НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Навчально-науковий інститут прикладного системного аналізу
Кафедра системного проектування

«На правах рукопису»
УДК _____

До захисту допущено:
Завідувач кафедри
_____ Вадим МУХІН
«___»_____2021 р.

Магістерська дисертація

на здобуття ступеня магістра
за освітньо-професійною програмою
“Інтелектуальні сервіс-орієнтовані розподілені обчислювання”
зі спеціальності 122 "Комп'ютерні науки"
на тему: «Моделі та методи організації та управління гетерогенними розподіленими базами даних з динамічною структурою»

Виконав (-ла):
студент (-ка) ІІ курсу, групи ДА-01мп
Проскурка Даниіл Миколайович _____
Науковий керівник:
Професор, д.т.н.
Мухін Вадим Євгенійович _____
Рецензент:
Професор кафедри інформаційних систем та технологій, д.т.н
Корнага Ярослав Ігорович _____

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

Київ – 2021

Національний технічний університет України
«Київський політехнічний інститут
імені Ігоря Сікорського»

Інститут/факультет Прикладного системного аналізу
(повна назва)

Кафедра Системного проектування
(повна назва)

Рівень вищої освіти – другий (магістерський) за освітньо-професійною програмою

Спеціальність (спеціалізація) Комп`ютерні науки
(код і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ __В.Є.Мухін._____
(підпис) (ініціали, прізвище)
«___»_____2021__ р.

ЗАВДАННЯ
на магістерську дисертацію студенту
(практика)
Проскурка Даниіл Миколайович
(прізвище, ім'я, по батькові)

1. Тема дисертації Моделі та методи організації та управління гетерогенними розподіленими базами даних з динамічною структурою
науковий керівник дисертації Мухін Вадим Євгенійович, д.т.н., професор,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «___»_____ 20__ р. №_____

2. Строк подання студентом дисертації 17 грудня 2021

3. Об'єкт дослідження
Гетерогенні розподілені бази даних з динамічною структурою.

4. Предмет дослідження (Вихідні дані – для магістерської дисертації за освітньо-професійною програмою)
Модель та методи організації і управління розподілених баз даних.

5. Перелік завдань, які потрібно розробити

Розглянути моделі та методи організації та управління гетерогенних баз даних з динамічною структурою
Розробити метод ефективного розподілення фрагментів баз даних
Порівняти та оцінити ефективність розробленого методу розподілення фрагментів бази даних

6. Перелік графічного (ілюстративного) матеріалу

Презентація

7. Орієнтовний перелік публікацій

8. Консультанти розділів дисертації[1*]

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

9. Дата видачі завдання _____

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Строк виконання етапів магістерської дисертації	Примітка
	Утвердження списку досліджуваних моделей розподілення баз даних	17 жовтня 2021	
	Розглядання алгоритмічних підходів вирішення проблеми розподілення баз даних у динамічному середовищі	10 листопада 2021	
	Розробка методу розподілення фрагментів баз даних у динамічній системі	25 листопада 2021	
	Захист дипломної роботи	17 грудня 2021	

Студент _____ _____Проскурка Д.М___
(підпис) (ініціали, прізвище)

Науковий керівник дисертації _____ _Мухін В. Є._____
(підпис) (ініціали, прізвище)

[1*] Консультантом не може бути зазначено наукового керівника

Реферат на магістерську дисертацію

виконано на тему: «Моделі та методи організації та управління гетерогенними розподіленими базами даних з динамічною структурою»

студентом: Проскуркою Даниілом Миколайовичем

Магістерська дисертація виконана на сторінках, містить 9 ілюстрацій та 42 таблиці. При підготовці дисертації використано літературу з 25 джерел.

Актуальність теми

Ефективне функціонування будь-якої системи розподілених баз даних «DDBS» сильно залежить від її правильного дизайну з точки зору прийнятих методів фрагментації та розподілу. Однак оптимальні методи проектування фрагментації та розподілу можуть бути дуже складними і потребують хорошого досвіду та знань для досягнення. Фрагментація великих глобальних баз даних виконується шляхом поділу відносин бази даних по горизонталі, вертикалі або як комбінація обох. Для того, щоб системи розподілених баз даних працювали ефективно, ці фрагменти мають бути розподілені між доступними сайтами таким чином, щоб зменшити витрати на зв’язок, тобто мінімізувати загальний обсяг даних, що передаються під час виконання запитів на сайтах.

Мета і завдання дослідження

Метою даної роботи є розроблення методів розподілення фрагментів бази даних, щоб системи розподілених баз даних працювали ефективно, на основі розподілення фрагменти розподілених між доступними сайтами таким чином, для підвищення швидкості передачі.

Рішення поставлених завдань і досягнуті результати

У цій дисертації представлена нова модель перерозподілу даних для реплікованих і нереплікованих обмежених DDBS шляхом внесення змін до

шаблону доступу до даних. Цей підхід передбачає, що розподіл фрагментів по сайтах мережі спочатку виконувався відповідно до належним чином прогнозованого набору значень частоти запитів, які можна було б використовувати на сайтах. Метод враховує обмеження сайтів на етапі перерозподілу. Він пропонує ефективний план перерозподілу фрагментів даних між сайтами на основі зв’язку та оновлення значень вартості для кожного фрагмента окремо. Процес перерозподілу буде виконуватися шляхом вибору максимального значення вартості оновлення для кожного фрагмента та відповідного перерозподілу. Результати експерименту підтвердили, що запропонована методика буде ефективно сприяти вирішенню проблеми перерозподілу фрагментів у середовищі динамічних розподілених реляційних баз даних.

Об’єкт досліджень

Гетерогенні розподілені бази даних з динамічною структурою.

Предмет досліджень

Модель та методи організації і управління розподілених баз даних.

Наукова новизна

Пропонується метод ефективного розподілення фрагментів розподілених баз даних у динамічному середовищі.

Практичне значення отриманих результатів

Отримані результати дослідження можуть використовуватися для подальшого розвитку систем розподілених баз даних та правильної розробки нових проектів. У майбутньому цей проект має стати проектом, який дає надію стати прибутковим та унікальним стартапом.

Ключові слова

Розподілена база даних, розподіл фрагментів, евристичний алгоритм, алгоритм перерозподілу

Реферат на магистерскую диссертацию

выполнена на тему: «Модели и методы организации и управления гетерогенными распределенными базами данных с динамической структурой»

студентом: Проскуркой Даниилом Николаевичем

Магистерская диссертация выполнена на страницах, содержит 9 иллюстраций и 42 таблицы. При подготовке диссертации использована литература из 25 источников.

Актуальность темы

Эффективное функционирование любой системы распределенных баз данных DDBS сильно зависит от ее правильного дизайна с точки зрения принятых методов фрагментации и распределения. Однако оптимальные методы проектирования фрагментации и распределения могут быть очень сложными и требуют хорошего опыта и знаний для достижения. Фрагментация больших глобальных баз данных производится путем разделения отношений базы данных по горизонтали, вертикали или как комбинация обоих. Для того, чтобы системы распределенных баз данных работали эффективно, эти фрагменты должны быть распределены между доступными сайтами таким образом, чтобы уменьшить расходы на связь, то есть минимизировать общий объем данных, передаваемых при выполнении запросов на сайтах.

Целью и задачей исследования

Целью данной работы является разработка методов распределения фрагментов базы данных, чтобы системы распределенных баз данных работали эффективно, на основе распределения фрагменты распределенных между доступными сайтами таким образом, для повышения скорости передачи.

Решения поставленных задач и достигнутые результаты

В этой диссертации представлена новая модель перераспределения данных для репликированных и нерепликированных ограниченных DDBS путем внесения изменений в шаблон доступа к данным. Этот подход предполагает, что распределение фрагментов по сайтам сети первоначально выполнялось в соответствии с должным образом прогнозируемым набором значений частоты запросов, которые можно использовать на сайтах. Метод учитывает ограничение сайтов на этапе перераспределения. Он предлагает эффективный план перераспределения фрагментов данных между сайтами на основе связи и обновления значений стоимости каждого фрагмента отдельно. Процесс перераспределения выполняется путем выбора максимального значения стоимости обновления для каждого фрагмента и соответствующего перераспределения. Результаты эксперимента подтвердили, что предложенная методика будет эффективно способствовать решению проблемы перераспределения фрагментов в динамических распределенных реляционных баз данных.

Объект исследований

Гетерогенные распределенные базы данных с динамической структурой.

Предмет исследований

Модель и методы организации и управления распределенных баз данных.

Научная новизна

Предлагается способ эффективного распределения фрагментов распределенных баз данных в динамической среде.

Практическое значение получаемых результатов

Полученные результаты исследования могут использоваться для дальнейшего развития систем распределенных баз данных и правильной разработки новых проектов. В будущем этот проект должен стать проектом, который дает надежду стать прибыльным и уникальным стартапом.

Ключевые слова

Распределенная база данных, распределение фрагментов, эвристический алгоритм, алгоритм перераспределения

The abstract for the master's dissertation is

made on the topic: "Models and methods of organization and management of heterogeneous distributed databases with dynamic structure"

student: Danyil Proskurka

Master's dissertation is made on pages, contains 9 illustrations and 42 tables. Literature from 25 sources was used in the preparation of the dissertation.

Relevance of the topic

The effective functioning of any system of distributed databases "DDBS" strongly depends on its proper design in terms of accepted methods of fragmentation and distribution. However, the best methods of designing fragmentation and distribution can be very complex and require good experience and knowledge to achieve. Fragmentation of large global databases is performed by dividing database relationships horizontally, vertically, or a combination of both. In order for distributed database systems to work effectively, these fragments must be distributed among the available sites in such a way as to reduce communication costs, ie to minimize the total amount of data transmitted during queries on the sites.

Aim and objectives of the study

The aim of this work is to develop methods for distributing database fragments so that distributed database systems work efficiently, based on the distribution of fragments distributed between available sites in such a way as to increase the transfer rate.

Solutions to the tasks and results achieved

This dissertation presents a new model of data redistribution for replicated and non-replicated limited DDBS by making changes to the data access template. This approach assumes that the distribution of snippets across network sites was

initially performed according to a properly predicted set of query frequency values that could be used on the sites. The method takes into account the limitations of sites at the stage of redistribution. It offers an effective plan for redistributing fragments of data between sites based on communication and updating value values for each fragment separately. The redistribution process will be performed by selecting the maximum value of the update cost for each fragment and the corresponding redistribution. The results of the experiment confirmed that the proposed technique will effectively help solve the problem of redistribution of fragments in the environment of dynamic distributed relational databases.

Object of research

Heterogeneous distributed databases with dynamic structure.

Subject of research

Model and methods of organization and management of distributed databases.

Scientific novelty

The method of efficient distribution of fragments of distributed databases in a dynamic environment is offered.

Practical significance of the obtained results

The obtained research results can be used for further development of distributed database systems and proper development of new projects. In the future, this project should become a project that hopes to become a profitable and unique startup.

Keywords

Distributed database, fragment distribution, heuristic algorithm, redistribution algorithm.

Зміст

Перелік умовних скорочень

DDBS - distributed database system

DDBMS - distributed database management system

ACID - atomicity consistency isolation durability

ANSI - American National Standards Institute

SPARC - Scheme for Promotion of Academic and Research Collaboration

DDB - distributed database

СУБД - система управління базами даних

БД - база даних

Вступ

Основні системи обробки, які використовують бази даних, прагнуть об’єднати всі дані, що обробляються в компанії, з єдиною метою та забезпечити контрольований доступ до них. Потужні компанії часто мають велику кількість підрозділів, які фізично можуть бути віддалені один від одного на велику кількість і навіть тисячі кілометрів.

Кожен з цих підрозділів має свою локальну базу даних. Коли ці бази мають різних архітекторів і використовують різні протоколи зв’язку, вони вважаються інформаційними загонами, до яких іншим важко отримати доступ. У цьому випадку виникає необхідність об’єднання небезпечних баз даних в логічне ціле, тобто створення спеціальних баз даних.

Розділена база даних — це циклічність структурно та логічно пов’язаних локальних баз даних або частин бази даних, які розподілені на декілька географічно розподілених комп’ютерних вузлів і які в цілому мають відповідні функції адміністрування бази даних. Таким чином, розподілена база даних реалізується в різних адміністративних правопорушеннях для очищення разом з організаційними, технічними та програмними методами їх створення та ведення.

Однією з найактуальніших проблем при створенні таких розподілених баз даних є необхідність скорочення часу відповіді на запити на обробку даних.

З урахуванням того, що дані географічно розподілені за неоднорідними ресурсами. Для вирішення цієї проблеми використовуються механізми розподілу ресурсів гетерогенних комп'ютерних систем з їх адаптацією до особливостей обробки даних у базах даних.

Для вирішення поставленої задачі в дипломній роботі були поставлені та опрацьовані такі завдання: детально розглянути методи розв’язання даних у великих базах даних, створити математичні моделі, проаналізувати швидкість листування в запитах для різних типів комп’ютерних мереж. , проаналізувати створені за допомогою базового програмного забезпечення отримані експериментальні дані, визначити оптимальні

Конфігурації ресурсів для зберігання даних у розподілених базах даних Предметом дослідження є авторизації гетерогенних баз даних.

Дослідження зосереджено на методах оптимального розподілу даних у певних базах даних.

Метою роботи є створення моделей та механізмів, що забезпечують оптимальне розділення даних у гетерогенно розподілених базах даних з огляду на швидкість порівняння із запитами.

Методи дослідження: прикладні методи математичного аналізу, методи оптимізації та методи індукції та аналізу математичних моделей аналізу експериментальних даних

Задача: Визначити оптимальну (з точки зору коефіцієнта відповідності на запит) конфігурацію ресурсів гетерогенно розподілених баз даних та визначити найменшу можливу кількість серверів, які слід використовувати для управління певним обсягом інформації в базі даних.

Розділ 1. Порівняльний аналіз існуючих методів поширення даних в гетерогенних розподілених баз даних з динамічною структурою

1.1 Особливості реалізації розподілених базах даних

Розподілені бази даних використовуються для горизонтального масштабування, і вони розроблені, щоб задовольнити вимоги робочого навантаження без необхідності вносити зміни в програму бази даних або вертикальне масштабування однієї машини.

Розподілені бази даних вирішують різні проблеми, такі як доступність, відмовостійкість, пропускна здатність, затримка, масштабованість та багато інших проблем, які можуть виникнути при використанні однієї машини та однієї бази даних.

Розподілена база даних являє собою кілька взаємопов’язаних баз даних, розкиданих на кількох сайтах, з’єднаних мережею. Оскільки всі бази даних підключені, вони відображаються для користувачів як єдина база даних.

Розподілені бази даних використовують декілька вузлів. Вони масштабуються горизонтально і розвивають розподілену систему. Більше вузлів у системі забезпечують більшу обчислювальну потужність, забезпечують більшу доступність та вирішують проблему єдиної точки збою.

Різні частини розподіленої бази даних зберігаються в кількох фізичних місцях, а вимоги до обробки розподіляються між процесорами на кількох вузлах бази даних.

Централізована система управління розподіленою базою даних (DDBMS) керує розподіленими даними так, ніби вони зберігаються в одному фізичному місці. DDBMS синхронізує всі операції з даними між базами даних і

гарантує, що оновлення в одній базі даних автоматично відображаються на базах даних на інших сайтах.

Деякі загальні особливості розподілених баз даних:

- Незалежність розташування – дані фізично зберігаються на кількох сайтах і керуються незалежною DDBMS.
- Розподілена обробка запитів. Розподілені бази даних відповідають на запити в розподіленому середовищі, яке керує даними на кількох сайтах. Запити високого рівня перетворюються на план виконання запитів для спрощення керування.
- Управління розподіленими транзакціями — забезпечує узгоджену розподілену базу даних за допомогою протоколів фіксації, методів розподіленого контролю паралельності та розподілених методів відновлення на випадок багатьох транзакцій і збоїв.
- Безпроблемна інтеграція. Бази даних у колекції зазвичай являють собою єдину логічну базу даних, і вони взаємопов’язані.
- Мережне зв’язування – усі бази даних у колекції пов’язані мережею та взаємодіють одна з одною.
- Обробка транзакцій. Розподілені бази даних включають обробку транзакцій, яка є програмою, що включає набір однієї або кількох операцій з базою даних. Обробка транзакцій - це атомарний процес, який виконується повністю або не виконується взагалі.

Існує два типи розподілених баз даних:

- Однорідні
- Гетерогенні

Однорідна розподілена база даних — це мережа ідентичних баз даних, що зберігаються на кількох сайтах. Сайти мають однакову операційну систему, DDBMS та структуру даних, що робить їх легко керованими.

Однорідні бази даних дозволяють користувачам безперешкодно отримувати доступ до даних з кожної з баз даних.

На малюнку 1 показано приклад однорідної бази даних.

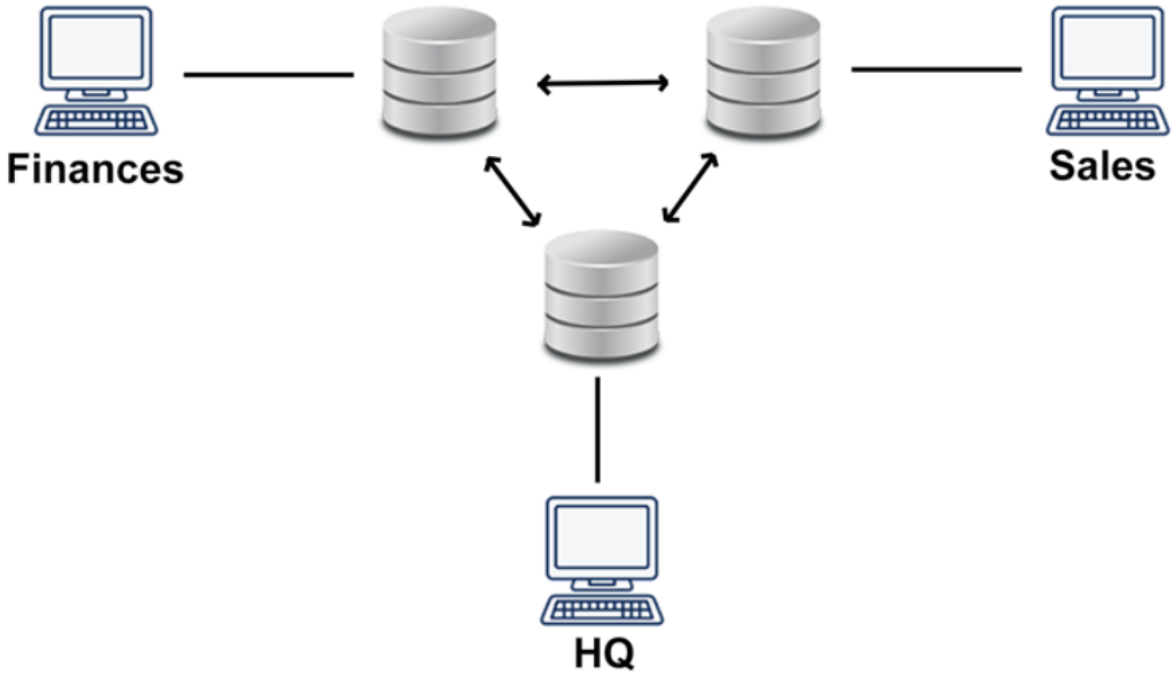

Рисунок 1 - Схема однорідної розподіленої бази даних

Гетерогенна розподілена база даних використовує різні схеми, операційні системи, DDBMS та різні моделі даних.

У разі неоднорідної розподіленої бази даних певний сайт може повністю не знати про інші сайти, що спричиняє обмежену співпрацю в обробці запитів користувачів. Обмеження полягає в тому, що для встановлення зв’язку між сайтами потрібні переклади.

На малюнку 1.2 показано приклад гетерогенної бази даних.

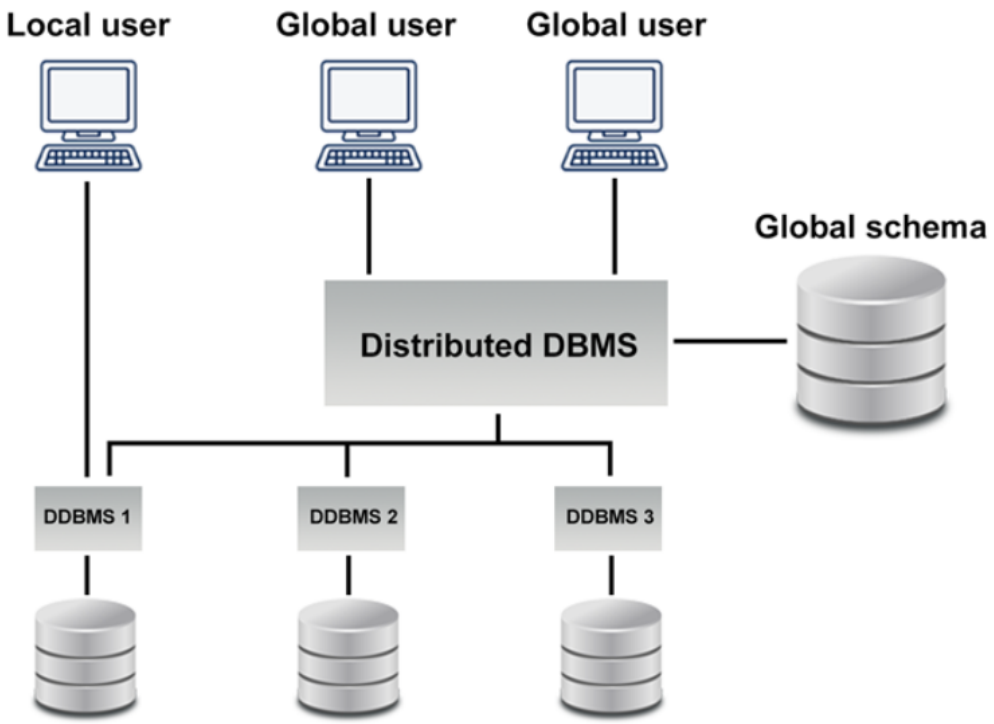

Рисунок 2 - Схема гетерогенної розподіленої бази даних

1.2 Механізми розподіленого зберігання даних

Розподілена база даних керується двома способами:

- Реплікація
- Фрагментація

Реплікація

При реплікації бази даних системи зберігають копії даних на різних сайтах. Якщо ціла база даних доступна на кількох сайтах, це повністю надлишкова база даних. Зображено на малюнку 3.

Перевага реплікації бази даних полягає в тому, що вона підвищує доступність даних на різних сайтах і дозволяє обробляти паралельні запити запитів.

Однак реплікація бази даних означає, що дані вимагають постійного оновлення та синхронізації з іншими сайтами для підтримки точної копії бази даних. Будь-які зміни, внесені на одному сайті, мають бути записані на інших сайтах, інакше виникнуть невідповідності.

Постійні оновлення спричиняють значні витрати на сервер і ускладнюють контроль одночасності, оскільки багато одночасних запитів потрібно перевіряти на всіх доступних сайтах.

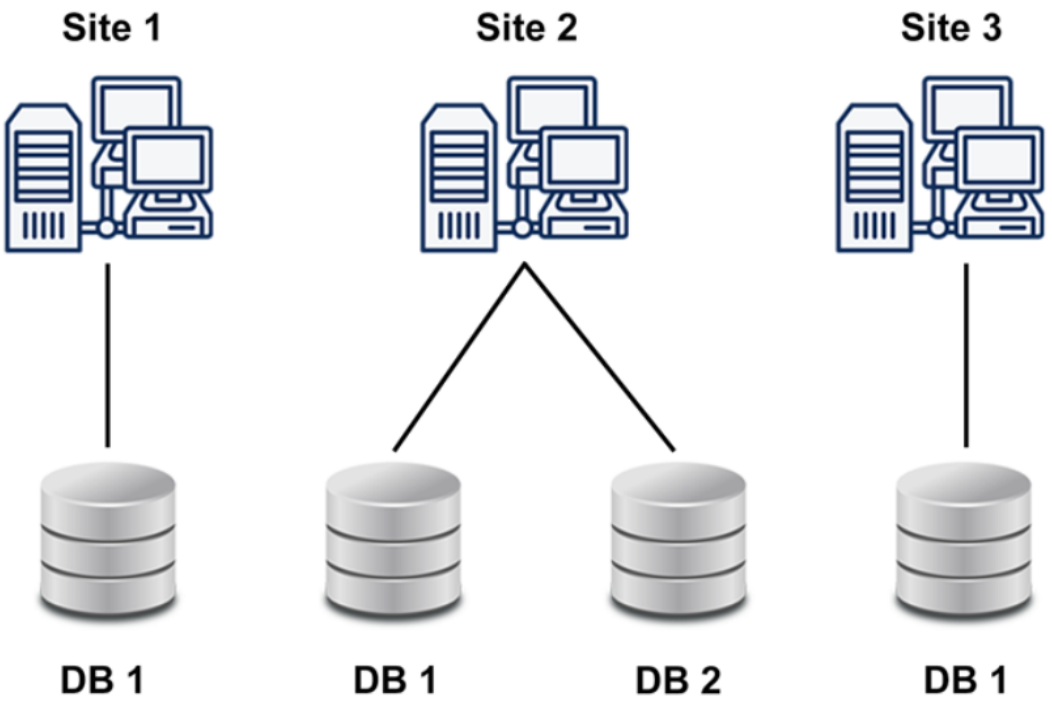

Рисунок 3 - Реплікація розподіленої бази даних

Фрагментація

Коли справа доходить до фрагментації розподіленого сховища бази даних, відносини фрагментовані, що означає, що вони розбиті на менші частини. Кожен із фрагментів зберігається на окремому сайті, де це потрібно.

Необхідною умовою для фрагментації є переконатися, що фрагменти можна згодом реконструювати у вихідне відношення без втрати даних.

Перевага фрагментації полягає в тому, що немає копій даних, що запобігає невідповідності даних.

Існує два типи фрагментації:

- Горизонтальна фрагментація – схема відношення фрагментується на групи рядків, і кожна група (кортеж) призначається одному фрагменту.

- Вертикальна фрагментація – схема відношень фрагментована на менші схеми, і кожен фрагмент містить загальний ключ-кандидат, який гарантує з’єднання без втрат.

Також цих два типи фрагментації можливо використовувати одночасно, так звана “змішана фрагментація”

Переваги використання розподілених баз даних

- Модульна розробка. Модульна розробка розподіленої бази даних передбачає, що систему можна розширити до нових місць або підрозділів шляхом додавання нових серверів і даних до існуючих установок і безперервного підключення їх до розподіленої системи. Цей тип розширення не викликає перебоїв у функціонуванні розподілених баз даних.
- Надійність. Розподілені бази даних пропонують більшу надійність на відміну від централізованих баз даних. У разі збою бази даних в централізованій базі даних система повністю зупиняється. У розподіленій базі даних система функціонує навіть у разі виникнення збоїв, забезпечуючи лише зниження продуктивності, поки проблема не буде вирішена.
- Нижча вартість зв'язку. Локальне зберігання даних зменшує витрати на зв’язок для маніпулювання даними в розподілених базах даних. Локальне зберігання даних у централізованих базах даних неможливо.
- Краща відповідь. Ефективний розподіл даних у розподіленій системі баз даних забезпечує швидшу відповідь, коли запити користувачів задовольняються локально. У централізованих базах даних запити користувачів проходять через центральну машину, яка обробляє всі

запити. Результатом є збільшення часу відповіді, особливо з великою кількістю запитів.

Недоліки використання розпоілених баз даних

- Дороге програмне забезпечення. Забезпечення прозорості та координації даних на кількох сайтах часто вимагає використання дорогого програмного забезпечення в системі розподіленої бази даних.
- Великі накладні витрати. Багато операцій на кількох сайтах вимагають численних обчислень і постійної синхронізації, коли використовується реплікація бази даних, що спричиняє великі витрати на обробку.
- Цілісність даних. Можливою проблемою під час використання реплікації бази даних є цілісність даних, яка порушується оновленням даних на кількох сайтах.
- Неправильне розповсюдження даних. Відповідність запитам користувачів багато в чому залежить від правильного розподілу даних. Це означає, що швидкість реагування може бути знижена, якщо дані не розподіляються належним чином на кількох сайтах.

1.3 Механізми обробки транзакцій в розподілених даних

Розподілена транзакція — це набір операцій над даними, які виконуються в двох або більше сховищах даних (особливо в базах даних). Зазвичай він координується між окремими вузлами, з'єднаними мережею, але також може охоплювати кілька баз даних на одному сервері.

Є два можливих результату:

- всі операції успішно завершені;
- жодна з операцій не виконується взагалі через збій десь у системі.

В останньому випадку, якщо деяка робота була виконана до збою, ця робота буде скасована, щоб переконатися, що мережна робота не була виконана. Цей тип роботи відповідає принципам «ACID» (atomicity-consistency-isolation-durability) баз даних, які забезпечують цілісність даних. ACID найчастіше асоціюється з транзакціями на одному сервері бази даних, але розподілені транзакції поширюють цю гарантію на кілька баз даних.

Атомність

У контексті баз даних атомарність означає, що ви:

- Візьміть на себе зобов'язання щодо всієї транзакції
- Не мати жодної транзакції взагалі

По суті, атомарна транзакція гарантує, що будь-яка фіксація успішно завершить всю операцію. Або, у разі втрати з'єднання в середині операції, база даних повертається до свого стану до початку фіксації.

Це важливо для запобігання збоям або відключенням, які створюють випадки, коли транзакція була частково завершена до невідомого загального стану. Якщо під час транзакції без атомарності відбувається збій, ви не можете точно знати, як далеко пройшов процес до того, як транзакція була перервана.

Використовуючи атомарність, ви гарантуєте, що або вся транзакція успішно завершена, або що жодної з них не було.

Послідовність

Узгодженість відноситься до підтримки обмежень цілісності даних.

Послідовна транзакція не порушуватиме обмежень цілісності, накладених на дані правилами бази даних. Забезпечення узгодженості гарантує, що якщо база даних переходить у незаконний стан (якщо відбувається порушення обмежень цілісності даних), процес буде перервано, а зміни повернуться до попереднього легального стану.

Іншим способом забезпечення узгодженості в базі даних протягом кожної транзакції є також застосування декларативних обмежень, накладених на базу даних.

Прикладом декларативного обмеження може бути те, що всі рахунки клієнтів повинні мати додатний баланс. Якщо транзакція призведе до від'ємного балансу на рахунку клієнта, ця транзакція буде відкатана. Це гарантує, що зміни успішно підтримують цілісність даних або вони повністю скасовані.

Ізоляція

Ізольовані транзакції вважаються «серіалізованими», що означає, що кожна транзакція відбувається в окремому порядку без жодних транзакцій, що відбуваються в тандемі.

Будь-яке читання або запис, виконані в базі даних, не будуть вплинути на інші читання та записи окремих транзакцій, що відбуваються в одній базі даних. Глобальне замовлення створюється, коли кожна транзакція стоїть у черзі, щоб гарантувати, що транзакції повністю завершені до початку іншої.

Важливо, що це не означає, що дві операції не можуть виконуватися одночасно. Декілька транзакцій можуть відбуватися, якщо ці транзакції не мають можливості вплинути на інші транзакції, що відбуваються одночасно.

Це може вплинути на швидкість транзакцій, оскільки це може змусити багато операцій чекати, перш ніж вони зможуть ініціюватися. Однак цей компроміс вартий додаткової безпеки даних, яку забезпечує ізоляція.

Ізоляція може бути досягнута за допомогою використання ковзної шкали вседозволеності, яка знаходиться між тим, що називають оптимістичними трансакціями, і песимістичними транзакціями:

Оптимістична схема транзакцій передбачає, що інші транзакції будуть завершені без читання або запису в одне й те саме місце двічі. За оптимістичної схеми обидві транзакції будуть скасовані та повторені у випадку, якщо транзакція двічі потрапляє в одне й те саме місце.

Песимістична схема транзакцій надає менше свободи та блокує ресурси на основі припущення, що транзакції впливатимуть на інші. Це призводить до меншої кількості переривань і повторних спроб, але це також означає, що транзакції змушені частіше чекати своєї черги в порівнянні з оптимістичним підходом до транзакцій.

Знаходячи солодке місце між цими двома ідеалами, часто ви знайдете найкращий загальний результат.

Довговічність

Останнім аспектом підходу ACID до управління базою даних є довговічність.

Довговічність гарантує, що зміни, внесені в базу даних (транзакції), які успішно фіксуються, зберігатимуться назавжди, навіть у разі системних збоїв. Це гарантує, що дані в базі даних не будуть пошкоджені:

- Відключення послуги
- Збої
- Інші випадки відмови

Довговічність досягається завдяки використанню журналів змін, на які посилаються при перезапуску баз даних (або частин бази даних).

Принцип роботи транзакцій у розподіленому середовищі

Розподілені транзакції мають такі ж вимоги до завершення обробки, як і звичайні транзакції бази даних, але ними потрібно керувати кількома ресурсами, що робить їх більш складним у реалізації для розробників баз даних. Кілька ресурсів додають більше точок збою, наприклад, окремі програмні системи, які запускають ресурси (наприклад, програмне забезпечення бази даних), додаткові сервери обладнання та збої в мережі. Це робить розподілені транзакції вразливими до збоїв, тому для збереження цілісності даних необхідно запровадити запобіжні заходи.

Щоб відбулася розподілена транзакція, менеджери транзакцій координують ресурси (або декілька баз даних, або декілька вузлів однієї бази даних). Менеджер транзакцій може бути одним із сховищ даних, які будуть оновлюватися в рамках транзакції, або це може бути повністю незалежний окремий ресурс, який відповідає лише за координацію. Менеджер транзакцій вирішує, чи здійснити успішну транзакцію, чи відкотити невдалу транзакцію, остання з яких залишає базу даних незмінною.

Спочатку програма запитує розподілену транзакцію до менеджера транзакцій. Потім менеджер транзакцій розгалужується до кожного ресурсу, який матиме власного «диспетчера ресурсів», щоб допомогти йому брати участь у розподілених транзакціях. Розподілені транзакції часто виконуються в два етапи, щоб захистити від часткових оновлень, які можуть статися у разі збою. Перший етап передбачає визнання наміру здійснити або етап «підготовки до вчинення». Після підтвердження всіх ресурсів їх просять виконати остаточну фіксацію, а потім транзакцію завершують.

Ми можемо розглянути основний приклад того, що відбувається, коли відбувається збій під час розподіленої транзакції. Скажімо, один або кілька ресурсів стають недоступними на етапі підготовки до фіксації. Коли запит закінчується, менеджер транзакцій повідомляє кожному ресурсу видалити статус підготовки до фіксації, і всі дані будуть скинуті до початкового стану. Якщо натомість будь-який із ресурсів стане недоступним під час фази фіксації, то менеджер транзакцій повідомить іншим ресурсам, які успішно зафіксували свою частину транзакції, скасувати або «відкати» цю транзакцію, і знову дані повертаються до своїх вихідний стан. Потім програма має повторити спробу транзакції, щоб переконатися, що вона завершена.

1.4 Архітектури гетерогенних DDBMS

Серед різноманітних гетерогенних систем DDB ми можемо знайти різні рівні зв'язку, від більш слабо зв'язаних систем до більш тісно пов'язаних, відповідно до класифікації, запропонованої Шетом і Ларсоном (1990), на основі рівня автономії компонентних БД. У слабо пов’язаних системах кожен компонент БД обробляє вхід і вихід системи, а також схему даних і обмін даними з іншими БД, зберігаючи при цьому свою власну автономію. Навпаки, тісно пов’язані системи вимагають узгодження, на шкоду їх власній автономії, для досягнення глобальної схеми для всіх користувачів DDB, на додаток до функціональної архітектури для вирішення глобальних доступів.

Тісно пов’язані системи зазвичай використовують п’ятирівневу архітектуру схеми, що дозволяє отримати глобальну схему для гетерогенної DDB. Під час роботи гетерогенних DDB досить часто використовують функціональну архітектуру, засновану на посередниках та обгортках, з перевагою, що це дозволяє різні ступені зв’язку. Крім того, зараз використовуються сучасні однорангові системи, і для них також доступна еталонна функціональна архітектура. Оскільки однорангові системи допускають різні ступені зв'язку, здавалося б доцільним реалізувати більш тісно пов'язані системи з посередниками та обгортками, тоді як більш слабо пов'язані системи будуть реалізовані за допомогою однорангових систем. У наступних розділах будуть розглянуті три типи архітектури, які ми тут визначили.

1.4.1 П'ятирівнева архітектура схем

Архітектура еталонної схеми для тісно пов'язаних гетерогенних DDB базується на архітектурі трьохрівневої схеми ANSI/SPARC, порівнянної з архітектурою для однорідних DDB.

Враховуючи необхідність створення гетерогенної DDB з різних гетерогенних баз даних, використовуваний процес відомий як проектування знизу вгору. Замість створення логічного проекту з нуля, як у випадку однорідних DDB, він починається з локальних проектів БД, які сформують нову DDB.

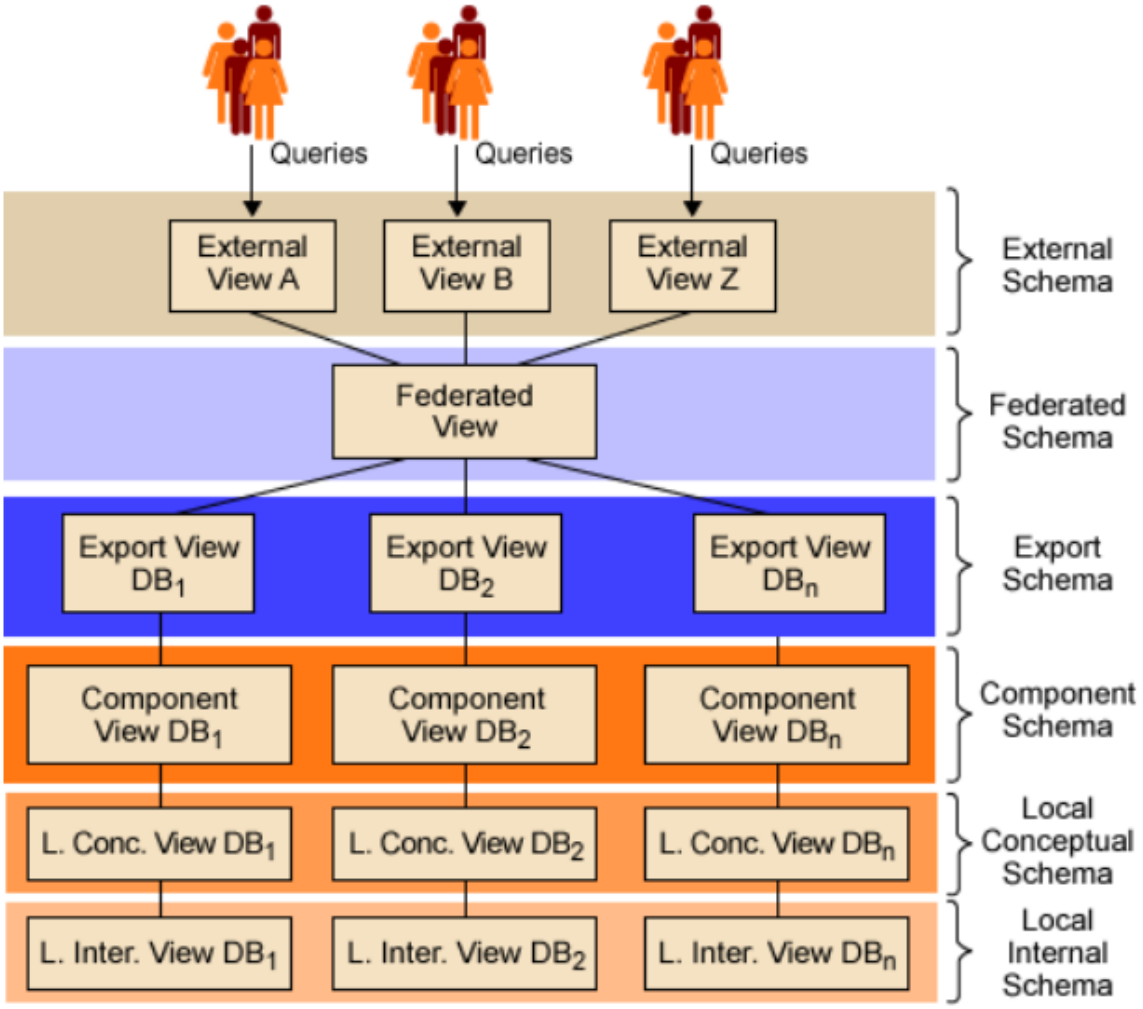

Рисунок 4 - Тісно пов'язана гетерогенна еталонна архітектура DDB

На малюнку 7 показано внутрішню схему та концептуальну схему – як локальну внутрішню схему та локальну концептуальну схему – кожного компонента БД. Перший додатковий рівень схеми - це рівень схеми компонента. Метою цього рівня схеми є вирішення синтаксичної неоднорідності моделі

даних компонентних БД. Так само, як концептуальна схема виражається як залежна від локальної СУБД, схема компонентів виражається за допомогою канонічної моделі даних, обраної як модель даних гетерогенної системи. Цей рівень схеми представляє всі локальні схеми, виражені відповідно до тієї ж моделі даних.

Другий додатковий рівень схеми, рівень схеми експорту, визначає підмножину даних компонентної бази даних, які будуть спільно використовуватися через гетерогенну DDB. Як і схема компонентів, схема експорту також виражається в термінах канонічної моделі даних.

Метою об'єднаної схеми (див. малюнок 4), яка діє як глобальна схема, є обслуговування гетерогенної DDB у розв'язанні всіх можливих семантичних неоднорідностей між різними схемами експорту компонентних DB. Ми обговорюємо різні аспекти, пов’язані з неоднорідністю. Об’єднана схема виражається в термінах канонічної моделі даних.

Як і в інших архітектурах схем, зовнішні схеми визначаються з об’єднаної схеми. Користувачі гетерогенної DDB зроблять глобальний доступ на основі зовнішніх схем. Є два варіанти вираження зовнішніх схем. По-перше, зовнішні схеми можуть бути виражені через канонічну модель даних гетерогенної системи (одномовна архітектура). Таким чином, користувачам може знадобитися звертатися до системи по-різному залежно від того, локальний чи глобальний доступ. По-друге, кожна зовнішня схема може бути виражена відповідно до моделі даних компонента БД, до якої належить користувач. У цьому випадку буде багатомовна архітектура, що дозволить користувачеві працювати однаково як локально, так і глобально.

Відображення, які пов’язують елементи на різних рівнях схеми еталонної архітектури гетерогенних DDB, настільки ж важливі, як і в архітектурі однорідних DDB.

1.4.2 Функціональна архітектура обгортки-посередника

З функціональної точки зору, основна відмінність між гетерогенними та однорідними DDB – це автономні СУБД, які існують у кожній БД, що містить гетерогенну DDB. Неоднорідні DDB отримують з програмного рівня, який працює над компонентною СУБД (як показано у функціональній еталонній архітектурі на малюнку 5) і що забезпечує доступ глобальним користувачам до різних БД. Цей рівень діє так, ніби це просто інша програма, яка надсилає запити та отримує відповіді.

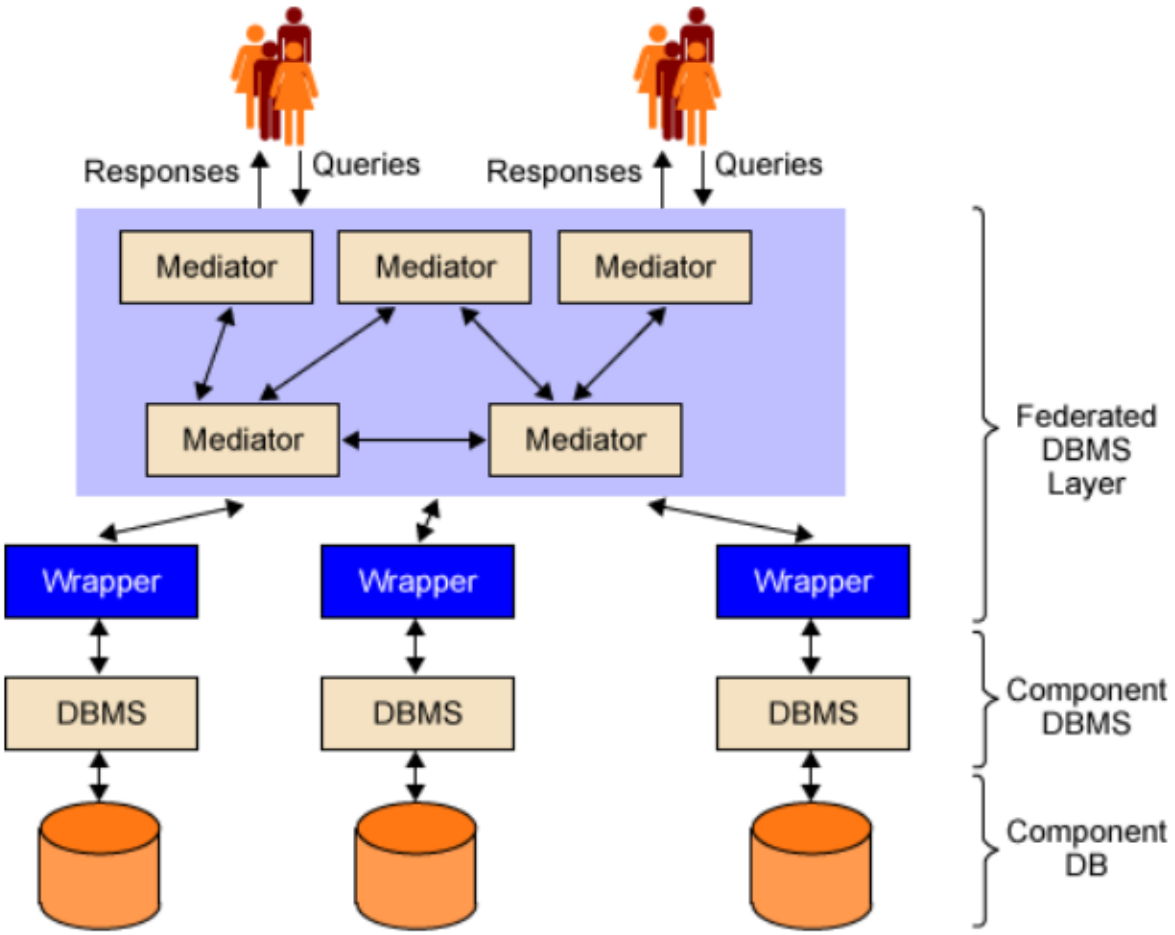

Рисунок 5 - Функціональна архітектура обгортки-посередника

Найпопулярніша реалізація функціональної архітектури для гетерогенних DDB базується на посередниках і обгортках, як показано на малюнку 5.

У Wiederhold (1992) посередник визначається як «програмний модуль, який використовує закодовані знання про певні набори або підмножини даних для створення інформації для більш високого рівня додатків».

Кожен посередник виконує задану функцію через чітко визначений інтерфейс. Таким чином, різнорідний рівень DDBMS реалізується через посередник – або через їх ієрархію –, який має справу із запитами на доступ гетерогенних користувачів DDB, охоплюючи всю функціональність глобального менеджера.

Медіатори зазвичай працюють, використовуючи модель даних і загальну мову інтерфейсу. Таким чином, обгортки використовуються для вирішення потенційних неоднорідностей, що виникають із різних компонентних СУБД. Кожна обгортка отримується з відображення між представленням компонентної СУБД і поданням посередника.

1.4.3 Однорангова функціональна архітектура

З часом з’явилося кілька типів однорангових (P2P) систем, усі з різними цілями. Ми розглянемо сучасні P2P-системи, які зосереджені на обміні даними і характеризуються:

- Масове поширення, оскільки вони складаються з тисяч вузлів (рівників) з широким географічним поширенням і мають можливість утворення груп у певних місцях.
- Притаманна неоднорідність одноліткових і їх повна індивідуальна автономія.
- Нестабільність системи, оскільки кожен одноранговий пристрій зазвичай є персональним комп’ютером, який за бажанням входить і виходить із системи, робить управління даними більш складним.

Щоб забезпечити функції керування даними цих систем, ми повинні мати справу з розташуванням даних, обробкою запитів, інтеграцією даних та узгодженістю даних.

На малюнку 6 показано функціональну еталонну архітектуру для однорангового партнера, який бере участь у системі обміну даними P2P. Найважливішим моментом пропонованої архітектури є поділ її функціональних можливостей на три основні компоненти: інтерфейс, який використовується для надсилання запитів на доступ, рівень керування даними, який обробляє запити та інформацію з каталогу метаданих, та інфраструктуру P2P, яка включає підпорядковану структуру. -рівень мережі P2P і P2P мережі як такої. Залежно від функціональних можливостей системи P2P, один або кілька з цих компонентів можуть не існувати, або вони можуть бути об’єднані або реалізовані в спеціалізованих однорангових системах.

Запити доступу, як до локальних даних, так і до глобальних даних, передаються через інтерфейс користувача або через API керування даними та надсилаються на рівень керування даними. Рівень, який отримує запит, має менеджера запитів, що відповідає за його виконання.

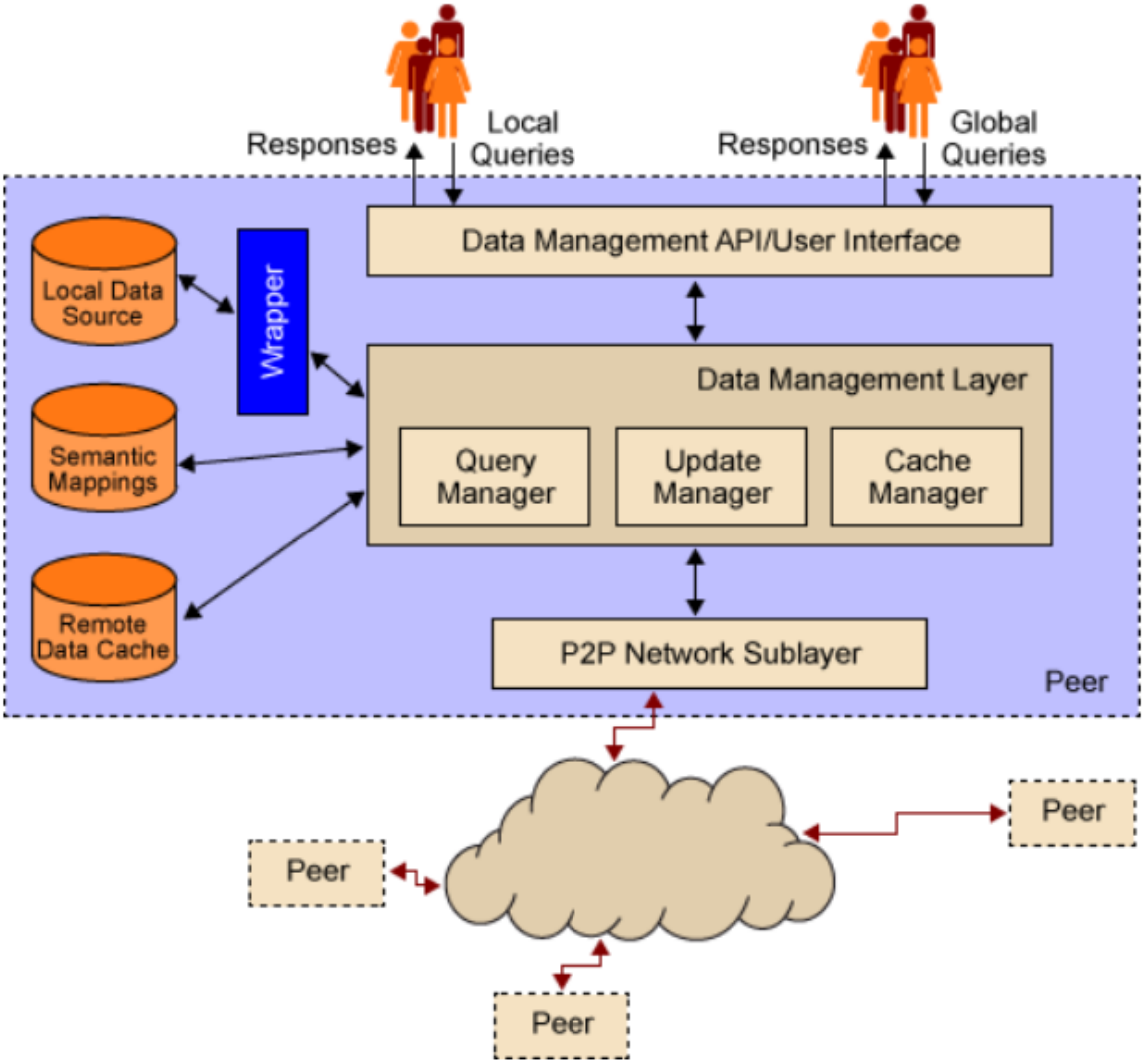

Рисунок 6 - Однорангова еталонна архітектура

Усі завдання, необхідні менеджеру запитів для виконання запиту, залежать від неоднорідності системи. Якщо однорангові партнери неоднорідні, менеджер запитів повинен перевірити семантичні відображення, щоб ідентифікувати однорангові системи, які зберігають відповідні дані для вирішення запиту. Потім запит потрібно переформулювати, щоб кожен задіяний

одноранговий зв’язок міг зрозуміти запит. Щоб спілкуватися з одноранговими, менеджер запитів викликає служби, реалізовані підрівнем мережі P2P. Враховуючи, що деякі системи P2P мають спеціалізовані однорангові для зберігання семантичних відображень, одноранговий вузол, який отримує запит на доступ, повинен зв’язатися зі спеціалізованим однорангом, щоб виконати запит.

Більше того, якби всі однорангові партнери в системі P2P мали однакову схему даних, функціональність переформулювання запиту не була б потрібна, а також не потрібно було б зберігати семантичні відображення.

Незалежно від того, чи координується виконання однорангом, який отримує запит, чи спеціалізованим вузлом, коли одноранговий партнер, який отримує запит, отримує відповіді від набору залучених однорангових партнерів, він має можливість зберігати результати локально за допомогою менеджера кешу, щоб прискорити виконання подібних запитів у майбутньому.

Менеджер запитів також відповідає за виконання локальної частини кожного глобального запиту, створеного іншим однорангом. У цьому випадку існування обгортки може усунути неоднорідності. Крім того, коли потрібне оновлення даних, менеджер оновлень відповідає за координацію оновлення між одноранговими партнерами, які містять репліки даних, які потрібно змінити.

Мережева інфраструктура P2P, незалежно від того, як вона реалізована, надає послуги зв'язку на рівні управління даними. Зауважте, що виконання запитів чутливе до реалізації цієї інфраструктури, оскільки це накладена мережа над фізичною мережею (як правило, Інтернет). Ця мережа накладання може бути чистою (усі однорангові рівні) або гібридною (деякі однорангові мають спеціальні функції). Набір однорангових вузлів може утворювати

неструктуровану топологію (немає обмежень щодо розташування даних у однорангових вузлах) або структуровану топологію (розташування даних контролюється для досягнення більшої масштабованості, що ставить під загрозу автономію). Ключ до виконання запиту полягає в тому, щоб побачити, як ресурси індексуються і як вони будуть шукатися. У неструктурованих топологіях використовуються індекси, що зберігаються централізовано або децентралізовано. Структуровані топології зазвичай використовують динамічну хеш-таблицю, за допомогою якої хешування до ключа, який є ідентифікатором об’єкта, генерує ідентифікатор однорангового партнера, де зберігаються дані, пов’язані з об’єктом.

1.5 Аналіз моделі розподіленої обробки запитів

Обробка запитів у розподіленій базі даних включає багато чого обмін даними між сайтами-учасниками. Це міжсайтовий зв'язок/передача відповідних даних домінуючий фактор або вартість для обмеження загального запиту обробка та оптимізація. Ступінь передачі даних безпосередньо пов’язана з неоднорідністю сайтів, до яких здійснюється доступ у плані запиту. Наприклад, у плані запиту все відносини доступні з різних сайтів, то це стратегія призведе до найбільшого обсягу передачі даних серед сайтів, тому запит не надає перевагу оптимізатор. У DQP різні понесені витрати включають процесор, Вартість введення-виведення та зв'язку між сайтом. Серед у них вартість міжоб’єктового зв’язку є домінуючим вартість. Щоб обробити запит користувача в розподіленому системи баз даних, необхідні дані можуть бути отримано з кількох сайтів, розподілених на комп’ютері мережі. Крім того, як кількість сайтів, що містять відношення, до яких звертається запит, збільшуються, число можливих дійсних планів запитів також збільшується. Так що стає вкрай необхідним для розробки плану обробки запиту що тягне за собою оптимальні витрати на обробку запиту. Однак кількість таких можливих планів запитів збільшується в геометричній прогресії зі збільшенням кількості відносини в запиті, а також зі збільшенням кількість сайтів, що їх містять. Таким чином, великий простір пошуку, що містить усі можливі плани запитів необхідно дослідити, щоб обчислити оптимальний запит.

Обробка запитів у такому середовищі є складним завданням для процесора запитів, як і для запиту користувача є можливі еквівалентні варіанти кількох запитів. Ці альтернативи запиту еквівалентні за умовами виведення. Семантика всіх еквівалентів подібна до них отримати доступ до схожого набору

відносин, але з різних оперативні сайти. Політика розміщення бази даних відтворює важливу роль відіграє вибір відповідних сайтів відносини. Вибір найкращих альтернатив для обробки з створений пул альтернатив є критичним завданням оптимізатори запитів, основною метою яких є вибір оптимальні (кращі) рішення. Підбір оптимального плани відповідно до компромісів між різними дизайнами цілі або евристики.

Висновки

У цьому розділі аналізується структура сучасного розподілу даних в розподілених базах даних. В цілому процес розробки розподілених баз даних дуже трудомісткий і пов'язаний зі значними матеріальними витратами. Завдання вибору найкращої відповідності властивостей RBD і методів розподілу даних в мережі вимагає всебічного аналізу, так як проектні рішення безпосередньо впливають на реалізацію і подальшу роботу інформаційної системи.

Розділ 2. Метод ефективного розподілення фрагментів розподілених баз даних у динамічному середовищі.

2.1 Постановка проблеми ефективного розподілення фрагментів розподілених баз даних

Проблеми фрагментації, реплікації та розподілу розглядаються як важливі проблеми проектування, які мають великий вплив на продуктивність DDBS і призводять до оптимальних рішень, особливо в динамічному розподіленому середовищі.

Зазвичай фрагментація виконується без урахування виділення. У той час як розподіл, з іншого боку, завжди залежить від фрагментації і виконується в припущенні, що фрагментація отримана апріорі.

Насправді, при проектуванні розподіленої бази даних слід враховувати чотири важливі моменти. По-перше, як глобальні відносини повинні бути фрагментовані. По-друге, скільки копій фрагмента потрібно відтворити. По-третє, яка техніка буде використовуватися для розподілу фрагментів на різних сайтах мережі та яка інформація для фрагментації та розподілу потрібно знати.

Імовірнісні та евристичні методи є одними з багатьох різних підходів, які дослідники прийняли для вирішення проблеми розподілу. Імовірнісний метод дає багато можливих сценаріїв пошуку оптимального рішення, але він дуже дорогий. Тоді як евристичний підхід призводить до майже оптимального рішення і рідко до оптимального. Результати, отримані евристичним методом, визначають якість цього методу. Запропонована нами модель розглядається як евристичний підхід, а не імовірнісний метод, хоча вона дещо поєднує концепції обох.

Для кращого часу відповіді необхідний оптимальний розподіл фрагментів по сайтах мережі. У [21, 26, 15, 2, 1] обговорено декілька рішень для вирішення проблеми розподілу. Однак через складність цієї задачі (NP-повна) в літературі були запропоновані більш евристичні алгоритми, тобто алгоритми меншої складності, які дають лише наближене рішення.

Однією з перших робіт з вивчення проблеми розміщення файлів було розглянуто в [8], де була розроблена загальна оптимізаційна модель для мінімізації загальних операційних витрат з огляду на обмеження часу, ємність зберігання та кількість копій для кожного файлу. Підхід SAGA був запропонований у [1] для вибору оптимальних місць для фрагментів даних. Шаблон доступу до даних і вартість передачі були використані для визначення оптимального розміщення. Однак [2] представив жадібний підхід, де пропонується математична модель розподілу даних у DDBS. Він врахував витрати на мережевий зв'язок, локальну обробку та зберігання даних для розподілу фрагментів по сайту, враховуючи обмеження пропускної здатності сайту та обмеження фрагментів.

У [17] була запропонована модель динамічного перерозподілу, яка перерозподіляє фрагменти з метою мінімізації переданих даних. Проте більш комплексний підхід до розподілу фрагментів для мінімізації загальних витрат на передачу даних був розроблений у [4]. З іншого боку, [18] дав інтегровану техніку для фрагментації та розподілу для отримання оптимальної продуктивності. Тоді як у [7] з цією ж метою використовується метод лагранжової релаксації. У [12] представлена формулювання цілочисельного програмування для ненадлишкової версії задачі розподілу фрагментів. Проте високопродуктивний обчислювальний метод для розподілу даних у DDBS

використовується в [13]. Проблема розповсюдження фрагментів віртуальних XML-сховищ через Web розглянута в [5].

У більшості описаних вище робіт розподіл даних здійснювався на основі підходів до статичного розподілу, які забезпечують найкращі рішення для середовищ, де ймовірність доступу сайтів до фрагментів ніколи не змінюється. Але рішення для статичного розподілу погіршить продуктивність бази даних у спеціальному середовищі, де ці ймовірності доступу змінюються з часом.

Структура для перерозподілу даних щодо динамічного розподілу даних була представлена в [6]. У [3] запропоновано динамічний алгоритм розподілу даних для нереплікованих систем баз даних, але для аналізу алгоритму не проводиться моделювання. Модель динамічного розподілу та реплікації даних для перерозподілу даних була представлена в [25 та 10]. У роботі [16] розроблено оптимальний алгоритм для нереплікованих систем баз даних. Однак [23] запропонував пороговий алгоритм для нереплікованих розподілених баз даних, де фрагменти безперервно перерозподіляються відповідно до змінних шаблонів доступу до даних. У [14] запропоновано алгоритм динамічного розподілу даних, який перерозподіляє дані шляхом зміни шаблонів доступу до даних у межах обмеження часу. Евристичний алгоритм, заснований на оптимізації колоній мурах для мінімізації вартості передачі та розподілу даних між сайтами мережі, був представлений в [11].

У цій роботі запропоновано динамічну модель перерозподілу даних для реплікованої та нереплікованої DDBS відповідно до заданих фрагментів Матриця оновлення (UM) та Матриця витрат на відстань (DM). Оцінюються витрати на перерозподіл фрагмента на сайті мережі, а потім приймається рішення про міграцію, вибираючи сайт Sj, який має найвищу вартість

оновлення запиту для фрагмента Fi, який буде обраний як сайт-кандидат для зберігання фрагмента Fi для мінімізації зв’язку. вартість.

Цей метод використовує техніку пріоритету фрагментів (FP), щоб уникнути дублювання фрагментів (що може мати місце на початковому етапі розподілу) і порушень обмежень сайтів, коли вони трапляються.

2.1.1 Запропоновані визначення в методі розподілення фрагментів

Запропонований метод припускає, що існує повністю підключена мережа, що складається з багатьох сайтів S = {S1, S2, …., Sm}, кожен сайт має ємність (C), фрагмент limit (FL) і локальну систему баз даних. Зв'язок між двома сайтами Si і Sj має ціле додатне число (CCij), пов'язане з ними, що представляє вартість одиничної передачі даних з сайту Si на сайт Sj. Кожен сайт має набір запитів Q = {Q1, Q2, ….., Qk}, які вважаються найбільш часто виконуваними запитами, що становлять понад 75% обробки сайту в DDBS. Кожен запит Qk може виконуватися з будь-якого сайту з певною частотою, частоти виконання k запитів на m сайтах можуть бути представлені матрицею m× k (QFij). Нехай S містить набір фрагментів F = {F1, F2, …, Fn}, що представляють розділи відношень під час фази фрагментації проектування DDBS. Щоб зробити розподіл більш ефективним, нам потрібно визначити не тільки кількість копій для кожного фрагмента на кожному сайті, але й знайти правильний розподіл для кожного з них на сайтах відповідно до зібраної інформації запиту.

Відповідно до [24] існує два визначення оптимальності: перше; мінімальна вартість (вартість зберігання кожного фрагмента Fi на сайті Sk, + вартість запиту Fi на сайті Sk, + вартість оновлення Fi на сайтах, де він

зберігається + вартість передачі даних); другий; продуктивність (мінімізація часу реакції та максимальна пропускна здатність системи).

Ми прийняли перший варіант (мінімальна вартість) у нашому запропонованому методі для вимірювання оптимальності. Таким чином, задачу оптимальності відповідно до нашого методу перерозподілу можна визначити як мінімізацію вартості зв’язку в процесі перерозподілу фрагмента Fi на сайт Sj .

Використані позначення в цій статті описані далі в таблиці 1;

Таблиця 1 - Позначення методу

M	Кількість сайтів мережі DDBS
Sj	j-й сайт
N	Кількість фрагментів даних у DDBS
Fi	I-й фрагмент даних
K	Кількість запитів DDBS
RN	Кількість відношень
Q	Набір запитів
Qk	k-й запит
Qj#	Кількість запитів над сайт j
RFij	Значення частоти доступу для пошукового запиту i на сайті j
UFij	Значення частоти доступу для оновлення запиту i на сайті j

QFki	Частота доступу k-го запиту на сайті j
Cj	Ємність сайту j
CCij	Вартість зв'язку між сайтом і та сайтом j
FLi	Максимальна кількість фрагментів для сайту j
TC	Загальна вартість

2.2 Особливості розподіленого динамічного середовища

Щоб отримати формули вартості, такі як функції вартості, деяку інформацію потрібно проаналізувати заздалегідь [20]; це може включати: (1) інформацію бази даних, (2) інформацію про поведінку запитів, (3) інформацію про сайт і, нарешті, (4) інформацію про мережу. Далі вони будуть описані.

2.2.1 Представлення архітектури бази даних

Зазвичай відношення складається з багатьох фрагментів. Кожен з яких має розмір Z(Fi)), який необхідно заздалегідь визначити, оскільки він відіграє ключову роль у обчисленні вартості зв'язку.

$Z(Fi) = Card(Fi) * L(Fi)$, $i = 1,\ldots, n$.

Де L – довжина кортежу фрагмента, а card(Fi) – це потужність фрагмента (тобто кількість записів фрагментів), довжину (Fi) можна обчислити за такою формулою:

$Length(Fi) = \sum LA$, $1 <= A < = h$.

Де h – кількість атрибутів для цього фрагмента, а A вказує на A-й атрибут. Отже, $Z(Fi) = n * \sum Lj$, де n – кількість записів фрагментів. Щоб спростити нашу модель, ми припустимо, що розміри фрагментів вказані в таблиці 2.

Таблиця 2 - Розміри фрагментів

Фрагмент	F1	F2	F3	F4	F5	F6
Розмір	25	45	32	60	36	57

Нам також потрібно визначити матрицю Xij, яка показує віднесення фрагментів до сайтів. Ця матриця буде визначена матрицею пошуку (RM) і використовуватиметься на початковому етапі розподілу.

Xij= 1, if F is assigned to S, otherwise 0

2.2.2 Представлення поведінки запитів у відповідності їх типу

Оскільки будь-який запит або витягує, чи оновлює дані, ми визначаємо дві матриці доступу: Матриця пошуку RM (таблиця 3) і матриця оновлення UM (таблиця 4), що описує поведінку пошуку та оновлення відповідно для всіх запитів. Елементи Rkj або Ukj в RM або UM відповідно дають частоту доступу до фрагмента Fj за запитом k.

RM (Rkj) = 1, if Qk retrieve Fj, otherwise 0

UM (Ukj) = 1, if Qk update Fj, otherwise 0

Таблиця 3 - RM

Запит/ Фрагмент	F1	F2	F3	F4	F5	F6
Q1	2	3	0	1	0	0
Q2	2	0	0	0	1	1
Q3	6	0	3	0	2	0
Q4	3	0	2	0	0	0
Q5	0	1	0	2	0	1

Таблиця 4 - UM

Запит/фрагмент	F1	F2	F3	F4	F5	F6
Q1	0	0	2	0	1	0
Q2	0	1	1	0	0	2
Q3	3	0	0	2	0	1
Q4	0	1	0	0	1	1
Q5	0	0	2	0	1	0

Наприклад, у таблиці 3 запит Q1 отримує фрагменти F1 двічі, F2 три рази і F4 лише один раз. Він також оновлює фрагменти F3 двічі і F5 один раз у таблиці 4. Однак для будь-якого запиту, що використовується на будь-якому сайті, він повинен мати значення частоти на цьому сайті. Таким чином, ми повинні визначити частотну матрицю для запитів (QF), яка показує частоту виконання запитів на кожному сайті (табл. 5 зображує це).

Таблиця 5 - Частота запитів

Запит/сайт	S1	S2	S3	S4
Q1	0	2	3	1
Q2	0	3	0	0

Q3	2	0	1	0
Q4	0	0	4	0
Q5	1	1	0	2

2.2.3 Архітектура сайту

Архітектура сайту представлена в нашому методі відповідно до обмежень сайту, система повинна відповідати обмеженням ємності сайту (C) і обмеженням фрагментів (FL), які представляють максимальний розмір і кількість фрагментів, які сайт може обробити. Далі представлені обмеження, які використовуються для управління процесом розподілу;

$$\sum Xij >= 1,\ 1<= i <= n \ (1)$$ // фаза початкового розподілу

$$\sum Qij * \text{розмір}(Fi) <= Ci,\ 1 <= j <= m \ (2)$$

$$\sum Qij <= FLi,\ 1 <= i <= n \ (3)$$

$$\sum Qij = 1,\ 1<= i <= n \ (4)$$ //фази перерозподілу

Рівняння(1) вказує, що кожен фрагмент F повинен бути виділений принаймні одному сайту на початковому етапі фаза розподілу. Рівняння (2) стверджує, що жоден сайт не отримає більше, ніж його потужність. Рівняння (3) стверджує, що кожен сайт не оброблятиме більше заданої кількості фрагментів (FL) і, нарешті, рівняння (4) стверджує, що фрагмент буде виділено лише

одному сайту на етапі після розподілу (перерозподілу). Таблиця 6 зображує обмеження нашого методу.

Таблиця 6 - Мережні сайти з обмеженнями

Сайт	Місткість(C)	Ліміт фрагментів (FL)
S1	100	6
S2	80	5
S3	60	4
S4	90	4

2.2.4 Архітектура мережі

У запропонованому підході передбачається, що сайти мережі DDBS є повністю підключені мережі. І кожне посилання між сайтами Si та Sj має значення вартості зв’язку (CCij), що представляє вартість зв’язку між ними, як показано на рисунку 7.

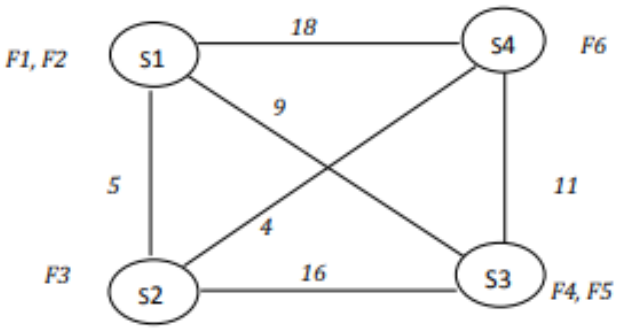

Рисунок 7 - Мережні сайти

Для спрощення нашої моделі матриця витрат на зв’язок наведена як симетрична матриця, тобто вартість зв'язку між сайтами Si і Sj така ж, як і між сайтами Sj і Si, а вартість зв'язку в межах одного сайту дорівнює нулю, як показано в таблиці 7.

Таблиця 7 - Матриця витрат на зв'язок

Сайти	S1	S2	S3	S4
S1	0	5	9	18
S2	5	0	16	4
S3	9	16	0	11
S4	18	4	11	0

Після застосування мінімального алгоритму з [20] до матриці витрат на зв'язок ми отримуємо матрицю витрат на відстань (DM), як показано в (таблиця 8).

Таблиця 8 - Матриця відстаней (DM)

Сайти	S1	S2	S3	S4
S1	0	5	9	9
S2	5	0	14	4

S3	9	14	0	11
S4	9	4	11	0

2.3 Запропонований метод ефективного розподілення баз даних

У цьому розділі запропоновано метод, будуть представлені функції вартості, які будуть використані для розрахунку вартості перерозподілу фрагментів.

2.3.1 Представлення початкових даних

Запропонований метод передбачає наявність попередньо зібраної інформації про частоту оновлення, яка бере участь у розподілі фрагментів між сайтами. Щоб належним чином виконати процес розподілу, кожен фрагмент повинен бути призначений одному або кільком сайтам у DDBS на початковому етапі розподілу. Однак, оскільки проблема розподілу передбачає пошук оптимального розподілу фрагментів по сайтах. Тоді для заданого набору фрагментів різного розміру та кількості мережевих сайтів, де кожен сайт має певну кількість виконаних запитів, оптимальна модель розподілу передбачатиме мінімізацію загальної вартості оновлення у всій мережі без порушення обмежень сайту.

У запропонованому методі передбачається, що якщо один і той самий запит надходить на кількох сайтах, він буде розглядатися як інший запит і може мати різні значення частоти. Однак для досягнення оптимального динамічного перерозподілу ми використовуємо значення частоти оновлення запиту для всіх розподілених фрагментів, щоб використовувати їх щоразу, коли потрібен процес перерозподілу. Щоб створити початкову фазу розподілу фрагментів (яка може містити дублювання фрагментів), запропонована методика буде припускати, що фрагменти вже розподілені між сайтами мережі на основі заданої матриці пошуку (RM) і матриці частоти запитів (QF). Запропонований підхід передбачає, що запити на пошук оброблятимуться без витрат на зв’язок шляхом

виділення кожного фрагмента сайту, що запитує. Наприклад, фрагмент F3 в RM (таблиця 3) може бути запитаний лише запитами Q3 і Q4. А оскільки запити Q3 і Q4 видаються на сайтах S1 і S3, то фрагмент F3 буде виділено на обидва сайти.

У цій роботі представлена нова техніка перерозподілу отриманих фрагментів із початкової фази без зайвих дій. Ця техніка перерозподілу виконується з використанням матриці оновлення лише разом із частотою її запитів, оскільки запит на оновлення зазвичай вимагає більше витрат, ніж запити на пошук. Таким чином, таблиці 4 і 5, які є матрицями UM і QF, будуть використовуватися як вхідні дані для побудови нової матриці під назвою Матриця частоти оновлення фрагментів, FUFM (таблиця 10) як основа для перерозподілу. Матрицю FUFM можна визначити як значення оновлень запитів на певному сайті для уражених фрагментів.

Матриця FUFM буде використана для створення таблиці кумулятивної частоти оновлення, CUFT (таблиця 11). І помноживши CUFT на DM, ми отримаємо матрицю оплати фрагмента (FUPM), представлену в таблиці 12. Нарешті, використовуючи матрицю FUPM, сайт із максимальним значенням вартості оновлення для конкретного фрагмента буде обрано як сайт-кандидат для зберігання цього фрагмента. Однак, якщо є якісь порушення обмежень сайту для сайту-кандидата, фрагмент буде переміщено на сайт із наступним найвищим значенням вартості оновлення. Більше того, якщо існує більше одного сайту, що має однакове значення вартості оновлення для певного фрагмента, на цих сайтах буде запущена процедура пріоритету фрагментів (FP), щоб призначити цей фрагмент сайту з найвищим значенням FP, а згодом ця дія

гарантує, що поточна копія фрагмента буде єдиною копією у всій мережі. Фази запропонованої нами системи зображені на малюнку 8.

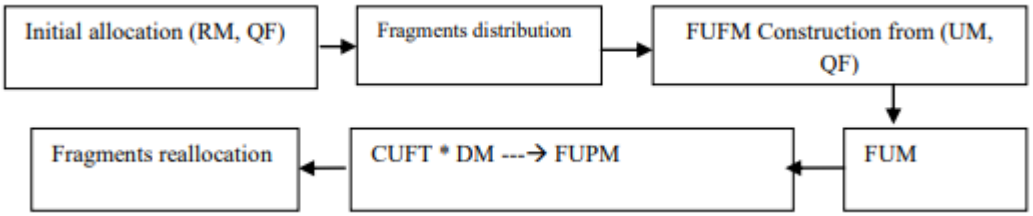

Рисунок 8 - Фази перерозподілу

2.3.2 Процедура задання пріоритету фрагментів

Процедура задання пріоритету фрагментів (FP) дає кількість звернень до фрагментів, тобто скільки разів запити кожного сайту (QS) досягають передбачуваний фрагмент. Це використовується, якщо є запити на той самий фрагмент Fi з більш ніж одного сайту з однаковим значенням вартості оновлення для цього фрагмента. Таким чином, щоб уникнути дублювання фрагментів на сайтах, FP буде розраховано для кожного сайту і використовуватиметься для прийняття рішення про виділення фрагмента Fi сайту, який має найвище значення FP.

$$FP(Sj, Fi) = \sum (QFhi * QShi) * DMj,h,\ 1 <= j <= m \qquad (1)$$

$$QShi = \sum RMhi + \sum UMhi,\ 1 <= h <= h \qquad (2)$$

Рівняння (1) використовується для обчислення FP, щоб уникнути дублювання фрагментів. Однак рівняння (2) видає сукупну кількість звернень, здійснених запитами.

2.3.3 Оцінка функції вартості запитів

Витрати виникають внаслідок запитів у процесі доступу та оновлення фрагментів на певному сайті. Як згадувалося раніше, для обчислення FUPM ми беремо матрицю FUFM як вхідні дані та використовуємо такі функції вартості:

$$CUFT = \sum \sum (QFi * UF(Fi)) \quad (3)$$

$$DMij = Min (CCij),\ 1 <= i , j <= m \quad (4)$$

$$FUPM = \sum CUFT (Fi) * DMjj' ,\ 1 <= j, j' <= m \quad (5)$$

$$\sum Max (FUPMij),\ 1 <= i <= n \quad (6)$$

Рівняння (3) використовується для створення таблиці CUFT, рівняння (4) застосовує «мінімальний» алгоритм до матриці витрат на зв’язок, рівняння (5) використовується для побудови FUPM і, нарешті, рівняння (6) представляє рішення про перерозподіл. Загалом, процес перерозподілу можна вважати оптимальним при досягненні схеми розподілу, яка призводить до мінімальних загальних витрат на оновлення запитів. Оскільки ця задача практично нерозв’язна, то вирішино використовувати евристичний метод.

2.4 Метод розподілення фрагментів бази даних

Алгоритм перерозподілу фрагментів

вхідні дані:

для кожного відношення {1,… , K}

Q = { Q1……, Qn} // набір запитів

F = { F1……., Fn } // РБД фрагменти

S = {S1..... .., Sm} // набір мережевих сайтів RM матриці, матриця UM, QF матриці і матриці вартість зв'язку

Вихід: фрагмент схеми перерозподілу

// початковий розподіл фаз

Початок

Початок первинний розподіл використовуючи RM і QF матриці

Матриця витрат на відстань (DM) = min (CCM)

Calculate_DM(CCM);

Calculate_CUFT (QFM, FUFM);

Calculate_FUPM (DM, CUTF);

Кінець

// Формування матриці відстаней

Min(CCM);{

// Вихід – це матриця відстаней і значення найкоротшого шляху

Для k=1 до m do

Для i=1 до m do

Для j=1 до m do

DMij = мінімум (ccij,ccik + cckj)

зберегти (SPA[i],SPA[j], valueij);

End for j

End for i

End for k

// Сума обчислення вартості пошуку та оновлення

Calculate_CUFT (QFM,FUFM);

// Вихід – таблиця CUFT;

Початок Для j=1 до m // сайти

Для h=1 до k // запити сайту

Для i=1 до n // фрагменти

*CUTFjhi = QFjh * UMhi*

End i;

End h;

End j;

End ;

// Розрахунок оплати витрат на оновлення сайтів для доступу до фрагментів Calculate_FUPM(CCM, SRUM);

вихід – FUPM;

Початок

Для j=1 до m //сайти

Для h=1 до m //сайти

Для i=1 до n // фрагмент

*FUPMji= CUFTji * DMj,h*

End j;

End h;

End i;

End ;

// Фаза перерозподілу

Почати після розподілу або перерозподілу за допомогою матриці (FUPM)

// Розміщення фрагмента Fi на сайті з найвищою вартістю оновлення

Початок

Для j=1 до m // сайтів

Для i=1 до n // фрагменти

Розподіл (Fi to Sj) = Макс (FUPMji)

Якщо (Sj .constraints-violated= true),

виділення Fi для Sj+1 має друге найвище значення вартості оновлення

кінець, якщо

If Sj .Fi (FUPM_value) = Sj+!.Fi (FUPM_value)

FP (Sj ,Sj+1,Fi ,FUPM_value);

End if

End for i

End for cni

End;

// Якщо для кількох сайтів потрібен той самий фрагмент, скористайтеся процедурою пріоритету фрагментів

Allocate-FP(Sj ,Sj+1,Fi ,FUPM_value);

//Вихід – це розміщення передбачуваного фрагмента на його новому сайті

Початок

// побудувати матрицю доступу

Для j=1 до m do// сайтів

Для h=1 до k зробити // запити

Для i=1 до n зробити // атрибути

*ACCji =(RMi,h,j+UMi,h,j) * QFji)*

End for

End for

End for

// Щоб обчислити пріоритет фрагмента (FP) і точно виконати фазу перерозподілу, виберіть значення (v), таке що Payv(Fi) = Maxs m і знайти сайт з максимальною вартістю оновлення для фрагмента (Sj). Виділити Fi на Sj (v)

// розподілити фрагмент і у відповідний сайт Sj (v)

Якщо багато сайтів вимагає одного фрагмента Fi

Запустіть процедуру FP на всіх сайтах, які потребують Fi

Розподіліть фрагмент у sj, який має максимальний FP

ENDFOR;

//

Для j=1 до m do

Для i=1 до n do

Для h=1 до m do

*FP(Fi , Sj)= Max (ACChji * DMjh), // враховуючи, що j' <> j*

End for j

End for i

End for h

End;

2.5 Оцінка ефективності розподілення фрагментів бази даних

Кожного разу, коли атрибут Ai призначається сайту Sk з найвищим значенням вартості оновлення, вартість зв'язку буде різко мінімізована.

Максимум (Sj) [∑j ∑k (QFjk * UMkl)], призначити тобто атрибут Ai сайту Sj, на якому кількість посилань на оновлення запиту на Ai є найбільшою.

Доказ цієї теорії полягає в наступному:

Припустимо, що вартість зв’язку надається як підсумок усіх доступів до оновлення, здійснюваних запитами сайту, до таких атрибутів, що:

Comm_Cost = QF11 * UM11 + QF21 * UM12 + ……. + QFjk * UMkl

= ∑ ∑ ∑ (QFjk * UMkl)

За умови, що цільовим сайтом розподілу є Sh і h [1….m], враховуючи, що h < > j, отже:

Comm_Cost=∑ ∑ [∑ QFjk * UMkl]

=∑ ∑ [∑ QFjk * UMkl]

=∑ ∑ [∑ QFjk * UMkl – QFhk * UMkl]

=∑ ∑ ∑ QFjk * UMkl – ∑ ∑ QFhk * UMkl

І вважаючи, що ∑∑ ∑ QFhk * UMkl є констанотою, то максимізація ∑ ∑QFhk * UMkl мінімізує витрати на зв'язок. Однак, якщо сайт Sh і Sj збігаються, де h,j [1…..m], тоді перерозподіл буде заснований на виборі max ∑ ∑ QFhk * UMkl

Як згадувалося раніше, два способи вирішення проблеми розподілу через ймовірності та евристичні методи. Запропонований нами підхід використовує концепції обох і таким чином намагається підвищити продуктивність DDBS в динамічному середовищі таким чином, щоб рішення про перерозподіл фрагментів було прийнято на основі змін інформації запиту,

щоб відобразити динамічну природу DDBS та мінімізувати вартість зв'язку та час відповіді на запит. Він вводить спрощену модель для процесу перерозподілу фрагментів. Крім того, мінімізація матриці витрат на зв'язок для отримання матриці витрат на відстань (DM) є ключовим фактором у підвищенні продуктивності DDBS. Однак у нашому методі процес перерозподілу заснований лише на вартості оновлення, оскільки операції оновлення потребують більше витрат, ніж вилучення, особливо коли фрагменти реплікуються на кількох сайтах під час початкового розподілу.

Таким чином, запропонований метод зосереджений на пошуку оптимального методу перерозподілу фрагментів для покращення продуктивності DDBS за рахунок мінімізації вартості зв’язку та часу відповіді на запит. Зрештою, забезпечення кращої доступності даних шляхом підтримки однієї копії фрагмента через DDBS на етапі після розподілу.

Переваги та недоліки нашого методу підсумовані в наступних пунктах:

1. Процес перерозподілу є динамічним методом, заснованим на отриманій інформації (значення частоти пошуку та оновлення) з існуючої DDBS. Отже, будь-які зміни запитів сайту та їх частоти вплинуть на процес перерозподілу.
2. Це гарантує відсутність дублювання фрагментів (дублювання дозволено лише на етапі початкового розподілу) на етапі після розподілу, тобто коли є більше одного сайту з однаковим значенням вартості оновлення, у цьому випадку буде використовуватися процедура пріоритету фрагментів вказати оптимальне місце для розміщення. На етапі після розподілу дозволяється лише дублювання первинного ключа між сайтами.

3. Менша складність обчислень. Наприклад, якщо k — кількість запитів, m — кількість сайтів, а n — кількість фрагментів, то складність перерозподілу фрагментів буде O (n + m^2 * n).

Єдиний недолік цього методу полягає в тому, що коли інформація запитів постійно змінюється швидше або коли кількість сайтів і фрагментів значно збільшується, метод буде складнішим.

Висновки

У цій роботі представлена динамічна евристична модель перерозподілу для пошуку оптимального рішення для перерозподілу фрагментів у середовищі DDBS. Ідея цього підходу полягає в тому, щоб перерозподілити фрагменти даних у їх оптимальне розташування на основі максимальної вартості оновлення фрагмента на призначеному для розміщення місці. Цей метод «максимальної вартості оновлення» забезпечує оптимальне рішення та враховує обмеження сайту та топологію мережі.

Розділ 3. Розробка засобів реалізації ефективного розподілення фрагментів розподілених баз даних

3.1 Початкове розміщення фрагментів бази даних

Щоб перевірити нашу методику, ми впровадили її на існуючу DDBS. Враховуючи вимоги та на основі матриць RM та QF, ми отримали початкову таблицю розподілу (таблиця 9).

Таблиця 9 - Початковий розподіл

/F	F1	F2	F3	F4	F5	F6
1	0	1	1	1	1	1
2	1	1	0	1	1	1
3	1	1	1	1	1	0
4	1	1	0	1	0	1

Основною проблемою на цьому етапі є дублювання фрагментів (наприклад, F1 з'являється в трьох сайтах). Однак ця проблема буде вирішена після розподілу (фаза перерозподілу).

Також маємо матрицю вартості зв’язку яка дана в таблиці 10, та матрицю мінімальної вартості яка вираховується з матриці зв’язку і дана в таблиці 11.

Таблиця 10 - Матриця витрат на зв'язок

Сайти	S1	S2	S3	S4
S1	0	5	9	18
S2	5	0	16	4
S3	9	16	0	11
S4	18	4	11	0

Таблиця 11 - Матриця відстаней (DM)

Сайти	S1	S2	S3	S4
S1	0	5	9	9
S2	5	0	14	4
S3	9	14	0	11
S4	9	4	11	0

Матриця пошуку RM (таблиця 12) і матриця оновлення UM (таблиця 13), що описує поведінку пошуку та оновлення відповідно для всіх запитів.

Таблиця 12 - RM

Запит/Фрагмент	F1	F2	F3	F4	F5	F6
Q1	2	3	0	1	0	0
Q2	2	0	0	0	1	1
Q3	6	0	3	0	2	0

Q4	3	0	2	0	0	0
Q5	0	1	0	2	0	1

Таблиця 13 -UM

Запит/фрагмент	F1	F2	F3	F4	F5	F6
Q1	0	0	2	0	1	0
Q2	0	1	1	0	0	2
Q3	3	0	0	2	0	1
Q4	0	1	0	0	1	1
Q5	0	0	2	0	1	0

Також визначена частотна матриця для запитів (QF), яка показує частоту виконання запитів на кожному сайті (табл. 14 зображує це).

Таблиця 14 - Частота запитів

Запит/сайт	S1	S2	S3	S4
Q1	0	2	3	1
Q2	0	3	0	0
Q3	2	0	1	0
Q4	0	0	4	0

Q5	1	1	0	2

3.2 Процес перерозподілу фрагментів бази даних

Оскільки, як згадувалося раніше, FUFM будується з використанням матриць UM і QF.

Таблиця 15 - Матриця FUFM

S	Q	QF	F1	F2	F3	F4	F5	F6
S1	Q3	2	3	0	0	2	0	1
	Q5	1	0	0	2	0	1	0
S2	Q1	2	0	0	2	0	0	1
	Q2	3	0	1	1	0	0	2
	Q5	1	0	0	2	0	1	0
S3	Q1	3	0	0	2	0	1	0
	Q3	1	3	0	0	2	0	1
	Q4	4	0	1	0	0	1	1
S4	Q1	1	0	0	2	0	1	0
	Q5	2	0	0	2	0	1	0

Використання матриці, ми отримуємо таблицю CUFT, яка створюється шляхом множення кожного значення QF на відповідне значення UM для кожного запиту на кожному сайті, а потім намацування результатів на основі сайтів, як представлено в таблиці (16).

Таблиця 16 - Матриця CUFT

S/F	F1	F2	F3	F4	F5	F6

S1	6	0	2	4	1	2
S2	0	3	9	0	3	6
S3	3	4	6	2	7	5
S4	0	0	6	0	3	0

Потім, щоб створити FUPM, DM матриця множиться на матрицю CUFT як показано в таблиці 17.

Таблиця 17 - FUPM

S/F	F1	F2	F3	F4	F5	F6
S1	27	51	153	18	105	75
S2	72	56	118	48	115	80
S3	54	42	210	36	84	102
S4	87	56	120	58	98	97

Нарешті, на основі таблиці FUPM буде виконано процес перерозподілу фрагментів. Кожен фрагмент буде перерозподілено на сайт, який має найвищу вартість його оновлення. З таблиці 17 можна помітити, що сайти S2 і S4 мають однакове значення вартості оновлення для F2. Але фрагмент F2 розподіляється на сайті S2, оскільки пріоритет фрагмента (FP) сайту S2 вищий, ніж у сайту S4. У таблиці 18 наведено остаточні результати перерозподілу.

Таблиця 18 - Розподіл фрагментів

F	F1	F2	F3	F4	F5	F6

S	S4	S2	S3	S4	S2	S3

3.3 Оцінка ефективності запропонованого методу розподілення фрагментів бази даних

У таблиці 18 показано розподіл фрагментів для сайтів на початковій фазі розподілу відповідно до нашого прикладу, розглянутого в розділі (VA). Він дає розподіл фрагментів між сайтами для існуючої DDBS перед застосуванням нашого методу, є кілька запитів, які запитують атрибути для отримання 19 виділень. На відміну від цього, у таблиці 19 показано процес перерозподілу після використання нашого підходу під час пост-розподілу, який дає лише 6 розподілів після застосування нашого методу, що матиме великий вплив на покращення продуктивності DDBS, що чітко показано у відмінностях між початковим та кінцевим розподілом для фрагменти (таблиця 20).

Таблиця 18 - Початковий розподіл

S/F	F1	F2	F3	F4	F5	F6
S1	0	1	1	1	1	1
S2	1	1	0	1	1	1
S3	1	1	1	1	1	0
S4	1	1	0	1	0	1

Таблиця 19 - Розподіл фрагментів

F	F1	F2	F3	F4	F5	F6
S	S4	S2	S3	S4	S2	S3

Таблиця 20 і малюнок 9 показують кількість фрагментів, розподілених по сайтах на початковій і пост-фазах розподілу.

Таблиця 20 - Оцінка ефективності методу

Номер сайту	Кількість фрагментів (початкове)	Початкове розповсюдження	Кількість фрагментів (пост)	Після розповсюдження	Оптимізація
S1	5	19	-	6	-
S2	5		2		60%
S3	5		2		60%
S4	4		2		50%

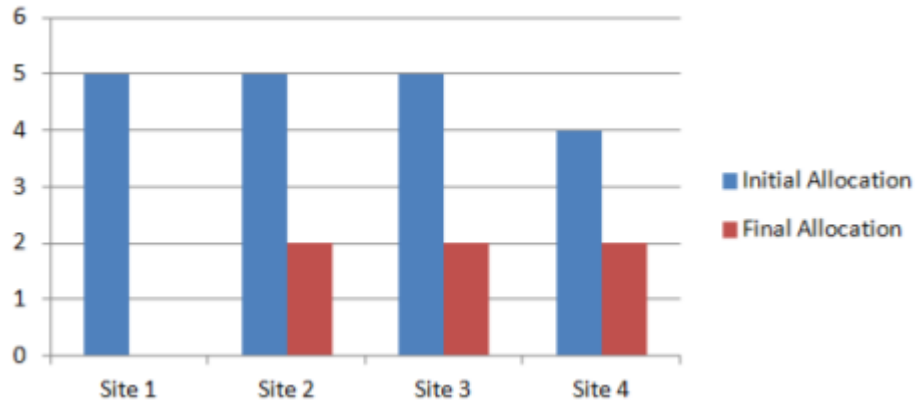

Рисунок 9 - Процес розподілу

Висновок

Запропонований метод покращує загальну продуктивність DDBS і призводить до значного зниження частоти міграцій фрагментів з однієї сторони на іншу і, отже, зменшує витрати на передачу даних. Якщо для певного фрагмента існує більше одного сайту, що має однакове значення вартості оновлення, цей фрагмент буде розподілено на оптимальному сайті відповідно до запропонованої нами процедури пріоритету фрагментів (FP). Цей метод гарантує уникнення дублювання фрагментів за рахунок використання процедури FP, що призводить до низької обчислювальної складності.

Розділ 4. Розробка стартап проекту

4.1 Опис ідеї стартапу

Ідея створення програм для моделюваня розподілених систем не нова. Однак не один із наявних додатків не має відкритого вихідного коду. Розроблений програмний продукт призначений для демонстрації розробленого алгоритму ефективного розподілення фрагментів баз даних у динамфсному средовищі. На даному етапі розроблений продукт не може виступати як повноцінний продукт, однак, за умови подальшої розробки можна спланувати ідеї, скласти стратегію розвитку та дій для подальшого просування та створення конкурентно-спроможного продукту.

У даному розділі магістерської дисертації буде описано розробку стартапу програми для проектування та моделювання розподілених систем, який можна використати як модуль в сторонніх рішеннях. У таблиці 4.1 показано зміст запропонованої ідеї, напрямки для застосування, переваги та вигоди у цих напрямках та різниця цієї ідеї від існуючих рішень.

Таблиця 4.1 - Зміст та ідея розроблюваного стартап-проекту

Зміст та ідея стартапу	Можливості та напрямки застосування	Переваги для користувача
Програма перерозподілу фрагментів у середовищі DDBS	Використання розробленого способу у сторонніх додатках для перерозподілу фрагментів у середовищі DDBS	Можливість эфетктивно прерозподілювати фрагменти у середовищі DDBS

	Використання отриманої інформації у програмах проектування розподілених систем	Можливість отримати інформацію про розташування фрагментыв та ограничения систем
	Прогнозування часу відгуку системи та використання ресурсів сайтів у розподіленому серидовищі	Прогнозування стійкості системи та часу відгуку у фоновому режимі

Аналіз потенційних техніко-економічних переваг (тобто аналіз тих факторів, що відрізняються розроблена ідея від існуючих аналогів та замінників) показано у таблиці 4.2. Прямими аналогами є Cisco Modeling Labs.

Таблиця 4.2 -Визначення слабких, сильних та нейтральних характеристик ідеї

№	Технічні та економічні характеристики розроб. ідеї-проекту	Розроб. проект	Конкурент	W	N	S
1	Можливість застосовувати у інших проектах чи платформах	+	-			+
2	Підтримка можливості будування розподіленої системи	+	+		+	

Таблиця 4.2 - продовження

№	Технічні та економічні характеристики розроб. ідеї-проекту	Розроб. проект	Конкуренти	W	N	S
3	Підтримка можливості оптимізації розташування фрагментів системи	+	-			+
4	Візуалізація отриманих результатів	+	+		+	

У таблиці вище було визначено слабкі, сильні та нейтральні характеристики ідеї потенційного товару, що є фундаментальним для формування його конкурентоздатності. Як видно із таблиці, основною перевагою ідеї є можливість використання у інших проектах чи платформах, на що і варто робити ставку. Також великою перевагою є можливість автоматичної оптимізації розташування фрагментів розподіленої бази даних у системі за допомогою запропонованого алгоритму, на що і потрібно робити ставку.

4.2 Технологічний аудит ідеї

У таблиці 4.3 показано обрані технології для реалізації ідеї проекту. Розробка програми на мові програмування C#, що побудовано для роботи із Asp .Net (тобто використовує бібліотеки данної платформи), алгоритм розподілення фрагментів бази даних у розподіленії систем, що запропоновано у цій магістерській дисертації.

Таблиця 4.3 -Можливість реалізації стартап проекту

№	Головна ідея для стартап проекту	Необхідні технології	Наявність	Доступність
1	Програма перерозподілу фрагментів у середовищі DDBS	Мова програмування C#	Є в наявності	Доступна безкоштовно
2		Бібліотеки для роботи із Asp .Net	Є в наявності	Доступна безкоштовно
3		Алгоритм перерозподілу фрагментів у середовищі	Є в наявності	Доступна безкоштовно

Аналіз ринкових можливостей запуску стартап-проекту

У цьому пункті буде визначено ринкові можливості, які можна використати під час ринкового впровадження проекту, а також ринкові загрози, що можуть виникнути під час реалізації проекту. Цей аналіз дозволяє

спланувати напрямки розвитку проекту враховуючи стан ринкового середовища, потреб клієнтів та пропозицій конкурентів. Таблиця 4.4. показує стан потенційного ринку для розроблюваного стартапу.

Таблиця 4.4 - Характеристика існуючого ринку стартап-проекту

№	Показник	Характеристика
1	Кількість головних гравців, од	> 2
2	Динаміка ринку (якісна оцінка)	У сфері програм ефетивного та автоматичного розподілення фрагментів бази даних ринок знаходиться на етапі стрімкого зросту
3	Наявність обмежень для входу (вказати характер обмежень)	Велика кількість конкурентів, що пропонують більший набір можливостей для аналізу отриманих результатів
4	Специфічні вимоги до стандартизації та сертифікації	Відсутні

Таблиця 4.5 - Основна характеристика потенційних клієнтів

№	Потреба у стартап-проекті	Потенційні клієнти	Відмінності у поведінці різних	Вимоги споживачів до товару

			потенційних цільових груп клієнтів	
1	Підвищений інтерес до ефективного розподілення даних	Користувачі, що зацікавлені в оптимізації розподілених систем	Архітектура розподілених систем	Необхідність зручного моделювання та оброблення даних для подальшого аналізу
2	Необхідність відстежувати та аналізувати стану системи подальшого використання та аналізу отриманої інформації	Корпоративні користувачі, що мають системи, які використовують інформацію для подальшої обробки	Безпосереднє використання у моніторингу стану системи	Можливість отримувати зібрану інформацію через використання відкритого API

Таблиця 4.6 - Основні фактори загроз при виході на ринок

№	**Фактор**	**Зміст**	**Реакція**
1	Висока	Існує велика	Реагувати на

	конкуренція	кількість подібних додатків, які подібні за своїм функціоналом	зворотній зв’язок від користувачів, їх побажання; Оновлення функціоналу та підтримка програми
2	Мала кількість функцій та можливостей	Ринок насичений подібними додатками, тому необхідно на нього вийти якомога скоріше, реалізувавши основні функції, які забезпечить конкурентоспроможний продукт	Затримка виходу на ринок для реалізації необхідного функціоналу
3	Економічний	Фінансова обмеженість, що може затримати вихід на ринок	Складання точного та чіткого бізнес-плану для залучення інвесторів

Таблиця 4.7 - Основні фактори можливостей для розроблюваного стартап-проекту

№	Фактор	Зміст	Реакція
1	Залучення клієнтів	Склавши чітку	Спланувати

		маркетингову кампанію, залучити клієнтів	маркетингову кампанію, провести презентацію продукту та можливостей, що він має
2	Залучення інвесторів	Створення бізнес-плану допоможе залучити інвестиції у проект, що дозволить найняти персонал для прискорення виходу на ринок	Збільшення функціоналу, прискорений вихід на ринок

4.3 Аналіз конкуренції на ринку

Таблиця 4.8 - Аналіз конкуренції на ринку

Основні властивості конкурентів	**Проявлення властивості**	**Вплив**
Схожість продукту з аналогами	Велике насичення подібних продуктів на ринку, що унеможливлює створення унікальних функцій	Виконувати експерименти зі створенням нових функцій
Доступність до технологій, що використовуються у проекті	Система не потребує важкодоступних технологій, та є можливість реалізувати даний алгоритм на різних технологіях що дає можливість швидкого роспувсюдження.	Реагувати на вихід нових продуктів на ринок, їх функціонал

	Прямі конкуренти	**Потенційні конкуренти**	**Клієнти**	**Замінники**
Складові аналізу	Cisco Modeling Labs	Net tracker	Компанії та користувачі, що зацікавленні у оптимізації побудови розподілених систем та ефетивного зберугання фрагментів	Усі додатки, що мають функціонал для моделювання таких систем
Висновки	Даний конкуренти мають перенасичений функціонал, а пошук необхідної інформації доступний	Необхідно враховувати, що функціонал проекту обмежений тільки інструментам	Аудиторія у даного додатку досить велика	Є безліч додатків, що мають подібний функціонал, однак, вони всі розроблені великими корпораціями, що може викликати недовіру

	після довгого блукання у меню та налаштуваннях	и побудови системи		користувача у плані безпеки збереження отриманої інформації

Таблиця 4.10 - Основні фактори конкурентоспроможності проекту

	Фактор конкурентоспроможності	**Обґрунтування**
	Простий та зручний інтерфейс	Враховуючи недоліки користувацького інтерфейсу та досвіду конкурентів, можна створити більш приємний та простий інтерфейс
	Бета-тестування	Впровадження бета-тестування для виявлення найбільш важливих нових функцій аби максимального адаптувати продукт до потреб кінцевого споживача
	Кросплатформеність ядра (алгоритму)	Ядро, тобто сам алгоритм, можна запустити на будь-якій платформі, що має підтримку C#, що робить можливим вихід на більш широкий ринок

Таблиця 4.11 - Відносна оцінка стартап-проекту до наявних конкурентів

№	Фактор конкурентоспроможності	Бали 1-20	Оцінка у відношенні розроблюваного проекту до конкурентів						
			-3	-2	-1	0	1	2	3
1	Простий та зручний інтерфейс	17		+					
2	Бета-тестування	12			+				
3	Кросплатформеність ядра (алгоритму)	10				+			

Таблиця 4.12 - SWOT-аналіз стартап-проекту

Сильні сторони:	Слабкі сторони:
- Зручний та простий інтерфейс; - Можливість використовувати алгоритм на будь-якій платформі; - Врахування досвіду конкурентів; - Бета-тестування для виявлення недоліків та	- Досить велика конкуренція - Наявність більшого функціоналу у конкурентів - Фінансова обмеженість, що може затримати вихід на ринок

побажань клієнтів	
Можливості: - Отримати можливість продати ядро (алгоритм) програми	Загрози: - Відсутність конкурентного функціоналу для залучення клієнтів; - Висока конкуренція

Таблиця 4.1 - Альтернативні сценарії для впровадження розроблюваного стартап-проекту

	Можлива альтернатива	**Ймовірність отримання ресурсів**	**Термін реалізації**
	Відкриття вихідного коду та розповсюдження у форматі open-source	Дуже низька. Основними ресурсами будуть пожертви від відданих користувачів	Миттєво
	Представлення проекту на конференціях та презентаціях	Висока. Представленням на конференціях можна поширити інформацію про розроблений продукт та залучити клієнтів	Від півроку до 5 років
	Участь у навчальних інтесивах	Середня. Отримання ресурсів	Від місяця до кількох років

	по проектуванню систем	залежить у зацікавленості навчальної організації та фінансуванні цієї організації	

Таблиця 4.1 - Переваги та недоліки розроблюваного стартап-проекту у порівнянні з конкурентами

	Профільні групи	**Бажання та готовність сприймати продукт**	**Попит у аудиторії**	**Конкуренція на ринку**	**Складн. виходу на ринок**
	Приватні компанії	Не всі готові прийняти	Необхідний вже	Середня конкуренція	Помірно-складна
	Навчальні організації	Не всі готові прийняти одразу	Необхідний вже	Середня конкуренція	Середня
	Дослідницькі	Не всі готові прийняти	Необхідний вже	Середня конкуренція	Середня

	програми	одразу			
Висновок: надано перевагу у роботі із приватними компаніями, однак розглядаються пропозиції від усіх цільових груп					

Таблиця 4.15 - Базова стратегія розвитку для стартап-проекту

№	**Альтернатива**	**Обраний спосіб для розвитку та впровадження стартапу**	**Ключові конкурентоспроможні позиції відповідно до обраної альтернативи**	**Базова стратегія розвитку**
1	Просування додатку за допомогою маркетингової компанії та участі у конференціях, презентаціях, тощо	Вплив на цільову аудиторію	Безкоштовне поширення для благодійних та наукових цілей	Впровадження нового функціоналу згідно відгукам користувачів та бета-тестувальників

Таблиця 4.16 - Визначення базової стратегії конкурентної поведінки

	Проект-першопрохідець	**Тип розвитку**	**Схожість проекту з конкурентами**	**Стратегія поведінки на конкурентному ринку**
	Ні	Змішаний тип розвитку: пошук нових споживачів та залучення споживачів конкурентів	Продукти конкурентів схожі, тому компанія буде копіювати основні характеристики конкурентів	Оновлення функціоналу згідно побажань користувачів та впровадженню їх у конкурентів

Таблиця 4.17 - Стратегія для позиціонування продукту

	Вимоги до продукту від аудиторії	**Базова стратегія розвитку**	**Ключові конкуренто-спроможні позиції стартап-проекту**	**Асоціації, які формують сприйняття проекту на ринку**
	Впровадження нових	Збільшити кількість	Швидка реакція на	Ефективне

	функцій, часті оновлення	персоналу, досліджувати поведінку користувачів конкурентів, їх побажання	недоліки, їх виправлення	розподілення даних

4.4 Розроблення маркетингової програми стартап-проекту

Таблиця 4.18 - Концепція продукту

№	Потреба	Вигода	Існуюча перевага та дії для створення цієї переваги
1	Проектування розподілених систем	Використання отриманої інформації для аналізу та планування системи	Можливість проектувати систему та автоматичної оптимізації. Використовувати отриману інформацію для глибокого аналізу та планування
2	Детальний звіт про модель	Можливість переглядати детальний звіт про модель	Частина програми, що відображає ключові характеристики
3	Оновлення	Врахування недоліків які виявленні та впровадження нового функціоналу	Створення служби підтримки для отримання інформації про недоліки та побажання користувачів

Таблиця 4.19 - Три рівні моделі товару стартап-проекту

Рівень	Складові та характеристика

<table>
<tr><td>I. Товар за задумом</td><td colspan="2">Програмний алгоритм для ефективного розподілу фрагментів бази даних у системі, а також приклад реалізації на веб додатку</td></tr>
<tr><td rowspan="5">II. Товар у реальному виконанні</td><td>Властивості/характеристики</td><td>Розмір</td></tr>
<tr><td>1. Алгоритм для ефективного розподілу фрагментів бази даних у системі</td><td>24 МБ</td></tr>
<tr><td>2. Реалізація у веб додатку</td><td>15 МБ</td></tr>
<tr><td colspan="2">Якість: внутрішнє автоматизоване тестування алгоритму, відправлення інформації про помилки, система зворотного зв’язку</td></tr>
<tr><td colspan="2">Пакування: продаж реалізації через систему підписок</td></tr>
<tr><td></td><td colspan="2">Марка: DisModel</td></tr>
<tr><td rowspan="2">III. Товар із підкріпленням</td><td colspan="2">До продажу: програмний код</td></tr>
<tr><td colspan="2">Після продажу: запакований архів у форматі dll</td></tr>
<tr><td colspan="3">За рахунок чого потенційний товар буде захищено від копіювання: захист від копіювання буде забезпечено стандартними засобами та правилами розповсюдження веб додатків</td></tr>
</table>

Таблиця 4.20 - Межі ціни для стартап-проекту

п/п	**Рівень цін на товари-замінники**	**Рівень цін на товари-аналоги**	**Рівень доходів цільової групи споживачів**	**Верхня та нижня межі становлення ціни на товар/послугу**
	0$ - 5$	0$ - 10$	> 5000$/місяць	0,99$-4,99$

Таблиця 4.21 - Система для формування ринку збуту продукту

п/п	**Закупівельна поведінка**	**Функції збуту постачальника**	**Глибина каналу збуту**	**Оптимальна система збуту**
	Продаж програми через систему продажу	Просування та забезпечення продажу продукту, залучення нових клієнтів	Одноріневий	Вертикальна (властність на продукт залишається у розробника)

Таблиця 4.22 - Визначення маркетингових комунікацій для стартап-проєкту

№	**Специфіка клієнтів**	**Канали зв'язку клієнтів**	**Позиціонування**	**Мета рекламного повідомл.**	**Концепція рекламного повідомл.**
1	Клієнти – покупці товару на вимогу	Конференції, виступи, презентації на спеціалізо-ва них заходах	Надання послуг у благодійних та комерційних цілях	Демонстр. можливостей додатку	Пояснення цільовій аудиторії переваг додатку та його можливостей

Висновок

У цьому розділі було проведено розробку стартап-проекту для програмного продукту, опис якого було представлено у попередніх розділах. У рамках цього розділу було описану ідею для стартап проекту, проведено порівняльний аналіз можливостей конкурентів та їх переваг та недоліків та вирішено наступні завдання:

1. Описано ідею та її мінімальний функціонал для виходу на ринок. Проаналізовано переваги та недоліки конкурентів, визначено переваги розробленого продукту та його можливий розвиток;
2. Проведено розгорнутий технічний аудит проекту;
3. Проведено аналіз ринку та можливості для запуску проекту на цьому ринку. Розроблено ринкову стратегію розвитку та план розвитку в умовах наявного ринку. Визначено цільові групи та способи просування проекту та продукту;
4. Описано план з маркетингового просування проекту на ринку та сформовано основний канал для продажу продукту, а також визначено цільові групи користувачів.

Отже, у цьому розділі було проведено аналіз можливостей запуску додатку у якості стартап-проекту, оцінено ризики та можливості його запуску.

Загальний висновок

У цій дисертації проаналізовано проблему оптимізації запитів до розподіленої бази даних у динамічному середовищі. У ході дослідження розроблено евристичний метод для ефективного розподілу фрагментів бази даних у розподіленому середовищі з урахуванням певних граничних значень системи, що відображають реальну постановку задачі.

У першому розділі проаналізовано найважливіші сучасні класифікації методів зберігання даних у розподілених базах даних, їх переваги та недоліки. Ми також проаналізували нюанси пошуку інформації в складних розподілених мережах, що в нашому випадку допомогло створити кращі математичні методи.

У другому розділі були описані всі побудовані математичні методи, кожна з яких описує принципово іншу мережу та моделює середовище часу, необхідного для пошуку інформації.

У третьому розділі були представлені експерименти, які були проведені з отриманими моделями з використанням реалізованого продукту. Експерименти показали, що запропонований метод оптимізації дозволяє ефективно розташовувати фрагменти в розподіленому середовищі.

У четвертому розділі представлені економічні розрахунки для цього програмного продукту та розраховані можливі економічні дивіденди для компаній, які скористаються цією розробкою.

Список використаної літератури

1. Абдалла Х., «Ефективний підхід до розміщення даних у розподілених системах», П’ята міжнародна конференція FTRA з мультимедіа та всюдисущої інженерії, (2011).

2. Абдалла Х., «Використання жадібного підходу для вирішення проблеми розподілу даних у розподіленому середовищі», щоб з’явитися в працях Міжнародної конференції 2008 року з паралельних і розподілених методів і додатків (PDPTA'08), (2008).).

3. Brunstrom A, Leutenegger ST і Simha R, «Експериментальна оцінка стратегій динамічного розподілу даних у розподіленій базі даних зі змінними робочими навантаженнями», у працях Міжнародної конференції з управління інформацією та знаннями 1995 року, Балтімор, штат Меріленд, США, (1995).), с. 395-402.

4. Apers P, “Розміщення даних у розподілених базах даних”, ACM Trans. Системи баз даних, вип. 13, № 3, (1988) вересень, с. 263-304.

5. Abiteboul S, Bonifati A, Cobena G, Manolescu I and Milo T, “Dynamic XML Documents with Distribution and Replication”, Proc. 2003 ACM SIGMOD Int'l Conf. Управління даними, (2003), с. 527-538.

6. Wilson B and Navathe SB, «Analytical Framework for the Redesign of Distributed Database», в Proceedings of the 6th Advanced Database Symposium, Tokyo, Japan, (1986), pp. 77-83.

7. Chiu G and Raghavendra C, “A Model for Optimal Database Allocation in Distributed Computing Systems”, Proc. IEEE INFOCOM 1990, вип. 3, (1990) червень, с. 827-833. [8] Chu WW, “Оптимальне розміщення файлів у кількох комп’ютерних системах”, IEEE Transaction on Computers, vol. C18, № 10, (1969).

8. Cormen TH, Leisorson CE, Rivest RL and Stein C, “Introduction to Algorithms 2nd Edition”, McGraw Hill (2001).

9. Lin WJ and Veeravalli B, “A Dynamic Object Allocation and Replication Algorithm for Distributed System with Centralized Control”, International Journal of Computer and Application, vol. 28, no. 1, (2006), pp. 26- 34.

10. Karimi Adl R and Rouhani Rankoohi SMT, “A new ant colony optimization based algorithm for data allocation problem in distributed databases”, Knowl Inf Syst 2009, vol. 20, (2009) January 23, pp. 349–373.

11. Menon S, “Allocating Fragments in Distributed Databases”, IEEE Transactions on Parallel and Distributed Systems, vol. 16, no. 7, (2005) July.

12. Hababeh IO, Ramachandran M and Bowring N, “A high-performance computing method for data allocation in distributed database systems”, Springer J Supercomput, vol. 39, (2007), pp. 3-18.

13. Singh A and Kahlon KS, "Non-replicated Dynamic Data Allocation in Distributed Database Systems", IJCSNS International Journal of Computer Science and Network Security, vol. 9, no. 9, (2009) September, pp. 176-180.

14. Sleit A, AlMobaideen W, Al-Areqi S and Yahya A, "A Dynamic Object Fragmentation and Replication Algorithm In Distributed Database Systems", American Journal of Applied Sciences, vol. 4, no. 8, (2007), pp. 613-618.

15. John LS, “A Generic Algorithm for Fragment Allocation in Distributed Database System”, ACM (1994).

16. Tâmbulea L and Horvat M, "Dynamic Distribution Model in Distributed Database", Int. J. of Computers, Communications & Control, ISSN 1841-9836, E-ISSN 1841-9844, Vol. III, (2008), Suppl. issue: Proceedings of ICCCC 2008, pp. 512-515.

17. Tamhankar A and Ram S, “Database Fragmentation and Allocation: An Integrated Methodology and Case Study,” IEEE Trans. Systems,Man and Cybernetics—Part A, vol. 28, no. 3, (1998) May.

18. Ozsu MT and Valduriez P, “Principles of Distributed Database Systems”, 2nd ed.Prentice-Hall International Editions, (1999).

19. Toadere T, “Graphs. Theory, Algorithms and Applications”, Editura Albastra, Cluj-Napoca, ROMANIA, (2002).

20. Upadhyaya S and Lata S, "Task allocation in Distributed computing VS distributed database systems: A Comparative study", IJCSNS International Journal of Computer Science and Network Security, vol. 8, no. 3, (2008) March.

21. Ulus T and Uysal M, “Heuristic Approach to Dynamic Data Allocation in Distributed Database Systems”, Pakistan Journal of Information and Technology, vol. 2, no. 3, (2003), pp. 231-239.

22. Rivera-Vega PI, Varadarajan R and Navathe SB, “Scheduling Data Redistribution in Distributed Databases”, in IEEE Proceedings of the Sixth International Conference on Data Engineering, (1990), pp. 166-173.

23. Dowdy LW and Foster DV, “Comparative models of the file assignment problem,” ACM Computing Survey, vol. 14, no. 2, (1982), pp. 287-313.

24. Wolfson O, Jajodia S and Huang Y, “An Adaptive Data Replication Algorithm”, ACM Trans. Database Systems, vol. 22, no. 2, (1997), pp. 255-314.

25. Wolfson O and Jajodia S, "An Algorithm for Dynamic Data Distribution", Proceedings of the 2nd Workshop on the Management of Replicated Data (WMRD-II), Monterey, CA, (1992) November.