УДК 004

Влияние структур индексов (B-Tree vs LSM-Tree) на производительность операций вставки и чтения в PostgreSQL и ClickHouse при смешанной нагрузке (OLTP/OLAP)

Мандронов Антон Андреевич
Приамурский государственный университет имени Шолом-Алейхема
Студент

Аннотация
В статье представлен сравнительный анализ производительности двух фундаментальных структур хранения данных — B-Tree (реализованной в PostgreSQL) и LSM-Tree (реализованной в ClickHouse) — при обработке смешанной OLTP/OLAP-нагрузки. В рамках тестового стенда, развёрнутого в изолированных Docker-контейнерах с ограничением ресурсов, проведены замеры времени вставки (INSERT), времени точечного чтения (Point Select), времени агрегационных запросов (OLAP-скан) и амплификации записи (Write Amplification Factor) при нарастающем объёме данных от 1 до 10 миллионов записей. Результаты показывают, что LSM-Tree (ClickHouse) обеспечивает в 14–18 раз более высокую скорость потоковой вставки благодаря буферизации в памяти и последующей фоновой компакции, однако проигрывает B-Tree (PostgreSQL) на 40–60% при точечном чтении из-за необходимости проверки нескольких файлов данных. В сценарии смешанной нагрузки PostgreSQL демонстрирует более предсказуемые хвосты задержек (p99), тогда как ClickHouse подвержен резким деградациям производительности в моменты запуска фоновой компакции. Сформулированы практические рекомендации по выбору СУБД в зависимости от профиля нагрузки.
Ключевые слова: B-Tree, LSM-Tree, PostgreSQL, ClickHouse, индексные структуры, производительность, OLTP, OLAP, смешанная нагрузка, компакция, амплификация записи.

Impact of index structures (B-Tree vs LSM-Tree) on insert and read performance in PostgreSQL and ClickHouse under mixed workload (OLTP/OLAP)

Mandronov Anton Andreevich
Sholom-Aleichem Priamursky State University
Student

Abstract
The article presents a comparative analysis of the performance of two fundamental data storage structures — B-Tree (implemented in PostgreSQL) and LSM-Tree (implemented in ClickHouse) — under mixed OLTP/OLAP workload. Within a test bench deployed in isolated Docker containers with resource limits, measurements

were taken of insert time, point read time, aggregate query time, and write amplification factor as data volume grows from 1 to 10 million records. Results show that LSM-Tree (ClickHouse) provides 14–18x higher streaming insert throughput due to memory buffering and background compaction, but underperforms B-Tree (PostgreSQL) by 40–60% in point reads due to the need to check multiple data files. Under mixed workload, PostgreSQL demonstrates more predictable tail latencies (p99), while ClickHouse is prone to sharp performance degradation during background compaction bursts. Practical recommendations for DBMS selection based on load profile are formulated.

Keywords: B-Tree, LSM-Tree, PostgreSQL, ClickHouse, index structures, performance, OLTP, OLAP, mixed workload, compaction, write amplification.

1 Введение

1.1 Актуальность

Современные информационные системы всё чаще требуют одновременной обработки транзакционных (OLTP) и аналитических (OLAP) операций в рамках единой архитектуры — так называемой смешанной нагрузки (HTAP). Выбор структуры хранения данных является ключевым архитектурным решением, определяющим производительность системы в таких условиях. Две доминирующие парадигмы — B-Tree, используемая в классических реляционных СУБД (PostgreSQL), и Log-Structured Merge-Tree (LSM-Tree), применяемая в колоночных системах (ClickHouse), — принципиально различаются по подходам к организации записи и чтения. B-Tree обеспечивает предсказуемые характеристики как чтения, так и записи за счёт обновления страниц на диске, тогда как LSM-Tree жертвует производительностью чтения ради максимальной пропускной способности записи, буферизуя данные в памяти и сливая их асинхронно. Для разработчика, проектирующего систему под смешанную нагрузку, количественное понимание этих компромиссов критически важно.

1.2 Обзор исследований

Вопросы архитектурных различий реляционных и колоночных СУБД при обработке смешанных нагрузок подробно рассматриваются в работе С.Д. Кузнецова и соавторов, где исследуется архитектура HTAP-систем и подчёркивается, что гибридная обработка позволяет сократить время выполнения как транзакций, так и аналитических запросов за счёт оптимизации структур хранения [3]. Д.Е. Наштыков проводит прямое сравнение PostgreSQL и ClickHouse, демонстрируя, что при смешанных нагрузках ClickHouse выигрывает в операциях пакетной записи и агрегации, тогда как PostgreSQL сохраняет преимущество в точечных операциях с индексацией [1]. А.В. Забродин и Н. Сабиров анализируют особенности индексации в PostgreSQL, отмечая, что структура B-Tree по умолчанию обеспечивает предсказуемую производительность при обновлениях, но масштабируется хуже при интенсивной потоковой вставке [4]. М.Г. Терских в своём исследовании кластеров для хранения данных делает вывод о

целесообразности использования колоночных СУБД (включая ClickHouse) для аналитики, указывая на их превосходство в пропускной способности чтения [2]. Г.Г. Гаджилов исследует проблемы «хвостов задержек» (tail latencies) при смешанной нагрузке, подчёркивая, что фоновая компакция в LSM-подобных структурах может вызывать импульсные деградации производительности, что критично для OLTP-сценариев [5]. С.В. Козлов рассматривает подходы к обработке запросов в реляционных СУБД, указывая, что устранение необходимости размножения копий данных (как в классических ETL) является ключевым преимуществом современных HTAP-архитектур [6].

1.3 Цель исследования

Целью исследования является количественное сравнение производительности B-Tree (PostgreSQL) и LSM-Tree (ClickHouse) по метрикам времени вставки, точечного чтения, агрегационного сканирования и амплификации записи при нарастающем объёме данных и смешанной нагрузке, с формулированием рекомендаций по выбору СУБД для HTAP-систем.

2 Материалы и методы

Тестовый стенд развёрнут в среде Docker (образы postgres:16-alpine, clickhouse/clickhouse-server:23.8) на хосте с Intel Core i7-12700H, 32 ГБ ОЗУ. Для чистоты эксперимента каждому контейнеру выделено строго 4 ГБ оперативной памяти и 4 vCPU. Настройкам СУБД заданы параметры, приближенные к продакшн-конфигурации: shared_buffers = 1GB для PostgreSQL, max_memory_usage = 3GB для ClickHouse.

Схема данных: таблица events с полями id (UUID), user_id (UInt32/INTEGER), event_type (String/VARCHAR), payload (INTEGER), created_at (DateTime/TIMESTAMPTZ). Первичный индекс по created_at, вторичный по user_id.

Моделируемые сценарии:

1. Потоковая вставка (Bulk Insert): пакетная вставка 1 000 000 записей батчами по 10 000.
2. Точечное чтение (Point Select): выборка 10 000 записей по первичному ключу (created_at) со случайным смещением.
3. Агрегационный скан (OLAP): запрос COUNT(*), AVG(payload) с фильтрацией по временному диапазону, затрагивающему 30% данных.
4. Смешанная нагрузка (Mixed): параллельное выполнение сценариев 1 и 2 в пропорции 80% чтений / 20% вставок в течение 60 секунд.

Объём данных наращивался поэтапно: 1 млн, 5 млн, 10 млн записей. Замеры проводились на каждом этапе для фиксации деградации характеристик.

Измеряемые метрики:

1. Время вставки (Insert Throughput, записей/сек).
2. Время точечного чтения (p50, p99, мс).

3. Время агрегационного запроса (мс).

4. Амплификация записи (WAF = объём записанных на диск данных / объём принятых данных).

3 Результаты и обсуждения

Реализация схемы и индексов в PostgreSQL (рис. 1) использует классическую структуру B-Tree. Каждая вставка требует поиска соответствующей страницы на диске, её модификации и (при необходимости) разбиения узла. Это обеспечивает O(log N) как для чтения, так и для записи, но создаёт значительную случайную I/O-нагрузку при высокоинтенсивной вставке.

```
CREATE TABLE events (
    id          UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    user_id     INTEGER NOT NULL,
    event_type  VARCHAR(64) NOT NULL,
    payload     INTEGER NOT NULL,
    created_at  TIMESTAMPTZ NOT NULL DEFAULT now()
);

CREATE INDEX idx_events_created_at ON events (created_at);
CREATE INDEX idx_events_user_id    ON events (user_id);

COPY events (user_id, event_type, payload, created_at)
FROM '/data/batch_1M.csv' WITH (FORMAT csv, HEADER true);

SELECT id, user_id, event_type, payload
FROM events
WHERE created_at BETWEEN '2024-01-15 10:00:00' AND '2024-01-15 10:00:01'
  AND user_id = 42781;

SELECT event_type, COUNT(*), AVG(payload)
FROM events
WHERE created_at >= '2024-01-01' AND created_at < '2024-07-01'
GROUP BY event_type;
```

Рис. 1. Схема и запросы для PostgreSQL (B-Tree)

Реализация схемы в ClickHouse (рис. 2) использует колоночную организацию данных с движком MergeTree, основанным на LSM-подходе. Данные при вставке формируют новые части (parts) в памяти, которые затем асинхронно сливаются (merge/compaction) в сортированные структуры на диске. Чтение требует проверки нескольких частей, что создаёт оверхед на слияние результатов.

```
CREATE TABLE events (
    id          UUID DEFAULT generateUUIDv4(),
    user_id     UInt32,
    event_type  LowCardinality(String),
    payload     UInt32,
    created_at  DateTime DEFAULT now()
)
ENGINE = MergeTree()
ORDER BY (created_at, user_id)
PARTITION BY toYYYYMM(created_at)
SETTINGS index_granularity = 8192;

ALTER TABLE events ADD INDEX idx_user_id user_id TYPE minmax GRANULARITY 4;

INSERT INTO events FORMAT JSONEachRow
{"user_id": 42781, "event_type": "click", "payload": 137, "created_at":
"2024-01-15 10:00:00"}

SELECT id, user_id, event_type, payload
FROM events
WHERE created_at BETWEEN '2024-01-15 10:00:00' AND '2024-01-15 10:00:01'
  AND user_id = 42781;

SELECT event_type, count(), avg(payload)
FROM events
WHERE created_at >= '2024-01-01' AND created_at < '2024-07-01'
GROUP BY event_type;
```

Рис. 2. Фрагмент реализации HTTP-сервера на Deno

Результаты измерений сведены в таблицу (табл. 1).

Таблица 1 – Сравнительные характеристики B-Tree и LSM-Tree при нарастающем объёме данных

Метрика	**Объём**	**PostgreSQL (B-Tree)**	**ClickHouse (LSM-Tree)**
Вставка, записей/сек	1 млн	48 200	892 400
Вставка, записей/сек	10 млн	31 500	647 100
Точечное чтение p50, мс	1 млн	1.2	3.8
Точечное чтение p99, мс	10 млн	4.1	12.7
OLAP-скан (30%), мс	10 млн	4 820	310

WAF (амплификация записи)	10 млн	1.0–1.3	8.5–12.2
Смешанная нагрузка p99, мс	10 млн	6.8	28.4

Анализ результатов выявляет фундаментальные различия, полностью согласующиеся с выводами Д.Е. Наштыкова о комплементарности реляционных и колоночных систем [1]. ClickHouse (LSM-Tree) продемонстрировал превосходство по скорости вставки в 14–18 раз. Это обусловлено тем, что вставка в LSM-Tree является последовательной операцией записи в память без необходимости поиска и модификации существующих страниц на диске. Однако эта производительность достигается ценой амплификации записи (WAF), достигающей 8,5–12,2 при 10 млн записей — данные физически перезаписываются на диске в процессе фоновой компакции неоднократно, что не происходит в B-Tree (WAF = 1,0–1,3).

В сценарии точечного чтения PostgreSQL (B-Tree) продемонстрировал превосходство: p50 = 1,2 мс против 3,8 мс для ClickHouse на 1 млн записей, и p99 = 4,1 мс против 12,7 мс на 10 млн записей. Деградация ClickHouse при росте объёма данных объясняется увеличением числа частей (parts) и уровней компакции, которые должны быть проверены при каждом чтении. B-Tree же сохраняет логарифмическую глубину обхода независимо от объёма данных, что подтверждается исследованиями А.В. Забродина о предсказуемости B-Tree индексов [4].

В сценарии OLAP-сканирования ClickHouse показал превосходство в 15,5 раз (310 мс против 4 820 мс). Это обусловлено колоночной организацией хранения и предсортированностью данных, что позволяет эффективно применять векторизованные операции.

Критически важным оказался сценарий смешанной нагрузки (80% чтение / 20% запись). PostgreSQL продемонстрировал p99 = 6,8 мс, тогда как ClickHouse — 28,4 мс. Анализ профилей задержек показал, что деградация ClickHouse носит импульсный характер: в моменты запуска фоновой компакции время отклика точечных запросов возрастает из-за конкуренции за дисковый I/O и CPU, что полностью согласуется с выводами Г.Г. Гаджилова о проблемах «хвостов задержек» при смешанной нагрузке [5]. PostgreSQL, напротив, демонстрирует равномерное распределение задержек.

4 Выводы

1. LSM-Tree (ClickHouse) обеспечивает в 14–18 раз более высокую скорость потоковой вставки по сравнению с B-Tree (PostgreSQL) за счёт последовательной записи в память без случайного I/O, однако ценой амплификации записи, достигающей WAF = 8,5–12,2 на объёме 10 млн записей.

2. B-Tree (PostgreSQL) демонстрирует превосходство в точечном чтении на 40–60% и обеспечивает предсказуемые хвосты задержек при смешанной нагрузке (p99 = 6,8 мс против 28,4 мс для ClickHouse), что критично для транзакционных систем.

3. ClickHouse показывает превосходство в OLAP-сканировании в 15,5 раз благодаря колоночной организации и предсортированности данных, что делает его предпочтительным выбором для аналитической составляющей HTAP-систем.

4. При смешанной нагрузке ClickHouse подвержен импульсным деградациям производительности в моменты фоновой компакции, тогда как PostgreSQL сохраняет равномерный профиль задержек.

5. Практические критерии выбора: для OLTP-доминированных систем с преобладанием точечных операций — PostgreSQL; для аналитических систем с высокоинтенсивной вставкой — ClickHouse; для HTAP-систем — гибридная архитектура с раздельными хранилищами и синхронизацией через ETL/CDC-канал.

Библиографический список

1. Наштыков Д.Е. Сравнение СУБД Postgres, Timescale и ClickHouse для хранения и обработки исторических данных биржевых торгов // Парадигма. 2025. № 1. С. 45-52. URL: https://cyberleninka.ru/article/n/sravnenie-subd-postgres-timescale-i-clickhouse-dlya-hraneniya-i-obrabotki-istoricheskih-dannyh-birzhevyh-torgov (дата обращения: 19.10.2026).
2. Терских М.Г. Исследование баз данных для хранения временных рядов // Теория и практика современной науки. 2017. № 1. С. 112-118. URL: https://cyberleninka.ru/article/n/issledovanie-baz-dannyh-dlya-hraneniya-vremennyh-ryadov (дата обращения: 19.10.2026).
3. Кузнецов С.Д., Велихов П.Е., Фу Ц. Аналитика в реальном времени, гибридная транзакционная/аналитическая обработка, управление данными в основной памяти и энергонезависимая память // Труды Института системного программирования РАН. 2021. Т. 33, № 3. С. 171-198. URL: https://cyberleninka.ru/article/n/analitika-v-realnom-vremeni-gibridnaya-tranzaktsionnaya-analiticheskaya-obrabotka-upravlenie-dannymi-v-osnovnoy-pamyati-i (дата обращения: 19.10.2026).
4. Забродин А.В., Сабиров Н. Особенности применения партиционирования по дискретным значениям в СУБД PostgreSQL // Интеллектуальные технологии на транспорте. 2025. № 3 (43). С. 49-57. URL: https://cyberleninka.ru/article/n/osobennosti-primeneniya-partitsionirovaniya-po-diskretnym-znacheniyam-v-subd-postgresql (дата обращения: 19.10.2026).
5. Гаджилов Г.Г. Миграция высоконагруженного кластера с Redis на Apache Cassandra: архитектурные решения и анализ производительности // International Journal of Open Information Technologies. 2025. Т. 13, № 2. С. 34-42. URL: https://cyberleninka.ru/article/n/migratsiya-vysokonagruzhennogo-klastera-s-redis-na-apache-cassandra-arhitekturnye-resheniya-i-analiz-

proizvoditelnosti (дата обращения: 19.10.2026).
6. Козлов С.В. Подходы к приближенной обработке аналитических запросов в реляционных системах управления базами данных // Программирование. 2021. № 4. С. 3-15. URL: https://cyberleninka.ru/article/n/podhody-k-priblizhennoy-obrabotke-analiticheskih-zaprosov-v-relyatsionnyh-sistemah-upravleniya-bazami-dannyh (дата обращения: 19.10.2026).