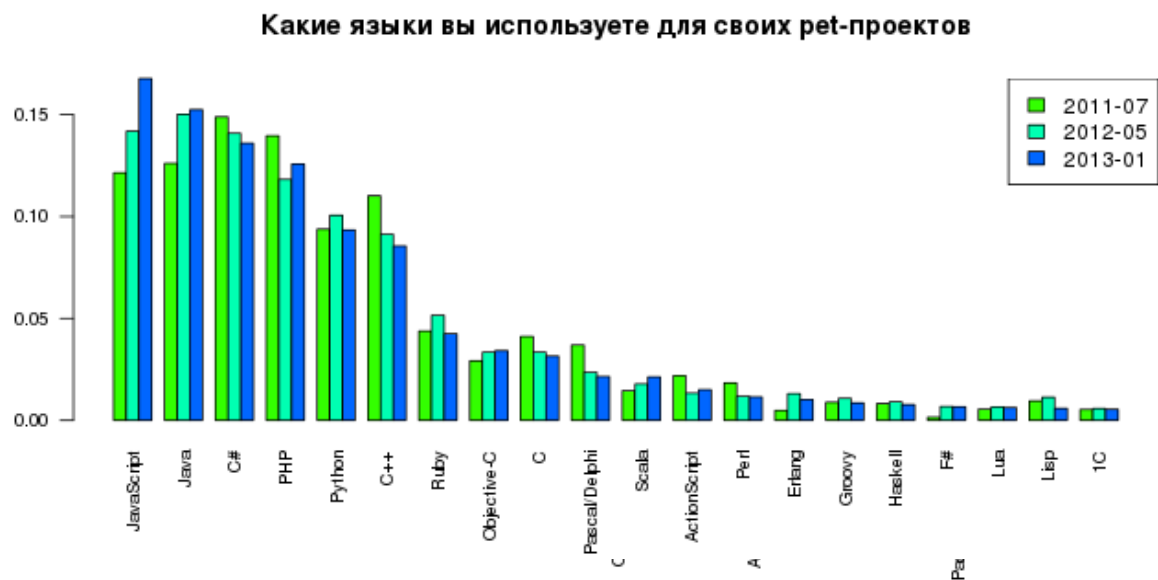


Рисунок Б.2. Статистика використання мов програмування за власними проектами

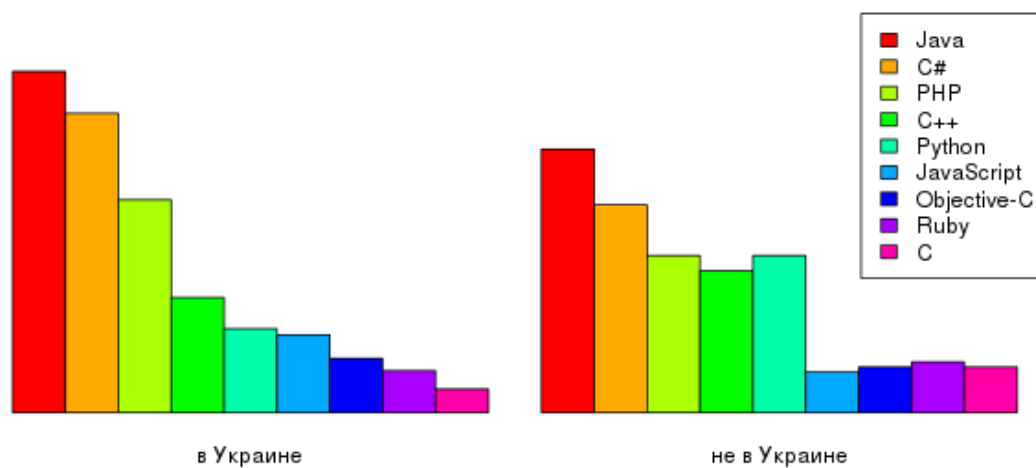


Рисунок Б.4. Статистика використання мов програмування в Україні та за її межами

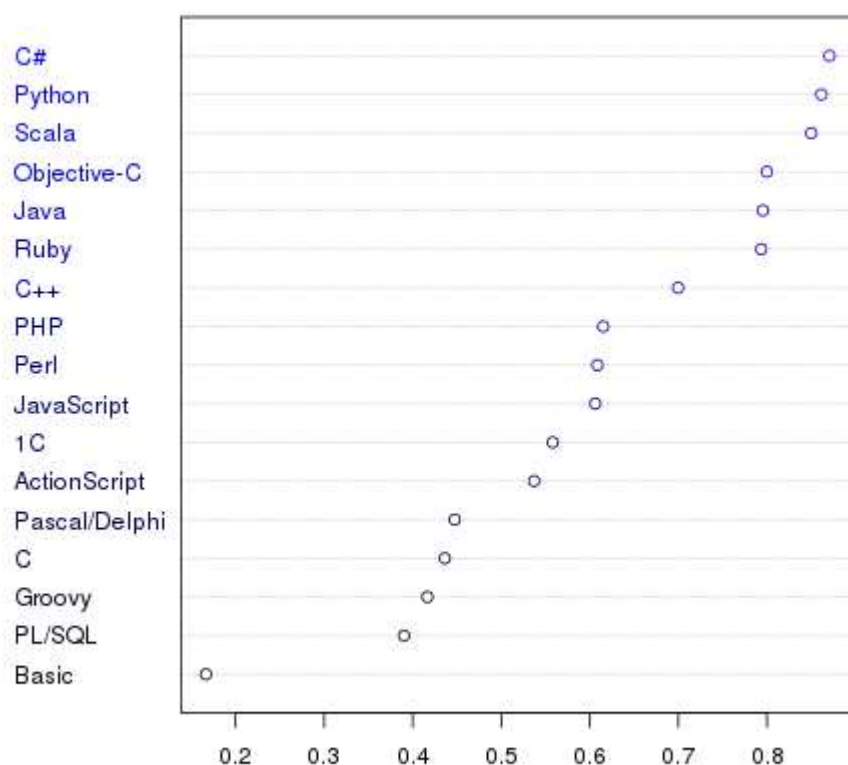


Рисунок Б.5. Розподіл мов програмування за вподобаннями

Додаток В. Реалізація алгоритму обчислення вагових коефіцієнтів та вартості відповідно до критеріїв оцінки вимог до сховища даних корпоративної ІС (фрагмент коду)

```
def main(argv):  
    k = [0] * n # створення масиву з n елементів  
    c = [0] * n # створення масиву з n елементів  
    result = 0.0  
    choices = None  
    # заповнення масиву ...  
    for i in len(n):  
        print "Enter c"  
        c_i = sys.readline()  
        c_i = float(c_i)  
        c[i] = c_i # значеннями від 0 до n-1  
    # заповнение масиву ...  
    for i in len(n):  
        print "Enter k"  
        k_i = sys.readline()  
        k_i = float(c_i)  
        k[i] = k_i # значеннями від 0 до n-1  
  
    for c_i in c: # перебирання всіх елементів  
        # до заданого числа  
        for k_i in k:  
            f = target(k_i, c_i)  
            if f < result:  
                choices = (k_i, c_i)  
    print "k: ", choices[0]  
    print "c: ", choices[1]  
    return choices  
  
#import sys  
if __name__ == "main":  
    main(sys.argv)
```