Д. И. Солодков, З. Р. Цупа, В. Д. Савицкий

АНАЛИЗ ЭФФЕКТИВНОСТИ СТРАТЕГИЙ ИНДЕКСИРОВАНИЯ ВРЕМЕННЫХ РЯДОВ В POSTGRESQL

Проведен сравнительный анализ эффективности механизмов индексирования в СУБД PostgreSQL при работе с временными рядами. На базе эмпирического исследования датасета объемом 5 млн записей проведено сравнение стратегий B-Tree и BRIN. Исследованны возможности оптимизации архитектуры хранилищ для систем мониторинга и IoT.

Введение

Экспоненциальный рост объемов временных рядов (Time-Series) в системах мониторинга и IoT ставит задачу минимизации затрат на хранение при сохранении высокой скорости записи. Традиционные индексы B-Tree зачастую становятся «узким местом» из-за деградации производительности вставки при росте дерева. Целью работы является эмпирическое обоснование выбора стратегии индексирования для временных рядов в зависимости от физического порядка данных и типа нагрузки.

I. Классификация индексов в PostgreSQl

Индексы – это вспомогательные структуры данных, которые СУБД использует для оптимизации поиска строк, чтобы не проверять базу данных каждый раз. В PostgreSQL предусмотрено 6 основных типов индексов, каждый из которых построен на уникальной структуре данных и предназначен для специфических задач. Понимание их различий критично для оптимизации высоконагруженных систем [1]. Результаты сравнения индексов представлены в таблице 1.

Таблица 1 – Сравнение индексов PostgreSQL

Тип	Основное назначение	Поиск	Размер
B-Tree	Универсал, сортировка	$O(\log N)$	Высокий
BRIN	Огромные упорядоченные данные	Блочный	Минимал.
GIN	Текстовый поиск, JSONB	Ключи	Средний
GiST	Геометрия, интервалы, R-tree	$O(\log N)$	Высокий
SP-GiST	Не сбаланс. данные	Префикс.	Средний
Hash	Точное совпадение (=)	$O(1)$	Средний

Индекс B-Tree обеспечивает высокую селективность и детерминированную сложность поиска O(logN). Он выбран в качестве контрольной группы для оценки максимально достижимой производительности чтения, несмотря на значительные накладные расходы на хранение и запись.

Индекс BRIN (Block Range Index) выбран как наиболее перспективное решение для сверхбольших временных рядов. В отличие от древовидных структур, он оперирует метаданными о диапазонах блоков, что минимизирует влияние на подсистему ввода-вывода при вставке и сокращает объем индекса на три порядка.

II. Архитектура и математическая модель индексации BRIN

В СУБД PostgreSQL индекс BRIN (Block Range Index) функционирует на основе агрегации метаданных о физических диапазонах страниц таблицы. Данный подход позволяет радикально сократить объем метаданных, необходимых для индексирования сверхбольших таблиц (VLDB). Основным преимуществом такой архитектуры является минимизация нагрузки на подсистему ввода-вывода в процессе актуализации индекса. Эффективность индексации можно выразить через коэффициент ускорения записи (A_w):

$$A_w = T_{base}/T_{idx} \tag{1}$$

где T_{base} – время выполнения операции вставки без индекса; T_{idx} – время вставки с использованием индекса.

Высокая эффективность A_w при использовании BRIN достигается за счет записи в индекс только в случае, если новое вставляемое значение выходит за границы текущего диапазона [Min; Max] последнего блока страниц. В противном случае структура индекса остается неизменной.

III. Архитектура и функциональные особенности индексации B-Tree

Классическая стратегия индексирования в реляционных СУБД базируется на использовании структуры B-Tree (сбалансированного дерева). В отличие от блоковых методов, данный механизм реализует принцип прямой адресации, где каждый листовой узел дерева содержит точный указатель (TID) на конкретный кортеж в физической памяти [2].

Иерархическая природа B-Tree обеспечивает детерминированную сложность поиска $O(\log N)$, что делает его эффективным для высокоселективных запросов, требующих извлечения единичных записей или узких диапазонов данных. Однако поддержание сбалансированности дерева при интенсивном входном потоке (сценарий Write-Heavy) сопряжено с существенными накладными расходами. Интенсивность влияния индекса на производительность системы можно оценить через коэффициент избыточности хранения R_{storage}:

$$R_{storage} = S_{index}/S_{table} \tag{2}$$

где S_{index} – объём дискового пространства, занимаемый структурой B-Tree; S_{table} – физический размер индексируемой таблицы.

IV. Методология и экспериментальная база исследования

Экспериментальная часть реализована в среде PostgreSQL на базе таблицы sensor_data, имитирующей поток промышленной телеметрии объёмом $5 \cdot 10^6$ записей. Временные метки created_at генерировались функцией random() в интервале 90 суток для моделирования данных с высокой энтропией. Тестирование охватывало четыре сценария:

– Проектирование базовой таблицы *sensor_data* с генерацией $5 \cdot 10^6$ кортежей, где временные метки *created_at* распределены случайным образом с использованием функции *random()* в интервале 90 суток;

– проведение серии из 20 итераций запроса агрегации (*count*) с фильтрацией по диапазону *BETWEEN* в анонимном блоке *DO* для получения усредненного времени выполнения (*Execution Time*) в отсутствие индексов;

– построение структуры *B-Tree* и измерение её объема с помощью функции *pg_size_pretty*, зафиксировавшее избыточность в 107 *MB*;

– добавление инкрементальной нагрузки (5000 новых строк) с последующим профилированием плана выполнения запроса через *EXPLAIN ANALYZE*;

– имплементация блокового индекса *BRIN*(*idx_brin_time*) объемом 32 *kB* и фиксация отказа оптимизатора от его использования в пользу *Sequential Scan* из-за отсутствия физической корреляции записей на диске;

– создание специализированной структуры *sensor_data_ordered* с детерминированным последовательным заполнением для имитации реального хронологического потока данных;

– сравнительный анализ эффективности *BRIN* на упорядоченном наборе данных, подтвердивший переход от полного сканирования к целевой выборке диапазонов блоков.

Измерения производительности выполнялись при естественном управлении кэшем PostgreSQL без принудительного сброса буферов, что моделирует реальные условия эксплуатации.

V. Вывод

В результате эмпирического исследования на датасете объёмом 5 млн записей установлено, что индекс BRIN обеспечивает экономию дискового пространства в 3300 раз по сравнению с B-Tree (32 КБ против 107 МБ), однако его эффективность критически зависит от физического порядка данных. При хронологической записи оптимизатор применяет Bitmap Heap Scan, тогда как на хаотичных данных происходит деградация до Sequential Scan из-за отсутствия корреляции значений на диске. B-Tree демонстрирует стабильное ускорение операций чтения в 26 раз независимо от порядка записей, а BRIN ускоряет операции вставки в 2,9 раза за счёт минимальных накладных расходов на обновление метаданных. Полученные данные подтверждают целесообразность применения BRIN исключительно в сценариях с последовательным поступлением временных рядов, тогда как для произвольных выборок или неупорядоченных данных предпочтителен B-Tree. Практическая значимость работы заключается в формировании обоснованной методологии выбора стратегии индексирования для высоконагруженных систем мониторинга и IoT-платформ, позволяющей оптимизировать совокупную стоимость владения за счёт рационального баланса между требованиями к дисковому пространству, скоростью чтения и записи.

Список литературы

1. PostgreSQL Global Development Group. (2024). BRIN Indexes. In *PostgreSQL 16 Documentation*. URL: https://www.postgresql.org/docs/16/brin.html.
2. Stonebraker, M., & Hellerstein, J. M. (2005). What Goes Around Comes Around. *Readings in Database Systems* (pp. 2-41).

Солодков Д. И., студент кафедры информационных технологий автоматизированных систем БГУИР, solodkov.dm@gmail.com.

Цупа З. Р., студент кафедры информационных технологий автоматизированных систем БГУИР, zlatushh0@gmail.com.

Савицкий В. Д., студент кафедры информационных технологий автоматизированных систем БГУИР, vadim29112005@gmail.com.

Научный руководитель: Трофимович А. Ф., старший преподаватель кафедры информационных технологий автоматизированных систем БГУИР, trofimaf@bsuir.by.