НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
ІНСТИТУТ ПРИКЛАДНОГО СИСТЕМНОГО АНАЛІЗУ
КАФЕДРА МАТЕМАТИЧНИХ МЕТОДІВ СИСТЕМНОГО АНАЛІЗУ

На правах рукопису
УДК __004.772_____

До захисту допущено
В. о. завідувача кафедри ММСА
О.Л.Тимощук
«___» _____ 2018 р.

Магістерська дисертація
на здобуття ступеня магістра за спеціальністю 124 Системний аналіз
на тему: «Методи та засоби ефективного розподілу даних у гетерогенно розподілених базах даних»

Виконав:
студент ІІ курсу, групи КА-71 мп
Коваленко Денис Юрійович _____

Керівник:
професор, д.т.н. Мухін В.Є. ._____

Рецензент:
д.т.н. Корнага Я. І. ._____

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

Київ

2018

РЕФЕРАТ

Магістерська дисертація: 95 с., 18 рис., 6 табл., 2 додатки, 25 джерел.

В роботі досліджується задача знаходження оптимальної структури мережі бази даних для мінімізації часу відповіді мережі на запити, а саме наступні типи мереж: ієрархічна, кільцева, решітчаста. Побудовано математичні моделі для кожної з них. Створено програмний продукт на основі розробених математичних моделей для порівняння та дитального аналізу всіх типів мереж.

Об'єктом дослідження є гетерогенні розподілені бази данних.

Предметом дослідження є методи оптимального розподілення даних в розподілених базах даних.

Мета роботи – розробити моделі та механізми, які забезпечать оптимальний розподіл даних в гетерогених розподілених баз даних з точки зору швидкості відповіді на запити.

Методи дослідження: для побудови та аналізу математичних моделей використані методи математичний аналізу, методи оптимізації та методи аналізу експериментальних даних.

Робота складається з п'яти розділів. У першому з них наведена структурна інформація про існуючі моделі розподілення даних в РБД, та проаналізовані недоліки існуючих моделей. У другому описаний алгоритм створення математичних моделей. В третьому та четвертому розділ присвячений опису створення програмного продукту порівняння досліджуваних мереж, а в п'ятому проаналізований стартап проект по досліджуваній проблематиці.

Галузь використання: Системи використвоуючі розподілені бази даних як основний метод зберігання інформації.

ГЕТЕРОГЕННІ БАЗИ ДАНИХ, ОПТИМАЛЬНИЙ РОЗПОДІЛ, ІНФОРМАЦІЇ, ІНФОРМАЦІЙНІ МЕРЕЖІ, МАТЕМАТИЧНІ МОДЕЛІ МЕРЕЖ

ABSTRACT

The topic: Methods and tools for efficient data sharing in the databases

Master's thesis: 95 pp., 18 img., 6 tab., 2 applications, 23 sources.

The task of finding the optimal database structure network structure to minimize the response time of the network to inquiries is investigated, in particular the following types of networks: hierarchical, ring, lattice. Mathematical models for each of them are constructed. A software product was developed based on the developed mathematical models for comparison and analysis of all types of networks.

The object of the study is the heterogeneous distribution of the database.

The subject of the research is the methods of optimal data distribution in distributed databases.

The purpose of the work is to develop optimal models of data distribution in heterogeneously distributed databases.

The research method is an analysis of information distribution methods in databases, mathematical analysis and optimization methods.

The work consists of five sections. The first of them presents the structural information about existing models of data distribution in the RBD, and analyzes the disadvantages of existing models. The second describes the algorithm for creating mathematical models.

Field of use: Systems using distributed databases as the main method of storing information.

HETEROGENIC BASES OF DATA, OPTIMAL DISTRIBUTION, INFORMATION, INFORMATION NETWORKS, MATHEMATIC NETWORK MODELS

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

БД – база даних

ЕОМ – електронно обчислюючи машина

МО – методи оптимізації

ММ – математична модель

СУБД – система управління базами даних;

СУРБД - система управління розподіленими базами даних;

РБД – розподілена база даних

ВСТУП

Основною передумовою розробки систем, що використовують бази даних, є прагнення об’єднати всі дані, які обробляються на підприємстві, в єдине ціле та забезпечити контрольований доступ до них. Сучасні потужні підприємства часто мають велику кількість підрозділів, які можуть фізично розташовуватись за сотні та навіть тисячі кілометрів один від одного. Кожний з цих підрозділів має свою локальну базу даних. Якщо ці бази мають різні архітектури та використовують різні протоколи зв’язку, то їх розглядають як інформаційні острови, важкодоступні для інших. В такому випадку виникає необхідність об’єднати розрізнені бази даних в одне логічне ціле, тобто створити розподілену базу даних.

Розподілена база даних – це сукупність структурно та логічно зв’язаних локальних баз даних або частин однієї бази даних, які розподілені між декількома територіально-рознесеними комп’ютерами-вузлами і мають відповідними засоби управління базами як єдиним цілим. Таким чином, розподілена база даних реалізується на різних просторово розосереджених обчислювальних засобах, разом з організаційними, технічними і програмними засобами її створення і ведення.

Однією з **актуальних проблем** при створенні таких розподілених баз даних є необхідність зниження часу відповіді на запити щодо обробки даних з врахуванням того факту, що дані є територіально розподілені на гетерогенних ресурсах. Для вирішення цієї задачі застосовуються механізми розподілу ресурсів гетерогенних комп’ютерних систем із адаптацією їх до специфики обробки даних в базах даних.

Для вирішення поставленої задачі в дипломній роботі поставлені та виконані **наступні завдання**: детально розглянути методи розподілення даних в великих базах даних, побудувати математичні моделі аналізу швидкості відповіді на запити для різних типів комп’ютерних мереж,

провести аналіз експериментальних даних, отриманих за допомогою створеного прикладного програмного забезпечення, визначити оптимальні конфігурації ресурсів для зберігання даних в розподілених базах даних.

Об'єктом дослідження є гетерогенні розподілені бази данних.

Предметом дослідження є методи оптимального розподілення даних в розподілених базах даних.

Мета роботи – розробити моделі та механізми, які забезпечать оптимальний розподіл даних в гетерогених розподілених баз даних з точки зору швидкості відповіді на запити.

Методи дослідження: для побудови та аналізу математичних моделей використані методи математичний аналізу, методи оптимізації та методи аналізу експериментальних даних.

Постановка задачі: визначити найбільш оптимальну (в плані швидкості відповіді на запит) конфігурацію ресурсів гетерогенних розподілених баз даних та визначити найменшу можливу кількість серверів, які мають бути використані для підтримки заданого об'єму інформації в базі даних.

РОЗДІЛ 1 КЛАСИФІКАЦІЯ ІСНУЮЧИХ МЕТОДІВ РОЗПОДІЛУ ДАНИХ

1.1 Загальна інформація про розподілені бази даних

Розподілена база даних (англ. distributed database, DDB) — сукупність логічно взаємопов'язаних баз даних, розподілених у комп'ютерній мережі. Логічний зв'язок баз даних в розподіленій базі даних забезпечує система управління розподіленою базою даних, яка дозволяє управляти розподіленою базою даних таким чином, щоб створювати у користувачів ілюзію цілісної бази даних.

Система управління розподіленою базою даних складається з (можливо, порожнього) набору вузлів прийому запитів і набору вузлів збереження даних. Вузли прийому запитів реалізують прозорий інтерфейс доступу до даних, що зберігаються в вузлах збереження даних, та приховують фрагментацію даних між вузлами. Кожен з вузлів може бути представлений незалежним комп'ютером в комп'ютерній мережі з власною (можливо відмінною від інших вузлів) операційною системою.

Система управління розподіленою базою даних є однорідною (гомогенною), якщо на кожному з вузлів збереження даних застосовуються однакові СКБД, в іншому випадку система управління розподіленою базою даних є неоднорідною (гетерогенною). В наслідок застосування стандартизованих механізмів доступу до баз даних відмінності між однорідними та неоднорідними системами нівелюються і не є критичними.

В архітектури програмно-технічного комплексу розподілених СКБД є три основні характеристики:

- Неоднорідність. Важливою характеристикою є міра однорідності програмно-технічних засобів СКБД.

- Розподіленість. Спосіб розподілу компонентів системи баз даних за комп'ютерами мережі визначається тим, чи є в мережі єдиний комп'ютер з повноваженнями розподіленої СКБД, або ж їх кілька.
- Автономність. Характеризує, наскільки самостійно компоненти СКБД можуть виконувати свої функції.

Виділяють такі основні різновиди архітектури програмно-технічних засобів розподіленої СКБД, як:

- Архітектура однорангової мережі. Якщо всі комп'ютери мережі є серверами та на кожному комп'ютері розміщено розподілену СКБД і базу даних та з кожного комп'ютера можна надіслати до іншого запит на отримання необхідних даних. То така архітектура буде архітектурою однорангової мережі.
- Архітектура з багатьма незалежними серверами. Якщо існують багато серверів, що мають доступ до своїх локальних баз даних. У такому випадку на комп'ютерах клієнтів має зберігатись інформація стосовно того, які дані на яких серверах розташовані, а також має бути розміщене програмне забезпечення, що дає змогу розкласти запити для їхнього виконання на різних серверах і потім об'єднати результати.
- Архітектура із взаємодіючими серверами.

У цьому випадку передбачається, що кожен сервер містить повну інформацію стосовно того, які дані на яких серверах зберігаються, а також здатний обробляти розподілені запити. У разі потреби один сервер може звернутися до іншого для одержання необхідних даних. Кожен клієнт має свій сервер, до якого звертається для виконання операцій над розподіленою базою даних.

При даному виді архітектури передбачено наявність єдиного комп'ютера-сервера і багатьох комп'ютерів-клієнтів, що взаємодіють між собою через канали зв'язку. На сервері розташована СКБД та інтегрована (централізована) база даних. Ніякого розподілу баз даних за вузлами мережі немає. На

клієнтських комп'ютерах виконуються додатки, які працюють із серверною базою даних, а також розміщене програмне забезпечення для зв'язку з віддаленою СКБД. Як клієнти, так і сервер оснащені комунікаційним програмним забезпеченням.

Розподіл даних між вузлами збереження даних забезпечується на основі механізмів фрагментації та реплікації, і досягається шляхом вертикального (на окремі поля записів бази даних) або горизонтального (на окремі записи бази даних) поділу даних. Наприклад, при вертикальному поділі даних інформація про покупців може зберігатись на одному вузлі збереження даних, про їх купівлі — на другому, про товари — на третьому тощо; при горизонтальному поділі даних інформація про покупців, купівлі та товари зберігається на одному вузлі, а поділ між вузлами здійснюється за країнами покупців (для кожної країни — окремий вузол збереження даних), або за типами товарів тощо. Фрагментація здійснюється за принципами, що дозволяють наблизити дані до місць їх найбільш інтенсивного використання для зменшення витрат на пересилання даних.

Наближення даних до місць їх найбільш інтенсивного використання також забезпечується реплікацією даних.

Доступ до даних в розподіленій базі даних звичайно забезпечується в трирівневій моделі: клієнт — сервер додатків — вузли збереження даних. При інтернет-доступі до даних роль сервера додатків відіграє веб-сервер або спеціальний додаток на боці клієнта.

Для вирішення спеціальних задач обслуговування даних можливий локальний доступ до окремих вузлів збереження даних, при цьому недоступні можливості доступу, що базуються на прозорості інтерфейсів доступу до даних.

1.2 Розподілене зберігання даних

Основним механізмом розподілу даних є фрагментація. Суть фрагментації полягає в тому, щоб поділити логічну базу даних на фрагменти з метою зберігання кожного фрагмента на певному вузлі мережі. Одиницями фрагментації можуть бути відношення та складені відношення. У випадку, коли одиницею фрагментації є відношення, вирішується проблема, яке відношення в якій базі даних має зберігатися. За іншого підходу допускається, що будь-яке відношення може бути зображене у вигляді сукупності фрагментів, що розподіляються за різними базами даних.

Фрагментацію, що здійснюється розподілом відношень за базами даних, теоретично здійснити нескладно, тому розглянемо проблему фрагментації власне відношень.

Фрагментація відношень. Завдання фрагментації відношень формулюється в такий спосіб. Нехай задане відношення R. Його потрібно зобразити у вигляді сукупності відношень R1, …, Rn так, щоб ця сукупність відповідала критеріям ефективності (за часом доступу, пам'яттю, завантаженістю комп'ютерів тощо).

Фрагментація є коректною, якщо вона повна, не містить перетинів і може бути реконструйована. Пояснимо ці терміни.

Декомпозиція відношення R на фрагменти R1, R2, … Rn є повною тоді й лише тоді, коли кожен елемент даних з R належить якомусь із відношень Ri.

Декомпозиція відношення R на фрагменти R1, R2, … Rn може бути реконструйована, якщо існує такий реляційний вираз φ(R1, R2, … Rn), що R=φ(R1, R2, … Rn).

Декомпозиція відношення R на фрагменти R1, R2, … Rn не містить перетинів, якщо будь-який елемент даних з R міститься не більш ніж в одному фрагменті.

Є три типи фрагментації відношень:

1) Горизонтальна фрагментація

Горизонтальна фрагментація полягає в розподілі кортежів відношення за фрагментами. Формально горизонтальну фрагментацію можна визначити в такий спосіб. Нехай задане відношення R і на ньому визначений предикат Fi. Тоді горизонтальний фрагмент Ri відношення визначається так:

$$R_i = \sigma_{F_i}(R)$$

Тобто горизонтальний фрагмент Ri — це множина кортежів R, що задовольняють умову Fi.

2) Вертикальна фрагментація

Суть вертикальної фрагментації полягає в тому, що відношення поділяється на дві чи більше проекцій, тобто схема відношення поділяється на певну множину підсхем.

Для відновлення вихідного відношення з фрагментів необхідно, щоб усі підсхеми містили первинний ключ. Існує інший підхід, коли під час поділу схеми до кожного фрагмента автоматично додається поле з ідентифікатором кортежу. Значення цього ідентифікатора зазвичай встановлюються системою автоматично.

3) Змішана фрагментація. Змішана фрагментація передбачає послідовне застосування вертикальної і горизонтальної фрагментацій.

Після отримання усіх необхідних фрагментів відношень постає проблема розподілу цих фрагментів за вузлами мережі. Єдиних рекомендацій стосовно того, як це робити, немає. Потрібно знайти оптимальний розподіл фрагментів F за вузлами мережі S за умови, що відомий розподіл додатків Q за вузлами мережі.

Реплікація є механізмом розподілу даних за вузлами, що в свою чергу дозволяє зберігати копії тих самих даних на різних вузлах мережі для прискорення пошуку і підвищення стійкості до відмов. Відношення або фрагмент є реплікованим, у випадку якщо його копії(або репліки) зберігаються на двох або більше вузлах . За повної реплікації відношення його копії зберігаються на всіх вузлах мережі. Допускається ситуація, коли вся база даних зберігається на всіх вузлах мережі — це називається повною реплікацією бази даних. Виділяють 3 основних механізми реплікації:

- видавець — сервер, що надає розміщені на ньому дані для копіювання на інші сервери;
- дистриб'ютор — сервер, що підтримує розподілену базу даних;
- передплатник — сервер, що отримує копії даних, надані видавцем.

Реплікація за запитом полягає в тому, що передплатник періодично звертається до дистриб'ютора із запитом про зміни, що відбулися з моменту останнього з'єднання.

Примусова реплікація, с свою чергу, коли дистриб'ютор сам встановлює з'єднання з передплатником і пересилає йому необхідні дані.

Залежно від методу реплікації, передплатники можуть, або не можуть вносити зміни в репліковані дані. У найпростішому випадку змінювати дані може тільки видавець, у складніших моделях реплікації — передплатники і видавці. Змінені дані, отримані від усіх передплатників, синхронізуються і поєднуються з даними видавця, а потім розсилаються передплатникам.

Топологія реплікацій описує характер взаємозв'язків між учасниками реплікації:

реплікація «один-до-багатьох» передбачає наявність одного видавця і кількох передплатників;

реплікація «багато-до-одного» має місце, коли дані від кількох видавців пересилаються одному передплатнику;

реплікація «багато-до-багатьох» означає, що дані від кількох видавців пересилаються кільком передплатникам.

1.3 Обробка транзакцій до розподілених даних

Транзакція — набір команд, що виконується як єдине ціле. У транзакції або всі команди будуть виконані, або жодна з них не виконається. Якщо хоча б одна з команд транзакції не може бути виконана, здійснюється відкочування (відновлюється стан системи, в якому вона перебувала до початку виконання транзакції).

Транзакції мають задовольняти вимоги ACID (Atomicity, Consistency, Isolation, Durability — атомарність, несуперечність, ізольованість, довговічність), що гарантують правильність і надійність роботи системи.

Атомарність передбачає:

- виконуються всі операції транзакції або жодна з них не виконується;
- якщо виконання транзакції було перерване, то всі зроблені транзакцією зміни мають бути скасовані.

Несуперечність означає, що транзакція, яка працює з несуперечною базою даних, після завершення роботи залишає її також у несуперечному стані. Транзакція не повинна порушувати цілісності бази даних

Для вирішення проблем одночасного доступу інститут ANSI розробив спеціальний стандарт, який визначає чотири рівні блокування (кожний вищий рівень передбачає виконання умов усіх нижчих рівнів)

Рівень 0. Заборона «забруднення» даних.

Рівень 1. Заборона некоректного зчитування.

Рівень 2. Заборона неповторюваного зчитування.

Рівень 3. Заборона «фантомів».

Властивість ізольованості означає, що на роботу транзакції не мають впливати інші транзакції. Транзакція «бачить» дані в тому стані, в якому вони перебували до початку роботи іншої транзакції, або в тому стані, в якому вони перебувають після її завершення.

Після того, як було підтверджено успішне завершення роботи транзакції (Commit), система має гарантувати, що її результати не будуть втрачені, незважаючи на можливі перебої. Це й називається довговічністю.

Властивості розподілених баз даних

12 властивостей розподілених баз даних, були сформульовані Крістофером Дейтом, одним з найбільших діячів в галузі баз даних. А саме:

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 СУБД мережевих протоколів не має бути обмеженням системи, заснованої на розподіленій БД

Незалежність від баз даних — в розподіленій системі можуть працювати СУБД різних виробників, і можливі операції пошуку і оновлення в базах даних різних моделей і форматів.

1.4 Сутність розподілених баз даних

Завдяки досягнутим успіхам в області обчислювальної техніки і засобів телекомунікацій безліч територіально розосереджених машин і терміналів об'єднуються в глобальні та локальні мережі. Це створює сприятливі умови для розширення області використання інформаційних систем.

Сучасні інформаційні системи будуються на відомій концепції баз даних, сутність якої полягає в інтеграції даних і централізації управління ними для забезпечення багатоаспектного використання. Цим забезпечується необхідний

рівень незалежності між технічними, програмними та інформаційними засобами систем, що дозволяє адаптувати останні до поточних вимог користувачів, а також удосконалювати в процесі експлуатації.

Більшість спочатку створювалися баз даних відносилося до класу централізованих, тобто зберігають інформацію в одній машині. Багато об'єктів (організації, підприємства) мали кілька таких баз, розміщених в різних комп'ютерах в силу наступних обставин:

- об'єкт за своєю природою є територіально розосередженим;
- інформаційні потреби цього об'єкта настільки великі, що не могли покриватися однією централізованою базою даних;
- різні бази даних створювалися в різний час в процесі еволюційного розвитку об'єкта.

Впровадження в практику управління персональних комп'ютерів ще більш ускладнило становище, так як суттєво збільшилася кількість відносно невеликих баз даних, що створюються в інтересах окремих посадових осіб або їх нечисленних груп.

У той же час очевидна потреба в швидкому і надійному доступі різних користувачів до інформації, що міститься в декількох локальних базах даних. Об'єднання розосереджених обчислювальних ресурсів об'єкта в глобальну або локальну мережу не дозволяє існуючу проблему. Навіть при наявності розвиненої комунікаційної системи необхідно вирішити п'ять завдань, пов'язаних з наданням даних:

- визначити комп'ютери, що містять запитувані дані;
- сформувати кілька запитів, які будуть виконуватися на різних машинах;
- скопіювати або передати результати в одну машину;
- об'єднати результати;
- виділити з об'єднаного результату відповідь на вихідний запит.

Рішення перерахованих завдань може зажадати значних витрат часу, бути досить рутинним і схильним до помилок. Тому потрібна реалізація принципово нових підходів до організації інформаційного процесу в комп'ютерних мережах. Якісний стрибок в цій області можливий завдяки створенню гнучких, високопродуктивних і економічних розподілених інформаційних систем на основі концепції баз даних.

Поширення названої концепції на новий рівень дозволяє визначити розподілену інформаційну систему як комплекс логічно інтегрованих і територіально розосереджених баз даних, технічних, програмних, мовних і організаційних засобів, призначених для накопичення, ведення і використання інформації. У свою чергу, розподілена база даних (РБД) визначається як інтегрована база даних, фізично розміщується на декількох територіально розподілених комп'ютерах мережі.

Для таких систем характерними є такі функції:

- накопичення, оновлення та зберігання даних в географічно віддалених вузлах мережі;
- логічна інтеграція територіально розподілених даних, процесів обробки, оновлення та пошуку інформації;
- забезпечення автоматичного взаємодії між локальними базами даних в процесі виконання запитів і вирішення завдань користувачів.

Як користувачів подібних інформаційних систем можуть виступати посадові особи організацій, підприємств, прикладні та системні програмісти, адміністрація бази даних. Адміністрування забезпечує створення розподіленої і локальних баз даних і їх реорганізацію, управління функціонуванням інформаційної системи, накопичення і видачу статистичних відомостей про її роботі, поточний аналіз стану. Функції адміністрування виконуються на рівні розподіленої і локальних баз даних. На рівні РБД ці функції полягають у підтримці в актуальному стані всіх видів засобів, що забезпечують інтеграцію

локальних баз даних. На рівні ЛБД вони збігаються з функціями адміністратора централізованої БД.

Управління виконанням основних функцій інформаційної системи здійснює комплекс програмних засобів, який включає систему управління розподіленими базами даних (СУРБД) в якості основного компонента, а також мережеву операційну систему, систему розмежування доступу та ін.

Під СУРБД розуміють систему, що реалізовує принцип логічної інтеграції даних, фізично розподілених між різними взаємопов'язаними комп'ютерами. До основних функцій СУРБД можна віднести аналіз і розподіл запитів, а також управління їх проходженням.

Матеріальну основу розподілених банків даних складають технічні засоби комп'ютерних мереж. Їх вдосконалення ведеться в двох напрямках. Перше є традиційним і полягає в розробці високопродуктивних і надійних мереж. Друге полягає в спеціалізації технічних засобів і пов'язане з розробкою ЕОМ, що володіють великою оперативною пам'яттю, а також машин баз даних. Збільшення обсягу оперативної пам'яті дозволить забезпечити достатню продуктивність всієї системи при вирішенні завдань з великим числом невпорядкованих звернень до пам'яті за рахунок скорочення операцій з обміну з зовнішніми пристроями, що запам'ятовують. Машини баз даних орієнтовані на реалізацію основних функцій управління БД, що забезпечує більш ефективне функціонування інформаційної системи в цілому. Вони можуть виконуватися в наступних модифікаціях: допоміжної ЕОМ, пакета прикладних програм, апаратно реалізованої СУБД.

В цілому необхідно відзначити, що реалізація концепції розподілених баз даних забезпечує незалежність програм по відношенню до розподілу даних в обчислювальній мережі. Це означає, що користувач може мати доступ до будь-яких даних мережі в межах своїх повноважень як до власної локальної базі даних. У цьому контексті значний інтерес представляє розгляд різних способів

розподілу даних по вузлах мережі для забезпечення ефективного доступу до них.

1.5. Стратегії розподілу даних в комп'ютерній мережі

Стратегії розподілу даних по вузлах комп'ютерної мережі можуть класифікуватися за різними ознаками. До основних з них можна віднести наявність дублювання інформації і кількість вузлів, що містять копії даних. Відповідно до зазначених класифікаційними ознаками представляється можливим виділити чотири альтернативних стратегії розподілу:

- централізація (єдина копія бази даних, розташована в одному вузлі);
- розчленування (єдина копія бази даних, непересічні фрагменти якої розподілені по декільком вузлам);
- дублювання (кілька копій бази даних, в кожному вузлі розташовується повна копія всієї бази);
- змішана (кілька копій бази даних, в кожному вузлі розташовується довільний фрагмент бази).

Ранжування стратегій по перевагу їх використання в загальному випадку не представляється можливим. Визначення ситуацій, в яких та чи інша стратегія є найбільш придатною, доцільно після виявлення переваг і недоліків кожної з них.

Стратегія централізації. Названа стратегія характеризує граничний випадок розподілу даних. Строго кажучи, база даних, яка за цієї стратегії, не є розподіленою. Проте, системи, що реалізують стратегію централізації, можуть

бути винесені в клас систем розподіленої обробки, в яких розосереджені обчислювальні ресурси і програмне забезпечення. Тому розгляд названої стратегії представляє певний інтерес.

До безперечних достоїнств централізованої бази даних слід віднести її простоту, так як всі операції звернення до даних виконуються під управлінням одного вузла.

У той же час недоліки, властиві даної стратегії, пов'язані саме з локальним розташуванням даних. Природно, що обсяг бази даних обмежений ємністю запам'ятовуючих пристроїв центрального вузла. До того ж запити на операції над даними повинні направлятися тільки в цей вузол з усіма наслідками, що випливають транспортними витратами. Якщо обчислювальний ресурс центрального вузла представлений єдиною ЕОМ, то на паралельну обробку даних накладаються істотні обмеження. Центральний вузол стає вузьким місцем всієї мережі бо, по-перше, швидкість обробки даних обмежується швидкодією його обчислювальних засобів, по-друге, база даних стає недоступною для інших вузлів при появі несправностей в мережі передачі даних і повністю відмовляє при виході з ладу центрального вузла.

Будь-яка з трьох інших стратегій допомагає подолати зазначені недоліки ціною певних витрат.

Стратегія розчленування. При використанні цієї стратегії база поділяється на підмножини даних, звані логічними фрагментами. Кожен такий фрагмент розміщується в окремому вузлі. Число вузлів, що зберігають дані, може бути довільним, в граничному випадку рівним загальної кількості вузлів мережі.

Переваги розчленування даних перед централізованим розміщенням наступні. Розмір бази обмежується об'ємом запам'ятовуючих пристроїв всій мережі (її частини). Знижується чутливість до вузьких місць в мережі передачі даних, оскільки навантаження на неї розподіляється більш рівномірно.

Підвищується доступність даних. Навіть якщо мережа передачі даних повністю або частково виходить з ладу, то доступними для користувачів можуть залишатися фрагменти бази даних в окремих вузлах або вузлах, що залишаються пов'язаними. Це забезпечує реалізацію функцій інформаційної системи, хоча і в скороченому обсязі.

Важливу роль при визначенні потенційної доступності даних в критичних ситуаціях грає ступінь локалізації посилань. Під локалізацією посилань розуміють характер розташування даних щодо запитів користувачів. Найвищий ступінь локалізації посилань існує в разі, якщо дані, розташовані в одному вузлі, запитуються виключно користувачами цього вузла. Навпаки, ступінь локалізації посилань мала, якщо подібне розчленовування бази неможливо. Таким чином, якщо запит користувача може бути виконаний за допомогою локально збережених даних (ступінь локалізації посилань висока), то несправності в інших вузлах або в мережі передачі даних не вплинуть на результат. Якщо ж ступінь локалізації посилань незначна або запити є складними, то в більшості випадків користувачам можуть знадобитися дані від інших вузлів. Недоступність хоча б одного вузла призведе до невиконання запиту користувача. У подібній ситуації доступність даних в розчленованої базі може бути гірше, ніж в централізованій, так як ймовірність того, що щонайменше один вузол мережі буде недоступний, вище ймовірності недоступності єдиного конкретного вузла. Отже, стратегія розчленування забезпечить меншу за часом доступність бази даних, ніж стратегія централізації.

Ступінь локалізації посилань впливає і на тимчасові характеристики розчленованої бази даних. Так, якщо більша частина запитів обслуговується локально, зменшуються витрати часу на передачу даних. З іншого боку, якщо виконання запиту вимагає доступу до багатьох вузлів, середній час затримки може перевищувати аналогічний показник централізованої бази. Проте,

зменшення часу реакції може бути досягнуто, якщо в повній мірі використовується паралелізм в обчислювальної мережі.

В цілому стратегія розчленування найбільш прийнятна у випадках, коли-небудь обсяг зовнішньої пам'яті в кожному вузлі істотно менше розміру бази даних, або недостатня надійність центрального вузла або мережі передачі даних. Також використання даної стратегії рекомендується, якщо запити мають високий ступінь локалізації посилань. В іншому випадку ефективність функціонування бази може виявитися занадто низькою внаслідок великих витрат на передачу даних.

Стратегія дублювання. Реалізація цієї стратегії передбачає розміщення повної копії бази даних в декількох вузлах мережі (в граничному випадку в кожному вузлі). Відсутня завдання визначення, яку частину бази повинен містити той чи інший вузол. Однак першочергового значення набувають питання, пов'язані з узгодженням численних копій даних.

Вже згадана стратегія забезпечує найвищий рівень доступності та надійності даних, але при явних витратах зовнішньої пам'яті мережі. Обсяг бази даних обмежується меншою ємністю запам'ятовуючих пристроїв в вузлах, що зберігають дані. Обробка даних здебільшого проводиться локально, але узгодження безлічі копій вимагає синхронізації. Способи узгодження в конкретних додатках можуть бути різними, а рівень транспортних витрат істотно залежить від ступеня сталості даних.

Необхідність узгодження копій і складність управління цим процесом є причиною того, що стратегія дублювання дозволяє реалізувати паралельну обробку не настільки просто, як стратегія розчленування. У свою чергу, кожен вузол може працювати асинхронно, а час реакції на запити може бути невеликим, особливо при виконанні запитів, які не потребують узгодження копій, наприклад, при виконанні пошукових процедур. У наявності простота заміни зруйнованої копії або відновлення процесу обробки в разі виходу з ладу

вузла. Узгоджена копія може бути отримана з будь-якого робочого вузла, після чого звернення до відновленого вузла може проводитися в повному обсязі. Слід зауважити, що якщо частина мережі недоступна з якої-небудь причини, необхідним стає накладення обмежень на виконання операцій оновлення даних для підтримки глобальної узгодженості копій бази даних. Дійсно, якщо дозволити дві операції оновлення в двох вузлах при відсутності узгодження, то при поновленні нормального функціонування мережі можливе порушення узгодженості бази даних.

Таким чином, стратегія дублювання найбільш застосовна в ситуаціях, коли обсяг даних невеликий, фактор доступності та надійності звернення до даних відіграє значну роль, а інтенсивність відновлення даних невисока. Остання обставина найбільш характерно для інформаційно-пошукових систем.

Змішана стратегія. Ця стратегія розподілу даних об'єднує підходи, які використовуються при розчленуванні і дублювання, з метою придбання переваг останніх. Але в той же час змішана стратегія не позбавлена недоліків, властивих своїм прототипам.

Використання змішаної стратегії передбачає поділ бази даних на логічні фрагменти (розчленування) і наявність в мережі довільного числа їх фізичних копій, званих збереженими фрагментами (дублювання). Спільність стратегії полягає в тому, що будь-яка частина бази може бути дубльована довільне число раз, при цьому в кожному вузлі може зберігатися бажане підмножина даних.

Основною перевагою змішаної стратегії, безсумнівно, є гнучкість. Це важлива властивість дозволяє досягти компромісу між обсягом пам'яті, використовуваної в цілому і в кожному вузлі, з одного боку, і забезпечуваним рівнем надійності і оперативності, з іншого. Наприклад, архівні дані можуть зберігатися в одному вузлі, а часто використовувані можуть бути дубльовані. При дублюванні логічного фрагмента ускладнюються процедури поновлення збережених фрагментів, але значно більшу кількість даних стає локально

доступним, тобто підвищується ступінь локалізації посилань. Останнє веде до зниження транспортних витрат за рахунок зменшення числа пересилань. Стратегія допускає досить просту організацію паралельної обробки, що веде до зменшення часу реакції на запити. З'являється можливість обходу вузьких місць в мережі передачі даних.

Недоліком змішаної стратегії є необхідність зберігання інформації про місцезнаходження даних в мережі. Неважко помітити, що без наявності такої довідкової інформації неможлива реалізація стратегії розчленування. Однак в останньому випадку завжди можна вказати однозначна відповідність між логічним фрагментом, з одного боку, і вузлом, в якому розташована його фізична копія, з іншого.

У свою чергу, змішана стратегія характеризується багатозначністю такої відповідності, що істотно збільшує обсяг довідкових даних і вимагає введення процедур оптимізації доступу до збережених фрагментів. Складнощі виникають також при узгодженні довільного числа збережених фрагментів, пов'язаних з кожним логічним фрагментом. Обробка і оптимізація запитів є нетривіальними завданнями.

Вже згадана стратегія може бути рекомендована до реалізації тоді, коли жодна з більш простих стратегій не зізнається задовільною. Наприклад, необхідно забезпечити виконання високих вимог по надійності, що пред'являються до окремих частин великий за обсягом бази даних. При цьому кожен вузол може звертатися до деяких частин бази досить часто, а до решти - значно рідше. В цьому випадку стратегія розчленування не може забезпечити достатньої надійності, а стратегія дублювання може бути неприйнятною через вимоги на обсяг пам'яті в вузлах.

1.6. Специфіка проектування розподілених баз даних

Реалізація концепції РБД не може не поставити перед розробниками розподілених інформаційних систем ряду проблем. Першочергові проблеми, пов'язані з розподілом даних в мережі, підлягають вирішенню на стадії проектування РБД. Тому названа стадія життєвого циклу РБД має ряд особливостей в порівнянні з проектуванням локальних баз даних.

У розподіленої інформаційної системи логічно цілісна база даних може бути фрагментована і широко розподілена по мережі з метою поліпшення продуктивності і надійності інформаційної системи. Фрагментація і розподіл даних без централізованого планування часто є причиною неузгодженого використання даних. Вже згадана послідовність етапів проектування розподіленої бази даних враховує цей факт.

Етап аналізу предметної області. Його реалізація передбачає вивчення і опис предметної області, а також аналіз призначених для користувача потреб в інформації.

Етап концептуального проектування. За результатами попереднього етапу розробляється інфологіческая модель (інформаційна структура) предметної області.

Етап логічного проектування (проектування реалізації). Під час його проведення здійснюється вибір системи управління розподіленою базою даних і накладення обмежень обраної СУРБД на інформаційну структуру. Результатом логічного проектування виступає глобальна структура бази даних.

Етап розчленування бази даних. Даний етап пов'язаний з розподілом глобальної бази даних на логічні фрагменти. При вирішенні завдання розчленування враховують вимоги до обробки даних, характеристики обраної

СУРБД, а також характеристики технічних і програмних засобів в вузлах мережі. Результатами є сукупність логічних фрагментів і розмір кожного з них.

Етап розміщення бази даних. На цьому етапі вирішується завдання вибору вузлів мережі для розміщення в них зберігаються фрагментів, відповідних логічним фрагментами бази даних. Певні обмеження накладають вимоги по обробці даних і розмежування доступу, особливості мережі передачі даних (її топологія, пропускна здатність каналів), а також характеристики апаратури і програмного забезпечення вузлів обчислювальної мережі. Рішення являє собою перелік вузлів, з кожним з яких пов'язаний список фрагментів бази даних.

Етап проектування локальних баз даних. Є заключним в розглянутій послідовності. На ньому здійснюється проектування фізичних структур локальних баз даних, утворених в результаті виконання всіх процедур попередніх кроків.

Головна відмінність перерахованої сукупності етапів від послідовності проектування централізованих баз даних полягає в наявності етапів розчленування і розміщення БД. Тому названі етапи заслуговують більш докладного розгляду.

При розчленуванні вихідна глобальна база поділяється на безліч логічних фрагментів, званих також розділами. Природно, що повинно виконуватися вимога про збереження інформації, тобто розділи повинні містити всі відомості, наявні у вихідній глобальної базі. Додатково на процес формування розділів накладаються обмеження по їх допустимому розміру, часу реакції на запит і надійності звернення. Внаслідок цього в розділ рекомендується об'єднувати такі використовувані часто спільні записи, щоб він покращував характеристики часу відповіді на запит. Також слід прагнути до отримання необхідного рівня надійності, використовуючи по можливості меншу кратність дублювання, тобто ступінь локалізації посилань повинна бути високою при мінімальному

числі копій зберігаються фрагментів. Допустимий розмір кожного розділу, як неподільної сукупності даних, визначається фіксованим обсягом пам'яті в кожному вузлі мережі. І в загальному випадку обмеження на клас допустимих розчленовувань накладає ємність зовнішніх запам'ятовуючих пристроїв у вузлах.

Завдання розміщення розподіленої бази даних вирішується порівняно просто для двох стратегій: централізації і дублювання. Цілком очевидно, що відпадає необхідність у виконанні процедури розчленування бази даних. Якщо прийнята перша стратегія, то Типовою є одне питання: в якому вузлі слід розмістити базу даних? Реалізація другої стратегії для кожного вузла мережі вимагає вирішення питання: розміщувати або не розміщувати повну копію бази? Відповіді на ці запитання найчастіше зумовлені і залежать від структури мережі, обсягу пам'яті в її вузлах, переліку альтернатив і здорового глузду.

Значно складніше представляється задача розміщення при використанні стратегії розчленування і особливо складної при змішаній стратегії. У разі реалізації стратегії розчленування необхідно: по-перше, розчленувати базу на логічні фрагменти і, по-друге, розмістити кожен фрагмент в конкретному вузлі з урахуванням обмежень на розміщення. Завдання є итеративной і можливо, що розчленовування бази даних потрібно проводити неодноразово. Якщо ж використовується змішана стратегія, рішення стає більш складним: кожен логічний фрагмент може бути розміщений в будь-якій кількості вузлів. Кількість перестановок фрагментів зростає дуже швидко, і це є однією з причин того, що обмежуються знаходженням не оптимальні, а раціонального розміщення.

Рішення задачі оптимального розміщення фрагментів по вузлах мережі методами лінійного програмування можливо при досить жорстких припущеннях про характер потоку запитів до бази, зумовленому числі і незмінності цих запитів, заздалегідь відомому числі збережених фрагментів.

Кількість змінних і обмежень прогресує зі збільшенням числа вузлів мережі і тому отримання рішення можливо лише для задач малої розмірності. Рішення також ускладнюється при пред'явленні менш жорстких вимог до характеру запитів користувачів. Підходи до вирішення використовують в своїй основі метод динамічного програмування. Якщо ж типові запити спочатку невідомі, то використовуються статистичні методи для визначення цих запитів, а результати їх визначення служать вхідними даними для вирішення задачі розміщення.

В цілому процес розробки розподілених баз даних відрізняється високою трудомісткістю і значними матеріальними витратами. Завдання вибору найкращої відповідності між характеристиками РБД і методами розподілу даних в мережі вимагає всебічного аналізу, так як прийняті проектні рішення безпосередньо впливають на реалізацію і подальше функціонування інформаційної системи.

З концепцією баз даних тісно пов'язана відома ідея багаторівневого представлення даних, запропонована дослідницькою групою в області баз даних ANSI / SPARC. Структурна основа цього подання включає в себе три рівні, кожному з яких ставиться у відповідність модель даних.

Очевидно, що такі складні освіти як розподілені бази даних можуть зажадати більшого числа рівнів подання даних для реалізації принципу незалежності опису структури баз від даних. Дійсно, розглянута трирівнева архітектура в разі розподіленої бази даних розширена до п'яти рівнів (Рис. 1.1).

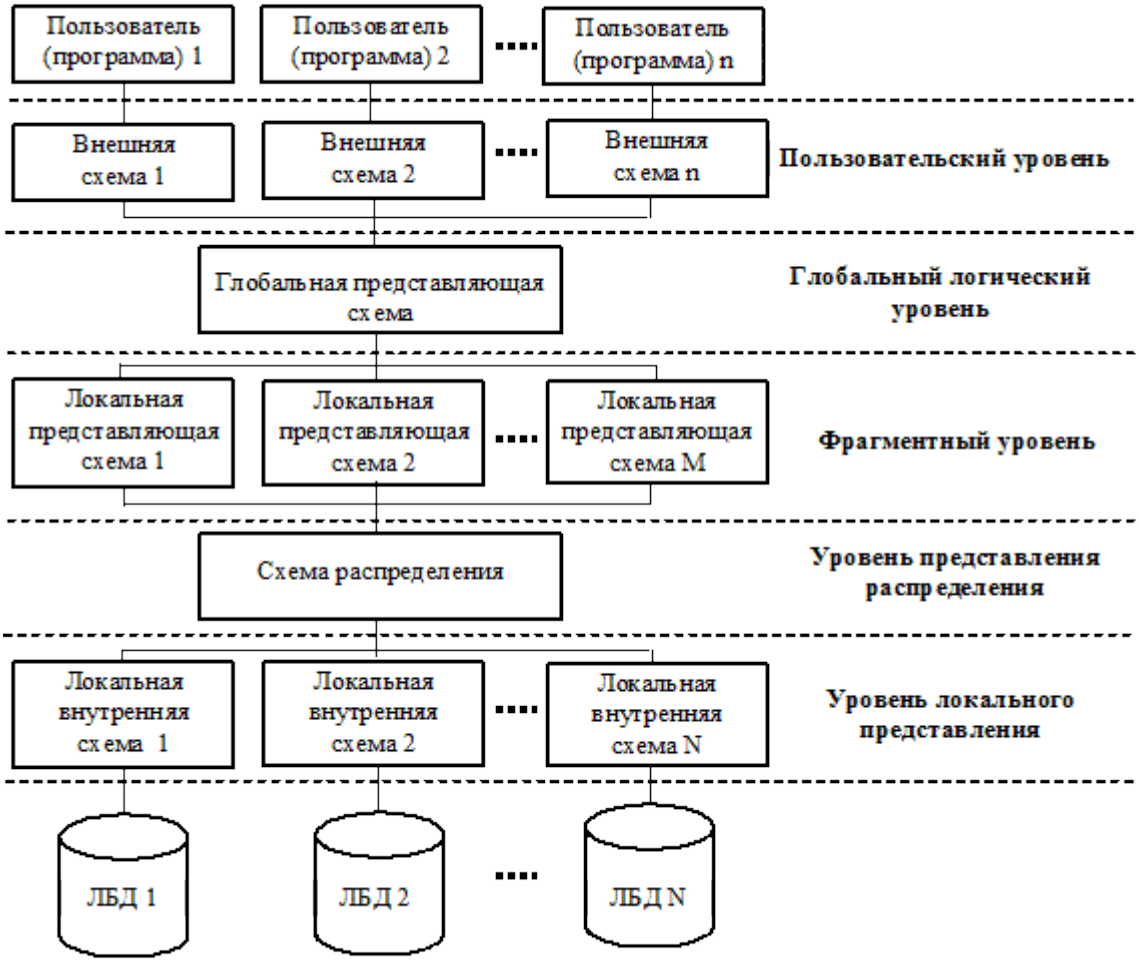

Рисунок 1.1 – Трьохрівнева структура зберігання даних

Призначений для користувача рівень представлення даних. Служить для опису частини бази даних, доступної одному конкретному користувачеві або групі користувачів, що розглядаються як один. Ця частина багаторівневого подання є, по суті, зовнішньої моделлю в концепції ANSI. Кожен користувач може мати відмінне від інших користувачів уявлення, що відповідає його вимогам і вимогам розмежування доступу.

Глобальний логічний рівень представлення даних. Цей рівень подібний до концептуального рівня уявлення концепції ANSI. Використовується для опису логічної структури всієї розподіленої бази даних, тобто в уявленні адміністратора РБД. При описі РБД цього рівня ставиться у відповідність глобальна представляє схема.

Існування третього і четвертого рівнів пояснюється розподіленої природою бази даних.

Фрагментний рівень представлення даних. Використовуючи цей рівень, адміністратор РБД визначає незв'язані підмножини бази даних, тобто логічні фрагменти, і описує їх засобами СУБД у вигляді локальних представляють схем.

Рівень представлення розподілу даних. На даному рівні визначається географічне розташування примірників кожного логічного фрагмента. Рівень допускає існування декількох фізичних фрагментів, відповідних логічного фрагменту. Цьому рівню відповідає схема розподілу даних.

Рівень локального представлення даних. Відповідає опису тієї частини бази даних, яка існує в конкретному вузлі. Безсумнівно, що ця локальна база може розглядатися як з точки зору логічної, так і фізичної структури. Однак локальне уявлення вважається описом логічної структури, при цьому фізична структура є прихованою від адміністратора РБД. Місцева база даних, як база в повному розумінні цього слова, має кілька рівнів подання, але в даному розгляді ці рівні не беруть участі.

Концепція банку даних передбачає незалежне від завдань користувачів накопичення і ведення позбавлених надмірності взаємопов'язаних даних. Всі функції з управління даними і взаємодії користувачів або прикладних програм з базами покладаються на системи управління базами даних, а користувачеві надаються спеціальні мовні засоби для опису, модифікації і вибірки даних. У разі розподілу даних однієї з проблем, які підлягають вирішенню, є інтеграція баз даних, створених в рамках різних локальних СУБД, і розробка систем управління розподіленими базами даних.

У загальному випадку СУРБД трактується як система, що забезпечує можливість роботи з декількома локальними базами даних і реалізує принцип

логічної інтеграції даних, фізично розподілених між взаємопов'язаними обчислювальними ресурсами. У функції СУРБД входять: управління доступом до даних, аналіз і розподіл транзакцій, пару різних програмних засобів (операційних систем, локальних СУБД), а також управління трансляцією запитів.

Реалізація функції управління доступом до даних забезпечує узгодженість операцій, пов'язаних з перевіркою формальної правильності запитів, управлінням одночасним зверненням до даних багатьох користувачів, захистом даних. Що стосується аналізу і розподілу транзакцій, то відповідні модулі СУРБД проводять аналіз надійшли від користувачів запитів, їх декомпозицію на підзапити і вибір дистанційного або локального методу доступу для кожного з цих підзапитів. Виконання функції сполучення забезпечує породження макрокоманд для узгодження взаємодії різних операційних систем і організації методів доступу до локальних СУБД, автоматичний виклик трансляторів. Нарешті, функція трансляції реалізує перевірку коректності текстів призначених для користувача запитів, перетворення цих текстів в форму внутрішнього мови мережі і мов маніпулювання даними локальних СУБД.

Різноманіття вже створених до теперішнього часу СУРБД передбачає їх групування з метою всебічної характеристики.

Класифікація СУРБД. В основу цієї класифікації можуть бути покладені різноманітні ознаки. Це призводить до необхідності вказівка ознак класифікації з подальшою короткою характеристикою виділяються угруповань.

Тип підтримуваних моделей даних. За цією ознакою розрізняють СУРБД, що підтримують однорідні і неоднорідні моделі даних в вузлах мережі.

Клас використовуваних ЕОМ. Тут виділяють СУРБД, орієнтовані на роботу в середовищі великих машин, з одного боку, і міні- і мікроЕОМ, з іншого.

Тип обчислювальних мереж. Відповідно до такого поділу можна вказати СУРБД для глобальних і локальних мереж відповідно.

Використовувані принципи управління. Введення в розгляд цієї ознаки припускає наявність СУРБД з централізованим і децентралізованим управлінням.

Централізоване управління розподіленою базою даних передбачає розташування СУРБД в одному з вузлів мережі. Локальні СУБД підпорядковані із сайтом. Перевагою централізованих СУРБД безсумнівно є відносна простота програмного забезпечення і реалізації. До недоліків слід віднести необхідність проходження всіх запитів через головний вузол і відсутність зв'язку між локальними СУБД при виході з ладу СУРБД.

Децентралізація управління вимагає багаторазового копіювання модулів СУРБД і розміщення їх у вузлах мережі. Кожна копія має можливість управління доступом до будь-якої локальної базі даних. Підвищення надійності таких систем і зниження транспортних витрат у порівнянні з централізованими СУРБД досягаються за рахунок істотного ускладнення програмного забезпечення. Дійсно, кожна копія СУРБД в довільний момент часу повинна мати інформацію про процеси, що протікають у всіх локальних базах даних, і враховувати її при синхронізації запитів до РБД.

Принципи проектування систем. Розподіл всієї множини СУРБД за цією ознакою дозволяє виявити системи, що базуються на стандартних, промислово експлуатованих типових СУБД, і системи, в основі яких лежать унікальні розробки.

Використання стандартних СУБД має те незаперечне достоїнство, що дозволяє об'єднувати в рамках розподіленого банку даних існуючі бази при відносно невеликих витратах. Але складності розробки інтерфейсів між рівнем локального уявлення і верхніми рівнями не гарантують високих

експлуатаційних характеристик таких систем. По суті, СУРБД розглянутого класу є надбудовами над існуючими СУБД і забезпечують інтеграцію локальних логічних уявлень в рамках логічної структури розподіленої базою даних. Програмний апарат виконується у вигляді драйверів.

Набагато більшою ефективністю в конкретних додатках мають СУРБД, засновані на оригінальних розробках. Це досягається значним подовженням періоду їх розробки, який починається практично з нуля, а адаптація існуючих систем до потреб користувачів має свої труднощі.

Створення таких складних програмних систем як СУРБД не може не поставити перед розробниками ряду проблем. До їх числа слід віднести проблему управління в неоднорідних РБД. Появі таких баз даних сприяли ті ж обставини, які призвели до інтеграції локальних баз даних в єдині системи.

При створенні неоднорідних РБД розробник має справу з різними моделями даних в локальних базах, синтаксичним і семантичним розходженням локальних СУБД навіть в рамках однієї моделі, а також з різними методами управління в кожній з цих СУБД. Тому природним є прагнення до того, щоб системи управління неоднорідними базами даних забезпечували "прозорість" бази не тільки в сенсі її распределенности, але і в сенсі неоднорідності. По всій видимості ідеальною є ситуація, коли СУРБД може забезпечити працездатність обчислювальної мережі, в якій будь-який користувач в довільному вузлі отримував би цілісне і поряд з цим індивідуальне уявлення, в той час як інформація могла б зберігатися в фізично рознесених базах, керованих різними локальними СУБД. В даний час СУРБД з настільки широкими можливостями відсутні.

Розвиток методів доступу до інформації в неоднорідних середовищах починалося з вивантаження / завантаження файлів. Суть цього методу полягає в наступному. Інформація вивантажується з вихідною програмно-апаратної середовища і запам'ятовується в загальному форматі, що розпізнається у

вихідній і об'єктної середовищах, потім завантажується в об'єктну середу. При необхідності здійснюється перетворення форматів. На практиці в якості загального формату часто використовується код ASCII, проте в ряді випадків потрібне застосування спеціально розроблених форматів, призначених для перенесення інформаційних описів та іншої семантичної інформації від відправника до одержувача. Прикладом може служити трансформація даних супутникової телеметрії, географічної інформації та ін.

Програми пересилки файлів, розроблені на базі найпростіших методів вивантаження / завантаження, не можуть бути визнані задовільними для широкого використання в якості компонентів СУРБД. Причиною є те, що вся вкрай важлива інформація про взаємозв'язок даних, їх індексації тощо може виявитися втраченої в процесі перетворення БД в сукупність окремих файлів. В результаті завантаження в об'єктну середу зажадає заново здійснити визначення бази, виконати індексацію і ряд інших дій. Часткове застосування розглянутий метод знайшов при створенні програм вивантаження інформації з нереляційних бази даних і завантаження їх в реляційну БД, а також програм розвантаження частини бази даних з великою ЕОМ в персональну. Основна проблема обробки вивантажених інформації полягає у забезпеченні її узгодженості при оновленні даних. На практиці використовується підхід, суть якого полягає у відстеженні істотних змін головною БД і періодичного розвантаженні вихідної середовища, а також у забороні оновлення інформації в об'єктному середовищі.

Спонукальною причиною пошуку нових способів з'єднання в неоднорідних середовищах стало досить бурхливе розповсюдження реляційних СУБД. Сутність робіт зводиться до створення реляційного сполучення з існуючими нереляціоннимі СУБД. Очевидно, що цей напрямок досліджень в якості відправної точки використовує розробки в області створення програм вивантаження / завантаження з нереляційних БД в реляційну. Тому дані методи можуть розглядатися з єдиних позицій як методи трансляції даних. На жаль,

основним стримуючим фактором при розробці реляційних інтерфейсів є відсутність стандартів на реляційні бази даних. Тому до теперішнього часу не вдалося створити стандартного реляційного сполучення для різноманітних СУБД.

Наступним етапом є розробка методів відображення або трансляції структур даних і відповідних їм команд мови маніпулювання даними. При використанні цих методів користувач формує запит до даних, представленим за допомогою доступної йому схеми, мовою оригіналу маніпулювання даними. Проводиться трансляція схеми вихідної БД в схему об'єктної бази, а також трансляція команд запиту мовою оригіналу маніпулювання даними в команди об'єктного мови. Типи проміжних схем і мов можуть бути різними. Число рівнів трансляції не обмежується двома і може лежати в межах від трьох до п'яти. Дані методи складні в реалізації і тому можливо спрощення: з глобальних запитів до розподіленої базі потрібно обробляти тільки запити на читання даних, а корекції здійснюються локально в вузлах мережі. Подібне спрощення обумовлено складнощами актуалізації надлишкових даних.

Спроби кардинального вирішення проблеми "прозорості" в неоднорідних розподілених базах даних складаються у відмові від застосування користувачем якогось спільної мови маніпулювання даними або від обробки певної групи запитів, наприклад, пов'язаних з читанням даних. Розглянутий далі метод є продовженням і узагальненням попередніх. В рамках опису РБД передбачається створення глобальних концептуальної і внутрішньої моделей даних. Перша являє опис логічної структури бази і є основою для організації процесів аналізу запитів користувачів і їх декомпозиції з метою формування команд звернення до локальних баз. Друга модель також є описом структури інтегрованої бази, в якому особливу увагу приділено способам організації доступу до інформації. Глобальна внутрішня модель точно характеризує інформаційні структури даних і можливі шляхи доступу до них (мережеві

маршрути, зв'язку між локальними базами даних), що не входять в компетенцію кожної локальної СУБД. При цьому модель залишається не прив'язаної до конкретної реалізації шляхів доступу і фізичного представлення інформації.

Виконання запиту до даної базі даних здійснюється наступним чином. Транслятор глобальних запитів обробляє сформульоване користувачем завдання і на підставі інформації про відповідну йому зовнішньої моделі транслює в форму, адекватну глобальної концептуальної моделі, а потім глобальної внутрішньої моделі. Остання дозволяє програмі декомпозиції запиту провести формування підзапитів і сформувати для кожного з них шляху доступу до даних. Далі відбувається трансляція підзапитів на мову маніпулювання даними конкретних локальних СУБД, вибраних для виконання, і оттранслировать запити передаються до відповідних вузли. Відповіді на підзапити підлягають передачі в вихідний вузол мережі, де програма-формувач відповіді об'єднує їх і видає користувачеві на доступному йому мовою.

Основні труднощі реалізації даного підходу, безсумнівно, полягає в побудові глобальної внутрішньої моделі розподіленої бази даних, яка враховує різноманітні зв'язки між локальними БД і шляхи доступу до даних і дозволяє проводити різноманітні трансляції.

1.7. Моделі розподіленої обробки запитів

При зверненні безлічі користувачів або прикладних програм до розподіленої базі даних на перший план висувається проблема управління паралельним виконанням запитів до РБД. Обов'язковою умовою є те, що кожен запит, пов'язаний з коригуванням даних, повинен залишати РБД в

несуперечливому стані. У такому випадку після закінчення виконання запиту необхідно встановити, що СУРБД завершила виконання всіх підзапитів, що змінюють дані. При подібному підході виключається можливість використання інформації, яка визначається перехідним станом бази даних. Дана умова може виконуватися по-різному в залежності від часу проведення операцій оновлення даних. Оперативна актуалізація проводиться в реальному масштабі часу, а періодична - в деякі встановлені моменти після накопичення змін даних. В останньому випадку виконання запитів на вибірку даних при оновленні блокується. Реалізація періодичної актуалізації представляється досить простий, однак даний підхід не може бути визнаний задовільним для використання в базах даних, які є інформаційними моделями систем, стан яких схильне до частих змін.

При оперативної актуалізації дії над даними, що проводяться у відповідності з різними запитами, можуть виконуватися СУРБД паралельно. При цьому виникає кілька ситуацій, коли операція над даними може бути не завершена. Це тупикові ситуації і циклічні рестарту. Тупикові ситуації характеризуються тим, що операції в декількох запитах (транзакціях) змушені чекати завершення один одного. Явище циклічного рестарту пов'язано з періодичним попаданням транзакції в такий стан, коли її виконання стає неможливим, після чого вона скасовується, а потім проводиться повторний запуск.

Механізм управління паралельним виконанням транзакцій повинен або виключати подібні ситуації, або забезпечувати їх дозвіл. До основних методів, що дозволяє коректно завершувати виконання оперативної актуалізації розподіленої бази даних і залишати її в несуперечливому стані, відносять методи: блокувань даних, згоди більшості і попереднього аналізу конфлікту транзакцій.

Метод блокувань даних. Даний метод є найбільш поширеним. На практиці застосовуються дві його модифікації, що розрізняються обраним способом управління. При централізованої блокування в мережі виділяється головний розподільник ресурсів, якому направляються всі вимоги транзакцій по блокуванню змінюваних даних. З отриманням права користування ресурсом транзакція актуалізації здійснює доступ до даних в будь-якому вузлі, що зберігає копію. Для забезпечення гарантованого використання несуперечливих даних транзакції, які здійснюють лише операції читання, повинні дотримуватися прийнятої дисципліни доступу до даних. Реалізація децентралізованої блокування усуває головний недолік попередньої модифікації - низький ступінь паралелізму і малу живучість системи управління базою даних, але при цьому не виключається можливість утворення тупиків.

Метод згоди більшості. При використанні названого методу вузлів надано право вирішувати питання про зміну даних за конкретним запитом «голосуванням». Це дозволяє вирішити конфліктні ситуації, що виникають при зверненні до даних і повністю виключити тупикові ситуації. Ступінь паралельного виконання транзакцій та ж, що і при децентралізованої блокування даних.

Суть методу згоди більшості полягає в наступному. З кожним компонентом бази даних зв'язується тимчасова мітка, яка фіксує момент його останньої зміни. Оператори зміни даних приймаються до виконання тільки в тому випадку, якщо вони відносяться до транзакцій, виконання яких почалося пізніше часу, зазначеного в мітці. В операторах зміни вказуються: унікальний код транзакції, її тимчасова мітка, список змінних ресурсів, нові значення даних, список базових ресурсів (використовуваних при формуванні зміни), значення їх тимчасових міток і т.д. При отриманні оператора вузол зобов'язаний виконати одну з наступних дій:

- проголосувати негативно (відхилити оператор) в разі, якщо порушено умова співвідношення тимчасових міток, названу раніше, і тимчасові мітки базових ресурсів, повідомлені в операторі, відносяться до більш ранніх моментів, ніж фактично зафіксовані в вузлі;
- проголосувати позитивно (ухвалити оператор) за умови, що всі співвідношення тимчасових міток є задовільними і даний оператор не вступає в конфлікт з іншим активним оператором (конфлікт можливий, якщо активний оператор, тобто не прийнятий і не відхилений, намагається змінити базовий ресурс розглянутого оператора або модифікується розглядаються оператором ресурс є базовим для будь-якого активного оператора);
- утриматися від голосування, коли співвідношення тимчасових міток є задовільним, але має місце конфлікт з іншими активними операторами.

Після отримання вузлом-ініціатором поновлення даних голосів від всіх вузлів, що містять змінні дані, проводиться аналіз результатів голосування. Наявність більшості позитивних голосів при відсутності негативних дозволяє проведення зміни даних, про що сповіщаються вузли, які беруть участь у цій операції.

Метод попереднього аналізу конфлікту транзакцій. Цей метод передбачає аналіз можливості виникнення конфлікту і його характеру. Кожному виду конфлікту ставиться у відповідність спеціалізований протокол синхронізації транзакцій, ефективно вирішує ситуацію, що виникла. В основі роботи протоколу перебуває техніка тимчасових міток. Попередній аналіз ситуації, що проводиться центральним вузлом, виключає утворення тупикових ситуацій.

Вибір того чи іншого методу для його подальшої реалізації залежить від конкретних додатків. Вироблення загальних рекомендацій є скрутної внаслідок відсутності відомостей про досягаються значеннях показників ефективності функціонування різних СУРБД.

Висновки

В даному розділі було проаналізовано структуру сучасного розподілу даних в розподілених базах даних.

В цілому процес розробки розподілених баз даних відрізняється високою трудомісткістю і значними матеріальними витратами. Завдання вибору найкращої відповідності між характеристиками РБД і методами розподілу даних в мережі вимагає всебічного аналізу, так як прийняті проектні рішення безпосередньо впливають на реалізацію і подальше функціонування інформаційної системи

РОЗДІЛ 2 ПОСТАНОВКА ЗАДАЧІ ТА ПОБУДОВА МАТЕМАТИЧНИХ МОДЕЛЕЙ

2.1 Математична постановка задачі

Строго математично поставлену задачу можно формалізувати наступним чином:

Яку оптимальну кількість серверів треба підтримувати в мережі (а – ієрархічного, б – кільцевого в - кубічного) типу, для того щоб середня затримка відповіді на запид до будь-якого з цих серверів буда мінімальною.

Як не важко зрозуміти суть проблеми полягає в тому, що при збільшенні кількості серверів зменшується час пошуку інформації на одному сервері, але водночас з цим збільшується час на переселання даних між серверами.

В далі розглянутих моделях будемо вважати що інформація (дані нашої бази даних) розподіляється рівномірно по всіх серверах, кількість яких необхідно знайти, і запити між серверами пересилаються одинакові проміжки часу, що не дуже відрізняється від реальної суті речей, але суттєво спрощує опис і розуміння математичних моделей.

Загальна формула для всіх моделей буде виглядати наступним чином:

$$T(n) = \frac{\mathrm{C}}{n} t_{\text{пош}} + t_{\text{пер}} F(n), \tag{2.1}$$

де:

С – загальні кільність інформаціх;

$t_{\text{пош}}$ – час пошуку необхідної одиниці інформації на сервері;

n – загільна кількість серверів

$F(n)$ – функція яка буде моделювати середню кількість переходів в середині заданої моделі мережі;

$t_{\text{пер}}(n)$ – час на перехід одного запиту через одни каназ зв’язку між серверами;

Якщо поверхносно проаналізувати отриману функцію можна, помітити цілком логічний факт, що перший доданок $\frac{\text{C}}{n} t_{\text{пош}}$ показує що при збільшенні загальної кількості серверів, як наслідок: зменшується загальний час на обробку запиту. Якщо вважати що C і $t_{\text{пош}}$ задані на перед, то графік часу середнього часу пошуку одиниці інформації буде мати вигляд звичайної спадаючої функції.

Далі вся складність заключається в тому, щоб правильно розрахувати функцію $F(n)$ для кожного типу мережі

2.2 Ієрархічна мережа

В ієрархічних локальних мережах є один або кілька спеціальних комп'ютерів — серверів, на яких зберігається інформація, яка спільно використовується різними користувачами.

Сервер в ієрархічних мережах — це постійне сховище спільних ресурсів. Сам сервер може бути клієнтом тільки сервера вищого рівня ієрархії. Тому ієрархічні мережі іноді називаються мережами з виділеним сервером. Сервери

зазвичай являють собою високопродуктивні комп'ютери, інколи з кількома паралельно працюючими процесорами, з вінчестерами великого об'єму, із високошвидкіс-ною мерёжною картою (100 МБіт/с і більше). Комп'ютери, з яких здійснюється Доступ до інформації на сервері, називаються станціями або клієнтами.

2.2.1 Математична модель

В даній постановці завдання будемо вважати, що ієрархічне дерево мережі має однорідну структуру (з вершини верхнього рівня завжди виходить R зв’язків до вузлів нижнього рівня). Також будемо вважати, що кінцева мережа буде зберігати інформацію (сервери) буде тільки на листках.

Тобто задача зводиться до того, щоб знайти найбільш оптимальну кількість рівнів ієрархічного дерева.

Як зазначалось до цього функція розподілу навантаження на сервера буде однаковою для всіх типів мереж: $\frac{C}{n} t_{\text{пош}}$

Далі необхідно розрахувати функцію яка буде розраховувати середню кількість зв’язків з вершини в графі нашої мережі яку необхідно подолати запиту який з однаковою ймовірністю звертається до будь якого серверу.Загальна структура тауого графу зображена на рисунку 2.1

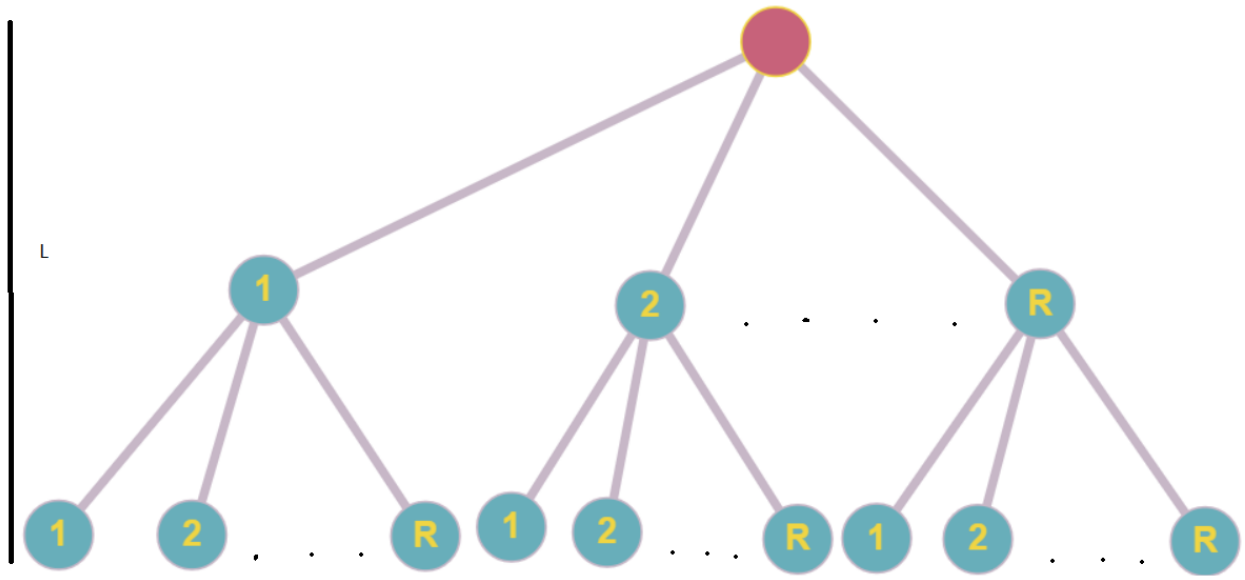

Рисунок 2.1 Загальна структура ієрархічного графа

Як видно з малюнку 1 для доступу до даних запиту необхідно пройти L (кількість рівнів ієрархії) зв'язків до серверу. Залишається тільки знайти закономірність між кількістю серверів і кількістю рівнів ієрархії в графі подібної структури для їх розміщення.

Максимальна кількість серверів для дерева з L рівнями розраховується по формулі:

$$n = R^L \tag{2.2}$$

далі необхідно визначити закономірність між L та n, не забуваючи про те, що L повинно буди цілим числом:

$$n = R^L \rightarrow \ln n = \mathrm{L} \ln R \rightarrow L = [\log_R n] \tag{2.3}$$

де оператор [] – визначає наступне ціле число після переданого в нього аргументу (якщо аргумент і так ціле число, то функція вертає це саме ціле число).

Після цього підставляємо наші отримані для даного типу мережі функції в результуючу і отримуємо:

$$T(n) = \frac{C}{n} t_{\text{пош}} + 2\, t_{\text{пер}} [\log_R n] \tag{2.4}$$

Отримана функція й використовуються як функція залежності між середнім часом відповіді мережі на запит та кількістю загальних серверів на листках ієрархічної мереж.

2.2.2 Оптимізація отриманої математичної моделі

Для знаходження мінімуму отриманої функції використовуються звичайний метод визначення екстремуму для функції однієї змінної через похідну.

Для знаходження похідної знайденої функції опустимо оператор [] з другого доданку, але після визначення точки мінімуму (позначимо її n_0) знайдемо попереднє набільше число серверів які б повністю покрили попередній рівень ієрархії нашої мережі (позначимо сю кількість як n_{-0}).

Потім аналогічним чином знайтело наступне найменше число серверів для повного заповнення рівня ієрархії який відповідає n_0 серверів (позначимо цю кількість як n_{+0}).

$$\left(\frac{C}{n} t_{\text{пош}} + 2\, t_{\text{пер}} \log_R n\right)'_n = -\frac{C}{n^2} t_{\text{пош}} + 2\, t_{\text{пер}} \frac{1}{n \ln R} \tag{2.5}$$

далі прирівнюємо отриману функцію до нуля і визначчаємо n_0

$$-\frac{C}{n^2}t_{\text{пош}} + 2\,t_{\text{пер}}\,\frac{1}{n\,\ln R} = 0 \;\rightarrow\; n_0 = \frac{C\,t_{\text{пош}}\ln R}{2\,t_{\text{пер}}} \qquad (2.6)$$

Для визначення n_{-0} і n_{+0} скористаємося формулою (2.3) нам необхідно визначити мінімальний необхідний рівень ієрархії для кількості серверів n_0. Для цього скористаємося формумою (*) підставивши в неї знайдене n_0

$$L = [\log_R n_0] \rightarrow\; n_{-0} = R^{([\log_R n_0]\,-\,1)};\; n_{+0} = R^{[\log_R n_0]} \qquad (2.7)$$

Після знаходження n_{-0} і n_{+0} залишається тільки порівняти значення нашої функції $T(n)$ в цих двох точках і та яка буде відповідати меншому часу й буде оптимальної для даної мережі.

2.3 Кільцева мережа

Кільце - топологія, в якій кожен комп'ютер з'єднаний лініями зв'язку тільки з двома іншими: від одного він тільки отримує інформацію, а іншому тільки передає. На кожній лінії зв'язку, як і у випадку зірки, працює тільки один передавач і один приймач. Це дозволяє відмовитися від застосування зовнішніх термінаторів.

Робота в мережі кільця полягає в тому, що кожен комп'ютер ретранслює (відновлює) сигнал, тобто виступає в ролі повторювача, тому загасання сигналу в усьому кільці не має ніякого значення, важливо тільки загасання між сусідніми комп'ютерами кільця. Чітко виділеного центру в цьому випадку немає, всі комп'ютери можуть бути однаковими. Однак досить часто в кільці виділяється спеціальний абонент, який управляє обміном або контролює обмін. Зрозуміло, що наявність такого керуючого абонента знижує надійність мережі, тому що вихід його з ладу відразу ж паралізує весь обмін.

Комп'ютери в кільці не є повністю рівноправними (на відміну, наприклад, від шинної топології). Одні з них обов'язково отримують інформацію від комп'ютера, який веде передачу в цей момент, раніше, а інші - пізніше. Саме на цій особливості топології і будуються методи керування обміном по мережі, спеціально розраховані на «кільце». У цих методах право на наступну передачу (або, як ще кажуть, на захоплення мережі) переходить послідовно до наступного по колу комп'ютера.

Щоб визначити функцію яка буде відповідати середній кількості ребер які необхідно пройти запиту для знаходження інформації яка знаходится на будь якому з n серверів скористаємося методом послідовного збільшення максимальної кількості серверів.

Спочатку розрахуємо функцію для n=2, а накдалі будемо збільшувати кількість серверів і прослідкуємо закономірність.

2.3.1 Математична модель

Функція розподілу навантаження на сервера буде такоюж, як і в попередній мережі : $\frac{C}{n} t_{\text{пош}}$

Далі необхідно розрахувати функцію яка буде розраховувати середню кількість зв’язків з вершини в графі нашої мережі яку необхідно подолати запиту який з однаковою ймовірністю звертається до будь якого серверу. Загальна структура тауого графу зображена на рисунку 2.2

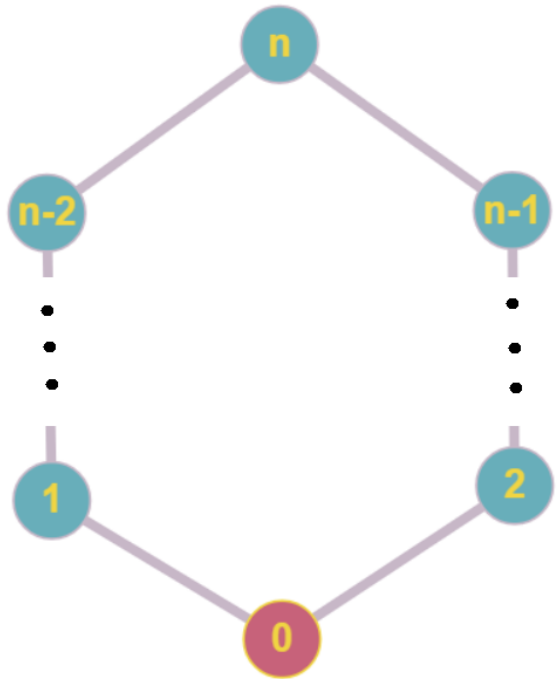

Рисунок 2.2. Загальна структура кільцевого графа

Щоб визначити функцію яка буде відповідати середній кількості ребер які необхідно пройти запиту для знаходження інформації яка знаходится на будь якому з n серверів скористаємося методом послідовного збільшення максимальної кількості серверів. Спочатку розрахуємо функцію для n=2, а накдалі будемо збільшувати кількість серверів і прослідкуємо закономірність.

Для n = 2 :

$$T_{\text{пер}} = 1 * 2 * t_{\text{пер}} = \frac{4}{2} * t_{\text{пер}} \quad (2.7)$$

Для n = 3 :

$$T_{\text{пер}} = \frac{2}{3} * 2 * t_{\text{пер}} + \frac{1}{3} * 4 * t_{\text{пер}} = \frac{8}{3} * t_{\text{пер}} \quad (2.8)$$

$\frac{2}{3}$ та $\frac{1}{3}$ в даній формулі це вірогідність того, що необхідні запиту данні знаходяться на відстанні одне і два ребра від точки входу відповідно.

Для n = 4 :

$$T_{\text{пер}} = \frac{2}{4} * 2 * t_{\text{пер}} + \frac{2}{4} * 4 * t_{\text{пер}} = \frac{12}{4} * t_{\text{пер}} \quad (2.9)$$

Для n = 5 :

$$T_{\text{пер}} = \frac{2}{5} * 2 * t_{\text{пер}} + \frac{2}{5} * 4 * t_{\text{пер}} + \frac{1}{5} * 6 * t_{\text{пер}} = \frac{18}{5} * t_{\text{пер}} \quad (2.10)$$

Для n = 6 :

$$T_{\text{пер}} = \frac{2}{6} * 2 * t_{\text{пер}} + \frac{2}{6} * 4 * t_{\text{пер}} + \frac{2}{6} * 6 * t_{\text{пер}} = \frac{24}{6} * t_{\text{пер}} \quad (2.11)$$

Для n = 7 :

$$T_{\text{пер}} = \frac{2}{7} * 2 * t_{\text{пер}} + \frac{2}{7} * 4 * t_{\text{пер}} + \frac{2}{7} * 6 * t_{\text{пер}} +$$

$$+ \frac{1}{7} * 8 * t_{\text{пер}} = \frac{32}{7} * t_{\text{пер}} \quad (2.12)$$

З данної послідовності можна прослідкувати закономірність для довільного *n:*

$$T_{\text{пер}}\,(n) = \frac{\{n\}|\frac{n+1}{2}|}{n}\,t_{\text{пер}} \tag{2.13}$$

де оператор {} – означає наступне парне число після переданого в нього аргументу (якщо аргумент і так парне число, то {} вертає це саме ціле число). Оператор | | – означає попереднє ціле число перед переданого в нього аргументу (якщо аргумент і так ціле число, то | | вертає це саме ціле число).

Після цього просто підставляєм значення отриманої функції в загальну формулу для наших мереж:

$$T(n) = \frac{C}{n}t_{\text{пош}} + \frac{\{n\}|\frac{n+1}{2}|}{n}\,t_{\text{пер}} \tag{2.14}$$

2.3.2 Оптимізація отриманої математичної моделі

Для знаходження мінімуму отриманої функції використовуються звичайний метод визначення екстремуму для функції однієї змінної через похідну.

Для знаходження похідної знайденої функції опустимо оператори {} і | | з другого доданку, але після визначення точки мінімуму (позначимо її n_0) знайдемо попереднє і наступне ціле число серверів для того щоб орівняти значення $T(n_0)$, $T(n_0-1)$, $T(n_0+1)$ і мінімальне з них і буде оптималіним для даного типу мережі.

$$\left(\frac{C}{n}t_{\text{пош}} + \frac{n(\frac{n+1}{2})}{n}\,t_{\text{пер}}\right)'_n = -\frac{C}{n^2}t_{\text{пош}} + \frac{t_{\text{пер}}}{2} \tag{2.15}$$

далі прирівнюємо отриману функцію до нуля і визначчаємо n_0

$$-\frac{C}{n^2}t_{\text{пош}} + \frac{t_{\text{пер}}}{2} = 0 \rightarrow n_0 = \frac{2\,C\,t_{\text{пош}}}{t_{\text{пер}}} \tag{2.16}$$

Після цього визначаємо значення $T(n_0)$, $T(n_0-1)$, $T(n_0+1)$ і порівнюємо аргумент найменшого з них і буде оптимальним.

2.4 Решітчаста мережа

Решітка (англ. Grid network) - це топологія, в якій вузли утворюють регулярну багатовимірну грати. При цьому кожне ребро решітки паралельно її осі і з'єднує два суміжних вузла уздовж цієї осі.

Одновимірна «решітка» - це ланцюг, що з'єднує два зовнішніх вузла (мають лише одного сусіда) через кілька внутрішніх (у яких по два сусіда - зліва і справа). При з'єднанні обох зовнішніх вузлів виходить топологія «кільце». Дво- і тривимірні решітки використовуються в архітектурі суперкомп'ютерів (частіше в варіанті багатовимірного тора). Раніше також певною популярністю користувалися мережі з топологією гіперкуб (багатовимірний куб, кожна розмірність якого дорівнює 2, всього 2^n вузлів, де n - кількість вимірювань гіперкуба)

2.4.1 Математична модель

Функція розподілу навантаження на сервера буде такоюж, як і в попередній мережі : $\frac{C}{n} t_{\text{пош}}$

Далі необхідно розрахувати функцію яка буде розраховувати середню кількість зв’язків з вершини в графі нашої мережі яку необхідно подолати запиту який з однаковою ймовірністю звертається до будь якого серверу.

В нашій науковій роботі будемо розглядати решіччестий граф в основі якого лежить простий квадрат 2 на 2 (Рис 2.3.)

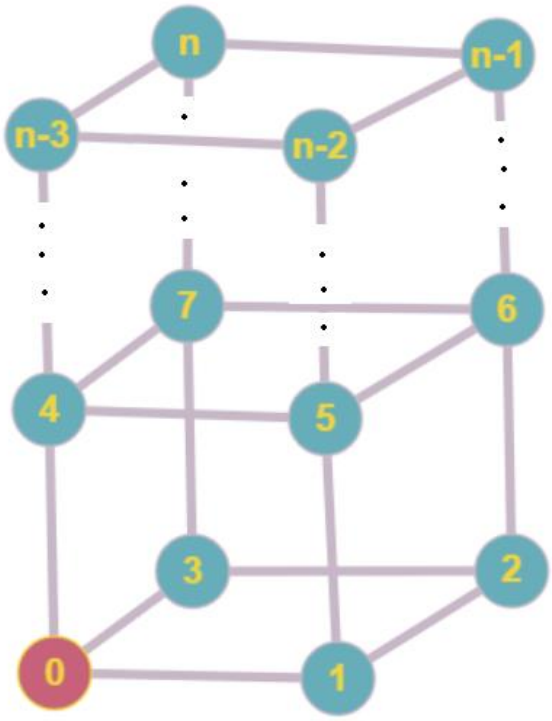

Рисунок. 2.3. Загальна структура решітчастого графа який досліджується в роботі

Математична модель для даної мережі буде дуже схожую на модель для ієрархічної мережі, бо там для знаходження оптимальної кількості серверів необхідно було знайти оптимальку кількість рівнів ієрархії, а випадку з решіточної мережі необхідно знайти оптимальну кількість рівнів (будемо вважати, що кожен рівень повинен бути повністю заповнений серверами, на кожному рівні по чотири сервера).

В данній постановці задачі будемо вважати що дані рівномірно розподілені по мережі, і зважаючи на це ймовірність того, що інформація по конкретному запиту з однаковою ймовірністю знаходится в усіх вузлах графа мережі.

Для знаходження функції яка б оцінювала середню кількість ребер яку необхідно подолати запиту для знаходження необхідного серверу з данними розглянемо спочатку решітчасту мережу з одним рівнем (Рис 2.4.):

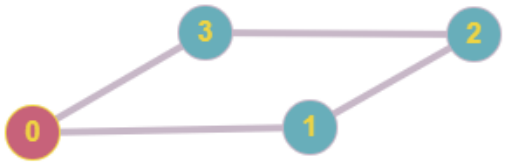

Рисунок 2.4. Решітчастий граф з одним рівнем

В цьому випадку задача зводиться до часткового випадку кільцевої мережі:

$$T_{\text{пер}} = \frac{2}{3} * 2 * t_{\text{пер}} + \frac{1}{3} * 4 * t_{\text{пер}} = \frac{8}{3} * t_{\text{пер}} \quad (2.17)$$

Для двох рівнів заданої мережі:

$$T_{\text{пер}} = \frac{3}{7} * 2 * t_{\text{пер}} + \frac{3}{7} * 4 * t_{\text{пер}} + \frac{1}{7} * 6 * t_{\text{пер}} = \frac{24}{7} * t_{\text{пер}} \quad (2.18)$$

Для трьох рівнів:

$$T_{\text{пер}} = \frac{3}{11} * 2 * t_{\text{пер}} + \frac{4}{11} * 4 * t_{\text{пер}} +$$

$$+ \frac{3}{11} * 6 * t_{\text{пер}} \ \frac{1}{11} * 8 * t_{\text{пер}} = \frac{48}{11} * t_{\text{пер}} \quad (2.19)$$

Проаналізувавши послідовність для більших значень l (кількість рівнів решітчастої мережі), можна отримати формулу для довільного цілого l:

$$T_{\text{пер}}(l) = \frac{4l(l+1)}{4l-1} t_{\text{пер}} \quad (2.20)$$

Після цього просто підставляєм значення отриманої функції в загальну формулу для наших мереж:

$$T(l) = \frac{C}{4l-1} t_{\text{пош}} + \frac{4l(l+1)}{4l-1} t_{\text{пер}} \quad (2.21)$$

2.4.2 Оптимізація отриманої математичної моделі

Для знаходження мінімуму отриманої функції використовуються звичайний метод визначення екстремуму для функції однієї змінної через похідну.

$$\left(\frac{C}{4l-1} t_{\text{пош}} + \frac{4l(l+1)}{4l-1} t_{\text{пер}}\right)'_l = -\frac{4C}{(4l-1)^2} t_{\text{пош}} + t_{\text{пер}} \frac{16l^2+8l-4}{(4l-1)^2} \quad (2.22)$$

Після перетворень і розв’язання рівняння $T(l)'_l = 0$ отримуэмо вираз для знаходження l_0:

$$l_0 = \frac{t_{\text{пер}} + \sqrt{5t_{\text{пер}}^2 + 4Ct_{\text{пер}}t_{\text{пош}}}}{4t_{\text{пер}}} \quad (2.23)$$

Висновки

В данному розділі було описано алгоритм розробки математичних моделей для кожної з типів мереж. Було представленно сам процес розробки моделей а також алгоритми мінімізації кількості серверів для кожного серверу.

РОЗДІЛ 3 РОЗРОБКА ПРОГРАМИ ДЛЯ ВІЗУАЛІЗАЦІЇ СТВОРЕНИХ МАТЕМАТИЧНИХ МОДЕЛЕЙ

3.1 Програмне середовище розробки Microsoft Visual Studio

Для створення системи було обрано програмне середовище Visual Studio 2015 Community.

Microsoft Visual Studio – серія продуктів фірми Microsoft, які включають інтегровану середу розробки та ряд інших інструментальних засобів. Ці продукти дозволяють розробляти як консольні програми, так і програми з графічним інтерфейсом, в тому числі з підтримкою технології Windows Forms. Дану середу розробки використовують для створення програмного забезпечення: від планування до розробки користувальницького інтерфейсу, написання коду, тестування, відладки, аналізу якості коду та продуктивності, розгортання у середах клієнтів та збору даних телеметрії по використанню. Ці інструменти необхідні для ефективної спільної діяльності [12].

Visual Studio включає в себе редактор вихідного коду з підтримкою технології IntelliSense і можливістю найпростішого рефакторингу коду. Вбудований відладчик може працювати як відладчик рівня вихідного коду, так і як відладчик машинного рівня. Решта вбудовані інструменти включають в себе редактор форм для спрощення створення графічного інтерфейсу додатку, веб-редактор, дизайнер класів і дизайнер схеми бази даних. Visual Studio дозволяє створювати і підключати сторонні доповнення (плагіни) для розширення функціональності практично на кожному рівні, включаючи додавання підтримки систем контролю версій вихідного коду (як наприклад, Subversion і Visual SourceSafe), додавання нових наборів інструментів (наприклад , для редагування і візуального проектування коду на предметно-орієнтованих мовах програмування або інструментів для інших аспектів процесу розробки програмного забезпечення (наприклад, клієнт Team Explorer для роботи з Team Foundation Server) [13].

Важливим доповненням в Visual Studio 2015 є інструменти для багатопотокової розробки з використанням як некерованого коду, так і .NET Framework.

3.2 Платформа .NET Framework

Дана інформаційна система написана мовою програмування C# з використанням .NET Framework. Для підключення баз даних використовувався сервіс ADO.NET. Дизайн додатку Windows Forms розроблений за допомогою мови програмування C#.

Програмна технологія Microsoft .NET — технологія, запропонована фірмою Microsoft як платформа для створення як звичайних програм, так і веб-застосунків. Багато в чому є продовженням ідей та принципів, покладених в технологію Java. Одною з ідей .NET є сумісність служб, написаних різними мовами. Хоча ця можливість рекламується Microsoft як перевага .NET, платформа Java має таку саму можливість [8].

Microsoft .Net Framework є так званою програмною платформою. У загальних рисах можна провести аналогію з відеофайлами, які не будуть відтворюватися якщо в системі не встановлений потрібний кодек. В даному випадку відеофайл – це програма, написана з використанням технології .Net, а кодек – це сама платформа Microsoft .Net Framework. Причому для роботи програми, написаної на конкретній версії фреймворка, потрібна установка саме цієї версії [19].

Зроблено це для того, щоб розробник міг максимально абстрагуватися від системного оточення на комп'ютері користувача. Його не повинно хвилювати,

яка операційна система встановлена, яка розрядність у процесора – 32-х або 64-бітна, яка у нього архітектура і т.д. Для запуску програми досить щоб під цю систему існувала і була встановлена реалізація .Net Framework [19].

Для операційних систем Windows розробкою платформи займається її творець – компанія Microsoft. Існують також незалежні реалізації. Таким чином технологія .NET — крос-платформова технологія, в цей час існує реалізація для платформи Microsoft Windows, FreeBSD (від Microsoft) і варіант технології для ОС Linux в проекті Mono (в рамках угоди між Microsoft з Novell), DotGNU [10].

Технологія .NET поділяється на дві основні частини — середовище виконання (по суті віртуальна машина) та інструментарій розробки [9].

Платформа складається з двох частин. Основою є середа Common Language Runtime (CLR), яка може виконувати як звичайні програми, так і серверні додатки. Друга, не менш важлива частина – це бібліотека класів Framework Class Library (FCL), що містить в собі безліч компонентів для роботи з базами даних, мережею, введенням/висновком, файлами, призначеним для користувача інтерфейсом і т.д. Це дозволяє розробнику не займатися низькорівневим програмуванням, а використовувати вже готові класи.

Мова програмування C# — об'єктно-орієнтована мова, з безпечною системою типізації для платформи.NET. Розроблена Андерсом Гейлсбергом, Скотом Вілтамутом та Пітером Гольде під егідою Microsoft Research (при фірмі Microsoft) [10].

Синтаксис C# близький до C++ і Java. Мова має строгу статичну типізацію, підтримує поліморфізм, перевантаження операторів, вказівники на функції-члени класів, атрибути, події, властивості, винятки, коментарі у форматі XML. Перейнявши багато що від своїх попередників — мов C++, Delphi, Модула і Smalltalk — C#, спираючись на практику їхнього використання, виключає деякі моделі, що зарекомендували себе як

проблематичні при розробці програмних систем, наприклад множинне спадкування класів (на відміну від C++) [11].

Мова програмування C# розроблялась як мова програмування прикладного рівня для CLR і тому вона залежить, перш за все, від можливостей самої CLR. Це стосується, перш за все, системи типів C#. Присутність або відсутність тих або інших виразних особливостей мови диктується тим, чи може конкретна мовна особливість бути трансльована у відповідні конструкції CLR. Так, з розвитком CLR від версії 1.1 до 2.0 значно збагатився і сам C#; подібної взаємодії слід чекати і надалі. (Проте ця закономірність буде порушена з виходом C# 3.0, що є розширеннями 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, збірка сміття не реалізована в самому C#, а проводиться CLR для програм, написаних на C# точно так, як і це робиться для програм на VB.NET, J# тощо [11].

Важливими частинами бібліотеки класів є:

− Windows Forms – відповідає за розробку графічного інтерфейсу. Фактично є обгорткою над Win32 API;

− ADO.NET – надає доступ даним. В основному використовується для роботи з базами даних;

− ASP.NET – технологія розробки веб-сайтів, веб-додатків і веб-сервісів;

− Language Integrated Query (LINQ) – реалізація мови запитів, що нагадує по синтаксису SQL в програмах на .Net;

− Windows Presentation Foundation (WPF) – система створення графічних інтерфейсів, що використовує мову розмітки XAML. На відміну від Windows Forms використовує графічну технологію DirectX, що забезпечує швидшу роботу за рахунок апаратного прискорення графіки [18];

– Windows Communication Foundation (WCF) – система обміну даними між додатками .Net. Використовується для створення розподілених додатків.

3.3 Розробка архітектури і функціональної схеми програми

Для здійснення розрахунків за наведеними математичними моделями було розроблено власний програмний продукт, призначений для знаходження оптимальної кількості серверів для кожної з мереж та для візуалізації (в графіках) поведінки мереж при різних початкових умовах.

Структурну схему програми можна спостерігати на рис. 3.1.

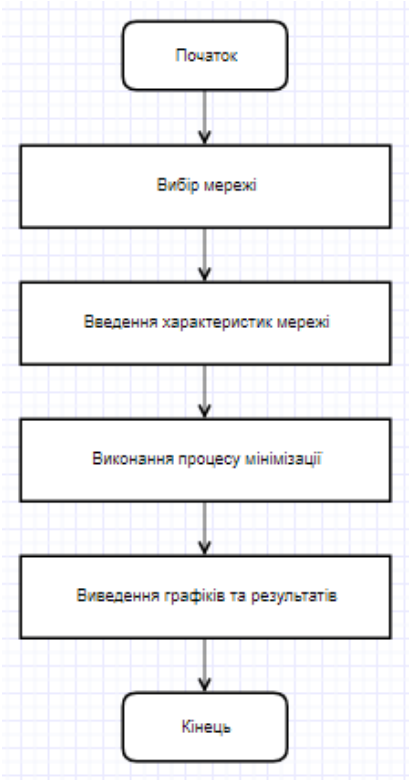

Рисунок 3.1 — Структурна схема програми

У ході розробки програмного продукту були розроблені наступні класи:

− интерфейс IFiller – интерфейс, що описує поведінку заповнення результатів розрахунків;

− абстрактний клас AParams – клас, що містить в собі універсальні властивості які характеризують всі типи мереж;

− клас Circle – клас для кільцевої мережі, забезпечує розрахунок точки мінімуму та побудову графіку;

− клас Hierarhy – клас для ієрархічної мережі, забезпечує розрахунок точки мінімуму та побудову графіку;

− клас Cube – клас для решітчастої мережі, забезпечує розрахунок точки мінімуму та побудову графіку;

− клас Result – загальний клас, що використовується для зберігання розрахованих результатів;

− клас GrafParams – клас, що забезпечує коректну і універсальну візуалізацію графіків для всіх типів мереж;

Вище описані класи графічно зображені на (Рис 3.2)

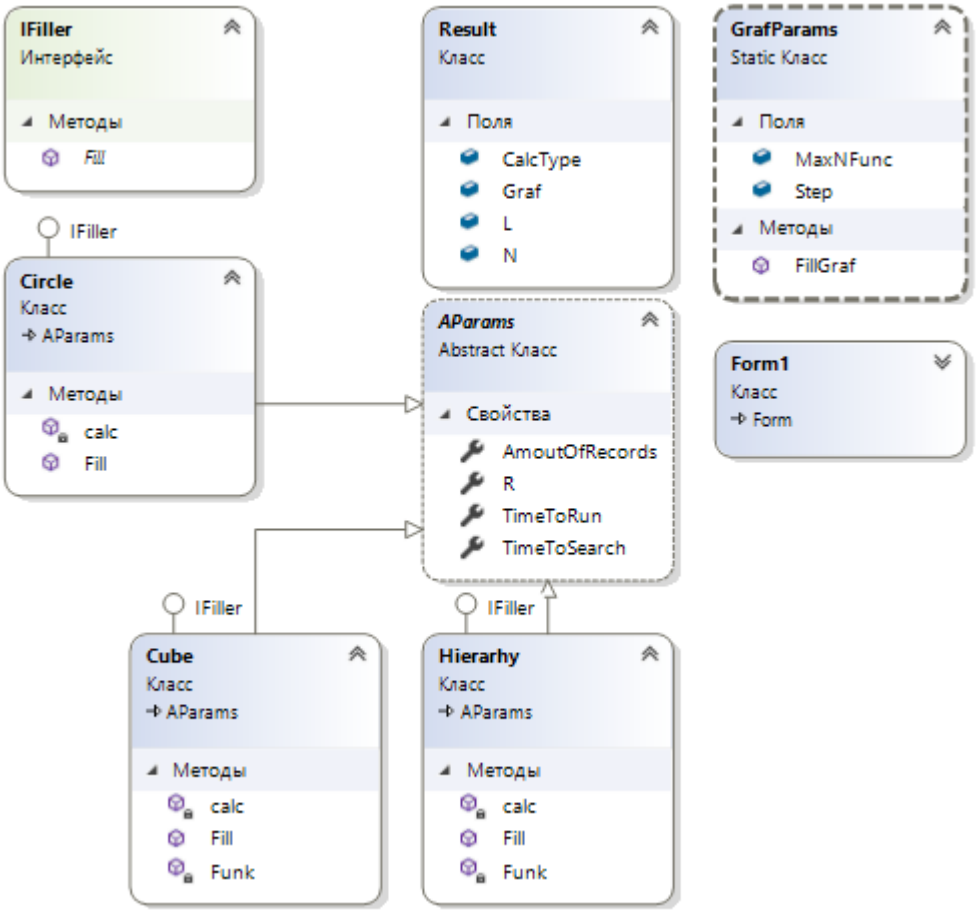

Рисунок 3.2 — Схема взаємодії класів програми

Інструкція по експлуатації програми:

1. Вибрати математичну модель в верхньому меню (Рис. 3.3).

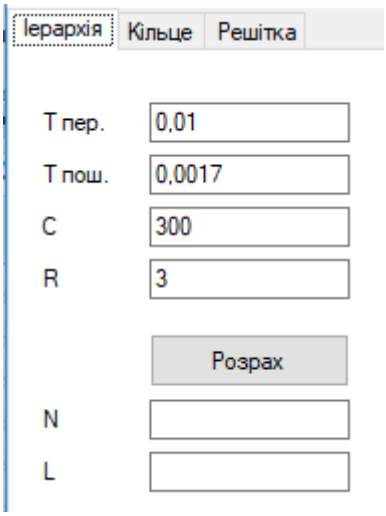

Рисунок 3.3 — Вибір типу мережі

2. Відповідно до обраної мережі на екрані з’являться додаткові (або зникнуть зайві) поля для введення необхідних даних (Рис. 3.4 - 3.5).

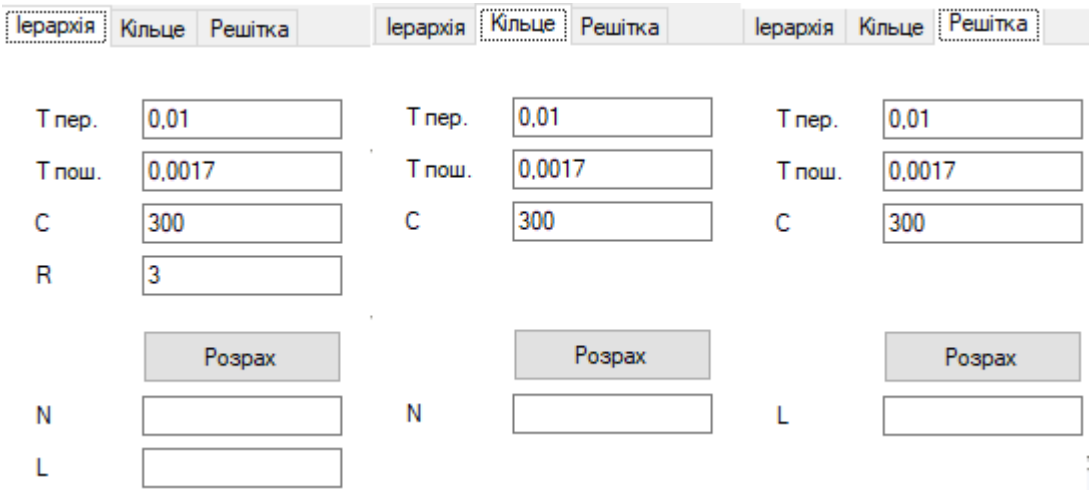

Рисунок 3.4 — Вигляд полів для вводу характеристик мереж

В правому блоці знаходиться елемент для відображення графіку мережі відповіно до введених характеристик (для зручності реалізована можливість порівнювати графіки різних типів мереж) (Рис 3.5.).

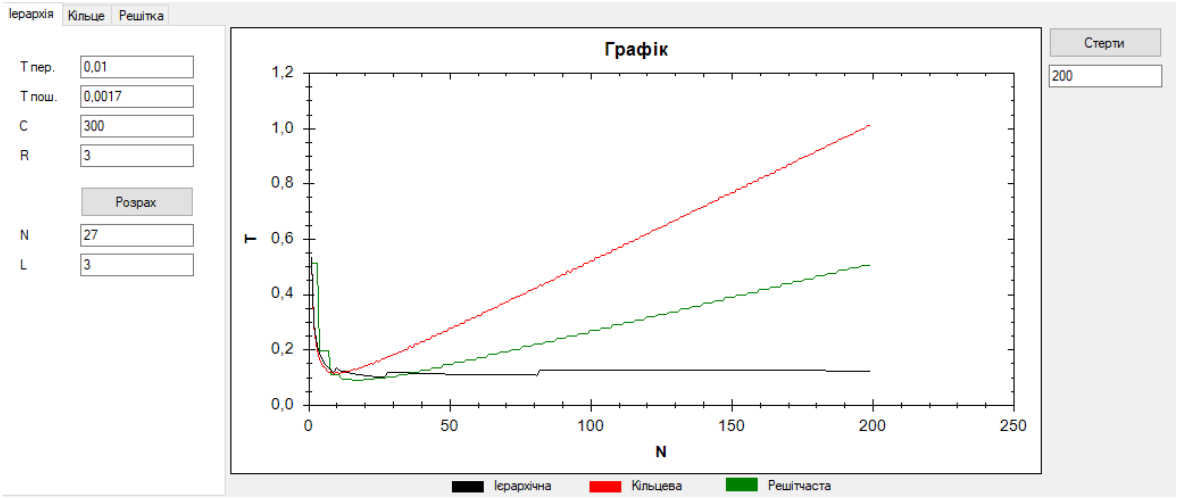

Рисунок 3.5 — Вигляд елементу для відображення графіків

Висновки

В розділі було експериментально досліджено та проаналізовано всі типи мереж, порівняно їх між собою. Було спостережено, що ієрархічна мережа найкраще проявляє себе при великій кількості даних або при малій швидкості обробки інформації. Інші ж мережі проявляють себе краще при менших обьемах даних.

РОЗДІЛ 4 АНАЛІЗ ДОСЛІДЖУВАНИХ МЕРЕЖ ЗА ДОПОМОГОЮ РЕАЛІЗОВАНОЇ ПРОГРАМИ

4.1 Ієрархічна мережа

Ієрархічна модель мережі є найбільш переважною, оскільки дозволяє створити найбільш стійку структуру мережі і раціональніше розподілити ресурси. Також гідністю ієрархічної мережі є вищий рівень захисту даних.

До недоліків ієрархічної мережі, в порівнянні з одноранговими мережами, відносяться:

- необхідність додаткової ОС для сервера;
- вища складність установки і модернізації мережі;
- необхідність виділення окремого комп'ютера як сервера [13].

Аналізуючи ієрархічну мережу з точки зору оптимальності для запитів, можно побачити доволі таки логічну але неочевидну спочатку річ. При переході кількості серверів до величини яка означає збільшення рівнів ієрархії; графік часу відповіді на запит різко робить підйом в верх. Це зпричинено тим, що через появу додаткового рівня ієрархії всім запитам необхідно проходити додатковий перехід «ребро» графу для доступу до серверу з інформацією, а в свою чергу самих серверів з данними стало лише на 1 більше, що не дуже то й скоротило навантаження на кожен з них. Тому час пошуку необхідної одиниці інформації займає майже той же час, що й при меншій кількості рівнів ієрархії.

В свою чергу при збільшенні кількості серверів в рамках одного рівня ієрархії можна спостерігати монотонне зменшення часу на обробку запиту, яка пояснюється тим, що переходів «ребер» запиту потріно долати туж саму кількість, але в свою чергу кожен з серверів має менше навантаження за рахунок рівномірного розподілу(Рис. 4.1).

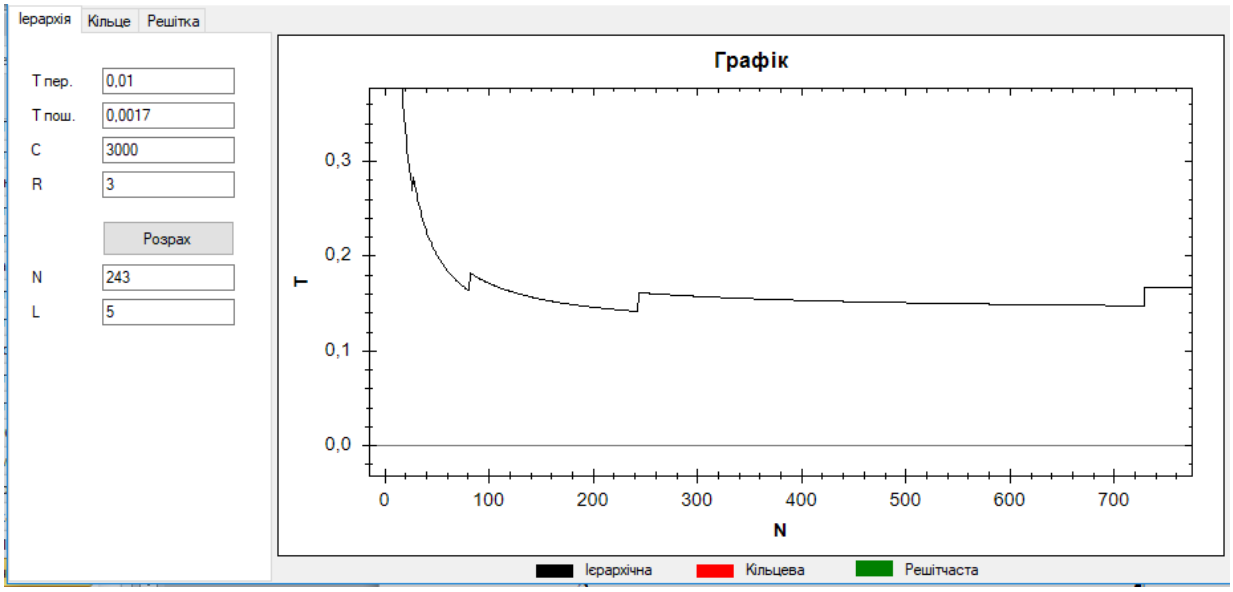

Рисунок 4.1 — Вигляд графіку для ієрархічної мережі біля точки мінімуму

З першого погляту може здатися что функція спадаюча і N=243 не є оптимальною кількістю серверів для найменшого часу відповіді на запит (при заданих характеристиках мережі). Проте якщо подивитися в більшому масштабі то помітно, що при збільшенні рівнів ієрархії суттєво збільшується час відповіді на запит. Це спричинено тим, що сервери й так не переповнені, але в той самий час збільшується кількість переходів «ребер» які потрібно пройти для доступу до даних, що збільшує час передачі (Рис. 4.2).

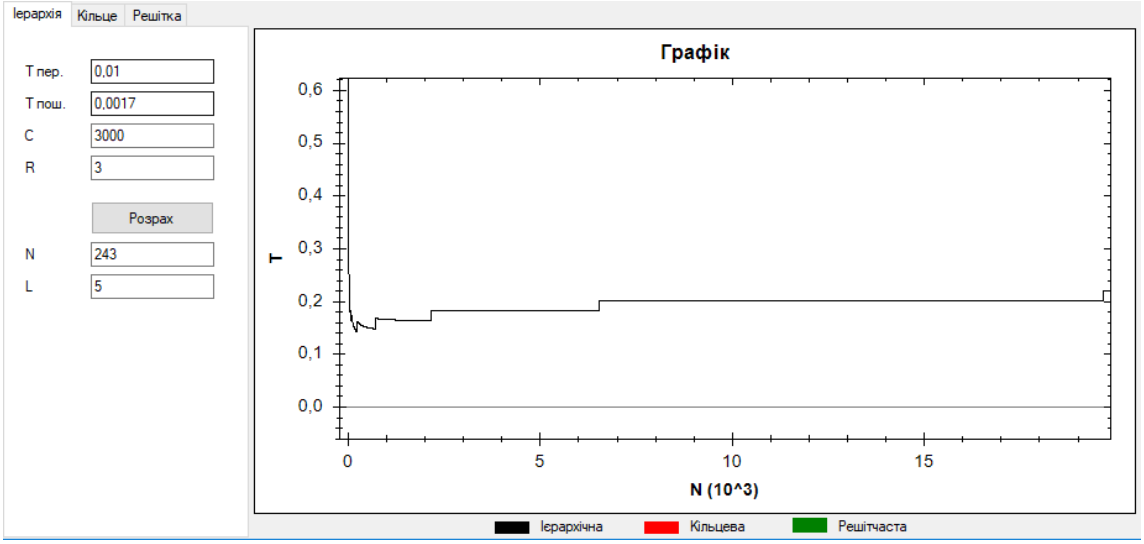

Рисунок 4.2 — Повний вигляд графіку ієрархічної мережі

4.2 Кільцева мережа

Аналізуючи кільцеву мережу модна спостерігати більш очікувану поведінку: при невеликій кількості серверів мережа доволі довго буде відповідати на запит, бо сервера переповнені даними і пошук в кожному з них займає багато часу. В свою чергу безкінечне збільшення кільца, додаванням все більшої кількості серверів, також не має сенсу, бо через збільшення величини кільця збільшується час передачі запиту від вузла до вузла за рахунок чого зменшується ефективність всієї мережі.

Зважаючи на вище перечислені властивості, і на те що для даного типу мережі не властиві перепади, як для ієрархічній мережі графік кільцевої мережі нагадує звичайний графік опуклої функції з одним глобальним екстремумом (Рис. 4.3).

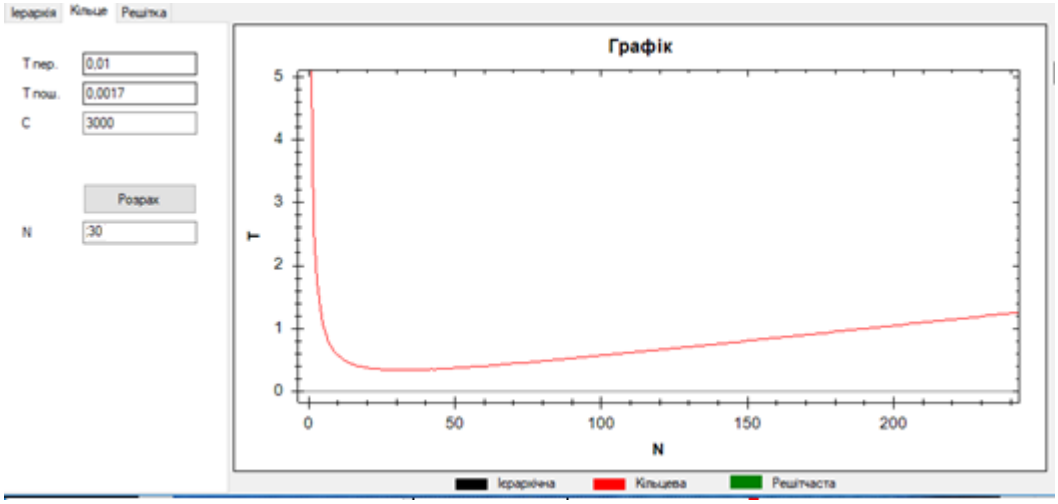

Рисунок 4.3 — Повний вигляд графіку кільцевої мережі

Проте якщо збільшити масштаб можна помітити, що і тут не обійшлося без дискретизації. Така поведінка графіку пов’язана з тим що при непарній кількості серверів *2k-1* запиту необхідно долати ту саму кількість ребер, що і для парної кількості *2k* , але в свою чергу для *2k* загальна кількість серверів в мережі збільшується на один, що в свою чергу веде до зменшення середнього часу обробки на одному сервері, що в свою чергу не значною мірою зменшує загальний час відповіді мережі на запит (Рис. 4.4).

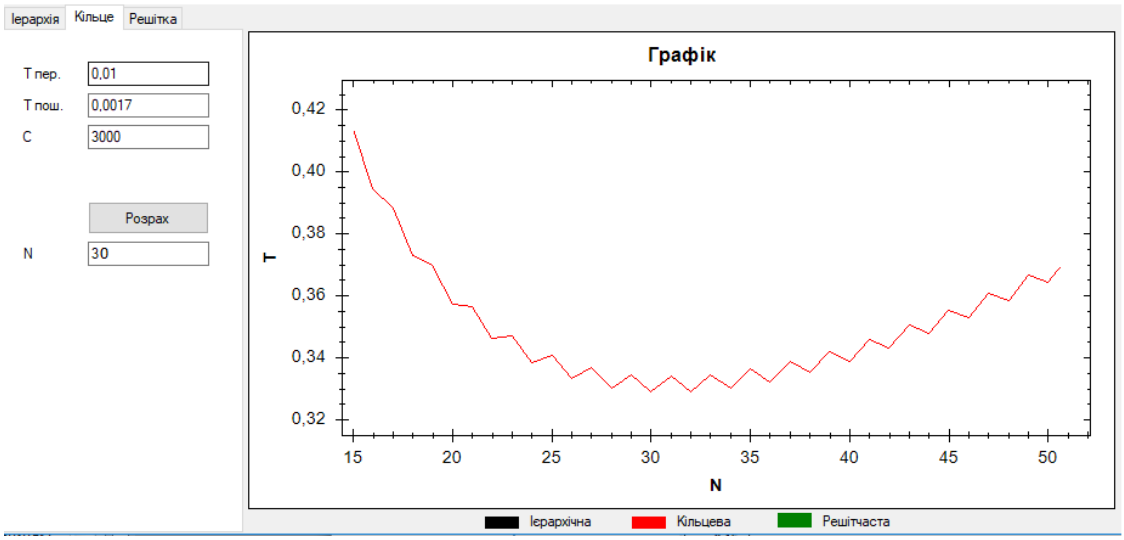

Рисунок 4.4 — Вигляд графіку кільцевої мережі біля точки мінімуму

4.3 Решітчаста мережа

Решітчаста мережа по своїй суті повністю аналогічний до кільцевої мережі за виключенням того, що тут присутня дискретизація по рівнях кожен з яких містить в собі 4 сервери (Рис. 4.5).

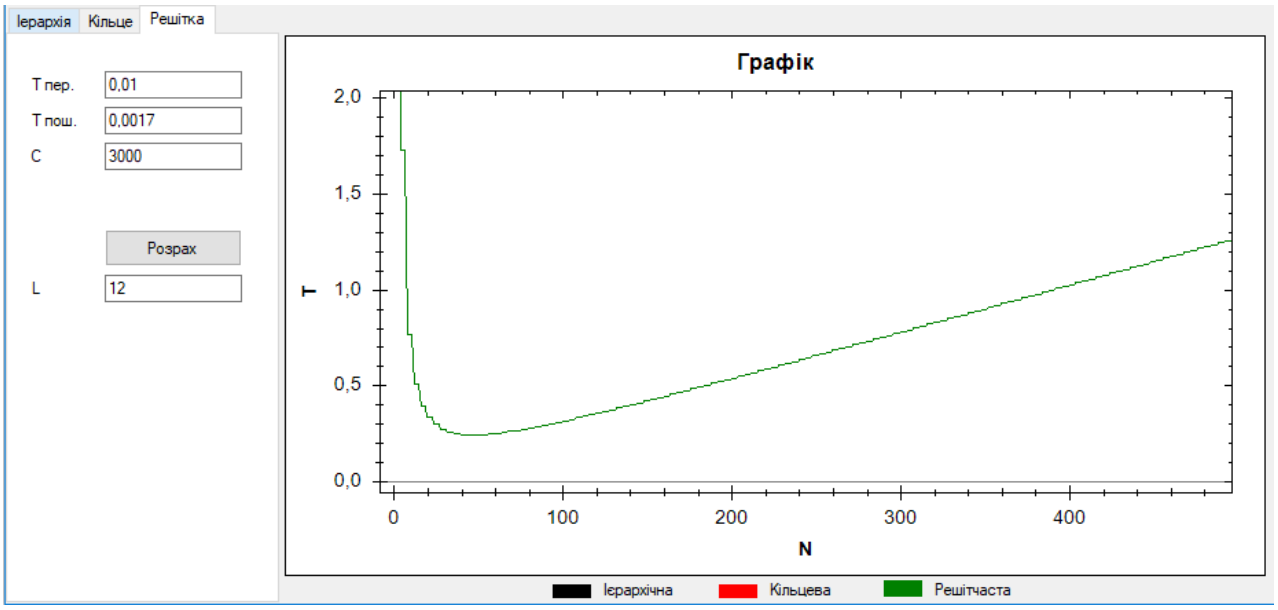

Рисунок 4.5 — Повний вигляд графіку решітчастої мережі

Вигляд графіку для решітчастої мережі біля точки мініму також майже аналогічний до кільцевої мережі, за виключенням того факту, що крива даного типу мережі до точки мініму тільки не зростає, а після неї тільки не спадає. Це пов’язано з тим, що в даній математичній можелі в нас всі рівні куба повністю заповнені серверами (Рис. 4.6).

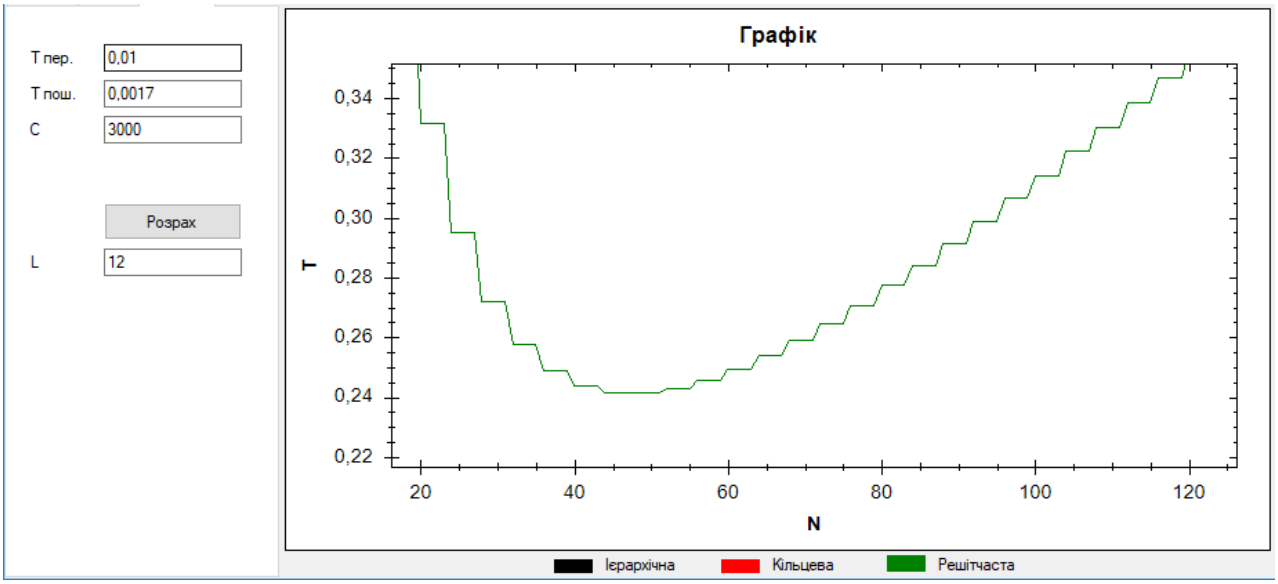

Рисунок 4.6 — Вигляд графіку кільцевої мережі біля точки мінімуму

4.4 Порівняння всіх типів мереж між собою

Користуючись написаною прикладною програмою є можливість порівняти всі типи мереж за однакових, або за різних характеристик мереж:

1. Порівняємо мережі при наступних вхідних значенях характеристик мереж (Рис. 4.7):

C = 3000; $t_{пош}$ = 0,0002; $t_{пер}$ = 0,02; R = 3 (кількість зв’язків для ієрархічної мережі)

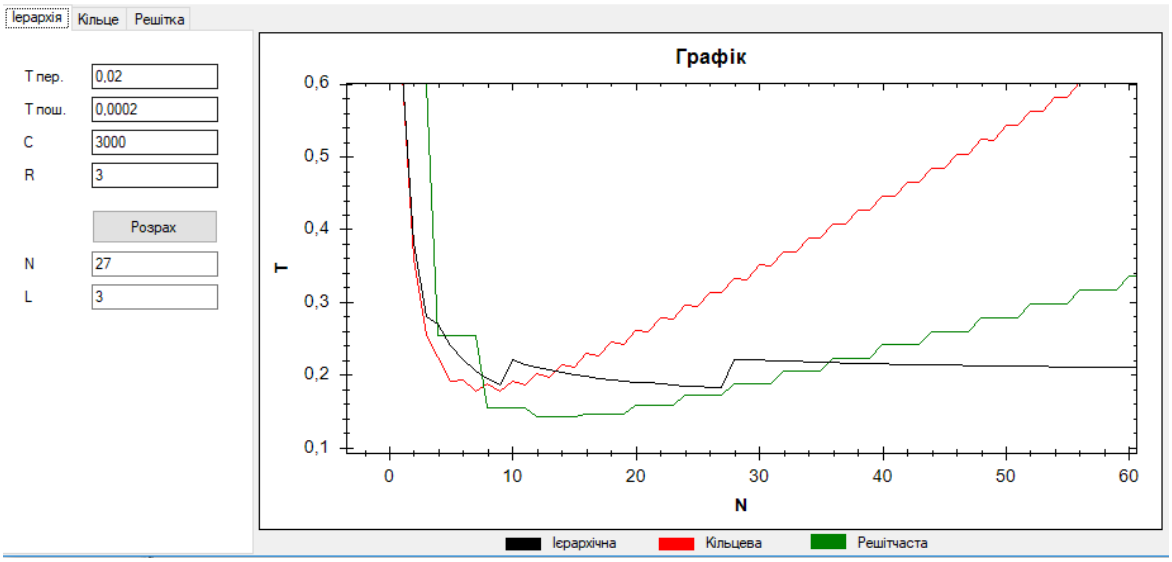

Рисунок 4.7 — Порівняння мереж при звичайних вхідних параметрах

При заданих характеристиках найбільш оптимальною виявилася решітчаста мережа, спробуємо поекспереннтувати з початковими характеристиками і проаналізуємо поведінку графіків.

2. При збільшенні $t_{\text{пош}}$ в 10 разів не можно не помітит, що ієрархічна мережа є найбільш оптимальною, причина цього, те що при зменшенні швидкості пошуку в вузліх оптимальниє є збільшення їх кількості, що в свою чергу веде до збільшення переходів, а в ієрархічній мережі порівняно з іншими набагато менше переходів необхідно зробити в середньому одному запиту для доступу до серверу (Рис. 4.8).

C = 3000; $t_{\text{пош}}$ = 0,002; $t_{\text{пер}}$ = 0,02; R = 3 (кількість зв’язків для ієрархічної мережі)

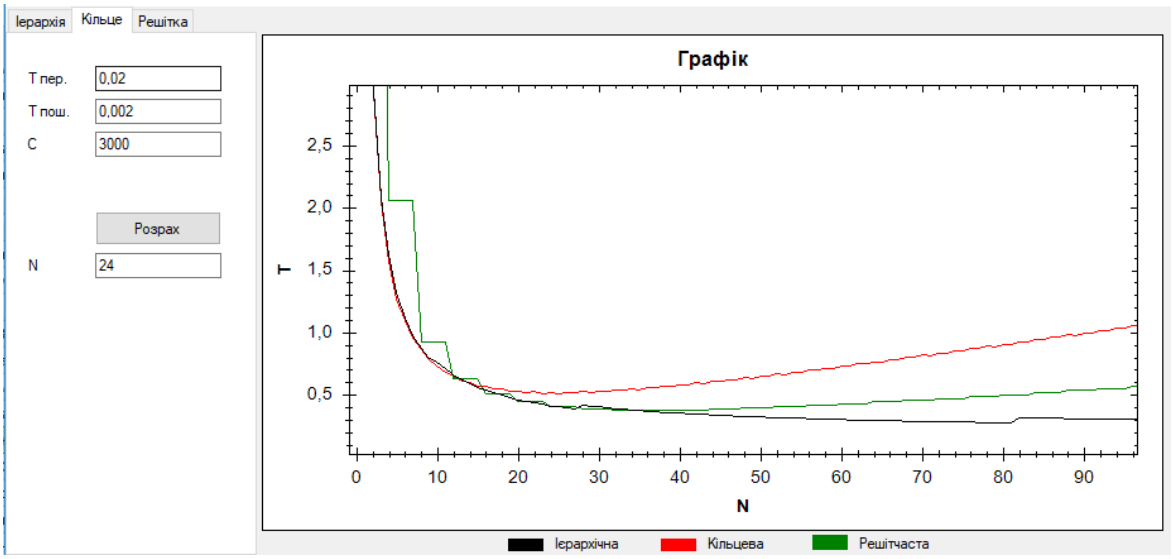

Рисунок 4.8 — Порівняння мереж при звичайних вхідних параметрах

3. Доволі цікаві результати результати при сутевому збільшенні швидкості обробки запиту на кінцевих серверах, в такому випадку оптимальною виявляється решітчаста, але зважіючи на те, що для її фізичної реалізації необхідно більше переходів і серверів ніж для реалізації кільцевої мережі, яка досягає свого мінімуму за 2 рази меншої кількості серверів ніж решітчаста, то цілком доцільно використовувати саме такий тип мережі (Рис. 4.9).

C = 3000; $t_{\text{пош}}$ = 0,00003; $t_{\text{пер}}$ = 0,02; R = 3 (кількість зв’язків для ієрархічної мережі)

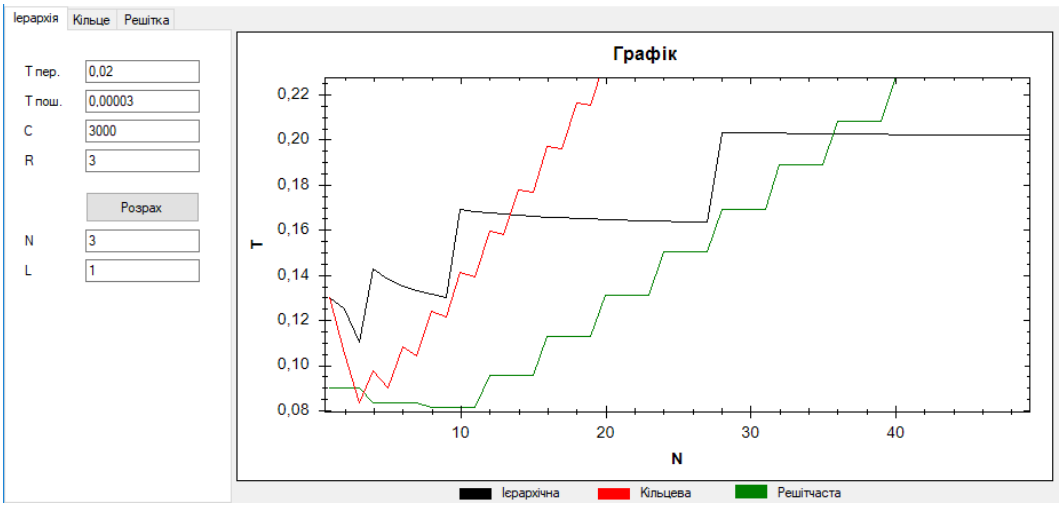

Рисунок 4.9 — Параметри при яких кільцева мережа проявляє себе найкраще

Висновки

В розділі було експериментально досліджено та проаналізовано всі типи мереж, порівняно їх між собою. Було спостережено, що ієрархічна мережа найкраще проявляє себе при великій кількості діних або при малій швидкості обробки інформації. Інші ж мережі проявляють себе краще при менших обьемах даних.

РОЗДІЛ 5 ФУНКЦІОНАЛЬНО-ВАРТІСНИЙ АНАЛІЗ ПРОГРАМНОГО ПРОДУКТУ

5.1 Резюме проекту

Даний бізнес-проект передбачає створення нової системи моделювання режимів роботи інтегрованих систем енергозабезпечення.

Розроблена система належить до сфери оптимізації управління ресурсами. Реалізація даної системи дозволить споживачу отримати конкретний набір основних функціональних елементів майбутньої системи енергозабезпечення, яка реалізує оптимальну модель забезпечення електричною енергією споживача. Це система керування, заснована на проведенні типових вимірювань і перевірок, що забезпечують таку роботу споживачів електричної енергії, при якій споживається тільки цілковито необхідна для користувача кількість енергії [1].

Також існує змога проаналізувати різні режими узгодженої роботи обладнання на протязі тижня для подальшої реалізації проекту. Даний продукт характеризується високою точністю в розрахунках та відносно простим для сприйняття інтерфейсом.

Було проведено аналіз програмного продукту у якості стартап проекту. Можна зазначити що у проекту є можливість комерціалізації, оскільки ринок технологій оптимізації витрат ресурсів динамічно розвивається, створюються нові додатки модульної структури, для яких необхідні додаткові можливості. Система може бути інтегрована в модульні системи і тісно взаємодіяти з ними.

Через те, що продукт є повністю програмним, його розробка не потребує витрат на різноманітні матеріали та обладнання.

Місце знаходження майбутньої організації планується у столиці країни де площі вже мають комунальне, енергетичне та інформаційне забезпечення.

Компанія гарантує підтримку користувачів продукту.

Можна сказати, що подальший розвиток проекту є доцільним, оскільки він знайде свою цільову аудиторію, а також зможе окупитися за пару років.

5.2 Організація проекту

Як відомо, кожне виробниче підприємство так чи інакше вирішує питання енергозбереження та раціонального використання паливно-енергетичних ресурсів. Енергетичний менеджмент є обов’язковим елементом в структурі підприємства, яке поставило собі за мету скорочення споживання енергетичних ресурсів, шляхом їх ефективного використання [2].

Питання скорочення витрат споживання електроенергетичних ресурсів актуальне не лише у сфері підприємців, проте і для звичайних власників будівель.

Зважаючи на актуальність поставленого питання та унікальність втілення рішення, відсутність аналогів програмних продуктів на ринку та нагальна потреба втілення проекту в життя за наявних проблем у суспільстві було вирішено створити окрему організацію, на базі якої планується реалізація окресленого проекту.

Бажання надати можливість кожному отримати чіткий перелік обладнання для подальшого проектування енергозабезпечення власної домівки являється головною метою створення організації.

Реалізація та використання розроблених рішень щодо застосування альтернативної електричної енергії дасть змогу удосконалити діючу в Україні методику встановлення відповідних тарифів на електроенергію, стимулювати

споживачів до здійснення контролю та керування потоками альтернативної енергії, що, в свою чергу, дозволить підтримувати окремі показники якості електроенергії в мережі на необхідному рівні.

5.3 Опис ідеї проекту та її технологічний аудит

Опис ідеї стартап проекту наведено у таблиці 5.1

Таблиця 5.1 – Опис ідеї стартап проекту

Зміст ідеї	Напрямки застосування	Вигоди для користувача
Система підтримки прийняття рішень для оцінки та прогнозування фінансових ризиків на фондовому ринку США.	Надання інвесторам інформації про ймовірні фінансові ризики.	Інвесторам: допомога у прийнятті рішення щодо інвестування.
	Надання користувачам інструментів аналізу і прогнозування динаміки зміни торгових ризиків на фондовому ринку США.	Гравцям на біржі: допомога при розробці стратегії торгівлі на біржі, набору позиції на ринку.
		Фінансовим аналітикам і консультантам: економія часу при аналізі потенційних позицій на ринку, порівняння отриманих даних щодо фін. Ризику, розрахованих на основі різних моделей.

Визначення сильних, слабких та нейтральних характеристик ідеї проекту продемонстровано у таблиці 5.2

Таблиця 5.2 – Визначення сильних, слабких та нейтральних характеристик ідеї проекту

№ п/п	Техніко-економічні характеристики ідеї	(потенційні) товари/концепції конкурентів			W (слабка сторона)	N (нейтральна сторона)	S (сильна сторона)
		Мій проект	prognoz.ru	Margincal.com			
1.	Ціна	Низька	Середня	Висока	+		
2.	Ефективність	Висока	Середня	Середня		+	
3.	Функціонал	Середній	Вузький	Широкий			+

Таблиця 5.3 – Технологічна здійсненність ідеї проекту

№ п/п	Ідея проекту	Технології її реалізації	Наявність технологій	Доступність технологій
1.	Розробка системи підтримки прийняття рішень для оцінки та прогнозування фінансових ризиків на фондовому ринку США.	Python	Наявна	Доступна
2.		MathLab	Наявна	Доступна
3.		MS Excel	Наявна	Доступна
Обрана технологія реалізації ідеї проекту: MS Excel				

5.4 Аналіз ринкових можливостей запуску стартап проекту

Таблиця 5.4 – Попередня характеристика потенційного ринку стартап проекту

№ п/п	Показники стану ринку (найменування)	Характеристика
1.	Кількість головних гравців, од	2
2.	Загальний обсяг продаж, грн/ум.од	8500000
3.	Динаміка ринку (якісна оцінка)	Зростає

Продовження таблиці 5.4 – Попередня характеристика потенційного ринку стартап проекту

4.	Наявність обмежень для входу (вказати характер обмежень)	Важкодоступність певних даних
5.	Специфічні вимоги до стандартизації та сертифікації	Немає
6.	Середня норма рентабельності в галузі (або по ринку), %	30%

Таблиця 5.5 – Характеристика потенційних клієнтів стартап проекту

№ п/п	Потреба, що формує ринок	Цільова аудиторія (цільові сегменти ринку)	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
1.	Потреба у єдиному джерелі даних, що впливають на динаміку котирувань цінних паперів.	Інвестори	Зацікавлені у своєчасній і якісній аналітиці	Вимоги до своєчасності і достовірності даних, що публікуються
2.		Трейдери	Потребують об’єктивну оцінку потенційних фінансових ризиків для формування позицій на ринку	
3.		Фінансові аналітики і консультанти	Потребують своєчасну інформацію для проведення аналізу	

Таблиця 5.6 – Фактори загроз

№ п/п	Фактор	Зміст загрози	Можлива реакція компанії
1.	Новий продукт	Потенційні користувачі з підозрою ставляться до нових продуктів	Поширення рекламної кампанії

Таблиця 5.7 – Фактори можливостей

№ п/п	Фактор	Зміст можливості	Можлива реакція компанії
1.	Потреба у єдиному джерелі даних, що впливають на динаміку котирувань цінних паперів	Інвестори, трейдери та фінансові аналітики потребують ефективної платформи для аналізу потенційних можливостей і загроз на фондовому ринку	Задоволені інвестори, трейдери та фінансові аналітики

Таблиця 5.8 – Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)
Тип конкуренції: олігополія	У сфері домінує невелика кількість подібних СППР	Поширення рекламної кампанії
За рівнем конкурентної боротьби: міжнаціональний	Наявна національна конкуренція	-
За галузевою ознакою: внутрішньогалузева	Наявна конкуренція в рамках одної галузі	-
Конкуренція за видами товарів: товарно-видова	Наявна конкуренція між схожими продуктами	Підвищення швидкості збору інформації, надання інструментів аналізу і прогнозування, збільшення кількості джерел інформації
За характером конкурентних переваг: нецінова	Наявна конкуренція завдяки розширенню бази даних та функціональних можливостей	Можливість вийти на ринок з недорогим продуктом
За інтенсивністю: немарочна	Наявна конкуренція, де роль торгової марки незначна	-

Таблиця 5.9 – Аналіз конкуренції в галузі за М. Портером

Складові аналізу	Прямі конкуренти в галузі	Потенційні конкуренти	Постачальники	Клієнти	Товари-замінники
	prognoz.ru; Margincall.com	Доступ до каналів розподілу	Компанії на фондовому ринку, портали новин, банки	Швидкість надання інформації, її достовірність	Ціна
Висновки	Висока інтенсивність конкурентної боротьби з боку прямих конкурентів	Є можливості входу в ринок. Потенційних конкурентів немає.	Постачальники є джерелами інформації для СППР.	Клієнти вимагають своєчасності та достовірності інформації.	Аналогічні продукти конкурентів є дорожчими, але їхній функціонал не є ширшим.

Таблиця 5.10 – Обгрунтування факторів конкурентоспроможності

№ п/п	Фактор конкурентоспроможності	Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проектів значущим)
1.	Ціна	Ціна запропонованого продукту нижча за ціни конкуруючих, але при цьому функціонал та якість інформації не поступається програмним продуктам конкурентів.
2.	Ефективність	За рахунок забезпечення своєчасності, повноти і достовірності інформації СППР дає можливість оцінити поточний стан на ринку, об’єктивно оцінити фінансові ризики та відкрити максимально вигідні позиції на фондовому ринку США.

Продовження таблиці 5.10 – Обгрунтування факторів конкурентоспроможності

3.	Поріг входження	Так як у сфері не так багато аналогічний програмних продуктів, шанси увійти на ринок високі.

Таблиця 5.11 – Порівняльний аналіз сильних та слабких сторін проекту

№ п/п	Фактор конкурентоспроможності	Бали 1-20	Рейтинг товарів-конкурентів у порівнянні з запропонованим продуктом						
			–3	–2	–1	0	+1	+2	+3
1.	Ціна	13	+						
2.	Ефективність	17						+	
3.	Поріг входження	18							+

Таблиця 5.12 – SWOT аналіз стартап проекту

Сильні сторони: Ефективність продукту Необхідність продукту Ціна продукту	Слабкі сторони: Невідомість продукту Вузьконаправленість продукту
Можливості: Охоплення широкої аудиторії зацікавлених користувачів – інвесторів, фінансових аналітиків, трейдерів. Підвищення ефективності роботи користувачів за рахунок розширення функціональних можливостей.	Загрози: Можлива незацікавленість продуктом через його специфічність і часту недовіру клієнтів до нових продуктів.

Таблиця 5.13 – Альтернативи ринкового впровадження стартап проекту

№ п/п	Альтернатива (орієнтовний комплекс заходів) ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
1.	Розробка програмного забезпечення та грамотна маркетингова програма	Велика	4-5 місяців

5.5 Розробка ринкової стратегії проекту

Таблиця 5.14 – Вибір цільових груп потенційних клієнтів

№ п/п	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
1.	Інвестори	Значна готовність	Високий	Низька	Середня
2.	Трейдери	Значна готовність	Високий	Низька	Висока
3.	Фінансові аналітики і консультанти	Значна готовність	Високий	Низька	Висока
Які цільові групи обрано: Інвестори, трейдери, фінансові консультанти та аналітики					

Таблиця 5.15 – Визначення базової стратегії розвитку

№ п/п	Обрана альтернатива розвитку проекту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку
1.	Розробка програмного забезпечення та грамотна маркетингова програма	За рахунок потреби в широкофункціональному продукті	Ефективність	Стратегія спеціалізації

Таблиця 5.16 – Визначення базової стратегії конкурентної поведінки

№ п/п	Чи є проект «першопрохідцем» на ринку?	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, і які?	Стратегія конкурентної поведінки
1.	Ні	Шукати нових споживачів і забирати існуючих у конкурентів	Буде розроблений продукт із ширшим функціоналом і повнішою інформаційною базою	Стратегія заняття конкурентної ніші.

Таблиця 5.17 – Визначення стратегії позиціонування

№ п/п	Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартап-проекту	Вибір асоціацій, які мають сформувати комплексну позицію власного проекту (три ключових)
1.	Необхідність СППР з повною, достовірною і своєчасною інформацією та інструментарієм для аналізу ринку	Стратегія спеціалізації	Ефективність	Висока ефективність Простота у використанні

5.6 Розробка маркетингової програми стартап проекту

Таблиця 5.18 – Визначення ключових переваг концепції потенційного товару

№ п/п	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити
1.	Потреба в ефективному продукті	Пропонує ефективний продукт	Ефективність товару вище ніж ефективність товару конкурентів

Таблиця 5.19 – Опис трьох рівнів моделі товару

<table>
<tr><td>Рівні товару</td><td colspan="3">Сутність та складові</td></tr>
<tr><td>I. Товар за задумом</td><td colspan="3">Система підтримки прийняття рішень для оцінки та прогнозування фінансових ризиків на основі різних моделей та підходів на фондовому ринку за США.</td></tr>
<tr><td rowspan="5">II. Товар у реальному виконанні</td><td>Властивості/характеристики</td><td>М/Нм</td><td>Вр/Тх /Тл/Е/Ор</td></tr>
<tr><td>1. Мультиплатформенність
2. Зручний інтуітивний інтеріейс</td><td>-</td><td>-</td></tr>
<tr><td colspan="3">Якість: стандарти ефективності</td></tr>
<tr><td colspan="3">Пакування: електронне розповсюдження</td></tr>
<tr><td colspan="3">Марка: JMarketRisk</td></tr>
<tr><td rowspan="2">III. Товар із підкріпленням</td><td colspan="3">До продажу: -</td></tr>
<tr><td colspan="3">Після продажу: технічна підтримка</td></tr>
<tr><td colspan="4">За рахунок чого потенційний товар буде захищено від копіювання: захист інтелектуальної власності</td></tr>
</table>

Таблиця 5.20 – Визначення меж встановлення ціни

№ п/п	Рівень цін на товари-замінники	Рівень цін на товари-аналоги	Рівень доходів цільової групи споживачів	Верхня та нижня межі встановлення ціни на товар/послугу
1.	-	9500000-63000000 грн.	30000+ грн.	10000-20000 грн.

Таблиця 5.21 – Формування системи збуту

№ п/п	Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
1.	Закупівля через інтернет	Підтримка нормального функіонування сайту	0	Електронне розповсюдження

Таблиця 5.22 – Концепція маркетингових комунікацій

№ п/п	Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуютьс я цільові клієнти	Ключові позиції, обрані для позиціонува ння	Завдання рекламного повідомленн я	Концепція рекламного звернення
1.	Потреба в ефективном у продукті	Інтернет-мережі	Висока ефективніст ь Простота у використанн і	Провести якісну маркетингов у кампанію	Донести специфіку продукту

Висновки

Отже, існує можливість ринкової комерціалізації проекту, так як існує попит, наявна динаміка ринку та рентабельність роботи на маркеті. Звичайно, існують перспективи впровадження проекту з огляду на потенційні групи клієнтів, бар’єри входження, поточну конкуренцію та конкурентноспроможність стартапу. Раціонально обрати альтернативу розробку програмного забезпечення та грамотну маркетингову програму для ринкової реалізації проекту. Вважаємо, що подальша імплементація проекту доцільна.

ВИСНОВКИ

В даній роботі було проаналізовано задачу мінімізації шляху між сторонами одного кута, який проходить через задану точку в цьому куті. В ході дослідження було розроблено 3 математичні моделі які з різних сторін по різному відтворюють дійсну постановку задачі.

В першому розділі були проаналізовані основні існуючу на данний момент класифікацію методів зберігання даних в розподілених базах данних, переваги і недоліки кожного з них. Також було проаналізовано нюанси пошуку інформації в складних розподілених мережах, що в нашому випадку допомогло краще побудувати математичні моделі для кожно з типів мереж.

В другому розділі було описано всі побудовані математичні моделі, кожна з яких описує принципово різну мережу і моделює середій час який потрібно запратити запиту для знаходження інформації.

Третій розділ був присвячений опису програмного продукту, який був створений для мінімізації знайдених в другому розділі функцій. Програма була написана в середовищі VisualStudio з використанням мови програмування C#.

В четвертому розділі було представлені експерименти які проводились з отриманими моделями за допомогую реалізованого продукту. Єксперимети показали, що ієрархічна мережа найкраще проявляе себе при великій кількості діних або при малій швидкості обробки інформації. Інші ж мережі проявляють себе краще при менших обьемах даних.

В п’ятому розділі представлені економічні розрахунки для даного програмного продукту, підраховані можливі економічні дивіденди для компаній які будуть застосовувати дану розробку.

ПЕРЕЛІК ПОСИЛАНЬ

1. Моделювання за методом Монте-Карло [Електронний ресурс]. – Режим доступу http://www.palisade.com/risk/ru/monte_carlo_simulation.asp
2. Основні принципи управління ризиком[Електронний ресурс]. – Режим доступу http://stud.com.ua/19854/strahova_sprava/osnovni_printsipi_upravlinnya
3. Поняття й класифікація фінансових ризиків [Електронний ресурс]. – Режим доступу http://www.finansystam.ru/uk/content/ponyattya-y-klasifikaciya-finansovih-rizikiv
4. Управління ризиками, системний аналіз і моделювання. [Електронний ресурс]. – Режим доступу http://stud.com.ua/24511/menedzhment/upravlinnya_rizikami_sistemniy_analiz_i_modelyuvannya
5. Управління фінансовими ризиками. сутність і класифікація фінансових ризиків підприємства. [Електронний ресурс]. – Режим доступу http://pidruchniki.com/1640022143678/finansi/finansoviy_menedzhment_dodatkovi_rozdili
6. Кирюшкин В. Основы риск-менеджмента [Текст] / В. Кирюшкин, И. Ларионов. – М.: Анкил, 2009. – 132 с.
7. Балабанов И. Риск-менеджмент [Текст] / И. Балабанов. – М.: Финансы и статистика, 1996. – 192 с.
8. Методичні вказівки з інспектування банків «Система оцінки ризиків» [Електронний ресурс]. – Режим доступу http://zakon3.rada.gov.ua/laws/show/v0104500-04
9. Методичні рекомендації щодо організації та функціонування систем ризик-менеджменту в банках України [Електронний ресурс]. – Режим доступу http://zakon5.rada.gov.ua/laws/show/v0361500-04

10. Amendment to the Capital Accord to incorporate market risks [Електронний ресурс]. – Режим доступу http://www.bis.org
11. Киселев В. Управление банковским капиталом: теория и практика [Текст] / В. Киселев. – М.: Экономика, 1997. – 192 с.
12. Ульянова М. Управление рыночным риском [Текст] / М. Ульянова // Молодой ученый. - 2014. - № 21.2. - С. 99-102.
13. Jorion Ph. Finansial risk-management: Second edition [Text] / Ph. Jorion. – Hobokon, New Jersey: John Wiley & Sons Ltd., 2003. – 708 p.
14. Примостка Л. Управління банківськими ризиками [Текст] / Л. Примостка, П. Чуб, Т. Карчева. – К: КНЕУ, 2007. – 600 с.
15. Poon S.-H. A practical guide for forecasting financial market volatility [Text] / Ser Huang Poon. – Chichester: John Wiley & Sons Ltd., 2005. – 238 p.
16. Giraitis L. Recent advances in ARCH modelling [Text] / Liudas Giraitis, Remigijus Leipus, Donatas Surgailis // Econometric Theory. – 2013. - №17. – С. 608-631.
17. Xekalaki E. ARCH Models for Financial Applications [Text] / Evdokia Xekalaki, Stavros Degiannakis. – Chichester: John Wiley & Sons Ltd., 2010. – 550 p.
18. Застосування методу Монте-Карло при оцінюванні фінансовиxx ризиків [Електронний ресурс]. – Режим доступу https://sibac.info/studconf/econom/xxvii/40330
19. Еволюція мір ризику інвестиційного портфеля. Сучасні модицікації мір ризику VaR [Електронний ресурс]. – Режим доступу http://www.beintrend.ru/evolution-var-value-at-risk
20. Класифікація фінансових ризиків [Електронний ресурс]. – Режим доступу http://studopedia.com.ua/1_251858_klasifikatsiya-finansovih-rizikiv.html
21. Ймовірносні моделі аналізу ризиків [Електронний ресурс]. – Режим доступу https://www.cfin.ru/finanalysis/monte_carlo2.shtml

ДОДАТОК А ЛІСТИНГ ПРОГРАМИ

```
using System;
using System.Collections.Generic;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Windows.Forms;
using ZedGraph;

namespace Lab_1
{
    public static class GrafParams
    {
        public static double Step = 1;
        public static Func<double, double> MaxNFunc = (a) => { return a * 3.3; };


        public static void FillGraf(ZedGraphControl graf, Result result)
        {
                    var panel = graf.GraphPane;
                    Color color;
                    switch (result.CalcType.Name)
                    {
                           case "Hierarhy":
                                  color = Color.Black;
                                  break;
                           case "Circle":
                                  color = Color.Red;
                                  break;
                           case "Cube":
                                  color = Color.Green;
                                  break;
                           default:
                                  color = Color.Black;
                                  break;
                    }
                    var listCurv =  panel.CurveList.Where(w => w.Color == color);
                    foreach (var curv in listCurv)
                    {
                           curv.Clear();
                    }
                    //panel.CurveList.Clear();
                    panel.GraphObjList.Clear();
            panel.Title.Text = "Графік";
            panel.XAxis.Title.Text = "N";
            panel.YAxis.Title.Text = "T";



            var max = result.Graf.Values.Where(x => !double.IsPositiveInfinity(x)).Max();

            // Установим цвет осей
            panel.XAxis.Color = Color.Gray;
            panel.YAxis.Color = Color.Gray;

                    // Включим сетку

                    PointPairList list = new PointPairList();

            foreach (var n in result.Graf.Keys)
            {
                list.Add(n, result.Graf[n]);
```

```
                }


                panel.AddCurve("", list, color, SymbolType.None);

                    panel.Title.FontSpec.FontColor = Color.Black;
            panel.Title.FontSpec.IsBold = true;

            graf.AxisChange();

            graf.Invalidate();
        }
    }

    internal abstract class AParams
    {
        public double TimeToSearch { get; set; }

        public double TimeToRun { get; set; }

        public double AmoutOfRecords { get; set; }

        public int R { get; set; }
    }

    public interface IFiller
    {
        void Fill(Form form);
    }
    public class Result
    {
        public double N;
        public double L;
        public Dictionary<double, double> Graf;
             public Type CalcType;
             //public int MaxN;
    }

    class Hierarhy : AParams, IFiller
    {
             private double Funk(double n)
             {
                 return (AmoutOfRecords * TimeToSearch / n) + (Math.Ceiling((Math.Log(n)
/ Math.Log(R)) + 1) * 2 * TimeToRun);
             }
        private Result calc()
        {

                 var n_opt = (int)Math.Truncate(
                     AmoutOfRecords *
                     TimeToSearch *
                     Math.Log(R) /
                     (2*TimeToRun));//todo formula

            var res = new Result()
            {
                N = n_opt,
                L = (int) (Math.Ceiling(Math.Log(n_opt)/Math.Log(R)))
            };
                 res.Graf = new Dictionary<double, double>();

            for (double n = 1; n < Form1.MaxN; n += GrafParams.Step)
```

```
                {
            res.Graf.Add(n, Funk(n));
                }
                if (res.L >= 1)
                {
                    var n1 = (int)Math.Pow(R, res.L);
                    var n2 = (int)Math.Pow(R, res.L - 1);
                    n_opt = Funk(n1) < Funk(n2) ? n1 : n2;
                    res.N = n_opt;
                    res.L = (int)(Math.Ceiling(Math.Log(n_opt) / Math.Log(R)));
                }
                res.CalcType = this.GetType();
                return res;
        }

        public void Fill(Form form)
        {
            var res = calc();
            var N = form.Controls.Find("N_h", true).First() as TextBox;
            var L = form.Controls.Find("L_h", true).First() as TextBox;
            var graf = form.Controls.Find("g", true).First() as ZedGraphControl;

            N.Text = res.N.ToString();
            L.Text = res.L.ToString();
            GrafParams.FillGraf(graf, res);

        }
    }

    class Circle : AParams, IFiller
    {
        private Result calc()
        {
            var n_opt = (int)Math.Sqrt((2 * (AmoutOfRecords * TimeToSearch - TimeToRun)) /
TimeToRun);
            var res = new Result()
            {
                N = n_opt
            };
            res.Graf = new Dictionary<double, double>();

            for (double n = 1; n < Form1.MaxN; n += GrafParams.Step)
            {
                res.Graf.Add(n,
                    (AmoutOfRecords * TimeToSearch / n) + (((Math.Floor((double)n / 2) + 1)
* (n + 1)) * TimeToRun / n));
            }
                    res.CalcType = this.GetType();
                    return res;
        }

        public void Fill(Form form)
        {
            var res = calc();
            var N = form.Controls.Find("N_c", true).First() as TextBox;
            var graf = form.Controls.Find("g", true).First() as ZedGraphControl;
            N.Text = res.N.ToString();
            GrafParams.FillGraf(graf, res);
        }
    }

       class Cube : AParams, IFiller
       {
```

```
		private double Funk(double l)
		{
			return Math.Abs( (AmoutOfRecords * TimeToSearch / (4*l - 1)) +
				((4*l*TimeToRun*(l+1))/(4*l - 1)));
		}
		private Result calc()
		{
			var l_opt = (int)(
				(Math.Sqrt(5*TimeToRun*TimeToRun + 4 * AmoutOfRecords * TimeToRun
* TimeToSearch) + TimeToRun)/
				(4 * TimeToRun));

			var res = new Result() { L = l_opt};
			res.Graf = new Dictionary<double, double>();

			for (double n = 1; n < Form1.MaxN/*GrafParams.MaxNFunc(l_opt*4)*/; n +=
GrafParams.Step)
			{
				res.Graf.Add(n, Funk((int)(n / 4)));
			}
			if (l_opt > 0)
			{
				res.L = Funk(l_opt) < Funk(l_opt + 1) ? l_opt : l_opt + 1;
			}
			res.CalcType = this.GetType();
			return res;
		}

		public void Fill(Form form)
		{
			var res = calc();
			var L = form.Controls.Find("L_k", true).First() as TextBox;
			var graf = form.Controls.Find("g", true).First() as ZedGraphControl;

			L.Text = res.L.ToString();
			GrafParams.FillGraf(graf, res);
		}
	}
}
```