УДК 004.658; 004.62

Костенко П.П.

МЕТОД АВТОМАТИЧНОЇ ЗОВНІШНЬОЇ ОПТИМІЗАЦІЇ SQL-ЗАПИТІВ В УМОВАХ НЕВИЗНАЧЕНОСТІ ФІЗИЧНОЇ ТА ЛОГІЧНОЇ СТРУКТУРИ БАЗИ ДАНИХ

Кременчуцький національний університет імені Михайла Остроградського

В роботі розглянуто фактори, які впливають на швидкість отримання інформації в інформаційних системах. Викладено метод автоматичної зовнішньої оптимізації SQL-запитів на основі локальної моделі керованого процесу, який дозволяє проводити оптимізацію SQL-запитів незалежно від застосованої системи керування базами даних та її налаштувань. Представлена структурно-функціональна схема адаптивної системи зовнішньої оптимізації SQL-запитів.

Вступ

Неможливо уявити сучасну інформаційну систему (ІС) без програмного засобу зберігання інформації: бази даних (БД), сховища даних (СД), тощо. Найбільш розповсюдженими, бюджетними та легкими в використанні і обслуговування є реляційні бази даних (РБД). Системи керування базами даних (СКБД) стали невід'ємною частиною сучасних інформаційних систем (рис. 1).

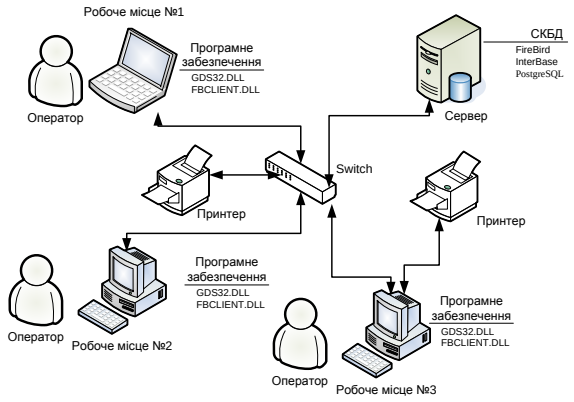

Рис. 1. Структурна схема інформаційної системи

Особливе місце в таких системах займають автоматизовані системи управління робочим місцем (АСУ РМ), які активно застосовуються для прискорення організації та роботи складних, багатоітераційних та розгалужених виробничих процесів і автоматизації медичних, юридичних закладів, бухгалтерських, економічних, статистичних відділів. Наразі розроблено спектр вітчизняного та іноземного програмного забезпечення (система автоматизації медичного закладу амбулаторного типу «Нейрон» ТОВ АЛЮР, автоматизована інформаційна система «Поликлиника» ТОВ ПАРУС, система автоматизації медичного закладу «ArchiMed» MLS IT Systems-Ukraine та ін..), що використовує в своїй організації АСУ РМ.

Максимальна автоматизація роботи оператора (лікаря, економіста тощо), ось головний чинник використання АСУ РМ. Тому постійно актуальною проблемою є максимізація швидкодії ІС на базі СКБД.

Постановка завдання

На швидкість отримання інформації в ІС впливають декілька чинників: програмна реалізація клієнтських частин ІС, апаратні характеристики клієнтських, серверних ПК та мережі передачі даних, структура БД, SQL–запити до БД тощо. Прискорення швидкодії ІС вбачається в оптимізації зазначених чинників.

До клієнтських частин ІС частіше застосовуються методи рефакторінгу [1,2] ніж оптимізації, це обумовлено значною їх складністю та насиченістю програмного коду. Оптимізація програмних кодів призводить до погіршення їх логічного та синтаксичного розуміння, тому оптимізація на цьому рівні проводиться в край рідко

Оптимізація апаратного забезпечення зводиться то систематичної заміни (модернізації) апаратних частин. В робо-

тах [3,4] детально розглянуто особливості апаратного забезпечення, методи взаємодії та підвищення розрахункових можливостей його складових.

Оптимізація структури БД передбачає реорганізацію інформації в базі даних на логічному рівні (нормалізацію бази даних)[5].

Під нормалізацією розуміється приведення інформації в базі даних до нормальних форм, що дозволяє звільнитися від збитковості та забезпечити цілісність даних. На сьогоднішній день виділяють шість нормальних форм [6].

Форми нормалізації дозволяють знизити збитковість даних, появи їх дублікатів, залучити методи підтримки цілісності даних. Як наслідок, значно зменшується ймовірність появи суперечливих даних, полегшується адміністрування бази та оновлення інформації в ній, скорочується обсяг дискового простору.

Проте така оптимізація призводить до конструювання «дуже» складних запитів, отримання даних значно уповільнюється, через, головним чином, велику кількість з'єднань таблиць. Тому, щоб збільшити швидкість вибірки даних і спростити програмування запитів, нерідко доводиться проводити вибіркову денормалізацію БД, та оптимізацію SQL – запитів.

Основні методи та алгоритми оптимізації SQL – запитів розглядались в роботах [7-13]. Говорячи про оптимізацію запитів в реляційних СКБД, зазвичай мають на увазі такий спосіб обробки запитів, коли за початковим поданням запиту шляхом його перетворень виробляється такий процедурний план його виконання, найбільш оптимальний за наявних у базі даних керуючих структурах. Відповідні перетворення початкового подання запиту виконуються спеціальним компонентом СКБД - оптимізатором, і оптимальність виробленого ним плану запиту носить умовний характер: план оптимальний відповідно до критеріїв, закладених в оптимізаторі, що породжує можливі відмінності від реальної оптимальності.

Оптимізатор запитів вибирає «найкращий» спосіб виконання запиту на основі відомих стратегій виконання елементарних складових запиту і способів композиції більш складних стратегій на основі елементарних. Тим самим, простір пошуку оптимального плану виконання запиту обмежений заздалегідь фіксованими елементарними стратегіями.

Внутрішній оптимізатор SQL володіє всіма необхідними даними для коректної оптимізації плану виконання запиту (мета дані БД, індекси, статистика тощо.), єдиним слабким місцем даної системі є правила оптимізації. Проте стандартний оптимізатор обмежений в своїй роботі одним запитом користувача, що призводить до генерації обмеженої кількості варіантів оптимізації. Вибір не оптимального плану виконання запиту обумовлюється некоректною формою запиту користувача.

Модифікація структури запиту, на боці клієнта, реалізується технологією рефакторінгу, що описана в роботі [14]. Рефакторінг SQL додатків – запитів кропіткий процес, що потребує періодичного застосування, в наслідок динамічного ускладнення структури та об’єму БД. Тому постає проблема оптимізації структури SQL – запитів, до їх виконання на сервері СКБД, підчас роботи ІС.

Аналіз попередніх досліджень

Фундаментальна робота Е. Кодда [15] з теорії реляційної моделі даних та його послідовників, зокрема М. Мейера [16], зумовила широке застосування РБД. Реляційна модель даних поєднує математичні та логічні моделі. Зазначені моделі встановлюють правила організації та роботи з інформацією, що зберігається в БД.

До переваг теорії реляційної моделі даних відносять: сувору логічну формалізацію, незалежність даних та можливість оптимізації структури бази даних за допомогою нормалізації. Отримання інформації з БД в переважній більшості випадків реалізовано з допомогою SQL–запитів.

Найбільшу питому вагу в аспекті використання розрахункових ресурсів та впливу на час обробки запиту мають етапи логічної оптимізації, розбору запиту та вибору оптимального плану виконання. Етап оптимізації запиту породжує ефективну стратегію виконання запиту користувача, тому розробка оптимальної системи генерації планів виконання запитів є актуальною задачею з початку 80-х років минулого століття.

Наразі відомо чотири стратегії оптимізації запитів:

• висхідна стратегія пошуку (Bottom-Up Optimizers).

• низхідна стратегія пошуку (Top-Down Optimizers).

• гібридна стратегія пошуку.

• Fuzzy Logic стратегія пошуку.

Недоліками зазначених стратегій вважаємо:

Недоліки Bottom-Up Optimizers:

1.Алгоритм динамічного програмування має експоненціальну залежність об'єму необхідної пам'яті від кількості поєднаних таблиць, що обмежує оптимізацію запитів з поєднанням більше 15 таблиць [17].

2.Евристичні правила базуються тільки на логічній інформації, а не вартісних оцінках, що інколи може призводити до вибору неоптимального плану виконання запиту.

3.Оптимізація складних не реляційних операторів потребує значної трудомісткої адаптації Starburst, що можна вважати суттєвим недоліком [18].

Недоліки Top-Down Optimizers:

Побудована стратегія пошуку призвела до відповідності продуктивності Volcano алгоритму Starburst в зв'язку з генерацією множини можливих логічних виразів до початку генерації фізичних виразів [19].

Недоліки гібридних оптимізаторів:

До недоліків, що знижують продуктивність, можна віднести використання віртуальних функцій, методи динамічного виділення і звільнення пам'яті, значне розділення середовища оптимізатора та програмного забезпечення адміністрування СКБД, надзвичайно малу ймовірність включення алгоритму «запам'ятовування» груп [20].

Недоліки Fuzzy Logic стратегії:

1.До недоліків адаптивних оптимізаторів відносять дворівневий механізм введення в експлуатацію. Перший рівень вимагає тривалого навчання з великою ймовірністю вибору неоптимального плану виконання запиту (QEP). Другий рівень передбачає поступове відключення адаптивного механізму та вибір оптимальних запитів, основаних на надійних статистичних і кореляційних фактах [21].

2.Сталий режим роботи при недостатньо тривалому зборі статистичних даних призводить до вибору QEP тільки за надійними фактами, незважаючи на оптимальний QEP, підтверджений сумнівними кореляційними та статистичними даними.

3.Вибір оптимального плану для одного запиту - процес не детермінований (аперіодичний), через випадковий вибір початкового плану та поступові «мутації» в процесі оптимізації. Внаслідок, результатом, для одного запиту, можуть виступити абсолютно різні плани з суттєво відмінним часом виконання.

Аналіз алгоритмів оптимізації SQL – запитів вказав на їх суттєву обмеженість та численні недоліки. Всі системи оптимізації, що було досліджено, залежні від структури та характеристик БД (метаданих), є невід'ємною частиною СУБД, обмежені в контексті структурної модернізації адміністратором СУБД.

Тому актуальною є проблема розробки оптимізатора, що буде незалежним від СУБД та структури БД (метаданих), потребуватиме мінімального терміну навчання, розглядатиме оптимізацію запитів, як періодичний процес, що дозволить усунути недоліки генетичних оптимізаторів.

Мета роботи. Розробка методу автоматичної зовнішньої оптимізації SQL-запитів в умовах невизначеності фізичної та логічної структури бази даних, який

дозволить проводити оптимізацію незалежно від СУБД та клієнтського програмного забезпечення.

Основна частина

Структурну схему ІС наведеної на рис. 1. в розгорнутому вигляді можна представити відповідно до рис. 2. Як видно з рис. 2. Структурно-функціональна схема частини ІС, що відповідає за обробку запитів користувача складається з трьох частин: робоче місце користувача (РМ), СКБД та БД. В зв'язку з необхідністю застосування оптимізатору до функціонуючих ІС та різних СКБД пропонується інтеграція додаткового модулю між РМ та СКБД. За таких умов оптимізатор обмежений до ступі до БД, тому процес оптимізації проводиться в умовах інформаційної невизначеності фізичної та логічної структури БД.

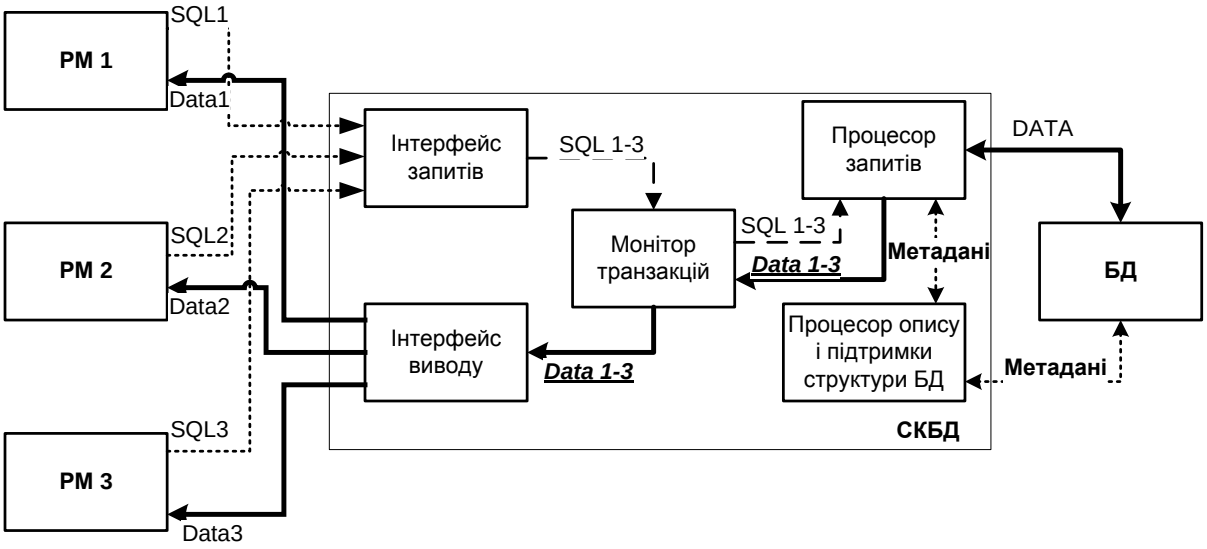

Рис. 2. Структурно-функціональна схема частини ІС, що відповідає за обробку запитів користувача

Система оптимізації має вигляд рис. 3. Та складається з адаптеру. оптимізатору, серверу та клієнту. Оскільки керування проводиться в умовах інформаційної невизначеності то застосування статистичних (ймовірнісних), еволюційних (нейромережі, генетичні алгоритми, апарат нечіткої математики та логіки) робастних системи керування неможливе. Це обумовлено їх суттєвим обмеженнями: статистичні системи керування потребують часу для накопичення статистики поведінки об'єкту та характеристик збурень; еволюційні системи керування використовують алгоритми зворотного поширення похибок з низькою швидкістю збіжності, потребують тривалого навчання; робастні системи керування потребують параметричної робастності об'єкта керування (ОК), точної моделі ОК, великої кількості розрахункових ітерацій.

В [22] показано, що шляхом проведення експерименту в режимі реального часу можливо отримати модель керованого процесу. Подібна локальна модель відображає в даний момент часу динаміку об'єкта керування та збурень, що діють на нього. Тому пропонується застосувати систему керування на основі локальної моделі керованого процесу в аспекті її адаптації з задачі стабілізації параметрів динамічної системи в часі до задачі оптимального вибору в дискретних системах.

Вибір системи керування на основі побудови локальної моделі керованого процесу (ЛМКП) для ядра методу оптимізації дозволяє використати наступні її переваги: отримання інформації про керований процес в реальному часі шляхом активного експерименту, побудова ЛМКП не потребує повної апріорної інформації про ОК, побудова ЛМКП не потребує тривалого накопичення статистичних да-

них про ОК, побудова ЛМКП не вимагає тривалого навчання.

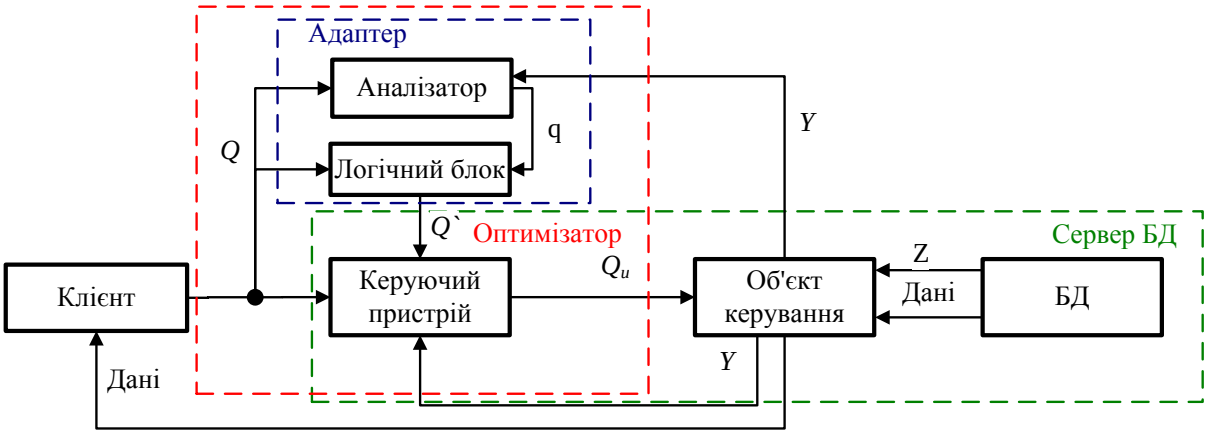

Рис. 3. Структурно-функціональна схема системи оптимізації

Метод оптимізації складається з 4 етапів:

1. Підготовчий етап:

- Моніторинг запитів та збереження їх до внутрішньої БД, з постановкою на оптимізацію.

- Оптимізація запитів в підчас простою системи.

2. Етап оптимізації:

- Визначення вагових коефіцієнтів запитів.

- Групування запитів за ваговими коефіцієнтами

- Побудова ЛМКП для запитів з черги на оптимізацію.

- Збереження оптимальної синтаксичної конструкції до внутрішньої бази даних.

3. Етап ін'єкції (підміни) запитів:

- Виявлення запитів до їх виконання СУБД.

- Визначення вагових коефіцієнтів запитів.

- Пошук оптимізованих запитів за ваговим коефіцієнтом у внутрішній базі даних.

- Підміна запиту та передача його на виконання.

4. Етап адаптивної оптимізації:

- Моніторинг часу виконання та вагових коефіцієнтів запиту.

- Постановка запиту на повторну оптимізацію у випадку збільшення часу його виконання.

- Генерація синтаксично відповідної структури для запитів з однаковими ваговими коефіцієнтами.

Висновки

Проведено аналіз існуючих стратегій оптимізації SQL-запитів. Виявлено їх переваги та недоліки.

Вперше запропоновано метод автоматичної зовнішньої оптимізації SQL-запитів в умовах невизначеності фізичної та логічної структури бази даних, який полягає у створенні та використанні локальної моделі процесу виконання запиту на основі вибору таких пробних SQL-запитів, які дають такий же час виконання в СКБД, як і SQL-запит, що ідеально відповідає тій частині фізичної та логічної структури даних, яка приймає участь у виконанні вхідного запиту та дозволяє зменшити час отримання інформації шляхом визначення оптимальної синтаксичної конструкції запиту.

Подальший розвиток роботи вбачається в розробці автоматичного програмного забезпечення перехвату та оптимізації запитів до їх виконання засобами СКБД.

Список літератури

1. Фаулер М. Рефакторинг: улучшение существующего кода = Refactoring: Improving the Design of Existing Code (2000)/ Фаулер М., — Спб: Символ-Плюс, 2004. — С. 430.

2. Скотт В. Эмблер, Рефакторинг баз данных: эволюционное проектирование = Refactoring Databases: Evolutionary

Database Design (Addison-Wesley Signature Series) / Скотт В. Эмблер, Прамодкумар Дж. Садаладж //. — М.: «Вильямс», 2007. — С. 368.
3. Mueller, Scott: Upgrading and repairing PCs / Scott Mueller. — 19th ed. 2010 p. 1156.
4. Terry W. Ogletree, Upgrading and Repairing Networks / Terry W. Ogletree, Mark Edward Soper //, Fifth Edition, Que, 2006 p. 1200.
5. Коломейчук В.В. Розробка та дослідження бази даних для системної обробки статистичної інформації / В.В. Коломейчук // Математичні машини і системи. — 2009. — № 4. — С. 89-95
6. Дейт К. Дж. Введение в системы баз данных / Дейт К. Дж. — 8-е изд. — М.: «Вильямс», 2006. – С. 1328.
7. Pirahesh Hamid. Extensible/Rule Based Query Rewrite Optimization in Starburst /. Pirahesh Hamid, Hellerstein Joseph M., Hasan Waqar //. – In Proceedings of the 1992 ACM SIGMOD International Conference on Management of Data, San Diego, California. – June. – 1992. – p. 39-48.
8. Chaudhuri S. An Overview of Query Optimization in Relational Systems / Chaudhuri S. – PODS-98. – Seattle WA, USA. – 1998.
9. Graefe Goetz. The Volcano Optimizer Generator: Extensibility and Efficient Search. / Graefe Goetz, McKenna William // In Proceeding of the 12th International Conf. on Data Engineering, – 1993. P. 209-218.
10. Stillger M. LEO - DB2's LEarning Optimizer. / M. Stillger, G. M. Lohman, V. Markl, and M. Kandil. // In Proc. VLDB. – 2001. – P. 19–28.
11. Markl V. Robust Query Processing through Progressive Optimization. / V. Markl, V. Raman, D. E. Simmen, G. M. Lohman, H. Pirahesh.// In Proc. ACM SIGMOD. – 2004. – P. 659–670.
12. Babu S. Adaptive Query Processing in the Looking Glass. / S. Babu, P. Bizarro// In Proc. CIDR. – 2005.
13. Deshpande A. Adaptive query processing. / A. Deshpande, Z. Ives, V. Raman // Foundations and Trends in Databases. – 1(1). – 2007. – P. 1–140.
14. Фаро С. Рефакторинг SQL приложений / С. Фаро, Л. Паскаль // Пер. с англ. – СПб: Символ-Плюс, 2009. – 336 с., ил.
15. Codd, E.F. A Relational Model of Data for Large Shared Data Banks. / E.F. Codd // Communications of the ACM 13 (6). – 1970. – P. 377–387.
16. Мейер М. Теория реляционных баз данных / Мейер М. – Москва: Мир, 1998. – 608 с.
17. Selinger P. Access Path Selection in a Relational Database Management System / P. Selinger, M. Astrahan, D. Chamberlin // Proceedings of the ACM SIGMOD International Conference Management Data. – Boston. – 1979. – P. 23-34.
18. Pirahesh H. Extensible: Rule Based Query Rewrite Optimization in Starburst / Pirahesh H., Hellerstein J., Hasan W. // Proceedings of the ACM SIGMOD International Conference on Management of Data. – San Diego. – 1992. – P. 39-48.
19. Graefe G. The Cascades Framework for Query Optimization / Graefe G. – Bulletin of the IEEE Technical Committee on Data Engineering (Washington). – 1995. – Vol. 18. – N.3. – P. 19-29.
20. Ozcan F. A Region Based Query Optimizer Through Cascades Query Optimizer Framework / Fatma Ozcan, Sena Nural, Pinar Koksal, Mehmet Altinel, Asuman Dogac //. IEEE Data Eng. Bull. 18(3). – 1995. – P. 30-40.
21. Markl V. LEO: An autonomic query optimizer for DB2. / V. Markl, G. M. Lohman, V. Raman. // IBM Systems Journal, Vol. 42. – Num. 1. – 2003.
22. Гученко М.І. Активно-резонансний принцип керування / М.І. Гученко // 16-а Міжнародна конференція з автоматичного управління «Автоматика-2009». Тези доповідей. – Чернівці. – 2009. – С. 59-61