**Оглавление**

[Введение 3](#_Toc185955180)

[Глава I. Теоретические аспекты разработки базы данных для управления ассортиментом и продажами магазина промышленного оборудования "ИнтерТех" 6](#_Toc185955181)

[1.1 Постановка задачи 6](#_Toc185955182)

[1.2 Описание основных функциональных подсистем 7](#_Toc185955183)

[1.3 Обоснование и выбор СУБД, средств моделирования и разработки 8](#_Toc185955184)

[1.4 Инфологическое моделирование предметной области 10](#_Toc185955185)

[1.4.1 Описание предметной области 10](#_Toc185955186)

[1.4.2 Проблемная область 12](#_Toc185955187)

[1.4.3 Инфологическая модель базы данных 13](#_Toc185955188)

[1.4.4 Концептуальная модель базы данных 16](#_Toc185955189)

[Глава II. Практические аспекты разработки базы данных информационной системы 22](#_Toc185955190)

[2.1 Создание таблиц в СУБД PostgreSQL 22](#_Toc185955191)

[2.2 Заполнение таблиц БД 25](#_Toc185955192)

[2.3 Создание запросов 26](#_Toc185955193)

[2.4 Создание триггеров 29](#_Toc185955194)

[2.5 Создание форм 32](#_Toc185955195)

[2.5 Полнофункциональный интерфейс 34](#_Toc185955196)

[Заключение 38](#_Toc185955197)

[Список использованных источников 40](#_Toc185955198)

[Приложение 43](#_Toc185955199)

[Приложение 1 43](#_Toc185955200)

[Приложение 2 46](#_Toc185955201)

[Приложение 3 51](#_Toc185955202)

[Приложение 4 53](#_Toc185955203)

[Приложение 5 55](#_Toc185955204)

[Приложение 6 59](#_Toc185955205)

Введение

В условиях современной экономики и высокой конкуренции на рынке промышленного оборудования, эффективность управления ассортиментом и продажами становится определяющим фактором успеха для торговых предприятий. Магазин "ИнтерТех", специализирующийся на реализации промышленного оборудования, не является исключением. В настоящее время магазин сталкивается с необходимостью обработки значительных объемов информации, включающих данные о товарах, поставщиках, клиентах и заказах. Использование устаревших методов ручного учета, разрозненных электронных таблиц или неспециализированного программного обеспечения приводит к снижению оперативности управления, возникновению ошибок в учете, затрудняет анализ данных и, как следствие, негативно влияет на прибыльность и конкурентоспособность магазина.

В связи с этим, *целью* данной работы является разработка базы данных, призванной автоматизировать процессы управления ассортиментом и продажами в магазине промышленного оборудования "ИнтерТех". Достижение этой цели предполагает решение ряда взаимосвязанных задач. Прежде всего, необходимо провести всесторонний анализ бизнес-процессов магазина, связанных с ассортиментом и продажами, чтобы выявить существующие проблемы и определить потребности в автоматизации. На основе проведенного анализа будут сформулированы требования к функциональности и производительности разрабатываемой базы данных, учитывающие специфику деятельности "ИнтерТех".

Следующим важным этапом является *выбор* системы управления базами данных (СУБД), оптимально подходящей для реализации поставленной задачи. При выборе будут учитываться такие факторы, как объем обрабатываемых данных, предполагаемая нагрузка на систему, стоимость владения и эксплуатации, а также наличие квалифицированных специалистов, способных работать с данной СУБД. После выбора СУБД будет спроектирована структура базы данных, разработаны инфологическая и даталогическая модели, обеспечивающие эффективное хранение и обработку информации о товарах, клиентах, поставщиках, заказах и продажах.

Непосредственная *реализация* базы данных в выбранной СУБД будет включать в себя создание таблиц, определение связей между ними, настройку индексов для оптимизации производительности, а также разработку необходимых SQL-запросов для выборки и обработки данных. В случае, если это предусмотрено техническим заданием, будет разработан и пользовательский интерфейс, обеспечивающий удобный доступ к данным и функциям системы. Завершающим этапом станет тестирование и отладка работы базы данных для проверки ее соответствия заявленным требованиям и обеспечения корректности обработки информации. Также будет разработана необходимая документация, включающая в себя руководство пользователя и описание структуры базы данных для администратора.

*Объектом* данного исследования является магазин промышленного оборудования "ИнтерТех", а предметом - процессы управления ассортиментом и продажами в этом магазине. Разработка базы данных позволит автоматизировать рутинные операции, связанные с учетом товаров, обработкой заказов и регистрацией продаж. Это, в свою очередь, приведет к повышению эффективности работы персонала, сокращению количества ошибок, оптимизации складских запасов и, как следствие, к увеличению прибыли магазина "ИнтерТех".

Для достижения поставленной цели и решения обозначенных задач в работе будут использованы следующие *методы* исследования:

* *Анализ документации:* изучение существующих документов, регламентирующих процессы управления ассортиментом и продажами в магазине "ИнтерТех".
* *Наблюдение:* непосредственное наблюдение за работой персонала магазина, связанной с учетом товаров, оформлением заказов и проведением продаж.
* *Моделирование:* построение моделей базы данных с использованием специализированных программных средств.
* *Тестирование:* проверка работоспособности и производительности разработанной базы данных с использованием тестовых данных и сценариев.

*Практическая значимость* данного исследования заключается в том, что разработанная база данных станет инструментом повышения эффективности и конкурентоспособности магазина "ИнтерТех" за счет автоматизации и оптимизации ключевых бизнес-процессов.

Глава I. Теоретические аспекты разработки базы данных для управления ассортиментом и продажами магазина промышленного оборудования "ИнтерТех"

1.1 Постановка задачи

Разработка базы данных для магазина "ИнтерТех" ставит своей целью создание единого информационного пространства, которое позволит автоматизировать и оптимизировать процессы, связанные с управлением ассортиментом и продажами. Внедрение современной системы управления базами данных (СУБД) предоставит инструменты для эффективного учета товаров, управления заказами, взаимодействия с клиентами и поставщиками, а также регистрации и анализа продаж.

**Учет товаров**. Учет подразумевает хранение информации о каждом товаре, включая наименование, артикул, описание, технические характеристики, производителя, страну-производитель, закупочную цену, розничную цену, штрих-код. Для удобного поиска и анализа необходимо предусмотреть возможность классификации товаров по категориям и подкатегориям. Кроме того, система должна отслеживать текущее количество товара на складе и уведомлять о товарах, количество которых достигло минимального уровня *(точки перезаказа)*.

Важной функцией разрабатываемой базы данных является **управление заказами**. Система должна позволять менеджерам формировать новые заказы от клиентов, включающих список товаров, их количество и предварительную стоимость. Также необходимо отслеживать статусы заказов на каждом этапе выполнения (*новый, в обработке, собран, отгружен, выполнен, отменен*) и предоставлять возможность редактирования заказов (*изменение состава, количества, контактных данных*). Хранение истории заказов по каждому клиенту обеспечит удобный доступ к этой информации в будущем.

Для эффективного взаимодействия с клиентами база данных должна обеспечивать **учет клиентов**, который включает в себя хранение информации о клиентах: наименование организации для юридических клиентов или ФИО для физических лиц, контактные данные, реквизиты, история заказов. Также должна быть предусмотрена возможность разделения клиентов на группы (*оптовые и розничные покупатели*) и учет скидок для постоянных клиентов.

**Учет поставщиков**. База данных должна хранить информацию о поставщиках (*наименование, контактные данные, условия сотрудничества*) и отслеживать историю поставок от каждого поставщика.

**Регистрация продаж**. Это еще одна ключевая функция системы. Она должна фиксировать факт продажи с указанием даты, клиента, проданных товаров, количества, цены и суммы продажи. При этом система должна автоматически списывать проданные товары со склада. Также необходимо предусмотреть возможность формирования чеков для клиентов.

База данных должна предоставлять возможность **формирования отчетов**. Это позволит руководству магазина получать отчеты по продажам за определенный период в разрезе товаров, клиентов, категорий, анализировать остатки товаров на складе, выявлять наиболее и наименее продаваемые товары, формировать отчеты.

Разрабатываемая база данных должна быть надежной, безопасной, обеспечивать целостность и непротиворечивость данных, а также иметь потенциал для дальнейшего развития и масштабирования.

1.2 Описание основных функциональных подсистем

Для реализации всех этих функций база данных будет состоять из нескольких взаимосвязанных функциональных подсистем: ***"Товары", "Клиенты", "Поставщики", "Заказы", "Продажи" и "Отчеты".***

Подсистема ***"Товары"*** является центральной, обеспечивающей ведение каталога товаров и позволяющей добавлять новые товары, редактировать информацию о существующих, классифицировать товары по категориям и отслеживать остатки на складе.

Подсистема ***"Клиенты"*** предназначена для ведения базы данных клиентов магазина, регистрации новых клиентов, хранения и редактирования информации о них, а также просмотра истории заказов и взаиморасчетов.

Подсистема ***"Поставщики"*** обеспечивает учет информации о поставщиках товаров, позволяя хранить данные о поставщиках, условиях поставок и контактных лицах.

Подсистема ***"Заказы"*** автоматизирует процесс формирования и обработки заказов от клиентов, позволяя создавать новые заказы, отслеживать их статусы, вносить изменения и контролировать оплату.

Подсистема ***"Продажи"*** фиксирует информацию о совершенных продажах, списывает проданные товары со склада и формирует чеки.

Подсистема ***"Отчеты"*** предоставляет инструменты для анализа данных и формирования отчетов по различным аспектам деятельности магазина, таким как продажи, остатки, клиенты и поставщики.

1.3 Обоснование и выбор СУБД, средств моделирования и разработки

Для реализации базы данных магазина "ИнтерТех" необходимо выбрать СУБД, отвечающую следующим требованиям: надежность, производительность, масштабируемость, безопасность, поддержка SQL. Исходя из этих требований, а также учитывая доступность и распространенность, для разработки данной базы данных была выбрана ***СУБД PostgreSQL***.

***PostgreSQL*** — это свободно распространяемая объектно-реляционная система управления базами данных. Она обладает рядом преимуществ:

* *Надежность и стабильность*: PostgreSQL известна своей надежностью и способностью работать под высокими нагрузками.
* *Широкая функциональность*: поддерживает множество типов данных, включая JSON, XML, массивы, геометрические типы и другие, что позволяет хранить разнообразную информацию о товарах и клиентах.
* *Развитый SQL*: поддержка стандарта SQL и наличие процедурных языков для создания хранимых процедур и функций.
* *Масштабируемость*: PostgreSQL хорошо масштабируется как вертикально (за счет увеличения ресурсов сервера), так и горизонтально (за счет распределения данных по нескольким серверам).
* *Безопасность*: наличие развитых средств обеспечения безопасности, включая разграничение прав доступа, шифрование данных, аудит действий пользователей.[[1]](#footnote-1)

Альтернативные варианты, такие как MySQL или коммерческие СУБД (Oracle, MS SQL Server), уступают PostgreSQL по совокупности факторов: MySQL менее функциональна, а коммерческие СУБД требуют значительных затрат на лицензирование.

Для проектирования инфологической и даталогической моделей базы данных будет использоваться ***ERwin Data Modeler***. ***ERwin Data Modeler*** является мощным CASE-средством для моделирования данных, которое обладает следующими преимуществами:

* *Поддержка различных нотаций*: ERwin поддерживает различные нотации моделирования данных, что позволяет выбрать наиболее подходящий вариант для проекта.
* *Обратная разработка*: возможность создания модели на основе существующей базы данных.
* *Удобный графический интерфейс*: интуитивно понятный интерфейс для создания и редактирования диаграмм.
* *Интеграция с другими средствами разработки*: ERwin может интегрироваться с другими средствами разработки, что упрощает процесс создания и сопровождения базы данных.

В качестве инструмента разработки был выбран ***Python*** с библиотекой ***PyQt***. *Python* - это высокоуровневый язык программирования, который отличается простотой изучения, лаконичностью и большой библиотекой модулей. Его преимуществами являются: простота и читаемость кода, большое количество библиотек для решения широкого круга задач *(включая разработку GUI, работу с базами данных, веб-разработку)*, кроссплатформенность и большое сообщество разработчиков. *PyQt*, в свою очередь, является реализацией фреймворка *Qt* для языка программирования *Python*. *Qt* - это мощный кроссплатформенный инструментарий для разработки программного обеспечения, который включает в себя обширный набор библиотек для создания графического пользовательского интерфейса (*GUI*) и других задач. *PyQt* позволяет использовать все возможности *Qt* в приложениях на *Python*. Выбор *PyQt* обусловлен следующими преимуществами:

* *Мощный и гибкий GUI*: *PyQt* предоставляет широкий набор виджетов и инструментов для создания современных и функциональных пользовательских интерфейсов.
* *Кросплатформенность*: приложения, разработанные с использованием *PyQt*, могут работать на различных операционных системах без изменения кода.
* ***Поддержка Qt Designer***: *Qt Designer* — это визуальный редактор, который позволяет создавать пользовательские интерфейсы методом "перетаскивания", что значительно ускоряет процесс разработки.
* *Хорошая интеграция с Python*: *PyQt* органично интегрируется с языком *Python*, позволяя использовать все его преимущества при разработке *GUI*.
* *Обширная документация и примеры*: для *PyQt* доступна обширная документация и множество примеров, облегчающих освоение и использование библиотеки.[[2]](#footnote-2)

Альтернативой *PyQt* могла бы быть стандартная библиотека *Python* ***Tkinter***. Однако *Tkinter* обладает более ограниченными возможностями и предоставляет меньший набор виджетов. Кроме того, интерфейсы, созданные с помощью *Tkinter*, часто выглядят устаревшими. Таким образом, выбор связки *Python* + *PyQt* является оптимальным решением для разработки функционального и современного пользовательского интерфейса для данной базы данных.

1.4 Инфологическое моделирование предметной области

1.4.1 Описание предметной области

Магазин "ИнтерТех" занимается розничной и оптовой торговлей промышленным оборудованием. Ассортимент магазина включает в себя различные категории товаров, такие как:

* Электроинструменты (дрели, перфораторы, шуруповерты, шлифовальные машины и т.д.)
* Станки (токарные, фрезерные, сверлильные и т.д.)
* Сварочное оборудование (инверторы, полуавтоматы, плазменные резаки и т.д.)
* Насосное оборудование (насосы, насосные станции)
* Садовая техника (газонокосилки, триммеры, культиваторы и т.д.)
* Генераторы (бензиновые, дизельные)
* Компрессоры (поршневые, винтовые)

Каждый товар обладает набором характеристик, специфичных для его категории.

Магазин работает как с физическими, так и с юридическими лицами. Заказы оформляются через менеджеров магазина. Клиенты могут иметь персональные скидки. Поставки товаров осуществляются от различных поставщиков. Организационная структура компании включает менеджеров по продажам, логистов и складских сотрудников. Менеджеры принимают заявки, консультируют клиентов, ведут базу данных. Логисты отвечают за координацию доставки, контролируют выполнение заказов. Складские сотрудники занимаются обработкой и подготовкой оборудования к отправке. Для работы используются таблицы Excel для учёта клиентов, заказов и складских остатков, а также такие каналы связи, как телефон, электронная почта и мессенджеры.

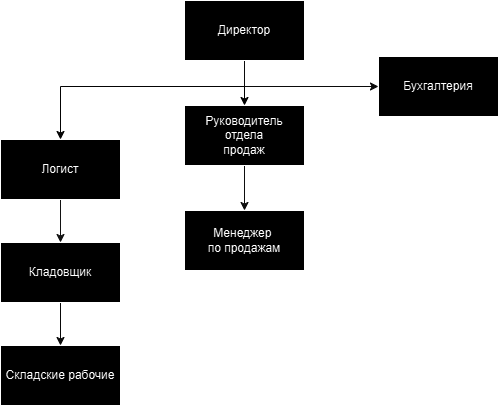
1.4.2 Проблемная область

Рисунок . Организационная структура.

В настоящее время в магазине "ИнтерТех" управление ассортиментом и продажами осуществляется с использованием разрозненных электронных таблиц и бумажных документов. Такой подход имеет ряд недостатков:

Сложность получения актуальной информации об остатках товаров: данные в разных таблицах могут дублироваться и противоречить друг другу, что затрудняет контроль за состоянием склада.

Высокая вероятность ошибок при ручном вводе данных: человеческий фактор приводит к ошибкам в наименованиях товаров, ценах, количестве, что может привести к финансовым потерям и недовольству клиентов.

Трудоемкость формирования заказов и отчетов: поиск нужных товаров и составление отчетов занимает много времени и требует значительных усилий.

Отсутствие единой базы данных клиентов и поставщиков: информация о клиентах и поставщиках разрознена, что затрудняет анализ и планирование.

Невозможность оперативного анализа продаж: отсутствие инструментов для анализа данных о продажах не позволяет своевременно реагировать на изменения спроса и оптимизировать ассортимент.

Внедрение базы данных позволит решить эти проблемы, автоматизировать рутинные операции, повысить эффективность управления и обеспечить руководство магазина актуальной и достоверной информацией.

1.4.3 Инфологическая модель базы данных

Инфологической моделью данных называют обобщенное неформальное описание создаваемой базы данных, выполненное с использованием естественного языка, математических формул, таблиц, графиков и других средств, понятных всем людям, работающим. Простыми словами, инфологическая модель – это потоки информации, сущности и связи данной области. В такой модели указываются связи между сущностями данной предметной области[[3]](#footnote-3)

Инфологическая модель представляет собой графическое описание предметной области, включающее основные сущности и связи между ними. Для наглядности она будет представлена в виде ER-диаграммы.

**Сущности и атрибуты:**

* Товар (Product):
  + ID\_Товара (Product\_ID) - уникальный идентификатор товара (первичный ключ)
  + Название (Name) - наименование товара
  + Описание (Description) - подробное описание товара
  + Артикул (VendorCode) - артикул товара
  + ID\_Категории (Category\_ID) - идентификатор категории товара (внешний ключ)
  + Цена\_закупки (Purchase\_Price) - цена закупки единицы товара
  + Цена\_продажи (Selling\_Price) - цена продажи единицы товара
  + Единица\_измерения (Unit) - единица измерения товара (шт., кг., м. и т.д)
  + ID\_Поставщика (Supplier\_ID) - идентификатор поставщика (внешний ключ)
* Категория (Category):
  + ID\_Категории (Category\_ID) - уникальный идентификатор категории (первичный ключ)
  + Название\_категории (Category\_Name) - название категории товара
  + Описание\_категории (Category\_Description) - описание категории
* Поставщик (Supplier):
  + ID\_Поставщика (Supplier\_ID) - уникальный идентификатор поставщика (первичный ключ)
  + Название\_поставщика (Supplier\_Name) - название компании-поставщика
  + Адрес (Address) - адрес поставщика
  + Телефон (Phone) - контактный телефон поставщика
  + Email (Email) - адрес электронной почты поставщика
  + ИНН (INN) - ИНН поставщика
  + Контактное\_лицо (Contact\_Person) - контактное лицо поставщика
* Клиент (Customer):
  + ID\_Клиента (Customer\_ID) - уникальный идентификатор клиента (первичный ключ)
  + Тип\_клиента (CustomerType) - Тип клиента: Юр. лицо, Физ. лицо
  + Наименования (Customer\_Name) - ФИО клиента (для физ. лиц) или название компании (для юр. лиц)
  + Телефон (Phone) - контактный телефон клиента
* Заказ (Order):
  + ID\_Заказа (Order\_ID) - уникальный идентификатор заказа (первичный ключ)
  + ID\_Клиента (Customer\_ID) - идентификатор клиента, сделавшего заказ (внешний ключ)
  + Дата\_заказа (Order\_Date) - дата оформления заказа
  + Сумма\_заказа (Total\_Amount) - общая сумма заказа
  + Статус\_заказа (Order\_Status) - статус заказа (Новый, В обработке, Собран, Отгружен, Выполнен, Отменен)
  + ID\_Сотрудника (Employee\_ID) - идентификатор сотрудника, оформившего заказ (внешний ключ)
  + Комментарий (Comment) - комментарий к заказу
* Состав\_заказа (Order\_Item):
  + ID\_Позиции (Order\_Item\_ID) - уникальный идентификатор позиции заказа (первичный ключ)
  + ID\_Заказа (Order\_ID) - идентификатор заказа (внешний ключ)
  + ID\_Товара (Product\_ID) - идентификатор товара (внешний ключ)
  + Количество (Quantity) - количество единиц товара в заказе
  + Цена (Price) - цена за единицу товара на момент заказа
* Склад (Warehouse):
  + ID\_Склада (Warehouse\_ID) - уникальный идентификатор склада (первичный ключ)
  + Название\_склада (Warehouse\_Name) - название склада
  + Адрес\_склада (Warehouse\_Address) - адрес склада
* Сотрудник (Employee):
  + ID\_Сотрудника (Employee\_ID) - уникальный идентификатор сотрудника (первичный ключ)
  + ФИО\_Сотрудника (Employee\_Name) - ФИО сотрудника
  + ID\_Должности (Position\_ID) - идентификатор должности (внешний ключ)
  + Телефон (Phone) - рабочий телефон сотрудника
  + Email (Email) - рабочий email сотрудника
* Должность (Position):
  + ID\_Должности (Position\_ID) - уникальный идентификатор должности (первичный ключ)
  + Название\_должности (Position\_Name) - название должности
* Остаток (Stock):
  + ID\_Остатка (Stock\_ID) - уникальный идентификатор записи об остатках (первичный ключ)
  + ID\_Товара (Product\_ID) - идентификатор товара (внешний ключ)
  + ID\_Склада (Warehouse\_ID) - идентификатор склада (внешний ключ)
  + Количество (Quantity) - количество товара на складе

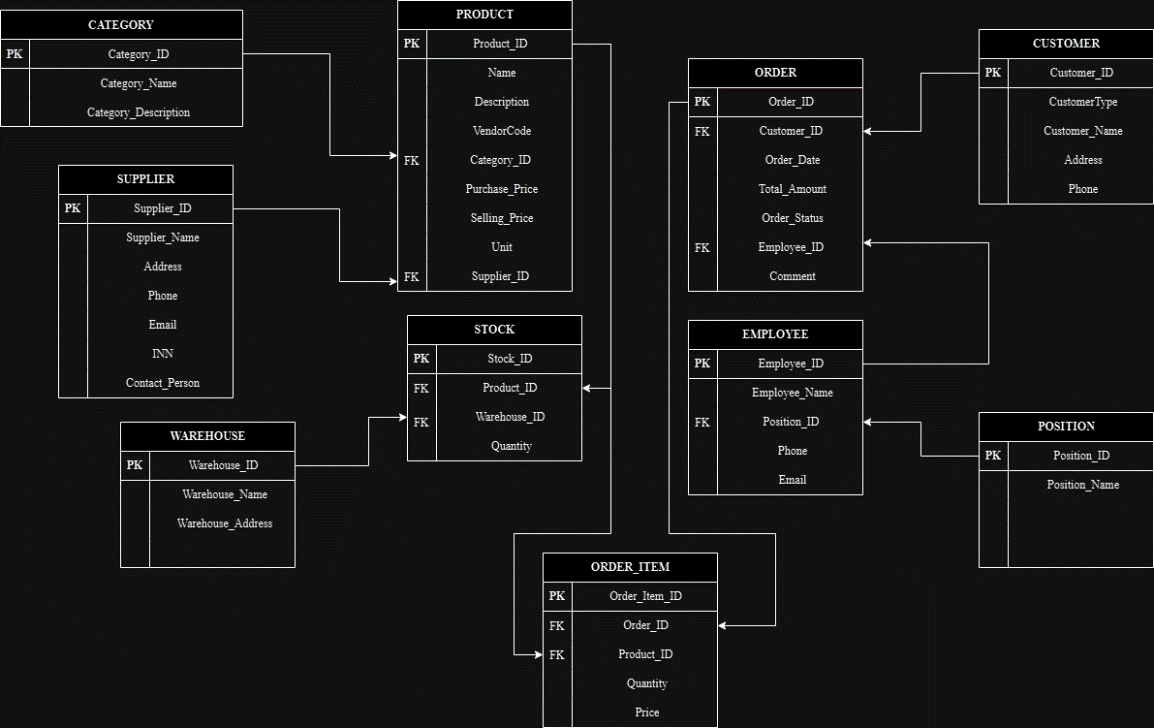


Рисунок 2. Инфологическая модель базы данных

1.4.4 Концептуальная модель базы данных

На основе проведенного анализа предметной области, описания существующих бизнес-процессов и выявленных проблем была разработана концептуальная модель базы данных для магазина промышленного оборудования "ИнтерТех". Данная модель описывает основные сущности предметной области, их атрибуты и взаимосвязи между ними на логическом уровне, независимо от конкретной СУБД.

Концептуальная модель включает в себя следующие таблицы:

**1. Таблица: Customers (Клиенты)**

|  |  |  |  |
| --- | --- | --- | --- |
| Поле | Тип данных | Описание | Ключ |
| Customer\_ID | UUID | Уникальный идентификатор клиента | PK |
| CustomerType | VARCHAR(20) | Тип клиента (Физ. лицо, Юр. лицо) |  |
| OrganizationName | VARCHAR(255) | Название организации (для юр. лиц) |  |
| Customer\_Name | VARCHAR(255) | ФИО клиента (для физ. лиц) |  |
| Address | TEXT | Адрес клиента |  |
| Phone | VARCHAR(20) | Телефон клиента |  |
| Email | VARCHAR(255) | Email клиента |  |
| INN | VARCHAR(20) | ИНН клиента (для юр. лиц) |  |

**2. Таблица: Warehouses (Склады)**

|  |  |  |  |
| --- | --- | --- | --- |
| Поле | Тип данных | Описание | Ключ |
| Warehouse\_ID | UUID | Уникальный идентификатор склада | PK |
| Warehouse\_Name | VARCHAR(255) | Название склада |  |
| Warehouse\_Address | TEXT | Адрес склада |  |

**3. Таблица: Suppliers (Поставщики)**

|  |  |  |  |
| --- | --- | --- | --- |
| Поле | Тип данных | Описание | Ключ |
| Supplier\_ID | UUID | Уникальный идентификатор поставщика | PK |
| Supplier\_Name | VARCHAR(255) | Название поставщика |  |
| Address | TEXT | Адрес поставщика |  |
| Phone | VARCHAR(20) | Телефон поставщика |  |
| Email | VARCHAR(255) | Email поставщика |  |
| INN | VARCHAR(20) | ИНН поставщика |  |
| Contact\_Person | VARCHAR(255) | Контактное лицо |  |

**4. Таблица: Products (Товары)**

|  |  |  |  |
| --- | --- | --- | --- |
| Поле | Тип данных | Описание | Ключ |
| Product\_ID | UUID | Уникальный идентификатор товара | PK |
| Name | VARCHAR(255) | Наименование товара |  |
| Description | TEXT | Описание товара |  |
| VendorCode | VARCHAR(50) | Артикул товара от поставщика |  |
| Category | ENUM('мало-габаритное', 'средне-габаритное', 'крупно-габаритное') | Категория товара |  |
| Purchase\_Price | DECIMAL(10, 2) | Цена закупки |  |
| Selling\_Price | DECIMAL(10, 2) | Цена продажи |  |
| Unit | VARCHAR(20) | Единица измерения |  |
| Supplier\_ID | UUID | Идентификатор поставщика (внешний ключ) | FK |

**5. Таблица: Orders (Заказы)**

|  |  |  |  |
| --- | --- | --- | --- |
| Поле | Тип данных | Описание | Ключ |
| Order\_ID | UUID | Уникальный идентификатор заказа | PK |
| Customer\_ID | UUID | Идентификатор клиента (внешний ключ) | FK |
| Order\_Date | DATE | Дата заказа |  |
| Total\_Amount | DECIMAL(10, 2) | Общая сумма заказа |  |
| Order\_Status | VARCHAR(50) | Статус заказа (Новый, В обработке, и т.д.) |  |
| Employee\_ID | UUID | Идентификатор сотрудника (внешний ключ) | FK |
| Comment | TEXT | Комментарий к заказу |  |
| Product\_ID | UUID | Идентификатор товара | FK |
| Quantity | INTEGER | Количество единиц товара |  |
| Price | DECIMAL (10,2) | Цена за единицу товара на момент заказа |  |

**6. Таблица: Employees (Сотрудники)**

|  |  |  |  |
| --- | --- | --- | --- |
| Поле | Тип данных | Описание | Ключ |
| Employee\_ID | UUID | Уникальный идентификатор сотрудника | PK |
| Employee\_Name | VARCHAR(255) | ФИО сотрудника |  |
| Position | VARCHAR(255) | Должность сотрудника |  |
| Phone | VARCHAR(20) | Телефон сотрудника |  |
| Email | VARCHAR(255) | Email сотрудника |  |

**7. Таблица: Stock (Остатки)**

|  |  |  |  |
| --- | --- | --- | --- |
| Поле | Тип данных | Описание | Ключ |
| Stock\_ID | UUID | Уникальный идентификатор записи | PK |
| Warehouse\_ID | UUID | Идентификатор склада (внешний ключ) | FK |
| Product\_ID | UUID | Идентификатор товара (внешний ключ) | FK |
| Quantity | INTEGER | Количество товара на складе |  |

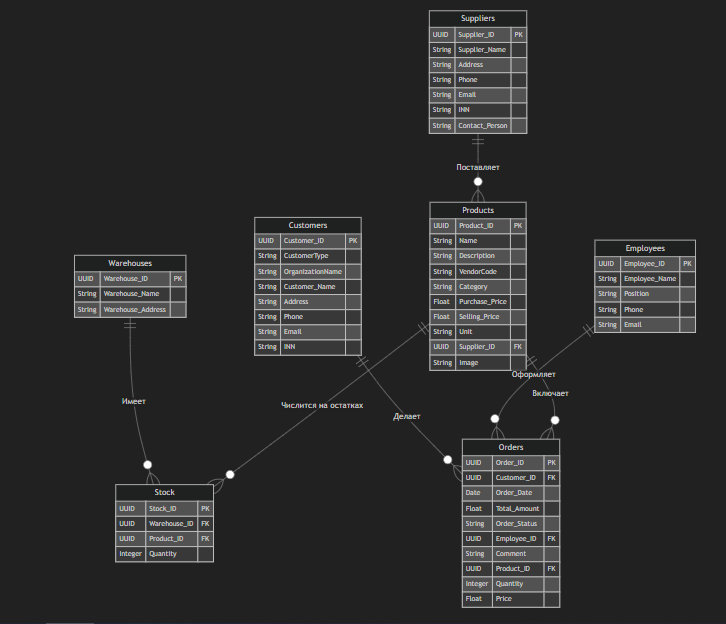


Рисунок 3. Визуальная схема базы данных

Разработанная концептуальная модель соответствует требованиям трех нормальных форм (3НФ):

*Первая нормальная форма (1НФ*): Все атрибуты таблиц являются атомарными (неделимыми). В нашей модели это требование выполняется, так как каждое поле содержит только одно значение.

*Вторая нормальная форма (2НФ):* Каждое неключевое поле таблицы полностью зависит от первичного ключа, а не от его части. Это требование выполняется, так как в таблицах с составными первичными ключами (Stock, Orders) отсутствуют неключевые поля, которые зависели бы только от части составного ключа.

*Третья нормальная форма (3НФ):* Отсутствуют транзитивные зависимости неключевых полей от первичного ключа. Другими словами, все неключевые поля зависят только от первичного ключа и не зависят друг от друга. В нашей модели это требование также выполняется.

Вывод по главе I

В первой главе были детально рассмотрены теоретические аспекты разработки базы данных для управления ассортиментом и продажами магазина "ИнтерТех". Проведен анализ предметной области, включающий описание деятельности магазина, его организационной структуры, основных бизнес-процессов и существующих проблем. Сформулированы цели и задачи проекта, обоснован выбор СУБД PostgreSQL, а также инструментов моделирования (ERwin Data Modeler) и разработки интерфейса (Python, PyQt). Особое внимание уделено инфологическому моделированию, результатом которого стала концептуальная модель базы данных, полностью описывающая предметную область в терминах сущностей, атрибутов и связей. Разработанная модель представлена в виде ER-диаграммы и в табличном формате, а также проверена на соответствие требованиям трех нормальных форм. Таким образом, первая глава закладывает прочный фундамент для дальнейшей практической реализации проекта, предоставляя детальное описание предметной области и четко сформулированную концептуальную модель базы данных.

Глава II. Практические аспекты разработки базы данных информационной системы

На данном этапе осуществляется непосредственное создание базы данных в выбранной СУБД *PostgreSQL* на основе разработанной логической модели (п. 1.4.4). Для каждой таблицы, описанной в модели, будет создан соответствующий SQL-код, определяющий структуру таблицы, типы данных полей, первичные и внешние ключи, а также другие необходимые ограничения.

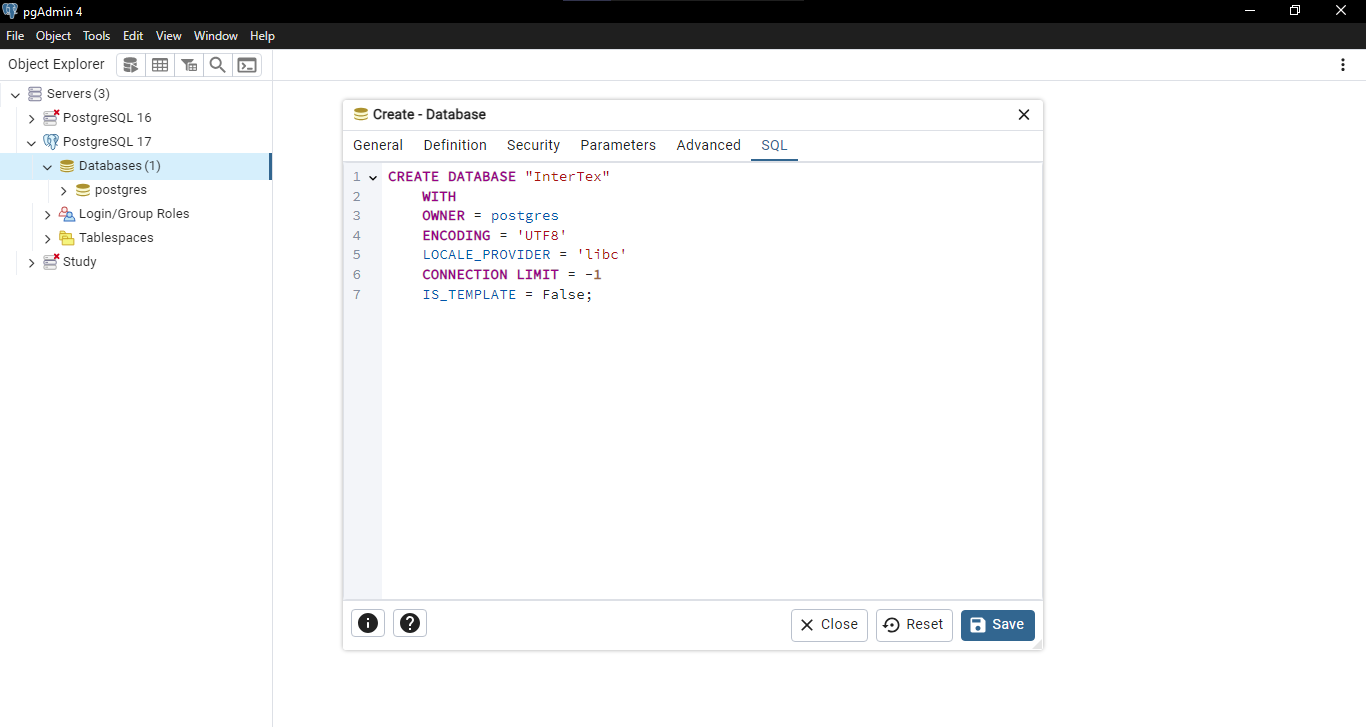
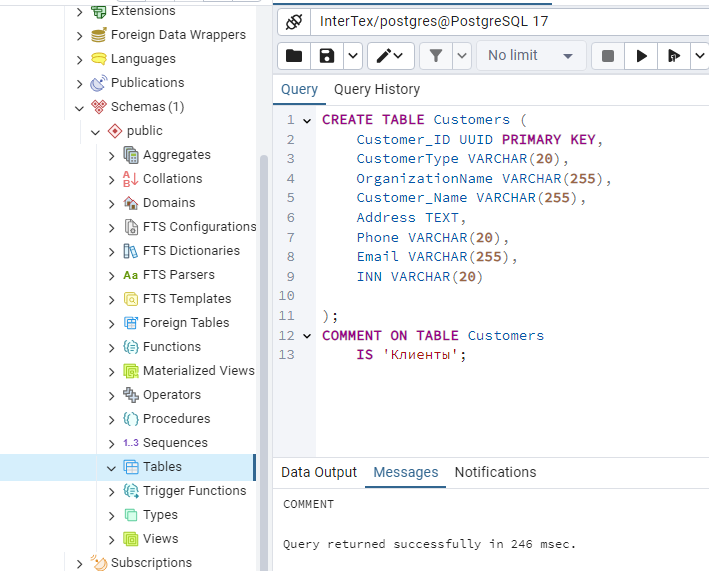


Рисунок 4. Создание БД в PostgreSQL

2.1 Создание таблиц в СУБД PostgreSQL

Первым действием создаем базы данных и соответствующие таблицы в ней. Затем заполняем их данными.

Создание таблиц представлено на рисунках 5-11. Полный код для создание таблиц представлен в приложении (*Приложение 1*).



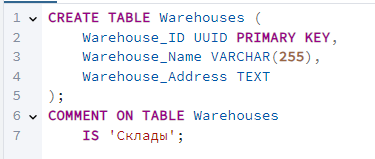
Рисунок 5. Создание таблицы Customers(Клиенты)

Рисунок 6. Создание таблицы Warehouse(Склад)

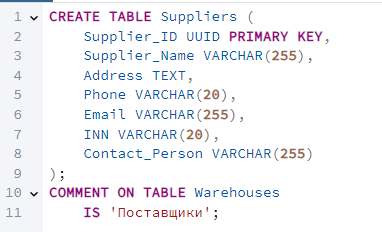


Рисунок 7. Создание таблицы Suppliers(Поставщики)

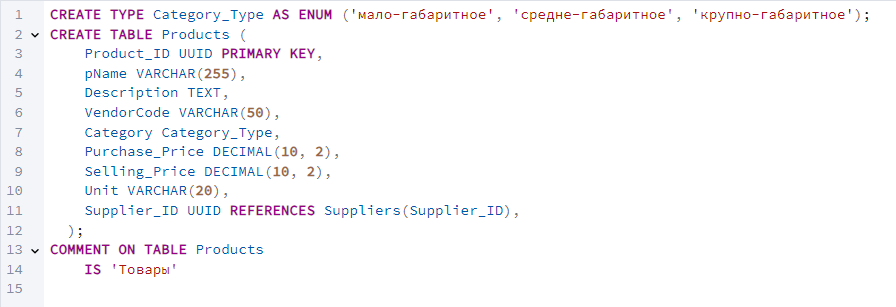


Рисунок 8. Создание таблицы Products(Товары)

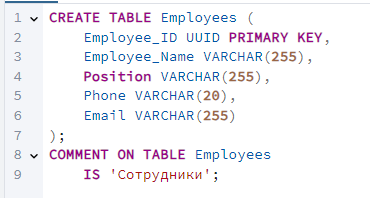


Рисунок 9. Создание таблицы Employees(Сотрудники)

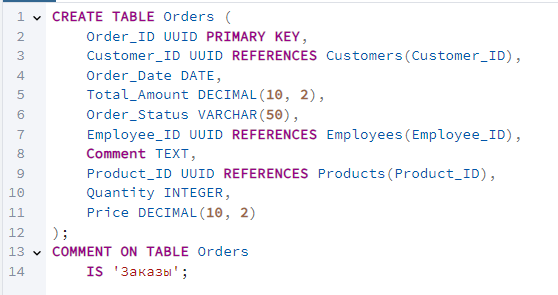


Рисунок 10. Создание таблицы Orders(Заказы)

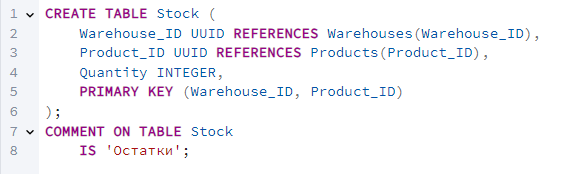


Рисунок 11. Создание таблицы Stock(остатки)

2.2 Заполнение таблиц БД

После создания всех таблиц приступаем к их заполнению. Все таблицы представлены на рисунках 12-18. Полный скрипт заполнения таблиц находится в приложении(Приложение 2).

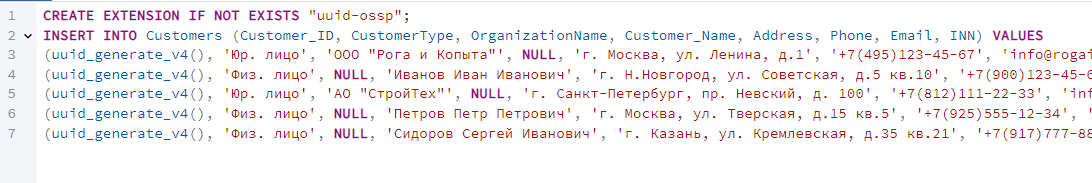
[[4]](#footnote-4)

Рисунок 12. Заполнение таблицы Customers

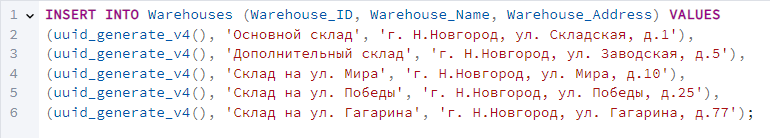


Рисунок 13. Заполнение таблицы Warehouse

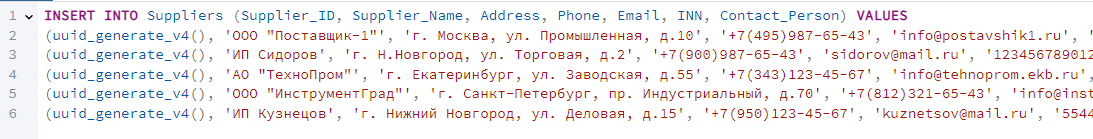


Рисунок 14. Заполнение таблицы Suppliers

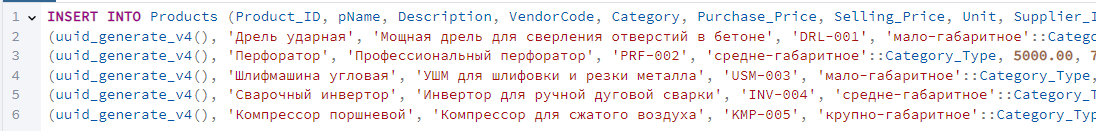


Рисунок 15. Заполнение таблицы Products

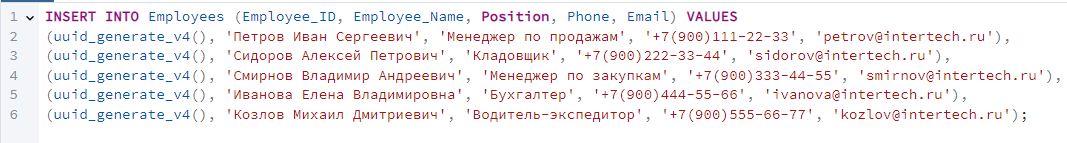


Рисунок 16. Заполнение таблицы Employees

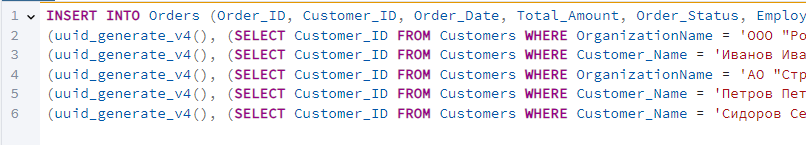


Рисунок 17. Заполнение таблицы Orders

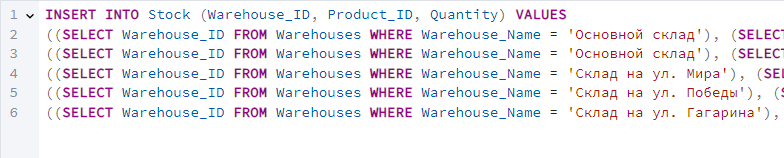


Рисунок 18. Заполнение таблицы Stock

2.3 Создание запросов

**SQ**L (Structured Query Language - язык структурированных запросов) - это декларативный язык программирования, который используется для управления данными в реляционных базах данных. SQL-запросы позволяют выполнять различные операции с данными, такие как:

* Выборка (*SELECT*): Извлечение данных из одной или нескольких таблиц с возможностью фильтрации, сортировки и группировки.
* Вставка (*INSERT*): Добавление новых данных в таблицы.
* Обновление (*UPDATE*): Изменение существующих данных в таблицах.
* Удаление (DELETE): Удаление данных из таблиц.
* Создание и изменение структуры БД (*CREATE, ALTER, DROP*): Cоздание новых таблиц, изменение структуры, удаление.[[5]](#footnote-5)

Для демонстрации работы с базой данных магазина "ИнтерТех" сформируем 5 основных запросов:

1. Выборка всех товаров определенной категории (Рисунок 18)
2. Выборка информации о клиенте по его ID (Рисунок 19)
3. Выборка всех заказов, оформленных конкретным сотрудником за определённый период (Рисунок 20)
4. Выборка товаров, остаток которых на складе меньше заданного количества (Рисунок 21)
5. Выборка полной информации о составе заказа по его номеру (Рисунок 22)

Коды для реализации запросов и фильтров представлены в приложении (*Приложение 3*)

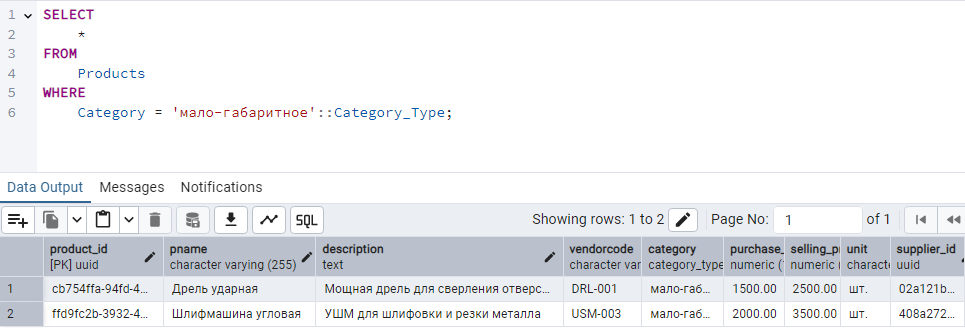


Рисунок 18. Выборка всех товаров определенной категории(мало-габаритные)

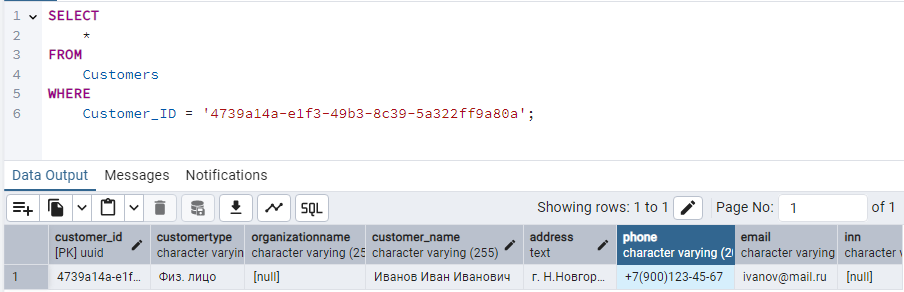


Рисунок 19. Выборка информации о клиенте по его ID

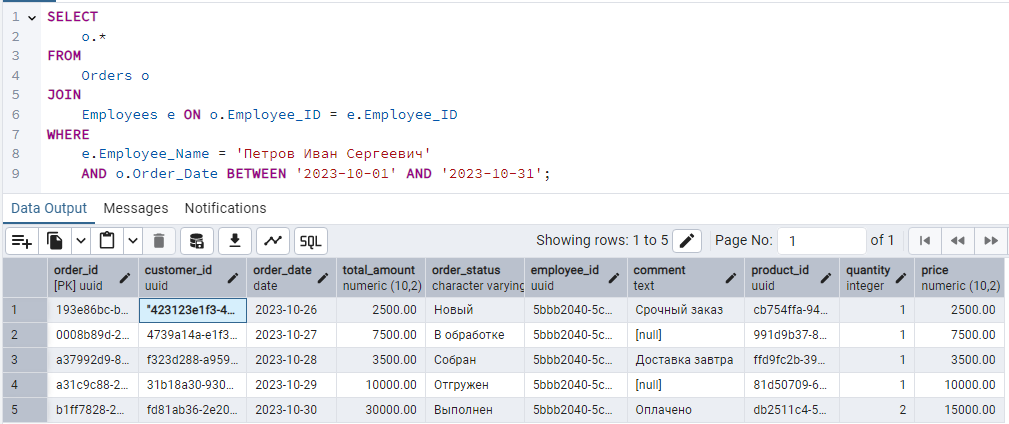


Рисунок 20. Выборка всех заказов, оформленных конкретным сотрудником за определённый период

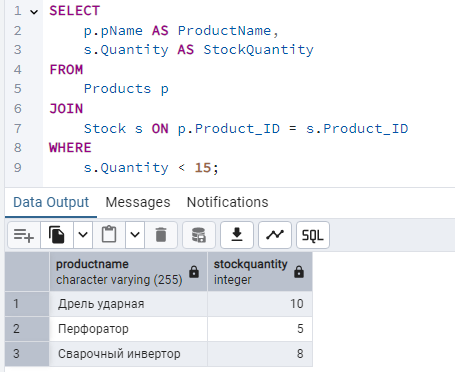
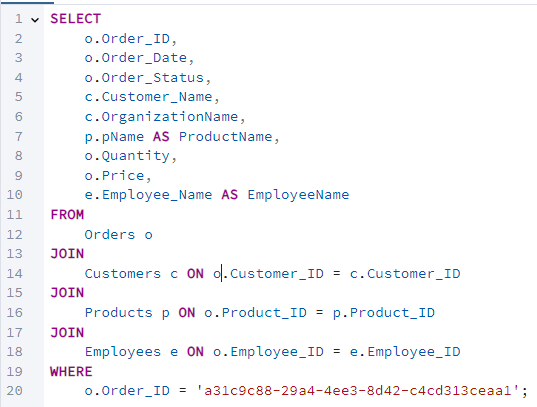


Рисунок 21. Выборка товаров, остаток которых на складе меньше заданного количества



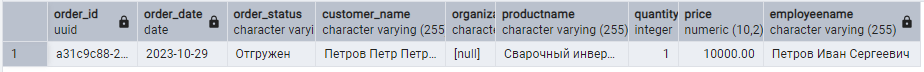


Рисунок 22. Выборка полной информации о составе заказа по его номеру

2.4 Создание триггеров

**Триггер** — это хранимая процедура в базе данных, которая автоматически вызывается при возникновении определенного события, связанного с таблицей. Триггеры используются для обеспечения целостности данных, автоматизации операций и реализации сложной бизнес-логики на уровне базы данных.[[6]](#footnote-6) В PostgreSQL триггеры создаются с помощью команды *CREATE TRIGGER.*

Коды для создания триггеров представлены в приложении (*Приложение 4*)

**Триггер 1:** Автоматический пересчёт суммы заказа при добавлении или изменении товара в заказе.

При добавлении нового товара в заказ (в таблицу *Orders*) или изменении количества (*Quantity*) или цены (*Price*) товара в заказе необходимо автоматически пересчитывать общую сумму заказа (*Total\_Amount*) в таблице заказов.

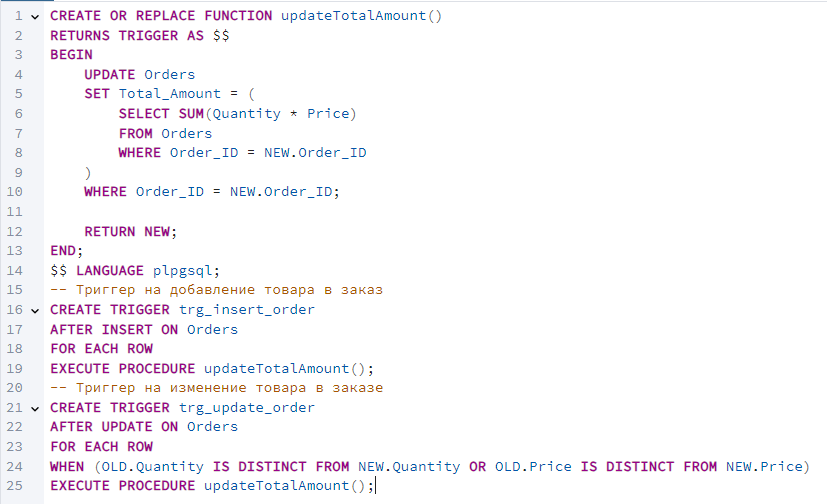


Рисунок 23. Создание триггера пересчета суммы заказа при его изменении

Этот триггер предназначен для автоматического пересчета общей суммы заказа в таблице «Заказы» при добавлении нового товара в заказ или изменении количества/цены существующего товара.

Триггер trg\_insert\_order срабатывает после (AFTER) добавления новой строки (INSERT) в таблицу Orders, а триггер trg\_update\_order срабатывает после (AFTER) обновления существующей строки (UPDATE) в таблице Orders. Триггер trg\_update\_order имеет дополнительное условие WHEN, он сработает только в том случае, если изменилось количество (Quantity) или цена (Price) товара в заказе.

Функция updateTotalAmount() получает данные о добавленной или измененной строке в таблице Orders. Эти данные доступны через специальную переменную NEW. С помощью SQL-запроса UPDATE функция обновляет поле Total\_Amount в таблице Orders для заказа, к которому относится измененная строка. Функция возвращает NEW. Это важно для триггеров типа AFTER, чтобы изменения, внесенные в строку, были сохранены.

**Триггер 2:** Контроль остатков при оформлении заказа.

При добавлении товара в заказ (в таблицу Orders) необходимо проверять, достаточно ли товара на складе. Если товара недостаточно, нужно отменять операцию и выдавать сообщение об ошибке.



Рисунок 24. Создание триггера контроля остатков

Этот триггер предназначен для контроля остатков товара на складе при оформлении заказа. Он проверяет, достаточно ли товара на складе, прежде чем разрешить добавление товара в заказ.

Триггер *trg\_check\_stock* срабатывает перед (*BEFORE*) добавлением новой строки (*INSERT*) в таблицу *Orders*. Триггер вызывает функцию *checkStockQuantity*(). Функция *checkStockQuantity*() получает данные о добавляемой строке в таблице *Orders*. Эти данные доступны через специальную переменную *NEW*.

С помощью SQL-запроса *SELECT* *INTO* функция получает количество товара (*Quantity*) на "Основном складе" (*Warehouse\_ID* которого соответствует названию "Основной склад") для товара, который добавляется в заказ (*Product\_ID* которого равен *NEW.Product\_ID*). Результат запроса сохраняется в переменную *stock\_quantity*.

Если товара на складе не найдено или количество товара на складе меньше, чем количество, указанное в заказе (*stock\_quantity < NEW.Quantity*), то генерируется исключение с помощью команды *RAISE EXCEPTION*. Это исключение прерывает операцию добавления товара в заказ и выводит сообщение об ошибке, в котором указывается название товара, его текущее количество на складе и требуемое количество.

Если товара на складе достаточно, функция завершает работу и возвращает *NEW*, разрешая тем самым добавление товара в заказ.

2.5 Создание форм

В контексте баз данных, формы — это элементы графического пользовательского интерфейса (*GUI*), которые предоставляют удобный способ взаимодействия с данными. Формы позволяют пользователям просматривать, добавлять, изменять и удалять данные в базе данных без необходимости написания SQL-запросов вручную.

*PyQt* - это библиотека *Python*, которая является реализацией фреймворка *Qt*. *Qt* предоставляет обширный набор инструментов для создания кроссплатформенных приложений с графическим интерфейсом. Именно *PyQt* я и буду использовать для создания форм

Полные скрипт создания форм на *Python* находятся в приложении(*Приложение 5*).

**Форма «Состав заказа»**

Данная форма предназначена для получения детальной информации о конкретном заказе.

*Описание формы:*

* Поле ввода "ID заказа": Текстовое поле для ввода уникального идентификатора заказа.
* Кнопка "Загрузить": По нажатию на кнопку выполняется запрос к базе данных для получения информации о заказе с указанным ID.
* Таблица "Состав заказа": Отображает информацию о товарах, входящих в заказ:
* ID товара
* Наименование товара
* Количество
* Цена за единицу
* Сумма (количество \* цена)
* Информация о клиенте:
* ФИО клиента (или название организации)
* Информация о сотруднике:
* ФИО сотрудника, оформившего заказ
* Общая сумма заказа: Рассчитывается автоматически.

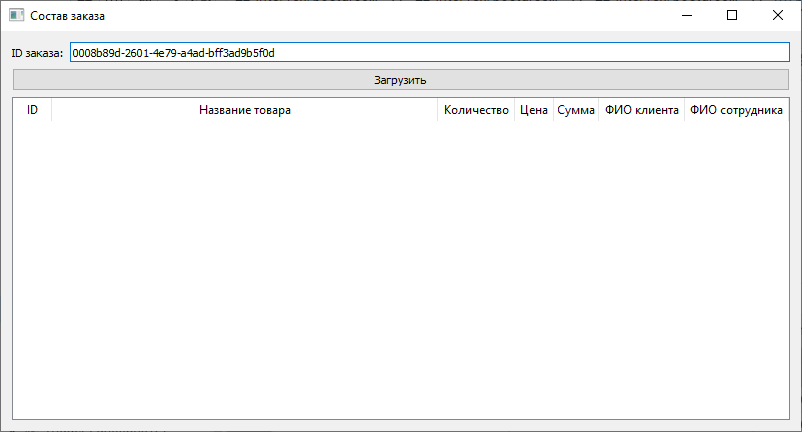


Рисунок 25. Форма «Состав заказа»

**Форма авторизации**

Данная форма предназначена для аутентификации пользователей перед предоставлением им доступа к базе данных

**Описание формы:**

* Поле ввода "Логин": Текстовое поле для ввода имени пользователя.
* Поле ввода "Пароль": Текстовое поле для ввода пароля.
* Кнопка "Войти": По нажатию на кнопку выполняется проверка введенных данных.
* Кнопка "Отмена": Закрывает форму авторизации.

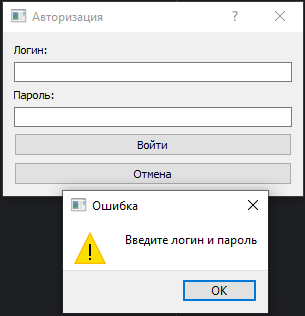


Рисунок 26. Форма авторизации

2.5 Полнофункциональный интерфейс

*Интерфейс пользователя* (UI - User Interface) - это совокупность средств и методов, при помощи которых пользователь взаимодействует с различными системами, программами или устройствами. В контексте данного проекта, интерфейс обеспечивает взаимодействие пользователя с базой данных.

*Графический интерфейс пользователя* (GUI - Graphical User Interface) - это тип интерфейса, который позволяет пользователям взаимодействовать с электронными устройствами через графические элементы, такие как окна, кнопки, меню, значки и т.д., вместо использования текстовых команд.[[7]](#footnote-7)

Полный код представлен в приложении (*Приложение 6*)

Разработка полнофункционального интерфейса для базы данных магазина "ИнтерТех" включает в себя:

1. Форму авторизации: Для входа в систему и разграничения прав доступа.
2. Главное окно: Содержит меню или кнопки для доступа к основным функциям приложения.
3. Формы для просмотра и редактирования данных: Для каждой таблицы базы данных (Customers, Warehouses, Suppliers, Products, Orders, Employees, Stock) создается отдельная форма, позволяющая просматривать, добавлять, изменять и удалять записи.
4. Форму "Состав заказа": Для просмотра детальной информации о заказе.
5. Форму поиска: Для поиска клиентов по имени или телефону.
6. Форму оформления заказа: Для создания новых заказов с возможностью выбора клиента, товаров, сотрудника, указания даты, количества, цены и комментария.
7. Форму заказа у поставщика: Для оформления заказа на поставку товаров.

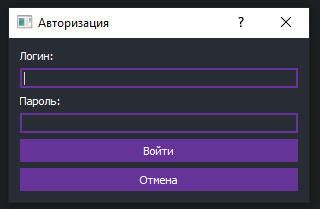


Рисунок 27. Окно авторизации

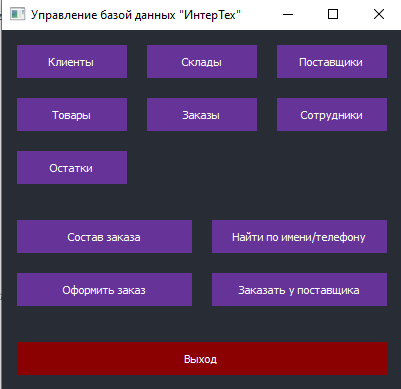


Рисунок 28. Главное меню

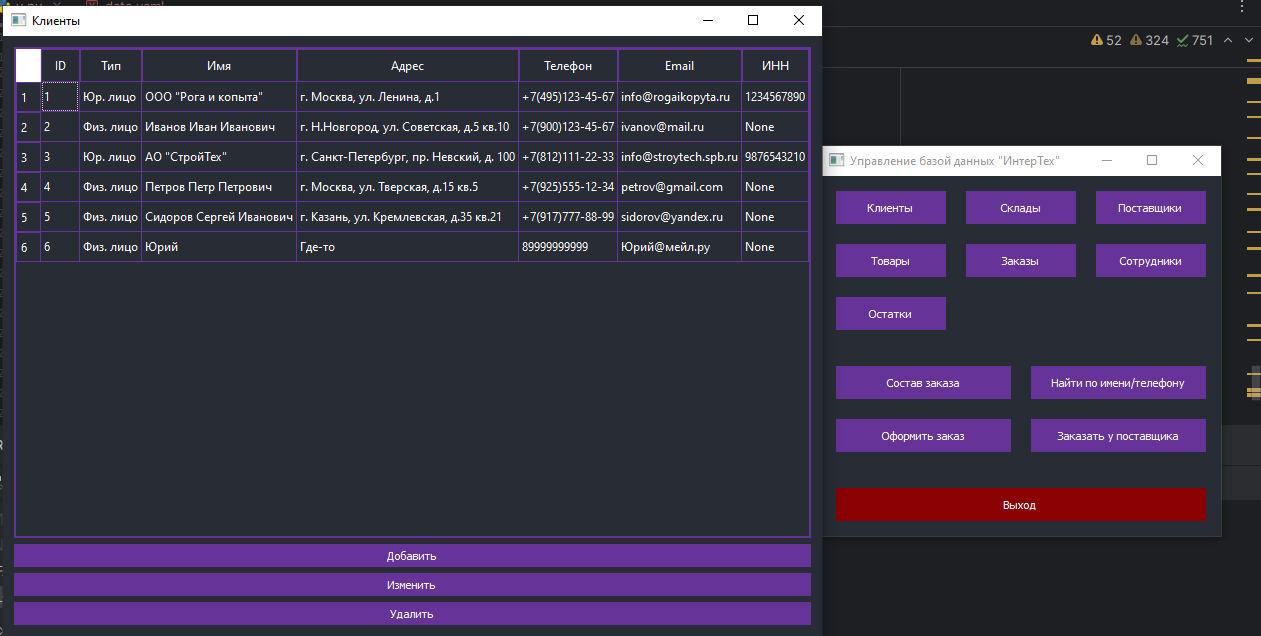


Рисунок 29. Форма для таблицы «Клиенты»

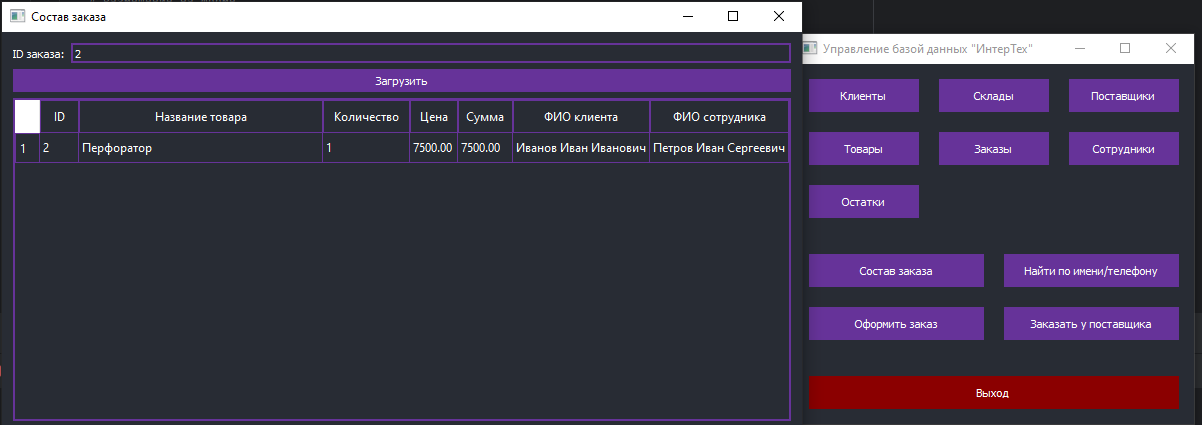


Рисунок 29. Форма для просмотра деталей заказа

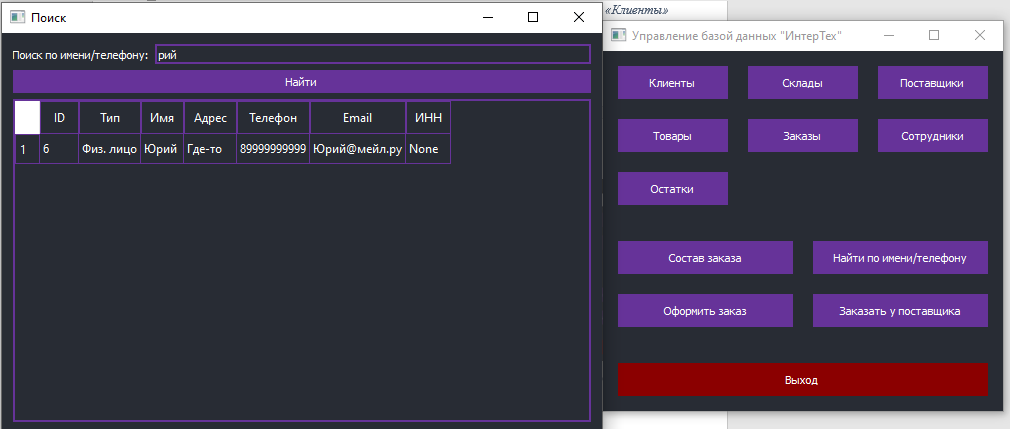


Рисунок 30. Форма поиска клиентов

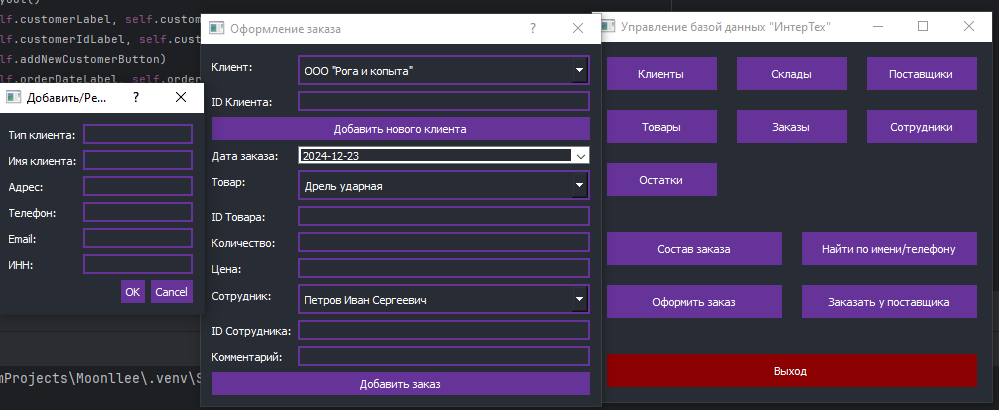


Рисунок 31. Форма для создание нового заказа

Вывод по главе II

Вторая глава посвящена практической реализации базы данных, спроектированной в первой главе. На основе концептуальной модели были созданы таблицы в СУБД PostgreSQL с использованием языка SQL. Определены типы данных для каждого поля, а также первичные и внешние ключи для обеспечения целостности данных. Таблицы были заполнены тестовыми данными для демонстрации работоспособности системы. Разработаны и протестированы SQL-запросы, реализующие основные операции с данными: выборку, добавление, изменение и удаление. Важным элементом практической части стала разработка графического интерфейса пользователя на языке Python с использованием библиотеки PyQt. Созданы формы, предоставляющие удобный доступ к данным и функциям системы, включая форму авторизации, формы для просмотра и редактирования данных всех таблиц, а также форму "Состав заказа". Кроме того, реализованы два триггера: один для автоматического пересчета суммы заказа, другой - для контроля остатков на складе, что демонстрирует возможности автоматизации бизнес-процессов. Таким образом, во второй главе была успешно выполнена практическая реализация базы данных, включая создание таблиц, заполнение их данными, разработку SQL-запросов, создание триггеров и разработку графического интерфейса, обеспечивающего удобное взаимодействие с базой данных.

Заключение

В первой главе данной курсовой работы были рассмотрены теоретические аспекты разработки базы данных для управления ассортиментом и продажами магазина промышленного оборудования "ИнтерТех". Был проведен анализ предметной области, который выявил ряд проблем, связанных с использованием устаревших методов учета, таких как разрозненные электронные таблицы. Эти проблемы приводили к сложности получения актуальной информации об остатках, высокой вероятности ошибок при ручном вводе, трудоемкости формирования заказов и отчетов, а также невозможности оперативного анализа продаж.

На основе проведенного анализа была сформулирована цель работы - разработка базы данных для автоматизации процессов управления ассортиментом и продажами в магазине "ИнтерТех". Также были определены основные задачи, включающие в себя выбор СУБД, проектирование инфологической модели, реализацию базы данных, а также разработку пользовательского интерфейса. В качестве СУБД была выбрана PostgreSQL, как надежная, производительная, масштабируемая и свободно распространяемая система, поддерживающая все необходимые функции. Для моделирования данных выбрано CASE-средство ERwin Data Modeler, а для разработки интерфейса - язык программирования Python с библиотекой PyQt5.

В результате, была спроектирована концептуальная (логическая) модель базы данных, включающая в себя таблицы "Клиенты", "Склады", "Поставщики", "Товары", "Заказы", "Сотрудники" и "Остатки", а также связи между ними. Разработанная модель соответствует требованиям трех нормальных форм, что обеспечивает целостность и непротиворечивость данных.

Во второй главе были рассмотрены практические аспекты реализации базы данных. Были созданы таблицы в СУБД PostgreSQL с использованием языка SQL, определены типы данных, первичные и внешние ключи. Таблицы были заполнены тестовыми данными. Кроме того, были разработаны и протестированы SQL-запросы для выборки, добавления, изменения и удаления данных. Также были созданы формы графического интерфейса пользователя, включая форму авторизации, формы для просмотра и редактирования данных таблиц, а также форму "Состав заказа". Разработанный интерфейс, выполненный в единой стилистике, предоставляет удобный доступ к данным и функциям системы, позволяя пользователям эффективно взаимодействовать с базой данных без необходимости знания языка SQL. В заключение, созданные формы на данный момент позволяют вносить изменения в базу данных, просматривать информацию, а так же добавлять новые записи.

Исходя из всего изложенного, можно сделать вывод, что разработанная база данных оптимизирует рабочий процесс рассматриваемой в курсовом проекте организации, обеспечит надежное хранение данных и повысит качество работы сотрудников.

В заключение можно сказать, что задачи курсового проекта были выполнены в полном объеме, цель исследования достигнута.

Список использованных источников

1. Работа с базами данных SQL в Python: подробный гайд // Timeweb Cloud. URL: https://timeweb.cloud/tutorials/python/rabota-s-bazami-dannyh-sql-v-python (дата обращения: 09.12.2024).
2. Прохоренок, Н. А. Python 3 и PyQt 5. Разработка приложений. — 2-е изд., перераб. и доп. / Н. А. Прохоренок, В. А. Дронов. — СПб.: БХВ-Петербург, 2018. — 832 с.: ил. — (Профессиональное программирование) (дата обращения: 09.12.2024).
3. Создаём и наполняем базу данных SQLite в Python // Журнал «Код». URL: https://thecode.media/sqlite-py/ (дата обращения: 07.12.2024).
4. Python + SQLite: основы работы с базами данных // Proglib.io. URL: https://proglib.io/p/samouchitel-po-python-dlya-nachinayushchih-chast-22-osnovy-raboty-s-sqlite-2023-06-15 (дата обращения: 09.12.2024).
5. Работа с базами данных в Python // Skypro. URL: https://sky.pro/wiki/python/rabota-s-bazami-dannyh-v-python (дата обращения: 07.12.2024).
6. Как подружить Python и базы данных SQL. Подробное руководство // Proglib.io. URL: https://proglib.io/p/kak-podruzhit-python-i-bazy-dannyh-sql-podrobnoe-rukovodstvo-2020-02-27 (дата обращения: 09.12.2024).
7. Создание базы данных SQL для pyodbc // Microsoft Learn. URL: https://learn.microsoft.com/ru-ru/sql/connect/python/pyodbc/step-2-create-a-sql-database-for-pyodbc-python-development?view=sql-server-ver16& (дата обращения: 09.12.2024).
8. База данных Python: обзор совместимых и как их подключить // Foxminded.ua. URL: https://foxminded.ua/ru/baza-dannyh-python (дата обращения: 09.12.2024).
9. Python. Лекция 10. Работа с базой данных // Ideafix.su. URL: https://ideafix.su/wp-content/uploads/2012/05/Python-10.pdf (дата обращения: 07.12.2024).
10. Гвоздева, Т. В., Баллод, Б. А. Проектирование информационных систем. Стандартизация: учебное пособие для вузов. 2-е изд., стер. Санкт-Петербург: Лань, 2023. 252 с. ISBN 978-5-8114-7963-4. Текст: электронный // Лань: электронно-библиотечная система. URL: https://e.lanbook.com/book/169810 (дата обращения: 09.12.2024).
11. Федорова, Г. Н. Информационные системы: учебник для студентов учреждений среднего профессионального образования. 3-е изд., стер. Москва: Издательский центр «Академия», 2013. 208 с. ISBN 978-5-7695-9642-1. Текст: электронный // Издательский центр «Академия». URL: https://academia-moscow.ru/ftp\_share/\_books/fragments/fragment\_21064.pdf (дата обращения: 12.12.2024).
12. Гордеев, С. И., Волошина, В. Н. Организация баз данных в 2 ч. Часть 1: учебник для среднего профессионального образования. 2-е изд., испр. и доп. Москва: Издательство Юрайт, 2023. 310 с. (Профессиональное образование). ISBN 978-5-534-11626-7. Текст: электронный // Образовательная платформа Юрайт. URL: https://urait.ru/bcode/51851(дата обращения: 09.12.2024).
13. Стружкин, Н. П.  Базы данных: проектирование : учебник для вузов / Н. П. Стружкин, В. В. Годин. — Москва : Издательство Юрайт, 2024. — 477 с. — (Высшее образование). — ISBN 978-5-534-00229-4. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: <https://urait.ru/bcode/536006> (дата обращения: 12.12.2024).
14. Проектирование информационных систем : учебник и практикум для вузов / Д. В. Чистов, П. П. Мельников, А. В. Золотарюк, Н. Б. Ничепорук. — 2-е изд., перераб. и доп. — Москва : Издательство Юрайт, 2024. — 273 с. — (Высшее образование). — ISBN 978-5-534-20361-5. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: <https://urait.ru/bcode/558007> (дата обращения: 09.12.2024).
15. Пирогов, В. Ю. Информационные системы и базы данных: организация и проектирование: учебное пособие. Москва: ЛитРес, 2023. 320 с. ISBN 978-5-534-11629-8. Текст: электронный // ЛитРес. URL: https://www.litres.ru/book/vladislav-pirogov/informacionnye-sistemy-i-bazy-dannyh-organizaciya-i-proe-648725/ (дата обращения: 12.12.2024).
16. Мартишин, С. А., Симонов, В. Л., Кузнецов, М. В. Базы данных: проектирование и разработка информационных систем с использованием СУБД MySQL и языка Go: учебное пособие. Москва: Знаниум, 2023. 400 с. ISBN 978-5-534-11630-4. Текст: электронный // Знаниум. URL: https://znanium.ru/catalog/document?id=422052 (дата обращения: 12.12.2024).
17. Рочев, К. В. Проектирование информационных систем: учебное пособие. Санкт-Петербург: Лань, 2023. 220 с. ISBN 978-5-534-11632-8. Текст: электронный // Электронно-библиотечная система Лань. URL: https://reader.lanbook.com/book/208946?demoKey=a3d998f29126fcd799a1dfdbf4466a77 (дата обращения: 09.12.2024).
18. Чистов, Д. В. Основы проектирования баз данных: учебник для среднего профессионального образования. Москва: КноРус, 2023. 280 с. ISBN 978-5-534-11633-5. Текст: электронный // Издательство КноРус. URL: https://knorus.ru/catalog/programmirovanie-subd/675339-osnovy-proektirovaniya-baz-dannyh-spo-uchebnшл/ (дата обращения: 12.12.2024).

# Приложение

## Приложение 1

CREATE TABLE **Customers** (

Customer\_ID UUID PRIMARY KEY,

CustomerType VARCHAR(20),

OrganizationName VARCHAR(255),

Customer\_Name VARCHAR(255),

Address TEXT,

Phone VARCHAR(20),

Email VARCHAR(255),

INN VARCHAR(20)

);

COMMENT ON TABLE Customers

IS **'Клиенты'**

go

CREATE TABLE **Warehouses** (

Warehouse\_ID UUID PRIMARY KEY,

Warehouse\_Name VARCHAR(255),

Warehouse\_Address TEXT

);

COMMENT ON TABLE Warehouses

IS **'Склады'**

go

CREATE TABLE **Suppliers** (

Supplier\_ID UUID PRIMARY KEY,

Supplier\_Name VARCHAR(255),

Address TEXT,

Phone VARCHAR(20),

Email VARCHAR(255),

INN VARCHAR(20),

Contact\_Person VARCHAR(255)

);

COMMENT ON TABLE Suppliers

IS **'Поставщики'**

go

CREATE TYPE **Category\_Type** AS **ENUM** ('мало-габаритное', 'средне-габаритное', 'крупно-габаритное');

CREATE TABLE Products (

Product\_ID UUID PRIMARY KEY,

pName VARCHAR(255),

Description TEXT,

VendorCode VARCHAR(50),

Category **Category\_Type,**

Purchase\_Price DECIMAL(10, 2),

Selling\_Price DECIMAL(10, 2),

Unit VARCHAR(20),

Supplier\_ID UUID REFERENCES Suppliers(Supplier\_ID),

);

COMMENT ON TABLE Products

IS **'Товары'**

go

CREATE TABLE Employees (

Employee\_ID UUID PRIMARY KEY,

Employee\_Name VARCHAR(255),

Position VARCHAR(255),

Phone VARCHAR(20),

Email VARCHAR(255)

);

COMMENT ON TABLE Employees

IS 'Сотрудники'

go

CREATE TABLE Orders (

Order\_ID UUID PRIMARY KEY,

Customer\_ID UUID REFERENCES Customers(Customer\_ID),

Order\_Date DATE,

Total\_Amount DECIMAL(10, 2),

Order\_Status VARCHAR(50),

Employee\_ID UUID REFERENCES Employees(Employee\_ID),

Comment TEXT,

Product\_ID UUID REFERENCES Products(Product\_ID),

Quantity INTEGER,

Price DECIMAL(10, 2)

);

COMMENT ON TABLE Orders

IS 'Заказы'

go

## Приложение 2

**CREATE** **EXTENSION** IF NOT EXISTS *"uuid-ossp";*

**INSERT** INTO **Customers** (Customer\_ID, CustomerType, OrganizationName, Customer\_Name, Address, Phone, Email, INN) VALUES

*(uuid\_generate\_v4(), 'Юр. лицо', 'ООО "Рога и Копыта"', NULL, 'г. Москва, ул. Ленина, д.1', '+7(495)123-45-67', 'info@rogaikopyta.ru', '1234567890'),*

*(uuid\_generate\_v4(), 'Физ. лицо', NULL, 'Иванов Иван Иванович', 'г. Н.Новгород, ул. Советская, д.5 кв.10', '+7(900)123-45-67', 'ivanov@mail.ru', NULL),*

*(uuid\_generate\_v4(), 'Юр. лицо', 'АО "СтройТех"', NULL, 'г. Санкт-Петербург, пр. Невский, д. 100', '+7(812)111-22-33', 'info@stroytech.spb.ru', '9876543210'),*

*(uuid\_generate\_v4(), 'Физ. лицо', NULL, 'Петров Петр Петрович', 'г. Москва, ул. Тверская, д.15 кв.5', '+7(925)555-12-34', 'petrov@gmail.com', NULL),*

*(uuid\_generate\_v4(), 'Физ. лицо', NULL, 'Сидоров Сергей Иванович', 'г. Казань, ул. Кремлевская, д.35 кв.21', '+7(917)777-88-99', 'sidorov@yandex.ru', NULL)*

go

**INSERT** INTO **Warehouses** (Warehouse\_ID, Warehouse\_Name, Warehouse\_Address) VALUES

*(uuid\_generate\_v4(), 'Основной склад', 'г. Н.Новгород, ул. Складская, д.1'),*

*(uuid\_generate\_v4(), 'Дополнительный склад', 'г. Н.Новгород, ул. Заводская, д.5'),*

*(uuid\_generate\_v4(), 'Склад на ул. Мира', 'г. Н.Новгород, ул. Мира, д.10'),*

*(uuid\_generate\_v4(), 'Склад на ул. Победы', 'г. Н.Новгород, ул. Победы, д.25'),*

*(uuid\_generate\_v4(), 'Склад на ул. Гагарина', 'г. Н.Новгород, ул. Гагарина, д.77')*

go

**INSERT** INTO **Suppliers** (Supplier\_ID, Supplier\_Name, Address, Phone, Email, INN, Contact\_Person) VALUES

*(uuid\_generate\_v4(), 'ООО "Поставщик-1"', 'г. Москва, ул. Промышленная, д.10', '+7(495)987-65-43', 'info@postavshik1.ru', '9876543210', 'Петров П.П.'),*

*(uuid\_generate\_v4(), 'ИП Сидоров', 'г. Н.Новгород, ул. Торговая, д.2', '+7(900)987-65-43', 'sidorov@mail.ru', '123456789012', 'Сидоров С.С.'),*

*(uuid\_generate\_v4(), 'АО "ТехноПром"', 'г. Екатеринбург, ул. Заводская, д.55', '+7(343)123-45-67', 'info@tehnoprom.ekb.ru', '112233445566', 'Иванов И.И.'),*

*(uuid\_generate\_v4(), 'ООО "ИнструментГрад"', 'г. Санкт-Петербург, пр. Индустриальный, д.70', '+7(812)321-65-43', 'info@instrumentgrad.spb.ru', '998877665544', 'Смирнов А.В.'),*

*(uuid\_generate\_v4(), 'ИП Кузнецов', 'г. Нижний Новгород, ул. Деловая, д.15', '+7(950)123-45-67', 'kuznetsov@mail.ru', '554433221100', 'Кузнецов К.К.')*

go

**INSERT** INTO **Products** (Product\_ID, pName, Description, VendorCode, Category, Purchase\_Price, Selling\_Price, Unit, Supplier\_ID) VALUES

*(uuid\_generate\_v4(), 'Дрель ударная', 'Мощная дрель для сверления отверстий в бетоне', 'DRL-001', 'мало-габаритное'::Category\_Type, 1500.00, 2500.00, 'шт.', (SELECT Supplier\_ID FROM Suppliers WHERE Supplier\_Name = 'ООО "Поставщик-1"')),*

*(uuid\_generate\_v4(), 'Перфоратор', 'Профессиональный перфоратор', 'PRF-002', 'средне-габаритное'::Category\_Type, 5000.00, 7500.00, 'шт.', (SELECT Supplier\_ID FROM Suppliers WHERE Supplier\_Name = 'ИП Сидоров')),*

*(uuid\_generate\_v4(), 'Шлифмашина угловая', 'УШМ для шлифовки и резки металла', 'USM-003', 'мало-габаритное'::Category\_Type, 2000.00, 3500.00, 'шт.', (SELECT Supplier\_ID FROM Suppliers WHERE Supplier\_Name = 'АО "ТехноПром"')),*

*(uuid\_generate\_v4(), 'Сварочный инвертор', 'Инвертор для ручной дуговой сварки', 'INV-004', 'средне-габаритное'::Category\_Type, 7000.00, 10000.00, 'шт.', (SELECT Supplier\_ID FROM Suppliers WHERE Supplier\_Name = 'ООО "ИнструментГрад"')),*

*(uuid\_generate\_v4(), 'Компрессор поршневой', 'Компрессор для сжатого воздуха', 'KMP-005', 'крупно-габаритное'::Category\_Type, 10000.00, 15000.00, 'шт.', (SELECT Supplier\_ID FROM Suppliers WHERE Supplier\_Name = 'ИП Кузнецов'))*

go

**INSERT** INTO **Employees** (Employee\_ID, Employee\_Name, Position, Phone, Email) VALUES

*(uuid\_generate\_v4(), 'Петров Иван Сергеевич', 'Менеджер по продажам', '+7(900)111-22-33', 'petrov@intertech.ru'),*

*(uuid\_generate\_v4(), 'Сидоров Алексей Петрович', 'Кладовщик', '+7(900)222-33-44', 'sidorov@intertech.ru'),*

*(uuid\_generate\_v4(), 'Смирнов Владимир Андреевич', 'Менеджер по закупкам', '+7(900)333-44-55', 'smirnov@intertech.ru'),*

*(uuid\_generate\_v4(), 'Иванова Елена Владимировна', 'Бухгалтер', '+7(900)444-55-66', 'ivanova@intertech.ru'),*

*(uuid\_generate\_v4(), 'Козлов Михаил Дмитриевич', 'Водитель-экспедитор', '+7(900)555-66-77', 'kozlov@intertech.ru')*

go

**INSERT** INTO **Orders** (Order\_ID, Customer\_ID, Order\_Date, Total\_Amount, Order\_Status, Employee\_ID, Comment, Product\_ID, Quantity, Price) VALUES

*(uuid\_generate\_v4(), (SELECT Customer\_ID FROM Customers WHERE OrganizationName = 'ООО "Рога и копыта"'), '2023-10-26', 2500.00, 'Новый', (SELECT Employee\_ID FROM Employees WHERE Employee\_Name = 'Петров Иван Сергеевич'), 'Срочный заказ', (SELECT Product\_ID FROM Products WHERE pName = 'Дрель ударная'), 1, 2500.00),*

*(uuid\_generate\_v4(), (SELECT Customer\_ID FROM Customers WHERE Customer\_Name = 'Иванов Иван Иванович'), '2023-10-27', 7500.00, 'В обработке', (SELECT Employee\_ID FROM Employees WHERE Employee\_Name = 'Петров Иван Сергеевич'), NULL, (SELECT Product\_ID FROM Products WHERE pName = 'Перфоратор'), 1, 7500.00),*

*(uuid\_generate\_v4(), (SELECT Customer\_ID FROM Customers WHERE OrganizationName = 'АО "СтройТех"'), '2023-10-28', 3500.00, 'Собран', (SELECT Employee\_ID FROM Employees WHERE Employee\_Name = 'Петров Иван Сергеевич'), 'Доставка завтра', (SELECT Product\_ID FROM Products WHERE pName = 'Шлифмашина угловая'), 1, 3500.00),*

*(uuid\_generate\_v4(), (SELECT Customer\_ID FROM Customers WHERE Customer\_Name = 'Петров Петр Петрович'), '2023-10-29', 10000.00, 'Отгружен', (SELECT Employee\_ID FROM Employees WHERE Employee\_Name = 'Петров Иван Сергеевич'), NULL, (SELECT Product\_ID FROM Products WHERE pName = 'Сварочный инвертор'), 1, 10000.00),*

*(uuid\_generate\_v4(), (SELECT Customer\_ID FROM Customers WHERE Customer\_Name = 'Сидоров Сергей Иванович'), '2023-10-30', 30000.00, 'Выполнен', (SELECT Employee\_ID FROM Employees WHERE Employee\_Name = 'Петров Иван Сергеевич'), 'Оплачено', (SELECT Product\_ID FROM Products WHERE pName = 'Компрессор поршневой'), 2, 15000.00)*

go

**INSERT** INTO **Stock** (Warehouse\_ID, Product\_ID, Quantity) VALUES

*((SELECT Warehouse\_ID FROM Warehouses WHERE Warehouse\_Name = 'Основной склад'), (SELECT Product\_ID FROM Products WHERE pName = 'Дрель ударная'), 10),*

*((SELECT Warehouse\_ID FROM Warehouses WHERE Warehouse\_Name = 'Основной склад'), (SELECT Product\_ID FROM Products WHERE pName = 'Перфоратор'), 5),*

*((SELECT Warehouse\_ID FROM Warehouses WHERE Warehouse\_Name = 'Склад на ул. Мира'), (SELECT Product\_ID FROM Products WHERE pName = 'Шлифмашина угловая'), 15),*

*((SELECT Warehouse\_ID FROM Warehouses WHERE Warehouse\_Name = 'Склад на ул. Победы'), (SELECT Product\_ID FROM Products WHERE pName = 'Сварочный инвертор'), 8),*

*((SELECT Warehouse\_ID FROM Warehouses WHERE Warehouse\_Name = 'Склад на ул. Гагарина'), (SELECT Product\_ID FROM Products WHERE pName = 'Компрессор поршневой'), 20)*

go

## Приложение 3

1. **Выборка всех товаров определённое категории**

SELECT

\*

FROM

Products

WHERE

Category = 'мало-габаритное'::Category\_Type;

1. **Выборка информации о клиенте по его ID**

SELECT

\*

FROM

Customers

WHERE

Customer\_ID = '4739a14a-e1f3-49b3-8c39-5a322ff9a80a';

1. **Выборка всех заказов, оформленных конкретным сотрудником за определённый период**

SELECT

o.\*

FROM

Orders o

JOIN

Employees e ON o.Employee\_ID = e.Employee\_ID

WHERE

e.Employee\_Name = 'Петров Иван Сергеевич'

AND o.Order\_Date BETWEEN '2023-10-01' AND '2023-10-31';

1. **Выборка товаров, остаток которых на складе меньше заданного количества**

SELECT

p.pName AS ProductName,

s.Quantity AS StockQuantity

FROM

Products p

JOIN

Stock s ON p.Product\_ID = s.Product\_ID

WHERE

s.Quantity < 15;

1. **Выборка полной информации о составе заказа по его номеру**

SELECT

o.Order\_ID,

o.Order\_Date,

o.Order\_Status,

c.Customer\_Name,

c.OrganizationName,

p.pName AS ProductName,

o.Quantity,

o.Price,

e.Employee\_Name AS EmployeeName

FROM

Orders o

JOIN

Customers c ON o.Customer\_ID = c.Customer\_ID

JOIN

Products p ON o.Product\_ID = p.Product\_ID

JOIN

Employees e ON o.Employee\_ID = e.Employee\_ID

WHERE

o.Order\_ID = 'a31c9c88-29a4-4ee3-8d42-c4cd313ceaa1';

## Приложение 4

**Триггер 1**

CREATE OR REPLACE FUNCTION updateTotalAmount()

RETURNS TRIGGER AS $$

BEGIN

UPDATE Orders

SET Total\_Amount = (

SELECT SUM(Quantity \* Price)

FROM Orders

WHERE Order\_ID = NEW.Order\_ID

)

WHERE Order\_ID = NEW.Order\_ID;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

-- Триггер на добавление товара в заказ

CREATE TRIGGER trg\_insert\_order

AFTER INSERT ON Orders

FOR EACH ROW

EXECUTE PROCEDURE updateTotalAmount();

-- Триггер на изменение товара в заказе

CREATE TRIGGER trg\_update\_order

AFTER UPDATE ON Orders

FOR EACH ROW

WHEN (OLD.Quantity IS DISTINCT FROM NEW.Quantity OR OLD.Price IS DISTINCT FROM NEW.Price)

EXECUTE PROCEDURE updateTotalAmount();

**Триггер 2**

CREATE OR REPLACE FUNCTION checkStockQuantity()

RETURNS TRIGGER AS $$

DECLARE

stock\_quantity INTEGER;

BEGIN

SELECT INTO stock\_quantity

s.Quantity

FROM Stock s

WHERE s.Product\_ID = NEW.Product\_ID

AND s.Warehouse\_ID = (SELECT Warehouse\_ID FROM Warehouses WHERE Warehouse\_Name = 'Основной склад');

IF stock\_quantity IS NULL OR stock\_quantity < NEW.Quantity THEN

RAISE EXCEPTION 'Недостаточно товара "%" на складе. В наличии: %, требуется: %', (SELECT pName FROM Products WHERE Product\_ID = NEW.Product\_ID), stock\_quantity, NEW.Quantity;

END IF;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

-- Триггер на проверку остатков при добавлении товара в заказ

CREATE TRIGGER trg\_check\_stock

BEFORE INSERT ON Orders

FOR EACH ROW

EXECUTE PROCEDURE checkStockQuantity();

## Приложение 5

**Форма «Состав заказа»**

import sys  
from PyQt5.QtWidgets import (QApplication, QWidget, QTableWidget,  
 QTableWidgetItem, QVBoxLayout, QLabel,  
 QLineEdit, QPushButton, QMessageBox,  
 QFormLayout, QHeaderView)  
from PyQt5.QtCore import Qt  
import uuid  
import psycopg2  
  
  
DATABASE = {  
 "host": "localhost:5432",  
 "database": "InterTex",  
 "user": "postgres",  
 "password": "postgres",  
 "options": "-c client\_encoding=UTF8"  
}  
  
class OrderDetailsForm(QWidget):  
 def \_\_init\_\_(self):  
 super().\_\_init\_\_()  
 self.setWindowTitle('Состав заказа')  
 self.setGeometry(100, 100, 800, 400)  
  
 self.initUI()  
  
 def initUI(self):  
 self.orderIdLabel = QLabel("ID заказа:")  
 self.orderIdEdit = QLineEdit()  
 self.loadButton = QPushButton("Загрузить")  
 self.loadButton.clicked.connect(self.loadOrderDetails)  
  
 self.tableWidget = QTableWidget()  
 self.tableWidget.setColumnCount(7)   
 self.tableWidget.setHorizontalHeaderLabels(["ID", "Название товара", "Количество", "Цена", "Сумма", "ФИО клиента", "ФИО сотрудника"])  
 header = self.tableWidget.horizontalHeader()  
 header.setSectionResizeMode(0, QHeaderView.ResizeToContents)  
 header.setSectionResizeMode(1, QHeaderView.Stretch)  
 header.setSectionResizeMode(2, QHeaderView.ResizeToContents)  
 header.setSectionResizeMode(3, QHeaderView.ResizeToContents)  
 header.setSectionResizeMode(4, QHeaderView.ResizeToContents)  
 header.setSectionResizeMode(5, QHeaderView.ResizeToContents)  
 header.setSectionResizeMode(6, QHeaderView.ResizeToContents)  
  
 formLayout = QFormLayout()  
 formLayout.addRow(self.orderIdLabel, self.orderIdEdit)  
 formLayout.addRow(self.loadButton)  
  
 self.layout = QVBoxLayout()  
 self.layout.addLayout(formLayout)  
 self.layout.addWidget(self.tableWidget)  
 self.setLayout(self.layout)  
  
 def loadOrderDetails(self):  
 order\_id = self.orderIdEdit.text()  
 if not order\_id:  
 QMessageBox.warning(self, 'Внимание', 'Введите ID заказа')  
 return  
  
 try:  
 conn = psycopg2.connect(\*\*DATABASE)  
 cursor = conn.cursor()  
  
 cursor.execute("""  
 SELECT  
 o.Order\_ID,  
 p.pName AS ProductName,  
 o.Quantity,  
 o.Price,  
 o.Quantity \* o.Price AS TotalPrice,  
 c.Customer\_Name,  
 e.Employee\_Name  
 FROM  
 Orders o  
 JOIN  
 Customers c ON o.Customer\_ID = c.Customer\_ID  
 JOIN  
 Products p ON o.Product\_ID = p.Product\_ID  
 JOIN  
 Employees e ON o.Employee\_ID = e.Employee\_ID  
 WHERE  
 o.Order\_ID = %s  
 """, (order\_id,))  
  
 rows = cursor.fetchall()  
  
 if not rows:  
 QMessageBox.information(self, 'Информация', 'Заказ с таким ID не найден')  
 return  
  
 self.tableWidget.setRowCount(len(rows))  
 for i, row in enumerate(rows):  
 for j, col in enumerate(row):  
 item = QTableWidgetItem(str(col))  
 item.setFlags(item.flags() & ~Qt.ItemIsEditable) # Запрет редактирования  
 self.tableWidget.setItem(i, j, item)  
  
 cursor.close()  
 conn.close()  
  
 except Exception as e:  
 QMessageBox.critical(self, 'Ошибка', f'Не удалось загрузить данные: {e}')  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 app = QApplication(sys.argv)  
 form = OrderDetailsForm()  
 form.show()  
 sys.exit(app.exec\_())

**Форма авторизации**

import sys  
from PyQt5.QtWidgets import (QApplication, QWidget, QLabel, QLineEdit,  
 QPushButton, QVBoxLayout, QMessageBox,  
 QDialog)  
import psycopg2  
  
# Параметры подключения к базе данных (замените на свои)  
DATABASE = {  
 "host": "localhost:5432",  
 "database": "InterTex",  
 "user": "postgres",  
 "password": "postgres",  
 "options": "-c client\_encoding=UTF8"  
}  
  
class LoginForm(QDialog):  
 def \_\_init\_\_(self):  
 super().\_\_init\_\_()  
 self.setWindowTitle('Авторизация')  
 self.setGeometry(200, 200, 300, 150)  
  
 self.initUI()  
  
 def initUI(self):  
 self.loginLabel = QLabel("Логин:")  
 self.loginEdit = QLineEdit()  
 self.passwordLabel = QLabel("Пароль:")  
 self.passwordEdit = QLineEdit()  
 self.passwordEdit.setEchoMode(QLineEdit.Password) # Скрываем ввод пароля  
  
 self.loginButton = QPushButton("Войти")  
 self.loginButton.clicked.connect(self.checkLogin)  
 self.cancelButton = QPushButton("Отмена")  
 self.cancelButton.clicked.connect(self.reject)  
  
 layout = QVBoxLayout()  
 layout.addWidget(self.loginLabel)  
 layout.addWidget(self.loginEdit)  
 layout.addWidget(self.passwordLabel)  
 layout.addWidget(self.passwordEdit)  
 layout.addWidget(self.loginButton)  
 layout.addWidget(self.cancelButton)  
 self.setLayout(layout)  
  
 def checkLogin(self):  
 login = self.loginEdit.text()  
 password = self.passwordEdit.text()  
  
 if not login or not password:  
 QMessageBox.warning(self, 'Ошибка', 'Введите логин и пароль')  
 return  
  
 try:  
 conn = psycopg2.connect(\*\*DATABASE)  
 cursor = conn.cursor()  
  
 # ВНИМАНИЕ: Этот код уязвим для SQL-инъекций в реальном приложении  
 # используйте параметризованные запросы или хранимые процедуры для  
 # проверки логина и пароля  
 cursor.execute(f"SELECT \* FROM Employees WHERE Email = '{login}' AND Employee\_Name = '{password}'") # небезопасно!  
  
 user = cursor.fetchone()  
  
 cursor.close()  
 conn.close()  
  
 if user:  
 QMessageBox.information(self, 'Успех', 'Авторизация прошла успешно')  
 self.accept() # Закрываем окно с кодом, что всё хорошо  
 else:  
 QMessageBox.warning(self, 'Ошибка', 'Неверный логин или пароль')  
  
 except Exception as e:  
 QMessageBox.critical(self, 'Ошибка', f'Не удалось подключиться к базе данных: {e}')  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 app = QApplication(sys.argv)  
 login\_form = LoginForm()  
 if login\_form.exec\_() == QDialog.Accepted:  
 sys.exit(app.exec\_())  
 else:  
 sys.exit()

## Приложение 6

[**https://github.com/Polubomy/FesikDataBse**](https://github.com/Polubomy/FesikDataBse)

1. [7]Создание базы данных SQL для pyodbc // Microsoft Learn. URL: https://learn.microsoft.com/ru-ru/sql/connect/python/pyodbc/step-2-create-a-sql-database-for-pyodbc-python-development?view=sql-server-ver16 [↑](#footnote-ref-1)
2. [2]Прохоренок, Н. А. Python 3 и PyQt 5. Разработка приложений. — 2-е изд., перераб. и доп. / Н. А. Прохоренок, В. А. Дронов. — СПб.: БХВ-Петербург, 2018. — 832 с. [↑](#footnote-ref-2)
3. [17]Рочев, К. В. Проектирование информационных систем: учебное пособие. Санкт-Петербург: Лань, 2023. 220 с. ISBN 978-5-534-11632-8. Текст: электронный // Электронно-библиотечная система Лань. URL: https://reader.lanbook.com/book/208946?demoKey=a3d998f29126fcd799a1dfdbf4466a77 [↑](#footnote-ref-3)
4. CREATE EXTENSION IF NOT EXISTS "uuid-ossp" – установка расширения для генерации UUID [↑](#footnote-ref-4)
5. [10] Гвоздева, Т. В., Баллод, Б. А. Проектирование информационных систем. Стандартизация: учебное пособие для вузов. 2-е изд., стер. Санкт-Петербург: Лань, 2023. 252 с. ISBN 978-5-8114-7963-4. Текст: электронный // Лань: электронно-библиотечная система. URL: https://e.lanbook.com/book/169810 [↑](#footnote-ref-5)
6. [7] Создаём и наполняем базу данных SQLite в Python // Журнал «Код». URL: https://thecode.media/sqlite-py/ [↑](#footnote-ref-6)
7. [2]Прохоренок, Н. А. Python 3 и PyQt 5. Разработка приложений. — 2-е изд., перераб. и доп. / Н. А. Прохоренок, В. А. Дронов. — СПб.: БХВ-Петербург, 2018. — 832 с. [↑](#footnote-ref-7)