

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. Михелев І.Л.

" ____ " _____ 2025 р.

КВАЛІФІКАЦІЙНА РОБОТА**на здобуття ступеня вищої освіти "магістр"****на тему:**«Дослідження технологій штучного інтелекту в організації
сховищ даних розподілених інформаційних систем»

Виконав: студент групи 6141 м


_____ **Буртник А.О.**
(підпис)

Керівник роботи:

доцент кафедри ІУСТ, к.т.н.,
(посада, науковий ступінь вчене звання)
_____ **Гайда А.Ю.,**

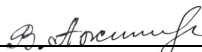
м. Миколаїв – 2025 р.

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 технологійСпеціальність 122 "Комп'ютерні науки"Освітня програма "Комп'ютерні науки"**"ЗАТВЕРДЖУЮ"**

Гарант освітньої програми



к.т.н, доц. Ажищев В.Ф.

"_____" "_____" 2025 р.

**ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ
на здобуття ступеня вищої освіти "магістр"**Студенту Буртнику А. О.

(прізвище, ім'я, по батькові)

1. Тема роботи: «Дослідження технологій штучного інтелекту в організації сховищ даних розподілених інформаційних систем»

Керівник роботи Гайда А. Ю.Затверджені наказом ректора № 1355-уч. від "14" листопада 2025 року.2. Термін подання роботи: 15 грудня 2025 р.

3. Вихідні дані до проєкту (роботи): ДСТУ щодо обробки інформації, літературні джерела, технічна документація на існуючі аналоги систем, матеріали наукового стажування

4. Перелік питань, що належать до розробки (найменування розділів):

Аналіз методів та засобів керування даними в розподілених інформаційних системах; Розробка моделей та механізмів ефективного доступу до даних в проєктах розподілених АІС; Розробка проєктних рішень по підсистемі управління структурою, верифікації БД і впровадження результатів

5. Перелік презентаційних матеріалів: загальні тематичні рисунки по темі дослідження, моделі оптимізації сховищ даних розподілених інформаційних систем, математичні моделі вибору оптимальної структури сховища даних, моделі впровадження розроблених у роботі рішень.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1	Гайда А. Ю., доцент каф. ІУСТ	05.09.2024	22.12.2024
2	Гайда А. Ю., доцент каф. ІУСТ	23.12.2024	15.06.2025
3	Гайда А. Ю., доцент каф. ІУСТ	15.06.2025	14.11.2025

7. Дата видачі завдання 05 вересня 2024 р.*КАЛЕНДАРНИЙ ПЛАН*

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів роботи	Примітка
1	Дослідження та аналіз предметної галузі	22.12.2024	
2	Розробка концепції	15.03.2025	
3	Розділ з розробки проєктних рішень по системі (та її частинах)	15.06.2025	
4	Реалізація проєкту та розробка робочої документації	25.09.2025	
5	Обчислення економічної ефективності	14.11.2025	
6	Вирішення питань охорони праці та навколишнього середовища	14.11.2025	
7	Оформлення пояснювальної записки	14.11.2025	
8	Подання магістерської роботи на перевірку на плагіат	16.11.2025	
9	Подання магістерської роботи рецензенту	20.11.2025	
10	Подання магістерської роботи на державний захист	15.12.2025	

Студент


 (підпис)

Буртник А.О.
 (ПІБ)

Керівник роботи


 (підпис)

Гайда А.Ю.
 (ПІБ)

Анотація

В роботі “Дослідження технологій штучного інтелекту в організації сховищ даних розподілених інформаційних систем” розглядаються питання підвищення ефективності процесів створення і експлуатації сховищ даних розподілених інформаційних систем шляхом впровадження технологій автоматизованого аналізу результатів виконання операцій над сховищем даних з використанням засобів штучного інтелекту та машинного навчання. Також розглядаються питання організації даних і доступу до даних в розподілених інформаційних системах, підтримки цільності та верифікації баз даних, визначення критеріїв оцінки ефективності застосування окремих рішень.

В роботі представлені результати аналізу технологій збереження, організації, верифікації та доступу до даних в розподілених інформаційних системах, розглянуті та визначені особливості розподілу даних в розподіленій базі даних, застосування інструментальних засобів партиціювання даних, визначені критерії оцінки відповідності технологій та інструментів типам сховища даних і проектів, в яких такі банки даних застосовуються.

В роботі запропонований науково-обґрунтований підхід до оцінки якості розподілу даних, інструментальних засобів аналізу результатів виконання операцій у сховищі, створення і керування банками даними в розподілених інформаційних системах, наведені результати застосування розроблених моделей та механізмів у реальних проектах.

Робота викладена на 91 сторінках друкованого тексту, містить 20 рисунків, 11 таблиць, перелік джерел з 26 найменувань, 3 додатки.

Ключові слова: корпоративна автоматизована інформаційна система, сховище даних, штучний інтелект, модифікація структури даних, оцінка стану сховища даних, керування даними.

Abstract

The paper “Research on Artificial Intelligence Technologies in the Organization of Data Warehouses of Distributed Information Systems” examines the issues of increasing the efficiency of the processes of creating and operating data warehouses of distributed information systems by implementing technologies for automated analysis of the results of operations on the data warehouse using artificial intelligence and machine learning. It also examines the issues of data organization and access to data in distributed information systems, maintaining the integrity and verification of databases, and determining criteria for assessing the effectiveness of the application of individual solutions.

The paper presents the results of the analysis of data storage, organization, verification and access technologies in distributed information systems, considers and defines the features of data distribution in a distributed database, the use of data partitioning tools, defines the criteria for assessing the compliance of technologies and tools with data storage types and projects in which such data banks are used.

The paper proposes a scientifically sound approach to assessing the quality of data distribution, tools for analyzing the results of operations in the storage, creating and managing data banks in distributed information systems, and presents the results of applying the developed models and mechanisms in real projects.

The paper is presented on 91 pages of printed text, contains 20 figures, 11 tables, a list of sources with 26 titles, 3 appendices.

Key words: corporate automated information system, information system, artificial intelligence, data structure modification, data warehouse status assessment, data management.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ І СКОРОЧЕНЬ	9
Вступ	10
1. АНАЛІЗ МЕТОДІВ ТА ЗАСОБІВ КЕРУВАННЯ ДАНИМИ В РОЗПОДІЛЕНИХ ІНФОРМАЦІЙНИХ СИСТЕМАХ	15
1.1. Огляд практик керування даними в корпоративних інформаційних системах	15
1.1.1. Моделювання бізнес процесів в корпоративних інформаційних системах	15
1.1.2. Властивості основних типів СКБД	19
1.1.3. Процеси керування даними в корпоративних інформаційних системах	22
1.1.4. Висновки щодо методів та засобів збереження даних	25
1.2 Огляд сервісів корпоративних ІС та протоколів доступу до даних	25
1.2.1. Механізми доступу та сервіси ІС	25
1.2.2. Сховища даних та інтернет-сервери HTTP та HTTPS	27
1.2.3. Висновки щодо методів та засобів передачі даних	28
1.3. Проблеми доступу до даних в корпоративних інформаційних системах та забезпечення надійності зберігання даних	29
1.3.1. Проблеми захисту даних та каналів передачі даних	29
1.3.2. Проблеми, викликані модифікацією структур даних	31
1.3.3. Проблеми синхронізації даних	31
1.3.4. Проблеми забезпечення надійності збереження даних	33
1.3.5. Проблеми доступу до даних в багатокористувацьких корпоративних ІС	34
1.3.6. Проблеми, що визначаються інструментальним забезпеченням	36
1.3.7. Проблеми на рівні каналів передачі даних	38
1.4. Аналіз технологій застосування ІІІ та ML в корпоративних сховищах даних	38

1.4.1. Використання ІІІ та ML для партиціювання БД	38
1.4.2. Використання ІІІ та ML для оптимізації структури БД	40
1.4.3. Застосування штучного інтелекту для оптимізації шляхів доступу до бази даних	42
1.4.4. Проблеми впровадження ІІІ та ML в корпоративних інформаційних системах	44
1.5. Постановка задачі дослідження	46
1.6. Висновки до розділу 1	47
2. РОЗРОБКА МОДЕЛЕЙ ТА МЕХАНІЗМІВ ЕФЕКТИВНОГО ДОСТУПУ ДО ДАНИХ В ПРОЕКТАХ РОЗПОДІЛЕНИХ АІС	48
2.1. Визначення критеріїв оцінки методів та засобів розробки проектів розподілених АІС	48
2.1.1. Визначення критеріїв оцінки методів та засобів збереження та доступу та даних в корпоративних АІС	48
2.1.2. Визначення критеріїв оцінки методів та засобів розробки сховищ даних корпоратив ІС	50
2.2. Розробка моделей опису структури бази даних	51
2.3. Розробка моделей оцінки якості механізмів збереження та доступу до даних	53
2.3.1. Критерії оцінки якості технологій збереження та доступу до даних в розподілених інформаційних системах	53
2.3.2. Механізми забезпечення якості збереження даних	56
2.3.3. Механізми забезпечення якості доступу до даних	59
2.3.4. Механізми доступу та збереження даних клієнта	61
2.4. Розробка моделей адаптивного партиціювання сховища даних корпоративної ІС	65
2.4.1. Розробка моделі збирання даних про виконання запитів до сховища даних	65
2.4.2. Математична модель оптимізації партиціювання таблиць БД	66

2.5. Розробка моделей вибору методів та засобів збереження та доступу та даних в корпоративній ІС	67
2.6. Вибір інструментального забезпечення розробки сховищ даних корпоративних ІС	69
2.7. Визначення властивостей мов програмування, придатних для створення розподілених інформаційних систем	72
2.8. Висновки до розділу 2	74
3. РОЗРОБКА ПРОЕКТНИХ РІШЕНЬ ПО ПІДСИСТЕМІ УПРАВЛІННЯ СТРУКТУРОЮ, ВЕРИФІКАЦІЇ БД І ВПРОВАДЖЕННЯ РЕЗУЛЬТАТІВ	76
3.1. Розробка адаптивної моделі управління структурою і верифікації сервера баз даних	76
3.2. Результати тестування для різних “engines” РСКБД MySQL	82
3.3. Висновки до розділу 3	83
Загальні висновки по роботі	84
Список використаних джерел	85
Додаток А. Статистика використання різних мов програмування в Україні (дані сервера ІТ-вакансій на початок 2014 року)	88
Додаток Б. Приклад реалізації компонентної моделі кластеризації і верифікації структури БД	90
Додаток В. Реалізація алгоритму вибору належності даних до партицій відповідно до критеріїв оцінки вимог до ІС	91

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ І СКОРОЧЕНЬ

AIC – автоматизована інформаційна система

АРМ – автоматизоване робоче місце

БД – база даних

ІС – інформаційна система

ІТ – інформаційні технології

ОО – об’єктно-орієнтований, об’єктно-орієнтована

ОП – оперативна пам’ять

ОС – операційна система

ОЦД – обмеження цілісності даних

ПЗ – програмне забезпечення

РСКБД – реляційна система керування базами даних

СКБД – система керування базами даних

ШІ – штучний інтелект

ШНМ – штучна нейронна мережа

ЦП – центральний процесор

DML – Data Manipulation Language, мова маніпуляції даними

ML – машинне навчання

UML – уніфікована мова моделювання

Вступ

Обсяги даних, що в той чи інший спосіб доступні для використання в сховищах даних корпоративних інформаційних систем становлять сотні терабайт і завдяки використанню даних з мережі Інтернет продовжують стрімко зростати. Такі дані включають нормативні, конструкторські та технологічні документи, бухгалтерські та управлінські звіти, медійні матеріали тощо. Як наслідок, спеціальні алгоритми традиційних систем керування базами даних (СКБД) не здатні забезпечити прийнятний час відгуку на окремих запитах щодо отримання і обробки даних в корпоративних задачах. Крім того, такі задачі часто виконуються на спеціальних обчислювальних системах з масивно-паралельною архітектурою, що дозволяють розпаралелити обробку даних і винести частину обробки з рівня СКБД на рівень окремих серверів зберігання даних, що, у свою чергу, призводить до переходу на розподілені сховища даних корпоративних інформаційних систем. Як наслідок, ефективне використання таких систем можливе лише за наявності надійних засобів і механізмів збереження і керування розподілом корпоративних даних.

Отже, надійність застосування зазначених засобів організації сховищ даних і відповідних механізмів має виявлятися як в умовах нормальної експлуатації інформаційних систем, так і в умовах, що виникають через конкуренцію між окремими підприємствами, економічне та політичне протистояння між окремими політичними угрупованнями всередині країни або окремими країнами тощо.

Також активно розвиваються хмарні обчислення (Cloud computing). Ця технологія є привабливою як для окремих фізичних користувачів, так і для корпоративного сектору. Її переваги визначаються тим, що кожен окремий користувач може відносно легко і недорого отримати у тимчасове користування окремі сервіси для зберігання і обробки даних. Наприклад, можна замовити комп'ютер із заданим об'ємом пам'яті, дисків, кількістю

процесорів, з попередньо встановленими операційною системою (ОС) і СКБД або іншими пакетами програмного забезпечення (ПЗ) і працювати з всім цим з власного офісу, хоча фізично і обладнання, і програмне забезпечення розташовуються і обслуговуються у віддаленому центрі збереження і обробки даних. В умовах всеосяжного проникнення інтернет у всі сфери суспільного життя поширені служби, де можна замовити потрібний дисковий ресурс і зберігати на ньому свої файли. Прикладами таких систем з 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). Такі популярні сервіси є тим більш важливими в умовах конфліктів, війни та техногенних катастроф.

СКБД є одним з найбільш поширених класів прикладних систем для серверів, що випускаються більшістю компаній-розробників корпоративного програмного забезпечення. При цьому додатки, орієнтовані на використання баз даних, і самі СКБД сильно різняться за своєю організацією. Абсолютна більшість інсталюваних у світі систем збереження даних є реляційними і ця архітектура просувається на ринок і підтримується такими знаними виробниками ПЗ, як Oracle, Sybase, Microsoft, Ingres, Informix тощо. Тим не менше, складність структур даних і, особливо, бізнес-логіки додатків корпоративних інформаційних систем призводять до необхідності пошуку технологій, які дозволили б віднайти розумний компроміс між високою продуктивністю реляційних СКБД і низькою продуктивністю розробки ПЗ для роботи з такими СКБД за умов низького рівня організації даних та високої складності структур корпоративних даних. Крім того, навіть за наявності стандартизації типів даних і синтаксису запитів до РСКБД між різними РСКБД є значні відмінності, що ускладнюють перенесення даних з однієї РСКБД на іншу та ускладнюють вирішення питань забезпечення високої ефективності обробки, а також надійності збереження даних.

Останнім часом реляційні СКБД дещо поступились системам з альтернативними моделями даних. Частково це викликано спробами підвищення продуктивності РСКБД за рахунок спрощення структур даних при зберіганні, що відносно легко досягається в РСКБД. З іншого боку, корпоративні системи вимагають вдосконалення засобів організації даних і доступу до даних [1] за умов підвищення складності процесів обробки і стрімкого зростання обсягів даних, у тому числі – за рахунок переходу до більш складних інформаційних моделей, на яких РСКБД поступаються об'єктним СКБД або СКБД на базі файлових сховищ. Якщо системи на базі файлових серверів порівняно просто розділити за типами робочого навантаження на два принципово різних класи (з інтенсивною обробкою атрибутів файлів і з інтенсивною обробкою самих даних), то провести подібну класифікацію серед додатків баз даних і СКБД просто неможливо. Однією з причин, що унеможливають таку класифікацію є стандартна мова реляційних СКБД (мова SQL), яка виявляється набагато багатшою, ніж набір операцій мережевої файлової системи.

Одним із шляхів вирішення зазначеної проблеми є впровадження адаптивних технологій керування розподіленими сховищами даних і інтеграція СКБД з файловими системами, коли частина задач з організації і пошуку даних покладається не на СКБД, а на ядро операційної системи (ОС), що дозволяє значно підвищити пропускну спроможність СКБД і зменшити час обробки запитів. Подібна технологія активно використовується в продуктах компанії Oracle.

Ще однією істотною проблемою проектування і експлуатації сховищ даних корпоративних інформаційних систем є постійні зміни у діяльності компаній та зміни у їх інформаційному оточенні. Оскільки діяльність постійно змінюється, змінюється і структура даних, і структура процесів, і, власне, сама предметна область такої діяльності. Отже, ці зміни мають відбиватись і на моделях та процесах обробки даних, способах збереження та

структурах самих даних у сховищі даних. Для розподілених сховищ даних таке керування має визначати, у тому числі, локацію даних у розподіленій системі збереженні і обробки даних, а отже і механізми визначення оптимальної локації. Оскільки така задача є погано формалізованою для її вирішення буде доцільним використання засобів машинного навчання та штучного інтелекту.

З вищенаведеного можна зробити висновок, що питання керування базами та банками даних в розподілених інформаційних системах корпоративного сектору є важливими і актуальними, а наявність значної кількості невирішених проблем вказує на необхідність проведення подальших досліджень.

Актуальність роботи визначається широким використанням корпоративних інформаційних систем та наявними проблемами, що визначаються високим навантаженням сховищ даних та СКБД таких систем, та механізмами доступу до даних [1] в корпоративних інформаційних системах в частині розподілу, міграції та верифікації даних; складністю моделювання бізнес процесів, що постійно змінюються та вимагають модифікації алгоритмів обробки даних та структур даних сховища даних.

Метою дослідження є підвищення ефективності доступу до даних, процесів організації і верифікації сховища даних в корпоративних інформаційних системах за рахунок обґрунтованого вибору моделей та механізмів організації, розподілу, міграції, модифікації, верифікації даних та їх моделей.

Об'єктом дослідження є технології організації сховищ даних корпоративних інформаційних систем та технології доступу до даних в корпоративних інформаційних системах.

Предметом дослідження є процеси вибору технологій організації, розподілу, міграції, модифікації, верифікації даних та їх моделей в корпоративних інформаційних системах.

Задачі дослідження: виконати аналіз існуючих технологій збереження та доступу до даних, визначити критерії оцінки інструментальних засобів, розробити моделі та механізми вибору інструментальних засобів та технологій міграції, модифікації, верифікації даних і їх моделей в корпоративних інформаційних системах.

Наукова новизна роботи полягає в розробці моделей адаптивного вибору та оцінки механізмів розподілу, міграції, модифікації, верифікації сховища даних СКБД в корпоративних інформаційних системах відповідно до особливостей предметної області.

1. АНАЛІЗ МЕТОДІВ ТА ЗАСОБІВ КЕРУВАННЯ ДАНИМИ В РОЗПОДІЛЕНИХ ІНФОРМАЦІЙНИХ СИСТЕМАХ

1.1. Огляд практик керування даними в корпоративних інформаційних системах

1.1.1. Моделювання бізнес процесів в корпоративних інформаційних системах

Складність вирішення завдання моделювання бізнес процесів загалом та реорганізації структур даних зокрема в системах з реляційними базами даних (які набули найбільшого поширення) зростає з ускладненням структури даних 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, що, в цілому, призводить до необхідності побудови і виконання над базами даних вкрай складних запитів. Значна кількість методологій щодо проектування та організації баз даних лише підтверджують складність та різноплановість процесів створення баз даних та їх істотну залежність від бізнес логіки задач, для яких такі бази даних створюють. Наслідком спрощеного підходу до проектування баз даних та невиправдане використання найбільш поширених реляційних баз даних призводить до невідповідності функціоналу, що забезпечується СКБД вимогам до механізмів доступу до даних, які визначаються програмним забезпеченням зокрема та бізнес-логікою в цілому [1]. 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. При цьому будь-які зміни структури бази даних внаслідок змін у бізнес логіці або розвитку систем змушує розробників виконувати переробку раніше створених запитів.

Серед різних компонентів апаратури сервера СКБД конфігурація пам'яті

системи зазвичай має найбільше вплив на її продуктивності – більша частина основної пам'яті в системах СКБД використовується як буфер (кеш) даних, що дозволяє в багатьох випадках усунути необхідність виконання фізичного введення-виведення з диска. Оскільки звернення до основної пам'яті виконується приблизно в 30000 разів швидше, ніж звернення до швидкого диску, мінімізація дискового вводу-виводу є першочергово важливим завданням.

Існуюче в даний час співвідношення між цінами підсистеми пам'яті і дискової підсистеми показує, що економічно доцільно кешувати дані, до яких здійснюються звернення більше одного разу кожні п'ять хвилин. Це означає, що дані, до яких звернення відбуваються більш часто, ніж один раз на п'ять хвилин, повинні кешуватися в основній пам'яті. Відповідно, щоб оцінити розмір кешу даних необхідно підсумувати об'єми всіх даних, які додаток припускає використовувати більш часто, ніж один раз на п'ять хвилин на рівні всієї системи. Тому при визначенні необхідного обсягу основної пам'яті розраховують, принаймні, на такий розмір кешу даних. Додатково зазвичай резервують ще 5-10% для розміщення верхніх рівнів індексів В-дерев, збережених процедур та іншої керуючої інформації СКБД.

Кожна комерційна СКБД забезпечує механізм для визначення вартості чи переваг зміни розміру різних кешів СКБД. Коли система інстальована і починає працювати, необхідно використовувати ці механізми для визначення ефекту від зміни розмірів кешу вводу-виводу СКБД.

Хоча основною метою СКБД на макрорівні є керування томами даних, які за визначенням зазвичай мають великий обсяг і неминуче у багато разів перевищують обсяг оперативної пам'яті, проведені дослідження показали, що доступ до даних в загальному випадку підкоряється правилу "90/10": 90% всіх звернень виконуються до 10% даних. Більше того, виявилось, що до "гарячих" даними, звернення до яких становлять 90% усіх звернень, знову застосовано правило "90/10". Таким чином, приблизно 80% усіх звернень до

даних пов'язані з доступом до приблизно 1% агрегованих даних. Слід зазначити, що це відношення включає звернення до внутрішніх метаданих СКБД (що включають індекси В-дерев і т. п.), звернення до яких зазвичай приховані від прикладного програміста. Хоча бажано мати коефіцієнт попадань в кеш приблизно на рівні 95%, з економічних міркувань неможливо сліпо забезпечити обсяг кешу в пам'яті, що дозволяє розмістити 10% всіх даних: навіть для скромних баз даних об'ємом 5 Гбайт це зажадало б збільшення розміру основної пам'яті до 500 Мбайт. Проте зазвичай завжди можливо забезпечити кеш обсягом в 1% всіх даних навіть для дуже великих баз даних. Максимальний обсяг основної пам'яті сучасних серверів може бути дуже великим.

Природно, конфігурація системи повинна забезпечувати також простір для традиційного використання пам'яті. В СКБД на базі Unix і NT завжди необхідно забезпечити принаймні 16 Мбайт для базової операційної системи. Далі необхідно передбачити простір об'ємом 2-4 Мбайт для ряду програм СКБД (програм ведення журналу, перевірки узгодженого стану, архіваторів і т. п.) і досить простору для розміщення в пам'яті двійкових кодів програми. Обсяг двійкових кодів додатка становить зазвичай 1-2 Мбайт, але іноді вони можуть досягати 16-20 Мбайт. Операційна система забезпечує режим поділу двійкових кодів між безліччю користуються ними процесів, тому необхідно резервувати простір тільки для однієї копії. Простір для розміщення самого коду сервера СКБД залежить від його архітектури.

Оброблювана кожним ЦП додаткова інформація вимагає виділення деякого простору для того, щоб обійти інтенсивну фрагментацію пам'яті. Ці задачі виявляються тим більше складними, що помилки вибору структури бази даних важко виправляти через істотні розбіжності в механізмах організації даних в базах даних (Рисунок 1.1).

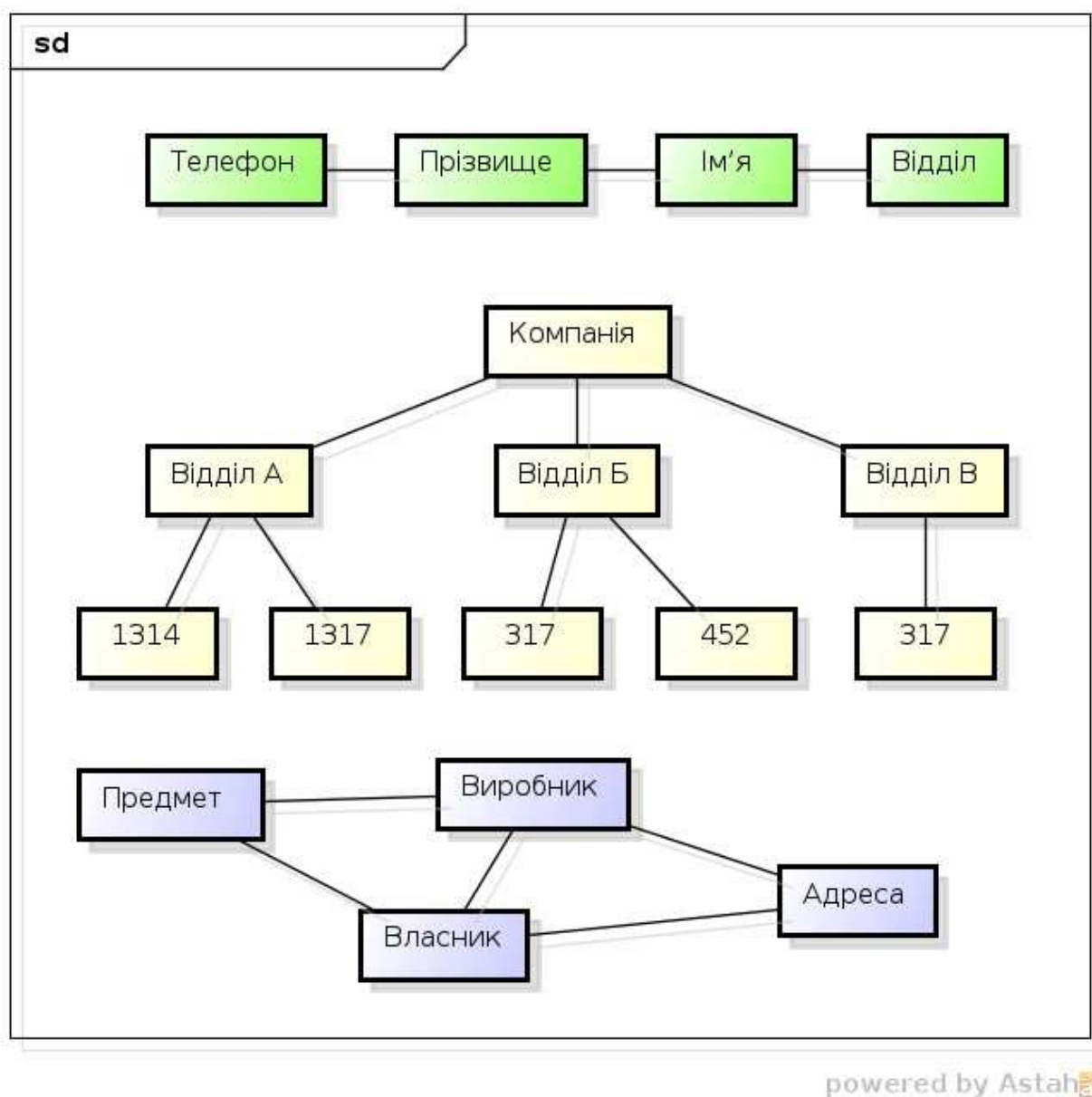


Рисунок 1.1 – Приклади типів структур БД (зверху – донизу: реляційні, ієрархічні, мережеві)

З метою визначення особливостей реалізації бізнес логіки в корпоративних інформаційних системах та її відображення в моделі та механізми збереження та доступу до даних для СКБД розглянемо особливості основних типів сучасних баз даних та визначимо їх переваги та недоліки стосовно розробки систем збереження даних Інтернет-проектів.

1.1.2. Властивості основних типів СКБД

Реляційних СКБД є традиційними для багатьох проектів 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 (Рис. 1.2).

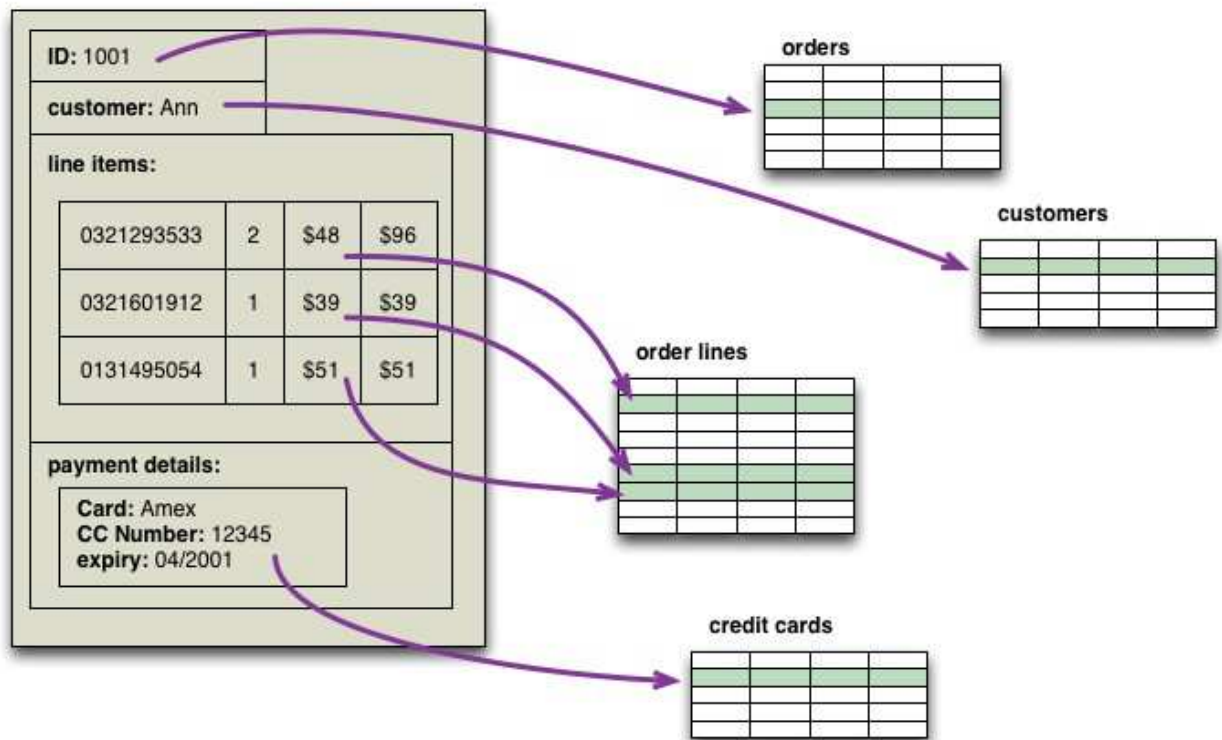


Рисунок 1.2 – Відображення предметної області в реалізації інформаційної системи для РСКБД

NoSQL. Для вирішення завдань, пов'язаних зі складними методами обробки даних і складною організацією даних (що характерно при автоматизації складних бізнес процесів), набули поширення так звані NoSQL бази даних та сховища (Рис 1.3). Вони істотно відрізняються від РСКБД і застосовуються частіше не для зберігання всіх даних системи, а лише для вирішення специфічних завдань (журналювання, кешування, черги завдань), і тому менш поширені в простих проектах.

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.

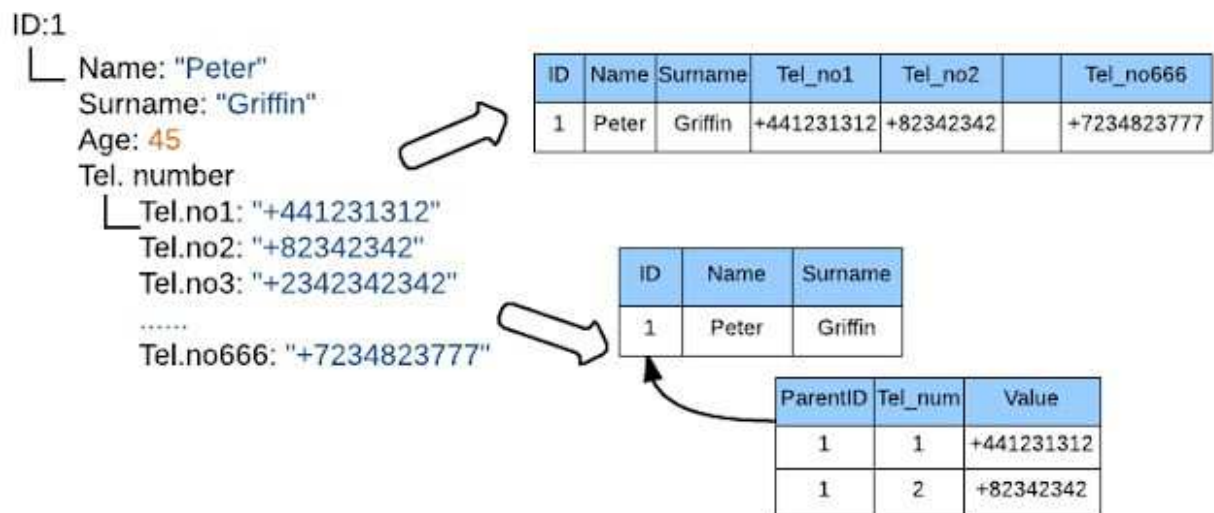


Рисунок 1.3 – Відображення предметної області в реалізації інформаційної системи для NoSQL СКБД

ОО СКБД. Зазначена проблема могла б бути подолана за рахунок використання об'єктно-орієнтованих систем керування базами даних (СКБД), що дозволяють повною мірою задовольнити потреби в організації даних з боку об'єктів програмного забезпечення, але, на жаль, об'єктно-орієнтовані СКБД на поточний момент не набули широкого поширення, насамперед – через свою складності і надмірної вартості. Перші публікації були викликані

проблема РСКБД при високій складності відношень (Рисунок 1.1).

Об'єктно-орієнтована база даних зазвичай рекомендовані для тих випадків, коли потрібна високопродуктивна обробка даних, що мають складну структуру (Рисунок 1.4).

Обов'язкові характеристики, яким має відповідати будь-яка ООБД наступні:

- Підтримка складних об'єктів. В системі має бути передбачена можливість створення складових об'єктів за рахунок застосування конструкторів. Необхідно, щоб будь-який з конструкторів можна було використовувати з будь-яким об'єктом.
- Підтримка індивідуальності об'єктів. Всі об'єкти мають мати унікальний ідентифікатор, котрий не залежить від значень їх атрибутів.
- Підтримка інкапсуляції. Інкапсуляція досягається за рахунок того, що програмісти мають право доступу тільки до специфікації інтерфейсу методів, а дані і реалізація методів приховані всередині об'єктів.
- Підтримка типів і класів. Вимагається, щоб ООБД підтримувалась хоча б одна концепція відмінності між типами і класами.
- Підтримка наслідування типів і класів від їх предків. Підтип або підклас має наслідувати атрибути і методи від супертипу або суперкласу.
- Перезавантаження у поєднанні з повним пов'язуванням. Реалізація методу має залежати від типу об'єктів, для якого даний метод викликається.
- Обчислювальна повнота. Мова операцій над даними має бути мовою програмування загального призначення.
- Набір типів даних має бути розширюваним. Користувач повинен мати засоби створення нових типів даних на основі набору зумовлених системних типів. Більш того, між способами використання системних і користувацьких типів даних не повинно бути жодних відмінностей.

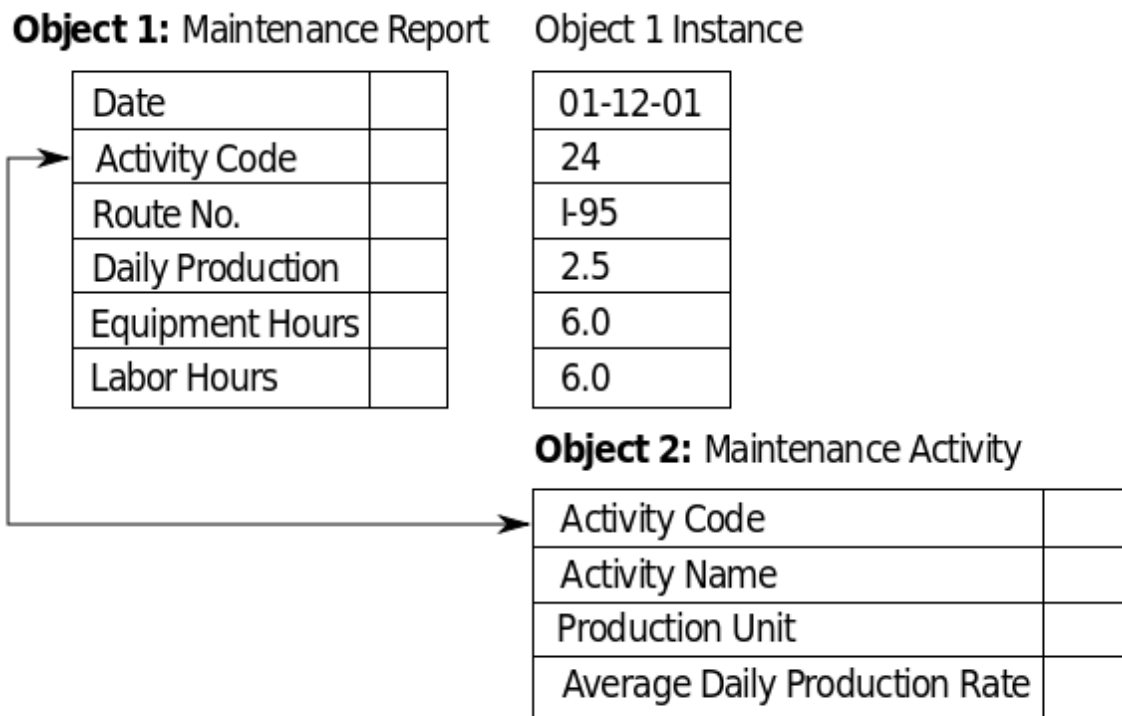


Рисунок 1.4 – Відображення даних в ОО СКБД

1.1.3. Процеси керування даними в корпоративних інформаційних системах

З метою втілення зазначених концепцій та подолання наявних в РСКБД 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. Таким чином, використання ORM надає розробнику можливість відносно вільно маніпулювати даними на рівні об'єктів і абстрагуватися від особливостей реалізації структур даних на конкретному сервері баз даних. Це також дозволяє відносно простими засобами отримувати зрозумілий і якісний код, спрощує його тестування, подальшу модифікацію і супровід.

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 пропускної 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, які його використовують, будуть працювати повільніше і використовувати більше оперативної пам'яті, ніж програми, що містять «вручну» написані запити. Тим не менше, ORM позбавляє розробника від необхідності написання великої кількості коду, часто одноманітного і схильного до помилок, і, головне, який тісно пов'язаний зі структурами даних, що унеможливорює оперативну модифікацію цих структур. Тим самим ORM 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 проектів, 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.

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 в 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. Такий підхід виявляється цінним для систем, в яких обсяг трафіку між клієнтами та сервером є критичним.

Розподілені бази даних 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 можна відзначити:

- великий обсяг даних у середовищі передачі даних (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 подібних корпоративних систем.

1.1.4. Висновки щодо методів та засобів збереження даних

Для звичайних проектів простіше використовувати MySQL або PostgreSQL, оскільки на простих операціях не надто помітна різниця між різними базами даних. Однак, якщо проект передбачає складну логіку обробки даних, то вибір бази варто проводити виходячи з технічних характеристик та області використання кожної окремої системи та її сховища даних (Таблиця 1.1, рисунок 1.5).

1.2 Огляд сервісів корпоративних ІС та протоколів доступу до даних

1.2.1. Механізми доступу та сервіси ІС

Окрім структури організації даних бази даних також істотно відрізняються механізмами доступу, які можуть бути реалізовані на різних

рівнях реалізації інформаційної системи: на рівні файлової системи ОС, СКБД або навіть на рівні додатку [1].

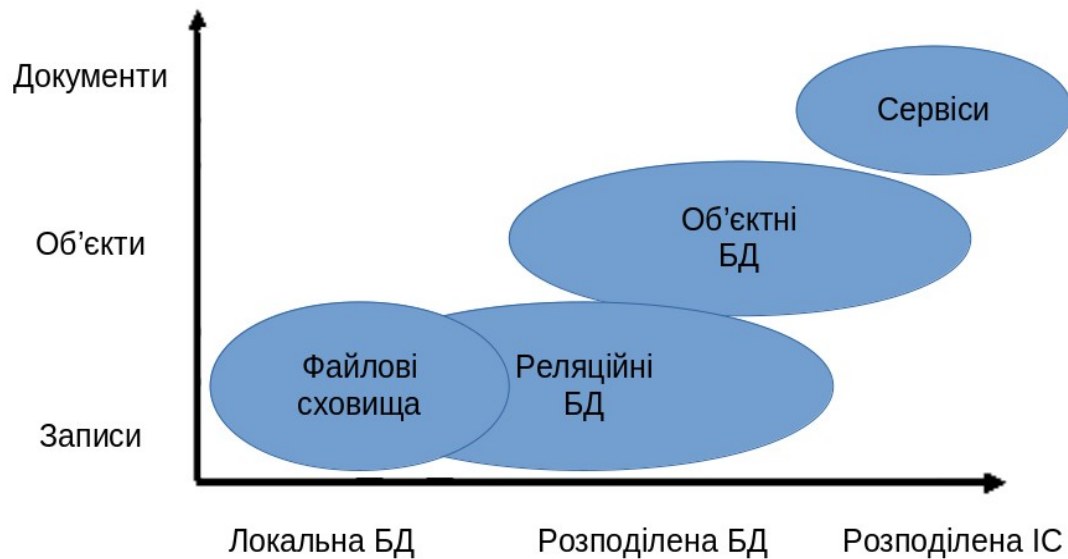


Рисунок 1.5 – Области використання СКБД з різною організацією

Таблиця 1.1 – Типи сховищ баз даних за областями призначення

№	Тип сховища	Область використання сховища
	Реляційні СКБД	Класичні проекти з відносно простою бізнес логікою
	NoSQL СКБД	Проекти зі складною бізнес логікою та відносно невеликим обсягом даних
	ОО СКБД	Проекти зі складною бізнес логікою та відносно великим обсягом даних
	Документо-орієнтовані СКБД	Проекти з ієрархічно-впорядкованими даними
	Розподілені СКБД	Проекти систем обробки даних з низькими вимогами до надійності
	Файлові сховища	Проекти з масовим доступом до окремих файлів

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).

1.2.2. Сховища даних та інтернет-сервери HTTP та HTTPS

Найбільш популярний серед інтернет-розробників вважається комплект LAMP (Linux, Apache, MySQL, PHP). Цей інтернет-сервер дозволяє підключати зовнішні модулі для роботи з інтерпретуються мовами програмування (наприклад, з PHP, Ruby, Python) і для надання даних, може використовувати СКБД для аутентифікації користувачів, можлива модифікація повідомлень про помилки і багато іншого. Apache є крос-платформним ПЗ, підтримує операційні системи Linux, BSD, Mac OS, Microsoft Windows, Novell NetWare, BeOS, хоча найбільш часто використовується тільки на серверних дистрибутивах Linux і BSD.

Сам по собі інтернет-сервер вельми потужний і надійний, але споживання ресурсів при його роботі зазвичай досить високо. Часто Apache використовують як бекенд-сервера, встановлюючи позаду інтернет-сервера nginx.

HTTPS – 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 (HyperText Transfer Protocol Secure) – 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 [8-11].

На відміну від HTTP, який використовує TCP-порт 80, для HTTPS за замовчуванням використовується TCP-порт 443. Щоб підготувати інтернет-

сервер для обробки https-з'єднань, адміністратор повинен отримати і встановити в систему SSL-сертифікат для цього інтернет-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 тощо. 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 [12].

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.

1.2.3. Висновки щодо методів та засобів передачі даних

Широкий спектр технологій доступу до даних породжує проблеми вибору цих технологій в окремих проектах і проблеми забезпечення цілісності даних.

Вибір конкретних технологій здійснюється, як правило, за результатами визначення вимог до інформаційного сховища, вибору інструментальних

засобів реалізації проекту корпоративної розподіленої інформаційної системи та вибору архітектури бази даних для проекту.

Очевидно, що засоби захисту даних від не санкціонованого доступу є важливими і будуть продовжувати вдосконалюватися [9]. Будуть ускладнюватися алгоритми кодування даних на всіх рівнях (в БД, архівах, при передачі по мережі), способи авторизації і аутентифікації тощо.

1.3. Проблеми доступу до даних в корпоративних інформаційних системах та забезпечення надійності зберігання даних

1.3.1. Проблеми захисту даних та каналів передачі даних

Останнім часом з'явилася тенденція виносу механізму управління користувачами (їх обліковими записами та засобами конфігурації робочих місць) з окремих СКБД і додатків в єдину централізовану систему організації (Identity & Access management). Такий підхід не тільки спрощує управління користувачами в масштабах організації, але також дозволяє в автоматичному режимі керувати ними з урахуванням поточних вимог бізнесу. Наслідком впровадження таких технологій є удосконалення методів аудиту, створення інтегрованих середовищ збору та аналізу аудит-інформації всієї організації (із безлічі різних додатків і, власне, самих 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 – глобальна мережа (Інтернет) / корпоративна локальна мережа, номер терміналу і тому подібні).

Наступним аспектом, що визначає захищеність даних, є їх обсяг [10] і застосовуються для зменшення обсягу механізми 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 операцій DML 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. Але для сховищ даних та історичних даних такий підхід цілком прийнятний. Адміністратор повинен мати можливість для різних даних, різних таблиць і навіть частин таблиць вибирати ті варіанти стиснення та зберігання (за строками, по стовпцях), які найбільш підходять з точки зору бізнес-використання цих даних. Причому, оскільки для додатків способи зберігання і стиснення прозорий, DBA може періодично міняти способи зберігання і стиснення. Цей підхід буде дуже корисний користувачам великих БД (наприклад, SAP-користувачам), які сьогодні змушені постійно збільшувати свої багато-терабайтні системи зберігання. Ну, і розумний механізм стиснення дозволяє залишати найсвіжіші та найбільш активно використовувані дані деякий час не стиснутими. Пізніше вони стискаються автоматично у фоновому режимі.

Ще один перспективний напрям розвитку – ізоляція DBA від даних. Сьогодні адміністратор може бачити і змінювати всі дані в БД. Нові засоби захисту (такі як Oracle Data Vault option) дозволять DBA виконувати всі операції по адмініструванню БД, не дозволяючи йому бачити і змінювати дані. Також адміністратору можна буде обмежити набір дозволених до

виконання операцій. Обмеження може бути прив'язане до імені DBA, часу, точці входу в систему.

1.3.2. Проблеми, викликані модифікацією структур даних

Як показано на рисунку 1.1 структура баз даних може істотно відрізнятися. 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, а їх перенесення і забезпечення повноти та внутрішньої узгодженості може стати складною і відповідальною задачею.

1.3.3. Проблеми синхронізації даних

Проблеми синхронізації даних розглянемо на прикладі мережі біт-торент, яка покликана забезпечувати таку синхронізацію.

BitTorrent Sync синхронізує файли використовуючи однорангову мережу P2P, цей протокол може ефективно використовуватися для передачі великих файлів між безліччю пристроїв, цей принцип передачі даних використовується в таких додатках як Torrent, Transmission і BitTorrent. BitTorrent Sync відрізняється від хмарних сервісів тим, що користувацькі дані знаходяться на локальному носії, а синхронізація вимагає як мінімум одного пристрою підключеного до мережі.

BitTorrent Sync використовує шифрування даних за допомогою алгоритму AES з довжиною ключа в 128 біт, який може бути як створеним випадково так і вибраним користувачем. Ключ є похідною від секрету, який

може бути поширений серед користувачів, які обмінюються даними. Дані надсилаються безпосередньо від пристрою до пристрою, якщо приймаючий пристрій недоступний (наприклад, перебуває за фаєрволом або виключено), то дані будуть спрямовані іншому вузлу мережі. Але дані будуть прийняті вузлом тільки в тому випадку, якщо йому відомий секретний ключ для розшифровки і перегляду даних. Вузол мережі може бути доданий до синхронізації, якщо йому буде передано відповідний секретний ключ, що дозволяє йому переглядати (RO-доступ) або змінювати (RW-доступ) дані.

Синхронізація робочих файлів між пристроями – найбільш очевидний спосіб використання, можна просто синхронізувати дані між декількома ПК або серверами, забезпечуючи наявність на кожному з пристроїв ідентичною інформації. Даний спосіб корисний в організації зберігання особистих і корпоративних даних.

Резервне копіювання – BitTorrent Sync 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 сховищах даних [14, 15] і додатках розподілених інформаційних систем 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.

1.3.4. Проблеми забезпечення надійності збереження даних

Оскільки значна частина сучасних АІС заснована на використанні реляційних баз даних, необхідно розглянути фактори, що визначають якість збереження даних в цілому. Однією з найважливіших характеристик якості БД є інформаційна безпека, яка, в свою чергу, включає в себе цілісність інформації. В теорії реляційних БД цілісність розуміється як логічна узгодженість і семантична коректність даних. До початку розробки БД замовник формулює вимоги цілісності, які вважає значущими в даній предметній області. Завдання розробника БД полягає в тому, щоб забезпечити програмну реалізацію цих вимог у вигляді спеціальних програмних об'єктів БД – обмежень цілісності даних (ОЦД) і тригерів. Підтримка цілісності в БД виконується на необхідному рівні, якщо забезпечені:

- придатність ОЦД і тригерів, тобто ОЦД і тригери реалізують всі специфіковані вимоги цілісності;
- коректність ОЦД і тригерів, тобто робота ОЦД і тригерів призводить до очікуваних результатів, зокрема, ОЦ і тригери не містять випадкових або навмисних декларованих можливостей.

Недостатня придатність і коректність ОЦД і тригерів може стати причиною порушення цілісності інформації, що, в свою чергу, може привести до збоїв в роботі АІС і пов'язаних з ними проблем у виробничій діяльності організації, зниження якості продукції та послуг, економічним збитком, людських жертв.

Існуючі методи верифікації цілісності реляційних БД мають наступні недоліки:

- в процесі верифікації не використовуються формальні моделі вимог цілісності, в результаті чого нерідко виникає двозначність формулювання вимог, яка ускладнює ідентифікацію помилок;

- розробка тестових сценаріїв для виявлення НВД в БД вимагає значних витрат людської праці і негативно позначається на оперативності виявлення НВД;
- відсутній спосіб формального опису властивостей наявних НВД в ОЦД і тригерах;
- на практиці має місце неповнота виявлення НВД в ОЦД і тригерах, особливо це стосується умисних НВД (програмних закладок), що частково пов'язано з відсутністю формального опису НВД.

Таким чином, завдання формалізації вимог і функцій цілісності реляційних БД, а також підвищення оперативності та повноти виявлення НВД в БД є актуальними в області верифікації БД.

1.3.5. Проблеми доступу до даних в багатокористувацьких корпоративних ІС

Багатокористувацькі системи стикаються з низкою критичних проблем доступу до даних, які виникають, коли кілька користувачів або процесів одночасно намагаються читати або змінювати одні й ті самі дані. Ці проблеми є фундаментальними для розробки надійних і послідовних систем керування базами даних.

Основні проблеми доступу до даних включають: проблеми паралелізму та цілісності даних; проблеми блокування та продуктивності; проблеми масштабованості.

Проблеми паралелізму та цілісності даних є найбільш поширеними проблемами, що виникають внаслідок одночасного виконання транзакцій:

- "Брудне" читання (Dirty Reads) / Читання непідтверджених даних (Reading Uncommitted Data). Виникає, коли транзакція А читає дані, які були змінені транзакцією В, але транзакція В ще не завершилася (не підтвердилася – COMMIT). Якщо транзакція В пізніше відкотиться (ROLLBACK), дані, прочитані А, виявляться недійсними або

"брудними".

- Втрачене оновлення (Lost Update). Виникає, коли дві транзакції читають одні й ті самі дані, змінюють їх локально, а потім намагаються записати назад. Остання записана зміна "перетирає" (затирає) зміну, зроблену першою транзакцією. Наприклад, два менеджери одночасно додають товари до кошика на складі. Оновлення одного менеджера ігнорується системою.
- Неповторюване читання (Non-repeatable Reads). Виникає, коли транзакція А читає дані двічі, але між двома читаннями транзакція В змінює або видаляє ці дані. В результаті, друге читання транзакції А дає інший результат, ніж перше, що порушує логіку роботи А.
- Фантомні читання (Phantom Reads). Виникає подібно до неповторюваного читання, але стосується наборів даних (результатів запитів WHERE). Транзакція А виконує запит, що повертає певний набір рядків. Транзакція В додає або видаляє рядки, які відповідають критеріям запиту. При повторному виконанні запиту транзакція А бачить "фантомні" нові або відсутні рядки.

Проблеми блокування та продуктивності виникають внаслідок вирішення проблем паралелізму (наприклад, блокування даних):

- Взаємне блокування (Deadlock) / "Клінч". Має місце, коли дві або більше транзакцій чекають одна на одну, щоб звільнити ресурси. Транзакція А блокує ресурс Х і чекає ресурс У. Транзакція В блокує ресурс У і чекає ресурс Х. Жодна з транзакцій не може продовжити роботу.
- Зниження пропускну здатності (Contention). Виникає при надмірному використанні блокувань і може змусити багато транзакцій чекати на доступ до популярних даних. Це значно знижує загальну продуктивність системи, хоча й забезпечує цілісність даних.
- Голодування (Starvation). Виявляється у ситуаціях, коли деякі транзакції

постійно "відкладаються" на користь інших транзакцій, що мають вищий пріоритет або просто отримують доступ до ресурсів раніше, і тому за критичний для клієнта час не можуть завершити свою роботу.

Проблеми масштабованості виникають через так звані централізовані вузькі місця, коли у міру зростання кількості користувачів, централізовані механізми керування доступом (наприклад, єдиний сервер бази даних або системи контролю доступу) можуть стати вузьким місцем і є наслідком високої відносної складності розподілених систем. При шардингу або використанні розподілених баз даних забезпечення цілісності транзакцій між різними фізичними вузлами стає надзвичайно складним (проблема двофазного коміту) [25].

Вирішення цих проблем досягається за допомогою впровадження різних рівнів ізоляції транзакцій (Isolation Levels: Read Uncommitted, Read Committed, Repeatable Read, Serializable), які визначають, наскільки суворо система запобігає вищезгаданим аномаліям за рахунок використання різних механізмів блокування або керування версіями (MVCC – Multiversion Concurrency Control).

1.3.6. Проблеми, що визначаються інструментальним забезпеченням

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

Для написання програм на низькому рівні абстракції, докладного і детального опису операцій, що враховують архітектура комп'ютера та

операційної системи використовують низькорівневі мови програмування, такі як асемблер та С. У веб програмуванні асемблер 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 програмістів, отже збільшується і ймовірність появи помилок. Це можна вважати недоліком програмування на мовах низького рівня.

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 спрощують програмування з використанням технології об'єктно-орієнтованого програмування (Python, Java [2-4] тощо), що, з точки зору можливого документально-орієнтованого представлення інформації, є корисним.

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

1.3.7. Проблеми на рівні каналів передачі даних

При зростанні обсягу БД вузьким місцем стають канали передачі даних між сервером БД і пристроями зберігання. Можливість збільшити кількість і пропускну здатність таких каналів, а також знизити обсяг переданих даних за рахунок виконання частини операцій з перегляду та вибірки даних на рівні осередків зберігання, дозволяє зняти проблему каналів передачі. Такі зв'язки обладнання та спеціального ПЗ відомі як машини баз даних, доволі детально описані в спеціальній літературі. Приклади популярних машин баз даних: Netezza, Datallegro, Teradata.

1.4. Аналіз технологій застосування ШІ та ML в корпоративних сховищах даних

1.4.1. Використання ШІ та ML для партиціювання БД

Штучний інтелект пропонує революційні рішення для оптимізації партиціювання баз даних (БД), автоматизуючи складні процеси та підвищуючи продуктивність. Замість ручного налаштування, ШІ використовує машинне навчання для аналізу закономірностей використання даних, допомагаючи визначити оптимальні стратегії розподілу [23, 24].

Для оптимізації партиціювання ШІ аналізує і за результатами аналізу оптимізує розподіл даних між окремими розділами БД:

- Автоматизований аналіз навантаження. Алгоритми ШІ можуть постійно відстежувати патерни запитів та інтенсивність використання різних частин БД. Це дозволяє динамічно перерозподіляти дані між партиціями для запобігання "гарячих точок" (hotspots) та балансування навантаження.
- Прогнозування майбутніх потреб. Використовуючи історичні дані, моделі машинного навчання можуть прогнозувати зростання обсягів даних та майбутні патерни доступу. Це дає змогу заздалегідь планувати

та автоматично створювати нові партиції (наприклад, для даних наступного місяця чи року), забезпечуючи стабільну продуктивність.

- Динамічний вибір ключа партиціювання (Shard Key). Традиційний вибір ключа партиціювання може бути складним і часто призводить до нерівномірного розподілу даних. ШІ може аналізувати різні атрибути даних та пропонувати або автоматично застосовувати оптимальні ключі, які забезпечують більш рівномірний розподіл, наприклад, за допомогою хеш-функцій або комбінованих методів.
- Адаптивні стратегії. Замість використання однієї статичної стратегії (як-от партиціювання за діапазоном чи списком), ШІ може адаптивно поєднувати різні методи (комполитне партиціювання), оптимізуючи як швидкість читання, так і запису залежно від поточного робочого навантаження.
- Оптимізація використання ресурсів. ШІ допомагає ефективніше використовувати апаратні ресурси, розподіляючи дані між різними серверами (при шардингу) таким чином, щоб максимізувати пропускну здатність та мінімізувати затримки [26].

Переваги використання ШІ для вирішення зазначених задач аналізу і оптимізації наступні:

- Підвищення продуктивності та масштабованості. Оптимізований розподіл даних значно покращує час відгуку запитів та дозволяє системі легше масштабуватися горизонтально.
- Зниження операційної складності. ШІ автоматизує процеси, які раніше вимагали значного ручного втручання та експертних знань адміністраторів БД.
- Вища надійність. Завдяки кращому балансуванню навантаження та ізоляції збоїв (якщо один шард виходить з ладу, інші продовжують працювати), загальна стійкість системи зростає [25].

Загалом, ШІ трансформує керування базами даних, роблячи системи

більш адаптивними, ефективними та простішими в управлінні.

1.4.2. Використання ШІ та ML для оптимізації структури БД

Штучний інтелект (ШІ) відіграє ключову роль в оптимізації структури баз даних (БД), виходячи за рамки простого партиціювання. Він використовує машинне навчання та автоматизований аналіз для переосмислення традиційних підходів до проектування схем і структур даних, індексації даних та генерації матеріалізованих представлень, що призводить до значного підвищення продуктивності, зниження витрат та спрощення керування розподіленим сховищем даних.

Оптимізація структури БД за допомогою ШІ охоплює кілька ключових напрямків: автоматизоване проектування схеми та денормалізація БД; інтелектуальне керування індексами; інтелектуальне створення представлень; адаптивне фізичне розташування даних

Автоматизоване проектування схеми та денормалізація забезпечує вирішення наступних задач:

- Традиційне проектування передбачає ручний вибір між нормалізацією (зменшення надмірності даних) і денормалізацією (пришвидшення читання за рахунок дублювання).
- ШІ може аналізувати всі запити, які виконуються до бази даних, і пропонувати оптимальні гібридні схеми. Він може автоматично ідентифікувати групи таблиць, які часто об'єднуються (JOINS), і пропонувати стратегії денормалізації або використання колоночних сховищ для цих конкретних робочих навантажень.

Інтелектуальне управління індексами здійснюється для вирішення постійної проблеми – визначення переліку «правильних» індексів. Занадто багато індексів уповільнюють запис, занадто мало – уповільнюють читання. Для інтелектуального управління індексами на рівні ШІ застосовуються наступні механізми:

- "Навчальні" (learned) індекси. Це новий підхід, коли ШІ створює компактні моделі, які фактично замінюють або доповнюють традиційні В-дерева. Ці моделі "вивчають" розподіл даних і можуть передбачати місце знаходження даних швидше, ніж класичні структури даних.
- Автоматичне видалення/створення індексів. ШІ-агенти можуть відстежувати невикористані індекси та рекомендувати їх видалення, або ж ідентифікувати повільні запити та автоматично створювати індекси, які б їх пришвидшили, без втручання адміністратора.
- Оптимізація матеріалізованих представлень (Materialized Views). Матеріалізовані представлення прискорюють складні аналітичні запити, але потребують регулярного оновлення.

Інтелектуальне створення представлень забезпечує вирішення наступного основної задачі: ШІ може аналізувати патерни запитів та визначати, які саме представлення варто створити, а також автоматично планувати оптимальний час для їхнього оновлення, щоб мінімізувати вплив на основну роботу системи [26].

Адаптивне фізичне розташування даних (Data Layout). ШІ може вирішувати, чи зберігати дані у форматі рядків (row-oriented) для транзакційних систем (OLTP), чи у форматі стовпців (column-oriented) для аналітичних систем (OLAP). Деякі сучасні системи використовують ШІ для динамічного перетворення даних між цими форматами на льоту, залежно від типу поточного робочого навантаження. Впровадження адаптивного фізичного розташування даних надає наступні переваги:

- Експоненційне зростання продуктивності: "Навчальні" індекси та інтелектуальна денормалізація можуть значно скоротити час виконання запитів.
- Зниження витрат: Оптимізована структура потребує менше обчислювальних ресурсів (процесор, пам'ять, сховище) для підтримки того ж рівня продуктивності.

- Постійна адаптація: На відміну від ручного проектування, яке є статичним, ШІ-оптимізація постійно адаптується до мінливих бізнес-потреб та патернів використання даних.

Впровадження ШІ в керування розподіленими базами даних перетворює їх із статичних сховищ інформації на самокеровані, адаптивні системи, здатні автономно оптимізувати власну внутрішню роботу. Крім того, впровадження ШІ автоматизує завдання, які раніше вимагали високооплачуваних експертів з оптимізації БД (DBA).

1.4.3. Застосування штучного інтелекту для оптимізації шляхів доступу до бази даних

Штучний інтелект (ШІ) революціонізує оптимізацію шляхів доступу до бази даних, замінюючи традиційні статичні оптимізатори запитів на динамічні, самонавчальні системи. Ці системи використовують машинне навчання для аналізу історичних даних та вибору найефективнішого способу отримання інформації в режимі реального часу.

Традиційні оптимізатори запитів покладаються на фіксовані правила та статистичні моделі вартості, які можуть стати неточними в умовах великих обсягів даних, що постійно змінюються. ШІ долає ці обмеження за допомогою таких підходів:

- Навчання на основі даних (Machine Learning Models): моделі машинного навчання (особливо навчання з підкріпленням та навчання з вчителем) навчаються на основі даних про виконання тисяч попередніх запитів. Вони аналізують різні фактори, такі як структура запиту, розмір таблиць, наявність індексів та фактичний час виконання, щоб передбачити найефективніший план виконання (Query Execution Plan, QEP).
- Адаптивна оптимізація в реальному часі (Adaptive Query Optimization): на відміну від статичних планів, які обираються один раз на початку

виконання, ШІ-системи можуть адаптувати стратегію виконання запиту на льоту. Якщо під час виконання виявляється, що обраний шлях неоптимальний (наприклад, через несподіваний розподіл даних), ШІ може динамічно перемкнутися на більш ефективний план.

- "Навчальні" індекси (Learned Indexes): ШІ може створювати спеціалізовані моделі, які виступають як ефективні заміники або доповнення до традиційних структур індексів (таких як B-дерева). Ці моделі "вивчають" розподіл даних і можуть передбачати місце знаходження записів з більшою точністю та швидкістю.
- Прогнозування навантаження (Workload Prediction): ШІ аналізує історичні патерни використання БД, щоб передбачити майбутнє навантаження та можливі вузькі місця. Це дозволяє системі проактивно розподіляти ресурси в розподіленій інформаційній системі або оптимізувати індекси ще до виникнення проблем.
- Зниження ручної роботи адміністраторів БД: ШІ автоматизує рутинні завдання з налаштування продуктивності, визначення оптимальної локалізації даних, керування індексами та моніторингу, дозволяючи адміністраторам зосередитися на стратегічних ініціативах.
- Підвищена відмовостійкість та адаптивність: система стає більш стійкою до змін у патернах даних та запитів до сховища даних, або робочому навантаженні без необхідності ручного переналаштування системи.
- Демократизація доступу: інструменти на основі ШІ (наприклад, AI2SQL) дозволяють користувачам без глибоких знань SQL створювати високоефективні запити, використовуючи природну мову.

Отже, ШІ надає значні переваги, у тому числі - значне прискорення виконання запитів: компанії, що впроваджують ШІ-оптимізацію, повідомляють про покращення швидкості запитів, інколи від 10х до 140х разів. Інтеграція ШІ в оптимізацію шляхів доступу є ключовим кроком до

створення повністю автономних (self-tuning) баз даних, які можуть самостійно керувати своєю продуктивністю та ефективністю.

1.4.4. Проблеми впровадження ІІІ та ML в корпоративних інформаційних системах

Впровадження ІІІ та ML в корпоративні інформаційні системи супроводжується низкою суттєвих проблем, які можна розділити на кілька ключових категорій: проблеми з даними; технічні та технологічні проблеми, організаційні проблеми та проблеми, що визначаються людським фактором; проблеми, пов'язані з правом; фінансові проблеми.

Проблеми з даними визначаються наступним переліком основних чинників:

- Якість та доступність даних. Алгоритми ІІІ потребують великих обсягів високоякісних, чистих та релевантних даних для ефективного навчання. Багато компаній мають розрізнені дані низької якості, що зберігаються в ізольованих системах, що ускладнює їх використання.
- Підготовка даних. Процес збору, очищення, маркування та трансформації даних є надзвичайно трудомістким, дорогим і може займати до 80% часу проекту ІІІ.

Технічні та технологічні проблеми визначаються наступними основними чинниками:

- Складність інтеграції. Нові рішення ІІІ необхідно інтегрувати з існуючими застарілими (legacy) корпоративними системами, що може бути складним і вимагати значних змін в ІТ-інфраструктурі.
- Вибір правильних інструментів. Існує величезна кількість інструментів, фреймворків та платформ для ІІІ. Вибір найбільш відповідного стеку технологій для конкретних бізнес-завдань може бути складним завданням.
- Потреба у високопродуктивному обладнанні. Деякі моделі ІІІ

(особливо глибоке навчання) потребують значних обчислювальних ресурсів, таких як потужні графічні процесори (GPU) або хмарні обчислення, що тягне за собою додаткові витрати.

Організаційні та людські фактори визначаються наступними основними чинниками:

- Нестача кваліфікованих фахівців. На ринку праці спостерігається дефіцит спеціалістів із ШІ, таких як інженери з машинного навчання, аналітики даних та фахівці з Data Science. Цей чинник є тим більш істотним зважаючи на військові дії і стан війни в Україні починаючи з 2014 року і по теперішній час.
- Опір змінам. Співробітники можуть чинити опір автоматизації процесів, побоюючись втрати робочих місць або необхідності перекваліфікації. Для успішного впровадження потрібна ефективна програма керування змінами (Change Management) та навчання персоналу.
- Відсутність чіткої стратегії. Багато компаній починають проєкти ШІ без чітко визначених бізнес-цілей, очікуваної рентабельності інвестицій (ROI) або довгострокової стратегії впровадження.

Етичні, правові та регуляторні проблеми визначаються наступними основними чинниками:

- Упередженість (Bias) моделей. Моделі ШІ можуть успадковувати упередження з даних, на яких вони навчалися, що може призвести до дискримінаційних рішень (наприклад, при найманні на роботу чи видачі кредитів).
- Прозорість та пояснюваність (Explainability). Деякі складні моделі ШІ (так звані "чорні скриньки") не дозволяють чітко зрозуміти і пояснити логіку їх рішень. У певних галузях (наприклад, фінанси, медицина, транспорт, національна безпека) це є неприйнятним з регуляторної точки зору.

Проблеми з безпекою та конфіденційністю визначаються наступними основними чинниками:

- **Нормативні обмеження.** Впровадження ІІІ часто передбачає роботу з чутливими даними, що вимагає дотримання суворих нормативних вимог (наприклад, GDPR в ЄС або законів про конфіденційність в Україні) та забезпечення надійного захисту від компрометації даних шляхом їх спотворення або витоків.
- **Відповідальність.** Визначення того, хто несе відповідальність за помилки, допущені системою ІІІ, залишається складним юридичним питанням.

Фінансові проблеми визначаються наступним переліком основних чинників:

- **Висока вартість.** Початкові інвестиції у впровадження ІІІ (ліцензії, обладнання, зарплати фахівців, послуги консультантів) можуть бути дуже значними.
- **Складність оцінки ROI.** Не завжди легко точно спрогнозувати фінансову віддачу від інвестицій у ІІІ до завершення пілотного проєкту.

Успішне подолання зазначених проблем вимагає комплексного підходу, який поєднує технологічні рішення з увагою до бізнес-процесів, корпоративної культури та етичних аспектів.

1.5. Постановка задачі дослідження

На основі зазначених чотирьох критеріїв та прикладів їх застосування для класифікації проєктів можна зробити висновок, що будь які проєкти можуть бути класифіковані в системі з чотирьох критеріїв. Це значить, що за наявності двійкової оцінки за кожним з критерієм існує всього 16 типів проєктів

Виходячи з результатів аналізу виконаємо постановку задачі

дослідження у наступному формулюванні: визначити можливості науково-обґрунтованого підходу до вибору адаптивних технологій організації сховищ даних розподілених інформаційних систем і механізмів доступу до даних для забезпечення оптимальної щодо вирішуваних у системі задач локації даних і засобів їх обробки, та можливостей їх застосування в проектах розподілених інформаційних систем, що дозволить з наявного спектру інструментального забезпечення, методів та засобів збереження, транспортування, захисту та обробки даних виконувати розробку таких систем з найкращими експлуатаційними характеристиками.

1.6. Висновки до розділу 1

В розділі були розглянуті технології організації і доступу до даних у корпоративних розподілених інформаційних системах, що визначаються організацією сховищ даних, інструментальним забезпеченням, методами та засобами збереження, передачі, захисту та обробки даних корпоративних ІС.

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

Виконано постановку задачі.

2. РОЗРОБКА МОДЕЛЕЙ ТА МЕХАНІЗМІВ ЕФЕКТИВНОГО ДОСТУПУ ДО ДАНИХ В ПРОЕКТАХ РОЗПОДІЛЕНИХ АІС

2.1. Визначення критеріїв оцінки методів та засобів розробки проектів розподілених АІС

2.1.1. Визначення критеріїв оцінки методів та засобів збереження та доступу та даних в корпоративних АІС

Враховуючи проблеми, пов'язані з бізнес-логікою і обробкою даних, обсягом даних та обсягом запитів до СКБД, сховища даних та СКБД розподілених інформаційних систем можуть розглядатись як:

- системи з великими обсягами даних;
- системи зі складною організацією даних;
- системи з великим обсягом запитів (як правило – зі значним числом користувачів).

Візуально різновиди сховищ даних і СКБД проектах розподілених інформаційних систем можуть бути представлені у вигляді моделі (рис 2.1).

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

- Відповідності – сервіси та протоколи за своїми функціональними можливостями повинні відповідати призначенню проекту, наприклад, в комерційних проектах повинні застосовуватися захищені канали передачі даних, сертифікація, стиснення даних тощо;
- Критерій розвитку (в даному випадку стабільні краще);
- Критерій підтримки СКБД інструментальними засобами (бажана підтримка сучасних мов програмування);
- Критерій вартості входження та супроводу СКБД (прості і безкоштовні краще у зв'язку з коротким часом життя проектів);

- Критерій підтримки інструментів з сучасними технологіями програмування (кращі ті інструменти, що відносно простими засобами дозволяють реалізувати більш складну бізнес логіку, наприклад з підтримкою технологій метапрограмування);
- Критерій простоти реорганізації модифікації структури бази даних (так, 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, може однотипних, але маленьких).

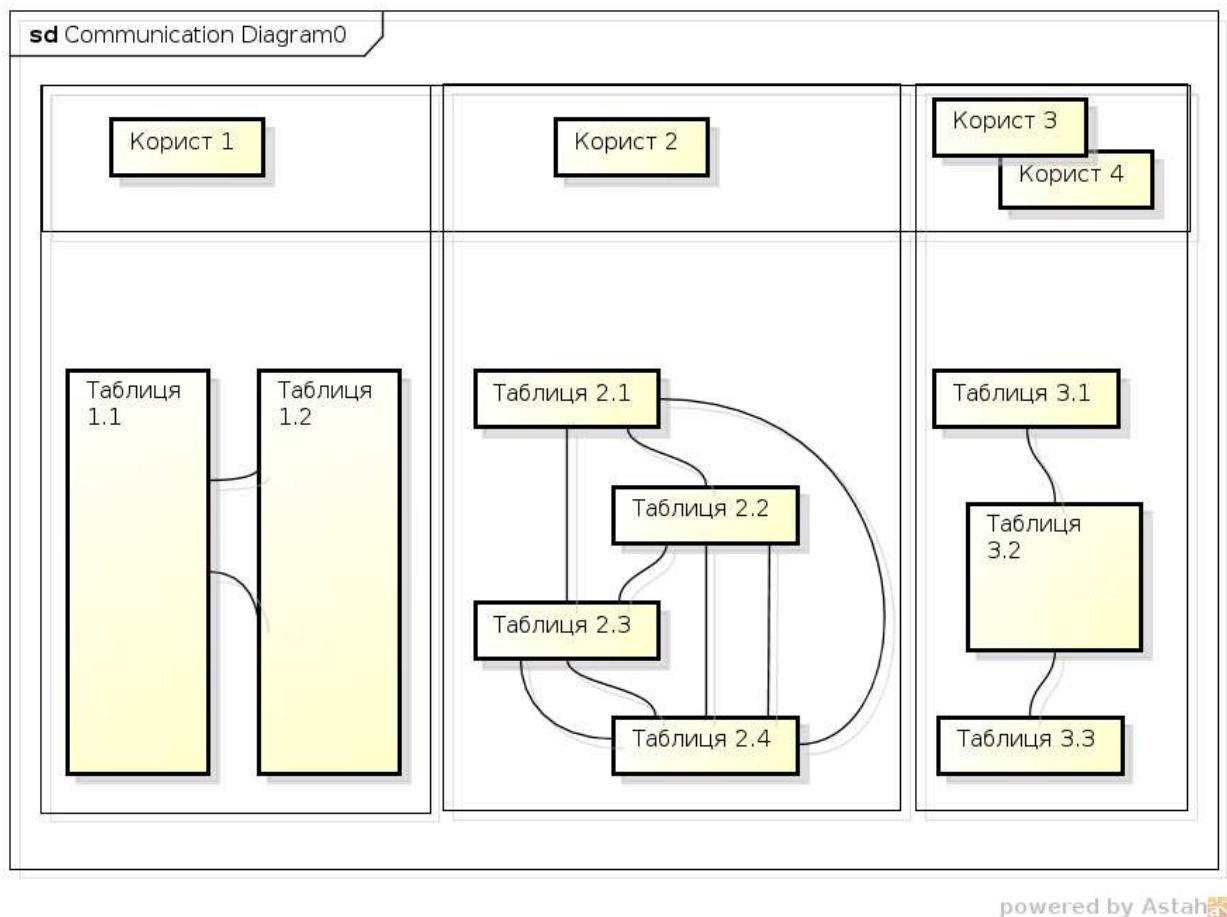


Рисунок 2.1 – Різновиди СКБД проектів розподілених інформаційних систем: з великим обсягом даних (ліворуч), зі складною структурою (у центрі), зі значним числом користувачів (праворуч)

2.1.2. Визначення критеріїв оцінки методів та засобів розробки сховищ даних корпоратив ІС

Враховуючи, що однією з основних проблем вибору інструментального забезпечення для реалізації сховищ даних корпоратив ІС слід вважати специфіку задач кожного конкретного проекту з точки зору вимог особливостей методів обробки даних, що визначаються бізнес-процесами. Зважаючи на вищенаведене, визначимо наступні ключові критерії щодо застосування інструментальних засобів у сховищах даних корпоративних ІС (Таблиця 1.3).

Залежно від вимог щодо забезпечення кожної з цих областей наявні проекти розподілених інформаційних систем можуть бути поділені на категорії з різними вимогами. Нижче наведені приклади (таблиця 2.1.).

Таблиця 2.1 – Ключові критерії щодо застосування інструментальних засобів у корпоративних ІС

<i>№</i>	<i>Критерій</i>	<i>Примітка</i>
1	Збереження та надання інформаційних ресурсів	
2	Комунікації та обмін інформацією	
3	Масові обчислення	
4	Управління соціальними та технічними системами	

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.

Таблиця 2.2 – Приклади ранжування корпоративних ІС за категоріями вимог

№	Предметна область	Значення для критерію			
		Збереження та надання ресурсів	Комунікації, захист та обмін інформацією	Масові обчислення	Управління іншими системами
1	Електронні ЗМІ	високе	низьке	низьке	низьке
2	Електронна торгівля	високе	середнє	низьке	низьке
3	Навчання	середнє	середнє	середнє	середнє
4	Інженерні розрахунки	середнє	середнє	високе	низьке
5	Соціальні мережі	низьке	високе	низьке	низьке
6	Електронний уряд	високе	середнє	низьке	високе
7	Системи моніторингу та управління	високе	високе	високе	високе

2.2. Розробка моделей опису структури бази даних

Для забезпечення верифікації бази даних необхідно мати інформацію про її структуру. Тоді зміна такої інформації може бути відображена в зміні структури бази даних, а отже – автоматизована. Інформація про інформацію називається метаданими. Визначимо наступні складові такої інформації (Рис. 2.2):

- Метакласи (угруповання сутностей) дозволять об'єднувати сутності предметної області в групи. Застосування: сутності за своєю природою

можуть бути не однорідні, вони відносяться до різних рівнів абстракції однієї предметної області або ж до різних предметних областях, які обслуговує однією базою даних або ж ми хочемо виділити в предметній області кілька кластерів або зон [14]. Варіанти вирішення: використання префіксів в іменах таблиць, розфарбування структур баз даних різними кольорами, але частіше за все про групи «знає» безпосередньо програмне забезпечення, яке працює з базою даних, угруповання «зашиті» в код і модель без коду не повна.

- Угруповання атрибутів сутностей в смислові блоки, пов'язані функціонально, семантично або відносяться до різних аспектів моделювання сутностей. Застосування: наприклад, буває зручно згрупувати поля, що відносяться до адресою (країна, місто, вулиця, будинок, індекс і тому подібне) або два поля описують один атрибут (як широта і довгота описують атрибут «Координата»). Варіанти вирішення: введення префіксів в імена полів, перерахування полів по порядку, але в основному, цей аспект інформаційної моделі «зашивається» в призначений для користувача інтерфейс і ми бачимо, що модель не повна без інтерфейсу.
- Метакласи зв'язків (угруповання зв'язків) – зв'язки між таблицями не завжди іменовані, тому угруповання зв'язків по іменах практично не зустрічається, але є завдання, до яких вона вкрай необхідна. Застосування: приклад #1 – особи поєднані у відношенні «багато-до-багатьох» через перехресну таблицю і нам потрібно виділити частину зв'язків як «родинні», а частина як «дружні» або «робочі». Є ще один поширений випадок (приклад #2) угруповань, коли є кілька перехресних таблиць багато-до-багатьох, і нам потрібно згрупувати власників транспортних засобів, власників нерухомості і власників домашніх тварин в кілька груп: колишні власники, поточні власники і особи заперечують володіння. Варіанти вирішення: для випадку #1 –

просто вводиться прапор в перехресної таблиці або додається довідник груп зв'язків. Для випадку #2 – вводити однакові прапори і довідники для різних таблиць звичайно можна, але хотілося б потім одним запитом вибирати всіх «поточних власників». Для цього ми можемо застосувати абстрактну «згортку», як описано в першій частині.



Рисунок 2.2 – Елементи компонентної моделі БД на основі метамоделі

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

2.3.1. Критерії оцінки якості технологій збереження та доступу до даних в розподілених інформаційних системах

Враховуючи визначені в попередньому розділі (пп. 1.4) критерії, що визначають якість механізмів збереження та доступу до даних, визначимо підходи до розробки моделей оцінки якості механізмів збереження та доступу до даних.

Ці критерії визначені як наступні:

- Критерій відповідності вимогам;

- Критерії захисту ресурсів;
- Критерій розвитку проекту;
- Критерій підтримки проекту інструментальними засобами;
- Критерій підтримки сучасних технологій програмування;
- Критерій простоти і вартості СКБД;
- Критерій простоти реорганізації бази даних;
- Критерій простоти модифікації структури БД.

Дамо розгорнуте трактування окремих із зазначених критеріїв.

Критерій відповідності полягає в тому, що сервіси та протоколи за своїми функціональними можливостями повинні відповідати призначенню проекту розподіленої інформаційної системи, наприклад, в комерційних проектах, повинні застосовуватися захищені канали передачі даних, сертифікація, стиснення даних і тому подібне [9].

Критерій підтримки проекту інструментальними засобами полягає в тому, що окрім поточної розробки будь яку систему необхідно підтримувати і надалі. При цьому непопулярність інструментальних засобів в майбутньому може призвести до проблем з пошуком та залученням відповідних фахівців. Як наслідок, в проекті бажана орієнтація на сучасні високорівневі мови програмування.

Критерій простоти і вартості СКБД полягає в тому, що прості і безкоштовні СКБД є кращим вибором з погляду на відносно короткий термін життя проектів розподілених інформаційних систем.

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, може однотипних, але маленьких. Приклад: корпоративний чат або інтернет форум, в якому повідомлення форуму краще зберігати не в одній великій таблиці по всіх темах форуму, а в декількох невеликих таблицях, але містять повідомлення тільки однієї теми.

Критерій простоти модифікації структури БД визначається тим, що у зв'язку з високим навантаженням і коротким часом життя проектів розподілених інформаційних систем структура баз даних часто зазнає змін через відносно спрощені технології початкового проектування ПЗ корпоративних автоматизованих інформаційних систем.

Визначимо, що оцінка якості проектів має враховувати два основні аспекти:

- якість використання програмної системи, тобто система має задовольняти експлуатаційним вимогам;
- якість розробки програмної системи, тобто система має задовольняти технологічним вимогам.

Виходячи з того, що система, яка не може використовуватись, має нульову якість так само як і система, що не може бути реалізована, визначаємо, що ці аспекти можуть бути враховані в загальній оцінці якості як середнє геометричне за кожним з аспектів:

$$c = \sqrt{c_u * c_d}, \quad (2.1)$$

де: c – оцінка, c_u – відповідність вимогам замовника; c_d – відповідність вимогам розробника.

У свою чергу, відповідність моделі доступу вимогам замовника визначається відповідністю механізмів доступу до даних та механізмів збереження даних:

$$c_u = \sqrt{c_{ua} * c_{us}}, \quad (2.2)$$

де: c_{ua} – коефіцієнт відповідності вимог доступу, c_{us} – коефіцієнт відповідності вимог збереження.

Зважаючи на результати аналізу, визначаємо, що відповідність вимогам розробника забезпечується відповідністю механізмів збереження та доступу до даних залежно від предметної області задачі, а відповідність вимогам розробника – відповідністю інструментальних засобів (в тому числі мови

програмування) щодо можливостей забезпечення збереження та доступу до даних.

2.3.2. Механізми забезпечення якості збереження даних

Відповідно до визначених вище критеріїв якості, згідно до таблиці 2.1 та (2.2) визначимо значення критерію c_s як сумму:

$$c_{us} = k_o * c_{uo} + k_t * c_{ut} + k_m * c_{um}, \quad (2.3)$$

де: c – коефіцієнт значення критерію за експлуатаційними вимогами для проекту відповідно до його предметної області, k – ваговий коефіцієнт; o , t , m – значення щодо обсягу, часу виконання запитів та складності бізнес логіки.

Коефіцієнти k , c будуть визначатись експертним шляхом на основі усереднення оцінок, наданих експертами.

Серед основних механізмів забезпечення якості в системах збереження даних найбільш цікавим, з точки зору технологій створення розподілених інформаційних систем, є масштабування баз даних – партиціювання, реплікація і шардинг.

СКБД – це дуже часто «вузьке місце» в продуктивності додатків розподілених інформаційних систем. У момент, коли сервер баз даних не може впоратись з навантаженнями, проводиться масштабування. Вище були розглянуті основні способи збільшення продуктивності СКБД. Розглянуті механізми масштабування застосовні як для реляційних баз даних, так і для NoSQL-сховищ.

Зрозуміло, що у всіх СКБД, баз даних і сховищ є своя специфіка, тому додатково і більш детально розглянемо тільки основні напрямки без акценту на деталі реалізації.

Партиціювання (partitioning) – це розбиття таблиць, що містять велику кількість записів, на логічні частини за деякими обраним адміністратором критеріям. партиціювання таблиць ділить весь обсяг операцій з обробки

даних на кілька незалежних і паралельно виконуються потоків, що істотно прискорює роботу СКБД. Для правильного конфігурування параметрів партиціювання необхідно, щоб в кожному потоці було приблизно однакову кількість записів [26].

Наприклад, на новинних сайтах має сенс партиціонувати 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 [17].

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 проектів розподілених інформаційних систем операцій читання на кілька порядків більше, ніж операцій запису. Однак у більш складній схемі реплікації може бути і кілька серверів-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 локації 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 серверів БД, до яких відносяться БД проектів розподілених інформаційних систем, 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 дуже гарній налаштування додатки, може дати час відгуку порядку декількох мілісекунд. Однак для багатьох додатків (наприклад, для систем, що працюють в реальному часі) потрібно швидкодію 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 тощо. 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). Багато хто не дуже важливі 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. Наприклад, Oracle сьогодні 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 дискової СКБД можна підвісити 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), але кешується за певною політиці лише частину даних. Важливо, що для роботи з використанням такого кешу не потрібне спеціальне програмування, можна використовувати стандартні протоколи (ODBC, JDBC), звичну мову дискової СКБД.

2.3.3. Механізми забезпечення якості доступу до даних

Відповідно до визначених вище критеріїв якості, згідно до пункту 1.4.2 та (2.2) визначимо значення критерію c_a як сумму:

$$c_{us} = k_b * c_{ub} + k_p * c_{up}, \quad (2.4)$$

де: $k_b * c_{ub}$ — ваговий коефіцієнт та значення критерію безпеки передачі даних;
 $k_p * c_{up}$ — ваговий коефіцієнт та значення критерію доступності протоколів передачі 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 програмування та проектування. 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.

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.

Крім цього, 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 керування 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 на сайтах, де користувачі можуть публікувати свій контент, роблячи його доступним для інших. Зловмисник, при відсутності належної фільтрації, може або безпосередньо використовувати уразливість, або заражати системи користувачів. Розробники веб проектів для запобігання атак цього типу змушені або просто обмежувати можливості користувачів публікувати на сайті що-небудь, або ретельно фільтрувати надходять від користувачів дані.

2.3.4. Механізми доступу та збереження даних клієнта

Браузерне або клієнтське кешування. Кешування статичних ресурсів (картинок, скриптів, стилів) і незмінних сторінок на стороні браузера може заощадити час завантаження сторінок, якщо користувач відвідує сайт багаторазово або при відвідуванні переглядає кілька сторінок, які використовують однакові ресурси.

Найчастіше на стороні браузера кешуються файли зображень, JS і CSS файли. Трохи рідше має сенс кешувати сторінки і бінарні файли (медіа-файли, PDF, завантажувані документи, архіви і т. д.).

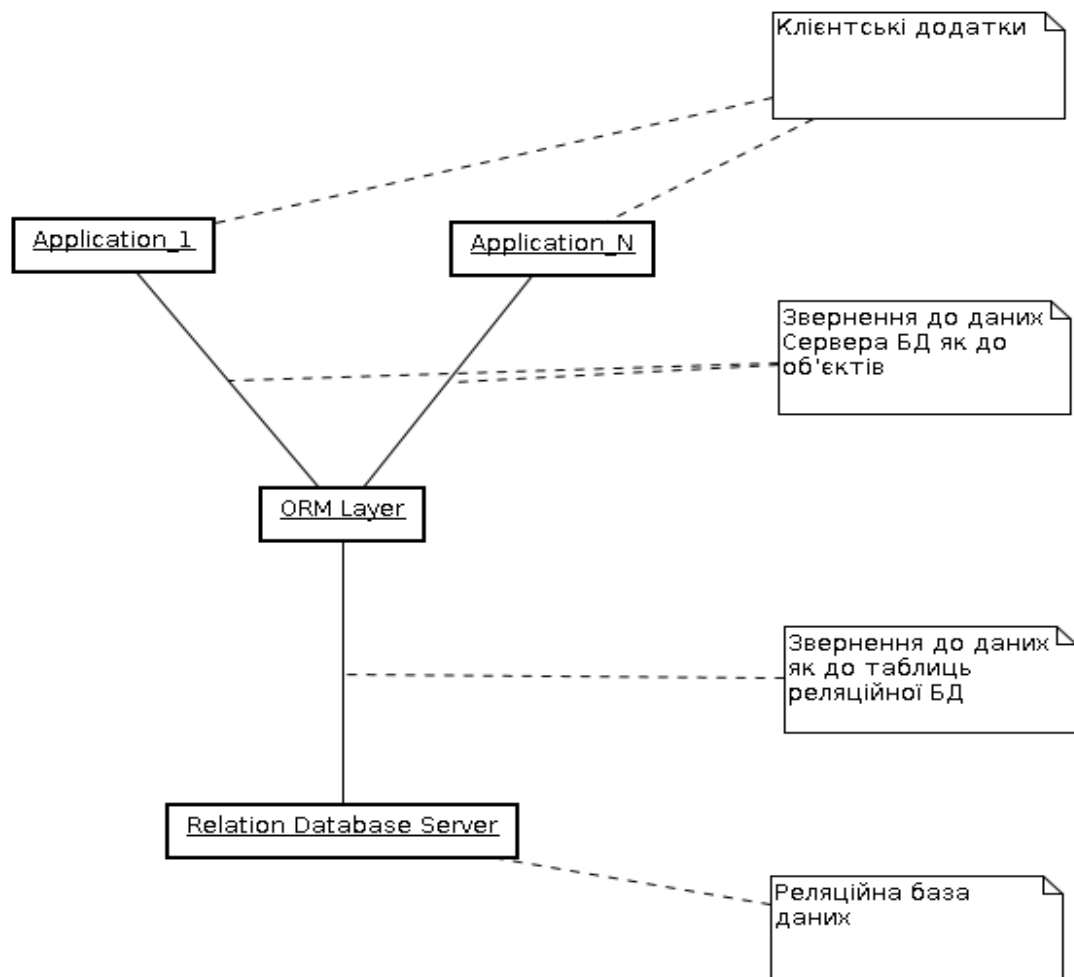


Рисунок 2.3 – Компонентна модель доступу до реляційної бази даних з Object Relational Mapping (ORM)

Статичні ресурси повинні мати хоча б тижневе час життя кешу, а краще кешувати їх відразу на місяць – з врахуванням сезонного характеру окремої інформації. Таким чином, ці ресурси будуть тільки один раз отримують з сервера, а потім браузер буде або відразу використовувати локальну копію (якщо вказано заголовок Expires або Cache-Control), або після перевірки на незмінність (якщо вказано заголовок Last-Modified або Etag).

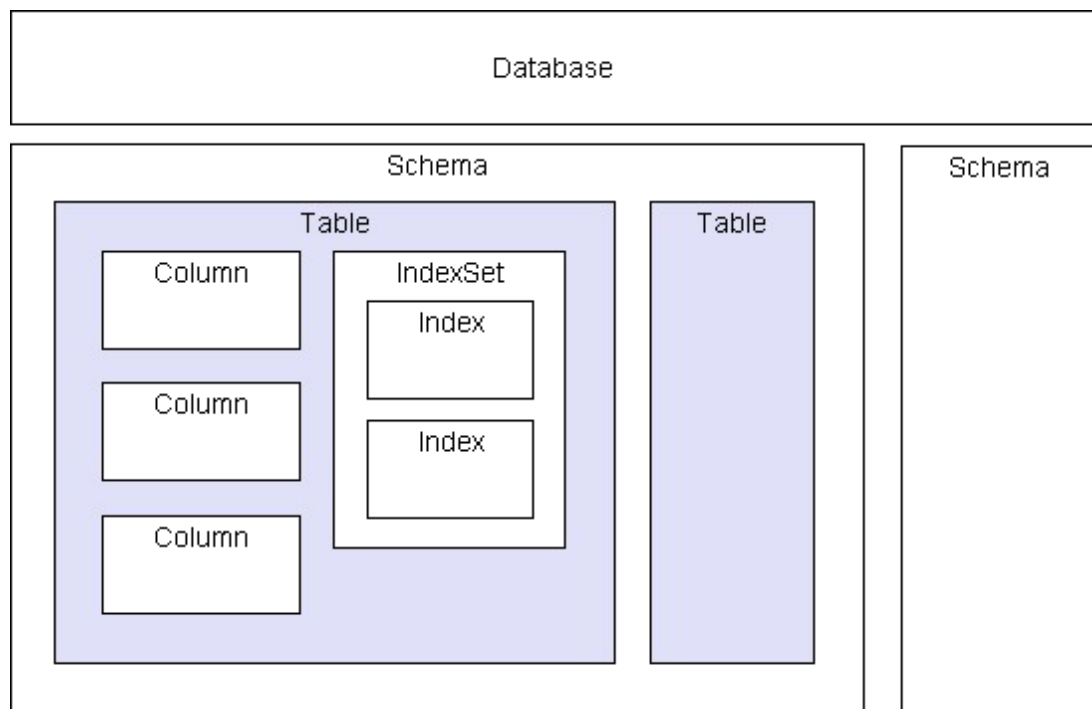


Рисунок 2.4 – Логічна структура даних в ORM-системі збереження даних

Заголовки Expires і Cache-Control: max-age. Як значення у цих заголовків використовується дата. До тих пір, поки вона не настала, браузер буде без будь-яких додаткових перевірок використовувати кешовану версію ресурсу. Expires підтримується трохи ширше, ніж Cache-Control: max-age, тому краще використовувати саме його.

Заголовки Last-Modified і Etag. Як значення заголовка Last-Modified використовується дата, а для заголовка ETag – довільна рядок. Якщо

використовуються ці заголовки, то браузер, перш ніж використовувати кешований ресурс, отримає з сервера поточне значення заголовка і порівняє його з заголовком кешованої версії – якщо дані співпадуть, то буде використовуватися локальна версія, а якщо ні – відбудеться повторне завантаження. Запити перевірки відбуваються швидше, ніж повне завантаження ресурсів, що дає приріст продуктивності сайту при повторному використанні ресурсів.

Таблиця 2.3 – Типи та періодичність кешування ресурсів інтернет-сторінок на стороні клієнта

<i>№</i>	<i>Типи ресурсів</i>	<i>Призначення</i>	<i>Період</i>
1	Файли зображень	Медійна інформація	1 місяць
2	Скрипти JS	Анімація, обробка даних на стороні клієнта	1 місяць
3	CSS файли	Стили представлення, оформлення	1 місяць
4	Статичні сторінки	Інформація, що змінюється розробником контенту	1 день – 1 тиждень
5	Бінарні файли	Інформація, що змінюється сервером за запитом	до 1 дня

Оптимальна стратегія клієнтського кешування загалом може бути наступною. Для значно і неочікувано мінливих файлів варто поставити Expires на дату, яка настане через місяць, а в момент зміни цих ресурсів використовувати метод «відбитків пальців» (fingerprinting) – додавати до імені файлу невелику частину хешу, яка утворюється на основі його змісту. Таким чином, вся статика завжди буде кешована на клієнтському робочому місці, а при зміні цієї самої статички будуть змінені імена файлів і вони

автоматично будуть перезавантажуватися клієнтським додатком.

Таблиця 2.4 – Елементи сторінки, що впливають на періодичність кешування ресурсів інтернет-сторінок на стороні клієнта

№	Типи заголовку	Тип	Примітки
1	Expires	дата	
2	Cache-Control: max-age	дата	Обмежена підтримка
3	Last-Modified	дата	
4	Etag	рядок	

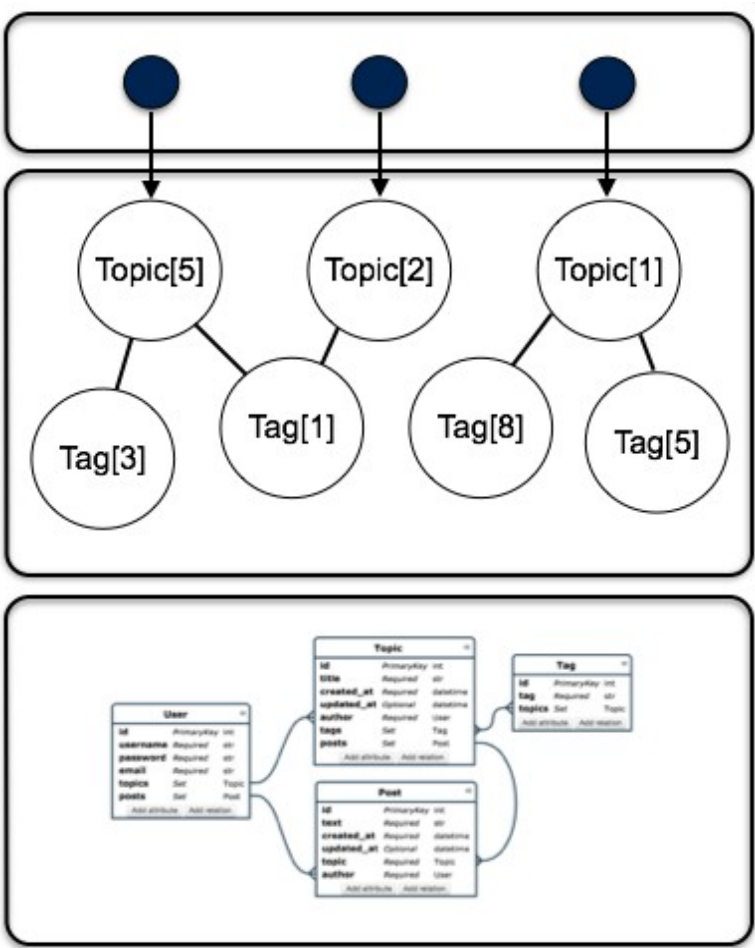


Рисунок 2.5 – Фізична структура даних в ORM-системі (зверху-донизу: рівень доступу; рівень організації, рівень збереження)

Власне техніка *fingerprinting* активно використовується для статичних ресурсів (стилів і скриптів) у багатьох сучасних фреймворках (зокрема, в Django для Python та Ruby on Rails для мови Ruby). А масові для всіх типів статичних файлів виставлення і заповнення заголовку *Expires* можна реалізувати на стороні інтернет-сервера (наприклад, на сервері *nginx* або подібному проксі-сервері).

2.4. Розробка моделей адаптивного партиціювання сховища даних корпоративної ІС

2.4.1. Розробка моделі збирання даних про виконання запитів до сховища даних

Для визначення можливостей забезпечення адаптивного партиціювання даних у сховищі даних розподіленої ІС розглянемо наступний приклад запиту до сховища:

```
SELECT city_id, store_id, store, total_sales,  
DENSE_RANK()  
OVER (PARTITION BY city ORDER BY total_sales DESC) AS  
sales_rank  
FROM store_sales  
ORDER BY city, total_sales DESC;
```

Як видно з такого запиту, звернення здійснюється до даних з двох таблиць реляційної СКБД з використання поділу даних на партиції. У найбільш простому і прийнятному з точки зору обробки запиту всі необхідні дані містяться в одному розділі розподіленої БД, а всі дані таблиць — в одній партиції. Але, при наявності значної кількості даних хоча б у одній з таблиць час виконання такого запиту може бути значним через необхідність виділення під обробку запиту значного обсягу оперативної пам'яті. Саме тому стають можливими і інші, дещо гірші з точки зору часу і споживаних ресурсів,

комбінації варіантів розподілу даних:

- дані з однієї таблиці містяться одному розділі розподіленої БД, дані з іншої таблиці — в іншому розділі;
- дані кожної з окремих таблиць містяться в декількох окремих партиціях.

Саме для зменшення часу обробки запиту і обсягу споживаних ресурсів і здійснюється поділ даних окремих таблиць за партиціями. Наприклад, для поточного запиту такий поділ міг би бути здійснений за ідентифікатором міста (`city_id`).

Зважаючи, що структура даних корпоративної ІС зазвичай є вкрай складною і може складатись з сотень і тисяч таблиць БД постає складна задача визначення такого поділу даних між окремими сховищами даних розподіленої інформаційної системи та розподілу даних між окремими файлами, що дозволить з високою ймовірністю для більшості запитів отримувати необхідні дані з найближчого сховища і з мінімальної кількості партицій кожної з таблиць. Складність такої задачі визначається головне тим, що структура запитів відповідає поточній й операційній діяльності компанії і ця діяльність постійно (за днями тижня, декадами, сезонами року тощо) змінюється. Це значить, що дані про виконання запитів до сховища даних, які можуть бути зібрані, виявляться вкрай нечіткими. Наприклад, інформація про виконання таких запитів до сховища могла б бути представлена у табличній формі (Таблиця 2.5) [19].

Як наслідок, потрібні механізми обробки статистики звернень до сховища даних, що дозволили б на основі зібраної статистики прийняти зважені рішення щодо розподілу даних у розподіленому сховищі та у партиціях БД.

2.4.2. Математична модель оптимізації партиціювання таблиць БД

Для відомих t_i , c_i , s_i , p_i знайти таку функцію оцінки часу виконання

запитів F від $[c_i, s_i, p_i]$ та $[t_i]$:

$$t_i = F(c_i, s_i, p_i), \quad (2.5)$$

що дозволить обчислити доцільність віднесення даних до окремої партиції x при забезпеченні мінімального часу обробки запиту, а отже і доступу до даних:

$$x \in [P] \forall x | F(c_i, s_i, p_x) \rightarrow \min \quad (2.6)$$

Таблиця 2.5 – Приклад представлення статистичних даних щодо виконання запитів до сховища даних

N	city_id	store_id	PARTITION_id	Time
...	...	...	...	...
n	cn	sn	pn	tn
n+1	cn+1	sn+1	pn+1	tn+1
...	...	...	...	...

Оскільки дані про результати виконання запитів до сховища даних ІС є експериментальними і залежать від багатьох інших чинників, що не можуть бути коректно враховані (загальне навантаження на систему, завантаженість мережі, поточна конфігурація апаратних засобів, наявність апаратних збоїв тощо), моделювання цільової функції доцільно здійснити засобами штучних нейронних мереж (рисунок 2.6) [6, 7, 22].

2.5. Розробка моделей вибору методів та засобів збереження та доступу та даних в корпоративній ІС

Вище були визначені критерії оцінки методів та засобів збереження інформації, які при бінарному значенні («потрібні», «непотрібні») дозволяють визначити вимоги до будь якого набору даних або його складових з 16 (або $2^{**}4$) можливих типів даних у наборі (первинна інформація, ключі, індекси

тощо) [19]. Очевидно, що необхідні механізми визначення наслідків порушення цих критеріїв для визначення напрямків та заходів, що мають бути застосовані при покращенні роботи як самого сховища даних так і кожного окремого клієнського місця корпоративної розподіленої АІС.

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 дозволяють обрати рішення, щоб найкращим чином досягти визначених критеріїв.

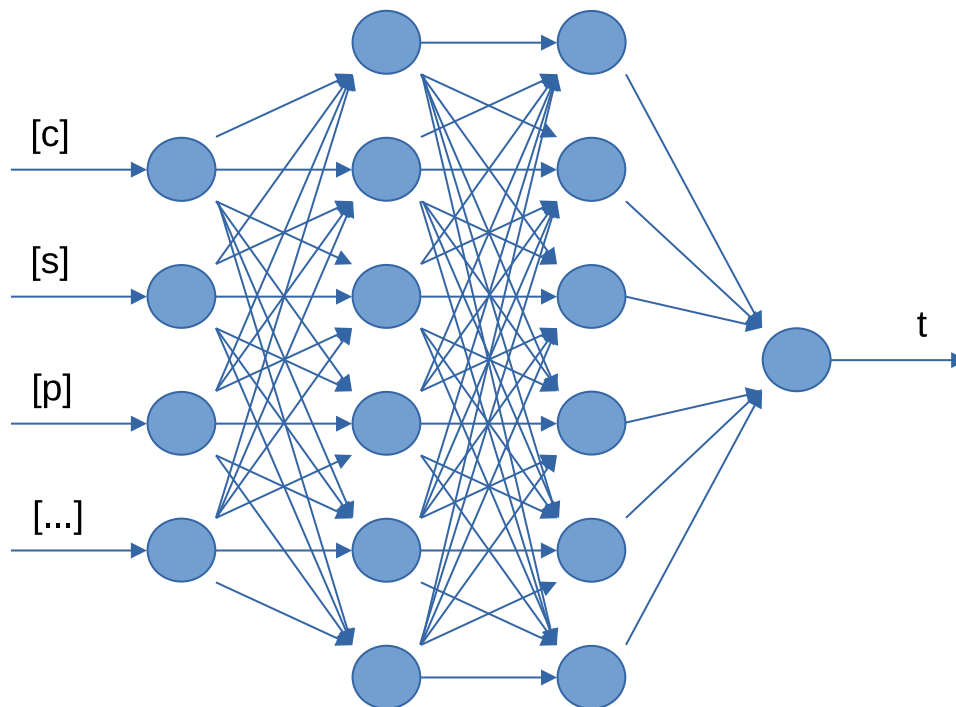


Рисунок 2.6 – Модель ШНМ для вирішення задачі раціонального партиціювання даних у БД

Очевидно, що для кожної з окремих предметних областей значення цих вагових коефіцієнтів будуть відрізнятись (що було показано вище в п. 1.4.2). В якості прикладу розглянемо окремі проекти з двох предметних областей (табл. 2.6).

Таблиця 2.6 – Основні значення параметрів варіантів ІС за критеріями оцінки

<i>Критерій</i>	<i>Навчання</i>	<i>Соціальні мережі</i>
Збереження та надання інформаційних ресурсів	0.5	0.2
Комунікації та обмін інформацією	0.6	0.8
Масові обчислення, обробка медійних файлів	0.4	0.2
Управління соціальними та технічними системами	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 наведено для мов програмування C/C++ та Python).

Також визначимо якості кожного з інструментальних засобів розробки стосовно розробників системи (табл. 2.6 – 2.8, для прикладу для мов C/C++ та Python, Табл. 2.9 для інших мов). Також цікавим є порівняння популярності мов програмування на поточний момент і, скажімо, станом на 10 років тому (Табл. 2.10).

2.6. Вибір інструментального забезпечення розробки сховищ даних корпоративних ІС

Класичний підхід передбачає розробку структур баз даних, де всі сутності інформаційної моделі знаходяться на одному абстрактному рівні, є однорідними. Однак, складні і слабо-структуровані предметні області призводять реляційну декомпозицію до комбінаторному вибуху, непропорційного зростання кількості таблиць і зв'язків. А динамічні

предметні області, в яких щоденні зміни є нормою життєвого циклу, вимагають постійного реінжиніринга структури реляційної бази даних.

Таблиця 2.7 – Основні параметри інструментальних засобів щодо забезпечення властивостей системи

<i>Критерій</i>	<i>C/C++</i>	<i>Python</i>
<i>Збереження та надання інформаційних ресурсів</i>	<i>0.5</i>	<i>0.5</i>
<i>Комунікації та обмін інформацією</i>	<i>0.6</i>	<i>0.6</i>
<i>Масові обчислення і обробка медійних файлів</i>	<i>0.9</i>	<i>0.2</i>
<i>Управління соціальними та технічними системами</i>	<i>0.4</i>	<i>0.2</i>

Таблиця 2.8 – Основні параметри інструментальних засобів щодо складності розробки

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

За таких умов, підвищення рівня абстрагування структур даних може вирішити задачу тільки частково, адже переходячи від конкретики до абстрактної моделі втрачається специфіка предметної області. Отже, зберігати в одній базі даних потрібно два, логічно пов'язаних шару абстракції.

Логічну же зв'язку повинен виконувати шар метамоделей, який визначає параметри взаємно однозначного відображення одного абстрактного шару моделі в інший.

Таблиця 2.9 – Розподіл популярності мов програмування в Україні (дані на осінь 2025 року)

Позиція	Мова	Коментар
1	TypeScript	Популярний для великих, масштабованих front-end та full-stack проєктів.
2	Python	Є ядром для розробки в сферах Data Science, AI/ML, а також активно використовується для бекенду.
3	JavaScript	Є фундаментальною мовою для веб-розробки і широко використовується як основна або додаткова мова у багатьох проєктах [2,3].
4	C#	Популярний для розробки корпоративних додатків та ігрової індустрії (платформа .NET).
5	Java	Популярний у великих корпоративних та бекенд-системах.

Таблиця 2.10 – Розподіл проєктів за мовами програмування в Україні (дані на початок 2014 року)

№	Мова	Доля ринку, %	Зміна, %	Робочий на 2014	Перспект. доля ринку	Планують на 2014	Не комерц. проєкти	Задоволення, %
1	Java	24.50	-1.60	719	24.25	828	731	79
2	C#	21.34	0.00	626	21.38	730	652	87
3	PHP	15.23	0.70	447	9.70	331	603	61
4	C++	8.62	-1.80	253	7.50	256	410	70
5	Python	6.65	0.30	351	10.28	351	448	86
6	JavaScript	5.45	1,75	160	5,71	195	804	60
7	Ruby	3.13	-0.69	92	5.53	189	204	79
8	Perl			23		17	55	60

Перехід до більш високого ступеня абстракції дозволяє виділити

сутності вищого порядку, які в структурі бази даних перетворюються на таблицю з рекурсивної посиланням. З такої таблиці можливо розгорнути ієрархію з необмеженою вкладеністю. Але при узагальненні може бути втрачена семантика. Інформаційна система повинна забороняти користувачеві створювати зв'язок такого типу на логічному рівні, що є аспектом цілісності інформації і є обов'язковою функцією СКБД (Рисунок 2.7).

2.7. Визначення властивостей мов програмування, придатних для створення розподілених інформаційних систем

Враховуючи результати аналізу, наведені в розділі 1, можна стверджувати, що розробку проектів розподілених інформаційних систем на поточний момент доцільно виконувати на одній з наступних мов: РНР, Python, Perl, Ruby та Java [5]. Кожна з цих мов має розвинуті інструментальні засоби (фреймворки), написані цими ж мовами.

Хоча кожна з зазначених мов програмування має свої особливості, їх спільні переваги з погляду технологій програмування розподілених інформаційних систем наступні:

- Висока швидкість розробки. Це обумовлено як технічними особливостями фреймворку (наприклад, продумані угоди спрощують конфігурацію), так і інструментами розробки (консольні утиліти і генератори, готові бібліотеки, розширення і модулі).
- Простота реалізації складної бізнес-логіки – конвенції написання програмного коду на базі цих мов дозволяють створювати зрозумілий програмний код, який згодом може бути без проблем модифікований в адекватні терміни, а дотримання цих конвенцій робить програмний код супроводжуваним не тільки початковим розробником, але і будь-яким іншим спеціалістом або будь-якою іншою командою. Відсутність прив'язки до основного розробника є вкрай важливим у складних проектах.

- Висока надійність і супроводжуваність рішень – в цих мовах 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 нічого не зламається. Для бізнес-проектів ця складова вкрай важлива – від стабільності роботи додатка часто залежить ефективність роботи бізнесу в цілому.
- Масштабованість, продуктивність – 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 інтернет-сервісів розподілених інформаційних 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 корпоративних інформаційних систем. Чим більше є компетентними розробники, тим швидше (дешевше) розробляється проект і тим він більш продуманий, масштабований і надійний.

Будемо вважати, що інтегральна оцінка кожного з інструментів може бути обрахована як середнє геометричне якостей кожного окремого продукту (таблиця 2.5):

$$c_d = k_e * c_e + k_q * c_q + k_v * c_v + k_w * c_w + k_r * c_r, \quad (2.7)$$

де: $k_e * c_e$ – ваговий коефіцієнт та коефіцієнт якості щодо захищеності; $k_q * c_q$ – ваговий коефіцієнт та коефіцієнт якості щодо швидкості; $k_v * c_v$ – ваговий

коефіцієнт та коефіцієнт якості щодо наочності; $k_w * c_w$ – ваговий коефіцієнт та коефіцієнт якості щодо швидкості обробки запитів; $k_r * c_r$ – ваговий коефіцієнт та коефіцієнт якості щодо валідності.

В результаті виконання серії підстановок з (2.7), (2.4), (2.3), (2.2), в (2.1) отримаємо формулу розрахунку значення оцінки якості проекту.

Для визначення вагових коефіцієнтів можуть бути взяті дані з таблиць 2.3, 2.4, 2.5 або такі коефіцієнти можуть бути сформовані шляхом проведення експертизи, наприклад, за популярністю кожної з мов програмування.

За умов можливого створення декількох варіантів рішень постає задача вибору альтернативних рішень. Загалом, такі рішення було б можливо обирати за близькими інтегральними оцінками якості, але, оскільки оцінка якості є зваженою сумою декількох складових, кращими альтернативними рішеннями були б такі, які мали б близькі значення складових, але не інтегральної оцінки. Можливим варіантом вибору такого рішення було б порівняння за сумою відстаней по кожній складовій інтегральної оцінки, тобто шляхом представлення оцінки як вектора у багатовимірному просторі з вибором альтернативи, як вектора, найменш віддаленого від базової оцінки. Мінімальна декартова відстань між такими векторами буде вказувати на близькі за якостями альтернативи. Декартова відстань оцінок для двох варіантів (базового і альтернативного) може бути обчислена як:

$$d_{i,j} = \sqrt{(C_{ei} - C_{ej})^2 + (C_{qi} - C_{qj})^2 + (C_{wi} - C_{wj})^2 + (C_{vi} - C_{vj})^2 + (C_{ri} - C_{rj})^2}, \quad (2.8)$$

де C – значення відповідних складових коефіцієнтів якості окремого варіанту проєкту, наведених у (2.5).

2.8. Висновки до розділу 2

В розділі були розглянуті технології організації і доступу до даних у

проектах корпоративних розподілених інформаційних систем, що визначаються інструментальним забезпеченням, методами та засобами організації, визначення структури і локалізації, збереження, передачі, захисту та обробки даних проектів корпоративних розподілених інформаційних систем.

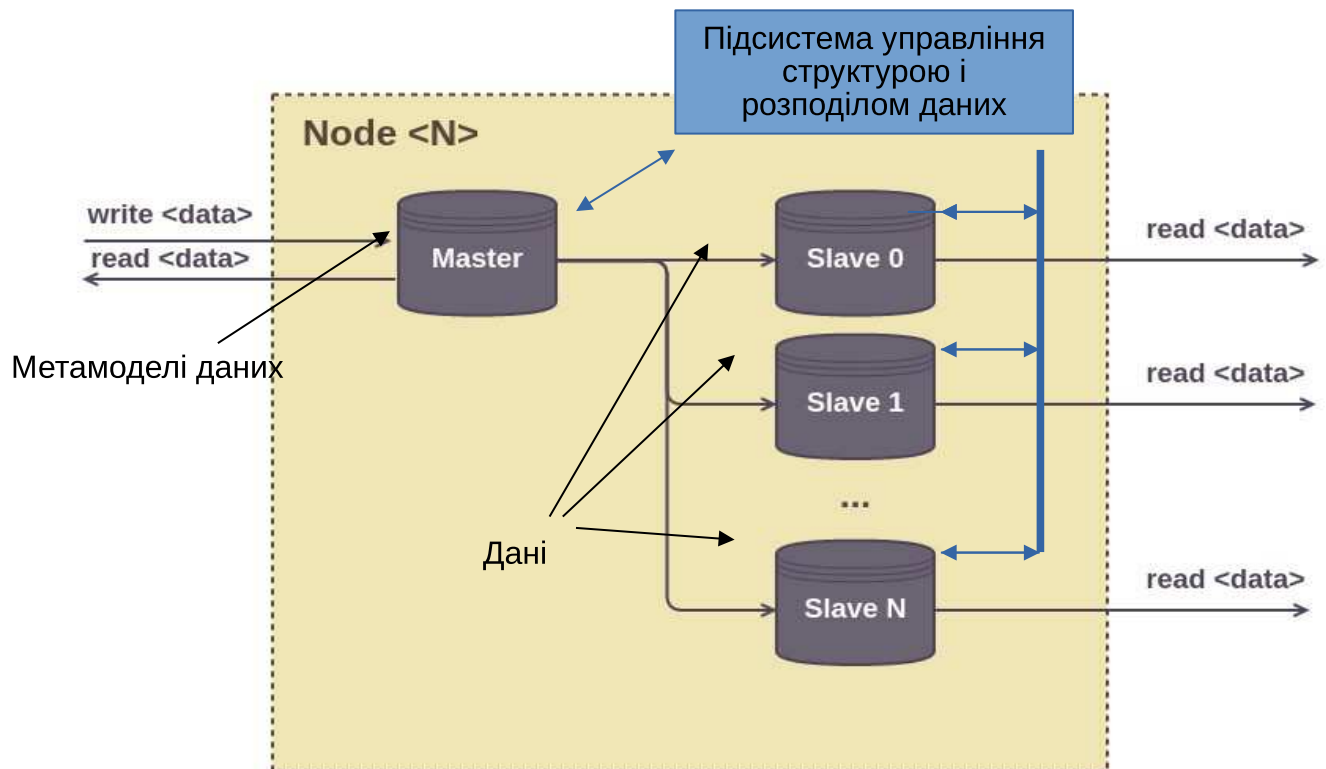


Рисунок 2.7 – Кластеризація даних в розподілений СКБД з підсистемою управління структурою і розподілом даних

Виконано розробку математичної моделі обчислення відповідності технологій проекту 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 проекту.

3. РОЗРОБКА ПРОЕКТНИХ РІШЕНЬ ПО ПІДСИСТЕМІ УПРАВЛІННЯ СТРУКТУРОЮ, ВЕРИФІКАЦІЇ БД І ВПРОВАДЖЕННЯ РЕЗУЛЬТАТІВ

3.1. Розробка адаптивної моделі управління структурою і верифікації сервера баз даних

Враховуючи результати досліджень та з метою перевірки адекватності отриманих рішень розробимо адаптивну модель організації даних для серверів розподілених інформаційних систем. В якості основних механізмів забезпечення адаптивності будемо використовувати наступні:

- нормалізація даних в БД (генерація додаткових індексів);
- денормалізація даних в БД (зміна структури БД);
- збереження проміжних результатів (створення тимчасових таблиць).

Для вибору механізмів адаптивного перетворення будемо використовувати результати профілювання СКБД та оригінальні утиліти збору даних, які базуються на часі виконання запитів.

Алгоритми процедур наведені на рис. 3.1 – 3.4.

З метою проведення оцінки якості інструментального забезпечення проектів розподілених інформаційних систем був реалізований спрощений алгоритм обчислення значення цільової функції, який виконується методом перебору (Рис. 3.1).

Реалізація виконана мовою програмування Python (Додаток В).

В більш складному випадку (при значному числі інструментальних засобів, обмеженнях у часі тощо) можуть бути застосовані методи вирішення оптимізаційних задач (лінійного програмування, сімплекс методи тощо).

На основі розрахунків, проведених на основі розробленого алгоритму (рис. 3.1) визначено, що у навчальних системах доцільно використовувати хмаро-орієнтовані сховища, зокрема Google Drive, Microsoft OneDrive, Dropbox, які дозволять викладачам зберігати файли поза системою MOODLE, з можливістю додавати до курсів посилання на ці файли, використовуючи стандартні сховища або вбудовувати вміст цих файлів в деякі ресурси та види

діяльності курсу.

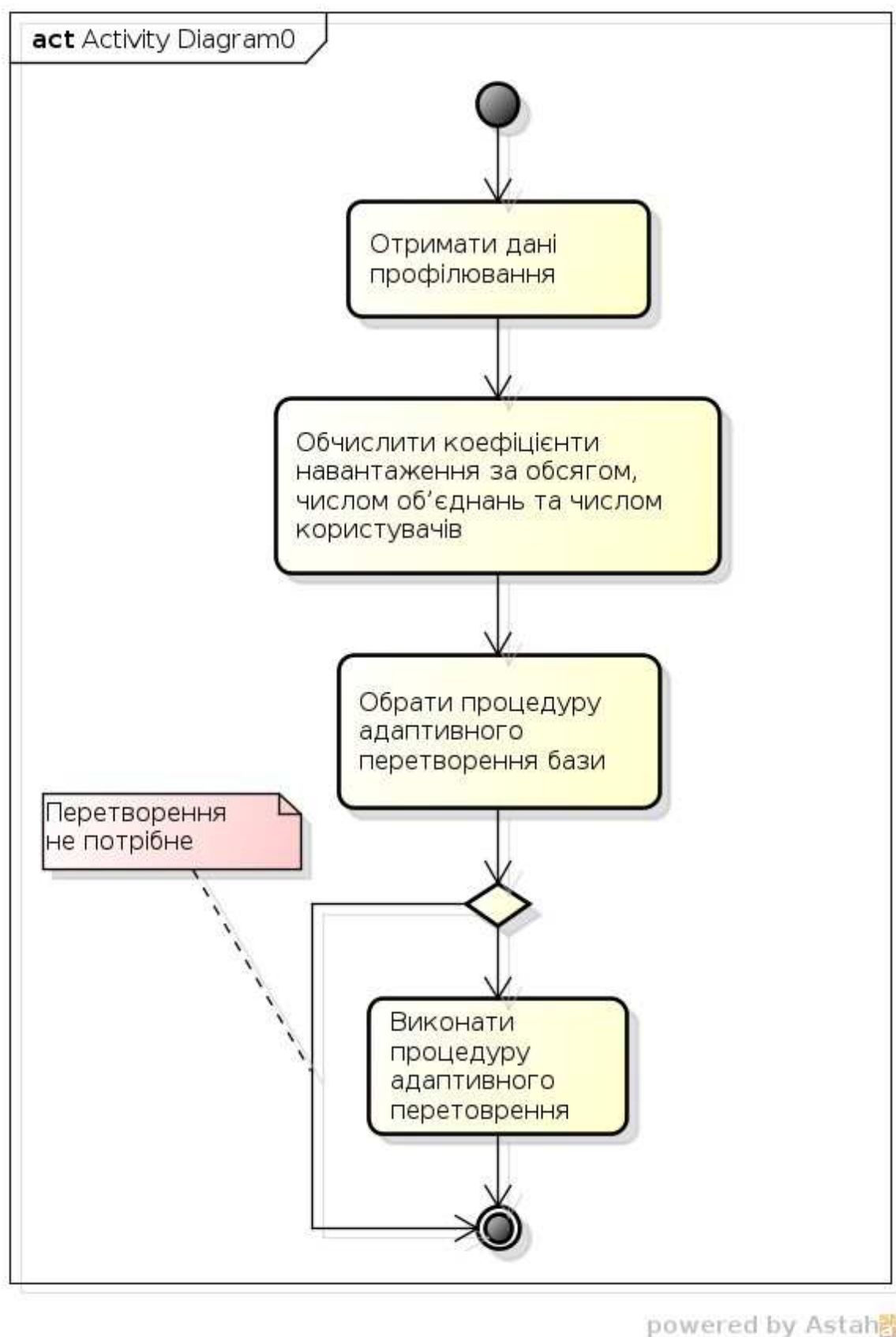


Рисунок 3.1 – Алгоритм процедури забезпечення адаптивності

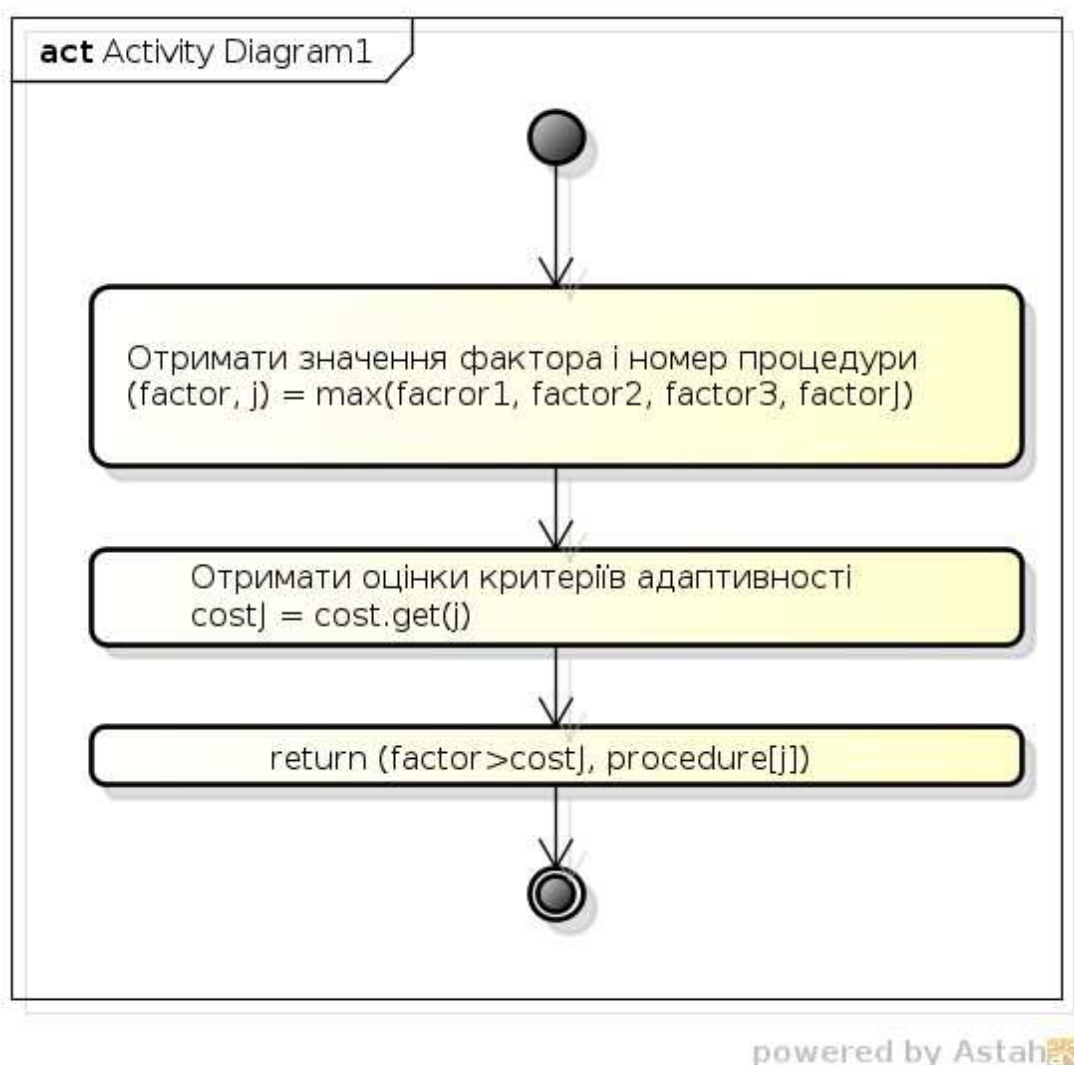


Рисунок 3.2 – Алгоритм процедури забезпечення адаптивності

Результати досліджень були застосовані в окремих модулях проекту автоматизованої інформаційної системи управління міським комунальним господарством.

З метою забезпечення можливостей горизонтальної інтеграції підсистем проектованої автоматизованої системи між собою і вертикальної інтеграції системи в інформаційну систему управління процесами розвитку технічних систем комунального господарства міста, у проекті автоматизованої інформаційної системи повинні бути передбачені адаптивні механізми

керування структурою розподіленого сховища даних, що відповідають складності і особливостям вирішуваних завдань, а також засоби інтеграції окремих підсистем розподіленої інформаційної системи в єдину корпоративну систему.

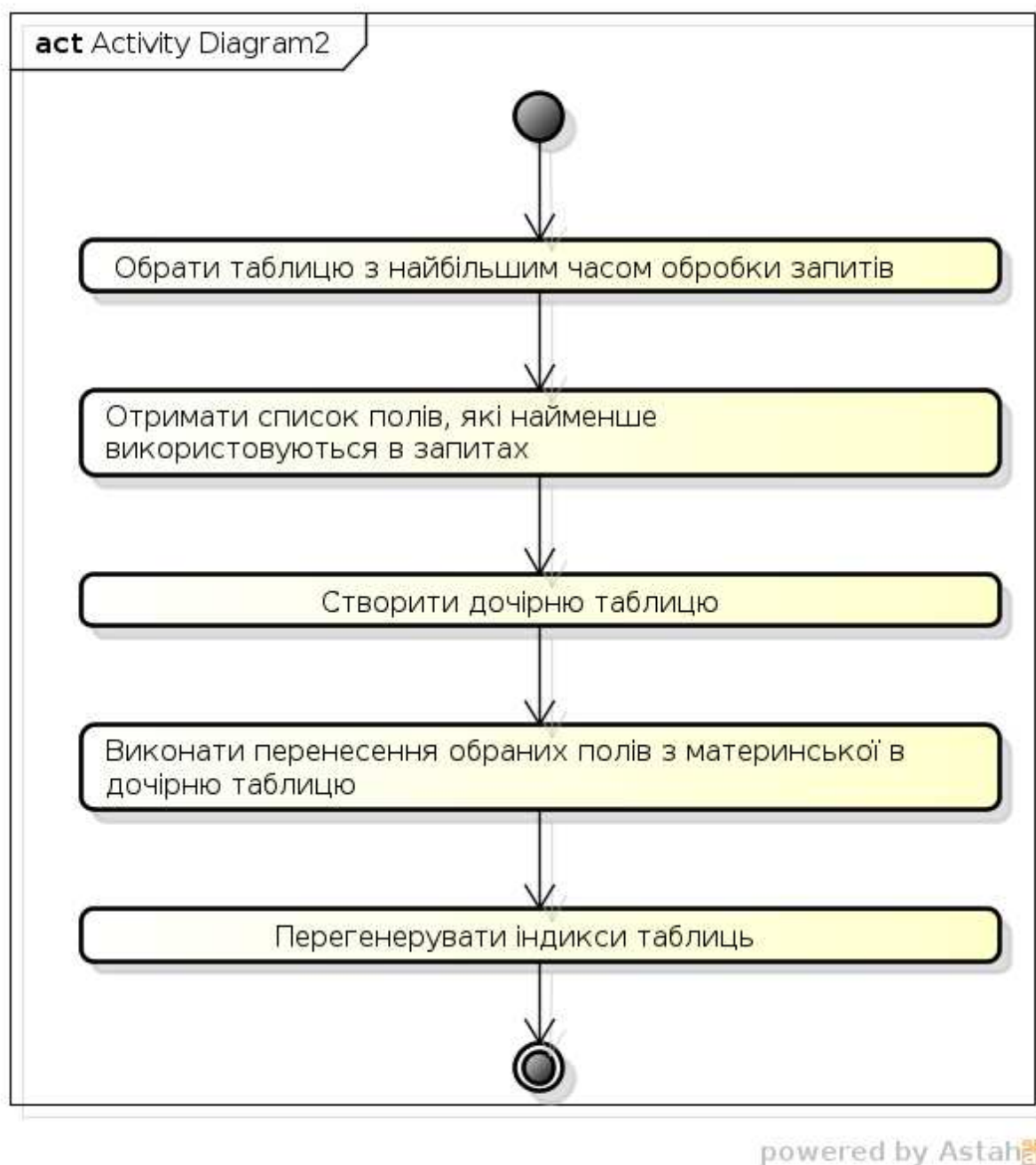


Рисунок 3.3 – Алгоритм адаптивної нормалізації структури індексів за атрибутами БД

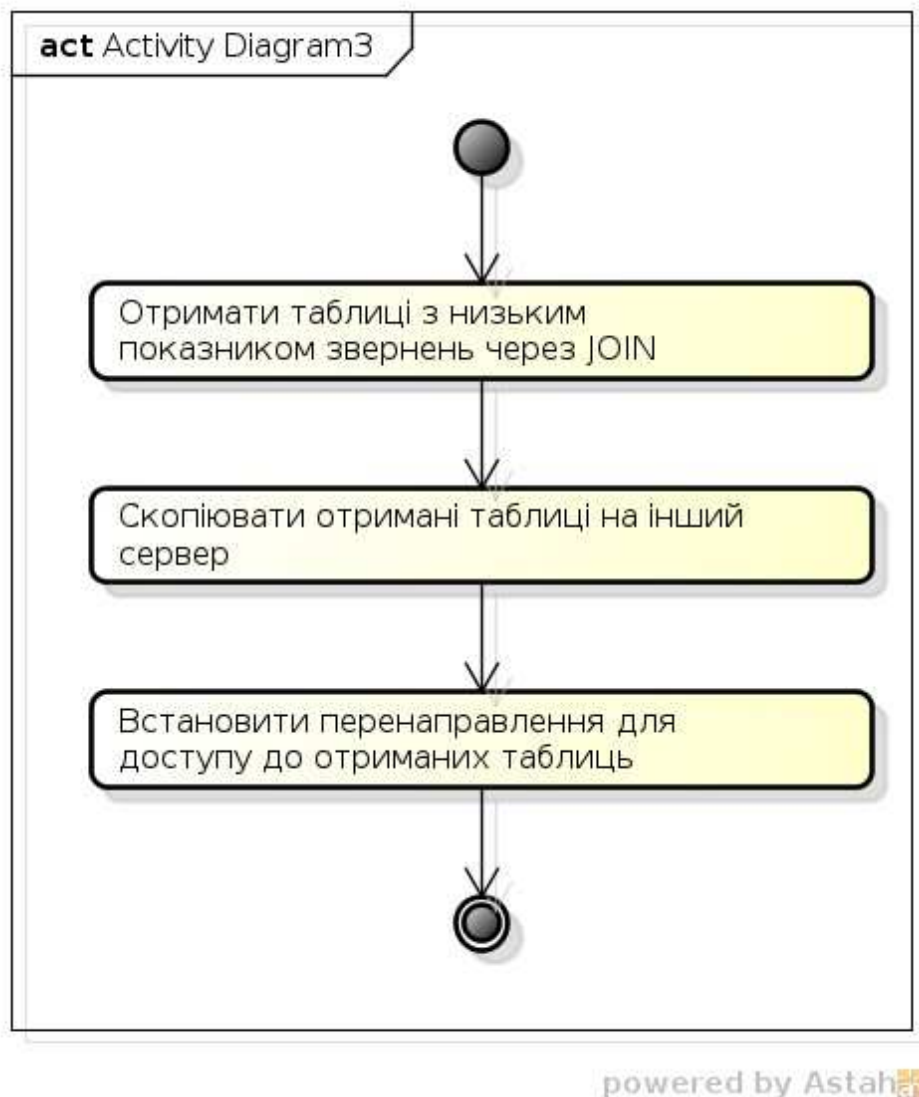


Рисунок 3.4 – Алгоритм адаптивної кластеризації даних розподіленої інформаційної системи за серверами збереження даних

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 керування між окремими 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, 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 та HTTP/HTTPS.

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. Результати тестування для різних “engines” РСКБД MySQL

На основі результатів дослідження розробимо рекомендації з вибору конфігурації системи (Рис. 3.2):

- Визначити можливість використання монітору транзакцій.
- Визначити можливість використовувати систему в конфігурації клієнт-сервер.
- Визначити число одночасно активних користувачів.
- Визначити стратегії індексації даних.
- Визначити чистий розмір бази даних.
- Визначити ємність та швидкість виконання операцій читання-запису для дискової пам'яті необроблених даних, індексів, тимчасових табличних просторів.
- Узгодити передбачувану стратегія резервного копіювання.

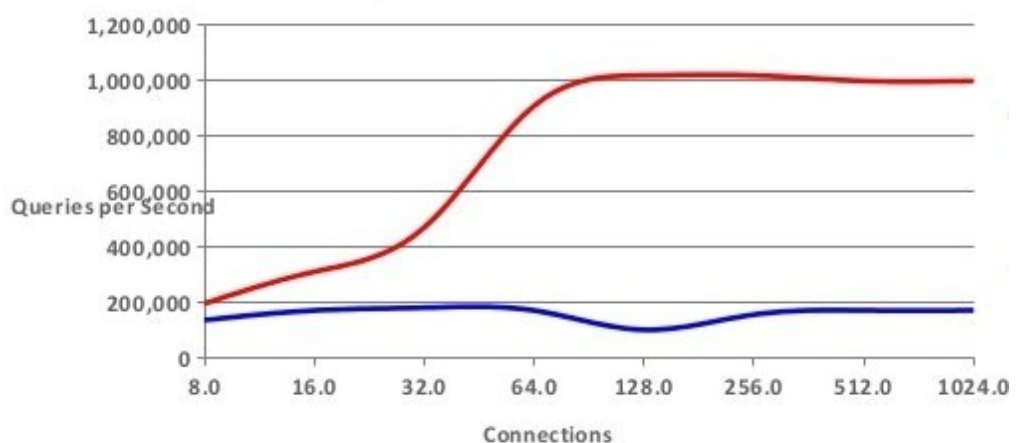


Рисунок 3.5 – Тести навантаження сервера (внизу – з фіксованою структурою, зверху – з вибором структури збереження даних)

3.3. Висновки до розділу 3

В розділі запропоновано спрощений алгоритм пошуку рішення на основі цільової функції оцінки якості, розглянуті можливості її застосування щодо проектів з декількох предметних областей. Також були проведені стрес-тести запропонованого спрощеного алгоритму пошуку раціонального рішення на основі цільової функції оцінки якості та верифікації структури БД. Були розглянуті можливості її застосування щодо проектів з декількох предметних областей.

Розроблені пропозиції щодо використання інструментальних засобів в проектах розподілених інформаційних систем та забезпечення адаптивності структури БД.

Загальні висновки по роботі

В роботі в цілому вирішені задачі, викладені у постановці задачі, що дозволило досягти основної мети дослідження: підвищення ефективності процесів організації, доступу і верифікації даних в сховищах даних корпоративних розподілених автоматизованих інформаційних систем за рахунок обґрунтованого вибору моделей та механізмів верифікації.

Запропоновані рішення можуть бути застосовані при вирішенні задач проектування сховищ даних розподілених інформаційних систем та для інших програмних проектів, що потребують високої інтенсивності обробки великих наборів даних різної природи.

Список використаних джерел

1. Антонюк, А. О. Основи захисту інформації в автоматизованих системах / А. О. Антонюк. – К.: КМ Академія, 2006. – 244 с.
2. Баррет Д. JavaScript. Web-професіоналам. – К: БХВ – Київ, 2001.
3. Вагнер, Р. JavaScript. Энциклопедия пользователя. – Киев: ДиаСофт, 2001.
4. Вайк, А. JavaScript в примерах. – Киев: ДиаСофт, 2000.
5. Вандик, Д. К., Вестгейт, М. Pro Drupal 7 Development: Third Edition / Todd Tomlinson. John K. VanDyk – Apress, 2010 .
6. Гайда А. Ю., Михелев І. Л., Морозова Г. С. До проблеми зникаючого градієнту при навчанні глибоких штучних нейронних мереж. Інновації в суднобудуванні та океанотехніці: XIV Міжнародна науково-технічна конференція: матеріали. — Миколаїв: НУК, 2023.
7. Гайда А. Ю., Морозова Г. С. Механізм адаптивного навчання глибоких штучних нейронних мереж. Вчені записки Таврійського національного університету імені В. І. Вернадського. Серія: Технічні науки. Том 35 (74) No 3 2024 — Херсон: 2024. С. 48-53.
8. ДСТУ 3396.2 – 97. Захист інформації. Технічний захист інформації. Терміни і визначення. – К.: Держстандарт України, 1998.
9. Закон України «Про державну таємницю». – К.: Відомості Верховної Ради України, 1994. – N 16. – Ст. 93.
10. Закон України «Про електронний цифровий підпис». – К.: Відомості Верховної Ради України, 2003. – N 36. – Ст. 276.
11. Закон України «Про захист інформації в інформаційно-телекомунікаційних системах». – К.: Відомості Верховної Ради України, 1994. – N 31. – Ст. 286.
12. Закон України «Про інформацію». – К.: Відомості Верховної Ради України, 1992. – N 48. – Ст. 650

13. Brownlee J. Two-Dimensional (2D) Test Functions for Function Optimization. URL: <https://machinelearningmastery.com/2d-test-functions-for-function-optimization/> — 2001.
14. Chawla N. V., Bowyer K. W., Hall L. O., Kegelmeyer W. P. SMOTE: Synthetic Minority Over-sampling Technique / Journal Of Artificial Intelligence Research, Volume 16, pages 321-357, URL: <https://arxiv.org/abs/1106.1813> — 2002.
15. Dean J. MapReduce: Simplified Data Processing on Large Clusters / Jeffrey Dean and Sanjay Ghemawat // OSDI'04: Sixth Symposium on Operating System Design and Implementation. — San Francisco, CA. — December, 2004.
16. Elreedy D., Atiya A. F., Kamalov F. A theoretical distribution analysis of synthetic minority oversampling technique (SMOTE) for imbalanced learning // Springer, URL: <https://link.springer.com/article/10.1007/s10994-022-06296-4> — 2023.
17. Jain A., Murty M., Flynn P. Data Clustering: A Review. // ACM Computing Surveys. Vol. 31, no. 3. — 1999.
18. Hochreiter S., Bengio Y., Frasconi P., Schmidhuber J. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies. / Kremer, S. C.; Kolen, J. F. A Field Guide to Dynamical Recurrent Neural Networks. IEEE Press. ISBN 0-7803-5369-2. — 2001.
19. Learning from Reasoning Failures via Synthetic Data Generation. Gabriela Ben Melech Stan Estelle Aflalo Avinash Madasu and others. URL:// <https://arxiv.org/pdf/2504.14523> // Cornell University. — 2025.
20. Leskovec J. Mining of Massive Datasets / Jure Leskovec Anand Rajaraman, Jeffrey David Ullman // Stanford Univ. — 2010.
21. McLachlan G. Discriminant Analysis and Statistical Pattern Recognition. John Wiley & Sons Inc. USA. — 2004.

22. Model Comparison and Calibration Assessment: User Guide for Consistent Scoring Functions in Machine Learning and Actuarial Practice. // Tobias Fissler, Christian Lorentzen, Michael Mayer URL: <https://arxiv.org/abs/2202.12780> / USA: Cornell University — 2022.
23. Nabil T., Agoua G., Cauchois P., De Moliner A, Grossin B. A synthetic dataset of French electric load curves with temperature conditioning. URL: <https://arxiv.org/abs/2504.14046> // Cornell University. — 2025.
24. Thousand Voices of Trauma: A Large-Scale Synthetic Dataset for Modeling Prolonged Exposure Therapy Conversations. // Suhas BN, Mattioli D. Abdullah S. and others. URL:// <https://arxiv.org/pdf/2504.13955> // Cornell University. — 2025.
25. Wang, Gang; Shi, Zhijie Jerry; Nixon, Mark; Han, Song (21 October 2019). "SoK". Proceedings of the 1st ACM Conference on Advances in Financial Technologies. pp. 41–61.
26. Yu, Mingchao; Sahraei, Saeid; Nixon, Mark; Han, Song (18 July 2020). "SoK: Sharding on Blockchain". Proceedings of the 1st ACM Conference on Advances in Financial Technologies. pp. 114–134.

Додаток А. Статистика використання різних мов програмування в Україні

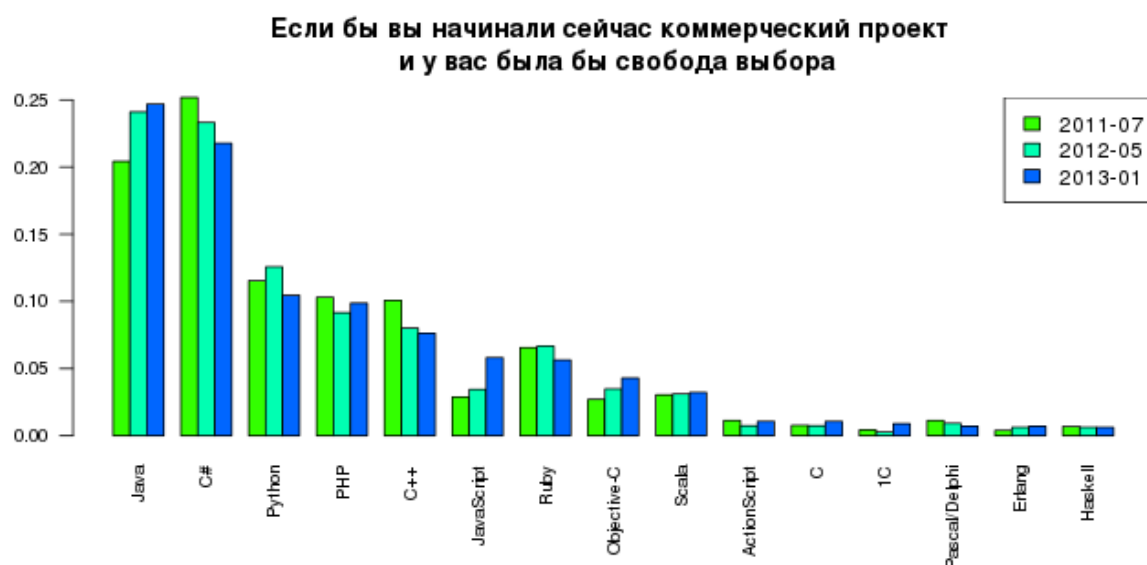


Рисунок А.1 – Статистика використання мов програмування за вподобаннями (дані сервера ІТ-вакансій на початок 2014 року)

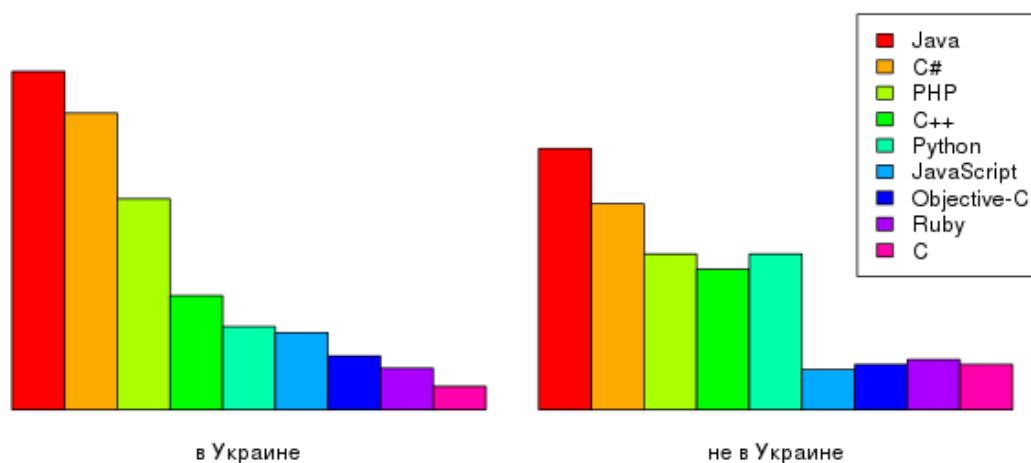


Рисунок А.2 – Статистика використання мов програмування в Україні та за її межами (дані сервера ІТ-вакансій на початок 2014 року)

Відповідей: 8075

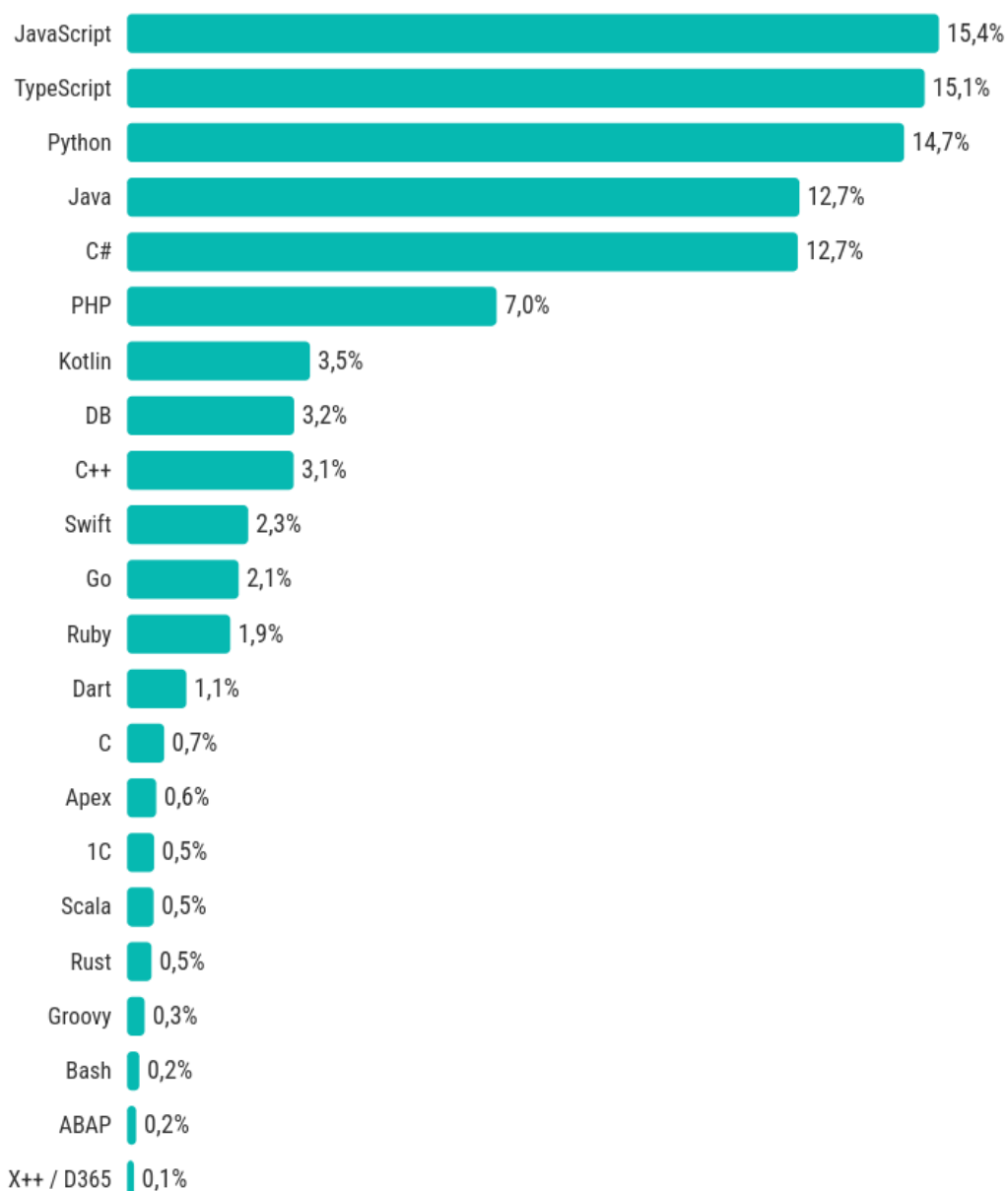


Рисунок А.3 – Статистика використання мов програмування в Україні (дані сервера ІТ-вакансій на кінець 2025 року)

Додаток Б. Приклад реалізації компонентної моделі кластеризації і верифікації структури БД

```

class MetaModelCluster(object):
    """провайдер доступу до таблиці або груп таблиць БД через їх моделі"""
    class MetaModelResult(dict):
        """провайдер верифікації таблиць або груп таблиць БД за моделями"""
        def __init__(self, o):
            dict.__init__(self, o)
            self._data = o.copy()

        def __getitem__(self, fname):
            return self._data[fname]

        def __getattr__(self, fname):
            return self._data[fname]

    def verification(self, model):
        assert Db.exec_select(model._froms, model._cfields._names, None,
            limit=0, strict=False) == True

    def __new__(cls, from_expr, field_expr=None, filter_expr=None):
        """створює об'єкт _SingleModelCluster або _MultiModelCluster (за виразом з
        @from_expr)"""
        if isinstance(from_expr, cssql.Join):
            return _MultiModelCluster.__new__(_MultiModelCluster, from_expr,
            field_expr, filter_expr).
        else:
            return _SingleModelCluster.__new__(_SingleModelCluster, from_expr,
            field_expr, filter_expr)

    def __init__(self, cfroms, cfields, cfilter):
        self._name = self.__class__.__name__
        self._cfroms = cfroms
        self._froms = cfroms._froms
        self._cfields = cfields
        self._filter = cfilter

```

Додаток В. Реалізація алгоритму вибору належності даних до партицій відповідно до критеріїв оцінки вимог до ІС

```
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 масива значеннями від 0 до n-1
    for i in len(n):
        print "Enter c"
        c_i = sys.readline()
        c_i = float(c_i)
        c[i] = c_i
    # заповнення масиву значеннями від 0 до n-1
    for i in len(n):
        print "Enter k"
        k_i = sys.readline()
        k_i = float(c_i)
        k[i] = k_i

    for c_i in c: # перебір 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]

import sys
if __name__ == "main":
    main(sys.argv)
```