УДК 004.657

Реализация вставки в PostgreSQL с использованием дерева отрезков

Мартынов В. А.✉, Плотникова Н. П.

Национальный исследовательский Мордовский государственный университет им. Н. П. Огарёва, Саранск, 430005, Республика Мордовия, Российская Федерация

Постановка задачи: *Данная статья является продолжением статьи [1], в которой рассматривался способ оптимизации агрегирующих запросов на непрерывном диапазоне строк к базам данных PostgreSQL с помощью дерева отрезков [2]. Недостатком рассмотренного подхода была длительная операция вставки новых значений в таблицу, что влекло за собой полное перестроение дерева.* ***Целью работы*** *является разработка способа, позволяющего значительно ускорить вставку новых элементов в таблицу с поддержанием актуальной информации в дереве отрезков, при этом полностью сохранив выигрыш в скорости при агрегирующих запросах.* ***Новизна*** *данного подхода заключается в том, что он основан на особенности дерева отрезков, которая позволяет обновлять элементы в нем за асимптотику* $O(\log_2 N)$. *Тогда можно свести задачу добавления новых значений к обновлению существующих элементов, что можно решить более эффективно. Для этого при построении дерева выделяются дополнительные нейтральные элементы. При добавлении новых значений в таблицу обновляется один из этих дополнительных элементов. Когда свободные элементы в дереве заканчиваются, дерево перестраивается, снова выделяя дополнительную память.* ***Результатом*** *работы является разработка алгоритма обновления элементов в дереве и его реализация в виде расширения PostgreSQL Extensions. Проведены замеры скорости полученного решения, сделаны выводы по итогам работы и планы по дальнейшей оптимизации операций.*

Ключевые слова: *база данных, дерево отрезков, PostgreSQL, insert, асимптотика*

Введение

В статье [1] был рассмотрен подход с деревом отрезков [2–4], позволяющий значительно ускорить агрегирующие запросы к таблице на непрерывном диапазоне строк. С помощью данного подхода удалось в 87 раз сократить время выполнения агрегирующих запросов для таблицы из 100 млн строк.

Рассмотрим задачу множественных агрегирующих запросов к таблице на непрерывном диапазоне строк. Как правило, такие запросы поступают к таблицам, в которых данные хранятся в хронологическом порядке естественным образом, и требуется сформировать отчет за некоторый временной промежуток [5]. Примерами таких данных на практике могут служить:

– журнал банковских транзакций;
– история активности пользователей социальной сети;
– история расходов и доходов на предприятии.

Библиографическая ссылка на статью:
Мартынов В. А., Плотникова Н. П. Реализация вставки в PostgreSQL с использованием дерева отрезков // Информационные технологии и телекоммуникации. 2023. Т. 11. № 2. С. 1–7. DOI: 10.31854/2307-1303-2023-11-2-1-7

Reference for citation:
Martynov V., Plotnikova N. Implementation of INSERT in PostgreSQL in a Segment Tree. *Telecom IT.* 2023; 11(2):1–7. DOI: 10.31854/2307-1303-2023-11-2-1-7

Однако в большинстве подобных задач необходимо также достаточно эффективно добавлять новые записи в таблицу. Недостаток данного подхода – необходимость полностью перестраивать дерево при добавлении новых записей. Это занимает $O(N)$ времени, где N – количество строк таблицы, что при больших объемах данных становится неприемлемым.

Чтобы данное решение оставалось применимым в реальных задачах, необходимо добиться удовлетворительного времени вставки новых записей. Для этого был разработан подход, позволяющий свести операцию вставки новых записей к операции обновления, которую можно реализовать за асимптотику $O(\log_2 N)$ в дереве отрезков. В данной статье будет рассмотрен этот подход, позволяющий снизить асимптотику вставки до значения $O(\log_2 N)$, а также проведены замеры скорости.

Изменение элемента в дереве отрезков

Рассмотрим дерево отрезков на сумму для массива [3, 1, 2, 5, 6, 5, 3, 2], изображенное на рисунке 1. При изменении элемента массива необходимо пересчитать значения некоторых вершин дерева. Например, если мы меняем шестой элемент с 5 на 8, изменятся вершины, выделенные красным на рисунке 2.

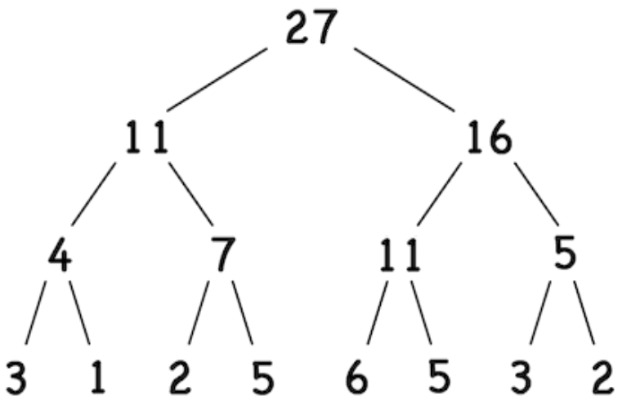

Рис. 1. Дерево отрезков на сумму для массива [3, 1, 2, 5, 6, 5, 3, 2]

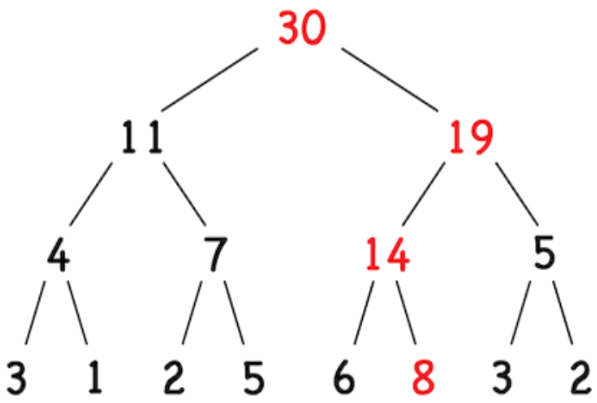

Рис. 2. Изменение элемента в дереве отрезков

Как видно из рисунка 2, при изменении элемента необходимо обновить только родительские вершины, количество которых, исходя из высоты дерева, будет порядка $O(\log_2 N)$ штук. Таким образом, мы можем изменять элементы массива и поддерживать актуальную информацию в дереве отрезков за асимптотику $O(\log_2 N)$ на одно изменение. Применим эту возможность для операции вставки нового элемента в исходный массив.

Дополнительные элементы

Основная идея заключается в том, чтобы свести операцию вставки нового элемента к операции изменения элемента в массиве. Выделим при первичном построении дерева еще N дополнительных элементов, которые будут нейтральными по отношению к текущей функции. Для функции суммы нейтральным

DOI: 10.31854/2307-1303-2023-11-2-1-7
URL: https://www.sut.ru/doci/nauka/1AEA/ITT/2023_2/1-7.pdf

элементом будет 0. Например, изначальный массив состоял из четырех элементов [3, 1, 2, 5], добавим еще четыре элемента равных 0. Получив массив [3, 1, 2, 5, 0, 0, 0, 0], построим на нем дерево отрезков (рисунок 3).

Теперь вместо добавления нового элемента будем изменять первый не используемый нейтральный элемент. Например, если в массив добавили число 3, дерево изменится, как изображено на рисунке 4.

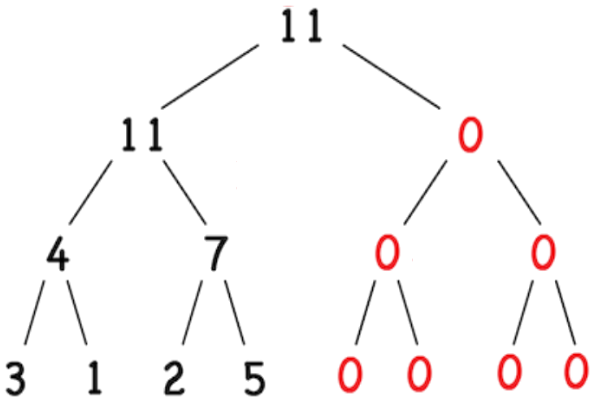

Рис. 3. Дополнительные нейтральные элементы в дереве отрезков

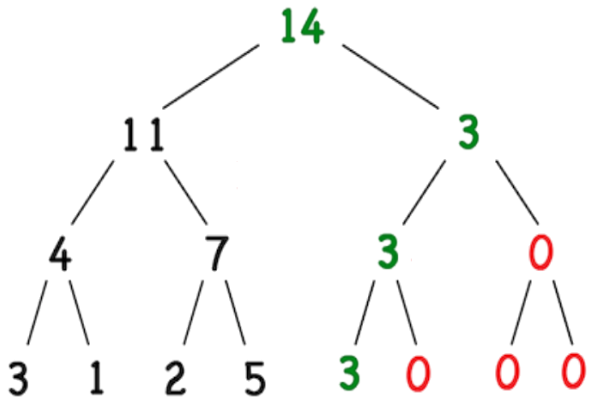

Рис. 4. Обновление нейтрального элемента

Таким образом, пока есть свободные нейтральные элементы, мы можем имитировать операцию вставки нового элемента за асимптотику $O(\log_2 N)$. Когда нейтральные элементы будут заканчиваться, перестроим дерево отрезков, снова выделив в два раза больше памяти. Тогда перестраивать дерево придется все реже, так как дополнительных элементов будет выделяться каждый раз в два раза больше.

Перенесем все рассуждения на базу данных PostgreSQL [6], описанную в статье [1]. При создании дополнительной таблицы добавим N строк со значением $AggField$, равным 0, и на этих $2 \cdot N$ строках построим дерево отрезков. Реализуем функцию *insert* на языке SQL [7] в виде PostgreSQL Extension [8], дополнив функционал уже имеющегося расширения.

Реализация функции *insert*

Insert выполняется в два этапа:

1) стандартная вставка строки в исходную таблицу и возвращение Id;

2) изменение элемента с индексом Id в дереве отрезков.

Изменение элемента происходит путем рекурсивного вызова от корня дерева. Тогда для текущей вершины есть два случая:

– вершина является листом (тогда обновляем значение поля $AggField$ соответствующей строки на заданное значение);

– вершина не является листом (тогда смотрим, отрезку какого из «детей» принадлежит Id, и запускаемся рекурсивно в этого «ребенка»; после этого пересчитываем значение для текущей вершины).

DOI: 10.31854/2307-1303-2023-11-2-1-7
URL: https://www.sut.ru/doci/nauka/1AEA/ITT/2023_2/1-7.pdf

Тесты скорости

Протестируем скорость реализованной функции и сравним со скоростью стандартной вставки в таблицу. Все эксперименты проводились на машине со следующими характеристиками:

– CPU – AMD Ryzen 5 3600;

– RAM – 64 Gb;

– SSD – Force MP600 1 Tb;

– OS – Ubuntu 22.04.1;

– PostgreSQL 14.8 [9].

Ниже (таблица 1) представлены замеры скорости для различных размеров таблицы для реализованной и стандартной [10–11] функций. Время посчитано для 1000 вставок.

Таблица 1 – Сравнение времени вставки

Размер таблицы, строка	Время реализованной функции, с	Время обычной вставки, с
1 000	2,715	0,109
5 000	2,738	0,100
100 000	2,886	0,139
1 000 000	3,205	0,090
10 000 000	3,719	0,094
100 000 000	4,069	0,094

Так как помимо вставки необходимо поддерживать актуальную информацию в дереве, время реализованной функции уступает времени обычной вставки. Однако при этом сохранен выигрыш в скорости агрегирующих запросов. Для наглядности в таблице 2 приведены коэффициенты прироста скорости агрегирующих запросов [1] для разных размеров таблицы, а также коэффициент проигрыша в скорости операции вставки.

Таблица 2 – Коэффициенты прироста и проигрыша скорости операций

Размер таблицы, строка	Коэффициент прироста скорости агрегирующих запросов	Коэффициент проигрыша скорости операции вставки
1 000	0,28	24,91
5 000	0,15	27,38
100 000	1,45	20,76
1 000 000	8,80	35,61
10 000 000	32,74	39,56
100 000 000	87,13	43,29

DOI: 10.31854/2307-1303-2023-11-2-1-7
URL: https://www.sut.ru/doci/nauka/1AEA/ITT/2023_2/1-7.pdf

Заключение

В рамках исследования реализована вставка новых строк в таблицу PostgreSQL с поддержанием актуальной информации в дереве отрезков за асимптотику $O(\log_2 N)$.

Весь функционал реализован в рамках имеющегося расширения, описанного в статье [1], при этом его возможности дополнены операцией вставки. Таким образом разработан совершенно новый подход, опирающийся на хранение частичной информации по агрегируемым данным. Он открывает новые возможности анализа данных, позволяя выполнять запросы на агрегацию больших объемов информации на непрерывном диапазоне асимптотически эффективнее существующих решений.

Реализованная функция работает в 25–43 раза медленнее обычной вставки из-за необходимости параллельного обновления дерева отрезков. Однако позволяет использовать все основные операции с базой данных, сохранив при этом 87-кратное преимущество подхода в агрегирующих запросах.

В дальнейшем планируется оптимизировать функции путем применения массовых запросов к базе данных и произвести более детальное сравнение всех функций с имеющимися решениями для подобных задач.

Литература

1. Мартынов В. А., Плотникова Н. П. Применение дерева отрезков в PostgreSQL // Инженерный вестник Дона. 2023. № 9 (105). С. 142–151.

2. Chang Y. K., Lin Y. C. Dynamic Segment Trees for Ranges and Prefixes // IEEE Transactions on Computers. 2007. Vol. 56. Iss. 6. PP. 769–784. DOI: 10.1109/TC.2007.1037

3. Wang L., Wang X. A Simple and Space Efficient Segment Tree Implementation // MethodsX. 2019. Vol. 6. PP. 500–512. DOI: 10.1016/j.mex.2019.02.028

4. Zheng C., Shen G, Li S., Shenker S. Distributed Segment Tree: Support of Range Query and Cover Query over DHT // Proceedings of the 5th International Workshop on Peer-to-Peer Systems (IPTPS). 2006. URL: http://web2.cs.columbia.edu/~cxz/publications/iptps2006-DST.pdf (дата обращения 11.08.2023)

5. Кудрявцев Ю. OLAP-технологии: обзор решаемых задач и исследований // Бизнес-информатика. 2008. № 1(3). С. 66–70.

6. Momjian B. PostgreSQL: Introduction and Concepts. New York: Addison–Wesley, 2001.

7. Супрунов С. PostgreSQL: функции и триггеры // Системный администратор. 2004. № 10. С. 42–47.

8. PostgreSQL Extensions. URL: https://wiki.postgresql.org/wiki/Extensions (дата обращения 11.08.2023)

9. PGTune // PGTune. URL: https://pgtune.leopard.in.ua (дата обращения 11.08.2023)

10. Arge L., Hinrichs K. H., Vahrenhold J., Vitter J. S. Efficient Bulk Operations on Dynamic R-Trees // Algorithmica. 2002. Vol. 33. PP. 104–128. DOI: 10.1007/s00453-001-0107-6

11. Imasheva B., Nakispekov A., Sidelkovskiy A., Sidelkovskaya A. The Practice of Moving to Big Data on the Case of the NoSQL Database, Clickhouse // Proceedings of the 6th World Congress on Global Optimization of Complex Systems: Theory, Models, Algorithms and Applications (WCGO 2019). Cham: Springer, 2019. PP. 820–828. DOI: 10.1007/978-3-030-21803-4_82

Статья поступила 20 ноября 2023 г.
Одобрена после рецензирования 30 ноября 2023 г.
Принята к публикации 11 декабря 2023 г.

Информация об авторах

Мартынов Владислав Александрович – соискатель ученой степени кандидата технических наук, тренер Центра олимпиадной подготовки по программированию, Национальный исследовательский Мордовский государственный университет им. Н. П. Огарёва. E-mail: vladmart1996@gmail.com

Плотникова Наталья Павловна – кандидат технических наук, директор Центра олимпиадной подготовки по программированию, Национальный исследовательский Мордовский государственный университет им. Н. П. Огарёва. E-mail: linsierra@yandex.ru

Implementation of INSERT in PostgreSQL in a Segment Tree

V. Martynov ✉, N. Plotnikova

National Research Ogarev Mordovia State University,
Saransk, 430005, Republic of Mordovia, Russian Federation

***Purpose**: This article is a continuation of the article [1], which considered a method of optimizing aggregation queries on a continuous range of PostgreSQL databases rows using a segment tree [2–4]. The disadvantage of the considered approach was that the operation of inserting new values into the table took too much time, as it led to a complete rebuilding of the tree. **The purpose** of this work is to develop a method that allows to significantly speed up the insertion of new items into a table while maintaining up-to-date information in the segment tree. In this article, we propose a method that can significantly speed up the insertion of new items into the table while maintaining up-to-date information in the segment tree. At the same time the gain in aggregating queries is fully preserved. The method is based on the peculiarity of the segment tree, which allows to update elements in it for $O(log_2 N)$ asymptotic. Then we can reduce the problem of adding new values to updating existing elements, which can be solved more efficiently. For this purpose, additional neutral elements are allocated during the construction of the tree. When new values are added to the table, one of these additional elements is updated. When the free elements in the tree run out the tree is rebuilt, again allocating additional memory. **The result** of this work is the development of an algorithm for updating elements in the tree and its implementation in the form of PostgreSQL Extensions. The speed of the obtained solution has been measured. Conclusions on the results of work and plans for further optimization of operations are made.*

***Key words:** database, segment tree, PostgreSQL, insert, asymptotic*

Information about Authors

Vladislav Martynov – Candidate for a degree of Ph.D. of Engineering Sciences, Coach of the Center of Olympiad Training in Programming, National Research Ogarev Mordovia State University. E-mail: vladmart1996@gmail.com

Natalya Plotnikova – Ph.D. of Engineering Sciences, Director of the Programming Olympiad Training Center, National Research Ogarev Mordovia State University. E-mail: linsierra@yandex.ru

DOI: 10.31854/2307-1303-2023-11-2-1-7
URL: https://www.sut.ru/doci/nauka/1AEA/ITT/2023_2/1-7.pdf