УДК 004-658

Я.І. Корнага

ПОРІВНЯЛЬНІ ОЦІНКИ ЗАСТОСУВАННЯ МЕТОДІВ ПІДВИЩЕННЯ ШВИДКОСТІ ПОШУКУ ТА ЗАПИСУ ДАНИХ В БАЗАХ ДАНИХ

Анотація: Запропоновані нові методи оптимізації запису та пошуку інформації в базах даних. Проведено експериментальне дослідження збільшення швидкості запису та пошуку інформації. Здійснене порівняння швидкості пошуку зі стандартними методами, що використовують індекси на основі B+-дерев.

Ключові слова: запис даних, пошук даних, індекси, B+-дерева.

Вступ

В сучасних СУБД процесами в роботі з інформацією є запис та пошук її в них. Основною характеристикою цього процесу є швидкість. В сучасних СУБД при кожному зберіганні даних в таблицю відбувається перебудова індексів, що значно впливає на швидкість запису в БД. Користувачі повинні витрачати час не тільки на закінчення попереднього збереження, а і на перебудову індексів. Відповідно до цього час затримки перед наступною операцією запису (модифікації) даних доходить навіть до кількох хвилин, що абсолютно не допустимо при роботі баз даних в яких кілька сотень, а то і тисяч, клієнтів [1,2]. А якщо їх мільйони, то збереження буде тривати досить тривалий час та продуктивність роботи серверу БД відповідно буде дуже швидко падати [3,4,5].

Постановка задачі

Потрібно створити нові методи підвищення ефективності запису та пошуку інформації в базах даних, які б дозволили пришвидшити дані процеси та ефективніше використовувати ресурси СУБД.

Методи підвищення швидкості пошуку та запису в БД

Для підвищення ефективності швидкості пошуку однакових даних в різних таблицях було запропоновано *метод об'єднаних індексів*, який ефективно застосовується в тому випадку, коли в базі даних присутнє використання полів з однаковим типом та значенням в різних таблицях [6].

В листі дерева такого об'єднаного індексу потрібно зберігати запис-посилання на однакові поля в різних таблицях в форматі, як для звичайних B+-дерев, але з тією відмінністю, що запис крім посилання на розміщення даних в таблиці повинен містити ще і посилання на саму таблицю, як це показано на рисунку 1.

ключ1 запис1.i	ключ2 запис2.j	...	ключ k запис k.m

Рис. 1 – Структура листової сторінки дерева об'єднаного індексу

Значення запису-посилання $1, 2, \ldots, k$ показують конкретне посилання на дані, а значення запису-посилання $i, j, \ldots, m$ – номер таблиці в якій знаходяться дані.

Це дає можливість суттєво економити час пошуку даних в різних таблицях одночасно тому що, пошук буде здійснюватися по одному дереву.

Приведемо метод пошуку на основі алгоритму використання об'єднаних індексів:

1. Клієнт створює запит на пошук даних по кількох таблицях.
2. Запит обробляється компілятором та перевіряється на наявність об'єднаних індексів по даних полях та таблицях.
3. Вразі наявності об'єднаного індексу, використовується він, а якщо об'єднаний індекс відсутній, то використовують стандартні табличні індекси для пошуку даних.
4. Клієнт отримує результати пошуку даних по кількох таблицях.

Для того, щоб не відбувалось затримок в роботі серверу БД потрібно використовувати *метод тимчасових таблиць* для зберігання даних [6]. Потік інформації з запитами запису, який іде від клієнтів буде записуватися в них з тією умовою, що при запису даних не відбувається перебудова індексів по причині їх відсутності в даній таблиці. Тимчасові таблиці створюватимуться для таблиць, в які проводяться операції запису даних найчастіше та в яких використовується багато індексів для проведення пошуку даних.

Тимчасові таблиці створюються на основі існуючих основних таблиць та мають абсолютно ідентичний вигляд з ними. Єдиною відмінністю може бути добавляння службових полів для визначення історії записів, в які можуть входити: час запису та ідентифікатор користувача, який проводив запис.

Перенесення даних з тимчасової таблиці в основну може відбуватися за чотирма варіантами:

1. Перенесення за кількістю полів заповнення тимчасової таблиці.
2. Перенесення за плином часу заповнення.
3. Перенесення при найменшій активності користувачів.
4. Перенесення за вимогою адміністратора (користувача) БД.

Для кращого розуміння роботи тимчасових таблиць опишемо їх життєвий цикл, який показаний на рисунку 2.

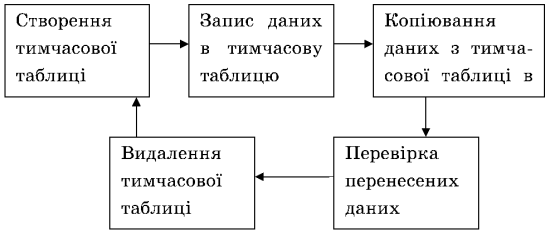

Рис. 2 – Схема роботи тимчасових таблиць

Для того, щоб почати роботу з тимчасовими таблицями, СУБД створює їх в форматі, який описаний вище та фізично розміщує робочому табличному файлі. Дані, які потрапляють на сервер від користувачів записуються в них на протязі часу, аж поки не настане момент копіювання даних в основні таблиці. Перед самим початком копіювання створюється нова тимчасова таблиця для того, щоб не переривалася робота по внесенню даних користувачами СУБД. Після завершення копіювання даних в основну таблицю відбувається перевірка даних, що були перенесенні. В разі, якщо все пройшло успішно, то відбувається перебудова індексів основної таблиці. Після проведення всіх вище вказаних операцій тимчасова таблиця видаляється, в зв'язку з тим, що на її місці уже працює нова тимчасова таблиця, в яку уже заповнюються нові дані.

Враховуючи вище сказане, опишемо метод запису (модифікації) даних на основі алгоритму використання методу тимчасових таблиць відповідно до варіантів перенесення даних до основних таблиць:

1. Клієнт створює запит на запис даних.
2. Компілятор обробляє запит та визначає таблиці, в які потрібно проводити запис (модифікацію).
3. Якщо для таблиці, в яку потрібно проводити запис, існує тимчасова таблиця, то запис проводиться у неї, а якщо відсутня – то запис проводиться в основну таблицю.
4. Якщо дані, які потрібно модифікувати, знаходяться в основній таблиці, то проводиться модифікація даних та перебудова індексів основної таблиці, а якщо дані, які потрібно модифікувати, не було знайдено в основній таблиці, то проводиться пошук і модифікація даних в додатковій таблиці.

Після завершення запису (модифікації) клієнту передається кількість даних, які були записані (модифіковані).

Аналітична оцінка параметрів методів запису та пошуку даних

Під час розрахунку часу пошуку даних по *методу об'єднаних індексів* візьмемо до уваги тай факт, що як зображено на рисунку 4 пошук може відбуватися за двома варіантами. Для першого варіанту характерно те, що пошук посилань на дані з різних таблиць у дереві відбувається по одних і тих самих вузлах, тоді економиться час $T_Д$ на зчитування з жорсткого диску вузлів індексу (дерева) для однієї з таблиць, в зв'язку з тим, що вони уже були зчитані для другої таблиці. Відповідно час пошуку $T_{ПО}$ буде розраховуватися за формулою:

$$T_{ПО} = m(T_Д + 2T_В), \tag{1}$$

де $T_Д$ – час зчитування вузлів дерева індексу пошуку з жорсткого диску, $T_В$ – час пошуку в вузлі дерева для першої та другої таблиці та m – висота дерева.

Для другого варіанту характерно те, що пошук посилань на дані з різних таблиць у дереві відбувається не по одних і тих самих вузлах, тоді економиться час тільки на зчитування з жорсткого диску вузлів індексу (дерева) для однієї з таблиць до того моменту, поки не розходяться шляхи пошуку по дереву не розійдуться. Відповідно час пошуку $T_П$ буде розраховуватися за формулою:

$$T_П = m(T_Д + 2T_В) + dT_Д, \tag{2}$$

де $T_Д$ – час зчитування вузлів дерева індексу пошуку з жорсткого диску для першої таблиці, $T_В$ – час пошуку в вузлі дерева для першої та другої таблиці, m– висота дерева та d – кількість вузлів, що додатково потрібно зчитати з жорсткого диску (d завжди менше m).

Отже, для розрахунку часу пошуку даних по методу об'єднаних індексів потрібно використовувати формулу 2, а в разі відсутності розбіжності у вузлах пошуку, то $d = 0$ та час пошуку розраховується за першим варіантом.

Фактично відбувається економія часу пошуку даних, що складає: для першого варіанту розрахунку $mT_Д$, а для другого варіанту - $(m - d)T_Д$.

Для оцінки параметрів запису інформації в БД за допомогою *методу тимчасових таблиць* потрібно ввести наступні позначення:

$T_З$ – час запису інформації в БД,

$T_{ПР}$ – час однієї перебудови індексів для відповідної таблиці,

$T_{ЗТ}$ – час одного запису інформації,

$T_{ПД}$ – час переносу даних з тимчасової таблиці в основну,

n – кількість записів, які проводяться у відповідну таблицю.

В загальному випадку при запису даних час запису розраховується за формулою:

$$T_З = n(T_{ЗТ} + T_{ПР}), \tag{3}$$

тобто, під час кожного запису тратиться час на сам запис інформації $T_{ЗТ}$ та час на перебудову індексів $T_{ПР}$.

Для методу використання тимчасових таблиць характерною особливістю є відсутність багаторазової перебудови індексів для таблиці, тому формула часу запису інформації в БД матиме вигляд:

$$T_З = nT_{ЗТ} + T_{ПР} + T_{ПД}, \tag{4}$$

тобто, замість виконання n перебудов індексів ми перебудовуємо їх тільки один раз.

Так як час $T_{ПР}$ перебудови індексів значно більший як час $T_{ПД}$ запису з тимчасової таблиці в основну, то $T_{ПД}$ можна знехтувати, тоді виграш часу запису даних з використання методу тимчасових таблиць складає $(n-1)T_{ПР}$, що дозволяє виконувати більше операцій запису одночасно.

Експериментальні оцінки параметрів методів запису та пошуку даних

Для проведення експериментів було розроблено спеціальний програмний комплекс для дослідження ефективності пошуку та запису інформації в БД, який можна адаптувати під різні операційні системи. Проведення експерименту по ефективності застосування індексів на основі методу об'єднаних індексів в базах даних було вибрано операційну систему: Linux Ubuntu.

При проведенні експерименту було проведено порівняння швидкості пошуку по індексах на основі B+-дерев та індексах побудованих на основі методу об'єднаних індексів.

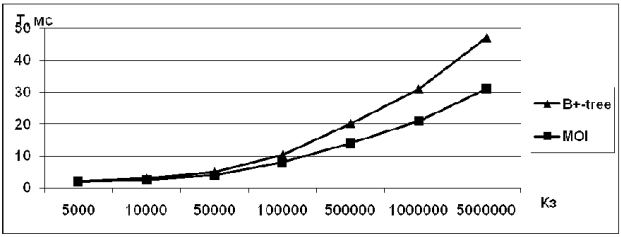

Рис. 3 – Залежність часу пошуку від кількості записів для методів формування індексів в БД за допомогою B+-дерев та MOI

В цілому ефективність використання методу об'єднаних індексів, як показано на рисунку 3 для заданої ОС на заданому проміжку кількості записів в таблицях бази даних складає майже 25% в

порівнянні з B+-деревами, що дає значне пришвидшення в роботі бази даних та дозволяє витрачати її ресурси на інші потреби.

Середньоквадратичне відхилення склало для B+-дерев майже 0,91%, а для методу об'єднаних індексів – 0,92%, з цього можна зробити висновок, що експеримент був проведений в рамках допустимих правил та отриманий результат відповідає потрібній точності.

Для проведення експерименту по ефективності застосування методу тимчасових таблиць в базі даних було створено тимчасову таблицю по якій не будувалися індекси. У цю таблицю добавлялися записи шляхом простого дописування їх в її кінець. А потім при заповнені таблиці на відповідну кількість записів вона копіювалася в основну таблицю і уже після збереження скопійованих даних відбувалася перебудова індексу. Експеримент проводився на операційній системі Linux Ubuntu.

Був проведений експеримент, завдяки якому визначилися час запису в таблицю, час копіювання в таблицю, час перебудови індексу та застосувавши формулу 5 розрахунку загального часу для проведення n операцій запису в БД було отримано, як показано на рисунку 4, графік в програмі MS Excel, апроксимація якого і дала функцію за допомогою якої можна розрахувати час побудови індексу для відповідної кількості записів в таблиці.

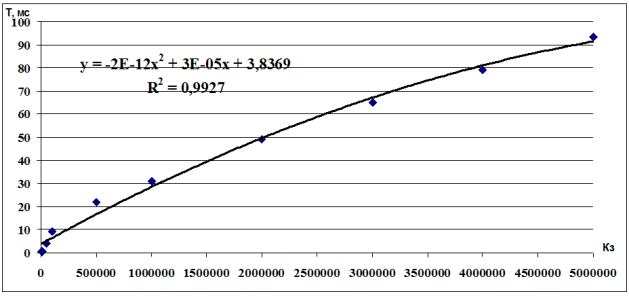

Рис. 4 – Час побудови індексів в залежності від кількості записів в таблиці

Відповідно формула для визначення екстрополярних даних наступна:

$$y = -2E - 12x^2 + 3E - 05x + 3.8, \tag{5}$$

де x – це кількість записів, а y – час за який відбувається перебудова індексу.

А для визначення часу побудови індексів для відповідної ділянки кількості записів про інтегруємо її та отримуємо:

$$y = -2E - 12x^3/3 + 3E - 05x^2/2 + 3.8x \tag{6}$$

Отже, маючи формулу 6 ми можемо розрахувати (таблиця 1) сумарний час запису відповідної кількості записів в основну таблицю та у тимчасову таблицю.

Таблиця 1

Порівняння часу запису в БД з застосування методу тимчасових таблиць та без застосування методу тимчасових таблиць

Кількість записів	Час запису в файл без застосвання методу тимчасових таблиць	Час запису в файл з застосвання методу тимчасових таблиць	Логарифм часу запису без МТТ	Логарифм часу запису з МТТ
5000	5,15	4,85	0,69	0,71
10000	14,60	7,65	0,88	1,16
50000	64,97	31,35	1,50	1,81
100000	279,73	82,38	1,92	2,45
500000	1966,67	294,28	2,47	3,29
1000000	6733,33	681,80	2,83	3,83
2000000	23866,67	1229,82	3,09	4,38
3000000	49800,00	1765,91	3,25	4,70
4000000	82933,33	2300,10	3,36	4,92
5000000	121666,67	2843,60	3,45	5,09

В зв'язку з тим, що порядок даних значно відрізняється для різної кількості записів, тому для кращого наглядного прикладу побудуємо графік залежності кількості записів від сумарного часу запису в таблицю взявши десятковий логарифм від сумарного часу.

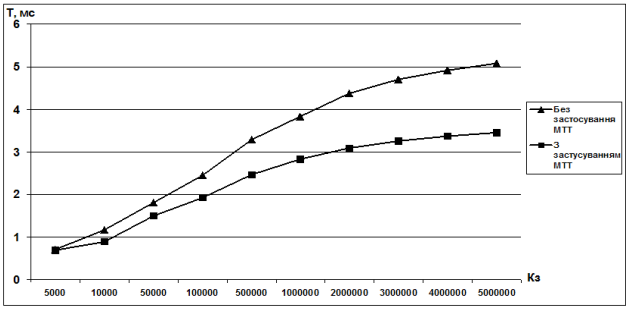

Рис. 5 – Логарифмічна функція часу добавлення відповідної кількості записів в таблицю без застосування та з застосуванням МТТ

На рисунку 5 чітко можна прослідкувати ефективність застосування тимчасових таблиць в базах даних, які не містять індексів.

Висновки

Пошук на основі методу об'єднаних індексів доцільно використовувати, для тих таблиць, які найчастіше попадають під пошук та поля в яких найчастіше порівнюють між собою. Зазвичай він буде використовуватися для двох таблиць, але зі збільшенням об'ємів БД та розподіленістю її елементів він може використовуватися і для більшої кількості таблиць. Використання методу хешування індексів призводить до покращення часу пошуку та швидкості перебудови індексів при запису (модифікації) даних в БД в тих випадках коли один і той самий індекс використовується в багато разів за короткий проміжок часу на 25%, а використання методу тимчасових таблиць для запису даних в БД веде до значного зменшення сумарного часу запису, в зв'язку з відсутністю в тимчасовій таблиці БД індексів та потребою в їх перебудові.

Таким чином, вибір певного методу запису та пошуку даних залежить від декількох параметрів побудови бази даних, а саме: кількості таблиць та записів у них, кількості звернень до відповідної таблиці, кількості побудованих індексів на відповідну таблицю та типів даних у відповідних полях БД.

Біблиографічний список

1. Marcus Hirt, Marcus Lagergren Oracle JRockit: The Definitive Guide / Packt Publishing, 2010. – 588 с.
2. Teo Lachev Applied Microsoft SQL Server 2008 Reporting Services / Prologika Press, 2008. – 768 с.
3. Горєв А., Ахаян Р., Макашарипов С. Ефективна робота із СУБД / Питер Кому, 2006. – 704 с.
4. Джен Л. Харрингтон Проектування реляционних баз даних / Лорі, 2006. – 230 с.
5. Коннор МакДональд Oracle PL/SQL для професіоналів. Практичні рішення / ДиаСофтЮП, 2006. – 560 с.
6. Мухін В.Є., Корнага Я.І. Механізми підвищення ефективності процедури моніторингу безпеки в розподілених базах даних / Вісник Національного технічного університету "Харківський політехнічний інститут". Збірник наукових праць. Серія: Інформатика та моделювання. / Харків: НТУ "ХПІ", 2012. – № 38., 128–135 с.

Отримано 08.02.2013