Issue 15, 2024

https://doi.org/10.23939/sisn2024.15.065

УДК 004.9

ПІДВИЩЕННЯ ПРОДУКТИВНОСТІ ЗАПИТІВ ДО БАЗ ДАНИХ: АНАЛІЗ ТЕХНІК ІНДЕКСАЦІЇ

Віталій Голубінка[1], Андрій Худий[2]

Національний університет "Львівська політехніка",
кафедра інформаційних систем та мереж, Львів, Україна
[1] E-mail: vitalii.p.holubinka@lpnu.ua, ORCID: 0009-0001-3676-3566
[2] E-mail: andrii.m.khudyi@lpnu.ua, ORCID: 0000-0003-2029-7270

Потреба у підвищенні продуктивності запитів до баз даних у сучасному інформаційному середовищі є ключовою для забезпечення ефективного функціонування різних галузей. У статті проаналізовано техніки індексації баз даних, щоб з'ясувати їх вплив на продуктивність та ефективність виконання запитів. Детально розглянуто різні типи індексів, зокрема B-дерева, хеш-таблиці та текстові індекси, проаналізовано їхні переваги та обмеження. Дослідження впливу цих технік на час виконання запитів показало, що продуктивність залежить від різних факторів, таких як складність запитів та обсяг даних.

Особливу увагу приділено вибору оптимального типу індексу залежно від конкретних потреб та характеристик бази даних. Дослідження показує, що врахування обсягу даних та структури бази даних є ключовим для вибору ефективних методів індексації. Додатково проаналізовано переваги та недоліки різних технік індексації, враховано їх вплив на швидкодію та ефективність виконання запитів.

Результати дослідження вказують на те, що правильний вибір та використання відповідних стратегій індексації може істотно підвищити продуктивність систем управління базами даних, забезпечуючи швидкий та ефективний доступ до інформації для користувачів. Зроблено важливий внесок у розуміння та практичне застосування технік індексації для підвищення продуктивності запитів до баз даних.

Ключові слова: індексація баз даних; продуктивність запитів; доступ до даних; оптимізаційні техніки; пошук інформації; управління базами даних; підвищення ефективності; структура даних.

Вступ

У сучасному інформаційному суспільстві доступ до швидкого та ефективного опрацювання даних надзвичайно важливий для успішного функціонування різних галузей. Завдяки постійному зростанню обсягів інформації та потреби в миттєвому доступі до неї бази даних відіграють ключову роль у забезпеченні цього доступу. Вони дають змогу зберігати величезні обсяги даних і забезпечують можливість виконання різноманітних запитів для отримання необхідної інформації.

Проте зі зростанням обсягів даних та складності запитів виникає проблема продуктивності виконання запитів до баз даних. Швидкість та ефективність виконання запитів стають критично важливими, оскільки впливають на роботу різних систем та сервісів. Навіть невеликі затримки у виконанні запитів можуть призвести до значних проблем, особливо в галузях, де

потрібне оперативне опрацювання великих обсягів даних, таких як фінансові послуги, медична діагностика, телекомунікації тощо.

Водночас, із розвитком сучасного інформаційного суспільства, обсяги даних, які зберігаються в базах даних, стають все більшими та різноманітнішими. Це зумовлює необхідність ефективного та швидкого доступу до цих даних для різних потреб, зокрема бізнесу, науки, медицини та інших сфер. Проте зі зростанням обсягів даних та складності запитів виникає серйозна проблема продуктивності у контексті баз даних.

Продуктивність запитів – це важливий аспект, який визначає швидкість та ефективність виконання запитів до бази даних. Зі зростанням обсягів даних та різноманіття запитів ця проблема стає ще актуальнішою. У сучасних системах управління базами даних, які опрацьовують величезні обсяги інформації в реальному часі, навіть незначні затримки у виконанні запитів можуть призвести до істотних проблем з продуктивністю та спричинити недосяжність сервісу для користувачів.

Отже, важливість оптимізації продуктивності запитів до баз даних очевидна. Це потребує розроблення та впровадження нових технік індексації, які забезпечують ефективне управління даними та мінімізують час виконання запитів. Підвищення продуктивності запитів є важливим завданням для забезпечення стабільної та ефективної роботи баз даних у сучасному інформаційному середовищі.

Аналіз останніх досліджень та публікацій

Останні дослідження у сфері оптимізації запитів до баз даних свідчать про значущість покращення методів індексації та оптимізації SQL-запитів з метою підвищення продуктивності баз даних. В контексті оцінювання витрат оптимізатора на основі згаданих умов і вказаних даних дослідження А. Бари, І. Лунгу та інших [2] підкреслюють важливість аналізу вартості виконання SQL-запитів у базах даних Oracle, що є критичним для ефективної вибірки даних.

К. Ахмад [1] розглядає удосконалення методів індексації як ключ до зменшення часу опрацювання запитів, вказуючи на потенційні недоліки наявних технік налаштування, які можуть виявитися неадекватними для сучасних застосувань, що інтенсивно використовують дані.

Дослідження К. Бема та інших [3] пропонує погляд на структури індексів та алгоритми для мультимедійних баз даних, що обіцяє підвищення продуктивності запитів у базах даних з великими обсягами даних і високими вимогами до розмірності даних.

Науковці А. Бойцеа та Ф. Радулеску [4] вивчали підходи до оптимізації продуктивності запитів у розподілених базах даних, демонструючи, як новітні техніки можуть знизити часові витрати на опрацювання множинних запитів, що відіграє ключову роль у підвищенні ефективності великих розподілених систем.

Ці дослідження колективно ілюструють значення продовження розвитку в галузі методів оптимізації та індексації баз даних, з особливим акцентом на адаптацію до дедалі більших обсягів даних та складності запитів у різноманітних сферах застосування.

Формулювання цілі статті

Мета цієї статті полягає в аналізуванні технік індексації з метою підвищення продуктивності запитів до баз даних. Вона спрямована на систематичний розгляд та оцінювання різних методів індексації, вивчення їх переваг та недоліків у контексті ефективності виконання запитів. Автори поставили мету здійснити огляд сучасних підходів до індексації баз даних і визначити оптимальні стратегії для підвищення швидкості та продуктивності опрацювання даних у системах управління базами даних.

Виклад основного матеріалу

Індексація у контексті баз даних є процесом створення структури даних, яка дає змогу швидко здійснювати пошук та доступ до інформації. Основна ідея полягає в тому, щоб створити певний тип внутрішньої організації даних, що дає змогу знаходити необхідну інформацію за допомогою

швидкого й ефективного пошуку. Індекс може бути створений для будь-якого поля чи комбінації полів у базі даних і зазвичай використовується для підвищення швидкодії пошуку, сортування та фільтрації даних. Він істотно зменшує час пошуку необхідної інформації, оскільки замість сканування усіх записів бази даних система може швидко локалізувати необхідні дані за допомогою індексу. Отже, індексація в базах даних відіграє критичну роль у забезпеченні швидкого доступу до даних і вдосконаленні продуктивності операцій з ними. Цей процес є особливо важливим у великих базах даних, де ефективний пошук і фільтрація даних можуть стати істотними факторами для успішної роботи системи.

Завдяки індексації бази даних можуть швидко відгукуватись на запити користувачів, забезпечуючи оперативне опрацювання інформації та зменшення часу очікування результатів. Без ефективних індексів операції пошуку можуть стати дуже часоємкими, особливо за великого обсягу даних, що може призвести до зниження продуктивності та задоволення користувачів. Отже, правильне використання індексації є ключовим для забезпечення оптимальної продуктивності та ефективності управління базами даних [5].

Підвищення продуктивності запитів до баз даних широко використовується у різних сферах – бізнесі, медицині, науці, технологіях тощо. Наприклад, у банківській сфері швидкий доступ до фінансових даних може вирішити питання безпеки та шахрайства, забезпечуючи миттєвий доступ до інформації про транзакції. У медичних дослідженнях швидкий пошук медичних даних може допомогти у виявленні нових лікувальних методів або здійсненні швидкої діагностики.

Крім того, з розвитком інтернету та хмарних технологій опрацювання великих обсягів даних стає все важливішим. Індексація даних дає змогу оптимізувати швидкість роботи вебсайтів, віддалених серверів та інших онлайн-систем, що є критичним для задоволення потреб сучасних користувачів, які очікують миттєвих результатів та безперебійного доступу до інформації.

Існує кілька типів індексів, які використовують у базах даних для оптимізації пошуку та доступу до даних. Один із найпоширеніших типів – B-дерева, що є структурою даних для організації та зберігання індексованих ключів у вигляді бінарного дерева. B-дерева дають змогу швидко знаходити необхідні дані через операції пошуку в логарифмічному часі.

Інший тип індексів – хеш-таблиці, які використовують для швидкого пошуку значень за ключем за допомогою хешування. Цей метод пришвидшує доступ до даних, але може виникнути конфлікт хешів, який потребує вирішення [6].

Крім того, індекси можуть бути створені для текстових полів, що дає змогу швидко здійснювати повнотекстовий пошук у текстових даних, таких як документи або вебсторінки. Це особливо корисно для систем пошуку, в яких потрібно здійснювати пошук за ключовими словами чи фразами.

Усі ці типи індексів мають певні особливості та характеристики, які впливають на їхню ефективність і застосування у конкретних ситуаціях. Наприклад, B-дерева – деревоподібна структура даних, яка дає змогу швидко виконувати операції пошуку, вставляння та видалення у відсортованих даних. Їх часто використовують у реляційних базах даних зі збереженням впорядкованої інформації.

Хеш-таблиці, з іншого боку, основані на хеш-функціях і уможливлюють швидкий доступ до даних за допомогою хеш-значень ключів. Вони добре підходять для швидкого пошуку унікальних значень, але можуть виявитися менш ефективними у випадку колізій хешування.

Текстові індекси призначені для оптимізації пошуку текстової інформації, такої як повнотекстовий пошук у документах або статтях. Вони дають змогу виконувати складні запити, які охоплюють текстові дані, такі як пошук за ключовими словами або фразами.

Окрім зазначених типів індексів, є й інші цікаві методи оптимізації продуктивності запитів до баз даних. Наприклад, геопросторові індекси використовують для ефективного пошуку об’єктів на основі їхнього географічного розташування. Це дає змогу, наприклад, швидко знаходити найближчі до певного місця об’єкти або аналізувати просторові взаємозв’язки між даними [7].

Кластерні індекси дають змогу групувати подібні дані разом, що може підвищити швидкодію пошуку та зменшити кількість звернень до диска. Це особливо корисно для великих обсягів даних, де групування даних може спростити процеси пошуку та аналізу.

Крім того, останніми роками активно розробляють нові технології індексації, такі як індексація векторного простору для пошуку за векторними представленнями даних або індексація для опрацювання графових структур. Ці нові підходи відкривають нові можливості для аналізу та оптимізації даних у різних областях, таких як машинне навчання, аналіз соціальних мереж або обробка великих графів даних [8].

Під час вибору того чи іншого типу варто також звертати увагу як на їх переваги, так і на недоліки, для вибору найраціональнішого варіанта залежно від конкретного запиту. Переваги та недоліки різних технік індексації в базах даних можуть бути значними і залежать від конкретних умов та вимог проєкту. Наприклад, B-дерева забезпечують швидкий доступ до даних і хорошу підтримку для операцій пошуку, вставляння та видалення, але можуть виявитися менш ефективними для додавання та видалення даних у випадку частих змін обсягу даних (рис. 1).

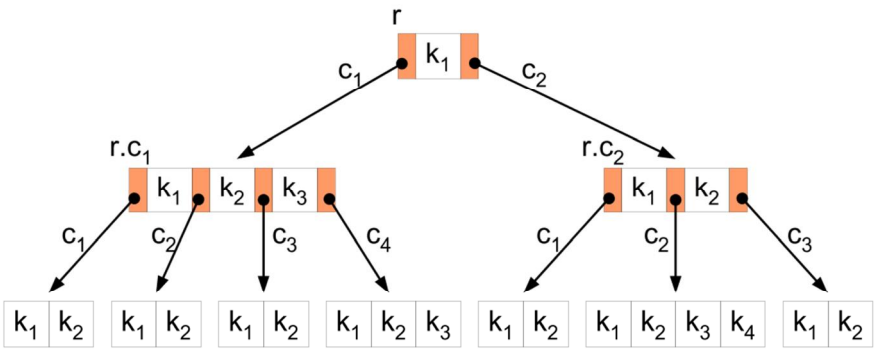

Рис. 1. Приклад графічного зображення "B-дерево"

Хеш-таблиці, з іншого боку, можуть пришвидшувати пошук, але вони стають менш ефективними за великої кількості колізій хешування (рис. 2).

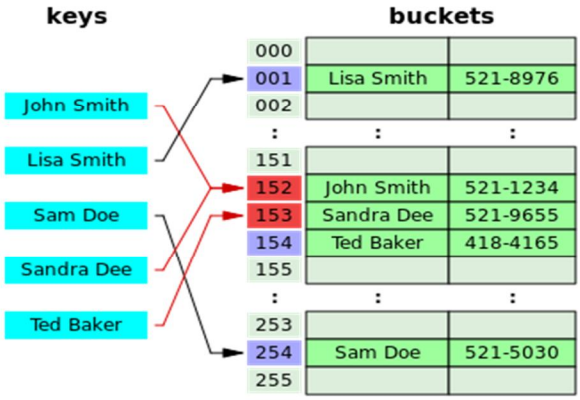

Рис. 2. Приклад графічного зображення "Хеш-таблиці"

Текстові індекси дають змогу швидко знаходити текстову інформацію, але можливі обмеження щодо розділових символів та обсягу індексованих даних.

Крім того, деякі техніки індексації можуть потребувати значних ресурсів пам'яті або обчислювальної потужності для створення та підтримки індексу. Наприклад, побудова B-дерев потребує

значних обчислювальних ресурсів та часу для великих обсягів даних. Також треба враховувати, що деякі типи індексів можуть бути менш ефективними для певних типів запитів чи структур даних.

Також слід враховувати, що деякі типи індексів можуть бути менш ефективними для певних типів запитів чи структур даних. Наприклад, B-дерева, хоча і забезпечують швидкий доступ до даних та ефективне виконання операцій вставляння та видалення, можуть стати менш ефективними за широкого діапазону значень, коли у рядку ключа велика кількість повторюваних значень. У таких випадках, швидше за все, краще використовувати хеш-таблиці, які можуть забезпечити швидкий доступ до даних за конкретним хеш-ключем. Проте хеш-таблиці також можуть бути менш ефективними, коли потрібно виконати діапазонний пошук або сортування даних за ключем [9].

Додатково текстові індекси можуть стати неефективними у випадку значного обсягу текстової інформації з великою кількістю унікальних слів або фраз. У таких ситуаціях може виникнути проблема з обсягом пам'яті, потрібного для підтримки індексу, а також зі швидкістю пошуку через велику кількість записів у індексі.

Додатково зазначимо, що ефективність різних типів індексів може змінюватися залежно від конкретної архітектури системи баз даних та характеристик апаратного забезпечення. Наприклад, наявність кеш-пам'яті або рівня розподілення даних на диску може вплинути на швидкодію деяких типів індексів.

Крім того, розробники постійно працюють над створенням нових технологій і методів індексації, спрямованих на вдосконалення продуктивності та ефективності роботи з базами даних. Наприклад, індексація векторного простору, використовувана в аналізі текстів або зображень, відкриває нові можливості для швидкого та ефективного пошуку за векторними представленнями даних.

Крім того, індексація для роботи з графовими структурами набуває все більшої популярності, особливо в контексті аналізу соціальних мереж, транспортних систем або біологічних мереж. Використання спеціалізованих графових баз даних та методів індексації дає змогу ефективно аналізувати та оптимізувати великі графи даних [10].

Аналіз впливу на час виконання запитів критично важливий у контексті оптимізації продуктивності баз даних. Час виконання запиту може бути безпосередньо пов'язаний із різними факторами, зокрема типами та кількістю індексів, обсягом даних, архітектурою системи баз даних, а також специфічними властивостями запитів.

Наявність індексів може істотно зменшити час виконання запитів, оскільки вони дозволяють базі даних ефективно шукати та фільтрувати дані. Однак велика кількість індексів може призвести до збільшення часу на вставляння, оновлення та видалення даних, тому оптимальне використання індексів потребує збалансованого підходу.

Обсяг даних також істотно впливає на час виконання запитів. Зі зростанням обсягу даних зазвичай збільшується час виконання, оскільки база даних повинна обробляти більше інформації. Великі обсяги даних також можуть призвести до необхідності оптимізації запитів та використання ефективніших методів опрацювання даних.

Архітектура системи баз даних також впливає на час виконання запитів. Наприклад, у розподілених системах баз даних іноді складніший механізм обміну даними між вузлами, що може призвести до затримок у виконанні запитів.

Нарешті, специфічні властивості запитів теж можуть істотно впливати на час їх виконання. Наприклад, складність умов, введених у запит, може суттєво вплинути на продуктивність. Запити з багатьма складними умовами або об'єднаннями іноді потребують більших обчислювальної потужності та ресурсів бази даних для їх виконання [11].

Крім того, обсяг даних, які потрібно вибирати, також впливає на час виконання запиту. Великі обсяги даних можуть збільшувати час, необхідний для їх оброблення та передавання, особливо в разі виконання запитів на великих обсягах даних або із використанням агрегатних функцій.

Тип операцій також може мати значення для часу виконання запиту. Наприклад, операції з об'єднанням, групуванням та сортуванням даних зазвичай потребують додаткових обчислювальних ресурсів та можуть збільшувати час виконання запиту.

Усі ці фактори потребують уважного аналізу та оптимізації для забезпечення максимальної продуктивності та швидкодії виконання запитів до баз даних. Це може передбачати використання індексів, оптимізацію структури бази даних, а також вдосконалення запитів для оптимального використання ресурсів.

Крім того, важливо враховувати, що ефективність виконання запитів може залежати від того, чи використовують паралельні обчислення. У паралельних системах баз даних запити можуть обробляти одночасно різні процесори або вузли, що може істотно зменшити час виконання, особливо для великих обсягів даних та складних запитів.

Додатково можливість кешування результатів запитів також може позитивно вплинути на час їх виконання. Кешування дає змогу зберігати попередні результати запитів у швидкодіючій пам'яті, що дозволяє уникнути повторних обчислень у випадку повторного запиту, зменшуючи час відповіді та виконання [11, 12].

Також варто враховувати, що продуктивність баз даних може залежати від оптимізації самого запиту. Наприклад, використання оптимальних алгоритмів пошуку та обробки даних, а також правильне використання індексів у запитах істотно підвищує їх продуктивність.

Тож для підвищення ефективності роботи з базами даних необхідно враховувати не лише самі дані та їх опрацювання, а й контекст виконання запитів, зокрема можливість паралельних обчислень, кешування результатів та оптимізацію самого запиту.

Також зазначимо, що врахування обсягу даних та структури бази даних має велике значення для аналізу часу виконання запитів. Обсяг даних може впливати на продуктивність через збільшення часу на пошук та обробку інформації. У великих базах даних зазвичай необхідно виконувати складні операції пошуку та фільтрації, що може призводити до збільшення часу виконання запитів.

Структура бази даних також відіграє ключову роль у визначенні часу виконання запитів. Наприклад, нормалізовані бази даних, які дотримуються принципів нормалізації даних, іноді потребують більшої кількості об'єднань таблиць для виконання складних запитів, що може збільшити час виконання. На відміну від цього, денормалізовані бази даних можуть забезпечувати швидший доступ до даних за рахунок зберігання пов'язаних даних в одній таблиці, що зменшує кількість об'єднань [13, 14].

Крім того, врахування обсягу даних та структури бази даних також важливе для вибору оптимальних методів індексації. Наприклад, великі таблиці можуть вимагати використання композитних індексів або індексів, які покривають більше від одного стовпчика, для ефективного пошуку та фільтрації даних.

Композитні індекси дають змогу створювати індекс для комбінації декількох стовпчиків у таблиці [15]. Це особливо корисно в ситуаціях, коли запити часто використовують кілька стовпчиків для фільтрації або умов. Завдяки композитним індексам можна значно підвищити швидкодію таких запитів, оскільки база даних швидко локалізуватиме потрібні записи за допомогою одного індексу.

Крім того, індекси, які покривають більше від одного стовпчика, також корисні для великих таблиць. Вони дають змогу оптимізувати пошук за декількома критеріями одночасно. Наприклад, якщо таблиця містить інформацію про товари та їх ціни, індекс, який охоплює обидва ці стовпчики, допоможе швидше знаходити товари в певному ціновому діапазоні [16].

Додатковим аспектом, який варто врахувати під час вибору методів індексації, є можливість використання альтернативних структур даних для оптимізації продуктивності запитів. Наприклад, для текстових даних чи даних типу "ключ-значення" можуть бути використані спеціалізовані текстові індекси або хеш-таблиці, які дають змогу ефективно шукати та виконувати запити до таких даних.

Деякі сучасні системи управління базами даних також надають можливість автоматичного створення індексів на підставі аналізу запитів та звітів про використання бази даних. Це дає змогу системі адаптуватися до змінних потреб користувачів та оптимізувати продуктивність без необхідності ручного втручання розробників.

Крім того, для деяких сценаріїв корисне використання розподілених баз даних, у яких дані розділені між кількома вузлами. У таких системах індексація може бути розподілена між різними вузлами, що допомагає розподілити навантаження та підвищити продуктивність виконання запитів.

Отже, урахування різноманітних альтернативних методів індексації та можливості автоматизованого управління індексами допомагає досягти істотного підвищення продуктивності запитів до баз даних у різних сценаріях використання.

Висновки

Виконавши детальний аналіз технік індексації баз даних, можна зробити кілька висновків.

По-перше, розглянуті різні типи індексів, такі як B-дерева, хеш-таблиці, та текстові індекси, мають певні унікальні переваги та обмеження, які потрібно враховувати під час вибору оптимального методу для конкретної бази даних.

По-друге, аналіз впливу на час виконання запитів показав, що швидкодія виконання залежить від багатьох факторів, таких як складність запитів, обсяг даних та тип операцій. Це підкреслює важливість ретельного вибору типу індексу для оптимізації продуктивності.

Крім того, врахування обсягу даних та структури бази даних визначає можливості використання композитних індексів та інших альтернативних технік індексації для забезпечення ефективного пошуку та фільтрації даних.

Нарешті, врахування усіх цих факторів та використання відповідних стратегій індексації сприятиме підвищенню продуктивності систем управління базами даних, забезпечуючи швидкий та ефективний доступ до інформації для користувачів у різних сценаріях використання.

Список літератури

1. Bâra, A., Lungu, I., Velicanu, M., & Diaconiţa, V. (2008). Improving query performance in virtual data warehouses. *WSEAS Transactions on Information Science and Applications*, 5(3), 295–302. https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=aeccdfbab928e994fe4a419130a4d24bea9571c8

2. Ahmad, K. (2020). Query Performance in Database Operation. https://www.ftsm.ukm.my/v5/file/research/technicalreport/PS-FTSM-2020-045.pdf

3. Böhm, C., Berchtold, S., & Keim, D. A. (2001). Searching in high-dimensional spaces: Index structures for improving the performance of multimedia databases. *ACM Computing Surveys (CSUR)*, 33(3), 322–373. https://kops.uni-konstanz.de/server/api/core/bitstreams/5c71837c-9d42-4eeb-b105-fc574ecb3fd1/content

4. Boicea, A., Radulescu, F., Truica, C. O., & Urse, L. (2016). Improving Query Performance in Distributed Database. *Journal of Control Engineering and Applied Informatics*, 18(2), 10–17. http://www.ceai.srait.ro/index.php?journal=ceai&page=article&op=download&path%5B%5D=3159&path%5B%5D=1393

5. Rupley, M., Jr. (2008). Introduction to query processing and optimization. *Indiana University South Bend Computer Science and Informatics.* https://clas.iusb.edu/computer-science-informatics/research/reports/TR-20080105-1.pdf

6. Bhajipale, R., Bisen, P., Meshram, A., & Thakur, S. S. (2016). SQL tuner. *International Journal of Computer Trends and Technology*, 33(1), 29–32. https://doi.org/10.14445/22312803/IJCTT-V33P106

7. Karthik, P., Reddy, G. T., & Vanan, E. K. (2012). Tuning the SQL query in order to reduce time consumption. *International Journal of Computer Science Issues*, 9(4/3), 418–423. https://www.ijcsi.org/papers/IJCSI-9-4-3-418-423.pdf

8. Habimana, J. (2015). Query optimization techniques – tips for writing efficient and faster SQL queries. *International Journal of Scientific & Technology Research*, 4(10), 22–26. https://www.ijstr.org/final-print/oct2015/Query-Optimization-Techniques-Tips-For-Writing-Efficient-And-Faster-Sql-Queries.pdf

9. Sahal, R., Nihad, M., Khafagy, M. H., & Omara, F. A. (2018). iHOME: Index based JOIN query optimization for limited big data storage. *Journal of Grid Computing*, 16(2), 345–380. https://doi.org/10.1007/s10723-018-9431-9

10. Sharma, M. (2012). Query optimization using SQL transformations. *International Journal of IT, Engineering and Applied Sciences Research*, 1(1), 100–104. http://www.irjcjournals.org/ijieasr/Oct2012/20.pdf

11. Srinivas, S. S., Naik, B. V., & Kumar, J. S. A. (2017). Query minimization methods. *International Journal of Scientific & Engineering Research*, 8(5), 30–33. https://www.ijser.org/researchpaper/Query-Minimization-Methods.pdf

12. Patel, D., & Patel, P. (2015). An approach for query optimization by using schema object base view. *International Journal of Computer Applications*, 119(16), 21–24. https://doi.org/10.5120/21152-4146

13. Patil, S., Damare, P., Sonawane, J., & Maitre, N. (2015). Study of performance tuning techniques. *Journal of Emerging Technologies and Innovative Research*, 2(3), 499–502. https://www.jetir.org/papers/JETIR1503018.pdf

14. Corlatan, C. G., Lazar, M. M., Luca, V., & Petricica, O. T. (2014). Query optimization techniques in Microsoft SQL server. *Database Systems Journal*, 5(2), 33–48. https://www.dbjournal.ro/archive/16/16_4.pdf

15. Lokhande, A. D., & Shete, R. M. (2012). The use of hints in SQL-Nested query optimization. *Journal of Data Mining and Knowledge Discovery*, 3(1), 54–57. https://bioinfopublication.org/files/articles/3_1_5_JDMKD.pdf

16. Ozar, B. (2022, July 21). How to Download the Stack Overflow Database. https://www.brentozar.com/archive/2015/10/how-to-download-the-stack-overflow-database-via-bittorrent/

References

1. Bâra, A., Lungu, I., Velicanu, M., & Diaconiţa, V. (2008). Improving query performance in virtual data warehouses. *WSEAS Transactions on Information Science and Applications*, 5(3), 295–302. https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=aeccdfbab928e994fe4a419130a4d24bea9571c8

2. Ahmad, K. (2020). Query Performance in Database Operation. https://www.ftsm.ukm.my/v5/file/research/technicalreport/PS-FTSM-2020-045.pdf

3. Böhm, C., Berchtold, S., & Keim, D. A. (2001). Searching in high-dimensional spaces: Index structures for improving the performance of multimedia databases. *ACM Computing Surveys (CSUR)*, 33(3), 322–373. https://kops.uni-konstanz.de/server/api/core/bitstreams/5c71837c-9d42-4eeb-b105-fc574ecb3fd1/content

4. Boicea, A., Radulescu, F., Truica, C. O., & Urse, L. (2016). Improving Query Performance in Distributed Database. *Journal of Control Engineering and Applied Informatics*, 18(2), 10–17. http://www.ceai.srait.ro/index.php?journal=ceai&page=article&op=download&path%5B%5D=3159&path%5B%5D=1393

5. Rupley, M., Jr. (2008). Introduction to query processing and optimization. *Indiana University South Bend Computer Science and Informatics.* https://clas.iusb.edu/computer-science-informatics/research/reports/TR-20080105-1.pdf

6. Bhajipale, R., Bisen, P., Meshram, A., & Thakur, S. S. (2016). SQL tuner. *International Journal of Computer Trends and Technology*, 33(1), 29–32. https://doi.org/10.14445/22312803/IJCTT-V33P106

7. Karthik, P., Reddy, G. T., & Vanan, E. K. (2012). Tuning the SQL query in order to reduce time consumption. *International Journal of Computer Science Issues*, 9(4/3), 418–423. https://www.ijcsi.org/papers/IJCSI-9-4-3-418-423.pdf

8. Habimana, J. (2015). Query optimization techniques – tips for writing efficient and faster SQL queries. *International Journal of Scientific & Technology Research*, 4(10), 22–26. https://www.ijstr.org/final-print/oct2015/Query-Optimization-Techniques-Tips-For-Writing-Efficient-And-Faster-Sql-Queries.pdf

9. Sahal, R., Nihad, M., Khafagy, M. H., & Omara, F. A. (2018). iHOME: Index based JOIN query optimization for limited big data storage. *Journal of Grid Computing*, 16(2), 345–380. https://doi.org/10.1007/s10723-018-9431-9

10. Sharma, M. (2012). Query optimization using SQL transformations. *International Journal of IT, Engineering and Applied Sciences Research*, 1(1), 100–104. http://www.irjcjournals.org/ijieasr/Oct2012/20.pdf

11. Srinivas, S. S., Naik, B. V., & Kumar, J. S. A. (2017). Query minimization methods. *International Journal of Scientific & Engineering Research*, 8(5), 30–33. https://www.ijser.org/researchpaper/Query-Minimization-Methods.pdf

12. Patel, D., & Patel, P. (2015). An approach for query optimization by using schema object base view. *International Journal of Computer Applications*, 119(16), 21–24. https://doi.org/10.5120/21152-4146

13. Patil, S., Damare, P., Sonawane, J., & Maitre, N. (2015). Study of performance tuning techniques. *Journal of Emerging Technologies and Innovative Research*, 2(3), 499–502. https://www.jetir.org/papers/JETIR1503018.pdf

14. Corlatan, C. G., Lazar, M. M., Luca, V., & Petricica, O. T. (2014). Query optimization techniques in Microsoft SQL server. *Database Systems Journal*, 5(2), 33–48. https://www.dbjournal.ro/archive/16/16_4.pdf

15. Lokhande, A. D., & Shete, R. M. (2012). The use of hints in SQL-Nested query optimization. *Journal of Data Mining and Knowledge Discovery*, 3(1), 54–57. https://bioinfopublication.org/files/articles/3_1_5_JDMKD.pdf

16. Ozar, B. (2022, July 21). How to Download the Stack Overflow Database. https://www.brentozar.com/archive/2015/10/how-to-download-the-stack-overflow-database-via-bittorrent/

ENHANCING DATABASE QUERY PERFORMANCE: ANALYSIS OF INDEXING TECHNIQUES

Vitalii Holubinka[1], Andrii Khydyi[2]

Lviv Polytechnic National University,
Information Systems and Networks Department, Lviv, Ukraine
[1] E-mail: vitalii.p.holubinka@lpnu.ua, ORCID: 0009-0001-3676-3566
[2] E-mail: andrii.m.khudyi@lpnu.ua, ORCID: 0000-0003-2029-7270

The need to enhance query performance in databases within the contemporary information environment is crucial for ensuring efficient operations across various domains. This paper is dedicated to an analysis of database indexing techniques aimed at understanding their impact on query performance and efficiency. It meticulously examines various types of indexes, including B-trees, hash tables, and textual indexes, analyzing their advantages and limitations. The investigation into the influence of these techniques on query execution time reveals that performance depends on various factors such as query complexity and data volume.

Special attention is given to selecting the optimal index type based on specific database needs and characteristics. The research underscores the importance of considering data volume and database structure in choosing effective indexing methods. Additionally, the advantages and disadvantages of different indexing techniques are scrutinized, taking into account their impact on query execution speed and efficiency.

Of the study indicate that the correct selection and utilization of appropriate indexing strategies can significantly enhance the performance of database management systems, providing swift and efficient access to information for users. Ultimately, this work makes a significant contribution to understanding and practically applying indexing techniques to improve query performance in databases.

Key words: database indexing; query performance; data access; optimization techniques; information retrieval; database management; efficiency enhancement; data structure.