МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«Тольяттинский государственный университет»

Кафедра Прикладная математика и информатика
(наименование)

09.03.03 Прикладная информатика
(код и наименование направления подготовки, специальности)

Цифровая трансформация бизнеса
(направленность (профиль) / специализация)

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)

на тему «Проект автоматизации процесса формирования аналитической отчетности системы управления взаимодействием с клиентами туроператора»

Обучающийся К.Ю. Коваленко
(Инициалы Фамилия) (личная подпись)

Руководитель канд. пед. наук, О.А. Крайнова
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Тольятти 2025

Аннотация

Выпускная квалификационная работа посвящена разработке проекта автоматизации процесса формирования аналитической отчётности системы управления взаимодействием с клиентами туроператора путём интеграции платформ Zendesk и Snowflake. Актуальность темы определяется стремительным ростом конкуренции на рынке туристических услуг и постоянной необходимостью получению достоверных данных для принятия обоснованных управленческих решений.

Целью исследования стало создание комплексного решения, позволяющего ежедневно формировать аналитическую отчетность о количестве обращений в контакт-центр в автоматическом режиме. Для достижения цели выполнены задачи анализа существующих бизнес-процессов (AS-IS), разработки целевой модели (TO-BE), выбора технологий, проектирования архитектуры проекта, написания хранимых процедур ETL, визуализации ключевых метрик и расчёта экономического эффекта от реализации проекта автоматизации.

Работа состоит из двух глав: аналитической (диагностика и обоснование решений) и проектной (загрузка и обработка данных, построение отчётности, оценка эффективности). Разработанное решение позволило снизить трудозатраты, повысить точность данных в отчетности и может быть масштабировано на другие предприятия, использующие контакт-центры и работающие во всех отраслях экономики.

Содержание

Введение

В условиях развития туристической отрасли и всё больше возрастающей конкуренции среди участников туристического рынка особую роль приобретает эффективное управление взаимодействием с клиентами туроператоров [10]. Контакт-центры, обрабатывающие ежедневно сотни и тысячи обращений, становятся не просто ключевым звеном в цепи бизнес-процессов, но и важным источником данных, позволяющих руководству предприятия принимать взвешенные управленческие решения. Качественная аналитическая отчетность на основе собранных данных обеспечивает понимание динамики количества обращений клиентов в контакт-центр, степени загруженности отдельных команд и общих перспектив дальнейшего развития бизнеса [15]. Вместе с тем, при отсутствии комплексного решения по автоматизации формирование такой отчетности может занимать значительное время и быть сопряжено с постоянными рисками возникновения ошибок, ведущих к принятию неправильных решений и последующим финансовым потерям. Именно поэтому тема данной дипломной работы представляется весьма актуальной и практически значимой.

Целью настоящей дипломной работы является разработка автоматизированного решения, которое позволит оперативно получать достоверные аналитические данные о работе контакт-центра туристического оператора. В рамках достижения поставленной цели нам необходимо реализовать следующие задачи:

- исследовать и описать текущие бизнес-процессы взаимодействия с клиентами «как есть» (AS-IS), выявить их основные недостатки и ограничения;
- разработать и предложить оптимизированные бизнес-процессы «как должно быть» (TO-BE), отражающие переход к автоматизированной выгрузке, обработке и визуализации аналитических данных.

- проанализировать доступные на рынке решения для хранения и обработки данных контакт-центров;
- обосновать выбор технологий и платформ для автоматизации сбора, хранения и анализа данных об обращениях клиентов;
- спроектировать структурные и последовательные схемы интеграции систем Zendesk и Snowflake, а также методики формирования итоговых аналитических таблиц и визуализации данных;
- оценить экономическую эффективность и потенциальные выгоды для предприятия туристической сферы от реализации проекта автоматизации, обусловленные снижением трудовых затрат и повышением точности аналитических данных.

Объектом исследования выступает процесс формирования аналитической отчетности в системе управления взаимодействием с клиентами туроператора. Предметом исследования служат предложенные подходы к автоматизации рассматриваемого процесса, а также методики интеграции облачных платформ, инструменты сбора и визуализации данных.

В ходе работы используются следующие методы: анализ и моделирование бизнес-процессов, объектно-ориентированный подход к проектированию баз данных и хранимых процедур, а также методы экономического анализа для оценки эффективности внедрения автоматизированной системы формирования отчетности.

Представленная дипломная работа состоит из двух основных глав. Первая глава посвящена анализу текущих бизнес-процессов в контакт-центре туристического оператора, определению основных проблем и обоснованию выбора решений для устранения выявленных недостатков. Во второй главе мы рассмотрели вопросы проектирования интеграции данных Zendesk и Snowflake, описали процесс внедрения системы автоматизированного формирования отчётности, а также провели оценку экономической эффективности проекта. Завершают дипломную работу выводы о достижении

поставленных целей и задач, а также рекомендации по дальнейшему развитию и масштабированию предложенного решения автоматизации.

Представленный подход к интеграции систем Zendesk и Snowflake позволяет обеспечить своевременное и точное формирование статистики по всем ключевым параметрам работы контакт-центра, предоставляя руководству предприятия данные для оперативного управления и стратегического планирования и прогнозирования. Результаты исследования могут быть использованы не только в туристической сфере, но и в других отраслях экономики, где система эффективного взаимодействия с клиентами является одним из важнейших факторов развития и повышения конкурентоспособности.

1 Анализ бизнес-процессов взаимодействия с клиентами туроператора

1.1 Описание автоматизируемого процесса

Ежедневно в контакт-центр предприятия поступают сотни запросов от клиентов по различным темам, которые обрабатываются сотрудниками контакт-центра в системе Zendesk. Руководству предприятия необходимо ежедневно получать данные о количестве поступивших и обработанных обращений в разрезе укрупненных регионов путешествий, данные о перемещении обращений между этими регионами и командами, а также понимать количество необработанных обращений в очереди на конец дня.

До автоматизации процесса формирования аналитической отчетности системы управления взаимодействием с клиентами туроператора такие отчеты формируются сотрудниками с помощью специального встроенного инструмента Zendesk Explorer, на основании этих данных заполняются аналитические таблицы в Excel. Недостаток такого подхода заключался большом количестве операций, которые выполняются вручную, что ведет к росту трудозатрат и повышению риска возникновения ошибок.

Для обеспечения наглядности на рисунках 1 и 2 сформируем процессную модель автоматизируемого процесса «как есть».

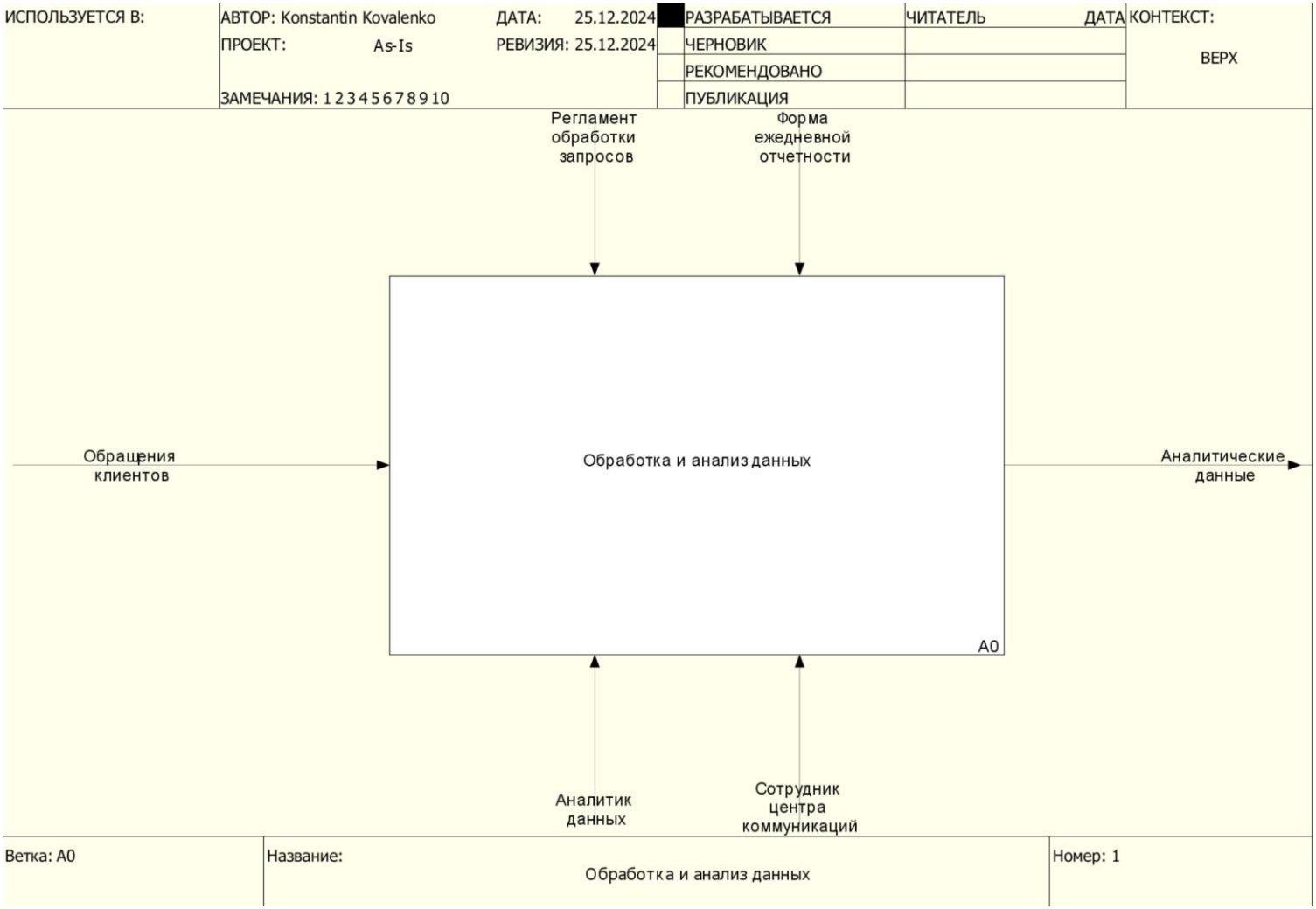

Рисунок 1 – AS-IS A0-диаграмма процесса формирования аналитической отчетности

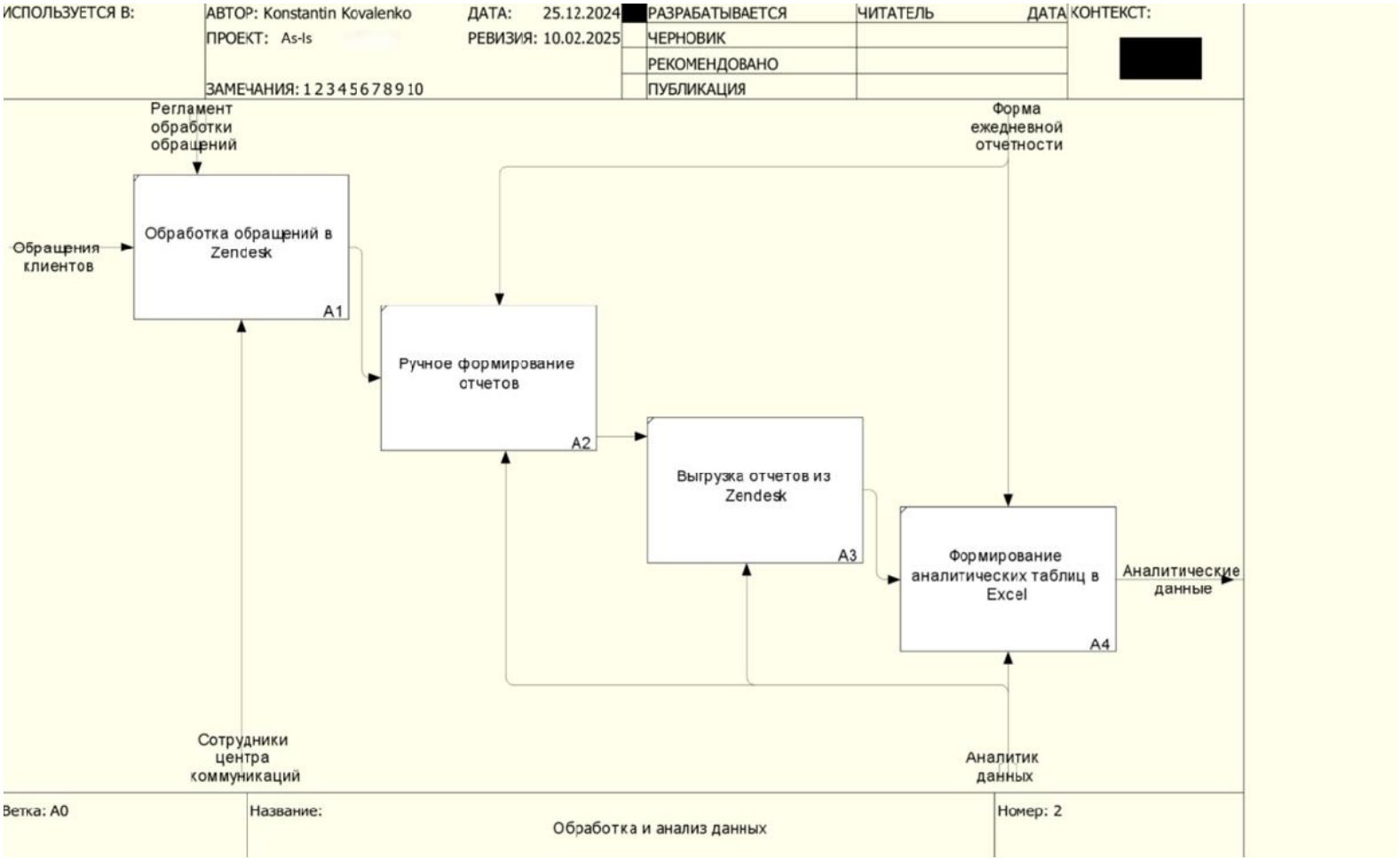

Рисунок 2 – AS-IS A1-диаграмма процесса формирования аналитической отчетности

На схеме «как есть» мы можем увидеть, что сотрудники контакт-центра занимаются обработкой обращений в Zendesk, а затем вручную формируют отчеты при помощи Zendesk Explorer и Excel. Здесь также показана сложная последовательность операций, которая включает в себя выгрузку данных из системы, их сортировку, а также последующее сведение в итоговые аналитические таблицы. Подобный подход требует постоянного внимания исполнителей, отнимает значительное количество рабочего времени и содержит в себе риск возникновения ошибок.

Решением данной проблемы могло бы стать внедрение системы автоматического формирования отчетности, при этом сотрудники контакт-центра продолжают обрабатывать запросы в системе Zendesk, в то же время должна быть налажена автоматическая выгрузка данных из API Zendesk в Snowflake и организовано формирование итоговых аналитических таблиц, а анализ и создание визуализаций данных будет происходить в Power BI и Excel [9]. При этом все описанные действия производятся автоматически.

Данный подход позволит существенно снизить количество операций, выполняемых вручную, уменьшить трудозатраты, а также исключить возможность возникновения ошибок исполнителей [17].

В результате мы получим ежедневно обновляемые данные о количестве поступивших и обработанных обращений в разрезе укрупненных регионов путешествий, данные о перемещении обращений между этими регионами, а также количество необработанных обращений на конец дня.

Полученные данные будут представлять огромную ценность: руководство предприятия сможет понимать, по каким именно регионам путешествий возникает больше всего запросов, какие сотрудники загружены больше всего, а какие меньше. Мы сможем также контролировать очередь из необработанных обращений, и самое главное, мы получаем возможность формировать прогнозы на будущее, что позволит наиболее эффективно

использовать трудовые ресурсы и соответственно повышать прибыльность предприятия [6].

После внедрения автоматической системы отчетности у нас также существенно расширяются возможности для получения аналитических данных. Например, существенную ценность представляет информация от каких организаций поступает больше всего запросов, а также анализ тем запросов в разрезе различных регионов путешествий и команд [20].

1.2 Анализ лучших практик и обоснование выбора решений для автоматизации процесса

Как уже отмечалось выше, контакт-центр рассматриваемого туроператора функционирует на платформе Zendesk, что обеспечивает эффективное взаимодействие с клиентами через обработку поступающих электронных писем. С целью повышения эффективности хранения, обработки данных и подготовки аналитической отчетности запланировано внедрение и использование Snowflake — облачной платформы для хранения, обработки и анализа данных.

Опишем основные требования к управлению современным контакт-центром предприятия:

- централизованное хранение данных: объединение данных из различных источников в единую платформу для обеспечения их целостности и доступности;
- автоматизация процессов: внедрение инструментов для автоматической обработки запросов, поступающих от клиентов, что сокращает время ответа и повышает уровень их удовлетворенности;
- интеграция с CRM-системами: связь контакт-центра с системами управления взаимодействия с клиентами для персонализации обслуживания и повышения качества такого взаимодействия;

– аналитика и отчетность: использование самых современных аналитических инструментов для мониторинга показателей эффективности, выявления тенденций и принятия управленческих решений на основе данных;
– обеспечение безопасности данных: применение методов защиты информации для предотвращения утечек данных и несанкционированного доступа к ним;
– масштабируемость решений: выбор систем, которые способны адаптироваться к постоянному росту объема данных и увеличению нагрузки без снижения производительности.

На рынке есть множество решений для хранения и обработки данных контакт-центров [14]. Мы рассмотрим основные из них:

а) традиционные реляционные базы данных (например, MySQL, PostgreSQL):
   1) подходят для небольших объемов данных,
   2) ограниченная масштабируемость и гибкость при обработке большого объема данных;

б) Data Lakes (например, Amazon S3, Azure Data Lake):
   1) обеспечивают хранение значительных объемов неструктурированных данных,
   2) требуют дополнительного инструментария для обработки и аналитики данных;

в) облачные хранилища данных (например, Google BigQuery, Amazon Redshift):
   1) высокая производительность и масштабируемость,
   2) интеграция с различными аналитическими инструментами,
   3) затраты могут значительно возрасти при больших объемах данных;

г) Snowflake:

1) облачное хранилище данных с высокой производительностью и масштабируемостью,
2) поддержка разных типов данных и простая интеграция с другими платформами,
3) гибкая модель ценообразования, основанная на объеме использования ресурсов,
4) встроенные возможности для обеспечения безопасности и соответствия всем нормативным требованиям [1].

Обоснуем оптимальность выбора Snowflake для проекта автоматизации процесса формирования аналитической отчетности системы управления взаимодействием с клиентами туроператора.

Snowflake представляет собой современную платформу для хранения, обработки и анализа данных, которая обладает рядом неоспоримых преимуществ, делающих ее оптимальным выбором для нашего предприятия [2]. В таблице 1 проведем сравнение Snowflake с альтернативными решениями по ряду критериев.

Таблица 1 – Сравнение Snowflake с альтернативными решениями

Критерии	Snowflake	Amazon Redshift	Google BigQuery	Традиционные базы данных
Масштабируемость	Высокая, динамическая	Высокая, но требует настройки	Высокая, полностью управляемая	Ограниченная
Производительность	Высокая	Высокая	Высокая	Средняя
Интеграция	Простая с Zendesk и другими системами	Хорошая с AWS экосистемой	Хорошая с Google экосистемой	Ограниченная
Стоимость	Гибкая, по использованию	Фиксированная и по использованию	Гибкая, по использованию	Низкая для небольших объемов

Продолжение таблицы 1

Безопасность	Высокий уровень защиты	Высокий уровень защиты	Высокий уровень защиты	Зависит от реализации
Удобство использования	Высокое, интуитивно понятное	Требует настройки	Высокое, интуитивно понятное	Среднее, зависит от СУБД

Snowflake превосходит все другие альтернативные решения по большинству критериев, включая масштабируемость, производительность, интеграцию и безопасность. В отличие от традиционных баз данных, Snowflake предоставляет облачное решение, которое легко можно адаптировать к постоянно меняющимся потребностям бизнеса, что делает его наиболее подходящим выбором для контакт-центра туристического предприятия [3].

Внедрение Snowflake позволит обеспечить эффективное управление большими объемами данных, повысить качество аналитической отчетности и снизить расходы на организацию работы контакт-центра.

Для обеспечения наглядности на рисунках 3 и 4 сформируем процессную модель автоматизируемого процесса «как должно быть».

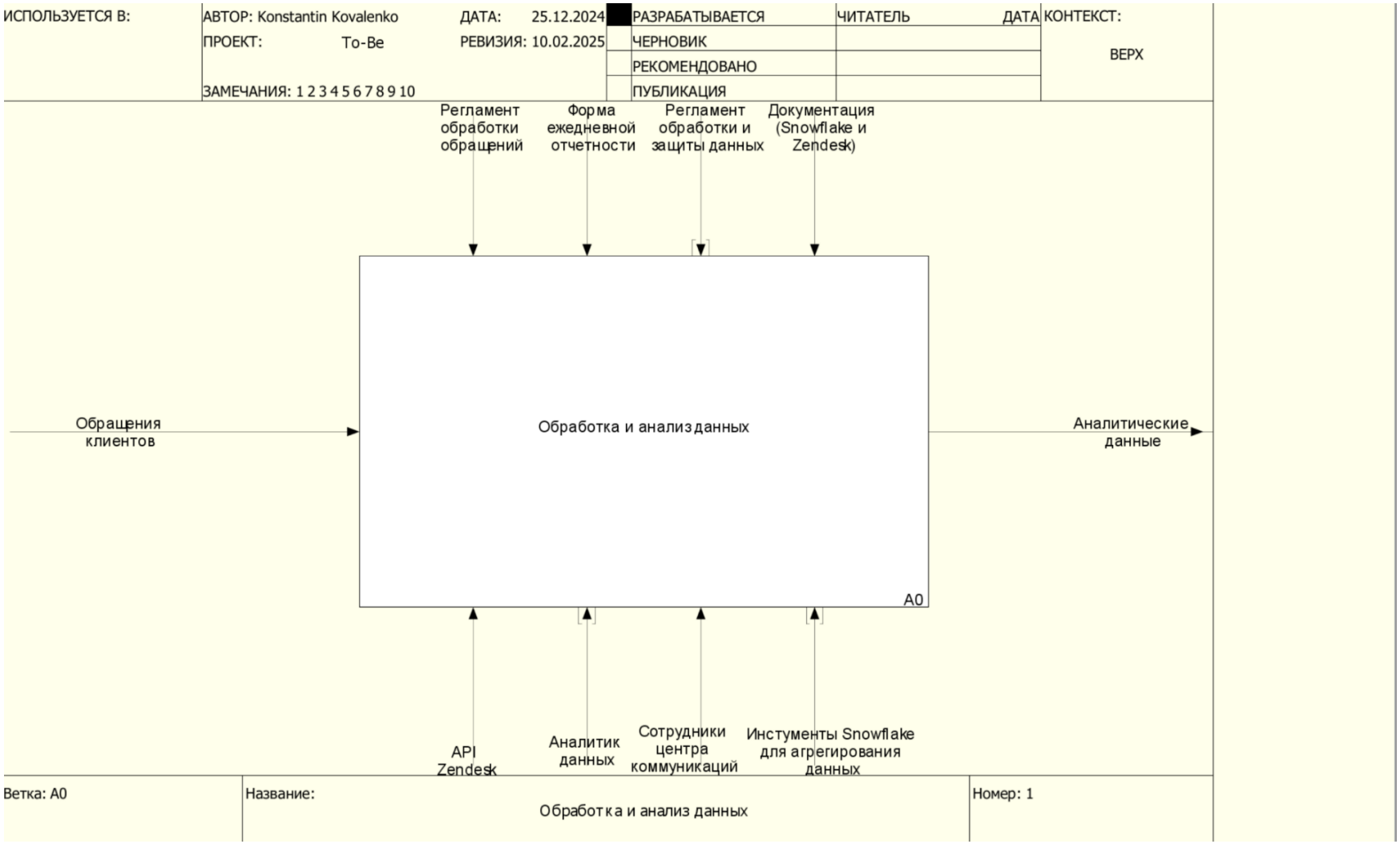

Рисунок 3 – TO-BE A0-диаграмма процесса формирования аналитической отчетности

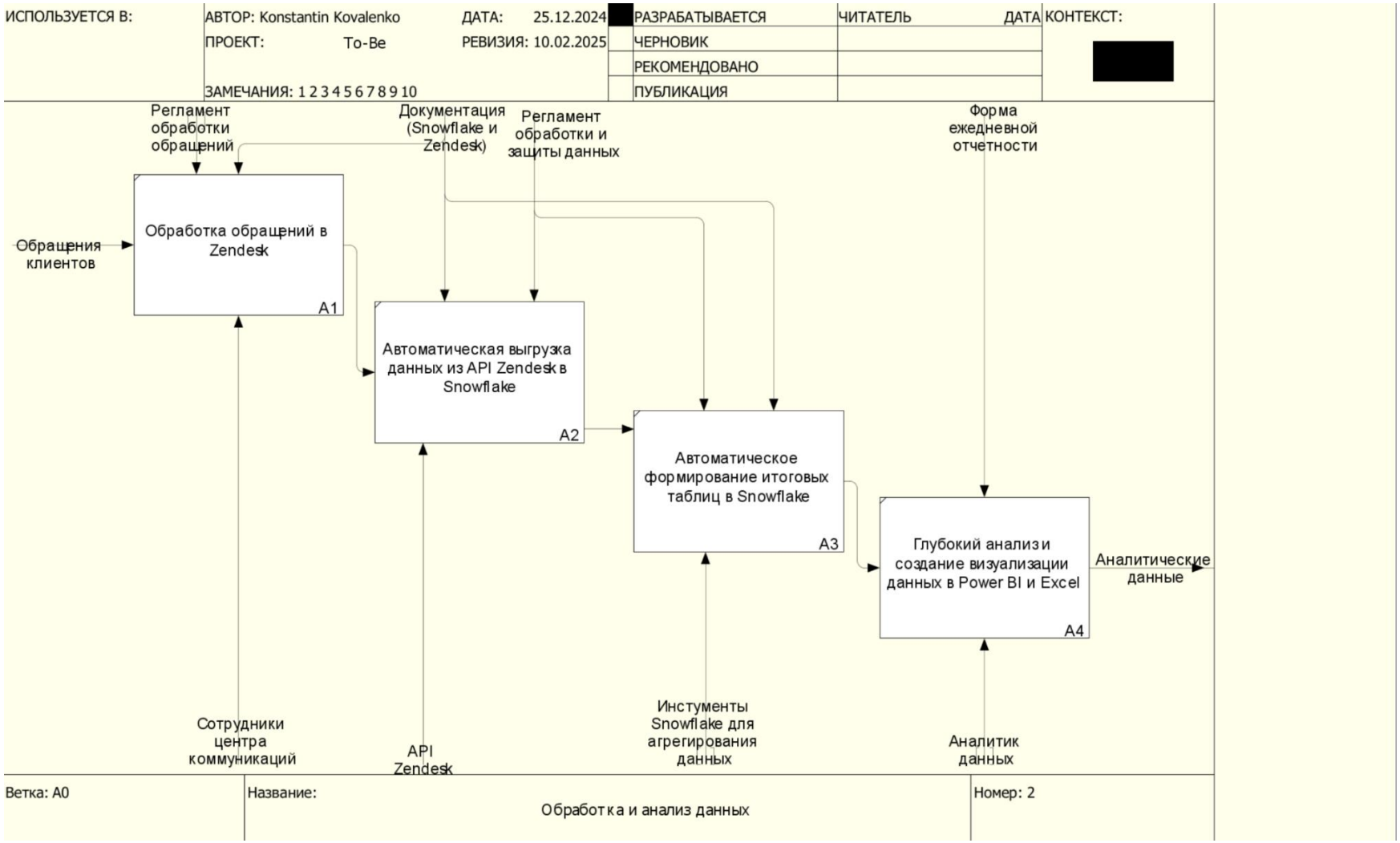

Рисунок 4 – TO-BE A1-диаграмма процесса формирования аналитической отчетности

На схеме «как должно быть» основной упор сделан на автоматизацию процесса формирования отчетности. После обработки обращений в Zendesk данные автоматически выгружаются в Snowflake, где формируется агрегированная информация, готовая к дальнейшему анализу. На следующем этапе создаются аналитические таблицы и визуализации в Power BI и Excel, эти операции производится без участия операторов. Такая модель позволит существенно сократить число ручных операций, снизить вероятность возникновения ошибок и в конечном итоге повысить качество и оперативность принимаемых управленческих решений.

1.3 Бизнес-цели и функциональные требования проекта автоматизации

Рассматриваемый проект автоматизации направлен на оптимизацию работы контакт-центра туристического оператора посредством интеграции системы Zendesk с облачной платформой Snowflake. Цель проекта – создание системы хранения, обработки и анализа данных о запросах, поступающих от клиентов через электронную почту, а также автоматизация процесса формирования аналитической отчетности.

Бизнес-цели проекта автоматизации определяют стратегические направления и результаты, которые предприятие стремится достичь посредством его реализации. Перечислим основные из них:

а) повышение качества обслуживания клиентов:
   1) быстрое и точное реагирование на поступающие запросы,
   2) сокращение времени обработки запросов благодаря грамотной расстановки кадров;

б) оптимизация процессов обработки данных:
   1) централизация хранения данных для обеспечения их целостности,

2) автоматизация ETL-процессов (Extract – извлечение, Transform – трансформация, Load – загрузка) для снижения трудовых затрат и минимизации ошибок при ручной обработке и анализе данных;

в) улучшение аналитических возможностей: создание аналитической отчетности для поддержки управленческих решений и прогнозирования;

г) снижение операционных затрат:

1) оптимизация расходов за счет гибкости модели ценообразования Snowflake,
2) снижение затрат на ИТ-инфраструктуру через облачное решение;

д) обеспечение безопасности и соответствия всем нормативным требованиям: повышение уровня защиты данных клиентов и самого предприятия;

е) повышение конкурентоспособности компании:

1) ускорение внедрения инновационных решений и адаптация к изменениям на рынке туристических услуг,
2) укрепление имиджа предприятия как технологически продвинутого туристического оператора.

Основные функциональные требования к проекту автоматизации представлены на рисунке 5.

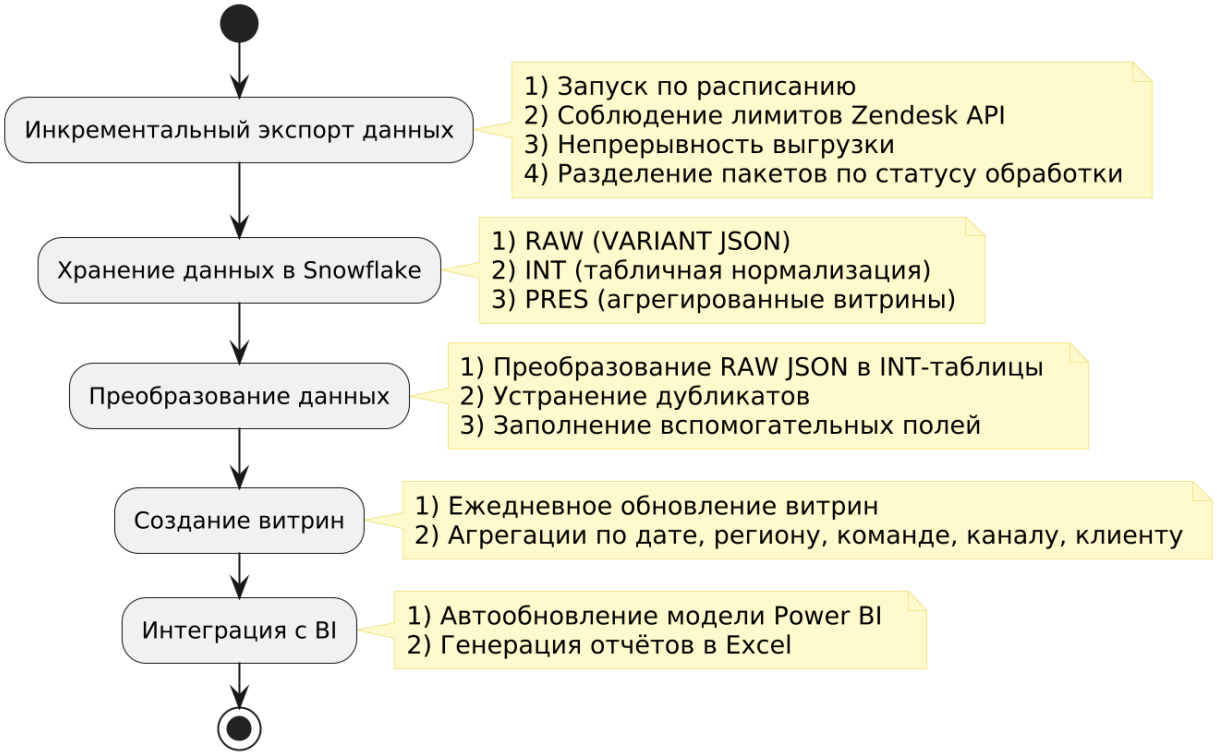

Рисунок 5 – Основные функциональные требования к проекту автоматизации

Итак, в качестве основных функциональных требований к рассматриваемому проекту автоматизации мы определили:

а) инкрементальный экспорт данных из Zendesk:

1) автоматический запуск экспорта по установленному расписанию,
2) соблюдение лимитов Zendesk API по числу запросов в минуту,
3) сохранение контрольной метки после каждой вставки для гарантированной непрерывности выгрузки,
4) отслеживание статуса обработки, позволяющее разделять необработанные и готовые к трансформации пакеты;

б) трёхуровневая схема хранения данных в Snowflake:

1) RAW - прием исходных JSON-выгрузок, тип VARIANT,
2) INT - нормализованные события в табличном формате,
3) PRES - агрегированные витрины метрик для отчётности;

в) преобразование и обработка данных в Snowflake:

1) автоматическое преобразование JSON-данных из слоя RAW в структурированный формат INT с разбором JSON и маппингом его полей на соответствующие колонки табличного слоя,
2) устранение дубликатов,
3) заполнение вспомогательных столбцов, необходимых для формирования статистических таблиц;

г) формирование в Snowflake статистических витрин:

1) ежедневное обновление таблиц PRES с ключевыми показателями работы контакт-центра,
2) поддержка агрегаций по дате, региону, команде, каналу, клиенту;

д) интеграция с BI-инструментами:

1) построение единой модели данных в Power BI и необходимых визуализаций для экспорта данных в Excel, ежедневное автоматическое обновление данных через прямое подключение к Snowflake,
2) создание аналитических отчетов и визуализаций в Excel, позволяющих пользователям получать данные за произвольные периоды времени; связка файла Excel с моделью Power BI для автообновления при каждом открытии;

Перечисленные функциональные требования образуют основу надёжного и масштабируемого конвейера обработки данных, обеспечивая тем самым автоматическое формирование аналитической отчётности контакт-центра предприятия.

2 Проектирование интеграции данных и автоматизации процесса формирования аналитической отчетности

2.1 Архитектура проекта автоматизации и особенности его реализации

2.1.1 Проектирование модели данных в формате JSON

Выгрузки из системы Zendesk представляют собой полуструктурированные JSON-документы, которые сочетают в себе элементы как строго типизированных записей, так и свободно расширяемых структур. В отличие от классических реляционных моделей, где каждая строка в таблице однозначно соответствует фиксированному набору колонок, JSON-формат допускает вложенные объекты, массивы переменной длины и опциональные поля, которые могут появляться не в каждом событии. Это даёт большую гибкость при хранении информации, но требует принципиально иной методики проектирования данных. Сначала нам необходимо сформировать концептуальную модель ключевых сущностей и их связей, затем задать правила валидации через JSON Schema, а на физическом этапе моделирования мы организуем хранение в Snowflake с помощью специальных типов данных (VARIANT) и механизмов парсинга. Такой подход позволит нам одновременно сохранить гибкость полуструктурированного формата и обеспечить надежность и высокое качество данных.

На концептуальном уровне моделирования мы не рассматриваем технологии хранения данных и фокусируемся на ключевых сущностях предметной области и их взаимосвязях. В системе управления взаимодействием с клиентами туроператора мы выделили следующие основные сущности:

– Ticket – единица клиентского обращения, однозначно идентифицируемая ticket_id; содержит время регистрации created_at,

неизменный канал поступления channel и текущий статус обработки status;

- Event – любое действие или изменение в рамках конкретного тикета (комментарий, смена статуса, переназначение), каждое событие имеет уникальный event_id, тип event_type, временную метку event_ts, а также автора updater_id;
- User – автор события, характеризуемый user_id и ролями (agent, end-user), что позволяет нам анализировать активность каждого участника процесса;
- Channel и Area – динамические классификационные атрибуты, отражающие канал поступления обращения (например, «Web form», «Email») и область ответственности (например, «Europa», «Asia»), причем значение этих атрибутов может меняться и фиксируется внутри каждой записи Event.

Связи между сущностями формализуются через отношение один-ко-многим, то есть один Ticket может включать множество Event, каждый Event создаётся одним User и содержит конкретные значения Channel и Area на момент совершения определенного действия.

Для обеспечения наглядности на рисунке 6 нами разработана ER-модель, которая графически отображает перечисленные связи между сущностями. Для реализации данной диаграммы использовано средство PlantUML (PlantText) в нотации Information Engineering (IE-нотация).

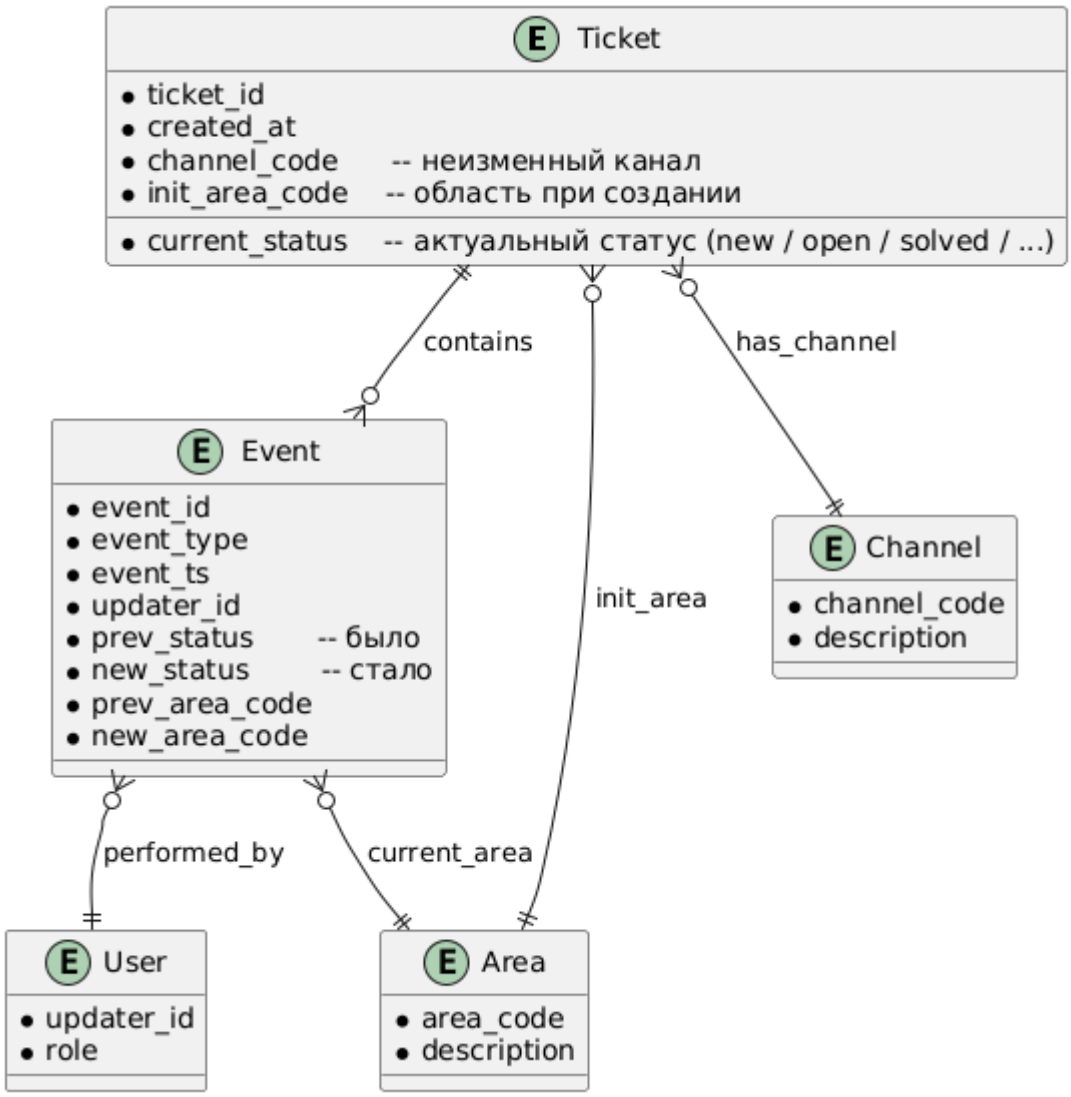

Рисунок 6 – ER-диаграмма концептуальной модели данных

На основе детального анализа JSON-документов и подготовленной ER-диаграммой концептуальной модели данных нами разработана JSON-Schema, которая описывает структуру полуструктурированных JSON-выгрузок – определяет обязательные поля и их типы, вложенные массивы и объекты, списки допустимых значений для статусов и типов событий, а также шаблоны регулярных выражений для кодов каналов и областей. Это позволит нам гарантировать, что все входные данные из Zendesk строго соответствуют концептуальной модели и готовы к автоматической загрузке и трансформации в Snowflake (Приложение А).

На логическом уровне мы определяем, как полуструктурированные JSON-документы из слоя RAW попадают в нормализованную структуру и каким ограничениям они при этом должны соответствовать, что представлено в таблице 2.

Таблица 2 – Логическая модель слоёв ETL-процесса (RAW, INT, PRES)

Элемент логической модели	Назначение	Ограничения
Слой RAW (RAW_JSON_ticket_events)	Приём необработанных данных JSON без изменений	Колонка ticket_events_json VARIANT; уникальность первичной метки ticket_events_next_timestamp; флаг ticket_events_processed – предотвращает повторный парсинг
Слой INT (Z_TICKET_EVENTS)	Нормализованные события в табличном виде	NOT NULL на ticket_id и event_id; CHECK (new_status IN ('new','open','pending','solved','closed')); составной ключ (ticket_id, event_id) устраняет дубли
Слой PRES (витрина Z_STAT)	Агрегированные метрики для BI	Обновление только за предыдущий день: ежедневно ETL-процедура добавляет новые и обновляет изменённые записи, сохраняя историю в рамках одной записи на дату

Таким образом, логический уровень задаёт сквозной «маршрут» – RAW-INT-PRES – и описывает, какие поля, индексы и проверки должны присутствовать на каждом этапе обработки.

На физическом уровне в Snowflake мы реализуем следующие механизмы:

– созданы DDL-скрипты, определяющие три основные таблицы: RAW_JSON_ticket_events с полем VARIANT для хранения вложенных JSON-структур, Z_TICKET_EVENTS с колонками типов NUMBER, STRING и TIMESTAMP_NTZ для распарсенных полей и Z_STAT для агрегированных метрик, что позволяет избежать избыточных преобразований данных при загрузке и обработке;

– таблица RAW_JSON_ticket_events кластеризована по ticket_events_next_timestamp, обеспечивая цепное считывание очередных порций данных; таблица Z_TICKET_EVENTS – по event_ts, что ускоряет аналитические диапазонные запросы;
– для безопасной работы Python-процедуры создана интеграция EXTERNAL_ACCESS_INTEGRATION = (ZENDESK_INTEGRATION) и таблица-хранилище секретных данных SECURE_ZENDESK_CREDENTIALS;
– единый Snowflake-task ежедневно запускает последовательность процедур: DOWNLOAD_JSON → PARSE_RAW_INT → FILL_STAT → AGGREGATE_INT_PRES.

Такое распределение обязанностей (RAW – прием, INT – нормализация, PRES – отчёт) гарантирует качество данных, а вариативные типы VARIANT и JSON-функции Snowflake позволяют при необходимости принять новые поля, которые могут появиться в будущих версиях API Zendesk.

2.1.2 Структурная схема и последовательность реализации проекта автоматизации

В рамках реализации проекта автоматизации процесса формирования аналитической отчетности системы управления взаимодействием с клиентами туроператора мы планируем провести интеграцию систем Zendesk и Snowflake с целью оптимизации процессов обработки и анализа данных. Для организации базы данных в Snowflake создадим структурную схему, которая состоит из трех основных окружений: CC.DEV (Development), CC.TEST (Testing) и CC.PROD (Production).

– CC.DEV: используется для разработки и отладки запросов,
– CC.TEST: предназначается для тестирования разработанных решений перед их внедрением,
– CC.PROD: служит для промышленного использования, обеспечивая доступ к данным в рабочей среде.

В каждом из вышеуказанных окружений предусмотрено три основных раздела:

- RAW – загрузка необработанных данных в формате JSON. Это позволяет сохранять исходные данные без изменений, обеспечивая при этом возможность последующей трансформации и анализа данных;
- INT – создание таблиц с данными и их подготовка для последующей статистической обработки. Этот этап включает в себя очистку, преобразование и объединение данных из различных источников;
- PRES – создание таблиц с готовыми статистическими данными, которые мы будем использовать для визуализации и отчетности.

Движение и трансформация данных будут организованы следующим образом:

- загрузка данных из Zendesk: с помощью API Zendesk будет осуществляться выгрузка данных, которые сохраняются в таблице в формате JSON в разделе RAW (json_ticket_events), что позволит обеспечить хранение исходных событий без их предварительной обработки;
- трансформация данных: необработанные данные из таблицы json_ticket_events будут преобразовываться и загружаться в таблицу z_ticket_events в разделе INT; на данном этапе производится очистка данных, их нормализация и подготовка для дальнейшего анализа;
- формирование статистических данных: из данных таблицы z_ticket_events в разделе PRES будет создаваться таблица z_stat, содержащая агрегированные статистические данные, которые включают ключевые метрики и показатели, необходимые для анализа эффективности работы контакт-центра;
- интеграция с PowerBI: таблица z_stat будет импортироваться в Power BI для создания визуальных отчетов, что позволит визуализировать

статистические данные и проводить глубокий анализ в удобном формате;

- создание отчетности в Excel: на основе данных из Power BI будут создаваться файлы отчетности в формате Excel, которые будут автоматически связываться с моделью данных в Power BI, что обеспечивает их актуальность и согласованность с визуальными отчетами.

Для доступа к данным Zendesk необходимо сформировать логин технического пользователя и токен. В целях обеспечения безопасности и конфиденциальности данных, первоочередной задачей является создание таблицы для хранения секретных данных. Это позволит избежать непосредственного указания чувствительной информации непосредственно в коде, что соответствует рекомендациям по безопасности, отраженным в документации [5].

Для наглядности на рисунке 7 представим последовательность реализации проекта в графическом формате, который выполнен в нотации UML Sequence Diagram (диаграмма последовательностей UML).

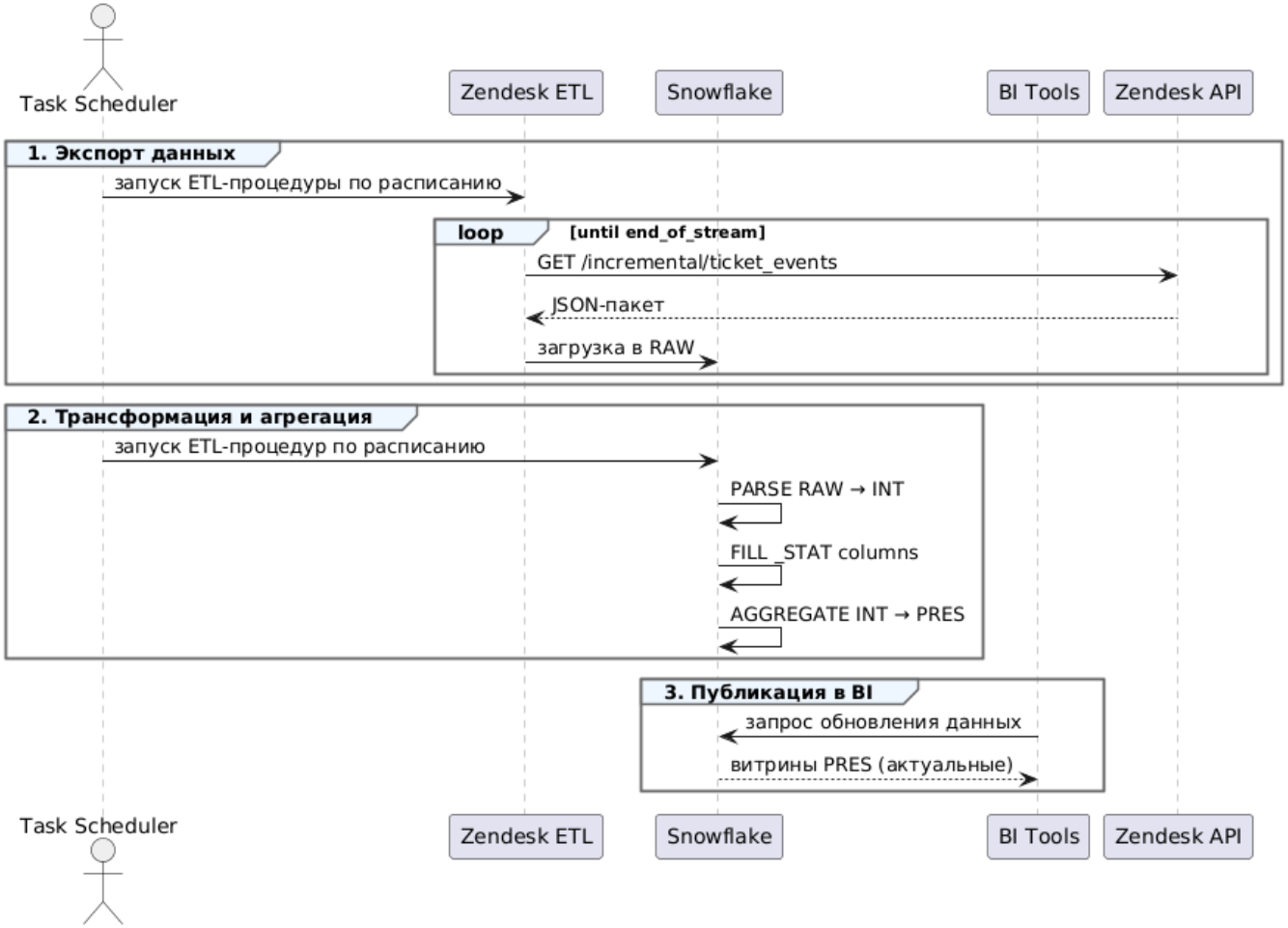

Рисунок 7 – Автоматизированный ETL-конвейер Zendesk, Snowflake и BI

На Рисунке 7 мы наглядно продемонстрировали последовательность взаимодействия систем Zendesk и Snowflake, начиная от аутентификации пользователя в системе Zendesk и заканчивая формированием итоговых аналитических данных, их интеграцией в инструменты визуализации (Power BI, Excel). Представленная диаграмма отражает, какие именно процедуры последовательно запускаются в среде Snowflake, а также то, как данные передаются между различными окружениями RAW, INT и PRES.

После загрузки и структурирования необработанных JSON-данных (RAW) в таблицу z_ticket_events (INT) выполняются операции фильтрации, очистки и нормализации данных. Затем формируется отчетная таблица z_stat (PRES), в которую вносятся агрегированные статистические данные, такие как количество новых, решённых и ожидающих в очереди обращений. Именно эти данные и послужат основой для обновления отчётов в Power BI и Excel, что позволит руководству предприятия и аналитикам оперативно контролировать динамику количества обращений и своевременно принимать управленческие решения, основанные на данных.

2.1.3 Формирование таблицы с данными в формате JSON

Для эффективной интеграции Zendesk и Snowflake необходимо, в первую очередь, правильно организовать процесс загрузки и хранения данных [11]. В данном разделе мы детально рассмотрим процесс формирования таблицы json_ticket_events в разделе RAW, которая служит для хранения необработанных данных о событиях тикетов в формате JSON.

Согласно документации Zendesk [7], API предоставляет возможность инкрементального экспорта трех типов данных:

- tickets: содержат актуальную информацию обо всех тикетах, включая последние изменения за период выгрузки;
- ticket_events: включают все события, связанные с тикетами, что позволяет нам отслеживать все изменения и действия, произошедшие с тикетами;

– users: информация о пользователях системы.

На данном этапе для нас будут актуальны только данные ticket_events, так как они содержат полную историю изменений тикетов, что необходимо для детального анализа. Данные users пока не требуются и могут быть исключены из процесса загрузки.

Хотя данные tickets предоставляют актуальную информацию о тикетах, но они не позволяют отслеживать все произошедшие события и изменения. Поэтому для получения полной картины изменений в тикетах для нас предпочтительнее использовать только таблицу ticket_events, которая фиксирует каждое событие, связанное с тикетом, включая создание, обновление, изменение статуса, а также и другие действия пользователей [4].

Выгрузка данных из Zendesk производится автоматически один раз в сутки в одно и то же время, охватывая данные за прошедшие сутки. Это обеспечивает актуальность данных и позволяет своевременно обновлять их в Snowflake.

При загрузке данных из API Zendesk нам необходимо учесть ограничения Snowflake на объем данных в одной ячейке таблицы. Это ограничение составляет 16 Мб на одну ячейку [5]. Поскольку API Zendesk возвращает данные по 1000 событий за один запрос [7], то загрузка всех данных в одну ячейку может привести к превышению этого лимита. Чтобы избежать этого, нам необходимо загружать данные частями, что обеспечит соответствие установленным ограничениям по объему данных.

Для хранения данных в формате JSON создадим таблицу json_ticket_events с тремя колонками:

– ticket_events_json: содержит данные о 1000 событиях в формате JSON;
– ticket_events_next_timestamp: хранит время последнего события на текущей странице данных, которое используется для инкрементальной выгрузки;

– ticket_events_processed: логическая колонка, по умолчанию имеющая значение FALSE, указывающая на то, что данные пока еще не обработаны.

Для автоматизации процесса загрузки данных из Zendesk в таблицу json_ticket_events мы разработали хранимую процедуру. Эта процедура написана на языке Python и использует библиотеки snowflake-snowpark-python и requests для взаимодействия с API Zendesk и для обработки данных [12].

Указанная процедура выполняет следующие шаги:

а) аутентификация: из таблицы с секретными данными извлекаются логин и токен для доступа к API Zendesk,

б) получение начального времени: определяется временная метка последнего загруженного события из таблицы json_ticket_events.

в) цикл выгрузки данных:

1) формируется URL для запроса данных с использованием временной метки start_time;
2) отправляется GET-запрос к API Zendesk для получения следующего набора событий;
3) полученные данные преобразуются в строку JSON и вставляются в таблицу json_ticket_events вместе с временной меткой для следующей страницы данных;
4) обновляется временная метка для следующего запроса;
5) процесс повторяется до тех пор, пока не будет достигнут конец потока данных (end_of_stream).

г) после успешной загрузки всех данных процедура возвращает сообщение о завершении этого процесса.

В приложении Б приведен полный код создания таблицы json_ticket_events и хранимой процедуры.

2.1.4 Преобразование загруженных данных в табличный формат

После загрузки необработанных данных в формате JSON их необходимо преобразовать в табличный формат для последующего анализа. Это позволит

нам выполнять запросы, агрегировать данные и строить статистические модели на основе структурированных данных.

Для хранения структурированных данных в разделе INT создана таблица z_ticket_events, а для ее ежедневного заполнения разработана специальная хранимая процедура, которая отвечает за извлечение данных из таблицы json_ticket_events, их парсинг и вставку в структурированную таблицу z_ticket_events.

Данная процедура выполняет следующие основные шаги:

а) создание временной таблицы с дубликатами:
   1) получает данные из таблицы json_ticket_events, распаковывает JSON-объекты и извлекает необходимые поля,
   2) фильтрует данные, чтобы избежать обработки уже обработанных событий;

б) вставка уникальных записей в таблицу z_ticket_events: из временной таблицы выбираются уникальные записи, которые затем вставляются в основную таблицу z_ticket_events;

в) обновление значений колонки ticket_events_processed в таблице json_ticket_events на TRUE для всех записей, которые были обработаны в текущей итерации. Это предотвратит повторную обработку тех же данных в будущем;

г) удаление временной таблицы: после успешной вставки данных временная таблица удаляется, чтобы освободить ресурсы системы.

После преобразования необработанных данных в структурированный формат и загрузки их в таблицу z_ticket_events следующим шагом станет заполнение дополнительных столбцов, предназначенных для статистического анализа. В частности, нас интересуют столбцы, название которых заканчивается на «_STAT» и которые содержат информацию о канале (channel) и области ответственности (area) тикета. Эти столбцы необходимы для более глубокого анализа и построения детализированных отчетов.

В процессе реализации проекта стало очевидно, что события, связанные с изменением статуса тикета, часто не содержат данных о канале и области ответственности (channel и area). Обычно эти данные содержатся в событиях, связанных с получением нового тикета. Это затрудняет проведение полного анализа, поскольку данные параметры являются важными метриками для оценки эффективности работы контактного центра.

Более того, в некоторых событиях изменяются одновременно несколько параметров area и channel. Это связано с особенностями функционирования системы Zendesk, где одно и то же событие может инициировать множественные изменения данных.

Для решения вышеуказанных проблем нами была разработана дополнительная хранимая процедура, которая отвечает за заполнение столбцов с окончанием «_stat» в таблице z_ticket_events [18]. Эта процедура выполняет следующие задачи:

а) установка целевой даты: процедура устанавливает целевую дату как предыдущий день по московскому времени, что позволяет ограничить обработку только актуальными данными за прошедший день;

б) обновление статистических столбцов area_stat и channel_stat:
   1) используются оконные функции для извлечения последних и первых значений area и channel для каждого тикета,
   2) заполняются соответствующие статистические столбцы только для тех записей, которые удовлетворяют определенным условиям изменения статуса;

в) учет множественных изменений в одном событии: процедура учитывает случаи, когда area и channel изменяются несколько раз в рамках одного события, гарантируя, что статистические столбцы заполняются корректными значениями;

г) фильтрация данных по целевой дате: обновления выполняются только для записей, соответствующих целевой дате, что обеспечивает их актуальность и релевантность;

д) финальное обновление статистических столбцов: дополнительные условия позволяют нам более точно заполнить статистические столбцы, учитывая особенности изменения статуса тикетов.

В приложении В приведен полный код создания таблицы z_ticket_events и вышеупомянутых хранимых процедур.

2.1.5 Формирование таблицы со статистическими данными и задания на запуск всех хранимых процедур

Для формирования отчетности нам необходимо создать и заполнить данными таблицу с основными показателями, которые мы хотели бы получить для дальнейшего анализа.

Для хранения отчетных данных в разделе PRES создана таблица z_stat, а для ее ежедневного обновления разработана хранимая процедура, которая отвечает за извлечение данных из таблицы z_ticket_events и их вставку в отчетную таблицу.

Данная процедура выполняет следующие основные шаги:

а) установка целевой даты: процедура вычисляет переменную target_date, устанавливая её как вчерашнюю дату по московскому времени, что достигается конвертацией текущей временной метки из часового пояса America/Los_Angeles в Europe/Moscow и последующим вычитанием одного дня.

б) инициализация данных в отчетной таблице: на основе вычисленной переменной target_date в таблицу z_stat вставляются записи для заранее определённых областей (z_europa, z_africa, z_asia), что создаст базовые строки для последующего обновления статистических показателей;

в) обновление статистических данных:

1) новые тикеты: выполняется подсчет тикетов со статусом «new» из таблицы z_ticket_events, соответствующих заданным условиям, обновляется столбец «new»;
2) входящие тикеты: считается количество тикетов, полученных от других команд (с учетом условий по изменению области и каналу), обновляется столбец «mb_in»;
3) исходящие тикеты: аналогичным образом обновляется столбец «mb_out», который отражает тикеты, отправленные в другие команды;
4) решенные тикеты: обновление столбца «solved» происходит на основе тикетов, где статус изменился на «solved», с учетом наличия предыдущего статуса;
5) вновь открытые тикеты: подсчитываются тикеты, которые после решения вновь стали активными, результаты записываются в столбец «new_open»;
6) удаленные тикеты: происходит обновление столбца «deleted» для тикетов, удаленных через определенный канал и при условии, что их предыдущий статус не был «solved»;
7) очередь необработанных тикетов: рассчитывается количество тикетов, остающихся в очереди (не имеющих финальных статусов: solved, closed, deleted) на определенную дату, обновляется столбец «backlog»;

г) завершение процедуры: после выполнения всех операций вставки и обновления статистики процедура возвращает сообщение: «Отчетная таблица обновлена успешно.».

В приложении Г приведен полный код создания таблицы z_stat и хранимой процедуры.

На следующем этапе нам необходимо создать в Snowflake единую процедуру по запуску всех созданных процедур и задание на ее ежедневный запуск [16]. Это позволит формировать статистические данные автоматически

без участия конкретных сотрудников, не зависимо от их отпусков или больничных.

В приложении Д приведен полный код создания единой процедуры и задания на ее выполнение в определенное время.

2.2 Автоматизированное формирование отчетности, визуализация и проверка данных

После настройки ежедневного выполнения всех хранимых процедур следующим шагом стала интеграция отчетной таблицы в Power BI. Это решение позволяет значительно сэкономить кредиты Snowflake, поскольку основная аналитика будет проводиться в Excel. При каждом открытии файла Excel данные будут запрашиваться напрямую у модели Power BI, а не у Snowflake, что и позволит снизить затраты.

Импортирование отчетной таблицы в Power BI из Snowflake будет производиться один раз в сутки. Для организации автоматической и регулярной актуализации данных создан специализированный технический пользователь в Power BI. После импорта таблицы z_stat в Power BI настроим ежедневное обновление модели, обеспечивая тем самым своевременное получение актуальной информации [8].

Для анализа данных нами был создан соответствующий файл в Excel, который содержит следующие листы: «Обзор», «Итого», «Европа», «Африка», «Азия» и «Экспорт».

Лист «Экспорт» мы связали с моделью в Power BI таким образом, что данные на этом листе обновляются при каждом открытии файла [19]. Листы «Европа», «Африка», «Азия» содержат все необходимые показатели для каждого дня, на листе «Итого» происходит суммирование данных на каждый день по всему предприятию в целом.

На листе «Обзор» пользователь может ввести любую дату либо месяц, за которые он хотел бы просмотреть отчетные данные. Также предусмотрена

возможность просмотра данных за весь год. Данный лист содержит ряд аналитических диаграмм, которые визуализируют текущую ситуацию в контакт-центре предприятия [13].

В приложении Е на рисунках Е.1 – Е.5 приведены примеры визуализации данных по различным направлениям путешествий.

Одним из важнейших этапов реализации нашего проекта является проверка полученных данных. С этой целью было сформировано задание для всех пользователей системы фиксировать номера тикетов, которые были обработаны в течение одного дня.

Далее были написаны запросы, которые позволяют получить список всех новых тикетов, полученных в определенный день, список решенных в этот день тикетов, а также список тикетов в очереди на конец дня.

После сравнения списка тикетов, подготовленных сотрудниками контакт-центра, и списка тикетов из базы данных, а также проверки истории обработки отдельных тикетов мы смогли убедиться в том, что отчетные данные верны и могут быть использованы для принятия управленческих решений и построения прогнозов.

В приложении Ж приведены запросы на получение списка необходимых тикетов.

2.3 Оценка экономической эффективности проекта

Реализованный проект по интеграции данных Zendesk и Snowflake позволил не только снизить трудозатраты, но и минимизировать риск возникновения ошибок, который появляется при ручном внесении данных и формировании отчётов. Выработанный подход позволил обеспечить своевременное получение достоверной информации о количестве поступающих и обработанных запросов, динамике их распределения по укрупнённым регионам и очереди из необработанных обращений.

Основные статьи расходов проекта включают в себя:

– затраты на труд одного специалиста: работа инженера данных в течение 3 месяцев, приблизительная оценка затрат основана на уровне рыночной зарплаты специалиста соответствующего уровня;
– приобретение лицензии Power BI, необходимой для создания аналитических визуализаций и глубокого анализа данных, данная статья расходов напрямую связана с использованием данного инструмента;
– использование Snowflake: проект автоматизации реализован с использованием минимального хранилища (XS), ежедневное время работы облачной платформы составило не более 10 минут, что гарантирует достаточно низкие эксплуатационные расходы на хранение и обработку наших данных.

Реализация данного проекта автоматизации даёт возможность добиться следующих положительных результатов:

– автоматизация и сокращение трудозатрат: исключение рутинных действий, выполняемых вручную, существенно снижает вероятность возникновения ошибок и высвобождает время сотрудников для выполнения более важных практических задач;
– повышение оперативности и точности принятия управленческих решений: ежедневно обновляемые аналитические данные позволяют руководству предприятия оперативно реагировать на изменения текущей ситуации, корректировать соответствующие рабочие процессы и оптимизировать распределение трудовых ресурсов;
– оптимальное использование персонала: анализ динамики входящих запросов и их распределения по направлениям путешествий позволяет разрабатывать прогнозы и планировать загрузку контакт-центра, сокращать издержки на оплату труда персонала, а также повысить эффективность работы всего предприятия;

- оценка эффективности работы различных команд: наличие данных о перемещении обращений позволяет нам проводить сравнительный анализ работы различных отделов, мотивировать сотрудников, а также разработать систему премирования для наиболее эффективных команд;
- улучшение качества обслуживания: своевременное получение данных о текущей нагрузке на контактный центр, а также о состоянии очереди обращений позволяет оперативно реагировать на пиковые нагрузки, что также сказывается на уровне удовлетворенности клиентов и, как следствие, на росте объема продаж.

Все вышеперечисленные результаты в конечном итоге приведут к росту конкурентоспособности предприятия и повышению рентабельности его бизнеса.

Заключение

В ходе выполнения дипломной работы была достигнута её главная цель – разработано автоматизированное решение, позволяющее оперативно получать достоверные аналитические данные о работе контакт-центра туристического оператора. Для достижения данной цели нами были решены следующие задачи:

- проведён анализ актуального состояния бизнес-процессов контакт-центра «КАК ЕСТЬ», были выявлены основные проблемы: высокий уровень выполняемых вручную операций, сложность и длительность процесса формирования отчётности, высокий риск возникновения ошибок;
- с учётом возможностей современных облачных технологий и инструментов аналитики разработаны оптимизированные бизнес-процессы «КАК ДОЛЖНО БЫТЬ», в которых все основные этапы – от первичной обработки обращений клиентов до конечной аналитической отчётности – осуществляются автоматически, без ручной выгрузки и сводки данных;
- обоснован и реализован выбор современных технологических решений для интеграции и автоматизации сбора, хранения, обработки и визуализации данных;
- спроектированы структурные схемы и последовательности интеграции систем Zendesk и Snowflake: определены три окружения DEV, TEST, PROD, включающие разделы RAW, INT и PRES, описан порядок инкрементальной выгрузки и обработки полученных данных;
- созданы и протестированы хранимые процедуры, обеспечивающие планомерную загрузку данных в JSON-формате из Zendesk, их

трансформацию в структурированный формат и формирование итоговых аналитических таблиц;

- определены параметры настройки процесса автоматической генерации отчётности с возможностью ежедневного обновления данных, при этом для экономии финансовых ресурсов признано целесообразным анализ данных выполнять в Excel с привязкой файла к модели Power BI;
- выработан и предложен метод проведения проверки полученных данных и подтверждения корректности отчётов путём сопоставления списков тикетов, сформированных сотрудниками предприятия вручную, с выгруженными из системы данными;
- проведена оценка экономической эффективности внедрения проекта автоматизации; благодаря сокращению ручных операций, повышению скорости формирования отчётности и точности данных существенно улучшается качество управления контакт-центром, что, в конечном итоге, приводит к росту конкурентоспособности компании на рынке туристических услуг.

Для внедрения разработанной автоматизированной системы формирования отчетности в промышленную эксплуатацию необходимо:

- организовать ежедневный регламент запуска всех хранимых процедур, выполняющих загрузку и обработку данных;
- настроить механизмы интеграции с Power BI и Excel, включая автоматическое обновление моделей и файлов отчётности;
- обеспечить обучение сотрудников контакт-центра и аналитиков данных работе с новыми инструментами и методами получения данных и формирования отчётности;
- разработать регламенты информационной безопасности и контроля доступа к данным, а также актуализировать внутреннюю

документацию в соответствии с утвержденными требованиями предприятия.

К перспективным направлениям развития проекта относятся:

- расширение объема анализируемых данных за счёт интеграции с дополнительными каналами взаимодействия туроператора с клиентами (соцсети, мессенджеры);
- применение методов машинного обучения и интеллектуального анализа данных (data mining) для построения прогнозов, выявления закономерностей и оптимизации загрузки сотрудников контакт-центра;
- интеграция с другими внутренними системами предприятия.

Таким образом, реализация проекта автоматизации процесса формирования аналитической отчётности показала свою эффективность и соответствие поставленным целям и задачам. Предложенная архитектура и методика внедрения позволили сократить количество операций, выполняемых вручную, усилить контроль за качеством данных и предоставить руководству предприятия актуальный и достоверный инструментарий для принятия обоснованных управленческих решений. Полученные результаты могут быть масштабированы и применены в различных сферах бизнеса, где важно оперативно анализировать большие объёмы получаемой в контакт-центрах информации и формировать на их основе аналитические отчёты.

Список используемой литературы и используемых источников

1. Boegner M. Use Case: Building an Advanced Analytics Platform Using Snowflake’s Cloud Data Warehouse [Электронный ресурс] / Upside’s Engineering Blog. – 16 January 2019. – URL: https://community.snowflake.com/article/Building-An-Advanced-Analytics-Platform-Using-Snowflake (дата обращения: 17.04.2025).
2. Ferle, M. Snowflake Data Engineering. – NY: Manning, 2025. – 368 p.
3. Gershkovich, S. Data Modeling with Snowflake: A practical guide to accelerating Snowflake development using universal data modeling techniques. – Birmingham: Packt, 2023. – 324 p.
4. Jacob C.F. Mastering Zendesk: Master the art of providing effective IT services to your customers by leveraging Zendesk. – Birmingham: Packt, 2017. – 412 p.
5. Snowflake Documentation [Электронный ресурс]. – URL: https://docs.snowflake.com/en/ (дата обращения: 17.04.2025).
6. So K. K. F., Li X. (Robert). Customer Engagement in Hospitality and Tourism Services // Journal of Hospitality & Tourism Research. – 2020. – Vol. 44, № 2. – P. 171–177.
7. Zendesk help [Электронный ресурс]. – URL: https://support.zendesk.com/hc/ (дата обращения: 17.04.2025).
8. Документация по Power BI [Электронный ресурс]. – URL: https://learn.microsoft.com/ru-ru/power-bi/ (дата обращения: 17.04.2025).
9. Еропкина, А. С., Зобнин Ю.А. Современные информационные технологии для автоматизации бизнес-процессов. – Тюмень: Тюменский индустриальный университет, 2018. – 156 с.
10. Есаулова, С. П. Информационные технологии туристической индустрии: учебное пособие. – М: Дашков и К, Ай Пи Эр Медиа, 2010. – 152 с.

11. Грошев, А. С. Основы работы с базами данных: учебное пособие. – М.: Интернет-Университет Информационных Технологий (ИНТУИТ), Ай Пи Ар Медиа, 2025. – 255 с.

12. Кара-Ушанов, В. Ю. SQL – язык реляционных баз данных: учебное пособие. – Екатеринбург: Уральский федеральный университет, ЭБС АСВ, 2016. – 156 с.

13. Пульс, К. Приручи данные с помощью Power Query в Excel и Power BI. – М: ДМК Пресс, 2022. – 572 с.

14. Савельев, А. О. Введение в облачные решения Microsoft: учебное пособие. – М: Интернет-Университет Информационных Технологий (ИНТУИТ), Ай Пи Ар Медиа, 2025. – 229 с.

15. Самолюбова А. Call Center на 100%. Практическое руководство по организации Центра обслуживания вызовов. – М.: Альпина Паблишер, 2013. – 392 с.

16. Селиванова, И. А. Построение и анализ алгоритмов обработки данных: учебно-методическое пособие. – Екатеринбург: Уральский федеральный университет, ЭБС АСВ, 2015. – 108 с.

17. Сивошенко А. Как повысить эффективность контакт-центра и снизить нагрузку на оператора [Электронный ресурс]. URL: https://companies.rbc.ru/news/hif8H3kmoF/kak-povyisit-effektivnost-kontakt-tsentra-i-snizit-nagruzku-na-operatora/ (дата обращения: 17.04.2025).

18. Токмаков, Г. П. Базы данных: модели и структуры данных, язык SQL, программирование баз данных: учебное пособие. – Ульяновск: Ульяновский государственный технический университет, 2021. – 362 с.

19. Феррари, А. Анализ данных при помощи Microsoft Power BI и Power Pivot для Excel. – М: ДМК Пресс, 2020. – 288 с.

20. Шнарева, Г. В. Анализ данных: учебно-методическое пособие. – Симферополь: Университет экономики и управления, 2019. – 129 с.

Приложение А

JSON-схема полуструктурированных выгрузок событий тикетов для ETL-слоя INT

```
{
  "$schema": "https://json-schema.org/draft/2020-12/schema",
  "title":  "Ticket Event for INT Loading",
  "type":   "object",

  "required": [
    "ticket_id",
    "id",
    "timestamp",
    "updater_id",
    "via",
    "child_events"
  ],

  "properties": {
    "ticket_id": {
      "type": "integer"
    },
    "id": {
      "type": "integer"
    },
    "timestamp": {
      "type": "integer"
    },
    "updater_id": {
      "type": "integer"
    },
    "via": {
      "type": "string"
    },
    "child_events": {
      "type": "array",
      "minItems": 1,
      "items": { "$ref": "#/$defs/childEvent" }
    }
  },

  "additionalProperties": false,

  "$defs": {
    "childEvent": {
      "type": "object",
      "required": ["id","event_type"],
      "properties": {
        "id": {
          "type": "integer"
        },
        "event_type": {
          "type": "string",
          "enum": ["Create","Change","Comment","Audit"]
        },
```

```
        "status": {
          "type": ["string","null"]
        },
        "previous_value": {
          "type": ["string","null"]
        },
        "custom_ticket_fields": {
          "type": "object",
          "properties": {
            "25649581": { "type": ["string","null"] },
            "26150212": { "type": ["string","null"] }
          },
          "additionalProperties": true
        }
      },
      "additionalProperties": false
    }
  }
}
```

Приложение Б

Реализация слоя RAW: таблица RAW_JSON_ticket_events и процедура инкрементальной загрузки JSON-событий из Zendesk

Создание таблицы CC_DEV.RAW.RAW_JSON_ticket_events

```
CREATE OR REPLACE TABLE CC_DEV.RAW.RAW_JSON_ticket_events (
    ticket_events_json VARIANT,
    ticket_events_next_timestamp STRING,
    ticket_events_processed BOOLEAN DEFAULT FALSE
)
CLUSTER BY (ticket_events_next_timestamp);
```

Создание процедуры загрузки данных в таблицу CC_DEV.RAW.RAW_JSON_ticket_events

```
CREATE OR REPLACE PROCEDURE CC_DEV.RAW.DOWNLOAD_ZENDESK_TICKET_EVENTS_JSON()
RETURNS STRING
LANGUAGE PYTHON
RUNTIME_VERSION = '3.9'
PACKAGES = ('snowflake-snowpark-python', 'requests')
HANDLER = 'run'
EXTERNAL_ACCESS_INTEGRATIONS = (ZENDESK_INTEGRATION)
AS
$$
def run(session):
    import requests
    from requests.auth import HTTPBasicAuth
    import json

    # Извлечение учетных данных из защищенной таблицы
    credentials_df =
session.table('SECURE_ZENDESK_CREDENTIALS').select('USERNAME', 'TOKEN')
    credentials = credentials_df.collect()

    if not credentials:
        raise Exception("The SECURE_ZENDESK_CREDENTIALS table does not contain
credentials.")

    user = credentials[0]['USERNAME']
    token = credentials[0]['TOKEN']
    auth = HTTPBasicAuth(user, token)
    headers = {"Content-Type": "application/json"}
    session_requests = requests.Session()

    # Получение начального времени выгрузки
    start_time_query = """
    SELECT COALESCE(MAX(ticket_events_next_timestamp), '1729461600')
```

```
    FROM CC_DEV.RAW.z_RAW_JSON_ticket_events;
    """
    start_time = session.sql(start_time_query).collect()[0][0]

    while True:
        # Формирование URL для запроса
        url =
f"https://mktour.zendesk.com/api/v2/incremental/ticket_events.json?start_time={st
art_time}"
        response = session_requests.get(url, auth=auth, headers=headers)
        data = response.json()

        ticket_events = data.get('ticket_events', [])
        next_page = data.get('next_page', None)
        end_of_stream = data.get('end_of_stream', None)

        ticket_events_json_str = json.dumps(ticket_events)

        # Вставка данных в таблицу
        insert_json_query = """
        INSERT INTO CC_DEV.RAW.z_RAW_JSON_ticket_events (ticket_events_json,
ticket_events_next_timestamp)
        SELECT
          PARSE_JSON(?) AS ticket_events_json,
          SPLIT_PART(?, 'start_time=', 2) AS ticket_events_next_timestamp
        WHERE
          ARRAY_SIZE(PARSE_JSON(?)) > 0
        """
        session.sql(insert_json_query, [ticket_events_json_str, next_page,
ticket_events_json_str]).collect()

        # Обновление временной метки для следующего запроса
        update_start_time_query = """
        SELECT COALESCE(MAX(ticket_events_next_timestamp), '0')
        FROM CC_DEV.RAW.z_RAW_JSON_ticket_events;
        """
        start_time = session.sql(update_start_time_query).collect()[0][0]

        if end_of_stream:
            break

        time.sleep(6.5)

    return 'Загрузка событий в формате JSON завершена успешно.'
END;
$$;
```

Приложение В

Реализация слоя INT: таблица CC_DEV.INT.z_ticket_events и процедуры загрузки и заполнения статистических столбцов

Создание таблицы CC_DEV.INT.z_ticket_events

```
CREATE OR REPLACE TABLE CC_DEV.INT.z_ticket_events (
    ticket_events_ticket_id STRING,
    ticket_events_id STRING,
    ticket_events_timestamp STRING,
    TICKET_EVENTS_DATE DATE,
    ticket_events_via STRING,
    ticket_events_child_event_id STRING,
    ticket_events_status STRING,
    ticket_events_previous_status STRING,
    ticket_events_area STRING,
    ticket_events_previous_area STRING,
    ticket_events_channel STRING,
    ticket_events_previous_channel STRING,
    TICKET_EVENTS_AREA_STAT STRING,
    TICKET_EVENTS_PREVIOUS_AREA_STAT STRING,
    TICKET_EVENTS_CHANNEL_STAT STRING,
    TICKET_EVENTS_PREVIOUS_CHANNEL_STAT STRING
)
CLUSTER BY (ticket_events_timestamp);
```

Создание процедуры загрузки данных в таблицу CC_DEV.INT.z_ticket_events

```
CREATE OR REPLACE PROCEDURE CC_DEV.INT.DOWNLOAD_ZENDESK_TICKET_EVENTS()
RETURNS VARCHAR
LANGUAGE SQL
AS
$$
BEGIN
    -- Шаг 1: Создание временной таблицы с распакованными данными
    CREATE OR REPLACE TABLE CC_DEV.INT.z_ticket_events_with_duplicates AS
    SELECT
        json_data.value:ticket_id::STRING AS ticket_events_ticket_id,
        json_data.value:id::STRING AS ticket_events_id,
        json_data.value:timestamp::STRING AS ticket_events_timestamp,
        DATE(json_data.value:timestamp::TIMESTAMP) AS TICKET_EVENTS_DATE,
        json_data.value:via::STRING AS ticket_events_via,
        child.value:id::STRING AS ticket_events_child_event_id,
        child.value:status::STRING AS ticket_events_status,
        IFF(child.value:status IS NOT NULL, child.value:previous_value::STRING,
NULL) AS ticket_events_previous_status,
```

```
        child.value:custom_ticket_fields::OBJECT['2564951']::STRING AS
ticket_events_area,
        IFF(child.value:custom_ticket_fields::OBJECT['2564951'] IS NOT NULL,
child.value:previous_value::STRING, NULL) AS ticket_events_previous_area,
        child.value:custom_ticket_fields::OBJECT['2615021']::STRING AS
ticket_events_channel,
        IFF(child.value:custom_ticket_fields::OBJECT['2615021'] IS NOT NULL,
child.value:previous_value::STRING, NULL) AS ticket_events_previous_channel,
    FROM
        CC_DEV.RAW.JSON_ticket_events,
        LATERAL FLATTEN(input => JSON_ticket_events.ticket_events_json)
json_data,
        LATERAL FLATTEN(input => json_data.value:child_events) child
    WHERE
        JSON_ticket_events.ticket_events_processed = FALSE
        AND (
            child.value:status IS NOT NULL
            OR child.value:custom_ticket_fields::OBJECT['2564951'] IS NOT NULL
            OR child.value:custom_ticket_fields::OBJECT['2615021'] IS NOT NULL
        );
    -- Шаг 2: Вставка уникальных записей в основную таблицу z_ticket_events
    INSERT INTO CC_DEV.INT.z_ticket_events (
        ticket_events_ticket_id,
        ticket_events_id,
        ticket_events_timestamp,
        TICKET_EVENTS_DATE,
        ticket_events_via,
        ticket_events_child_event_id,
        ticket_events_status,
        ticket_events_previous_status,
        ticket_events_area,
        ticket_events_previous_area,
        ticket_events_channel,
        ticket_events_previous_channel
    )
    SELECT DISTINCT
        ticket_events_ticket_id,
        ticket_events_id,
        ticket_events_timestamp,
        TICKET_EVENTS_DATE,
        ticket_events_via,
        ticket_events_child_event_id,
        ticket_events_status,
        ticket_events_previous_status,
        ticket_events_area,
        ticket_events_previous_area,
        ticket_events_channel,
        ticket_events_previous_channel
    FROM CC_DEV.INT.z_ticket_events_with_duplicates
    WHERE ticket_events_timestamp <> (
```

```
        SELECT MAX(ticket_events_next_timestamp)
        FROM CC_DEV.RAW.JSON_ticket_events
    );
    -- Шаг 3: Обновление статуса обработанных записей
    UPDATE CC_DEV.RAW.JSON_ticket_events
    SET ticket_events_processed = TRUE
    WHERE ticket_events_processed = FALSE;

    -- Шаг 4: Удаление временной таблицы
    DROP TABLE CC_DEV.INT.z_ticket_events_with_duplicates;

    RETURN 'Обработка событий тикетов Zendesk завершена успешно.';
END;
$$;
```

Создание процедуры для заполнения статистических столбцов

```
CREATE OR REPLACE PROCEDURE CC_DEV.INT.DOWNLOAD_ZENDESK_TICKET_EVENTS_STAT()
RETURNS VARCHAR
LANGUAGE SQL
AS
$$
BEGIN
    -- Установка целевой даты (предыдущий день по московскому времени)
    SET TARGET_DATE = DATE(CONVERT_TIMEZONE('America/Los_Angeles',
'Europe/Moscow', CURRENT_TIMESTAMP())) - 1;

    -- Шаг 1: Обновление столбцов AREA_STAT и CHANNEL_STAT для определенных
условий
    UPDATE CC_DEV.INT.z_ticket_events z
    SET
        TICKET_EVENTS_AREA_STAT = m.TICKET_EVENTS_AREA,
        TICKET_EVENTS_CHANNEL_STAT = m.TICKET_EVENTS_CHANNEL
    FROM (
        SELECT
            ticket_events_id,
            LAST_VALUE(ticket_events_area IGNORE NULLS) OVER (
                PARTITION BY ticket_events_id
                ORDER BY ticket_events_child_event_id ASC
                ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING
            ) AS TICKET_EVENTS_AREA,
            LAST_VALUE(ticket_events_channel IGNORE NULLS) OVER (
                PARTITION BY ticket_events_id
                ORDER BY ticket_events_child_event_id ASC
                ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING
            ) AS TICKET_EVENTS_CHANNEL
        FROM CC_DEV.INT.z_ticket_events
    ) m
    WHERE
```

```
        z.ticket_events_id = m.ticket_events_id
        AND (
            (z.ticket_events_status IN ('new', 'open', 'hold') AND
z.ticket_events_previous_status IS NULL)
            OR
            (z.ticket_events_status = 'closed' AND
z.ticket_events_previous_status = 'new')
        )
        AND z.TICKET_EVENTS_DATE = $TARGET_DATE;
    -- Шаг 2: Обновление статистических столбцов AREA_STAT и PREVIOUS_AREA_STAT
    UPDATE CC_DEV.INT.z_ticket_events z
    SET
        TICKET_EVENTS_AREA_STAT = m.TICKET_EVENTS_AREA_STAT,
        TICKET_EVENTS_PREVIOUS_AREA_STAT = m.TICKET_EVENTS_PREVIOUS_AREA_STAT
    FROM (
        SELECT
            ticket_events_id,
            FIRST_VALUE(ticket_events_area) OVER (
                PARTITION BY ticket_events_id
                ORDER BY ticket_events_child_event_id DESC
                ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING
            ) AS TICKET_EVENTS_AREA_STAT,
            FIRST_VALUE(ticket_events_previous_area) OVER (
                PARTITION BY ticket_events_id
                ORDER BY ticket_events_child_event_id ASC
                ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING
            ) AS TICKET_EVENTS_PREVIOUS_AREA_STAT
        FROM CC_DEV.INT.z_ticket_events
        WHERE ticket_events_area IS NOT NULL
          AND ticket_events_previous_area IS NOT NULL
    ) m
    WHERE
        z.ticket_events_id = m.ticket_events_id
        AND z.ticket_events_area IS NOT NULL
        AND z.ticket_events_previous_area IS NOT NULL
        AND z.TICKET_EVENTS_DATE = $TARGET_DATE;

    -- Шаг 3: Обновление статистического столбца CHANNEL_STAT
    UPDATE CC_DEV.INT.z_ticket_events z
    SET
        TICKET_EVENTS_CHANNEL_STAT = m.TICKET_EVENTS_CHANNEL
    FROM (
        SELECT
            ticket_events_id,
            FIRST_VALUE(ticket_events_channel IGNORE NULLS) OVER (
                PARTITION BY ticket_events_ticket_id
                ORDER BY ticket_events_timestamp, ticket_events_child_event_id
ASC
                ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING
            ) AS TICKET_EVENTS_CHANNEL
```

```
        FROM CC_DEV.INT.z_ticket_events
    ) m
    WHERE
        z.ticket_events_id = m.ticket_events_id
        AND z.ticket_events_area IS NOT NULL
        AND z.ticket_events_previous_area IS NOT NULL
        AND z.TICKET_EVENTS_DATE = $TARGET_DATE;

    -- Шаг 4: Финальное обновление статистических столбцов AREA_STAT и
CHANNEL_STAT
    UPDATE CC_DEV.INT.z_ticket_events z
    SET
        TICKET_EVENTS_AREA_STAT = m.AREA,
        TICKET_EVENTS_CHANNEL_STAT = m.CHANNEL
    FROM (
        SELECT
            z1.ticket_events_id,
            z1.ticket_events_ticket_id,
            LAST_VALUE(z1.ticket_events_area IGNORE NULLS) OVER (
                PARTITION BY z1.ticket_events_ticket_id
                ORDER BY z1.ticket_events_timestamp,
z1.ticket_events_child_event_id
                ROWS BETWEEN UNBOUNDED PRECEDING AND 1 PRECEDING
            ) AS AREA,
            FIRST_VALUE(z1.ticket_events_channel IGNORE NULLS) OVER (
                PARTITION BY z1.ticket_events_ticket_id
                ORDER BY z1.ticket_events_timestamp
                ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING
            ) AS CHANNEL
        FROM CC_DEV.INT.z_ticket_events z1
    ) m
    WHERE
        z.ticket_events_id = m.ticket_events_id
        AND (
            (z.ticket_events_status = 'closed'
             AND z.ticket_events_previous_status = 'new'
             AND z.ticket_events_via = 'Web service'
             AND z.TICKET_EVENTS_AREA_STAT IS NULL)
            OR z.ticket_events_status IN ('solved', 'deleted')
            OR (
                z.ticket_events_status NOT IN ('closed', 'deleted')
                AND z.ticket_events_status IS NOT NULL
                AND z.ticket_events_previous_status = 'solved'
            )
        )
        AND z.TICKET_EVENTS_DATE = $TARGET_DATE;
    RETURN 'Заполнение статистических столбцов Zendesk завершено успешно.';
END;
$$;
```

Приложение Г

Реализация слоя PRES: таблица CC_DEV.PRES.z_stat и процедура UPDATE_STAT

Создание таблицы CC_DEV.PRES.z_stat

```
CREATE OR REPLACE TABLE Z_STAT (
    DATE DATE,
    AREA STRING,
    NEW STRING,
    MB_IN INTEGER,
    NEW_OPEN INTEGER,
    MB_OUT INTEGER,
    SOLVED INTEGER,
    DELETED INTEGER,
    BACKLOG INTEGER
);
```

Создание процедуры загрузки данных в таблицу CC_DEV.PRES.z_stat

```
CREATE OR REPLACE PROCEDURE UPDATE_STAT()
RETURNS STRING
LANGUAGE SQL
AS
$$
BEGIN
    -- Установка TARGET_DATE на вчерашнюю дату по московскому времени
    LET TARGET_DATE DATE := DATE(CONVERT_TIMEZONE('America/Los_Angeles',
'Europe/Moscow', CURRENT_TIMESTAMP())) - 1;

    -- Вставка даты и AREA в таблицу STAT
    INSERT INTO CC_DEV.PRES.Z_STAT (DATE, AREA)
    SELECT
        :TARGET_DATE AS DATE,
        AREA_list.AREA
    FROM
        (VALUES
            ('z_europa'),
            ('z_africa'),
            ('z_asia')
        ) AS AREA_list(AREA);

    -- Обновление таблицы STAT данными о новых тикетах
    UPDATE Z_STAT AS tgt
    SET NEW = ct.NEW
    FROM (
        SELECT
```

```
        ct.EVENT_DATE AS DATE,
        ct.AREA AS AREA,
        COUNT(ct.TICKET_ID) AS NEW
    FROM (
        SELECT
            TICKET_EVENTS_TICKET_ID AS TICKET_ID,
            TICKET_EVENTS_AREA_STAT AS AREA,
            TICKET_EVENTS_DATE AS EVENT_DATE
        FROM CC_DEV.INT.Z_TICKET_EVENTS
        WHERE
            TICKET_EVENTS_CHANNEL_STAT = 'z_ch_e01'
            AND TICKET_EVENTS_STATUS = 'new'
            AND TICKET_EVENTS_AREA_STAT IS NOT NULL
            AND TICKET_EVENTS_DATE = :TARGET_DATE
    WHERE
       ct.AREA LIKE 'z_%'
       AND excluded_tickets.TICKET_ID IS NULL
    GROUP BY ct.EVENT_DATE, ct.AREA
) AS ct
WHERE tgt.DATE = ct.DATE
  AND tgt.AREA = ct.AREA;

-- Обновление таблицы STAT данными о полученных от других команд тикетах
UPDATE Z_STAT AS tgt
SET mb_in = COALESCE(src.mb_in, 0)
FROM (
    SELECT
        TICKET_EVENTS_DATE AS EVENT_DATE,
        TICKET_EVENTS_AREA_STAT AS AREA,
        COUNT(TICKET_EVENTS_TICKET_ID) AS mb_in
    FROM CC_DEV.INT.Z_TICKET_EVENTS
    WHERE
        TICKET_EVENTS_AREA_STAT LIKE 'z_%'
        AND TICKET_EVENTS_AREA_STAT != TICKET_EVENTS_PREVIOUS_AREA_STAT
        AND TICKET_EVENTS_CHANNEL_STAT = 'z_ch_e01'
        AND TICKET_EVENTS_VIA IN ('Web service', 'Web form')
        AND TICKET_EVENTS_DATE = :TARGET_DATE
    GROUP BY 1, 2
) AS src
WHERE tgt.DATE = src.EVENT_DATE
  AND tgt.AREA = src.AREA
  AND tgt.DATE = :TARGET_DATE;

-- Обновление таблицы STAT данными о направленных в другие команды тикетах
UPDATE Z_STAT AS tgt
SET mb_out = COALESCE(src.mb_out, 0)
FROM (
    SELECT
        TICKET_EVENTS_DATE AS EVENT_DATE,
        TICKET_EVENTS_PREVIOUS_AREA_STAT AS AREA,
```

```
            COUNT(TICKET_EVENTS_TICKET_ID) AS mb_out
        FROM CC_DEV.INT.Z_TICKET_EVENTS
        WHERE
            TICKET_EVENTS_PREVIOUS_AREA_STAT LIKE 'z_%'
            AND TICKET_EVENTS_AREA_STAT != TICKET_EVENTS_PREVIOUS_AREA_STAT
            AND TICKET_EVENTS_CHANNEL_STAT = 'z_ch_e01'
            AND TICKET_EVENTS_VIA IN ('Web service', 'Web form')
            AND TICKET_EVENTS_DATE = :TARGET_DATE
        GROUP BY 1, 2
    ) AS src
    WHERE tgt.DATE = src.EVENT_DATE
      AND tgt.AREA = src.AREA
      AND tgt.DATE = :TARGET_DATE;


    -- Обновление таблицы STAT данными о решенных тикетах
    UPDATE Z_STAT AS tgt
    SET SOLVED = COALESCE(src.SOLVED, 0)
    FROM (
        SELECT
            TICKET_EVENTS_DATE AS event_date,
            TICKET_EVENTS_AREA_STAT AS AREA,
            COUNT(TICKET_EVENTS_TICKET_ID) AS SOLVED
        FROM CC_DEV.INT.Z_TICKET_EVENTS
        WHERE
            TICKET_EVENTS_CHANNEL_STAT = 'z_ch_e01'
            AND TICKET_EVENTS_STATUS = 'solved'
            AND TICKET_EVENTS_PREVIOUS_STATUS IS NOT NULL
            AND TICKET_EVENTS_AREA_STAT LIKE 'z_%'
          GROUP BY 1, 2
    ) AS src
    WHERE tgt.DATE = src.event_date
      AND tgt.AREA = src.AREA
      AND tgt.DATE = :TARGET_DATE;

    -- Обновление таблицы STAT данными о вновь открытых тикетах
    UPDATE Z_STAT AS tgt
    SET NEW_OPEN = src.NEW_OPEN
    FROM (
        SELECT
            TICKET_EVENTS_DATE AS event_date,
            TICKET_EVENTS_AREA_STAT AS AREA,
            COUNT(TICKET_EVENTS_TICKET_ID) AS NEW_OPEN
        FROM CC_DEV.INT.Z_TICKET_EVENTS
        WHERE
            TICKET_EVENTS_CHANNEL_STAT = 'z_ch_e01'
            AND (TICKET_EVENTS_STATUS NOT IN ('closed', 'deleted') AND
TICKET_EVENTS_STATUS IS NOT NULL)
            AND TICKET_EVENTS_PREVIOUS_STATUS = 'solved'
            AND TICKET_EVENTS_AREA_STAT LIKE 'z_%'
```

```
            AND TICKET_EVENTS_DATE = :TARGET_DATE
        GROUP BY 1, 2
    ) AS src
    WHERE tgt.DATE = src.event_date
      AND tgt.AREA = src.AREA
      AND tgt.DATE = :TARGET_DATE;

    -- Обновление таблицы STAT данными об удаленных тикетах
    UPDATE Z_STAT AS tgt
    SET DELETED = src.DELETED
    FROM (
        SELECT
            TICKET_EVENTS_DATE AS event_date,
            TICKET_EVENTS_AREA_STAT AS AREA,
            COUNT(TICKET_EVENTS_TICKET_ID) AS DELETED
        FROM CC_DEV.INT.Z_TICKET_EVENTS
        WHERE
            TICKET_EVENTS_CHANNEL_STAT = 'z_kan_e01'
            AND TICKET_EVENTS_STATUS = 'deleted'
            AND TICKET_EVENTS_PREVIOUS_STATUS != 'solved'
            AND TICKET_EVENTS_AREA_STAT LIKE 'z_%'
            AND TICKET_EVENTS_DATE = :TARGET_DATE
        GROUP BY 1, 2
    ) AS src
    WHERE tgt.DATE = src.event_date
      AND tgt.AREA = src.AREA
      AND tgt.DATE = :TARGET_DATE;

    -- Обновление таблицы STAT данными о количестве тикетов в очереди
    UPDATE Z_STAT AS tgt
    SET BACKLOG = src.BACKLOG
    FROM (
        SELECT
            TICKET_EVENTS_LAST_AREA AS AREA,
            COUNT(DISTINCT TICKET_EVENTS_TICKET_ID) AS BACKLOG
        FROM CC_DEV.INT.Z_TICKET_EVENTS
        WHERE
            TICKET_EVENTS_LAST_STATUS NOT IN ('solved', 'closed', 'deleted')
            AND TICKET_EVENTS_CHANNEL_STAT = 'z_kan_e01'
            AND TICKET_EVENTS_LAST_AREA LIKE 'z_%'
            AND TICKET_EVENTS_DATE <= :TARGET_DATE
        GROUP BY TICKET_EVENTS_LAST_AREA
    ) AS src
    WHERE tgt.DATE = :TARGET_DATE
      AND tgt.AREA = src.AREA;

    RETURN 'Отчетная таблица обновлена успешно.';
END;
$$;
```

Приложение Д

Реализация оркестрации ETL-процедур

Создание единой процедуры MASTER_ALL_PROCEDURES

```
CREATE OR REPLACE PROCEDURE CC_DEV.PRES.MASTER_ALL_PROCEDURES()
RETURNS STRING
LANGUAGE SQL
AS
$$
BEGIN
    CALL CC_DEV.RAW.DOWNLOAD_ZENDESK_TICKET_EVENTS_JSON();
    CALL CC_DEV.INT.DOWNLOAD_ZENDESK_TICKET_EVENTS();
    CALL CC_DEV.INT.DOWNLOAD_ZENDESK_TICKET_EVENTS_STAT();
    CALL CC_DEV.PRES.UPDATE_STAT();
END;
$$;
```

Создание задания на запуск единой процедуры DAILY_ALL_PROCEDURES

```
CREATE OR REPLACE TASK DAILY_ALL_PROCEDURES
  WAREHOUSE = CC_COMPUTE
  SCHEDULE = 'USING CRON 01 01 * * * Europe/Moscow'
AS
  CALL master_all_procedures();

ALTER TASK daily_all_procedures RESUME;
```

Приложение Е

Примеры визуализации данных

Введите дату или месяц (например, декабрь)

30.12.2024 Пн

Дневной обзор

Цель	**Очередь на начало дня**	Новые сообщения	Получено от других команд	Вновь открытые сообщения	**Итого обработать**	Обработано сообщений	Направлено в другие команды	**Очередь на конец дня**	Изменение очереди
Европа	**68**	75	1	10	**154**	77	4	73	5
Африка	**90**	72	7	6	**175**	105	5	65	-25
Азия	**78**	32	3	5	**118**	35	3	80	2
Итого	**236**	**179**	**11**	**21**	**447**	**217**	**12**	**218**	**-18**

Рисунок Е.1 – Дневной обзор и аналитические диаграммы

Итого по предприятию

Дата		Очередь на начало дня	Новые сообщения	Получено от других команд	Вновь открытые сообщения	Итого обработать	Обработано сообщений	Направлено в другие команды	Очередь на конец дня	Изменение очереди
Ноябрь		**0**	**4 451**	**301**	**541**	**5 293**	**4 831**	**257**	**205**	**205**
01.12.2024	Вс	205	79	0	0	284	0	0	284	79
02.12.2024	**Пн**	**284**	**177**	**16**	**24**	**501**	**223**	**6**	**272**	**-12**
03.12.2024	Вт	272	173	4	19	468	199	9	260	-12
04.12.2024	Ср	260	166	14	22	462	217	11	234	-26
05.12.2024	Чт	234	157	16	18	425	209	12	204	-30
06.12.2024	Пт	204	158	10	24	396	214	4	178	-26
07.12.2024	Сб	178	100	9	22	309	77	6	226	48
08.12.2024	Вс	226	58	0	0	284	0	0	284	58
09.12.2024	**Пн**	**284**	**175**	**12**	**27**	**498**	**205**	**10**	**283**	**-1**
10.12.2024	Вт	283	159	14	18	474	204	9	261	-22
11.12.2024	Ср	261	161	2	24	448	210	8	230	-31
12.12.2024	Чт	230	170	7	24	431	200	11	220	-10
13.12.2024	Пт	220	148	6	24	398	206	8	184	-36
14.12.2024	Сб	184	90	11	24	309	68	11	230	46
15.12.2024	Вс	230	53	0	0	283	0	0	283	53
16.12.2024	**Пн**	**283**	**175**	**15**	**22**	**495**	**214**	**13**	**268**	**-15**
17.12.2024	Вт	268	142	8	17	435	213	13	209	-59
18.12.2024	Ср	209	161	12	18	400	203	10	187	-22
19.12.2024	Чт	187	158	5	18	368	197	12	159	-28
20.12.2024	Пт	159	158	15	17	349	155	7	187	28
21.12.2024	Сб	187	94	14	23	318	85	7	226	39
22.12.2024	Вс	226	62	0	0	288	0	0	288	62
23.12.2024	**Пн**	**288**	**186**	**4**	**14**	**492**	**201**	**11**	**280**	**-8**
24.12.2024	Вт	280	149	10	22	461	224	13	224	-56
25.12.2024	Ср	224	145	5	17	391	191	6	194	-30
26.12.2024	Чт	194	171	8	22	395	209	8	178	-16
27.12.2024	Пт	178	149	15	14	356	222	10	124	-54
28.12.2024	Сб	124	104	6	21	255	68	12	175	51
29.12.2024	Вс	175	61	0	0	236	0	0	236	61
30.12.2024	**Пн**	**236**	**179**	**11**	**21**	**447**	**217**	**12**	**218**	**-18**
31.12.2024	Вт	218	155	11	21	405	216	8	181	-37
Декабрь		**205**	**4 273**	**260**	**537**	**5 275**	**4 847**	**247**	**181**	**-24**
Год		**0**	**8 724**	**561**	**1 078**	**10 363**	**9 678**	**504**	**181**	**181**

Рисунок Е.2 – Данные по всему предприятию за декабрь 2024 года

Европа Дата	Очередь на начало дня	Новые сообщения	Получено от других команд	Вновь открытые сообщения	Итого обработать	Обработано сообщений	Направлено в другие команды	Очередь на конец дня	Изменение очереди
Ноябрь	**0**	**1 332**	**131**	**199**	**1 662**	**1 517**	**87**	**58**	**58**
01.12.2024 Вс	58	11	0	0	**69**	0	0	69	11
02.12.2024 Пн	**69**	**69**	**7**	**10**	**155**	**74**	**3**	**78**	**9**
03.12.2024 Вт	78	51	3	6	**138**	66	1	71	-7
04.12.2024 Ср	71	55	4	9	**139**	75	2	62	-9
05.12.2024 Чт	62	42	6	6	**116**	77	2	37	-25
06.12.2024 Пт	37	47	7	9	**100**	76	1	23	-14
07.12.2024 Сб	23	33	5	11	**72**	21	3	48	25
08.12.2024 Вс	48	18	0	0	**66**	0	0	66	18
09.12.2024 Пн	**66**	**61**	**4**	**10**	**141**	**66**	**3**	**72**	**6**
10.12.2024 Вт	72	40	6	6	**124**	67	4	53	-19
11.12.2024 Ср	53	55	1	7	**116**	67	2	47	-6
12.12.2024 Чт	47	58	1	5	**111**	69	5	37	-10
13.12.2024 Пт	37	39	2	10	**88**	70	1	17	-20
14.12.2024 Сб	17	25	6	7	**55**	15	3	37	20
15.12.2024 Вс	37	17	0	0	**54**	0	0	54	17
16.12.2024 Пн	**54**	**63**	**7**	**10**	**134**	**69**	**5**	**60**	**6**
17.12.2024 Вт	60	38	5	6	**109**	67	5	37	-23
18.12.2024 Ср	37	51	5	7	**100**	65	3	32	-5
19.12.2024 Чт	32	52	0	9	**93**	67	3	23	-9
20.12.2024 Пт	23	41	6	6	**76**	15	4	57	34
21.12.2024 Сб	57	22	6	11	**96**	17	4	75	18
22.12.2024 Вс	75	14	0	0	**89**	0	0	89	14
23.12.2024 Пн	**89**	**71**	**1**	**5**	**166**	**66**	**3**	**97**	**8**
24.12.2024 Вт	97	43	3	7	**150**	70	4	76	-21
25.12.2024 Ср	76	37	1	6	**120**	63	1	56	-20
26.12.2024 Чт	56	47	3	11	**117**	70	3	44	-12
27.12.2024 Пт	44	38	5	7	**94**	71	1	22	-22
28.12.2024 Сб	22	35	3	5	**65**	13	5	47	25
29.12.2024 Вс	47	21	0	0	**68**	0	0	68	21
30.12.2024 Пн	**68**	**75**	**1**	**10**	**154**	**77**	**4**	**73**	**5**
31.12.2024 Вт	73	41	0	10	**124**	70	5	49	-24
Декабрь	**58**	**1 310**	**98**	**206**	**1 672**	**1 543**	**80**	**49**	**-9**
Год	**0**	**2 642**	**229**	**405**	**3 276**	**3 060**	**167**	**49**	**49**

Рисунок Е.3 – Данные по цели путешествий Европа за декабрь 2024 года

Африка Дата		Очередь на начало дня	Новые сообщения	Получено от других команд	Вновь открытые сообщения	Итого обработать	Обработано сообщений	Направлено в другие команды	Очередь на конец дня	Изменение очереди
Ноябрь		**0**	**2 301**	**141**	**230**	**2 672**	**2 460**	**119**	**93**	**93**
01.12.2024	Вс	93	35	0	0	128	0	0	128	35
02.12.2024	**Пн**	**128**	**82**	**6**	**10**	**226**	**109**	**1**	**116**	**-12**
03.12.2024	Вт	116	89	1	5	211	97	4	110	-6
04.12.2024	Ср	110	79	8	7	204	102	5	97	-13
05.12.2024	Чт	97	85	8	10	200	92	7	101	4
06.12.2024	Пт	101	82	3	9	195	104	3	88	-13
07.12.2024	Сб	88	54	1	6	149	39	1	109	21
08.12.2024	Вс	109	31	0	0	140	0	0	140	31
09.12.2024	**Пн**	**140**	**81**	**6**	**12**	**239**	**102**	**7**	**130**	**-10**
10.12.2024	Вт	130	89	6	8	233	101	4	128	-2
11.12.2024	Ср	128	71	1	12	212	106	2	104	-24
12.12.2024	Чт	104	85	6	11	206	95	2	109	5
13.12.2024	Пт	109	75	4	12	200	95	5	100	-9
14.12.2024	Сб	100	48	3	10	161	42	4	115	15
15.12.2024	Вс	115	29	0	0	144	0	0	144	29
16.12.2024	**Пн**	**144**	**88**	**8**	**6**	**246**	**107**	**4**	**135**	**-9**
17.12.2024	Вт	135	79	1	7	222	111	6	105	-30
18.12.2024	Ср	105	81	6	7	199	99	3	97	-8
19.12.2024	Чт	97	72	5	6	180	98	6	76	-21
20.12.2024	Пт	76	82	7	7	172	101	3	68	-8
21.12.2024	Сб	68	57	7	10	142	55	2	85	17
22.12.2024	Вс	85	37	0	0	122	0	0	122	37
23.12.2024	**Пн**	**122**	**84**	**3**	**5**	**214**	**102**	**4**	**108**	**-14**
24.12.2024	Вт	108	71	5	10	194	115	7	72	-36
25.12.2024	Ср	72	80	4	8	164	92	4	68	-4
26.12.2024	Чт	68	89	5	9	171	103	4	64	-4
27.12.2024	Пт	64	86	7	5	162	114	7	41	-23
28.12.2024	Сб	41	51	2	9	103	39	7	57	16
29.12.2024	Вс	57	33	0	0	90	0	0	90	33
30.12.2024	**Пн**	**90**	**72**	**7**	**6**	**175**	**105**	**5**	**65**	**-25**
31.12.2024	Вт	65	87	8	8	168	109	3	56	-9
Декабрь		**93**	**2 164**	**128**	**215**	**2 600**	**2 434**	**110**	**56**	**-37**
Год		**0**	**4 465**	**269**	**445**	**5 179**	**4 894**	**229**	**56**	**56**

Рисунок Е.4 – Данные по цели путешествий Африка за декабрь 2024 года

Азия Дата	Очередь на начало дня	Новые сообщения	Получено от других команд	Вновь открытые сообщения	Итого обработать	Обработано сообщений	Направлено в другие команды	Очередь на конец дня	Изменение очереди
Ноябрь	**0**	**818**	**29**	**112**	**959**	**854**	**51**	**54**	**54**
01.12.2024 Вс	54	33	0	0	87	0	0	87	33
02.12.2024 Пн	**87**	**26**	**3**	**4**	**120**	**40**	**2**	**78**	**-9**
03.12.2024 Вт	78	33	0	8	119	36	4	79	1
04.12.2024 Ср	79	32	2	6	119	40	4	75	-4
05.12.2024 Чт	75	30	2	2	109	40	3	66	-9
06.12.2024 Пт	66	29	0	6	101	34	0	67	1
07.12.2024 Сб	67	13	3	5	88	17	2	69	2
08.12.2024 Вс	69	9	0	0	78	0	0	78	9
09.12.2024 Пн	**78**	**33**	**2**	**5**	**118**	**37**	**0**	**81**	**3**
10.12.2024 Вт	81	30	2	4	117	36	1	80	-1
11.12.2024 Ср	80	35	0	5	120	37	4	79	-1
12.12.2024 Чт	79	27	0	8	114	36	4	74	-5
13.12.2024 Пт	74	34	0	2	110	41	2	67	-7
14.12.2024 Сб	67	17	2	7	93	11	4	78	11
15.12.2024 Вс	78	7	0	0	85	0	0	85	7
16.12.2024 Пн	**85**	**24**	**0**	**6**	**115**	**38**	**4**	**73**	**-12**
17.12.2024 Вт	73	25	2	4	104	35	2	67	-6
18.12.2024 Ср	67	29	1	4	101	39	4	58	-9
19.12.2024 Чт	58	34	0	3	95	32	3	60	2
20.12.2024 Пт	60	35	2	4	101	39	0	62	2
21.12.2024 Сб	62	15	1	2	80	13	1	66	4
22.12.2024 Вс	66	11	0	0	77	0	0	77	11
23.12.2024 Пн	**77**	**31**	**0**	**4**	**112**	**33**	**4**	**75**	**-2**
24.12.2024 Вт	75	35	2	5	117	39	2	76	1
25.12.2024 Ср	76	28	0	3	107	36	1	70	-6
26.12.2024 Чт	70	35	0	2	107	36	1	70	0
27.12.2024 Пт	70	25	3	2	100	37	2	61	-9
28.12.2024 Сб	61	18	1	7	87	16	0	71	10
29.12.2024 Вс	71	7	0	0	78	0	0	78	7
30.12.2024 Пн	**78**	**32**	**3**	**5**	**118**	**35**	**3**	**80**	**2**
31.12.2024 Вт	80	27	3	3	113	37	0	76	-4
Декабрь	**54**	**799**	**34**	**116**	**1 003**	**870**	**57**	**76**	**22**
Год	**0**	**1 617**	**63**	**228**	**1 908**	**1 724**	**108**	**76**	**76**

Рисунок Е.5 – Данные по цели путешествий Азия за декабрь 2024 года

Приложение Ж

SQL-запросы для проверки корректности отчётности

Получение списка решенных тикетов

```
SELECT
    TICKET_EVENTS_TICKET_ID AS TICKET_ID
FROM CC_DEV.INT.Z_TICKET_EVENTS
WHERE
    TICKET_EVENTS_CHANNEL_STAT = 'z_ch_e01'
    AND TICKET_EVENTS_STATUS = 'solved'
    AND TICKET_EVENTS_PREVIOUS_STATUS IS NOT NULL
    AND TICKET_EVENTS_AREA_STAT LIKE 'z_%'
    AND TICKET_EVENTS_DATE = '2024-12-07';
```

Получение списка тикетов в очереди

```
SELECT
    DISTINCT TICKET_EVENTS_TICKET_ID AS TICKET_ID
FROM CC_DEV.INT.Z_TICKET_EVENTS
WHERE
    TICKET_EVENTS_LAST_STATUS NOT IN ('solved', 'closed', 'deleted')
    AND TICKET_EVENTS_CHANNEL_STAT = 'z_kan_e01'
    AND TICKET_EVENTS_LAST_AREA LIKE 'z_%'
    AND TICKET_EVENTS_DATE <= '2024-12-07';
```