МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»

Кафедра теории функций и стохастического анализа

ПРОЕКТИРОВАНИЕ И РЕАЛИЗАЦИЯ ПЛАТФОРМЫ ОБРАБОТКИ И АНАЛИЗА ПОТОКОВЫХ ДАННЫХ В РЕАЛЬНОМ ВРЕМЕНИ НА БАЗЕ KAFKA, SPARK STRUCTURED STREAMING, AIRFLOW И CLICKHOUSE

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

студента 4 курса 451 группы
направления 38.03.05 — Бизнес-информатика

механико-математического факультета
Боярчук Олеси Алексеевны

Научный руководитель
доцент, к. ф.-м. н. _____ О. А. Мыльцина

Заведующий кафедрой
д. ф.-м. н., доцент _____ С. П. Сидоров

Саратов 2026

ВВЕДЕНИЕ

Актуальность темы исследования. Современные информационные системы формируют большие объёмы данных, которые поступают непрерывно и требуют оперативной обработки. В таких областях, как электронная коммерция, логистика, финансовый мониторинг и цифровые сервисы, важным становится не только накопление данных, но и их анализ практически сразу после возникновения событий. Традиционные подходы пакетной обработки не всегда позволяют обеспечить необходимую скорость получения аналитических результатов, поэтому возрастает роль платформ потоковой обработки данных.

Особую значимость имеют сквозные аналитические платформы, которые обеспечивают полный путь данных: от источника и операционной базы данных до брокера сообщений, потоковой обработки, аналитического хранилища, витрин данных и визуализации. В рамках данной выпускной квалификационной работы рассматривается проектирование и реализация такой платформы на основе Apache Kafka, Spark Structured Streaming, Apache Airflow, ClickHouse, MinIO и Apache Superset.

Цель работы — проектирование и реализация платформы обработки и анализа потоковых данных в реальном времени на базе Kafka, Spark Structured Streaming, Airflow и ClickHouse.

Для достижения поставленной цели необходимо решить следующие задачи:

— изучить теоретические основы потоковой обработки данных и архитектур real-time analytics;
— рассмотреть технологии Kafka, Debezium, Spark Structured Streaming, MinIO, ClickHouse, Airflow и Superset;
— спроектировать архитектуру платформы и определить схему движения данных между компонентами;
— реализовать получение данных из внешнего API и запись событий в PostgreSQL;
— настроить механизм Change Data Capture для передачи изменений из PostgreSQL в Kafka через Debezium;

— реализовать потоковую обработку событий с помощью Spark Structured Streaming и сохранение данных в MinIO в формате Parquet;
— реализовать загрузку данных из MinIO в ClickHouse;
— построить аналитические витрины и реализовать ML scoring;
— настроить оркестрацию пайплайна в Airflow и визуализацию результатов в Superset;

Объектом исследования являются процессы потоковой обработки и анализа данных в распределённых аналитических системах.

Предметом исследования являются методы и инструменты построения платформы real-time analytics с использованием Kafka, Spark Structured Streaming, Airflow и ClickHouse.

Практическая значимость работы заключается в создании работоспособной программной платформы, обеспечивающей автоматизированное получение событий, их передачу через брокер сообщений, потоковую обработку, сохранение в объектное хранилище, загрузку в аналитическую СУБД, построение витрин, расчёт прогнозов и визуализацию результатов. Разработанная система может быть использована как прототип архитектуры потоковой аналитики и как основа для дальнейшего расширения функциональности.

Структура работы. Выпускная квалификационная работа состоит из введения, трёх разделов, заключения и списка использованных источников. В первом разделе рассматриваются теоретические основы потоковой обработки данных и применяемые технологии. Во втором разделе описывается проектирование архитектуры платформы, модель данных, структура хранения и логика взаимодействия компонентов. В третьем разделе представлена практическая реализация системы, фрагменты программного кода, результаты тестирования и анализ полученных данных.

Основное содержание работы. В первом разделе «Теоретические основы потоковой обработки данных»выпускной квалификационной работы рассмотрены теоретические основы потоковой обработки данных и технологии, используемые при построении платформ real-time analytics.

Потоковая обработка данных представляет собой подход, при котором информация обрабатывается непрерывно по мере её поступления в систему. В отличие от пакетной обработки, где данные сначала накапливаются,

а затем обрабатываются через определённые интервалы времени, потоковая обработка позволяет сократить задержку между возникновением события и получением аналитического результата. Это особенно важно для систем, работающих с событиями заказов, платежей, логистических операций, пользовательской активности и другими динамически изменяющимися данными.

В работе показано, что архитектуры потоковой аналитики обычно включают несколько ключевых компонентов: источник данных, операционную базу, механизм передачи изменений, брокер сообщений, систему потоковой обработки, хранилище данных, аналитическую СУБД и средство визуализации. Каждый из этих компонентов выполняет отдельную функцию, а их совместная работа обеспечивает построение сквозного пайплайна обработки данных.

Отдельное внимание в первом разделе уделено механизму Change Data Capture. Данный подход используется для отслеживания изменений в операционных базах данных и передачи этих изменений в другие системы без полного перечитывания исходных таблиц. В рамках разработанной платформы механизм CDC реализован с использованием Debezium, который считывает изменения из PostgreSQL и публикует их в Apache Kafka. Такой подход позволяет автоматически передавать новые события заказов в потоковую часть системы.

Apache Kafka в рассматриваемой архитектуре выполняет роль брокера сообщений. Он обеспечивает приём, хранение и передачу событий между компонентами платформы. Использование Kafka позволяет отделить источник данных от системы обработки и обеспечить асинхронный обмен сообщениями. Это повышает устойчивость архитектуры и упрощает масштабирование отдельных компонентов.

Для обработки событий в работе используется Spark Structured Streaming. Данная технология позволяет описывать потоковую обработку с помощью DataFrame API и выполнять преобразование данных в режиме микропакетов. В разработанной платформе Spark Structured Streaming читает события из Kafka, извлекает данные из структуры Debezium-сообщения, приводит временные метки к нужному формату и сохраняет результат в объектное хранилище MinIO.

MinIO используется как S3-совместимое объектное хранилище. В нём

обработанные события сохраняются в формате Parquet. Выбор данного формата обусловлен тем, что Parquet является колоночным форматом хранения, хорошо подходящим для аналитической обработки данных. Партиционирование данных по дате события позволяет упростить дальнейшую пакетную загрузку и ускорить выборки за конкретные периоды.

В качестве аналитического хранилища в работе используется ClickHouse. Эта СУБД ориентирована на выполнение аналитических запросов по большим объёмам данных и хорошо подходит для построения витрин. В рамках платформы ClickHouse используется для хранения сырых событий заказов, агрегированных витрин и результатов ML scoring.

Apache Airflow рассматривается как инструмент оркестрации. Он позволяет управлять последовательностью выполнения задач, задавать зависимости между этапами обработки, отслеживать статусы запусков и выполнять повторные запуски при необходимости. В разработанной системе Airflow используется как управляющий слой, объединяющий получение данных, потоковую обработку, загрузку в ClickHouse и расчёт прогнозов.

Для визуализации результатов применяется Apache Superset. С его помощью на основе таблиц и витрин ClickHouse строятся графики и дашборды, отражающие распределение заказов по статусам, динамику событий, прогнозы оплаты и показатели времени до оплаты.

Таким образом, в первом разделе сформирована теоретическая основа для дальнейшего проектирования и реализации платформы. Рассмотренные технологии образуют единый стек, позволяющий построить end-to-end решение для обработки и анализа потоковых данных в реальном времени.

Во втором разделе «Проектирование платформы обработки потоковых данных»выпускной квалификационной работы выполнено проектирование платформы обработки и анализа потоковых данных в реальном времени. Основное внимание уделено общей архитектуре системы, схеме движения данных, структуре хранения в MinIO, проектированию таблиц ClickHouse, аналитических витрин, ML scoring и оркестрации пайплайна в Apache Airflow.

Разработанная платформа построена по многоуровневому принципу. В её состав входят источник данных, операционный слой, слой захвата изменений, брокер сообщений, слой потоковой обработки, объектное хранилище,

аналитическое хранилище, слой машинного обучения и слой визуализации. Такое разделение позволяет логически отделить этапы получения, передачи, обработки, хранения и анализа данных.

Общая архитектура разработанной платформы представлена на рисунке 1.

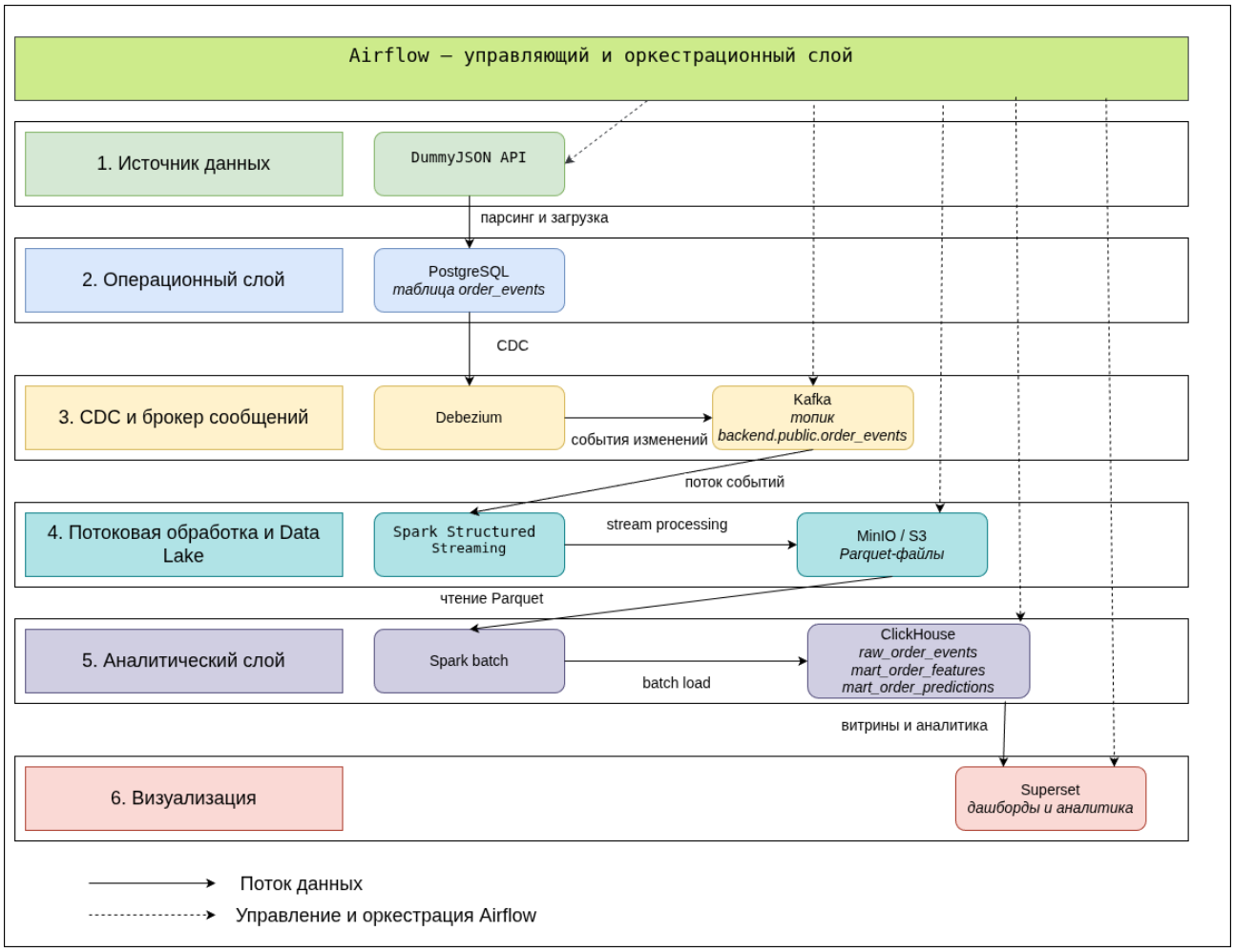

Рисунок 1 – Общая архитектура платформы обработки и анализа потоковых данных

В качестве источника данных в проекте используется внешний API DummyJSON. На его основе формируются события заказов, отражающие изменение их состояния. Полученные события записываются в таблицу `public.order_events` операционной базы данных PostgreSQL. Данная таблица является исходной точкой для дальнейшей передачи данных в потоковую часть платформы.

Для автоматической передачи изменений из PostgreSQL используется механизм Change Data Capture. Он реализован с помощью Debezium, кото-

рый отслеживает изменения в таблице `order_events` и публикует соответствующие события в Kafka topic `backend.public.order_events`. В результате новые записи, появляющиеся в PostgreSQL, автоматически становятся доступными для потоковой обработки без ручного экспорта данных.

Apache Kafka выполняет роль промежуточного брокера сообщений между операционным слоем и слоем обработки. Использование Kafka позволяет связать компоненты платформы асинхронно: источник данных и система обработки не зависят напрямую друг от друга. Это повышает устойчивость архитектуры и позволяет масштабировать отдельные части системы независимо.

Потоковая обработка реализована с использованием Spark Structured Streaming. Spark читает сообщения из Kafka, извлекает актуальное состояние записи из структуры Debezium-сообщения, преобразует временные метки и формирует поле даты события. После обработки данные сохраняются в MinIO в формате Parquet. Для хранения используется бакет `prod` и путь `stream/order_events/`, внутри которого данные партиционируются по дате события.

Выбор MinIO в качестве промежуточного хранилища обусловлен тем, что оно предоставляет S3-совместимый интерфейс и позволяет организовать Data Lake-слой в локальной инфраструктуре. Использование формата Parquet обеспечивает компактное колоночное хранение данных и удобную последующую загрузку в аналитическую СУБД.

Аналитический слой платформы реализован на базе ClickHouse. В базе данных `rt_analytics` спроектированы таблицы для хранения сырых событий, агрегированных витрин и результатов ML scoring. Основная таблица `raw_order_events` содержит события заказов, загруженные из MinIO. На её основе формируются витрины `mart_order_status_realtime` и `mart_order_features`.

Витрина `mart_order_status_realtime` предназначена для анализа количества событий и уникальных заказов по статусам в разрезе дат. Она позволяет оценивать распределение заказов по состояниям `created`, `paid` и `shipped`. Данная витрина используется для построения графиков в Apache Superset.

Витрина `mart_order_features` предназначена для формирования признаков по каждому заказу. В ней рассчитываются последний статус заказа, количество событий по заказу, флаги оплаты и отгрузки, а также время от создания заказа до оплаты. Эти признаки используются как для аналитики, так и в качестве входных данных для модуля ML scoring.

В рамках проектирования ML scoring задача прогнозирования оплаты заказа сформулирована как задача бинарной классификации. Целевая переменная показывает, был ли заказ оплачен. В качестве признаков используются количество событий по заказу, флаг отгрузки и время до оплаты. Результаты прогнозирования сохраняются в таблицу `mart_order_predictions`, содержащую фактическое значение целевой переменной, предсказанный класс, вероятность оплаты и время расчёта прогноза.

Оркестрация всех этапов обработки данных спроектирована в Apache Airflow. Главный DAG `E2E_DummyJSON_Order_Pipeline` объединяет последовательность операций: получение данных из DummyJSON API, загрузку событий из Kafka в MinIO, загрузку данных из MinIO в ClickHouse, построение витрин и запуск ML scoring. Такая структура позволяет контролировать выполнение всего пайплайна из единой точки.

Таким образом, во втором разделе была спроектирована архитектура платформы, обеспечивающая полный путь данных от внешнего источника до аналитического дашборда. Проектирование охватывает как инфраструктурные компоненты, так и модель данных, аналитические витрины, логику прогнозирования и управление пайплайном.

В третьем разделе «Реализация и тестирование платформы обработки потоковых данных»выпускной квалификационной работы представлена практическая реализация разработанной платформы, описание программных компонентов, настройка пайплайна, результаты тестирования и анализ работоспособности системы.

Реализация платформы выполнена в контейнеризированной среде `Docker Compose`. Такой подход позволил объединить все необходимые сервисы в единую локальную инфраструктуру и обеспечить воспроизводимость запуска. В состав окружения входят PostgreSQL, Apache Kafka, Zookeeper, Kafka Connect с Debezium, MinIO, ClickHouse, Apache Airflow, Apache Superset и вспомога-

тельные сервисы.

На первом этапе была реализована загрузка данных из внешнего API DummyJSON. Для этого разработан Python-скрипт, который обращается к API, получает данные о заказах и на их основе формирует события жизненного цикла заказа. Для каждого заказа создаются события со статусами `created`, `paid` и `shipped`. Полученные события записываются в таблицу `public.order_events` базы данных PostgreSQL.

Таблица `order_events` содержит идентификатор заказа, статус события и временную метку. Данная таблица используется как операционный источник данных, изменения в котором далее передаются в потоковую часть системы. После добавления новых строк Debezium фиксирует изменения в PostgreSQL и публикует соответствующие сообщения в Kafka.

Для передачи изменений был настроен Debezium-коннектор PostgreSQL. Коннектор регистрируется через Kafka Connect API и отслеживает таблицу `public.order_events`. При появлении новых записей Debezium формирует JSON-сообщения, содержащие служебную информацию и блок `after`, в котором находится актуальное состояние записи. Эти сообщения публикуются в Kafka topic `backend.public.order_events`.

Далее реализована потоковая обработка сообщений с помощью Spark Structured Streaming. Spark-приложение читает события из Kafka, преобразует сообщение из JSON-формата, извлекает поля заказа, приводит временную метку к типу `timestamp` и формирует поле `event_date`. Обработанные данные сохраняются в MinIO в формате Parquet. Для контроля уже обработанных сообщений используется checkpoint, который позволяет Spark отслеживать Kafka offset-ы.

Следующий этап реализован в виде пакетной Spark-задачи, которая читает Parquet-файлы из MinIO и загружает данные в ClickHouse. Данные попадают в таблицу `rt_analytics.raw_order_events`. После загрузки сырых событий выполняются SQL-запросы построения аналитических витрин.

Витрина `mart_order_status_realtime` агрегирует события по датам и статусам. Она содержит количество событий и количество уникальных заказов для каждого статуса. Данная витрина используется для анализа распределения заказов по состояниям и построения графиков динамики событий.

Витрина `mart_order_features` формирует признаки по каждому заказу. В ней рассчитываются текущий статус заказа, количество событий по заказу, наличие оплаты, наличие отгрузки и время от создания заказа до оплаты. Эти признаки позволяют анализировать поведение заказов и используются как входные данные для ML scoring.

Модуль ML scoring реализован отдельным Python-скриптом. Скрипт читает данные из витрины `mart_order_features`, выполняет подготовку признаков, заполняет пропущенные значения и обучает базовую модель логистической регрессии. После обучения модель применяется ко всей выборке, а результаты сохраняются в таблицу `mart_order_predictions`. В результате для каждого заказа фиксируются фактический класс, предсказанный класс, вероятность оплаты и время расчёта прогноза.

Оркестрация всех этапов реализована в Apache Airflow. Главный DAG `E2E_DummyJSON_Order_Pipeline` выполняет полный сценарий обработки данных. Он последовательно запускает получение данных из API, загрузку Kafka-событий в MinIO, загрузку данных из MinIO в ClickHouse и расчёт ML-прогнозов. Такой подход позволяет контролировать весь пайплайн из единого интерфейса и отслеживать состояние каждой задачи.

Для визуализации результатов был настроен Apache Superset. В Superset создан дашборд, отображающий основные аналитические показатели: распределение событий по статусам, количество заказов по последнему статусу, результаты прогнозирования оплаты, распределение вероятностей оплаты, среднее время до оплаты и заказы с задержкой оплаты.

Пример разработанного дашборда представлен на рисунке 2.

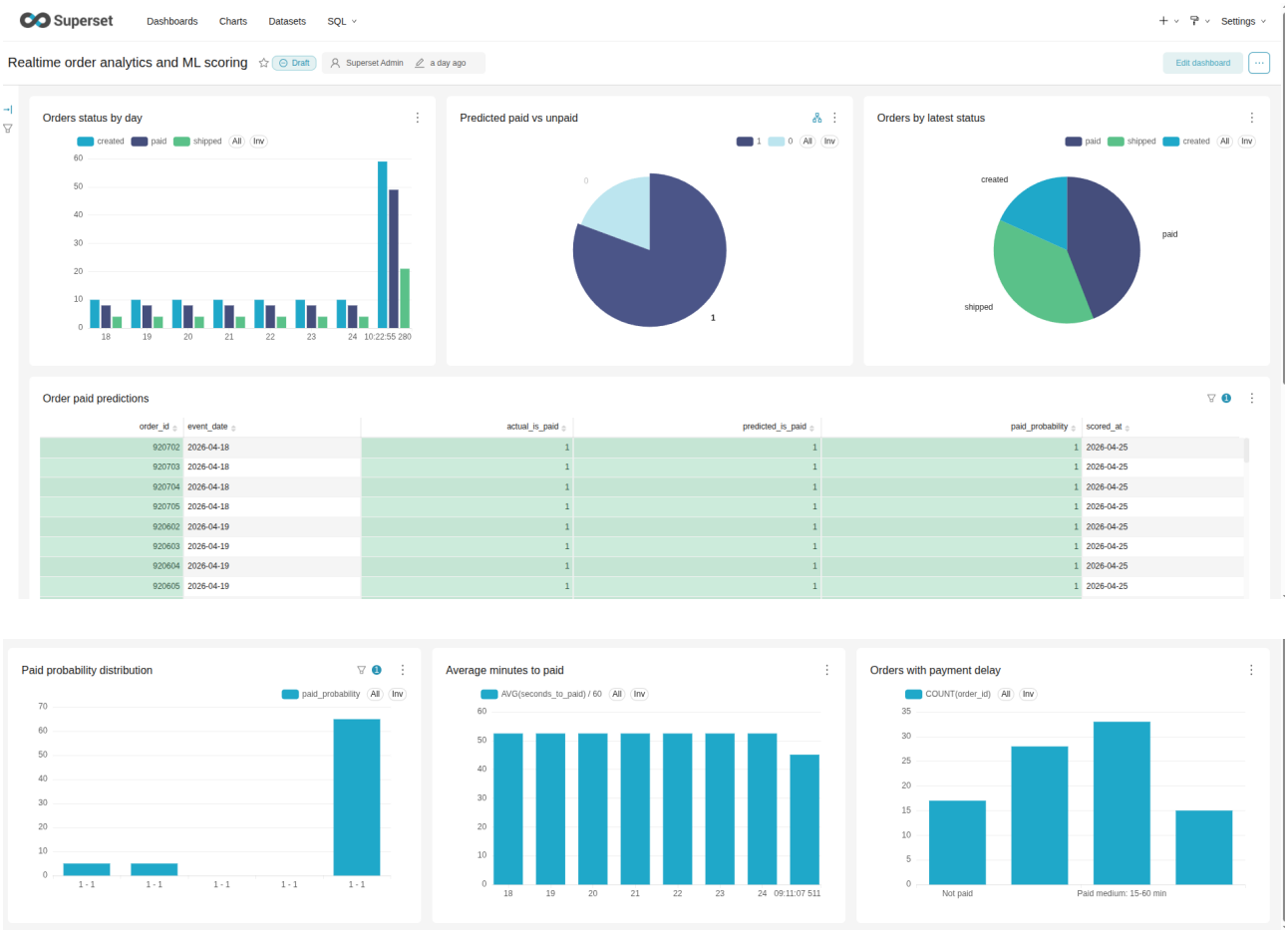

Рисунок 2 – Дашборд Apache Superset для анализа событий заказов

В завершение реализации было выполнено end-to-end тестирование платформы. В ходе проверки новые события заказов были добавлены в PostgreSQL, после чего они последовательно прошли через Debezium, Kafka, Spark Structured Streaming, MinIO, Spark batch и ClickHouse. Затем были построены аналитические витрины и рассчитаны ML-прогнозы.

Результаты тестирования подтвердили работоспособность разработанного пайплайна. В таблице `raw_order_events` появились свежие Это показывает, что данные корректно проходят через все этапы платформы: от внешнего источника до аналитического хранилища и визуализации.

Таким образом, в третьем разделе была реализована и протестирована платформа обработки и анализа потоковых данных в реальном времени. Практическая реализация подтвердила возможность построения сквозного data pipeline на основе Kafka, Spark Structured Streaming, Airflow, ClickHouse и Superset.

Заключение. В ходе выполнения выпускной квалификационной работы была спроектирована и реализована платформа обработки и анализа потоковых данных в реальном времени на базе Kafka, Spark Structured Streaming, Airflow и ClickHouse. Разработанная система обеспечивает полный путь данных: от получения событий из внешнего источника до их обработки, хранения, построения аналитических витрин, расчёта ML-прогнозов и визуализации результатов.

В работе были изучены теоретические основы потоковой обработки данных, рассмотрены технологии Kafka, Debezium, Spark Structured Streaming, MinIO, ClickHouse, Airflow и Superset, а также спроектирована архитектура платформы. Практическая реализация включала получение данных из DummyJSON API, запись событий в PostgreSQL, передачу изменений через Debezium и Kafka, обработку данных в Spark, сохранение в MinIO и загрузку в ClickHouse.

На основе загруженных данных были построены аналитические витрины для анализа статусов заказов и формирования признаков. Также был реализован модуль ML scoring, позволяющий прогнозировать вероятность оплаты заказа. Для управления пайплайном использован Apache Airflow, а для визуализации результатов — Apache Superset.

End-to-end тестирование подтвердило работоспособность платформы: новые события успешно проходят через все этапы обработки и становятся доступны для аналитики и прогнозирования. Таким образом, цель работы была достигнута, а разработанная платформа может рассматриваться как работоспособный прототип системы потоковой аналитики, пригодный для дальнейшего расширения и развития.