DOI 10.34185/1562-9945-3-164-2026-18
УДК 004.672

М.Ю. Павлюшин, І.О. Саяпіна

МЕТОД ТА ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ ОПТИМІЗАЦІЇ ПРОДУКТИВНОСТІ БАЗ ДАНИХ

Анотація. Сучасні підходи до оптимізації продуктивності баз даних переважно орієнтовані на окремі аспекти – індексацію, оптимізацію SQL-запитів або налаштування серверного середовища, що змушує адміністраторів використовувати декілька різних інструментів одночасно та підвищує ймовірність пропуску критичних недоліків. Невирішеною залишається проблема комплексного автоматизованого аналізу бази даних без необхідності прямого підключення до сервера. У роботі запропоновано комбінований алгоритмічно-програмний метод оптимізації продуктивності баз даних, що відрізняється від існуючих підходів можливістю комплексного офлайн-аналізу SQL-дампів та конфігураційних файлів без прямого підключення до СУБД. Метод ґрунтується на статичному аналізі SQL-дампів та конфігураційних файлів без розгортання бази даних та інтегрує три незалежних аналітичних модулі: аналіз індексації, структурний аналіз та аналіз параметрів серверної конфігурації, а також окремий модуль інтерактивного рефакторингу SQL-запитів у реальному часі. Для кількісної оцінки ефективності оптимізації запропоновано комплексний індекс продуктивності (КІП), що об'єднує три часткових показники – індексаційний, структурний та конфігураційний з визначеними ваговими коефіцієнтами. Метод реалізовано у вигляді клієнт-серверного вебзастосунку з рольовою моделлю розмежування доступу. За рахунок використання запропонованого методу досягається комплексна автоматизована оптимізація бази даних та формування рекомендацій для адміністратора без значних додаткових витрат на адміністрування.
Ключові слова: оптимізація баз даних, SQL-дамп, статичний аналіз, індексація, структурний аналіз, конфігурація сервера, комплексний індекс продуктивності, FastAPI, React, СУБД.

Постановка проблеми. Сьогодні інформаційні системи оперують величезними обсягами даних, і продуктивність баз даних є ключовим чинником ефективності будь-якого програмного комплексу. Навіть незначні затримки у виконанні SQL-запитів або помилки у конфігурації серверного середовища можуть призводити до суттєвого зниження швидкодії, втрати даних або перевитрати ресурсів серверів.

У працях [1–3] розглянуто підходи до оптимізації запитів, структури даних, а також використання хешування та індексації. Існуючі підходи до оптимізації переважно орієнтовані на окремі аспекти: одні інструменти спеціалізуються на аналізі індексів, інші на перевірці структури схеми, треті на налаштуванні серверних параметрів. Для

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

проведення комплексного аналізу зазвичай необхідно використовувати декілька різних інструментів, що ускладнює процес і збільшує ймовірність пропуску критично важливих недоліків у структурі чи конфігурації бази даних. Крім того, більшість існуючих рішень потребують безпосереднього підключення до живого сервера або виконання профілюючих запитів для збору статистики, що унеможливлює їх застосування там, де прямий доступ є технічно недоступним або небажаним з міркувань безпеки.

Таким чином, невирішеною залишається задача побудови універсального методу комплексного аналізу продуктивності баз даних на основі статичних артефактів без необхідності прямого доступу до СУБД, що і визначає напрям дослідження.

Аналіз останніх досліджень і публікацій. Проблема підвищення продуктивності баз даних активно досліджується як у промислових, так і в академічних колах.

Структурні методи орієнтовані на нормалізацію бази даних, усунення надлишковості та підтримку цілісності даних. Вбудовані оптимізатори PostgreSQL та MySQL пропонують аналіз ключів і статистики таблиць, однак вони не завжди враховують логіку бізнес-процесів і зв'язки між запитами. У роботі [4] проведено систематичний огляд методів оптимізації баз даних і показано, що жоден з існуючих підходів не охоплює комплексно всі аспекти продуктивності одночасно, а ефективність структурних методів у великих схемах залишається обмеженою.

Методи оптимізації SQL-запитів, представлені у працях [1, 2, 5], базуються на перетворенні запитів, використанні індексів, реорганізації об'єднань (JOIN) і заміні вкладених підзапитів. Зокрема, у [1] запропоновано підходи до побудови самоналаштовуваних систем керування базами даних, а у [2] – ефективний інструмент автоматичного вибору індексів на основі аналізу навантаження. У дослідженні [6] експериментально підтверджено, що застосування методів оптимізації до реляційних баз MySQL дозволяє суттєво скоротити час виконання запитів, зокрема за рахунок коректної індексації та реструктуризації схеми. Проте зазначені методи потребують попереднього профілювання або прямого доступу до СУБД, що унеможливлює їх застосування при офлайн-аналізі SQL-дампів.

Серверна оптимізація передбачає регулювання параметрів конфігурації. У дослідженнях [7, 8] розглянуто методи машинного навчання для автоматичного підбору параметрів, зокрема для рушія InnoDB, а у [9] досліджено вплив конфігураційних налаштувань MySQL на продуктивність у різних сценаріях навантаження. Систематичний огляд [10] показав, що системи автоматичного налаштування схеми та параметрів баз даних потребують тривалої фази навчання на реальному навантаженні і не надають рекомендацій, придатних для безпосереднього застосування адміністратором.

Таким чином, жоден із наявних методів не вирішує комплексно задачу статичного аналізу SQL-дампів із наданням інтегрованих рекомендацій одразу за кількома напрямами без необхідності підключення до сервера.

Мета дослідження. Метою даного дослідження є підвищення продуктивності обробки запитів у базах даних шляхом розроблення комбінованого алгоритмічно-програмного методу та відповідного програмного забезпечення. Наукова новизна робо-

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

ти полягає у розробленні алгоритмічно-програмного методу аналітично-рекомендаційної оптимізації баз даних, який відрізняється від відомих підходів можливістю статичного аналізу SQL-дампів та конфігураційних файлів без прямого підключення до СУБД. Метод інтегрує три аналітичні модулі – індексаційний, структурний та конфігураційний в єдину систему кількісного оцінювання, а також окремий модуль інтерактивного рефакторингу SQL-запитів у реальному часі. Для об'єктивного вимірювання ефекту оптимізації введено комплексний індекс продуктивності (КІП) як інтегральний показник, що об’єднує результати трьох аналітичних модулів і дозволяє кількісно порівнювати стан бази даних до та після застосування рекомендацій.

Викладення основного матеріалу дослідження. Одним із найпопулярніших прикладних протоколів в мережі Інтернет є протокол HTTP(S) для вебсторінок, оскільки зазвичай саме вебсайт є кінцевим інтерфейсом для отримання інформації чи надання послуг. Тому запропонований метод розроблявся для протоколу HTTP(S).

Запропонований метод оптимізації продуктивності баз даних побудовано на статичному аналізі вхідних артефактів без необхідності підключення до живого сервера СУБД. Основна ідея полягає у формуванні трьох груп показників, а саме індексних, структурних та конфігураційних, які нормалізуються в інтервалі [0; 1] і використовуються як для надання рекомендацій користувачу, так і для розрахунку підсумкового значення комплексного індексу продуктивності (КІП). Метод об'єднує три незалежні аналітичних модулі, кожен з яких орієнтований на окремий аспект функціонування бази даних. Загальну схему роботи методу наведено на рис. 1.

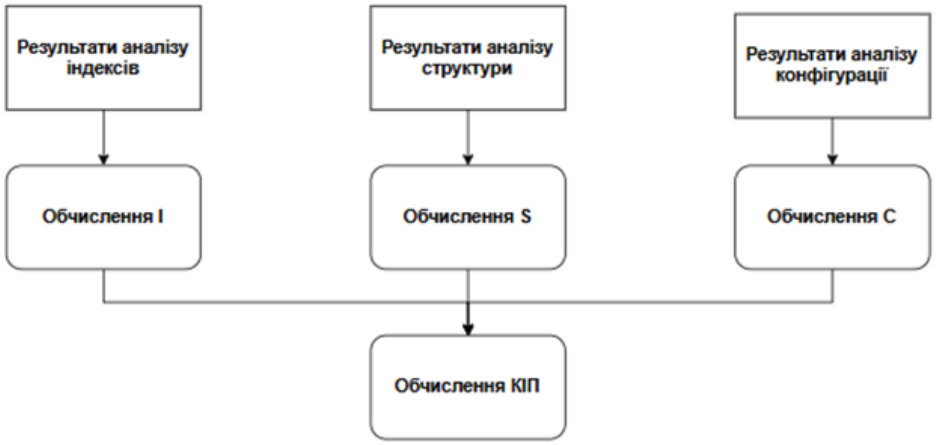

Рисунок 1 – Схема роботи запропонованого методу

Метод реалізується у вигляді послідовності етапів. На першому етапі зчитуються вхідні дані – SQL-дамп або конфігураційний файл СУБД. На другому етапі виконується статичний аналіз: синтаксичний розбір структури SQL-дампу, аналіз індексації та аналіз конфігураційних параметрів. Кожний показник масштабується до інтервалу [0; 1], де 0 відповідає негативному стану, а 1 – оптимальному. На третьому етапі обчислюються часткові індекси I, S та C для кожної групи показників. На четвертому етапі на їх основі обчислюється інтегральний показник КІП. На останньому етапі система формує звіт із рекомендаціями.

Модуль аналізу індексації (indexing). Вхідними даними є SQL-дамп бази даних. Модуль виконує синтаксичний розбір дампу та перевіряє наявність первинних і зовнішніх ключів, дублікатів індексів та таблиць без індексації. Визначається кількість таблиць без жодного індексу ($T_{noindex}$) та загальна кількість таблиць (T_{all}). Частковий індекс модуля індексації обчислюється за формулою:

$$I = 1 - \frac{T_{noindex}}{T_{all}} \quad (1)$$

Значення I = 1 свідчить про те, що всі таблиці мають щонайменше один індекс, тоді як низькі значення вказують на наявність таблиць, пошук у яких виконуватиметься повним скануванням.

Модуль структурного аналізу (structure). Вхідними даними також є SQL-дамп. Модуль перевіряє відповідність схеми вимогам нормальних форм, виявляє відсутність первинних ключів (T_{pk}), застарілі або застарілі типи даних (T_{types}) та надлишкові поля (T_{fk}). Частковий індекс структурного модуля обчислюється як:

$$S = \frac{T_{pk} + T_{fk} + T_{types}}{3} \quad (2)$$

Додатково застосовується критерій відповідності третій нормальній формі: якщо частка атрибутів із транзитивними функціональними залежностями перевищує 0,3 від загальної кількості атрибутів, система надає рекомендації щодо декомпозиції відповідних таблиць.

Модуль аналізу конфігурації (tuning). Вхідними даними є текстовий конфігураційний файл СУБД (наприклад, my.cnf для MySQL). Модуль порівнює значення ключових параметрів – innodb_buffer_pool_size, max_connections, query_cache_size, tmp_table_size із рекомендованими діапазонами($C_{optimal}$). Частковий індекс конфігураційного модуля визначається як частка параметрів, що відповідають рекомендованим значенням:

$$C = \frac{C_{optimal}}{C_{total}} \quad (3)$$

Модуль рефакторингу запитів (refactoring). На відміну від попередніх модулів, цей модуль працює в інтерактивному режимі реального часу: користувач вводить SQL-вираз, а система миттєво надає рекомендації щодо його покращення. Аналізуються типові антипатерни: використання SELECT *, відсутність обмежень WHERE у запитах на оновлення та видалення, надлишково складні вкладені підзапити, що можуть бути замінені конструкцією JOIN. Оскільки модуль працює з окремими виразами, а не з цілісним дампом, його результати не входять до розрахунку КІП.

Комплексний індекс продуктивності (КІП). Результати трьох аналітичних модулів – індексації, структури та конфігурації інтегруються в єдиний показник за формулою:

$$CPI = w_1 \cdot I + w_2 \cdot S + w_3 \cdot C \quad (4)$$

w_1, w_2, w_3 – вагові коефіцієнти значущості, вагові коефіцієнти, що задовольняють умові $w_1 + w_2 + w_3 = 1$. На основі аналізу джерел [1–3, 5, 7, 8] та проведених експериментів визначено такі значення коефіцієнтів: $w_1 = 0{,}40$, $w_2 = 0{,}35$, $w_3 = 0{,}25$. Індексація отримала найбільшу вагу, оскільки безпосередньо впливає на швидкість вибірок і JOIN-

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

операцій. Структура визначає узгодженість і ефективність зберігання даних. Конфігурація забезпечує стабільність системи та ефективне використання ресурсів.

КІП нормується в діапазоні від 0 до 1, де значення до 0,3 свідчить про суттєві структурні недоліки, 0,3-0,6 – середній рівень оптимізації, 0,6-0,8 – високий рівень, а 0,8-1,0 – про добре оптимізовану систему. Отримане значення КІП дозволяє перевіряти стан бази даних та порівнювати його до та після оптимізації без фактичного виконання запитів, що особливо важливо при статичному аналізі SQL-дампів.

Архітектура програмного забезпечення. Програмну реалізацію методу виконано у вигляді багаторівневої клієнт-серверної системи. Серверна частина реалізує основну логіку обробки вхідних даних: синтаксичний розбір SQL-дампів, виконання аналітичних модулів та обчислення показників продуктивності. Клієнтська частина забезпечує інтерактивну взаємодію з користувачем – завантаження даних, перегляд сформованих рекомендацій, роботу з модулем рефакторингу в реальному часі та перегляд історії оптимізацій. Обмін даними між рівнями здійснюється через REST API у форматі JSON. Система підтримує рольову модель розмежування доступу (RBAC) з двома ролями – користувач та адміністратор.

Для перевірки ефективності запропонованого методу було проведено тестування на двох базах даних різного масштабу.

База DB1 містила 3 таблиці. Система виявила наступні проблеми: 1 таблицю без індексів, 1 таблицю без первинного ключа та занижене значення tmp_table_size. Після застосування рекомендацій було створено відсутні індекси, додано первинний ключ та скориговано конфігураційні параметри.

База DB2 (DemoCRM) містила 10 таблиць загальним обсягом близько 180 тис. записів. Система виявила наступні проблеми: 3 таблиці без індексів, 2 таблиці без первинних ключів, 5 вкладених підзапитів у SQL-запитах, а також низьке значення innodb_buffer_pool_size (512 МБ при 8 ГБ RAM). Після застосування рекомендацій було створено індекси на ключових полях, додано первинні ключі, замінено вкладені підзапити конструкцією JOIN та скориговано конфігураційні параметри.

Результати обчислення КІП до та після застосування рекомендацій наведено у таблиці 1.

Таблиця 1

Результати експериментальної перевірки методу

База даних	Кількість таблиць	КІП до	КІП після	Покращення, %
DB1 (тестова)	3	0,52	0,7	+35
DB2 (DemoCRM)	10	0,65	0,85	+31

Експериментальні результати демонструють покращення значення КІП у середньому на 30–35%, що підтверджує ефективність запропонованого методу.

Висновки. У представленій роботі запропоновано метод оцінювання продуктивності баз даних на основі статичного аналізу SQL-дампів. Запроваджено новий показ-

ник комплексний індекс продуктивності, який дозволяє кількісно оцінити ступінь ефективності структури та конфігурації бази даних без її виконання. Розроблений програмний комплекс формує рекомендації для покращення індексації, структури таблиць і параметрів конфігурації. Експериментальна перевірка підтвердила середнє зростання КІП на 30–35% після їх застосування. Практична значущість роботи полягає у можливості застосування запропонованого методу для попереднього аналізу баз даних без доступу до продукційного середовища.

Подальші дослідження передбачають розширення системи для інтеграції з реальними СУБД і включення динамічного моніторингу виконання запитів.

ЛІТЕРАТУРА

1. Chaudhuri S., Weikum G. Rethinking Database System Architecture: Towards a Self-Tuning RISC-Style Database System. *Proceedings of the VLDB Endowment*. 2000.

2. Chaudhuri S., Narasayya V. An Efficient Cost-Driven Index Selection Tool for Microsoft SQL Server. *VLDB Journal*. 1998.

3. Gupta H. Selection of Views to Materialize in a Data Warehouse. *Proceedings of the International Conference on Database Theory (ICDT)*. 1997.

4. Ashari R. et al. A Systematic Literature Review: Database Optimization Techniques. *IEEE*. 2021. DOI: 10.1109/iccsai53272.2021.9609766

5. Chaudhuri S. An Overview of Query Optimization in Relational Systems. *ACM PODS*. 1998.

6. Győrödi C. A. et al. Performance Impact of Optimization Methods on MySQL Document-Based and Relational Databases. *Applied Sciences*. 2021. DOI: 10.3390/app11156794

7. Kunjir M. et al. Learning-Based Query Optimization. *Proceedings of the ACM SIGMOD International Conference on Management of Data*. 2020. DOI: 10.1145/3318464

8. Leis V. et al. How Good Are Query Optimizers, Really? *Proceedings of the VLDB Endowment*. 2015. Vol. 8, No. 1. P. 121–132. DOI: 10.14778/2850583.2850594

9. Gao P. et al. Research on Performance Optimization of MySQL Database. *IEEE*. 2023. DOI: 10.1109/ICIBA56860.2023.10165291

10. Mozaffari M. et al. Self-tuning Database Systems: A Systematic Literature Review of Automatic Database Schema Design and Tuning. *ACM*. 2024. DOI: 10.1145/3665323

REFERENCES

1. Chaudhuri, S., & Weikum, G. (2000). Rethinking database system architecture: Towards a self-tuning RISC-style database system. *Proceedings of the VLDB Endowment*.

2. Chaudhuri, S., & Narasayya, V. (1998). An efficient cost-driven index selection tool for Microsoft SQL Server. *VLDB Journal*.

3. Gupta, H. (1997). Selection of views to materialize in a data warehouse. *Proceedings of the International Conference on Database Theory (ICDT)*.

4. Ashari, R., et al. (2021). A systematic literature review: Database optimization techniques. *IEEE*. https://doi.org/10.1109/iccsai53272.2021.9609766

5. Chaudhuri, S. (1998). An overview of query optimization in relational systems. *ACM PODS*.

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

6. Győrödi, C. A., et al. (2021). Performance impact of optimization methods on MySQL document-based and relational databases. *Applied Sciences*.
https://doi.org/10.3390/app11156794
7. Kunjir, M., et al. (2020). Learning-based query optimization. *Proceedings of the ACM SIGMOD International Conference on Management of Data*. https://doi.org/10.1145/3318464
8. Leis, V., et al. (2015). How good are query optimizers, really? *Proceedings of the VLDB Endowment, 8*(1), 121–132. https://doi.org/10.14778/2850583.2850594
9. Gao, P., et al. (2023). Research on performance optimization of MySQL database. *IEEE*. https://doi.org/10.1109/ICIBA56860.2023.10165291
10. Mozaffari, M., et al. (2024). Self-tuning database systems: A systematic literature review of automatic database schema design and tuning. *ACM*. https://doi.org/10.1145/3665323

Received 13.04.2026.
Accepted 15.04.2026.
Published 30.04.2026

Method and software for database performance optimization

Modern approaches to database performance optimization are mostly focused on individual aspects – indexing, SQL query optimization, or server environment configuration which forces administrators to use multiple specialized tools simultaneously and significantly increases the likelihood of missing critical deficiencies in the database structure or configuration. The problem of comprehensive automated database analysis without direct server access remains unsolved, as most existing solutions require either a live connection to the database server or execution of profiling queries to collect runtime statistics, which makes them inapplicable in scenarios where direct access to the production system is restricted for security or technical reasons.

This paper proposes a combined algorithmic-programmatic method that integrates three analytical modules for static analysis – indexing, structural, and configuration as well as an additional interactive module for real-time SQL query refactoring. The three core modules contribute to the composite performance index (CPI), while the refactoring module operates independently as a real-time advisory tool. The method is based on static analysis of SQL dumps and server configuration files without deploying the database, which allows it to be applied in offline scenarios. Each module operates independently and targets a specific aspect of database performance: the indexing module detects missing or redundant indexes, the structural module verifies compliance with normal forms and identifies inefficient data types, the configuration module compares server parameters against recommended values, and the refactoring module provides real-time recommendations as the user writes SQL expressions.

To quantitatively evaluate optimization effectiveness, a composite performance index (CPI) is introduced, which aggregates three partial scores – indexing efficiency, structural efficiency, and configuration efficiency – using empirically determined weight coefficients of 0.40, 0.35, and 0.25 respectively. The CPI is normalized in the range from 0 to 1 and is calculated both before and after applying the recommendations, enabling objective comparison of the database state without executing any queries against the live system.

The method is implemented as a client-server web application with a React-based frontend and a FastAPI backend, supporting role-based access control with user and admin-

istrator roles. The proposed approach achieves comprehensive automated database optimization and generates actionable recommendations for the administrator without significant additional administration overhead.

Keywords: database optimization, SQL dump, static analysis, indexing, structural analysis, server configuration, comprehensive performance index, FastAPI, React, DBMS.

Павлюшин Максим Юрійович – магістрант кафедри програмного забезпечення комп'ютерних систем, Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського».
ORCID: https://orcid.org/0009-0004-3814-1615
Саяпіна Інна Олександрівна – к.т.н, доцент, доцент кафедри програмного забезпечення комп'ютерних систем, Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського».
ORCID: https://orcid.org/0000-0003-1541-1681

Pavliushyn Maksym Yuriyovych – Master's student of the Computer Systems Software Department, National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute".
ORCID: https://orcid.org/0009-0004-3814-1615
Saiapina Inna Oleksandrivna – Ph.D., Associate Professor, Associate Professor of the Computer Systems Software Department, National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute".
ORCID: https://orcid.org/0000-0003-1541-1681

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)