МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»
Кафедра математической кибернетики и компьютерных наук

СОЗДАНИЕ РЕШЕНИЯ ДЛЯ ОПЕРАТИВНОГО АНАЛИЗА ДАННЫХ НА БАЗЕ СУБД POSTGRESQL И CLICKHOUSE

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

студента 5 курса 551 группы
направления 09.03.04 — Программная инженерия
факультета КНиИТ
Плотникова Акима Антоновича

Научный руководитель
доцент, к. ф.-м. н. _____ М. И. Сафрончик

Заведующий кафедрой
к. ф.-м. н., доцент _____ С. В. Миронов

Саратов 2025

ВВЕДЕНИЕ

В последние десятилетия объём корпоративных данных возрастает экспоненциально: электронная коммерция, финансовые транзакции, датчики IoT и взаимодействия пользователей в цифровых сервисах порождают терабайты информации ежедневно. Быстрое и достоверное извлечение знаний из этого массива становится ключевым фактором конкурентоспособности компаний. Традиционные операционные СУБД (анг. On-Line Transaction Processing, OLTP-система обработки транзакций в реальном времени), оптимизированные для многопользовательских транзакций, неспособны эффективно выполнять ресурсоёмкие аналитические запросы, требующие многократных сканирований и агрегаций миллиардов строк. В ответ на эти вызовы появились хранилища данных (анг. Data Warehouse) и технологии оперативной многомерной аналитической обработки — OLAP (анг. On-Line Analytical Processing), которые позволяют консолидировать разрозненные источники, формировать целостное историческое представление бизнеса и предоставлять пользователям высокоуровневую аналитику в режиме, близком к реальному времени.

Настоящая работа посвящена созданию законченного OLAP-решения, включающего проектирование многомерной модели, организацию конвейера извлечения-трансформации-загрузки данных (анг. ETL - Extract, Transform, Load), построение куба, написание запросов и вычисление ключевых показателей эффективности (KPI). В качестве аналитического ядра выбрана колоночная MPP-СУБД ClickHouse, обеспечивающая высокую скорость агрегаций благодаря векторизованному исполнителю запросов и эффективному сжатию данных. ClickHouse интегрируется с операционной базой PostgreSQL посредством движка MaterializedPostgreSQL, который в фоновом режиме реплицирует изменения и создаёт материализованные копии таблиц для быстрого анализа. За автоматизацию потоков данных отвечает Apache NiFi — визуальный ETL-opchestrator, способный как к пакетной, так и к поточной обработке, а за интерактивную визуализацию — Metabase. Все компоненты развёрнуты в Docker-контейнерах; это упрощает тестирование, масштабирование и портирование решения.

Объектом исследования выступает процесс хранения и аналитической обработки корпоративных данных, предметом — методы проектирования многомерных моделей, настройки ETL-конвейеров и расчёта KPI с помощью clickhouse.

Цель выпускной квалификационной работы — разработать технологию,

позволяющую построить OLAP-куб, наполнить его корректными и актуальными данными и предоставить пользователям средства оперативного анализа и мониторинга бизнес-метрик. Для достижения указанной цели в ходе работы решаются следующие задачи:

- проводится обзор эволюции хранилищ данных, колоночных СУБД и стриминговых ETL-подходов;
- изучаются архитектурные особенности ClickHouse, дающие прирост производительности аналитических запросов;
- разрабатывается звёздная схема данных: таблица фактов и таблицы измерений, способные адекватно описать структуру продаж и расчётов;
- реализовывается ETL-конвейер в NiFi, обеспечивающий инкрементальное обновление измерений из CSV-файлов и SQL-источников и потоковую репликацию фактов через CDC;
- создается OLAP-куб, описываются выражения для получения сводов, детальных срезов и расчёта производных метрик;
- определяется набор KPI (например, валовая прибыль, количество выполненных заказов, уровень отказов), задается их шкала статусов и интегрируются индикаторы в дашборды Metabase;

В ходе работы использованы методы контент-анализа научной и практической литературы, объектно-ориентированное моделирование, инструментальные эксперименты в контейнеризованной среде, а также визуальная аналитика.

Практическая значимость работы проявляется в создании воспроизводимой методики быстрого развёртывания аналитической платформы, пригодной для коммерческих организаций, требующих непрерывного мониторинга финансово-экономических показателей, и для учебного процесса в качестве лабораторного стенда.

Структура и объём работы. Бакалаврская работа состоит из введения, трех разделов, заключения, списка использованных источников и трех приложений. Общий объем работы – 54 страницы, из них 48 страниц — основное содержание, включая 12 рисунков, список использованных источников информации — 21 наименование.

1 Анализ данных на основе хранилищ и OLAP

Аналитика предприятия начинается с интеграции разнородных источников в хранилище данных — интегрированное, предметно-ориентированное, неизменяемое и привязанное ко времени собрание информации, снимающее нагрузку с оперативных систем и создающее единую основу для отчётов. Для гибкого исследования этих массивов применяют OLAP, который реализует быстрый анализ разделяемой многомерной информации (FASMI) и бывает трёх типов: ROLAP, MOLAP и HOLAP, различающихся балансом между скоростью, актуальностью и объёмом хранимых агрегатов. OLAP позволяет рассматривать данные под разными углами, использовать иерархии, фильтрацию и агрегирование для получения нужных представлений. Ключевые показатели эффективности (KPI) добавляют к аналитике цели, статус и тренд, позволяя быстро оценивать, насколько фактические результаты соответствуют плану. Таким образом, связка «хранилище + OLAP + KPI» обеспечивает полный цикл: от консолидации данных до оперативной, наглядной и управляемой аналитики бизнеса.

1.1 ETL процесс

ETL — это ключевой процесс в аналитике, включающий извлечение, преобразование и загрузку данных из различных источников в хранилище, где они подготавливаются для анализа. С ростом объёмов и требований к скорости особую важность приобретают методы оптимизации: параллельная обработка, инкрементальные загрузки (CDC), упрощение трансформаций, масштабирование ресурсов и постоянный мониторинг. Apache NiFi — мощный инструмент для реализации ETL/ELT, который управляет потоками данных с помощью визуального интерфейса, разделения логики на независимые процессоры и использования надёжных очередей с контролем нагрузки. Он поддерживает потоковую и пакетную обработку, обеспечивает отслеживание происхождения данных и легко интегрируется с внешними системами через Controller Services. Для быстрой аналитики на данных из PostgreSQL в ClickHouse используется движок MaterializedPostgreSQL, который сначала копирует таблицы, а затем реплицирует изменения, сохраняя при этом преимущества колоночного хранения и высокой производительности.

2 Разработка и заполнение хранилища

2.1 Создание окружения

Для развёртывания полноценного аналитического стека данных используется Docker, связывающий между собой следующие компоненты.

1. ClickHouse. Сервис clickhouse поднимает контейнер с сервером ClickHouse на базе Alpine-образа, открывает порты 8123 и 9000, включает поддержку materialized_postgresql и настраивается через монтируемый XML-файл, подключён к сети backend
2. PostgreSQL. Контейнер с версией 16 и включённой логической репликацией (wal_level=logical), создаёт пользователя и базу diplom, использует том postgres_data, входит в сеть backend.
3. Metabase. Веб-интерфейс доступен на порту 3000, использует собственный образ с драйвером ClickHouse, запускается после PostgreSQL и ClickHouse, подключён к backend.
4. Apache NiFi. Запускается с версией 2.4, доступен по порту 8092, использует single-user режим, монтирует драйвер PostgreSQL и директорию пользователя, входит в backend.
5. NiFi Registry. Хранит версии потоков данных, сохраняет конфигурации в том nifi_registry_data.

Для обеспечения сохранности данных и удобства взаимодействия между контейнерами, в Compose-файле используются именованные тома и общая сетевая инфраструктура. Именованные тома:

— nifi_data — используется для хранения конфигурационных файлов Apache NiFi. Благодаря этому настройки сохраняются между перезапусками контейнера.

— nifi_registry_data — предназначен для хранения всех данных NiFi Registry, включая сохранённые версии потоков данных.

— postgres_data — содержит внутренние файлы базы данных PostgreSQL. Это обеспечивает постоянство данных, даже если контейнер будет пересоздан.

Сеть backend — это виртуальная внутренняя сеть Docker, объединяющая все сервисы. Благодаря ей контейнеры могут взаимодействовать друг с другом по имени, без необходимости использовать IP-адреса или пробрасывать дополнительные порты.

2.2 Создание хранилища

Хранилище данных будет базой в PostgreSQL, которая находится в контейнере, описанном в предыдущем разделе. Схема построена по звездной модели (star schema), где есть одна таблица фактов fact_orders, связанная с несколькими измерениями (dim_*), которые дают «разрезы» для аналитики: дата, клиент, провайдер, товар, валюта, статус заказа. Это классическая структура DWH, удобная для аналитики: измерения описывают "контекст", а факт содержит числовые метрики и ключи к измерениям. Были созданы следующие измере

1. Измерение времени (dim_date). Это календарная таблица, позволяет делать аналитику по временным метрикам.
2. Измерение клиента (dim_customer). Справочник пользователей, оформивших заказы, позволяет анализировать заказы по типу клиента.
3. Измерение провайдера (dim_provider). Справочник провайдеров, через которых проходят заказы или платежи. Может использоваться для анализа производительности и надёжности провайдеров.
4. Измерение товара или услуги (dim_item). Справочник товаров и услуг, участвующих в заказах. Используется для анализа по категориям, регионам и типам товаров.
5. Измерение валюты (dim_currency). Справочник валют, используемых при расчётах. Позволяет учитывать многовалютные операции
6. Измерение статуса заказа (dim_order_status). Справочник возможных состояний заказа, используется для анализа жизненного цикла заказа.

После создания таблицы справочника статусов заказов (dim_order_status) она автоматически заполняется на основе значений перечисления order_status. Для этого используется специальная функция enum_range, возвращающая все возможные значения перечисления.

Справочник дат dim_date создаётся с помощью конструкции generate_series, генерирующей последовательность дат с 2020 по 2030 год, для каждой из которых рассчитываются календарные атрибуты.

2.3 Заполнение хранилища

2.3.1 Dim_currency и dim_provider

Для заполнения хранилища используется Apache Nifi. Начнем с заполения таблиц измерений dim_currency и dim_provider. Для извлечения данных

из источника используется процессор ExecuteSQLRecord, который формирует записи в формате Avro с помощью AvroRecordSetWriter. Загрузка данных в хранилище осуществляется через PutDatabaseRecord с использованием AvroReader, при этом обновление выполняется по ключам currency_id и provider_id. Для отладки и аудита применяется процессор LogMessage, фиксирующий сообщения в журнале NiFi.

2.3.2 Dim_item

Таблица dim_item заполняется на основе excel файла. Для излечения данных из файла используется процессор GetFile, который преобразует его в FlowFile для дальнейшей обработки. Несоответствие названий столбцов устраняется с помощью JoltTransformRecord, где данные преобразуются по правилам JOLT, используя ExcelReader для чтения и AvroRecordSetWriter для записи. Завершается процесс загрузкой преобразованных данных в хранилище через PutDatabaseRecord.

2.3.3 Dim_customer

Таблица dim_customer заполняется из двух источников, основным источником является таблица в базе источике, там содержится информация по тарифу, а дополнительная информация приходит из другой системы в формате csv файла. Основные данные о покупателях извлекаются с помощью ExecuteSQLRecord, а дополнительные — через GetFile, после чего оба потока объединяются процессором JoinEnrichment на основе заданного ключа. Для указания роли каждого потока используется UpdateAttribute, а сами данные читаются с помощью AvroReader и CSVReader. После объединения данные загружаются в хранилище с помощью PutDatabaseRecord.

2.3.4 Fact_orders

Таблица fact_orders заполняется из таблицы источника. Данные для таблицы fact_orders извлекаются с помощью QueryDatabaseTableRecord, который отслеживает изменения по полю updated_at, а недостающие поля provider_id и item_id добавляются через LookupRecord по связке с таблицей provider_item. Затем JoltTransformRecord и UpdateRecord преобразуют данные в нужный формат, включая генерацию ключей для даты и статуса. Финальная загрузка осуществляется через PutDatabaseRecord.

3 Разработка аналитики

Для отображения аналитики используется metabase, чтобы его сконфигурировать достаточно настроить пользователя, а затем подключиться через tcp подключение

3.1 Примеры агрегированных запросов с помощью clickhouse

Общие показатели продаж по месяцам. Посчитаем количество заказов и суммарную выручку по месяцам за определённый год. Здесь соединяются факты с размерностью дат, отфильтровывается нужный год (2024) и группируется по месяцу. Результат может быть представлен в виде таблицы или визуализирован графиком (линейным или столбцовым) в разрезе месяцев.

Топ-5 категорий товаров по выручке. Этот запрос полезен для анализа ассортимента – показывает, какие категории приносят наибольшую прибыль. Агрегируем за всё время (если нужен период – можно добавить условие по дате, например d.year = 2024 подобно первому запросу, присоединив также dim_date). Метрики: общая выручка по категории и проданное количество. Результат – топ-5 категорий. Это можно отобразить как горизонтальный барчарт "Top 5 категорий". В ClickHouse такой запрос выполняется очень быстро даже на миллионах строк фактов, поскольку чтение идёт только по нужным колонкам (item_id, amount_in_rub, quantity) и всё агрегирование – в оперативной памяти.

Распределение заказов по статусам. Показывает, сколько заказов завершилось успехом, сколько отменено, в споре и т.д. Этот KPI важен для качественных показателей работы. Здесь через справочник статусов получаем читаемые имена статусов. Визуализировать можно круговой диаграммой (доля успешных vs неуспешных) либо просто показать эти цифры. Кроме того, на основе этих данных можно вычислить процент успешных заказов. В ClickHouse есть удобная функция sumIf для таких случаев, либо можно после получить результат и посчитать долю. Это даст процент успешных заказов от общего числа. Такая метрика (уровень успешности) – важный KPI для качества обслуживания. В PostgreSQL подобный запрос тоже выполним, но если данных очень много, то ClickHouse сделает его быстрее за счёт колонкового скана по status_key.

Географическая разбивка и метрики по клиентам. Посмотрим, как распределяется выручка по странам, и сколько уникальных клиентов в каждой стране. Агрегируем по стране клиента. Получаем, например, топ-10 стран по объёму

продаж. Дополнительно считаем количество уникальных клиентов (функция countDistinct в ClickHouse высокопроизводительная, она примерно подсчитывает на лету). Эти данные можно визуализировать на карте или барчартом. Средний чек по странам покажет платежеспособность аудитории.

3.2 Разработка дашборда

Развернув Metabase поверх ClickHouse, можно создавать интерактивные дашборды – наборы графиков, метрик и таблиц, отображающих ключевые показатели.

3.3 Разработка KPI

Разработанные KPI включают общую выручку, количество заказов и средний чек за выбранный период (например, месяц или квартал), представленные в виде числовых индикаторов на карточках.

3.4 Разработка графиков

Для создания графиков на основе агрегированных запросов, необходимо немного модифицировать запрос ограничив выборку данных по дате, затем используются стандартные средства визуализации metabase

ЗАКЛЮЧЕНИЕ

В ходе выполнения выпускной квалификационной работы была реализована полноценная аналитическая платформа, охватывающая весь жизненный цикл корпоративных данных — от извлечения первичной информации до представления обобщённых показателей руководству. Сначала был проведён обстоятельный обзор современных подходов к построению хранилищ и систем оперативной многомерной аналитики. На основе этого анализа обоснован выбор колоночной СУБД ClickHouse как высокопроизводительной основы для OLAP-нагрузок и движка MaterializedPostgreSQL, обеспечивающего логическую репликацию транзакционных таблиц в аналитическое хранилище практически без задержек.

Для интеграции разнородных источников данных разработан ETL-конвейер на базе Apache NiFi. Его визуально настраиваемые потоки выполняют извлечение, очистку и трансформацию записей, после чего данные автоматически загружаются в измерения и факт-таблицы модели «звезда». Конвейер развёрнут в изолированной Docker-среде и поддерживает инкрементальную обработку: при появлении новых или изменённых строк они мгновенно фиксируются в логах PostgreSQL и доставляются в ClickHouse, что гарантирует актуальность витрин без «ночных» пакетных окон.

Поверх этих выражений определены ключевые показатели эффективности (KPI), такие как валовая выручка, средний чек и коэффициент успешного выполнения заказов. На базе BI-системы Metabase созданы интерактивные дашборды: менеджер может «свернуть» показатели до уровня года или «развернуть» их до конкретного магазина, а цветовая индикация сразу подчёркивает отклонения KPI от целевых значений.

Практическая значимость заключается в том, что разработанное решение может быть оперативно внедрено в любой компании, нуждающейся в прозрачной системе мониторинга экономических показателей, а созданные инструкции и конфигурации могут использоваться в учебном процессе для демонстрации современных технологий бизнес-аналитики.

Таким образом, поставленная цель — создание масштабируемого OLAP-решения для анализа данных и оценки KPI — полностью достигнута. Разработанная методика подтверждает, что комбинация открытых инструментов позволяет организациям быстро получать достоверную, детализированную и визуаль-

но наглядную информацию, служащую надёжной основой для стратегических и оперативных управленческих решений.

В процессе работы были получены практические навыки проектирования аналитической архитектуры, разработки ETL-процессов, настройки потоков данных в Apache NiFi, организации репликации из PostgreSQL в ClickHouse, а также визуализации метрик в BI-среде Metabase. Работа охватывает весь цикл построения аналитической платформы — от построения модели данных до отображения ключевых показателей эффективности.

В дальнейшем планируется расширение функциональности платформы: внедрение витрин для дополнительных бизнес-процессов, автоматизация контроля качества данных, а также реализация механизма обратной связи от пользователей с целью повышения точности и актуальности метрик.