ORES
Научно-практический электронный журнал
Оригинальные исследования (ОРИС), *ISSN 2222-0402*
www.ores.su
original-research.ru

УДК 004.657

ВЫБОР ОПТИМАЛЬНОЙ АРХИТЕКТУРЫ ХРАНЕНИЯ И ОБРАБОТКИ ДАННЫХ ИЗ ФИНАНСОВЫХ ОТЧЁТОВ WILDBERRIES: ROW-STORE И COLUMN-STORE

Романовский Илья Олегович,
Студент группы ИУК5-11М
Калужский филиал Московского государственного технического университета имени Н.Э. Баумана
romanovskiyio@student.bmstu.ru

Ткаченко Анастасия Владимировна,
Старший преподаватель кафедры ИУК5
Калужский филиал Московского государственного технического университета имени Н.Э. Баумана
tkachenko_av@bmstu.ru

Серпинский Роман Эдуардович,
Студент группы ИУК5-11М
Калужский филиал Московского государственного технического университета имени Н.Э. Баумана
serpinskiyrea@student.bmst.ru

Аннотация

В работе проводится сравнительный анализ производительности строчно-ориентированной СУБД PostgreSQL и колоночной системы DuckDB при хранении и обработке «широких» финансовых отчётов маркетплейса Wildberries. Исследование охватывает типовые аналитические запросы (агрегации, группировки, оконные функции) и включает замер времени выполнения, нагрузки на CPU и RAM, а также эффективности дискового хранения. На основе Python-скрипта реализован автоматизированный прогон идентичных SQL-запросов и мониторинг вычислительных ресурсов, что позволяет объективно оценить преимущества и ограничения строчно- и колоночно-ориентированных архитектур в задачах бизнес-аналитики. Полученные результаты показывают существенное преимущество DuckDB по скорости обработки и компактности хранения по сравнению с PostgreSQL при работе с табличными данными отчётности Wildberries.

Ключевые слова: Wildberries; финансовые отчёты; PostgreSQL; DuckDB; row-store; column-store; аналитические запросы; производительность СУБД; бизнес-аналитика.

CHOOSING THE OPTIMAL ARCHITECTURE FOR STORING AND PROCESSING DATA FROM WILDBERRIES FINANCIAL REPORTS: ROW-STORE AND COLUMN-STORE

Romanovsky Ilya Olegovich,
Student of the IUK5-11M group
Kaluga Branch of the Bauman Moscow State Technical University
romanovskiyio@student.bmstu.ru

Tkachenko Anastasia Vladimirovna,
Senior Lecturer at the IUK5 Department
Kaluga Branch of the Bauman Moscow State Technical University
tkachenko_av@bmstu.ru

Serpinski Roman Eduardovich,
Student of the IUK5-11M group
Kaluga Branch of the Bauman Moscow State Technical University
serpinskiyrea@student.bmst.ru

ABSTRACT

This study presents a comparative performance analysis of the row-oriented DBMS PostgreSQL and the column-oriented system DuckDB for storing and processing "wide" financial reports from the Wildberries marketplace. The research focuses on typical analytical queries (aggregations, groupings, window functions) and measures query execution time, CPU and RAM utilization, as well as disk storage efficiency. An automated Python-based benchmarking pipeline executes identical SQL workloads and monitors computing resources, enabling an objective evaluation of row-store and column-store architectures for business analytics tasks. The experimental results demonstrate a significant advantage of DuckDB over PostgreSQL in terms of query performance and storage compactness when processing Wildberries reporting tables.

Keywords: Wildberries; financial reports; PostgreSQL; DuckDB; row-store; column-store; analytical queries; DBMS performance; business analytics.

Введение

В современных условиях электронная коммерция и маркетплейсы генерируют огромные объемы данных, которые требуют эффективных методов хранения и обработки. Wildberries, как один из крупнейших маркетплейсов в России, предоставляет бизнесу широкий ассортимент отчетной информации, необходимой для анализа продаж, возвратов, логистики и финансовых показателей. Такие отчеты часто представляют собой «широкие» таблицы с большим количеством колонок, что создает уникальные вызовы для традиционных систем управления базами данных.

Целью данной статьи является сравнительный анализ производительности двух основных архитектур систем хранения данных: традиционной строчно-ориентированной (row-store) базы данных PostgreSQL и современной колоночной (column-store) аналитической системы DuckDB. Анализ направлен на оценку эффективности обработки

широких таблиц отчетности Wildberries с точки зрения времени выполнения типовых аналитических запросов и оптимальности использования ресурсов.

В научной литературе детально показано, что колоночные СУБД способны на порядки превосходить классические строчно-ориентированные хранилища при аналитических нагрузках за счёт уменьшения объёма читаемых данных, более эффективного сжатия и оптимального использования кэш-памяти процессора. В работе Д. Абади и соавторов приведены экспериментальные результаты, демонстрирующие значительный выигрыш колоночных систем на типичных запросах хранилищ данных по сравнению с традиционными row-store архитектурами. [7]

Полученные выводы подтверждаются и более поздними исследованиями, в которых сравниваются производительность и ресурсоёмкость колоночного и строкового хранения на OLAP-нагрузках: колоночные решения показывают ускорение выполнения агрегирующих запросов в несколько раз за счёт чтения только нужных столбцов и агрессивного сжатия. В контексте маркетплейсов, где аналитические запросы часто обрабатывают «широкие» таблицы с десятками и сотнями атрибутов, такие преимущества делают колоночные архитектуры естественным кандидатом для реализации витрин данных и систем бизнес-аналитики. [8]

Результаты исследования позволят определить преимущества и ограничения каждой из технологий в контексте обработки данных маркетплейса и дадут рекомендации по выбору архитектуры для задач бизнес-аналитики в экосистеме Wildberries.

Проведённое исследование является этапом подготовки к внедрению программного решения для автоматизированной обработки отчетов Wildberries и направлено на выбор оптимальной архитектуры хранения данных для данного проекта

Критерии сравнения

Для объективной оценки эффективности различных систем хранения данных при обработке широких таблиц отчетности Wildberries используются следующие ключевые критерии:

Время выполнения типовых запросов. Замеряется среднее и медианное время ответа на операции агрегации, выборки по фильтру и группировки по основным полям.

Использование оперативной памяти и процессора во время выполнения запросов. Отслеживается нагрузка на ресурсы сервера во время выполнения стандартного сценария обработки.

Эффективность хранения данных. Сравнивается объем занимаемого места на диске после загрузки всего набора документов.

Выбор этих критериев обусловлен спецификой бизнес-аналитики маркетплейса, где приоритетом является быстрая обработка больших объемов табличных данных с возможностью гибкой обработки и низкими затратами на поддержку инфраструктуры.

Система PostgreSQL

PostgreSQL представляет собой свободно распространяемую объектно-реляционную систему управления базами данных с открытым исходным кодом, реализующую классическую строчно-ориентированную (row-store) модель хранения. Система использует клиент-серверную архитектуру: сервер PostgreSQL управляет данными и транзакциями, а приложения подключаются к нему по сети или локально для выполнения SQL-запросов.

Логически данные организованы в кластеры, базы данных, схемы и таблицы, а физически таблицы разбиваются на страницы фиксированного размера, внутри которых строки хранятся в так называемом «heap-хранилище». Такая архитектура ориентирована на операции, при которых чаще всего запрашивается вся строка целиком, что характерно для транзакционных и смешанных (OLTP/OLAP) нагрузок. [1]

PostgreSQL поддерживает широкий набор встроенных типов данных, включая числовые, строковые, временные, диапазонные типы, а также документы в формате JSON/JSONB и пользовательские типы. Функциональность системы расширяется за счёт модулей и расширений, что позволяет добавлять новые типы данных, индексы, функции и средства анализа без изменения ядра СУБД.

Система DuckDB

DuckDB — это встраиваемая аналитическая СУБД, реализующая колоночное хранение данных и векторизованный механизм выполнения запросов, ориентированный на локальную обработку аналитических нагрузок. Система работает «в процессе» приложения, не требуя отдельного серверного компонента: базы данных создаются и используются напрямую из программ на Python, R или других языках, а также в интерактивных средах анализа данных.

Колонко-ориентированная архитектура DuckDB позволяет считывать только необходимые столбцы и эффективно сжимать данные, что особенно полезно при работе с широкими таблицами и большими CSV- или Parquet-файлами. Векторизованное выполнение обрабатывает данные пакетами значений, что улучшает использование кэш-памяти процессора и обеспечивает высокую скорость выполнения агрегирующих и фильтрующих запросов в аналитических сценариях. [2]

Практическая часть

Для проведения сравнительного анализа PostgreSQL и DuckDB была подготовлена единая экспериментальная среда и сформирован набор данных, представляющий собой агрегированные еженедельные отчёты Wildberries. На начальном этапе была развернута виртуальная машина под управлением Ubuntu, на которой установлены обе тестируемые системы управления базами данных. Для PostgreSQL была создана отдельная база данных WB, внутри которой определена таблица wb_buyout_reports, полностью повторяющая структуру исходных CSV-отчетов маркетплейса. [4]

Восемь выгруженных Excel-отчётов Wildberries предварительно были конвертированы в единый CSV-файл. Процесс объединения выполнен средствами Python: данные были сведены в единую таблицу, цветовые и локализованные числовые форматы преобразованы в корректный числовой вид, а итоговый файл сохранён с разделителем «;» и кодировкой UTF-8. Готовый CSV-файл был загружен в PostgreSQL командой COPY, после чего выполнены проверки корректности: сопоставление количества строк, выборки случайных записей, сравнение типов и значений полей с исходными данными.

Реализация запросов

Для тестирования производительности СУБД PostgreSQL и DuckDB была выбрана группа запросов, ориентированных на реальные бизнес-метрики маркетплейса Wildberries. Основной целью было обеспечить репрезентативную нагрузку на систему с учётом структуры данных и объёма таблицы. [3]

Выбранные запросы

Продажи по артикулам с расчётом маржинальности

Сумма вознаграждений, сумма розничной цены, отношение вознаграждения к розничной цене.

Цель: проверить скорость агрегации и вычислений на уровне единицы товара.

Ежедневная аналитика по складам с учётом скидок и хранения

Сумма вознаграждений, средняя согласованная скидка, сумма расходов на хранение.

Цель: нагрузка на группировку по нескольким полям, расчёт средних значений и сумм.

Анализ возвратности товаров

Сумма возвратов, сумма доставок, коэффициент возвратности.

Цель: проверка вычислений с условными операциями (CASE WHEN) и работы с дробными значениями.

Полный финансовый KPI по артикулам

Сумма вознаграждений, комиссии эквайринга, комиссия маркетплейса, логистические расходы, чистая прибыль.

Цель: нагрузка на сложные арифметические выражения с суммированием нескольких полей.

Кумулятивная выручка по артикулам (оконная функция)

Построение накопительной суммы вознаграждений по дате операции для каждого артикула.

Цель: тестирование сложных аналитических операций, оконных функций и их влияния на использование CPU и RAM.

Выбор параметров

Агрегации по артикулу: проверка работы с повторяющимися значениями.

Группировки по дате и складу: имитация ежедневной аналитики на основе всего объёма данных.

Использование арифметики и оконных функций: создание вычислительной нагрузки, имитирующей реальные аналитические задачи.

Выбранные запросы позволяют:

Проверять производительность агрегаций и вычислений на уровне всей таблицы.

Оценивать работу с условными выражениями и дробными значениями (CASE WHEN, AVG, коэффициенты возвратности).

Тестировать сложные аналитические функции, включая оконные функции (cum_revenue).

Сравнивать эффективность PostgreSQL и DuckDB на типовых бизнес-задачах маркетплейса.

Таким образом, набор запросов отражает реальные сценарии аналитики, создаёт нагрузку на CPU, память и диск, и позволяет получить объективные результаты сравнения двух СУБД. [5]

Далее была разработана Python-программа, выполняющая автоматизированный прогон заранее подготовленных SQL-запросов. Скрипт последовательно подключался к PostgreSQL и выполнял один и тот же набор аналитических запросов, включающий группировки, агрегации, фильтрацию по типу документа и выборку топ-N значений. Для каждого запроса фиксировались время выполнения и количество возвращённых строк.

Для обеспечения объективности сравнения была реализована система мониторинга вычислительных ресурсов. В программу встроен класс ResourceMonitor, использующий библиотеку psutil для измерения:

средней и максимальной загрузки CPU;

средней и максимальной загрузки оперативной памяти;

нагрузки на раздел подкачки (swap);

общего количества байтов, прочитанных и записанных на диск во время выполнения запроса.

Мониторинг запускался в отдельном потоке одновременно с началом выполнения запроса и автоматически завершался после получения результата. Таким образом, каждая операция сопровождалась замером не только времени выполнения, но и всей сопутствующей нагрузки на ресурсы системы. [6]

После прогонки всех запросов в PostgreSQL аналогичный набор тестов был выполнен в DuckDB. В рамках эксперимента был использован файл wb.duckdb, содержащий ту же таблицу с отчетами Wildberries, загруженную ранее. Python-скрипт подключался к DuckDB

через официальный драйвер и исполнял идентичные запросы, содержащие такие же группировки и агрегации, но с корректированными именами столбцов, соответствующими формату таблицы в DuckDB. Набор метрик (время выполнения, количество строк, показатели CPU, RAM и дискового ввода-вывода) фиксировался тем же монитором, что и при тестировании PostgreSQL.

Результаты выполнения всех запросов — как PostgreSQL, так и DuckDB — автоматически сохранялись в итоговый CSV-файл для последующего анализа. Форма сохранения позволила структурировать данные и облегчить дальнейшее построение сравнительных таблиц и графиков.

Полученные результаты

Время выполнения

DuckDB показала значительно более быстрое выполнение всех запросов по сравнению с PostgreSQL.

Например, агрегация по 40 000 строк с оконной функцией (кумулятивная выручка) выполнялась в DuckDB за ~45–97 мс, тогда как PostgreSQL тратил на тот же запрос ~335–531 мс.

Для простых агрегатных запросов разница была ещё более заметной: DuckDB выполняла их за 2–5 мс, PostgreSQL — 97–232 мс.

Нагрузка на ресурсы

Средняя загрузка процессора во время выполнения запросов составила около 89 % для PostgreSQL и около 76 % для DuckDB, при том что пиковые значения в обоих случаях достигали 100 %.

Потребление RAM оставалось примерно одинаковым (~57–58 %), swap не использовался.

В PostgreSQL при выполнении запроса с большим объёмом данных наблюдались заметные дисковые операции (до ~5,7 МБ записи на диск), в то время как DuckDB минимально использовала диск, что подтверждает её оптимизацию для аналитики in-memory.

Объём занимаемых данных

Размер данных на диске в DuckDB составил ~ 10,5 МБ, тогда как в PostgreSQL — ~74,6 МБ, что свидетельствует о более эффективном хранении и сжатии в DuckDB.

Выводы

DuckDB демонстрирует значительное преимущество по скорости выполнения аналитических запросов и экономии дискового пространства при обработке таблиц.

PostgreSQL также успешно справляется с задачей, однако демонстрирует большую зависимость от дисковой подсистемы и более высокое время отклика.

Выбор DuckDB для подобных задач аналитики обоснован как с точки зрения производительности, так и с точки зрения компактного хранения данных.

Список литературы:

1. PostgreSQL Global Development Group. PostgreSQL Documentation. – URL: https://www.postgresql.org/docs/ (дата обращения: 10.12.2025). – Текст: электронный.
2. DuckDB Developers. DuckDB Documentation. – URL: https://duckdb.org/docs/stable/ (дата обращения: 10.12.2025). – Текст: электронный.
3. DuckDB Developers. DuckDB – an in-process SQL OLAP database management system. – URL: https://duckdb.org/ (дата обращения: 11.12.2025). – Текст: электронный.

4. Wildberries. Официальные отчёты и документация для партнёров маркетплейса Wildberries. – URL: https://seller.wildberries.ru/about-portal/ru/ru (дата обращения: 11.12.2025). – Текст: электронный.
5. Postgres Professional. Документация PostgreSQL: зеркало официальной документации. – URL: https://postgrespro.com/docs/postgresql/current/ (дата обращения: 11.12.2025). – Текст: электронный.
6. Tutorialspoint. Linux performance monitoring with vmstat and iostat commands. – URL: https://www.tutorialspoint.com/linux-performance-monitoring-with-vmstat-and-iostat-commands (дата обращения: 12.12.2025). – Текст: электронный.
7. Abadi D. J., Boncz P. A., Harizopoulos S. Column-stores vs. row-stores: how different are they really? // ACM SIGMOD International Conference on Management of Data. – 2008. – P. 967–980.
8. El-Sappagh S. H., Hassanien A. E., El Bastawissy A. H. Columnar storage vs. row-based storage for OLAP workloads // Journal of Scientific and Engineering Research. – 2022. – Vol. 9, No. 4. – P. 238–249.

References:

1. The global PostgreSQL Development Team. PostgreSQL documentation. – URL: https://www.postgresql.org/docs / (date of publication: 10.12.2025). – Text: electronic.
2. DuckDB developers. DuckDB documentation. – URL: https://duckdb.org/docs/stable / (date of request: 10.12.2025). – Text: electronic.
3. DuckDB developers. DuckDB is an integrated SQL OLAP database management system. – URL: https://duckdb.org / (date of request: 11.12.2025). – Text: electronic.
4. Wild berries. Official reviews and purchase for Wildberries market partners. – URL: https://seller.wildberries.ru/about-portal/ru/ru (date of request: 11.12.2025). – Text: electronic.
5. Postgres Professional. PostgreSQL download: an email mirror. – URL: https://postgrespro.com/docs/postgresql/current / (date of request: 11.12.2025). – Text: electronic.
6. Tutorialspoint. Linux performance monitoring using vmstat and iostat commands. – URL: https://www.tutorialspoint.com/linux-performance-monitoring-with-vmstat-and-iostat-commands (date of request: 12.12.2025). – Text: electronic.
7. Abadi D. J., Bonch P. A., Charizopoulos S. Storage in columns and storage in rows: how different are they really? // ACM SIGMOD International Conference on Data Management. - 2008. – pp. 967-980.
8. El-Sappag S. H., Hassanien A. E., El-Bastavissi A. H. Columnar storage and row-based storage for OLAP workloads // Journal of Scientific and Engineering Research. – 2022. – Volume 9, No. 4. – pp. 238-249.