УДК 004.6

ОПТИМІЗАЦІЯ ВИКОНАННЯ ЗАПИТІВ У РЕЛЯЦІЙНІЙ СИСТЕМІ УПРАВЛІННЯ БАЗАМИ ДАНИХ

Єрмолаєв О. Д., Суліма С.В.
Навчально-науковий Інститут телекомунікаційних систем КПІ ім. Ігоря Сікорського, Україна
E-mail: itssulima@gmail.com

OPTIMIZATION OF QUERIES EXECUTION IN THE RELATIONAL DATABASE MANAGEMENT SYSTEM

A method of optimizing the synthesis of complex SQL queries from many simple ones is presented, which allows to increase the speed of execution of the input query by the relational database while simultaneously ensuring high performance and ease of use.

Обробка SQL – це синтаксичний розбір, оптимізація, генерація джерела рядків та виконання оператора SQL. На рис. 1 зображені загальні етапи обробки SQL запиту.

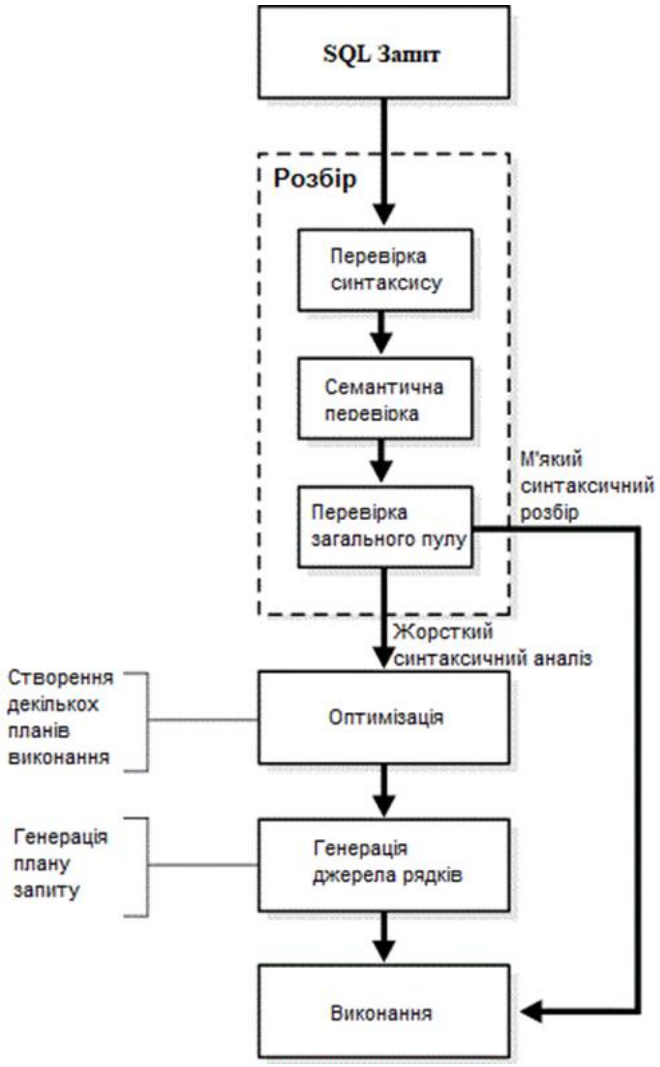

Рис.1. Загальні етапи обробки SQL запиту.

Щоб допомогти програмістам баз даних, на ринку існує безліч інструментів автоматизації налаштування SQL, які допомагають виявляти проблемні SQL-оператори та пропонують варіанти їх виправлення [1]. Незважаючи на те, що автоматичні інструменти можуть полегшити налаштування SQL, воно все одно потребує значної участі людини.

Щоб забезпечити оптимальну продуктивність складання набору запитів, рекомендується використовувати метод проб і помилок, тобто написати кілька варіантів запитів та порівняти їх плани планування та виконання. Дослідники запропонували багато різних методів та порад, які можна використовувати для підготовки та тестування оптимізації продуктивності запитів [1-6]. Найбільш поширеними методами визначення найкращих можливостей складання запитів є аналіз кількості логічних даних, створених запитів та перегляд графічних планів.

Тимчасові таблиці – функція, яка дозволяє зберігати та обробляти проміжні результати за допомогою тих самих можливостей вибору, оновлення та приєднання, які можна використовувати з типовими таблицями SQL Server [7]. Тимчасові таблиці мають різні варіанти, включаючи, серед іншого, локальні тимчасові таблиці (починаються з #), глобальні тимчасові таблиці (починаються з ##) та змінні таблиці (починаються з (@). Збережені процедури можуть посилатися на тимчасові таблиці, створені під час поточного сеансу.

Отже, удосконалений метод, який пропонується далі, засновується на уникненні оператора IN в реченні WHERE [8].

Рішення:

Створити тимчасову таблицю і вставити необхідні дані і ПРИЄДНАТИСЬ до основного запиту.

Якщо складний запит включає оператор IN, його необхідно видалити, а для створення тимчасової таблиці і включення її в запит необхідно виконати наступні кроки:

1. Визначте поля SELECT замість SELECT *: Якщо таблиця має багато полів і рядків, виділення всіх стовпців (за допомогою SELECT *) надмірно використовує ресурси бази даних для запиту великої кількості непотрібних даних. Визначення полів у операторі SELECT вкаже базу даних на запит лише необхідні дані.
2. Уникайте SELECT DISTINCT, якщо це можливо: SELECT DISTINCT працює, групуючи всі поля в запиті, щоб створити чіткі результати. Однак для досягнення цієї мети необхідна велика кількість обчислювальної потужності.
3. Використовуйте WHERE замість HAVING для визначення фільтрів: Відповідно до порядку операцій SQL, оператори HAVING обчислюються після операторів WHERE. Якщо нам потрібно відфільтрувати запит на основі умов, оператор WHERE є більш ефективним.
4. Використовуйте LIMIT для вибірки результатів запиту: Перед першим запуском запиту переконайтеся, що результати будуть бажаними та значущими, використовуючи оператор LIMIT. Слід зазначити, що в деяких СУБД – LIMIT просто не існує, тому зручно використовувати даний синтаксис:

SELECT TOP 3 * FROM Table;

5. Замініть підзапит на JOIN: Хоча підзапити корисні, їх часто можна замінити об’єднанням, яке, безумовно, виконується швидше. Розглянемо приклад нижче:

```
SELECT a.id, (SELECT MAX(created)
FROM posts
WHERE author_id = a.id)
AS latest_post FROM authors a
```

Щоб уникнути підзапиту, його можна переписати за допомогою об’єднання як:

```
SELECT a.id, MAX(p.created) AS latest_post
FROM authors a
INNER JOIN posts p
ON (a.id = p.author_id)
GROUP BY a.id
```

6. Видалити оператор IN, створити таблицю #temp і вставити в неї необхідні дані.

7. Створити індекс для таблиці #temp.

8. З'єднайте #temp з базовою таблицею

Запропоновано удосконалений метод оптимізації SQL запитів у випадку якщо швидкість вибірки даних почала просідати за часом, який використовує заміну оператора IN на тимчасову таблицю та використовує не кластеризований індекс, що дозволяє на відміну від існуючих пришвидшити вибірку за рахунок зменшення логічних звернень.

Література

1. M. L. Rupley, “Introduction to query processing and optimization,” Indiana University, South Bend, South Bend, IN, USA, techreport TR– 20080105–1, 2008.
2. R. Bhajipale, P. Bisen, A. Meshram, and S. S. Thakur, “SQL tuner,” International Journal of Computer Trends and Technology, vol. 33, no. 1, pp. 29–32, 2016.
3. P. Karthik, G. T. Reddy, and E. K. Vanan, “Tuning the SQL query in order to reduce time consumption,” International Journal of Computer Science Issues, vol. 9, no. 4/3, pp. 418–423, 2012.
4. J. Habimana, “Query optimization techniques – tips for writing efficient and faster SQL queries,” International Journal of Scientific & Technology Research, vol. 4, no. 10, pp. 22–26, 2015.
5. R. Sahal, M. Nihad, M. H. Khafagy, and F. A. Omara, “iHOME: Index based JOIN query optimization for limited big data storage,” Journal of Grid Computing, vol. 16, no. 2, pp. 345–380, 2018.
6. M. Sharma, “Query optimization using SQL transformations,” International Journal of IT, Engineering and Applied Sciences Research, vol. 1, no. 1, pp. 100–104, 2012.
7. Phil Factor Тимчасові таблиці URL: https://www.red-gate.com /simple-talk/sql/t-sql programming/temporary-tables-in-sql-server/.
8. Єрмолаєв, О. Д. Метод оптимізації SQL запитів : дипломна робота … бакалавра : 172 Телекомунікації та радіотехніка / Єрмолаєв Олександр Дмитрович. – Київ, 2021. – 56 с.