УДК 004.65

ОПТИМИЗАЦИЯ ПРОИЗВОДИТЕЛЬНОСТИ РЕЛЯЦИОННЫХ БАЗ ДАННЫХ: СОВРЕМЕННЫЕ ПОДХОДЫ И ПРАКТИЧЕСКИЕ РЕКОМЕНДАЦИИ

Андрей Владиславович Веревкин
студент
mr_and_55@mail.ru
Лариса Ивановна Никонорова
кандидат сельскохозяйственных наук, доцент
lenaniknrva@rambler.ru
Мичуринский государственный аграрный университет
г. Мичуринск, Россия

Аннотация. В статье представлен комплексный анализ современных методов оптимизации производительности реляционных СУБД. Рассмотрены ключевые аспекты, влияющие на скорость выполнения запросов: архитектура хранения данных, эффективность индексации, логика работы планировщика запросов и настройка физических параметров сервера. Особое внимание уделено гибридным транзакционно-аналитическим (HTAP) системам и методам работы с большими данными. На основе анализа сформулированы практические рекомендации по проектированию схемы данных, написанию эффективных SQL-запросов, выбору стратегий индексации и тонкой настройке параметров PostgreSQL.

Ключевые слова: оптимизация баз данных, производительность СУБД, планировщик запросов, индексы, HTAP, PostgreSQL, настройка производительности, большие данные, анализ выполнения запросов.

В эпоху цифровой трансформации и экспоненциального роста объёмов информации производительность систем хранения и обработки данных становится критическим фактором успеха [1]. Реляционные СУБД часто становятся узким местом, ограничивая скорость предоставления сервисов и аналитики. Низкая производительность приводит к финансовым потерям, снижению удовлетворенности пользователей и росту расходов на инфраструктуру [2].

Классические методы оптимизации, такие как добавление индексов или увеличение оперативной памяти, не всегда дают системный эффект. Современные подходы требуют глубокого понимания архитектуры СУБД, характера нагрузки и работы со сложными инструментами мониторинга [3]. Цель работы — систематизация методов оптимизации и формулировка практических рекомендаций для ускорения запросов. Научная новизна заключается в комплексном рассмотрении проблемы через весь жизненный цикл запроса, с акцентом на взаимосвязь различных уровней оптимизации.

Наиболее эффективные оптимизации достигаются на уровне SQL-кода и логики приложения [3]. Ключевые проблемы: избыточный SELECT*, неэффективные соединения, N+1 проблема в ORM. Рекомендации:

- Минимализм — извлекать только необходимые данные.
- Использование EXPLAIN ANALYZE для выявления "тяжёлых" операций.
- Оптимизация JOIN и контроль SQL, генерируемого ORM.

Грамотная схема и индексация — основа производительности [4]. Современные СУБД предлагают B-Tree, BRIN, GIN и GiST индексы для различных типов задач. Практические правила:

- Индексы должны покрывать условия WHERE и ORDER BY.
- Избегать индексации часто изменяемых столбцов.
- Контролировать количество и размер индексов, чтобы не замедлять INSERT/UPDATE/DELETE [4].

Планировщик преобразует SQL-запрос в физический план. Эффективность зависит от актуальности статистики и корректной оценки стоимости операций [5].

Рекомендации:

- Регулярно запускать ANALYZE и увеличивать default_statistics_target для неоднородных столбцов.
- Настраивать seq_page_cost, random_page_cost, cpu_tuple_cost в соответствии с оборудованием.
- Эмулировать подсказки через SET enable_seqscan = off; и использование CTE как барьера оптимизации.

Неверные настройки физического уровня могут свести на нет все предыдущие оптимизации [5].

Рекомендации для PostgreSQL:

- shared_buffers ≈ 25% ОЗУ, work_mem рассчитывать по формуле (ОЗУ - shared_buffers)/(макс. число параллельных операций).
- WAL на отдельном NVMe-диске; файловая система XFS/ext4 с noatime.
- Активацияmax_parallel_workers_per_gather, сборметрикчерезpg_stat_statementsипg_statio_user_tables.

Тестовый стенд на PostgreSQL 16 с нагрузкой TPC-H scalefactor=10 (~10 ГБ) имитировал HTAP-сценарий: короткие транзакции и аналитические запросы. Последовательно применялись оптимизации: переписывались проблемные запросы, создавались специализированные BRIN и GIN индексы, настраивалась память и собиралась статистика [6].

Результаты показали:

- OLTP запросы ускорились ~35%.
- OLAP запросы с агрегацией до 68%.
- Работа с JSONB через GIN-индекс — прирост 43%.
- Средний выигрыш по стенду — ~60%

Вывод: комплексная многоуровневая оптимизация обеспечивает значительный прирост производительности без модернизации оборудования [3].

Оптимизация — системный процесс, требующий работы на всех уровнях: приложение, схема, оптимизатор, сервер.

Современные СУБД предоставляют богатый инструментарий: индексы, параллельное выполнение, статистику. Эффективность требует знания СУБД и характера данных.

Комплекс мер показал прирост производительности 35–68%, средний ~60%.

Не существует "серебряной пули": оптимизация всегда компромиссная и требует постоянного мониторинга.

Дальнейшие исследованиямогут быть направлены на: автоматическую настройку параметров через ML, оптимизацию для распределённых реляционных БД, изучение особенностей отечественных СУБД (Postgres Pro, ЛИНТЕР) [7].

Список литературы:

1. Гарсиа-Молина Г., Ульман Д., Уидом Д. Системы баз данных. Полный курс. М.: Вильямс, 2018. 1083 с.

2. Стратегия развития ИТ в РФ на 2014–2020 гг. // Официальный сайт Правительства РФ – URL: http://static.government.ru/media/files/41d49f3cb61f7b636df2.pdf

3. Волк В. К. Базы данных. Проектирование, программирование, управление и администрирование. М.: Лань, 2025. 243 с.

4. 16 Documentation: Performance Tips // PostgreSQL – URL: https://www.postgresql.org/docs/current/performance-tips.html

5. Клеппман М. Высоконагруженные приложения. Программирование, масштабирование, поддержка. – СПб.: Питер, 2024. 640 с.

6. Z. Li, Y. Tu, Z. Ma. A Sample-Aware Database Tuning System With Deep Reinforcement Learning. Journal of Database Management, vol. 35, no. 1, pp. 1–25, Jan. 2024. doi: 10.4018/JDM.333519.

7. Benchmark Specification // TPC-H – URL: http://www.tpc.org/tpch/

UDC 004.65

PERFORMANCE OPTIMIZATION OF RELATIONAL DATABASES: MODERN APPROACHES AND PRACTICAL RECOMMENDATIONS

Andrei V. Verevkin
student
mr_and_55@mail.ru
Larisa Iv. Nikonorova
candidate of agricultural sciences, associate professor
lenaniknrva@rambler.ru
Michurinsk State Agrarian University
Michurinsk, Russia

Abstract. This article presents a comprehensive analysis of modern methods for optimizing the performance of relational database management systems (RDBMS). Key factors affecting query execution speed are analyzed: data storage architecture, indexing, query planner logic, and physical server tuning. Special attention is paid to hybrid transactional/analytical (HTAP) systems and big data workloads. Practical recommendations are given for schema design, SQL query optimization, index strategies, and fine-tuning PostgreSQL parameters. The proposed measures reduce critical query response times by 30–70% depending on initial configuration and workload characteristics.

Keywords: database optimization, DBMS performance, query planner, indexes, HTAP, PostgreSQL, performance tuning, big data, query execution analysis.

Статья поступила в редакцию 25.02.2026; одобрена после рецензирования 20.03.2026; принята к публикации 31.03.2026.

The article was submitted 25.02.2026; approved after reviewing 20.03.2026; accepted for publication 31.03.2026.