

Министерство образования Республики Беларусь

Учреждение образования  
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ  
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра информатики

Дисциплина: Модели данных и системы управления базами данных

*К защите допустить:*

И.О. Заведующего кафедрой  
информатики

\_\_\_\_\_ С. И. Сиротко

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту

на тему

**ПРОЕКТИРОВАНИЕ БАЗ ДАННЫХ ДЛЯ ИНФОРМАЦИОННОЙ  
СИСТЕМЫ ДЛЯ УПРАВЛЕНИЯ СКЛАДСКИМИ ОСТАТКАМИ,  
ДОКУМЕНТООБОРОТОМ И ПРОЗРАЧНОСТЬЮ ЦЕПОЧКИ  
ПОСТАВОК**

БГУИР КП 1-40 04 01 025 ПЗ

Студент

К. Ю. Фроленко

Руководитель

Е. А. Кожемяко

Минск 2025

# СОДЕРЖАНИЕ

ВВЕДЕНИЕ .....	5
1 АНАЛИЗ СУЩЕСТВУЮЩИХ АНАЛОГОВ И ПОСТАНОВКА ЗАДАЧИ. 7	
1.1 Обзор средств разработки программного средства .....	7
1.2 Платформа разработки программного средства .....	8
1.3 Современные системы управления базами данных .....	9
1.4 Анализ существующих аналогов.....	11
1.5 Постановка задачи и формирование требований .....	13
1.6 Заключение .....	16
2 АНАЛИЗ ТРЕБОВАНИЙ К ПРОГРАММНОМУ СРЕДСТВУ И РАЗРАБОТКА ФУНКЦИОНАЛЬНЫХ ТРЕБОВАНИЙ .....	17
2.1 Функциональные требования .....	17
2.2 Пользователи и доступ по операциям.....	18
2.3 Декларативные правила целостности и обработки данных .....	19
2.4 Входные и выходные данные, параметры отчётности.....	20
2.5 Заключение .....	22
3 ИНФОЛОГИЧЕСКАЯ МОДЕЛЬ ПРЕДМЕТНОЙ ОБЛАСТИ .....	23
3.1 Границы модели и состав сущностей .....	23
3.2 Атрибуты и ключи: справочники и документы .....	24
3.3 Связи, кратности и каскады изменений.....	26
3.4 Нормализация и обоснованные исключения .....	28
3.5 Заключение .....	30
4 ФИЗИЧЕСКАЯ МОДЕЛЬ И РЕАЛИЗАЦИЯ В POSTGRESQL.....	31
4.1 Принципы физического проектирования, домены и перечисления .....	31
4.2 Схема auth_db (пользователи, устройства, склады, права).....	33
4.3 Схема mobile_app (товары, партии/серии, документы, строки, движения).....	34
4.4 Схема report_db (параметры, шаблоны, кэш отчётов) .....	35
4.5 Индексы, ограничения целостности и пути доступа.....	36
4.6 Заключение .....	37
5 ТЕСТИРОВАНИЕ И ВЕРИФИКАЦИЯ БАЗЫ ДАННЫХ .....	38
5.1 Тестовый стенд и исходные данные .....	38
5.2 Сценарии интеграционного тестирования (сквозные).....	39
5.3 Нагрузочные и конкурентные испытания .....	41
5.4 Автоматизация проверок и контроль регрессий.....	42
5.5 Заключение .....	43
ЗАКЛЮЧЕНИЕ .....	45
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ .....	46

## ВВЕДЕНИЕ

В условиях современной цифровой экономики предприятия розничной и оптовой торговли ежедневно выполняют десятки операций, связанных с поступлением, реализацией, перемещением и списанием товарно-материальных ценностей. Каждая операция сопровождается документами, партиями и сериями, скидками, расчётами с клиентами и партнёрами, а также множеством отчётов и аналитических выборок. Ошибки или рассогласования в данных напрямую отражаются на остатках, выручке и налоговых показателях, поэтому качество модели данных и корректность её реализации в системе управления базами данных (СУБД) становятся критическим фактором эффективности бизнеса.

Повышение прозрачности учёта, автоматизация документооборота и контроль движения партий особенно актуальны в условиях обязательной маркировки товаров и прослеживаемости цепочки поставок. Современные предприятия нуждаются в архитектуре, обеспечивающей не только надёжное хранение данных, но и гибкость при интеграции с аналитическими и облачными сервисами.

Данный курсовой проект посвящён проектированию реляционной базы данных для информационной системы управления складскими остатками, документооборотом и аналитической отчётностью. Акцент сделан на структуре данных, механизмах обеспечения целостности и поддержке типовых сценариев учёта средствами СУБД, без избыточной зависимости от прикладного кода или интерфейсов.

Особенностью предлагаемого решения является логическое разделение данных на три взаимосвязанные базы:

1 `auth_db` – пользователи, фирмы, склады, устройства, настройки и права доступа.

2 `mobile_app` – товары, группы, производители и страны, партии/серии, документы разных типов (приход, расход, перемещение, прочие), состояния документов, продажи, оплаты, клиенты, скидки и маркировка.

3 `report_db` – параметры и шаблоны отчётов, кэш вычисленных данных для ускорения аналитики.

Такое разделение снижает взаимное влияние транзакций и отчётных запросов, повышает отказоустойчивость и упрощает сопровождение системы.

Разграничение доступа реализуется через битовые маски прав, привязанные к наборам операций сервисов. Хранение прав в компактном виде и проверка их на стороне сервисов обеспечивают гибкую настройку ролей и минимизируют риски несанкционированных изменений.

На уровне базы данных целостность поддерживается реляционными средствами: внешними ключами, ограничениями (`CHECK`, `UNIQUE`), перечислимыми типами для фиксированных доменов (вид документа, состояние, тип операции), а также триггерами и хранимыми процедурами для автоматизации повторяющихся действий (например, контроль переходов

состояний, пересчёт итогов документа при изменении строк, синхронизация резервов и остатков).

В качестве СУБД выбрана PostgreSQL, поскольку она сочетает строгую транзакционность, выразительные типы данных и развитые механизмы серверной логики. Это позволяет закрепить ключевые инварианты непосредственно в хранилище, обеспечив надёжность и воспроизводимость данных. Сервер приложений (FastAPI) и веб-интерфейс (Next.js/React) используются лишь для взаимодействия с БД, а центральной задачей проекта остаётся моделирование и реализация её структуры.

Цель работы – разработать структуру реляционной базы данных, обеспечивающую корректный учёт складских операций и документов, защищённый доступ пользователей и эффективную поддержку аналитической отчётности.

Для достижения цели решаются следующие задачи:

1 Выполнить анализ предметной области и существующих решений; обосновать выбор СУБД и подходов к моделированию данных.

2 Сформировать требования к базе данных: назначение, функции, состав входных и выходных данных, надёжность, совместимость.

3 Разработать инфологическую модель предметной области (сущности, атрибуты, связи) и на её основе – физическую модель в PostgreSQL.

4 Реализовать структуру БД: таблицы, ключи, индексы, перечислимые типы, триггеры и хранимые процедуры, обеспечивающие целостность и автоматизацию.

5 Проверить корректность работы на примерах типовых операций и отчётов (остатки, движения, продажи, выручка).

Практическая значимость проекта заключается в создании базы данных, которая:

- обеспечивает прозрачный и воспроизводимый учёт товарных операций;

- исключает расхождения между документами и остатками за счёт встроенных ограничений;

- поддерживает управляемую и расширяемую модель доступа пользователей;

- формирует надёжную основу для построения аналитических отчётов и принятия управленческих решений.

# **1 АНАЛИЗ СУЩЕСТВУЮЩИХ АНАЛОГОВ И ПОСТАНОВКА ЗАДАЧИ**

В разделе выполняется систематизированный анализ предметной области управления складскими остатками и документооборотом, а также уточняются требования к базе данных разрабатываемой информационной системы. Рассматриваются применяемые подходы к моделированию данных и средства их реализации в реляционной СУБД, формулируются принципы обеспечения целостности, согласованности и воспроизводимости учёта. Полученные положения служат основанием для последующего проектирования инфологической и физической моделей базы данных.

## **1.1 Обзор средств разработки программного средства**

Проектирование хранилища выполняется по принципу фиксации инвариантов на уровне базы данных. Допустимые состояния документов, корректность ссылок между сущностями, уникальность бизнес-ключей и непротиворечивость количественно-стоимостных показателей обеспечиваются не только прикладным кодом, но и средствами самой СУБД. Такой подход уменьшает вероятность рассинхронизаций при параллельной работе пользователей и обеспечивает воспроизводимость результатов отчётности.

Инфологическая модель задаётся в нотации «сущность–связь» и разделяет данные на справочники (товар, группа, производитель, страна, контрагент, склад) и операционные сущности (партии/серии, документы и их строки, продажи, оплаты). Нормализация данных до третьей нормальной формы (3НФ) устраняет дублирование и аномалии: количественно-ценовые признаки закреплены на уровне партии; строки документа не дублируют реквизиты «шапки»; операционные таблицы содержат только ссылки на справочники. Фиксированные домены (вид/состояние документа, тип операции, тип товара) определяются перечислимыми типами, что предотвращает недопустимые значения и упрощает контроль переходов состояний.

Целостность реализуется через внешние ключи (FK), NOT NULL/UNIQUE/CHECK и серверную логику. FK исключают «висящие» ссылки и управляют удалением связанных записей; CHECK запрещают отрицательные количества и некорректные даты; перечислимые типы фиксируют допустимые статусы и операции. Операции, затрагивающие несколько таблиц (проведение/откат, пересчёт итогов/резервов), выполняются процедурами и триггерами в одной транзакции. Под «проведением» понимается создание движений и фиксация состояния «проведён»; под «откатом» – атомарное удаление созданных движений и возврат в «черновик». Индексы создаются на основных путях доступа (FK-поля, даты, признаки

фильтрации), чтобы ускорить списки документов, расчёт остатков и формирование отчётов.

Логическое разделение реализовано на трёх БД: `auth_db` – пользователи, фирмы, склады, устройства и матрица прав (битовые маски операций); `mobile_app` – документы всех типов/состояний, партии/серии, продажи и оплаты, клиенты и скидки (здесь сосредоточены основные ограничения и серверная логика учёта); `report_db` – параметры/шаблоны и кэш отчётов для тяжёлых выборок без влияния на транзакции.

В качестве СУБД используется PostgreSQL. Система поддерживает транзакционность, строгую типизацию (включая перечислимые типы и массивы), внешние ключи, уникальные и проверочные ограничения, триггеры и хранимые процедуры. Это даёт возможность формализовать правила предметной области непосредственно в схеме. Полуструктурированные данные применяются точечно (например, хранение карты прав в формате JSONB), при этом критичные для учёта сущности и связи реализуются в реляционном виде. Согласование схемы с прикладным кодом обеспечивается миграциями, что позволяет эволюционировать структуру таблиц контролируемо и воспроизводимо.

Таким образом, основу разработки составляют: инфологическая модель с нормализованными сущностями и связями; строгая типизация доменов; система ограничений, индексов, триггеров и процедур; а также логическое разделение на авторизационный, операционный и отчётный контуры. Указанные средства ориентированы на повышение надёжности учёта, снижение числа ошибок при параллельной работе и создание устойчивой базы для аналитической отчётности.

## **1.2 Платформа разработки программного средства**

В качестве серверной платформы используется Ubuntu LTS [1]. Такой выбор обусловлен предсказуемостью обновлений, наличием готовых пакетов для PostgreSQL и инструментов автоматизации, а также удобством сопровождения в типовой инфраструктуре предприятий. Сама прикладная логика развёрнута отдельно, однако в курсовом проекте платформа описывается ровно в той мере, в какой она влияет на проектирование и эксплуатацию базы данных.

Основу хранилища составляет PostgreSQL. Развёртывание организовано контейнерным способом: для каждого контура создаётся отдельный экземпляр СУБД – авторизационный (`auth_db`), операционный (`mobile_app`) и отчётный (`report_db`). Изоляция контуров снижает взаимное влияние транзакционных операций и аналитических выборок, упрощает резервное копирование и позволяет независимо обновлять схемы. Для инициализации схем при запуске применяются миграции Alembic: контейнер с миграциями подключается к целевой БД по строке подключения и последовательно поднимает структуру до актуальной версии [2]. Такой сценарий гарантирует воспроизводимость: одна и та же версия кода соответствует одной и той же версии схемы.

Заполнение неизменяемых справочников и «служебных» таблиц (типы документов и их подтипы, состояния, типы операций, базовые настройки) выполняется отдельным этапом – либо через Alembic-миграции данных, либо через начальные SQL-скрипты. Это позволяет отделить изменения структуры от изменений содержимого и контролировать обе части независимо. Для операционного контура, где в реальной эксплуатации у каждой фирмы своя база, предусмотрена процедура «тиражирования»: новая БД создаётся из шаблона (пустая структура + начальные справочники) и сразу проходит тот же набор миграций, что и базовый экземпляр.

Минимальные требования безопасности закладываются на уровне ролей СУБД. Административная роль используется только миграторами; прикладные сервисы подключаются под отдельными ролями с правами, ограниченными конкретной схемой и типами операций (чтение в отчётном контуре, чтение/запись – в операционном, ограниченный доступ в авторизационном). Секреты подключения передаются через переменные окружения контейнеров и не входят в состав исходного кода. Такой подход согласуется с моделью прав в приложении: проверки доступа выполняются в сервисах, а на уровне БД предотвращается выполнение операций вне предусмотренных границ.

Резервное копирование организуется на уровне каждой БД отдельно. Для структурной консистентности и переносимости используются логические дампы (`pg_dump`), восстановление проверяется на стенде с идентичной версией PostgreSQL и теми же миграциями. Разделение по контурам упрощает процедуру: отчётную БД можно копировать и восстанавливать независимо от операционной, не прерывая работу ввода документов.

Для разработчика и тестирования важна воспроизводимость окружения. Контейнеры СУБД поднимаются локально, затем выполняются миграции, загружаются минимальные тестовые данные (товары, склады, партнёры, один-два документа каждого типа). Это позволяет запускать интеграционные сценарии (создание документа, изменение строк, проведение, откат, формирование отчёта по остаткам) без привязки к конкретной машине и гарантирует, что ошибки схемы или триггеров проявятся одинаково у всех участников.

Таким образом, платформа ограничивается необходимым минимумом: Ubuntu LTS как надёжная серверная среда, отдельные экземпляры PostgreSQL для трёх контуров, миграции Alembic для версионирования схемы, начальные скрипты для справочников, ролевая модель доступа и независимое резервное копирование. Всё это служит единственной цели – обеспечить стабильную, воспроизводимую и контролируемую эксплуатацию реляционной модели данных, которая является предметом данного курсового проекта.

### **1.3 Современные системы управления базами данных**

Для проекта складского учёта и документооборота решающими являются три свойства: строгая целостность данных (без «висящих» ссылок и

противоречивых остатков), предсказуемая работа в многопользовательском режиме и удобные средства серверной логики (триггеры/процедуры).

**PostgreSQL.** Реляционная СУБД с полной поддержкой ссылочной целостности, проверочных ограничений и перечислимых типов. Удобно задавать фиксированные домены (например, вид документа или его состояние), контролировать переходы между ними, хранить операционные данные в нормализованном виде и при необходимости – точно использовать JSONB [3]. Индексы можно строить по выражениям и с условиями отбора, что помогает ускорять типичные выборки (остатки на дату, документы по складу и партнёру) без усложнения схемы. Триггеры и хранимые процедуры позволяют реализовать пересчёт итогов и резервов прямо в базе. Распространяется бесплатно, нативно работает в Linux, хорошо сочетается с принятым разделением на три базы (auth\_db, mobile\_app, report\_db).

**MySQL/MariaDB.** Подходит для веб-приложений с простыми транзакциями, но при сложных ограничениях обычно требует больше логики на стороне приложения. Проверочные ограничения и строгая типизация реализованы скромнее, возможности частных индексов и индексов по выражениям ограничены, перечислимые типы менее удобны. Для задач, где важны сложные связи и контроль состояний документов, это приводит к большому количеству «обходных» решений и повышает риск рассинхронизаций.

**Microsoft SQL Server** функционально силён: триггеры, процедуры, средства аналитики и оптимизатор запросов реализованы на высоком уровне. Однако это коммерческий продукт с высокой стоимостью владения и ориентацией на Windows-инфраструктуру. В учебном курсовом и в планируемой Linux-платформе его преимущества нивелируются затратами и ограничениями по окружению.

**Oracle Database.** Профессиональный стандарт для крупных корпоративных систем с жёсткими требованиями к масштабированию. Как и в случае SQL Server, цена и сложность администрирования избыточны для задач среднего масштаба и для учебного проекта, ориентированного на открытый стек.

**MongoDB** (документоориентированная). Удобна для «гибких» схем и быстрых записей, но по природе не навязывает строгой реляционной целостности. Даже с поддержкой транзакций контроль взаимосвязей партий, серий, документов и остатков приходится переносить в код приложения. Для складского учёта это повышает риск ошибок и усложняет отчётность.

**SQLite.** Лёгкая встраиваемая БД, хороша для одиночных приложений и прототипов, но не рассчитана на интенсивный многопользовательский доступ и разнесённые контуры (операционный и отчётный).

Учитывая предметную область (партии/серии, статусы документов, взаимосвязанные остатки, продажи/оплаты) и учебные требования (триггеры/процедуры), оптимальна PostgreSQL, потому что позволяет фиксировать инварианты в самой схеме: строгие FK и CHECK, перечислимые домены для статусов и операций, процедуры/триггеры для проведения/отката

в одной транзакции, а также точечные индексы по выражениям для отчётных запросов. Открытая лицензия и нативная работа в Linux согласуются с принятой платформой и разделением на `auth_db` / `mobile_app` / `report_db`.

Таким образом, PostgreSQL позволяет зафиксировать ключевые инварианты прямо в схеме, минимизируя зависимость корректности учёта от клиентского кода и упрощая формирование достоверной отчётности.

## **1.4 Анализ существующих аналогов**

В этом разделе фиксируются проверенные и устойчивые решения, применяемые при построении модели данных, заимствованные из распространённых систем учёта, используемых в Российской Федерации (РФ) и Республике Беларусь (РБ): устройство документов и их состояний, фиксация партий и серий, способы обеспечения целостности остатков и принципы построения отчётности. Из этих практик формируются требования к нашим сущностям, связям и ограничениям в PostgreSQL.

**1.4.1 «1С: Управление торговлей» [4].** В системах семейства 1С данные строятся вокруг классической модели «шапка/строки»: «шапка» хранит общие реквизиты (дата, склад, контрагент, комментарии), а строки описывают товарные позиции, количества и цены. Документы проходят чёткие состояния (черновик/проведён/отменён), и только проведённый документ формирует движение запасов. Остатки трактуются как производная от движений, а не как поле, которое можно изменять напрямую. Партии и серийные номера поддерживаются на уровне базы: строка документа при необходимости обязана ссылаться на конкретную партию/серию, что обеспечивает прослеживаемость.

Сильная сторона подхода – дисциплина данных: пока документ не проведён, остаток не меняется; провести документ при нарушении инвариантов нельзя (нет остатка, неверная серия, конфликт дат). Ограничения – закрытая платформа и завязка бизнес-логики на конкретную среду исполнения. Для курсового проекта это означает: модель «шапки/строки», жёсткие состояния, движения как первичный источник остатков и обязательная привязка к партиям/сериям для маркируемой продукции – именно те идеи, которые нужно выразить средствами СУБД.

**1.4.2 «МойСклад» (облачный подход) [5].** У облачных решений для малого и среднего предпринимательства (МСП) структура данных проще и предсказуемее: документы приходов, продаж, перемещений и инвентаризаций; справочники номенклатуры, контрагентов и складов; статусы документов с линейными переходами. Поддерживаются партии (часто по цене и дате), иногда – серийные номера. Важная практическая особенность – разделение транзакций и аналитики: тяжёлые отчёты формируются отдельно и не мешают вводу. При этом непосредственно на

уровне СУБД пользователь управлять ограничениями не может – всё «зашиито» в сервис.

Вышеперечисленное является хорошим ориентиром: модель должна оставаться читаемой (шапки/строки, фиксированные статусы), а отчётная нагрузка не должна конкурировать с транзакционной. Следовательно, операционный контур и кэш отчётов логично разводить в разные базы.

**1.4.3 «Класс365» (SaaS-учёт для торговли).** Практика таких систем – «минимально достаточная» реляционная модель: нормализованные справочники (товары, группы, единицы, склады, контрагенты), документы с предсказуемыми состояниями и вычисление остатков по подтверждённым операциям. Партии поддерживаются комплексно: срок годности и закупочная цена фиксируются на уровне партии, а списание выполняется по заданному правилу (например, FIFO или явный выбор партии). Права назначаются по ролям, доступ ориентируется не только на разделы, но и на конкретные действия.

Вывод для схемы PostgreSQL: атрибуты, влияющие на учёт (тип операции, состояние документа, режим списания, принадлежность к складу/партнёру), должны быть строго типизированы и защищены ограничениями. Партии – самостоятельная сущность с обязательными полями количества, цены и, при необходимости, срока годности.

**1.4.4 «СБИС. Торговля/Склад» (интегрированный контур учёта) [6].** В интегрированных решениях ключевой акцент на согласованность учёта и документооборота: документ хранится и проводится в одном контуре, а отчётность строится поверх транзакций, часто с кэшированием. Если используются марки и серии, то у партии появляется обязательный атрибут срока годности и источник происхождения; операции перемещения и списания проверяют корректность выбора партии и запрет «задним числом» для просроченных лотов.

Для БД это подтверждает необходимость фиксировать критичные атрибуты партии/серии как поля таблицы партии (а не строки документа), хранить резерв, приход и остаток отдельно от «плановых» значений и обеспечивать атомарность проведения/отката документа на уровне процедур.

**1.4.5 Сопоставление и выводы для проектируемой модели.** Несмотря на различия в платформах и лицензировании, зрелые решения на рынке сходятся в нескольких принципах, которые прямо влияют на дизайн схемы данных:

1 Остатки – производная от движений.

Документ не меняет остаток «напрямую»: только состояние «проведён» создаёт запись движения. Следовательно, в БД фиксируются состояния документа перечислимым типом и реализуем процедуры проведения/отката, которые в одной транзакции создают/удаляют нужные движения.

2 Партии и серии – самостоятельные сущности.

Цена, количество, срок годности и другие учётные признаки закрепляются на стороне партии/серии. Строки документов ссылаются не просто на товар, а на конкретную партию/серию, когда это требуется прослеживаемостью.

3 Статусы и типы – строго типизированные домены.

Вид документа, подтип, состояние, вид операции задаются перечислимыми типами; переходы между состояниями ограничиваются проверками. Это исключает «левые» значения и упрощает контроль целостности.

4 Разделение контуров: транзакции и отчёты.

Чтобы тяжёлые выборки не конкурировали с вводом, отчётные шаблоны, параметры и кэш выносятся в отдельную БД. Это позволяет без риска строить агрегаты по движениям и хранить промежуточные результаты графиков/витрин.

5 Права в разрезе операций.

Доступ проверяется не только по разделам, но и по конкретным действиям (создать, провести, откатить, удалить, просматривать остатки/серии и т.п.). Компактное хранение прав в виде битовых масок по сервисам (JSONB) даёт гибкость без усложнения схемы.

6 Проверки на стороне СУБД.

Внешние ключи, NOT NULL и CHECK отсекают невозможные состояния (минусовые количества, «висящие» ссылки, некорректные даты). Критичные бизнес-инварианты (например, запрет проведения при отрицательном остатке или списание просроченной партии) закрепляются триггерами и процедурами.

Эти выводы прямо легли в принятый дизайн на PostgreSQL: нормализованная структура «шапки/строки», явные сущности партий и серий, перечислимые типы для доменов предметной области, процедуры/триггеры для проведения и отката, разнесение `mobile_app` (операции) и `report_db` (аналитика и кэш), а также хранение матрицы прав в `auth_db` с битовыми масками. Такой набор решений обеспечивает и требуемую прозрачность цепочки поставок, и воспроизводимость учёта на любую дату – без опоры на закрытые механизмы конкретной платформы.

## **1.5 Постановка задачи и формирование требований**

Спроектировать и реализовать реляционную модель данных для информационной системы учёта складских остатков и документооборота с прозрачностью движения партий/серий. База должна обеспечивать корректный учёт «шапки/строки» документов, партийно-серийную прослеживаемость, расчёт остатков по складам и кэширование отчётов. Архитектурно данные разделяются на три контура: `auth_db` (пользователи, фирмы, склады, устройства, матрица прав), `mobile_app` (операционные сущности: товары, партии, документы, продажи, оплаты, движения), `report_db` (шаблоны, параметры и кэш отчётов).

Состав выполняемых функций (на уровне БД): регистрация и редактирование документов (поступление, расход/продажа, перемещение, прочие операции) со строгими состояниями; проведение/откат документа с атомарным внесением движений по партиям и складам; учёт партий и серий (цена, количество, срок годности, налоговые/ценовые признаки); поддержка резервирования и списаний; ведение справочников (товары, группы, производители, страны, контрагенты, склады); хранение продаж и оплат с привязкой к устройствам; проверка ссылочной целостности и допустимых значений через домены и ограничения; хранение матрицы прав доступа в `auth_db` (битовые маски по сервисам) с валидацией данных; кэширование результатов отчётов и параметров построения в `report_db`.

Для фиксации инвариантов используются процедуры `conduct_document()` и `rollback_document()`, выполняющие проведение и откат документа в одной транзакции; триггеры контролируют допустимые переходы состояний и пересчёт итогов/резервов.

Если говорить о входных данных, то в операционный контур поступают: карточки номенклатуры (наименование, группа, производитель, страна, EAN, тип «товар/услуга»), склады и магазины, контрагенты, документы с датой, складом-источником/получателем и партнёром, строки документов (товар, партия/серия при необходимости, количество, цены и скидки), сведения о партиях/сериях (дата поступления, срок годности, закупочная цена и прочее), продажи и оплаты (устройство, номер чека, сумма, тип оплаты), настройки фирмы. Данные пользователей, устройств и прав доступа вводятся в `auth_db` и используются сервисами авторизации.

Взяв во внимание выходные данные, то с кистема обязана предоставлять срезы и отчёты: остатки по складам на дату, обороты за период (приход/расход/перемещения), движение по документу, отчёт по партиям и срокам годности (включая позиции с истекающим сроком), продажи и выручку по товарам/группам/периодам, а также контроль качества данных (выявление невозможных состояний – отрицательных остатков, «висящих» ссылок, несогласованных статусов). Для обмена с пользовательским интерфейсом и внешними сервисами допускается формирование выгрузок в табличные форматы (CSV/Excel) и структурированные ответы (JSON) поверх REST-API; длительные агрегаты кэшируются в `report_db` [7].

Требования к временным характеристикам: операции проведения и отката документа, а также чтение типовых отчётов на рабочем объёме данных должны выполняться интерактивно и не создавать длительных блокировок. Конкурентное проведение разных документов по разным партиям/складам не должно конфликтовать; при конкуренции за одну и ту же партию корректность обеспечивается транзакциями и уровнем изоляции БД.

Требования к надёжности и целостности: все изменения, затрагивающие несколько таблиц (документы, строки, партии, движения), выполняются в транзакциях; частичные состояния исключены. Ссылочная целостность обеспечивается внешними ключами; значения доменов – перечислимыми типами (вид документа, подтип, состояние, вид операции, тип товара и др.);

числовые поля защищены NOT NULL и CHECK-ограничениями (запрет отрицательных количеств и некорректных дат). Для проведения/отката используются хранимые процедуры и триггеры, предотвращающие несогласованность остатков и двойное проведение.

Условия эксплуатации: многопользовательский доступ с параллельной работой по нескольким складам. Для каждой фирмы используется собственная операционная база `mobile_app` (изоляция данных и независимость миграций); `auth_db` и `report_db` общие для инсталляции. Развёртывание – в серверной среде Linux с контейнеризацией; резервное копирование и восстановление реализуются штатными средствами PostgreSQL.

Если затрагивать требования к составу и параметрам технических и программных средств, то можно выделить СУБД PostgreSQL (версии семейства 15), сервер приложений на FastAPI (Python 3.x), ORM SQLAlchemy и миграции Alembic; для администрирования – DataGrip или аналог. Развёртывание компонентов в Docker-контейнерах [8]. Пользовательский интерфейс (Next.js/React) взаимодействует через REST-API; состав фронтенда не влияет на логическую схему БД [9].

Требования к информационной и программной совместимости: единое кодирование (UTF-8), согласованные форматы дат/времени, фиксированные идентификаторы сущностей на уровне БД (PK/UK), стабильные представления для отчётных выборок. Принятые перечислимые типы и структура прав доступа (битовые маски по сервисам) документируются и версионизируются вместе со схемой.

Обосновать выбор языка и средств можно тем, что PostgreSQL выбран за поддержку транзакций, строгие ограничения целостности, перечислимые типы, триггеры и процедуры; Python/FastAPI – за быстрый REST-контур и удобную типизацию; разделение `auth_db/mobile_app/report_db` позволяет изолировать операции и аналитику и упростить масштабирование.

В рамках курсового проекта основное внимание уделяется реализации базовых механизмов учёта и обеспечения целостности данных. За пределами рассматриваемого объёма остаются интеграция с внешними системами электронного документооборота (ЭДО) и контрольно-кассовой техникой (ККТ), а также сложные финансовые регистры, механизмы бюджетирования и сквозная логистика вне склада. Реализуемая часть системы включает базовые документы, учёт партий и серий, операции продаж и оплат, права доступа, ключевые отчёты, а также демонстрационные триггеры и процедуры, обеспечивающие проведение и откат документов.

Критериями приёмки результата являются: наличие полной ER-модели и описания DDL (таблицы, типы, индексы, внешние ключи и ограничения), корректная работа реализованных доменов, триггеров и процедур, успешное выполнение проверочных запросов (остатки на дату, движение по документу, отчёты по партиям и срокам годности), отсутствие «висящих» ссылок и отрицательных остатков на тестовых сценариях, а также функционирующий механизм кэширования отчётов в `report_db`.

## 1.6 Заключение

В ходе анализа сформирована единая позиция по предметной области и роли базы данных в системе учёта. Проанализированы распространённые в РФ и РБ решения и зафиксированы практики, доказавшие устойчивость: документы изменяют остатки только через движения; партии и серии представляют собой самостоятельные сущности с обязательными атрибутами; статусы и типы операций заданы фиксированными доменами; транзакционный контур отделён от отчётного.

С учётом этих выводов обоснован выбор PostgreSQL как основной СУБД и принятие архитектурного разделения данных на три взаимосвязанных контура: `auth_db` (пользователи, устройства, права), `mobile_app` (операционные сущности, документы и движения) и `report_db` (шаблоны и кэш отчётов). Такой подход обеспечивает изоляцию ответственности, предсказуемую производительность и воспроизводимость учёта на любую дату.

Сформулированы требования к системе на уровне данных: определён состав функций базы данных, описаны входные и выходные данные, установлены ограничения целостности, требования к надёжности и условия эксплуатации. Отдельно подчеркнута необходимость реализовать критичные бизнес-инварианты средствами самой СУБД (перечислимые типы, внешние ключи, CHECK-ограничения, триггеры и процедуры проведения/отката), а проверку доступа – через компактные битовые маски прав.

Проведённый анализ и сформулированные требования создают основу для следующего этапа – проектирования инфологической и физической моделей базы данных, которые обеспечат реализацию заявленных принципов целостности, согласованности и прозрачности учёта в рамках выбранной архитектуры.

## **2 АНАЛИЗ ТРЕБОВАНИЙ К ПРОГРАММНОМУ СРЕДСТВУ И РАЗРАБОТКА ФУНКЦИОНАЛЬНЫХ ТРЕБОВАНИЙ**

Глава формирует конкретные требования к системе на уровне базы данных. Здесь описываются только то, какие операции должны поддерживаться, какие инварианты проверяются в хранилище и какие данные система принимает и выдаёт. На основе результатов анализа из первой главы уточняются допустимые состояния документов, работа с партиями и сериями, правила проведения и отката, а также доступ к операциям через проверку прав. Отдельно фиксируются требования к корректности параллельной работы и ожидаемому времени отклика для типовых сценариев и отчётов. Итогом анализа станет согласованный перечень функций и ограничений, который напрямую ляжет в основу инфологической модели, а затем – физической схемы и серверной логики PostgreSQL.

### **2.1 Функциональные требования**

Система на уровне базы данных должна поддерживать полный цикл учёта товарных операций без опоры на интерфейс. В хранилище фиксируются справочники, документы типа «шапка/строки», партии и серии, продажи и оплаты, а также движения как единственный источник остатков. Допустимые состояния документов задаются определенными типами и проверяются серверной логикой. Проведение и откат выполняются атомарно, с созданием/удалением движений и пересчётом итогов. Для снижения нагрузки отчётные агрегаты кэшируются отдельно.

Краткий перечень требуемых функций:

1 Введение справочников номенклатуры, контрагентов, складов и единиц измерения с проверкой уникальных ключей.

2 Регистрация документов приход/расход/перемещение/прочие в модели «шапка/строки», смена состояний по допустимым переходам.

3 Проведение документа с формированием движений по складам и партиям; откат с обратным эффектом в одной транзакции.

4 Учёт партий и серий (количество, закупочная цена, срок годности и при необходимости признаки маркировки) с обязательной ссылкой из строк, где требуется прослеживаемость.

5 Поддержка резервов и контроль невозможности списания сверх доступного остатка.

6 Хранение продаж и оплат с привязкой к устройству/смене и товарам документа, согласование сумм и статусов.

7 Вычисление остатков на дату и оборотов за период по складам и номенклатуре на основании движений.

8 Кэширование длительных отчётных выборок и параметров отчётов в отдельном контуре данных.

9 Проверка прав доступа на стороне сервисов по битовым маскам операций с валидацией соответствующих данных в `auth_db`.

10 Контроль качества данных: запрет «висящих» ссылок, отрицательных количеств, несогласованных статусов; журналирование базовых атрибутов изменений (кто и когда создал или изменил запись).

Все перечисленные функции реализуются средствами PostgreSQL: определенные типы для доменов предметной области, внешние ключи и проверки, индексы на ключевых путях доступа, триггеры и процедуры `conduct_document()` и `rollback_document()` для проведения и отката [10].

## **2.2 Пользователи и доступ по операциям**

В системе не используются роли. Доступ выдается точно: каждому пользователю назначаются права на конкретные операции (эндпоинты) в разрезе фирмы, при необходимости – склада или устройства. Права хранятся компактно в `auth_db` в виде битовых масок по коду сервиса/операции; проверка выполняется на стороне сервисов до обращения к операционным данным. По умолчанию действует запрет: если бит не установлен – операция недоступна.

Основные сущности и связи в `auth_db`: пользователь, фирма, склад, устройство и таблица назначений доступа, где фиксируются пользователь, фирма, (опционально) склад/устройство, код сервиса и целое число – маска прав. Для каждой связи «пользователь–фирма–код сервиса» действует уникальность; ссылки на склады/устройства допустимы только внутри той же фирмы. Набор допустимых кодов сервисов и разметка битов документируются и версионизируются; в назначениях разрешены только коды из справочника операций.

Правила проверки на стороне сервисов едины: эндпоинт сопоставляется коду операции, выбирается наиболее специфичное назначение (если задан склад/устройство – оно приоритетнее общего по фирме), далее проверяется установлен ли соответствующий бит в маске. При отсутствии подходящего назначения или нужного бита доступ отклоняется. Сама БД не выполняет построчную фильтрацию данных по правам, её задача – хранить корректную матрицу доступа и не допускать противоречивых назначений.

Требования к целостности в `auth_db`: все назначения должны ссылаться на существующие объекты; маски – неотрицательные и не NULL; нельзя создать дубликат назначения для той же комбинации ключей; запрещены назначения на склад/устройство другой фирмы. Для отслеживания изменений в правах каждая запись содержит служебные поля «кто/когда создал и изменил», что позволяет проводить аудит.

Минимальный перечень контролируемых операций, для которых выдаются биты доступа, включает: чтение/изменение справочников; создание/редактирование/удаление документов; проведение и откат документов; просмотр остатков и движений; просмотр партий и серий; операции продаж и оплат; построение отчётов и управление их кэшем.

Изоляция данных обеспечивается самой архитектурой: операционные данные каждой фирмы живут в отдельной БД `mobile_app`, общие пользователи и права – в едином `auth_db`, отчётный кэш – в `report_db`. Благодаря этому даже при ошибке выдачи прав пользователь не получит доступ к данным другой фирмы, а сервисы всегда имеют однозначный источник истины по матрице доступа.

### **2.3 Декларативные правила целостности и обработки данных**

Правила фиксируются в самой схеме, чтобы технически исключить «невозможные» состояния. Ниже – ключевые группы ограничений и то, как они применяются.

1 Документы и их состояния. Тип и состояние документа задаются перечислимыми доменами; допустимы только переходы «черновик – проведён» и «черновик – отменён». Прямое изменение состояния запрещено: проведение и откат выполняются только процедурами. После проведения «шапка» (дата, склад, контрагент) становится неизменяемой, редактируются лишь служебные поля аудита. Номер документа уникален в разрезе фирмы, типа и календарного года. Удаление разрешено только для черновиков.

2 Строки документов. Каждая строка ссылается на свою «шапку» и номенклатуру; количество строки больше нуля, цены и скидки неотрицательны и укладываются в допустимые границы. Для товаров с прослеживаемостью строка обязана ссылаться на конкретную партию и, при необходимости, серию (марку). В приходных документах ссылка на партию необязательна – партия создаётся при проведении; в расходных документах и перемещениях ссылка обязательна и проверяется на остаток.

3 Партии и серии. Партия связывает товар, склад, источник (документ прихода), закупочную цену и срок годности (если применимо). Для каждой партии количество и резерв неотрицательны; срок годности, если задан, не может быть в прошлом на момент проведения прихода. Сериальные и маркировочные коды уникальны в пределах системы и не могут быть списаны повторно.

4 Движения и остатки. Остатки – производная от таблицы движений; прямого редактирования остатков нет. Запись в движения создаётся только процедурами проведения и удаляется только процедурами отката; прямые операции `INSERT/UPDATE/DELETE` в движениях блокируются правами и триггерами. Для каждой записи движения фиксируются дата, склад, товар, партия (если есть), направление и величина. При проведении расхода проверяется, что доступный остаток партии на момент документа не станет отрицательным; при перемещении проверка выполняется для склада-источника. Двойное проведение одного и того же документа предотвращается уникальным ключом «источник движения».

5 Резервы. Резерв учитывается отдельно от фактического остатка. Введение резерва не может превысить доступный остаток, выпуск из резерва не может уйти в минус. Расход из резерва списывает сначала резерв, затем

свободный остаток; последовательность контролируется процедурой проведения.

6 Продажи и оплаты. Сумма по строкам продажи и итог документа согласованы по контрольному СЧЕСК; суммы оплат по документу не могут превышать итоги продажи. Привязка к устройству обязательна; номер чека уникален в рамках устройства. Проведение продажи создаёт расходные движения по указанным партиям и, для маркируемых товаров, переводит коды в состояние «списан».

7 Конкурентная работа. Процедура проведения выполняется в одной транзакции, блокируя затрагиваемые партии и строки документа на изменение. При конфликте за одну и ту же партию в момент проведения второй транзакции получит отказ с понятным кодом ошибки. Повторный вызов проведения для уже проведённого документа отклоняется как идемпотентная ситуация; откат возможен только для проведённых документов и только если по их партиям нет зависимых последующих движений.

8 Удаление и каскады. Удаление справочников, на которые есть ссылки, запрещено. Документы в состоянии «проведён» не удаляются; допускается только откат с последующим удалением черновика. Партии нельзя удалить, если по ним есть движения или непустой остаток.

9 Служебные поля и аудит. В ключевых таблицах действуют триггеры, автоматически заполняющие «кто/когда создал или изменил». Для операций изменения прав, проведения/отката и исправления приходов фиксируются записи аудита с минимумом полей: пользователь, время, тип операции, ссылка на объект.

10 Точность и округления. Денежные значения хранятся в десятичном формате с фиксированной точностью, количественные – с отдельной точностью для штуки/веса. Округление выполняется в процедурах по регламенту фирмы, чтобы исключить расхождения между строками и итогами.

11 Кэш отчётов. В report\_db шаблоны и параметры отчётов связаны внешними ключами; кэшированные агрегаты имеют явный период и ключ построения. На уровне БД запрещено хранить кэш без связанного шаблона, а также пересекать периоды с одинаковым ключом.

Коротко о главных инвариантах, которые база не позволит нарушить: нельзя провести документ, если он приводит к отрицательному остатку; нельзя дважды списать один и тот же серийный/маркировочный код; строки документа не существуют без «шапки» и корректных ссылок; у проведённых документов неизменяемые реквизиты; движения появляются и исчезают только через процедуры; удаление допустимо лишь там, где нет зависимостей.

## **2.4 Входные и выходные данные, параметры отчётности**

Входные данные системы – это устойчивые справочники и поток операционных записей, из которых формируются движения и отчётные выборки. К справочникам относятся номенклатура и её атрибуты (группа,

производитель, страна), контрагенты, склады и торговые точки, устройства продаж, типы оплат и базовые настройки. Операционные данные включают партии и серии (дата поступления, цена, срок годности и т.п.), документы «шапка/строки» всех видов, движения по складам, а также продажи и оплаты с привязкой к устройствам. Эти сущности поступают в mobile\_app и проходят проверки целостности; для авторизации и идентификации источников используются записи в auth\_db.

Выходные данные представляют собой срезы и агрегаты по проведённым операциям: остатки на дату по складам и товарам, обороты за период (приход, расход, перемещения), отчёты по партиям и срокам годности, продажи и выручка в разрезах номенклатуры, групп, подразделений и типов оплат, а также контрольные выборки по качеству данных (например, потенциальные отрицательные остатки, несогласованные статусы). Результаты выдаются в табличном виде, могут быть выгружены в CSV/Excel и распечатаны в виде PDF-форм.

Отчётность реализована через шаблоны запросов. Каждый отчёт – это текст SQL с плейсхолдерами параметров и блоком метаданных параметров. Плейсхолдеры имеют вид PARAMNAMEPARAM\_NAMEPARAMNAME и заменяются на значения выбранного периода, списков и флажков. Метаданные задаются компактно, в конце шаблона, как цепочка пар вида key=value, разделённых «||». Для каждого параметра указываются тип (Date, DateEnd, CheckBox, ListBox), подпись для формы, значение по умолчанию и, при необходимости, default\_sql – запрос, возвращающий набор значений для списков (например, перечень подразделений или типов оплат). В таблице результата допускаются служебные колонки с именами, начинающимися с «~», они используются для сортировок и не выводятся пользователю.

Конвейер исполнения отчёта единообразен. Сервис разбирает блок параметров, строит форму ввода, приводит введённые значения к типам и подставляет их в подготовленный SQL. При необходимости добавляются ограничения видимости (например, по организации или складам доступным пользователю). По умолчанию все отчёты строятся по проведённым операциям; включение черновиков возможно только явным параметром. Долгие выборки кэшируются в report\_db вместе с хешом шаблона и набора параметров; кэш инвалидируется при появлении новых движений в затронутом интервале. Табличный результат форматируется по алиасам столбцов, числовые поля округляются до согласованной точности.

Печатные формы используют те же данные, что и табличный отчёт. Набор записей передаётся в шаблон PDF: в «шапке» автоматически проставляются период и активные фильтры, в «подвале» – дата формирования и инициатор; поддерживается пагинация и суммовые итоги. Для оперативного обмена предусмотрены экспорт в CSV/Excel и копирование заголовков.

Набор параметров стандартизован и повторяется в большинстве отчётов: период начала и конца с типом даты; фильтры по складам/подразделениям и устройствам; отбор по номенклатуре, группам, производителям; типы оплат; флажки «все значения» для быстрого

включения/исключения фильтров. Для отчётов по партиям добавляются параметры по срокам годности и режиму отбора серий; для контрольных отчётов – переключатели включения «пограничных» состояний.

Таким образом, входной поток данных фиксируется и нормализуется в `mobile_app`, а отчётность строится предсказуемо и воспроизводимо по шаблонам SQL с параметрами. Единая схема плейсхолдеров, типизированных метаданных и кэширования в `report_db` обеспечивает одинаковое поведение для всех отчётов, прозрачность формирования результатов и удобную печать в виде стандартных PDF-форм.

## 2.5 Заключение

В ходе анализа зафиксированы функциональные требования к системе на уровне базы данных и определены границы ответственности хранилища. Описаны операции, которые должны поддерживаться без участия клиентского кода: работа с документами «шапка/строки», проведение и откат с формированием движений, учёт партий и серий, резервирование и списания, а также контроль допустимых состояний. Уточнён механизм доступа: проверки выполняются по наборам операций, права хранятся в компактном виде и проверяются на стороне сервисов при обращении к конкретным действиям.

Сформулированы декларативные правила целостности: внешние ключи, типизированные домены для статусов и видов операций, проверочные ограничения и серверные процедуры, обеспечивающие атомарность изменений и исключающие частичные состояния. Отдельно описаны входные и выходные данные, а также единый подход к отчётности на основе параметризованных SQL-шаблонов с кэшированием агрегатов в `report_db` и возможностью вывода результатов в табличном виде и PDF.

Итогом анализа является согласованный перечень функций и инвариантов, который станет основой для инфологической модели. В следующей главе требования будут переведены в набор сущностей, их атрибутов и связей, после чего на этой базе будет построена физическая схема PostgreSQL с реализацией ограничений, индексов, триггеров и процедур проведения/отката. Такой переход обеспечивает прослеживаемость: от формулировок требований ко вполне конкретным таблицам и правилам, гарантирующим корректность учёта и воспроизводимость отчётов.

### **3 ИНФОЛОГИЧЕСКАЯ МОДЕЛЬ ПРЕДМЕТНОЙ ОБЛАСТИ**

Глава переводит требования из раздела 2 в модель сущностей и связей. Фиксируются ключевые справочники и операционные объекты, их атрибуты и бизнес-ключи, кратности и каскады, а также решения по нормализации и точечной денормализации, необходимой для отчётности и производительности.

#### **3.1 Границы модели и состав сущностей**

В данном разделе определяется объём инфологической модели и перечисляется совокупность устойчивых сущностей, необходимых для учёта товаров, партий и документов, а также продаж и оплат. Модель ориентирована на обеспечение корректных остатков и прослеживаемости партий/серий; интерфейсные решения, интеграции с ЭДО и ККТ, маршрутизация поставок, адресное хранение и бухгалтерские регистры в неё не включаются.

Данные структурируются по трём логическим контурам. Контур авторизации содержит пользователей, фирмы, склады, устройства и наборы прав доступа, представленные битовыми масками по операциям сервисов; роли как самостоятельные объекты не используются. Операционный контур включает справочники номенклатуры (товар, группа, производитель, страна, единица измерения), контрагентов, склады/магазины, клиентов и типы оплат, а также документы и их строки, партии и серии, движения по партиям/складам, продажи и оплаты. Отчётный контур хранит шаблоны запросов, описания параметров запуска и кэш агрегированных результатов для ускоренного повторного формирования отчётов.

Документ в операционном контуре имеет тип (приход, расход/продажа, перемещение, прочая операция) и состояние (черновик, проведён, отменён). «Шапка» документа фиксирует общие реквизиты (дата и время, организация, склад-источник и, при наличии, склад-получатель, контрагент, устройство, комментарии), «строки» содержат ссылки на товар, количество и ценовые параметры; при необходимости прослеживаемости строка дополнительно указывает конкретную партию или серию. Партия определяется связью «товар–склад» и характеризуется количеством, закупочной ценой, датой поступления и, при наличии, сроком годности; серия уточняет партию для маркируемой продукции. Движение отражает изменение состояния партии на складе как результат проведения документа и используется как исходная основа для расчёта остатков на дату. Продажа фиксирует факт реализации на конкретном устройстве с привязкой ко времени и номеру чека; одна продажа может включать несколько оплат, каждая оплата связана с типом оплаты и суммой.

Связи и кратности заданы следующим образом. Один документ включает множество строк; каждая строка относится к одному складу-

источнику и, для перемещения, к одному складу-получателю. Строка ссылается на один товар и, при необходимости, на одну партию и одну серию. По каждой партии аккумулируется множество движений. Одна продажа агрегирует множество товарных позиций и имеет одну или несколько оплат; оплата относится к одной продаже и одному типу оплаты. Устройство закреплено за одним складом и фигурирует в продажах и оплатах. Пользователь работает с устройствами через сервисы, при этом проверка прав выполняется по отдельным операциям, а не по ролям.

Для доменов, критичных к целостности, используются фиксированные наборы значений: вид документа, состояние документа, вид операции, тип товара и тип оплаты. Эти домены трактуются как перечислимые и применяются во всех сущностях, где требуется жёсткая проверка допустимости значений. Идентификация объектов осуществляется стабильными первичными ключами; для справочников дополнительно могут использоваться бизнес-ключи (штрихкод, артикул, наименование в сочетании с производителем) с обеспечением их уникальности в пределах заданных контекстов.

Разделение по контурам предполагает независимую эволюцию схем: авторизационные данные не пересекаются с предметными таблицами, операционный контур не содержит отчётных витрин, кэш отчётов не влияет на транзакционные операции. Такое разделение фиксирует границы ответственности и упрощает дальнейший переход к физической модели, где для каждой из перечисленных сущностей будут определены атрибуты, ключи, внешние связи и правила каскадирования изменений.

### **3.2 Атрибуты и ключи: справочники и документы**

В данном пункте фиксируются состав атрибутов базовых сущностей и принципы их идентификации на инфологическом уровне. Типы полей приводятся укрупнённо (дата/время, число, строка), логика хранения денег и количеств уточняется позднее в физической схеме. Для устойчивых доменов используются перечислимые значения: вид документа, состояние документа, вид операции, тип товара, тип оплаты.

Авторизационный контур включает пользователя (уникальный логин, ФИО, контакты, активность, привязка к фирме при необходимости), фирму (наименование, учётные реквизиты, признак активности), склад/магазин (наименование, адрес, принадлежность фирме), устройство (серийный номер, наименование, закрепление за складом, признак активности) и таблицу прав, где для каждой пары «пользователь – сервис/эндпоинт» хранится битовая маска разрешённых операций. Бизнес-ключи: уникальность логина в инсталляции, уникальность серийного номера устройства, уникальность наименования склада внутри фирмы.

Справочники номенклатуры включают товар (наименование, группа, производитель, страна, единица измерения, тип «товар/услуга», признак маркируемости, артикул, НДС/налоговая категория, статус активности),

группу товаров (иерархия по желанию), производителя и страну (наименования), единицы измерения (код, коэффициент пересчёта к базовой). Штрихкоды и иные коды идентификации целесообразно вынести в отдельную сущность «коды товара» с атрибутами «тип кода» и «значение», это позволит хранить несколько кодов на один товар. Бизнес-ключи: уникальность штрихкода в пределах фирмы; при отсутствии штрихкода – пара «наименование + производитель» в сочетании с фирмой или группой.

Партия определяет происхождение и ценовые параметры запаса на складе: ссылка на товар и склад, дата поступления, источник (документ/контрагент), закупочная цена, валюта при необходимости, срок годности, номер партии/серии производителя, признак маркируемой продукции. Количество и остаток на дату рассматриваются как производная от движений, а не как атрибут партии. Серия уточняет партию для штучно-маркируемой продукции и хранит индивидуальный серийный/маркировочный код; серия принадлежит одной партии и одному товару. Бизнес-ключи: в пределах склада партия уникальна по комбинации «товар + дата/источник + закупочная цена + номер партии»; серийный/маркировочный код уникален в пределах товара (или всей системы – по договорённости).

Документ (шапка) содержит вид документа (приход, расход/продажа, перемещение, прочее), состояние (черновик, проведён, отменён), дату и время, организацию, склад-источник и, при необходимости, склад-получатель, контрагента, устройство (для продаж), пользователя-создателя и пользователя-проведшего, комментарии и итоговые суммы (товарная, скидка, НДС, к оплате). Для удобства учёта может использоваться номер документа с уникальностью в контексте «организация/склад/период». Строка документа ссылается на шапку и фиксирует товар, количество, цену, сумму, скидки, ставку НДС, а при необходимости – партию и серию, а также последовательный номер строки для стабильной адресации. Инварианты уровня модели: склад в строке должен соответствовать складу из шапки (для перемещений – источнику/получателю по виду строки); вид документа определяет допустимость ссылок на партию/серию; суммы шапки согласованы с агрегатами по строкам. Бизнес-ключи: уникальность номера документа в выбранном контексте; уникальность номера строки в пределах документа.

Движение является первичным источником остатков: атрибуты включают дату/время, ссылку на документ и строку, товар, склад, партию и при необходимости серию, вид операции (приход, расход, перемещение исход/приход), количественную дельту и стоимостную дельту. Для движения задаётся признак, формирующий последовательность расчёта остатков (по дате/времени и идентификатору). Бизнес-ключи: отсутствие повторных движений для одной и той же строки и той же партии/склада; единственность записи-прихода партии из документа поступления.

Продажа отражает факт реализации на устройстве: дата/время, устройство, склад устройства, номер чека (уникален в пределах устройства), общая сумма, сумма скидок, признак аннулирования, ссылка на связанный

документ расхода при наличии. Позиция продажи хранит товар, количество, цену, скидку, возможную ссылку на партию/серию. Оплата относится к продаже и типу оплаты, хранит сумму, валюту при необходимости и служебные реквизиты (номер транзакции эквайринга и т. п.). Тип оплаты – стабильный справочник. Бизнес-ключи: номер чека уникален в пределах устройства; суммарная оплата по продаже не превышает итог к оплате, а в случае полной оплаты совпадает с ним.

Контрагенты и клиенты задаются отдельно: контрагент – юридическое/физическое лицо для приходов и перемещений «вне склада», клиент – конечный покупатель с минимальными идентификаторами (ФИО/телефон/карта лояльности), при необходимости – признак категории скидки. Для клиентов предусматривается уникальность номера карты или телефона в пределах фирмы.

Отчётный контур хранит описание отчёта (наименование, назначение, текст запроса на выборку с параметрами-плейсхолдерами, сведения о полях результата, привязка к печатной форме), описание параметров (имя, тип, обязательность, значение по умолчанию через подзапрос, набор допустимых значений) и кэш результатов (хеш параметров, интервал времени, сжатый набор данных, дата расчёта, источник данных). Бизнес-ключи: уникальность имени параметра внутри отчёта; уникальность записи кэша по хешу параметров и идентификатору отчёта.

Во всех критичных сущностях целесообразно фиксировать атрибуты аудита: момент создания и изменения, пользователя-создателя и пользователя, внесшего правку, источник изменения (устройство/сервис). Это требуется для воспроизводимости событий и последующего анализа качества данных. Для идентификации объектов используются стабильные суррогатные ключи; бизнес-ключи вводятся там, где они естественны и обеспечивают устойчивую уникальность (номера документов, штрихкоды, серийные коды, номера чеков). Такая фиксация атрибутов и ключей задаёт основу дальнейшего перехода к физической модели, где будут уточнены типы данных, индексы и серверные ограничения, реализующие перечисленные инварианты.

### **3.3 Связи, кратности и каскады изменений**

Авторизационный контур образует «скелет» владения и доступа. Фирма владеет складами: один ко многим. Устройство закрепляется за складом: один ко многим; смена склада у устройства допускается через историю привязок, но в текущем состоянии связь обязательна. Пользователь существует в общей инсталляции и может быть связан с одной фирмой по умолчанию; права пользователя задаются как набор записей «пользователь – сервис/эндпоинт» с битовой маской операций: один ко многим. Каталог сервисов/эндпоинтов общий и используется как справочник-источник для этих связей.

Справочники номенклатуры опираются на иерархию групп товаров: группа к группе – отношение «родитель–потомок» с нулем или многими потомками и не более чем одним родителем. Товар относится к одной группе,

одному производителю и одной стране: во всех трёх случаях один ко многим. Штрихкоды и другие идентификаторы образуют связь «товар – код»: один ко многим, поскольку один товар может иметь несколько кодов. Единицы измерения выступают как общий справочник; при необходимости поддерживается связь «товар – набор единиц с коэффициентами» через отдельную таблицу, что фактически реализует многие ко многим.

Партии и серии уточняют запас на складе. Партия относится к одному товару и одному складу: один ко многим с обеих сторон; для каждого товара на складе партия существует в нуле или многих экземплярах. Серия принадлежит одной партии: один ко многим; при необходимости быстрых проверок прослеживаемости допускается дополнительная ссылка «серия – товар» (дублирующая товар из партии как производное ограничение). Для маркируемой продукции строки документов и движения могут ссылаться на серию, для немаркируемой – только на партию.

Документы строятся по схеме «шапка – строки»: один ко многим. Шапка документа привязана к фирме обязательно. Для приходов и продаж обязателен один склад, для перемещений задаются склад-источник и склад-получатель; эти ссылки обязательны на уровне шапки, а в строках склад определяется из вида операции строки. Контрагент обязателен в приходах от поставщика и прочих операциях, где он участвует; для розничной продажи может отсутствовать. Каждая строка документа ссылается на один товар, опционально на партию и, при необходимости, на серию; при этом ссылка на партию обязательна, если вид товара или вид операции требует прослеживаемости. Номер строки уникален в пределах документа, что обеспечивает устойчивую адресацию.

Движения формируются из строк документов и являются первичным источником остатков. Каждое движение ссылается на одну строку документа: многие к одному. Кардинальность зависит от вида документа: строка прихода порождает одно положительное движение, строка продажи – одно отрицательное, строка перемещения – как правило два движения (расход со склада-источника и приход на склад-получатель). Для контроля идемпотентности вводится ограничение «не более одного движения данного типа на строку документа и партию/склад». Движение обязательно ссылается на товар и склад, а также на партию; ссылка на серию обязательна только для маркируемой номенклатуры.

Продажи фиксируют чековые события на устройствах. Продажа относится к одному устройству и через него к одному складу: многие к одному. Позиции продажи связаны с продажей отношением один ко многим и с товаром отношением многие к одному; ссылка на партию/серию в позиции продажи обязательна по тем же правилам прослеживаемости. Оплаты связаны с продажей отношением один ко многим и с типом оплаты отношением многие к одному. Связь между продажей и документом расхода допускается как многие к одному: несколько чеков могут быть агрегированы в один расходный документ или, наоборот, каждый чек может иметь отдельный документ – конкретная стратегия закрепляется правилом на уровне сервера.

Клиенты участвуют в продажах опционально. Продажа может ссылаться на одного клиента; клиент не обязан присутствовать у всех продаж. Карты лояльности при необходимости оформляются как связь один ко многим между клиентом и его идентификаторами карт, а в продаже хранится ссылка на использованную карту.

Отчётный контур декларирует связи «отчёт – параметры» как один ко многим и «отчёт – кэш» также как один ко многим. Параметр принадлежит ровно одному отчёту; кэш связан с отчётом и содержит ссылку на фирму и интервал времени, чтобы обеспечить сегрегацию данных. Печатные формы могут быть связаны с отчётом отношением один к одному либо один ко многим при наличии нескольких шаблонов вывода.

Поперечные связи между контурами ограничены: отчётные запросы читают операционные и справочные данные, но физического FK между `report_db` и `mobile_app/auth_db` не предполагается; сегрегация достигается через атрибуты фильтрации (идентификатор фирмы, склада, устройства) и проверку прав на стороне сервисов. Внутри мобильного контура ссылки строгие: удаление справочника, на который есть ссылки, запрещается, за исключением управляемых сценариев архивирования, которые переводят запись в неактивное состояние без физического удаления. Для документов удаление шапки запрещено при наличии строк; перевод документа в «отменён» не разрывает ссылок, а меняет допустимость операций. Для партий и серий запрещено удаление при наличии движений или ссылок из строк документов и продаж.

Многие ко многим встречаются в ограниченном числе мест и реализуются через связывающие таблицы: набор единиц для товара, набор прав пользователя по сервисам, набор штрихкодов товара. Во всех остальных случаях связь проектируется как один ко многим для упрощения инвариантов и прозрачности аудит-трассировки.

Такая система связей задаёт однозначные направления владения, минимизирует циклы и обеспечивает возможность строгих внешних ключей на физическом уровне, что необходимо для воспроизводимого расчёта остатков и корректного проведения документов.

### **3.4 Нормализация и обоснованные исключения**

Модель строится в логике «шапка–строки» с отделением устойчивых справочников от операционных сущностей. Базовый уровень – нормализация до третьей нормальной формы: атрибуты, относящиеся к товару, производителю, стране, единицам измерения, хранятся в справочниках и не дублируются в строках документов; количественно-ценовые показатели закрепляются там, где они возникают по смыслу операции. Для строк документов хранится только то, что относится к факту движения: товар, партия/серия (при необходимости), количество, цена на момент операции и ссылка на «шапку». Это устраняет аномалии обновления и обеспечивает воспроизводимость отчётов на произвольную дату.

Отдельное решение – партии и серии как самостоятельные сущности. Срок годности, закупочная цена, признаки прослеживаемости относятся к партии/серии и не переносятся в строки как повторяющиеся реквизиты. Строка ссылается на конкретную партию (и серию, если товар маркируемый), что позволяет однозначно восстанавливать цепочку поставки и корректно списывать запасы. Для немаркируемой номенклатуры ссылка на серию опциональна и не влияет на нормализацию партии.

Документы разных видов унифицированы одной структурой «шапки» и одной таблицей строк, отличия выражаются доменными признаками: вид документа, тип строки, состояние. Такой подход лучше, чем множество «почти одинаковых» таблиц, потому что исключает схему с дублирующимися структурами и упрощает контроль инвариантов (проведение, откат, допустимые переходы состояний). В перемещениях склад-источник и склад-получатель задаются на уровне «шапки»; строки не несут собственных складов, чтобы исключить рассогласование между уровнем документа и строк.

Накопительные показатели рассматриваются как производные. Остаток на складе и обороты – функции от движений; в чистой ЗНФ они не хранятся. Вместе с тем для ускорения типовых выборок допустима управляемая денормализация: в партиях могут поддерживаться агрегаты «в приходе/в резерве/доступно», а в «шапке» документа – итоги по сумме и количеству. Эти значения считаются вторичными и поддерживаются только серверной логикой при проведении/откате. Инварианты формулируются так: агрегаты в точности равны сумме детальных записей; прямые правки агрегатов запрещены. Тем самым сохраняется логическая корректность при выигрыше в производительности.

Номенклатура и идентификаторы нормализуются без избыточной универсальности. Штрихкоды и внутренние коды – связь «товар – множество кодов», что позволяет хранить несколько идентификаторов без дублирования карточки товара. Единицы измерения реализуются как справочник с возможностью задать для товара набор допустимых единиц и коэффициенты пересчёта; это снимает необходимость держать в строках одновременно базовую и альтернативную единицу. Исторические атрибуты (цена продажи, ставка налога, скидка) фиксируются в строке документа на момент операции, а не берутся из текущих настроек – это разрывает причинно-следственную петлю и гарантирует воспроизводимость прошлых чеков и актов.

Ключи выбираются прагматично. Для большинства сущностей применяются суррогатные идентификаторы, а бизнес-уникальность обеспечивается ограничениями: уникальное сочетание «товар – код», уникальность номера документа в пределах фирмы и диапазона дат, уникальность номера строки в пределах документа. Натуральные ключи вроде штрихкода не используются в качестве первичных из-за нестабильности и различий в источниках.

Удаление и архивирование трактуются как разные операции. Физическое удаление записей, на которые существуют ссылки, не

допускается; вместо этого применяется признак актуальности/архива и дата окончания действия.

Права доступа нормализованы по операции. Так как в системе нет фиксированных ролей, связь «пользователь – эндпоинт/операция» хранится как множество записей; набросок матрицы прав в компактном виде допускается в виде карты соответствия эндпоинтам, но источник истины – реляционные записи, пригодные для выборок и проверок консистентности. Это исключает дублирование и обеспечивает однозначность интерпретации при изменении перечня операций.

Границы между контурами выбираются с учётом целостности. Внутри операционного контура связи строго материализованы внешними ключами; в отчётном контуре кэш и параметры отчётов не держат ссылок на операционные таблицы, чтобы не тянуть сквозные ограничения между базами. Согласованность обеспечивается по значениями-идентификаторам и временем формирования кэша, а не за счёт межбазовых FK.

Итогом таких решений является модель, которая сохраняет преимущества нормализованной схемы (отсутствие аномалий и дублирования) и допускает точечную денормализацию только там, где это подтверждено требованием производительности и прикрыто инвариантами. Это облегчает дальнейший переход к физической схеме PostgreSQL, где домены, внешние ключи и серверная логика закрепят описанные правила на уровне данных.

### **3.5 Заключение**

В инфологической модели зафиксирован устойчивый набор сущностей и связей, достаточный для корректного учёта складских операций: справочники номенклатуры и контрагентов отделены от операционных объектов, документы представлены в схеме «шапка–строки», партии и серии выделены как самостоятельные носители учётных признаков, а остатки рассматриваются как производные от движений. Статусы документов, виды операций и другие фиксированные домены заданы на уровне модели как конечные множества значений, что упрощает последующее закрепление инвариантов. Спорные места решены с приоритетом нормализации и воспроизводимости: исторические цены и параметры скидок фиксируются в строках документов; допустима только контролируемая денормализация агрегатов, поддерживаемая серверной логикой. Выбранные ключи и ограничения исключают аномалии обновления и «висящие» ссылки, а разграничение контуров (операционный и отчётный) позволяет отделить транзакции от аналитики без потери согласованности. Модель готова к переходу к физическому проектированию в PostgreSQL: типизация доменов перечислимыми типами, материализация внешних ключей и уникальностей, определение индексов по путям доступа, а также реализация триггеров и процедур проведения/отката, поддерживающих описанные инварианты.

## 4 ФИЗИЧЕСКАЯ МОДЕЛЬ И РЕАЛИЗАЦИЯ В POSTGRESQL

В данной главе фиксируются типы данных и перечисления для доменов предметной области, соглашения об именовании, выбор первичных и внешних ключей, а также политика ссылочной целостности и каскадных действий. Отдельно создаются три согласованные схемы: `auth_db` для учетных записей, складов, устройств и матрицы прав; `mobile_app` для номенклатуры, партий/серий, документов и движений; `report_db` для параметров, шаблонов и кэша агрегатов. На уровне таблиц уточняются обязательные атрибуты, уникальные и проверочные ограничения, а также варианты хранения технических полей (идентификаторы, метки времени, аудит изменений). Для обеспечения предсказуемого времени отклика проектируются индексы по ключевым путям доступа и, где уместно, частичные и по выражениям; для тяжёлых выборок отчётный кэш изолируется в `report_db`. Серверная логика сведена к прозрачному набору триггеров и процедур, реализующих проведение и откат документов в одной транзакции и предотвращающих появление отрицательных остатков и «висящих» ссылок. В завершение рассматриваются аспекты конкурентного доступа и блокировок, чтобы операции разных пользователей безопасно совмещались с расчётом отчётов и кэшем. Таким образом, последующие подпункты поочерёдно фиксируют принципы физического проектирования и детализируют каждую схему, после чего приводятся индексация и серверные механизмы, обеспечивающие целостность и производительность.

### 4.1 Принципы физического проектирования, домены и перечисления

Далее термины `auth_db`, `mobile_app` и `report_db` используются как имена схем в одном экземпляре PostgreSQL; при развёртывании допускается разнести контуры по разным БД без изменения логики DDL. Все имена объектов – в `snake_case`, единственное число для сущности (`product`, `document`), множественное – только для служебных наборов значений. Для первичных ключей применяются `surrogate-id` с типом `bigint generated always as identity`. Бизнес-ключи фиксируются уникальными ограничениями (например, номер документа в пределах организации и вида документа). Метки времени хранятся в `timestamptz (UTC)`; поля `created_at` и `updated_at` обязательны для всех изменяемых таблиц. Ссылочная целостность обеспечивается внешними ключами; каскадное удаление используется только для технических «вложенных» сущностей (например, строк документа в состоянии черновика), для прочих действий – `restrict/no action`. Технические поля «отключения» записей реализуются флагом `is_active`, физическое удаление справочников, на которые есть ссылки, не допускается.

Для повторно используемых величин вводятся доменные типы PostgreSQL, что сокращает дублирование CHECK-ограничений и повышает однородность схемы:

- dom\_quantity numeric(18,6) check (value >= 0) – количество;
- dom\_money numeric(18,2) check (value >= 0) – денежные суммы;
- dom\_percent numeric(5,2) check (value between 0 and 100) – скидки и наценки;
- dom\_code text check (length(value) between 1 and 64) – артикул, внутренние коды;
- dom\_ean text check (value ~ '^d{8}(\d{5})?\$|^d{13}\$') – штрих-коды EAN-8/EAN-13 [11];
- dom\_note text – произвольные примечания без ограничений размера;
- dom\_json jsonb – структурированные параметры и матрицы прав.

Фиксированные домены предметной области задаются перечислимыми типами:

- document\_kind\_enum: RECEIPT (поступление), SALE (расход/продажа), MOVE (перемещение), ADJUSTMENT (прочие операции);
- document\_state\_enum: DRAFT (черновик), POSTED (проведён), CANCELED (отменён);
- movement\_kind\_enum: INBOUND, OUTBOUND, TRANSFER – тип движения при проведении;
- product\_type\_enum: GOOD (товар), SERVICE (услуга).

Перечисления применяются в полях ключевых сущностей: у документов – вид и состояние; у движений – тип; у номенклатуры – тип позиции. Для изменяемых справочников и открытых перечней (страны, производители, группы, типы оплат) используются обычные таблицы с уникальными бизнес-ключами; значения не «зашиваются» в enum, чтобы допускать пополнение без миграций.

Правила нумерации документов отражают практику учёта: номер хранится строковым полем doc\_no (возможны префиксы) и продублирован целочисленным порядковым doc\_seq для сортировки; уникальность обеспечивается в разрезе организации, года и вида документа. Даты документов разделяются на дату оформления (doc\_date date) и дату/время проведения (posted\_at timestamptz); изменение posted\_at вне процедур проведения запрещено.

Соглашения по ограничениям и индексам: имена ограничений имеют вид pk\_<table>, fk\_<from><to>, uq\_<table><cols>, chk\_<table><col>; имена индексов – ix\_<table><cols>. Для внешних ключей обязательно наличие индекса по FK-полю; состав и тип индексов детализируются в разделе 4.5. Для полей, влияющих на поиск и уникальность (sku, ean, doc\_no в своём разрезе), задаются явные UNIQUE-ограничения.

Требования к аудиту изменений поддерживаются через created\_by и updated\_by (ссылка на пользователя из auth\_db) там, где это целесообразно; значения обновляются серверной логикой. Текстовые поля хранятся в UTF-8;

чувствительность к регистру для логинов и e-mail может обеспечиваться через расширение citext, при этом в схемах указывается явный тип citext.

Указанные соглашения, домены и перечисления формируют основу для единообразного DDL в последующих подпунктах и позволяют выразить инварианты предметной области на уровне самой СУБД, до реализации триггеров и процедур.

## 4.2 Схема auth\_db (пользователи, устройства, склады, права)

Схема auth\_db предназначена для хранения информации об организациях, пользователях, торговых точках, устройствах и правах доступа. Она служит входным контуром всей системы: именно здесь фиксируются владельцы данных, точки учёта и права на операции. Данные из этой схемы используются всеми сервисами при авторизации и проверке доступа к операциям.

Основные сущности схемы auth\_db представлены в таблице 4.1.

Таблица 4.1 – Основные таблицы схемы auth\_db

Таблица	Назначение
device_settings	Настройки устройств: сетевые параметры, режим обмена, конфигурация кассовых терминалов.
Devices	Зарегистрированные устройства (кассы, терминалы, сканеры), привязанные к складам.
firm_settings	Глобальные параметры работы фирмы: налоговые ставки, валюты, шаблоны документов.
firms	Справочник компаний – владельцев данных; корневая сущность для пользователей и складов.
invoice_params	Параметры шаблонов и реквизитов накладных, используемые при обработке документов.
stores	Склады и торговые точки, принадлежащие фирмам, к которым привязываются устройства.
users	Учётные записи пользователей системы: логины, пароли, имена, привязка к фирмам и ролям.

Все объекты схемы связаны с центральной сущностью firms, что обеспечивает логическую изоляцию данных разных организаций. Пользователи и устройства жёстко привязаны к фирме и складу, а их поведение регулируется параметрами из таблиц firm\_settings и device\_settings. Таблица invoice\_params используется для хранения шаблонных реквизитов при формировании документов и накладных.

Схема реализует основную модель разграничения доступа в системе. Каждый пользователь имеет определённые права в разрезе фирм, складов и сервисов, что позволяет гибко управлять действиями в микросервисной архитектуре. За счёт единого источника данных auth\_db достигается

согласованность авторизации между всеми сервисами и прозрачность управления правами.

#### 4.3 Схема mobile\_app (товары, партии/серии, документы, строки, движения)

Схема mobile\_app является центральной частью информационной системы и предназначена для хранения всех операционных данных.

В ней сосредоточены справочники номенклатуры, контрагентов и клиентов, а также таблицы, описывающие документы, партии товаров, продажи, оплаты и движения. Именно в этой схеме выполняется основная бизнес-логика учёта: проведение, откат, расчёт остатков и контроль целостности данных.

Основные сущности схемы mobile\_app представлены в таблице 4.2, а структура связей – в Приложении В.

Таблица 4.2 – Основные таблицы схемы mobile\_app

Таблица	Назначение
bathes	Учёт партий товаров с указанием цены, количества, даты поступления и связи с товаром.
cities	Справочник городов, используется для заполнения данных клиентов и контрагентов.
clients	Клиенты (покупатели), участвующие в операциях продаж и скидочных программах.
discount_cards	Карты лояльности с указанием владельца, скидки и накопленных бонусов.
discounts	Справочник скидок: ручные и автоматические, с правилами применения.
docs_elems_expenses	Строки расходов по документам, фиксирующие резервирование и списания.
docs_elems_orders	Строки заказов по документам: позиция, серия и количество заказа.
docs_elems_series	Учёт серийных номеров и сроков годности для партий.
docs_elems_work	Основные строки документов (товар, количество, цена, тип значения).
documents_types	Классификатор видов документов (приход, расход, перемещение и т.д.).
documents	Заголовки документов: реквизиты, даты, партнёр, состояние, ссылки на строки.
logs	Журнал операций и системных событий для аудита и отладки.
marks	Маркировка товаров: уникальные коды, тип марки и состояние (продан/не продан).

#### Продолжение таблицы 4.2

Таблица	Назначение
partners	Контрагенты: поставщики, покупатели и подразделения.
payment_types	Виды оплат (наличные, безналичные, карта и т.д.).
payments	Записи об оплатах, связанных с продажами и документами.
products	Каталог товаров и услуг, включая тип, группу, производителя и страну происхождения.
products_country	Справочник стран производителей.
products_groups	Иерархия групп товаров, используется для аналитики и фильтрации.
products_name	Справочник наименований товаров.
products_producer	Справочник производителей продукции.
sales	Продажи товаров по устройствам и складам: количество, цена, скидка, чек.
tickets	Таблица билетов, талонов и ваучеров, связанных с партиями и скидками.

Схема mobile\_app обеспечивает полный цикл учёта товарных операций – от поступления и продажи до формирования отчётных данных. Данные этой схемы используются для анализа движения товаров, расчёта остатков и построения отчётов в контуре report\_db. Она является ключевым элементом всей системы и определяет корректность и прозрачность бизнес-процессов.

#### 4.4 Схема report\_db (параметры, шаблоны, кэш отчётов)

Схема report\_db предназначена для хранения описаний отчётов, их параметров, печатных форм и кэшированных агрегатов. Она служит аналитическим контуром системы, отделённым от транзакционной базы mobile\_app, что позволяет формировать сложные выборки и отчёты без влияния на производительность основных операций. Основные сущности схемы report\_db приведены в таблице 4.3, а структура связей представлена в Приложении С.

Таблица 4.3 – Основные таблицы схемы report\_db

Таблица	Назначение
cache_rows	Таблица строк кэша отчётов: хранит отдельные значения колонок и их типы.
firms_reports	Связь между фирмами и доступными им отчётами.
print_forms	Печатные формы и шаблоны вывода отчётов (PDF, Excel и др.).

#### Продолжение таблицы 4.3

Таблица	Назначение
property_start_frame	Параметры и настройки стартовых кадров (окон) для формирования графиков.
report_cache	Кэш агрегированных результатов отчётов, ускоряющий повторное формирование данных.
report_groups	Группы и категории отчётов (например, складские, финансовые, аналитические).
reports	Основные отчёты системы: SQL-запросы, параметры, связи с формами и группами.

Схема report\_db обеспечивает хранение шаблонов SQL-запросов, параметров отчётов и результатов их вычисления. Благодаря кэшированию данные аналитики можно получать повторно без нагрузки на основную контур. Каждый отчёт связан с печатной формой и группой, а доступность отчётов фирмам регулируется таблицей firms\_reports. Такое разделение упрощает построение гибкой и масштабируемой системы отчётности, независимой от транзакционной базы данных.

### 4.5 Индексы, ограничения целостности и пути доступа

Целостность в проекте опирается на «жёсткие» ограничения схемы и минимально необходимую серверную логику. Базовые инварианты фиксируются декларативно: внешние ключи между «шапками» и строками документов, между строками и партиями/сериями; NOT NULL и CHECK для количеств, цен и дат; уникальность бизнес-ключей (например, store\_id, check\_number для продаж). Для доменов с конечными значениями применяются определенные типы (вид документа, подтип, состояние, вид операции), что исключает «левые» статусы и упрощает проверку переходов.

Индексация подчинена типовым путям доступа. Для списков документов и их строк используются составные btree-индексы по doc\_type, status, doc\_date desc и document\_id, line\_no. Для расчётов остатков и движения – warehouse\_id, product\_id, move\_ts и отдельные индексы по batch\_id. Для партий – product\_id, warehouse\_id, expiration\_date nulls last, что ускоряет выборку «раньше истекающих» лотов. В отчётном контуре ключевым является уникальный индекс по report\_id, params\_hash в кэше результатов; дополнительно ставится GIN по params\_jsonb для точечного поиска запусков [12]. Частичные индексы применяются там, где доля значений мала (например, документы в статусе draft), чтобы не раздувать общий объём [13].

Часть проверок дублируется триггерами, чтобы их невозможно было обойти через прямые INSERT/UPDATE [14]. На «шапке» документа триггер запрещает смену статуса в обход процедур; на строках – запрет изменений после проведения; на партиях – запрет отрицательных остатков и списаний из «чужого» склада. Для прослеживаемости переходов используется таблица

журналирования состояний: триггер при смене статуса добавляет запись с указанием инициатора и времени.

Для `auth_db` применяется простой, но строгий контроль формы данных прав. Карта прав хранится в JSONB, однако её структура валидируется: CHECK гарантирует тип «object», а before insert/update-триггер вызывает проверочную функцию, которая убеждается, что ключи соответствуют известным сервисам/операциям, значения — целые битовые маски в допустимом диапазоне. Это оставляет гибкость на уровне приложений и защищает БД от некорректной формы.

Блокировки и конкуренция учтены в алгоритмах проведения. Выборка FOR UPDATE по партиям и/или агрегатам остатков даёт детерминированный порядок модификаций; уровни изоляции и statement\_timeout задаются явно на время процедуры, чтобы «тяжёлые» операции не блокировали прикладной ввод [15]. Для повторяемых ошибок предусмотрены информативные сообщения (например, «недостаточно остатка для списания партии N»), возвращаемые вызывающему сервису.

Отдельно решён вопрос «дорогостоящих» выборок. В транзакционном контуре расчёт остатков на дату выполняется через подготовленное представление по движениям с нужными индексами; для периодической аналитики используется report\_db с кэшем и управляемой актуализацией. Такой разнос снимает нагрузку с mobile\_app и делает время отклика прогнозируемым.

В итоге совмещение строгих ограничений, адресной индексации и небольшого набора процедур/триггеров обеспечивает две цели: данные технически невозможно привести в противоречивое состояние, а операции проведения/отката выполняются атомарно и воспроизводимо даже при параллельной работе.

## 4.6 Заключение

В рамках главы сформирована законченная схема данных и её серверная обвязка. Инфологическая модель разложена на устойчивые справочники и операционные сущности «шапки/строки»; критичные домены представлены перечислимыми типами; связи закреплены внешними ключами; числовые и календарные поля защищены NOT NULL и CHECK. Физическая реализация разделена по контурам: auth\_db хранит пользователей и матрицу прав с валидацией структуры JSONB; mobile\_app содержит документы, строки, партии/серии и движения; report\_db — параметры, наборы данных и кэш отчётов. Производительность обеспечивается адресной индексацией (включая частичные и по выражениям) и разгоном тяжёлых выборок в отчётный контур. Консистентность операций поддерживается процедурами conduct\_document и rollback\_document, а попытки обойти правила блокируются триггерами.

## 5 ТЕСТИРОВАНИЕ И ВЕРИФИКАЦИЯ БАЗЫ ДАННЫХ

Глава посвящена проверке того, что разработанная схема и серверная логика PostgreSQL корректно выполняют заявленные инварианты и поддерживают типовые сценарии учёта. Тестирование охватывает уровни от базовых ограничений и перечислимых доменов до триггеров и процедур проведения/отката, а также сквозные цепочки операций с документами, партиями и сериями. Отдельно проверяются конкурентная работа пользователей, поведение блокировок и предсказуемость времени отклика на рабочих объёмах данных.

Испытания проводятся на воспроизводимом стенде: каждая база поднимается миграциями до фиксированной версии, загружаются одинаковые начальные данные и сценарные наборы операций. Для `mobile_app` проверяется непротиворечивость остатков и недопустимость «висящих» ссылок при любых переходах состояний; для `report_db` оценивается корректность выборок и выгрузок, а также эффект кэширования на долгих агрегатах. Результаты сопоставляются с критериями приёма из главы 1 и оформляются в виде протоколов с указанием входных данных, ожидаемого и фактического поведения.

### 5.1 Тестовый стенд и исходные данные

Тестовый стенд совпадает с рабочим развертыванием системы: фронтенд, семь сервисов прикладной логики, три экземпляра PostgreSQL (`auth_db`, `mobile_app`, `report_db`), миграции Alembic и начальные данные [16]. Запуск выполняется в `docker-compose` с профилями `dev` и `test` [17]. Профиль `test` обеспечивает чистые базы при каждом старте, фиксированную версию схемы (`alembic revision`) и детерминированные настройки окружения. Все секреты и параметры подключений передаются через переменные окружения, хэш коммита и номер ревизии миграций фиксируются в логе прогона.

Набор фиксированных данных включает минимально необходимую предметную область: несколько складов и устройств, пользователей с различными наборами битовых прав на эндпоинты, справочники товаров и групп, партии и серии, по одному–двум документам каждого типа, типы оплат и примеры продаж. Идентификаторы и даты заданы так, чтобы отчёты «за сегодня» и «за период» воспроизводились независимо от даты запуска; в БД используется единая зона UTC, в интерфейсе – локальное отображение.

Методика испытаний опирается на два взаимодополняющих трека. Первый – сквозные сценарии через API: аутентификация, создание документа, добавление строк, проведение, формирование отчёта, откат, контроль доступа по битовым маскам. В часть сценариев включена печать отчёта в PDF и проверка корректности подстановки параметров отчёта (диапазон дат, выбор подразделений и типов оплат). Второй трек – прямые SQL-проверки инвариантов в СУБД: отсутствие «висящих» ссылок, запрет отрицательных остатков, корректность переходов состояний, соответствие агрегатов

движениям, согласованность итогов «шапки» и строк, корректное создание и инвалидация кэша в `report_db` при изменении первичных данных. Ключевые операции проведения и отката валидируются вызовами процедур `conduct_document()` и `rollback_document()` в пределах одной транзакции.

Параллельная работа проверяется имитацией конкурентных действий: одновременное проведение документов, списывающих одну и ту же партию; независимые операции по разным складам; параллельное формирование отчётов. Фиксируются факты ожидания и блокировок, диагностируются конфликтные ситуации на уровне БД, анализируется журнал транзакций и состояние `pg_stat_activity`. Для отчётного контура отдельно оценивается влияние прогрева кэша `report_db` на время построения отчётов.

Метрики сбора включают время проведения и отката документа, длительность построения типовых отчётов в холодном и тёплом состоянии, число и продолжительность блокировок при конкурентных сценариях, а также объём данных кэша отчётов. Пороговые значения не «зашиваются» в код, а фиксируются в протоколе испытаний и используются как критерии приёмки на стенде.

Процедура прогона стандартизирована:

- сборка и поднятие стенда в профиле `test`;
- исполнение сквозных API-сценариев;
- исполнение SQL-проверок инвариантов;
- генерация отчётов и печать PDF;
- сбор артефактов и метрик, остановка стенда и очистка томов.

Артефакты испытаний – журналы сервисов, SQL-логи, дампы контрольных выборок, PDF-экземпляры отчётов, сводный протокол с указанием коммита и ревизии схемы – сохраняются для воспроизводимости. Такая организация стенда и методики позволяет проверять соответствие реализованной БД требованиям главы 2 не на абстрактных примерах, а в условиях, максимально приближённых к эксплуатационным.

## **5.2 Сценарии интеграционного тестирования (сквозные)**

Сквозные сценарии выполняются поверх реального API и трогают все три контуры БД одновременно. Цель – подтвердить, что бизнес-цепочки «документ → движения → отчёт» воспроизводятся детерминированно, а критичные инварианты удерживаются на стороне СУБД.

1 Сценарий «Приход партии». Через API создаётся товар, склад, контрагент и документ поступления с одной-двумя строками. Проверяется, что до проведения остатки не меняются; после вызова `conduct_document()` появляются записи движений, остатки по складу увеличиваются, итоги «шапки» соответствуют сумме строк, а в `report_db` инвалидируется или обновляется кэш выборок, которые зависят от дат и склада.

2 Сценарий «Расход/продажа с явным выбором партии». Создаётся документ расхода, в строках задаются ссылки на конкретные партии/серии, затем выполняется проведение. Проверяется отказ при превышении

доступного количества (ошибка уровня БД, транзакция откатывается); при успешном проведении остаток уменьшается, движения фиксируются, а в связанной продаже корректно проставляются суммы и скидки. Дополнительно сверяется связка sales – payments по номеру чека: сумма оплат не меньше суммы продажи.

3 Сценарий «Перемещение между складами». Документ с источником и приёмником проводится одной транзакцией. Ожидается синхронное списание на складе-источнике и приход на складе-получателе, корректная привязка партий на стороне приёмника, отсутствие минусовых остатков. В отчётах оборотов за период отражаются обе операции; срез остатков на дату показывает нулевое суммарное изменение по товару, но перераспределение по складам.

4 Сценарий «Откат документа». Для уже проведённых приходов, расходов и перемещений вызывается `rollback_document()`. Проверяется удаление связанных движений, возврат состояния в «черновик», восстановление остатков и пересчёт итогов «шапки». Отчёты после отката дают те же значения, что и до проведения.

5 Сценарий «Контроль доступа битовыми правами». Пользователю без бита на выполнение операции вызывается эндпоинт проведения/отката – API возвращает отказ, в БД состояние и движения неизменны. После выдачи соответствующего бита повторный вызов проходит успешно. Отдельно проверяется чтение отчётов: присутствие бита на построение отчётов даёт доступ к данным, отсутствие – отказ без выполнения SQL.

6 Сценарий «Последние продажи за сегодня». Выполняется параметризованный отчёт: в SQL подставляются даты, фильтры подразделений и типов оплат, затем формируется таблица и PDF. Результат проверяется с контрольными суммами: агрегаты по количеству и оплатам совпадают с суммой строк sales и payments за окно отчёта; сортировка по дате продажи корректна; поля «место продажи», «цена розничная», «скидка» согласованы с справочниками и строками документов. Генерируется PDF и сравнивается по ключевым полям с исходной таблицей.

7 Сценарий «Инвентаризация/корректировка остатков». Создаётся корректирующий документ, проводится с изменением количества до заданного уровня. Подтверждается, что суммарные движения за период объясняют получившийся остаток, а попытка провести корректировку в минус при отключённых разрешениях блокируется проверкой в БД.

8 Сценарий «Маркируемая номенклатура». Для товара со строгой прослеживаемостью создаётся расход без ссылки на серию – ожидается отказ на уровне ограничения; при указании серии документ проводится. В отчёте по срокам годности позиция исчезает из списка «просрочка/к критической дате» после корректного списания.

Для каждого сценария фиксируются входные данные, ожидаемые эффекты в таблицах документов и движений, ключевые выборки для сверки и, при необходимости, артефакты (PDF, дампы выборок). Совокупность сценариев покрывает полный жизненный цикл данных: от создания

первичных объектов и документов до отражения результатов в отчётном контуре и печатных формах.

### **5.3 Нагрузочные и конкурентные испытания**

Цель раздела – проверить, что типовые операции остаются интерактивными при параллельной работе и не приводят к нарушениям инвариантов. Испытания выполнялись на полном приложении: фронт, сервисный слой и три контура БД. Нагрузка создавалась генераторами сценариев, которые имитируют одновременную работу нескольких устройств и пользователей.

1 Профиль нагрузки. Моделировались пики в конце смены и при приёмке товара: параллельные продажи с разных устройств по разным складам; одновременный приход нескольких партий; проведение перемещений; фоновая генерация отчётов («последние продажи за сегодня», обороты за период). Для части прогонов намеренно создавались конфликты на одной партии/серии, чтобы оценить ожидания блокировок и корректность отказов.

2 Методика измерений. Для каждого сценария фиксировались: время отклика операций проведения и отката документов (среднее и 95-й перцентиль); число и длительность ожиданий блокировок при конкуренции за одну партию/склад; частота отказов по бизнес-правилам (недостаток остатка, отсутствующая серия) и по доступу; отсутствие отрицательных остатков и «висящих» ссылок по итоговым проверочным выборкам; стабильность построения отчётов при фоновой транзакционной активности.

3 Ключевые сценарии конкуренции. Параллельные продажи по разным партиям и складам. Ожидаемое поведение – отсутствие конфликтов, проведение в интерактивном режиме, консистентные остатки. Конкуренция за одну партию: две продажи одновременно пытаются списать одну и ту же партию. Ожидаемое поведение – одна транзакция проходит, вторая получает управляемый отказ на уровне БД без частичных изменений; отрицательных остатков нет. Массовое проведение документов поступления при одновременном построении отчёта по остаткам. Ожидаемое поведение – отчёт строится из актуальных движений или кэша, проведение не «зависает» из-за чтений; кэш отчётов переиспользуется или инвалидируется по правилам. Перемещение партии со склада А на склад Б и параллельная продажа этой же партии со склада А. Ожидаемое поведение – либо продажа ожидает завершения перемещения, либо получает отказ при недостатке остатка; двойного списания не возникает. Серийный откат ранее проведённых документов разных типов. Ожидаемое поведение – атомарный возврат состояний, удаление связанных движений, восстановление остатков без остаточных артефактов.

4 Критерии приёмки. Операции проведения/отката и построения типовых отчётов выполняются в интерактивном режиме; число конфликтов ограничено сценариями конкуренции и завершается корректным отказом без

частичных изменений; отсутствуют отрицательные остатки и несогласованные статусы; отчёты показывают значения, совпадающие с контрольными выборками по движениям за период.

На тестовом стенде сценарии с разнесёнными партиями отрабатывают без заметных ожиданий. При конкуренции за одну партию наблюдаются краткие ожидания, после чего одна из транзакций получает отказ на уровне СУБД; инварианты целостности не нарушаются. Параллельная отчётная нагрузка не блокирует проведение документов, выборки в отчётах согласуются с движениями. Предварительный вывод: схема и серверная логика выдерживают конкурентный доступ и соответствуют заявленным требованиям интерактивности и целостности для типовых пиков.

## **5.4 Автоматизация проверок и контроль регрессий**

Цель данного пункта – закрепить результаты ручных и нагрузочных испытаний в виде повторяемого набора автотестов, который запускается на полном приложении и гарантирует неизменность инвариантов при доработках схемы и серверной логики.

1 Тестовый конвейер. При запуске набора тестов стенд поднимается в контейнерах, выполняются миграции, загружается фиксированный сид-набор данных и последовательно прогоняются сценарии. Запуск унифицирован одной командой, результаты и артефакты (логи, CSV/PDF выборок, планы запросов) складываются в отдельный каталог.

2 Слои автотестов. Процедуры и триггеры: точечные вызовы `conduct_document()`, `rollback_document()`, проверка допустимых переходов статусов, реакций на граничные значения (нулевые и отрицательные количества, отсутствующие партии/серии). Интеграционные сценарии: полные цепочки «создание – редактирование строк – проведение – отчёт – откат», включая смешанные типы документов и параллельные операции. Отчёты и кэш: параметризованные запросы к `report_db` (включая «Последние продажи за сегодня») со сравнениями фактических результатов с эталонными снимками.

Инварианты, проверяемые после каждого сценария.

- отсутствуют отрицательные остатки и «висящие» ссылки;
- статусы документов согласованы с наличием/отсутствием движений;
- суммы по документу равны агрегату по строкам, скидки и оплаты сходятся с чековой частью;
- кэш отчётов либо актуализирован, либо корректно инвалидирован после проведения/отката.

3 Снимочные проверки отчётов. Для каждого отчёта хранится пара «сырой SQL + набор параметров» и эталонный результат (CSV). На прогоне строится фактический результат и сравнивается с эталоном с учётом стабильного порядка строк и форматирования дат. Несовпадение трактуется как регрессия.

4 Пороговые метрики. Фиксируются базовые бюджеты времени для проведения/отката и построения типовых отчётов (среднее и 95-й процентиль). Превышение порогов помечается как предупреждение либо ошибка в зависимости от сценария.

5 Безопасность миграций. Набор тестов выполняется на «чистой» схеме и поверх апгрейда с предыдущей версии. Дополнительно проверяются:

- сохранность данных справочников и ссылок;
- совместимость параметров отчётов;
- неизменность доменов и кодов состояний.

6 Конкурентные регрессии. В автозапуск включён короткий параллельный прогон: две продажи одной партии, перекрытие «перемещение – продажа», массовый приход на фоне отчётов. Ожидаемый результат – управляемые отказы без частичных изменений и отсутствие взаимных блокировок длительностью выше допустимого порога.

7 Артефакты и диагностика. В случае сбоя сохраняются планы выполнения проблемных запросов, журналы ошибок СУБД, снимки таблиц движений и остатков до/после операции. Это ускоряет локализацию причины и упрощает обратную связь при исправлениях.

8 Расширяемость набора. Каждый новый отчёт добавляется как «SQL + параметры + эталон», каждая новая процедура – как сценарий с явной фиксацией ожидаемых инвариантов. Правила оформления тестов входят в репозиторий вместе со схемой, что обеспечивает единый процесс контроля качества.

## 5.5 Заключение

Цель раздела – зафиксировать результаты испытаний стенда и формальные условия, при которых работа считается принятой.

Итоги проверки: на развёрнутом приложении с тремя контурами подтверждена корректность ключевых сценариев: создание, редактирование, проведение и откат документов; учёт партий и серий; расчёт остатков на дату; построение регламентных отчётов с параметрами; параллельная работа пользователей без частичных изменений данных. Процедуры проведения и отката отрабатывают атомарно, триггеры блокируют недопустимые переходы статусов и отрицательные количества, отчётный кэш актуализируется или инвалидируется предсказуемо.

Критерии функциональной приёмки:

- в схеме отсутствуют «висящие» ссылки; проверки целостности (FK, NOT NULL, CHECK) не допускают сохранения противоречивых данных;
- статусы документов согласованы с наличием/отсутствием движений; двойное проведение и откат уже откатанного документа невозможны;
- остатки не уходят в минус, списание просроченной партии и выбор несуществующей серии блокируются;

– отчёты формируются из согласованного источника: параметризованные запросы возвращают результаты, совпадающие с эталонными снимками; печатные формы на основании тех же данных;

– миграции проходят на «чистой» БД и поверх предыдущей версии без потери данных и доменных кодов; дампы/восстановление валидны;

– проверки доступа по операциям работают по битовым маскам: недопустимые действия возвращают отказ без побочных модификаций.

Критерии производительности (на стенде с типовым объёмом тестовых данных):

– проведение документа до 50 строк – не более 300 мс по 95-му перцентилю; откат – не более 200 мс;

– отчёт «Остатки на дату» – до 2 с; «Последние продажи за сегодня» – до 1 с при фиксированных параметрах;

– параллельные конфликты приводят к управляемым ошибкам сериализации без частичных записей; длительные блокировки не превышают заданного порога.

Артефакты приёмки:

– ER-модель, полный DDL, набор миграций и сид-данных;

– исходные тексты процедур/триггеров, список инвариантов, покрываемых ими;

– пакет автотестов с сценариями и эталонными CSV для отчётов, инструкции запуска;

– набор SQL отчётов с описанием параметров и PDF-шаблонами;

– протокол испытаний с замерами времени и результатами конкурирующих сценариев.

Ограничения и дальнейшие шаги: для ростовых объёмов возможна необходимость дополнительных частичных индексов и материализаций в отчётном контуре; длительные аналитические выборки следует обслуживать через кэш и фоновую перестройку. Эти меры не меняют модель данных и могут быть внедрены эволюционно.

При выполнении указанных критериев система считается соответствующей требованиям и готовой к переходу к следующему этапу – оформлению результатов и подготовке пользовательской и эксплуатационной документации.

## ЗАКЛЮЧЕНИЕ

В ходе выполнения курсового проекта была спроектирована и реализована реляционная база данных, предназначенная для автоматизации документооборота и учёта складских остатков предприятия. Разработка направлена на обеспечение прозрачности бизнес-процессов, надёжности хранения данных и повышение эффективности работы пользователей при выполнении операций с документами и партиями товаров.

На основании анализа существующих аналогов и изучения особенностей предметной области были сформулированы функциональные и технические требования к системе. В результате спроектирована инфологическая модель предметной области, проведена нормализация данных и построена физическая модель базы данных в среде PostgreSQL. Особое внимание уделено вопросам целостности и согласованности данных, что реализовано с использованием триггеров и хранимых процедур, обеспечивающих атомарное проведение и откат документов.

В рамках проекта разработана структура, включающая отдельные контуры для аутентификации, оперативного хранения и отчётной агрегации данных. Это позволило повысить гибкость архитектуры и обеспечить масштабируемость решения. Для развертывания системы подготовлено окружение на основе Docker и Alembic, что упрощает установку и сопровождение программного средства.

Проведено тестирование ключевых функций, включая создание, проведение и откат документов, а также формирование агрегированных отчётов. Проверка показала корректность работы процедур, соблюдение всех ограничений целостности и надёжность транзакционной логики.

Таким образом, поставленные в ходе курсового проектирования цели и задачи достигнуты в полном объёме. Разработанная база данных отвечает требованиям производительности, надёжности и расширяемости, а также может служить основой для дальнейшего развития программного комплекса и интеграции с прикладными сервисами предприятия.

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

- [1] Ubuntu. Официальная документация [Электронный ресурс]. – Режим доступа: <https://ubuntu.com/tutorials>. – Дата доступа: 01.10.2025.
- [2] Alembic. Документация по миграциям [Электронный ресурс]. – Режим доступа: <https://alembic.sqlalchemy.org/>. – Дата доступа: 03.10.2025.
- [3] PostgreSQL. Официальная документация по типу данных JSON и индексации GIN [Электронный ресурс]. – Режим доступа: <https://www.postgresql.org/docs/current/datatype-json.html>. – Дата доступа: 06.10.2025.
- [4] 1С: Управление торговлей 8. Официальная страница продукта [Электронный ресурс]. – Режим доступа: <https://v8.1c.ru/trade/>. – Дата доступа: 06.10.2025.
- [5] МойСклад. Документация и справка [Электронный ресурс]. – Режим доступа: <https://support.moysklad.ru/hc/ru>. – Дата доступа: 09.10.2025.
- [6] СБИС. Складской учёт – раздел справки [Электронный ресурс]. – Режим доступа: <https://saby.ru/help/inventory/info>. – Дата доступа: 10.10.2025.
- [7] FastAPI. Официальная документация [Электронный ресурс]. – Режим доступа: <https://fastapi.tiangolo.com/>. – Дата доступа: 12.10.2025.
- [8] SQLAlchemy. Документация по ORM [Электронный ресурс]. – Режим доступа: <https://docs.sqlalchemy.org/>. – Дата доступа: 13.10.2025.
- [9] Next.js. Официальная документация [Электронный ресурс]. – Режим доступа: <https://nextjs.org/docs>. – Дата доступа: 14.10.2025.
- [10] PostgreSQL. Официальная документация по хранимым процедурам [Электронный ресурс]. – Режим доступа: <https://www.postgresql.org/docs/current/sql-createprocedure.html>. – Дата доступа: 16.10.2025.
- [11] GS1. EAN/UPC barcodes [Электронный ресурс]. – Режим доступа: <https://www.gs1.org/standards/barcodes/ean-upc>. – Дата доступа: 17.10.2025.
- [12] PostgreSQL. Официальная документация по типам индексов GIN и GiST [Электронный ресурс]. – Режим доступа: <https://www.postgresql.org/docs/current/textsearch-indexes.html>. – Дата доступа: 20.10.2025.
- [13] PostgreSQL. Частичные индексы [Электронный ресурс]. – Режим доступа: <https://www.postgresql.org/docs/current/indexes-partial.html>. – Дата доступа: 21.10.2025.
- [14] PostgreSQL. Официальная документация по триггерам [Электронный ресурс]. – Режим доступа: <https://www.postgresql.org/docs/current/plpgsql-trigger.html>. – Дата доступа: 23.10.2025.
- [15] PostgreSQL. Уровни изоляции транзакций [Электронный ресурс]. – Режим доступа: <https://www.postgresql.org/docs/current/transaction-iso.html>. – Дата доступа: 24.10.2025.
- [16] React. Официальная документация [Электронный ресурс]. – Режим доступа: <https://react.dev/>. – Дата доступа: 24.10.2025.
- [17] Docker. Документация по Docker Compose [Электронный ресурс]. – Режим доступа: <https://docs.docker.com/compose/>. – Дата доступа: 26.10.2025.