Применение дерева отрезков в PostgreSQL

В.А. Мартынов, Н.П. Плотникова

Национальный исследовательский Мордовский государственный университет им. Н.П. Огарёва»

Аннотация: В статье рассматривается подход к решению задачи оптимизации скорости агрегирующих запросов к непрерывному диапазону строк таблицы базы данных PostgreSQL. Создан программный модуль на базе расширения PostgreSQL Extensions, обеспечивающий построение дерева отрезков для таблицы и запросов к нему. Достигнуто увеличение скорости запросов более чем в 80 раз для таблицы размером 100 миллионов записей по сравнению с существующими решениями.
Ключевые слова: PostgreSQL, дерево отрезков, запрос, агрегация, оптимизация, PosgreSQL Extensions, асимптотика, индекс.

Введение

Скорость запросов к базе данных часто является краеугольным камнем скорости работы информационной системы, обрабатывающей большой объем данных. Для различных видов задач и запросов разрабатываются свои способы оптимизации. Это может быть либо специальная архитектура базы данных [1,2], либо различные индексы [3] к таблицам.

Рассмотрим задачи, в которых требуется выполнять агрегирующие запросы к последовательным строкам таблицы. Например, задача построения отчетов на предприятии за определенный период или задача получения статистики по истории активности пользователей социальных сетей.

В данной статье предложен подход, который обеспечивает значительное ускорение выполнения подобных запросов.

1. Существующие решения

Рассмотрим, как происходят агрегирующие запросы в таких популярных СУБД, как PostgreSQL [4] и MySQL [5] в базовом варианте без специальных настроек:

1. перебираются все строки таблицы;

© Электронный научный журнал «Инженерный вестник Дона», 2007–2023

2. для каждой строки проверяется, подходит ли она под условия запроса;
3. подходящие строки агрегируются в результат.

В данном случае будут рассмотрены все строки, даже если запрос затрагивает небольшую их часть. Таким образом, если всего в таблице N строк, то асимптотика [6] запроса будет составлять $O(N)$.

Построение индекса [7] для поля, которое участвует в условии запроса, позволяет немного оптимизировать скорость. За счет этого можно перебирать только подходящие строки. В таком случае, если запрашивались строки с L по R, асимптотика запроса будет составлять $O(R-L)$. При этом сохраняется проблема, связанная с необходимостью перебора всех подходящих строк для получения результата.

Предложенный в настоящей статье подход состоит в том, чтобы построить некоторую структуру данных, которая за счет $O(N)$ дополнительной памяти позволит отвечать на запросы за $O(log_2 N)$ времени. Такой структурой является дерево отрезков [8].

2. Дерево отрезков

Дерево отрезков — это структура данных, которая позволяет за асимптотику $O(log_2 N)$ реализовать любые операции, определяемые на множестве, на котором данная операция ассоциативна.

Дерево отрезков представляет собой корневое дерево, листьями которого являются элементы исходного массива, а другие вершины имеют по два ребёнка. Каждой вершине в соответствие поставлен отрезок, являющийся объединением отрезков ее детей (если у вершины есть дети), либо отрезок, содержащий конкретный элемент массива (для листьев). Кроме того, для каждой вершины хранится значение некоторой ассоциативной функции на данном отрезке. Пример дерева отрезков для функции суммы изображен на рис.1.

© Электронный научный журнал «Инженерный вестник Дона», 2007–2023

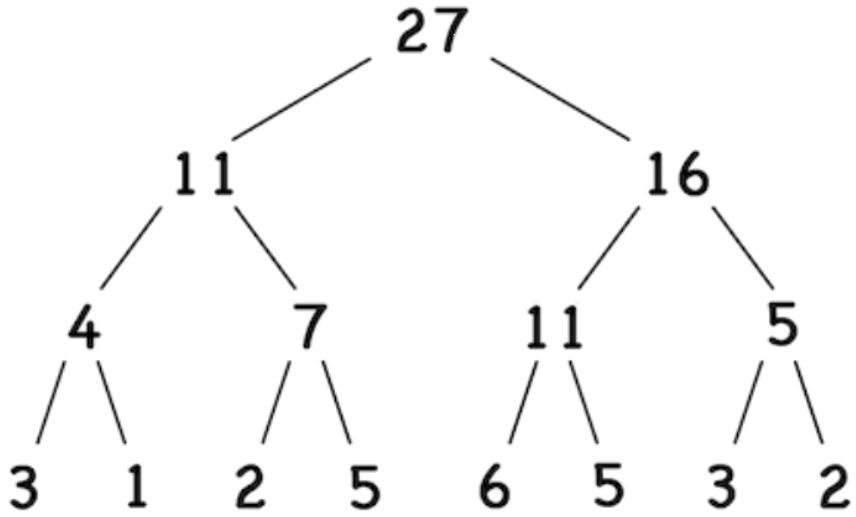

Рис. 1. – Дерево отрезков на сумму для массива [3, 1, 2, 5, 6, 5, 3, 2]

Данное дерево имеет высоту $O(log_2 N)$ и потребляет не более $2 \cdot N$ памяти. Утверждается, что можно найти значение функции на любом отрезке массива посетив не более $2 \cdot O(log_2 N)$ вершин дерева [9].

3. Реализация дерева отрезков в PostgreSQL

Рассмотрим применение подобной логики к базе данных PostgreSQL. Пусть в базе есть таблица вида, изображенного на рис.2.

Id	Sorted Field	Agg Field
1	01.01.2001	3
2	17.02.2001	1
3	05.04.2009	2
4	22.05.2020	5

Рис. 2. – Пример таблицы

Здесь «Sorted Field» — отсортированное поле, на которое будут накладываться условия запросов, а «Agg Field» — поле, значения которого мы хотим агрегировать.

© Электронный научный журнал «Инженерный вестник Дона», 2007–2023

Создадим еще одну таблицу, в которой будем хранить элементы дерева. Структура таблицы будет аналогична исходной, кроме лишних столбцов. В качестве индексов строк будем использовать значения поля «Id». Тогда в таблице нам понадобятся всего два поля: «Id» и «Agg Field».

Пронумеруем вершины дерева следующим образом: корень будет вершиной номер 1, левый ребенок вершины i будет иметь номер $2 \cdot i$, правый — $2 \cdot i + 1$. Тогда будем хранить значение вершины в строке таблицы с «Id», равным номеру вершины. На рис.3 изображена заполненная таблица для рассмотренного выше примера дерева.

Id	Agg Field
1	27
2	11
3	16
4	4
5	7
6	11
7	5
8	3

Id	Agg Field
9	1
10	2
11	5
12	6
13	5
14	3
15	2

Рис. 3. – Хранение дерева отрезков в таблице

Для реализации базовой версии дерева отрезков нам понадобятся две функции:

1. build, которая будет создавать и заполнять таблицу для дерева отрезков
2. get, которая будет обрабатывать запросы на агрегацию данных на отрезке.

Реализуем данные функции на языке SQL в виде PostgreSQL Extension [10].

© Электронный научный журнал «Инженерный вестник Дона», 2007–2023

Все эксперименты проводились на рабочей станции со следующими характеристиками:

- CPU — AMD Ryzen 5 3600
- RAM — 64Gb
- SSD — Force MP600 1Tb
- OS — Ubuntu 22.04.1
- PostgreSQL 14.8

4. Реализация функции build

Построение дерева будет проходить в два этапа:

1. создание таблицы под дерево отрезков размера $2 \cdot N$, где N — количество строк в исходной таблице;
2. рекурсивное заполнение таблицы.

Заполнение происходит следующим образом:

- запускаем рекурсивный обход из корня дерева;
- если текущая вершина лист, копируем из исходной таблицы значение соответствующего элемента;
- для остальных вершин запускаем рекурсивный обход левого и правого ребенка, агрегируем их результаты.

Для работы со значениями в таблице будем использовать стандартные запросы «SELECT» и «UPDATE». В таблице №1 приведены замеры времени построения дерева отрезков для различных размеров исходной таблицы.

Таблица №1

Время построения дерева отрезков

Размер таблицы, строк	Время построения, секунд
1'000	0,13

© Электронный научный журнал «Инженерный вестник Дона», 2007–2023

5'000	0,64
100'000	13,24
1'000'000	139,32
10'000'000	1439,3

5. Реализация функции get

На вход нам поступают два значения поля «Sorted Field», между которыми нужно вычислить значение. Запрос можно разделить на два этапа:

1. вычисление самого первого и последнего «Id» строк в исходной таблице, попадающих в нужный диапазон. Это делается двумя запросами «SELECT» к исходной таблице;
2. рекурсивный спуск по дереву и получение результата.

Для рекурсивного спуска нам необходимо поддерживать границы отрезка таблицы, соответствующего текущей вершине. Изначально запускается обход из корня дерева. Далее есть три случая:

1. отрезок вершины полностью входит в отрезок запроса — добавляем значение, записанное в вершине, к ответу и останавливаем обход;
2. отрезок вершины не пересекается с отрезком запроса — останавливаем обход;
3. отрезок вершины частично пересекается с отрезком запроса — рекурсивно спускаемся в обоих детей.

В таблице №2 приведены замеры времени 1000 запросов суммы на случайном отрезке для различных размеров исходной таблицы.

Таблица №2

Время работы get для 1000 запросов

Размер таблицы, строк	Время для 1000 запросов, секунд
1'000	2,1

© Электронный научный журнал «Инженерный вестник Дона», 2007–2023

5'000	2,12
100'000	2,76
1'000'000	4,12
10'000'000	5,03
100'000'000	14,29

6. Сравнение скорости

Сравним скорость реализованной функции «get» и обычного запроса на сумму к таблице. Для обычного запроса обеспечим оптимальную производительность, путем создания индекса для столбца «Sorted Field». Произведены замеры для 1000 случайных запросов и различных размеров таблиц. Результаты приведены в таблице №3.

Таблица №3

Сравнение времени агрегирующих запросов

Размер таблицы, строк	Время реализованной функции, секунд	Время обычного запроса, секунд
1'000	2,1	0,58
5'000	2,12	0,31
100'000	2,76	4,0
1'000'000	4,12	36,24
10'000'000	5,03	164,67
100'000'000	14,29	1245,04

В целях наглядной демонстрации прироста скорости, построим график отношения среднего времени запроса реализованной функции ко времени обычного запроса. Результаты представлены на рис.4.

© Электронный научный журнал «Инженерный вестник Дона», 2007–2023

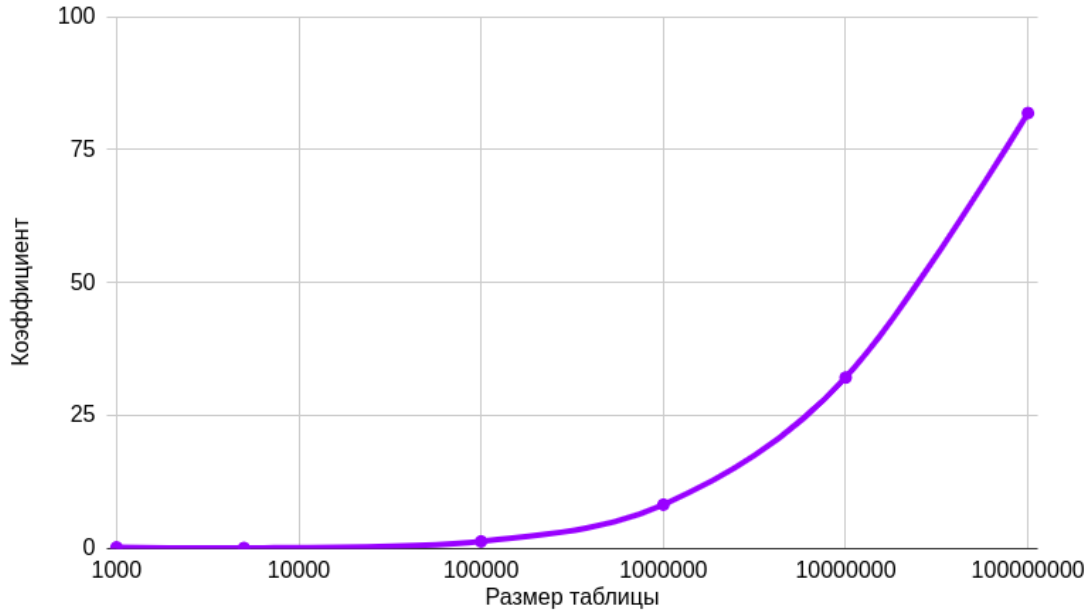

Рис. 4. – Коэффициент прироста скорости

Заключение

В рамках проведенного исследования разработан подход к решению задачи оптимизации скорости агрегирующих запросов к непрерывному диапазону строк таблицы базы данных PostgreSQL.

Создан программный модуль на базе расширения PostgreSQL Extensions обеспечивающий построение дерева отрезков для таблицы и запросов к нему.

Как видно из графика на рис.4, прирост скорости на таблице из 100'000'000 строк примерно 80-кратный. При этом, с ростом размеров таблицы прирост также увеличивается. Это достигается за счет разной асимптотики запросов.

В дальнейшем планируется реализовать программный модуль, поддерживающий добавление новых записей в исходную таблицу без перестроения всего дерева отрезков, а также оптимизировать текущие функции путем применения массовых запросов к базе данных.

© Электронный научный журнал «Инженерный вестник Дона», 2007–2023

Литература

1. Алгазали С.М.М, Айвазов В.Г., Кузнецова А.В. Совершенствование процесса поиска неэффективных SQL-запросов в СУБД Oracle // Инженерный вестник Дона, 2017, №4. URL: ivdon.ru/ru/magazine/archive/n4y2017/4511.
2. Земцов А.Н., Болгов Н.В., Божко С.Н. Многокритериальный выбор оптимальной системы управления базы данных с помощью метода анализа иерархий // Инженерный вестник Дона, 2014, №2. URL: ivdon.ru/magazine/archive/n1y2009/250/.
3. Roh H., Park S., Kim S. B+-tree Index Optimization by Exploiting Internal Parallelism of Flash-based Solid State Drives. URL: arxiv.org/abs/1201.0227 (дата обращения: 26.08.23).
4. Ahmed I., Smith G., Pirozzi E. PostgreSQL 10 High Performance: Expert techniques for query optimization, high availability, and efficient database maintenance. Pact Publishing, 2018, 508 p.
5. Лакхатария Д., Мехта Ч., Патель К., Чаллавала Ш. MySQL 8 для больших данных. ДМК Пресс, 2018. 226 с.
6. Bruijn N. G. Asymptotic Methods in Analysis. Courier Corporation, 1981, 200 p.
7. Maymala J. PostgreSQL for Data Architects. 2015, 272 p.
8. Laaksonen A. Guide to Competitive Programming: Learning and Improving Algorithms Through Contests (Undergraduate Topics in Computer Science). Springer, 2018, Pp. 125-128.
9. Brass P. Advanced Data Structures. Cambridge University Press, 2008, 154 p.
10. PostgreSQL Extensions // PostgreSQL. URL: wiki.postgresql.org/wiki/Extensions (дата обращения: 11.08.2023).

© Электронный научный журнал «Инженерный вестник Дона», 2007–2023

References

1. Algazali S.M.M, Ayvazov V.G., Kuznetsova A.V. Inzhenernyj vestnik Dona, 2017, №4. URL: ivdon.ru/ru/magazine/archive/n4y2017/4511.
2. Zemtsov A.N., Bolgov N.V., Bozhko S.N. Inzhenernyj vestnik Dona, 2014, №2. URL: ivdon.ru/magazine/archive/n1y2009/250/.
3. Roh H., Park S., Kim S. B+-tree Index Optimization by Exploiting Internal Parallelism of Flash-based Solid State Drives. arxiv.org URL: arxiv.org/abs/1201.0227 (date of reference: 26.08.23).
4. Ahmed I., Smith G., Pirozzi E. PostgreSQL 10 High Performance: Expert techniques for query optimization, high availability, and efficient database maintenance. Pact Publishing, 2018, 508 p.
5. Lakkhatariya D., Mekhta Ch., Patel' K., Challavala Sh. MySQL 8 dlya bol'shikh dannykh [MySQL 8 for big data]. DMK Press, 2018, 226 p.
6. Bruijn N. G. Asymptotic Methods in Analysis. Courier Corporation, 1981, 200 p.
7. J. Maymala PostgreSQL for Data Architects. 2015, 272 p.
8. A. Laaksonen Guide to Competitive Programming: Learning and Improving Algorithms Through Contests (Undergraduate Topics in Computer Science). Springer, 2018. pp. 125-128.
9. Brass P. Advanced Data Structures. Cambridge University Press, 2008. 154 p.
10. PostgreSQL Extensions. PostgreSQL. URL: wiki.postgresql.org/wiki/Extensions (date of reference: 11.08.2023).

© Электронный научный журнал «Инженерный вестник Дона», 2007–2023