УДК 004.652.4:004.421.2

ОПТИМІЗАЦІЯ SQL-ЗАПИТІВ У РЕЛЯЦІЙНИХ БАЗАХ ДАНИХ ЯК СКЛАДОВА ПІДВИЩЕННЯ ЕФЕКТИВНОСТІ ІНФОРМАЦІЙНИХ СИСТЕМ

Нестеров Ю. М.
Національний університет «Одеська політехніка»
Україна, Одеса
nesterov@op.edu.ua

Анотація. *У роботі досліджено методи підвищення продуктивності SQL-запитів у реляційних базах даних. Проаналізовано ключові фактори, що впливають на швидкодію запитів, та проведено експериментальне порівняння до і після оптимізації. Результати показали суттєве скорочення часу обробки даних і можуть бути використані для підвищення ефективності інформаційних систем.*
Ключові слова: *реляційні бази даних; оптимізація SQL-запитів; індекси; продуктивність; інформаційні системи.*

Вступна частина. У сучасному інформаційному суспільстві, де щодня генеруються величезні обсяги структурованих даних, питання ефективності їх зберігання та обробки набуває особливого значення. Реляційні бази даних залишаються основним інструментом для організації, управління та доступу до інформації в різних галузях – від бізнесу до наукових досліджень [1, 2]. Однак зі збільшенням обсягів даних і складності аналітичних запитів зростають вимоги до швидкодії систем, що призводить до появи затримок у виконанні SQL-запитів і зниження продуктивності інформаційних систем [3].

Актуальність теми зумовлена необхідністю розроблення ефективних підходів до оптимізації запитів, які забезпечують швидке отримання даних при мінімальних витратах обчислювальних ресурсів. Сучасні дослідження доводять, що правильне проєктування індексів, використання кешування та аналіз планів виконання здатні суттєво зменшити час обробки запитів [4, 5]. Проте відомі методи часто не враховують специфіку конкретних СУБД і характер даних, що створює потребу у практичному порівнянні підходів і розробленні рекомендацій для реальних систем.

Метою дослідження є дослідження методів підвищення продуктивності SQL-запитів у реляційних базах даних шляхом аналізу факторів, що впливають на швидкодію, та експериментальної перевірки ефективності оптимізаційних прийомів.

Основна частина. У ході дослідження було детально проаналізовано архітектуру систем керування базами даних, у якій особливе місце посідає оптимізатор запитів – один із ключових компонентів, що визначає ефективність роботи всієї СУБД. Його основне завдання полягає у формуванні плану виконання запиту – набору інструкцій, які визначають, у якому порядку система здійснюватиме зчитування таблиць, застосовуватиме індекси, виконуватиме операції фільтрації, сортування та об’єднання даних. Іншими словами, оптимізатор виконує функцію «мозку» бази даних, перетворюючи декларативний SQL-код на оптимізовану послідовність конкретних дій, спрямованих на мінімізацію витрат часу й ресурсів.

Процес формування плану виконання складається з кількох етапів: аналізу запиту, оцінки вартості можливих стратегій його виконання та вибору найоптимальнішої. На першому етапі оптимізатор перевіряє синтаксичну й семантичну коректність запиту. Далі система генерує кілька можливих способів його виконання, наприклад, різні варіанти

порядку з’єднання таблиць, методів сканування (повне або індексне), типів об’єднань (Nested Loop, Merge Join, Hash Join). Кожен із варіантів оцінюється за вартісною моделлю, яка враховує обсяг даних, кількість доступів до диску, використання пам’яті, завантаження процесора тощо. Найменш затратна стратегія обирається як остаточний план виконання, який і буде реалізовано виконавчим модулем СУБД.

Аналіз показав, що ефективність виконання запиту безпосередньо залежить від якості сформованого оптимізатором плану. Вона визначається низкою факторів, серед яких — структура таблиць, наявність первинних і зовнішніх ключів, статистика даних (розподіл значень у стовпцях, кількість унікальних записів), а також синтаксис запиту. Наприклад, некоректне використання підзапитів або надмірна вкладеність умов у секції WHERE може призвести до генерації неефективного плану, який викликатиме повне сканування таблиць (Full Table Scan) замість індексного пошуку.

Додатковим чинником є актуальність статистики таблиць, яку використовує оптимізатор під час оцінки вартості операцій. Якщо статистика застаріла, система може неправильно оцінити обсяг даних або частоту появи певних значень, що призведе до вибору неоптимальної стратегії виконання. Тому регулярне оновлення статистичних метаданих є необхідною умовою для стабільної роботи оптимізатора.

Одним із найрезультативніших і водночас найважливіших способів оптимізації SQL-запитів є використання індексів. Індекс у базі даних – це спеціальна структура, призначена для прискорення пошуку записів у таблиці без необхідності повного перегляду її вмісту. За своєю суттю індекс виконує роль покажчика, який дає змогу системі швидко знайти потрібні рядки за значеннями ключових полів, мінімізуючи кількість звернень до диску та обсяг оброблюваних даних.

Використання індексів є особливо ефективним для запитів, що містять операції фільтрації, сортування або об’єднання таблиць. Без індексації система змушена виконувати повне сканування таблиці, перевіряючи кожен запис, що значно збільшує навантаження на ресурси при великих обсягах даних. Індексація дозволяє виконувати вибіркове сканування, завдяки чому система звертається лише до тих рядків, які відповідають заданим умовам.

Серед основних типів індексів розрізняють кластеризовані, некластеризовані та комбіновані (складені). Кластеризований індекс визначає фізичний порядок зберігання рядків у таблиці, забезпечуючи впорядкування даних на рівні сховища. Це значно пришвидшує виконання запитів вибірки, особливо тих, що використовують сортування або фільтрацію за первинними ключами. Однак у межах однієї таблиці може існувати лише один кластеризований індекс, оскільки він визначає фізичну структуру даних.

Некластеризовані індекси не змінюють фізичного розташування рядків, а створюють окрему логічну структуру, яка містить посилання на відповідні записи таблиці. Такий підхід дозволяє створювати кілька незалежних індексів для різних полів, що забезпечує гнучкість і високу ефективність під час виконання запитів із різними умовами відбору. Водночас велика кількість індексів може негативно впливати на швидкість оновлення даних, оскільки кожна зміна в таблиці потребує оновлення відповідних індексних структур.

Комбіновані або складені індекси охоплюють кілька полів одночасно. Вони особливо ефективні для запитів, які використовують кілька умов фільтрації або сортування за кількома ознаками. Завдяки цьому система може одночасно враховувати кілька критеріїв пошуку, зменшуючи кількість оброблюваних рядків та обсяг пам’яті, необхідної для виконання запиту.

Використання індексів також позитивно впливає на ефективність операцій об’єднання таблиць, сортування та групування, оскільки дозволяє системі швидше зіставляти значення між таблицями та формувати результати запитів. У результаті підвищується загальна продуктивність реляційної бази даних, особливо у системах, де обробляються великі обсяги інформації або виконується значна кількість однотипних запитів.

Однак важливо враховувати, що створення надмірної кількості індексів може призвести до протилежного ефекту – збільшення часу оновлення, вставлення чи видалення даних. Тому проєктування індексів потребує комплексного підходу, який передбачає баланс між швидкістю читання й ефективністю операцій модифікації. Оптимальна стратегія індексації визначається з урахуванням структури таблиць, частоти виконання запитів і характеру їхніх умов.

Експериментальні результати показали, що використання правильно спроєктованої системи індексів дає змогу скоротити час виконання складних запитів у кілька разів, не вимагаючи додаткових апаратних ресурсів. У різних системах керування базами даних — таких як PostgreSQL, MySQL, Oracle чи Microsoft SQL Server – індекси реалізовані з урахуванням особливостей архітектури, проте їхня загальна мета єдина – забезпечити ефективний доступ до даних і стабільну роботу інформаційної системи.

Додатково було досліджено вплив кешування результатів запитів та використання матеріалізованих представлень. Кешування знижує кількість повторних звернень до диску, тоді як матеріалізовані представлення зберігають результати обчислень складних аналітичних запитів, що дозволяє скоротити час обробки в кілька разів.

Практичну частину проведено у середовищах PostgreSQL, MySQL та Microsoft SQL Server. Для типових запитів, що містили операції JOIN, GROUP BY і фільтрацію за декількома умовами, було створено експериментальні набори даних. Результати показали, що використання індексів і оптимізація умов WHERE скорочують час виконання запиту у 3–5 разів, а впровадження матеріалізованих представлень – до 60 %. Аналіз планів виконання за допомогою команд EXPLAIN та EXPLAIN ANALYZE дозволив виявити «вузькі місця», зокрема повні сканування таблиць та неефективні з’єднання, і усунути їх за рахунок зміни логіки запиту.

Отримані результати демонструють доцільність застосування комбінованого підходу до оптимізації, який поєднує автоматичні засоби оптимізатора СУБД з ручним аналізом і корекцією запитів. Такий підхід дозволяє моделювати процеси обробки даних з урахуванням реальних умов використання системи, забезпечуючи більш збалансовану продуктивність і стабільність роботи.

Висновки. У результаті дослідження проаналізовано основні фактори, що впливають на ефективність виконання SQL-запитів, та визначено найбільш результативні методи оптимізації. Встановлено, що використання індексів, кешування результатів, матеріалізованих представлень і вдосконалення структури запитів дозволяють суттєво зменшити час обробки даних без необхідності оновлення апаратних ресурсів.

Практичні експерименти підтвердили, що системний підхід до оптимізації – із залученням аналізу планів виконання та корекції структури запитів – є найефективнішим. Отримані результати можуть бути використані при проєктуванні корпоративних інформаційних систем, розробці аналітичних платформ та побудові моделей продуктивності баз даних у межах системного аналізу. Подальші дослідження доцільно спрямувати на автоматизацію процесу оптимізації із використанням алгоритмів машинного навчання для прогнозування ефективності SQL-запитів у реальному часі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

[1] Date, C.J. (2019). *An Introduction to Database Systems*, 8th ed. Pearson Addison Wesley, 1024 p.
[2] Elmasri, R., & Navathe, S.B. (2020). *Fundamentals of Database Systems*, 8th ed. Pearson, 1272 p.
[3] Garcia-Molina, H., Ullman, J.D., & Widom, J. (2009). *Database Systems: The Complete Book*, 2nd ed. Pearson Prentice Hall, 1080 p.
[4] Іващук, О.В. (2021). *Бази даних та інформаційні системи: навчальний посібник.* Видавництво Львівської політехніки, 276 с.
[5] Ramakrishnan, R., & Gehrke, J. (2003). *Database Management Systems*, 3rd ed. McGraw-Hill, 1065 p.

Nesterov Yu. M.

Optimization of SQL queries in relational databases as a component of enhancing information systems efficiency

Abstract. *The paper investigates methods for improving the performance of SQL queries in relational databases. Key factors affecting query execution speed are analyzed, and an experimental comparison before and after optimization is conducted. The results show a significant reduction in data processing time and can be applied to enhance the efficiency of information systems.*

Keywords: *relational databases; SQL query optimization; indexes; performance; information systems.*