МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЧОРНОМОРСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ПЕТРА МОГИЛИ

Кваліфікаційна наукова
праця на правах рукопису

ДВОРЕЦЬКИЙ МИХАЙЛО ЛЕОНІДОВИЧ

УДК 004.658.3

МОДЕЛІ ТА ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ ОПТИМІЗАЦІЇ СТРУКТУРИ БАЗИ ДАНИХ ВУЗЛА У КОРПОРАТИВНИХ ІНФОРМАЦІЙНИХ СИСТЕМАХ

05.13.06 – інформаційні технології

Подається на здобуття наукового ступеня кандидата технічних наук

Дисертація містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело _____М. Л. Дворецький

Науковий керівник:
Фісун Микола Тихонович,
доктор технічних наук, професор

Миколаїв – 2020

АНОТАЦІЯ

Дворецький М. Л. Моделі та інформаційна технологія оптимізації структури бази даних вузла у корпоративних інформаційних системах. – Кваліфікаційна наукова праця на правах рукопису.

Дисертація на здобуття наукового ступеня кандидата технічних наук за спеціальністю 05.13.06 – інформаційні технології. – Чорноморський національний університет ім. Петра Могили, Миколаїв, 2020.

Дисертаційна робота присвячена підвищенню рівня доступності даних та ефективності використання розподілених та територіально розосереджених комп'ютерних систем шляхом реалізації інформаційної технології оптимізації структури бази даних (БД) вузла розподіленої корпоративної інформаційної системи (РКІС) на основі статистики SQL-запитів. Це досягається за рахунок розробки відповідних моделей та інформаційної технології. Розроблені моделі описують SQL-запит, формалізують критерії оптимальності структури БД вузла РКІС та описують їх залежність від значення маркеру представленості даних. Інформаційна технологія базується на удосконаленій моделі SQL-запиту, методі аналізу ієрархій із представленням результату у вигляді вектору нечітких чисел і приведенням до чіткого значення та класифікацією нових даних відповідно до необхідності їх представлення на вузлі РКІС.

Об'єктом дослідження є процес оптимізації структури БД розподілених корпоративних інформаційних систем, в основі яких лежать реляційні бази даних. Предмет дослідження – моделі SQL-запитів реляційних БД та методи вибору кращої альтернативи структури вузла РКІС. Загальний науковий результат роботи – вирішення актуальної науково-практичної задачі створення моделей та інформаційної технології оптимізації структури бази даних вузла у корпоративних інформаційних системах.

У вступі обґрунтовано актуальність обраної теми дисертації і її зв'язок із державними науковими програмами й темами. Сформульовано мету й задачі

дослідження, характеристики наукової новизни, теоретичного і практичного значення отриманих результатів.

В першому розділі розглянуто підходи та методи підвищення швидкості роботи інформаційних систем (ІС). Загальним недоліком проаналізованих підходів визначено необхідність модифікації самої ІС, що у випадку закритої реалізації останніх робить неможливим їх використання. Розглядаючи проблеми використання «універсальних» ІС, відмічається тенденція використання набору окремих спеціалізованих рішень із подальшою синхронізацією даних. Холдингова структура об’єкта автоматизації також може обумовити необхідність створення та використання БД розподіленої структури. Під корпоративною інформаційною системою розуміємо сукупність ІС окремих підрозділів підприємства, об’єднаних загальним документообігом, таких, що кожна система виконує частину задач по управлінню прийняттям рішення, а всі системи разом забезпечують функціонування підприємства.

Розглянуто різні концепції представлення даних в розподілених БД, серед яких виділено комбіновану, що поділяє базу даних на логічні фрагменти та дозволяє мати довільну кількість фізичних копій кожного фрагмента. Ключовим фактором, який впливає на надійність і доступність БД, є локалізація посилань. Враховуючи закритість коду SQL-запитів РКІС, задача оптимального проєктування структури БД зводиться до пошуку оптимального розподілу даних по вузлах РКІС. Імітаційні моделі РКІС, що є вкрай складною системою, зазвичай дуже спрощені та не враховують ряд важливих факторів, що впливають на її поведінку.

Запропоновано накопичення статистичних даних виконання SQL-запитів РКІС до розподіленої БД, що дає змогу врахувати всі особливості РКІС. Парсинг текстів SQL-запитів дозволить виявити множини відношень, атрибутів та кортежів БД, задіяних у кожному запиті. Подальший аналіз накопиченої статистики дозволить виявити частоту та характер звернень до окремих відношень, їх атрибутів і кортежів. Вводиться набір критеріїв оптимальності структури БД та формулюється задача багатокритеріальної оптимізації.

Проаналізовано різні методи її розв'язання, серед яких виділяється метод аналізу ієрархій (MAI).

Зроблено висновок про доцільність досліджень, спрямованих на підвищення рівня доступності даних та ефективності використання розподілених та територіально розосереджених комп'ютерних систем шляхом реалізації інформаційної технології оптимізації структури БД вузла РКІС на базі статистики SQL-запитів. Для виконання оптимізації структури БД сформульовано ряд завдань, а саме: накопичення статистичних даних; розробка інформаційної технології парсингу тексту SQL-запитів; розробка реляційної та багатовимірної БД для зберігання та аналізу даних SQL-запитів; створення математичних моделей залежності критеріїв оптимальності структури БД вузла РКІС від граничного рівня маркеру представленості даних вузла; розв'язання задачі вибору найкращої альтернативи; класифікації нових записів відношень БД.

У другому розділі на основі реляційної моделі даних та операцій реляційної алгебри реалізовано модель SQL-запиту, що включає дані про застосунок, хост, користувача, а також набір відношень, атрибутів та кортежів БД, які входять до результуючого набору рядків (кортежів). Запит представляється у багатовимірній БД у базисі вимірів, що включає «відношення», «атрибут» та «кортеж» БД.

При виконанні аналізу щодо представленості даних для елементів вимірів вводиться характеристика «маркер представленості даних», що відображає рівень необхідності представлення даних на вузлі РКІС. Для кожного елементу значення маркеру приймається одне з множини лінгвістичних значень {«необхідно», «бажано», «не потрібно»}, що визначає ступінь необхідності представлення даних того чи іншого робочого місця, ролі користувача або застосунка.

Визначення значення маркеру при виконанні консолідації рядків таблиці фактів по значенням $<R, A, tup>$ виконується із використанням методу ковзного середнього із введенням вагових коефіцієнтів елементів вимірів. Виконавши переведення нечислової лінгвістичної змінної маркерів у числове значення («обов'язково» – «2», «необхідно» – «1», «бажано» – «0», «не потрібно» – «–1»,

«заборонено» – «–2»), визначено функцію агрегації маркеру представленості даних. При прийнятті рішення по представленості даних на віддаленому вузлі, виконується консолідація рядків таблиці фактів за $<R, A, tup>$ та розраховується значення маркеру для кожного її елементу.

Для розв'язання завдання формалізації критеріїв оптимальності граничного рівня маркеру представленості даних визначено наступні критерії оптимальності: «незалежність БД», «розмір БД» та «необхідність синхронізації». Для кожного з критеріїв визначено математичну модель, що описує його залежність від значення граничного рівня маркеру представленості даних на вузлі РКІС.

При синхронізації даних вузла РКІС запропоновано використання динамічного визначення періоду обміну даними. Наявність механізмів обчислення моменту наступної синхронізації даних дозволяє підвищити актуальність даних, та розвантажити ресурси серверів баз даних. При їх реалізації враховані наступні фактори: кількість змін у журналі транзакцій; якість цих змін (на основі частоти використання даних у користувацьких запитах визначаються коефіцієнти впливу на роботу вузла РКІС); значення точки актуальності даних; завантаженість серверу БД поточними запитами.

У третьому розділі розроблено метод вибору найкращої альтернативи для граничного рівня маркеру представленості даних. Використано метод аналізу ієрархій із автоматичною ініціалізацією матриці попарних порівнянь альтернатив відповідно до отриманих математичних моделей та нормалізації значень; із використанням елементів нечіткого логічного висновку для дефазифікації вектору глобальних пріоритетів; та виконанням класифікації нових даних із використанням алгоритму наївного Байєса.

При побудові ієрархічної моделі процесу прийняття рішення вибору кращої альтернативи було використано 4 рівня ієрархії, а саме: «мета», «зацікавлені особи», «критерії», «альтернативи». З метою спрощення задачі було визначено 5 альтернатив рівня маркеру представленості: «низький» (Н) – «–1», «нижче середнього» (НС) – «–0,5», «середній» (С) – «0», «вище середнього» (ВС) – «0,5», та «високий» (В) – «1». Перелік критеріїв оптимальності моделі на рівні ієрархії

«критерії» відрізняється для різних зацікавлених осіб. Використовуючи шкалу відносної важливості критеріїв та із залученням людини, що приймає рішення (ЛПР), якою на даному етапі виступає зацікавлена особа «Власник» об’єкту автоматизації, виконується побудова матриці попарних порівнянь для зацікавлених осіб. На третьому рівні ієрархії складаються відповідні матриці попарних порівнянь за критеріями оптимальності за кожною зацікавленою особою. Тут у ролі ЛПР виступає зацікавлена особа, за якою заповнюється матриця.

Наявність математичних моделей критеріїв оптимальності, сформульованих у другому розділі роботи, дозволяє виконати розрахунок та початкову ініціалізацію матриць попарних порівнянь альтернатив на базі числових значень маркеру представленості даних для кожної альтернативи. Отримана матриця подається на розгляд ЛПР для затвердження. Керуючись принципами парних порівнянь, виконується нормування значень із використанням дещо модифікованої формули природньої нормалізації. Відповідно до вимог ЛПР, також виконується обмеження області значень критеріїв оптимальності рівня маркеру представленості даних на вузлі РКІС.

З метою підвищення точності результату, запропоновано представити вектор глобальних пріоритетів альтернатив у вигляді вектору нечітких чисел для маркеру представленості даних. Для отримання числового значення оптимального рівня маркеру представленості даних виконується об’єднання результатів та дефазифікація.

Оскільки на рівень представленості при аналізі користувацьких запитів до БД впливає комбінація значень атрибутів кортежу відношення, запропоновано представити завдання визначення рівня необхідності представлення нових даних у вигляді задачі класифікації, що розв’язується із використанням Байєсівського наївного алгоритму.

Четвертий розділ присвячено створенню технології обліку та аналізу SQL-запитів, як складової інформаційної технології оптимізації структури БД вузла РКІС. Для можливості отримання набору відношень, атрибутів та кортежів SQL-

запиту, було реалізовано підсистему парсингу тексту SQL-запиту. Для цього розроблено граматику мови SQL, що дозволило із використанням програмного продукту ANTLR виділити множини відношень, атрибутів та вкладених підзапитів.

Завдання визначення множин кортежів запиту вирішено за рахунок виконання додаткового запиту для кожної таблиці із колекції відношень базового SQL-запиту, список атрибутів у якому замінюється на первинний ключ відношення. Результат запиту запам’ятовується як множина кортежів таблиці, що використовується поточним запитом.

Враховуючи великі обсяги статистичних даних, які фіксують виконання операцій користувачів із БД РКІС, а також необхідність виконання подальшого аналізу накопичених даних з точки зору множинності вимірів, дані SQL-запитів представлено у вигляді багатовимірної бази даних (ББД). Для реалізації таблиці фактів та таблиць вимірів у вигляді схеми «зірка», на рівні реляційної БД ІС обліку користувацьких запитів уведено ряд представлень. Даний підхід підвищує прозорість структури БД та є своєрідним інтерфейсом при взаємодії багатовимірної моделі із реляційною.

ББД реалізовано на базі SQL Analisys Server із використанням пакету SQL Server Development Tools for Business Intelligence. Структура всіх вимірів багатовимірного кубу є типовою. Вона містять один описовий атрибут «назва» та унікальний ідентифікатор об’єкту, що є первинним ключем для відношення. На базі створених вимірів та двох таблиць фактів, окремо для атрибутів та кортежів відношення, створюється два OLAP-куба, що мають в своїй основі схему «зірка» – по одній таблиці фактів та по одній таблиці на кожний вимір ББД.

У якості клієнтського застосунку в межах поточного дослідження використано MS Excel, що має механізми з підключення та отриманню даних із SQL Analysis Services та надає достатній рівень функціональних можливостей для побудови зрізів даних, необхідних ЛПР для прийняття рішення щодо подальшої фільтрації даних при розрахунку значень критеріїв оптимальності структури вузла РКІС.

У п’ятому розділі представлено програмну складову інформаційної технології визначення оптимального рівня маркеру представленості даних на вузлі РКІС, що виконана у вигляді сервіс-орієнтованого вебзастосунку на основі трьохланкової архітектури. Бекенд та фронтенд-компоненти взаємодіють через REST-API із обміном даними у json-форматі. Наведена архітектура є досить гнучкою, та дозволяє за необхідності змінювати СКБД та підключатись до бекенд Application Programming Interface (API), використовуючи стороннє програмне забезпечення.

Фронтенд складова реалізована у вигляді набору компонент за сервісів. Користувач інформаційної технології має змогу переглянути всі етапи формування глобального вектору пріоритетів альтернатив. Це дає змогу оцінити вплив кожної складеної матриці попарних порівнянь на всіх рівнях ієрархії моделі та, відповідно, приймати більш осмислене рішення щодо визначення оптимального рівня маркеру представленості даних на вузлі РКІС.

Етап обробки отриманого результату полягає у представленні отриманого вектору глобальних пріоритетів альтернатив у вигляді набору нечітких множин однієї змінної із подальшим об’єднанням та дефазифікацію результату для отримання числового значення оптимального рівня маркеру представленості даних. Даний етап виконано у середовищі MathLab із використанням модуля Fuzzy Logic Toolbox.

При розрахунку значення ефективності використання вузла РКІС, було виведено співвідношення отриманого результату до витрачених ресурсів на базі сформульованих у розділі 2 критеріїв оптимальності структури БД, та отриманого у ході вирішення задачі вибору кращої альтернативи вектору відносної ваги критеріїв оптимальності. Отримані результати дозволяють зробити висновок про підвищення ефективності у межах від 16 % до 30 % у порівнянні з розміщенням на вузлі РКІС всіх необхідних даних або лише критичних даних відповідно.

У висновках сформульовано основні наукові та практичні результати дисертаційної роботи.

У додатках наведено акти впровадження результатів дисертаційної роботи,

список публікацій за темою дисертації, лістинг скрипту створення структури БД обліку користувацьких запитів та граматики для опису мови T-SQL.

Ключові слова: корпоративна інформаційна система, реляційна база даних, SQL-запит, парсинг тексту, метод аналізу ієрархій, вебсервіс.

СПИСОК ПУБЛІКАЦІЙ ЗДОБУВАЧА ЗА ТЕМОЮ ДИСЕРТАЦІЇ

1. Фісун М. Т., Дворецький М. Л. Пошук асоціативних правил засобами мови SQL при інтелектуальному аналізі даних. *Вісник Херсонського національного технічного університету*. 2005. № 1 (21). С. 185–189; **внесок автора:** реалізовано пошуку асоціативних правил засобами мови SQL та аналіз отриманих результатів; **база (и):** *Google Scholar, CrossRef*.

2. Фісун М. Т., Дворецький М. Л. Застосування OLAP-технологій в інформаційно-управляючих системах. *Інформаційні технології та комп'ютерна інженерія: Міжнародний науково-технічний журнал Вінницького національно-технічного університету.* 2006. №2(6). С. 128–134; **внесок автора:** засобами SQL Server на базі OLAP розроблено інформаційно аналітичну систему для аналізу трендів продажів торгівельного підприємства; **база (и):** *Google Scholar*.

3. Фісун М. Т., Дворецький М. Л. Синхронізація оновлення даних в гетерогенних інформаційних системах. *Наукові праці: Науково-методичний журнал Миколаївського державного гуманітарного університету ім. П. Могили. Сер. Комп'ютерні технології.* 2006. Вип. 44. Т. 57. С. 61–66; **внесок автора:** запропоновано та виконано апробацію алгоритму розрахунку моменту наступної синхронізації БД різнорідних ІС; **база (и):** *Google Scholar*.

4. Дворецький М. Л. Реалізація задачі класифікації засобами мови Transact-SQL при інтелектуальному аналізі даних. *Системні технології: Регіональний міжвузівський збірник наукових праць. Дніпропетровськ, ДНВП «Системні технології»*. 2006. Випуск №6 (47). С. 102–111; **база (и):** *Google Scholar, Ulrichsweb, Index Copernicus*.

5. Дворецький М. Л. Створення перехресного набору даних (crosstab) засобами Transact SQL. *Вісник Херсонського національного технічного університету*. 2006. № 1 (24). С. 503–507; **база (и):** *Google Scholar, CrossRef*.

6. Дворецький М. Л. Розв’язання задачі класифікації в багатовимірних базах даних. *Вісник Херсонського національного технічного університету*. 2006. № 1 (30). С. 198–202; **база (и):** *Google Scholar, CrossRef*.

7. Дворецький М. Л. Інтелектуальний аналіз даних в 1С:8.0. *Наукові праці: Науково-методичний журнал Миколаївського державного гуманітарного університету ім. П. Могили. Сер. Комп’ютерні технології.* 2007. Вип. 55. Т. 68. С. 141–149; **база (и):** *Google Scholar*.

8. Дворецький М. Л. Проектування та оцінка оптимальності структури сховища даних та багатовимірної БД. *Наукові праці: Науково-методичний журнал Миколаївського державного гуманітарного університету ім. П. Могили. Сер. Комп’ютерні технології.* 2008. Вип. 77. Т. 90. С. 216–221; **база (и):** *Google Scholar*.

9. Дворецький М. Л. Інтеграція підсистем обліку та оперативно-аналітичної обробки даних в інформаційній системі супермаркету. *Комп’ютерні науки: освіта, наука, практика. Миколаїв. Національний університет кораблебудування*. 2014. С. 58–60; **база (и):** *Google Scholar, CrossRef*.

10. Дворецький М. Л. Використання SQL/OLAP в T-SQL при автоматизації операцій консолідації та деталізації кросс-таблиць на базі реляційних джерел даних. *Могилянські читання: досвід та тенденції розвитку суспільства в Україні: глобальний, національний та регіональний аспекти*: матеріали всеукр. наук.-метод. конф. Миколаїв: Вид-во ЧНУ ім. П. Могили, 2014. С. 44–46.

11. Дворецький М. Л., Динамічне формування запиту із використанням SQL/OLAP в T-SQL при автоматизації кросс-таблиць на базі реляційних джерел даних. *Наукові праці: Науково-методичний журнал Чорноморського державного університету ім. П. Могили. Сер. Комп’ютерні технології.* 2014. Вип. 238. Т. 250. С. 32–37; **база (и):** *Google Scholar, Ulrichsweb, Index Copernicus*.

12. Дворецький М. Л. Задача синхронізації даних в РБД. Синхронне та асинхронне оновлення. *Могилянські читання: досвід та тенденції розвитку суспільства в Україні: глобальний, національний та регіональний аспекти*: матеріали всеукр. наук.-метод. конф., 14-18 лист. 2016 р. Миколаїв: Вид-во ЧНУ ім. П. Могили, 2016. С. 105–108.

13. Дворецький М. Л., Кулаковська І. В. Порівняльний ABC-XYZ аналіз на базі різних факторів із використанням ієрархічних даних. *Проблеми інформаційних технологій Херсонського національного технічного університету.*

2016. № 19. С. 200–209; **внесок автора:** запропоноване ієрархічне представлення даних та реалізовані механізми подальшого поєднання результатів аналізу за різними критеріями; **база (и):** *Index Copernicus, Google Scholar, Research Bible, Open Academic Journals Index, Directory of Open Access Journals.*.

14. Фісун М. Т., Дворецький М. Л., Юхатов А. В. Порівняльний аналіз методів побудови OLAP-систем із використанням засобів MS SQL SERVER та ORACLE. *Наукові праці: Науково-методичний журнал Чорноморського державного університету ім. П. Могили. Сер. Комп'ютерні технології.* 2016. Вип. 271. Т. 283. С. 36–42; **внесок автора:** виконано аналіз засобів та показників продуктивності щодо автоматизації OLAP на базі СКБД Oracle та SQL Server; **база (и):** *Google Scholar, Ulrichsweb, Index Copernicus*.

15. Дворецький М. Л., Давиденко Є. О., Боровльова С. Ю. Проектування структури розподіленої БД на базі парсингу SQL-запитів. *Наукові праці: Науково-методичний журнал Чорноморського державного університету ім. П. Могили. Сер. Комп'ютерні технології.* 2016. Вип. 275. Т. 287. С. 53–61; **внесок автора:** виконано розробку парсеру та багатовимірної моделі SQL-запиту, розроблено СППР оптимізації структури РБД; **база (и):** *Google Scholar, Ulrichsweb, Index Copernicus*.

16. Дворецький М. Л., Дворецька С. В. Оптимізація механізмів пошуку та оцінка якості знань на базі статистичних даних користувацьких запитів. *Інформаційні технології та взаємодії* : матеріали міжнар. наук.-практ. конф., 8–10 лист. 2017 р. Київ, 2017. С. 157–158; **внесок автора:** реалізація підсистем накопичення та аналізу статистики користувацьких запитів.

17. Fisun M., Dvoretskyi M., Shved A. Davydenko Y. Query parsing in order to optimize distributed DB structure. *9th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS), Bucharest*. 2017, pp. 172–178. doi: 10.1109/IDAACS.2017.8095071; **внесок автора:** розроблено технологію парсингу тексту запиту для оптимізації структури БД; **база (и):** *Google Scholar, Web of Science, SCOPUS, IEEE, DBLP.*.

18. Fisun M., Dvoretskyi M., Horban H. The usage of the feedback with user activities in company knowlagde management system. *12th International Scientific and Technical Conference on Computer Sciences and Information Technologies (CSIT), Lviv*. 2017. pp. 143–146. doi: 10.1109/STC-CSIT.2017.8098755; **внесок автора:** використання зворотнього зв'язку користувацької активності в системах

управління знаннями та реалізація відповідної технології; **база (и):** *Google Scholar, SCOPUS, IEEE*.

19. Дворецький М. Л. Підходи щодо підвищення швидкості роботи SQL-запитів при використанні індексів БД. *Ольвійський форум: стратегії країн Причорноморського регіону в геополітичному просторі*: матеріали XII міжнар. наук.-практ. конф., 7-10 червня 2018 р. Миколаїв : Вид-во ЧНУ ім. Петра Могили, 2018. С. 30–32.

20. Дворецький М. Л., Дворецька С. В., Давиденко Є. О. Розробка системи управління знаннями організації на базі CMS WORDPRESS. *Проблеми інформаційних технологій Херсонського національного технічного університету.* 2018. №1 (023). С. 173–180; **внесок автора:** виконано розробку системи управління знаннями на базі CMS Wordpress; **база (и):** *Index Copernicus, Google Scholar, Research Bible, Open Academic Journals Index, Directory of Open Access Journals.*

21. Fisun M., Horban H., Dvoretskyi M. Methods of Searching for Association Dependencies in Multidimensional Databases. *13th International Scientific and Technical Conference on Computer Sciences and Information Technologies (CSIT), Lviv*. 2018, pp. 88–93. doi: 10.1109/STC-CSIT.2018.8526737; **внесок автора:** виконано аналіз результатів пошуку асоціацій на базі багатовимірної бази даних; **база (и):** *Google Scholar, SCOPUS, IEEE*.

22. Дворецький М. Л., Дворецька С. В., Давиденко Є. О. Використання angular та symfony при реалізації web-орієнтованого застосунку автоматизації обліку торгової точки. *Наукові праці: Науково-методичний журнал Чорноморського національного університету ім. П. Могили. Сер. Комп'ютерні технології.* 2018. Вип. 305. Т. 317. С. 70–77; **внесок автора:** реалізовано взаємодію бекенд та фронтенд фреймворків при розробці сервіс-орієнтованого вебзастосуноку; **база (и):** *Google Scholar, Ulrichsweb, Index Copernicus*.

23. Fisun M., Dvoretskyi M., Horban H., Komar M. Knowledge management applications based on user activities feedback. *International Journal of Computing, Ternopil*. 2019. 18 (1). pp. 32–44; **внесок автора:** дослідження впливу зворотнього зв'язку активності користувачів систем управління знаннями; **база (и):** *Google Scholar, SCOPUS*.

24. Дворецький М. Л., Боровльова С. Ю., Дворецька С. В. WEB-застосунок складського обліку в неавтоматизованих торгових точках. *Наукові праці:*

Науково-методичний журнал Чорноморського національного університету ім. П. Могили. Сер. Комп'ютерні технології. 2018. Вип. 308 Т. 320. С. 45–52; **внесок автора:** виконано аналіз різних типів автоматизації облікових систем та реалізовано відповідний web-застосунок; **база (и):** *Google Scholar, Ulrichsweb, Index Copernicus*.

25. Дворецький М. Л., Дворецька С. В. Використання brain.js при визначенні корисності кортежу для віддалено вузла РБД. *Могилянські читання: досвід та тенденції розвитку суспільства в Україні: глобальний, національний та регіональний аспекти:* матеріали XXII Всеукр. наук.-метод. конф., 11-16 лист. 2019 р. Миколаїв: Вид-во ЧНУ ім. П. Могили, 2019. С. 123–125; **внесок автора:** реалізація підсистеми визначення корисності кортежу на вузлі РКІС із використанням нейронних мереж.

26. Дворецький М. Л., Дворецька С. В., Давиденко Є. О. Інформаційна технологія визначення корисних даних при оптимізації структури та мінімізації обсягів вузла розподіленої БД. *Вісник Черкаського державного технологічного університету*. 2019. № 4. С. 26–35. doi: 10.24025/2306-4412.4.2019.184808; **внесок автора:** реалізація інформаційної технології оптимізації структури БД вузла РКІС; **база (и):** *Google Scholar, Crossref, Index Copernicus, Eurazian Scientific Journal Index, Directory of Open Access Journals.*

27. Dvoretskyi M., Dvoretska S., Nezdoliy Y., Borovlova S. Data Utility Assessment while Optimizing the Structure and Minimizing the Volume of a Distributed Database Node, *1st International Workshop on Information-Communication Technologies & Embedded Systems (ICTES), Mykolaiv, Ukraine, November 14-15.* 2019. pp. 128–137; **внесок автора:** вирішення задачі багатокритеріальної оптимізації при проєктуванні структури БД вузла РКІС; **база (и):** *SCOPUS.*

28. Фісун М. Т., Дворецький М. Л., Дворецька С. В. Побудова моделей для оптимізації структури бази даних вузла у корпоративних інформаційних системах. *Інформаційні технології та комп'ютерна інженерія: Міжнародний науково-технічний журнал Вінницького національно-технічного університету*. Vol 48, № 2. 2020. С. 52–60. doi: https://doi.org/10.31649/1999-9941-2020-48-2-52-60; **внесок автора:** реалізація моделей SQL-запиту та критеріїв оптимальності структури БД вузла РКІС; **база (и):** *Google Scholar*.

ABSTRACT

Dvoretskyi M. L. Models and information technology for optimizing the structure of the node database in corporate information systems. – Qualification research paper, manuscript copyright.

Thesis for the degree of a candidate of technical sciences in specialty 05.13.06 – information technology. – Petro Mohyla Black Sea National University, Mykolaiv, 2020.

The thesis is dedicated to increasing the level of data availability and efficiency of distributed computer systems by implementing the information technology to optimize the database (DB) structure of the distributed corporate information system (DCIS) based on SQL query statistics. This is reached through the development of appropriate models and information technology. The developed models describe the SQL query, formalize the criteria for the optimality of the database structure of the DCIS node and describe their dependence on the value of the data representation marker. Information technology is based on an improved model of SQL query and uses a modified method of analytic hierarchy process.

The object of research is the process of optimizing the database structure of distributed corporate information systems, which is based on relational databases. The subject of research – models of SQL queries of relational databases and methods of choosing the best alternative of the structure of the DCIS node database. The general scientific result of the work is the solution for the actual scientific and practical task of creating models and information technology for optimizing the structure of the node database in corporate information systems.

The first section considers approaches and methods to increase the speed of information systems (IS). The general disadvantage of the analyzed approaches is the need to modify the IS itself, which commonly is impossible. Considering the using "universal" IS, there is a tendency to use a set of separate specialized solutions with the next data synchronization. The holding structure of the company may also necessitate the creation and use of the database with a distributed structure. Under the corporate

information system, we understand a set of IS of the individual departments, united by the shared document flow, and all systems together ensure the functioning of the enterprise.

Various concepts of data representation in distributed databases are considered, among which the combined one stands out. It divides the database into logical fragments and allows having many numbers of physical copies of each fragment. The key factor that affects the reliability and availability of the database is the localization of links. Given the code of SQL queries of DCIS is closed, the task of optimal design of the database structure is to find the optimal distribution of data on the nodes of DCIS. Simulation models of DCIS, which is an extremely complex system, are usually very simplified and do not take into account several important factors that influence its behavior.

It is proposed to accumulate the statistics of the execution of SQL queries to a distributed database, which allows to take into account all the features of DCIS. Parsing SQL query texts will determine the set relations, attributes, and tuples of the database involved in each query. Further analysis will reveal the frequency and nature of appeals to individual relations, their attributes and tuples. A set of criteria for the optimality of the database structure is introduced and the problem of multicriteria optimization is formulated. Among various methods, the analytic hierarchy process stands out.

The conclusion is made about the relevance of research aimed at increasing the level of data availability and efficiency of distributed computer systems by implementing information technology to optimize the structure of the DCIS database based on SQL query statistics. To optimize the structure of the database, several tasks are formulated: accumulation of statistical data; development of information technology for parsing the text of SQL queries; development of relational and multidimensional databases for storage and analysis of SQL query data; creation of mathematical models of optimality criteria of the DB structure of DCIS node and formalize its dependence from a boundary level of data representation marker; solving the task of choosing the best alternative; classification of new records of database relations.

In the second section, based on the relational data model and relational algebra operations, the SQL query model is implemented, which includes data about the application, host, user, and a set of relations, attributes and tuples of the database, which are part of the resulting set of rows (tuples). The query is represented in the multidimensional database in the dimension base, which includes the "relation", "attribute" and "tuple" of the database.

When performing the analysis for data presenting on the node, the term "data representation marker" for the dimension elements was proposed, which reflects the level of need for data presentation at the DCIS node. For each element of the marker value, one of the linguistic values {"necessary", "neutral", "not required"} is taken, which determines the degree of need to present data of the particular workplace, user role or application.

Determining the value of the marker when performing the consolidation of rows of the fact table on the values of <R, A, tup> is made using the moving average method with the introduction of the dimension elements weights. By converting a non-numeric linguistic variable of markers into a numerical value ("obligatorily" – "2", "necessary" – "1", "neutral" – "0", "not required" – "–1", "forbidden" – "–2"), defined the function of aggregation of the data representation marker. When deciding on the data representation on a remote node, the consolidation of the rows of the fact table for <R, A, tup> is made by calculating the value of the marker for each of its elements.

To solve the task of formalizing the optimality criteria of the data representation marker, the following optimality criteria are defined: "database independence", "database size" and "synchronization necessity". For each of the criteria, a mathematical model is defined that describes its dependence on the value of the threshold level of the data representation marker at the DCIS node.

When synchronizing the data of the DCIS node, the dynamic determination of the data exchange period is proposed. The presence of mechanisms for calculating the moment of the next data synchronization allows to increase the relevance of data and unload the resources of database servers. Their implementation takes into account the following factors: the number of changes in the transaction log; the quality of these

changes (based on the intensity of data usage in SQL queries, the coefficients of influence on the DCIS node are determined); the value of the point of the data relevance; database server load with current SQL queries activity.

In the third section, the method for selecting the best alternative for the level of the data representation marker was developed. The analytic hierarchy process with automatic initialization of the matrix of pairwise comparisons of alternatives according to the obtained mathematical models and normalization of values was used; using elements of fuzzy inference to dephase the vector of global priorities, and performing the classification of new data using the naive Bayesian algorithm.

When building a hierarchical model of the decision-making process for choosing the best alternative, 4 levels of hierarchy were used: "goal", "decision makers", "criteria", "alternatives". In order to simplify the task, 5 alternatives to the level of the representation marker were identified: "low" – "–1", "lower then medium" – "–0.5", "medium" – "0", "higher then medium" – "0.5", and "high" – "1". The list of criteria for model optimality at the level of the "criteria" hierarchy differs for different decision makers. Using the scale of the relative importance of the criteria, the construction of the matrix of pairwise comparisons for the level of decision makers was made. At the third level of the hierarchy, the corresponding matrices of pairwise comparisons are formed according to the criteria of optimality for each decision maker.

The presence of mathematical models of optimality criteria formulated in the second section of the work allows performing the calculation and initialization of matrices of pairwise comparisons of alternatives based on numerical values of the data representation marker for each alternative. The obtained matrix is submitted to the decision maker for approval. Guided by the principles of pairwise comparisons, the normalization of values is performed using a slightly modified formula of natural normalization. In accordance with the requirements of the decision maker, the restriction of the values range of criteria of optimality is also performed.

To increase the accuracy of the result, it is proposed to represent the vector of global priorities of alternatives in the form of a vector of fuzzy numbers for the data

representation marker. To obtain the numerical value of the optimal level of the data representation marker, the results are aggregated and dephasified.

Since the level of representation in the analysis of user queries to the database is influenced by a combination of values of the attributes of the relation tuple, it is proposed to present the task of determining the level of need for new data as a classification problem and use of naive Bayesian algorithm to solve it.

The fourth section is devoted to the creation of information technology for the accounting and analysis of SQL queries. To be able to obtain a set of relations, attributes and tuples of SQL query, the subsystem of parsing the text of SQL query was implemented. To do this, the grammar of the SQL language was developed, which allowed using the software product ANTLR to select sets of relations, attributes and subqueries.

The task of determining the sets of query tuples is solved by executing an additional query for each table from the collection of relations of the basic SQL query, the list of attributes in which is replaced by the primary key of the relation. The result of the query is stored as the set of table tuples used by the current query.

Given the large number of statistics that record the performance of user activities with the DCIS database, as well as the need for further analysis of the accumulated data in terms of multi-dimensional model, SQL query data is presented as a multidimensional database (MDB). To implement the fact table and dimension tables in the form of a scheme "star", at the level of the relational database the set of views were created. This approach increases the transparency of the database structure and is a kind of interface for the interaction of the multidimensional model with the relational one.

The multidimensional database is implemented on the basis of SQL Analysis Server using the SQL Server Development Tools for Business Intelligence package. The structure of all dimensions of a multidimensional cube is typical. It contains one descriptive attribute "name" and a unique object identifier, which is the primary key to the relation. Based on the created dimensions and two fact tables, separately for the

attributes and tuples of the relation, two OLAP-cubes are created, based on the "star" scheme - one fact table and one table for each dimension of the MDB.

As a client application the MS Excel was used, which has mechanisms for connecting and receiving data from SQL Analysis Services and provides a sufficient level of functionality to build data slices needed by decision maker to determine the further data filtering when calculating the values of structure optimality criteria of DCIS node.

The fifth section presents the software implementation of information technology to determine the optimal level of the data representation marker on the DCIS node, which is made in the form of a service-oriented web application based on a three-tier architecture. The backend and frontend components interact through the REST-API with data exchange in JSON format. This architecture is quite flexible, and allows to change the database platform, and connect to the backend Application Programming Interface (API), using third-party software.

The frontend is implemented as a set of components and services. It is available for the user to view all stages of the formation of the global vector of priorities for alternatives. This allows to assess the impact of each matrix of pairwise comparisons at all levels of the hierarchy model and, accordingly, to make a more meaningful decision to determine the optimal level of the data representation marker at the DCIS node. At the stage of processing the obtained result, the vector of global priorities is presented in the form of fuzzy sets of one variable, followed by combining and dephasifying the result to obtain a numerical value of the optimal level of the data representation marker. This step is performed in the MathLab environment using the Fuzzy Logic Toolbox module.

When calculating the value of the DCIS node efficiency, the ratio of the obtained result to the spent resources was derived. It was done on the basis of database structure optimality criteria, formulated in section 2, and calculated vector of the relative weight of optimality criteria. The obtained results allow to make a conclusion about efficiency increase in the range from 16 % to 30 % in comparison with placement on the DCIS node of all necessary data or only critical data accordingly.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

ББД – багатовимірна база даних

БД – база даних

БКО – багатокритеріальна оптимізація

ІУ – індекс узгодженості

ІС – інформаційна система

КІС – корпоративна інформаційна система

ЛПР – людина, що приймає рішення

МАІ – метод аналізу ієрархій

ОДД – оперативне джерело даних

ПЗ – програмне забезпечення

РБД – розподілена база даних

РКІС – розподілена корпоративна інформаційна система

СД – сховище даних

СКБД – система керування базами даних

СППР – система підтримки прийняття рішень

ACID – Acomicity, Consistancy, Isolation, Durability

DDB – Distributed Databases

ETL – Extracting, Transformation, Loading

OLAP – Online Analytical Processing (укр. оперативно-аналітична обробка даних)

OLTP – Online Transactional Processing (укр. системи оперативної обробки транзакцій)

SQL – Structured Query Language (укр. мова структурованих запитів)

ВСТУП

Обґрунтування вибору теми дослідження

Широке використання інформаційних технологій багаторівневих, територіально розосереджених та розподілених корпоративних інформаційних систем (РКІС) на основі баз і сховищ даних, у тому числі розподілених, обумовлює необхідність забезпечення певної автономності роботи користувачів, зниження навантаження на сервери баз даних (БД) та сервіси синхронізації даних шляхом оптимізації структури БД віддаленого вузла корпоративної інформаційної системи (КІС). Під розподіленою базою даних (РБД) розуміємо сукупність взаємопов'язаних баз даних, розподілених у мережі (у тому числі Інтернет). Основна функція розподілених систем керування базами даних (СКБД) – координування спільної роботи багатьох користувачів з розподіленою інформацією. Розв'язання задачі автономності роботи користувачів розподіленої системи створює багато специфічних проблем в організації баз даних, оскільки різні користувачі можуть працювати паралельно з одними й тими самими даними, виконуючи з ними різні перетворення.

Ключовим фактором, що впливає на надійність і доступність БД, є локалізація посилань. Про високий ступінь локалізації посилань свідчить представлення на поточному вузлі тих даних, що викликаються винятково його користувачем. Комбінована стратегія розподілу даних є найбільш привабливою, але при її використанні, окрім задачі синхронізації дубльованої інформації, актуальною постає задача оптимального проєктування структури БД з точки зору приналежності даних до вузла РКІС.

Комбінована стратегія розподілу даних є найбільш виправданою із точки зору можливості поєднання переваг стратегій з та без дублювання. Але при її використанні, окрім задачі синхронізації дубльованої інформації, актуальною постає задача оптимального проєктування структури БД з точки зору приналежності даних до категорії того чи іншого вузла мережі. Крім того,

продуктивність системи напряму буде залежати від прийняття рішення щодо необхідності часткового або повного дублювання даних.

Відношення БД представляються на вузлі РКІС після застосування операцій проєкції та вибірки. Тобто для оптимального представлення даних необхідно використати елементи вертикальної та горизонтальної фрагментації даних. Імітаційні моделі РКІС зазвичай дуже спрощені та не враховують ряд важливих факторів, що впливають на її поведінку. У свою чергу накопичення статистичних даних виконання SQL-запитів до розподіленої БД на реально працюючих серверах дає змогу врахувати всі особливості РКІС.

Наведені питання розглядались у роботах таких вітчизняних вчених, як: Стогній А. О., Пасічник В. В., Шаховська Н. Б., Кунгурцев О. Б., Філатов В. О., Малахов Є. В., Скобцов Ю. А., Зіноватна С. Л., Новохатська С. Ю. та ін., а також розглянуті такими зарубіжними вченими, як Дейт К. Дж., Кодд Е. Ф., Гарсіа-Моліна Г., Барсегян А. А., Ульман Дж. Д., Райордан Р. М., Бергер А. Б., Конноллі Т., Петкович Д. та ін. Однак, не дивлячись на певний прогрес в методах та інформаційних технологіях проєктування структур РБД, питання оптимізації структури БД вузлів РКІС шляхом нових підходів до моделювання та статистики SQL-запитів обумовлює актуальність теми даного дослідження.

Так, у роботах Стогнія А. О., Дейта К. Дж., Кодда Е. Ф. висвітлюються питання проєктування автоматизованих систем управління та обробки даних, у роботах Пасічника В. В., Барсегяна А. А. – питання організації сховищ даних та багатовимірних моделей. Невисвітленими залишаються питання побудови моделей при використання комбінованої стратегії розподіленого представлення даних у КІС. Вчені Кунгурцев О. Б., Філатов В. О., Зіноватна С. Л., Новохатська С. Ю. у своїх роботах розглядають питання підвищення продуктивності автоматизованих систем за рахунок використання матеріалізованих представлень, реструктуризації БД та денормалізації відношень. Однак, аспекту оптимальності структури окремого вузла РКІС у проаналізованих дослідженнях приділяється мало уваги. У роботах Шаховської Н. Б. розглянуто підходи до аналізу тексту, а також використання

консолідації та просторів даних при роботі із багатовимірними моделями. Проте, використання багатовимірних моделей при представленні SQL-запитів до реляційної БД не розглядається.

Отже, задача розробки моделей та інформаційної технології оптимізації структури бази даних вузла у корпоративних інформаційних системах є актуальною.

Зв'язок роботи з науковими програмами, планами, темами

Дисертація виконана у відповідності до завдань науково-дослідницьких робіт Чорноморського національного університету (надалі – ЧНУ) імені Петра Могили, в яких автор брав участь як виконавець:

- «Інтеграція інформаційних технологій баз даних, баз знань та Data Mining» (номер держ. реєстрації № 0112U007749);

- «Інтелектуалізація інформаційних систем та СППР за рахунок впровадження методів ситуаційного моделювання, сценарного аналізу, Data Mining та OLAP технологій» (номер держ. реєстрації № 0114U004402);

- «Розроблення найсучаснішого інтерактивного навчально-тренажерного та аналітично-консультативного комплексу військово-цивільного призначення» номер держ. реєстрації № 0118U000193).

Мета і завдання дослідження

Метою дослідження є підвищення рівня доступності даних та ефективності використання розподілених та територіально розосереджених комп’ютерних систем шляхом реалізації інформаційної технології оптимізації структури БД вузла РКІС на основі статистики SQL-запитів.

Для досягнення поставленої мети в роботі сформульовано такі завдання:

- проаналізувати особливості задач використання інтегрованого комплексу територіально розосереджених інформаційних систем та методів їх оптимізації;

- побудувати модель SQL-запиту та формалізувати критерії оптимальності структури БД віддаленого вузла КІС;

- розробити інформаційну технологію парсингу та подальшого обліку SQL-запитів до реляційної БД, а також обчислення значень критеріїв оптимальності відповідно до результатів операцій зрізу та консолідації;

- сформулювати та розв'язати задачу багатокритеріальної оптимізації для визначення оптимального рівня представленості даних на вузлі РКІС;

- розробити інформаційну технологію обчислення вектору глобальних пріоритетів альтернатив із представленням результату у вигляді вектору нечітких чисел та приведенням до чіткого значення;

- виконати класифікацію нових даних БД відповідно до необхідності їх представлення на вузлі РКІС.

Об'єктом дослідження є процес оптимізації структури БД розподілених корпоративних інформаційних систем, в основі яких лежать реляційні БД.

Предмет дослідження – моделі SQL-запитів реляційних БД та методи вибору кращої альтернативи структури вузла РКІС.

Наукова гіпотеза – існує ряд критеріїв оптимальності структури вузла РКІС, значення яких можна визначити у вигляді залежності від представлення даних інформаційної системи локально або віддалено. Вирішення багатокритеріальної задачі та знаходження оптимального рівня представленості даних дозволить підвищити ступінь доступності даних та ефективності використання вузла РКІС.

Методи дослідження

Методи досліджень базуються на основних засадах теорії реляційних баз даних (у тому числі розподілених), методах парсингу тексту, багатокритеріальної оптимізації та шаблонного проєктування. Зокрема, на:

а) методах представлення даних та обробки транзакцій у розподілених БД – при аналізі особливостей використання РКІС;

б) методах реляційної алгебри, теорії реляційної моделі даних, багатовимірній моделі даних – при побудові моделі SQL-запиту;

в) методі аналізу ієрархій та апараті нечіткого логічного висновку – при вирішенні задачі вибору найкращої альтернативи рівня представленості даних;

г) наївному методі Байєса – при класифікації нових даних що надходять на вузол РКІС після останньої реструктуризації;

д) методах парсингу тексту на основі формальних граматик – при розв'язанні задачі визначення складових SQL-запиту;

е) методах шаблонного проєктування та розробки вебзастосунків при реалізації сервіс-орієнтованої інтернет-технології підтримки прийняття рішень при проєктуванні структури віддаленого вузла РКІС.

Наукова новизна отриманих результатів

1. ВПЕРШЕ уведено поняття маркеру представленості даних на вузлі РКІС для елементів вимірів моделі SQL-запиту та розроблена функція агрегації, що дозволяє визначити рівень необхідності атрибутів та кортежів відношення БД на вузлі РКІС на основі статистики SQL-запитів.

2. ВПЕРШЕ побудовано модель залежності критеріїв оптимальності структури БД вузла РКІС від значення маркера представленості даних, що на відміну від існуючих підходів дозволяє визначити оптимальне граничне значення маркера на основі статистики SQL-запитів.

3. ВПЕРШЕ розроблено інформаційну технологію розрахунку критеріїв оптимальності структури БД вузла РКІС, що дозволяє визначити граничний рівень маркеру представленості даних і прийняти рішення про представленість даних на вузлі РКІС та, на відміну від існуючих, використовує аналітичні дані та результати парсингу тексту SQL-запитів до серверів РКІС.

4. УДОСКОНАЛЕНО модель SQL-запиту, яка, на відміну від існуючих, представлена моделлю багатовимірної БД в аналітиці множини атрибутів та кортежів відношення, а також інших характеристик запиту, що дозволяє провести оперативно-аналітичний аналіз, зважаючи на множинність вимірів з метою розрахунку рівня представленості атрибутів та кортежів відношення БД на вузлі РКІС.

5. ОТРИМАВ ПОДАЛЬШОГО РОЗВИТКУ метод аналізу ієрархій за рахунок автоматичної ініціалізація матриці попарних порівнянь альтернатив згідно отриманих математичних моделей та нормалізації значень та

представлення результату у вигляді вектору нечітких чисел із приведенням до чіткого значення, що дозволило підвищити ефективність вузла РКІС на 4 %.

6. ОТРИМАВ ПОДАЛЬШОГО РОЗВИТКУ метод асинхронної реплікації даних за рахунок динамічного визначення моменту наступної синхронізації даних на основі статистики SQL-запитів, що дозволяє підвищити актуальність даних та розвантажити ресурси серверів баз даних.

Практичне значення отриманих результатів

Практична значимість одержаних результатів полягає у розробленні інформаційної технології підтримки прийняття рішення при визначенні представленості даних на вузлі РКІС, що дозволяє оптимізувати структуру БД вузла.

Результати дослідження впроваджено на ППФ «Юнікс Трейд Ко» при проєктуванні структури БД віддаленого вузла РКІС. Використання розробленої інформаційної технології оптимізації структури БД вузла РКІС дозволило виявити закономірності між значеннями критеріїв оптимальності структури БД та рівнем представленості даних інформаційної системи на базі статистики SQL-запитів. Ефект від впровадження зазначених результатів полягає у підвищенні ефективності використання БД вузла РКІС на 12 %.

Розроблені моделі та інформаційна технологія впроваджені на «ТОВ Еліт білдінг» для виявлення залежності критеріїв оптимальності структури БД та рівня представленості даних. Ефект від впровадження полягає у підвищенні швидкості виконання запитів на 14 % та доступності даних на 21 %.

Результати роботи також впроваджено у навчальний процес на кафедрі «Інженерії програмного забезпечення» при викладанні дисциплін «Інформаційні технології OLTP, OLAP та DM на клієнт-серверній платформі», «Клієнт-серверні СКБД та аналітичні системи», «Database Development», а також у курсовому та дипломному проєктуванні для студентів спеціальності «Інженерія програмного забезпечення».

Результати роботи увійшли у звіт науково-дослідницької роботи «Розроблення найсучаснішого інтерактивного навчально-тренажерного та

аналітично-консультативного комплексу військово-цивільного призначення» ЧНУ ім. Петра Могили, Миколаїв, 2019 (держ. реєстр. № 0118U000193).

Особистий внесок здобувача

Наукові положення, теоретичні розробки та практичні результати у представленій дисертаційній роботі отримані автором особисто. У наукових роботах, опублікованих у співавторстві, здобувачу належить наступне. В [1] виконано практичну реалізацію задачі пошуку асоціативних правил засобами мови SQL та аналіз отриманих результатів. В [2] засобами SQL Server на базі OLAP розроблено інформаційно аналітичну систему для аналізу трендів продажів торгівельного підприємства. В рамках [3] запропоновано та виконано апробацію алгоритму розрахунку моменту наступної синхронізації БД різнорідних інформаційних систем (ІС). В [13] запропоноване ієрархічне представлення даних при виконанні ABC-XYZ аналізу та реалізовані механізми подальшого поєднання результатів аналізу за різними критеріями. У ході [14] здобувачем виконано аналіз наявності інструментальних засобів та деяких показників продуктивності щодо автоматизації OLAP на базі СКБД Oracle та SQL Server. В [15] виконано розробку парсеру та багатовимірної моделі SQL-запиту, на базі якої розроблено СППР при оптимізації структури РБД. В рамках [16] здобувачу належить реалізація підсистем накопичення та аналізу статистики користувацьких запитів. В [17] розроблено технологію парсингу тексту запиту для оптимізації структури БД. В роботі [18] та [23] автору належить дослідження використання зворотного зв’язку користувацької активності в системах управління знаннями та реалізація відповідної технології. В рамках [20] виконано розробку системи управління знаннями на базі CMS Wordpress. У роботі [21] проаналізовано результати пошуку асоціацій на базі багатовимірної бази даних. У [22] реалізовано взаємодію бекенд- та фронтенд-фреймворків при розробці сервіс-орієнтованого вебсервісу. У [24] автор виконав порівняльний аналіз різних типів автоматизації облікових інформаційних систем та реалізацію відповідного вебзастосунку. У [25] здобувачу належить реалізація підсистеми визначення корисності кортежу на вузлі РКІС із використанням нейронних мереж. У рамках [26], [27] та [28] автором

запропоновано реалізацію моделей та інформаційної технології оптимізації структури БД вузла РКІС.

Апробація результатів дисертації

Основні положення і наукові результати дисертаційної роботи доповідалися та отримали позитивну оцінку на 18 міжнародних, всеукраїнських та регіональних наукових конференціях, а саме: Всеукраїнська науково-методична конференція «Могилянські читання» (2014–2019); Міжнародна науково-технічна конференція «Комп'ютерні науки: освіта, наука, практика» (2014); Міжнародна науково-практична конференція «Молодь у світі сучасних технологій» (2016); Міжнародна науково-практична конференція «Інформаційні технології та взаємодії» (2017); Міжнародна науково-практична конференція Ольвійський форум (2016–2019); 9th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems (2017); 12th & 13th International Scientific and Technical Conference on Computer Sciences and Information Technologies (2017–2018); 1st International Workshop on Information-Communication Technologies & Embedded Systems (2019).

Публікації

За темою дисертації було опубліковано 28 наукових робіт, з них: 5 публікацій проіндексовано у наукометричній базі Scopus; 17 надруковано у фахових виданнях; 6 – у збірниках матеріалів міжнародних та всеукраїнських науково-технічних конференцій.

Структура дисертації

Дисертація складається зі вступу, п’яти розділів, висновків, списку використаних джерел із 163 найменувань та чотирьох додатків. Загальний обсяг дисертації – 213 сторінок, з них 4 додатки на 36 сторінках, 25 таблиць, 71 рисунок.

РОЗДІЛ 1
ХАРАКТЕРИСТИКА ТЕХНОЛОГІЙ, МОДЕЛЕЙ ТА МЕТОДІВ ПІДВИЩЕННЯ ЕФЕКТИВНОСТІ РОЗПОДІЛЕНИХ КОРПОРАТИВНИХ ІНФОРМАЦІЙНИХ СИСТЕМ

1.1 Технології підвищення швидкості роботи інформаційних систем

Інформаційні системи (ІС), що є взаємозалежною сукупністю засобів, методів і персоналу та мають на меті зберігання, обробку та представлення інформації, звичайно мають справу з великими обсягами інформації, що нерідко має досить складну структуру. Майже будь-яка ІС для збереження даних використовує бази даних (БД), що перебувають під управлінням системи керування базами даних.

Система керування базами даних (СКБД) — це комплекс програмних і мовних засобів, необхідних для створення баз даних, підтримки їх в актуальному стані й організації пошуку в них необхідної інформації. Розрізняють файлові та клієнт-серверні СКБД. Специфікою архітектури «клієнт-сервер» є використання спеціальної мови структурованих запитів, або Structured Query Language (SQL) – простого й потужного інструменту для доступу до даних. Запит клієнта на виконання якої-небудь операції з даними провокує на сервері пошук і добування даних. Витягнуті дані транспортуються по мережі від сервера до клієнта.

Один із основних методів підвищення ефективності роботи клієнт-серверних ІС полягає у зменшення сумарного часу, необхідного на обробку клієнтського запиту та повернення результату його виконання, більшу частину якого складає, як правило, його виконання на сервері БД [1–4].

1.1.1 Оптимізація запитів та індексація даних

Оптимізація запитів є одним із найбільш важливих і цікавих напрямків досліджень і розробок в області баз даних [5; 6]. Існує перелік стандартних підходів, що дозволяють зменшити час виконання SQL-запиту на сервері БД [7]. Так, у роботах [8–10] наведено рекомендації використання конкретних імен

атрибутів, зведення до мінімуму використання підзапитів, уникання використання оператору «in» із великими підмножинами, використання фільтрації даних на якомога ранніх етапах вибірки, використання правильного порядку поєднання таблиць та ін. Дослідження [11] наголошує на можливих перевагах використання тимчасових таблиць, оператора «join» перед вкладеним підзапитом та уникання конструкцій корельованих підзапитів, якщо це можливо. У роботі Філатова В. А. [12] розглянуті питання побудови оптимальної послідовності з'єднання відношень в запитах реляційної бази даних, а Кунгурцев А. Б. у [13] використовує ієрархічну модель об'єктів для дослідження запитів до БД. В окремих випадках, зменшення часу виконання запиту може складати від десятків до сотень, і навіть тисяч разів [5; 7–10]. Порівняльна характеристика показників складності виконання запитів в реляційних СКБД наводиться у роботі [14].

Всі сучасні СКБД мають у своєму складі спеціальні модулі та підсистеми, що мають спільну назву «оптимізатор запиту». Робота оптимізатора запиту є ключовою у обробці даних, а розуміння принципів його роботи допомагає при побудові швидких запитів [15]. На етапі вибірки даних СКБД намагається найбільш швидким та менш витратним чином використовувати свої ресурси. Цю роботу виконує оптимізатор запитів, і, як результат, видає «план виконання запиту». У більшості випадків оптимізатор запиту виконує перетворення тексту запиту відповідно до загальних рекомендацій підвищення продуктивності його роботи. Відповідно, навіть при реалізації неоптимальних запитів з боку розробників ІС, СКБД частково вирішує цю проблему. Крім того, оптимізація запитів стає можливою тільки при наявності доступу до модифікації їх тексту, що у більшості випадків у користувача ІС відсутній [3; 16–19].

Один із ефективних шляхів підвищення швидкості SQL-запитів полягає в упорядкуванні даних за рахунок створення індексів [20–21]. Дані можуть бути упорядковані двома шляхами: через зміну реального порядку збереження даних на дисковому носії, або через створення додаткової структури, де зберігається поле, за яким виконується сортування, та посилання на рядок таблиці. У другому

випадку виконується сортування додаткової структури, а не самих даних. Даним типам сортування в реляційних СКБД відповідають кластерний на некластерний типи індексів.

Кластерна індексація дає більш високий рівень швидкості пошуку необхідних даних за полем індексації. Головний недолік полягає у неможливості використання декількох кластерних індексів для одного відношення. Тому даний тип індексації може бути використаний лише у частині запитів на вибірку даних. При вибірці із використанням некластерної індексації виконується пошук у відсортованій структурі, із подальшим переходом за посиланням для читання основних даних. Недолік даного підходу полягає у необхідності виконання додаткового пошуку при переході за посиланням. Окремої уваги заслуговують покриваючі некластерні індекси [20; 22], що дозволяють представити у індексі додаткові поля, які не приймають участі у сортуванні, але необхідні запиту на вибірку даних. Слід також звернути увагу на спільний недолік всіх типів індексів, що полягає у сповільненні виконання операцій модифікації даних.

При виконанні запиту, оптимізатор запиту виходячи з існуючих індексів, структур даних і наявної статистики, намагається у відведений проміжок часу знайти кращий план виконання запиту [23]. Під час компіляції плану запиту, при переборі можливих індексів, якщо сервер СКБД не виявив кращого варіанту індексації даних, то в плані запиту даний індекс позначається, як не знайдений. Сервер СКБД веде статистику щодо таких індексів – скільки разів сервер скористався б цим індексом і якою була би очікувана швидкість виконання запиту [24].

Тобто можна стверджувати, що оптимізатор запитів разом зі статистикою СКБД надає достатній обсяг рекомендацій розробникам ІС та БД щодо створення та модифікації існуючих індексів, які можуть позитивно вплинути на швидкість виконання SQL-запитів. Практика використання ІС показує, що у переважній більшості випадків всі необхідні індекси, що можуть позитивно вплинути на роботу ІС, уже створені і ефективно використовуються [17; 25].

1.1.2 Модифікація структури БД

Іншим підходом оптимізації запитів та підвищення ефективності ІС є модифікація структури БД на базі контрольованої денормалізації. Усунення аномалій даних відповідно до теорії реляційних БД вимагає, щоб будь-яка БД була нормалізована, тобто відповідала вимогам нормальних форм. Відповідність вимогам нормалізації мінімізує надмірність даних в БД і забезпечує відсутність багатьох видів логічних помилок оновлення та вибірки даних [26–29]. Однак, при виконанні вибірки великих обсягів даних операція з'єднання нормалізованих відношень виконується неприйнятно довго. Внаслідок цього, в ситуаціях, коли продуктивність таких запитів неможливо підвищити іншими засобами, може проводитись контрольована денормалізація, що полягає у композиції кількох відношень (таблиць) в одну [30].

За рахунок такого перепроєктування операція сполучення при вибірці даних стає непотрібною, і, як результат, запити вибірки, які раніше вимагали з'єднання, працюють швидше. Слід пам'ятати, що денормалізація завжди виконується за рахунок підвищення ризику порушення цілісності даних при операціях модифікації. Тому денормалізацію слід проводити в крайньому випадку, якщо інші заходи підвищення продуктивності неможливі, або у випадках, коли денормалізована БД використовується тільки на читання. Крім того, слід враховувати, що прискорення одних запитів на денормалізованій БД може супроводжуватися уповільненням інших запитів, які раніше виконувалися окремо на нормалізованих відношеннях.

Порівняння вартості виконання запитів до та після денормалізації реляційної БД та підвищення продуктивності автоматизованої системи методами реструктуризації БД розглядається в роботах Зіноватної С. Л. [31; 32]. Також автори [33] наводять імітаційні моделі для дослідження ефективності денормалізації реляційної бази даних в інформаційних системах. Слід зауважити, що підходи, пов’язані із оптимізацією запитів шляхом модифікації структури БД мають недолік, спільний із підходом модифікації текстів SQL-запитів. Даний

недолік пов’язаний передусім із відсутністю доступу до модифікації коду ІС [3; 4; 11; 34]. Відповідно зміни структури БД ніяким чином не впливатимуть на швидкість роботи клієнтських SQL-запитів, що залишаються сталими.

1.1.3 Використання матеріалізованих представлень

Окремим випадком модифікації структури БД, та ще одним із шляхів підвищення швидкості виконання запитів є використання матеріалізованих представлень [35–36]. Матеріалізоване представлення – фізичний об'єкт БД, що містить результат виконання запиту. Так як в матеріалізованих представленнях зберігаються вже заздалегідь обчислені результати запиту, включаючи підсумки і результати з'єднань таблиць (JOIN), отримання даних з них виконується значно швидше, ніж у випадку звичайних представлень. В деяких СКБД, таких як Oracle, вже є штатний механізм для створення матеріалізованих представлень на рівні мови SQL (CREATE MATERIALIZED VIEW). В інших СКБД такий механізм може бути відсутній [37], але це не означає, що створити матеріалізоване представлення там неможливо.

Як випливає з визначення, щоб представлення працювало як матеріалізоване, воно повинно зберігати результати виконання запиту на фізичному рівні. У разі використання СКБД, де не передбачено матеріалізованих представлень явним чином, цього можна домогтися шляхом створення для представлення кластерного індексу. Тоді результати зберігаються фізично в індексі і час звернення до них зменшується. Технології створення матеріалізованих представлень для реляційних баз даних розглядаються в роботі [38]. Також питання підвищення ефективності застосування матеріалізованих представлень в автоматизованих комп'ютерних системах з реляційними базами даних розглядається у роботі [39].

Незважаючи на свої переваги, даний підхід має недолік, спільний із індексацією даних, що полягає у сповільненні виконання операцій модифікації даних. Також відсутність доступу до модифікації коду ІС [17; 40–41] обумовлює неможливість їх використання останніми для підвищення ефективності ІС.

1.2 Використання моделей СД, OLAP-технологій та ETL-систем

Ще один підхід підвищення ефективності ІС пов’язаний із поняттям сховища даних (СД) [42]. Системи, що вирішують задачі вводу та збереження даних та інформаційно-пошукові системи обробки даних, використовують для обробки даних транзакційний підхід. Розвинений механізм керування транзакціями в сучасних СКБД зробив їх основним засобом побудови OLTP-систем, основною задачею яких є забезпечення виконання операцій із БД.

Практика використання OLTP-систем показала неефективність їх застосування для повноцінного аналізу інформації. Основною причиною невдач є протиріччя у вимогах, що пред’являються до систем OLTP та систем підтримки прийняття рішень (СППР), серед яких різний ступінь деталізації даних, допущення надлишковості, кількість даних, що зберігаються, час обробки звернення до даних та ін. Наявність наведених суперечностей веде до появи концепції сховищ даних, основна ідея якої полягає у розділенні БД для OLTP систем і БД для виконання аналізу і подальшому їх проєктуванні з врахуванням відповідних вимог [43]. При реалізації в СППР концепції СД дані з різних оперативних джерел даних (ОДД) акумулюються в єдиному сховищі, що неминуче приводить до дублювання інформації в ОДД і в СД.

Процес та аспекти створення та використання сховищ даних висвітлюються у роботах Пасічника В. В. та Шахорської Н. Б. [44; 45]. Малахов Є. В наводить проблеми створення інформаційних сховищ, та опис і представлення об’єктів у сховищах даних в множині предметних областей [46]. Недоліками концепції СД є необхідність інтеграції даних неоднорідних джерел у розподіленому середовищі, потреба в ефективному зберіганні великих об’ємів інформації, необхідність багаторівневих довідників метаданих та підвищення вимог до безпеки даних. Використання віртуального СД вирішує частину з них, але веде до виникнення ряду інших, серед яких збільшення часу обробки запитів до СД, необхідність постійної доступності всіх ОДД, та неможливість отримання даних за довгий період часу.

Задача перенесення даних із ОДД до СД покладається на окремий клас систем, що отримали назву Extraction Transformation Loading (ETL) системи. Організація процесу переносу інформації з баз даних в інформаційні сховища детально розглянута у роботі [47]. Задача асинхронної реплікації даних, що є актуальною для розподілених СКБД, також може бути покладена на ETL-систему. Технології інтеграції гетерогенних інформаційних систем і розподілених баз даних розглянуті у роботі [48].

Процес перенесення, включає етапи витягання, перетворення і завантаження. Серед способів витягання даних виділяються: отримання даних допоміжними програмними засобами безпосередньо із структур зберігання інформації та вивантаження даних засобами OLTP-систем в проміжні структури.

Перший підхід характеризує відсутність необхідності розширення OLTP-системи, та витягання даних з врахуванням потреб процесу перенесення. При цьому структура БД може бути невідомою або відсутні механізми доступу до даних. Крім того, зміна структури БД ОДД веде до необхідності перепроєктування підсистеми витягання даних. Другий характеризує можливість використання засобів OLTP-систем, адаптованих до структур даних та зміни засобів вивантаження разом із змінами OLTP-систем і ОДД.

Етап перетворення включає узагальнення, приведення значень та очищення даних. Процедура очищення даних направлена на виявлення і видалення помилок і невідповідностей у даних з метою поліпшення їх якості. Після того, як дані перетворені, здійснюється етап їх завантаження.

1.3 Проблеми використання багаторівневих, розподілених та територіально розосереджених КІС

У межах одного підприємства часто існує необхідність автоматизації різних типів обліку. У якості прикладу можна навести складський, бухгалтерський облік, облік кадрів, розробка інформаційних порталів, систем відеоспостереження, контролю прав доступу до приміщень та ін. Спроба одночасної автоматизації різних видів обліку привело до виникнення так званих «універсальних» або

«комплексних» облікових систем, що створюють єдине облікове середовище організації, та забезпечують доступ до всіх необхідних даних моніторингу, контролю і оцінки ефективності роботи організації, а також для підтримки прийняття управлінських рішень.

Підхід, пов’язаний із використанням такого роду систем, має цілий ряд недоліків. По-перше, реалізація всіх видів обліку у межах однієї ІС веде до перевантаження БД такої системи. Враховуючи специфіку різних типів обліку, структура такої БД є досить складною, відповідно транзакції запитів на вибірку та зміну даних є тривалими, що приводить до виникнення взаємних блокувань користувачів при паралельній роботі. Іншим аспектом є низька відмовостійкість та вразливість такої системи, оскільки тимчасова недоступність центральної БД комплексної ІС приводить до зупинки роботи інформаційної систем всіх структурних підрозділів організації. Також слід зауважити, що жодна з «універсальних» облікових систем не реалізовує у повному обсязі специфіку всіх видів діяльності компанії.

1.3.1 Централізована та розподілена моделі КІС

Наведені недоліки можуть бути мінімізовані за допомогою використання окремих спеціалізованих ІС. Перевагами спеціалізованих облікових систем є більш високий рівень якості реалізації необхідних облікових механізмів та локалізація використання даних кожної ІС у вигляді окремої БД. Тимчасова недоступність БД або серверу застосунків однієї з ІС не впливає на роботу інших. Але даний підхід веде до появи різних платформ СКБД, що потребують подальшої синхронізації.

Холдингова структура об’єкта автоматизації та/або географічна віддаленість філій підприємства також може обумовити необхідність створення та використання БД розподіленої структури. У деяких випадках можливе використання центральної БД із роботою через виділений канал зв’язку або розміщення БД у хмарі. Але при цьому збільшується навантаження на центральну БД, канали зв’язку, та знижується відмовостійкість системи. Крім того, існують

фрагменти даних, що мають бути оперативно доступними 24/7 незалежно від наявності зв’язку, навіть за рахунок втрати їх актуальності.

У розвитку сучасних ІС намітилася тенденція переходу від локальних БД до створення розподілених БД [49–53]. Це обумовлено як наведеними вище причинами, так і розвитком інформаційних технологій, а саме появою і широким поширенням технічних засобів високошвидкісної передачі даних і вдосконаленням засобів побудови розподілених систем. Відповідно все більше підприємств, що мають територіально розподілену структуру впроваджують і використовують розподілені корпоративні інформаційні системи (РКІС).

Під корпоративними інформаційними системами (КІС), згідно з [54], розуміємо сукупність інформаційних систем окремих підрозділів підприємства, об’єднаних загальним документообігом, таких, що кожна система виконує частину задач по управлінню прийняттям рішення, а всі системи разом забезпечують функціонування підприємства [18].

У свою чергу розподілені корпоративні інформаційні системи (РКІС) – це один із складних видів комп’ютерних систем, що створюється як правило для крупних підприємств або корпорацій, що об’єднують декілька організацій, у тому числі територіально розподілених. Зв’язок забезпечується за допомогою комп’ютерних мереж, у тому числі із використанням мережі Інтернет, а дані зберігаються у РБД. Питанням проєктування структури БД КІС приділяється увага у роботі [55], а методи і засоби проєктування ІС та РБД розглянуті у роботі [56].

1.3.2 Технології представлення даних та використання розподілених транзакцій

Серед загальних моделей та стратегій розподілу даних між вузлами РКІС, без урахування особливостей та обмежень конкретної розподіленої СКБД, виділяють централізовану та розподілену моделі БД. Стратегії представлення даних на базі розподіленої моделі [57-59] включають наступні класи: розподілена без дублювання; розподілена з дублюванням; та комбінована.

Централізована стратегія характеризується розміщенням всіх даних в одному вузлі мережі та наявністю системи управління доступом інших вузлів до даних. Вона має ряд переваг, серед яких простота реалізації забезпечення цілісності та захисту даних, створення та ведення файлів БД, проєктування структури БД, відсутність необхідності у синхронізації даних між ОДД та можливістю проведення інтегрованого аналізу даних у межах всієї предметної області конкретної БД. Однак також характерно виникання великих черг, та збільшення часу реакції системи, що особливо актуально у високонавантажених системах та/або при використанні довгих транзакцій. Обсяг бази даних обмежений ресурсами одного серверу баз даних. Актуальним є також питання навантаження на мережу при передачі даних на вузол РКІС, особливо у випадку відсутності надійного каналу зв’язку між сервером БД та клієнтським застосунком.

При розподіленій стратегії без дублювання структура РБД проєктується, як неперетинні між собою підмножини даних, розподілені по вузлах РКІС, що є досить складною задачею. Ця стратегія оптимальна у випадку мінімальної кількість логічних посилань взаємозв'язків вузлів одного з одним, коли кожен вузол працює зі своїми даними та майже не використовує дані інших вузлів РКІС. При цьому зменшуються витрати на передавання інформації та забезпечується доступність та швидкість доступу до даних, зменшується вірогідність виникнення черг, збільшується швидкість обробки локальних запитів. Однак наряду із цим виникає необхідність використання розподілених транзакцій для організації доступу до даних на різних вузлах РКІС. Використання розподілених транзакцій у свою чергу передбачає одночасну доступність всіх вузлів, а блокування, накладені на кожному окремому вузлі РКІС будуть зняті лише після повного завершення роботи розподіленої транзакції.

За розподіленої стратегії з дублюванням проєктування БД виконується як за централізованого підходу, але із фізичним її дублюванням в кожному вузлі РКІС. Дана стратегія ефективно розв'язує проблеми доступу та вибірки даних з мінімальними витратами часу, а система досить проста при проєктуванні. При

цьому цей підхід характеризується складністю адміністрування та розв'язання проблеми узгодженості файлів БД у різних вузлах РКІС, особливо при виникненні розбіжностей у даних, що вимагає спеціальних механізмів їх узгодження.

Комбінована стратегія поєднує підходи розподілу без дублювання та з дублюванням даних. Ця стратегія поділяє БД на багато логічних фрагментів та дозволяє мати довільну кількість фізичних копій кожного фрагмента. Система, побудована за цією стратегією, допускає реалізацію паралельної обробки даних, що скорочує час відгуку та забезпечує більш високу надійність даних за рахунок їх часткового дублювання. Однак залишається проблема узгодженості копій фрагментів БД у всіх вузлах РКІС.

У роботах [53; 60–64] розглядаються підходи до проєктування БД, у тому числі розподілених, із урахуванням горизонтальної та вертикальної фрагментації даних та стратегій з/без дублюванням даних; розглядається процес інтеграції програмних компонентів розподіленої ІС; а також роль і місце інтегрованих БД у розподілених ІС. Враховуючи результати аналізу наведених наукових робіт, а також особливості різних стратегій представлення даних в РКІС, комбінована стратегія є найбільш виправданою, однак актуальною постає задача оптимізації структури БД вузла РКІС, а також задача синхронізації даних.

1.3.3 Технології СКБД та ІС щодо інтеграції даних

Синхронізація даних між однорідними та/або різнорідними джерелами даних може бути здійснена за допомогою різних методів. По-перше, у розпорядженні майже бідь-якої сучасної СКБД існують механізми по керуванню розподіленими транзакціями, забезпеченню дзеркального відображення та реплікації БД та їх окремих елементів [53; 65-70]. Іншим підходом може бути використання інструментів синхронізації сторонніх розробників, що дозволяють із одного середовища взаємодіяти із різними СКБД, наприклад EMS Data Comparer [71] або Akeneo [72].

Деякі ІС мають у своєму розпорядженні механізми файлового обміну даними із можливістю налаштування автоматичного або напівавтоматичного режиму оновлення, що реалізовано безпосередньо розробниками OLTP-систем та враховують особливості представлення сутностей в інформаційній моделі [73]. Крім того, при відомій структурі БД OLTP-системи завжди існує можливість розробки власної системи синхронізації, яка в більшості випадків буде поєднувати методи СКБД та ІС для можливості використати переваги кожного з підходів, врахувавши їх недоліки.

Реплікація є набором технологій, за допомогою яких дані або об'єкти БД можна копіювати і переносити з однієї БД в іншу, а потім синхронізувати ці БД для забезпечення узгодженості. Виділяють три категорії даних, що беруть участь в процесі реплікації:

– оновлювані тільки на центральному вузлі;

– відфільтровані дані, що оновлюються тільки на одному з віддалених вузлів;

– дані, які можуть оновлюватися декількома користувачами незалежно один від одного.

В останньому випадку при синхронізації даних можуть виникати конфлікти. Процес виявлення і вирішення конфліктів може займати багато часу і ресурсів, тому рекомендовано мінімізувати кількість конфліктів застосунку, створюючи секції даних так, щоб різні вузли РКІС змінювали різні підмножини даних. Одним з основних параметрів є фільтрація даних, що дозволяє фільтрувати стовбці і рядки так, що таблиці міститимуть тільки дані, необхідні для застосунку.

1.4 Методи та технології оптимізації структури БД на вузлах РКІС

Ключовим фактором, який впливає на надійність і доступність БД, є так звана «локалізація посилань». Якщо БД розподілена таким чином, що розміщені на вузлі дані викликаються винятково його користувачами, це свідчить про високий ступінь локалізації посилань. Якщо подібну фрагментацію даних здійснити неможливо і для виконання запитів потрібно звертатись за

інформацією до інших вузлів, то це свідчить про невисокий ступінь локалізації посилань.

Комбінована стратегія розподілу даних є найбільш виправданою із точки зору можливості поєднання переваг стратегій з та без дублювання та дозволяє досягти найвищого рівня локалізації посилань при мінімальній збитковості даних. Але при її використанні, окрім задачі синхронізації дубльованої інформації, актуальною постає задача оптимального проєктування структури БД з точки зору приналежності даних до того чи іншого вузла РКІС. Крім того, продуктивність системи напряму буде залежати від прийняття рішення про необхідність часткового або повного дублювання даних.

Враховуючи закритість коду SQL-запитів РКІС, задача оптимального проєктування структури БД полягає у пошуку оптимального розподілу даних по вузлах РКІС.

Класичні методи пошуку оптимуму, такі як метод Гоморі, метод гілок та границь не можуть бути застосовані у зв’язку із характером задачі, що вирішується, а також її великою розмірністю. Метод повного перебору, що дозволяє знайти глобальний оптимум, при великих розмірах дискретної задачі пошуку оптимального розподілу даних по вузлах РКІС [74] не може бути застосований через неприпустимі часові витрати на проведення розрахунків.

У роботі Землянської С. Ю. для пошуку квазіоптимального рішення запропоновано використовувати еволюційні методи [49]. Розглянуто три групи еволюційних алгоритмів (генетичні методи, методи поведінки натовпу, методи колонії мурах) та зроблено висновки про переваги групи генетичний алгоритмів над іншими. Генетичні алгоритми імітують еволюційний процес, наближаючись до оптимального значення, починаючи із деякого вихідного покоління. Ключовим аспектом є використання імітаційної моделі, що моделює поведінку РКІС та надає можливість для розрахунку фітнес-функції при зміні параметрів системи.

Однак імітаційні моделі РКІС, що є вкрай складною системою, зазвичай дуже спрощені та не враховують багатьох факторів, що впливають на її поведінку.

Так, наприклад, у роботі [49] залишається без уваги вплив особливостей виконання однакових SQL-запитів у середовищах різних СКБД. Не розглядаються особливості транзакційного режиму роботи із даними та роботи на різних рівнях ізоляції, що безпосередньо впливає на кількість блокувань, та взаємний вплив процесу виконання одних запитів на інші. Вкрай важливими також є налаштування серверу БД, від яких напряму залежать як продуктивність окремого запиту, так і їх взаємний вплив один на одного. Ускладнення моделі поглиблюється із уведенням розподілених запитів та розподілених транзакцій з двофазним протоколом їх фіксації та можливою наявністю різних СКБД у випадку гетерогенних джерел даних [51] або СД [75]. Також більшість існуючих підходів оптимізації структури РБД та КІС розглядають оптимальне розташування попередньо визначених фрагментів даних на вузлах РКІС, але не розглядають оптимізацію самого процесу фрагментації.

Відповідно до наведеного, використання імітаційних моделей можна вважати недоцільним. Натомість запропоновано накопичення статистичної вибірки виконання SQL-запитів РКІС до РБД на реально працюючих серверах, що дає змогу врахувати всі вищенаведені особливості РКІС. Парсинг текстів SQL-запитів та розщеплення на окремі підзапити дозволить виявити множини відношень, атрибутів та кортежів БД, задіяних у кожному запиті [76–78]. Подальший аналіз накопиченої та обробленої статистики в аналітиці доступних вимірів у вигляді програмного застосунку, хоста, користувача та ін. дозволить виявити частоту та характер звернень до окремих відношень та їх атрибутів і кортежів.

1.5 Оптимізація структури БД вузла РКІС за критеріями оптимальності

При формулюванні задачі оптимізації структури БД вузла РКІС запропоновано виконання оцінки оптимальності структури за значеннями відповідних критеріїв оптимальності, а саме: доступності та швидкості отримання даних, незалежності від центрального вузла БД, розміру БД, ступеня

достовірності даних, необхідності у подальшій синхронізації. Деякі з критеріїв, що мають спільну природу, об’єднано у спільні групи, після чого отримано три групи критеріїв: «незалежність БД», «розмір БД» та «необхідність синхронізації». Кожен атрибут та кортеж відношення згідно з результатами обробки статистики SQL-запитів маркується відповідним значенням необхідності представлення на вузлі РКІС. Значення критеріїв виражається у вигляді залежності від рівня представленості даних на вузлі РКІС, а задача зводиться до визначення оптимального значення маркеру представленості даних на вузлі РКІС.

1.5.1 Формулювання задачі багатокритеріальної оптимізації

Процес прийняття рішень, та методи його підтримки відносяться до теорії прийняття рішень та часто реалізуються у комплексах комп’ютерних програм, що мають загальну назву системи підтримки прийняття рішень (СППР). В процесі прийняття рішення, людина, що приймає рішення (ЛПР) може також виступати у ролі експерта. Прийняття рішення передбачає вибір одного з можливих варіантів дій, що прийнято називати альтернативами. У деяких випадках альтернатив може виявитись забагато для ЛПР, що вимагає залучення додаткових заходів підтримки вибору рішення. До цього випадку потрапляє також і задача визначення оптимального рівня маркеру представленості даних.

Кожне значення рівня маркеру представленості даних характеризується показниками їх привабливості для ЛПР, а саме такими критеріями оптимальності, як незалежність від центрального вузла БД, співвідношення розміру локальної БД до центральної, та показник рівня необхідності синхронізації даних. Всі вони є критеріями вибору рішення. У процесі прийняття рішення нерідко до участі залучаються експерти, але враховуючи конфіденційність даних, самого процесу прийняття рішення, та його результату, для ЛПР це не завжди є прийнятним. Крім того, наявність експертів завжди підвищує вартість системи. Враховуючи це, ставиться на меті реалізація методів, при використанні яких експерти можуть бути залучені, але ЛПР також може виступати в ролі експерта.

При визначенні маркеру представленості даних зв’язок між його значенням (альтернативою) та значеннями критеріїв для оцінки цих альтернатив (незалежності локальної БД, необхідності у синхронізації та розміру БД) задається у вигляді математичних моделей. Отже, це дозволяє, обравши один із методів вирішення багатокритеріальної задачі дослідження операцій, перейти до знаходження оптимального рішення.

Традиційний підхід дослідження операцій передбачає наявність єдиного критерію оцінки якості прийнятого рішення [79]. Одним із перших підходів, що використовується для 2-х критеріїв, яким є метод «вартість-результат», що полягає у побудові моделей результативності та вартості, та подальшого синтезу їх оцінок. Однак метод має ряд недоліків [80], крім того, у нашому випадку при наявності трьох критеріїв, одна з моделей (в залежності від того, зарахувати критерій необхідності синхронізації даних до вартості або до результату) перетворюється на багатокритеріальну.

Задача визначення оптимального рівня маркеру представленості даних є добре структурованою, оскільки всі залежності можуть бути представленні у чисельному вигляді. Також критерії оптимальності (рівень незалежності локальної БД, ступінь необхідності у синхронізації даних та розмір локальної БД) є незалежними, і може буди задано напрямок покращення значень кожного з критеріїв. Так, при збільшенні значення маркеру представленості даних, ступінь необхідності у синхронізації зростає або залишається сталою, розмір локальної БД збільшується, рівень незалежності локальної БД також зростає або залишається сталим. Виходячи із наведеного, дану задачу можна віднести до класу задач багатокритеріальної оптимізації (БКО).

У випадку багатокритеріальних задачах частина інформації, необхідна для повного та однозначного визначення вимог до рішення, принципово відсутня [81]. Так, було визначено критерії оптимальності, та встановлено зв’язок між ними та альтернативами (значенням маркеру представленості даних). Але кращі поєднання критеріїв не можуть бути визначені на основі об'єктивної інформації,

наявної в розпорядженні дослідника. Враховуючи наведене, дана задача може бути визначена, як слабо структурована.

Наявність декількох критеріїв призводить до зміни характеру розв'язуваної задачі і ролі ЛПР, суб'єктивні переваги якої мають стати основою для вироблення рішення. Рішення хоч і буде суб'єктивним, але в процесі вирішення будуть використані об'єктивні моделі. Використання багатокритеріальних методів дозволяє ЛПР уникнути поверхневих рішень та схильності до спрощення задачі, внаслідок чого замість декількох критеріїв буде розглянуто лише один, що вбачається ЛПР більш важливим.

Набір рішень, що містить в собі ті рішення, які можна використовувати при пошуку найкращого, прийнято називати простором рішень (далі *W*). Деякі з елементів простору рішень (що задається неоднозначно) неможливо реалізувати з тих чи інших причин. Ті рішення, які можуть бути використані при пошуку найкращого рішення, утворюють підмножину *X* простору рішень *W*. Таку множину $X \subset W$ називаємо множиною допустимих рішень. Нехай задана числова функція *f*, значення якої описують рівень переваги рішень. В цьому випадку природно вважати, що метою є збільшення значення функції *f*. У даному вигляді рішення трактується як пошук деякого елемента множини *X*, при якому значення *f* максимальне [82]. Таким чином, приходимо до задачі однокритеріальної (скалярної) оптимізації: $f(x) \to max, x \in X$.

Елемент $x^* \in X$ називається рішенням скалярної задачі оптимізації, якщо $f(x^*) > f(x)$ для всіх $x \in X$. Набір з трьох критеріїв вибору рішення є сукупність функцій (f_1, f_2, f_3), заданих на просторі *W*. Поряд з простором рішень *W*, можна представити критеріальний простір *W'*, що включає в себе прямий добуток шкал критеріїв (множин можливих значень окремого критерію). Таким чином, сукупність критеріальних функцій задає відображення $f = (f_1, f_2, f_3)$, що діє з *W* в *W'*.

Важливою умовою є незалежність критеріїв за перевагою, при якій бажані для ЛПР зміни значень кожного з окремих критеріїв при незмінних значеннях

інших критеріїв не повинні залежати від конкретних значень інших критеріїв. Для будь-якої пари значень з шкали кожного з критеріїв (скажімо, y_j' и y_j'') ЛПР в змозі визначити, що одне значення краще іншого $y_j' \succ y_j''$, або значення рівноцінні $y_j' \approx y_j''$. При цьому повинна виконуватися вимога несуперечності тверджень про переваги, тобто якщо $y_j' \succ y_j''$ та $y_j'' \succ y_j'''$, то $y_j' \succ y_j'''$, і якщо $y_j' \approx y_j''$ та $y_j'' \approx y_j'''$, то $y_j' \approx y_j'''$.

Таким чином, переваги, сформульовані ЛПР, зручно виразити за допомогою математичного поняття, що характеризує співвідношення між парами об'єктів, тобто через бінарні відношення.

Критерії оптимальності значення маркеру представленості даних, а саме рівень незалежності локальної БД, ступінь необхідності у синхронізації даних та розмір локальної БД є числовими, незалежними за перевагою та ЛПР у більшості випадків може порівняти будь-яку пару значень за шкалою кожного з окремих критеріїв. Критеріальний простір є тривимірним лінійним простором, а окремі критерії не тільки є числовими, але і набори їх значень можуть додаватись та помножатись на число.

Як вже зазначалося, критерії є монотонними за перевагою, що означає збільшення (або зменшення) значення окремого критерію (при постійних значеннях інших критеріїв) для ЛПР. Щоб звести задачу до класичної задачі багатокритеріальної максимізації, необхідно дотримання умови бажаності для ЛПР збільшення значення кожного із окремих критеріїв. У нашому випадку бажаним є збільшення значення критерію рівня незалежності локальної БД, а ступінь необхідності у синхронізації даних та розмір локальної БД бажано зменшувати. Тому, для зведення задачі до виду багатокритеріальної максимізації, останні два критерія мають бути помножені на «–1». Враховуючи всі зазначені особливості, для порівняння критеріальних векторів у нашому випадку може бути використане алгебраїчне порівняння значень окремих критеріїв.

При відсутності конфлікту між критеріями, можливе рішення задачі може бути представлене у вигляді ідеальної точки. Під ідеальною точкою розуміють

такий вектор $y^* \in W'$, компоненти якого є максимумами окремих критеріальних функцій $f(x)$ окремих критеріїв. Але враховуючи той факт, що у нашому випадку зміна значення рівня маркеру представленості даних (зміна альтернативи) веде до покращення значення одних критеріїв за рахунок погіршення значення інших, ідеальної точки на критеріальному просторі допустимих рішень не існує, а отже не існує і абсолютно оптимального рішення $x^* \in X$. Отже, для знаходження єдиного оптимального рішення, задача має бути зведена до однокритеріальної.

1.5.2 Аналіз методів розв'язання задачі БКО

При вирішенні задачі БКО основною задачею є перехід до однокритеріальної задачі, або послідовності таких задач, що може бути вирішена одним із методів дослідження операцій [83]. Серед основних підходів слід згадати отримання суперкритерію шляхом згортки критеріїв, методи звуження множини альтернатив, групування критеріїв, а також методи головного критерія, ідеальної точки та послідовних поступок [84; 85].

Методи звуження множини альтернатив у більшості випадків базуються на визначенні множини рішень, ефективних за Парето (Стейтором, Смейлом) [86; 87]. На першому кроці моделювання на віддаленому вузлі БД територіально розосередженої КІС запропоновано створення повної копії БД центрального вузла. Далі із її складу виключаються всі надлишкові дані, до яких не виконується жодних звернень користувацьких SQL-запитів. Всі інші альтернативи, що перебувають між цим станом і станом повної відсутності БД у віддаленому вузлі, входять до множини рішень, ефективних за Парето, оскільки призводять до покращення рівня значень одних критеріїв за рахунок погіршення інших. Методи групування критеріїв використовуються при великій кількості останніх, що у розглянутому випадку при наявності трьох критеріїв оптимальності (незалежність від центрального вузла БД, розмір локальної БД, та показник рівня необхідності синхронізації даних) неактуально.

Серед методів згортки критеріїв виділяють адитивні, мультиплікативні та агреговані (мінімаксні та максимінні) методи. Так згортка може бути виконана

наступним чином $F(A_i) = \sum_{j=1}^{n} \alpha_j * f_j(A_i)$, або за однією з наступних формул $F(A_i) = \prod_{j=1}^{n} f_j(A_i)^{\alpha^j}$, $F(A_i) = 1 - \prod_{j=1}^{n}(1 - \frac{\alpha^j * f_j(A_i)}{S_j})$, $F(A_i) = \frac{\prod_{j=1}^{m} f_j(A_i)^{\alpha^j}}{\prod_{j=m+1}^{n} f_j(A_i)^{\alpha^j}}$, або ж із використанням агрегування $F(A_i) = extr_j(\alpha_j * f_j(A_i))$, де $F(A_i)$ – розраховане значення суперкритерію за і-ю альтернативою, $f_j(A_i)$ – значення j-го критерія оптимальності за і-ю альтернативою, а α_j – ваговий коефіцієнт j-го критерія оптимальності. Недоліками методів згортки є складність аргументування вибору конкретного методу (різні методи дають різні результати), обґрунтування вибору вагових коефіцієнтів альтернатив та можливість компенсування малих значень одних параметрів великими значеннями інших.

Метод головного критерію, що полягає у виборі одного критерію у якості вирішального, та використання двох інших у вигляді обмежень, переважає за рахунок простоти інтерпретації результатів та відсутності додаткових вимог до експертів та обчислювальних засобів. Однак у нашому випадку залучення експертів вважається невиправданим через конфіденційність даних, тому переваги перекриваються цілим рядом суттєвих недоліків, а саме: надмірним спрощенням структури задачі; суб'єктивізмом при виборі головного критерія; можливістю втрати сукупного впливу двох другорядних критеріїв і, як наслідок, отримання неефективного рішення. Також при деякій комбінації обмежень по двох неосновних критеріях існує ймовірність отримання порожньої множини допустимих рішень.

Повертаючись до методу ідеальної точки та методу послідовних поступок слід звернути увагу на можливість отримання у вигляді рішення неефективних за Парето альтернатив. Крім того, обчислення відстані від ідеальної точки у нашому випадку дає різні результати при використанні різних методів його визначення (Евклідова відстань, манхеттенівська відстань, відстань Чебишева тощо) [80]. Визначення відсотку поступки та міри співвідношення отриманого виграшу за

неосновним критерієм у порівнянні із програшом за основним також може виявитися для ЛПР складною, а іноді, навіть невизначено складною операцією.

Серед інших методів виділено метод аналізу ієрархій (MAI), що є загальною методологією розв'язання широкого класу задач прийняття рішень і дозволяє поєднати порівняно простий математичний апарат зі знаннями та досвітом ЛПР. Основою даного метода є представлення процесу рішення у вигляді багаторівневої ієрархії, яка має відображати всі складові задачі, що вирішується [88; 89].

MAI дозволяє виконувати такі етапи аналізу багатокритеріальних задач прийняття рішень, як: структурування задачі та формалізація зв'язків між складовими; формування системи переваг ЛПР для критеріїв та альтернатив. Перевагами методу є наочність моделей, простота інтерпретації результатів, можливість оцінки альтернатив по якісним та суб'єктивно-визначеним критеріям, а також стійкість до порушення узгодженості оцінок ЛПР. Основу метода складають принципи декомпозиції, парних порівнянь та ієрархічної композиції. Основними етапами методу є побудова ієрархії, оцінювання значимості та пріоритетів, перевірка узгодженості пріоритетів та синтез рішення [90; 91].

Висновки до розділу 1

Розглянуті технології, моделі та методи підвищення швидкості роботи інформаційних систем (ІС), серед яких оптимізація запитів, індексація даних, модифікація структури БД, контрольована денормалізація відношень та використання матеріалізованих представлень дозволяє виділити загальний недолік, який полягає у необхідності модифікації самої ІС, що у випадку закритої реалізації останніх робить неможливим їх використання.

Ряд проблем використання «універсальних» або «комплексних» ІС, обумовлює тенденцію використання набору окремих спеціалізованих рішень із подальшою синхронізацією даних. Комбінована стратегія представлення даних в розподілених БД поділяє базу даних на логічні фрагменти. При цьому ключовим фактором, який впливає на надійність і доступність БД, є локалізація посилань.

Враховуючи закритість коду SQL-запитів РКІС, задача оптимального проєктування структури БД зводиться до пошуку оптимального розподілу даних по вузлах РКІС. Імітаційні моделі РКІС, що є вкрай складною системою, зазвичай дуже спрощені та не враховують ряд важливих факторів, що впливають на її поведінку. Натомість запропоновано накопичення статистичної вибірки виконання SQL-запитів РКІС до розподіленої БД на реально працюючих серверах, що дає змогу врахувати особливості роботи РКІС.

Парсинг текстів SQL-запитів та розщеплення їх на окремі підзапити дозволить виявити множини відношень, атрибутів та кортежів БД, задіяних у кожному запиті. Подальший аналіз накопиченої та обробленої статистики в аналітиці доступних вимірів у вигляді програмного застосунка, хоста, користувача та ін. дозволить виявити частоту та характер звернень до окремих відношень, їх атрибутів і кортежів. Вводиться набір критеріїв оптимальності структури БД та формулюється задача багатокритеріальної оптимізації. Проаналізовано різні методи її розв’язання, серед яких виділяється метод аналізу ієрархій.

Отже, дослідження, спрямоване на підвищення рівня доступності даних та ефективності використання розподілених та територіально розосереджених комп’ютерних систем шляхом реалізації інформаційної технології оптимізації структури БД вузла РКІС на базі статистики SQL-запитів, є актуальним.

Для виконання оптимізації структури БД вузла у РКІС, підвищення рівня доступності даних та ефективності використання розподілених та територіально розосереджених комп’ютерних систем шляхом реалізації інформаційної технології оптимізації структури БД віддаленого вузла із використанням багатовимірної моделі SQL-запитів необхідно вирішити ряд актуальних науково-практичних задач.

На першому етапі постає завдання реалізації накопичення статистичних даних SQL-запитів та технології парсингу тексту із розщеплення запитів з метою виявлення множин відношень, атрибутів та кортежів БД, задіяних в SQL-запитах. Розробити реляційну та багатовимірну БД збереження отриманих даних SQL-

запитів, що дозволить обчислювати значення маркеру представленості даних для окремих атрибутів та кортежів відношень БД.

Наступним завданням є створення математичних моделей для описання залежності критеріїв оптимальності структури БД вузла РКІС від граничного рівня маркеру представленості даних вузла. Формалізація критеріїв дає можливість визначити задачу БКО, розв’язання якої запропоновано виконувати, використовуючи МАІ. Необхідно визначити ієрархічну модель та виконати програмну реалізацію обчислення вектору глобальних пріоритетів із урахуванням розбиття інтервалу допустимих значень рівня маркеру представленості на скінченну кількість альтернатив та введення обмежень на область значень критеріїв оптимальності.

Отримане значення оптимального рівня маркеру представленості даних дозволить прийняти рішення з представленості кожного окремого атрибуту та кортежу на вузлі РКІС. Для нових даних БД, зважаючи на складність повторного виконання аналізу, необхідно вирішити задачу класифікації для визначення приналежності їх до одного з двох класів – «потрібних» та «непотрібних» на вузлі РКІС. Для розв’язання задачі може бути використана множина проаналізованих кортежів відношення щодо рішення про представленість на віддаленому вузлі РКІС.

РОЗДІЛ 2
МОДЕЛІ SQL-ЗАПИТУ, СТРУКТУРИ БД ВУЗЛА РКІС ТА КРИТЕРІЇВ ЇЇ ОПТИМАЛЬНОСТІ

2.1 Модель відношень та структури реляційної БД на базі теорії множин

Реляційна модель даних базується на простому й у той же час потужному математичному апараті, що спирається, головним чином, на теорію множин і математичну логіку [92-93]. При побудові математичної моделі РБД а також формалізації роботи розподіленої транзакції, механізмів синхронного та асинхронного оновлення даних [66; 94], Extracting Transformation Loading (ETL) процесів вважається за доцільне використання базових понять теорії множин та реляційної моделі даних із їх подальшим розширенням та доповненням за рахунок описання нових елементів, що не були присутні при побудові моделі одиничної централізованої реляційної БД.

Основними поняттями реляційних БД є тип даних, домен, атрибут, кортеж, ключ (первинний та зовнішній) та відношення (рис.2.1).

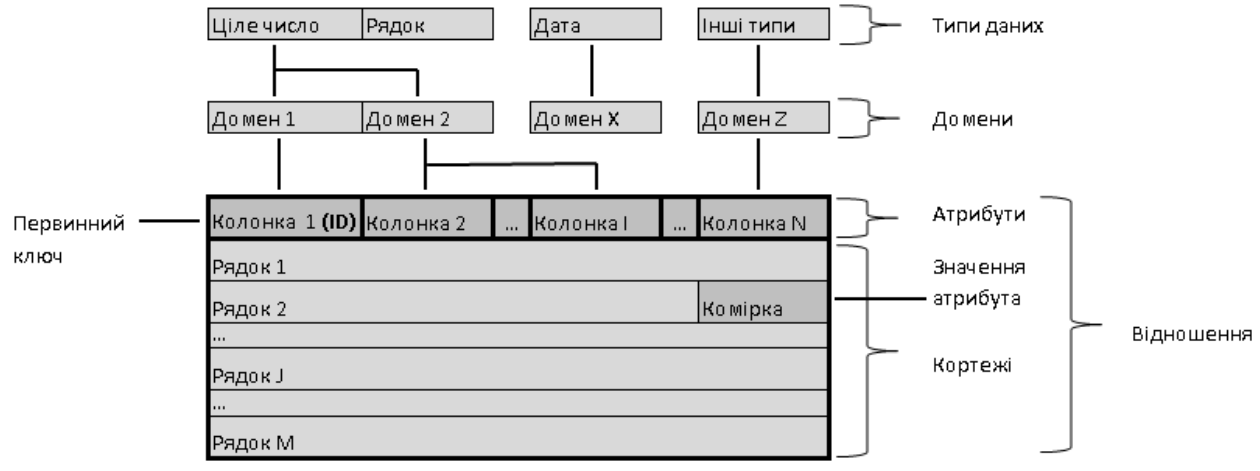


Рисунок 2.1 – Взаємозв’язок типів даних, доменів, атрибутів, кортежів, первинних ключів та відношень

Поняття типу даних в реляційній моделі співставне із поняттям типу даних в мовах програмування [95–98]. Також активно розвиваються механізми по

розширенню можливостей реляційних систем абстрактними, або користувацькими типами даних, що визначаються на базі вбудованих типів, правил та обмежень цілісності даних.

Множина вбудованих типів може бути визначена, як:

$$T_{inner}=\{t_{inner\_1},\ \ldots,\ t_{inner\_n}\},$$

де t_{inner} – вбудований тип даних;

n – загальна кількість доступних вбудованих типів.

Множину користувацьких типів представляється, як:

$$T_{outer}=\{t_{outer\_1},\ \ldots,\ t_{outer\_m}\},$$

де t_{outer} – користувацький тип даних;

m – загальна кількість визначених користувацьких типів.

Загальна множина доступних для визначення атрибутів типів даних може бути визначена, як об'єднання $T= T_{inner}\ U\ T_{outer}$.

Поняття домену в загальному вигляді визначається заданням деякого базового типу даних, до якого входять елементи домена і довільного логічного вираження, що застосовується до елементу типу даних [26, 28]. Якщо обчислення цього логічного виразу дає результат «істина», елемент даних є елементом домена. На домен іноді накладаються додаткові обмеження, що не можуть бути реалізовані за допомогою звичайних правил. У цьому випадку можуть бути використані обмеження у вигляді зовнішнього ключа або тригера. У загальному вигляді домен може бути виражений як множина: $D = \{d_1, \ldots, d_n\}$, де d – значення елементу, що визначається на домені D, n – кількість цих значень [28, 92]. При чому, кожен елемент d обмежений областю значень свого типу даних, тобто

$$\forall\ D\ \exists\ (t_j \in T)\ \forall\ (\ d_i \in D)\ (d_i \in t_j),$$

де t_j – тип даних, на якому визначається домен D. Тобто, фактично $D \subset t_j$. Множина всіх наявних у БД доменів може бути визначена, як $D_{all} = \{D_1, \ldots, D_n\}$.

Атрибут визначається, як пара (A_{name}, D), де A_{name} – ім'я атрибута, D – домен, на якому він визначається [28, 29]. У різних атрибутів домени можуть бути однаковими, а їх імена мають бути різними (унікальними). Схема відношення

(таблиці БД) – це іменована множина атрибутів, тобто пар (ім’я атрибута, домен). Ступінь, або арність (потужність) схеми відношення визначається кількістю елементів цієї множини.

Під терміном «відношення» R надалі розуміємо підмножину декартового добутку доменів $R \subset d_1 x\ d_2\ x\ \ldots\ x\ d_n = \{< a_1, \ldots, a_n >\}$ для якої висловлювання $R(A_1, \ldots, A_n) = істина$ [28]. Графічно відношення представляється у вигляді таблиці. Схему відношенні R визначено, як $R_{shema} = \{A_1, \ldots, A_n\}$, де $A = (A_{name}, D_j)$ – атрибут відношення.

До складу відношення входить деяка (в окремих випадках порожня) множина кортежів (або рядків таблиці) [92]. Відношення має просту графічну інтерпретацію (рис. 2.2), воно може буде представлене у вигляді таблиці, стовпці (поля, атрибути) якої відповідають входженням доменів у відношення, а рядки (записи, кортежі) — наборам з n значень, що взяті з початкових доменів. Кількість рядків n, називають кардинальним числом відношення або потужністю відношення.

Рисунок 2.2 – Графічне представлення відношення [99]

Тобто, з точки зору множини кортежів, відношення може бути представлене наступним чином: $R_{data} = \{Tup_1, \ldots, Tup_p\}$, де Tup – кортеж відношення.

У свою чергу, кожен кортеж є множиною значень атрибутів цього відношення: $Tup = \{V_1, \ldots, V_n\}$, де $V_k = (A_{k\_name}, Val)$ – значення атрибуту відношення. Слід зауважити, що кожне значення визначається на домені відповідного атрибуту $Val \in D_j$.

Відповідно до реляційної моделі відношення не може мати двох однакових кортежів [26, 29]. Тобто існує така підмножина на множині атрибутів відношення, значення яких для певного кортежу однозначно його ідентифікує, що у теорії реляційних БД має назву первинний або потенційний ключ [100–102]: $R_{primary} = \{ A_1^{pr}, \ldots, A_n^{pr} \} \subset R_{shema}$.

Також в окремих випадках відношення має множину зовнішніх ключів в вигляді посилань на первинні (або потенційні) ключі інших відношень, кожен з яких може бути представлено у вигляді наступної множини: $R_i^{foreign} = \{A_1^{foreign}, \ldots, A_n^{foreign}\}$, $R_{foreign} \subset R_{shema}$, $A_1^{foreign} \in D_1^{primary}$, де $D_1^{primary}$ – домен, на якому визначається відповідний атрибут первинного ключа іншого відношення, на який посилається поточний зовнішній ключ.

Множина всіх таблиць БД може бути представлена наступним чином: $DB = \{R_1, \ldots, R_n\}$, де $R_k = (R_{shema}, R_{data}, R_{primary}, R_{foreign})$, причому кожна з таблиць (відношень) має множину відношень, на які виконано посилання за допомогою конструкції зовнішній ключ $R^{main} = \{R_1, \ldots R_m\}$. На основі множини R^{main} для всіх таблиць БД, для кожного відношення визначено множину залежних відношень $R^{dep} = \{R \mid R^{dep} \in R^{main} \}$.

При відсутності зовнішніх ключів відношення, множина $R^{main} = \emptyset$. Також, якщо таблиця перебуває на найнижчому рівні ієрархіє, може бути порожньою і множина R^{dep} . У випадку, якщо порожніми є обидві множини, можна говорити, що відношення не має жодних логічних зв’язків з іншими.

2.2 Формалізація підмножин даних на базі користувацьких запитів та вибір стратегії представлення даних

Виконуючи розгляд користувацьких запитів до реляційної БД, наведемо 8 операцій реляційної алгебри, що були визначені Коддом та розбиті на 2 групи: традиційні оператори: декартів добуток, об'єднання, перетин та різниця; та спеціальні оператори: «вибірка», «проєкція», «з'єднання» і «розподіл» [92]. Властивість замкнутості наведених операцій говорить про те, що результатом кожної операції над відношенням також є відношення.

Зважаючи на особливості задачі проєктування структури БД територіально віддаленого вузла з точки зору представлення необхідних вузлу даних, де має місце використання горизонтальної та вертикальної фрагментації даних, далі розглянуто операції «проєкції» та «вибірки». Нехай *tup* – кортеж з відношення *R*, *tup[P]* – частина цього кортежу, що містить тільки значення атрибутів, які входять до підмножини *P* схеми відношення R_{shema} $(P \subset R_{shema})$. Тоді проєкцією *R* на *P* буде відношення, що складаються з кортежів усіх значень з множини *P*, що існують у відношенні *R*, тобто $R[P] = \{tup[P] \mid tup \in R_{data}\}$. Схема результуючого відношення може бути визначена наступною множиною атрибутів: $R[P]_{shema} = \{A_1 \ldots, A_m\}$, де $\forall A \in R_{shema}$.

Якщо проєкція будує відображення одного атрибута на інші, то вибірка виконує відображення кортежів, результатом якого є відношення, що містить підмножину всіх кортежів відношення *R*, для яких виконується певна логічна умова: $R[S] = \{ tup \mid tup \in R_{data} \wedge F(tup, S) = істина \}$, де *S* – логічна умова, а *F(tup, S)* – функція, що відображає її виконання для відповідного кортежу. Схема результуючого відношення буде збігатись зі схемою базового відношення, тобто $R[S]_{shema} = R_{shema}$.

SQL-запит на вибірку даних може включати деяку множину відношень, кожне з яких є результатом послідовного виконання операцій вибірки та проєкції до базового відношення (таблиці БД). $R'' = R'[P]$, де $R' = R[S]$, тобто

$$R'' = \{tup[P] \mid tup[P] \in R[P]_{data} \wedge F(tup, S) = \text{істина}\} \quad (2.1)$$

Розглядаючи множину запитів до БД, результуюча підмножина R''_{union} базового відношення R визначається, як об'єднання підмножин R'' всіх запитів, що надійшли до БД з віддаленого вузла $R''_{union} = \bigcup_{i=1}^{n} R_i''$, або $R''_{union} = \{tup[P_{union}] \mid tup[P_{union}] \in R[P_{union}]_{data} \wedge F(tup, S_{union}) = \text{істина}\}$, де $tup[P_{union}] = \bigcup_{i=1}^{n} tup[P_i]$, а $S_{union} = \bigvee_{i=1}^{n} S_i$.

Враховуючи, що деякі дані, необхідні на вузлі РБД, для запобігання необхідності подальшої реплікації можуть бути представлені лише на центральному вузлі БД та приймати участь у роботі за рахунок використання розподілених запитів, результуюче відношення R^{remote} буде лише підмножиною R''_{union}. Для представлення даних на вузлі територіально розосередженої КІС необхідно використати елементи як вертикальної, так і горизонтальної фрагментації даних (як проекцію, так і вибірку), відповідно до чого підмножина базового відношення R, що буде описувати відношення вузла РКІС, представлено наступним чином:

$$R_{schema}^{remote} = \{A \mid A \in R_{shema}, R_{primary} \subset R_{schema}^{remote}, A \in R_{primary} \vee F_a(Node, A) = істина\} \quad (2.2)$$

Для прийняття рішення про представленість певного атрибуту відношення на тому чи іншому вузлі, використано функцію $F_a(Node, A)$. Зауважимо, що множина атрибутів первинного ключа відношення у будь-якому випадку має бути представлена на віддаленому вузлі.

Множина кортежів у свою чергу визначатиметься за формулою:

$$R_{data}^{remote} = \{ tup \mid tup \in R_{data}, tup_{primary} \in R_{data}^{remote-dep} \vee F_{tup}(Node, tup) = \text{істина}\} \quad (2.3)$$

Таким чином, кортеж має бути представлений у випадку входження його первинного ключа до множини даних відношень, залежних від поточного. В

іншому випадку необхідність наявності даних вирішується за допомогою оціночної функції $F_{tup}(Node, tup)$.

Модель представлення користувацьких запитів має підтримувати можливість їх подальшої класифікації згідно приналежності до того чи іншого хосту, географічного розташування, ролі користувача та інших критеріїв, що можливо додати до моделі під час її використання згідно з особливостями тієї чи іншої предметної області [103]. Тобто, користувацький запит визначається, як

$$Q = \{Хост, Користувач, Застосунок, ДодВластивості, R_{set}'', Q_{set}^{inner}\}, \quad (2.4)$$

де $Хост = \{Тип, Розташування\}$ – пристрій, що характеризується типом та розташуванням;

$Користувач = \{Роль, Ім'я\}$ – користувач КІС із ім'ям та роллю;

$ДодВластивості = \{Властивість_1, \ldots Властивість_n\}$ – множина додаткових властивостей запиту згідно з особливостями предметної області ІС;

$R_{set}'' = \{ R'' \mid \{tup[P] \mid tup[P] \in R[P]_{data} \wedge F(tup, S) = істина\}$ – множина результуючих відношень, отриманих із базових відношень (таблиць) БД відповідним запитом;

Q_{set}^{inner} – множина вкладених запитів основного запиту Q.

Тип та розташування хоста, ролі користувачів та додаткові властивості задаються за допомогою відповідних класифікаторів.

При плануванні структури БД вузла РКІС та, відповідно, розрахунку значень функцій $F_a(Node, A)$ та $F_{tup}(Node, tup)$, що визначають необхідність представлення даних локально, приймає участь декілька факторів – доступність та швидкість отримання даних, незалежність від центрального вузла БД, розмір БД, ступінь достовірності даних, необхідність у подальшій синхронізації.

На першому кроці моделювання розпочинається з представлення на вузлі РКІС повної копії БД центрального вузла. При цьому доступність та незалежність від центрального вузла БД має максимальний рівень, швидкість отримання даних у порівнянні із центральним вузлом як правило нижча за рахунок менш потужних

обчислювальних ресурсів, але може бути збільшена за рахунок виконання операцій вибірки та проекції до відношень та зменшення кількості та обсягів блокувань даних. Розмір БД є великим, а отже даний критерій є неоптимальним. Також всі дані потребують синхронізації з центральним вузлом, що є досить ресурсоємною операцією.

Другий крок полягає у виключенні з віддаленого вузла всіх надлишкових даних. Для вирішення даної задачі на базі реляційної моделі обліку користувацьких запитів (2.4) створено багатовимірну базу даних (ББД). У запропонованій структурі багатовимірної БД у якості базового набору вимірів виступає наступна множина:

$$D = \{ДатаЧас,\ ТипХосту,\ РозташуванняХосту,\ РольКористувача,\ Застосунок,\ R,\ A,\ tup\} \quad (2.5)$$

Множина мір складається з пари значень «*Кількість*» та «*ЧасВиконання*»

$$M = \{Кількість,\ ЧасВиконання\} \quad (2.6)$$

Домени вимірів «*ТипХосту*», «*РольКористувача*» та «*Застосунок*» визначаються за допомогою відповідних класифікаторів. При виконанні аналізу щодо представленості даних для елементів вимірів вводиться характеристика «Маркер представленості даних», що відображає рівень необхідності представлення даних на вузлі РКІС. Для кожного елементу виміру за значення маркеру приймається одне з множини лінгвістичних значень {«*необхідно*», «*бажано*», «*не потрібно*»}, що визначає ступінь необхідності представлення даних того чи іншого хосту, ролі користувача або застосунку. Для виміру «*РозташуванняХосту*» маркування виконується автоматично значенням «*необхідно*» для відповідного віддаленого вузла, та «*не потрібно*» для всіх інших.

При визначенні значення маркеру представленості для рядку таблиці фактів використано функцію максимуму, що відображає принцип поглинання більш потужним значенням маркеру менш потужного. Визначення рівня маркеру при виконанні консолідації рядків таблиці фактів по значенням <*R, A, tup*>, тобто, для

комірки таблиці, може бути виконано декількома шляхами. Так, за песимістичними сценарієм, маємо алгоритм, аналогічний попередньому, тобто використовується функція агрегації максимум (у нашому випадку максимальне значення має маркер «*необхідно*», а відповідно мінімальне «*не потрібно*»). Даний підхід резервує дані, якщо вони необхідні хоча б в одному місці. За оптимістичним сценарієм, навпаки, дані маркуються як «*непотрібні*», якщо вони є такими хоча б за значенням одного виміру. Перший сценарій гарантовано забезпечує віддалений вузол необхідними даними, але веде до потенційної збитковості даних та розширює область даних, що потребуватимуть подальшої синхронізації. Другий веде до мінімізації обсягу даних на вузлі РКІС, але збільшує потребу у розподілених запитах, що у свою чергу веде до зниження рівня доступності та швидкість отримання даних, а також рівня незалежності від центрального вузла РКІС (центральної БД).

Застосування методу ковзного середнього вирішує це питання при умові урахування питомої ваги впливу маркеру кожного з вимірів. Крім того, слід врахувати, що для деяких підмножин вимірів має спрацьовувати саме песимістичний сценарій (дані потрібні, не зважаючи ні на що), а для деяких оптимістичний (дані не мають дублюватися в жодному разі). При чому, перший, як правило, поглинатиме останній.

Отже, маємо модель, у якій кожен атрибут виміру має значення, маркер та ваговий коефіцієнт: $A_{dim} = \{Val, Mrk, vol\}$, де $Mrk = \{$*«обов'язково», «необхідно», «бажано», «не потрібно», «заборонено»*$\}$, а vol – ваговий коефіцієнт (ігнорується для значень маркера «*обов'язково*» та «*заборонено*»).

Виконавши переведення нечислової лінгвістичної змінної маркерів у числове значення («*обов'язково*» – «2», «*необхідно*» – «1», «*бажано*» –«0», «*не потрібно*» – «–1», «*заборонено*» – «–2»), для них визначається відповідна функція агрегації:

$$Aggregate_{i=1}^{n} Mrk_i = \begin{cases} 2, \text{ якщо } \exists\ Mrk_i = 2 \\ -2, \text{якщо } \exists\ Mrk_i = -2 \wedge \nexists\ Mrk_i = 2 \\ \sum_{i=1}^{n} (Mrk_i * \frac{Vol_i}{\sum_{i=1}^{n} Vol_i}) \end{cases} \quad (2.7)$$

При прийнятті рішення з представленості даних на вузлі РКІС виконується консолідація рядків таблиці фактів за множиною <*R*, *A*, *tup*> та розраховується значення маркеру для кожного її елементу за формулою (2.7), після чого приймається рішення про представленість за наступним критерієм:

$$Предст(Node,R,A,tup) = (Aggregate(R, A, tup)_{i=1}^{n} Mrk_i > koef_{\text{предст}}^{node}) \quad (2.8)$$

де $koef_{\text{предст}}^{node}$ – пороговий коефіцієнт представленості даних у певному вузлі Node, область значень якого [–1, 1].

При встановленні коефіцієнту представленості даних (2.8) значення одиниці ($koef_{\text{предст}}^{node}) = 1$) всі дані, необхідні для роботи віддаленого вузла РБД розміщуються у локальній БД вузла РКІС [103]. Звідси, як і у випадку повного дублювання даних, маємо той же максимальний рівень доступності даних. Також, у порівнянні із повним дублюванням, не змінюється рівень достовірності та незалежності від центрального вузла БД. Але швидкість отримання даних збільшується за рахунок зменшення їх обсягу (передусім на операціях вибірки). Також розмір БД зменшується та зменшуються обсяги даних, що потенційно можуть потребувати подальшої синхронізації.

Отже, отримано покращення декількох показників при збереженні значень інших, що за Парето та Слейтором [86; 104] свідчить про вищій ступінь оптимальності БД вузла РКІС на другому кроці у порівнянні із першим кроком. Включення будь-якої частини надлишкових даних до складу БД віддаленого вузла призведе до зниження рівня хоча б одного з показників при збереженні рівня інших. Відповідно, всі можливі рішення по представленості даних, що знаходяться між першим та другим кроками, не входять до оптимальної множини за Парето.

2.3 Формалізація критеріїв оцінки якості структури БД вузла розподіленої та територіально розосередженої КІС

Третій крок полягає у повній відмові від локальної БД та розміщенні всіх даних на центральному вузлі (або в окремих випадках в інших вузлах) РКІС. У цьому випадку маємо максимізацію рівня оптимальності за показниками щодо необхідності подальшої синхронізації даних, оскільки дублювання даних відсутнє. Рівень достовірності також є максимальним, і розмір локальної БД вузла має мінімальне значення (локальна БД відсутня). Але, у той же час, максимально знижується доступність даних та швидкість їх отримання, а робота КІС є максимально залежною від каналів зв'язку та доступності центрального вузла.

Отже, на третьому кроці значення частини показників у порівнянні із другим кроком було покращено, але у той же час значення іншої частини показників погіршилося. Логічно припустити, що оптимальних значень показники якості структури вузла РКІС набувають на проміжку між другим та третім кроками. Для можливості виконання аналізу та знаходження оптимального розподілу даних між вузлами РКІС необхідно, передусім, формалізувати перераховані вище показники оптимальності структури БД.

Показники незалежності від центрального вузла БД, та, відповідно, доступності і швидкість їх отримання, напряму залежать від представленості даних користувацьких SQL-запитів на вузлі РКІС. Використавши модель запиту користувача (2.4) та результуючого відношення віддаленого вузла (2.1, 2.2), визначено функцію доступності даних запиту.

$$F_{\text{доступність}}(Node, Q) = \begin{cases} 1, \text{якщо } \forall R'' \exists R_{shema}^{remote}, R'' \in R_{shema}^{remote} \wedge \\ \quad \forall Q^{inner} F_{\text{доступність}}\left(Q^{inner}\right) = 1 \\ 0, \text{ якщо } \exists R'' \nexists R_{shema}^{remote}, R'' \in R_{shema}^{remote} \vee \\ \quad \exists Q^{inner} F_{\text{доступність}}\left(Q^{inner}\right) = 0 \end{cases} \quad (2.9)$$

Загальне значення рівня доступності даних та незалежності від центрального вузла БД визначається, як середнє значення рівня доступності (2.9) деякої підмножини користувацьких запитів:

$$F_{\text{доступність}} = \frac{\sum_{i=1}^{n} F_{\text{доступність}}(Q_{node})}{n},\ Q_{node} \in Q_{all} \tag{2.10}$$

Множина користувацьких запитів Q_{node} є підмножиною всіх користувацьких запитів Q_{all} ($Q_{node} \subset Q_{all}$), для кожного елементу якої функція приналежності до вузла РКІС дорівнює одиниці. Значення цієї функції визначається, як одиниця, якщо всі відношення, що входять до складу запиту та його вкладених підзапитів, мають значення маркеру представленості у цьому вузлі перевищує «-1». В іншому випадку значення функції дорівнює нулю: $Q_{node} = \{Q \mid F_{\text{приналежності}}(Node, Q) = 1\}$, де

$$F_{\text{приналежності}}(Node, Q) = \begin{cases} 1, \text{якщо } (\exists\, R'' \in R''_{set} \\ \rightarrow Aggregate(R'')_{i=1}^{n} Mrk_i > -1) \vee \\ (\exists\, Q^{inner} \in Q_{set}^{inner} \\ \rightarrow F_{\text{приналежності}}(Node, Q^{inner}) = 1) \\ \\ 0, \text{якщо } (\forall\, R'' \in R''_{set} \\ \rightarrow Aggregate(R'')_{i=1}^{n} Mrk_i \leq -1) \wedge \\ (\forall\, Q^{inner} \in Q_{set}^{inner} \\ \rightarrow F_{\text{приналежності}}(Node, Q^{inner}) = 0) \end{cases}$$

Слід зазначити, що до складу елементу Q множини Q_{all} окрім (2.4) також має бути включений порядковий номер. Дана операція виконується зважаючи на те, що деякі запити надходять від користувачів більше одного разу, і враховуючи властивість реляційної моделі, можуть бути згорнуті до одного рядку. У даному випадку ця особливість має бути врахована, та вага такого запиту при розрахунку загального значення критерію доступності даних віддаленого вузла РКІС збільшується у порівнянні із іншими запитами, відповідно до кількості надходжень, що і досягається за рахунок їх дублювання у множині Q_{all}.

Наступним розглянуто критерій розміру локальної БД, що впливає як на продуктивність запитів до локальної БД, так і на потужність обчислювальних ресурсів, необхідних для виконання операцій адміністрування БД та КІС (резервне копіювання, забезпечення відмовостійкості та ін.). БД під управлінням реляційної СКБД (у тому числі розподіленої) представлена на дисковому просторі у вигляді файлу або групи файлів [68; 70; 105]. У той же час будь-яка сучасна реляційна СКБД надає інформацію про обсяги, що займає окреме відношення на дисковому носії. В рамках виконання роботи експериментальним шляхом виявлено, що у переважній більшості випадків сумарне значення розміру відношень співпадає або майже співпадає із сумарним значенням розміру файлів БД:

$$\sum_{i=1}^{n} SizeR_i^{DBMS} - \sum_{j=1}^{m} SizeFile_j = \Delta, \qquad (2.11)$$

де $\frac{\Delta}{\sum_{i=1}^{n} \mathrm{SizeR_i^{DBMS}}} \leq 0{,}005$

Відповідно до (2.11) зроблено висновок про можливість використання розміру відношення при розрахунку значення критерію розміру локальної БД РКІС. Але інформація про розмір *R* не дає можливості визначити розмір *R''*, що є результатом послідовності операцій вибірки та проєкції, і входить до множини R^{remote}. З іншого боку, кожна СКБД надає інформацію про обсяг дискового простору, необхідного для збереження значення атрибуту, визначеного на певному домені [67; 69]. Звідки розмір кортежу визначено, як

$$Size_R = SizeR0_i^{DBMS} + p \; x \sum_{i=1}^{n} Size(Type_i), \qquad (2.12)$$

де $A_i \in D_i \in Type_i$, а p – кардинальне число або потужність відношення, а $SizeR0_i^{DBMS}$ – розмір *i*-го відношення за відсутності даних.

Однак, отримані за допомогою (2.12) значення не можуть бути використані в розрахунках без відповідних модифікацій, оскільки $Size_R$ практично ніколи не збігається з $SizeR^{dbms}$, а в деяких випадках відрізняється у декілька разів. Це пов’язано як із наявністю додаткових структур даних (індексів) відношення, так і

з особливостями побудови кластерних індексів, використання фактору заповнення, та інших властивостей представлення даних на дисковому носії тієї чи іншої СКБД.

Одночасне використання (2.11) та (2.12) дає можливість подолати вищенаведені невідповідності. Так, на першому кроці для кожного відношення визначено поправочний коефіцієнт $Koef_{sizeR} = \frac{SizeR_i^{DBMS} - SizeR0_i^{DBMS}}{p \times \sum_{i=1}^{n} Size(Type_i)}$.

Далі, при визначенні розміру відношення R'' (що є підмножиною R) використовуємо наступну формулу:

$$Size_{R''} = Koef_{sizeR} \times (SizeR0_i^{DBMS} + p' \times \sum_{i=1}^{n'} Size(Type_i)), \quad (2.13)$$

де p – кардинальне число R'', n' – кількість елементів множини R_{shema}^{remote} (кількість атрибутів), і кожен атрибут $A_i \in D_i \in Type_i$.

Сума розмірів (2.13) всіх підмножин R'' дає розмір БД віддаленого вузла. Але для кожного окремого випадку предметної області даний критерій буде відрізнятись, а отже, його абсолютне значення не має цінності. Тому остаточне значення критерію розміру локальної БД вузла представлено у вигляді відношення до загального розміру БД центрального вузла КІС.

$$F_{size} = \frac{\sum_{i=1}^{n} Size_{R'',i}}{\sum_{i=1}^{n} SizeR_i^{DBMS}} \quad (2.14)$$

Останнім із критеріїв оптимальності структури БД вузла РКІС є необхідність у подальшій синхронізації [103]. Спочатку визначається підмножина даних вузла, для якої виконуються операції зміни даних. Для цього визначимо модель SQL-запиту модифікації даних відношення: $Q^{modif} = \{Виміри, R''^{modif}, type\}$, де R''^{modif} – підмножина на відношенні R, що зазнає змін внаслідок виконання операцій модифікації даних, $type = \{«insert», «update», «delete»\}$ – тип операції. R''^{modif} визначаємо, як:

$$R''^{modif} = \{tup[P^{modif}] \mid tup[P^{modif}] \in R[P^{modif}]_{data} \wedge F(tup, S) = істина\}, \quad (2.15)$$

де *S* – деяка логічна умова, *F(tup, S)* – функція, що відображає її виконання для відповідного кортежу, а P^{modif} – множина атрибутів, що зазнає модифікації.

Розглядаючи множину запитів до БД, результуюча підмножина R''^{modif}_{node} базового відношення *R* визначається як об'єднання підмножин R''^{modif} всіх запитів (2.15), що надійшли до БД з віддаленого вузла: $R''^{modif}_{node} = \bigcup_{i=1}^{n} R''^{\text{modif}}_{i}$:

$$R''^{modif}_{node} = \{tup[P^{modif}_{node}] \mid tup[P^{modif}_{node}] \in R[P^{modif}_{node}]_{data} \wedge F(tup, S_{node}) = истина\}, \quad (2.16)$$

де $tup[P^{modif}_{node}] = \bigcup_{i=1}^{n} tup[P_i]$, а $S_{node} = \bigvee_{i=1}^{n} S_i$.

Аналогічним чином визначено множину R''^{modif}_{main}, що зазнає модифікацій на центральному вузлі або інших вузлах із подальшою синхронізацією із центральним. Перетин множин R''^{modif}_{node} та R''^{modif}_{main} визначає підмножину базового відношення, на якій можуть виникати конфлікти зміни та значення атрибутів якої потребують застосування більш ресурсоємних алгоритмів синхронізації [94; 106; 107].

$$R'''^{modif}_{node} = R''^{modif}_{node} \cap R''^{modif}_{main} \quad (2.17)$$

Ґрунтуючись на (2.16, 2.17), до багатовимірної моделі (2.5, 2.6) додається вимір *SyncroFlg = {true, false}*, що визначатиметься на кортежі <*R, A, tup*>. Далі, на підставі агрегатного значення маркеру представленості даних $Aggregate^{n}_{i=1} Mrk_i$ та коефіцієнту представленості $koef^{node}_{\text{предст}}$ виконується фільтрація даних багатовимірної БД обліку SQL-запитів згідно з рішенням про представленість (2.8) та *SyncroFlg = true*, із подальшим згортанням за <*R, A, tup*> та підрахунком кількості запитів. Співвідношення отриманого значення до загальної кількості запитів згідно з (2.8) і є показником рівня необхідності синхронізації даних

$$F_{synchro} = \frac{p^{modif}_{node}}{p_{node}}, \quad (2.18)$$

де p_{node}^{modif} – кардинальність відношення, що включає запити віддаленого вузла (згідно з рішенням про представленість), до яких входять значення атрибутів кортежів, що також входять до множини R''^{modif}_{node}, а p_{node} – кардинальність всіх запитів, атрибути та кортежі яких представлені на вузлі РКІС.

2.4 Підвищення ступеня актуальності даних та динамічне визначення часу асинхронної реплікації

При синхронізації даних вузла РКІС використовуються в основному підходи, пов’язані із асинхронною реплікацією, але у деяких випадках (при високій вірогідності виникнення конфліктів) синхронна реплікація може бути виправдана.

Вибір підходу щодо реплікації даних напряму залежить від віддаленого або розподіленого типу запиту. У випадку віддаленого запита, коли взаємодія з іншими вузлами необхідна лише для забезпечення синхронізації, використано режим асинхронної реплікації. Подібне рішення обґрунтовано можливістю виключити із транзакції всі «зайві» вузли, залишивши лише вузол, на якому знаходяться дані (рис. 2.3).

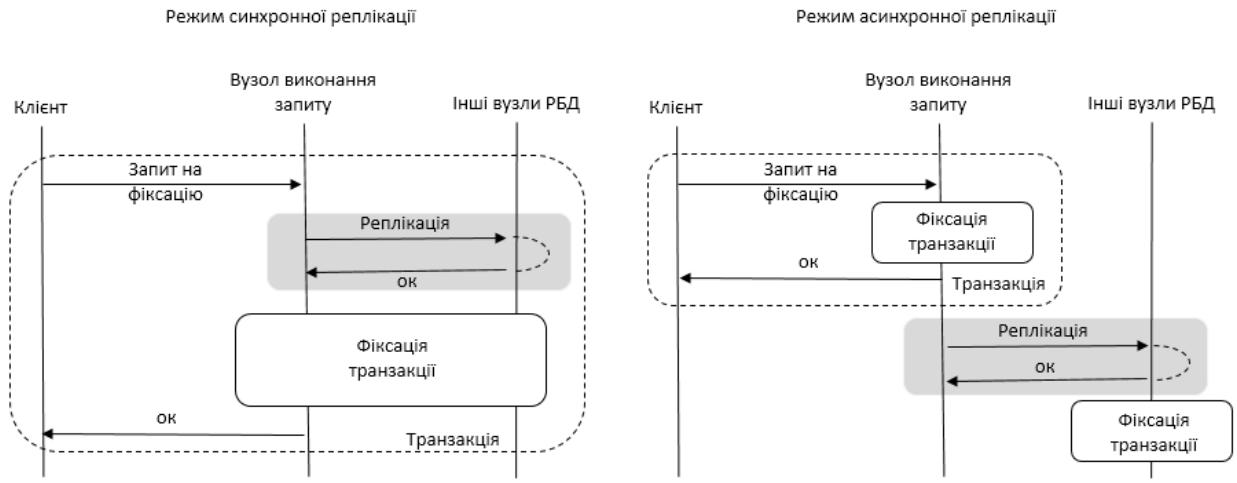


Рисунок 2.3 – Порівняння режимів реплікації у випадку віддаленого запита

При синхронізації даних у випадку розподіленого запиту взаємодія із вузлами не може бути винесена за межі транзакції. Це обумовлено тим фактом,

що до розподіленої транзакції у будь якому випадку входять фрагменти даних, які зберігаються на інших вузлах та не представлені на вузлі виконання запиту. Зважаючи на це, не вбачається за доцільне винесення процесу реплікації за межі транзакції, оскільки зменшення часу виконання транзакції є незначним у порівнянні із можливими суперечностями в даних, пов'язаними із несинхронністю оновлення (рис. 2.4).

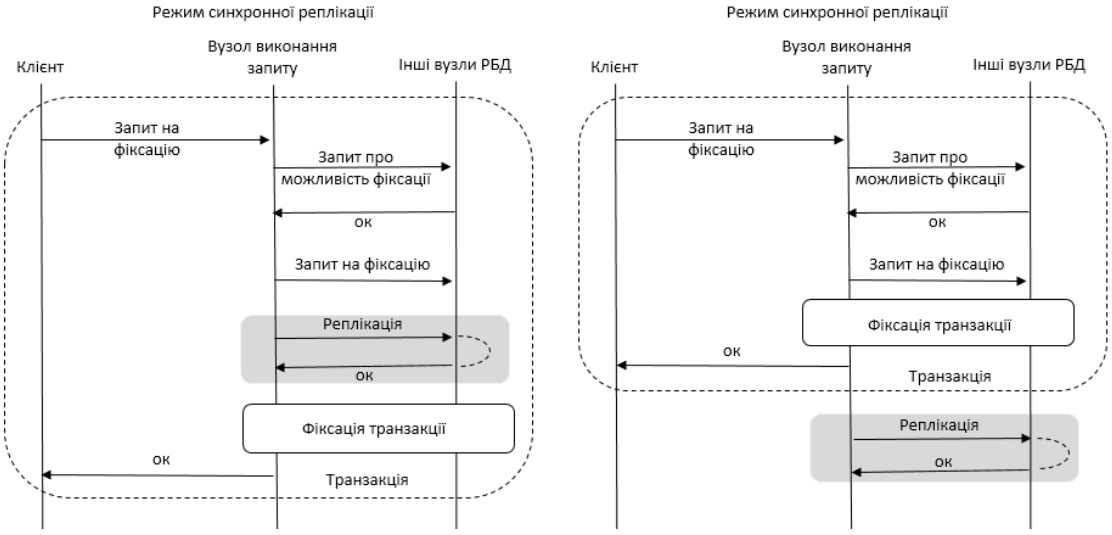


Рисунок 2.4 – Порівняння режимів реплікації у випадку розподіленого запита

У контексті реплікації даних між центральним та віддаленим вузлами, виділено дві множини даних, що потребують синхронізації. Перша – це дані, що змінюються на центральному вузлі РКІС, та представлені на віддаленому. Дана множина може бути представлена, як:

$$R_{set}{}''^{modif}_{main} = \{ R''^{modif}_{main} \mid (\exists A \mid A \in R''^{modif}_{main} \wedge A \in R''_{remote}) \wedge$$
$$\wedge ((\exists tup \mid tup \in R''^{modif}_{main} \wedge tup \in R''_{remote}) \} \quad (2.19)$$

Друга множина – це дані, що змінюються в БД віддаленого вузла РКІС:

$$R_{set}{}''^{\text{modif}}_{remote} = \{ R''^{\text{modif}}_{\text{remote}} \mid (\exists A \mid A \in R''^{\text{modif}}_{\text{remote}} \wedge A \in R''_{\text{main}}) \wedge$$
$$\wedge ((\exists tup \mid tup \in R''^{\text{modif}}_{\text{remote}} \wedge tup \in R''_{\text{main}}) \} \quad (2.20)$$

На базі (2.19) та (2.20) визначено три категорії даних для синхронізації – дані, що змінюються тільки на центральному вузлі, дані, що змінюються тільки на віддаленому вузлі, та дані, що зазнають змін на обох вузлах. Дане розділення виконано зважаючи на необхідність застосування різних механізмів при реплікації даних, що входять до цих множин.

$$R_{set}{}''^{modif}_{main-only} = R_{set}{}''^{modif}_{main} - R_{set}{}''^{modif}_{remote} \quad (2.21)$$

$$R_{set}{}''^{modif}_{remote-only} = R_{set}{}''^{modif}_{remote} - R_{set}{}''^{modif}_{main} \quad (2.22)$$

$$R_{set}{}''^{modif}_{main\&remote} = R_{set}{}''^{modif}_{remote} \cap R_{set}{}''^{modif}_{main} \quad (2.23)$$

При реплікації використовується асинхронна періодична реплікація із підтримкою реплікації за вимогою із використанням журналу транзакцій. Тобто виконується часткова синхронізація фрагменту даних, що зазнали реальних змін. Крім того, враховуються не тільки факт виникнення події зміни, а і кількісно-якісні його характеристики. Так, враховується рівень потреби у даних на замовнику, що визначається шляхом підрахунку кількості звернень до даних у межах користувацьких SQL-запитів. Також передбачені механізми виконання повної синхронізації даних у випадку виникнення збоїв синхронізації, порушення цілісності даних або інших позаштатних ситуацій.

Внаслідок того, що кожен віддалений вузол працює частково (або повністю) незалежно від центрального і один від одного, підтримуються так звані «точки актуальності даних» для кожного вузла у вигляді значення дати та часу останньої синхронізації даних кожної з трьох множин (2.21–2.23).

Завантаження даних множини, що змінюється тільки на віддаленому вузлі, є, як правило, менш складним завданням. Період синхронізації може бути більш довгим, оскільки дані не потрібні в режимі он-лайн, та можуть бути оброблені центральним вузлом із деякою затримкою у часі без суттєвих втрат при ухваленні управлінських рішень. Синхронізація проводиться зазвичай автоматично або користувачем КІС по завершенню робочої доби.

Обмін даними, що змінюються тільки на центральному вузлі, проводиться періодично із підтримкою режиму синхронізації за вимогою. Головна особливість запропонованого підходу полягає у динамічній зміні періоду обміну даними [51]. Виконуючи синхронізацію із чітко визначеною періодичністю, не враховується особливість нерівномірності змін даних у часі. Тому у деяких випадках період має бути зменшений для підтримки сталого рівня актуальності даних, а в інших навпаки збільшено, виходячи з міркувань зменшення навантаження на сервер СКБД та ETL-систему.

Отже, наявність механізмів обчислення моменту наступної синхронізації даних може з одного боку значно підвищити актуальність даних, а з іншого істотно розвантажити ресурси серверів БД (рис. 2.5). При їх реалізації враховані наступні фактори:

– кількість змін у журналі транзакцій;

– якість цих змін (на базі частоти використання даних у SQL-запитах визначаються коефіцієнти впливу на якість роботи вузла РКІС);

– значення точки актуальності даних (момент попередньої синхронізації по кожній з множин 2.21–2.23);

– завантаженість серверу БД поточними запитами.

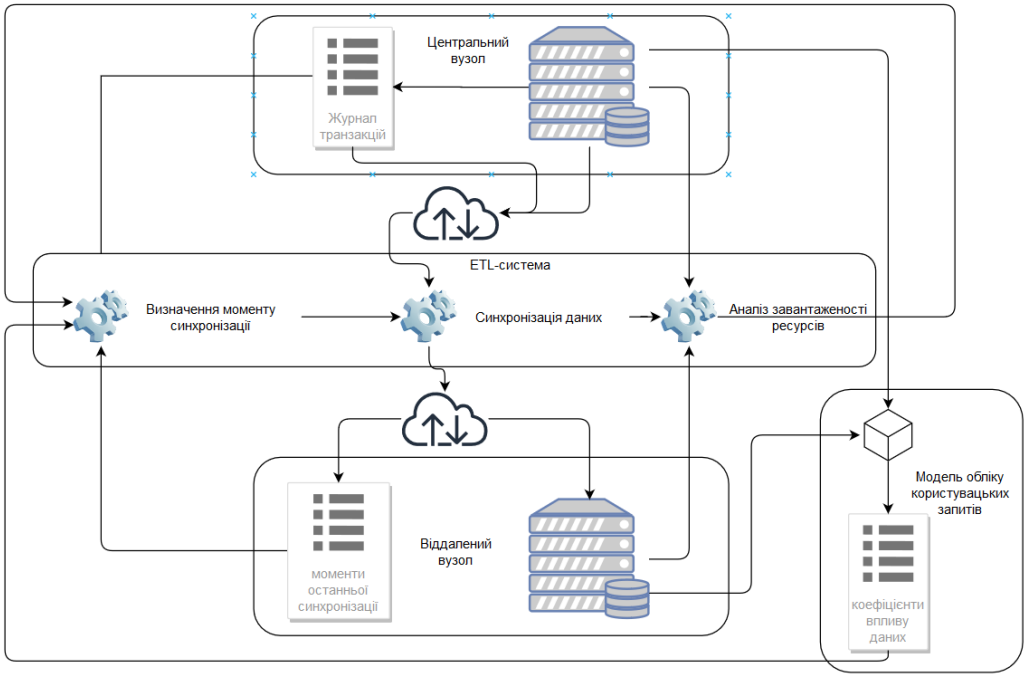


Рисунок 2.5 – Схема алгоритму реплікації даних

При визначенні моменту наступної синхронізації обчислюється значення рівня актуальності даних, далі $К_{акт}$. Якщо $К_{акт}$ перевищує гранично припустиме значення, далі $К_{гран}$, то відбувається синхронізація.

$$F_{синхронізації} = \begin{cases} 1, \text{якщо } К_{акт} - К_{гран} > 0 \\ 0, \text{якщо } К_{акт} - К_{гран} \leq 0 \end{cases}.$$

Рівень актуальності даних залежить від кількості змінених даних та від ступеня впливу цих даних на загальний рівень актуальності, і може бути обчислений наступним чином:

$$К_{акт} = \sum_{i=1}^{m}(k_{R(i)}{}^{x} \times (\sum_{j=1}^{n} k_{A(j)} \times \sum_{q=1}^{p}(modif_{(j,q)} \times k_{tup(q)}))),$$

де $К_{R(i)}$ – ваговий коефіцієнт ступеню впливу *i*-го відношення;

$К_{A(j)}$ – ваговий коефіцієнт ступеню впливу *j*-го атрибуту,

$К_{tup(q)}$ – ваговий коефіцієнт ступеню впливу q-го кортежу,

$modif_{(j,q)}$ – флаг, що вказує на факт зміни значення атрибуту кортежу *i*-го відношення {0,1};

$К_{R(i)} = \frac{p_{node}^{R(i)}}{p_{node}}$, де $p_{node}^{R(i)}$ – кардинальність відношення, що включає запити віддаленого вузла (згідно з рішенням про представленість), до яких входить відношення R_i;

p_{node} – кардинальність вибірки всіх запитів, атрибути та кортежі яких представлені на віддаленому вузлі;

$К_{A(j)} = \frac{p_{node}^{R(i),A(j)}}{p_{node}^{R(i)}}$, де $p_{node}^{R(i),A(j)}$ – кардинальність вибірки, що включає запити віддаленого вузла, до яких входить атрибут A_j відношення R_i;

$К_{tup(q)} = \frac{p_{node}^{R(i),tup(q)}}{p_{node}^{R(i)}}$, де $p_{node}^{R(i),A(j)}$ – кардинальність вибірки, що включає запити вузла РКІС, до яких входить кортеж tup_q відношення R_i;

$Modif_{(j,\,q)}$ – встановлюється програмно при зміні даних у БД.

Гранично припустиме значення залежить від дати останнього оновлення даних та від поточного завантаження сервера, та обчислюється за формулою

$$k_{\text{гран}} = f(\Delta t) \times (\frac{k_{\text{період}}}{\Delta t} + l_{\text{завант}}),$$

де $k_{\text{період}}$ – коефіцієнт періодичності оновлення даних (є елементом налаштування підсистеми);

Δt – інтервал, протягом якого дані не оновлювалися;

$l_{завант}$ – рівень завантаженості сервера (обчислюється за допомогою спеціальних моніторингових програм (наприклад, SQL Profiler або Performance Monitor);

$f(\Delta t)$ – порогова функція необхідності оновлення даних (приймає значення «0», якщо значення ΔT перевищує гранично допустимий інтервал, інакше дорівнює «1»).

Отже, обчислення наступного моменту реплікації даних відбувається, базуючись на значенні наступної функції [51]:

$$f(t) = \begin{cases} 1, \sum_{i=1}^{m}(k_{R(i)} * (\sum_{j=1}^{n} k_{A(j)} * \sum_{q=1}^{p}(modif_{(j,q)} * k_{tup(q)}))) - \\ f(\Delta t) \times \left(\frac{k_{\text{період}}}{\Delta t} + l_{\text{завант}}\right) > 0 \\ 0, \sum_{i=1}^{m}(k_{R(i)} * (\sum_{j=1}^{n} k_{A(j)} * \sum_{q=1}^{p}(modif_{(j,q)} * k_{tup(q)}))) - \\ f(\Delta t) \times \left(\frac{k_{\text{період}}}{\Delta t} + l_{\text{завант}}\right) \leq 0 \end{cases} \quad (2.24)$$

Запропонована модель, що враховує результат відстеження змін даних із визначенням суттєвості цих змін, дозволяє динамічно визначати момент наступної синхронізації даних, що у свою чергу дає можливість підвищити актуальність даних, не перевантажуючи сервер БД.

Висновки до розділу 2

На базі реляційної моделі даних та елементів теорії множин формалізовано поняття БД вузла РКІС у вигляді підмножини даних на множині відношень БД. Із використанням визначення операцій вибірки та зрізу, а також ієрархічної структури користувацьких запитів, побудована модель, що описує їх структуру. Дана модель, окрім відображення їх ієрархічної структури та підмножин даних і їх схем, включає аналітичні дані про приналежність типу автоматизованого

робочого місця, типу користувача (ролі), розташування та додаткових аналітичних вимірів.

Отримана модель користувацького запиту, що включає ієрархічний набір результуючих відношень, кожне з яких у свою чергу є підмножиною базового відношення схеми БД, дозволяє проаналізувати деякий їх набір з точки зору наявності спільних властивостей. Результати дозволяють виділити для кожного базового відношення деяку підмножину (відношення вузла РКІС), що складається із елементів, які входять до набору результуючих відношень послідовності SQL-запитів до БД КІС, що аналізуються.

Для виконання операцій консолідації та фільтрації множини користувацьких запитів при підрахунку значень необхідних показників запропоновано використати ББД. Елементи вимірів маркуються відповідно до необхідності представленості даних на віддаленому вузлі РКІС. Для визначення узагальненого значення маркеру представленості для відповідних даних введено специфічну функцію агрегації.

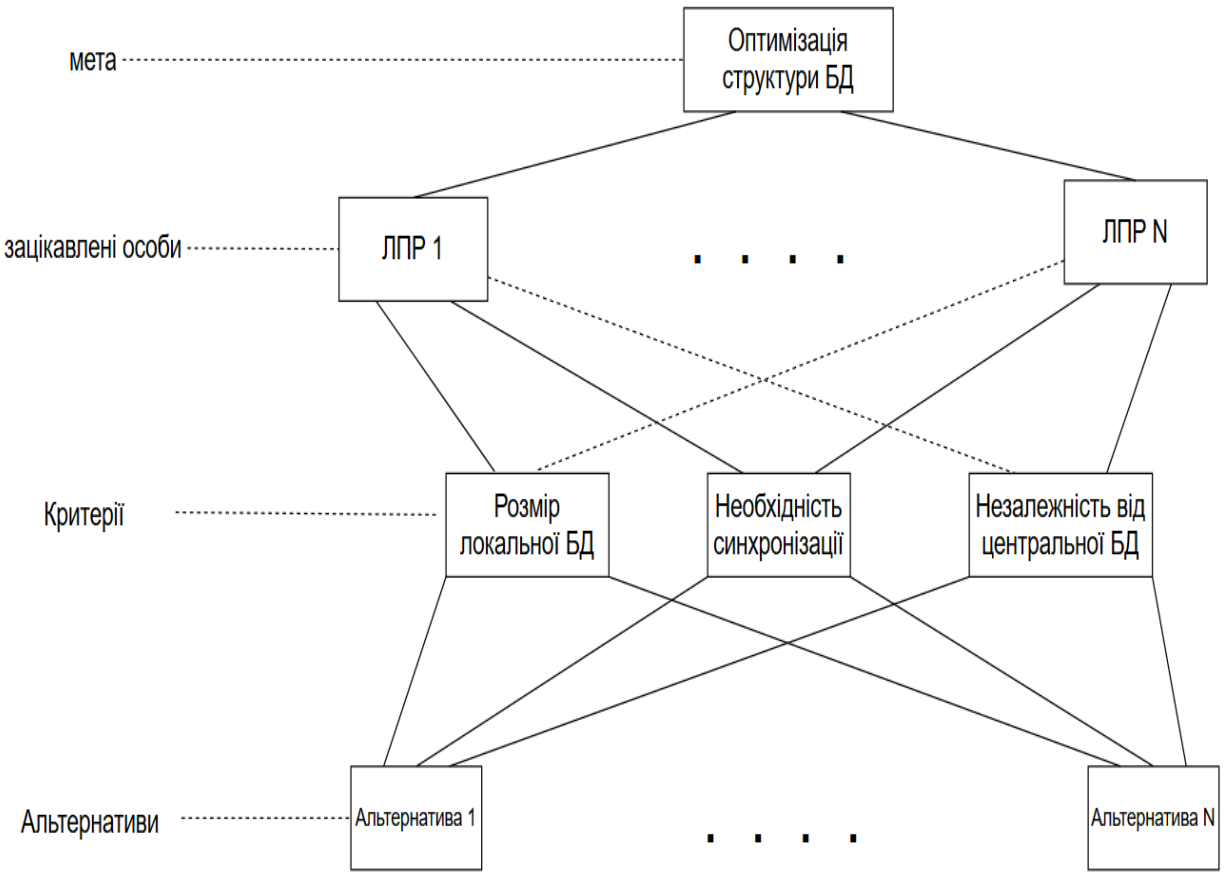
Для визначення оптимального значення рівня маркеру представленості введено критерії оптимальності структури БД та побудовано їх математичні моделі, що дозволяють розрахувати їх значення в залежності від рівня представленості даних у вузлі РКІС. Отже, отримано багатокритеріальну задачу, що має бути вирішена для визначення оптимального рівня представленості даних.

Окремо представлено модель динамічного визначення моменту асинхронної реплікації даних, що дозволяє підвищити ступінь актуальності даних без створення додаткового навантаження на сервер БД та ETL-системи синхронізації даних.

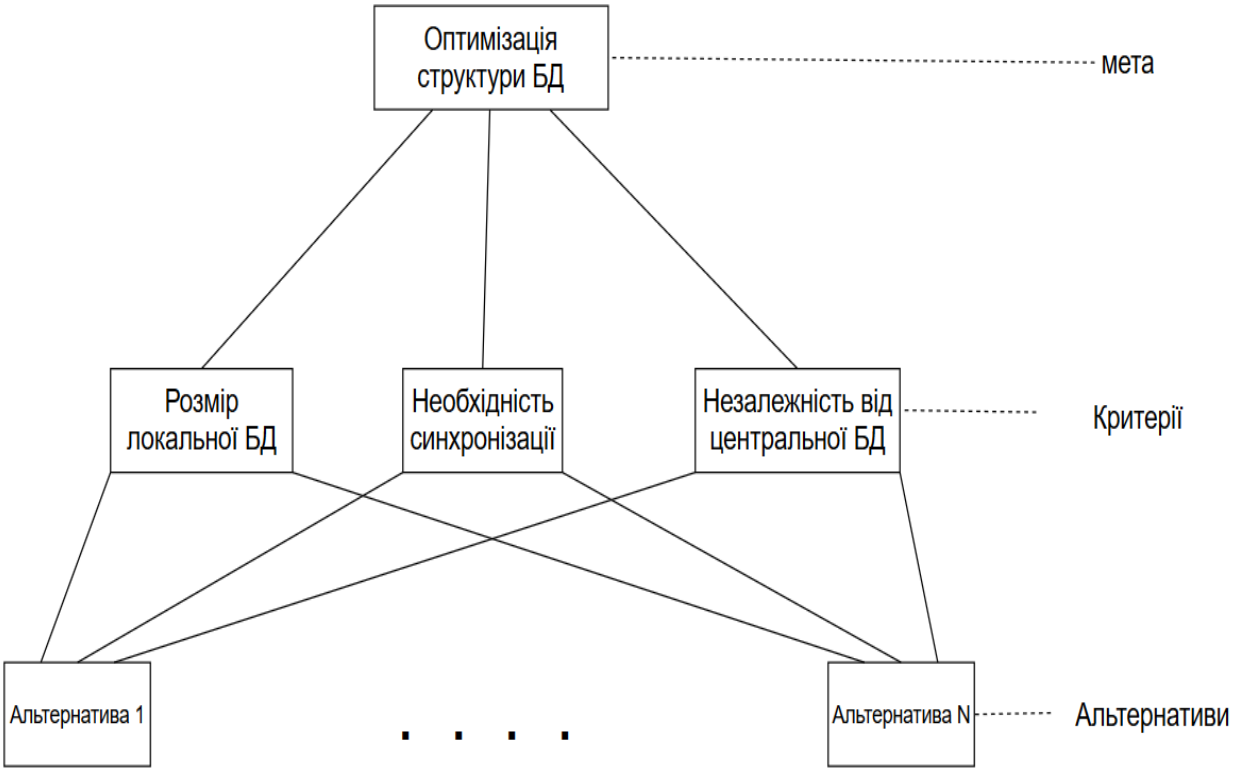
РОЗДІЛ 3
МЕТОДИ ВИБОРУ КРАЩОЇ АЛЬТЕРНАТИВИ ДЛЯ МАРКЕРУ ПРЕДСТАВЛЕНОСТІ ТА КЛАСИФІКАЦІЇ НОВИХ ДАНИХ

3.1 Побудова ієрархічної моделі MAI та заповнення матриць попарних порівнянь різних рівнів ієрархії

При складанні ієрархії використано відношення між елементами сусідніх рівнів: «ціль» – «зацікавлені особи» – «критерії» – «альтернативи». При відсутності можливості проведення аналізу із залученням декількох ЛПР, кількість рівнів ієрархії зменшується до трьох: «ціль» – «критерії» – «альтернативи» (рис. 3.1).



а)



б)

Рисунок 3.1 – Загальне представлення ієрархії моделі для чотирьох (а) та трьох (б) рівнів відповідно

Використання MAI у розглянутому випадку ускладняється тим фактом, що рівень маркеру представленості даних (що виступає у якості можливих альтернатив) є дійсним числом на інтервалі [-1, 1]. Оскільки виконати обчислення значень критеріїв оптимальності можливо відповідно до (2.10), (2.14) та (2.18) із використанням звернень до даних багатовимірної БД SQL-запитів, задача оптимізації може бути вирішена за допомогою підстановки значень та повного перебору (табулювання маркеру представленості із певним кроком). Даний підхід дає велику кількість альтернатив на четвертому рівні ієрархії навіть при збільшенні кроку табулювання, що ускладнює вирішення поставленої задачі.

Тому, на наступному етапі, із метою спрощення задачі та зменшення кількості альтернатив рівня маркеру представленості даних до п’яти інтервалів, запропоновано табулювання із кроком 0,5: «низький» (Н) – «–1», «нижче середнього» (НС) – «–0,5», «середній» (С) – «0», «вище середнього» (ВС) – «0,5»,

та «високий» (В) – «1». Рівень «зацікавлені особи» представлено елементами «Власник», «Адміністратор БД», «Розробник БД» та «Оператор КІС». Отримана ієрархічна модель наводиться на рис. 3.2.

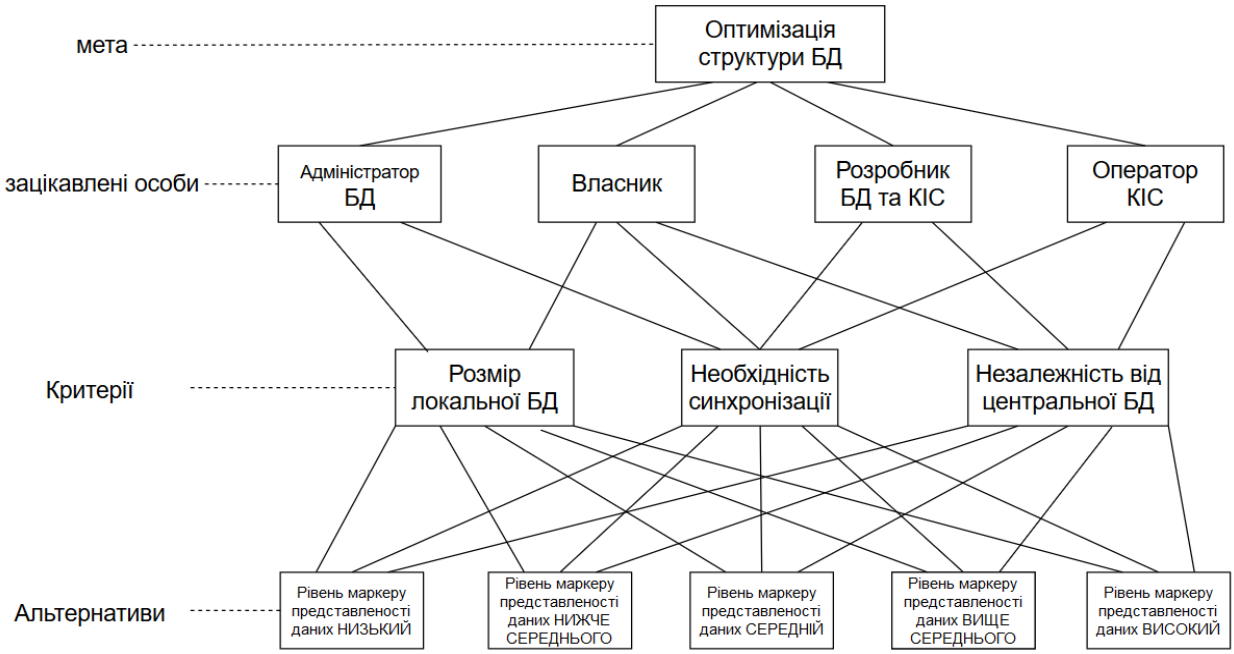


Рисунок 3.2 – Ієрархічна модель задачі оптимізації структури вузла РКІС після введення альтернатив на базі інтервалів значень маркеру представленості даних

Оскільки не завжди власник об’єкту автоматизації має безпосереднє відношення до впровадження та оптимізації процесів, пов’язаних з використанням інформаційних технологій (ІТ), під зацікавленою особою «Власник» тут і надалі мається на увазі топ-менеджер організації (корпорації), що виступає у ролі відповідальної особи за ІТ-сферу. Тобто у якості «Власника» може виступати як генеральний директор чи власник бізнесу, так і керівник ІТ- департаменту чи проєкт-менеджер.

Перелік критеріїв оптимальності моделі на рівні ієрархії «критерії» відрізняється для різних «зацікавлених осіб». Так, для зацікавленої особи «Власник» об’єкту автоматизації важливі всі три критерія («Розмір БД», «Необхідність синхронізації» та «Незалежність від центральної БД»). Цей факт пояснюється тим, що від рівня наведених критеріїв залежить як якість роботи КІС на віддаленому вузлі, так і вартість обладнання, що необхідна для забезпечення

роботи програмного забезпечення (ПЗ). У той же час для адміністратора БД важливими є критерії розміру БД та необхідності організації синхронізації даних. Рівень незалежності від центрального вузла (та інших вузлів) не є важливим у цьому випадку, оскільки ніяким чином не впливає на роботу зацікавленої особи. У свою чергу, для розробника БД та оператора КІС критерій розміру БД не є критичним, оскільки вони не мають відношення до забезпечення безперебійної роботи серверів та СКБД. Також зрозуміло, що відносна вага кожного з критеріїв для різних зацікавлених осіб відрізняється.

Використовуючи шкалу відносної важливості критеріїв [80] (табл. 3.1) та із залученням людини, що приймає рішення (ЛПР), якою на даному етапі виступає зацікавлена особа «Власник» об'єкту автоматизації, виконується побудова матриці попарних порівнянь для зацікавлених осіб.

Таблиця 3.1 – Шкала відносної важливості критеріїв

α_j	Відносна вага критерію або альтернативи
1	Рівна можливість порівнюваних критеріїв (альтернатив)
3	Помірна (слабка) перевага одного над іншим
5	Сильна (істотна) перевага
7	Очевидна перевага
9	Абсолютна перевага
2, 4, 6, 8	Проміжні значення

Так, наприклад, для випадку очевидної переваги «Власника» над «Оператором КІС», сильної переваги над «розробником БД» та помірної переваги над «Адміністратором БД»; слабкої переваги «Адміністратора БД» над «розробником БД» та помірної переваги над «Оператором КІС»; а також помірної переваги «Розробника БД» над «Оператором КІС», отримано матрицю переваг наступного вигляду (табл. 3.2).

Таблиця 3.2 – Матриця переваг зацікавлених осіб

	Власник	Адміністратор БД	Розробник БД та КІС	Оператор КІС
Власник	1	3	5	7
Адміністратор БД	1/3	1	3	5
Розробник БД та КІС	1/5	1/3	1	3
Оператор КІС	1/7	1/5	1/3	1

На третьому рівні ієрархії складаються відповідні матриці попарних порівнянь за критеріями оптимальності по кожній зацікавленій особі. Тут у ролі ЛПР виступає зацікавлена особа, за якою заповнюється матриця. Так, для зацікавленої особи «Власник», у випадку очевидної переваги критерію «Незалежність» над критерієм «Розмір БД» та сильної переваги над «необхідністю синхронізації», а також помірної переваги «Необхідності синхронізації» над «Розміром БД», отримано наступну матрицю переваг критеріїв оптимальності (табл. 3.3).

Таблиця 3.3 – Матриця переваг критеріїв оптимальності для зацікавленої особи «Власник»

	Розмір БД	Незалежність	Необхідність синхронізації
Розмір БД	1	1/7	1/3
Незалежність	7	1	5
Необхідність синхронізації	3	1/5	1

Виконавши розрахунок середньої ваги за формулою Баклі [108], що базується на побудові середнього геометричного:

$$W_j = \frac{r_j}{\sum_{i=1}^{n} r_i}, \text{ де } r_j = \left(\prod_{i=1}^{m} \alpha_{i,j}\right)^{\frac{1}{n}}, \quad (3.1)$$

отримуємо для матриці переваг ЛПР «Власник» наступний вектор відносної ваги критеріїв оптимальності моделі:

$$W_{\text{власник}}^{\text{крит}} = \begin{bmatrix} 0{,}08 \\ 0{,}73 \\ 0{,}19 \end{bmatrix}. \quad (3.2)$$

Для перевірки відсутності конфліктів між елементами матриці, виконується розрахунок індексу узгодженості (ІУ). Для даних табл. 3.3 ІУ = 3,2 %, що свідчить про допустимий рівень узгодженості (у випадку перевищення значення 10 % виникає необхідність корегування значень таблиці через повторне опитування ЛПР).

Наступним кроком у класичному методі аналізу ієрархій є заповнення матриць попарних порівнянь альтернатив окремо по кожному критерію оптимальності [88; 91], аналогічно табл. 3.2 та 3.3. У нашому випадку, наявність математичних моделей розрахунку значень критеріїв оптимальності, сформульованих у (2.10), (2.14) та (2.18), дозволяє виконати розрахунок та початкову ініціалізацію даних матриць на базі числових значень маркеру представленості даних для кожної альтернативи. Далі заповнена матриця подається на розгляд ЛПР для затвердження. У ролі ЛПР, так само, як і для попереднього випадку із критеріями оптимальності, виступає зацікавлена особа, за якою заповнюється матриця. Так, у табл. 3.4, наведено приклад зміни критерію «Розмір БД» вузла РКІС в залежності від однієї з п’яти альтернатив для однієї з предметних областей впровадження результатів роботи.

Таблиця 3.4 – Залежність значення розміру БД від обраної альтернативи

Рівень маркеру представленості даних	Низький	Нижче середнього	Середній	Вище середнього	Високий
Розмір локальної БД	0,02	0,24	0,47	0,55	0,75

Виходячи із наведених даних, розмір БД при низькому (min) та високому (max) рівні маркеру представленості даних відрізняється у $\frac{0{,}75}{0{,}02} = 37{,}5$ разів. Керуючись принципами парних порівнянь та аксіомою гомогенності [109], яка говорить про те, що на кожному рівні ієрархії порівнювані елементи не мають

сильно (згідно з табл. 3.1 не більше ніж у 9 разів) відрізнятися один від одного, виконується нормування значень, наведених у табл. 3.4, із використанням дещо модифікованої формули природньої нормалізації:

$$W_i^{norm} = \frac{(W_i - min_i W_i)}{(max_i W_i - min_i W_i)} * (k-1) + 1\,, \qquad (3.3)$$

де W_i – значення критерію оптимальності для i-ї альтернативи, а $k = 9$, згідно з табл. 3.1.

Пронормовані згідно з (3.3) значення розміру локальної БД (табл. 3.4) представлені у табл. 3.5.

Таблиця 3.5 – Пронормовані значення розміру БД в залежності від обраної альтернативи

Рівень маркеру представленості даних	Низький	Нижче середнього	Середній	Вище середнього	Високий
Пронормоване значення розміру локальної БД	1,00	3,41	5,93	6,81	9,00

Виконавши округлення до цілого за математичними правилами, для відповідності значень комірок матриці попарних порівнянь значенням табл. 3.1, будується матриця попарних порівнянь альтернатив за критерієм «Розмір локальної БД» для зацікавленої особи «Власник» (табл. 3.6).

Таблиця 3.6 – Матриця переваг альтернатив по критерію «розмір БД» для зацікавленої особи «Власник»

Розмір локальної БД	Низький	Нижче середнього	Середній	Вище середнього	Високий
Низький	1	3	6	7	9
Нижче середнього	1/3	1	2	2	3
Середній	1/6	1/2	1	1	2
Вище середнього	1/7	1/2	1	1	1
Високий	1/9	1/3	1/2	1	1

Оскільки заповнення матриці переваг виконано на базі розрахункових даних без участі ЛПР, розрахунок індексу узгодженості не має сенсу.

Далі, у табл. 3.7 наведено значення критеріїв «Необхідність синхронізації даних» та «Рівень незалежності» для кожної з п’яти альтернатив, отримані із багатовимірної БД користувацьких запитів відповідно до (2.10) та (2.18) для однієї з предметних областей впровадження результатів роботи.

Таблиця 3.7 – Залежність значення критеріїв «Рівень незалежності» та «Необхідність синхронізації» від обраної альтернативи

Рівень маркеру представленості даних	Низький	Нижче середнього	Середній	Вище середнього	Високий
Незалежність	0,35	0,71	0,92	0,96	0,97
Необхідність синхронізації	0,15	0,18	0,13	0,10	0,07

Відповідно до (3.3) виконується нормування та складається матриця попарних порівнянь альтернатив для критеріїв незалежності (табл. 3.8) та необхідності синхронізації (табл. 3.9). При розрахунку коефіцієнтів для альтернатив по критерію «Незалежність», враховується максимізація критерію як мета, і, відповідно до (3.3), формула набуває наступного вигляду:

$$W_i^{norm} = \frac{(W_i^{\sim} - min_i W_i^{\sim})}{(max_i W_i^{\sim} - min_i W_i^{\sim})} \times (k-1) + 1, \text{ де } W_i^{\sim} = (1 - W_i)$$

Таблиця 3.8 – Матриця переваг альтернатив по критерію «незалежності» від центрального вузла БД

Незалежність	Низький	Нижче середнього	Середній	Вище середнього	Високий
Низький	1	1/2	1/5	1/9	1/9
Нижче середнього	2	1	1/2	1/4	1/4
Середній	5	2	1	1/2	1/2
Вище середнього	9	4	2	1	1
Високий	9	4	2	1	1

Таблиця 3.9 – Матриця переваг альтернатив по критерію «необхідності синхронізації»

Необхідність синхронізації	Низький	Нижче середнього	Середній	Вище середнього	Високий
Низький	1	1	1	1/2	1/7
Нижче середнього	1	1	1/2	1/3	1/9
Середній	1	2	1	1/2	1/5
Вище середнього	2	3	2	1	1/3
Високий	7	9	5	3	1

3.2 Обмеження області значень критеріїв оптимальності та розрахунок вектору глобальних пріоритетів альтернатив

Відповідно до вимог ЛПР, у якості яких виступають елементи рівня ієрархії «зацікавлені особи» (рис. 3.2), виконується обмеження області значень критеріїв оптимальності рівня маркеру представленості даних на вузлі РКІС. У випадку розбіжності поглядів різних зацікавлених осіб, рішення залишається за пріоритетним експертом відповідно до вектору пріоритетів зацікавлених осіб, отриманого на базі матриці попарних порівнянь, наведеної у табл. 3.2.

Так, за критерієм оптимальності «Незалежність БД» для однієї з предметних областей впровадження результатів роботи введено обмеження мінімального значення на рівні 0,4, відповідно до якого щонайменше 40 % запитів до БД вузла РКІС має оброблятись локально, без звернення до інших вузлів із використанням каналів зв’язку. Розрахунок значень критерію оптимальності «Незалежність БД» відповідно до рівня маркеру представленості даних, виконаний на базі (2.10), результати якого наведено у першому рядку табл. 3.7. Це дозволяє визначити невідповідність альтернативи для значення маркеру представленості даних «Низький» введеному обмеженню для критерію оптимальності «Незалежність БД». Відповідно до цього, зазначена альтернатива виключається.

Обмеження за критерієм «розмір БД» для предметної області впровадження результатів роботи, було сформульовано на рівні максимально допустимого

значення 0,65, що говорить про максимальний розмір БД на вузлі РКІС на рівні 65 % від загального обсягу даних (після виключання дублікатів) РБД КІС. Згідно з розрахованими відповідно до (2.14) значеннями критерію оптимальності «Розмір БД» для кожної з п’яти альтернатив, що були наведені у табл. 3.4, визначається невідповідність введеному обмеженню альтернативи для значення маркеру представленості даних «Високий», для якого розмір БД складає 75 % від загального обсягу даних.

За критерієм «необхідність синхронізації» вводиться обмеження максимального значення на рівні 0,5, що відповідає не більше ніж 50 % запитів на зміну даних на вузлі РКІС, дані яких зазнають зміни на інших вузлах, від загальної кількості запитів на зміну даних на вузлі РКІС. Розраховані згідно з (2.18) значення критерію для кожної альтернативи, що наведені у другому рядку табл. 3.7, свідчать про відповідність всіх альтернатив маркеру представленості даних введеному обмеженню.

Після виключення альтернатив рішення, що не відповідають введеним обмеженням на область значень критеріїв оптимальності, початкова ієрархічна модель, наведена на рис. 3.2, зазнає відповідних змін, після виконання яких має наступного вигляду (рис. 3.3).

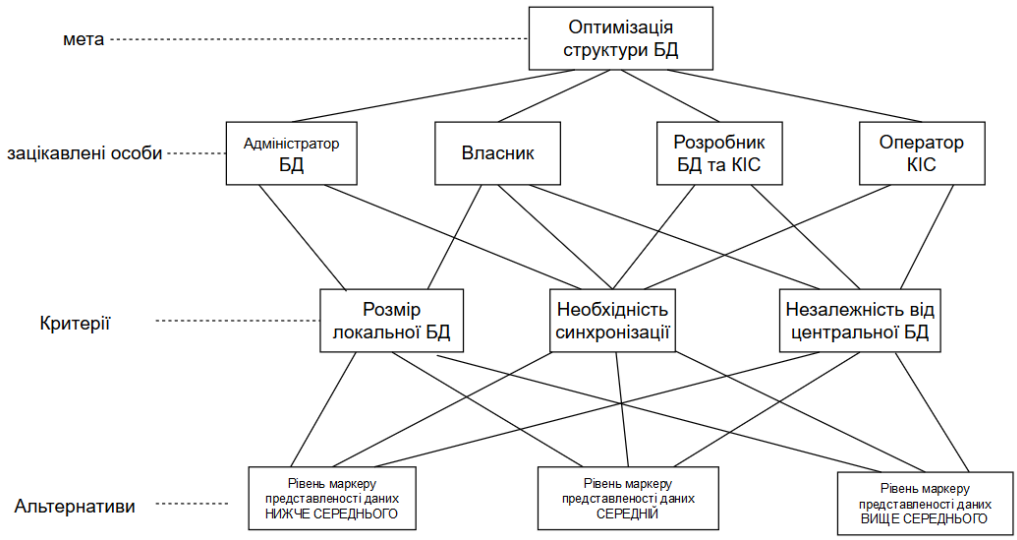


Рисунок 3.3 – Ієрархічна модель задачі оптимізації структури вузла РКІС після обмеження множини альтернатив

Матриці попарних порівнянь альтернатив, наведені у табл. 3.6, 3.8 та 3.9, також зазнають змін через виключення першої та останньої колонок, що відповідають альтернативам для значень маркеру представленості даних «Низький» та «Високий». На рис. 3.4 у якості прикладу розглянуто, як зазначені зміни вплинули матрицю попарних порівнянь за критерієм оптимальності «Розмір БД» для зацікавленої особи «Власник».

Розмір локальної БД	Низький	**Нижче середнього**	**Середній**	**Вище середнього**	Високий
Низький	1	3	6	7	9
Нижче середнього	1/3	**1**	**2**	**2**	3
Середній	1/6	**1/2**	**1**	**1**	2
Вище середнього	1/7	**1/2**	**1**	**1**	1
Високий	1/9	1/3	1/2	1	1

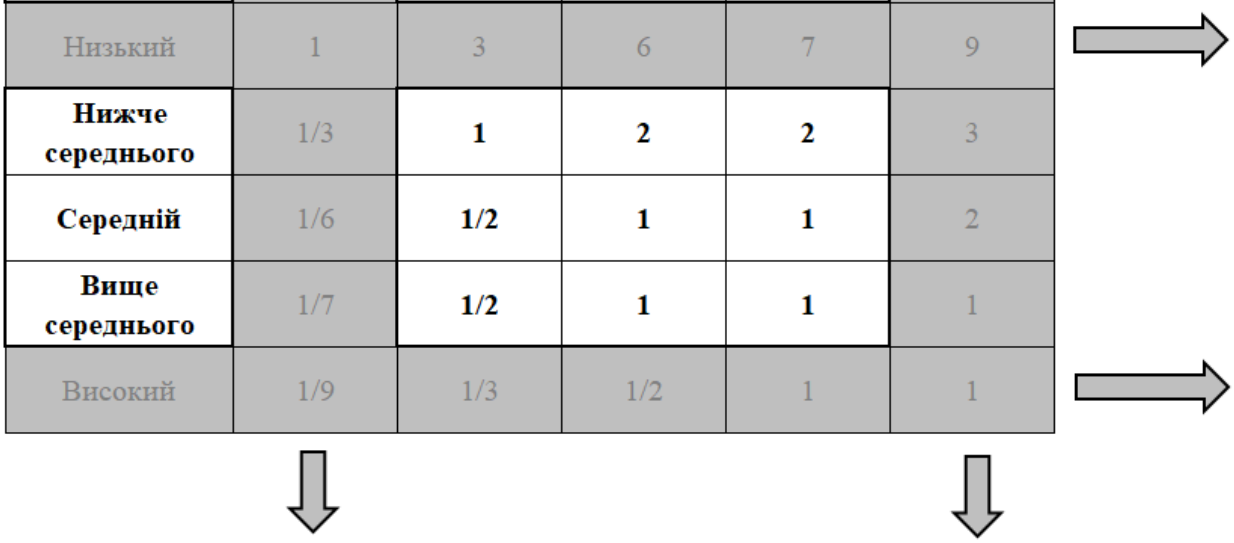


Рисунок 3.4 – Виключення альтернатив по матриці попарних порівнянь за критерієм оптимальності «Розмір БД» для зацікавленої особи «власник»

Отримані після зазначених змін матриці попарних порівнянь альтернатив за критеріями оптимальності для зацікавленої особи «Власник» наведено у табл. 3.10–3.12.

Таблиця 3.10 – Матриця переваг альтернатив по критерію «розмір БД» для зацікавленої особи «Власник»

Розмір локальної БД	Нижче середнього	Середній	Вище середнього
Нижче середнього	1	2	2
Середній	1/2	1	1
Вище середнього	1/2	1	1

Таблиця 3.11 – Матриця переваг альтернатив по критерію «незалежності» від центрального вузла БД

Незалежність	Нижче середнього	Середній	Вище середнього
Нижче середнього	1	1/2	1/4
Середній	2	1	1/2
Вище середнього	4	2	1

Таблиця 3.12 – Матриця переваг альтернатив по критерію «Необхідність синхронізації»

Необхідність синхронізації	Нижче середнього	Середній	Вище середнього
Нижче середнього	1	1/2	1/3
Середній	2	1	1/2
Вище середнього	3	2	1

Аналогічно до розрахунку матриці відносної ваги критеріїв оптимальності моделі (табл. 3.3), згідно з (3.1) виконується розрахунок матриці відносної ваги альтернатив за критерієм «Розмір БД» для зацікавленої особи «Власник». Отриманий результат для даних табл. 3.10 має наступний вигляд:

$$W_{\text{власник}}^{\text{розмБД}} = \begin{bmatrix} 0{,}500 \\ 0{,}250 \\ 0{,}250 \end{bmatrix} \tag{3.4}$$

Згідно (3.1) виконується розрахунок матриці відносної ваги альтернатив за критеріями «Незалежність» та «Необхідність синхронізації». Отриманий результат для даних табл. 3.11 та 3.12 представлений у (3.5) та (3.6) відповідно.

$$W_{\text{власник}}^{\text{незалежн}} = \begin{bmatrix} 0{,}260 \\ 0{,}327 \\ 0{,}413 \end{bmatrix} \tag{3.5}$$

$$W_{\text{власник}}^{\text{розмБД}} = \begin{bmatrix} 0{,}078 \\ 0{,}435 \\ 0{,}487 \end{bmatrix} \tag{3.6}$$

Глобальні пріоритети для альтернатив визначаються за формулою

$$W_i = W_1^{\text{крит}} \times W_i^{\text{розмБД}} + W_2^{\text{крит}} \times W_i^{\text{незалежн}} + W_3^{\text{крит}} \times W_i^{\text{синхр}} \qquad (3.7)$$

Отже, виходячи із (3.2), (3.4), (3.5) та (3.6), і відповідно до (3.7) отримано значення глобальних пріоритетів альтернатив, які наведено у табл. 3.13 та на (3.8).

Таблиця 3.13 – Розрахунок глобальних пріоритетів альтернатив для зацікавленої особи «Власник».

Альтернативи (рівень маркеру представленості)	Критерії			Глобальні пріоритети
	Розмір БД	Незалежність	Необхідність синхронізації	
	0,08	0,73	0,19	
Нижче середнього	0,500	0,260	0,078	0,245
Середній	0,250	0,327	0,435	0,341
Вище середнього	0,250	0,413	0,487	0,414

$$W_{\text{власник}}^{\text{глоб}} = \begin{bmatrix} 0{,}245 \\ 0{,}341 \\ 0{,}414 \end{bmatrix} \qquad (3.8)$$

Аналогічним чином складено матриці попарних порівнянь критеріїв оптимальності для інших зацікавлених осіб (табл. 3.14–3.16), та відповідно до отриманих за (2.10), (2.14) та (2.18) значень та відповідно до (3.3) виконано розрахунки векторів глобальних пріоритетів за вказаними особами (3.9–3.11). При складанні матриць враховується несуттєвість окремих критеріїв оптимальності для деяких зацікавлених осіб.

Таблиця 3.14 – Матриця переваг критеріїв оптимальності для зацікавленої особи «Адміністратор БД»

	Розмір БД	Необхідність синхронізації
Розмір БД	1	2
Необхідність синхронізації	1/2	1

Таблиця 3.15 – Матриця переваг критеріїв оптимальності для зацікавленої особи «Розробник БД»

	Незалежність	Необхідність синхронізації
Незалежність	1	7
Необхідність синхронізації	1/7	1

Таблиця 3.16 – Матриця переваг критеріїв оптимальності для зацікавленої особи «Оператор КІС»

	Незалежність	Необхідність синхронізації
Незалежність	1	3
Необхідність синхронізації	1/3	1

$$W^{\text{глоб}}_{\text{АдмінБД}} = \begin{bmatrix} 0{,}359 \\ 0{,}311 \\ 0{,}330 \end{bmatrix} \tag{3.9}$$

$$W^{\text{глоб}}_{\text{РозробникБД}} = \begin{bmatrix} 0{,}237 \\ 0{,}341 \\ 0{,}422 \end{bmatrix} \tag{3.10}$$

$$W^{\text{глоб}}_{\text{Оператор КІС}} = \begin{bmatrix} 0{,}214 \\ 0{,}355 \\ 0{,}431 \end{bmatrix} \tag{3.11}$$

Використавши отримані результати векторів глобальних пріоритетів за зацікавленими особами (3.9–3.11) та матрицю переваг зацікавлених осіб (табл. 3.2), аналогічно табл. 3.13 та згідно з (3.7), розраховано вектор глобальних пріоритетів альтернатив (табл. 3.17).

Таблиця 3.17. Розрахунок глобальних пріоритетів альтернатив

	Власник	Оператор КІС	Адмін. БД	Розробник БД	Глобальні пріоритети
	0,564	0,055	0,263	0,118	
Нижче середнього	0,245	0,214	0,359	0,237	0,273
Середній	0,341	0,355	0,311	0,341	0,334
Вище середнього	0,414	0,431	0,330	0,422	0,393

Відповідно до отриманого вектору глобальних пріоритетів альтернатив рівня маркеру представленості даних, найбільш привабливою виглядає альтернатива «вище середнього» із значенням 0,393, якому відповідає значення маркеру представленості даних, що дорівнює 0,50.

3.3 Використання елементів нечіткого логічного висновку для підвищення точності результату

При виконанні розбиття множини альтернатив значення маркеру представленості даних (що визначається на множині дійсних чисел в інтервалі [–1;1]) на більшу кількість інтервалів є можливість підвищити точність отриманого рішення. Так, розглянуто використання методу, описаного у підрозділах 3.1 та 3.2, та отримані результати для 5 та 21 інтервалу значень маркеру представленості даних. З метою спрощення не вводилось обмежень за критеріями оптимальності та використано трирівневу ієрархічну модель без рівня «зацікавлена особа». У якості матриці попарних порівнянь критеріїв оптимальності взято таку, що заповнювалась зацікавленою особою «Власник» (табл. 3.3). Також прийнято початкове заповнення матриць переваг альтернатив за критеріями оптимальності відповідно до математичних моделей (2.10), (2.14) та (2.18) як остаточне.

Враховуючи введені спрощення моделі, розрахунок вектору глобальних пріоритетів для п’яти альтернатив («низький» (Н) – «–1», «нижче середнього» (НС) – «–0,5», «середній» (С) – «0», «вище середнього» (ВС) – «0,5», та «високий» (В) – «1») дає наступний результат із оптимальною альтернативою «високий», якій відповідає рівень маркеру представленості даних, що дорівнює «1».

$$W_5^{\text{глоб}} = \begin{bmatrix} 0{,}09 \\ 0{,}09 \\ 0{,}15 \\ 0{,}30 \\ 0{,}37 \end{bmatrix} \tag{3.12}$$

Виконавши розбиття інтервалу дійсних чисел на 21 інтервал із кроком 0,1, отримано 21 альтернативу із наступною множиною значень маркеру представленості даних: *A = {–1,0; –0,9; –0,8; –0,7; –0,6; –0,5; –0,4; –0,3; –0,2; –0,1; 0; 0,1; 0,2; 0,3; 0,4; 0,5; 0,6; 0,7; 0,8; 0,9; 1,0}*. Одержані альтернативи позначено, як *A1, A2, A3, …, A21*. Виконавши розрахунки значень критеріїв оптимальності, згідно з (2.10), (2.14) та (2.18), нормування отриманих значень відповідно до (3.3), заповнення матриць попарних порівнянь та розрахунок вектору глобальних пріоритетів відповідно до (3.1) та (3.7), отримано відповідне значення вектору глобальних пріоритетів альтернатив, що наводиться у табл. 3.18.

Таблиця 3.18 – Вектор глобальних пріоритетів альтернатив для 21 інтервалу значень маркеру представленості даних

Альтернатива	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11
Маркер представленості	-1,0	-0,9	-0,8	-0,7	-0,6	-0,5	-0,4	-0,3	-0,2	-0,1	0,0
Елемент вектору пріоритетів	0,008	0,012	0,009	0,013	0,01	0,012	0,009	0,01	0,013	0,021	0,034
Альтернатива	A12	A13	A14	A15	A16	A17	**A18**	A19	A20	A21	–
Маркер представленості	0,1	0,2	0,3	0,4	0,5	0,6	**0,7**	0,8	0,9	1,0	–
Елемент вектору пріоритетів	0,05	0,054	0,063	0,076	0,082	0,101	**0,117**	0,111	0,103	0,093	–

У даному випадку зменшення кроку та, відповідно, підвищення точності виявляє альтернативу *A18*, як найкращу, якій відповідає значення маркеру представленості даних = 0,7. Однак використання даного підходу веде до необхідності подальшого корегування ЛПР матриці попарних порівнянь розміром 21×21, тобто вимагає виконати 210 операцій попарних порівнянь альтернатив по кожному критерію оптимальності кожною зацікавленою особою.

Крім того, ЛПР дуже важко визначати пріоритети альтернатив при незначній зміні їх якісних характеристик.

Виходячи із наведених складнощів реалізації та необхідності підвищення точності методу, прийнято рішення використати елементи апарату нечіткого логічного висновку [110–112], що дозволяє отримати точність результату, наближену до випадку використання 21 альтернативи, при використанні лише 5 інтервалів, як це було реалізовано у підрозділах 3.1 та 3.2.

В теорії множин елемент або належить множині, або ні. Нечітка множина визначається за допомогою функції приналежності, яка відповідає поняттю характеристична функція в класичній логіці. Кілька нечітких множин можна визначити через одну змінну, яка набуває декількох значень. Функція приналежності може приймати будь-яку форму, але найчастіше для її подання використовуються кусочно-лінійні лінії. Кусочно-лінійні функції приналежності зазвичай використовуються оскільки: вони характеризуються простотою; вони містять точки, що дозволяють задати області, де поняття є істинним, а де хибним, що спрощує опис системи [110; 113].

На рис. 3.5 представлено обрані у підрозділі 3.1 інтервали значення рівня маркеру представленості даних («низький», «нижче середнього», «середній», «вище середнього», «високий») у вигляді набору нечітких множин однієї змінної із кусочно-лінійними функціями приналежності лінійної форми.

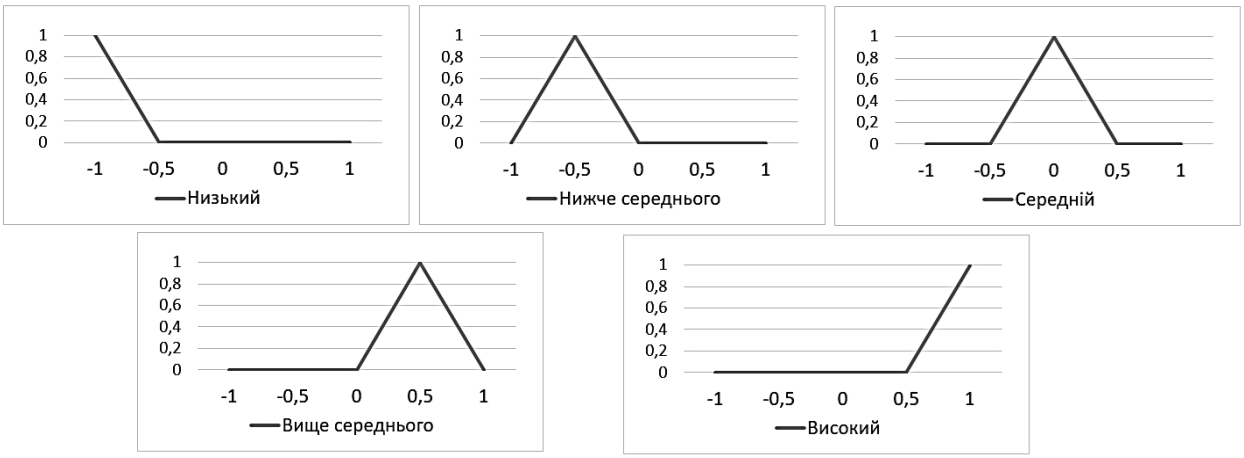


Рисунок 3.5 – Набор нечітких множин для опису рівня маркеру представленості даних

Класичний процес нечіткого логічного висновку складається з етапів фазифікації (приведення чіткої числової величини до нечіткої форми), безпосередньо нечіткого логічного висновку, як правило на базі набору правил, та дефазифікація (числове вираження нечіткого результату) [114].

У нашому випадку, у підрозділі 3.2 виконано вирішення задачі багатокритеріального аналізу із використанням методу аналізу ієрархій та отримано вектор глобальних пріоритетів альтернатив наступного вигляду:

$$W^{\text{глоб}} = \begin{bmatrix} 0{,}000 \\ 0{,}273 \\ 0{,}334 \\ 0{,}393 \\ 0{,}000 \end{bmatrix} \tag{3.12}$$

Далі, знайдений вектор глобальних пріоритетів подається, як вектор нечітких чисел для маркеру представленості даних. Тобто для отримання числового значення оптимального рівня маркеру представлення даних необхідно виконати об'єднання результатів та дефазифікацію. Дефазифікацією називаємо процес приведення нечіткого числа до його дійсної інтерпретації. В теорії нечітких множин дефазифікація аналогічна знаходженню характеристик положення випадкових величин (математичного очікування, моди, медіани) в теорії ймовірності. Серед способів дефазифікації є вибір чіткого числа з максимальним ступенем приналежності, або методи, що ґрунтуються на ідеї пошуку точки згущення (центру мас) області, розташованої між графіком функції приналежності і віссю абсцис. Це робиться з метою «усереднити» можливі прийняті нечітким числом значення, з урахуванням відповідних їм ступенів приналежності.

Серед найбільш розповсюджених методів – метод центра мас $a_r = \frac{\int_{min}^{max} u\mu_\varepsilon(u)du}{\int_{min}^{max} \mu_\varepsilon(u)du}$ та метод медіани $a_r : \int_{min}^{a^r} \mu_\varepsilon(u)du = \int_{a^r}^{max} \mu_\varepsilon(u)du.$

На рис. 3.6 наведено графічне представлення вектору нечітких чисел, отриманого для спрощеного варіанту задачі вираженого матрицею глобальних пріоритетів (3.12) для маркеру представленості даних.

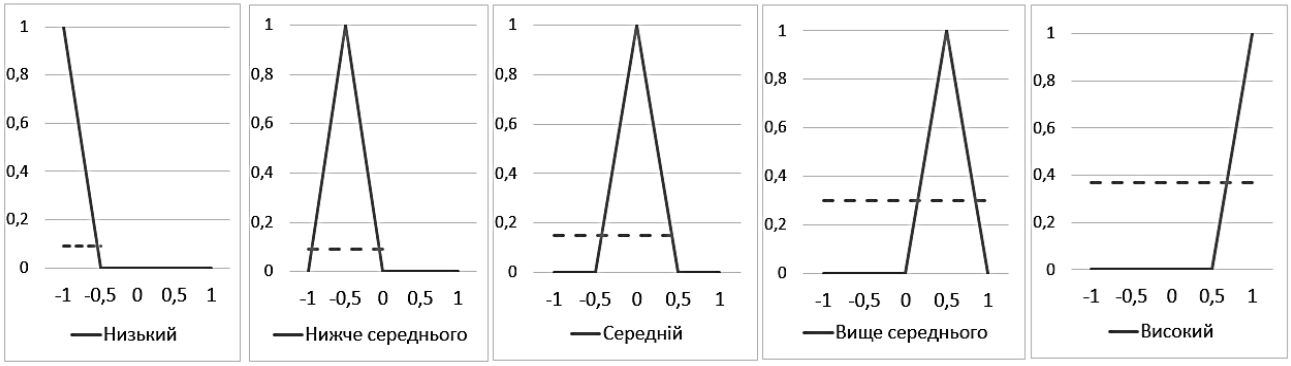


Рисунок 3.6 – Вектор нечітких чисел глобальних пріоритетів альтернатив

Після виконання операцій агрегації та дефазифікації отриманих результатів, отримано значення маркеру представлення даних на вузлі РКІС на рівні 0,7 (рис. 3.7).

Рисунок 3.7 –Етап агрегації та дефазифікації

Дане рішення співпадає із отриманим для 21 інтервалу значень маркеру представленості даних (табл. 3.18), із чого зроблено висновок про можливість використання елементів нечіткого логічного висновку, а саме представлення вектору глобальних пріоритетів альтернатив у вигляді вектору нечітких чисел із подальшою агрегацією та дефазифікацією результату, для підвищення точності рішення при невеликій кількості інтервалів значення маркеру представленості даних.

На рис. 3.7 наведено результат, отриманий згідно з даними вектору глобальних пріоритетів альтернатив підрозділу 3.2, за яким оптимальне значення маркеру представленості даних вузла РКІС для предметної області впровадження результатів роботи отримано на рівні 0,2.

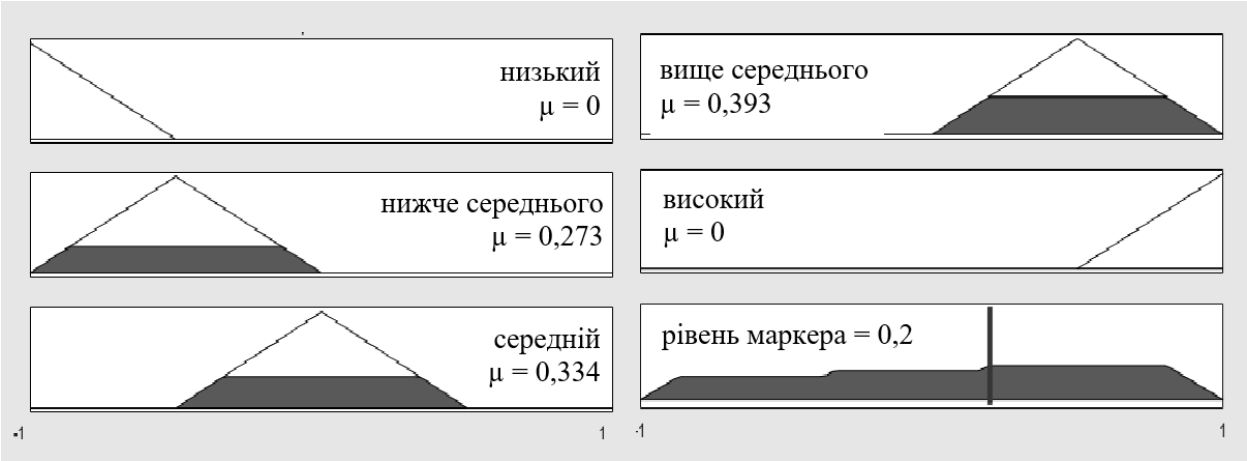


Рисунок 3.7 – Агрегація та дефазифікація отриманих результатів

Отже, використання методу аналізу ієрархій на базі обмеженої множини альтернатив дозволяє побудувати матриці переваг та виконати необхідні розрахунки для отримання значень глобальних переваг альтернатив, при чому у деяких випадках із залученням ЛПР, а у деяких – використовуючи математичні моделі критеріїв оптимальності, представлені у розділі 2. Використання елементів нечіткої логіки, а саме дефазифікації вектору нечітких чисел для маркеру представленості даних (вектору глобальних пріоритетів), дозволяє підвищити точність результату при визначенні оптимального значення рівня маркеру представленості даних.

3.4 Задача класифікації при визначенні маркеру представленості нових даних

У процесі експлуатації описаної технології виникає задача, пов’язана із додаванням до таблиць нових кортежів. Рівень необхідності представлення даних визначено відповідно до накопиченої статистики SQL-запитів та діапазону первинних ключів, що потрапляють під умову вибірки. Діапазон первинного

ключа доданих до БД кортежів не був доступний при виконанні аналізу, відповідно не був промаркований згідно зі статистикою запитів до БД. Для цих кортежів маємо відсутності значення рівня необхідності представленості даних у вузлі РКІС. Найбільш простим варіантом розв'язання задачі є реплікація до віддаленого вузла РКІС всіх нових даних (оскільки немає даних про ступінь їх корисності для вузла), тобто встановлення для них значення необхідності представлення на рівні «обов'язково». Даний підхід є не до кінця виправданим, оскільки, по-перше, не враховує впливу даних (доданих та особливо змінених) рядків на інші рядки через їх перетин у межах одного запиту (ПЗ, або типу хоста). Крім того, коли обсяг цих даних досягає критичної точки, виходячи, наприклад, із міркувань навантаження при операціях вибірки або синхронізації даних, настає момент, коли необхідно заново виконати ініціалізацію ББД SQL-запитів, та оновлення бази використання атрибутів та кортежів таблиць БД вузлом РКІС [103; 107; 115].

Виконання повного аналізу використовуваності атрибутів та кортежів таблиць БД є ресурсоємною операцією та не може виконуватись часто. Тому вищенаведений підхід є неприйнятним для великих та часто змінюваних БД. Наряду із цим, вторинний та подальші моніторинги активності користувачів БД ускладняється тим фактом, що БД віддаленого вузла вже використовується і запити до неї також мають бути враховані.

Враховуючи той факт, що головний вплив на рівень представленості при аналізі користувацьких запитів до БД відіграє у більшості випадків комбінація значень атрибутів кортежу відношення, запропоновано представити задачу із визначенням рівня маркеру представленості нових даних у вигляді задачі класифікації. Вхідними параметрами виступають назва таблиці (відношення) та перелік значень атрибутів кортежу, а результатом – рішення про корисність кортежу для вузла РКІС.

Серед множини алгоритмів вирішення задачі класифікації засобами машинного навчання [34; 116–120] виділено найбільш популярні підходи, серед яких лінійна та логічна регресія, дискрименантний аналіз, дерева рішень, наївний

Байєсівський алгоритм, *k*-найближчих сусідів та застосування різного виду нейронних мереж [121]. Лінійна та логічна регресія є одними з найбільш відомих методів, але виходячи із нечислової характеристики більшості вхідних змінних (атрибутів відношення) не є оптимальними у даному випадку. З тих же причин, а саме через складність визначення відстані, не розглядаємо k-найближчих сусідів.

Використання нейронних мереж є на даний момент найбільш популярним напрямком при вирішенні задачі класифікації [122-124]. Однак, по-перше, не всі таблиці містять достатню кількість рядків для здійснення якісного навчання (недостатній обсяг даних для навчання та тестування результатів). Крім того, кожна таблиця (відношення) має різну кількість атрибутів, кожен з яких визначається на своєму домені. Виходячи із цього, задачу класифікації нових даних для кожного відношення необхідно вирішувати, використовуючи окремо навчену нейронну мережу. Відповідно до вищенаведеного, використання нейронних мереж також не розглядається.

Для розв'язання задачі було використано Байєсівський наївний алгоритм [125], що є простим, але ефективним, та дозволяє швидко визначити ймовірність входження об'єкту до того чи іншого класу. Алгоритм, як відомо, базується на припущенні, що кожна вхідна змінна незалежна, що часто не відповідає дійсності. Але промислові реляційні БД у більшості випадків перебувають у 3-й нормальній формі [19; 126–127], що свідчить про відсутність транзитивних залежностей у межах відношення [2; 28; 46; 53]. Даний факт дозволяє стверджувати, що переважна більшість вхідних змінних відповідає базовому припущенню алгоритму.

Основу алгоритму складає теорема Байєса, що дозволяє розрахувати ймовірність приналежності об'єкту до того чи іншого класу [128]. Для поточної задачі, знайдено ймовірності, що кортеж X потрібен на віддаленому вузлі, відповідно до значення i-го атрибута x_i:

$$P_x(\text{потрібен} \mid x_i) = \frac{P(x_i \mid \text{потрібен}) * P(\text{потрібен})}{P(x_i)}, \qquad (3.13)$$

де $P(\text{потрібен})$ – загальна ймовірність за відношенням, що кортеж представлений на віддаленому вузлі;

$P(x_i)$ – ймовірність значення x_i атрибуту i;

$P(x_i \mid \text{потрібен})$ – ймовірність значення x_i атрибуту i на підмножині кортежів відношення, що представлені на віддаленому вузлі.

Також аналогічно розраховуємо

$$P_x(\text{непотрібен} \mid x_i) = \frac{P(x_i \mid \text{непотрібен}) * P(\text{непотрібен})}{P(x_i)} \quad (3.14)$$

Отримавши ймовірності $P_x(\text{потрібен} \mid x_i)$ для всіх атрибутів кортежу на базі (3.13) та (3.14), виконується розрахунок загального рівня імовірності необхідності представлення кортежу:

$$P(\text{потрібен} \mid X) = \frac{\prod_{i=1}^{n} P_x(\text{потрібен} \mid x_i)}{\prod_{i=1}^{n} P_x(\text{потрібен} \mid x_i) + \prod_{i=1}^{n} P_x(\text{непотрібен} \mid x_i)} \quad (3.15)$$

Якщо значення, отримане за (3.13) перевищує оптимальне значення рівня маркеру представленості даних, отримане за (3.9) або (3.10), приймається рішення про потрібність кортежу на вузлі РКІС.

$$F_{\text{X}}^{\text{Потрібен}} = \begin{cases} \text{істина}, & \frac{(koef_{\text{предст}}^{node} + 1)}{2} > P(\text{потрібен} \mid X) \\ \text{хибність}, & \frac{(koef_{\text{предст}}^{node} + 1)}{2} < P(\text{потрібен} \mid X) \end{cases}$$

Висновки до розділу 3

Для вибору найкращої альтернативи рівня маркеру представленості даних використано MAI. Для побудови матриць переваг альтернатив, їх множина була звужена до п’яти. Матриця переваг за критеріями оптимальності отримана класично, із залученням ЛПР та подальшою оцінкою індексу узгодженості. Матриці переваг альтернатив заповнюються автоматично, без участі ЛПР, на базі моделей критеріїв оптимальності, сформульованих у розділі 2. Також вводяться обмеження множини допустимих значень критеріїв оптимальності у вигляді

допустимих максимального та мінімального значень по кожному критерію, що веде до звуження кількості альтернатив на останньому рівні ієрархії моделі.

Після розрахунку та отримання вектору глобальних пріоритетів альтернатив, з метою підвищення точності отриманого результату, використано апарат нечітких множин, представивши знайдений вектор глобальних пріоритетів, як вектор нечітких чисел для маркеру представленості даних із подальшим приведенням до чіткого значення. Даний підхід дозволив зберегти точність отриманого результату при зменшенні кількості альтернатив рішень.

Для нових кортежів, що додаються до відношень БД після виконання розрахунків, запропоновано розв’язання задачі класифікації при визначенні їх приналежності до одного з двох класів – «потрібних» на віддаленому вузлі та «непотрібних». Порівнявши найбільш популярні підходи до вирішення задачі класифікації, було обрано алгоритм наївного Байєса, оскільки за умови відсутності транзитивних залежностей (вимога до третьої нормальної форми реляційної БД) всі атрибути відношення є незалежними. Після отримання ймовірності приналежності кортежу до класу «потрібен» і виконання нормування значення, вона приймається за рівень маркеру представленості та, відповідно, дані завантажуються на вузол, якщо це значення перевищує оптимальний рівень маркеру представленості для вузла РКІС.

РОЗДІЛ 4
ОБЛІК ТА АНАЛІЗ КОРИСТУВАЦЬКИХ ЗАПИТІВ, ЯК СКЛАДОВА ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ ОПТИМІЗАЦІЇ СТРУКТУРИ БД ВУЗЛА РКІС

4.1 Функціональне моделювання інформаційної технології

Функціональна модель розроблюваної інформаційної технології, що має на меті оптимізацію структури БД вузла РКІС, реалізована у вигляді моделі IDEF0 за методологією SADT [129-131], та на нульовому рівні ієрархії має наступний вигляд (рис. 4.1).

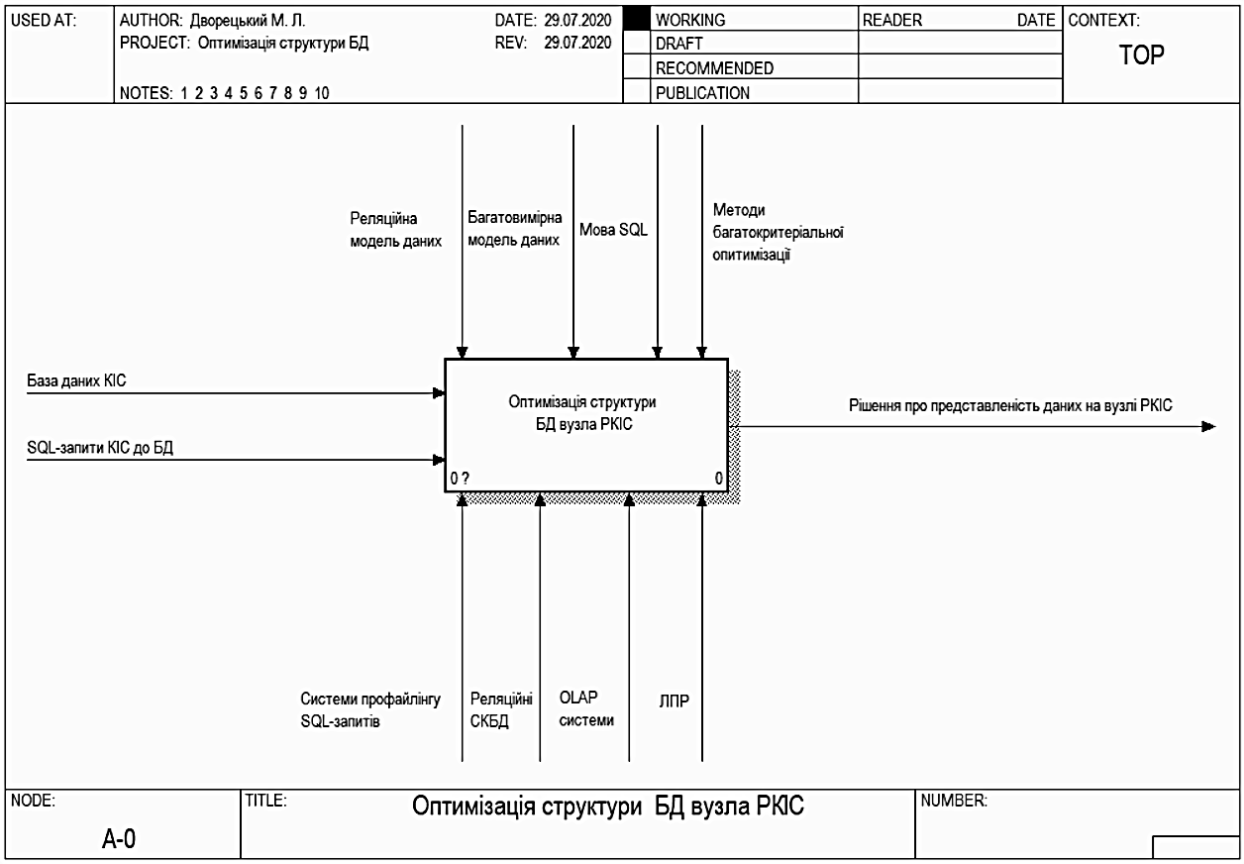


Рисунок 4.1 – Функціональна модель (0-й рівень ієрархії, A-0)

Вхідними даними виступає інформація, що описує SQL-запити, які надходять до БД від КІС та сама БД КІС. До даних SQL-запиту віднесено сам текст запиту, а також наявні у розпорядженні профайлінгових систем СКБД додаткові аналітичні дані, такі як: назва застосунку, ім’я користувача, транзакція,

назва та адреса хоста та ін. Щодо БД КІС, то тут важливі як структура БД, що описує відношення, їх атрибути, первинні та зовнішні ключі, так і самі дані, оскільки при використанні комбінованої стратегії розподілу виконується як горизонтальна, так і вертикальна фрагментація (вибірка та проекція).

Ресурсами та механізмами виступають системи профайлінгу СКБД, які необхідні для збору статистичних даних SQL-запитів; реляційні СКБД – при збереженні та обробці аналітичних даних SQL-запитів, а також при заповненні ЛПР значень маркеру представленості даних для елементів аналітичних вимірів; системи Online Analytical Processing (OLAP) [132] для виконання багатовимірного аналізу накопичених даних та розрахунку результуючих значень маркеру представленості для атрибутів та кортежів відношення відповідно до заданих ЛПР зрізів даних; та ЛПР при редагуванні маркеру представленості для елементів вимірів, виконанні зрізів даних ББД, та заповненні матриці попарних порівнянь для розрахунку векторів відносних пріоритетів альтернатив та критеріїв.

Керуючими впливами є реляційна модель даних, із набором обмежень та правил представлення даних та операцій над ними, та багатовимірна модель даних, що використовується для представлення множини SQL-запитів у вигляді гіперкубу. Перелік операторів мови SQL виступає у ролі стандарту при створенні формальної граматики, а методи БКО дозволяють виконати їх аналіз та обрати найкращу альтернативу для оптимального рівня маркеру представленості даних.

Вихідним потоком є рішення про представлення (або непредставлення) даних (атрибута чи кортежа відношення) на вузлі БД РКІС.

На рис. 4.2 представлено функціональну модель IDEF0 для 1-го рівня ієрархії. Задача оптимізації структури БД представлена у вигляді чотирьох підзадач: аналіз тексту та результату SQL-запиту; облік SQL-запитів КІС до БД; розрахунок рівня маркеру представленості даних та вибір найкращої альтернативи при визначенні оптимального рівня маркеру представленості даних на вузлі РКІС.

Задача аналізу тексту та результату SQL-запиту полягає у накопиченні статистики та виділенні із тексту SQL-запитів таких аналітичних характеристик,

як перелік відношень, атрибутів, та кортежів, до яких виконується звернення. Вхідними даними виступають SQL-запити та БД КІС. Ресурсами виступають системи профайлінгу та реляційні СКБД, керуючими впливами – стандарти мови SQL та реляційна модель даних, результатом є набір аналітичних характеристик SQL-запиту.

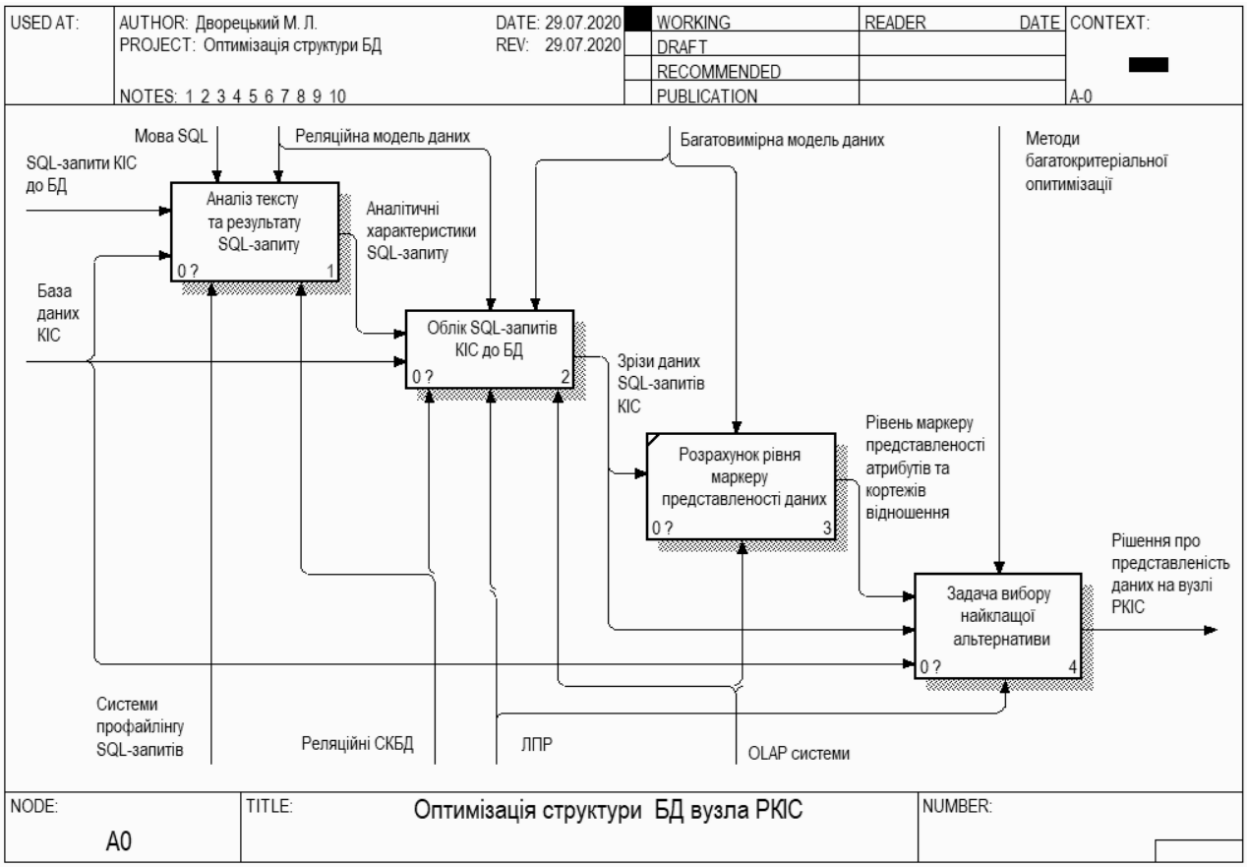


Рисунок 4.2 – Функціональна модель (1-й рівень ієрархії, A0)

Задача обліку SQL-запитів КІС до БД включає у себе розробку структури та наповнення даними реляційної та багатовимірної БД для отримання необхідних зрізів даних за набором аналітичних характеристик запиту (застосунку, місця розташування, ролі користувача, та ін.). Вхідними даними виступає БД КІС та набір SQL-запитів із аналітичними характеристиками. Ресурсами є системи управління реляційними та багатовимірними БД, та ЛПР, що виконує заповнення рівня маркеру представленості для елементів вимірів ББД. Керуючими впливами є стандарти та вимоги до реляційної та багатовимірної моделей даних. Виходом є підмножина SQL-запитів, що є

результатом операцій зрізу та консолідації за набором вимірів відношення–кортеж або відношення–атрибут.

За виходом, отриманим із попередньої задачі, за (2.7) виконується розрахунок рівня маркеру представленості даних для атрибутів та кортежів відношення. При цьому, керуючим впливом є багатовимірна модель, а ресурсом – система OLAP. Результатом є конкретні значення рівня маркеру представленості даних для кожного кортежу та атрибуту таблиці.

Останньою є задача визначення оптимального граничного значення маркеру представленості даних на вузлі КІС, для якої входом є БД КІС, зрізи даних SQL-запитів та рівень маркеру представленості атрибутів та кортежів відношення; керуючим впливом – методи БКО, а ресурсом – ЛПР (при заповненні матриці попарних порівнянь для розрахунку векторів відносних та глобальних пріоритетів альтернатив). Виходом є рішення про представлення або непредставлення відповідних даних на вузлі РКІС.

Перші дві із наведених задач є предметом технології визначення аналітичних характеристик SQL-запитів, їх обліку та аналізу, і детально розглянуті у даному розділі.

4.2 Задача парсингу SQL-запитів

Задача аналізу тексту та результату SQL-запиту з метою виявлення його аналітичних характеристик у вигляді набору відношень, атрибутів та кортежів, представлена у вигляді функціональної моделі IDEF0 на рис. 4.3.

Виділено задачі накопичення статистичних даних SQL-запитів до БД КІС [77; 133], створення граматики мови SQL, парсинг тексту запиту та визначення переліку кортежів відношення, що задіяні у межах запиту. Перша із них має вхідним потоком множину SQL-запитів КІС, керується реляційною моделлю даних, та використовує реляційні СКБД і механізми профайлінгу. Зауважимо, що на даному етапі механізми профайлінгу вже дозволяють визначити частину необхідних аналітичних характеристик запиту. Задача створення граматики керується синтаксисом мови SQL та дає на виході класи лексора та парсера тексту

SQL-команд, які використовуються задачею парсингу для отримання ієрархічного дерева відношень та їх атрибутів, що використовуються у межах запиту [76; 77]. Окремою паралельною задачею є визначення переліку кортежів відношення, вхідними потоками якої виступають SQL-запити, структура та дані БД КІС, у межах якої запит розбивається на множину підзапитів, кількість елементів якої відповідає кількості задіяних відношень [107; 115]. Виходи задач накопичення статистики запитів, парсингу та визначення діапазону ключів об’єднуються, і дають за виході повний перелік необхідних аналітичних характеристик SQL-запиту.

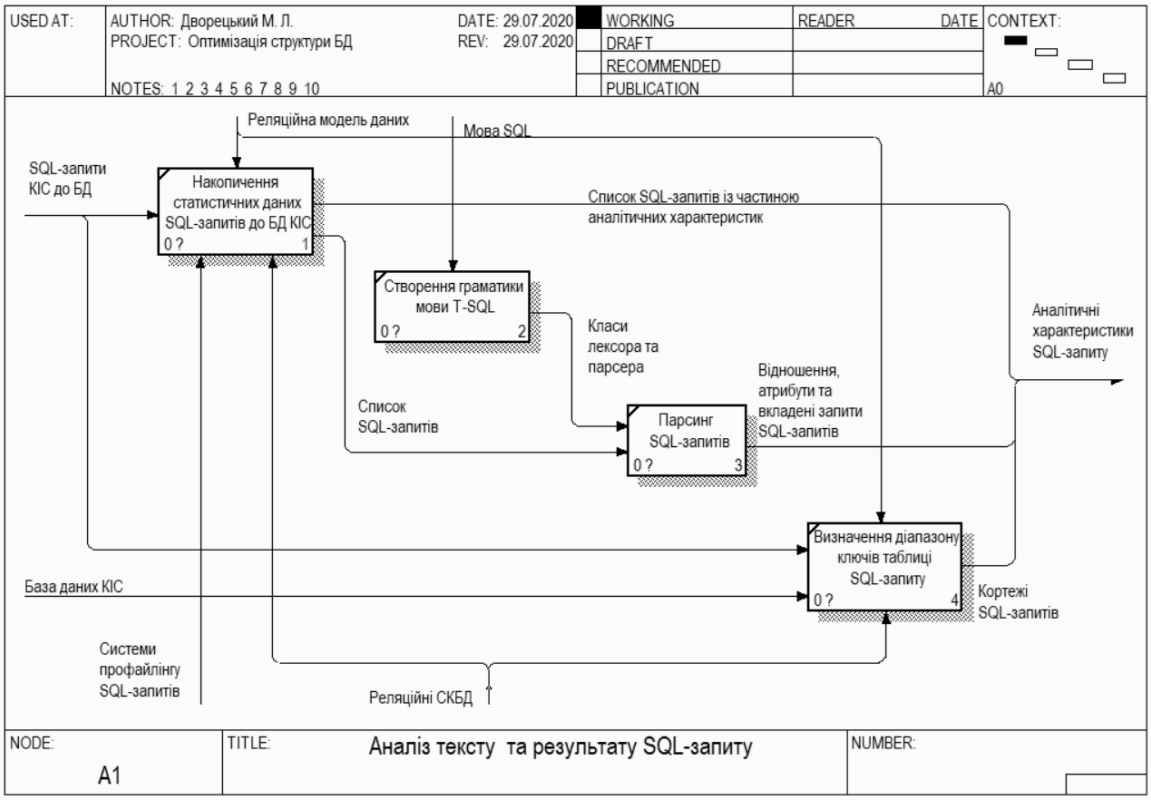


Рисунок 4.3 – Функціональна модель задачі аналізу тексту та результату SQL-запиту (2-й рівень ієрархії, A1)

Більшість сучасних СКБД мають у своєму розпорядженні розвинені механізми з відслідковування користувацької активності при роботі з БД [53; 67-68; 70]. Також на ринку ПЗ присутні програмні продукти сторонніх розробників, що дозволяють фіксувати сеанси взаємодії із сервером та відображають список запитів із прив’язкою до робочої станції, програмного забезпечення та

користувача [134-136]. Крім того, переважна більшість ІС мають у своєму розпорядженні власні інструменти профайлінгу користувацьких запитів. Вибір того чи іншого інструменту залежить від СКБД, що використовується для роботи із БД, наявності в ІС власних механізмів та від глибини необхідного аналізу отриманих даних.

В рамках даного дослідження обрано програмний продукт SQL Profiler, що входить до стандартного інсталяційного пакету SQL Server 2019 [137]. Даний вибір обумовлено достатнім переліком аналітичних властивостей, що надає функціонал ПЗ та сумісністю із версією СКБД, що використовується на підприємстві – базі автоматизації. При зміні вхідних умов вибір може бути змінено на користь іншого ПЗ. Створивши новий сеанс профайлінгу, є можливість автоматичного створення таблиці, у яку записуються результати спостереження за користувацькою активністю. У ході даного дослідження було обрано перелік полів, що можна побачити на рис. 4.13 у таблиці *profilingResultSet*, який наводиться під час розгляду логічної та фізичної моделей БД збереження даних SQL-запитів. Даний перелік дозволяє отримати необхідні для проведення подальшого аналізу дані.

Задачі створення граматики мови T-SQL та парсинту SQL-запитів (рис. 4.3) мають на меті визначити набір відношень, їх атрибутів, а також множину вкладених підзапитів, що задіяні у межах кожного SQL-запиту. Вхідними даними виступає список запитів, що є результатом розв’язання задачі накопичення статистичних даних (рис. 4.3), при створенні граматики керуючим впливом є стандартами мови SQL та на виході отримано перелік відношень та атрибутів, задіяних у межах запиту. Також розв’язання задачі створення граматики робить можливим генерацію класів лексора та парсера SQL-запиту для виконання подальшого парсингу тексту.

Кожна мова програмування має правила, які визначають синтаксичну структуру програмного коду. Синтаксис конструкцій мови програмування може бути описаний за допомогою контекстно-вільних граматик або нотації БНФ (Backus-Naur Form, форм Бекуса-Наура) [138–140]. До переваг граматик

відносять: точність специфікації; структурованість; гнучкість; автоматична побудова синтаксичних аналізаторів.

Формально граматика визначається, як *G = (Vn, Vt, S, P)*, де *Vn* і *Vt* – відповідно множини нетермінальних, і термінальних символів, що не перетинаються; *S* – виділений символ у *Vn*, що звичайно називають вихідним або початковим символом; *P* – кінцева множина продукцій (правил), за якими нетермінальні символи визначаються як упорядкована послідовність термінальних та/або нетермінальних символів. Об'єднання множин *Vn* і *Vt* називають словником граматики.

Найбільш ефективні спадні й висхідні методи граматичного розбору працюють тільки з підкласами граматик, однак деякі із цих підкласів, такі як LL-граматики і LR-граматики, досить виразні для опису більшості синтаксичних конструкцій мов програмування. Синтаксичні аналізатори для класу LR-граматик, як правило, створюються за допомогою автоматизованих генераторів (YACC, BIZON, ANTLR та інші).

SQL – декларативна мова програмування для взаємодії користувача з БД, що застосовується для формування запитів, оновлення і керування реляційними БД, створення схеми бази даних та її модифікації, системи контролю за доступом до бази даних [2; 141; 142]. SQL складається з наступних підрозділів: Data Definition Language (DDL) – робота зі структурою бази; Data Manipulation Language (DML) – робота з даними; Data Control Language (DCL) – робота з правами; Transaction Control Language (TCL) – робота з транзакціями. DML – це сімейство комп'ютерних мов, що використовуються в комп'ютерних програмах або користувачами баз даних для отримання, вставки, видалення або зміни даних в базах даних. Зараз найпопулярнішою DML є SQL, що використовується для отримання і маніпулювання даними в реляційній БД.

Нижче наведено спрощений синтаксис команди DML «SELECT» – оператора мови SQL, що повертає результуючий набір рядків однієї чи багатьох відношень. Синтаксис оператора «SELECT» є досить розгалуженим, проте в спрощеному вигляді його можна описати так:

```
SELECT <список_вибірки>
[ INTO <нова_таблиця> ]
FROM <таблиця>
[ WHERE <умови_пошуку> ]
[ GROUP BY <групувати_по_умові> ]
[ HAVING <умови_пошуку> ]
[ ORDER BY <сортувати_по_умові> [ ASC | DESC ] ]
```

У більшості застосунків, команда «SELECT» зустрічається найчастіше [7; 19; 142]. Реальний рівень складності команди «SELECT» значно вищий, наряду із цим задача ускладняється наявністю різних діалектів мови SQL, що використовується при роботі із різними СКБД. Відповідно, працюючи із певною СКБД, маємо необхідність у створенні специфічного механізму лінгвістичного та синтаксичного аналізу тесту запиту. У якості шляху розв'язання задачі запропоновано використання програмного продукту ANTLR на базі проекту Java 1.8 для парсингу користувацького запиту із попереднім створенням граматики для відповідного діалекту SQL. Надалі створення граматики розглядається на прикладі мови T-SQL для СКБД SQL Server. Даний вибір обумовлено СКБД системи обліку користувацьких запитів, а також приналежність основної БД підприємства – тестової бази автоматизації до сімейства SQL Server.

При використанні даної технології для парсингу запитів інших діалектів SQL виникає необхідність у розробці окремого файлу граматики із подальшою генерацією за допомогою ANTLR відповідних класів лексору та парсеру. Використання основних положень шаблонного проектування (Design Pattern) при реалізації інформаційної технології дозволяє виконувати дану операцію без необхідності зміни коду класів верхнього рівня [77]. Даний факт свідчить про високий рівень адаптованості технології під різні діалекти SQL.

Для створення граматики, виконується формальний опис всіх використовуваних у користувацьких запитах SQL-команд. При реалізації даного завдання для T-SQL було використано документацією Microsoft SQL Server Language References, а саме Transact-SQL Reference (Database Engine) [143]. Так, команда «SELECT» в T-SQL формалізується наступним чином:

```
<SELECT statement> ::=
    [ WITH { [ XMLNAMESPACES ,] [ <common_table_expression> [,...n] ] } ]
    <query_expression>
    [ ORDER BY { order_by_expression | column_position [ ASC | DESC ] }
  [ ,...n ] ]
    [ <FOR Clause>]
    [ OPTION ( <query_hint> [ ,...n ] ) ]

<query_expression> ::=
    { <query_specification> | ( <query_expression> ) }
    [  { UNION [ ALL ] | EXCEPT | INTERSECT }
        <query_specification> | ( <query_expression> ) [...n ] ]

<query_specification> ::=
SELECT [ ALL | DISTINCT ]
    [TOP ( expression ) [PERCENT] [ WITH TIES ] ]
    <select_list>
    [ INTO new_table ]
    [ FROM { <table_source> } [ ,...n ] ]
    [ WHERE <search_condition> ]
    [ <GROUP BY> ]
    [ HAVING < search_condition > ]
```

Наведений фрагмент не є завершеною граматикою для команди «SELECT», та потребує подальшого визначення таких токенів, як «common_table_expression», «order_by_expression», «column_position», «FOR Clause», «query_hint» та ін.

Генератор парсерів ANTLR є LL (*), існує вже понад 25 років, зараз його розробка ведеться на GitHub. В даний момент він дозволяє генерувати парсери на різних мовах, серед яких Java, C#, Python та JavaScript. Головна частина проєкту – генератор парсерів, що є консольною програмою, якій на вхід подається файл з описом граматики кінцевого компілятора. Генератор читає цей файл, і у відповідь генерує вихідні коди компілятора або помилки, якщо граматика порушена. Фактично ANTLR є інструментом, який перетворює граматику на парсер/лексор в Java (або іншої мови програмування із переліку, що підтримуються).

Використання ПЗ ANTLR у режимі командного рядку має свої переваги, однак є досить незручним при написанні великих граматик, тестуванні отриманих результатів та пошуку помилок. Існує ряд розширень, розроблених під більшість популярних інтегрованих середовищ розробки підтримуваних ANTLR мов програмування, що дозволяють інтегрувати інструменти ANTLR із середовищем

розробки та зробити процес створення, відлагодження та інтеграції граматик із основним проектом більш зручним та швидким.

Правила граматики пишуться на спеціальній мові, заснованій на регулярних виразах. Синтаксис, що використовується в ANTLR схожий на синтаксис мови C з деякими розширеннями для опису граматики. Також синтаксис дозволяє виконувати вставки на цільовій мові програмування. Дана можливість використовується при побудові ієрархії вкладених запитів із списками відношень, атрибутів та кортежів по кожному з них. На рис. 4.4 наведено фрагмент коду створеної граматики, що відповідає за команду «SELECT».

```
select_statement returns [HashSet<TsqlRelation> RelationList, TsqlCommand Command]
@init {
    $Command = new TsqlCommand();
}
    : with_expression? (query_expression {
        $RelationList = $query_expression.RelationList;
        $Command = TsqlCommand.getValueFrom($Command, $query_expression.Command);
    }) order_by_clause? for_clause? option_clause? ';'?
    ;
```

Рисунок 4.4 – Представлення *select_statement* у середовищі ANTLR v4

При переході через «query_expression» до «query_specification» виконується формалізація синтаксису команди «SELECT», що було описано вище (рис. 4.5).

```
: SELECT (ALL | DISTINCT)? (TOP expression PERCENT? (WITH TIES)?)?
 select_list {
   $RelationList.addAll($select_list.RelationList);
   $SelectCommand = $select_list.Command;
   $Command.setFieldList($select_list.Command.getFieldList());
 }
 // https://msdn.microsoft.com/en-us/library/ms188029.aspx
 (INTO table_name)?
 (FROM table_sources {
       $RelationList.addAll($table_sources.RelationList);
       $Command = TsqlCommand.getValueFrom($Command, $table_sources.Command);
 })?
 (WHERE where=search_condition {
       $RelationList.addAll($search_condition.RelationList);
       $SearchCommand = $search_condition.Command;

       HashSet<TsqlField> fldList = $Command.getFieldList();
       fldList.addAll($search_condition.Fields);
       $Command.setFieldList(fldList);
 })?
 // https://msdn.microsoft.com/en-us/library/ms177673.aspx
 (GROUP BY group_by_item (',' group_by_item)*)?
 (HAVING having=search_condition)?
;
```

Рисунок 4.5 – Фрагмент представлення query_specification в ANTLR v4

Повний текст розробленої граматики для опису мови T-SQL наведено у додатку Г даного дослідження.

Правила граматики, що представлені БНФ, можна представити у графічній формі, що робить граматику більш наглядною. Аналіз сполучень різноманітних елементів рядка для визначення того, чи є або не є даний рядок символів реченням мови, називається граматичним розбором. Для синтаксично правильного речення за допомогою створеної граматики можна побудувати дерево граматичного розбору. На рис. 4.6 наведено приклад такого дерева для запиту «select (select number from groups where id=students.id), name, age from students», що звертається до декількох атрибутів і має у своєму складі один вкладений запит.

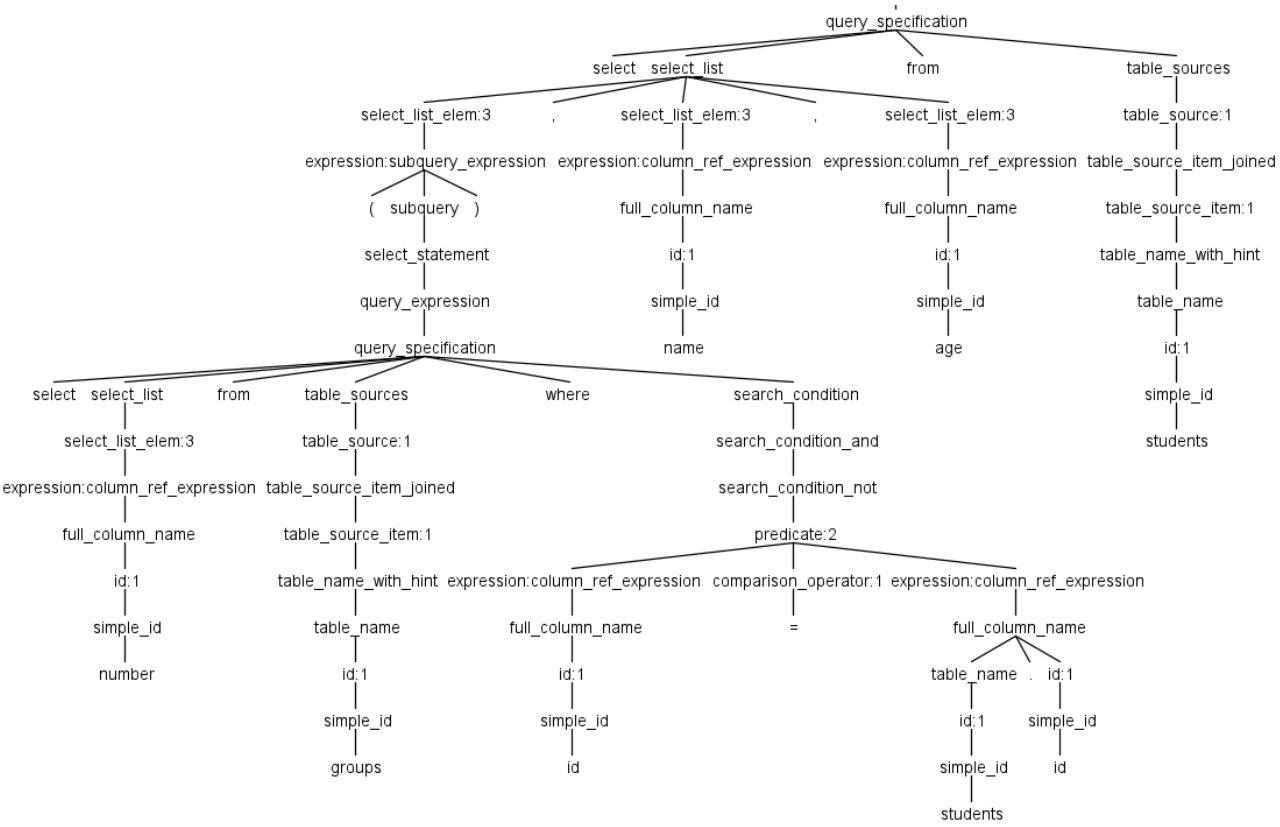


Рисунок 4.6 – Дерево парсингу запиту T-SQL

Далі кожен запит представляється у вигляді об’єкту, що має наступні атрибути: робочу станцію, користувача та програмне забезпечення, від яких надійшов запит; сам текст запиту; колекцію таблиць, до яких звертається запит; та колекцію вкладених запитів, якщо такі мають місце [77]. Фрагмент діаграми класів підсистеми парсеру SQL-запитів наведено на рис. 4.7.

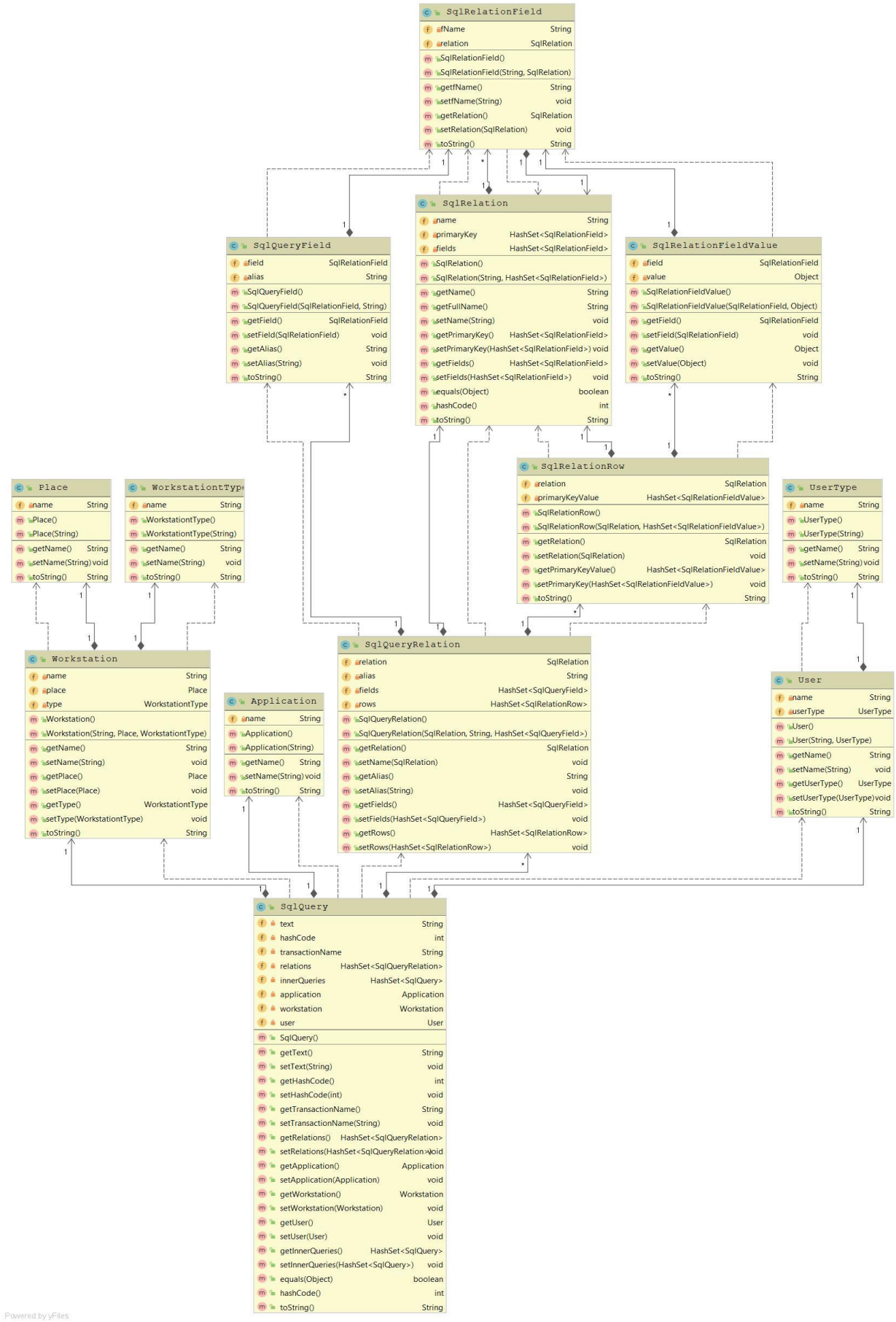


Рисунок 4.7 – Класи, що відповідають за представлення запитів у моделі

Сутність «таблиця» (SqlQueryRelation) у даному випадку є підмножиною таблиці БД (SqlRelation) і містить назву таблиці, колекцію атрибутів, які задіяні у запиті, та колекцію значень первинного ключа відношення, згідно з множиною кортежів, які повертаються як результат роботи запиту [115].

Отже, отримано множину відношень із атрибутами, що використовуються у запиті. Цього достатньо для визначення списку таблиць, представлених у вузлі розподіленої БД, та для обмеження загальної кількості даних через виконання операції проекції до них. Однак цього замало, оскільки у БД інформаційних систем існують таблиці із великою кількістю кортежів, що займають багато місця на дисковому носії та створюють навантаження при виконанні запитів до них [76; 107]. При цьому реально використовується досить невеликий відсоток від загального обсягу даних. Тому задача визначення множини кортежів, що використовуються у тому чи іншому запиті, також потребує розв’язання. Вхідними даними для неї виступає як набір SQL-запитів, так і сама БД КІС, при чому як її структура (для визначення ключових полів відношення), так і самі дані (для визначення діапазонів значень ключів відношення). У якості ресурсу виступають реляційні СКБД, а виходом є набір кортежів відношень, що задіяні у межах певного SQL-запиту (рис. 4.3).

Для цього клас SqlRelation доповнено колекцією типу SqlRelationField, що визначає первинний ключ відношення. У випадку, якщо первинний ключ у відношенні відсутній, або він складається із великої кількості атрибутів нечислового типу даних, і, як наслідок має великий об’єм, на рівні БД може бути створене унікальне автоінкрементне поле із обмеженням цілісності «not null», що буде використовуватись надалі для ідентифікації кортежу. На рівні запиту у клас SqlQueryRelation вводиться колекція SqlRelatonRow, що визначає набір рядків відношення, що задіяні у межах запиту. Якщо запит має у своєму складі вкладені підзапити, кожен із них обробляється окремо.

Для кожної таблиці із колекції відношень запиту виконується додатковий запит до БД, список атрибутів у якому замінюється на первинний ключ відношення. Результат запиту запам’ятовується, як множина кортежів таблиці,

що використовується поточним запитом. Так, наприклад, у випадку надходження запиту, що вибирає прізвища студентів із номерами студентських груп із двох таблиць із використанням вкладеного запиту для фільтрації за спеціальністю (рис. 4.8), він буде розбитий на наступну множину підзапитів (рис. 4.9).

```
select s.name, g.number
from students s inner join groups g on s.gr_code = g.code
where g.special_code in(
    select code from specialties where name = 'computer science'
)
```

Рисунок 4.8 – Приклад вхідного запиту

```
select code from specialties where name = 'computer science';

select distinct s.code from students s inner join groups g on s.gr_code = g.code
where g.special_code in(
    select code from specialties where name = 'computer science'
);

select distinct g.code from students s inner join groups g on s.gr_code = g.code
where g.special_code in(
    select code from specialties where name = 'computer science'
);
```

Рисунок 4.9 – Список підзапитів для отримання множин атрибутів відношень, що використовуються вхідним запитом

Після виявлення множин відношень, атрибутів та кортежів користувацького запиту, з’являється можливість розрахунку рівня маркеру представленості, використовуючи функцію агрегації (2.7), що, у свою чергу, робить можливим використати (2.8) для визначення оптимального рівня маркеру представленості, відповідно до оптимальних значень критеріїв (2.10, 2.14, 2.18). Наступною, згідно з функціональною моделлю A0 (рис. 4.2), є реалізація реляційної та багатовимірної БД для розв’язання задачі обліку та аналізу множини SQL-запитів.

4.3 Система обліку запитів та маркеру представленості даних

Задача обліку SQL-запитів КІС до БД розділяється на наступні чотири задачі: розробка структури реляційної БД обліку SQL-запитів; заповнення класифікаторів аналітичних характеристик SQL-запиту; заповнення рівня маркеру представленості для елементів вимірів SQL-запиту та створення багатовимірної БД обліку SQL-запитів (рис. 4.10). Результатом вирішення задачі аналізу тексту та результату SQL-запиту є журнал SQL-запитів із набором відповідних аналітичних характеристик («користувач», «застосунок», «робоча станція», «список відношень», «атрибутів» та «кортежів»). Вищенаведені дані мають бути збережені для можливості подальшої їх обробки, для забезпечення чого необхідно вирішити задачу розробки структури реляційної БД обліку SQL-запитів.

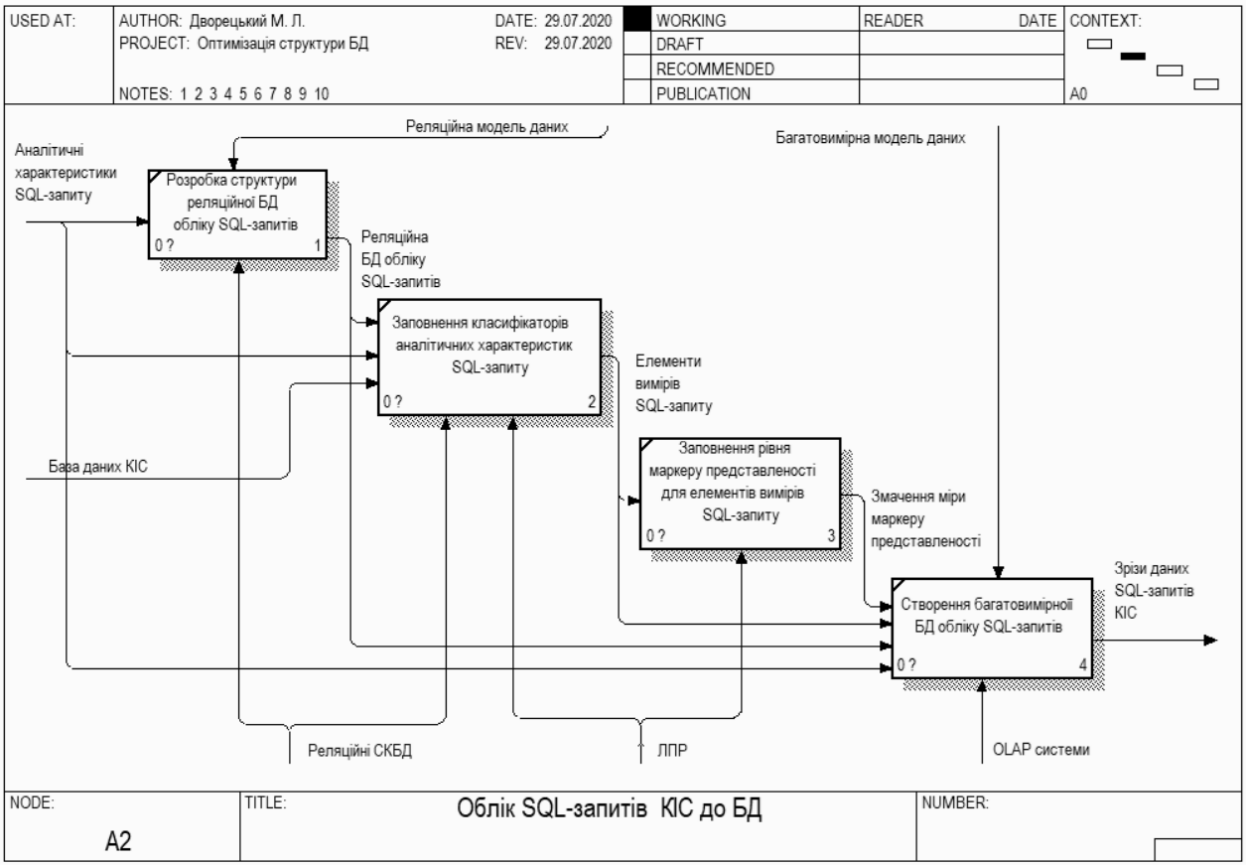


Рисунок 4.10 – Функціональна модель задачі обліку SQL-запитів КІС до БД (2-й рівень ієрархії, A2)

При проєктуванні структури БД на концептуальному рівні виділяється наступний перелік сутностей: «місце розташування», «робоча станція», «програмне забезпечення», «користувацький запит», «відношення», «атрибут», «кортеж», «транзакція». Відношення між ними виглядають наступним чином: «місце розташування» має «робочі станції» (або «робоча станція» належить до «місця розташування») із типом зв'язку 1:n; «програмне забезпечення» встановлено на «робочу станцію» (або «робоча станція» має «програмне забезпечення») із типом зв'язку *n:m*; «програмне забезпечення» генерує «транзакції», тип зв'язку *1:n*; «транзакції» містять «користувацькі запити», тип зв'язку n:m; «користувацькі запити» звертаються до «відношень», «атрибутів» та «кортежів»; «відношення» складаються з «атрибутів» та «кортежів», тип зв'язку *1:n* (рис. 4.11).

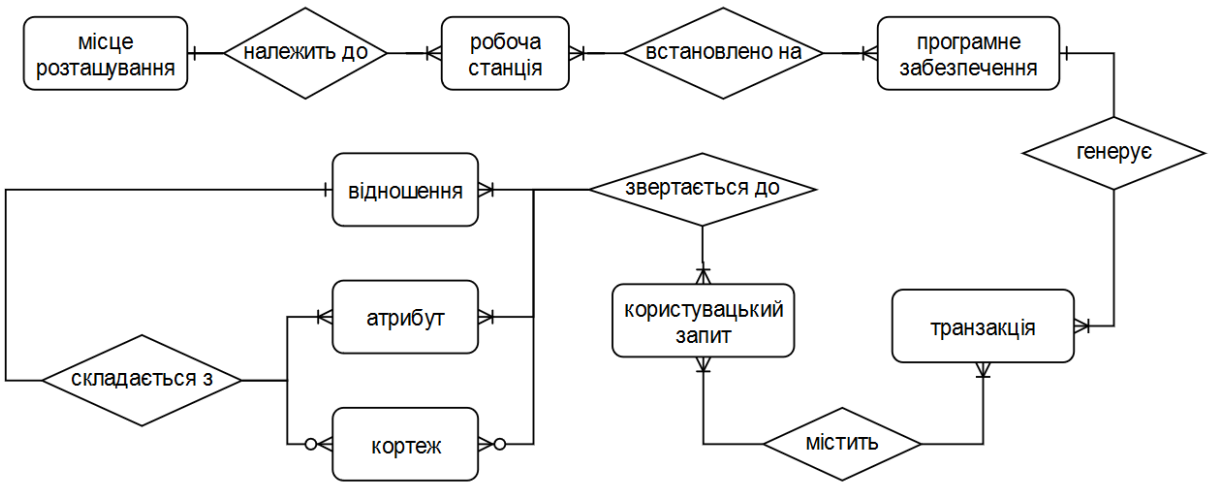


Рисунок 4.11 – Діаграма «сутність-зв'язок» до нормалізації відношень

Після приведення БД до третьої нормальної форми, до ERD додаються такі відношення, як «тип робочої станції», «ПЗ робочої станції», «журнал транзакцій», «вкладені запити», «список відношень запиту», «список атрибутів запиту», «список кортежів запиту» (рис. 4.12).

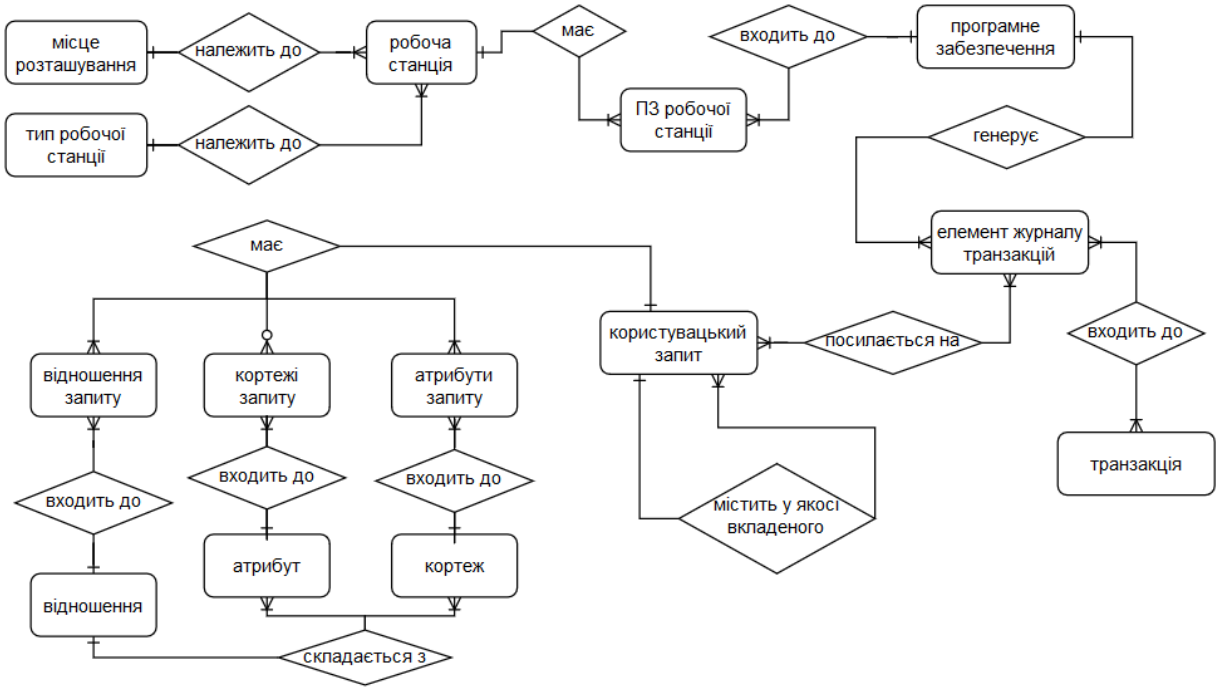


Рисунок 4.12 – Діаграма «сутність-зв’язок» після нормалізації відношень

Для спрощення механізму класифікації робочих станцій прийнято рішення про встановлення на рівні БД зв’язку між відношеннями «Робоча станція» та «Тип робочої станції» із типом зв’язку один до багатьох. При цьому для спрощення було знехтувано тим фактом, що можливе встановлення на одну робочу станцію програмного забезпечення, яке має відношення до різних типів хостів. Дана ситуація може мати місце при використанні одної робочої станції декількома співробітниками, або у випадку використання «універсальних» робочих місць, чи, наприклад, режиму роботи у вигляді термінальних сесій. Крім того, необхідної фільтрації можна досягнути, використавши для цього відношення «програмне забезпечення».

На наступному кроці проєктування визначається перелік атрибутів відношень, наведених на рис. 4.12. Отримана в результаті схема даних наведена на рис. 4.13. Повний скрипт реалізації фізичної моделі БД обліку користувацьких запитів наведено у додатку В.

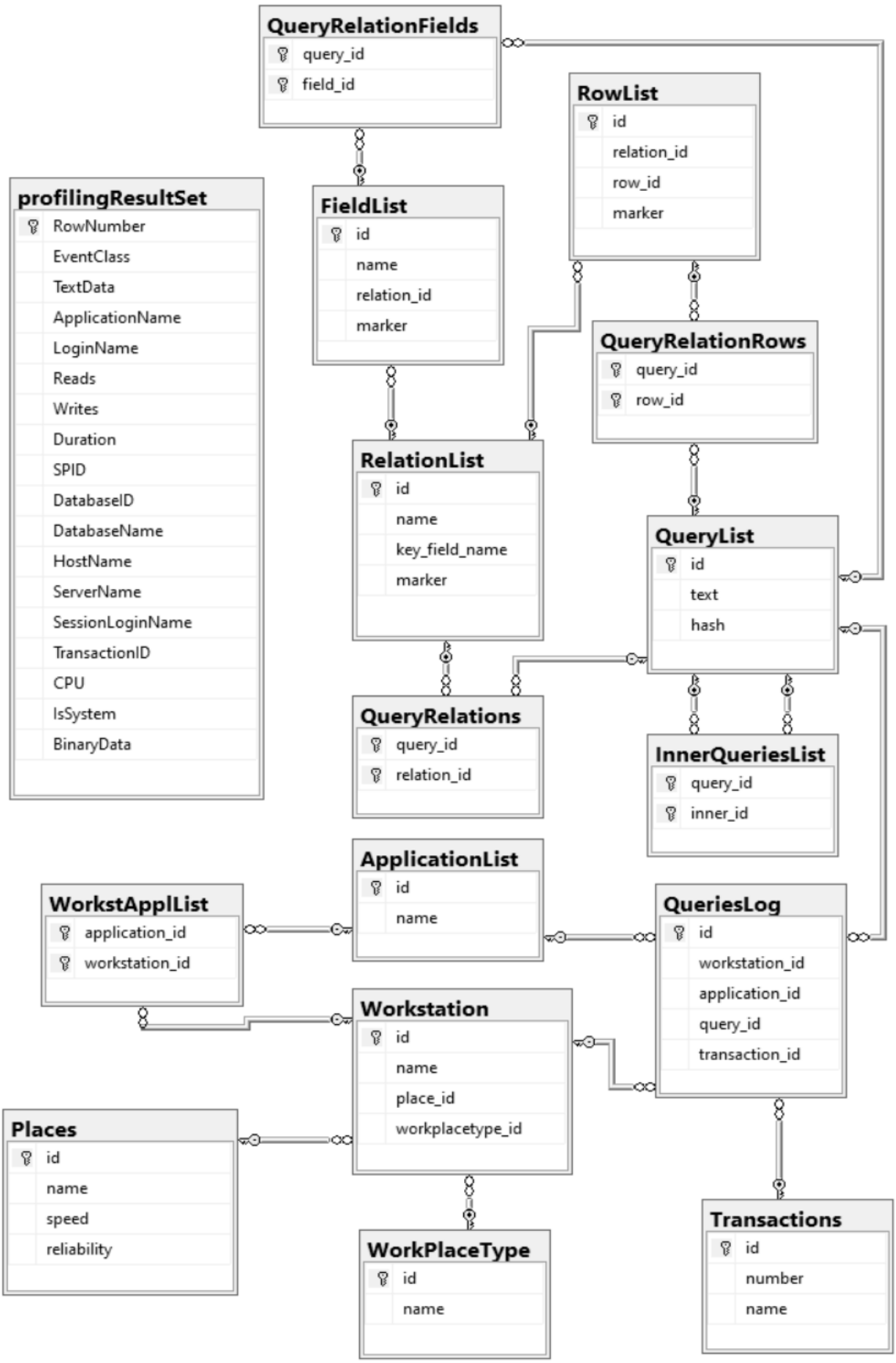


Рисунок 4.13 – Даталогічна модель підсистеми обліку користувацьких SQL-запитів

У системі можливе ручне уведення та редагування даних для таких сутностей, як «типи хосту», «робочі станції», «місця розташування» та «програмне забезпечення». Наряду із цим, передбачені механізми імпорту даних із текстових та csv-файлів для можливості взаємодії із сторонніми програмними продуктами при заповненні даних, наприклад по таким сутностям, як «програмне забезпечення» або «список хостів». Ця інформація може бути отримана, наприклад, із ActiveDirectory або із використанням таких безкоштовних програмних продуктів, як Advanced IP Scanner [144], NetWrix Inactive Users Tracker [145] або WinAudit Freeware [146]. Звісно, у випадку використання операційної системи, відмінної від Microsoft Windows, існує багато інших адміністративних засобів, що дозволяють отримати такого роду інформацію.

Поряд із звичними формами вводу та засобами імпорту даних із стороннього ПЗ, присутні сутності, що наповнюються даними виключно із використанням методів сканування структури БД. Так, наприклад, список відношень БД у випадку використання системи керування базами даних SQL Server може бути оновлено із використанням наступного скрипта SQL, наведеного на рис. 4.14.

```
delete from RelationList
where id not in(select query_id from QueryRelations)

declare @max_id int = (select isnull(max(id), 0) from RelationList)

insert into RelationList(id, name)
select ROW_NUMBER() over(order by object_id) + @max_id, name
from sys.objects
where type = 'U'
```

Рисунок 4.14 – T-SQL скрипт для оновлення даних у списку відношень бази даних

Даний фрагмент коду наведено для випадку використання однієї із версій MS SQL Server. Переважна більшість сучасних систем керування реляційними базами даних, таких як Oracle, MySQL, Postgree SQL, FireBird та ін., також мають у своєму розпорядженні механізми з отримання даних списку користувацьких таблиць, але синтаксис команд при цьому буде різнитися. Виходячи із цього,

програмно дана частина реалізована за допомогою композиції із використанням шаблону ООП «стратегія», отже достатньо лише обрати СКБД, що підтримується в налаштуваннях ПЗ.

Використовуючи аналіз отриманих даних щодо профайлінгу користувацької активності, отримано списки робочих станцій, користувачів та програмного забезпечення, що виконує звернення до інформації, розташованій в БД. Так, для отримання списку робочих станцій та оновлення його у відповідній таблиці підсистеми обліку користувацької активності, виконується SQL-код, наведений на рис. 4.15.

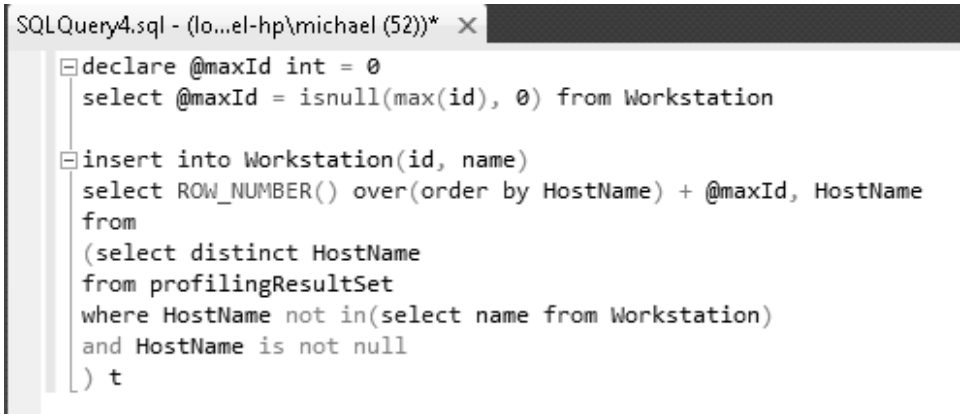


Рисунок 4.15 – T-SQL скрипт для оновлення даних списку робочих станцій

Аналогічним чином оновлюється список програмного забезпечення та користувачів БД (рис. 4.16–4.17).

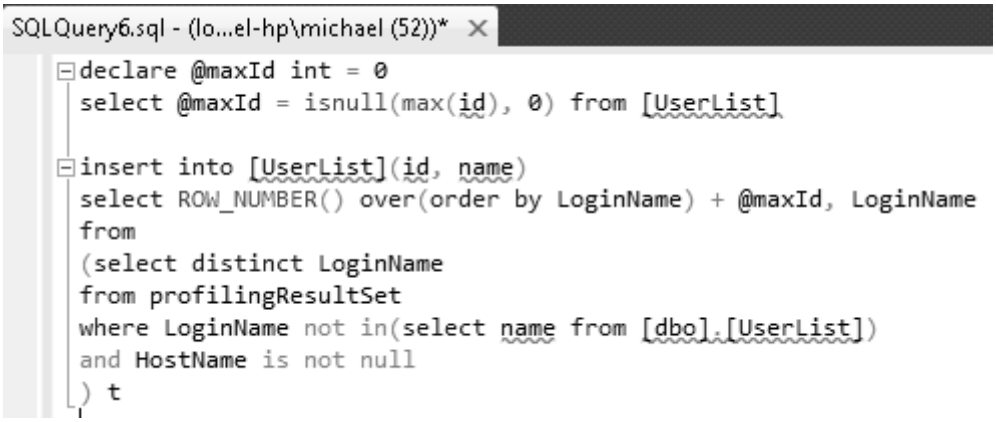


Рисунок 4.16 – T-SQL скрипт для оновлення даних списку користувачів

```
SQLQuery6.sql - (lo...el-hp\michael (52))*
declare @maxId int = 0
select @maxId = isnull(max(id), 0) from [ApplicationList]

insert into [ApplicationList](id, name)
select ROW_NUMBER() over(order by ApplicationName) + @maxId, ApplicationName
from
(select distinct ApplicationName
from profilingResultSet
where ApplicationName not in(select name from [dbo].[ApplicationList])
and ApplicationName is not null
) t
```

Рисунок 4.17 – T-SQL скрипт для оновлення даних списку програмного забезпечення

Базуючись на даних таблиць «Список ПЗ» та «Список робочих станцій», що заповнені за допомогою скриптів рис. 4.15 та рис. 4.17, а також таблиці «сирі дані», автоматично заповнюється вміст таблиці «ПЗ робочої станції». T-SQL скрипт для виконання зазначеної операції наведено на рис. 4.18.

```
SQLQuery6.sql - (lo...el-hp\michael (52))*
insert into [WorkstApplList](application_id, workstation_id)
select t.appId, t.wsId from
(select a.id appId, w.id wsId
from
(select distinct ApplicationName, HostName
from profilingResultSet
where ApplicationName is not null and HostName is not null) l
inner join [dbo].[ApplicationList] a on l.ApplicationName = a.name
inner join [dbo].[Workstation] w on l.HostName = w.name) t
left join [dbo].[WorkstApplList] wa on t.appId = wa.application_id and t.wsId = wa.workstation_id
where wa.application_id is null
```

Рисунок 4.18 – T-SQL скрипт для оновлення даних списку програмного забезпечення робочої станції

При реалізації серверної частини функціоналу використано елементи архітектури об’єктних БД на базі реляційної моделі даних. На рівні моделі сутність-зв’язок запропоновано використовувати три базові сутності, в яких представлені: список додаткових характеристик»; перелік можливих значень характеристик; та значення користувацьких характеристик для SQL-запитів [107]. Даний підхід дозволяє вирішувати задачу вводу нового аналітичного розрізу

обліку без необхідності зміни структури БД ІС. Фрагмент інфологічної моделі для підсистеми обліку користувацьких характеристик наведено на рис. 4.19.

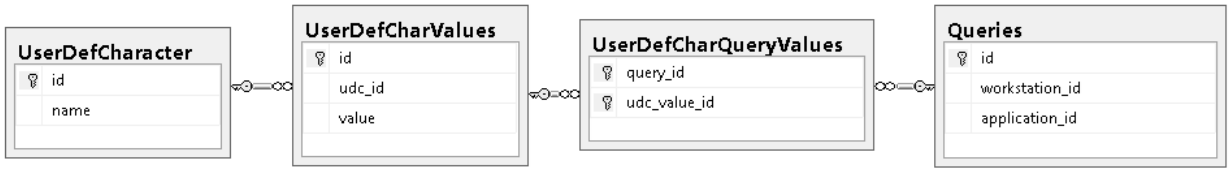


Рисунок 4.19 – Інфологічна модель підсистеми обліку користувацьких характеристик SQL-запиту

Вищенаведена реалізація розв'язує задачу ведення користувацьких аналітик лише на базі категоріальних значень. При необхідності використання інших типів значень виникає необхідність у додатковому розширенні структури БД ІС. Крім того, недоліком використання користувацьких класифікаторів є відхід від традиційної моделі «зірка» або «сніжинка», характерних для підсистем аналізу даних, що веде до помітного зниження швидкості аналітичних запитів до даних.

4.4 Багатовимірна БД обліку SQL-запитів

Враховуючи великі обсяги статистичних даних, які фіксують виконання операцій користувачів із БД ІС, а також необхідність виконання подальшого аналізу накопичених даних з точки зору множинності вимірів, прийнято рішення про представлення даних, необхідних для аналізу, у вигляді ББД [43; 75; 127; 147]. Для реалізації ББД обрано SQL Server Analysis Services - підсистему аналітики даних, яка використовується в прийнятті рішень і бізнес-аналітиці. При цьому, для візуалізації аналітичних даних можуть бути використані Power BI, MS Excel, звіти служб Reporting Services, а також інші інструменти візуалізації даних від сторонніх розробників.

Типовий робочий процес для Служби Analysis Services включає побудову моделі на основі багатовимірних або табличних даних, розгортання моделі як БД в екземплярі Служби Analysis Services, обробку БД для завантаження в неї даних

або метаданих і призначення дозволів на доступ до даних кінцевим користувачам [148; 149]. Після підготовки доступ до цієї багатоцільової семантичної моделі даних може здійснюватися будь-яким клієнтським застосунком, що підтримує служби Analysis Services як джерело даних [150].

Для реалізації таблиці фактів та таблиць вимірів у вигляді схеми «зірка», на рівні реляційної БД ІС обліку користувацьких запитів (детально описано у підрозділі 4.3) вводиться ряд представлень. Даний підхід підвищує прозорість структури БД для аналітиків та є своєрідним інтерфейсом при взаємодії багатовимірної моделі із реляційною. За умови використання такого підходу стає можливим внесення змін у реляційну модель БД без необхідності виконувати відповідні модифікації структури багатовимірної моделі за рахунок сталості переліку та типів значення полів створених представлень. Вищеописаний підхід є аналогом інкапсуляції в методології об’єктно-орієнтованого програмування [95–97].

Деякі з представлень не потребують додаткового огляду, оскільки виконані на основі однієї таблиці. Серед таких представлення, що використовуються при створенні вимірів програмних застосунків (ViewDimApplication), місць розташування (ViewDimPlace), атрибутів (ViewDimFields), відношень, або таблиць БД (ViewDimRelation), робочих місць (ViewDimWorkstation) та типів робочих місць (ViewDimType).

У порівнянні із ними, представлення, що утворюють таблиці фактів, мають набагато складнішу структуру. Так, у їх основі використано представлення, що повертає список складених запитів для поточного SQL-запиту із списку, що надійшли від користувацьких застосунків (рис. 4.20). Далі виконано представлення двох таблиць фактів для аналізу кількості запитів по атрибутах та кортежах відношення відповідно. Представлення для таблиці фактів по атрибутах відношення наведено на рис. 4.21. До складу запиту на вибірку даних, що його утворюють, входить 7 таблиць, серед яких список SQL-запитів, представлення вкладених підзапитів, довідники відношень та атрибутів, а також список атрибутів запиту.

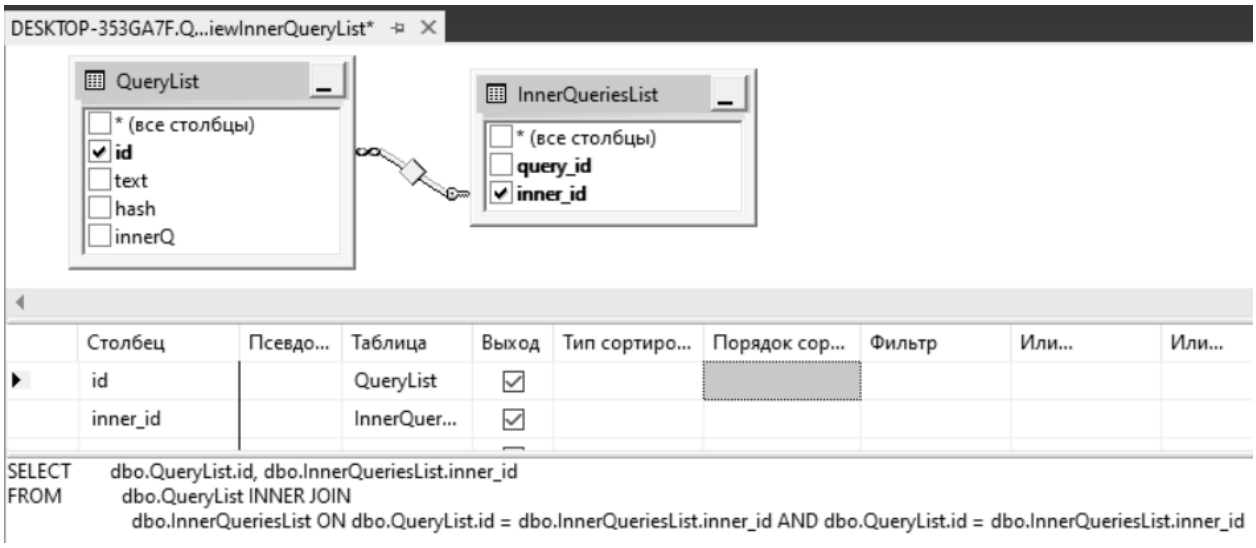


Рисунок 4.20 – Представлення ViewInnerQueryList

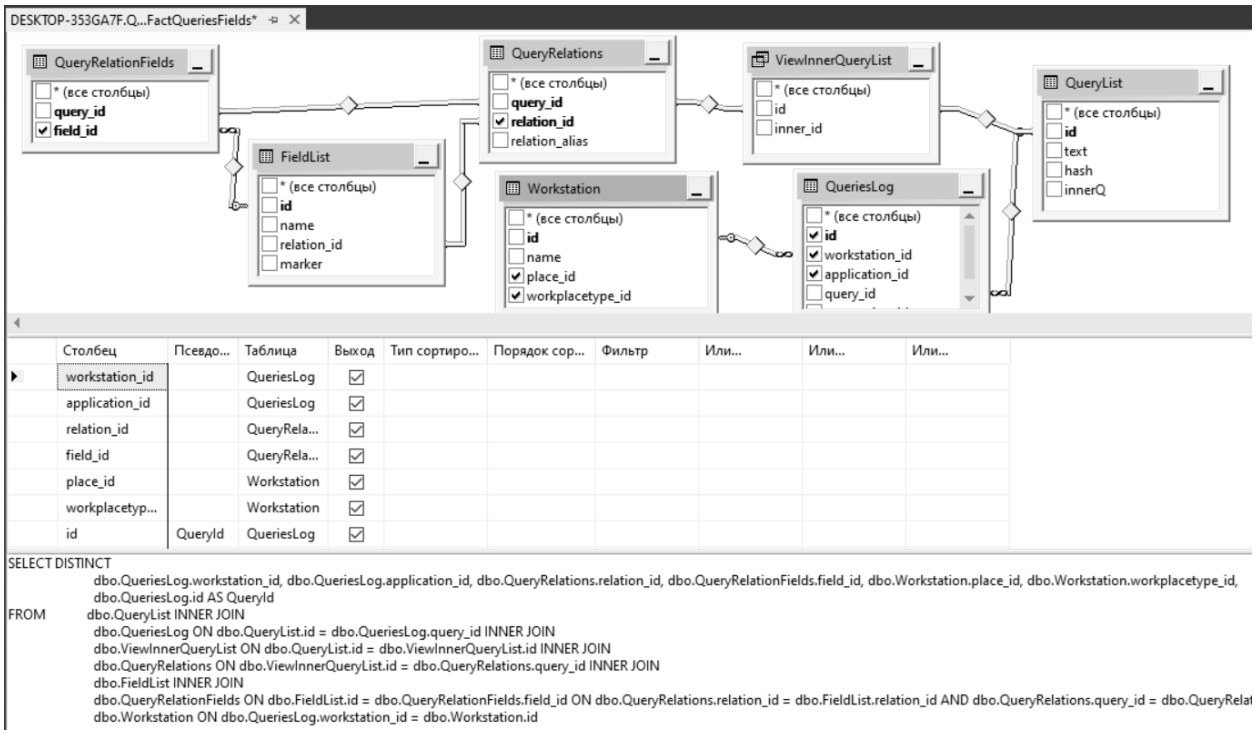


Рисунок 4.21 – Представлення ViewFactQueriesFields

Представлення для таблиці фактів по кортежах відношення має схожу структуру, за відміною входження до його складу списків рядків відношення та рядків запиту (рис. 4.22).

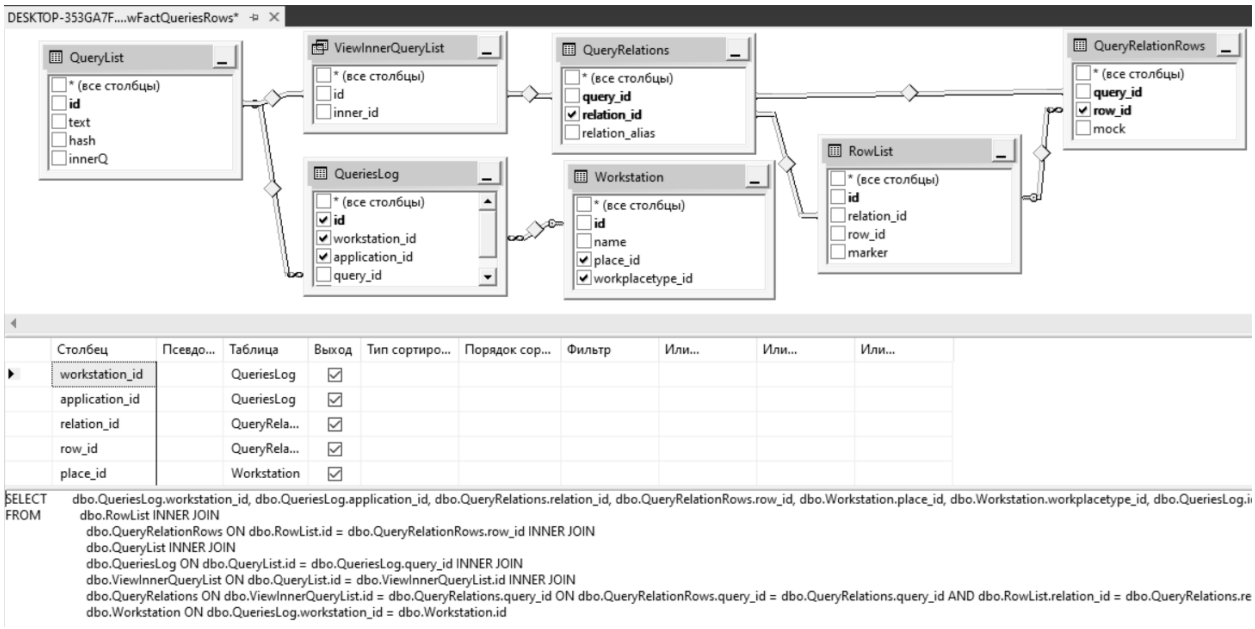


Рисунок 4.22 – Представлення ViewFactQueriesRows

Логічні первинні ключі представлень визначаються, як набір логічних зовнішніх ключів на представлення-виміри, а саме «workstation_id» – посилання на робочу станцію, «application_id» – на ПЗ, «relation_id» – на таблиці аналізуємої БД, «place_id» – місця розташування робочих станцій (рис. 4.21–4.22). Також для таблиці фактів по атрибутах – це посилання на список атрибутів «field_id», а для таблиці фактів по кортежах – це посилання на список кортежів «row_id». Вимір мір представлений єдиним значенням – це ідентифікатор запиту, що дозволить у майбутньому підрахувати значення рівня маркеру представленості відповідно до запитів, що надходять до БД, відфільтрованих відповідно до значень аналітик, що вказані при виконанні операцій зрізу.

Наступним етапом є створення самої багатовимірної моделі. Дана частина виконується у ПЗ Visual Studio із використанням пакету SQL Server Development Tools for Business Intelligence. В проєкті Analysis Services Multidimensional and Data Mining Project створюється підключення (Data Source) до реляційної БД підсистеми обліку запитів та визначається його представлення (Data Source View), у яке входять створені вище представлення рівня реляційної БД, що відповідають за таблиці вимірів та таблиці фактів багатовимірної моделі (рис. 4.23).

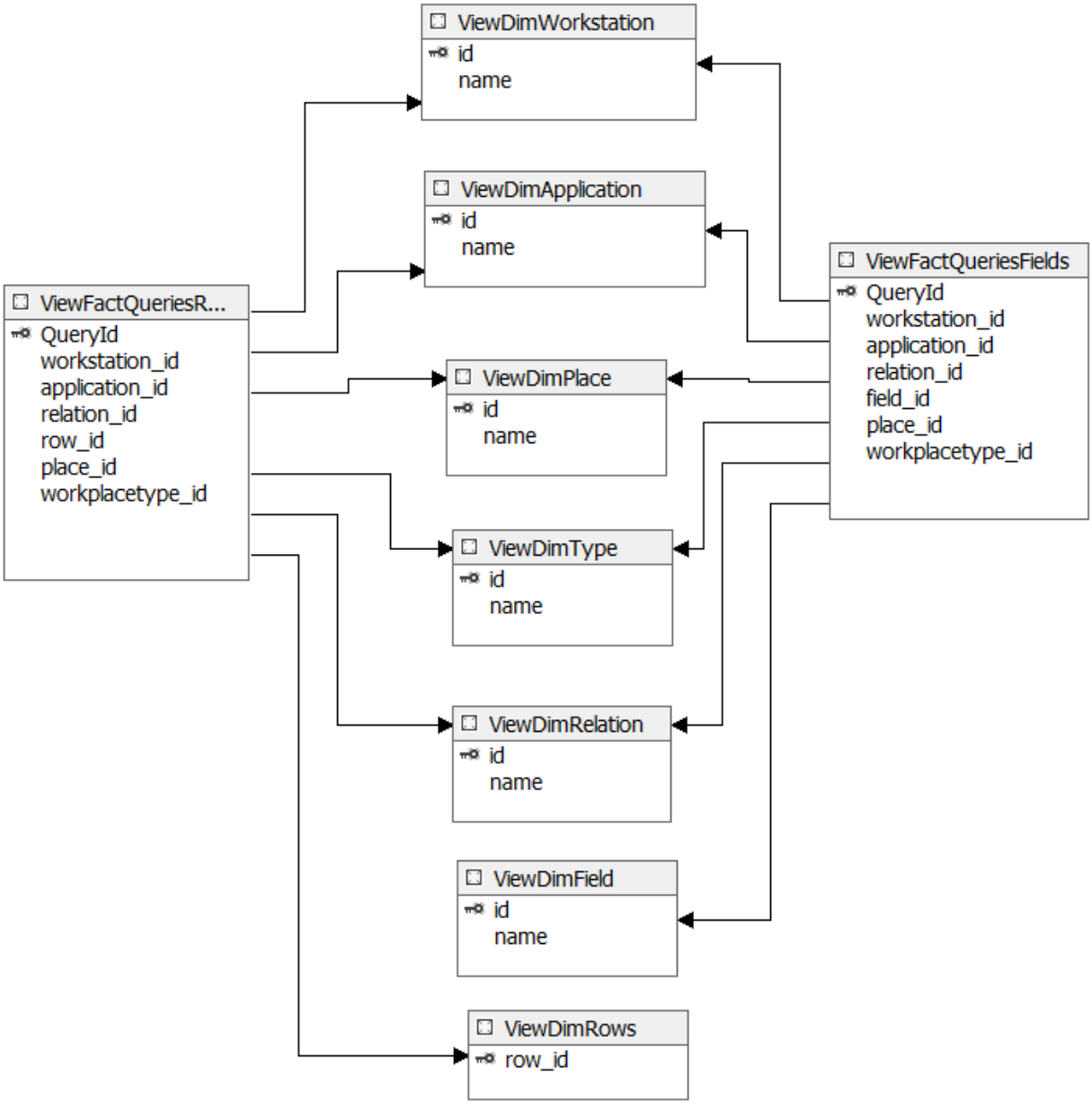


Рисунок 4.23 – Data Source View підсистеми оперативно-аналітичного аналізу запитів

Структура всіх вимірів багатовимірного кубу є типовою. Вона містить один описовий атрибут «назва» та унікальний ідентифікатор об’єкту, що є первинним ключем для відношення. Відповідно, на рис. 4.24 наведено тільки структура виміру «Application» (програмне забезпечення), всі інші відрізняються лише назвами представлень, що використовуються (та можливо назвами полів). Після визначення вимірів моделі створюється куб, що об’єднує набір вимірів із виміром мір. Куб, що характеризує кількість запитів в аналітиці атрибутів відношення, має

в своїй основі схему «зірка», одну таблицю фактів та по одній таблиці на кожний вимір ББД (рис. 4.25).

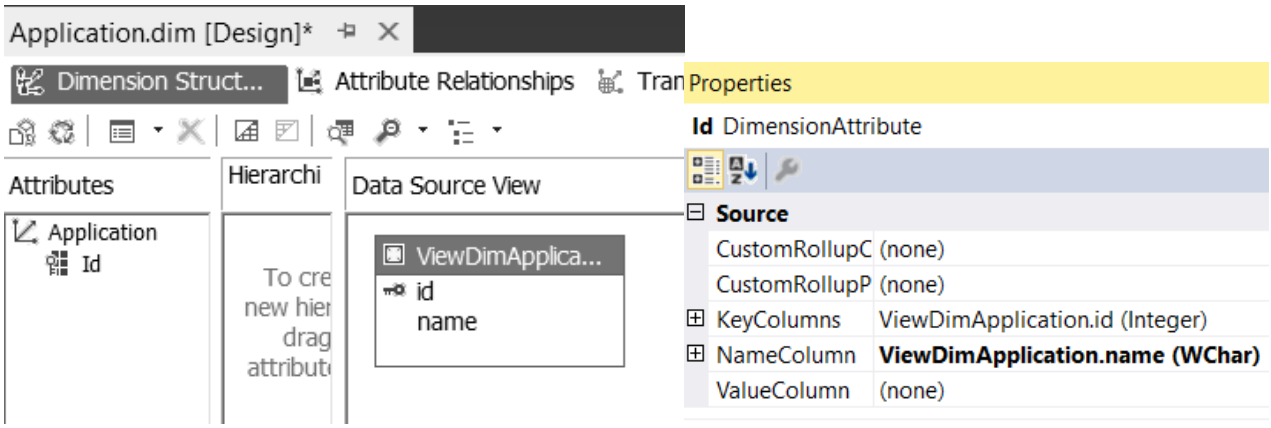


Рисунок 4.24 – Структура виміру «Програмне забезпечення»

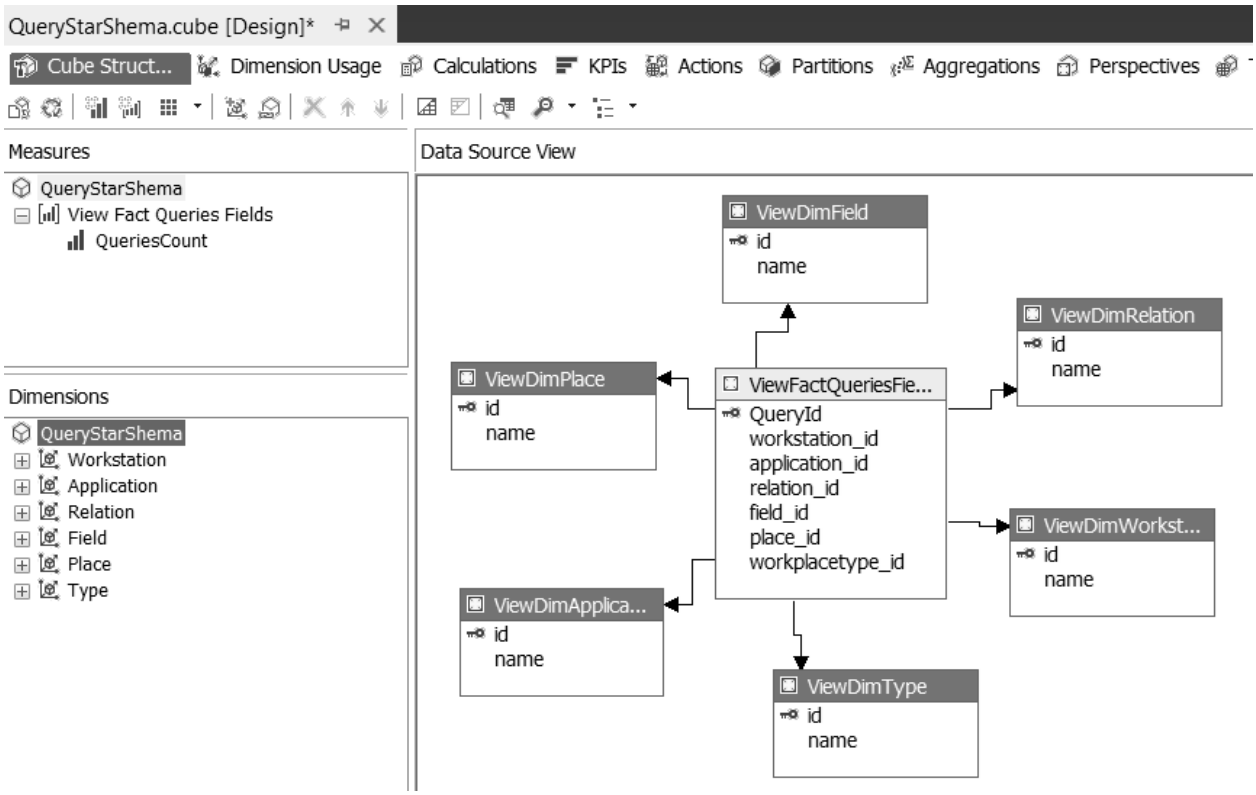


Рисунок 4.25 – Структура кубу QueryFieldsCube

При створенні міри, використано функцію агрегації підрахунку кількості значень без повторень по полю унікального ідентифікатора журналу SQL-запитів (рис. 4.26).

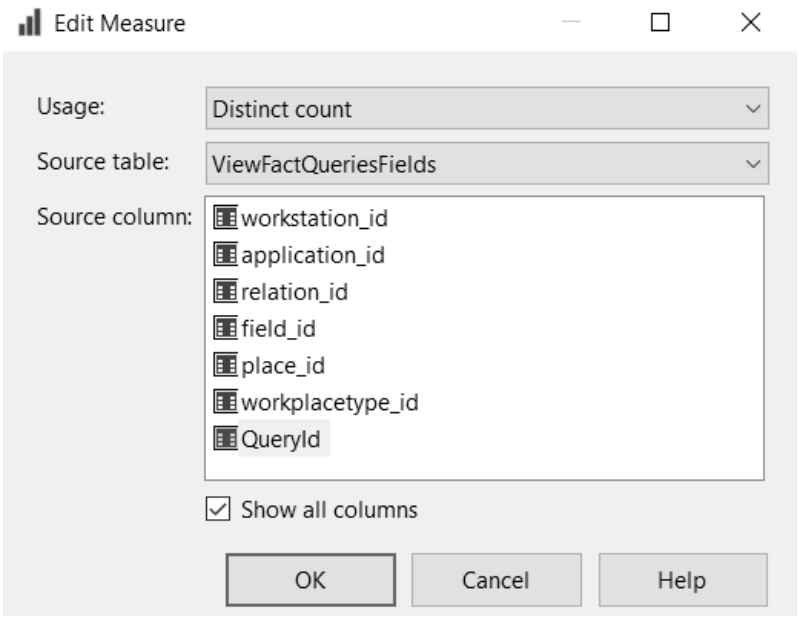


Рисунок 4.26 – Міра «кількість запитів» кубу QueryFieldsCube

Куб для роботи із рядками відношень, до яких виконуються звернення у межах SQL-запитів, має ідентичну до рис. 4.25 структуру, за винятком виміру «атрибут відношення», що замінюється виміром «кортеж відношення» (рис. 4.27).

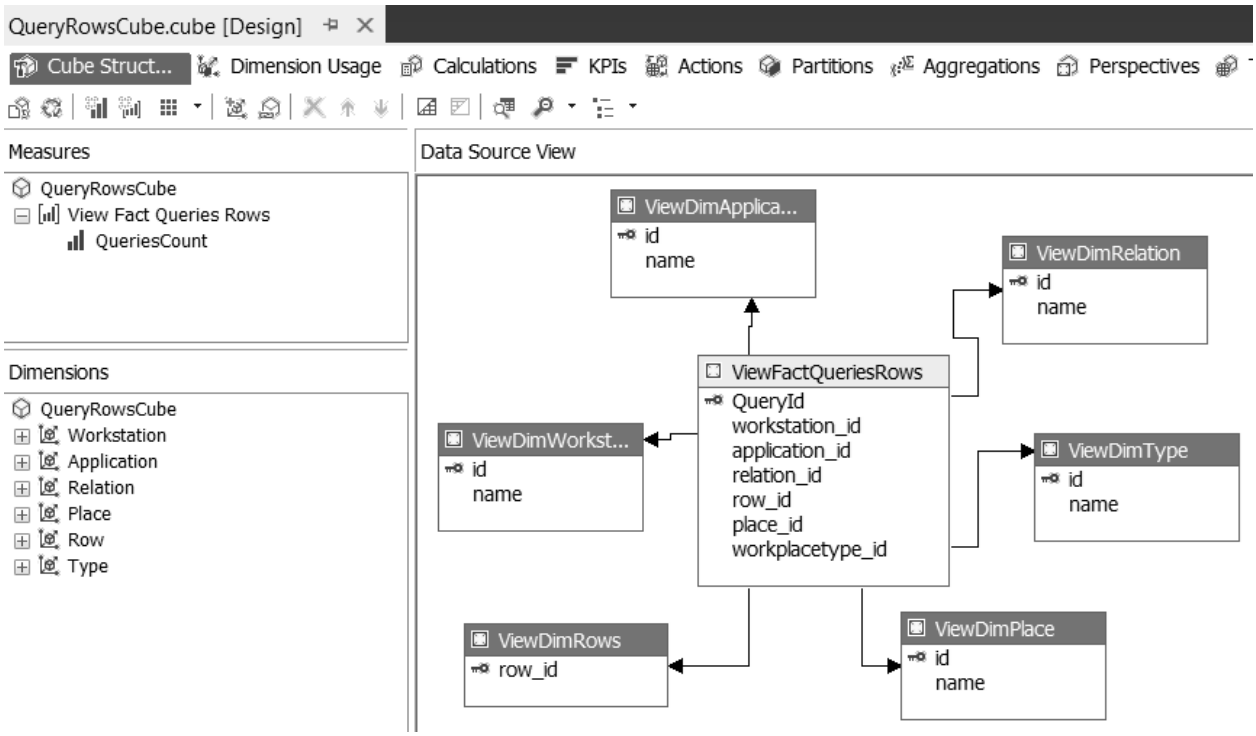


Рисунок 4.27 – Структура кубу QueryRowsCube

4.5 Отримання зрізів даних ББД

Створена ББД використовується для подальшого оперативно-аналітичного аналізу даних. У якості клієнтського застосунку може буди використане будь-яке програмне середовище, що має у своєму розпорядженні механізми по підключенню та отриманню даних від SQL Analysis Services починаючи від MS Excel і завершуючи розробкою власного ПЗ. В межах поточного дослідження використано MS Excel, що надає достатній рівень функціональних можливостей для побудови зрізів даних, необхідних ЛПР для прийняття рішення щодо подальшої фільтрації даних при розрахунку значень критеріїв оптимальності структури вузла РКІС.

Після виконання підключення до OLAP-серверу, отримуються відповідні зрізи даних ББД у вигляді зведеної таблиці. Так, наприклад, на рис. 4.28 наведено таблицю, що ілюструє кількість запитів до відношень БД в залежності від локації. Також використані фільтри за програмним забезпеченням, робочою станцією та типом робочого місця (рис. 4.29).

Workplace type name	(несколько элементов)				
Workstation name	All				
Application name	(несколько элементов)				
QueriesCount	**Названия столбцов**				
Названия строк	**Офіс Херсон**	**Торговий майданчик Соляні**	**Торговий майданчик Вознесенськ**	**Центральний офіс**	**Общий итог**
TOVAR	1898	94	230	3062	5284
RABFOLD	12		382	2952	3346
SKLAD	83	6	192	2920	3201
VARCST_ent				2304	2304
actionsod	895	30	24	595	1544
KLIENT	6		233	1261	1500
actionfold	822	22	24	595	1463
RABSOD	23		382	387	792
GROUPTOV	50	1	16	646	713
KART			149	120	269
MANAGER			67	168	235
UPAK				227	227
SKLGROUP				219	219
specSod			165	8	173
specSag			165	8	173
temp_zakazSod			101		101
TEMP_VERSIONS	28	7	8	53	96
TEMP_DISPERSON		92			92
klt_ent				77	77
ZAKAZSOD			71		71
ZAKAZSAG			67		67
DISCONT		66			66
PRNAME				53	53

Рисунок 4.28 – Таблиця кількості запитів до окремих відношень БД

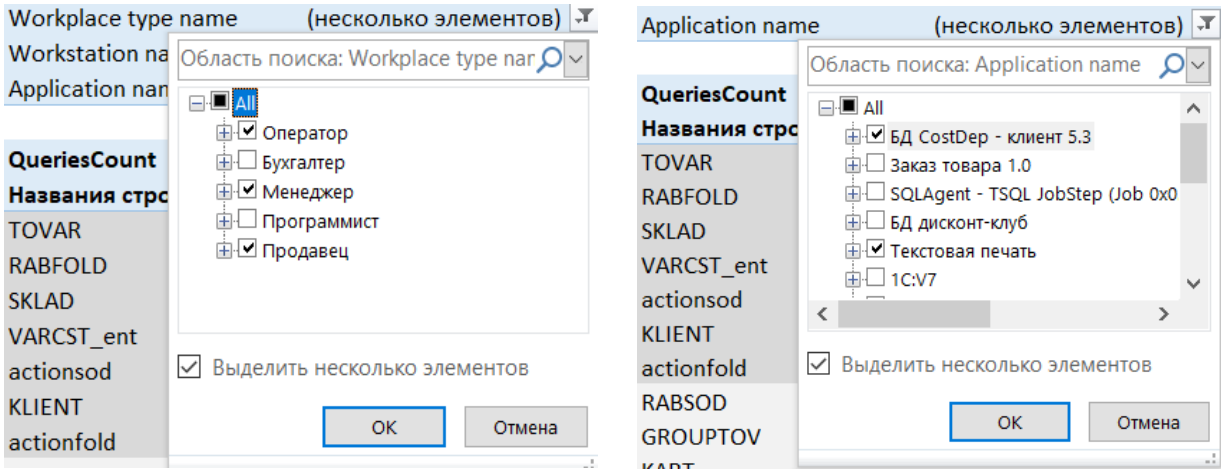


Рисунок 4.29 – Використання фільтрації даних

При виконанні більш глибокого аналізу на рівні атрибутів відношення, ЛПР має можливість виконати операцію деталізації та отримати дані щодо кількості звернень SQL-запитів до окремих полів таблиці. Отримані результати можуть бути відображені як у табличній формі, так і у формі графіків або діаграм (рис. 4.30).

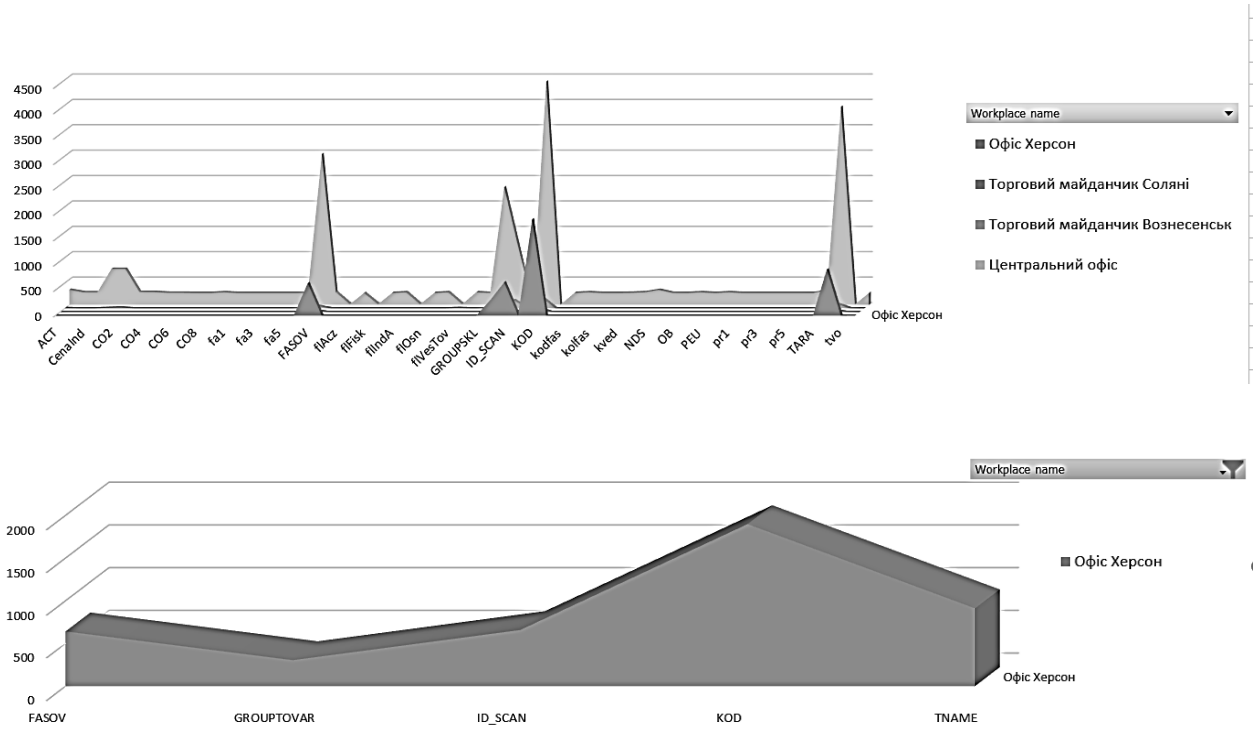


Рисунок 4.30 – Кількість запитів до окремих полів таблиць БД

Розроблена ББД дозволяє ЛПР виконувати оперативно-аналітичний аналіз даних з метою виявлення оптимального набору необхідного на вузлі РКІС програмного забезпечення. Виконання он-лайн операцій консолідації, деталізації та зрізу дозволяє визначити множини атрибутів та кортежів відношень БД, до яких будуть виконуватись SQL-запити КІС.

Висновки до розділу 4

Розроблена функціональна модель інформаційної технології дозволяє виділити чотири основні задачі, серед яких аналіз тексту та результату SQL-запиту, а також облік SQL-запитів КІС до БД. Після накопичення статистичних даних за допомогою профайлінгових утиліт СКБД, виконується їх аналіз із метою визначення аналітичних характеристик запиту. За допомогою стандартних механізмів СКБД моніторингу користувацької активності виділяється інформація щодо хоста, програмного забезпечення та користувача, від яких надійшов запит.

Далі тексти SQL-запитів розбираються на складові із використанням апарату формальних граматик із метою виявлення списку відношень та атрибутів, до яких відбувається звернення у межах запиту. Після цього кожен запит розбивається на множину підзапитів, кількість елементів у якій відповідає кількості відношень запиту. У кожному запиті даної множини набір атрибутів замінюється набором полів первинного ключа кожного відношення. Це робиться з метою виявлення діапазонів ключів кожного відношення, та визначення множини кортежів відношення, що задіяні у запиті.

Спроєктована структура реляційної БД обліку статистичних даних SQL-запитів КІС та на її основі побудована ББД, що дозволяє ЛПР проводити оперативно-аналітичний аналіз з метою виявлення множини атрибутів та кортежів відношення, до яких відбуваються звернення РКІС із виконанням відповідних операцій консолідації, деталізації та зрізу даних.

РОЗДІЛ 5
ВИЗНАЧЕННЯ ПРЕДСТАВЛЕНОСТІ ДАННИХ НА ВУЗЛІ РКІС, ЯК СКЛАДОВА ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ ОПТИМІЗАЦІЇ СТРУКТУРИ БД

5.1 Формалізація обмежень за критеріями оптимальності

Задача вибору найкращої альтернативи для рівня маркеру представленості даних на віддаленому вузлі розподіленої КІС включає чотири основних етапи, а саме: формалізацію критеріїв оптимальності структури БД; визначення оптимального рівня маркеру представленості даних; класифікацію атрибутів та кортежів відповідно по необхідності представлення на вузлі КІС та класифікацію нових даних відношення відповідно до необхідності представлення у вузлі РКІС (рис. 5.1).

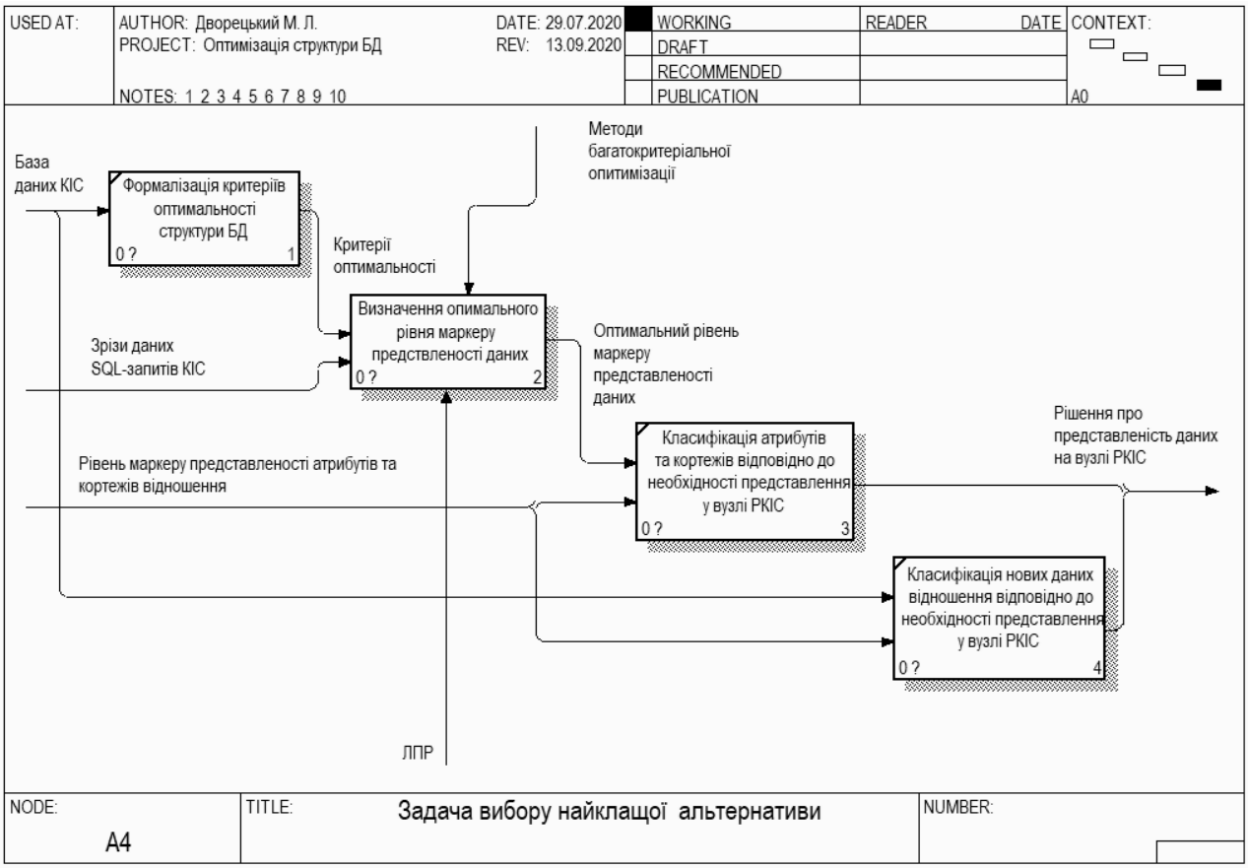


Рисунок 5.1 – Функціональна модель задачі вибору найкращої альтернативи (2-й рівень ієрархії, A4)

Перша із перерахованих задач була детально розглянута у підрозділі 2.3 і надає на виході математичні моделі критеріїв оптимальності структури БД у вигляді залежності від рівня маркеру представленості даних (2.10), (2.14) та (2.18).

Декомпозиція задачі визначення оптимального рівня маркеру представленості даних дає можливість визначити наступні її етапи: створення ієрархічної структури; формалізація обмежень за критеріями оптимальності; заповнення матриці попарних порівнянь та розрахунок вектору глобальних пріоритетів та дефазифікація одержаних результатів (рис. 5.2).

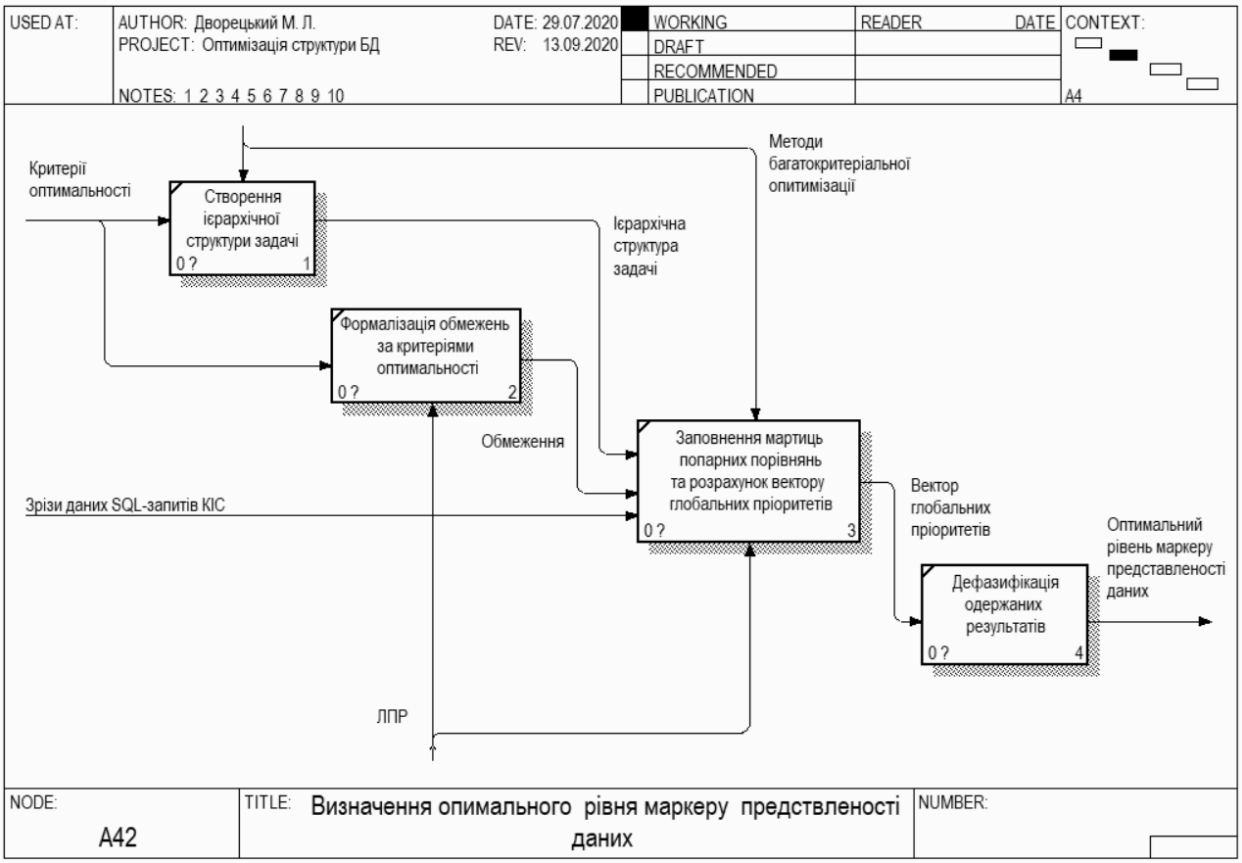


Рисунок 5.2 – Декомпозиція задачі визначення оптимального рівня маркеру представленості даних (3-й рівень ієрархії, A42)

Етап створення ієрархічної структури було розглянуто у підрозділі 3.1 та наведено на рис. 3.1 та рис. 3.2. Маючи три критерія оптимальності структури БД вузла РКІС та п’ять альтернатив щодо рівня маркеру представленості даних вводедено обмеження за критеріями оптимальності, визначивши мінімальний та максимальний рівень допустимих значень.

Програмна реалізація частини ІТ щодо визначення оптимального рівня маркеру представленості даних на вузлі РКІС виконана у вигляді сервіс-орієнтованого вебзастосунку на базі триланкової архітектури [151–153] (рис. 5.3).

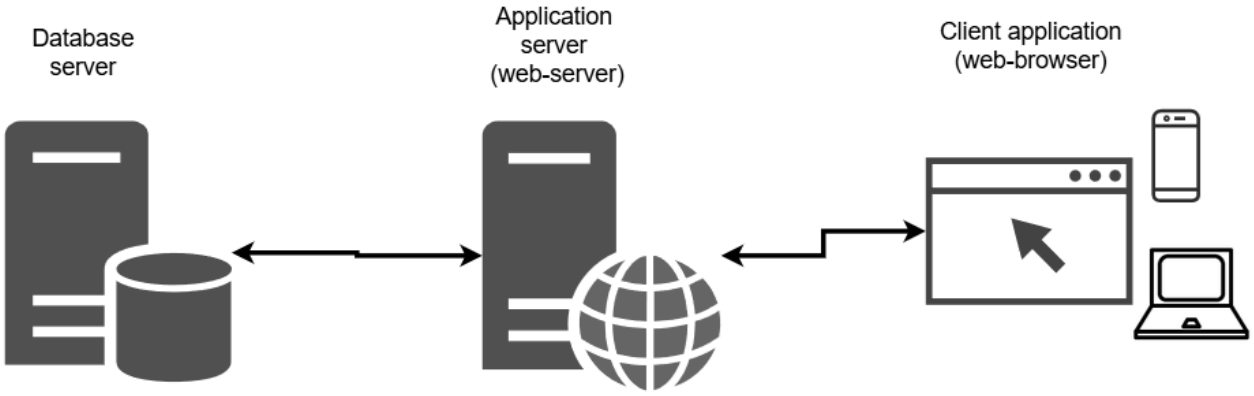


Рисунок 5.3 – Триланкова архітектура вебзастосунку

У якості сервера БД обрано SQL Server, бекенд-складова (Application Server) реалізована на базі фреймворку Symfony [154; 155] із використанням скриптової мови PHP. Взаємодія із БД реалізована із використанням Object-Relational Mapper (ORM) Doctrine [156–158], що дозволяє оперувати із даними БД за допомогою об’єктної моделі. Даний прошарок взаємодіє із сервером БД за допомогою команд мови SQL та повертає екземпляри класів сутностей у середовище бекенд. Фронтенд-складова реалізована на базі фреймворку Angular [155; 159; 160] та мови програмування TypeScript (рис. 5.4).

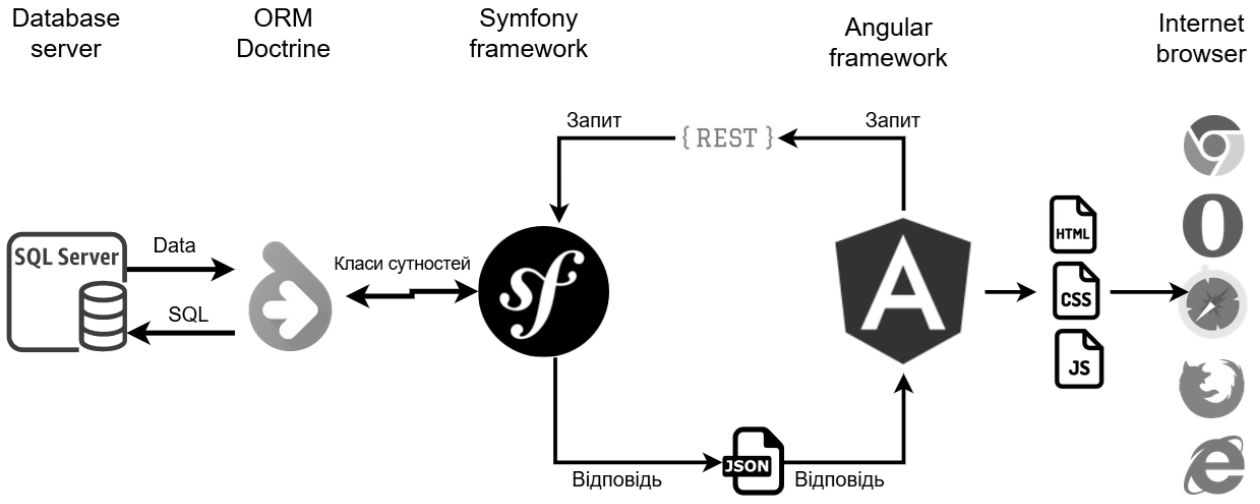


Рисунок 5.4 – Взаємодія елементів розробленої сервіс-орієнтованої ІТ

Бекенд- та фронтенд-компоненти взаємодіють через REST-API із обміном даними у json-форматі. Наведена архітектура є достатньо гнучкою та дозволяє за необхідності змінювати СКБД та підключатись до бекенд API, використовуючи стороннє ПЗ.

При реалізації обмежень за критеріями оптимальності на рівні бекенд-компоненти реалізован клас-сутність «Критерій оптимальності» із полями «Назва», «Мінімально допустимий рівень» та «Максимально допустимий рівень». Використовуючи механізми фреймворку Symfony та ORM Doctrine, після створення наведеного класу-сутності, таблиця БД генерується командою «bin/console doctrine:schema:update --force». Клас розширює інтерфейс «JsonSerialisable» для можливості експорту даних у JSON-форматі. Клас «Критерій оптимальності» та структуру таблиці БД наведено на рис. 5.5.

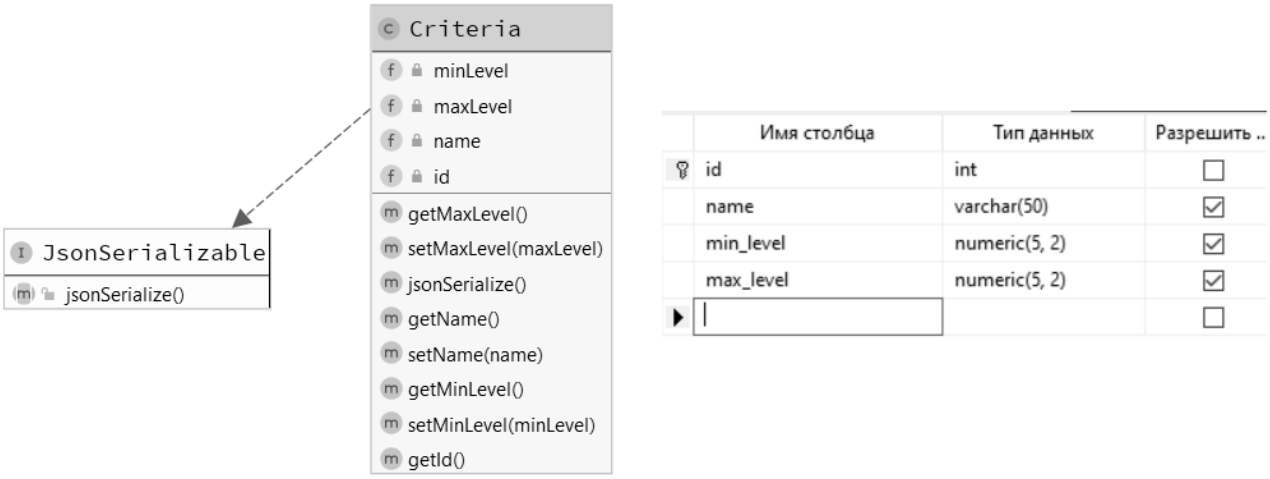


Рисунок 5.5 – Клас «Критерій оптимальності» та відповідна таблиця БД

Бекенд-компонента реалізує маршрути для отримання необхідних даних щодо критеріїв оптимальності. Так, звернення до маршруту /criteria/get-criteria/{criteriaId}, де {criteriaId} – ідентифікатор критерію ефективності, повертає його дані та розраховані значення для кожної альтернативи у json-форматі (рис. 5.6).

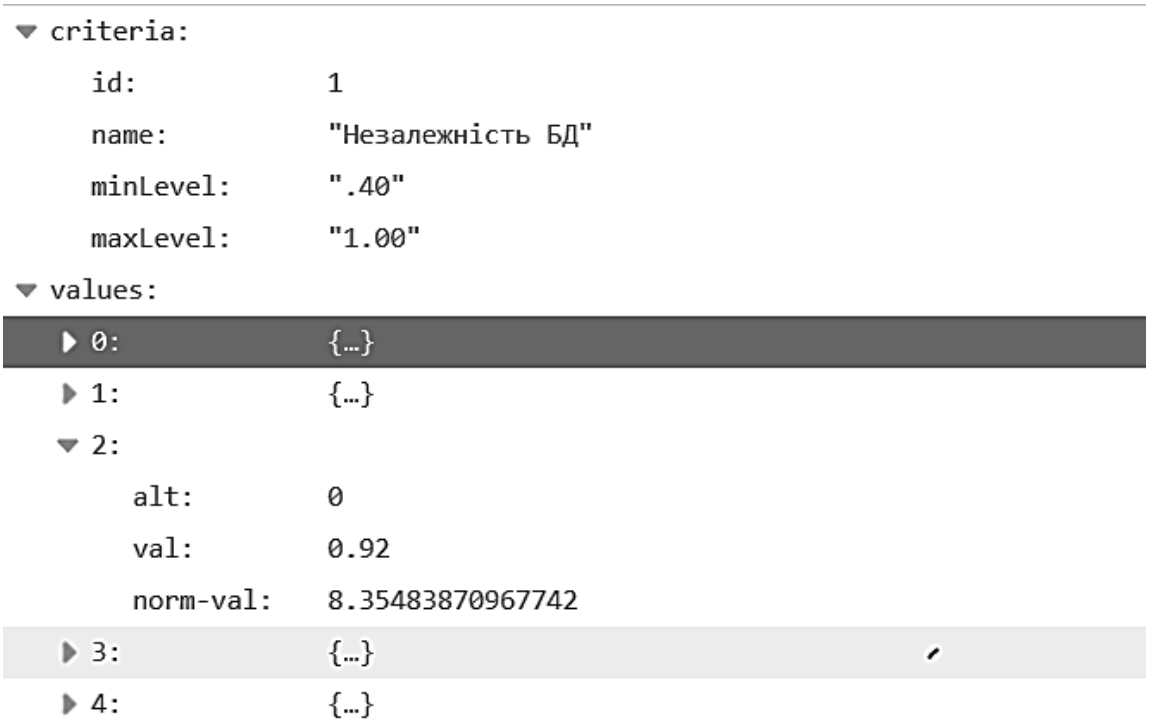


Рисунок 5.6 –Дані щодо критерію ефективності у json-форматі

Дані по критеріях оптимальності «Незалежність БД» та «Розмір БД» наведені на рис. 5.7.

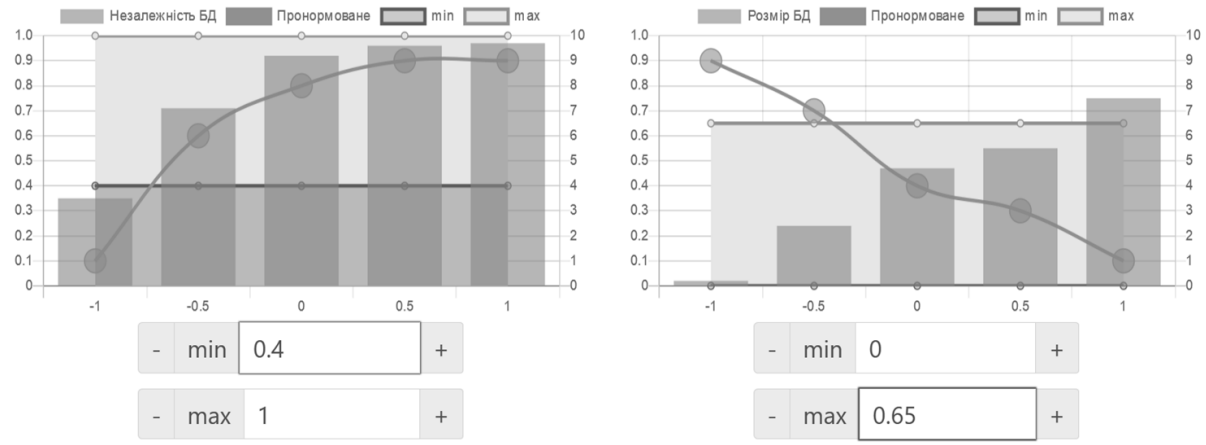


Рисунок 5.7 – Дані щодо критеріїв оптимальності «Незалежність БД» та «Розмір БД»

Для надання всієї необхідної інформації щодо критеріїв оптимальності та встановлених обмежень на бекенд реалізовано ще ряд маршрутів, перелік яких наведено у табл. 5.1.

Таблиця 5.1 – Перелік маршрутів бекенд-компоненти для роботи із критеріями оптимальності

URI	Метод / Призначення	Параметри /тіло запиту	Структура відповіді (json)	Описання відповіді
/criteria/get-criteria/ {criteriaId}	GET / Отримання даних щодо критерію оптимальності	criteriaId: number – ідентифікатор критерію оптимальності	{criteria: {id: number, name: string, minLevel: number, maxLevel: number}, values: [{alt: number, val: number, norm-val: number}] }	Назва критерію ефективності, обмеження (мінімальний та максимальний допустимий рівні значення), значення та нормоване значення критерію (3.3) для кожної альтернативи
/criteria/set-min-max	POST / Зміна значень мінімального та максимального рівня критерію оптимальності	Body: {code: number, min: number, max: number} – ідентифікатор критерію, мін. та макс. допустимі значення	{value: [{alt:number, disable: boolean}] }	Масив альтернатив для рівня маркеру представленості даних із зазначенням допустимості альтернативи відповідно до діючих обмежень
/criteria/get-alternative	GET / Отримання даних масиву альтернатив, що задовольняють обмеженням за критеріями		{value: [{alt:number, disable: boolean}] }	Масив альтернатив для рівня маркеру із зазначенням допустимості альтернативи відповідно до діючих обмежень

Масив допустимих альтернатив із урахуванням обмежень за критеріями оптимальності наведений на рис. 5.8.

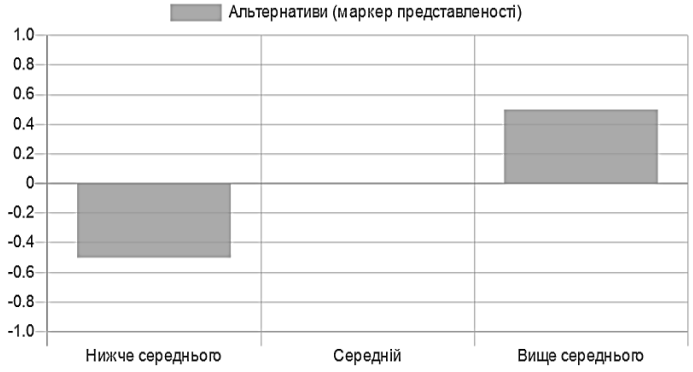


Рисунок 5.8 – Масив допустимих альтернатив із урахуванням обмежень за критеріями оптимальності

Наведений функціонал реалізовано у компоненті «CriteriaChartsComponent», що відповідає за логіку обробки та виведення даних бекенд; та двох сервісах: «CriteriaService» та «CompareMatrixService», у яких реалізована логіка взаємодії із бекенд-частиною застосунку. Діаграма класів для фронтенд-складової реалізації задачі обмежень за критеріями оптимальності наведено на рис. 5.9.

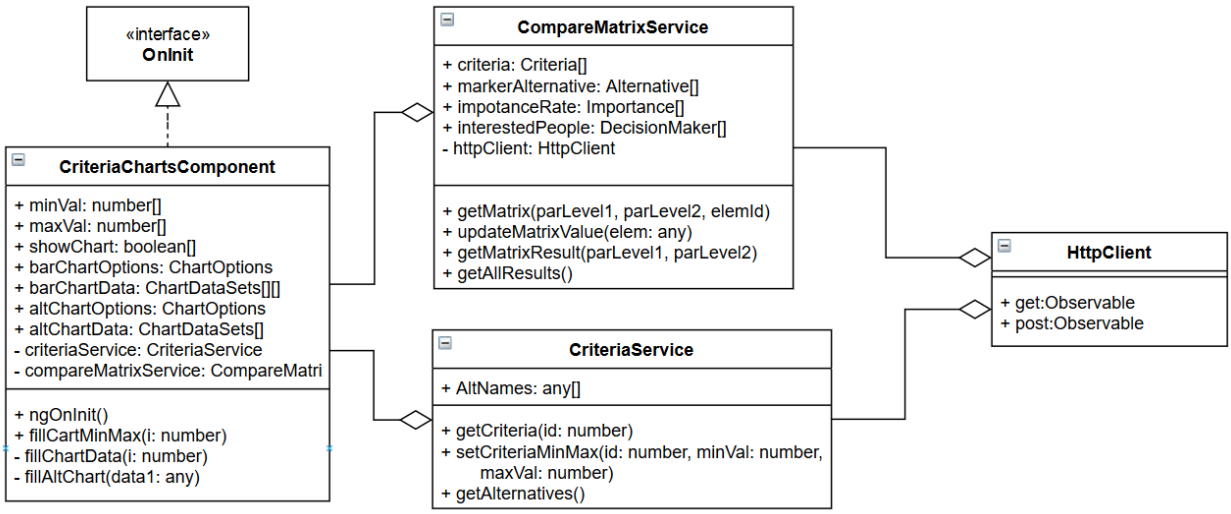


Рисунок 5.9 – Діаграма класів. Компонента CriteriaChartsComponent та сервіси CriteriaService та CompareMatrixService

5.2 Робота із матрицею переваг 1-го та 2-го рівня ієрархії

Робота із матрицею переваг здійснюється на 3 рівнях ієрархічної моделі, описаної у підрозділі 3.1 та наведеної на рис. 3.1–3.2. На 1-му рівні ієрархії виконується оцінка переваг зацікавлених осіб, якими виступають «Власник», «Адміністратор БД», «Розробник КІС» та «Оператор КІС». На рівні ORM Doctrine та серверу БД реалізовано один клас сутності та відповідну таблицю БД для роботи із матрицями попарних порівнянь на всіх рівнях ієрархії (рис. 5.10). Можливість використання однієї таблиці для зберігання даних по матрицях попарних порівнянь для всіх рівнів ієрархії моделі досягається за рахунок введення атрибутів «par_level1» та «par_level2». Так, елементи, що відповідають умові «par_level1 = 2 and par_level2 = 0», відносяться до матриці 2-го рівня ієрархії (par_level2 = 0) по зацікавленій особі із кодом «2» (par_level1 = 2).

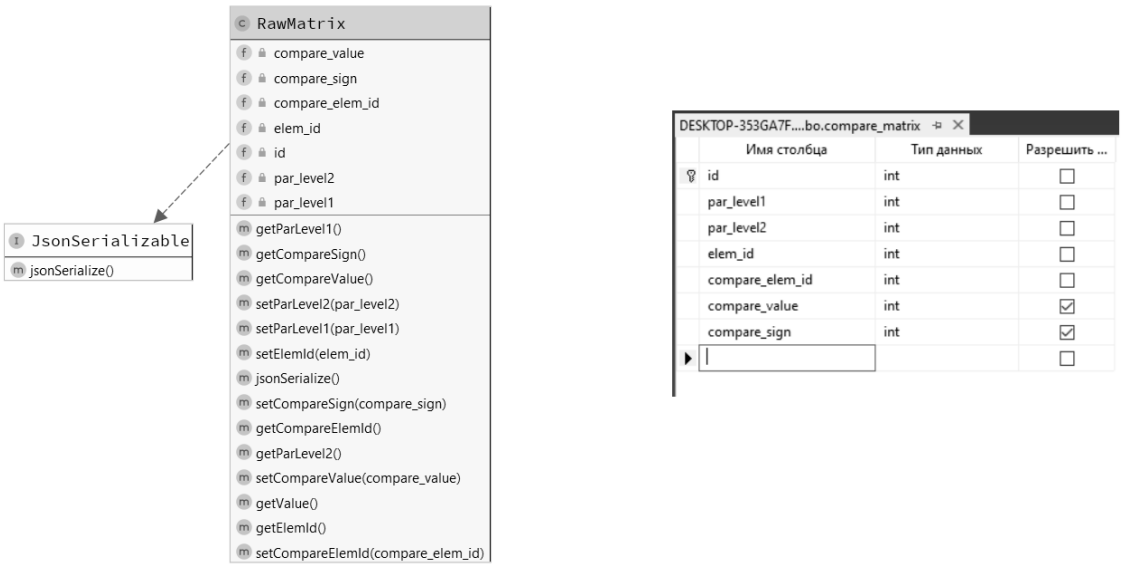


Рисунок 5.10 – Клас «Елемент матриці попарних порівнянь» та відповідна таблиця БД

Зі сторони бекенд-частини інформаційної технології реалізовано клас-контролер «MatrixController» для обслуговування маршрутів та класи сервісів «MatrixManager» та «AlternativeManager» для реалізації додаткової логіки обробки даних (рис. 5.11). Клас контролеру «MatrixController» реалізує ряд маршрутів для роботи із даними матриць попарних порівнянь, перелік яких наведено у табл. 5.2.

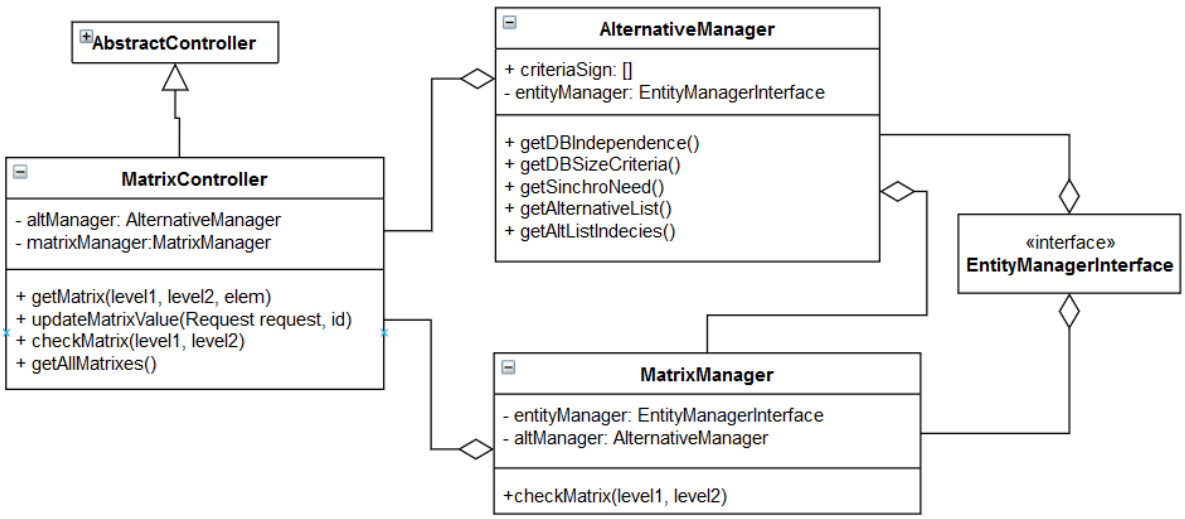


Рисунок 5.11 – Діаграма класів. Контролер MatrixController та сервіси MatrixManager та AlternativeManager

Таблиця 5.2 – Перелік маршрутів бекенд-компоненти для роботи із матрицями попарних порівнянь

URI	**Метод / Призначення**	**Параметри /тіло запиту**	**Структура відповіді (json)**	**Описання відповіді**
/compare-matrix/{level1} /{level2} /{elem}	GET / Отримання даних рядку матриці попарних порівнянь для елементу elem	level1: number – ідентифікатор зацікавленої особи (якщо 0 – 1 рівень ієрархії); level2: number – ідентифікатор критерію ефективності (якщо 0 – 2 рівень ієрархії), elem: number – ідентифікатор елементу	[{id: number, element: number, compareElement: number, value: number}]	Масив комірок певного рядку матриці попарних порівнянь. Елемент масиву складається з ідентифікатору комірки, ідентифікатору першого елементу (зацікавленої особи, критерію, альтернативи), ідентифікатору другого елементу та значення переваги першого елементу над другим
/compare-matrix/{id}	POST / Зміна комірки мартиці попарних порівнянь	id: number – ідентифікатор комірки матриці попарних порівнянь / Body: { value: number, sign: number }	[{id: number, element: number, compareElement: number, value: number}]	Масив із двох комірок матриці попарних порівнянь, що є симетричними відносно головної діагоналі та зазнають змін у наслідок операції модифікації однієї з них
/check-matrix/{level1}/ {level2}	GET / Розрахунок вектору пріоритетів та індексу узгодженості для матриці попарних порівнянь	level1: number – ідентифікатор зацікавленої особи (якщо 0 – 1 рівень ієрархії); level2: number – ідентифікатор критерію ефективності (якщо 0 – 2 рівень ієрархії)	{ vectorVal: {elem1: number, elem2: number, ... elemN: number}, indexVal: number }	Вектор пріоритетів, де elem1 – elemN відповідає ідентифікаторам елементів (зацікавленої особи, критерію, альтернативи) та індекс узгодженості матриці попарних порівнянь (якщо > 10 %, то необхідне корегування матриці)

Фронтенд-складова, що реалізує логіку відображення даних матриці попарних порівнянь зацікавлених осіб та взаємодію із бекенд-складовою, реалізована у вигляді компоненти «PeopleMatrixComponent». Оскільки існує ряд

компонент, що відповідають за виведення матриць попарних порівнянь для інших рівнів ієрархії, чия поведінка не суттєво відрізняється від «PeopleMatrixComponent», дана компонента та ряд інших, що будуть розглянуті пізніше, реалізовані на базі компоненти «CompareMatrixComponent» (рис. 5.12). Аналогічно компоненту «CriteriaChartsComponent», що наведений на рис. 5.9, «CompareMatrixComponent» використовує сервіси «CriteriaService» та «CompareMatrixService», у яких реалізована логіка взаємодії із бекенд-частиною застосунку.

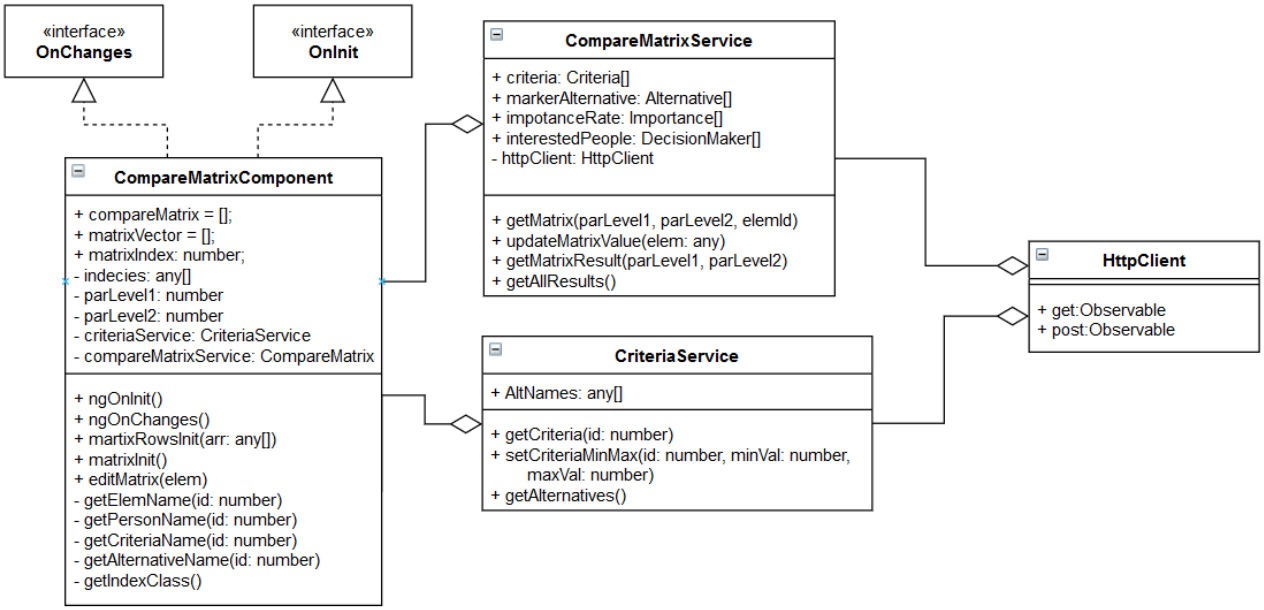


Рисунок 5.12 – Діаграма класів. Компонента CompareMatrixComponent та сервіси CriteriaService та CompareMatrixService

Відображення даних реалізоване на базі адаптивного інтерфейсу та дозволяє виконувати заповнення та редагування матриці попарних порівнянь на пристроях із різною роздільною здатністю (рис. 5.13).

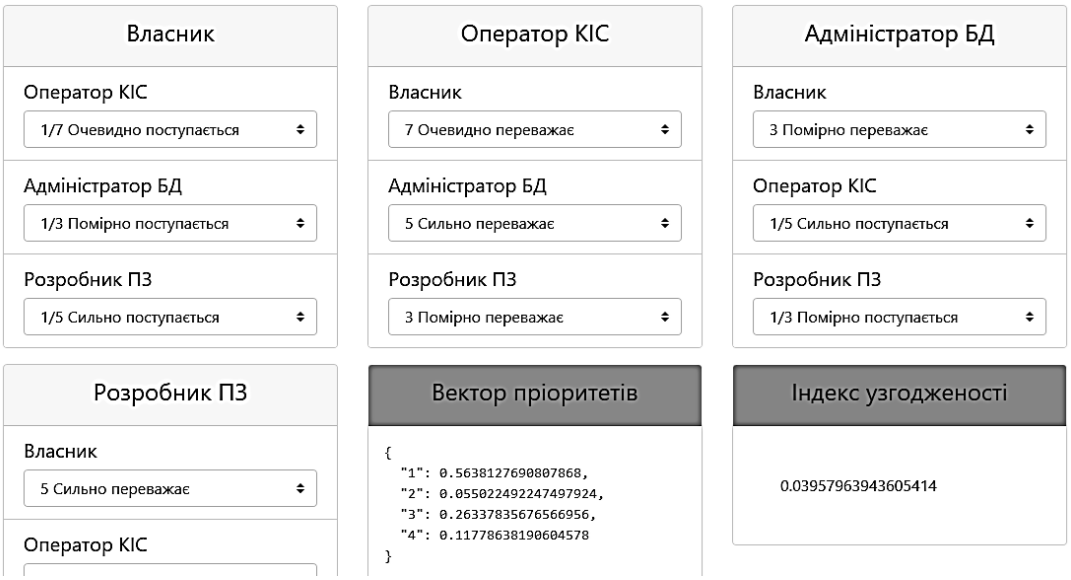


Рисунок 5.13 – Робота із матрицею попарних порівнянь зацікавлених осіб. Користувацький інтерфейс

При модифікації значення переваг елементів матриці динамічно оновлюються значення індексу узгодженості та вектору пріоритетів, що дає можливість ЛПР відразу оцінити наслідки свого рішення.

Універсальність реалізації компонента «CompareMatrixComponent» досягається за допомогою вхідних параметрів «parLevel1» та «parLevel2», аналогічно до схожих полів класу «Елемент матриці попарних порівнянь» бекенд-складової. Так, при реалізації компонента «PeopleMatrixComponent» для виведення даних матриці попарних порівнянь зацікавлених осіб, достатньо реалізувати html-шаблон наступного вмісту: <app-compare-matrix [parLevel1]="0" [parLevel2]="0"></app-compare-matrix>, або враховуючи наявність значень вхідних параметрів за замовченням, <app-compare-matrix></app-compare-matrix>.

Для роботи із матрицями попарних порівнянь критеріїв оптимальності моделі за кожною зацікавленою особою, реалізовано компонент «CriteriaMatrixComponent», що аналогічно із «PeopleMatrixComponent» реалізовано на базі «CompareMatrixComponent» за допомогою наступного коду <app-compare-matrix [parLevel1]="personId"></app-compare-matrix>. Також до користувацького інтерфейсу додається поле вибору зацікавленої особи (рис. 5.14).

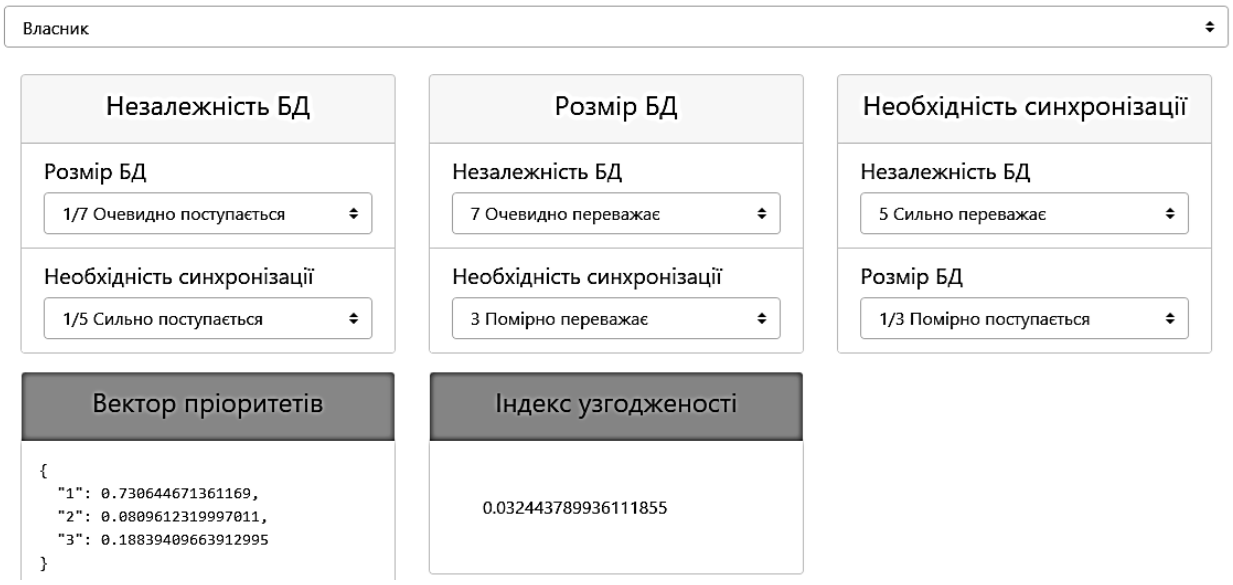


Рисунок 5.14 – Робота із матрицею попарних порівнянь критеріїв оптимальності (власник). Користувацький інтерфейс

Для різних ЛПР набір критеріїв оптимальності може відрізнятись. Так, на рис. 5.15 наведено приклад попарних порівнянь критеріїв для зацікавленої особи «Оператор КІС», для якого критерій «Розмір БД» не є суттєвим відповідно до ієрархічної моделі, що наведено на рис. 3.2.

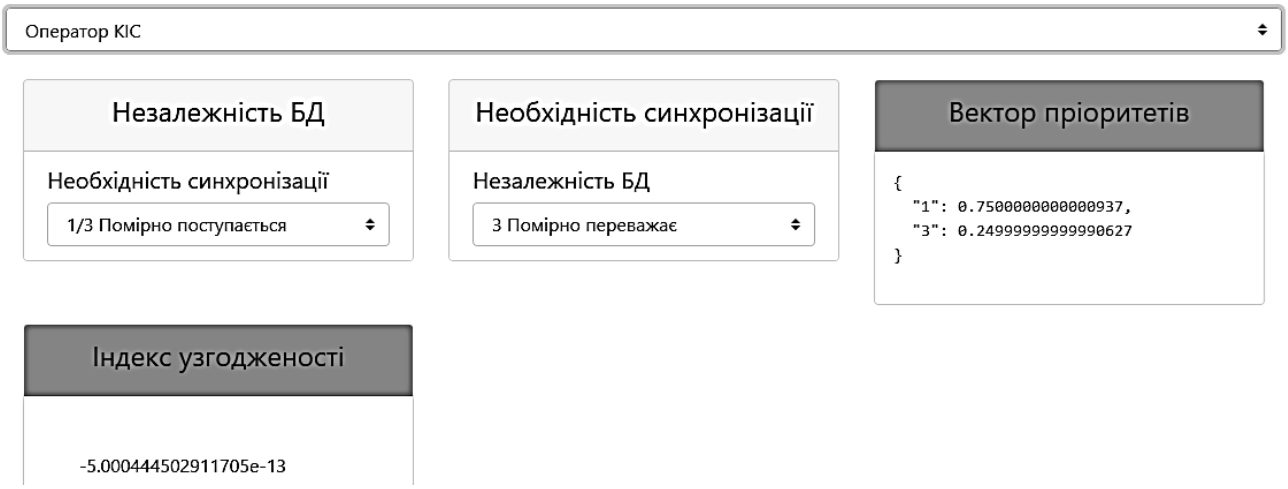


Рисунок 5.15 – Робота із матрицею попарних порівнянь критеріїв ефективності (оператор КІС). Користувацький інтерфейс

5.3 Маркер представленості для елементів вимірів ББД та робота із матрицею переваг альтернатив

Початкова ініціалізація матриці переваг за альтернативами по кожному критерію оптимальності на останньому рівні ієрархії виконується в автоматичному режимі на базі математичних моделей, сформульованих у підрозділі 2.3. Для можливості розрахунку рівня маркеру представленості даних для певних атрибутів та кортежів відношення, згідно з функцією агрегації (2.7), необхідно виконати початкове заповнення рівня маркеру представленості даних для елементів вимірів ББД, описаної у підрозділі 2.2. Для реалізації функціоналу заповнення рівня маркеру представленості даних для елементів вимірів з боку бекенд реалізовані класи сутностей «Application», «Place», «Workstation» та «WsType», що відповідають за роботу із вимірами «Застосунок», «Місце розташування», «Робоча станція» та «Тип робочої станції». Діаграму класів наведено на рис. 5.16.

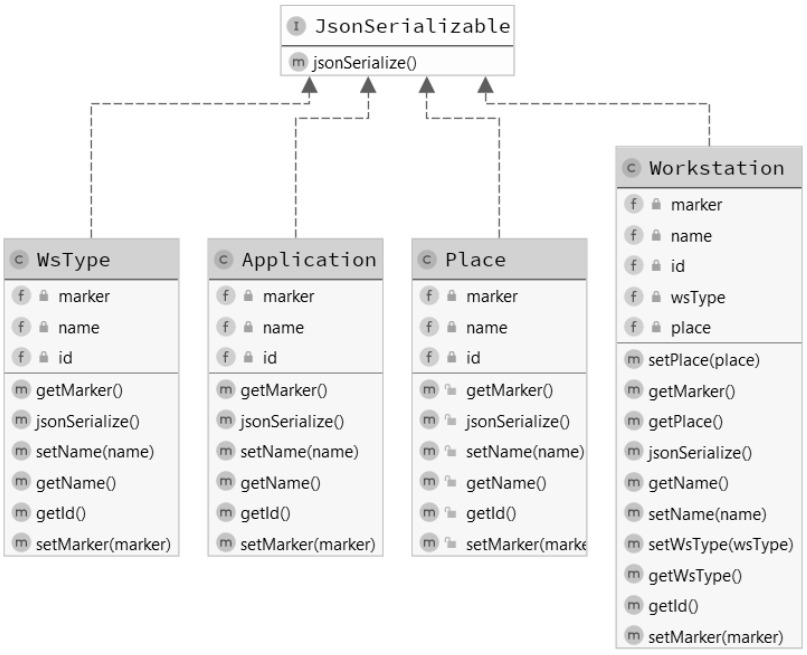


Рисунок 5.16 – Діаграма класів сутностей для роботи із рівнем маркеру представленості елементів вимірів ББД обліку користувацьких запитів

Також для реалізації відповідних маршрутів та взаємодії із фронтенд-складовою інформаційної технології реалізовано класи контролерів «ApplicationController», «PlaceController», «WorkstationController» та «WsTypeController». Діаграму класів контролерів для роботи із сутностями «Застосунок», «Місце розташування», «Робоча станція» та «Тип робочої станції» наведено на рис. 5.17.

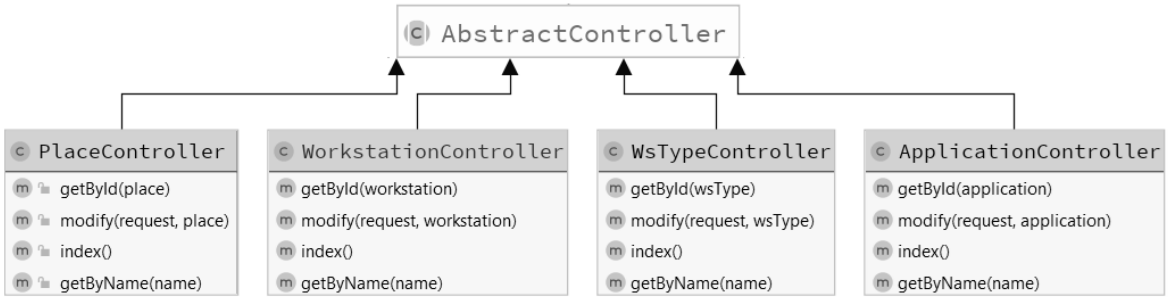


Рисунок 5.17 – Діаграма класів контролерів для модифікації равня маркеру представленості даних елементів вимірів

Наведені класи реалізують маршрути отримання списку елементів вимірів, отримання елементу за назвою або ідентифікатором, та модифікацію рівня маркеру представленості даних відповідного елементу виміру. Повний перелік маршрутів наводиться у табл. 5.3–5.5.

Таблиця 5.3 – Перелік маршрутів роботи із виміром «застосунок»

URI	**Метод / Призначення**	**Параметри /тіло запиту**	**Структура відповіді (json)**	**Описання відповіді**

/application	GET / Отримання списку елементів виміру «застосунок»		[
			{
			id: number,
			name: string,
			marker: number
			}
			]
				Масив елементів виміру «застосунок». Елемент масиву складається з ідентифікатору, назви та значення маркеру представленості даних

/application /{id}	GET / Отримання елементу виміру	Id: number – унікальний ідентифікатор	[{ id: number,	Масив із одного елементу виміру «застосунок» із

URI	Метод / Призначення	Параметри /тіло запиту	Структура відповіді (json)	Описання відповіді
	«застосунок» із унікальним ідентифікатором id	елементу виміру «застосунок»	name: string, marker: number }]	ідентифікатором id. Елемент масиву складається з ідентифікатору, назви та значення маркеру представленості даних
/application /{name}	GET / Отримання елементів виміру «застосунок» із назвою, у яку входить name	Name: string – частина назви елементу виміру «застосунок»	[{ id: number, name: string, marker: number }]	Масив елементів виміру «застосунок» із назвою, у яку входить name. Елемент масиву складається з ідентифікатору, назви та значення маркеру представленості даних
/application /{id}	POST / Зміна значення маркеру представленості даних для елементу виміру «застосунок»	Id: number – унікальний ідентифікатор елементу виміру «застосунок» / Body: { marker: number }	{ id: number, name: string, marker: number }	Елемент виміру «застосунок» після виконання зміни значення маркеру представленості даних

Таблиця 5.4 – Перелік маршрутів роботи із виміром «Місце розташування»

URI	Метод / Призначення	Параметри /тіло запиту	Структура відповіді (json)	Описання відповіді
/place	GET / Отримання списку елементів виміру «місце розташування»		[{ id: number, name: string, marker: number }]	Масив елементів виміру «місце розташування». Елемент масиву складається з ідентифікатору, назви та значення маркеру представленості даних
/place /{id}	GET / Отримання елементу виміру «місце розташування» із унікальним ідентифікатором id	Id: number – унікальний ідентифікатор елементу виміру «місце розташування»	[{ id: number, name: string, marker: number }]	Масив із одного елементу виміру «місце розташування» із ідентифікатором id. Елемент масиву складається з ідентифікатору, назви та значення маркеру представленості даних

URI	Метод / Призначення	Параметри /тіло запиту	Структура відповіді (json)	Описання відповіді

/place /{name}	GET / Отримання елементів виміру «місце розташування» із назвою, у яку входить name	Name: string – частина назви елементу виміру «місце розташування»	[
			{
			id: number,
			name: string,
			marker: number
			}
			]
				Масив елементів виміру «місце розташування» із назвою, у яку входить name. Елемент масиву складається з ідентифікатору, назви та значення маркеру представленості даних

/place /{id}	POST / Зміна значення маркеру представленості даних для елементу виміру «місце розташування»	Id: number – унікальний ідентифікатор елементу виміру «місце розташування» / Body:
		{
		marker: number
		}
			{
			id: number,
			name: string,
			marker: number
			}
				Елемент виміру «місце розташування» після виконання зміни значення маркеру представленості даних

Таблиця 5.5 – Перелік маршрутів роботи із виміром «Робоча станція»

URI	Метод / Призначення	Параметри /тіло запиту	Структура відповіді (json)	Описання відповіді

/workstation	GET / Отримання списку елементів виміру «Робоча станція»		[
			{
			id: number,
			name: string,
			marker: number,
			type: {
			id: number,
			name: string,
			marker: number
			},
			place: {
			id: number,
			name: string,
			marker: number
			},
			}
			]
				Масив елементів виміру «Робоча станція». Елемент масиву складається з ідентифікатору, назви та значення маркеру представленості даних

/workstation /{id}	GET / Отримання елементу виміру «Робоча станція» із	Id: number – унікальний ідентифікатор елементу виміру «Робоча станція»	[{ id: number, name: string,	Масив із одного елементу виміру «Робоча станція» із ідентифікатором id. Елемент масиву

URI	Метод / Призначення	Параметри /тіло запиту	Структура відповіді (json)	Описання відповіді

	унікальним ідентифікатором id		marker: number,
			type: {
			id: number,
			name: string,
			marker: number
			},
			place: {
			id: number,
			name: string,
			marker: number
			},
			}
			]
				складається з ідентифікатору, назви, значення маркеру представленості даних, місця розташування та типу робочої станції

/workstation /{name}	GET / Отримання елементів виміру «Робоча станція» із назвою, у яку входить name	Name: string – частина назви елементу виміру «Робоча станція»	[
			{
			id: number,
			name: string,
			marker: number,
			type: {
			id: number,
			name: string,
			marker: number
			},
			place: {
			id: number,
			name: string,
			marker: number
			},
			}
			]
				Масив елементів виміру «Робоча станція» із назвою, у яку входить name. Елемент масиву складається з ідентифікатору, назви, значення маркеру представленості даних, місця розташування та типу робочої станції

/workstation /{id}	POST / Зміна значення маркеру представленості даних для елементу виміру «Робоча станція»	Id: number – унікальний ідентифікатор елементу виміру «Робоча станція» / Body:
		{
		marker: number
		}
			{
			id: number,
			name: string,
			marker: number,
			type: {
			id: number,
			name: string,
			marker: number
			},
			place: {
			id: number,
			name: string,
			marker: number
			},
			}
				Елемент виміру «Робоча станція» після виконання зміни значення маркеру представленості даних

Таблиця 5.6 – Перелік маршрутів роботи із виміром «Тип робочої станції»

URI	Метод / Призначення	Параметри /тіло запиту	Структура відповіді (json)	Описання відповіді

/ws-type	GET / Отримання списку елементів виміру «Тип робочої станції»		[
			{
			id: number,
			name: string,
			marker: number
			}
			]
				Масив елементів виміру «Тип робочої станції». Елемент масиву складається з ідентифікатору, назви та значення маркеру представленості даних

/ws-type /{id}	GET / Отримання елементу виміру «Тип робочої станції» із унікальним ідентифікатором id	Id: number – унікальний ідентифікатор елементу виміру «Тип робочої станції»	[
			{
			id: number,
			name: string,
			marker: number
			}
			]
				Масив із одного елементу виміру «Тип робочої станції» із ідентифікатором id. Елемент масиву складається з ідентифікатору, назви та значення маркеру представленості даних

/ws-type /{name}	GET / Отримання елементів виміру «Тип робочої станції» із назвою, у яку входить name	Name: string – частина назви елементу виміру «Тип робочої станції»	[
			{
			id: number,
			name: string,
			marker: number
			}
			]
				Масив елементів виміру «Тип робочої станції» із назвою, у яку входить name. Елемент масиву складається з ідентифікатору, назви та значення маркеру представленості даних

/ws-type /{id}	POST / Зміна значення маркеру представленості даних для елементу виміру «Тип робочої станції»	Id: number – унікальний ідентифікатор елементу виміру «Тип робочої станції» /
		Body:
		{
		marker: number
		}
			{
			id: number,
			name: string,
			marker: number
			}
				Елемент виміру «Тип робочої станції» після виконання зміни значення маркеру представленості даних

З боку фронтенд-складової інформаційної технології для реалізації логіки відображення даних елементів вимірів «Застосунок», «Місце розташування», «Робоча станція» та «Тип робочої станції» та модифікації значення маркеру представленості даних реалізовано класи компонент «ApplicationComponent»,

«PlaceComponent», «WorkstationComponent» та «WsTypeComponent». Для виконання http-запитів взаємодії із бекенд-маршрутами, наведеними у табл. 5.3–5.6, реалізовано сервіс «MarkerEditorService». Діаграма класів для наведеного переліку класів фронтенд-складової наводиться на рис. 5.18.

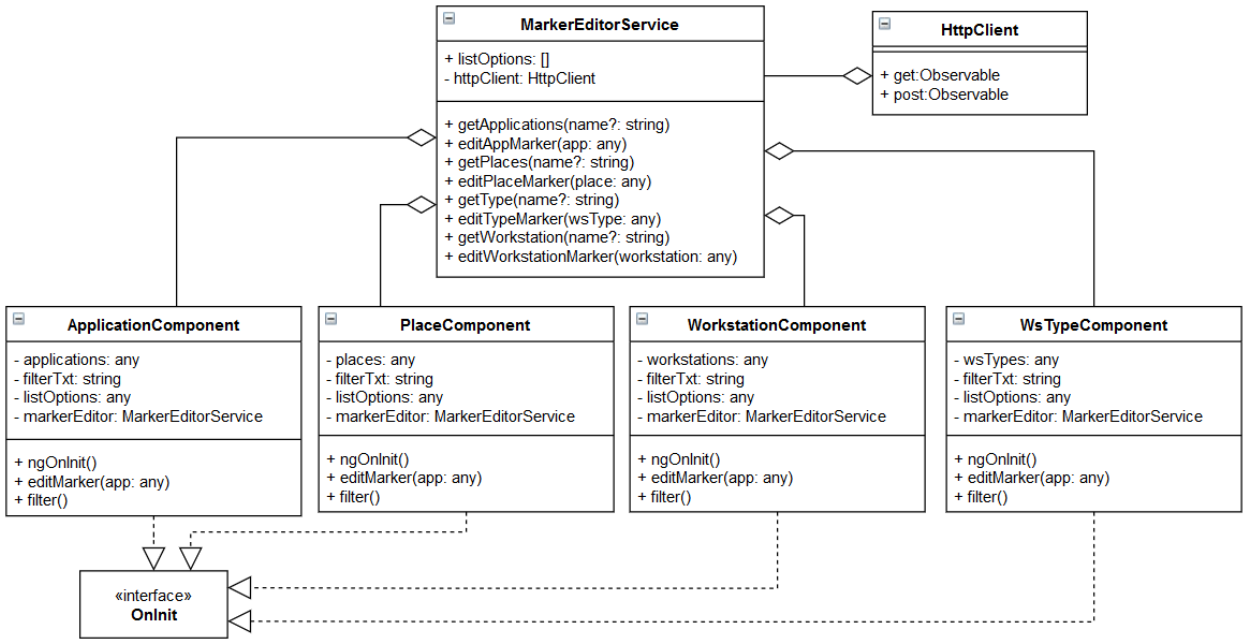


Рисунок 5.18 – Діаграма класів. Компоненти та сервіс по роботі із маркером представленості даних для елементів вимірів

Користувацький інтерфейс реалізовано із дотриманням принципів адаптивного інтерфейсу, відповідно до чого ПЗ може використовуватись на пристроях вводу із різною роздільною здатністю екранів. Зовнішній вигляд сторінки редагування значення маркеру представленості даних елементу виміру «Робоча станція» із використанням шкали {«необхідно», «бажано», «не потрібно»} наведено на рис. 5.19.

Реалізація операцій маніпулювання матрицею переваг альтернатив виконано у класі контролеру «MatrixController» та класах сервісів «MatrixManager» та «AlternativeManager», огляд яких був наведений на рис. 5.13. Початкова ж ініціалізація значень матриці виконується на базі математичних моделей, сформульованих у підрозділі 2.3 із застосуванням нормування значень критеріїв ефективності згідно з (3.3). Отримання значень критеріїв оптимальності

для кожної альтернативи описано у підрозділі 5.1, перелік відповідних маршрутів бекенд-складової наводиться у табл. 5.1, а зовнішній вигляд отриманих даних – на рис. 5.6 та рис. 5.7.

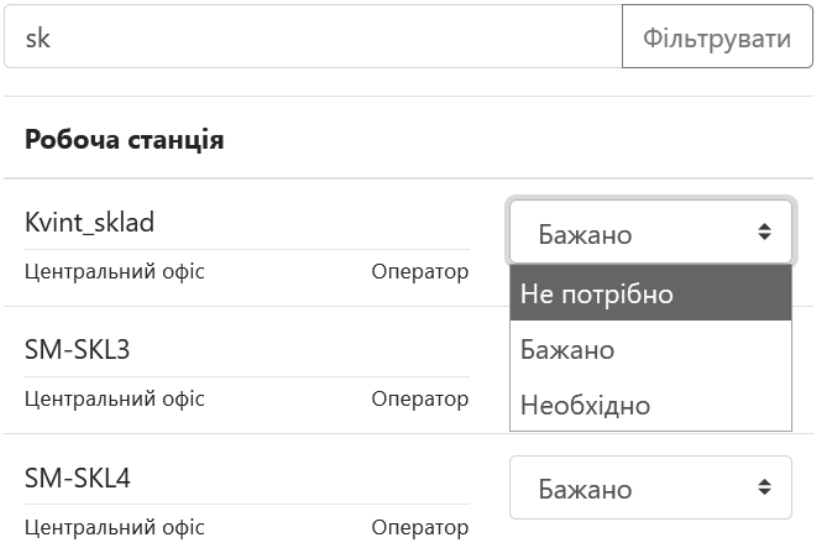


Рисунок 5.19 – Зовнішній вигляд сторінки редагування маркеру представленості для елемента виміру «Робоча станція»

Логіку відображення даних матриці попарних порівнянь альтернатив за критеріями ефективності в аналітиці зацікавлених осіб та взаємодію із бекенд-складовою, реалізована у вигляді компоненти «AltMatrixComponent» на базі компоненти «CompareMatrixComponent» (рис. 5.14). Аналогічно іншим реалізаціям фронтенд-компонент, «AltMatrixComponent» використовує сервіси, у яких реалізована логіка взаємодії із бекенд-частиною застосунку. При реалізації компонента для виведення даних матриці попарних порівнянь альтернатив, наприклад за першим критерієм ефективності по другій зацікавлений особі, використовується html-шаблон наступного вмісту: <app-compare-matrix [parLevel1]="2" [parLevel2]="1"></app-compare-matrix>. Також компонент реалізує вибір зацікавленої особи у елементі користувацького інтерфейсу. Вибір критерію оптимальності виконується у меню застосунку (рис. 5.20).

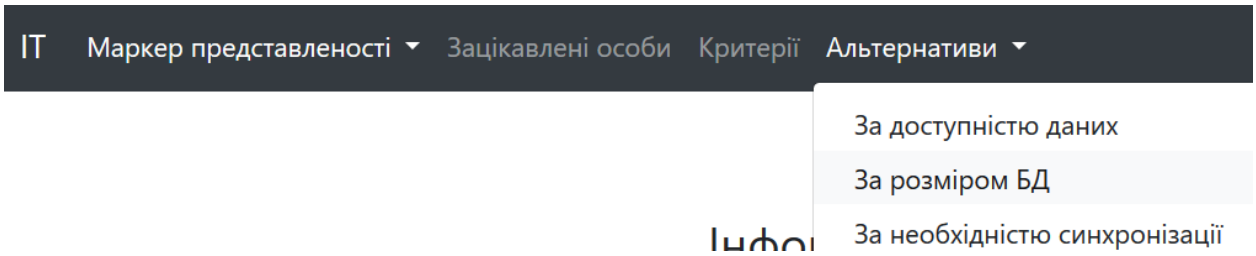


Рисунок 5.20 – Вибір критерію ефективності при роботі із матрицею переваг альтернатив

За аналогією із матрицями переваг першого та другого рівня ієрархії, робота з якими описана у підрозділі 5.2, при модифікації значення переваг елементів матриці динамічно оновлюються значення індексу узгодженості та вектору пріоритетів, що дає можливість ЛПР відразу оцінити наслідки свого рішення. Крім того, згідно з обмеженнями за критеріями оптимальності, кількість альтернатив у матриці переваг, що розглядається, може динамічно змінюватись, що також враховано при реалізації користувацького інтерфейсу (рис. 5.21).

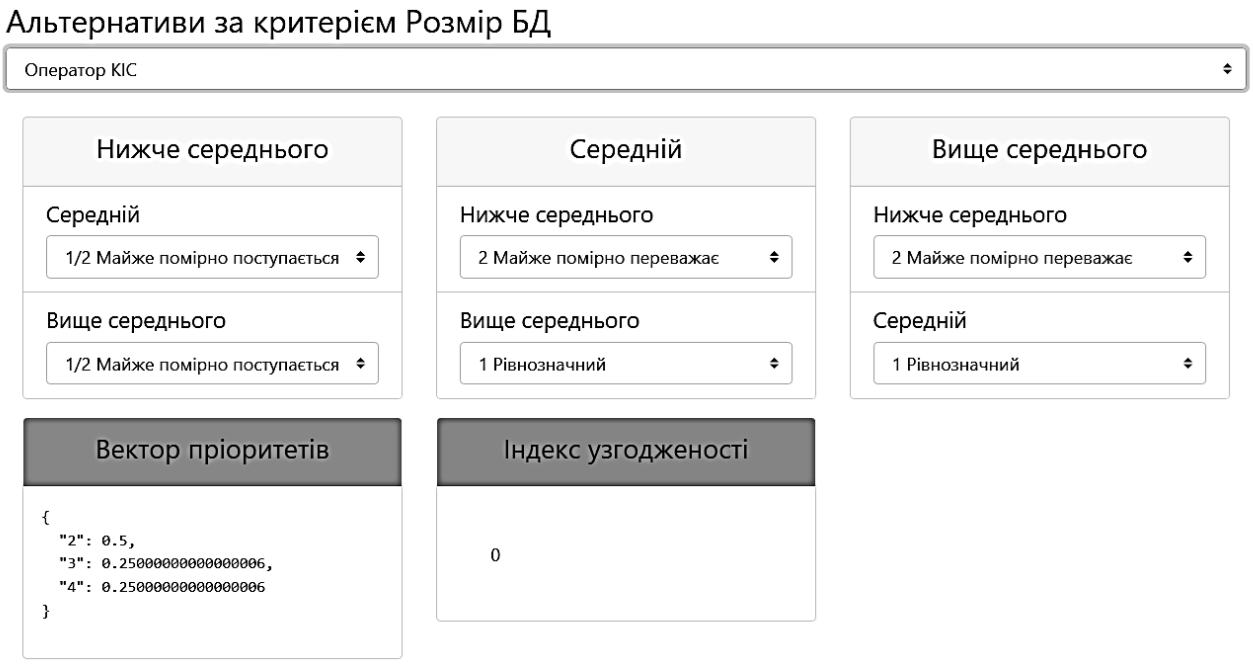


Рисунок 5.21 – Робота із матрицею попарних порівнянь альтернатив за критерієм оптимальності «Розмір БД» (оператор KIC). Користувацький інтерфейс

5.4 Обчислення вектору глобальних пріоритетів та дефазифікація результатів

Обчислення вектору глобальних пріоритетів виконується відповідно до ієрархічної моделі, сформульованої у підрозділі 3.1 знизу догори, починаючи із розрахунку векторів пріоритетів альтернатив за окремими критеріями оптимальності та зацікавленими особами (рис. 5.21). При цьому враховується звуження множини альтернатив відповідно до введених обмежень на область допустимих значень критеріїв оптимальності (підрозділ 5.1). Для визначення глобальних пріоритетів альтернатив по зацікавленій особі використано формулу (3.7), із врахуванням несуттєвості окремих критеріїв для відповідних зацікавлених осіб. Аналогічним чином виконується розрахунок глобального вектору пріоритетів альтернатив.

Реалізація наведеного завдання з боку бекенд-складової інформаційної технології виконана у вигляді методу «getAllMatrixes» класу-контролера «MatrixController» та методу «checkMatrix» сервісу «MatrixManager» (рис. 5.11). Метод «checkMatrix» повертає вектор пріоритетів відповідно до збережених у БД даних попарних порівнянь його елементів для відповідної матриці попарних порівнянь. Метод «getAllMatrixes» звертається до попереднього у циклі та повертає масив векторів пріоритетів для всіх матриць всіх рівнів ієрархії моделі. Описання маршруту для отримання наведених даних приводиться у табл. 5.7.

Фронтенд-складова інформаційної технології у даному випадку реалізує логіку не тільки відображення отриманих даних, а і логіку їх обробки при розрахунку вектору пріоритетів критеріїв оптимальності по зацікавленій особі та глобального вектору пріоритетів альтернатив згідно з (3.7). Для цього реалізовано методи «getPersonVector» та «getGlobalVector» класу компоненти «ResultChartComponent». Для виконання http-запитів взаємодії із бекенд-маршрутом, наведеним у табл. 5.7, використано сервіс «CompareMatrixService» та його метод «getAllResults». Діаграма класів для наведеного переліку класів фронтенд-складової приводиться на рис. 5.22.

Таблиця 5.7 – Маршрут для отримання масиву векторів пріоритетів

URI	Метод / Призначення	Параметри /тіло запиту	Структура відповіді (json)	Описання відповіді
/get-all-matrix	GET / Отримання масиву векторів пріоритетів для всіх матриць всіх рівнів ієрархії моделі		[0: []: number, 1: [0: []: number, 1: []: number, …], …]	Масив векторів пріоритетів для всіх матриць всіх рівнів ієрархії моделі. 0-й елемент масиву – вектор пріоритетів зацікавлених осіб (1-й рівень). Індекс масиву відповідає ідентифікатору елементу сутності «зацікавлена особа». 1-й і наступні елементи (*i*-й). 0-й елемент – вектор пріоритетів критеріїв оптимальності для *i*-ї зацікавленої особи. Індекс масиву відповідає ідентифікатору елементу сутності «критерій оптимальності». 1-й і наступні елементи (*j*-й) – вектор пріоритетів альтернатив для *i*-ї зацікавленої особи та *j*-го критерію оптимальності

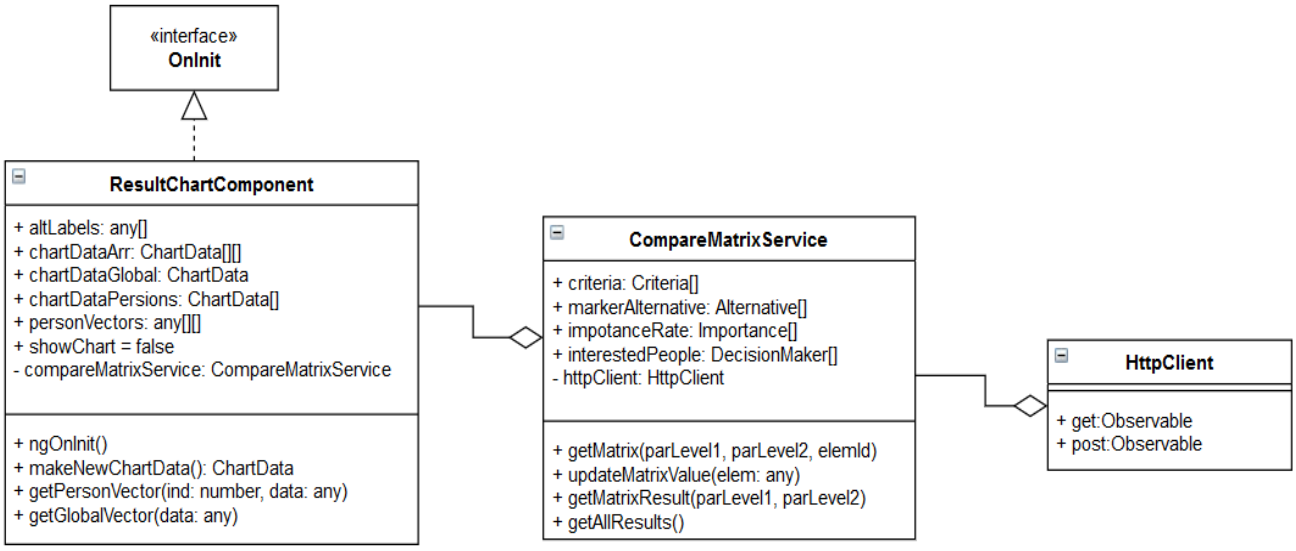


Рисунок 5.22 – Діаграма класів. Компоненти та сервіс для розрахунку та виведення глобального вектору альтернатив

Користувач інформаційної технології має змогу переглянути всі етапи формування глобального вектору пріоритетів альтернатив. Це дає змогу оцінити вплив кожної складеної матриці попарних порівнянь на всіх рівнях ієрархії моделі

та, відповідно, приймати більш осмислене рішення щодо визначення оптимального рівня маркеру представленості даних на вузлі РКІС. На рис. 5.23 наведено вектор глобальних пріоритетів альтернатив у вигляді діаграми із урахуванням звуження множини альтернатив відповідно до встановлених обмежень за критеріями оптимальності (рис. 5.9–5.10). Так, альтернатива «Низький» була виключена виходячи із обмеження мінімального рівня незалежності БД 0,4, а альтернатива «Високий» виключена згідно зі значенням максимального розміру БД вузла РКІС на рівні 0,65.

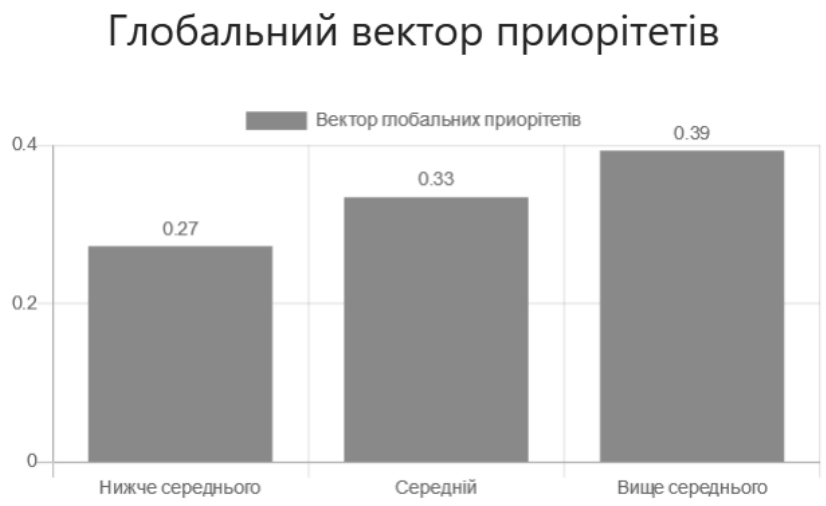


Рисунок 5.23 – Діаграма глобальних пріоритетів альтернатив

Пріоритети за критеріями оптимальності по зацікавленим особам відображаються також у вигляді відповідних діаграм (рис. 5.24). При обробці та виведенні даних враховується особливість ієрархічної моделі відносно несуттєвості деяких критеріїв оптимальності для окремих зацікавлених осіб. Так, на рис. 5.24 та 5.25 наведено діаграми векторів пріоритетів для зацікавлених осіб «Власник» та «Адміністратор БД» відповідно, для першого з яких суттєвими є всі три критерія оптимальності, другий же враховує лише «Розмір БД» та «Необхідність синхронізації» даних вузла РКІС.

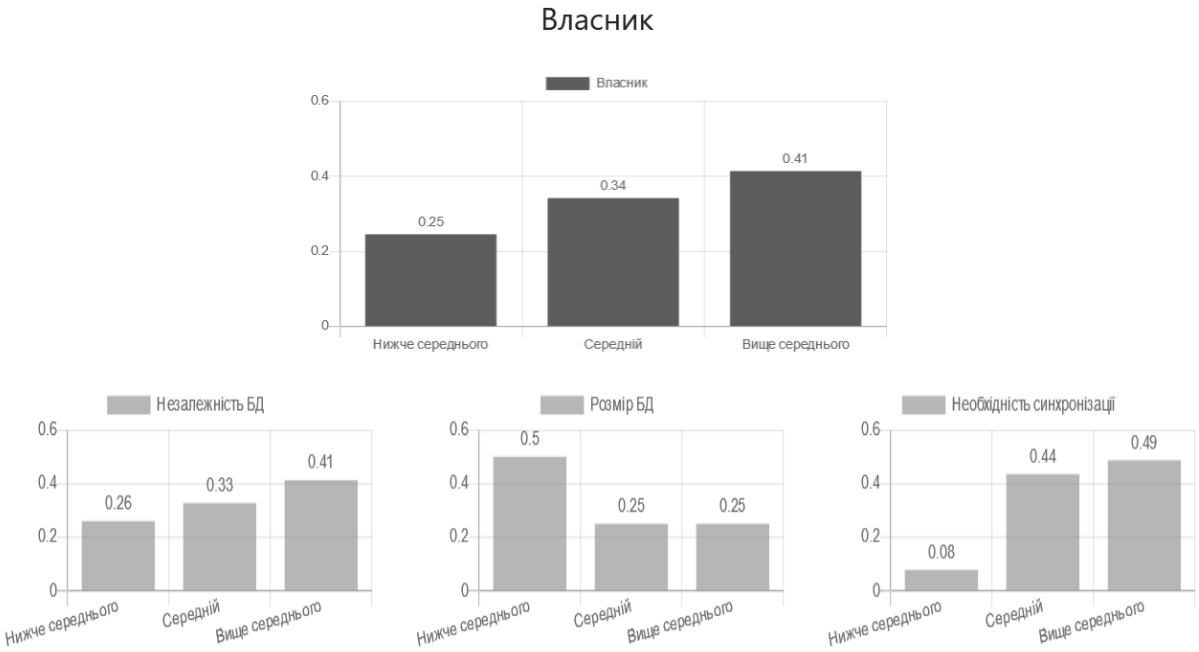


Рисунок 5.24 – Діаграми пріоритетів альтернатив за критеріями оптимальності. Зацікавлена особа «Власник»

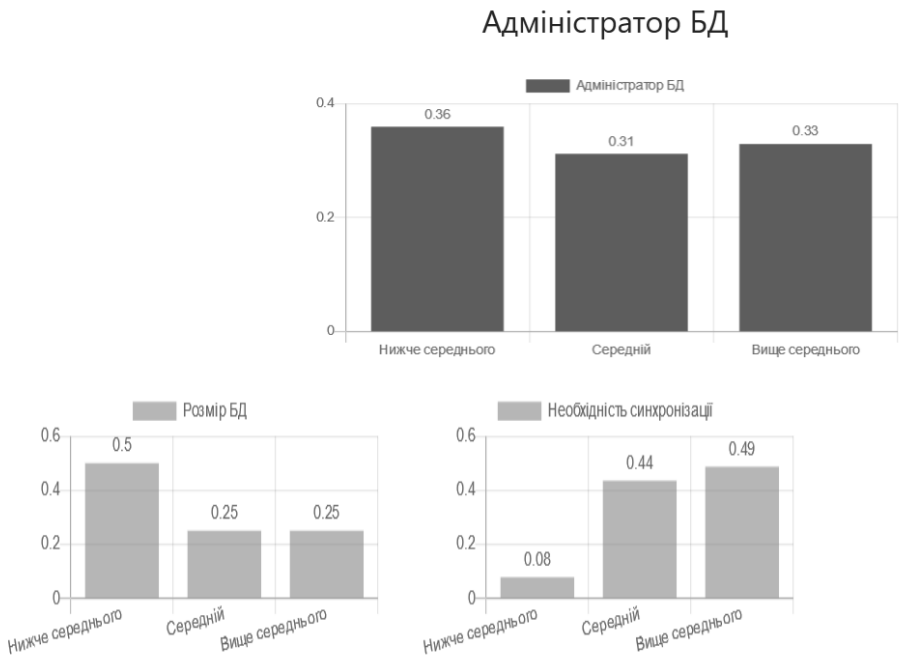


Рисунок 5.25 – Діаграми пріоритетів альтернатив за критеріями оптимальності. Зацікавлена особа «Адміністратор БД»

Останній етап обробки отриманого результату полягає у представленні отриманого вектору глобальних пріоритетів альтернатив у вигляді набору нечітких множин однієї змінної. Для цього значення множини альтернатив

фазифікується та подається у вигляді набору нечітких чисел трикутної форми із кусочно-лінійними функціями приналежності лінійної форми, зовнішній вигляд яких було наведено у підрозділі 3.3 на рис. 3.5. Представивши наведений на рис. 5.23 вектор як вектор нечітких чисел, виконуємо подальше об'єднання результатів та дефазифікацію результату для отримання числового значення оптимального рівня маркеру представленості даних.

Даний етап виконано у середовищі MathLab [161; 162] із використанням модуля Fuzzy Logic Toolbox (рис. 5.26), що дозволяє розробляти та виконувати симуляцію для систем нечіткого логічного висновку [163].

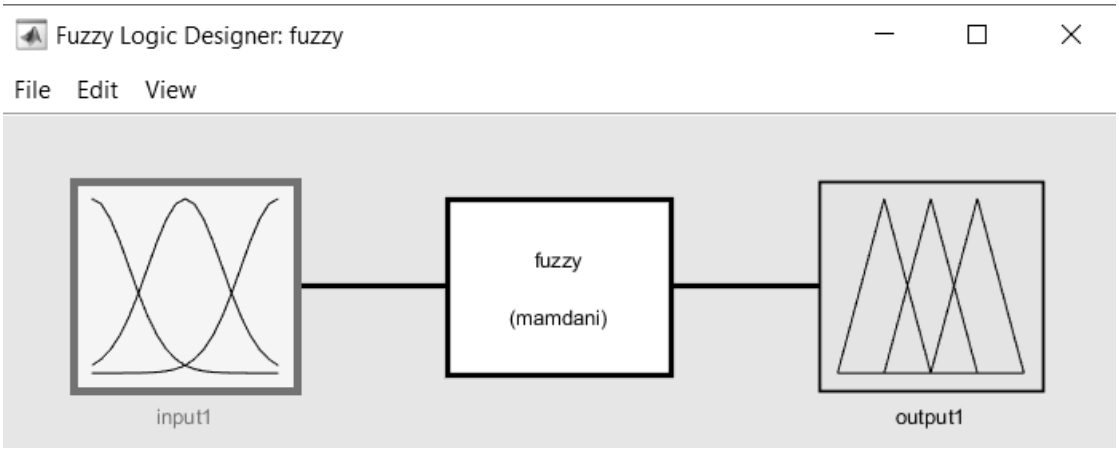


Рисунок 5.26 – Вікно Fuzzy Logic Toolbox у середовищі MathLab

Для запуску Fuzzy Logic Toolbox у середовищі MathLab у командному рядку виконується команда fuzzy. Задано константний вхід «mock» на рівні «1» для всього інтервалу значень маркеру представленості даних [–1; 1] (рис. 5.27).

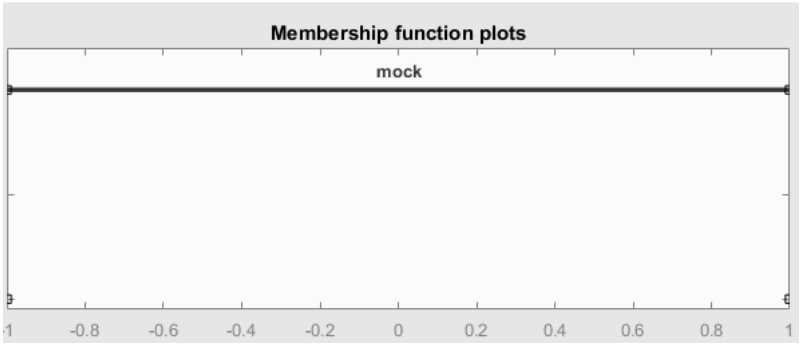


Рисунок 5.27 – Симуляція значення входу

Вихідний сигнал визначено на множині із п’яти інтервалів значення маркеру представленості даних у вигляді нечітких чисел трикутної форми (рис. 5.28).

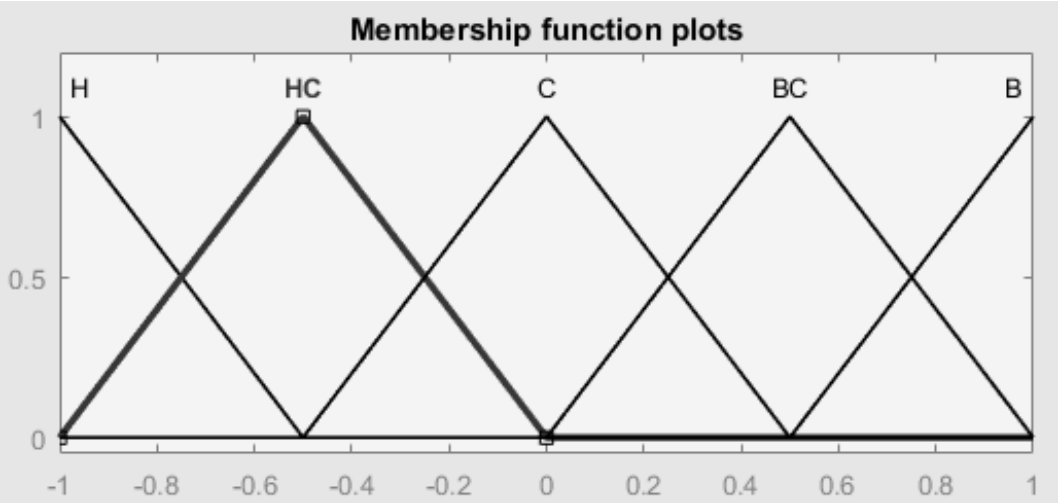


Рисунок 5.28 – Представлення маркеру представленості даних у вигляді п’яти інтервалів

Для представлення отриманого вектор глобальний пріоритетів альтернатив, що був наведений на рис.5.23, в елементі Rule Editor Fuzzy Logic Toolbox задано наступний набір правил:

Значення *«Низький»*: If (input is mock) then output is ***H (0);***

Значення *«Нижче середнього»*: If (input is mock) then output is ***HC (0,27);***

Значення *«Середній»*: If (input is mock) then output is ***C (0,33);***

Значення *«Вище середнього»*: If (input is mock) then output is ***BC (0,39);***

Значення *«Високий»*: If (input is mock) then output is ***B (0).***

Використання інструменту Rule Viewer Fuzzy Logic Toolbox дозволяє отримати дефазифіковане значення вихідної змінної після виконаної агрегації набору наведених правил. Для розглянутого випадку отримано оптимальне значення маркеру представленості даних на вузлі РКІС на рівні 0,2 (рис. 5.29).

Використовуючи агреговане значення маркеру представленості для кортежів та атрибутів відношень БД, отримане згідно з (2.7), визначається множина таких, для яких рівень маркеру перевищує отримане оптимальне

значення (2.8). Отримана підмножина даних БД і має бути представлена на вузлі РКІС.

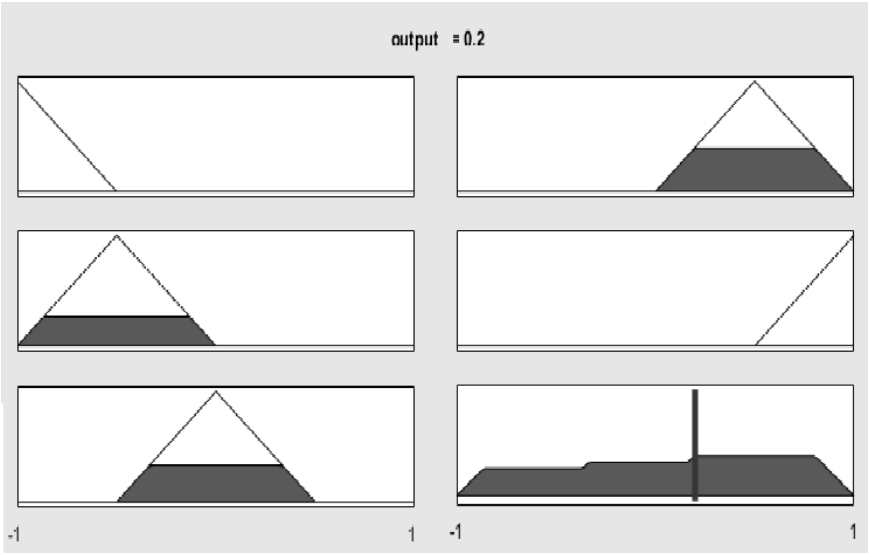


Рисунок 5.29 – Визначення оптимального рівня маркеру представленості даних

Висновки до розділу 5

Функціональна модель визначення оптимального рівня маркеру представленості даних виділяє етапи формалізації обмежень за критеріями оптимальності, роботу із матрицями попарних порівнянь, розрахунку вектору глобальних пріоритетів, а також представлення вектору у вигляді набору нечітких множин однієї змінної із подальшою дефазифікацією для отримання оптимального значення маркеру.

Програмна реалізація визначення оптимального рівня маркеру представленості даних на вузлі РКІС, як частини інформаційної технології оптимізації структури БД вузла РКІС, виконана у вигляді сервіс-орієнтованого вебзастосунку на базі триланкової архітектури. У якості сервера баз даних обрано SQL Server, бекенд-складова (Application Server) реалізована на базі фреймворку Symfony із використанням скриптової мови PHP, а взаємодія із БД реалізована із використанням Object-Relational Mapper (ORM) Doctrine. Фронтенд-складова реалізована на базі фреймворку Angular та мови програмування TypeScript. Запропонована архітектура є гнучкою, реалізує інформаційну технологію у

вигляді вебсервісу, та дозволяє за необхідності легко змінювати СКБД та підключатись до бекенд API, використовуючи стороннє ПЗ.

Наведено діаграми класів та структури окремих таблиць БД для реалізації логіки обробки даних з боку бекенд-складової інформаційної технології та таблиці маршрутів доступу до даних із вказанням форматів вводу-виводу даних та описом додаткових опцій та властивостей. Також наведені діаграми класів фронтенд складової, що реалізують логіку взаємодії із бекенд та користувачем інформаційної технології. Приведено огляд основних форм користувацького інтерфейсу.

Етап обробки отриманого результату, що полягає у представленні вектору глобальних пріоритетів альтернатив у вигляді набору нечітких множин однієї змінної, подальшому об’єднанні результатів та дефазифікації для отримання числового значення оптимального рівня маркеру представленості даних виконується у середовищі MathLab із використанням модуля Fuzzy Logic Toolbox.

У якості результату маємо рішення щодо представленості даних кортежів та атрибутів відношень БД вузла РКІС на базі отриманого оптимального рівня маркеру представленості даних та згідно з (2.6) та (2.7).

ВИСНОВКИ

У дисертації на основі виконаних теоретичних досліджень вирішено актуальну науково-технічну задачу створення моделей та інформаційної технології оптимізації структури БД вузла у КІС.

У ході виконання дослідження:

1. Виконано аналіз особливостей задач використання інтегрованого комплексу територіально розосереджених інформаційних систем та методів їх оптимізації.

2. Побудовано модель SQL-запиту та формалізовано критерії оптимальності структури БД віддаленого вузла КІС.

3. Розроблено інформаційну технологію парсингу та подальшого обліку SQL-запитів до реляційної БД, а також обчислення значень критеріїв оптимальності відповідно до результатів операцій зрізу та консолідації.

4. Сформульовано та вирішено задачу багатокритеріальної оптимізації для визначення оптимального рівня представленості даних на вузлі РКІС.

5. Розроблено інформаційну технологію обчислення вектору глобальних пріоритетів альтернатив із представленням результату у вигляді вектору нечітких чисел та приведенням до чіткого значення.

6. Реалізовано класифікацію нових даних БД відповідно до необхідності їх представлення на вузлі РКІС.

Наукова новизна отриманих результатів:

1. Вперше уведено поняття маркеру представленості даних на вузлі РКІС для елементів вимірів моделі SQL-запиту та розроблена функція агрегації, що дозволяє визначити рівень необхідності атрибутів та кортежів відношення БД на вузлі РКІС на основі статистики SQL-запитів.

2. Вперше побудовано модель залежності критеріїв оптимальності структури БД вузла РКІС від значення маркера представленості даних, що на відміну від існуючих підходів дозволяє визначити оптимальне граничне значення маркера на основі статистики SQL-запитів.

3. Вперше розроблено інформаційну технологію розрахунку критеріїв оптимальності структури БД вузла РКІС, що дозволяє визначити граничний рівень маркеру представленості даних і прийняти рішення про представленість даних на вузлі РКІС, та на відміну від існуючих використовує аналітичні дані та результати парсингу тексту SQL-запитів до серверів РКІС.

4. Удосконалено модель SQL-запиту, яка, на відміну від інших, представлена моделлю багатовимірної БД в аналітиці множини атрибутів та кортежів відношення, а також інших характеристик запиту, що дозволяє проведення оперативно-аналітичного аналізу зважаючи на множинність вимірів з метою розрахунку рівня представленості атрибутів та кортежів відношення БД на вузлі РКІС.

5. Отримав подальшого розвитку метод аналізу ієрархій за рахунок автоматичної ініціалізація матриці попарних порівнянь альтернатив відповідно до отриманих математичних моделей та нормалізації значень та представлення результату у вигляді вектору нечітких чисел із приведенням до чіткого значення, що дозволило підвищити ефективність вузла РКІС на 4 %.

6. Отримав подальшого розвитку метод асинхронної реплікації даних за рахунок динамічного визначення моменту наступної синхронізації даних на базі статистики SQL-запитів, що дозволяє підвищити актуальність даних, та розвантажити ресурси серверів баз даних.

Результати дослідження впроваджено на ППФ «Юнікс Трейд Ко» при проєктуванні структури БД вузла РКІС. Ефект від впровадження полягає у підвищенні ефективності використання БД вузла РКІС на 12 %. Розроблені моделі та інформаційна технологія впроваджені на ТОВ «Еліт Білдінг». Ефект від впровадження полягає у підвищенні швидкості виконання запитів на 14 % та доступності даних на 21 %.

Здійснено впровадження результатів дисертаційного дослідження в навчальний процес на кафедрі «Інженерії програмного забезпечення» при викладанні дисциплін «Клієнт-серверні СКБД та аналітичні системи», «Database Development» та ін.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

[1] Özsu M. T., Valduriez P. Principles of Distributed Database Systems, 3rd ed. Springler, 2011. 845p.

[2] К. Дж. Дейт. SQL и реляционная теория. Как грамотно писать код на SQL. Пер. с англ. СПб.: Символ-Плюс, 2010. 480 с.

[3] М. Л. Дворецький, Боровльова С. Ю., Дворецька С. В. WEB-застосунок складського обліку в неавтоматизованих торгових точках. Наукові праці: Науково-методичний журнал Чорноморського національного університету ім. П. Могили. Сер. Комп'ютерні технології. 2018. Вип. 308 Т. 320. С. 45-52.

[4] Филиппов Е. В. Настольная книга 1С:Эксперта по технологическим вопросам. Издание 2. М.: 1С-Паблишинг, 2015. 247 с.

[5] Matthias Jarke, Jurgen Koch. Query Optimization in Database Systems. Computing Surveys. 1984. Vol. 16, No. 2. P. 111-152.

[6] Дворецький М. Л., Динамічне формування запиту із використанням SQL/OLAP в T-SQL при автоматизації кросс-таблиць на базі реляційних джерел даних. Наукові праці: Науково-методичний журнал Чорноморського державного університету ім. П. Могили. Сер. Комп'ютерні технології. 2014. Вип. 238. Т. 250. С. 32–37.

[7] Как оптимизировать запросы в SQL? : веб-сайт. URL: https://vc.ru/newtechaudit/113408-kak-optimizirovat-zaprosy-v-sql (дата обращения: 20.10.2020).

[8] Оптимизация в распределенных системах управления базами данных : веб-сайт. URL: http://citforum.ru/database/articles/art_26_10.shtml (дата звернення: 20.10.2020).

[9] И. В. Чухраев, И. В. Жукова. Оптимизация работы с информацией в базах данных. Международный научных журнал "Инновационная наука". 2016. №4. С. 206-208

[10] Oracle Database Documentation : веб-сайт. URL: https://docs.oracle.com/en/database/oracle/oracle-database/index.html (Last accessed: 22.10.2020).

[11] Базы данных. Оптимизация запросов. Оптимизация структуры данных : веб-сайт. URL: https://www.youtube.com/watch?v=9yWZ-LIsAII (дата обращения: 20.10.2020).

[12] Пономаренко Л. А., Танянский С. С., Филатов В. А. Построение оптимальной последовательности соединений отношений в запросах реляционной базы данных. Системні дослідження та інформаційні технології. 2003. №2. С. 53-58

[13] Кунгурцев А. Б., Зиноватная С. Л. Иерархическая модель объектов для исследования запросов к базе данных. Труды Одесского политехнического университета. 2008. № 2. С. 130-134.

[14] Филатов В. А., Танянский С. С. Сравнительная характеристика показателей сложности выполнения запросов в реляционных СУБД. Системи обробки інформації. 2004. Вип. 2. С. 91-95.

[15] Данис Искаков. Практика программирования. Оптимизатор запроса : веб-сайт. URL: https://infostart.ru/1c/articles/1172359/ (дата звернення: 20.10.2020).

[16] Гладкий А. Складской учет на компьютере. Лучшие программы, включая 1С 8.2. М.: Литрес, 2013. 532 с.

[17] Радченко М. Г., Хрусталева Е. Ю. 1С:Предприятие 8.3. Практическое пособие разработчика. Примеры и типовые приемы. М.: 1С-Пабл., 2020. 503 с.

[18] Киселев А. Г. Корпоративная и комплексная система управления промышленного предприятия (КИС). Новосибирск, 2010. 408 с.

[19] Дворецький М. Л., Кулаковська І. В. Порівняльний ABC-XYZ аналіз на базі різних факторів із використанням ієрархічних даних. Проблеми інформаційних технологій Херсонського нац. технічного університету. 2016. № 19. С. 200–209.

[20] Markus Winand. SQL performance. DGS Druck u. Graphikservice GmbH Wien Austria, 2014. 197 p.

[21] Дворецький М.Л. Підходи щодо підвищення швидкості роботи SQL-запитів при використанні індексів БД. Ольвійський форум: стратегії країн Причорноморського регіону в геополітичному просторі: матеріали XII міжнар.

наук.-практ. конф., 7-10 червня 2018 р. Миколаїв : Вид-во ЧНУ ім. Петра Могили, 2018. С. 30-32.

[22] Petkovic D. Microsoft SQL Server 2019: A Beginner's Guide, Seventh Edition. McGraw-Hill Education, 2020. 864 p.

[23] Шварц Б., Зайцев П., Ткаченко В. MySQL. Оптимизация производительности (2-е издание). Символ, 2010. 823 с.

[24] Отсутствующие индексы в MS-SQL или оптимизация «по-быстрому» : веб-сайт. URL: http://kurenkov.pro/blog/87-otsutstvuyushchie-indeksy-v-ms-sql-ili-optimizatsiya-po-bystromu (дата звернення: 20.10.2020).

[25] Дворецький М. Л., Дворецька С. В., Давиденко Є. О. Розробка системи управління знаннями організації на базі CMS WORDPRESS. Проблеми інформаційних технологій Херсонського національного технічного університету. 2018. №1 (023). С. 173–180.

[26] Гарсиа-Молина Г., Ульман Дж.Д., Уидом Дж. Системы баз данных. Полный курс. Пер. с англ. М.: Изд. дом «Вильямс», 2003. 1088 с.

[27] Крёнке Д. Теорія і практика проектування баз даних. 8-е изд. Пер. с англ. СПб: Питер, 2003. 800 с.

[28] Дейт К. Дж. Введение в системы баз данных. 8-е изд. Пер. с англ. М.: Изд. дом «Вильямс», 2005. 1328 с.

[29] Новиков Б. А., Рогова Е. В. Основы технологий баз данных: учеб. пособие. М.: ДМК Пресс, 2019. 240 с.

[30] Денормализация как средство повышения производительности. Реализация денормализации : веб-сайт. URL: https://intellect.icu/denormalizatsiya-kak-sredstvo-povysheniya-proizvoditelnosti-realizatsiya-denormalizatsii-7877 (дата обращения: 20.10.2020).

[31] Кунгурцев О.Б., Зіноватна С.Л. Порівняння вартості виконання запитів до й після денормалізації реляційної БД. ВІСНИК ЖДТУ Одеського нац. політехнічного університету: Технічні науки. 2006. No 4 (39). С. 207-212.

[32] Зіноватна С. Л. Інформаційна технологія підвищення продуктивності автоматизованих систем методами реструктуризації бази даних : Автореф. дис. на здобуття наук. ступеня канд. техн. наук : 05.13.06. Одеса, 2008. 18 с.

[33] Кунгурцев А. Б., Зиноватная С. Л. Имитационная модель для исследования эффективности денормализации реляционной базы данных в информационной системе. Радіоелектроніка, інформатика, управління. Науковий журнал Запорожскького нацыонального технічного університету. 2008. Випуск 1 (19). С. 60-69.

[34] Дворецький М. Л. Інтелектуальний аналіз даних в 1С:8.0. Наукові праці: Науково-методичний журнал Миколаївського державного гуманітарного університету ім. П. Могили. Сер. Комп'ютерні технології. 2007. Вип. 55. Т. 68. С. 141–149.

[35] Кунгурцев А.Б., Возовиков Ю.Н. Поиск закономерностей в распределении запросов для управления материализованными представлениями. Труды Одесского политехнического университета. 2008. 2(30). С. 135— 140.

[36] Кунгурцев А.Б. Зиноватная С.Л. Модель реструктуризации реляционной базы данных путем денормализации схемы отношений. Труды Одесского политехнического университета. 2006. 2(26). С. 105–111.

[37] Биков Д. П., Фісун М. Т. Створення серверної СКБД, що підтримує сіткову модель даних на базі локальної. Наукові записки. Комп'ютерні науки. К. : Видавничий дім «КМ Академія». 2005. Том 36. С. 21–25.

[38] Кунгурцев, О. Б., Возовиков, Ю. М. Технология создания материализованных представлений для реляционных баз данных. Праці Одеського політехнічного університету. 2012. №2 (39). С. 170-176

[39] Нгуєн Чан Куок Вінь. Підвищення ефективності застосування матеріалізованих представлень в автоматизованих комп'ютерних системах з реляційними базами даних : Автореф. дис. на здобуття наук. ступеня канд. техн. наук : 05.13.06. Одеса, 2005. 19 с.

[40] Fisun M., Dvoretskyi M., Horban H., Komar M. Knowledge management applications based on user activities feedback. International Journal of Computing, Ternopil. 2019. 18 (1). P. 32-44.

[41] Fisun M., Dvoretskyi M., Horban H. The usage of the feedback with user activities in company knowlagde management system. 12th International Scientific and

Technical Conference on Computer Sciences and Information Technologies (CSIT), Lviv. 2017. P. 143-146. doi: 10.1109/STC-CSIT.2017.8098755

[42] Стулов А.В. Хранилища данных: основные архитектуры и принципы построения. Интуит, 2010. 487 с.

[43] Методы и модели анализа данных: OLAP и Data Mining / Барсегян А. А., Куприянов М. С., Степаненко В. В., Холод И. И. Питербург, 2004. 336 с.

[44] Пасічник В. В., Шаховська Н. Б., Сховища та простори даних: монографія. Львів : Вид-во Львівської політехніки, 2009. 244 с.

[45] Пасічник В. В., Шаховська Н. Б., Сховища даних: Навчальний посібник. Львів : "Магнолія, 2006", 2008. 492 с.

[46] Малахов Е. В., Востров Г. Н., Мороз В. В. Проблемы создания баз данных и информационных хранилищ : веб-сайт. URL: http://www.nbuv.gov.ua/Articles/OSPU/opu_97_2/1_19.htm (дата звернення: 24.06.2016).

[47] Малахов Е. В., Иванченко О.В. Организация переноса информации из баз данных в информационные хранилища. Тр. Одес. политехн. ун-та, Одесса. 1998. No 2(6). С. 52— 54.

[48] Філатов В. О. Мультіагентні технології інтеграції гетерогенних інформаційних систем і розподілених баз даних : автореф. дис. на здобуття наук. ступеня д-ра техн. наук : 05.13.06. Харків, 2005. – 32 с.

[49] Землянська С. Ю. Оптимізація розподілених корпоративних інформаційних систем з використанням еволюційних обчислень і об’єктного моделювання. : Автореф. дис. на здобуття наук. ступеня канд. техн. наук : 05.13.06. Донецьк, 2013. – 24 с.

[50] Методика проектирования распределенных информационных хранилищ : веб-сайт. URL: https://cyberleninka.ru/article/n/metodika-proektirovaniya-raspredelennyh-informatsionnyh-hranilisch-1 (дата обращения: 20.10.2020)

[51] Фісун М. Т., Дворецький М. Л. Синхронізаця оновлення даних в гетерогенних інформаційних системах. Наукові праці: Науково-методичний

журнал Миколаївського державного гуманітарного університету ім. П. Могили. Сер. Комп’ютерні технології. 2006. Вип. 44. Т. 57. С. 61-66.

[52] Дворецький М.Л. Інтеграція підсистем обліку та оперативно-аналітичної обробки даних в інформаційній системі супермаркету. Комп’ютерні науки: освіта, наука, практика. Миколаїв. Національний університет кораблебудування. 2014. С. 58-60.

[53] Малахов Є. В., Блажко О. А., Глава М. Г. Проектування баз даних та їх реалізація засобами стандартного SQL та PostgreSQL: Навчальний посібник для студентів вищих навчальних закладів. Одеса : ВМВ, 2012. 248 с.

[54] Моделирование и оптимизация распределенных информационных систем / Скобцов Ю. О. та ін. Донецк : Изд-во "Нуолидж", 2012. 300 с.

[55] Savchuk T., Kozachuk A. Development of cloud application efficiency evaluation criterion. EasternEuropean Journal of Enterprise Technologies. 2015. № 2 (77). P. 20 – 26. ISSN 1729-3774

[56] Филатов В. А., Семенец Р. В. Методы и средства проектирования информационных систем и распределенных баз данных. Вестник Херсонского национального технического университета. 2007. No 4(27). С. 203-207.

[57] Оззу М. Т., Валдуриз П. Распределенные и параллельные системы баз данных : веб-сайт. URL: https://www.sqlmanager.net/ru/products/datacomparer (дата обращения: 21.10.2020)

[58] Лаздынь С.В., Землянская С.Ю. Оптимизация распределенных корпоративных информационных сетей с использованием генетических алгоритмов и объектного моделирования. Наукові праці ДонНТУ. 2009. No 147. С.83-95.

[59] Иванов А.Ю. Мобильные распределенные базы данных втоматизированных информационно-управляющих систем МЧС России: монография. СПб.: Санкт-Петербургский ун-т ГПС МЧС России. 2008. 152 с.

[60] Пасічник В. В., Резніченко В. А. Організація баз даних та знань. К.: Видавнича група BHV, 2006. 384 с.

[61] Ковалюк О. А. Интеграция программных компонентов распределенной информационной системы. Информационные технологии и компьютерная техника. 2011. No 4. С. 95-96.

[62] Касаткіна Н. В., Пономаренко Л. А., Філатов В. О. Роль і місце інтегрованих баз даних у розподіленій інформаційній системі ВАК України. Проблеми системного підходу в економіці: Збірник наук. праць. 2009. №29. С. 3-7.

[63] Берко А. Ю., Верес О.М., Пасічник В. В. Системи баз даних та знань. Книга 1. Організація баз даних та знань: Навч. посібник. Львів: "Магнолія, 2006", 2008. 456 с.

[64] Чекалов А. П. Базы данных: от проектирования до разработки приложений. СПб.: БХВ-Питербург, 2003. 384 с.

[65] Фісун М. Т., Дворецький М. Л., Юхатов А. В. Порівняльний аналіз методів побудови OLAP-систем із використанням засобів MS SQL SERVER та ORACLE. Наукові праці: Науково-методичний журнал Чорноморського державного університету ім. П. Могили. Сер. Комп’ютерні технології. 2016. Вип. 271. Т. 283. С. 36–42.

[66] Дворецький М.Л. Задача синхронізації даних в РБД. Синхронне та асинхронне оновлення. Могилянські читання: досвід та тенденції розвитку суспільства в Україні: глобальний, національний та регіональний аспекти: матеріали всеукр. наук.-метод. конф., 14-18 лист. 2016 р. Миколаїв: Вид-во ЧНУ ім. П. Могили, 2016. С. 105-108.

[67] Кузнецов М., Симдянов И. MySQL 5. СПб.: БХВ-Петербург, 2010. 342 с.

[68] Blokdyk G. Microsoft SQL Server A Complete Guide - 2019 Edition. 5STARCooks, 2019. 312 p.

[69] Ward B. Pro SQL Server on Linux: Including Container-Based Deployment with Docker and Kubernetes. Apress, 2018. 632 p.

[70] Malcher M., Kuhn D. Pro Oracle Database 18c Administration: Manage and Safeguard Your Organization’s Data. Apress, 2019. 1022 p.

[71] EMS Software development. Data Comparer : Группа продуктов : веб-сайт. URL: https://www.sqlmanager.net/ru/products/datacomparer (дата обращения: 21.10.2020)

[72] Akeneo. Create superior product experiences : веб-сайт. URL: https://www.akeneo.com/ (Last accessed: 21.10.2020).

[73] Хрусталева Е. Ю. Технологии интеграции 1С:Предприятия 8.3. М.: 1С-Паблишинг, 2020. 503 с.

[74] Jin Hyun Son, Myoung Ho Kim. An adaptable vertical partitioning method in distributed systems. Journal of Systems and Software. 2004. Vol. 73, No 3. P. 551-561.

[75] Дворецький М. Л. Проектування та оцінка оптимальності структури сховища даних та багатовимірної БД. Наукові праці: Науково-методичний журнал Миколаївського державного гуманітарного університету ім. П. Могили. Сер. Комп'ютерні технології. 2008. Вип. 77. Т. 90. С. 216-221.

[76] Дворецький М. Л., Давиденко Є. О., Боровльова С. Ю. Проектування структури розподіленої БД на базі парсингу SQL-запитів. Наукові праці: Науково-методичний журнал Чорноморського державного університету ім. П. Могили. Сер. Комп'ютерні технології. 2016. Вип. 275. Т. 287. С. 53–61.

[77] Fisun M., Dvoretskyi M., Shved A. Davydenko Y. Query parsing in order to optimize distributed DB structure. 9th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS), Bucharest. 2017, P. 172-178. doi: 10.1109/IDAACS.2017.8095071

[78] Kandyba I., Davydenko Y., Panasyuk V., Shved A., Fisun M. ANTRL as a Development Platform for the Series DSL for the Learning Process. 10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS). 2019. P. 597-600. doi: 10.1109/IDAACS.2019.8924354

[79] Гермейер Ю.Б. Введение в теорию исследования операций. М.: Наука, 1971. 384 с.

[80] Лотов А.В., Поспелова И.И. Многокритериальные задачи принятия решений: Учебное пособие. М:МАКС Пресс, 2008. 197 с.

[81] Kondratenko Y., Kondratenko G., Sidenko I. Multi-criteria deci-sion making for selecting a rational IoT platform. IEEE 9thInt. Conf. Dependable Syst., Services Technol. (DESSERT). 2018. P. 147–152.

[82] Trunov A. Realization of the paradigm of prescribed control of a nonlinear object as the problem on maximization of adequacy. Eastern-European Journal of Enterprise Technologies. 2016. 4(82). P. 50-58. DOI: 10.15587/1729-4061.2016.75674.

[83] Кини Р.Л., Райфа Х. Принятие решений при многих критериях: предпочтения и замещения. М.: Радио и связь, 1981. 560 с.

[84] Ногин В. Д. Принятие решений в многокритериальной среде: количественный подход. М. : ФИЗМАТЛИТ, 2002. 144 с.

[85] Ногин В. Д. Методы оптимальных решений : учебное пособие. СПб. : Издательство «Юстас», 2006. 108 с.

[86] Коваленко І. І., Давиденко Є. О. Парето-оптимальний вибір при формуванні портфеля замовлень ІТ-проектів. Наукові праці : науково-методичний журнал : Комп’ютерні технології. Миколаїв : Вид-во ЧДУ ім. П. Могили, 2011. Вип. 161. Т. 173. С. 44-48.

[87] Подиновский В. В., Ногин В. Д. Парето-оптимальные решения многокритериальных задач. М. : Наука. Главная редакция физико-математической литературы, 1982. 256 с.

[88] Saaty T. L. Multicriteria Decision Making: The Analytic Hierarchy Process: Planning, Priority Setting, Resource Allocation 2nd Edition. RWS Pubns, 1990. 287 p.

[89] Саати Т. Л. Принятие решений при зависимостях и обратных связях. Аналитические сети. Ленанд, 2016. 360 с.

[90] Saaty T. L., Vargas L. G. Models, Methods, Concepts & Applications of the Analytic Hierarchy Process. Springer US, 2012. 346 p.

[91] Саати Т. Л. Принятие решений. Метод анализа иерархий : Пер. с англ. Москва : "Радио и связь", 1993. 278 с.

[92] Малахов Є.В. Основи проектування баз даних : Навч. посіб. для студ. вищих навч. закладів. О: Наука і техніка, 2006. 156 с.

[93] Пасічник В. В., Резніченко В.А. Організація баз даних та знань. К.: Видавнича група BHV, 2006. 384 с.

[94] Использование синхронных и асинхронных операций базы данных: веб-сайт. URL: http://help.adobe.com/ru_RU/as3/dev/WS5b3cc c516d4fbf351e63e3d118666ade46-7d39.html (дата обращения: 10.11.2019).

[95] Liang D. Y. Introduction to Java Programming and Data Structures, Comprehensive Version 11th Edition. Pearson, 2017. 1232 p.

[96] Price M. J. C# 8.0 and .NET Core 3.0 – Modern Cross-Platform Development. Packt Publishing, 2019. 818 p.

[97] Приемы объектно-ориентированного проектирования. Паттерны проектирования / Гамма Э., Хелм Р., Джонсон Р., Влиссидес Дж. СПб: Питер, 2001. 368 с.

[98] Паттерны проектирования / Фримен Э., Фримен Э., Сьерра К., Бейтс Б. СПб.: Питер, 2011. 656 с.

[99] Реляційна модель даних. Деякі аспекти проектування реляційних баз. URL: https://slide-share.ru/zanyattya-2-relyacijna-model-danikh-deyaki-aspekti-proektuvannya-relyacijnikh-baz-38275 (дата звернення: 22.10.2020).

[100] Кузнецов С. Д. Базы данных: языки и модели. Учебник. М.: ООО "Бином-Пресс", 2008. 720 с.

[101] Райордан Р. М. Основы реляционных баз данных. Пер, с англ. М.: Издательско-торговый дом «Русская Редакция», 2001. 384 с.

[102] Коннолли Т., Карелли Б. Базы данных. Проектирование, реализация и сопровождение. Теория и практика. 3-е издание. : Пер. с англ. М. : Издательский дом "Вильяме", 2003. 1440 с.

[103] Фісун М.Т., Дворецький М.Л., Дворецька С.В. Побудова моделей для оптимізації структури бази даних вузла у корпоративних інформаційних системах. Інформаційні технології та комп’ютерна інженерія: Міжнародний науково-технічний журнал Вінницького національно-технічного університету. vol 48, № 2. 2020. С. 52-60. doi: https://doi.org/10.31649/1999-9941-2020-48-2-52-60.

[104] Чернoруцкий И. Г. Методы оптимизации и принятия решений : учебное пособие. СПб.: Издательство "Лань", 2001. 383 с.

[105] Петкович Д. Microsoft SQL Server 2012. Руководство для начинающих: Пер. с англ. СПб. : БХВ-Петербург, 2013. 816 с.

[106] Автоматическая синхронизация распределенных баз данных в разделенном режиме : веб-сайт. URL: http://stimul.kiev.ua/materialy.htm ?a=avtomaticheskaya_sinkhronizatsiya_raspredelennykh_baz_dannykh_v_razdelenno m_rezh (дата обращения: 22.10.2020).

[107] Dvoretskyi M., Dvoretska S., Nezdoliy Y., Borovlova S. Data Utility Assessment while Optimizing the Structure and Minimizing the Volume of a Distributed Database Node, 1st International Workshop on Information-Communication Technologies & Embedded Systems (ICTES), Mykolaiv, Ukraine, November 14-15. 2019. P. 128-137.

[108] Buckley J. J. Fuzzy hierarchical analysis. Fuzzy sets and systems. 1985. Vol. 17, No 3. P. 233–247.

[109] Университет Синергия. Общая характеристика метода анализа иерархий : веб-сайт. https://www.youtube.com/watch?v=fp2JEgbSA8k&t=456s (дата обращения: 22.10.2020).

[110] Шеври Ф., Гели Ф. Нечеткая логика. Техническая коллекция Schneider Electric. 2009. Выпуск 31. URL: http://www.netkom.by/docs/N31-Nechetkaya-logika.pdf (дата обращения: 22.10.2020).

[111] Gozhyj A., Kalinina I., Gozhyj V. Fuzzy cognitive analysis and modeling of water quality. 9th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS). 2017. P. 289-294.

[112] Zhuravska I.M., Lernatovych D.O. Fuzzy modeling system of human behavior and biometric identification using cloud serviaces. 5th International Scientific Conference “Applied Sciences in Europe: tendencies of contemporary development”: Papers of the 5th International Scientific Conference. Stuttgart, Germany. March 24, 2014. P.30-32.

[113] Кутковецький В. Я. Розпізнавання образів : навчальний посібник. Миколаїв : Вид-во МДГУ ім. П. Могили, 2003. 196 с.

[114] Томский Государственный Университет Систем Управления и Радиоэлектроники. Решение задачи нечеткого вывода. Основные этапы нечеткого вывода : веб-сайт. URL: https://studfile.net/preview/3652151/page:3/ (дата обращения: 22.10.2020).

[115] Дворецький М. Л., Дворецька С. В., Давиденко Є. О. Інформаційна технологія визначення корисних даних при оптимізації структури та мінімізації обсягів вузла розподіленої БД. Вісник Черкаського державного технологічного університету. 2019. № 4. С. 26–35. doi: 10.24025/2306-4412.4.2019.184808

[116] Пател А. Прикладное машинное обучение без учителя с использованием Python. Киев: Изд-во Диалектика, 2020. 432 с.

[117] Вьюгин В. В. Математические основы машинного обучения и прогнозирования : электронное издание. М.: МЦНМО, 2014. 304 с.

[118] Дворецький М. Л. Реалізація задачі класифікації засобами мови Transact-SQL при інтелектуальному аналізі даних. Системні технології: Регіональний міжвузівський збірник наукових праць. Дніпропетровськ, ДНВП «Системні технології». 2006. Випуск №6 (47). С. 102-111.

[119] Фісун М. Т., Дворецький М. Л. Пошук асоціативних правил засобами мови SQL при інтелектуальному аналізі даних. Вісник Херсонського національного технічного університету. 2005. № 1 (21). С. 185–189.

[120] Дворецький М. Л. Розв’язання задачі класифікації в багатовимірних базах даних. Вісник Херсонського національного технічного університету. 2006. № 1 (30). С. 198–202.

[121] Обзор самых популярных алгоритмов машинного обучения : веб-сайт. URL: https://tproger.ru/translations/top-machine-learning-algorithms/ (дата звернення: 22.10.2020).

[122] Хайкин С. Нейронные сети. Киев: Изд-во Диалектика, 2020. 1104 с.

[123] Аггарвал Ч. Нейронные сети и глубокое обучение: учебный курс. Киев: Диалектика-Вильямс, 2020. 752 с.

[124] Дворецький М.Л., Дворецька С. В. Використання brain.js при визначенні корисності кортежу для віддалено вузла РБД. Могилянські читання: досвід та тенденції розвитку суспільства в Україні: глобальний, національний та регіональний аспекти: матеріали XXII Всеукр. наук.-метод. конф., 11-16 лист. 2019 р. Миколаїв: Вид-во ЧНУ ім. П. Могили, 2019. С. 123-125.

[125] Акбархужаев С. А., Абдурахманова Н. Н. Сравнительный анализ методов Наивного Байеса и SVM алгоритмов при классификации текстовых документов. Молодой ученый. 2019. № 29 (267). С. 8-10. URL: https://moluch.ru/archive/267/61568/ (дата обращения: 22.10.2020).

[126] Фаулер М. Архитектура корпоративных программных приложений : Пер. с англ. М.: Издательский дом "Вильяме", 2006. 544 с.

[127] Фісун М. Т., Дворецький М. Л. Застосування OLAP-технологій в інформаційно-управляючих системах. Інформаційні технології та комп'ютерна інженерія: Міжнародний науково-технічний журнал Вінницького національно-технічного університету. 2006. №2(6). С. 128–134.

[128] 6 простых шагов для освоения наивного байесовского алгоритма (с примером кода на Python) : веб-сайт. URL: http://datareview.info/article/6-prostyih-shagov-dlya-osvoeniya-naivnogo-bayesovskogo-algoritma-s-primerom-koda-na-python/ (дата обращения: 22.10.2020).

[129] Марка Д. А., МакГоуэн К. Методология структурного анализа и проектирования SADT. McGraw-Hill Companies, 1999. 231 с.

[130] Лобова Г. SADT - технология деятельности. LAP Lambert Academic Publishing, 2012. 132 с.

[131] Описание бизнес-процессов: SADT, IDEF0, IDEF3, DFD, UML, ARIS : веб-сайт. URL: http://www.interface.ru/home.asp?artId=22559 (дата обращения: 21.10.2020)

[132] Бергер А.Б., Горбач И.В., Меломед Э.Л. Microsoft SQL Server Analysis Services. OLAP и многомерный анализ данных. СПб.: БХВ-Петербург, 2007. 928 с.

[133] Дворецький М.Л. Дворецька С.В. Оптимізація механізмів пошуку та оцінка якості знань на базі статистичних даних користувацьких запитів.

Інформаційні технології та взаємодії : матеріали міжнар. наук.-практ. конф., 8-10 лист. 2017 р. Київ, 2017. С. 157-158.

[134] SQL Profiling and Analysis : веб-сайт. URL: https://www.enterprisedb.com/edb-docs/d/edb-postgres-enterprise-manager/user-guides/enterprise-features-guide/7.10/sql_profiler.html (Last accessed: 22.10.2020).

[135] MySQL Reference Manual. MySQL Server Administration. MySQL Server Logs : веб-сайт. URL: https://dev.mysql.com/doc/refman/5.7/en/server-logs.html (Last accessed: 22.10.2020).

[136] Exposing PostgreSQL server logs to users via SQL : веб-сайт. URL: https://www.cybertec-postgresql.com/en/exposing-postgresql-server-logs-to-users-via-sql/?gclid=Cj0KCQjw28T8BRDbARIsAEOMBcxEPRqRU7aPdwYkYgfkvv_TvPwC8yNOFzjy-vlI001omiByJpGe_PUaApkOEALw_wcB (Last accessed: 22.10.2020).

[137] Приложение SQL Server Profiler : веб-сайт. URL: https://docs.microsoft.com/ru-ru/sql/tools/sql-server-profiler/sql-server-profiler?view=sql-server-ver15 (дата обращения: 22.10.2020).

[138] Свердлов С. З. Языки программирования и методы трансляции. Учебное пособие. СПб.: Издательство "Лань", 2016. 564 с.

[139] Малявко А. А. Программирование: формальные языки и компиляторы. Учебное пособие для вузов. М.: Юрайт, 2016. 429 с.

[140] Хопкрофт Дж., Мотвани Р., Ульман Д. Введение в теорию автоматов, языков и вычислений. М.: Изд. дом «Вильямс», 2008. 528 с.

[141] Грофф Дж. Р., Вайнберг П. Н., Оппель Э. Дж., SQL: полное руководство. Диалектика-Вильямс, 2020. 960 с.

[142] Дворецький М. Л. Створення перехресного набору даних (crosstab) засобами Transact SQL. Вісник Херсонського національного технічного університету. 2006. № 1 (24). С. 503–507.

[143] SQL Docs. Transact-SQL (T-SQL) reference. Queries. SELECT : веб-сайт. URL: https://docs.microsoft.com/en-us/sql/t-sql/queries/select-transact-sql?redirectedfrom=MSDN&view=sql-server-ver15 (Last accessed: 22.10.2020)

[144] Advanced IP Scanner. Сканування мережі за кілька секунд : веб-сайт. URL: https://www.advanced-ip-scanner.com/ua/ (дата звернення: 22.10.2020).

[145] NETWRIX INACTIVE USER TRACKERQUICK-STARTGUIDE : веб-сайт. URL: http://www.netwrix.com/download/QuickStart/Netwrix_Inactive_User_Tracker_Quick-Start_Guide.pdf (Last accessed: 22.10.2020).

[146] Parmavex Services. WinAudit Freeware : веб-сайт. URL: http://www.parmavex.co.uk/winaudit.html (Last accessed: 22.10.2020).

[147] Дворецький М.Л. Використання SQL/OLAP в T-SQL при автоматизації операцій консолідації та деталізації кросс-таблиць на базі реляційних джерел даних. Могилянські читання: досвід та тенденції розвитку суспільства в Україні: глобальний, національний та регіональний аспекти: матеріали всеукр. наук.-метод. конф. Миколаїв: Вид-во ЧНУ ім. П. Могили, 2014. С. 44-46.

[148] Russo M., Ferrari A. Tabular Modeling in Microsoft SQL Server Analysis Services (Developer Reference) 2nd Edition. Microsoft Press, 2017. 488 c.

[149] Fisun M., Horban H., Dvoretskyi M. Methods of Searching for Association Dependencies in Multidimensional Databases. 13th International Scientific and Technical Conference on Computer Sciences and Information Technologies (CSIT), Lviv. 2018, P. 88-93. doi: 10.1109/STC-CSIT.2018.8526737

[150] Документация по службам Analysis Services. Основные сведения об Analysis Services : веб-сайт. URL: https://docs.microsoft.com/ru-ru/analysis-services/analysis-services-overview?redirectedfrom=MSDN&view=asallproducts-allversions&viewFallbackFrom=sql-server-ver15 (дата обращения: 22.10.2020).

[151] Аурелиус М. Эволюция приложений или куда мы идем : веб-сайт. URL: https://habrahabr.ru/post/326016/ (дата обращения: 21.10.2020)

[152] Миковски М. С., Пауэлл Дж. К. Разработка одностраничных веб-приложений. ДМК Пресс, 2018. 512 с.

[153] Bass L., Clements P., Kazman R. Software Architecture in Practice (SEI Series in Software Engineering). Addison-Wesley Professional, 2012. 624 p.

[154] Symfony Documentation. Getting Started : веб-сайт. URL: http://symfony.com/doc/current/index.html (Last accessed: 22.10.2020).

[155] Дворецький М. Л., Дворецька С. В., Давиденко Є. О. Використання angular та symfony при реалізації web-орієнтованого застосунку автоматизації

обліку торгової точки. Наукові праці: Науково-методичний журнал Чорноморського національного університету ім. П. Могили. Сер. Комп'ютерні технології. 2018. Вип. 305. Т. 317. С. 70–77.

[156] Dunglas K. Persistence in PHP with Doctrine ORM. Packt Publishing, 2013. 114 p.

[157] Wage J. H., Vesterinen K. Doctrine ORM for PHP. Sensio SA, 2009. 522 p.

[158] Romer M. PHP Persistence: Concepts, Techniques and Practical Solutions with Doctrine. Apress, 2016. 122 p.

[159] Файн Я., Моисеев А. Angular и TypeScript. Сайтостроение для профессионалов. СПб.: Питер, 2018. 464 с.

[160] The Complete Guide to Angular. Published in San Francisco, California by Fullstack.io, 2017. 683 p.

[161] Гилат Амос. MATLAB. Теория и практика. Учебное пособие. ДМК Пресс, 2016. 416 с.

[162] Васильев А. Н. MATLAB. Самоучитель. Практический подход. Наука и Техника, 2015. 448 с.

[163] Fuzzy Logic Toolbox. Design and simulate fuzzy logic systems : веб-сайт. URL: https://www.mathworks.com/products/fuzzy-logic.html (Last accessed: 22.10.2020).

ДОДАТОК А

Акти впровадження результатів дисертації

ЗАТВЕРДЖУЮ
Начальник департаменту ІТ
ППФ "Юнікс Трейд Ко"
_____ О. С. Віннічeнко
«19» 08 2020 р.

АКТ
впровадження результатів дисертаційної роботи
Дворецького Михайла Леонідовича

«МОДЕЛІ ТА ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ ОПТИМІЗАЦІЇ СТРУКТУРИ БАЗИ ДАНИХ ВУЗЛА У КОРПОРАТИВНИХ ІНФОРМАЦІЙНИХ СИСТЕМАХ»

Комісія у складі начальника відділу розробки 1С Сакова Д. В. та інженера-програміста Ярошевича Д. А. встановила, що результати дисертаційної роботи Дворецького Михайла Леонідовича використані ППФ "Юнікс Трейд Ко" при проектуванні структури бази даних (БД) віддаленого вузла розподіленої комп'ютерної інформаційної системи (РКІС).

Використання розробленої інформаційної технології підтримки прийняття рішення при визначенні представленості даних на вузлі РКІС дозволило виявити закономірності між значеннями критеріїв оптимальності структури БД та рівнем представленості даних інформаційної системи на базі статистики SQL-запитів. Ефект від впровадження зазначених результатів полягає у підвищенні ефективності використання БД вузла РКІС на 12%.

Члени комісії:

Начальника відділу розробки 1С Д. В. Саков

Інженер-програміст
відділу web-розробки Д. А. Ярошевич

ТОВ «ЕЛІТ БІЛДІНГ»

04050, Київська обл., м. Київ, вул. Юрія Іллєнка, буд. 2/10, корпус 14

Код за ЄДРПОУ 41852437, ІПН 418524326596

Поштова адреса: 54003, м. Миколаїв, пр. Центральний, 259, т.(0512) 500470

№ 01/07

Від «_7_» _липня_ 2020 р.

АКТ
впровадження результатів дисертаційної роботи
Дворецького Михайла Леонідовича

«МОДЕЛІ ТА ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ ОПТИМІЗАЦІЇ СТРУКТУРИ БАЗИ ДАНИХ ВУЗЛА У КОРПОРАТИВНИХ ІНФОРМАЦІЙНИХ СИСТЕМАХ»

Цей акт засвідчує те, що **ТОВ «ЕЛІТ БІЛДІНГ»** були використані моделі та інформаційна технологія, створені в результаті наукових дисертаційних досліджень Дворецького М. Л. Розроблена технологія була застосована для виявлення закономірностей між значеннями критеріїв оптимальності структури БД та рівнем представленості даних на базі статистики SQL-запитів.

За допомогою СППР вибору найкращої альтернативи рівня представленості даних із використанням МАІ на базі багатовимірної моделі SQL-запиту було виконано оптимізацію структури бази даних віддаленого вузла розподіленої комп'ютерної інформаційної системи.

Ефект від впровадження зазначених результатів полягає у швидкості виконання SQL-запитів на віддаленому АРМ на 14% та підвищення рівня доступності даних на 21%.

Директор ТОВ «ЕЛІТ БІЛДІНГ» В. В. Покуль

ЗАТВЕРДЖУЮ
Перший проректор
Чорноморського національного
університету ім. П. Могили
_____ Н. М. Іщенко
«___» _____ 2020 р.

АКТ

про впровадження результатів дисертаційної роботи
Дворецького Михайла Леонідовича

«МОДЕЛІ ТА ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ ОПТИМІЗАЦІЇ СТРУКТУРИ БАЗИ ДАНИХ ВУЗЛА У КОРПОРАТИВНИХ ІНФОРМАЦІЙНИХ СИСТЕМАХ»

Комісія у складі: в. о. завідувача кафедри ІПЗ, к.т.н., доцента Давиденка Є. О., д.т.н., професора Фісуна М. Т., д.т.н., професора Коваленка І. І., встановила та склала цей акт у тому, що результати дисертаційного дослідження «Моделі та інформаційна технологія оптимізації структури бази даних вузла у корпоративних інформаційних системах» впроваджені у навчальний процес на кафедрі інженерії програмного забезпечення Чорноморського національного університету імені Петра Могили.

Матеріали дисертаційної роботи використовуються при викладанні дисциплін «Інформаційні технології, OLTP, OLAP та DM на клієнт-серверній платформі», «Клієнт-серверні СКБД та аналітичні системи», «Database development», а також у курсовому та дипломному проєктуванні для студентів спеціальності «Інженерія програмного забезпечення».

Комісія:

В. о. завідувача кафедри ІПЗ, к.т.н., доцент		Є. О. Давиденко
д.т.н., професор		М. Т. Фісун
д.т.н., професор		І. І. Коваленко

ЗАТВЕРДЖУЮ
Проректор з наукової роботи
Чорноморського національного
університету ім. П. Могили
_____ В. П. Беглиця
_____ 2020 р.

АКТ

про впровадження результатів дисертаційної роботи
Дворецького Михайла Леонідовича

«МОДЕЛІ ТА ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ ОПТИМІЗАЦІЇ СТРУКТУРИ БАЗИ ДАНИХ ВУЗЛА У КОРПОРАТИВНИХ ІНФОРМАЦІЙНИХ СИСТЕМАХ»

Ми, що нижче підписались, завідувач відділу аспірантури, докторантури ЧНУ ім. П. Могили, к.т.н., доцент Андрєєв В. І., науковий керівник професор, д.т.н., Фісун М. Т. склали цей акт у тому, що результати дисертаційної роботи Дворецького М. Л. «Моделі та інформаційна технологія оптимізації структури бази даних вузла у корпоративних інформаційних системах» увійшли у звіт з науково-дослідницької роботи «Розроблення найсучаснішого інтерактивного навчально-тренажерного та аналітично-консультативного комплексу військово-цивільного призначення» ЧНУ ім. П. Могили, Миколаїв, 2019 (№ 0118U000193, науковий керівник Фісун М. Т., виконавець Дворецький М. Л.)

Зав. відділу
аспірантури, докторантури _____ В. І. Андрєєв

д.т.н., професор _____ М. Т. Фісун

ДОДАТОК Б

Список публікацій за темою дисертації

1. Фісун М. Т., Дворецький М. Л. Пошук асоціативних правил засобами мови SQL при інтелектуальному аналізі даних. *Вісник Херсонського національного технічного університету*. 2005. № 1 (21). С. 185–189; **внесок автора:** реалізовано пошуку асоціативних правил засобами мови SQL та аналіз отриманих результатів; **база (и):** *Google Scholar, CrossRef*.

2. Фісун М. Т., Дворецький М. Л. Застосування OLAP-технологій в інформаційно-управляючих системах. *Інформаційні технології та комп'ютерна інженерія: Міжнародний науково-технічний журнал Вінницького національно-технічного університету.* 2006. №2(6). С. 128–134; **внесок автора:** засобами SQL Server на базі OLAP розроблено інформаційно аналітичну систему для аналізу трендів продажів торгівельного підприємства; **база (и):** *Google Scholar*.

3. Фісун М. Т., Дворецький М. Л. Синхронізаця оновлення даних в гетерогенних інформаційних системах. *Наукові праці: Науково-методичний журнал Миколаївського державного гуманітарного університету ім. П. Могили. Сер. Комп'ютерні технології.* 2006. Вип. 44. Т. 57. С. 61–66; **внесок автора:** запропоновано та виконано апробацію алгоритму розрахунку моменту наступної синхронізації БД різнорідних ІС; **база (и):** *Google Scholar*.

4. Дворецький М. Л. Реалізація задачі класифікації засобами мови Transact-SQL при інтелектуальному аналізі даних. *Системні технології: Регіональний міжвузівський збірник наукових праць. Дніпропетровськ, ДНВП «Системні технології»*. 2006. Випуск №6 (47). С. 102–111; **база (и):** *Google Scholar, Ulrichsweb, Index Copernicus*.

5. Дворецький М. Л. Створення перехресного набору даних (crosstab) засобами Transact SQL. *Вісник Херсонського національного технічного університету*. 2006. № 1 (24). С. 503–507; **база (и):** *Google Scholar, CrossRef*.

6. Дворецький М. Л. Розв’язання задачі класифікації в багатовимірних базах даних. *Вісник Херсонського національного технічного університету*. 2006. № 1 (30). С. 198–202; **база (и):** *Google Scholar, CrossRef*.

7. Дворецький М. Л. Інтелектуальний аналіз даних в 1С:8.0. *Наукові праці: Науково-методичний журнал Миколаївського державного гуманітарного університету ім. П. Могили. Сер. Комп’ютерні технології.* 2007. Вип. 55. Т. 68. С. 141–149; **база (и):** *Google Scholar*.

8. Дворецький М. Л. Проектування та оцінка оптимальності структури сховища даних та багатовимірної БД. *Наукові праці: Науково-методичний журнал Миколаївського державного гуманітарного університету ім. П. Могили. Сер. Комп’ютерні технології.* 2008. Вип. 77. Т. 90. С. 216–221; **база (и):** *Google Scholar*.

9. Дворецький М. Л. Інтеграція підсистем обліку та оперативно-аналітичної обробки даних в інформаційній системі супермаркету. *Комп’ютерні науки: освіта, наука, практика. Миколаїв. Національний університет кораблебудування*. 2014. С. 58–60; **база (и):** *Google Scholar, CrossRef*.

10. Дворецький М. Л. Використання SQL/OLAP в T-SQL при автоматизації операцій консолідації та деталізації кросс-таблиць на базі реляційних джерел даних. *Могилянські читання: досвід та тенденції розвитку суспільства в Україні: глобальний, національний та регіональний аспекти*: матеріали всеукр. наук.-метод. конф. Миколаїв: Вид-во ЧНУ ім. П. Могили, 2014. С. 44–46.

11. Дворецький М. Л., Динамічне формування запиту із використанням SQL/OLAP в T-SQL при автоматизації кросс-таблиць на базі реляційних джерел даних. *Наукові праці: Науково-методичний журнал Чорноморського державного університету ім. П. Могили. Сер. Комп’ютерні технології.* 2014. Вип. 238. Т. 250. С. 32–37; **база (и):** *Google Scholar, Ulrichsweb, Index Copernicus*.

12. Дворецький М. Л. Задача синхронізації даних в РБД. Синхронне та асинхронне оновлення. *Могилянські читання: досвід та тенденції розвитку суспільства в Україні: глобальний, національний та регіональний аспекти*:

матеріали всеукр. наук.-метод. конф., 14-18 лист. 2016 р. Миколаїв: Вид-во ЧНУ ім. П. Могили, 2016. С. 105–108.

13. Дворецький М. Л., Кулаковська І. В. Порівняльний ABC-XYZ аналіз на базі різних факторів із використанням ієрархічних даних. *Проблеми інформаційних технологій Херсонського національного технічного університету.* 2016. № 19. С. 200–209; **внесок автора:** запропоноване ієрархічне представлення даних та реалізовані механізми подальшого поєднання результатів аналізу за різними критеріями; **база (и):** *Index Copernicus, Google Scholar, Research Bible, Open Academic Journals Index, Directory of Open Access Journals.*.

14. Фісун М. Т., Дворецький М. Л., Юхатов А. В. Порівняльний аналіз методів побудови OLAP-систем із використанням засобів MS SQL SERVER та ORACLE. *Наукові праці: Науково-методичний журнал Чорноморського державного університету ім. П. Могили. Сер. Комп'ютерні технології.* 2016. Вип. 271. Т. 283. С. 36–42; **внесок автора:** виконано аналіз засобів та показників продуктивності щодо автоматизації OLAP на базі СКБД Oracle та SQL Server; **база (и):** *Google Scholar, Ulrichsweb, Index Copernicus*.

15. Дворецький М. Л., Давиденко Є. О., Боровльова С. Ю. Проектування структури розподіленої БД на базі парсингу SQL-запитів. *Наукові праці: Науково-методичний журнал Чорноморського державного університету ім. П. Могили. Сер. Комп'ютерні технології.* 2016. Вип. 275. Т. 287. С. 53–61; **внесок автора:** виконано розробку парсеру та багатовимірної моделі SQL-запиту, розроблено СППР оптимізації структури РБД; **база (и):** *Google Scholar, Ulrichsweb, Index Copernicus*.

16. Дворецький М. Л., Дворецька С. В. Оптимізація механізмів пошуку та оцінка якості знань на базі статистичних даних користувацьких запитів. *Інформаційні технології та взаємодії* : матеріали міжнар. наук.-практ. конф., 8–10 лист. 2017 р. Київ, 2017. С. 157–158; **внесок автора:** реалізація підсистем накопичення та аналізу статистики користувацьких запитів.

17. Fisun M., Dvoretskyi M., Shved A. Davydenko Y. Query parsing in order to optimize distributed DB structure. *9th IEEE International Conference on Intelligent*

Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS), Bucharest. 2017, pp. 172–178. doi: 10.1109/IDAACS.2017.8095071; **внесок автора:** розроблено технологію парсингу тексту запиту для оптимізації структури БД; **база (и):** *Google Scholar, Web of Science, SCOPUS, IEEE, DBLP.*.

18. Fisun M., Dvoretskyi M., Horban H. The usage of the feedback with user activities in company knowlagde management system. *12th International Scientific and Technical Conference on Computer Sciences and Information Technologies (CSIT), Lviv*. 2017. pp. 143–146. doi: 10.1109/STC-CSIT.2017.8098755; **внесок автора:** використання зворотнього зв'язку користувацької активності в системах управління знаннями та реалізація відповідної технології; **база (и):** *Google Scholar, SCOPUS, IEEE*.

19. Дворецький М. Л. Підходи щодо підвищення швидкості роботи SQL-запитів при використанні індексів БД. *Ольвійський форум: стратегії країн Причорноморського регіону в геополітичному просторі*: матеріали XII міжнар. наук.-практ. конф., 7-10 червня 2018 р. Миколаїв : Вид-во ЧНУ ім. Петра Могили, 2018. С. 30–32.

20. Дворецький М. Л., Дворецька С. В., Давиденко Є. О. Розробка системи управління знаннями організації на базі CMS WORDPRESS. *Проблеми інформаційних технологій Херсонського національного технічного університету.* 2018. №1 (023). С. 173–180; **внесок автора:** виконано розробку системи управління знаннями на базі CMS Wordpress; **база (и):** *Index Copernicus, Google Scholar, Research Bible, Open Academic Journals Index, Directory of Open Access Journals.*

21. Fisun M., Horban H., Dvoretskyi M. Methods of Searching for Association Dependencies in Multidimensional Databases. *13th International Scientific and Technical Conference on Computer Sciences and Information Technologies (CSIT), Lviv*. 2018, pp. 88–93. doi: 10.1109/STC-CSIT.2018.8526737; **внесок автора:** виконано аналіз результатів пошуку асоціацій на базі багатовимірної бази даних; **база (и):** *Google Scholar, SCOPUS, IEEE*.

22. Дворецький М. Л., Дворецька С. В., Давиденко Є. О. Використання angular та symfony при реалізації web-орієнтованого застосунку автоматизації обліку торгової точки. *Наукові праці: Науково-методичний журнал Чорноморського національного університету ім. П. Могили. Сер. Комп'ютерні технології.* 2018. Вип. 305. Т. 317. С. 70–77; **внесок автора:** реалізовано взаємодію бекенд та фронтенд фреймворків при розробці сервіс-орієнтованого вебзастосуноку; **база (и):** *Google Scholar, Ulrichsweb, Index Copernicus*.

23. Fisun M., Dvoretskyi M., Horban H., Komar M. Knowledge management applications based on user activities feedback. *International Journal of Computing, Ternopil*. 2019. 18 (1). pp. 32–44; **внесок автора:** дослідження впливу зворотнього зв'язку активності користувачів систем управління знаннями; **база (и):** *Google Scholar, SCOPUS*.

24. Дворецький М. Л., Боровльова С. Ю., Дворецька С. В. WEB-застосунок складського обліку в неавтоматизованих торгових точках. *Наукові праці: Науково-методичний журнал Чорноморського національного університету ім. П. Могили. Сер. Комп'ютерні технології.* 2018. Вип. 308 Т. 320. С. 45–52; **внесок автора:** виконано аналіз різних типів автоматизації облікових систем та реалізовано відповідний web-застосунок; **база (и):** *Google Scholar, Ulrichsweb, Index Copernicus*.

25. Дворецький М. Л., Дворецька С. В. Використання brain.js при визначенні корисності кортежу для віддалено вузла РБД. *Могилянські читання: досвід та тенденції розвитку суспільства в Україні: глобальний, національний та регіональний аспекти:* матеріали XXII Всеукр. наук.-метод. конф., 11-16 лист. 2019 р. Миколаїв: Вид-во ЧНУ ім. П. Могили, 2019. С. 123–125; **внесок автора:** реалізація підсистеми визначення корисності кортежу на вузлі РКІС із використанням нейронних мереж.

26. Дворецький М. Л., Дворецька С. В., Давиденко Є. О. Інформаційна технологія визначення корисних даних при оптимізації структури та мінімізації обсягів вузла розподіленої БД. *Вісник Черкаського державного технологічного університету*. 2019. № 4. С. 26–35. doi: 10.24025/2306-4412.4.2019.184808;

внесок автора: реалізація інформаційної технології оптимізації структури БД вузла РКІС; **база (и):** *Google Scholar, Crossref, Index Copernicus, Eurazian Scientific Journal Index, Directory of Open Access Journals.*

27. Dvoretskyi M., Dvoretska S., Nezdoliy Y., Borovlova S. Data Utility Assessment while Optimizing the Structure and Minimizing the Volume of a Distributed Database Node, *1st International Workshop on Information-Communication Technologies & Embedded Systems (ICTES), Mykolaiv, Ukraine, November 14-15.* 2019. pp. 128–137; **внесок автора:** вирішення задачі багатокритеріальної оптимізації при проєктуванні структури БД вузла РКІС; **база (и):** *SCOPUS.*

28. Фісун М. Т., Дворецький М. Л., Дворецька С. В. Побудова моделей для оптимізації структури бази даних вузла у корпоративних інформаційних системах. *Інформаційні технології та комп'ютерна інженерія: Міжнародний науково-технічний журнал Вінницького національно-технічного університету*. Vol 48, № 2. 2020. С. 52–60. doi: https://doi.org/10.31649/1999-9941-2020-48-2-52-60; **внесок автора:** реалізація моделей SQL-запиту та критеріїв оптимальності структури БД вузла РКІС; **база (и):** *Google Scholar*.

ДОДАТОК В

Скрипт створення структури БД обліку користувацьких запитів

```
/****** Object:  Table [dbo].[ApplicationList]    Script Date: 30.06.2020 18:14:56 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_PADDING ON
GO
CREATE TABLE [dbo].[ApplicationList](
	[id] [int] IDENTITY(1,1) NOT NULL,
	[name] [varchar](150) NULL,
 CONSTRAINT [PK_ApplicationList] PRIMARY KEY CLUSTERED
(
	[id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS =
ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]

GO
SET ANSI_PADDING OFF
GO
/****** Object:  Table [dbo].[FieldList]    Script Date: 30.06.2020 18:14:56 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_PADDING ON
GO
CREATE TABLE [dbo].[FieldList](
	[id] [int] IDENTITY(1,1) NOT NULL,
	[name] [varchar](50) NOT NULL,
	[relation_id] [int] NOT NULL,
	[marker] [numeric](3, 2) NULL,
 CONSTRAINT [PK_FieldList] PRIMARY KEY CLUSTERED
(
	[id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS =
ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]

GO
SET ANSI_PADDING OFF
GO
/****** Object:  Table [dbo].[InnerQueriesList]    Script Date: 30.06.2020 18:14:56 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[InnerQueriesList](
	[query_id] [int] NOT NULL,
	[inner_id] [int] NOT NULL,
 CONSTRAINT [PK_InnerQueriesList] PRIMARY KEY CLUSTERED
(
	[query_id] ASC,
	[inner_id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS =
ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]

GO
```

```
/****** Object:  Table [dbo].[Places]    Script Date: 30.06.2020 18:14:56 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_PADDING ON
GO
CREATE TABLE [dbo].[Places](
	[id] [int] IDENTITY(1,1) NOT NULL,
	[name] [varchar](150) NULL,
	[speed] [int] NULL,
	[reliability] [int] NULL,
 CONSTRAINT [PK_Places] PRIMARY KEY CLUSTERED
(
	[id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS =
ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]

GO
SET ANSI_PADDING OFF
GO
/****** Object:  Table [dbo].[profilingResultSet]    Script Date: 30.06.2020 18:14:56 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[profilingResultSet](
	[RowNumber] [int] IDENTITY(0,1) NOT NULL,
	[EventClass] [int] NULL,
	[TextData] [ntext] NULL,
	[ApplicationName] [nvarchar](128) NULL,
	[LoginName] [nvarchar](128) NULL,
	[Reads] [bigint] NULL,
	[Writes] [bigint] NULL,
	[Duration] [bigint] NULL,
	[SPID] [int] NULL,
	[DatabaseID] [int] NULL,
	[DatabaseName] [nvarchar](128) NULL,
	[HostName] [nvarchar](128) NULL,
	[ServerName] [nvarchar](128) NULL,
	[SessionLoginName] [nvarchar](128) NULL,
	[TransactionID] [bigint] NULL,
	[CPU] [int] NULL,
	[IsSystem] [int] NULL,
	[BinaryData] [image] NULL,
PRIMARY KEY CLUSTERED
(
	[RowNumber] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS =
ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]

GO
/****** Object:  Table [dbo].[QueriesLog]    Script Date: 30.06.2020 18:14:57 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[QueriesLog](
	[id] [int] IDENTITY(1,1) NOT NULL,
	[workstation_id] [int] NOT NULL,
	[application_id] [int] NOT NULL,
	[query_id] [int] NOT NULL,
```

```
	[transaction_id] [int] NOT NULL,
 CONSTRAINT [PK_QueriesLog] PRIMARY KEY CLUSTERED
(
	[id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS =
ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]

GO
/****** Object:  Table [dbo].[QueryList]    Script Date: 30.06.2020 18:14:57 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[QueryList](
	[id] [int] IDENTITY(1,1) NOT NULL,
	[text] [ntext] NOT NULL,
	[hash] [bigint] NOT NULL,
 CONSTRAINT [PK_QueryList] PRIMARY KEY CLUSTERED
(
	[id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS =
ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]

GO
/****** Object:  Table [dbo].[QueryRelationFields]    Script Date: 30.06.2020 18:14:57
******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[QueryRelationFields](
	[query_id] [int] NOT NULL,
	[field_id] [int] NOT NULL,
 CONSTRAINT [PK_QueryRelationFields] PRIMARY KEY CLUSTERED
(
	[query_id] ASC,
	[field_id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS =
ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]

GO
/****** Object:  Table [dbo].[QueryRelationRows]    Script Date: 30.06.2020 18:14:57 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[QueryRelationRows](
	[query_id] [int] NOT NULL,
	[row_id] [int] NOT NULL,
 CONSTRAINT [PK_QueryRelationRows] PRIMARY KEY CLUSTERED
(
	[query_id] ASC,
	[row_id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS =
ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]

GO
/****** Object:  Table [dbo].[QueryRelations]    Script Date: 30.06.2020 18:14:57 ******/
SET ANSI_NULLS ON
GO
```

```
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[QueryRelations](
	[query_id] [int] NOT NULL,
	[relation_id] [int] NOT NULL,
 CONSTRAINT [PK_QueryRelations] PRIMARY KEY CLUSTERED
(
	[query_id] ASC,
	[relation_id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS =
ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]

GO
/****** Object:  Table [dbo].[RelationList]    Script Date: 30.06.2020 18:14:57 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_PADDING ON
GO
CREATE TABLE [dbo].[RelationList](
	[id] [int] IDENTITY(1,1) NOT NULL,
	[name] [varchar](150) NOT NULL,
	[key_field_name] [varchar](150) NULL,
	[marker] [numeric](3, 2) NULL,
 CONSTRAINT [PK_RelationList] PRIMARY KEY CLUSTERED
(
	[id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS =
ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]

GO
SET ANSI_PADDING OFF
GO
/****** Object:  Table [dbo].[RowList]    Script Date: 30.06.2020 18:14:57 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[RowList](
	[id] [int] IDENTITY(1,1) NOT NULL,
	[relation_id] [int] NOT NULL,
	[row_id] [int] NOT NULL,
	[marker] [numeric](3, 2) NULL,
 CONSTRAINT [PK_RowList] PRIMARY KEY CLUSTERED
(
	[id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS =
ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]

GO
/****** Object:  Table [dbo].[Transactions]    Script Date: 30.06.2020 18:14:57 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_PADDING ON
GO
CREATE TABLE [dbo].[Transactions](
	[id] [int] NOT NULL,
	[number] [int] NOT NULL,
```

```
	[name] [varchar](150) NULL,
 CONSTRAINT [PK_Transactions] PRIMARY KEY CLUSTERED
(
	[id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS =
ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]

GO
SET ANSI_PADDING OFF
GO
/****** Object:  Table [dbo].[WorkPlaceType]    Script Date: 30.06.2020 18:14:57 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_PADDING ON
GO
CREATE TABLE [dbo].[WorkPlaceType](
	[id] [int] IDENTITY(1,1) NOT NULL,
	[name] [varchar](50) NULL,
 CONSTRAINT [PK_WorkPlaceType] PRIMARY KEY CLUSTERED
(
	[id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS =
ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]

GO
SET ANSI_PADDING OFF
GO
/****** Object:  Table [dbo].[WorkstApplList]    Script Date: 30.06.2020 18:14:57 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[WorkstApplList](
	[application_id] [int] NOT NULL,
	[workstation_id] [int] NOT NULL,
 CONSTRAINT [PK_WorkstApplList] PRIMARY KEY CLUSTERED
(
	[application_id] ASC,
	[workstation_id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS =
ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]

GO
/****** Object:  Table [dbo].[Workstation]    Script Date: 30.06.2020 18:14:57 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_PADDING ON
GO
CREATE TABLE [dbo].[Workstation](
	[id] [int] IDENTITY(1,1) NOT NULL,
	[name] [varchar](150) NULL,
	[place_id] [int] NULL,
	[workplacetype_id] [int] NULL,
 CONSTRAINT [PK_Workstation] PRIMARY KEY CLUSTERED
(
	[id] ASC
```

```
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS =
ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]

GO
SET ANSI_PADDING OFF
GO
ALTER TABLE [dbo].[FieldList]  WITH CHECK ADD  CONSTRAINT [FK_FieldList_RelationList] FOREIGN
KEY([relation_id])
REFERENCES [dbo].[RelationList] ([id])
GO
ALTER TABLE [dbo].[FieldList] CHECK CONSTRAINT [FK_FieldList_RelationList]
GO
ALTER TABLE [dbo].[InnerQueriesList]  WITH CHECK ADD  CONSTRAINT
[FK_InnerQueriesList_QueryList] FOREIGN KEY([inner_id])
REFERENCES [dbo].[QueryList] ([id])
GO
ALTER TABLE [dbo].[InnerQueriesList] CHECK CONSTRAINT [FK_InnerQueriesList_QueryList]
GO
ALTER TABLE [dbo].[InnerQueriesList]  WITH CHECK ADD  CONSTRAINT
[FK_InnerQueriesList_QueryList1] FOREIGN KEY([inner_id])
REFERENCES [dbo].[QueryList] ([id])
GO
ALTER TABLE [dbo].[InnerQueriesList] CHECK CONSTRAINT [FK_InnerQueriesList_QueryList1]
GO
ALTER TABLE [dbo].[QueriesLog]  WITH CHECK ADD  CONSTRAINT [FK_QueriesLog_ApplicationList]
FOREIGN KEY([application_id])
REFERENCES [dbo].[ApplicationList] ([id])
GO
ALTER TABLE [dbo].[QueriesLog] CHECK CONSTRAINT [FK_QueriesLog_ApplicationList]
GO
ALTER TABLE [dbo].[QueriesLog]  WITH CHECK ADD  CONSTRAINT [FK_QueriesLog_QueryList] FOREIGN
KEY([query_id])
REFERENCES [dbo].[QueryList] ([id])
GO
ALTER TABLE [dbo].[QueriesLog] CHECK CONSTRAINT [FK_QueriesLog_QueryList]
GO
ALTER TABLE [dbo].[QueriesLog]  WITH CHECK ADD  CONSTRAINT [FK_QueriesLog_Transactions]
FOREIGN KEY([transaction_id])
REFERENCES [dbo].[Transactions] ([id])
GO
ALTER TABLE [dbo].[QueriesLog] CHECK CONSTRAINT [FK_QueriesLog_Transactions]
GO
ALTER TABLE [dbo].[QueriesLog]  WITH CHECK ADD  CONSTRAINT [FK_QueriesLog_Workstation]
FOREIGN KEY([workstation_id])
REFERENCES [dbo].[Workstation] ([id])
GO
ALTER TABLE [dbo].[QueriesLog] CHECK CONSTRAINT [FK_QueriesLog_Workstation]
GO
ALTER TABLE [dbo].[QueryRelationFields]  WITH CHECK ADD  CONSTRAINT
[FK_QueryRelationFields_FieldList] FOREIGN KEY([field_id])
REFERENCES [dbo].[FieldList] ([id])
GO
ALTER TABLE [dbo].[QueryRelationFields] CHECK CONSTRAINT [FK_QueryRelationFields_FieldList]
GO
ALTER TABLE [dbo].[QueryRelationFields]  WITH CHECK ADD  CONSTRAINT
[FK_QueryRelationFields_QueryList] FOREIGN KEY([query_id])
REFERENCES [dbo].[QueryList] ([id])
GO
ALTER TABLE [dbo].[QueryRelationFields] CHECK CONSTRAINT [FK_QueryRelationFields_QueryList]
GO
ALTER TABLE [dbo].[QueryRelationRows]  WITH CHECK ADD  CONSTRAINT
[FK_QueryRelationRows_QueryList] FOREIGN KEY([query_id])
REFERENCES [dbo].[QueryList] ([id])
```

```
GO
ALTER TABLE [dbo].[QueryRelationRows] CHECK CONSTRAINT [FK_QueryRelationRows_QueryList]
GO
ALTER TABLE [dbo].[QueryRelationRows]  WITH CHECK ADD  CONSTRAINT
[FK_QueryRelationRows_RowList] FOREIGN KEY([row_id])
REFERENCES [dbo].[RowList] ([id])
GO
ALTER TABLE [dbo].[QueryRelationRows] CHECK CONSTRAINT [FK_QueryRelationRows_RowList]
GO
ALTER TABLE [dbo].[QueryRelations]  WITH CHECK ADD  CONSTRAINT [FK_QueryRelations_QueryList]
FOREIGN KEY([query_id])
REFERENCES [dbo].[QueryList] ([id])
GO
ALTER TABLE [dbo].[QueryRelations] CHECK CONSTRAINT [FK_QueryRelations_QueryList]
GO
ALTER TABLE [dbo].[QueryRelations]  WITH CHECK ADD  CONSTRAINT
[FK_QueryRelations_RelationList] FOREIGN KEY([relation_id])
REFERENCES [dbo].[RelationList] ([id])
GO
ALTER TABLE [dbo].[QueryRelations] CHECK CONSTRAINT [FK_QueryRelations_RelationList]
GO
ALTER TABLE [dbo].[RowList]  WITH CHECK ADD  CONSTRAINT [FK_RowList_RelationList] FOREIGN
KEY([relation_id])
REFERENCES [dbo].[RelationList] ([id])
GO
ALTER TABLE [dbo].[RowList] CHECK CONSTRAINT [FK_RowList_RelationList]
GO
ALTER TABLE [dbo].[WorkstApplList]  WITH CHECK ADD  CONSTRAINT
[FK_WorkstApplList_ApplicationList] FOREIGN KEY([application_id])
REFERENCES [dbo].[ApplicationList] ([id])
GO
ALTER TABLE [dbo].[WorkstApplList] CHECK CONSTRAINT [FK_WorkstApplList_ApplicationList]
GO
ALTER TABLE [dbo].[WorkstApplList]  WITH CHECK ADD  CONSTRAINT
[FK_WorkstApplList_Workstation] FOREIGN KEY([workstation_id])
REFERENCES [dbo].[Workstation] ([id])
GO
ALTER TABLE [dbo].[WorkstApplList] CHECK CONSTRAINT [FK_WorkstApplList_Workstation]
GO
ALTER TABLE [dbo].[Workstation]  WITH CHECK ADD  CONSTRAINT [FK_Workstation_Places] FOREIGN
KEY([place_id])
REFERENCES [dbo].[Places] ([id])
GO
ALTER TABLE [dbo].[Workstation] CHECK CONSTRAINT [FK_Workstation_Places]
GO
ALTER TABLE [dbo].[Workstation]  WITH CHECK ADD  CONSTRAINT [FK_Workstation_WorkPlaceType]
FOREIGN KEY([workplacetype_id])
REFERENCES [dbo].[WorkPlaceType] ([id])
GO
ALTER TABLE [dbo].[Workstation] CHECK CONSTRAINT [FK_Workstation_WorkPlaceType]
GO
/****** Object:  StoredProcedure [dbo].[ExtractObjFromSQL]    Script Date: 30.06.2020
18:14:57 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE procedure [dbo].[ExtractObjFromSQL] as
declare @str varchar(8000), @chr varchar(1), @word varchar(255), @levelnum smallint
declare @words table(id int identity(1,1), word varchar(255), levelnum smallint)

select * from @words

select @str=cast(textdata as varchar(8000)) from trace_log
```

```
where DatabaseName='kvint_dt' and RowNumber=4237
select @str
set @word=''
while LEN(@str)>0
begin
	--вычленяем слово
	set @chr=LEFT(@str,1)
	--это конец слова, пишем его в таблицу, если оно не пустое
	if exists(select id from keywords where word=@chr and eow=1)
		begin
			if (@word<>'')
				insert into @words(word) values(@word)
			set @word=''
		end

	if not(exists(select id from keywords where word=@chr and ign=1))
		set @word=@word+@chr

	set @str=RIGHT(@str,len(@str)-1)

	--если это отдельное слова, пишем его в таблицу
	if exists(select id from keywords where word=@chr and is_own=1)
		begin
			if (@word<>'')
				insert into @words(word,levelnum) values(@word,@levelnum)
			set @word=''
		end
end

--пишем ссылки на скобки
declare @brackets table(id int, word varchar(1), brnum smallint, brparnum smallint)

declare @brnum smallint, @brparnum smallint, @brmaxnum smallint, @id int, @brk varchar(1)
select @brnum=0, @brparnum=0, @brmaxnum=0

declare cur cursor local for select id, word from @words where word in('(',')')
open cur
fetch next from cur into @id, @brk
while @@FETCH_STATUS=0
begin
	if (@brk='(')
	begin
		set @brparnum=@brnum
		set @brmaxnum=@brmaxnum+1
		set @brnum=@brmaxnum
	end

	insert into @brackets(id, word, brnum, brparnum) values(@id, @brk, @brnum, @brparnum)

	if (@brk=')')
		select @brnum=brnum, @brparnum=brparnum from @brackets where brnum=@brparnum
and word='('
	fetch next from cur into @id, @brk
end
close cur
deallocate cur

select * from @words
select * from @brackets
GO
/****** Object:  StoredProcedure [dbo].[fillTables]    Script Date: 30.06.2020 18:14:57
******/
SET ANSI_NULLS ON
GO
```

```
SET QUOTED_IDENTIFIER ON
GO
CREATE proc [dbo].[fillTables] @remove tinyint = 0
as
if @remove = 1
begin
	delete from WorkstApplList

	delete from ApplicationList
	delete from Workstation

	delete from fieldList
	delete from RelationList
end

--fill application list
insert into ApplicationList(name)
select distinct ApplicationName from profilingResultSet where ApplicationName is not null
	and ApplicationName not in(select name from ApplicationList)

--fill workstation list
insert into Workstation(name, place_id, workplacetype_id)
select distinct hostname, 1, 1 from profilingResultSet where HostName is not null
	and HostName not in (select name from Workstation)

--fill workstation's applications
insert into WorkstApplList(application_id, workstation_id)
select t.*
from
(select distinct a.id aid, w.id wid
from profilingResultSet p inner join ApplicationList a on p.ApplicationName = a.name
inner join Workstation w on p.HostName = w.name
where ApplicationName is not null and HostName is not null ) t
left join WorkstApplList w on t.aid = w.application_id and t.wid = w.workstation_id
where w.application_id is null

--fill list of tables
insert into RelationList(name)
select name from kvint_dt.dbo.sysobjects
where type = 'U' and lower(name) not in(select lower(name) COLLATE Cyrillic_General_CI_AS
from RelationList)

--fill fields list
insert into fieldList(name, relation_id)
select t.*
from
(SELECT sc.name cname, r.id
FROM kvint_dt.dbo.syscolumns sc
 JOIN kvint_dt.dbo.sysobjects so ON sc.id = so.id
 inner join RelationList r on so.name = r.name collate Cyrillic_General_CI_AS) t
 left join fieldList f on t.cname collate SQL_Latin1_General_CP1251_CI_AS = f.name and t.id =
f.relation_id
 where f.id is null

GO
USE [master]
GO
ALTER DATABASE [QueryProfileSystem] SET  READ_WRITE
GO
```

ДОДАТОК Г

Граматика БНФ для опису мови T-SQL

```
grammar tsql;

@header {

import java.util.*;
import proj.model.*;
}

tsql_file returns [LinkedList<TsqlCommand> CommandList]
    : (batch {$CommandList = $batch.CommandList; }) * EOF
    ;

batch returns [LinkedList<TsqlCommand> CommandList]
    : (sql_clauses {$CommandList = $sql_clauses.CommandList; }) go_statement?
    ;

sql_clauses returns [LinkedList<TsqlCommand> CommandList]
@init{$CommandList = new LinkedList<TsqlCommand>();}
    : (sql_clause SEMI? {
        $CommandList.add($sql_clause.Command);
    })+
    ;

sql_clause returns [TsqlCommand Command]
    : dml_clause {
        $Command = $dml_clause.Command;
    }

    | ddl_clause {$Command = new TsqlCommand(TsqlCommand.NOTDML);}

    | cfl_statement {$Command = new TsqlCommand(TsqlCommand.NOTDML);}

    | another_statement {$Command = new TsqlCommand(TsqlCommand.NOTDML);}
    ;

// Data Manipulation Language: https://msdn.microsoft.com/en-
us/library/ff848766(v=sql.120).aspx
dml_clause returns [TsqlCommand Command]
    : delete_statement {$Command = new TsqlCommand(TsqlCommand.DELETE);}
    | insert_statement {$Command = new TsqlCommand(TsqlCommand.INSERT);}
    | select_statement {
        $Command = new TsqlCommand();
        $Command = TsqlCommand.getValueFrom($Command, $select_statement.Command);
    }
    | update_statement {$Command = new TsqlCommand(TsqlCommand.UPDATE);}
    ;

// Data Definition Language: https://msdn.microsoft.com/en-us/library/ff848799.aspx)
ddl_clause
    : //create_function
    create_database
    | create_index
    | create_procedure
    | create_statistics
    | create_table
    | create_type
```

```
    | create_view

    | alter_table
    | alter_database

    | drop_index
    | drop_procedure
    | drop_statistics
    | drop_table
   | drop_type
    | drop_view
    ;

// Control-of-Flow Language: https://msdn.microsoft.com/en-us/library/ms174290.aspx
// Labels for better AST traverse.
cfl_statement
    : BEGIN ';'? sql_clauses? END ';'?              #block_statement
    | BREAK ';'?                                    #break_statement
    | CONTINUE ';'?                                 #continue_statement
    | GOTO id ';'?                                  #goto_statement
    | id ':' ';'?                                   #goto_statement
    // https://msdn.microsoft.com/en-us/library/ms182717.aspx
    | IF search_condition sql_clause (ELSE sql_clause)? ';'?  #if_statement
    | RETURN expression? ';'?                       #return_statement
    | THROW (
      error_number=(DECIMAL | LOCAL_ID) ',' message=(STRING | LOCAL_ID) ','
      state=(DECIMAL | LOCAL_ID))? ';'?             #throw_statement
    | BEGIN TRY ';'? try_clauses=sql_clauses? END TRY ';'?
      BEGIN CATCH ';'? catch_clauses=sql_clauses? END CATCH ';'?
#try_catch_statement
    | WAITFOR (DELAY | TIME)  expression ';'?
#waitfor_statement
    | WHILE search_condition (sql_clause | BREAK ';'? | CONTINUE ';'?)
#while_statement
    | PRINT expression ';'?
#print_statement
    | RAISERROR '(' msg=(DECIMAL | STRING | LOCAL_ID) ',' severity=constant_LOCAL_ID ','
        state=constant_LOCAL_ID (',' constant_LOCAL_ID)* ')' ';'?
#raiseerror_statement
    ;

another_statement
    : declare_statement
    | cursor_statement
    | execute_statement
    | security_statement
    | set_statement
    | transaction_statement
    | use_statement
    ;

delete_statement
    : with_expression?
      DELETE (TOP '(' expression ')' PERCENT?)?
      FROM? delete_statement_from
      insert_with_table_hints?
      output_clause?
      (FROM table_sources)?
      (WHERE (search_condition | CURRENT OF (GLOBAL? cursor_name |
cursor_var=LOCAL_ID)))?
      for_clause? option_clause? ';'?
```

```
    ;

delete_statement_from
    : table_alias
    | ddl_object
    | rowset_function_limited
    | table_var=LOCAL_ID
    ;

insert_statement
    : with_expression?
      INSERT (TOP '(' expression ')' PERCENT?)?
      INTO? (ddl_object | rowset_function_limited)
      insert_with_table_hints?
      ('(' column_name_list ')')?
      output_clause?
      insert_statement_value
      for_clause? option_clause? ';'?
    ;

insert_statement_value
    : table_value_constructor
    | derived_table
    | execute_statement
    | DEFAULT VALUES
    ;

select_statement returns [HashSet<TsqlRelation> RelationList, TsqlCommand Command]
@init {
    $Command = new TsqlCommand();
}
    : with_expression? (query_expression {
        $RelationList = $query_expression.RelationList;
        $Command = TsqlCommand.getValueFrom($Command, $query_expression.Command);
    }) order_by_clause? for_clause? option_clause? ';'?
    ;

update_statement
    : with_expression?
      UPDATE (TOP '(' expression ')' PERCENT?)?
      (ddl_object | rowset_function_limited)
      with_table_hints?
      SET update_elem (',' update_elem)*
      output_clause?
      (FROM table_sources)?
      (WHERE (search_condition_list | CURRENT OF (GLOBAL? cursor_name |
cursor_var=LOCAL_ID)))?
      for_clause? option_clause? ';'?
    ;

output_clause
    : OUTPUT output_dml_list_elem (',' output_dml_list_elem)*
      (INTO (LOCAL_ID | table_name) ('(' column_name_list ')')? )?
    ;

output_dml_list_elem
    : (output_column_name | expression) (AS? column_alias)?  // TODO: scalar_expression
    ;

output_column_name
    : (DELETED | INSERTED | table_name) '.' ('*' | id)
```

```
    | DOLLAR_ACTION
    ;

declare_statement
    : DECLARE LOCAL_ID AS? table_type_definition ';'?
    | DECLARE declare_local (',' declare_local)* ';'?
    ;

cursor_statement
    // https://msdn.microsoft.com/en-us/library/ms175035(v=sql.120).aspx
    : CLOSE GLOBAL? cursor_name ';'?
    // https://msdn.microsoft.com/en-us/library/ms188782(v=sql.120).aspx
    | DEALLOCATE GLOBAL? cursor_name ';'?
    // https://msdn.microsoft.com/en-us/library/ms180169(v=sql.120).aspx
    | declare_cursor
    // https://msdn.microsoft.com/en-us/library/ms180152(v=sql.120).aspx
    | fetch_cursor
    // https://msdn.microsoft.com/en-us/library/ms190500(v=sql.120).aspx
    | OPEN GLOBAL? cursor_name ';'?
    ;

execute_statement
    : EXECUTE (return_status=LOCAL_ID '=')? func_proc_name (execute_statement_arg (','
execute_statement_arg)*)? ';'?
    | EXECUTE '(' execute_var_string ('+' execute_var_string)* ')' (AS? (LOGIN | USER)
'=' STRING)? ';'?
    ;

execute_statement_arg
    : (parameter=LOCAL_ID '=')? ((constant_LOCAL_ID | id) (OUTPUT | OUT)? | DEFAULT |
NULL)
    ;

execute_var_string
    : LOCAL_ID
    | STRING
    ;

set_statement
    : SET LOCAL_ID ('.' member_name=id)? '=' expression ';'?
    | SET LOCAL_ID assignment_operator expression ';'?
    | SET LOCAL_ID '='
      CURSOR declare_set_cursor_common (FOR (READ ONLY | UPDATE (OF
column_name_list)?))? ';'?
    // https://msdn.microsoft.com/en-us/library/ms189837.aspx
    | set_special
    ;

transaction_statement
    : BEGIN DISTRIBUTED (TRAN | TRANSACTION) (id | LOCAL_ID)? ';'?
    | BEGIN (TRAN | TRANSACTION) ((id | LOCAL_ID) (WITH MARK STRING)?)? ';'?
    | COMMIT (TRAN | TRANSACTION) ((id | LOCAL_ID) (WITH '(' DELAYED_DURABILITY EQUAL
(OFF | ON) ')')?)? ';'?
    | COMMIT WORK? ';'?
    | ROLLBACK (TRAN | TRANSACTION) (id | LOCAL_ID)? ';'?
    | ROLLBACK WORK? ';'?
    | SAVE (TRAN | TRANSACTION) (id | LOCAL_ID)? ';'?
    ;

go_statement
    : GO (count=DECIMAL)?
```

```
    ;

execute_clause
    : EXECUTE AS clause=(CALLER | SELF | OWNER | STRING)
    ;

declare_local
    : LOCAL_ID AS? data_type ('=' expression)?
    ;

table_type_definition
    : TABLE '(' column_def_table_constraints ')'
    ;

column_def_table_constraints
    : column_def_table_constraint (','? column_def_table_constraint)*
    ;

column_def_table_constraint
    : column_definition
    | table_constraint
    ;

column_definition
    : id (data_type | AS expression) (COLLATE id)? null_notnull?
      ((CONSTRAINT constraint=id)? DEFAULT constant_expression (WITH VALUES)?
       | IDENTITY ('(' seed=DECIMAL ',' increment=DECIMAL ')')? (NOT FOR REPLICATION)?)?
      ROWGUIDCOL?
      column_constraint*
    ;

column_constraint
    :(CONSTRAINT id)? null_notnull?
      ((PRIMARY KEY | UNIQUE) clustered? index_options?
      | CHECK (NOT FOR REPLICATION)? '(' search_condition ')')
    ;

table_constraint
    : (CONSTRAINT id)?
       ((PRIMARY KEY | UNIQUE) clustered? '(' column_name_list (ASC | DESC)? ')'
index_options? (ON id)?
       | CHECK (NOT FOR REPLICATION)? '(' search_condition ')')
    ;

index_options
    : WITH '(' index_option (',' index_option)* ')'
    ;

declare_cursor
    : DECLARE cursor_name
      (CURSOR (declare_set_cursor_common (FOR UPDATE (OF column_name_list)?)?)?
      | INSENSITIVE? SCROLL? CURSOR FOR select_statement (FOR (READ ONLY | UPDATE | (OF
column_name_list)))?
      ) ';'?
    ;

declare_set_cursor_common
    : (LOCAL | GLOBAL)?
      (FORWARD_ONLY | SCROLL)? (STATIC | KEYSET | DYNAMIC | FAST_FORWARD)?
      (READ_ONLY | SCROLL_LOCKS | OPTIMISTIC)? TYPE_WARNING?
      FOR select_statement
```

```
    ;

fetch_cursor
    : FETCH ((NEXT | PRIOR | FIRST | LAST | (ABSOLUTE | RELATIVE) expression)? FROM)?
      GLOBAL? cursor_name (INTO LOCAL_ID (',' LOCAL_ID)*)? ';'?
    ;

set_special
    : SET id (id | constant_LOCAL_ID | on_off) ';'?
    // https://msdn.microsoft.com/en-us/library/ms173763.aspx
    | SET TRANSACTION ISOLATION LEVEL
      (READ UNCOMMITTED | READ COMMITTED | REPEATABLE READ | SNAPSHOT | SERIALIZABLE)
';'?
    | SET IDENTITY_INSERT table_name on_off ';'?
    | SET ANSI_NULLS on_off
    | SET QUOTED_IDENTIFIER on_off
    | SET ANSI_PADDING on_off
    ;

expression returns [HashSet<TsqlRelation> RelationList, TsqlCommand Command, TsqlField
Field]
@init {
    $Command = new TsqlCommand();
    $RelationList = new HashSet<TsqlRelation>();
    $Field = new TsqlField();
}
    : DEFAULT                                                   #primitive_expression
    | NULL                                                      #primitive_expression
    | LOCAL_ID                                                  #primitive_expression
    | constant                                                  #primitive_expression
    | function_call                                             #function_call_expression
    | expression COLLATE id                                     #function_call_expression
    // https://msdn.microsoft.com/en-us/library/ms181765.aspx
    | CASE caseExpr=expression switch_section+ (ELSE elseExpr=expression)? END
#case_expression
    | CASE switch_search_condition_section+ (ELSE elseExpr=expression)? END
#case_expression
    | full_column_name {
        $Field = $full_column_name.Field;
    }                                       #column_ref_expression
    | '(' expression ')'                                        #bracket_expression
    | '(' subquery {
//          $RelationList = $subquery.RelationList;
          $Command = TsqlCommand.getValueFrom($Command, $subquery.Command);
       }')'                                             #subquery_expression
    | '~' expression
#unary_operator_expression

    | expression op=('*' | '/' | '%') expression
#binary_operator_expression
    | op=('+' | '-') expression
#unary_operator_expression
    | expression op=('+' | '-' | '&' | '^' | '|') expression
#binary_operator_expression
    | expression comparison_operator expression
#binary_operator_expression

    | over_clause                                               #over_clause_expression
    ;

constant_expression
```

```
    : NULL
    | constant
    // system functions: https://msdn.microsoft.com/en-us/library/ms187786.aspx
    | function_call
    | LOCAL_ID         // TODO: remove.
    | '(' constant_expression ')'
    ;

subquery returns [HashSet<TsqlRelation> RelationList, TsqlCommand Command]
@init {
    $Command = new TsqlCommand();
}
    : select_statement {
        $RelationList = $select_statement.RelationList;
        $Command = TsqlCommand.getValueFrom($Command, $select_statement.Command);
    }
    ;

with_expression
    : WITH (XMLNAMESPACES ',')? common_table_expression (',' common_table_expression)*
    ;

common_table_expression
    : expression_name=id ('(' column_name_list ')')? AS '(' select_statement ')'
    ;

update_elem
    : (full_column_name | LOCAL_ID) ('=' | assignment_operator) expression
    | udt_column_name=id '.' method_name=id '(' expression_list ')'
    //| full_column_name '.' WRITE (expression, )
    ;

search_condition_list returns [HashSet<TsqlRelation> RelationList, TsqlCommand Command]
@init{
  $RelationList = new HashSet<TsqlRelation>();
  $Command = new TsqlCommand();
}
    : search_condition (',' search_condition)*
    ;

search_condition returns [HashSet<TsqlRelation> RelationList, TsqlCommand Command,
HashSet<TsqlField> Fields]
@init{
  $RelationList = new HashSet<TsqlRelation>();
  $Command = new TsqlCommand();
  $Fields = new HashSet<TsqlField>();
}
    : search_condition_and {
        $RelationList.addAll($search_condition_and.RelationList);
        $Command.transferInnerCommands($search_condition_and.Command);
        $Fields.addAll($search_condition_and.Fields);
    } (OR search_condition_and {
        $RelationList.addAll($search_condition_and.RelationList);
        $Command.transferInnerCommands($search_condition_and.Command);
        $Fields.addAll($search_condition_and.Fields);
    })*
    ;

search_condition_and returns [HashSet<TsqlRelation> RelationList, TsqlCommand Command,
HashSet<TsqlField> Fields]
@init{
```

```
  $RelationList = new HashSet<TsqlRelation>();
  $Command = new TsqlCommand();
  $Fields = new HashSet<TsqlField>();
}
    : search_condition_not {
       $RelationList.addAll($search_condition_not.RelationList);
       $Command.transferInnerCommands($search_condition_not.Command);
       $Fields.addAll($search_condition_not.Fields);
    } (AND search_condition_not {
        $RelationList.addAll($search_condition_not.RelationList);
        $Command.transferInnerCommands($search_condition_not.Command);
        $Fields.addAll($search_condition_not.Fields);
    })*
    ;

search_condition_not returns [HashSet<TsqlRelation> RelationList, TsqlCommand Command,
HashSet<TsqlField> Fields]
@init{
  $RelationList = new HashSet<TsqlRelation>();
  $Command = new TsqlCommand();
  $Fields = new HashSet<TsqlField>();
}
    : NOT? predicate {
         $RelationList.addAll($predicate.RelationList);
         $Fields.addAll($predicate.Fields);
         $Command.transferInnerCommands($predicate.Command);
    }
    ;

predicate returns [HashSet<TsqlRelation> RelationList, TsqlCommand Command,
HashSet<TsqlField> Fields]
@init{
    $RelationList = new HashSet<TsqlRelation>();
    $Command = new TsqlCommand();
    $Fields = new HashSet<TsqlField>();
}
    : EXISTS '(' subquery ')'
    | expression {
        $Fields.add($expression.Field);
    } comparison_operator expression {
        $Fields.add($expression.Field);
    }
    | expression comparison_operator (ALL | SOME | ANY) '(' subquery {
//         $RelationList.addAll($subquery.RelationList);
         $Command.addInnerCommand($subquery.Command);
         $Fields.add($expression.Field);
    }')'
    | expression {$Fields.add($expression.Field);} NOT? BETWEEN expression AND
expression
    | expression NOT? IN '(' (subquery {
//        $RelationList.addAll($subquery.RelationList);
        $Command.addInnerCommand($subquery.Command);
        $Fields.add($expression.Field);
    } | expression_list) ')'
    | expression {$Fields.add($expression.Field);} NOT? LIKE expression (ESCAPE
expression)?
    | expression {$Fields.add($expression.Field);} IS null_notnull
    | '(' search_condition ')'
   | DECIMAL
    ;
```

```
query_expression returns [HashSet<TsqlRelation> RelationList, TsqlCommand Command]
@init{
    $RelationList = new HashSet<TsqlRelation>();
    $Command = new TsqlCommand();
}
    : (query_specification {
        $RelationList.addAll($query_specification.RelationList);
        $Command = TsqlCommand.getValueFrom($Command, $query_specification.Command);
        $Command.setCommandText($query_specification.text);
    }
    | '(' inexpr = query_expression {
       $RelationList.addAll($inexpr.RelationList);
       $Command = TsqlCommand.getValueFrom($Command, $query_expression.Command);
    }  ')'  )
    (union {
        $RelationList.addAll($union.RelationList);
        $Command = TsqlCommand.getValueFrom($Command, $union.Command);
    }) *
    ;

union returns [HashSet<TsqlRelation> RelationList, TsqlCommand Command]
@init{
    $RelationList = new HashSet<TsqlRelation>();
    $Command = new TsqlCommand();
}
    : (UNION ALL? | EXCEPT | INTERSECT)
       (query_specification {
           $RelationList.addAll($query_specification.RelationList);
           $Command = TsqlCommand.getValueFrom($Command, $query_specification.Command);
           $Command.setCommandText($query_specification.text);
       }
       | ('(' query_expression ')' {
           $RelationList.addAll($query_expression.RelationList);
           $Command = TsqlCommand.getValueFrom($Command, $query_expression.Command);
       })+
    )
    ;

query_specification returns [
    HashSet<TsqlRelation> RelationList,
    TsqlCommand Command,
    TsqlCommand SelectCommand,
    TsqlCommand SearchCommand
]
@init{
    $RelationList = new HashSet<TsqlRelation>();
    $Command = new TsqlCommand(TsqlCommand.SELECT);
    $SelectCommand = new TsqlCommand();
    $SearchCommand = new TsqlCommand();
}
@after{
    $Command.setRelationList($RelationList);
    $Command.transferInnerCommands($SelectCommand);
    $Command.transferInnerCommands($SearchCommand);
}
     : SELECT (ALL | DISTINCT)? (TOP expression PERCENT? (WITH TIES)?)?
      select_list {
        $RelationList.addAll($select_list.RelationList);
        $SelectCommand = $select_list.Command;
        $Command.setFieldList($select_list.Command.getFieldList());
      }
```

```
      // https://msdn.microsoft.com/en-us/library/ms188029.aspx
      (INTO table_name)?
      (FROM table_sources {
            $RelationList.addAll($table_sources.RelationList);
            $Command = TsqlCommand.getValueFrom($Command, $table_sources.Command);
      })?
      (WHERE where=search_condition {
            $RelationList.addAll($search_condition.RelationList);
            $SearchCommand = $search_condition.Command;

            HashSet<TsqlField> fldList = $Command.getFieldList();
            fldList.addAll($search_condition.Fields);
            $Command.setFieldList(fldList);
      })?
      // https://msdn.microsoft.com/en-us/library/ms177673.aspx
      (GROUP BY group_by_item (',' group_by_item)*)?
      (HAVING having=search_condition)?
    ;

order_by_clause
    : ORDER BY order_by_expression (',' order_by_expression)*
      (OFFSET expression (ROW | ROWS) (FETCH (FIRST | NEXT) expression (ROW | ROWS)
ONLY)?)?
    ;

for_clause
    : FOR BROWSE
    | FOR XML AUTO xml_common_directives?
    | FOR XML PATH ('(' STRING ')')? xml_common_directives?
    ;

order_by_expression
    : expression (ASC | DESC)?
    ;

group_by_item
    : expression
    /*| rollup_spec
    | cube_spec
    | grouping_sets_spec
    | grand_total*/
    ;

option_clause
    : OPTION '(' option (',' option)* ')'
    ;

option
    : FAST number_rows=DECIMAL
    | (HASH | ORDER) GROUP
    | (MERGE | HASH | CONCAT) UNION
    | (LOOP | MERGE | HASH) JOIN
    | EXPAND VIEWS
    | FORCE ORDER
    | IGNORE_NONCLUSTERED_COLUMNSTORE_INDEX
    | KEEP PLAN
    | KEEPFIXED PLAN
    | MAXDOP number_of_processors=DECIMAL
    | MAXRECURSION number_recursion=DECIMAL
    | OPTIMIZE FOR '(' optimize_for_arg (',' optimize_for_arg)* ')'
    | OPTIMIZE FOR UNKNOWN
```

```
    | PARAMETERIZATION (SIMPLE | FORCED)
    | RECOMPILE
    | ROBUST PLAN
    | USE PLAN STRING
    ;

optimize_for_arg
    : LOCAL_ID (UNKNOWN | '=' constant)
    ;

// https://msdn.microsoft.com/en-us/library/ms176104.aspx
select_list returns [HashSet<TsqlRelation> RelationList, TsqlCommand Command]
@init{
    $RelationList = new HashSet<TsqlRelation>();
    $Command = new TsqlCommand();
}
    : select_list_elem {
        $RelationList.addAll($select_list_elem.RelationList);
        $Command.addInnerCommand($select_list_elem.Command);
        $Command.addField($select_list_elem.Field);
    } (',' select_list_elem {
        $RelationList.addAll($select_list_elem.RelationList);
        $Command.addInnerCommand($select_list_elem.Command);
        $Command.addField($select_list_elem.Field);
    })*
    ;

select_list_elem returns [HashSet<TsqlRelation> RelationList, TsqlCommand Command,
TsqlField Field]
@init{
    $RelationList = new HashSet<TsqlRelation>();
    $Command = new TsqlCommand();
    $Field = new TsqlField();
}
    : (table_name '.')? ('*' | '$' (IDENTITY | ROWGUID))
    | column_alias '=' expression {
        $RelationList = $expression.RelationList;
        $Command = TsqlCommand.getValueFrom($Command, $expression.Command);
    }
    | expression {
         $RelationList = $expression.RelationList;
         $Command = TsqlCommand.getValueFrom($Command, $expression.Command);

         $Field.fName = $expression.Field.fName;
         $Field.tName = $expression.Field.tName;
     } (AS? column_alias {
         $Field.fAlias = $column_alias.text;
     })?
    ;

table_sources returns [HashSet<TsqlRelation> RelationList, TsqlCommand Command]
@init{
    $RelationList = new HashSet<TsqlRelation>();
    $Command = new TsqlCommand();
}
    : table_source {
        $RelationList.addAll($table_source.RelationList);
        $Command = TsqlCommand.getValueFrom($Command, $table_source.Command);
    }
    (',' table_source {
        $RelationList.addAll($table_source.RelationList);
```

```
        $Command = TsqlCommand.getValueFrom($Command, $table_source.Command);
    })*
    ;

table_source returns [HashSet<TsqlRelation> RelationList, TsqlCommand Command]
@init{
    $RelationList = new HashSet<TsqlRelation>();
    $Command = new TsqlCommand();
}
    : (table_source_item_joined {
        $RelationList = $table_source_item_joined.RelationList;
        $Command = TsqlCommand.getValueFrom($Command,
$table_source_item_joined.Command);
    })
    | '(' (table_source_item_joined {
        $RelationList = $table_source_item_joined.RelationList;
        $Command = TsqlCommand.getValueFrom($Command,
$table_source_item_joined.Command);
    }) ')'
    ;

table_source_item_joined returns [HashSet<TsqlRelation> RelationList, TsqlCommand
Command]
@init{
    $RelationList = new HashSet<TsqlRelation>();
    $Command = new TsqlCommand();
}
    : (table_source_item {
        $RelationList.addAll($table_source_item.RelationList);
        $Command = TsqlCommand.getValueFrom($Command, $table_source_item.Command);
    })
    (join_part {
        $RelationList.addAll($join_part.RelationList);
        $Command = TsqlCommand.getValueFrom($Command, $join_part.Command, true);
    })*
    ;

table_source_item returns [HashSet<TsqlRelation> RelationList, TsqlCommand Command]
@init{
    $RelationList = new HashSet<TsqlRelation>();
    $Command = new TsqlCommand();
}
    : (table_name_with_hint {
        $RelationList.add($table_name_with_hint.relation);
    })       as_table_alias?
    | rowset_function             as_table_alias?
    | derived_table {
       $RelationList.addAll($derived_table.RelationList);
       $Command = TsqlCommand.getValueFrom($Command, $derived_table.Command);
    }             (as_table_alias column_alias_list?)?
    | change_table                as_table_alias
    | function_call               as_table_alias?
    | LOCAL_ID                    as_table_alias?
    | LOCAL_ID '.' function_call (as_table_alias column_alias_list?)?
    ;

change_table
    : CHANGETABLE '(' CHANGES table_name ',' (NULL | DECIMAL | LOCAL_ID) ')'
    ;

// https://msdn.microsoft.com/en-us/library/ms191472.aspx
```

```
join_part returns [HashSet<TsqlRelation> RelationList, TsqlCommand Command]
    // https://msdn.microsoft.com/en-us/library/ms173815(v=sql.120).aspx
    @init{
        $RelationList = new HashSet<TsqlRelation>();
        $Command = new TsqlCommand();
    }
    : (INNER? |
       join_type=(LEFT | RIGHT | FULL) OUTER?) (join_hint=(LOOP | HASH | MERGE |
REMOTE))?
       JOIN (table_source {
            $RelationList.addAll($table_source.RelationList);
            $Command = TsqlCommand.getValueFrom($Command, $table_source.Command);
       }) ON search_condition
    | CROSS JOIN (table_source {
        $RelationList.addAll($table_source.RelationList);
        $Command = TsqlCommand.getValueFrom($Command, $table_source.Command);
    })
    | CROSS APPLY (table_source {
        $RelationList.addAll($table_source.RelationList);
        $Command = TsqlCommand.getValueFrom($Command, $table_source.Command);
    })
    | OUTER APPLY (table_source {
        $RelationList.addAll($table_source.RelationList);
        $Command = TsqlCommand.getValueFrom($Command, $table_source.Command);
    })
    ;

table_name_with_hint returns [TsqlRelation relation]
    : (table_name {$relation = new TsqlRelation($table_name.text);}) with_table_hints?
    ;

derived_table returns [HashSet<TsqlRelation> RelationList, TsqlCommand Command]
@init {
    $Command = new TsqlCommand();
    $RelationList = new HashSet<TsqlRelation>();
}
    : subquery {
//        $RelationList = $subquery.RelationList;
        $Command = TsqlCommand.getValueFrom($Command, $subquery.Command);
    }
    | '(' subquery {
//        $RelationList = $subquery.RelationList;
        $Command = TsqlCommand.getValueFrom($Command, $subquery.Command);
    } ')'
    ;

function_call
    : ranking_windowed_function
    | aggregate_windowed_function
    | scalar_function_name '(' expression_list? ')'
    | BINARY_CHECKSUM '(' '*' ')'
    | CAST '(' expression AS data_type ')'
    | CONVERT '(' data_type ',' expression (',' style=expression)? ')'
    | CHECKSUM '(' '*' ')'
    | COALESCE '(' expression_list ')'
    | CURRENT_TIMESTAMP
    | CURRENT_USER
    | DATEADD '(' ID ',' expression ',' expression ')'
    | DATEDIFF '(' ID ',' expression ',' expression ')'
    | DATENAME '(' ID ',' expression ')'
    | DATEPART '(' ID ',' expression ')'
```

```
    | IDENTITY '(' data_type (',' seed=DECIMAL)? (',' increment=DECIMAL)? ')'
    | MIN_ACTIVE_ROWVERSION
    | NULLIF '(' expression ',' expression ')'
    | SESSION_USER
    | SYSTEM_USER
    ;

switch_section
    : WHEN expression THEN expression
    ;

switch_search_condition_section
    : WHEN search_condition THEN expression
    ;

as_table_alias
    : AS? table_alias
    ;

table_alias
    : id with_table_hints?
    ;

// https://msdn.microsoft.com/en-us/library/ms187373.aspx
with_table_hints
    : WITH? '(' table_hint (',' table_hint)* ')'
    ;

// https://msdn.microsoft.com/en-us/library/ms187373.aspx
insert_with_table_hints
    : WITH '(' table_hint (',' table_hint)* ')'
    ;

table_hint
    : NOEXPAND? ( INDEX '(' index_value (',' index_value)* ')'
                | INDEX '=' index_value
                | FORCESEEK ('(' index_value '(' ID  (',' ID)* ')' ')')?
                | SERIALIZABLE
                | SNAPSHOT
                | SPATIAL_WINDOW_MAX_CELLS '=' DECIMAL
                | ID)?
    ;

index_value
    : id | DECIMAL
    ;

column_alias_list
    : '(' column_alias (',' column_alias)* ')'
    ;

column_alias
    : id
    | STRING
    ;

table_value_constructor
    : VALUES '(' expression_list ')' (',' '(' expression_list ')')*
    ;

expression_list
```

```
    : expression (',' expression)*
    ;

// https://msdn.microsoft.com/en-us/library/ms189798.aspx
ranking_windowed_function
    : (RANK | DENSE_RANK | ROW_NUMBER) '(' ')' over_clause
    | NTILE '(' expression ')' over_clause
    ;

// https://msdn.microsoft.com/en-us/library/ms173454.aspx
aggregate_windowed_function
    : (AVG | MAX | MIN | SUM | STDEV | STDEVP | VAR | VARP)
      '(' all_distinct_expression ')' over_clause?
    | (COUNT | COUNT_BIG)
      '(' ('*' | all_distinct_expression) ')' over_clause?
    | CHECKSUM_AGG '(' all_distinct_expression ')'
    | GROUPING '(' expression ')'
    | GROUPING_ID '(' expression_list ')'
    ;

all_distinct_expression
    : (ALL | DISTINCT)? expression
    ;

// https://msdn.microsoft.com/en-us/library/ms189461.aspx
over_clause
    : OVER '(' (PARTITION BY expression_list)? order_by_clause? row_or_range_clause? ')'
    ;

row_or_range_clause
    : (ROWS | RANGE) window_frame_extent
    ;

full_table_name
    : (server=id '.' database=id '.'  schema=id   '.'
      |              database=id '.' (schema=id)? '.'
      |                               schema=id   '.')? table=id
    ;

table_name
    : (database=id '.' (schema=id)? '.' | schema=id '.')? table=id
    ;

simple_name
    : (schema=id '.')? name=id
    ;

func_proc_name
    : (database=id '.' (schema=id)? '.' | (schema=id) '.')? procedure=id
    ;

full_column_name returns [TsqlField Field]
@init {
    $Field = new TsqlField();
}
    : (table_name {
        $Field.tName = $table_name.text;
    } '.')? id {
        $Field.fName = $id.text;
    }
    ;
```

```
column_name_list
    : id (',' id)*
    ;

cursor_name
    : id
    | LOCAL_ID
    ;

null_notnull
    : NOT? NULL
    ;

scalar_function_name
    : func_proc_name
    | RIGHT
    | LEFT
    | BINARY_CHECKSUM
    | CHECKSUM
    ;

data_type
    /*: BIGINT
    | BINARY '(' DECIMAL ')'
    | BIT
    | CHAR '(' DECIMAL ')'
    | DATE
    | DATETIME
    | DATETIME2
    | DATETIMEOFFSET '(' DECIMAL ')'
    | DECIMAL '(' DECIMAL ',' DECIMAL
')'
    | FLOAT
    | GEOGRAPHY
    | GEOMETRY
    | HIERARCHYID
    | IMAGE
    | INT
    | MONEY
    | NCHAR '(' DECIMAL ')'
    | NTEXT
    | NUMERIC '(' DECIMAL ',' DECIMAL
')'
    | NVARCHAR '(' DECIMAL | MAX ')'
    | REAL
    | SMALLDATETIME
    | SMALLINT
    | SMALLMONEY
    | SQL_VARIANT
    | TEXT
    | TIME '(' DECIMAL ')'
    | TIMESTAMP
    | TINYINT
    | UNIQUEIDENTIFIER
    | VARBINARY '(' DECIMAL | MAX ')'
    | VARCHAR '(' DECIMAL | MAX ')'
    | XML*/
    : id IDENTITY? ('(' (DECIMAL | MAX)
(',' DECIMAL)? ')')?
    ;

sign
    : '+'
    | '-'
    ;

id
    : simple_id
    | DOUBLE_QUOTE_ID
    | SQUARE_BRACKET_ID
    ;

simple_id
    : ID
    | ABSOLUTE
    | APPLY
    | AUTO
    | AVG
    | BASE64
    | CALLER
    | CAST
    | CATCH
    | CHECKSUM_AGG
    | COMMITTED
    | CONCAT
    | CONTROL
    | COOKIE
    | COUNT
    | COUNT_BIG
    | DELAY
    | DELETED
    | DENSE_RANK
    | DISABLE
    | DYNAMIC
    | ENCRYPTION
    | EXPAND
    | FAST
    | FAST_FORWARD
    | FIRST
    | FOLLOWING
    | FORCE
    | FORCESEEK
    | FORWARD_ONLY
    | FULLSCAN
    | GLOBAL
    | GO
    | GROUPING
    | GROUPING_ID
    | HASH
    | IMPERSONATE
    | INSENSITIVE
    | INSERTED
    | ISOLATION
    | KEEP
    | KEEPFIXED
    | FORCED
    | KEYSET
    |
```

```
IGNORE_NONCLUSTERED_COLUMNSTORE_INDEX
    | LAST
    | LEVEL
    | LOCAL
    | LOCK_ESCALATION
    | LOGIN
    | LOOP
    | MARK
    | MAX
    | MAXDOP
    | MAXRECURSION
    | MIN
    | MODIFY
    | NAME
    | NEXT
    | NOCOUNT
    | NOEXPAND
    | NORECOMPUTE
    | NTILE
    | NUMBER
    | OFFSET
    | ONLINE
    | ONLY
    | OPTIMISTIC
    | OPTIMIZE
    | OUT
    | OUTPUT
    | OWNER
    | PARAMETERIZATION
    | PARTITION
    | PATH
    | PRECEDING
    | PRIOR
    | PRIVILEGES
    | RANGE
    | RANK
    | READONLY
    | READ_ONLY
    | RECOMPILE
    | RELATIVE
    | REMOTE
    | REPEATABLE
    | ROBUST
    | ROOT
    | ROW
    | ROWGUID
    | ROWS
    | ROW_NUMBER
    | SAMPLE
    | SCHEMABINDING
    | SCROLL
    | SCROLL_LOCKS
    | SELF
    | SERIALIZABLE
    | SIMPLE
    | SNAPSHOT
    | SPATIAL_WINDOW_MAX_CELLS
    | STATIC
    | STATS_STREAM
    | STDEV
    | STDEVP
    | SUM
    | TEXTIMAGE_ON
    | THROW
    | TIES
    | TIME
    | TRY
    | TYPE
    | TYPE_WARNING
    | UNBOUNDED
    | UNCOMMITTED
    | UNKNOWN
    | USING
    | VAR
    | VARP
    | VIEW_METADATA
    | VIEWS
    | WORK
    | XML
    | XMLNAMESPACES
    ;

comparison_operator
    : '=' | '>' | '<' | '<' '=' | '>' '=' | '<' '>' | '!' '=' | '!' '>' | '!' '<'
    ;

assignment_operator
    : '+=' | '-=' | '*=' | '/=' | '%=' | '&=' | '^=' | '|='
    ;

file_size:
    DECIMAL( KB | MB | GB | TB | '%' )?
    ;
```