УДК 004.42, 004.657

В. В. Свіжак
О. В. Романюк

ПРОГРАМНИЙ ДОДАТОК ДЛЯ ДОСЛІДЖЕННЯ ПРОДУКТИВНОСТІ SQL-ЗАПИТІВ

Вінницький національний технічний університет

Анотація

Проаналізовано основні причини зниження продуктивності виконання запитів до бази даних. Розглянуто особливості розробки додатку для дослідження продуктивності SQL-запитів.
Ключові слова: *SQL-запит, продуктивність, база даних, СКБД.*

Abstract

The main reasons for the decrease in database query performance are analyzed. The article discusses application for performance research SQL-query.
Keywords: *SQL- query, performance, database, DBMS.*

Вступ

Декларативні мови програмування – це мови програмування високого рівня, в яких програмістом не задається покроковий алгоритм рішення задачі («як» вирішити завдання), а деяким чином описується, «що» потрібно отримати як результат [1]. Механізм обробки зіставлення за зразком декларативних тверджень вже реалізовано у пристрої мови.

За статистикою [2] 7 із 10 найпопулярніших систем керування базами даних (СКБД) використовують реляційну модель бази даних. Реляційна модель визначає представлення даних (структура), захищеність від некоректних змін (цілісність) та операції, що можуть бути виконані з даними (операції з даними). Такі СКБД для написанні запитів використовують різні варіації основані на декларативній мові SQL. Найпопулярнішими реляційними СКБД, згідно зі статистикою, є Oracle, MySQL та Microsoft SQL Server. Через широкий вибір інструментів для розробників для дослідження було вибрано середовище Microsoft SQL Server. Мова T-SQL, що використовується для написання запитів, має велику функціональність, а вбудований SQL CLR дає змогу розробникам створювати власні функції за допомогою засобів .Net.

Оскільки в мовах для написання запитів, основаних на SQL, на користувачу не лежить відповідальність за алгоритм виконання – багато користувачів вважають, що послідовність умов у блоці WHERE, або порядок з'єднань таблиць через JOIN не має різниці, але це не є так. У кожній СКБД представлені власні механізми для виконання та оптимізації запитів. Тому, для максимальної швидкодії роботи БД, потрібно вміти правильно будувати структуру запитів.

Отже, метою дослідження є підвищення продуктивності виконання SQL-запитів у середовищі MS SQL.

Об'єкт дослідження – процес оптимізації SQL-запитів.

Предмет дослідження – алгоритми та засоби для аналізу продуктивності SQL-запитів.

Основна задача – розробка додатку для реалізації заданої мети.

Розробка додатку

Основною задачею розроблюваного додатку є аналіз запитів мовою T-SQL та надання користувачу вказівок по покращенню написаного ним запиту.

В результаті проведеного аналізу [3] було виявлено, що найбільш типовими помилками є зайва конвертація даних, використання констант у арифметичних операціях з індексами, некоректне використання функцій в умовах, використання порядкового виконання запиту.

Для прикладу можна розглянути ситуацію, коли у таблиці Purchasing існує індекс по полю PurchaseOrder, однак при умові «WHERE Purchasing.PurchaseOrder * 2 = 3400» цей індекс не буде

використаний, що призводить до сканування всієї таблиці. Дану умову можна просто замінити на подібну «WHERE Purchasing.PurchaseOrder = 3400 / 2» – індекси будуть коректно використовуватись, що дає зменшення операцій читання у декілька разів.

Іншою досить популярною проблемою [4] є використання оператора IN разом із підзапитами. Багато користувачів вважають його дуже зручним для використання не лише зі сталим переліком, а й з вкладеними запитами. Суть проблеми полягає в тому, що такий вкладений підзапит буде виконуватись окремо для кожного запису. Так в умові «WHERE CustomerId NOT IN (SELECT …)», отримуємо не 1 очікуване виконання під запиту, а для кожного запису, що сильно знижує швидкість виконання. Замість некоректного використання оператора IN слід використовувати оператори OUTHER JOIN. Таким чином, змінивши умову на «LEFT JOIN (SELECT …) sub ON main.CustomerId = sub.CustomerId WHERE main.CustomerId IS NULL», отримаємо кількість операцій, що буде рости лінійно залежно від кількості даних, а не експоненціально, як у випадку з використанням оператора IN.

Ще одним цікавим випадком є порядок переліку таблиць в з'єднаннях оператором JOIN. Якщо таблиці сильно відрізняються за кількістю записів, кращою практикою буде розміщення таблиць від менших до більших. Під час виконання для кожного запису з лівої таблиці знаходиться відповідний запис з правої таблиці. Тому, якщо при пошуку шукати відповідність для кожного запису з меншої таблиці – це займе набагато менше часу, ніж перевіряти кожен запис з більшої таблиці, навіть у випадку, якщо всі записи більшої таблиці мають зв'язок з записом меншої таблиці.

На основі цього було сформовано основний функціонал додатку:

- попередній аналіз часу виконання запиту;
- можливість перевіряти зміни перед внесенням їх до БД;
- пошук типових помилок в написанні запитів;
- перевірка коректності використання індексів;
- розбиття великих запитів на під-запити з використанням тимчасових таблиць.

Додаток є розширенням для середовища MS SQL, що полегшує його використання. Для розробки й інтеграції даного додатку використовувались інструменти SQL CLR та мови програмування C# та T-SQL. Для аналізу запиту використовувався інструмент Query Plan, який надає змогу отримати реальні команди, які будуть викликатись під час виконання запиту.

Висновки

Проаналізовано основні помилки при написанні запитів до бази даних. Розроблено додаток для аналізу запитів мовою T-SQL, який дозволяє користувачу знаходити проблеми в структурі запиту та створювати більш ефективні SQL-запити.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Lloyd, J.W., Practical Advantages of Declarative Programming, 1994. – 17 с
2. DB-Engines [Електронний ресурс] – Режим доступу: https://db-engines.com/
3. Семь смертных грехов программиста на T-SQL (Хабр) [Електронний ресурс] – Режим доступу: https://habr.com/ru/post/149235/
4. 10 Common SQL Programming Mistakes and How to Avoid Them (UpWork) [Електронний ресурс] – Режим доступу: https://www.upwork.com/hiring/data/common-sql-programming-mistakes/

Свіжак Віктор Вячеславович, студент групи 1ПІ-16б, факультет інформаційних технологій та комп'ютерної інженерії, Вінницький національний технічний університет, Вінниця, e-mail: viktor.svizhak@gmail.com

Романюк Оксана Володимирівна, к.т.н., доцент кафедри програмного забезпечення, Вінницький національний технічний університет, Вінниця, e-mail: romaniukoksanav@gmail.com

Viktor Svizhak, student of group 1PI-16b, Faculty for Information Technologies and Computer Engineering, Vinnytsia National Technical University, Vinnytsia, e-mail: viktor.svizhak@gmail.com

Oksana Romaniuk – Candidate of Technical Sciences, Associate Professor of the Software Chair, Vinnytsia National Technical University, Vinnytsia, e-mail: romaniukoksanav@gmail.com