УДК 004.65:004.421

АДАПТИВНЕ ІНДЕКСУВАННЯ РЕЛЯЦІЙНИХ БАЗ ДАНИХ З УРАХУВАННЯМ ДИНАМІЧНИХ РОБОЧИХ НАВАНТАЖЕНЬ

Щербань В. В.[1]; Озарінська О. В.[2]
[1,2]Національний університет «Одеська політехніка»
[1,2]Україна, Одеса
[1]shcherban.v@gmail.com; [2]o.v.ozarinska@op.edu.ua

Анотація. *Запропоновано методику адаптивного вибору індексів за зміни частоти та структури SQL-запитів. Розглянуто компроміс між прискоренням читання, витратами на модифікацію даних і дисковим простором. Розроблено п'ятиетапний процес моніторингу, формування кандидатів, оцінювання, перевірки й стабілізації конфігурації індексів. Визначено умови практичного застосування методики.*
Ключові слова: *адаптивне індексування; реляційні бази даних; SQL-запити; робоче навантаження; продуктивність.*

Вступна частина. Продуктивність реляційних систем керування базами даних (СУБД) залежить від фізичної організації доступу до даних. Індекси прискорюють вибірку записів, проте потребують дискового простору та додаткових операцій під час вставлення, оновлення і видалення рядків. Набір індексів, ефективний для одного профілю навантаження, може стати невигідним після зміни співвідношення операцій читання й запису. Тому вибір індексів має враховувати не лише окремі запити, а й сукупні витрати системи [1].

Минулорічна робота автора була присвячена традиційним засобам оптимізації SQL-запитів, зокрема індексуванню та аналізу планів виконання. Логічним продовженням є перехід від статичної конфігурації індексів до їх періодичного перегляду за результатами моніторингу. Відомі підходи до автоматизованого фізичного проєктування й адаптивного індексування підтверджують перспективність такого напряму, але потребують врахування вартості перебудови структур і змінності навантаження [2, 3].

Мета роботи – розробити методику адаптивного вибору індексів реляційної бази даних із комплексним урахуванням часу читання, витрат на модифікацію даних і використання дискового простору. Робота має методичний характер; наведений алгоритм потребує подальшої кількісної перевірки.

Основна частина. Під динамічним робочим навантаженням розуміємо послідовність SQL-запитів, у якій змінюються частота виконання, параметри фільтрації, обсяги таблиць і співвідношення операцій SELECT та INSERT, UPDATE, DELETE. Запропоновано аналізувати навантаження в послідовних часових вікнах, фіксуючи нормалізований шаблон запиту, кількість запусків, тривалість, використані індекси та характеристики плану виконання. Такий підхід дозволяє виявляти стійкі зміни, не реагуючи на кожне випадкове коливання.

Перший етап методики – моніторинг робочого навантаження шляхом аналізу статистики виконання SQL-запитів, їхньої частоти, тривалості та планів виконання. Отримані дані використовуються для визначення найбільш ресурсомістких операцій і формування рекомендацій щодо оптимізації індексної структури. Дані необхідно агрегувати за шаблонами SQL-запитів, оскільки різні значення параметрів часто породжують однакову структуру звернення до таблиць. Під час вимірювань важливо контролювати кешування, паралельні транзакції та зміни конфігурації сервера.

Для забезпечення зіставності результатів важливо розрізняти короткочасні коливання навантаження та структурні зміни у використанні бази даних. Наприклад, сезонне збільшення кількості аналітичних запитів не завжди виправдовує постійну перебудову індексів. Тому статистику доцільно оцінювати одночасно в короткому та довшому часових інтервалах. Перший допомагає виявляти нові тенденції, другий – перевіряти їх стійкість. Якщо

навантаження змінюється лише на короткий період, створення індексу може бути дорожчим за очікувану економію часу.

Другий етап – формування кандидатних індексів за повторюваними умовами фільтрації, з'єднання, сортування та групування. Для кількох умов відбору розглядаються складені індекси з урахуванням порядку полів і можливості використання префікса. Водночас потрібно перевіряти, чи не дублює новий кандидат уже створений індекс. Низька селективність умови, невелика таблиця або переважання операцій запису можуть робити нову індексну структуру недоцільною.

Під час формування кандидатів слід ураховувати не лише окремі стовпці, а й взаємозв'язок умов відбору. Складений індекс може бути корисним для поширеної комбінації предикатів, проте його ефективність залежить від порядку атрибутів, селективності та характеру запитів. Також важливо аналізувати надлишкові індекси: декілька подібних структур іноді збільшують витрати на модифікацію даних, не забезпечуючи пропорційного прискорення читання. Отже, перелік кандидатів має містити як пропозиції створення нових індексів, так і пропозиції перегляду наявних.

Третій етап – комплексне ранжування кандидатів. Запропоновано враховувати три групи критеріїв: очікуване скорочення сумарного часу читання з поправкою на частоту запитів; додатковий час вставлення, оновлення та видалення; витрати дискового простору і вартість створення індексу. Індекс, що прискорює один рідкісний SELECT, але погіршує численні операції UPDATE, не повинен автоматично отримувати високий пріоритет.

Для попереднього вибору пропонується порівняльна матриця з трьома рівнями очікуваного ефекту та трьома рівнями ресурсних витрат. Кандидат із високою користю для частих запитів і помірними накладними витратами отримує пріоритет для перевірки. Кандидат із низькою користю та високими витратами відкладається. Це правило є процедурою відбору, а не заміною вимірювань: остаточне рішення повинне спиратися на плани виконання й фактичні часові показники.

Четвертий етап – контрольована перевірка індексної конфігурації. Репрезентативний набір SQL-запитів виконується до і після зміни індексів на однакових даних. Порівнюються медіанний час читання, тривалість операцій модифікації та обсяг індексних структур. Вимірювання необхідно повторювати, відокремлюючи ефект індексації від прогріву кешу та коливань навантаження. Для промислової системи бажане попереднє тестування на окремому стенді.

Під час перевірки необхідно оцінювати результат не лише за середнім часом запиту, оскільки поодинокі тривалі операції можуть суттєво впливати на сприйняття швидкодії системи. Доцільно порівнювати медіану та верхні процентилі часу виконання, а також пропускну здатність за фіксованої кількості одночасних користувачів. Окремо слід вимірювати затримки операцій запису. Якщо індекс прискорює більшість запитів читання, але значно збільшує час критичних транзакцій, його впровадження потребує додаткового обґрунтування.

П'ятий етап – стабілізація конфігурації. Автоматичне створення й видалення індексів після кожної короткочасної зміни навантаження може спричинити надмірні витрати. Тому пропонується мінімальний інтервал між переглядами, поріг очікуваної користі та окрема процедура вилучення індексів, які тривалий час не використовуються. Перед видаленням слід перевіряти рідкісні критичні запити й можливість відновлення попередньої конфігурації.

Для практичного використання адаптивного механізму варто передбачити журнал прийнятих рішень. У ньому фіксують дату зміни, причину створення або вилучення індексу, очікуваний ефект і фактичні результати перевірки. Такий журнал дає змогу виявляти повторні неефективні рішення та забезпечує відтворюваність налаштувань. Додатково пропонується режим рекомендацій без автоматичного застосування змін: адміністратор підтверджує дію після аналізу прогнозованого виграшу, можливих ризиків і доступного часу на перебудову структури даних.

Розглянемо два умовні сценарії. У першому переважають часті запити SELECT із фільтрацією за двома полями; складений індекс потенційно зменшує обсяг сканування. У другому частота цих запитів знижується, а кількість INSERT і UPDATE зростає. За таких умов користь від додаткового індексу може не компенсувати витрат на його підтримку. Приклад демонструє логіку адаптації, але не є експериментальним доказом прискорення.

Науково-методичний результат полягає у формалізації послідовності дій для системи рекомендацій: зібрати статистику, визначити кандидати, оцінити компроміси, перевірити ефективність і застосувати зміни з контролем стабільності. На відміну від разового індексування, рішення переглядається відповідно до фактичного профілю SQL-навантаження. Запропонований підхід можна реалізувати як допоміжний модуль адміністратора бази даних.

Критерієм успішності методики пропонується вважати зменшення сукупного часу обробки репрезентативного навантаження за умови дотримання встановлених обмежень на дисковий простір та допустиме погіршення операцій запису. Для майбутнього експерименту доцільно передбачити три режими: відсутність додаткових індексів, статично сформований набір та адаптивний набір, який періодично переглядається. Порівняння цих режимів на однакових даних дозволить відокремити ефект початкової індексації від додаткової користі саме адаптивного керування.

Висновки. Розроблено п’ятиетапну методику адаптивного індексування реляційних баз даних, що враховує зміну структури та частоти SQL-запитів. Обґрунтовано необхідність спільного оцінювання виграшу від читання, витрат на модифікацію даних і використання дискового простору. Запропоновано механізм стабілізації, який обмежує часті перебудови індексів.

Практичне значення роботи полягає у можливості застосування методики як основи автоматизованої системи рекомендацій. Подальше дослідження має включати контрольований експеримент у PostgreSQL на різних профілях навантаження. Кількісні показники прискорення не заявляються без фактичних вимірювань.

Використання інструментів ШІ: не використовувались.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

[1] Graefe, G. (2011). Modern B-tree techniques. *Foundations and Trends in Databases*, 3(4), 203–402. https://doi.org/10.1561/1900000028

[2] Chaudhuri, S., & Narasayya, V. (1997). An efficient cost-driven index selection tool for Microsoft SQL Server. *Proceedings of the 23rd International Conference on Very Large Data Bases*, 146–155.

[3] Idreos, S., Kersten, M. L., & Manegold, S. (2007). Database cracking. *Proceedings of the 3rd Biennial Conference on Innovative Data Systems Research*, 68–78.

Shcherban V. V., Ozarinska O. V.

Adaptive indexing of relational databases under dynamic workloads

Abstract. *A methodology for adaptive index selection in relational databases under changing SQL workloads is proposed. The approach combines workload monitoring, candidate index generation, assessment of read performance against update overhead and storage costs, and controlled configuration changes. A stabilization mechanism is suggested to prevent excessive index rebuilding. The paper provides a methodological framework without claiming unverified experimental speedups.*

Keywords: *adaptive indexing; relational databases; SQL workload; index maintenance; query performance.*