Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет *Комп'ютерних наук*
(повна назва)

Кафедра *Системотехніки*
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти *другий (магістерський)*

Адаптація та дослідження методів масштабування баз даних для інформаційних технологій проєктування
(тема)

Виконав:
студент *II* курсу, групи *ІТПм-21-2*
Таран П. А.
(прізвище, ініціали)

Спеціальність *122 Комп'ютерні науки*
(код і повна назва спеціальності)
Тип програми *освітньо-професійна*
(освітньо-професійна або освітньо-наукова)
Освітня програма *Інформаційні технології проєктування*
(повна назва освітньої програми)

Керівник *проф. Безкоровайний В.В.*
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри *СТ* _____ *Гребеннік І. В.*
(підпис) (прізвище, ініціали)

2022 р.

Я, як здобувач ХНУРЕ, розумію і підтримую політику закладу із академічної доброчесності. Я не надавав і не одержував недозволену допомогу під час підготовки кваліфікаційної роботи. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

«19» грудня 2022 р. *Таран П.А.*

Кваліфікаційна робота оформлена у відповідності до вимог діючих стандартів та методичних вказівок.

Матеріали кваліфікаційної роботи не містять відомостей, що заборонені для опублікування у відкритих виданнях.

Попередній захист проведено.

Керівник кваліфікаційної роботи  *В. В. Безкоровайний*

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

Факультет *Комп'ютерних наук*
(повна назва)

Кафедра *Системотехніки*
(повна назва)

Рівень вищої освіти *другий (магістерський)*

Спеціальність *122 Комп'ютерні науки*
(код і повна назва)

Тип програми *освітньо–професійна*
(освітньо–професійна або освітньо–наукова)

Освітня програма *Інформаційні технології проєктування*
(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри _____
(підпис)
«____»_____ 20 ___ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові *Тарану Павлу Андрійовичу*
(прізвище, ім'я, по батькові)

1. Тема роботи *«Адаптація та дослідження методів масштабування баз даних для інформаційних технологій проєктування»*
затверджена наказом університету від *«21» листопада 2022 р. №1504 Ст*
2. Термін подання студентом роботи до екзаменаційної комісії *«21» грудня 2022 р.*
3. Вихідні дані до роботи *Об'єкт дослідження – бази даних інформаційних технологій проєктування. Предмет дослідження – процедури масштабування та фрагментації баз даних для інформаційних технологій проєктування. Предмет розробки – засіб масштабування та фрагментації баз даних для інформаційних технологій проєктування. Функція – оптимізації варіантів масштабування та фрагментації баз даних. Розмірність задачі: кількіть користувачів – до 1000; Кількість локальних баз – до 4. Технічне забезпечення: IBM–сумісний персональний комп'ютер. Перелік використовуваних програмних засобів: ОС Windows; MS Office, Microsoft Visual Studio, Google Chrome.*
4. Перелік питань, що потрібно опрацювати в роботі *Вступ. Аналіз сучасного стану проблеми масштабування баз даних для інформаційних систем проєктування. Технології проєктування як об'єкт автоматизації. Бази даних в інформаційних системах. Питання масштабування баз даних для інформаційних систем та технологій проєктування. Розподілені бази даних. Огляд методів масштабування баз даних Постановка задачі. Розробка математичного забезпечення задачі. Огляд методів проєктування розподілених БД. Фрагментація розподілених БД. Математичний опис методів горизонтальної фрагментації БД. Математичний опис методів вертикального фрагментування. Розробка програмного забезпечення задачі та експерименти Вибір середовища розробки та мови програмування. Розробка структури програми.. Планування та проведення експериментів. Аналіз результатів. Висновки. Перелік джерел посилання.*

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п. 5 включається до завдання за рішенням випускової кафедри). *Кресленики, схеми, плакати та/або комп'ютерні ілюстрації (слайди) на аркушах формату А4, що включаються до тексту пояснювальної записки або складу додатків: схема декомпозиції задачі структурної оптимізації БД як ТРО, схеми вертикального та горизонтального масштабування БД, реплікації, горизонтального та вертикального секціонування, алгоритмів горизонтальної фрагментації БД, процедури знаходження точки розщеплення, алгоритму розщеплення БД, графіки чи діаграми з результатами експериментів.*

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	*Отримання, аналіз завдання, уточнення плану*		
	роботи	*01.09.22*	*Виконано*
2	*Аналіз сучасного стану проблеми масштабування*		
	баз даних для інформаційних технологій	*08.09.22*	*Виконано*
	проєктування		
3	*Огляд математичних моделей і методів*		
	масштабування баз даних	*22.09.22*	*Виконано*
4	*Розробка математичного забезпечення задачі*	*13.10.22*	*Виконано*
5	*Розробка програмного забезпечення задачі*	*27.10.22*	*Виконано*
6	*Проведення експериментальних досліджень*	*10.11.22*	*Виконано*
7	*Підготовка публікацій за результатами*		
	дослідження	*24.11.22*	*Виконано*
8	*Оформлення пояснювальної записки*	*01.12.22*	*Виконано*
9	*Подання закінченої роботи науковому керівникові*	*05.11.22*	*Виконано*
10	*Усунення зауважень наукового керівника*	*10.12.22*	*Виконано*
11	*Подання роботи на рецензування*	*12.12.22*	*Виконано*
12	*Підготовка презентації*	*16.12.22*	*Виконано*
13	*Попередній захист*	*19.12.22*	*Виконано*
14	*Подання роботи до екзаменаційної комісії*	*21.12.22*	*Виконано*

Дата видачі завдання _____

Студент _____ *Таран П. А.*
(підпис)

Керівник роботи _____ *Безкоровайний В. В.*
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи: 88 с., 16 рис., 2 дод., 31 джерело.

АЛГОРИТМ, БАЗА ДАНИХ, ІНФОРМАЦІЙНА СИСТЕМА, ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ ПРОЄКТУВАННЯ, МАСШТАБУВАННЯ БАЗИ, ОПТИМІЗАЦІЯ, ФРАГМЕНТАЦІЯ.

Об’єктом дослідження є бази даних інформаційних технологій проєктування.

Предметом дослідження є процедури масштабування та фрагментації баз даних для інформаційних технологій проєктування.

Метою роботи є підвищення ефективності інформаційних технологій проєктування за рахунок оптимізації варіантів масштабування та фрагментації баз даних, що використовуються в них.

Методи дослідження – системний підхід, методи структурного аналізу і моделювання, сучасні інформаційні технології.

У ході виконання роботи розглянуто та проаналізовано поточний стан технологій проектвання як об’єктів автоматизації, та проблеми, які виникають при роботі з базами даних у даній предметній області. Також проведено аналіз та порівняння існуючих методів та алгоритмів масштабування і секціонування розподілених баз даних.

За результатами аналізу обрано підхід для вирішення поставленої задачі з оптимізації розподіленої БД шляхом масштабування та фрагментування.

Результати роботи представлені у вигляді застосунку з вирішення задачі масштабування бази даних для інформаційних технологій проєктування.

ABSTRACT

Master’s Thesis: 88 pages, 16 figures, 2 appendice, 31 sources.

ALGORITHM, DATABASE, DATABASE SCALING, FRAGMENTATION, INFORMATION SYSTEM, INFORMATION TECHNOLOGIES OF DESIGN, OPTIMIZATION.

The object of research is the database of information technologies of design.

The subject of the study are methods and procedures for scaling and fragmentation of databases for information technologies of design.

The purpose of the work is to increase the efficiency of information technologies of design with optimization of options for scaling and fragmentation of the databases they use.

Research methods – systematic approach, methods of structural analysis and modeling, modern information technologies.

In the course of the work, the current state of design technologies as objects of automation and the problems that arise when working with databases in this subject area were considered and analyzed. An analysis and comparison of existing methods and algorithms for scaling and partitioning of distributed databases was conducted.

Based on the results of the analysis, was chosen the approach to solve the task of optimizing a distributed database by scaling and fragmentation.

The results of the work are presented in the form of an app to solve the problem of database scaling for information technologies of design.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

БД – база даних.

ТРО – територіально розподілений об’єкт.

ІР – інформаційний ресурс.

ІС – інформаційна система.

ІТП – інформаційні технології проєктування.

ЛБ – локальна база.

ПО – предметна область.

РБД – розподілена база даних.

СКБД – система керування базами даних.

ВСТУП

В умовах швидкого зростання складності об’єктів проєктування, швидкого розвитку технологій інформаційних систем швидкими темпами зростають об’єми проєктних даних, що зберігаються, кількість звернень до них, та складність процедур їхньої обробки. Це підвищує вимоги до сховищ даних в інформаційних технологіях проєктування та до систем керування ними. Вирішити проблему зростаючого обсягу даних та навантаження на системи керування ними допомагає масштабування баз даних (БД).

В інформаційних технологіях проєктування для забезпечення роботи з базами даних використовуються системи керування базами даних (СКБД). При роботі з базами даних може виникнути ситуація, коли поточної потужності системи недостатньо для її нормальної роботи, це може бути пов'язано з тим, що кількість даних та/або користувачів збільшується з часом, коли поточна технічна інфраструктура тривіальна застаріває, або база даних просто не вписується в початкову конфігурацію інформаційної системи.

Таким чином, масштабування бази даних є одним із інструментів підвищення загальної продуктивності системи при роботі з даними в інформаційних системах проєктування і може допомогти вирішити проблему зростаючого обсягу даних та навантаження на бази даних.

Об’єктом дослідження є бази даних інформаційних технологій проєктування.

Предметом дослідження є процедури масштабування та фрагментації баз даних для інформаційних технологій проєктування.

Метою роботи є підвищення ефективності інформаційних технологій проєктування за рахунок оптимізації варіантів масштабування та фрагментації баз даних, що використовуються в них.

Методи дослідження – системний підхід, методи структурного аналізу і моделювання, сучасні інформаційні технології.

Наукова новизна отриманих результатів полягає в розробці математичних методів оптимізації структур РБД у інформаційних технологіях проєктування шляхом їх масштабування та секціонування.

Практичне значення атестаційної роботи зумовлено створенням інструментальних засобів з автоматизації застосування розроблених математичних методів відповідно до заданих умов та виведення результату цих розрахунків для подальшого їх використання задля оптимізації структури реляційних баз даних у інформаційних технологіях проєктування.

У ході виконання атестаційної роботи також було подано матеріали тез до наукових конференцій, які доступні за посиланнями [1] та [2].

1 АНАЛІЗ СУЧАСНОГО СТАНУ ПРОБЛЕМИ МАСШТАБУВАННЯ БАЗ ДАНИХ ДЛЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ПРОЄКТУВАННЯ

1.1 Технології проєктування як об’єкт інформаційного забезпечення

1.1.1 Поняття проєктування

Загалом проектуванням можна назвати комплекс робіт з пошуку, досліджень, розрахунків та обчислень з метою отримання якомога детальнішого опису для створення нового, або модернізації існуючого об’єкту, виробу, застосунку, тощо, згідно заданим вимогам [3].

У технічному плані поняття проєктування можна звузити до процесу створення прототипу. Або прообразу майбутнього об’єкту, стану та способів його виготовлення з розробкою технічної, конструкторської або будь-якої іншої документації призначеною закріпити усі вимоги та інструкції, необхідні для забезпечення створення цього нового об’єкту [3].

1.1.2 Проблеми проєктування великомасштабних об’єктів

З розвитком сучасних технологій все дедалі збільшуються масштаби створюваних об’єктів, а, одже, зростають вимоги до проєктування, та все більше проектів та систем використовують у своїй структурі топологічне, або просторове розподілення. Такі об’єкти та системи називаються територіально розподіленими (ТРО), або, якщо казати простіше, великомасштабними [4]. У процесі проєктування ТРО окрім звичайних проблем структурного синтезу також виникає потреба вирішувати задачі топологічної оптимізації. Топологія підсистем та елементів визначає топологію ліній зв’язку між цими підсистемами що забезпечує функціонування системи як єдиного цілого, та

реалізую обмін необхідною інформацією та ресурсами між цими елементами та підсистемами [4].

Огляд сучасного стану задачі оптимізації великомасштабних систем та об’єктів показує, що існуючі технології розроблені з припущенням умовного самостійного вирішення задач топологічної, структурної, технологічної та параметричної оптимізації, що не дозволяє у повній мірі використати можливості формальних методів та комп’ютерних потужностей для отримання найефективніших рішень у цій області [4].

Для забезпечення ефективності та безперервності прийняття рішень на всіх етапах життєвих циклів ТРО необхідна єдина методологія їх системної та топологічної оптимізації [4].

Великомасштабні об’єкти зазвичай складаються з великої кількості елементів зі складною схемою взаємозв’язків між ними, тому створення єдиного опису в процесі їх структурної оптимізації є складною задачею. Моделі та методи вирішення цих проблем ще не побудовані [4].

Схема декомпозиції задачі структурної оптимізації великомасштабних об’єктів надана на рисунку 1.1 [4].

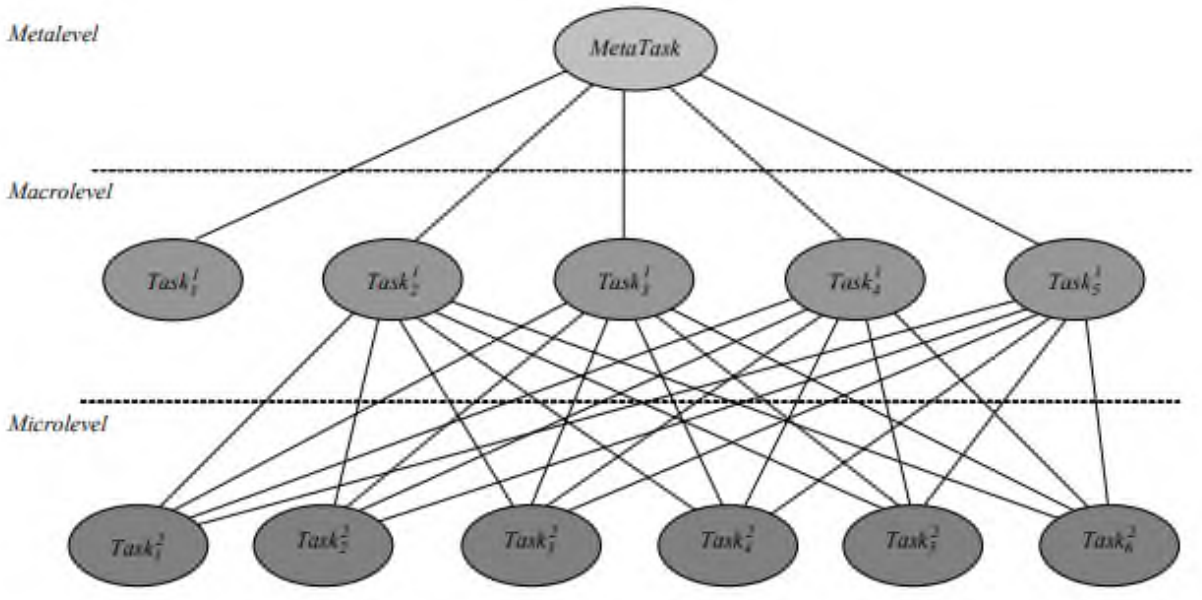

Рисунок 1.1 – Схема декомпозиції задачі оптимізації РБД як ТРО

З даної схеми видно, що загальна задача (мета-задача) з структурної оптимізації великомасштабних об’єктів складається з множин задач, які розміщені на різних ієрархічних рівнях, та мають складні зв’язки вхідної дати та результатів між ними.

Дослідження даної кваліфікаційної роботи направлені на оптимізацію великомасштабних об’єктів та систем при роботі з базами даних (БД) та системами керування базами даних (СКБД).

1.2 Бази даних в інформаційних системах

Для початку розглянемо саме поняття бази даних та системи керування базами даних.

База даних (від англ. Database) – сукупність даних, організованих відповідно до концепції, яка описує характеристику цих даних і взаємозв'язки між їх елементами; ця сукупність підтримує щонайменше одну з областей застосування. В загальному випадку база даних містить схеми, таблиці, подання, збережені процедури та інші об'єкти. Дані у базі організовують відповідно до моделі організації даних. Таким чином, сучасна база даних, крім саме даних, містить їх опис та може містити засоби для їх обробки [5].

У сучасних інформаційних системах для забезпечення роботи з базами даних використовують системи керування базами даних (СКБД). Система керування базами даних – це система, заснована на програмних та технічних засобах, яка забезпечує визначення, створення, маніпулювання, контроль, керування та використання баз даних [5].

В залежності від типу інформаційної системи, можна виділити 3 варіанти структури розміщення БД та СКБД:

– локальні ІС – коли БД та СКБД знаходяться на одному комп’ютері;

– файл–серверні ІС – БД знаходиться на файловому сервері мережі, а СКБД – на комп’ютері користувача;

– клієнт–серверні ІС – БД та головна СКБД знаходяться на сервері, а

локальна СКБД користувача посилає запит серверу та відображає результат [6].

Ці варіанти мають свою плюсі та мінуси, наприклад для локальних ІС перевагою буде велика автономність, але з такою БД складно працювати, адже одна робоча станція напряму працює з одним екземпляром бази локально, та її буде складно оновлювати при великій кількості користувачів, а зміни від різних користувачів майже неможливо синхронізувати [6].

У файл–серверній ІС вже декілька людей працюють з однією базою, але недоліками є слабкий захист, проблеми прі одночасній зміні даних, велике навантаження на мережу та вимоги до робочих станцій користувачів.

Клієнт–серверні ІС вирішують масу недоліків своїх попередників, так як основну роботу виконує окремий сервер, цей сервер легше модернізувати, мережа розвантажена, адже пересилає тільки потрібні частини даних, захист та права доступу налаштовані з боку сервера, що збільшує захист системи, та присутнє розділення доступу, тобто запити виконуються за чергою. Але всі ці переваги призводять до складнощів у налаштуванні такої системи, та до великих витрат на її підтримку [6].

1.3 Питання масштабування баз даних для інформаційних систем та технологій проєктування

З розвитком та розповсюдженням інформаційних систем постійно збільшуються об’єми, кількість звернень та складність обробки даних. При постійному зростанні усіх цих параметрів також збільшуються і вимоги до сховищ даних, та до систем керування сховищами даних. Вирішити питання постійно зростаючого обсягу даних та навантаження на системи керування даними допомагає масштабування баз даних.

При роботі з базами даних, може виникнути ситуація, коли поточних потужностей системи не вистачає для нормального її функціонування, це може бути пов’язано зі збільшенням кількості даних та/або клієнтів у із часом, коли

поточна технічна інфраструктура банально застаріває, або БД просто «не вміщується» у початкову конфігурацію інформаційної системи, або ж із короткочасними піковими навантаженнями на систему, які можуть бути пов’язані із сезонними явищами, такими як акції, чи періоди свят, коли потрібно обробляти запити від набагато більшої кількості користувачів, ніж система звикла у звичайному режимі. Таким чином масштабування баз даних є одним зі інструментів збільшення загальної продуктивність системи при роботі з даними в інформаційних системах, і може вирішувати різноманітні питання підвищення продуктивності інформаційної системи при роботі з даними в залежності від поточних умов та потреб. Сам процес масштабування БД так чи інакше пов’язаний з інфраструктурними змінами. Розділяють вертикальне (Scaling–up) та горизонтальне (Scaling–out) масштабування баз даних [7]. В залежності від вимог до інформаційної системи та для вирішення саме поточних задач в різні моменти часу може знадобитися як вертикальне, так і горизонтальне масштабування бази даних. Основну ідею та відмінності між вертикальним та горизонтальним масштабуванням можна побачити на рисунку 1.2 [8].

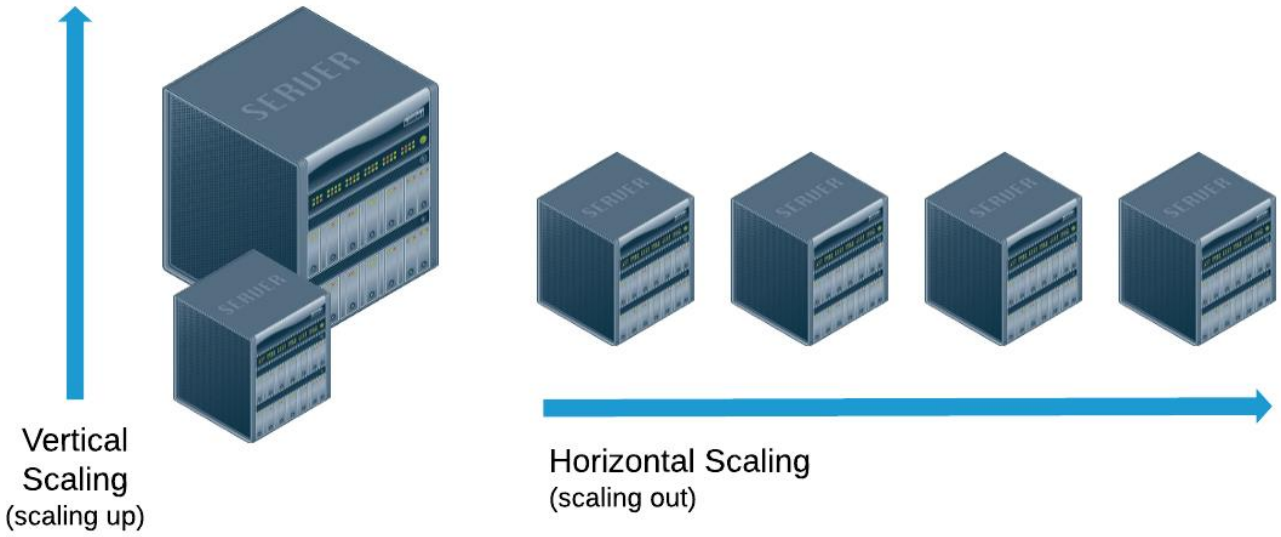

Рисунок 1.2 – Схеми вертикального та горизонтального масштабування БД

Масштабування БД також пов’язано з питанням проєктування інформаційних систем, адже для того щоб застосувати якийсь з методів масштабування, фізична інфраструктура та програмне забезпечення інформаційної системи повинно бути готовим до цього. Щоб досягти цього, зазвичай елементи інформаційної системи потрібно створювати максимально відокремленими та паралельними, що полегшити переміщення частин системи та додавання ресурсів під час масштабування не порушуючи ніяких зв’язків та відносин між модулями [8]. Такий підхід додає гнучкості системі, та спрощує процеси масштабування у майбутньому.

1.4 Розподілені бази даних

Говорячи про масштабування в рамках інформаційних систем, слід розглянути питання розподілених БД.

Розподілена база даних може бути визначена як набір із декількох логічно взаємопов'язаних баз даних, що розташовані у вузлах розподіленої системи. Розподіленою системою управління базами даних (розподіленою СКБД) називається програмна система, яка допускає управління розподіленою базою даних і робить цей розподіл прозорим для користувачів. Іноді, говорячи «розподілена система баз даних» (розподілена СКБД), може матися на увазі як розподілена БД, так та розподілена СКБД. Виділяються 2 важливі характеристики: логічна взаємопов'язаність даних та їх знаходження у розподіленій системі [9].

Основними ж положеннями, або, очікуваннями від розподілених БД є:

– прозоре управління розподіленими та реплікованими даними;

– надійний доступ до даних засобом розподілених транзакцій;

– покращена продуктивність;

– просте розширення системи [9].

В розподілених БД дані можуть бути представлені у сегментованому вигляді. При проектуванні розподіленої бази даних труднощі становить

локалізація запитів та транзакцій, що може призводити до сповільнення системи через виникнення розподілених запитів/транзакцій [9]. Тому створення розподіленої бази даних має сенс насамперед при проектуванні нової БД з нуля, або при глобальному реінжинірингу вже існуючої БД.

Фрагментація даних, може бути горизонтальною (шардінг) та вертикальною, але у даному випадку не варто плутати фрагментацію розподіленої БД з масштабуванням баз даних. Основою горизонтальної фрагментації служить оператор вибірки, предикат якого визначає спосіб фрагментації, тоді як вертикальна фрагментація здійснюється за допомогою оператора проєктування [9].

Загальними властивостями архітектури розподілених баз даних є неоднорідність, розподіленість та автономність, та Крістофер Дейт, один з найбільших діячів в галузі баз даних, сформував цілих 12 властивостей розподілених БД, а саме [10]:

– локальна автономія – управління даними на кожному з вузлів розподіленої системи виконується локально;

– незалежність вузлів – всі вузли рівноправні і незалежні, а розташовані на них БД є рівноправними постачальниками даних в загальний простір даних;

– безперервні операції – можливість безперервного доступу до даних в рамках розподіленої БД незалежно від їх розташування і незалежно від операцій, що виконуються на локальних вузлах;

– прозорість розташування – користувач, що звертається до БД, нічого не повинен знати про реальне, фізичне розміщення даних у вузлах інформаційної системи;

– прозора фрагментація – можливість розподіленого (тобто на різних вузлах) розміщення даних, логічно поєднаних в єдине ціле. Існує фрагментація двох типів: горизонтальна і вертикальна;

– прозоре тиражування – тиражування даних – це асинхронний процес перенесення змін об'єктів вихідної бази даних в бази, розташовані на інших вузлах розподіленої системи;

– обробка розподілених запитів – можливість виконання операцій вибірки даних з розподіленої БД, за допомогою запитів, сформульованих на мові SQL;

– обробка розподілених транзакцій – можливість виконання операцій оновлення розподіленої бази даних, які не порушують цілісність і узгодженість даних;

– незалежність від устаткування – як вузли розподіленої системи можуть виступати ПК будь–яких моделей і виробників;

– незалежність від операційних систем – різноманіття операційних систем, керуючих вузлами розподіленої системи;

– прозорість мережі – спектр підтримуваних конкретною СКБД мережевих протоколів не має бути обмеженням системи, заснованої на розподіленій БД;

– незалежність від баз даних – в розподіленій системі можуть працювати СКБД різних виробників, і можливі операції пошуку і оновлення в БД різних моделей і форматів.

1.5 Огляд методів масштабування баз даних

Вертикальним масштабуванням є нарощуванням чи зниженням обчислювальної потужності чи ресурсів баз даних [11]. Переважно мається на увазі додавання нових дисків до серверу бази даних, заміна на більш потужні чи додавання нових (за можливості) процесорів, або навіть повне перенесення бази даних на інший сервер, з додатковими обчислювальними можливостями та більшим розміром сховища.

Вертикальне масштабування має на меті вирішити загальні проблеми низької продуктивності інформаційної системи при роботі з даними, або для швидкої реакції на термінові проблеми із продуктивністю при пікових навантаженнях, чи відмовою частини обладнання.

Горизонтальне масштабування досягається шляхом додавання

додаткових баз даних або розділення великої бази даних на вузли меншого розміру з використанням секціонування даних методом сегментування. Горизонтальне масштабування застосовується коли поточних ресурсів вже недостатньо, та вертикальне масштабування неможливе, або немає сенсу з фінансової точки зору [11].

У випадку горизонтального масштабування, найпростішою буде техніка реплікації БД, коли створюються копії бази на інших серверах, і навантаження розподіляється між усіма доступними серверами. Простим варіантом є реплікація типу “Master–slave” (рис. 1.3) [12].

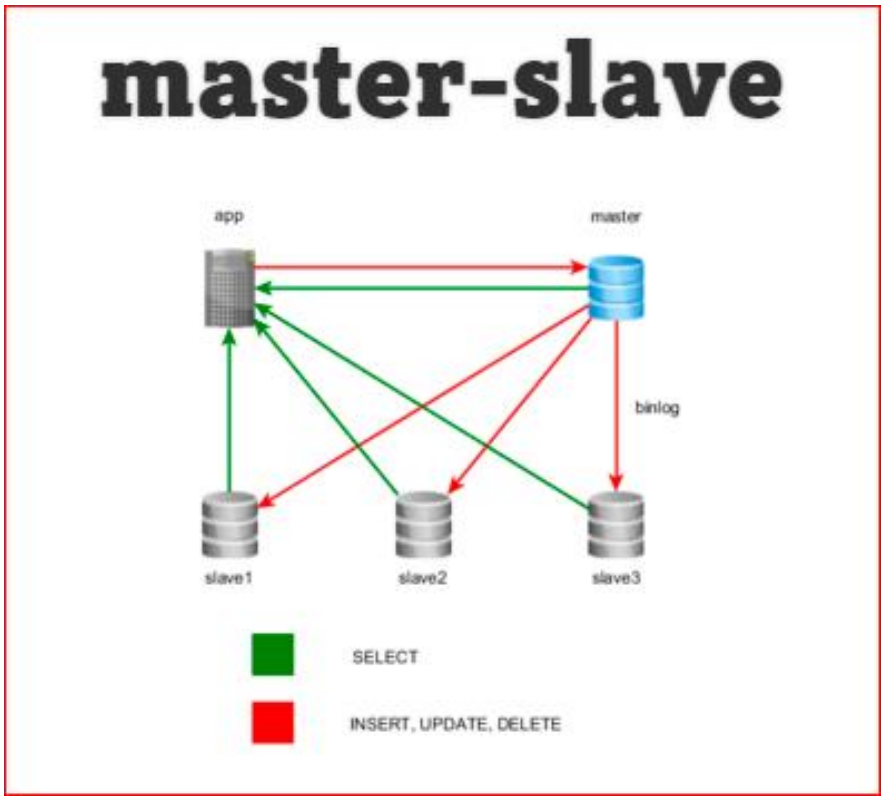

Рисунок 1.3 – Схема реплікації типу «Master–Slave»

З усього вищезазначеного, можна сказати, що масштабування баз даних є гнучким інструментом та вже невід’ємною частиною інформаційних технологій проєктування та можуть використовуватися для вирішення цілого спектру проблем, які виникають при найрізноманітніших сценаріях експлуатації високонавантажених інформаційних систем та систем з великими

обсягами даних, які постійно потребують розширення.

1.6 Постановка мети та задач дослідження

Для досягнення поставленої мети необхідно вирішити такі завдання:

– сформулювати постановку задачі масштабування баз даних для інформаційних технологій проєктування;

– обрати та проаналізувати методи масштабування та секціонування розподілених баз даних;

– розробити математичне забезпечення задачі;

– обрати мову програмування та середовище розробки програми;

– розробити програмне забезпечення задачі;

– провести перевірку працездатності програми;

– провести модельні експерименти та оцінити точність отриманих результатів.

2 ПОСТАНОВКА ЗАДАЧІ

2.1 Задача оптимізації структури розподіленої бази даних

Загальна задача оптимізації структури розподіленої бази даних (РБД) розглядається такій постановці [13]. Задані: множина користувачів РБД; множина інформаційних ресурсів (ІР); обсяги ІР; інтенсивності надходження запитів від користувачів до кожного з інформаційних ресурсів; обсяги запитів до інформаційних ресурсів; наведені витрати на зберігання ІР у вузлах комп'ютерної мережі; витрати на передачу одиниці інформації; обсяги інформації, що передається при оновленні інформаційних ресурсів; множина допустимих структур РБД.

Необхідно визначити найкращий варіант структури РБД з множини допустимих $x^o \in X$, кількість локальних баз (ЛБ), розподіл інформаційних ресурсів по ЛБ, розміщення ЛБ по вузлах мережі, обсяги пристроїв для зберігання ЛБ.

Критерії оптимізації: наведені витрати на x–реалізацію РБД $c(x) \to \min\limits_{x \in X}$; час доступу до ІР $t(x) \to \min\limits_{x \in X}$; загальний обсяг інформації, що передається $v(x) \to \min\limits_{x \in X}$. При цьому необхідним є забезпечення повноти бази за рахунок розподілу всіх ІР (з можливим дублюванням) по локальних базах та виконання обмеження на час доступу до інформаційних ресурсів бази даних $t(x) \le t^*$ (де t^* – максимально допустиме значення часу доступу до ІР).

Комбінаторний характер задач оптимізації РБД передбачає використання автоматичної кількісної оцінки варіантів із множини допустимих шляхом оцінки їхньої корисності $P(x)$.

Формально задача вибору найкращого варіанта структури РБД може бути зведена до задачі оптимізації виду [13]:

$$x^o = arg \max_{x \in X} P(x) = arg \max_{x \in X} \sum_{i=1}^{3} \lambda_i \xi_i(x). \quad (2.1)$$

$$k_1(x) = c(x) \to \min_{x \in X}, \; k_2(x) = t(x) \to \min_{x \in X}, \; k_3(x) = v(x) \to \min_{x \in X}. \quad (2.2)$$

де λ_i, $\xi_i(x)$ – вагові коефіцієнти та функція корисності часткових критеріїв $k_i(x)$, $i = \overline{1,3}$.

Математичні моделі та методи розв'язання багатокритеріальних задач оптимізації РБД, включаючи ситуації неповної визначеності даних детально розглянуті в роботах [13–27].

За результатами їх аналізу для розв'язання задачі багатокритеріальної оптимізації РБД шляхом горизонтального масштабування (2.1) – (2.2) пропонуються використати метод послідовного застосування критеріїв.

2.2 Постановка задачі масштабування баз даних для інформаційних технологій проєктування

У ході дослідження методів масштабування баз даних для інформаційних технологій проєктування, потрібно розробити застосунок, для визначення оптимального способу масштабування даних у рамках інформаційної системи. Для цього потрібно поставити прикладну задачу з вхідними умовами наближеними до реальних та представити її вирішення.

Система – застосунок з розрахунку масштабування баз даних для інформаційних технологій проєктування.

2.2.1 Призначення розробки

Створення даного застосунку є необхідним для програмної реалізації вирішення задачі масштабування розподіленої бази даних для інформаційних технологій проєктування.

Користувачами системи є проєктувальники та адміністратори баз даних.

2.2.2 Мета створення

Метою створення розробки є:

– аналіз вхідних даних та умов використання наданої бази даних;

– визначення методу масштабування бази даних за наданими вхідними параметрами;

– автоматизація процесу розрахунку оптимального шляху масштабування баз даних для інформаційних технологій проєктування.

2.2.3 Вхідні параметри

Вхідними параметрами є:

– множина користувачів РБД;

– множина інформаційних ресурсів;

– обсяги ІР;

– інтенсивності надходження запитів від користувачів до кожного з інформаційних ресурсів;

– обсяги запитів до ІР;

– наведені витрати на зберігання ІР у вузлах комп'ютерної мережі;

– витрати на передачу одиниці інформації;

– обсяги інформації, що передається при оновленні ІР;

– множина допустимих структур РБД.

2.2.4. Вихідні параметри

Вихідними параметрами є:

– кількість локальних баз (ЛБ);

- розподіл IP по ЛБ;
- розміщення ЛБ по вузлах мережі;
- обсяги пристроїв для зберігання ЛБ;
- найкращий варіант структури РБД.

3 РОЗРОБКА МАТЕМАТИЧНОГО ЗАБЕЗПЕЧЕННЯ ЗАДАЧІ

3.1 Огляд методів проєктування розподілених БД

Загалом існує 4 стратегії розміщення даних у інформаційній системі:

– централізоване розміщення – створення на одному вузлі однієї бази даних під керуванням СКБД, доступ до якої мають усі користувачі даною системою;

– фрагментоване розміщення – розподіл БД на фрагменти які не перетинаються між собою та розміщені на різних вузлах системи;

розміщення з повною репліцкаціїєю – стратегія при якій на кожному з вузлів системи розміщується повна копія всієї БД;

– розміщення з вибірковою репліцкаціїєю – комбінація методів реплікації, фрагментації та централізації, при якій деякі масиви даних розподіляються на фрагменти, що дозволяє домогтися для них високої локалізації посилань, тоді як інші, що використовуються на багатьох вузлах, але не схильні до частих змін (оновлень) піддаються реплікації. Також деяка частина даних може зберігатися централізовано [28].

Так як при проектуванні розподіленої БД вирішується задача переходу від однієї глобальної бази даних до розподілення даних між вузлами (такий підхід можна назвати проєктуванням зверху донизу), будуть розглядатися 2 основних варіанти розміщення даних у розподілених системах: секціонування та реплікація [9].

При секціонуванні (нереплікованому розміщенні даних) база даних розбивається на декілька секцій. Які не перетинаються між собою та кожна з котрих розміщена в окремому вузлі [9]. В такому випадку кожен окремий запис в БД розбивається на декілька частин – фрагментів, котрі потім розподіляються по різним вузлах згідно з логікою розміщення даних. Фрагментація буває горизонтальна та вертикальна. У першому випадку

фрагменти представлені як множина рядків (записів) бази даних, а у другому як множина стовпців (атрибутів) БД [28].

Реплікація може бути повною (повне дублювання), тоді в кожному вузлі розміщується уся база даних (повна її копія) або часткова (часткове дублювання), коли окрема секція БД зберігається у декількох, але не у всіх вузлах системи [9].

При проектуванні розподілених баз даних виникають 2 фундаментальних питання: фрагментація, тобто розбиття даних на секції та розподіл, тобто оптимальне розміщення фрагментів по різним вузлам системи [9].

З цим також пов’язана проблема проєктування системного каталогу та керування ним, адже якщо у централізованій системи цей каталог зберігає тільки метаінформацію про самі дані (тобто їх опис), то в розподіленій системі також потрібно зберігати додаткову інформацію про місцезнаходження даних.

3.2 Фрагментація розподілених БД

Реляційні таблиці можна фрагментувати горизонтально, або вертикально. При горизонтальному секціонуванні основою є оператор вибірки, предикат якого визначає спосіб фрагментації, тоді коли вертикальне секціонування здійснюється за допомогою оператора проєктування. Також фрагментація може бути вкладеною, а якщо на різних рівнях вкладеності застосовується фрагментація різного типу, то ми отримуємо гібридну фрагментацію даних [9].

На рисунку 3.1 зображено приклад горизонтального фрагментування, де відношення PROJ розбито на 2 фрагменти, перший фрагмент має дані про проекти з бюджетом менше 200 000 доларів, другий фрагмент – про проекти з більшим бюджетом, але обидва фрагменти мають повні дані про кожний свій запис [9].

$PROJ_1$

PNO	PNAME	BUDGET	LOC
P1	Instrumentation	150000	Montreal
P2	Database Develop.	135000	New York

$PROJ_2$

PNO	PNAME	BUDGET	LOC
P3	CAD/CAM	255000	New York
P4	Maintenance	310000	Paris

Рисунок 3.1 – Приклад горизонтального секціонування

На рисунку 3.2 зображено приклад вертикального фрагментування. Тут фрагменти відношення PROJ сформовані за атрибутами, и перший фрагмент має тільки інформацію про бюджети проектів, а другий фрагмент інформацію про їх назву та місце знаходження. Але треба звернути уваги, що первинний ключ присутній у обох фрагментах [9].

$PROJ_1$

PNO	BUDGET
P1	150000
P2	135000
P3	250000
P4	310000

$PROJ_2$

PNO	PNAME	LOC
P1	Instrumentation	Montreal
P2	Database Develop.	New York
P3	CAD/CAM	New York
P4	Maintenance	Paris

Рисунок 3.2 – Приклад вертикального секціонування

Систематичні прийоми фрагментації гарантують, що у процесі фрагментації не буде ніяких змін у семантиці бази даних, наприклад втрати частини даних [9]. Тому потрібно розглянути властивості фрагментування баз даних, такі як повнота, реконструкція та неперетинність (диз'юнктивність).

Повнота – властивість, яка відповідає за те, що кожен елемент даних з вихідного відношення, повинен бути присутнім щонайменше а одному фрагменті [29]. Тобто якщо відношення R розкласти на фрагменті $F_R = \{R_1, R_2, \dots, R_n\}$, то кожен елемент даних, присутній у відношенні R може бути знайдений у одному або декількох фрагментах R_i. Ця властивість, еквівалентна безвтратній декомпозиції нормалізації, важлива і при фрагментації, тому що гарантується, що дані глобального відношення відображаються на фрагменти без втрат. Також треба відмітити, що у разі горизонтальної фрагментації під «елементом» зазвичай мається на увазі кортеж, а у разі вертикальної – атрибут [9].

Реконструкція – означає, що вихідне відношення може бути відновлено з його фрагментів за допомогою функцій реляційної алгебри [29]. Таким чином, якщо відношення розкладено на фрагменти $F_R = \{R_1, R_2, \dots, R_n\}$, то повинна існувати можливість виявити реляційний оператор ∇, такий, що $R = \nabla R_i, \forall R_i \in F_R$. Оператор ∇ буде різним для різних форм фрагментації, але повинна існувати можливість його підібрати. Можливість реконструкції відношення з його фрагментів гарантує, що обмеження, визначені у вигляді його залежностей, зберігаються [9].

Неперетинність – властивість, яка означає, що один елемент неповинен бути присутнім в двох і більше фрагментах [29]. Загалом ця властивість правдива якщо відношення фрагментовано горизонтально на фрагменти $F_R = \{R_1, R_2, \dots, R_n\}$, і коли елемент даних d належить до фрагменту R_i, то він не належить жодному іншому фрагменту $R_k (k \neq j)$. Ця умова гарантує, що горизонтальні фрагменти є диз'юнктивними. Якщо відношення R фрагментоване по вертикалі, то атрибути, що визначають первинний ключ зазвичай повторюються у всіх фрагментах (для реконструкції). Тому у разі фрагментації по вертикалі диз'юнктність потрібна лише для атрибутів, які не входять до складу первинного ключа відносини [9].

Горизонтальна фрагментація загалом зустрічається частіше, але й вертикальна фрагментація найшла своє застосування у деяких стовпчикових

паралельних СКБД, таких як MonetDB та Vertica, які застосовуються для аналітичних застосунків, де потрібний швидкий доступ до конкретних атрибутів [9].

3.3 Математичний опис методів горизонтальної фрагментації БД

Існує два варіанти горизонтальної фрагментації: головна та похідна. Головна горизонтальна фрагментація відносини проводиться за допомогою предикатів, визначених на даному відношенні. У свою чергу похідна горизонтальна фрагментація – це фрагментація відносини, що виходить за допомогою предикатів, визначених на іншому плані [9].

3.3.1 Головна фрагментація

Для головної горизонтальної фрагментації існують алгоритми COM_MIN та PHORIZONTAL.

Перед розглядом цих алгоритмів потрібно визначити одне правило.

Правило 1: до кожного фрагменту хоча б один додаток звертається не так як інші [9].

Алгоритм COM_MIN починається з пошуку релевантного предикату, який секціонує вхідне відношення. У циклі repeat–until у множину ітеративно додаються предикати з дотриманням на кожному кроці умови мінімальності. Тому наприкінці циклу множина Pr' є одночасно мінімальним та повним [9].

На другому етапі процесу проєктування головної горизонтальної фрагментації потрібно знайти множину елементарних кон'юнкцій, які можна визначити над предикатами з множини Pr'. Ці елементарні кон'юнкції визначають фрагменти, які стануть кандидатами на кроці розміщення. Визначити окремі елементарні кон'юнкції просто, труднощі ж у тому, що множина таких елементарних кон'юнкцій може бути дуже великою (її розмір експоненційно залежить від числа простих предикатів) [9]. Схема алгоритму

COM_MIN зображена на рисунку 3.3.

У алгоритмі PHORIZONTAL входом є відношення *R*, що підлягає головній горизонтальній фрагментації, і множина простих предикатів *Pr*, знайдене відповідно до додатків, що звертаються до R [9]. Схема алгоритму PHORIZONTAL зображена на рисунку 3.4.

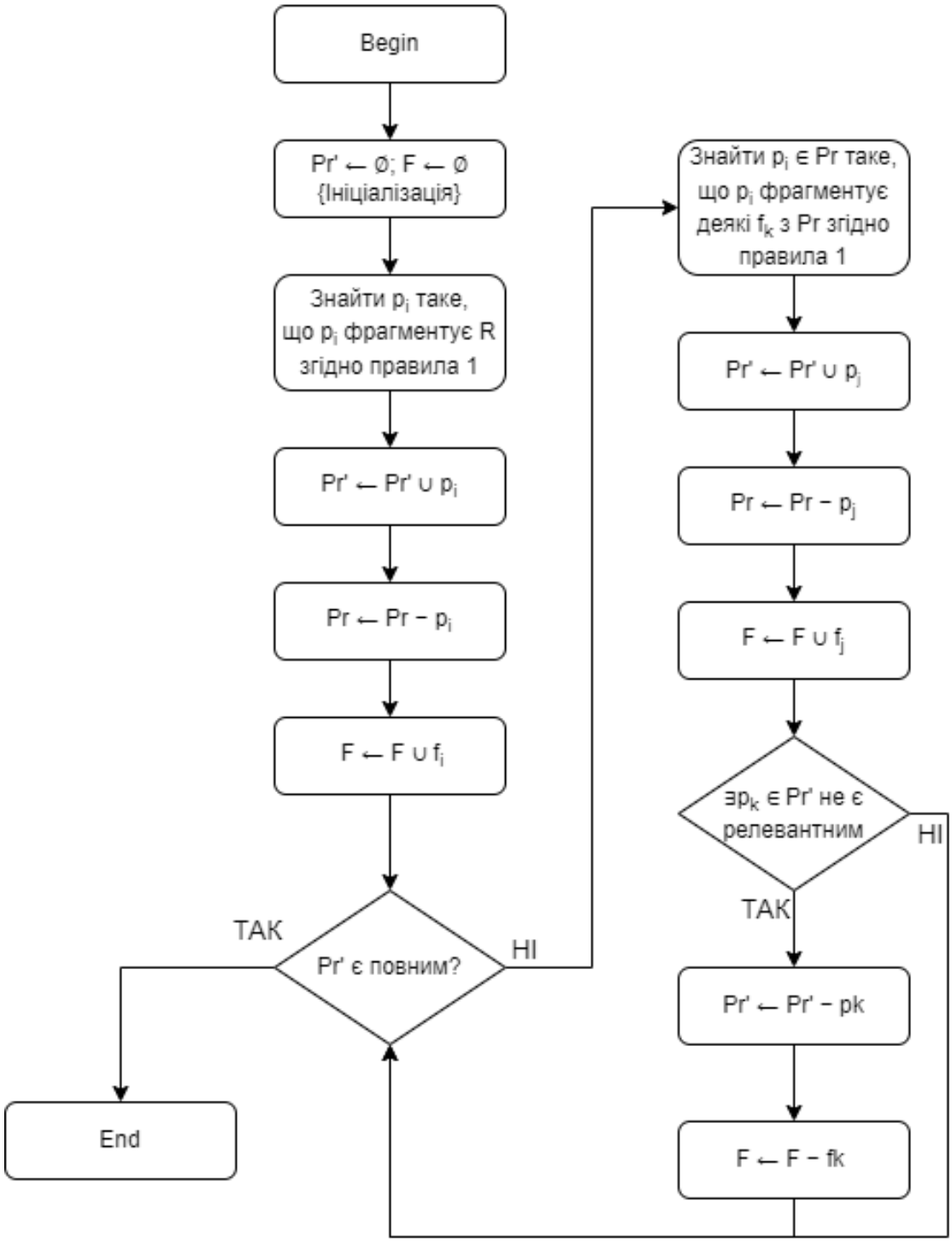

Рисунок 3.3 – Схема алгоритму COM_MIN

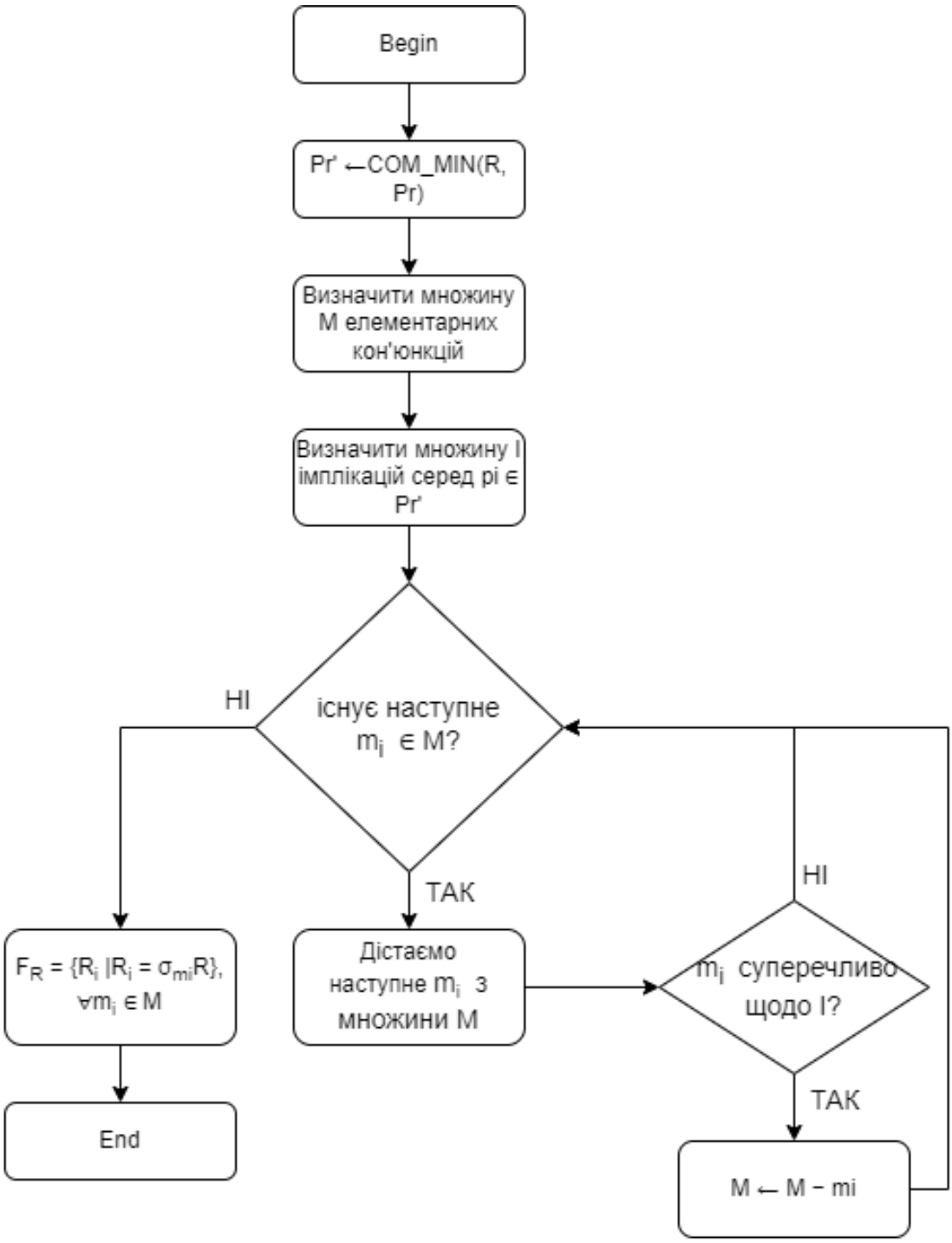

Рисунок 3.4 – Схема алгоритму PHORIZONTAL

3.3.2 Похідна фрагментація

Похідна горизонтальна фрагментація – це фрагментація відносини, що виходить за допомогою предикатів, визначених на іншому плані [9].

Похідна горизонтальна фрагментація застосовується до цільових відносин у графі з’єднань і виконується на основі предикатів, визначених над вихідним відношенням ребра графа з’єднань.

Якщо задано ребро L, де $source(L) = S$ та $target(L) = R$, то похідні горизонтальні фрагменти R визначаються наступним чином:

$$R_i = R \ltimes S_i, 1 \leq i \leq w,$$

де w – максимальне число фрагментів, визначених на R, а

$$S_i = \sigma_{F_i}(S),$$

де F_i – формула, за якою визначено головний горизонтальній фрагмент S_i [9].

Для виконання похідної горизонтальної фрагментації потрібно задати три види вхідних даних:

– множина фрагментів вихідного відношення;

– цільове відношення;

– множина предикатів напівз’єднання між початковим та цільовими відношеннями [9].

Існує одне потенційне ускладнення, що заслуговує на увагу. Часто буває, що у схемі бази даних є декілька ребер, що входять до відношення R.

У такому випадку є кілька можливих похідних фрагментацій R. Вибір однієї з них ґрунтується на двох критеріях:

а) фрагментація з найкращими характеристиками з'єднання;

б) фрагментація, що використовується у більшій кількості запитів.

Спочатку розглянемо другий критерій. Він цілком очевидний, якщо прийняти до уваги частоту звернення до даних при наявному робочому навантаженні. За можливістю слід спростити доступ «важким» користувачам, щоб звести до мінімуму їх впливом геть загальну продуктивність системи.

Але застосування першого критерію менш очевидно. Нам потрібно

пришвидшити з’єднання між відносинами:

– виконуючи його над меншими відношеннями (фрагментами)

– якомога більше розпаралелити операцію з’єднання.

Якщо з першою частиною все більш-менш зрозуміло, то у другому пункті ми маємо справу з паралелізмом з’єднання всередині запитів, тобто виконанням усіх з’єднань паралельно, що можливо лише за певних умов, наприклад, якщо ми маємо справу з простим графом з’єднань. Перевага структури, в якій з'єднання між фрагментами просте, в тому, що джерело і ціль ребра можна розмістити в одному вузлі, а з'єднання між різними парами фрагментів можна обробляти незалежно і паралельно. На жаль, звести справу до простих граф з'єднань не завжди можливо. Тоді наступна в порядку переваги альтернатива – знайти структуру, що призводить до розрізаного графа з'єднань. Розрізаний граф складається із двох або більше підграфів, між якими немає жодного ребра. Отримані таким чином фрагменти не можна розподілити для паралельного виконання так просто, як у разі простих графів з'єднань, але розміщення все ж таки можливе [9].

3.4 Математичний опис методів вертикального фрагментування

Для вертикального фрагментування нам потрібна деяка додаткова інформація про робоче навантаження. Оскільки при вертикальному секціонуванні в один фрагмент розміщуються атрибути, доступ до яких зазвичай проводиться спільно, то потрібна якась міра "Сумісності".

Цим заходом є спорідненість (affinity) атрибутів, що показує, наскільки тісно вони пов’язані між собою. Важко очікувати, що проектувальник або користувачі можуть точно вказати цю величину. Тому потрібно отримати її з більш примітивних даних.

Означимо $Q = \{q_1, q_2, \dots, q_q\}$ як множину користувацьких запитів, які звертаються до відношення $R(A_1, A_1, \dots, A_n)$. Тоді з кожним запитом q_i і кожним атрибутом A_j асоціюється показник використання атрибуту

$use(q_i, A_j)$, і можна вирахувати вектори $use(q_i, \bullet)$ для кожного запиту нескладно

$$use(q_i, A_j) = \begin{cases} 1, \text{якщо атрибут } A_j \text{ згадується у запиті } q_i, \\ 0 \text{ в іншому випадку.} \end{cases}$$

Показники використання атрибутів недостатньо загальні для того, щоб лягти в основу розщеплення та фрагментації, оскільки не відображають частоту виконання програми.

Цю частоту можна включити до визначення міри спорідненості $aff(A_i, A_j)$, яка вимірює зв'язок між атрибутами відношення відповідно до того, як до них звертаються запити.

Спорідненість атрибутів A_i та A_j відношення $R(A_1, A_1, \dots, A_n)$ відносно множини запитів $Q = \{q_1, q_2, \dots, q_q\}$ визначається наступним чином:

$$aff(A_i, A_j) = \sum_{k | use(q_k, A_i)=1 \Lambda use(q_k, A_j)=1} \sum_{\forall S_1} ref_l(q_k) acc_l(q_k),$$

де $acc_l(q_k)$ – частота доступу з боку програм, яка була визначена раніше і модифікована з урахуванням частот у різних вузлах;

$ref_l(q_k)$ – кількість звертань до атрибуту (A_i, A_j) для кожного виконання програми q_k в узлі S_l. Результатом цього обчислення є матриця $n \times n$, елементами якої є певні числа. Вона називається матрицею спорідненості атрибутів (AA).

Матриця спорідненості атрибутів використовується як орієнтир для алгоритмів вертикальної фрагментації [9].

3.4.1 Алгоритм кластеризації

Головне завдання при проектуванні алгоритму вертикальної

фрагментації – знайти спосіб угруповання атрибутів відношення на основі елементів матриці кревності атрибутів.

Для цього можна використати алгоритм енергії зв'язків (bond energy algorithm – BEA). BEA приймає матрицю спорідненості атрибутів для відношення $R(A_1, A_1, \dots, A_n)$, переставляє її рядки та стовпці та породжує кластерну матрицю спорідненості (CA).

Перестановки виконуються таким чином, щоб максимізувати наступний захід глобальної спорідненості (AM):

$$AM = \sum_{i=1}^{n}\sum_{j=1}^{n} aff(A_i, A_j)\begin{bmatrix} aff(A_i, A_{j-1}) + aff(A_i, A_{j+1}) + \\ + aff(A_{i-1}, A_j) + aff(A_{i+1}, A_j) \end{bmatrix},$$

де

$$aff(A_0, A_j) = aff(A_i, A_0) = aff(A_{n+1}, A_j) = aff(A_i, A_{n+1}) = 0.$$

Останній набір умов враховує випадки, коли атрибут вміщується у CA зліва від самого лівого атрибута або праворуч від самого правого при перестановці стовпців, або перед першим рядком або після останнього рядка при перестановці рядків.

Будемо позначати A_0 атрибут ліворуч від лівого атрибута та рядок перед першим рядком, а A_{n+1} – атрибут праворуч від самого правий атрибут або рядок після останнього рядка.

У цих випадках ми вважаємо рівними 0 значення aff між атрибутом, що розглядається для приміщення, та його лівим та правим (верхнім або нижнім) сусідами, оскільки їх немає в CA [9].

Функція максимізації враховує лише найближчих сусідів, тому великі значення групуються з більшими, а малі з малими. Крім того, матриця спорідненості атрибутів (AA) симетрична, тому цільова функція зводиться до

$$AM = \sum_{i=1}^{n}\sum_{j=1}^{n} aff(A_i, A_j)\left[aff(A_i, A_{j-1}) + aff(A_i, A_{j+1})\right].$$

Породження кластерної матриці спорідненості (CA) складається із трьох кроків:

– ініціалізація. Помістити та зафіксувати довільно обраний стовпець AA в CA;

– ітерація. Вибрати один з стовпців, що залишилися $n - i$ (де i – кількість стовпців, вже розміщених у CA) і спробувати помістити їх у решту $i + 1$ позицій матриці CA. Вибрати таке місце для розміщення, при якому робиться максимальний внесок в описану вище міру глобального кревності. Повторювати цей крок, доки не залишиться стовпців, які можна було б помістити;

– упорядкування рядків. Після того, як порядок стовпців визначено, слід змінити порядок рядків, щоб їх відносні позиції збігалися з відносними позиціями стовпців [9].

Схема алгоритму BEA зображена подана у додатку А, на рисунку А.1.

Щоб другий крок алгоритму працював, потрібно визначити, що розуміється під вкладом атрибута в міру спорідненості. Цей внесок можна визначити таким чином.

Нагадаємо, що міра глобальної спорідненості AM вище була визначена як

$$AM = \sum_{i=1}^{n}\sum_{j=1}^{n} aff(A_i, A_j)\left[aff(A_i, A_{j-1}) + aff(A_i, A_{j+1})\right],$$

що можна записати у вигляді

$$AM = \sum_{i=1}^{n}\sum_{j=1}^{n} aff(A_i, A_j)aff(A_i, A_{j-1}) + aff(A_i, A_j)aff(A_i, A_{j+1}) =$$

$$= \sum_{j=1}^{n}\left[\begin{matrix}\sum_{i=1}^{n} aff(A_i, A_j)aff(A_i, A_{j-1}) + \\ + \sum_{i=1}^{n} aff(A_i, A_j)aff(A_i, A_{j+1})\end{matrix}\right].$$

Визначимо зв'язок між двома атрибутами A_x та A_y:

$$bond(A_xA_y) = \sum_{i=1}^{n} aff(A_z, A_x)aff(A_z, A_y).$$

Тоді зв’язок між атрибутами A_x та A_y можна записати:

$$\sum_{i=1}^{n}[bond(A_jA_{j-1}) + bond(A_jA_{j+1})].$$

Тепер розглянемо наступні n атрибутів:

$$\underbrace{A_1\ A_2 \dots A_{i-1}}_{AM'}\ A_iA_j\ \underbrace{A_{j+1} \dots A_n}_{AM^1}.$$

Міру глобальної спорідненості для цих атрибутів можна записати так:

$$AM_{old} = AM' + AM^1 + bond(A_{i-1}A_i) + bond(A_jA_i) + bond(A_jA_{j+1}) =$$

$$= \sum_{l=1}^{i}[bond(A_lA_{l-1}) + bond(A_lA_{l+1})] +$$

$$+ \sum_{l=i+2}^{n} [bond(A_l A_{l-1}) + bond(A_l A_{l+1})] + 2bond(A_i A_j).$$

Розглянемо розміщення нового атрибуту A_k у кластерну матрицю спорідненості між атрибутами A_i та A_j. Нову міру глобальної спорідненості можна так само записати у вигляді:

$$AM_{new} = AM' + AM^1 + bond(A_i A_k) + bond(A_k A_i) +$$
$$+ bond(A_k A_j) + bond(A_j A_k)$$
$$AM_{new} = AM' + AM^1 + 2bond(A_i A_k) + 2bond(A_j A_k).$$

Таким чином, внесок у міру глобальної спорідненості, що виходить в результаті приміщення атрибуту A_k між A_i та A_j, дорівнює

$$cont(A_j A_k A_i) = AM_{new} - AM_{old} =$$
$$= 2bond(A_i A_k) + 2bond(A_k A_j) - 2bond(A_i A_j).$$

3.4.2 Алгоритм розщеплення

Мета розщеплення – знайти такі набори атрибутів, звернення до яких виробляються повністю або хоча б здебільшого різних наборів запитів. Наприклад, якщо вдасться знайти два атрибути A_1 та A_2, до яких звертається лише запит q_1, та атрибути A_3 та A_4, до яких звертаються тільки запити q_2 та q_3, то визначитися з фрагментами просто. Завдання в тому, щоб вигадати алгоритмічний метод знаходження таких груп.

Розглянемо кластерну матрицю атрибутів на рисунку 3.5.

Якщо зафіксувати точку на діагоналі, то вийде два набори атрибутів: $\{A_1, A_2, \ldots, A_i\}$ у лівому верхньому куті (позначений ТА) і $\{A_{i+1}, \ldots, A_n\}$ у правому нижньому куті (позначений ВА) [9].

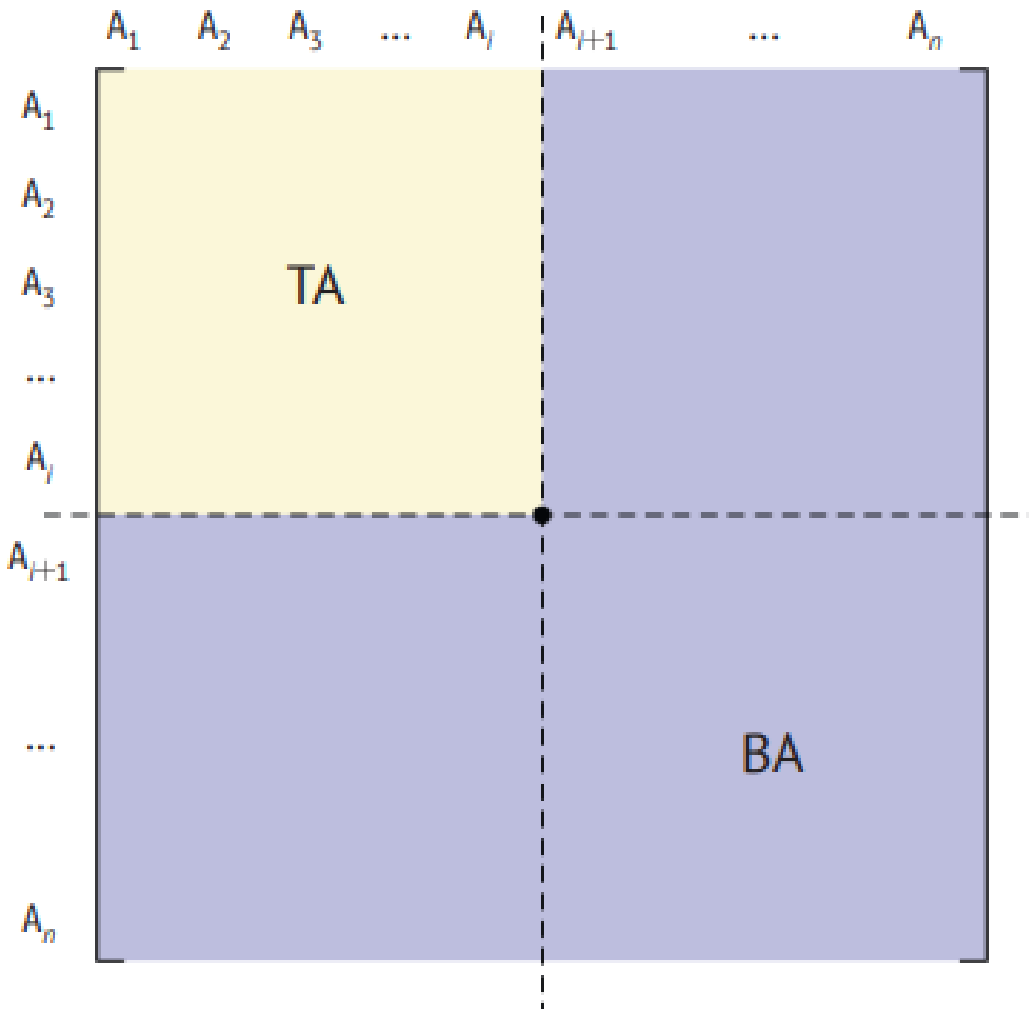

Рисунок 3.5 – Знаходження точки розщеплення

Тепер розіб'ємо множину запитів $Q = \{q_1, q_2, \dots, q_q\}$ н підмножини, які звертаються тільки до атрибутів з ТА, тільки з ВА, або і до тих і до інших. Ці підмножини визначаються наступним чином:

$$AQ(q_i) = \{A_j | use(q_i, A_j) = 1\};$$
$$TQ = \{q_i | AQ(q_i) \subseteq TA\};$$
$$BQ = \{q_i | AQ(q_i) \subseteq BA\};$$
$$OQ = Q - \{TQ \cup BQ\}.$$

Перша рівність визначає множину атрибутів, до яких звертається запит q_i; TQ і BQ – множина запитів, що звертаються тільки до атрибутів з ТА або ВА відповідно, а OQ – множина запитів, що звертаються до тих та інших

атрибутів.

Таким чином перед нами постає задача оптимізації. Якщо щодо n атрибутів, існує $n - 1$ способів поставити точку поділу на діагоналі кластерної матриці атрибутів.

Найкращим буде той, при якому породжуються такі множини TQ і BQ, для яких кількість звернень тільки до одного фрагмента досягає максимуму, а кількість звернень до обох фрагментів досягає мінімуму. Тому визначимо такі чотири величини:

$$CQ = \sum_{q_i \in Q} \sum_{\forall S_j} ref_i(q_i) acc_j(q_i),$$

$$CTQ = \sum_{q_i \in TQ} \sum_{\forall S_j} ref_i(q_i) acc_j(q_i),$$

$$CQB = \sum_{q_i \in BQ} \sum_{\forall S_j} ref_i(q_i) acc_j(q_i),$$

$$COQ = \sum_{q_i \in QO} \sum_{\forall S_j} ref_i(q_i) acc_j(q_i).$$

Кожна з них підраховує загальну кількість звернень до атрибутів із запитів, що належать відповідним класам. Тоді завдання оптимізації полягає в тому, щоб знайти точку x $(1 \leq x \leq n)$, що доставляє максимум виразу

$$z = CTQ * CBQ - COQ^2.$$

Важлива особливість цього виразу полягає в тому, що він визначає два фрагменти таких, що величини CTQ і CBQ настільки близькі, наскільки це можливо.

Це дозволить збалансувати навантаження, коли фрагменти розподілені за різними вузлами. Зрозуміло, що складність алгоритму секціонування лінійно залежить від кількості атрибутів відносини, тобто дорівнює $O(n)$ [9].

Ця процедура розбиває багато атрибутів на дві частини. Якщо множина атрибутів велика, то цілком може знадобитися m-шляхове секціонування. Спроектувати такий алгоритм можна, але він буде обчислювально складним. На діагоналі матриці CA потрібно буде поставити $1, 2, \ldots, m - 1$ точок поділу і кожної кількості точок знайти позиції, у яких z досягає максимуму.

Отже, складність алгоритму буде становити $O(2^m)$. Звичайно, визначення z доведеться модифікувати для випадку, коли точок поділу декілька.

Альтернативне рішення – рекурсивно застосовувати алгоритм бінарного секціонування до кожного фрагмента, отриманого на попередній ітерації. Потрібно буде обчислити TQ, BQ та OQ, а також асоційовані з ними заходи доступу для кожного фрагмента та продовжити їхнє розбиття [9].

До цього часу припускалося, що точка поділу одна, однозначно визначається і розбиває матрицю CA на лівий верхній блок і другий блок, що містить всі інші атрибути.

Однак розбиття можна сформувати і всередині матриці. І тут алгоритм потрібно трохи модифікувати. Найлівіший стовпець матриці CA зсувається і стає правим, а верхній рядок зсувається вниз. Операція зсуву супроводжується перевіркою *n – 1* положення на діагоналі з метою знаходження максимального *z*.

Ідея такого зсуву полягає в тому, щоб перемістити блок атрибутів, який повинен складати кластер, у верхній лівий кут матриці, де його легко знайти. Після додавання операції зсуву складність алгоритму секціонування зростає в n разів і виявляється рівною $O(n^2)$ [9].

Алгоритм розщеплення (SPLIT) надано на рисунку 3.6. На вході він приймає кластерну матрицю спорідненості CA, що підлягає фрагментації відношення, а також матриці використання атрибутів та частот доступу.

На виході виходить множина фрагментів $F_R = \{R_1, R_2\}$, де $R_i \subseteq \{A_1, A_2 \ldots, A_n\}$ і $R_1 \cap R_2 =$ множини атрибутів, що входять до складу первинного ключа R [9].

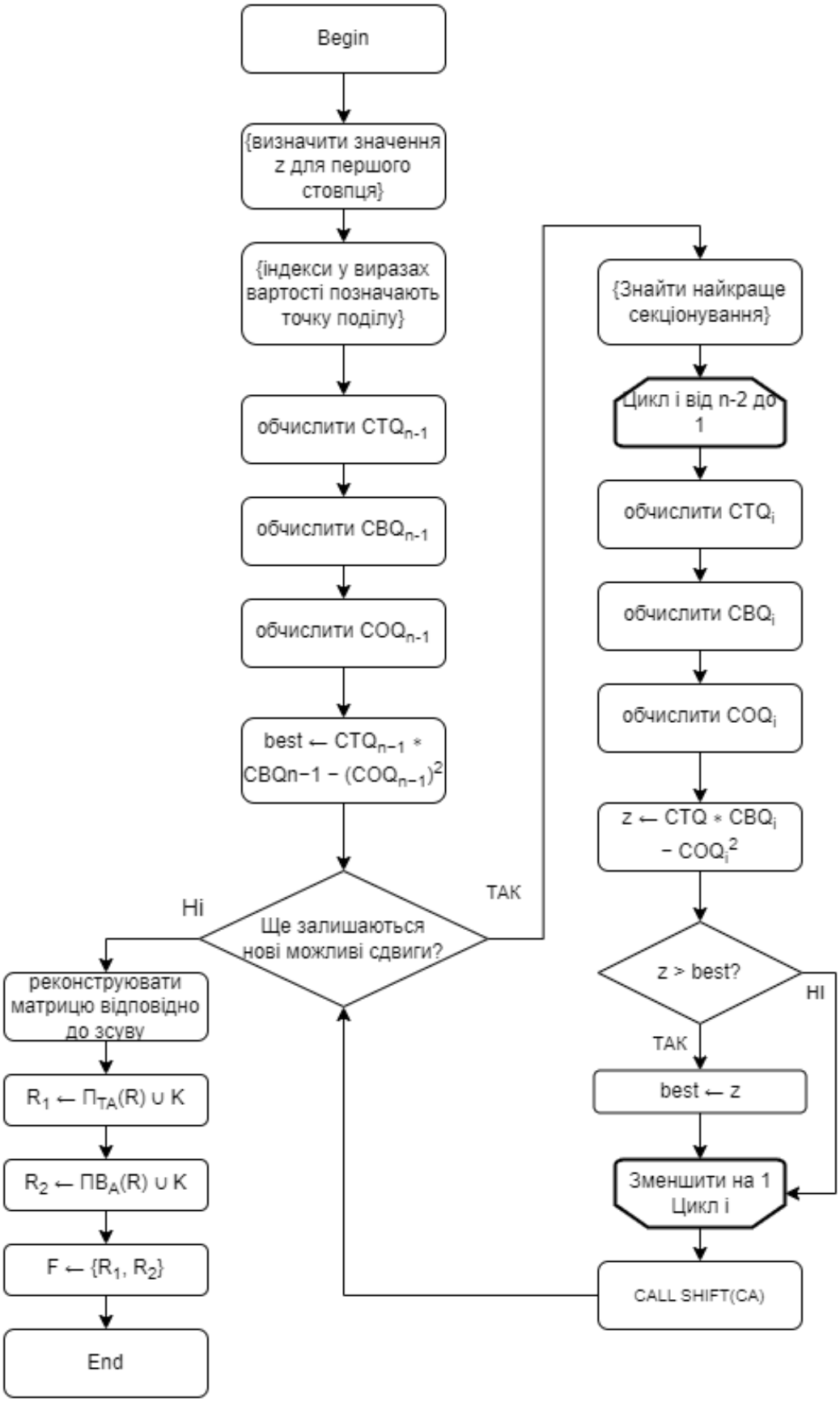

Рисунок 3.6 – Схема алгоритму розщеплення (SPLIT)

4 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ЗАДАЧІ ТА ЕКСПЕРИМЕНТИ

4.1 Вибір середовища розробки та мови програмування

4.1.1 Мова програмування C#

C# відноситься до сім'ї мов з C-подібним синтаксисом, з них синтаксис найбільш близький до C++ і Java. Мова має строгу статичну типізацію, підтримує поліморфізм, перевантаження операторів, покажчики на функції-члени класів, атрибути, події, властивості, виключення, коментарі у форматі XML. Перейнявши багато чого від своїх попередників – мов C++, Delphi – C# виключає деякі моделі, що зарекомендували себе як проблематичні при розробці програмних систем: так, C# не підтримує множинне успадкування класів або виведення типів [30].

C# розроблялася як мова програмування прикладного рівня для CLR (Common Language Runtime) і, як такий, залежить, перш за все, від можливостей самої CLR. Це стосується, перш за все, системи типів C#, яка відображає BCL (Base Class Library). Присутність або відсутність тих чи інших виразних особливостей мови диктується тим, чи може конкретна мовна особливість бути трансльований в відповідні конструкції CLR. Так, з розвитком CLR від версії 1.1 до 2.0 значно збагатився і сам C #; подібної взаємодії слід очікувати і в подальшому (проте, ця закономірність була порушена з виходом C# 3.0, що представляє собою розширення мови, що не спираються на розширення платформи .NET). CLR надає C#, як і всім іншим .NET-орієнтованим мовам, багато можливостей, яких позбавлені «класичні» мови програмування. Наприклад, прибирання сміття не реалізована в самому C#, а проводиться CLR для програм, написаних на C# точно так же, як це робиться для програм на VB.NET, J# та ін. [30].

4.1.2 Середовище розробки Microsoft Visual Studio

Microsoft Visual Studio зазвичай розуміють як інтегроване середовище розробки програмного забезпечення і ряд інших інструментальних засобів [28]. Дані продукти дозволяють розробляти як консольні програми, так і програми з графічним інтерфейсом, в тому числі з підтримкою технології Windows Forms, а також веб-сайти, веб-додатки, веб-служби як у рідному, так і в керованому коді для всіх платформ, підтримуваних Windows, Windows Mobile, Windows CE .NET Framework, Xbox, Windows Phone .NET Compact Framework і Silver light [30].

Visual Studio також включає в себе дуже потужний редактор коду, з підтримкою технології автодоповнення IntelliSense, яка пришвидшує написання коду, та допомагає не допускати розповсюджених помилок.

4.1.3 Технологія Windows Forms

Windows Forms — це структура інтерфейсу користувача для створення настільних програм Windows. Він надає один із найпродуктивніших способів створення настільних програм на основі візуального конструктора Visual Studio. Такі функціональні можливості, як розміщення візуальних елементів керування за допомогою перетягування, спрощують створення настільних програм [31].

Технологія Windows Forms інтерфейсу користувача для .NET має великий набір керованих бібліотек, які спрощують типові завдання програми, наприклад читання та запис у файлову систему. З використанням середовища розробки, наприклад Visual Studio, можна доволі просто створювати інтелектуальні клієнтські програми, які відображають інформацію, запитують введення від користувачів, спілкуються з віддаленими комп’ютерами через мережу, тощо [31].

Хоча на сьогоднішній день технологію Windows Forms можна вважати

дещо застарілою, вона все ще продовжує отримувати оновлення, та підходить до вирішення багатьох нескладних задач, які потребують швидко відтворити користувацький інтерфейс та елементи управління. Таким чином технологія Windows Forms підходить для прототипування та невеликих настільних застосунків загалом.

4.2 Опис структури програми

Для програмної реалізації було обрано оптимізувати розміщення інформаційних ресурсів по локальним базам даних за допомогою методу покординатної оптимізації. Суть процедури оптимізації полягає в поліпшенні деякого початкового розміщення m інформаційних ресурсів на n вузлах комп'ютерної мережі шляхом послідовного переміщення одного з ресурсів при фіксованому розміщенні m – 1 інших. При цьому початкове розміщення інформаційних ресурсів формується випадковим способом, а для кожного з варіантів розміщення визначаються необхідні ємності запам'ятовуючих пристроїв для зберігання локальних баз даних і пропускні здатності каналів зв'язку комп'ютерної мережі.

Для визначення кращого розміщення також визначається розташування користувачів, які звертаються до кожного з інформаційних ресурсів. Так як локальні бази окрім отримання запитів від користувачів, можуть отримувати запити і від інших локальних баз, потрібно враховувати і розташування інформаційних ресурсів відносно один одного.

Локальним базам та користувачам можна надавати вагові коефіцієнти, які покажуть різні сценарії розташування інформаційних ресурсів, при більшій залежності від користувачів, або від інших IP.

Цей метод дозволяє за прийнятний час отримувати раціональні розв'язки. Зміна якості (точність) розв'язків залежить від задання і вибору початкового варіанту розподілу інформаційних ресурсів по вузлах комп'ютерної мережі.

Код програмного застосунку наведено у додатку Б.

4.3 Планування та проведення експериментів

4.3.1 Експеримент 1

Запустимо застосунок та проведемо покоординатну оптимізацію при рівномірному розташуванні користувачів на координатній сітці і з найменшим впливом розташування інформаційних ресурсів один до одного. Для цього згенеруємо велику кількість (до 1000) рівнозначних користувачів (ваговий коефіцієнт користувача = 1) та знехтуємо розташуванням інформаційних ресурсів один до одного (ваговий коефіцієнт IP = 0)

Результати експерименту 1 надано на рисунку 4.1.

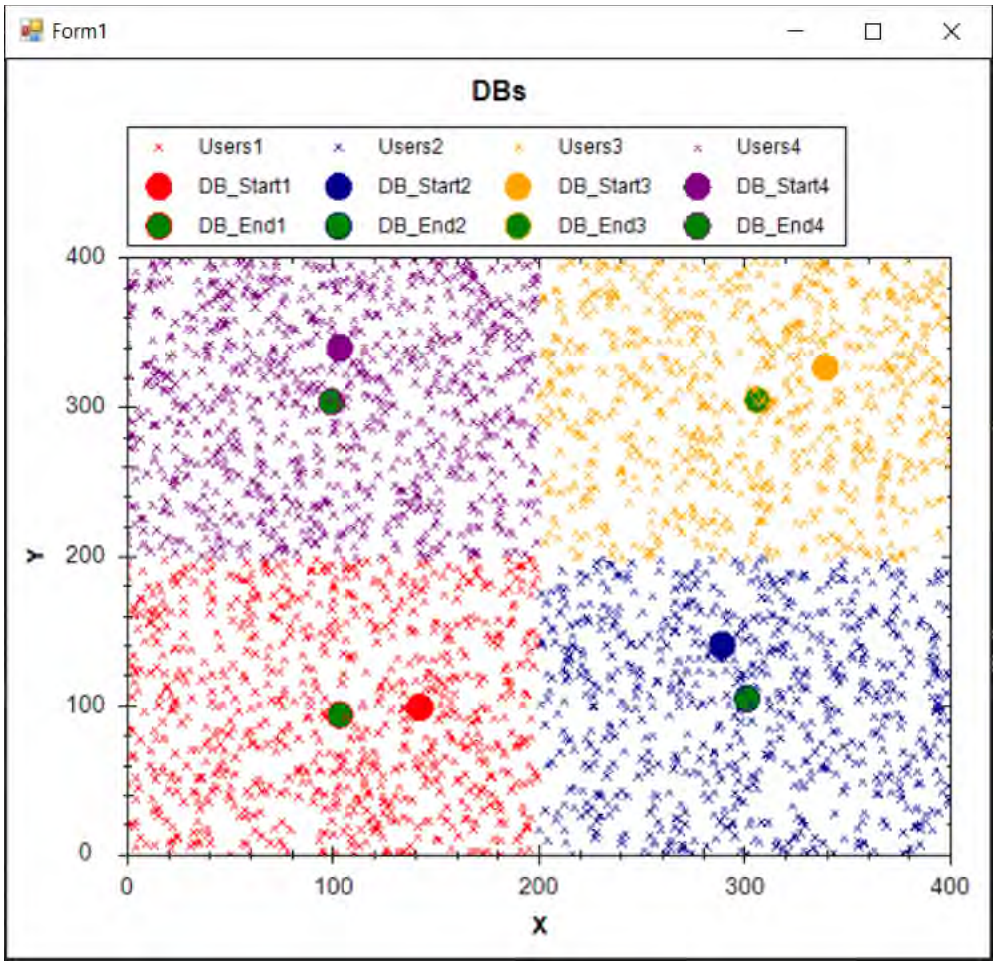

Рисунок 4.1 – Результати експерименту 1

4.3.2 Експеримент 2

Тепер запустимо застосунок та проведемо покоординатну оптимізацію при нерівномірному розташуванні користувачів на координатній сітці і з середнім впливом розташування інформаційних ресурсів один до одного. Для цього згенеруємо невелику кількість (до 20) нерівнозначних користувачів (ваговий коефіцієнт користувача у діапазоні від 1 до 2) та визначимо коефіцієнт розташування інформаційних ресурсів один до одного рівний 5.

Результати експерименту 2 надано на рисунку 4.2.

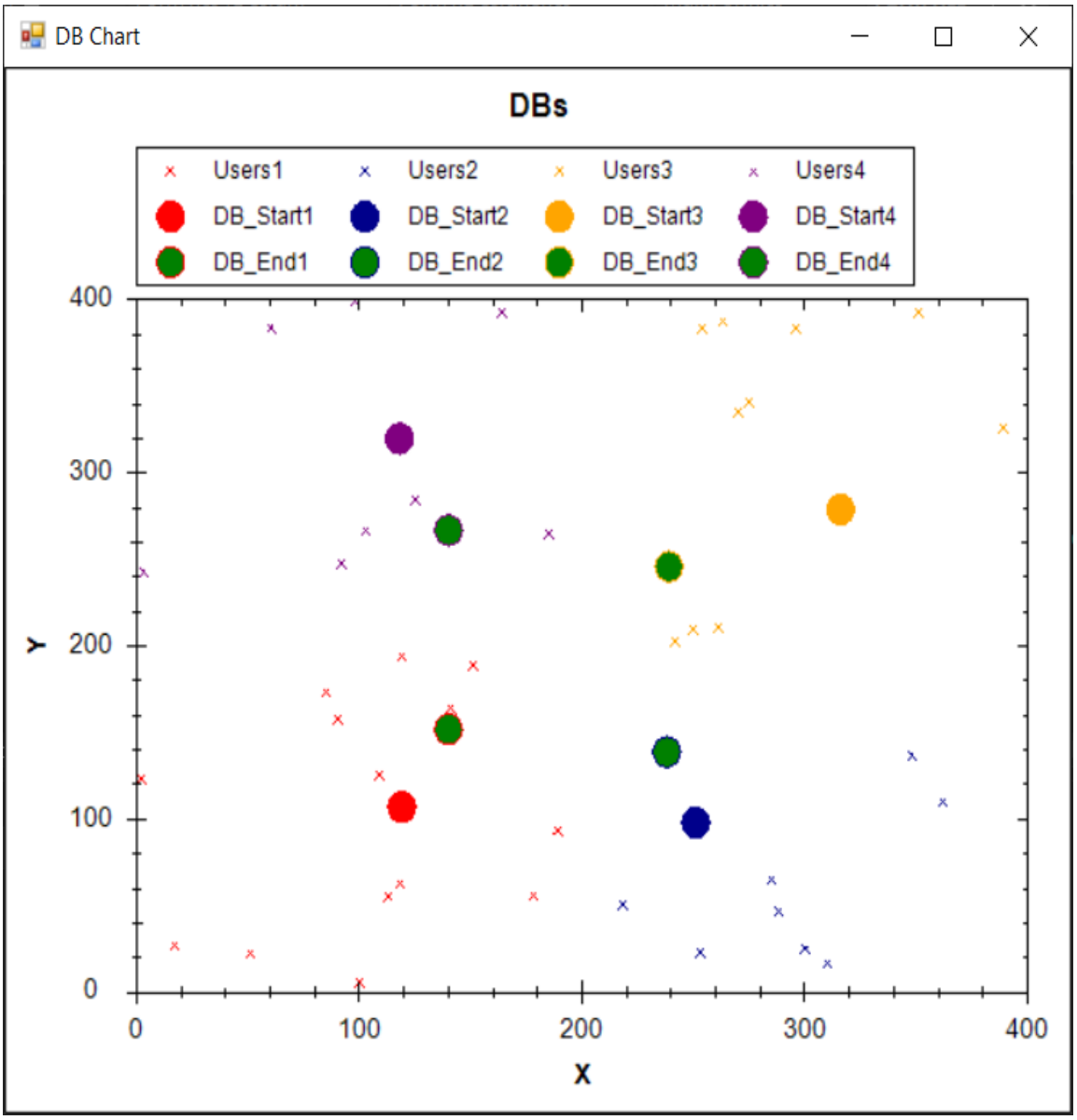

Рисунок 4.2 – Результати експерименту 2, прогін 1

Запустимо застосунок ще раз, результати другого прогону експерименту 2 надано на рисунку 4.3.

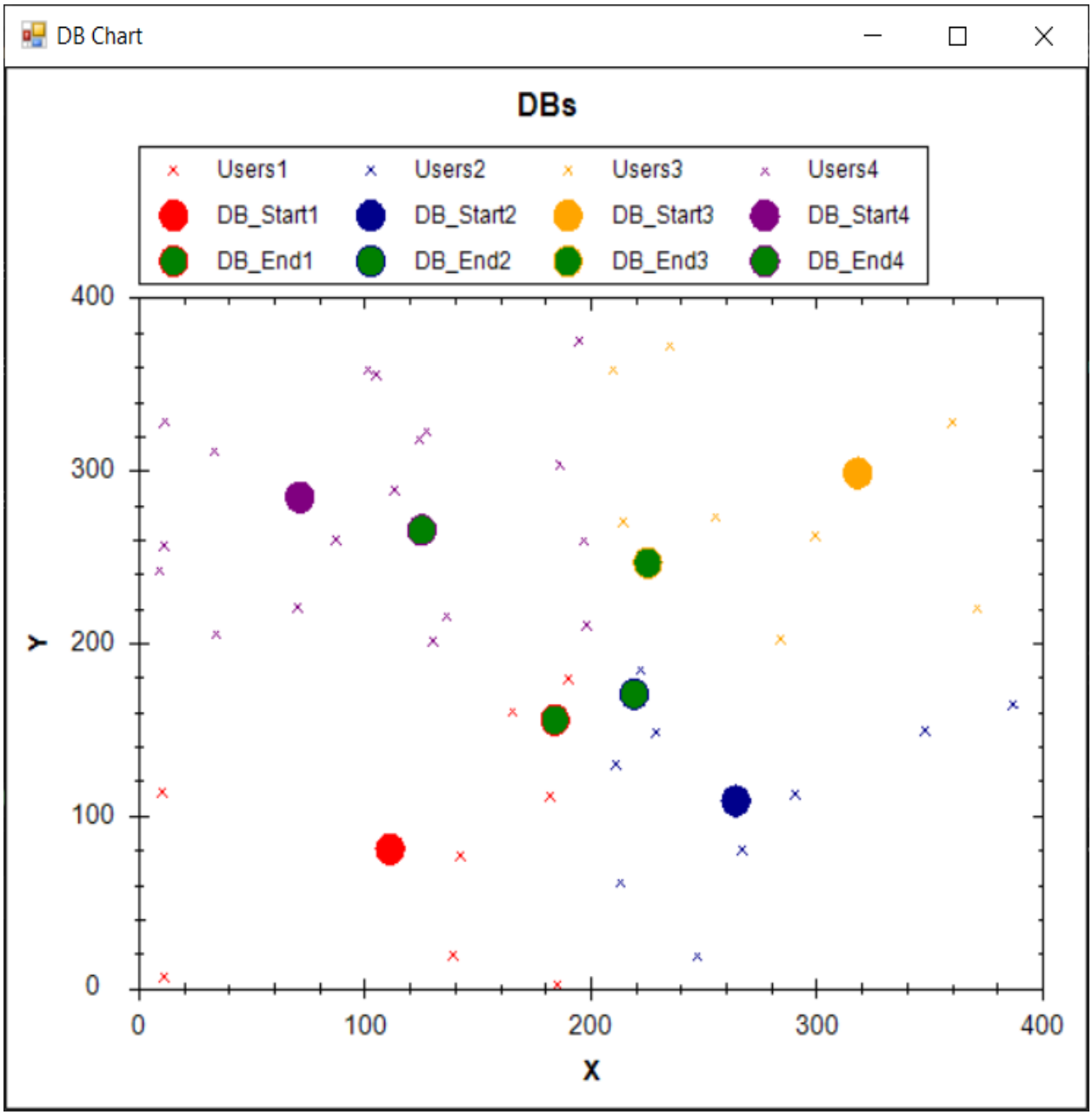

Рисунок 4.3 – Результати експерименту 2, прогін 2

Для другої частини експерименту 2, залишимо кількість користувачів та їх вагові коефіцієнти незмінними, але встановимо коефіцієнт розташування інформаційних ресурсів один до одного рівний 0. Результати даної частини експерименту 2 зображено на рисунку 4.4

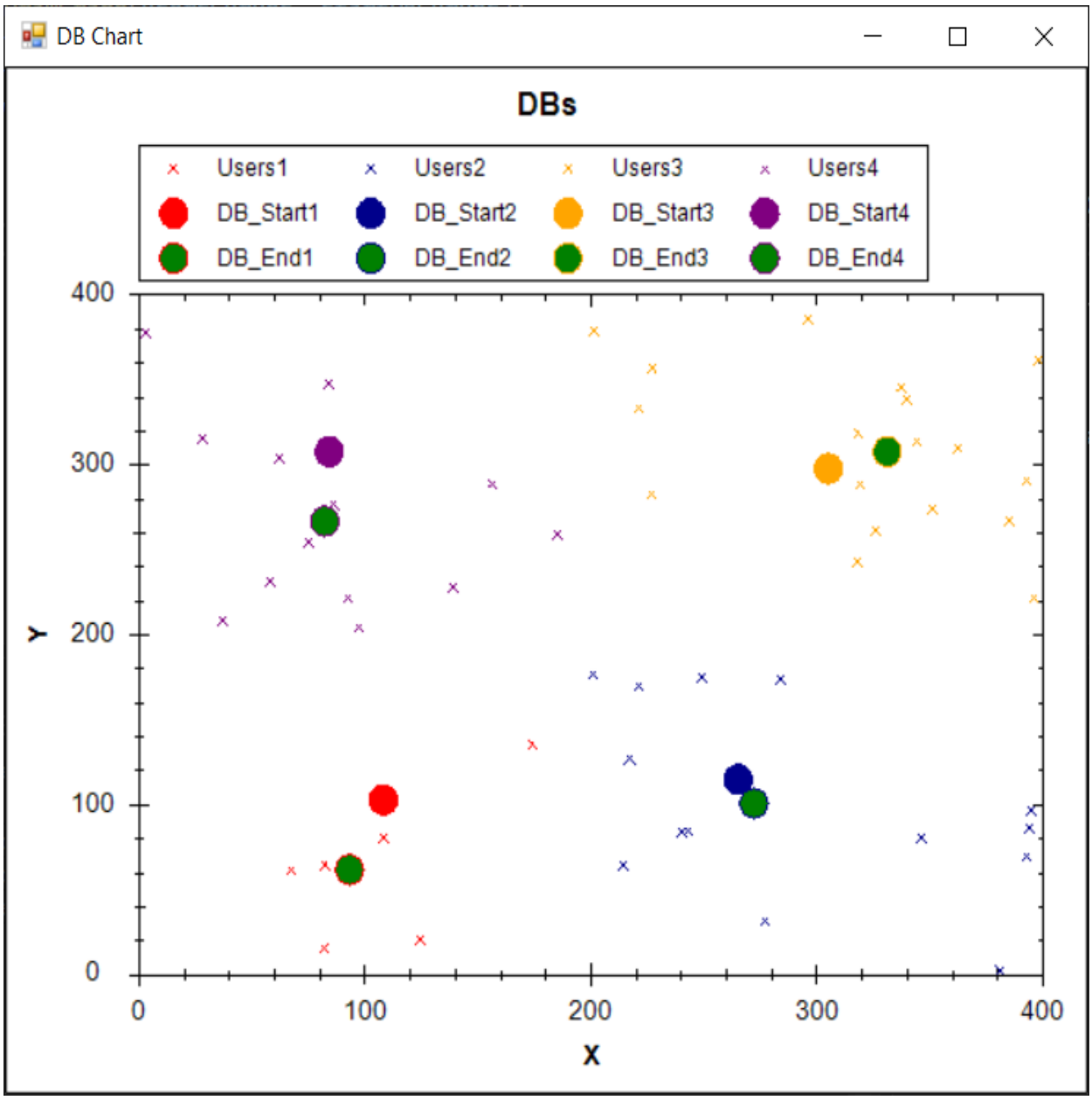

Рисунок 4.4 – Результати експерименту 2, частина 2

4.3.3 Експеримент 3

Наступний експеримент проведемо з керованим зміщенням розміщення користувачів до «кутів» системи координат. Для цього згенеруємо велику множину користувачів (від 50 до 200) з розподілом, який наближає більшість користувачів до кутових координат кожної секції та випадковими коефіцієнтами для кожного користувача у діапазоні від 1 до 2.

Після чого визначимо коефіцієнт розташування інформаційних ресурсів

один до одного рівний 30, що є доволі великим значенням з урахуванням кількості користувачів.

Результати проведення першої частини експерименту 3 надано на рисунку 4.5

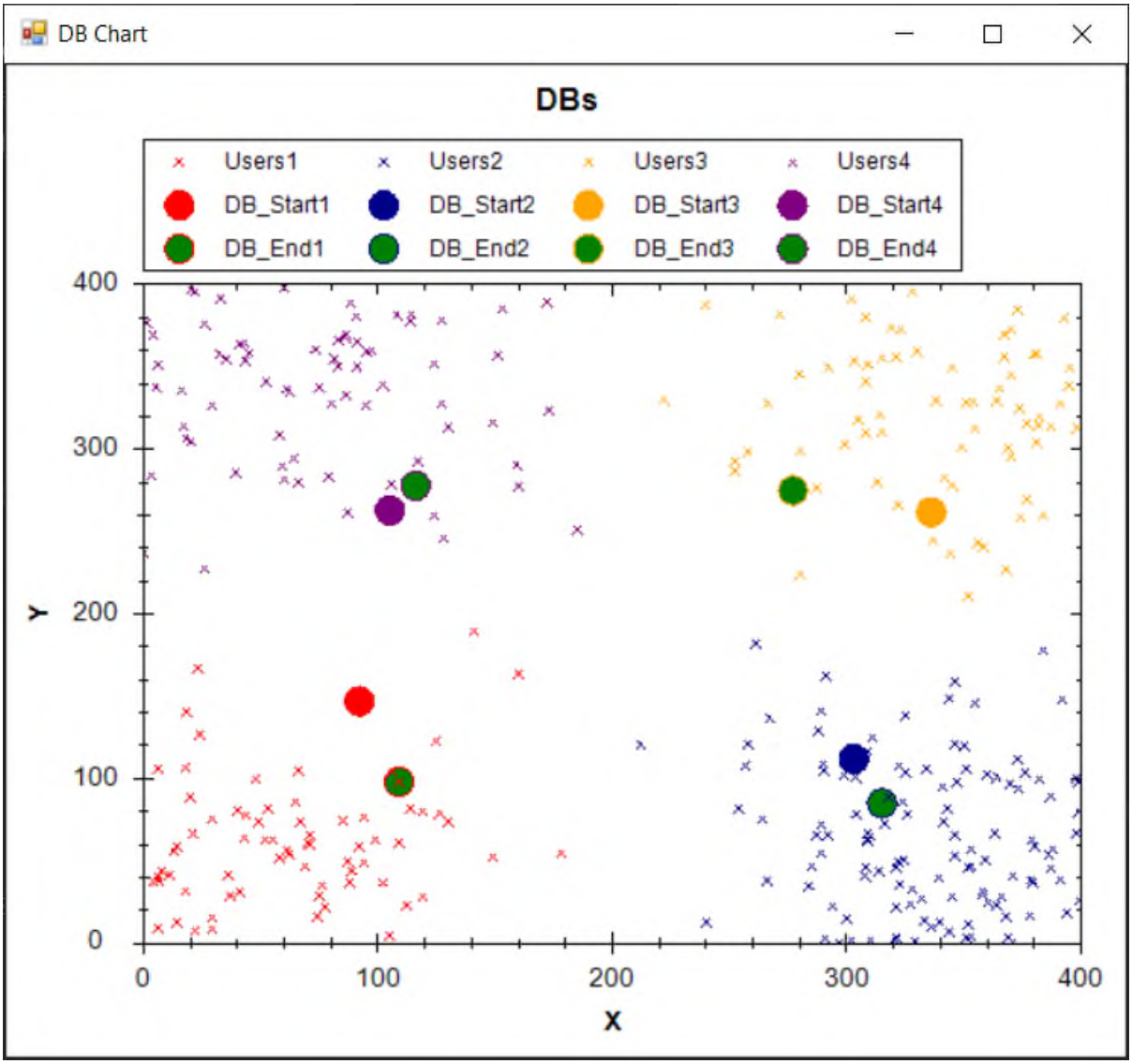

Рисунок 4.5 – Результати експерименту 3, частина 1

Тепер, не змінюючи спосіб генерування користувачів, зменшимо коефіцієнт розташування інформаційних ресурсів один до одного до 3, і запустимо застосунок ще раз.

Результати проведення другої частини експерименту 3 надано на рисунку 4.6.

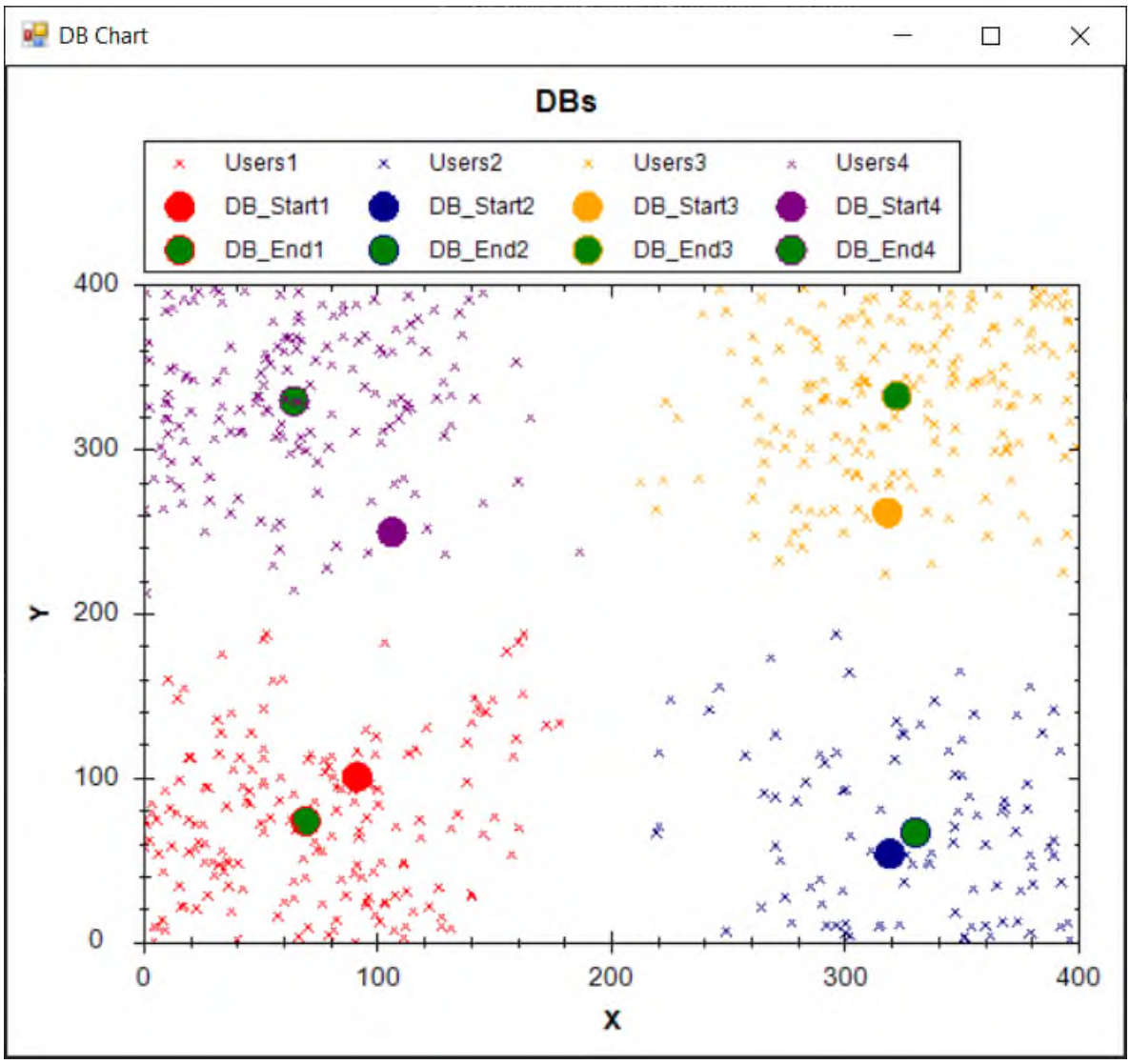

Рисунок 4.6 – Результати експерименту 3, частина 2

4.4 Аналіз результатів

4.4.1 Вагові коефіцієнти

Для розуміння аналізу наданих експериментів потрібно визначити роль вагових коефіцієнтів. Так ваговий коефіцієнт користувача kf_k представляє собою узагальнену змінну, яка позначає кількість та складність запитів, час їх обробки, пропускну здатність каналів зв'язку між користувачем та локальною базою, та будь-які інші змінні, які можуть вплинути на навантаження, яке створює конкретний користувач на локальну базу, до якої він відправляє

запит. Ваговий коефіцієнт локальної бази даних kf_d позначає те саме але у відношенні однієї локальної БД до іншої, таким чином позначаючи, наскільки важлива, чи навпаки, неважлива віддаленість локальних баз даних одна від одної.

Тепер можна перейти до аналізу отриманих у експериментальній частині результатів.

4.4.2 Аналіз експериментів

4.4.2.1 Аналіз експерименту 1

У експерименті 1, при рівномірному розподілі на координатній сітці великої кількості користувачів і без застосування коефіцієнтів kf_k. та kf_d.

Після запуску застосунку, на результатах видно, що фінальна точка оптимізації розміщення локальної бази даних максимально приближена до центру відповідного сектора, у якому розміщено інформаційний ресурс та користувачів. Така велика кількість користувачів з рівномірним їх розподілом за координатами дещо нівелює роль оптимізації загалом, але даний варіант необхідно розглянути для того щоб зрозуміти загальний принцип покрокової оптимізації розташування локальних баз даних. У нашому випадку метод покрокової оптимізації шукає такий варіант розміщення локальних баз даних, при якому середня відстань від локальної БД до користувача буде найменшим можливим значенням. Таким чином при поточних змінних, при рівномірному розподілі великої кількості користувачів ми отримали розміщення локальних баз даних максимально наближеним до центрів відповідних секторів.

4.4.2.2 Аналіз експерименту 2

Другий експеримент проводився за умови невеликої кількості користувачів кожної локальної БД, та із застосуванням випадкових вагових

коефіцієнтів користувача kf_k у діапазоні від 1 до 2, та зі зміною коефіцієнту kf_d на різних етапах експерименту.

Перша частина експерименту після двох прогонів показала, що через великий коефіцієнт розташування локальних баз даних kf_d, точки розташування мало реагують на розміщення клієнтів, через це фінальні оптимізовані точки локальних БД зміщені до центру системи координат.

У другій частині експерименту коефіцієнт розташування локальних баз даних kf_d було встановлено в 0, і тоді оптимізовані координати розміщення локальних БД наближаються до скупчень точок користувачів у більшій мірі відображають оптимізацію розміщення інформаційних ресурсів відносно користувачів, і при випадковому розміщенні невеликої кількості користувачів дана оптимізація має найбільший ефект при досить невеликих витратах на проведення самої операції оптимізації.

4.4.2.3 Аналіз експерименту 3

У третьому та останньому експерименті було змінено спосіб розташування клієнтів локальних баз даних, таким чином, що більша частина з множини клієнтів була розташована ближче до дальнього від центру системи координат кутку своєї секції, та було встановлено кількість згенерованих користувачів у діапазоні від 50 до 200.

З проведенням 2ух частин даного експерименту зі зміною вагового коефіцієнту, були здобуті результати схожі до тих, що вже були отримані у експерименті 2, але з більш наглядною репрезентацією. Так у частині 1 експерименту 3, було встановлено дуже великий коефіцієнт $kf_d = 30$. При такому коефіцієнті, навіть попри розміщення більшості користувачів локальної БД далеко від центру системи координат, покоординатна оптимізація не дає змінюваній локальній БД занадто віддалитися від інших локальних баз даних, що підтверджує результати отримані у експерименті 2.

У другій частині експерименту, при зменшенні коефіцієнту kf_d до 3,

було отримано прямо протилежні, але очікувані результати. Так, не стримані ваговим коефіцієнтом розташування локальних баз даних, фінальні оптимізовані точки розташування змістилися ближче до кутів відповідних секцій, що підтверджує вплив вагового коефіцієнту kf_d на напрямок оптимізації при даному методі.

4.4.3 Підсумки аналізу результатів

Після проведення усіх експериментів та аналізу їх результатів, можна зробити висновок, про ефективність використання методу покрокової оптимізації для оптимізації початкового розміщення інформаційних ресурсів на вузлах комп'ютерної мережі, а використання ваговиї коефіцієнтів kf_u та kf_d, які умовно репрезентують такі аспекти проектування розподіленої бази даниї як кількість та складність запитів, які клієнт/локальна БД до вузла, час обробки запитів, пропускну здатність каналів зв'язку між користувачем та локальною базою даних, або між однією локальною БД та іншою, тощо, дають змогу налаштувати напрямок оптимізації відносно розташування клієнтів локальних баз даних, або відносно інших локальних БД.

ВИСНОВКИ

В рамках кваліфікаційної робот було досліджено предметну область баз даних у інформаційних системах проєктування, приділено час розподіленим базам даних, так як вони є частиною дослідження. Також розглянуто базові питання масштабування та фрагментування баз даних у рамках інформаційних технологій проєктування. Після чого були обрані та описані існуючі методи масштабування баз даних, та методи фрагментування розподілених баз даних. Після чого було проведено аналіз отриманих даних та поставлено задачу дослідження.

У ході аналізу, було також детально розглянуті методи масштабування та фрагментування розподілених баз даних, які можуть бути застосовані для вирішення поставленої задачі, наведено їх математичний опис і схеми алгоритму. Таким чином отримані здобутки будуть використані при майбутньому прикладному вирішенні поставленої задачі та для подальшого аналізу результатів і формування висновку з дослідження.

В результаті роботи було розглянуто питання застосування баз даних у сфері інформаційних технологій проєктування, описано існуючи методи масштабування та фрагментації баз даних, поставлено задачу з масштабування розподілених баз даних та розглянуто та проаналізовано можливі методи та алгоритми для вирішення поставленої задачі, та наведено узагальнену задачу оптимізації структури та розташування вузлів розподіленої бази даних в інформаційних технологіях проєктування.

У ході проведення досліджень при виконанні даної кваліфікаційної роботи були опубліковані тези до наукових конференцій «Інформаційні системи та технології» [1] та «Комп’ютерно-інтегровані технології автоматизації технологічних процесів на транспорті та у виробництві» [2].

Напрямом подальших досліджень може бути розробка ефективних методів оптимізації структур РБД інформаційних технологіях проєктування за

умов неповної визначеності цілей та вхідних даних.

Напрямком подальшої розробки може бути покращення існуючого застосунку шляхом додавання нових та корегування існуючих функцій з оптимізації структури, розташування, та інших аспектів проектування розподілених баз даних.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Таран П., Безкоровайний В. Оптимізація структур розподілених баз даних з використанням масштабування // Інформаційні системи та технології: матеріали 11-ї Міжнародної наук.-техн. конф. Ч. 2 (м. Харків, 22-25 листоп. 2022 р.), Х.: ХНУРЕ, 2022. С. 5–6. URL: https://istconf.sedep.online/archive/ist_2022_part_2.pdf (дата звернення 03.12.2022).

2. Таран П. А., Безкоровайний В. В. Масштабування баз даних для інформаційних технологій проектування // Комп'ютерно-інтегровані технології автоматизації технологічних процесів на транспорті та у виробництві. Матеріали всеукраїнської наук.-практ. конф. здобув. вищої освіти і молод. учених (м. Харків, 23 листоп. 2022 р.), Харків: ХНАДУ. 2022. С. 238–241. URL: https://mf.khadi.kharkov.ua/fileadmin/user_upload/Матеріали_КІТ-2022.pdf (дата звернення 10.12.2022).

3. Проєктування. URL: https://uk.wikipedia.org/wiki/Проєктування (дата звернення 07.10.2022).

4. Beskorovainyi V., Imanhulova Z. Technology of large-scale objects system optimization // ECONTECHMOD. 2017. Vol. 06. №4. P. 3–8. URL: https://bibliotekanauki.pl/articles/410841 (дата звернення: 15.10.2022).

5. Бази даних та інформаційні системи. URL: https://wiki.cuspu.edu.ua/index.php/Бази_даних_та_інформаційні_системи (дата звернення: 20.10.2022).

6. Бази даних. Інформаційні системи. URL: https://ppt-online.org/127672 (дата звернення: 21.10.2022).

7. Горизонтальне і вертикальне масштабування веб додатків. URL: https://server-gu.ru/scaling-out-vs-scaling-up/ (дата звернення: 21.10.2022).

8. Scaling Horizontally vs. Scaling Vertically. URL: https://www.section.io/blog/scaling-horizontally-vs-vertically (дата звернення:

20.10.2022).

9. Тамер Есу М., Вальдуриес П. Принципи організації розподілених баз даних / пер. з англ. А. Слінкіна. Москва: ДМК Пресс, 2021. 672 с.

10. Розподілена база даних. URL: https://uk.wikipedia.org/wiki/Розподілена_база_даних (дата звернення: 05.11.2022).

11. Вертикальне та горизонтальне масштабування. Вступні відомості про масштабованість баз даних під час хмарних обчислень URL: https://azure.microsoft.com/ru-ru/resources/cloud-computing-dictionary/scaling-out-vs-scaling-up/#scale-up-vertically (дата звернення: 07.11.2022).

12. Горизонтальне масштабування. Що, навіщо, коли та як?. URL: Доступно: https://habr.com/ru/company/oleg-bunin/blog/319526 (дата звернення: 10.11.2022).

13. Beskorovainyi V., Kolesnyk L. Interval model of multi-criterion task of reengineering physical structures of distributed databases // Intelligent information systems for decision support in project and program management: Collective monograph edited by I. Linde. Riga: ISMA, 2021. P. 7–14.

14. Арсеньев В.П. Интегрированные распределенные базы данных. СПб.: Изд.-полигр. центр СПбГЭТУ (ЛЭТИ), 2004. 498 с.

15. Петров Э.Г., Писклакова В.П., Бескоровайный В.В. Территориально-распределенные системы обслуживания. К.: Техника, 1992. 208 с.

16. Бескоровайный В.В., Ульянова О.С. Математические модели многокритериального синтеза физических структур распределенных баз данных // Восточно-европейский журнал передовых технологий. 2010. № 4/4 (46). С. 44-48.

17. Калмыков С.А., Шокин Ю.И., Юлдашев З. Х. Методы интервального анализа. Новосибирск: Наука, 1986. 222с.

18. Дивак М. П. Задачі математичного моделювання статичних систем з інтервальними даними. Тернопіль: Видавництво ТНЕУ "Економічна думка",

2011. 216 с.

19. Яковлев А.Г. Машинная арифметика мультиинтервалов // Вопросы Кибернетики (Научный Совет по компл. проблеме "Кибернетика" АН СССР). 1986. Вып. 125. С. 66-81. 8. Extended interval arithmetics: new results and applications / N. S. Dimitrova, S. M. Markov, E. D. Popova, L. Atanassova, J. Herzberger, eds. // Computer Arithmetic and Enclosure Methods. Amsterdam: Eisevier, 1992. P. 225-232.

20. Овезгельдыев А.О., Петров Э.Г., Петров К.Э. Синтез и идентификация моделей многофакторного оценивания и оптимизации: монография. Наукова думка, 2002. 161 с.

21. Raskin L.G., Seraja O.V. Fuzzy Mathematics. Fundamentals of the theory. Kharkiv: Parus, 2008. P. 352.

22. Бескоровайный В.В., Соболева Е.В. Идентификация частной полезности многофакторных альтернатив с помощью S-образных функций // Бионика интеллекта. 2010. №1. С.50-54.

23. Beskorovainyi V., Berezovskyi H. Identification of preferences in decision support systems // Econtechmod. An International Quarterly Journal. 2017. Vol. 06, No. 4, P. 15-20.

24. Интроспективный анализ. Методы и средства экспертного оценивания / В.В. Крючковский, Э.Г. Петров, Н.А. Соколова, В.Е. Ходаков. Херсон: Гринь ДС, 2011. 284 с.

25. Бескоровайный В.В., Петров Э.Г., Трофименко И.В. Метод решения задачи компараторной идентификации моделей многофакторного оценивания // Бионика интеллекта. 2006. № 65. С. 3-7.

26. Beskorovainyi V., Petryshyn L., Shevchenko O. Specific subset effective option in technology design decisions // Applied Aspects of Information Technology. 2020. Vol. 3. No.1. P. 443–455. URL: https://aait.op.edu.ua/?fetch=articles&with=info&id=40. (дата звернення: 10.12.2022).

27. Beskorovainyi V. Combined method of ranking options in project

decision support systems // Innovative Technologies and Scientific Solutions for Industries. 2020. No 4 (14). P. 13–20. URL: http://journals.uran.ua/itssi/article/view/ITSSI.2020.14.013 (дата звернення 10.12.2022).

28. Розміщення даних у розподілених БД. URL: https://studref.com/521076/informatika/razmeschenie_dannyh_raspredelennyh (дата звернення: 10.11.2022).

29. Проектування розподілених баз даних. URL: https://helpiks.org/7-49596.html (дата звернення: 11.11.2022).

30. Таран П.А. Комп'ютерна гра-стратегія «Legendary Battles» із використанням технології Unity із використанням технології // Диплом бакалавра, спеціальність «Комп'ютерна інженерія». ХНУРЕ, 2021.

31. Desktop Guide (Windows Forms .NET). URL: https://learn.microsoft.com/en-us/dotnet/desktop/winforms/overview/?view=netdesktop-6.0 (дата звернення: 15.12.2022).