МИНИСТЕРСТВО ОБРАЗОВАНИЯ МОСКОВСКОЙ ОБЛАСТИ

Государственное бюджетное профессиональное образовательное учреждение Московской области «Люберецкий техникум имени Героя Советского Союза, летчика-космонавта Ю.А.Гагарина»

КУРСОВАЯ РАБОТА

Малиновской Софии Витальевны

МДК 11.01 Технология разработки и защиты баз данных

Курс 2 Группа № ИС-21

Тема: Разработать БД организации автоматизации учёта товаров организации оптовой или розничной торговли

Выполнил/а/ студент _____ Малиновская София Витальевна
(подпись) (ФИО полностью)

Руководитель _____ Тарджиманян Лия Николаевна
(подпись) (ФИО полностью)

Оценка _____

Дзержинский 2023

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Цель курсового проектирования:

Спроектировать базу данных (БД) организации автоматизации учёта товаров организации оптовой или розничной торговли.

Для достижения данной цели необходимо выполнить следующие задачи:

- создать ER-диаграмму по заданию (в Draw SQL)
- создать базу данных по ER-диаграмме (в SQLite)
- заполнить базу данными
- создать авторизацию (в Qt5 дизайнере)
- создать формы для автоматизированного заполнения данными (в Qt5 дизайнере)
- подключить форму к базе данных (в Visual Studio Code)

В данной курсовой работе будут рассмотрены все этапы создания базы данных: от анализа предметной области до автоматизированной базы данных.

Для разработки баз данных была выбрана СУБД Sqlite. Причины выбора данной СУБД:

- Надежность
- Практичность;
- Бесплатность;

Окончательным результатом курсовой работы будет готовая база данных, которая позволит вести учет наличие товара в на складе, контролировать взаимоотношения с поставщиками и клиентами, а также сотрудниками.

ГЛАВА 1 ОСНОВЫ БАЗ ДАННЫХ.

1.1. Архитектура СУБД

База данных- это представленная в объективной форме совокупность самостоятельных материалов,систематизированных таким образом, чтобы эти материалы могли быть найдены и обработаны с помощью ЭВМ (электронной вычислительной машиной).

СУБД - это совокупность программных и лингвистических средств общего или специального назначения, обеспечивающих управление созданием и использованием БД.

Основные функции СУБД:

- управление данными во внешней памяти (на дисках)
- управление данными в оперативной памяти с использованием дискового кэша
- журнализация изменений, резервное копирование и восстановление БД после сбоев
- поддержка языков БД (язык определения данных, язык манипулирования данных)

SQL:

- Structured query language - «язык структурированных запросов»
- Создание, изменение и управление данными

3 нормальные формы БД:

- **Первая нормальная форма** - отношение находится в 1НФ, если все его атрибуты являются простыми, все используемые домены должны

содержать только скалярные значения. Не должно быть повторений строк в таблице.

- **Вторая нормальная форма** - отношение находится во 2НФ, если оно находится в 1НФ и каждый неключевой атрибут неприводимо зависит от Первичного Ключа(ПК).
- **Третья нормальная форма** - Отношение находится в 3НФ, когда находится во 2НФ и каждый неключевой атрибут нетранзитивно зависит от первичного ключа.

1.2. Принципы разработки распределенных баз данных

Распределенная БД (РаБД) - набор логически связанных между собой разделяемых данных и их описаний, которые физически распределены по нескольким компьютерам (узлам) в некоторой компьютерной сети.

Основные принципы создания и функционирования распределенных БД:

- прозрачность размещения данных для пользователя (пользователю распределенная БД должна представляться точно так же, как и нераспределенная);
- изолированность пользователей друг от друга (на работу одного пользователя с БД не должна влиять работа других пользователей с ней);
- синхронизация БД и непротиворечивость состояния данных в любой момент времени.

Выделились несколько самостоятельных технологий распределенной обработки данных:

- клиент-сервер;
- реплицирования;

- объектного связывания.

1.3. Data Mining

Data Mining (добыча данных, интеллектуальный анализ данных, глубинный анализ данных или просто майнинг данных) — это процесс, используемый компаниями для превращения необработанных больших данных в полезную информацию. Также для этой технологии используется менее популярный термин «обнаружение знаний в данных» или KDD (knowledge discovery in databases).

Задачи Data Mining

- прогнозирование: оценка продаж, предсказание нагрузки сервера или его времени простоя;
- риск и вероятность: выбор подходящих заказчиков для целевой рассылки, определение точки баланса для рисковых сценариев, назначение вероятностей по диагнозам или другим результатам;
- рекомендации: определение продуктов, которые будут продаваться вместе, создание рекомендательных сообщений;
- поиск последовательностей: анализ выбора заказчиков во время совершения покупок, прогнозирование их поведения;
- группирование: разделение заказчиков или событий на кластеры, анализ и прогнозирование общих черт этих кластеров.

Где применяют Data Mining:

1. Торговля
2. Банки и телеком
3. Страхование

4. Производство
5. Социология
6. Медицина

Технология и методы Data Mining

- Постановка задачи. Этот шаг включает анализ бизнес-требований, определение области проблемы, метрик, по которым будет выполняться оценка модели, а также определение задач для проекта анализа.
- Подготовка данных: объединение и очистка. Эта работа включает не только удаление ненужных данных, но и поиск в них скрытых зависимостей, определение источников самых точных данных и создание таблицы для анализа.
- Изучение данных.
- Построение моделей.
- Исследование и проверка моделей. Точность их прогнозов можно проверить при помощи специальных средств.
- Развертывание и обновление моделей. Когда модель заработала, ее нужно обновлять по мере поступления новых данных, а затем выполнять их повторную обработку.

1.4. Модели данных.ООП

Модель данных - определяет абстракцию данных для приложений.

1.Реляционная модель данных

Реляционная модель представляет собой совокупность данных, состоящую из набора двумерных таблиц. В теории множеств таблице соответствует термин отношение (relation), физическим представлением которого является таблица,

отсюда и название модели – реляционная. Реляционная модель является удобной и наиболее привычной формой представления данных.

При табличной организации данных отсутствует иерархия элементов. Строки и столбцы могут быть просмотрены в любом порядке, поэтому высока гибкость выбора любого подмножества элементов в строках и столбцах.

Любая таблица в реляционной базе состоит из строк, которые называют записями, и столбцов, которые называют полями. На пересечении строк и столбцов находятся конкретные значения данных. Для каждого поля определяется множество его значений, например, поле «Месяц» может иметь двенадцать значений.

Структура таблицы в реляционной базе характеризуется следующим:

- совокупности столбцов;
- каждый столбец имеет уникальное, то есть не повторяющееся в других столбцах имя;
- последовательность столбцов в таблице не существенна;
- все строки таблицы организованы по одинаковой структуре, то есть имеют одно и то же количество реквизитов и имеют одинаковую длину;
- в таблице нет одинаковых строк;
- количество строк в таблице практически не ограничено;
- последовательность строк в таблице не существенна;
- при выполнении манипуляций с таблицей все строки и столбцы могут просматриваться в произвольном порядке безотносительно к их содержанию и смыслу.

2.Объектно-ориентированная модель

В объектно-ориентированной модели при представлении данных имеется возможность идентифицировать отдельные записи базы. Между записями базы данных и функциями их обработки устанавливаются взаимосвязи с помощью

механизмов, подобных соответствующим средствам в объектно-ориентированных языках программирования.

Структура объектно-ориентированной БД графически представима в виде дерева, узлами которого являются объекты. Свойства объектов описываются некоторым стандартным типом (например, строковым — string) или типом, конструируемым пользователем (определяется как class).

Логическая структура объектно-ориентированной БД внешне похожа на структуру иерархической БД. Основное отличие между ними состоит в методах манипулирования данными.

Основные понятия ООП применительно к объектно-ориентированной модели БД:

- **Инкапсуляция** - ограничивает область видимости имени свойства пределами того объекта, в котором оно определено.
- **Наследование** - распространяет область видимости свойства на всех потомков объекта.
- **Полиморфизм** в объектно-ориентированных языках программирования означает способность одного и того же программного кода работать с разнотипными данными.

ГЛАВА 2 БД ОРГАНИЗАЦИИ АВТОМАТИЗАЦИИ УЧЕТА ТОВАРОВ ОРГАНИЗАЦИИ ОПТОВОЙ ИЛИ РОЗНИЧНОЙ ТОРГОВЛИ.

2.1. Техническая часть.

Общее описание проекта: в организации необходима автоматизированная система учета товара на складе. В проекте необходимо разработать базу данных для хранения информации о товаре, сотрудниках, продажах, заказах и тд.

Требования к БД:

- БД должна быть реляционной и хранить информацию о товаре, сотрудниках, продажах, заказах и тд.
- Должна быть возможность добавления, удаления и изменения данных в таблицах.
- В базе должен быть поиск данных по таблицам.

Описание интерфейса:

Интерфейс базы данных должен включать в себя возможность удаления, добавления и изменения данных в таблицах. Также должна быть возможность выводить данные по заданным критериям.

2.2. Построение ER-диаграммы предметной области

Для проектирования базы данных, воспользуюсь построением ER-диаграммы, в которой можно увидеть сущности, атрибуты и связи между ними.

На основе задания в Er-диаграмме должны присутствовать такие сущности как: товары, сотрудники, поставщики, заказы, продажи, категории товаров, клиенты, также к этим сущностям должны быть атрибуты, которые можно увидеть на схеме ниже (рис. 2.1)

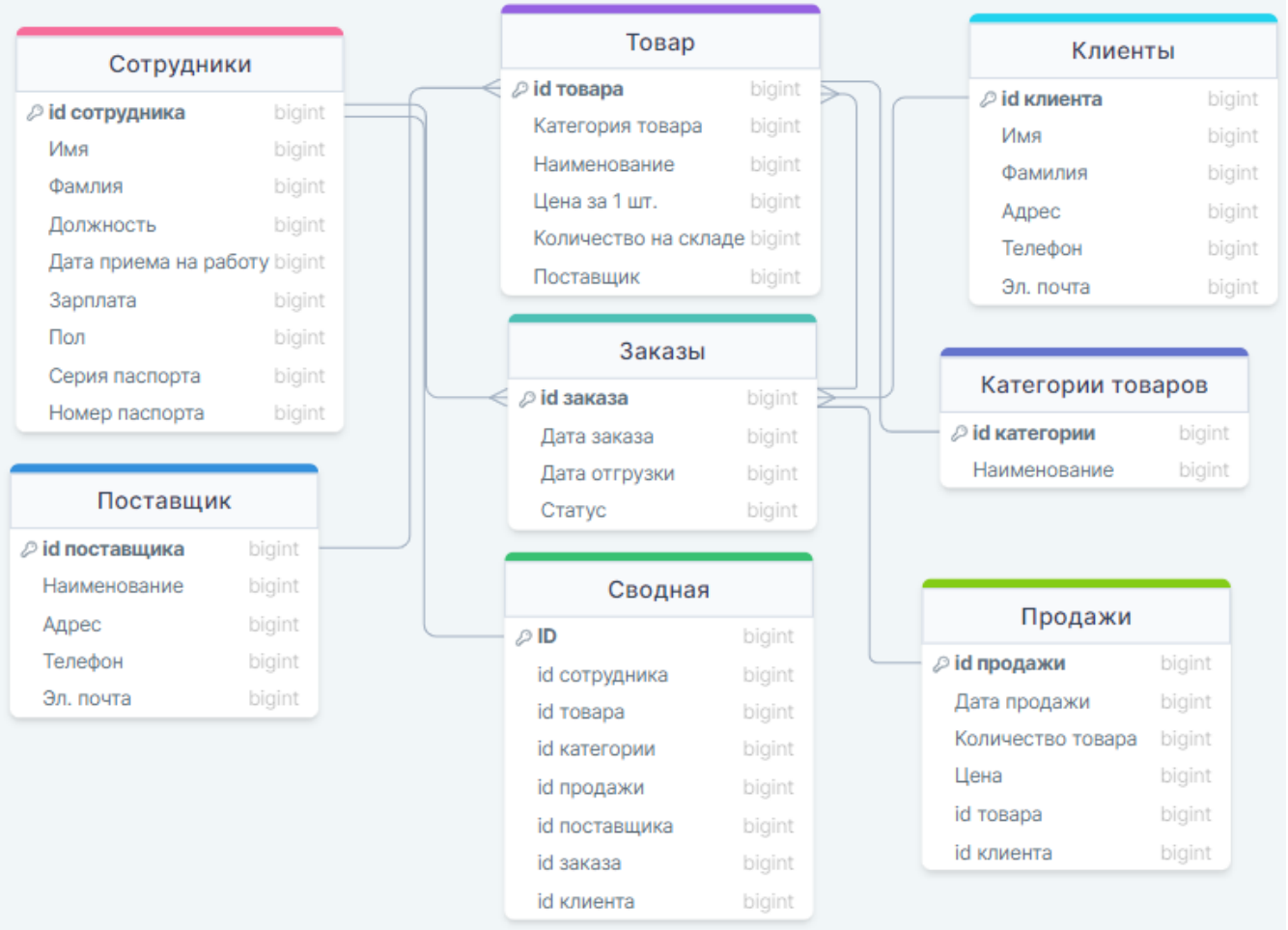


Рисунок 2.1 ER-диаграмма

2.3. Проектирование БД.

На основе ER-диаграммы создаем базу данных в SqLite. У нас есть 8 сущностей: сотрудники, товар, клиенты, поставщик, заказы, категории товаров, продажи, сводная.

Таблица “Сотрудники” имеет атрибуты:

ID сотрудника

Имя

Фамилия

Должность

Дата приема на работу

Зарплата

Пол

Серия паспорта

Номер паспорта

Таблица “Товар” имеет атрибуты:

ID товара

Категории товара

Наименование

Цена за 1 шт.

Количество на складе

Поставщик

Таблица “Клиенты” имеет атрибуты:

ID клиента

Имя

Фамилия

Адрес

Телефон

Почта

Таблица “Заказы” имеет атрибуты:

ID заказа

Дата заказа

Дата отгрузки

Статус

Таблица “Категории товаров” имеет атрибуты:

ID категории

Наименование

Таблица “Поставщик” имеет атрибуты:

ID поставщика

Наименование

Адрес

Телефон

Почта

Таблица “Продажи” имеет атрибуты:

ID продажи

Дата продажи

Количество товара

Цена

ID товара

ID клиента

Таблица “Сводная” имеет атрибуты:

ID

ID сотрудника

ID товара

ID клиента

ID заказа

ID категории

ID поставщика

ID продажи

Для контроля целостности были использованы встроенные функции СУБД SqLite, такие как: NOT NULL, PRIMARY KEY, AUTOINCREMENT.

Также для каждого атрибуты были выбраны подходящие для них типы данных, такие как : INTEGER и TEXT.

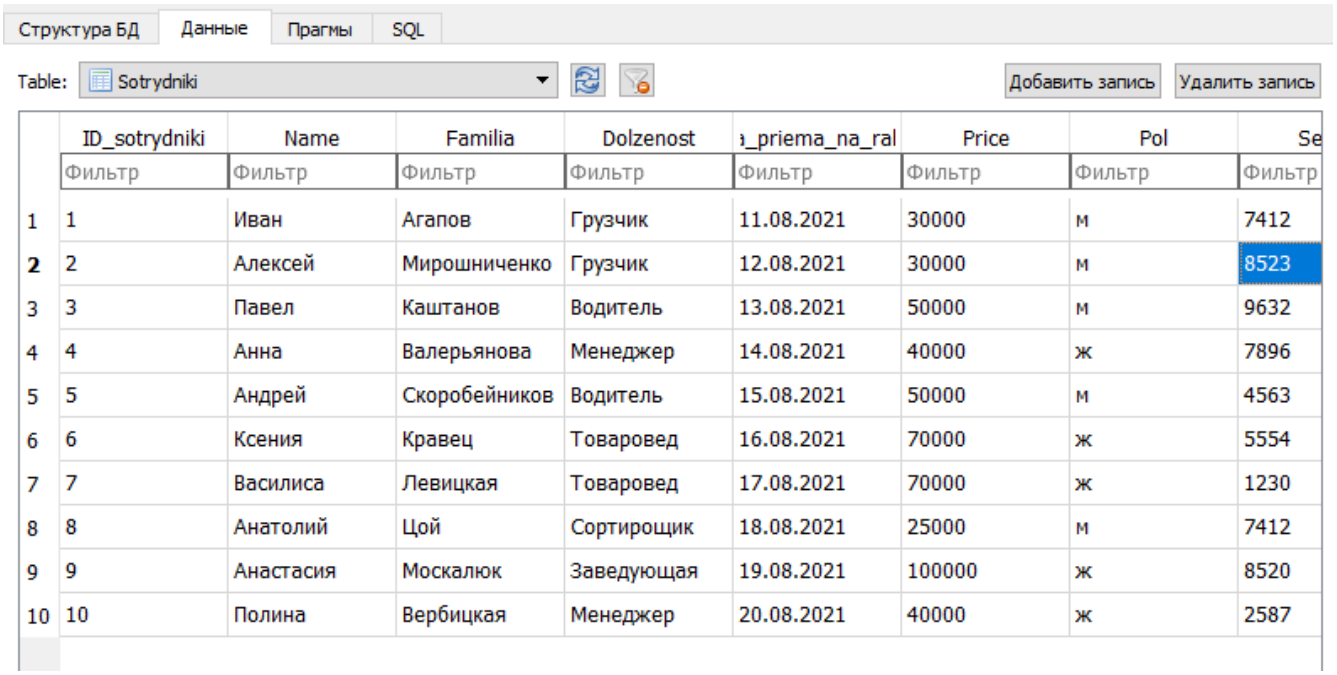
2.4. Организация ввода данных.

Все таблицы и их атрибуты сперва были введены в Google Таблицы см. (рис 2.2)

	A	B	C	D	E	F	G	H	I
1	ID_sotrydniki	Name	Familia	Dolzenost	Data_priema_na	Price	Pol	Seria	Nomer
2	1	Иван	Агапов	Грузчик	11.08.2021	30000	м	7412	456789
3	2	Алексей	Мирошниченкс	Грузчик	12.08.2021	30000	м	8523	78912
4	3	Павел	Каштанов	Водитель	13.08.2021	50000	м	9632	456789
5	4	Анна	Валерьянова	Менеджер	14.08.2021	40000	ж	7896	123456
6	5	Андрей	Скоробейников	Водитель	15.08.2021	50000	м	4563	456456
7	6	Ксения	Кравец	Товаровед	16.08.2021	70000	ж	5554	789789
8	7	Василиса	Левицкая	Товаровед	17.08.2021	70000	ж	1230	963963
9	8	Анатолий	Цой	Сортирощик	18.08.2021	25000	м	7412	147789
10	9	Анастасия	Москалюк	Заведующая	19.08.2021	100000	ж	8520	456789
11	10	Полина	Вербицкая	Менеджер	20.08.2021	40000	ж	2587	456123
12									

Рисунок 2.2 Ввод данных в Google Таблицы

Таблицы мы сохраняем в формате CSV, чтобы импортировать их в Sqlite см. (Рис 2.3)



	ID_sotrydniki	Name	Familia	Dolzenost	_priema_na_ral	Price	Pol	Se
	Фильтр	Фильтр	Фильтр	Фильтр	Фильтр	Фильтр	Фильтр	Фильтр
1	1	Иван	Агапов	Грузчик	11.08.2021	30000	м	7412
2	2	Алексей	Мирошниченко	Грузчик	12.08.2021	30000	м	8523
3	3	Павел	Каштанов	Водитель	13.08.2021	50000	м	9632
4	4	Анна	Валерьянова	Менеджер	14.08.2021	40000	ж	7896
5	5	Андрей	Скоробейников	Водитель	15.08.2021	50000	м	4563
6	6	Ксения	Кравец	Товаровед	16.08.2021	70000	ж	5554
7	7	Василиса	Левицкая	Товаровед	17.08.2021	70000	ж	1230
8	8	Анатолий	Цой	Сортирощик	18.08.2021	25000	м	7412
9	9	Анастасия	Москалюк	Заведующая	19.08.2021	100000	ж	8520
10	10	Полина	Вербицкая	Менеджер	20.08.2021	40000	ж	2587

Рисунок 2.3 Импорт таблиц в Sqlite

2.5 Разработка интерфейса

Для разработки интерфейса выбран Qt5 Designer. Необходимо создать форму для ввода данных и различных функций см. (рис 2.4)

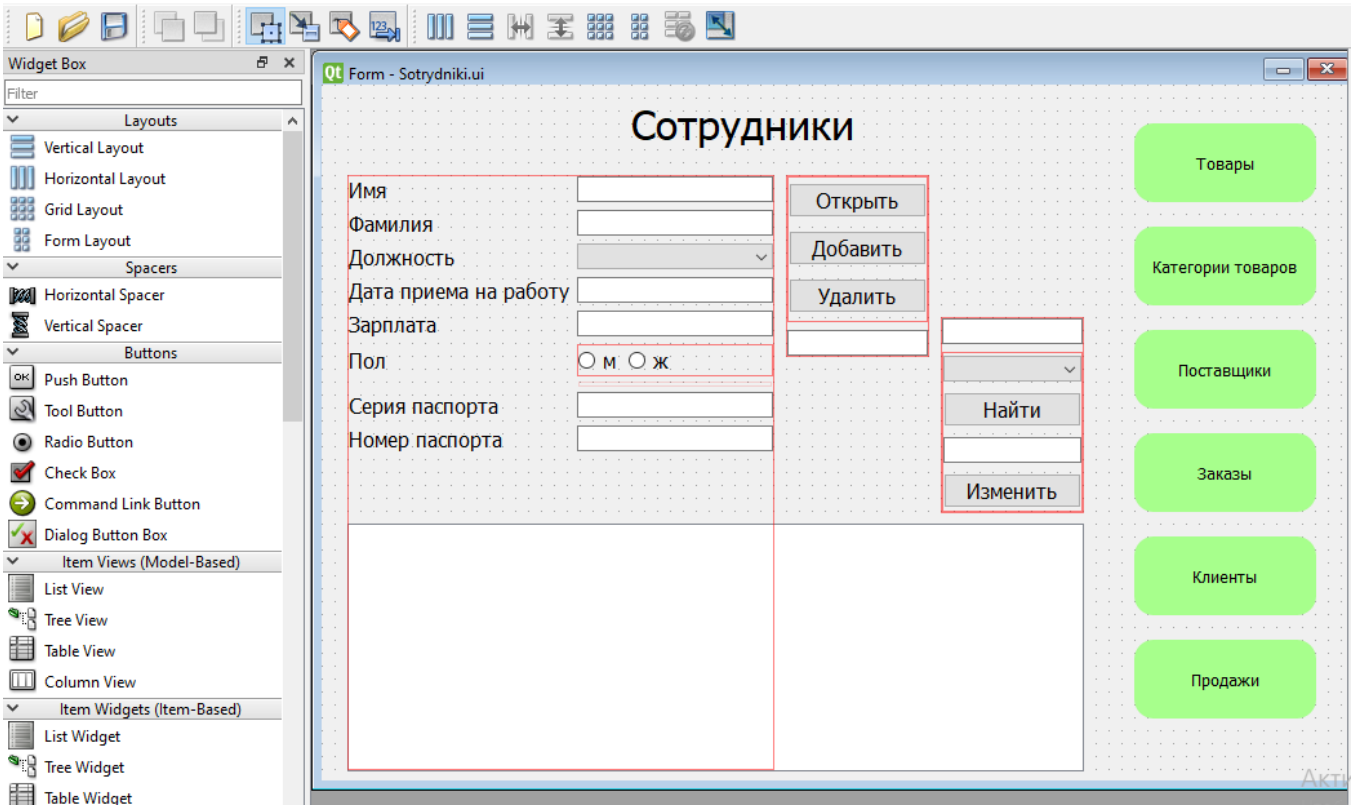


Рисунок 2.4 Создание формы в Qt5 Designer

После создания всех форм, их необходимо связать с базой данных, а также присвоить кнопкам соответствующие функции.

2.6 Разбор кода

Для начала мы добавим все формы и базу данных в VSCode и установим библиотеку PyQt5. Затем формы необходимо перевести в другой формат см. (рис 2.5)

```
PS C:\Users\Светлана Малиновская\Desktop\Kursovaya_store\kursovaya>
PS C:\Users\Светлана Малиновская\Desktop\Kursovaya_store\kursovaya> pyuic5 Sotrydniki.ui -o Sotrydniki.py
```

Рисунок 2.5 Перевод UI формы в PY

Потом импортируем набор библиотек и классы форм которые мы перевели. Импортируем в файл main.py см. (рис 2.6)

main.py > ...

```
1   import sqlite3
2   import sys
3   from PyQt5.QtGui import QPixmap
4   from PyQt5.QtWidgets import QApplication, QWidget, QTableWidgetItem, QFileDialog
5   from PyQt5 import QtWidgets
6   import login
7   import vxod
8   from Sotrydniki import Ui_Sotrydniki
9   from Tovar import Ui_Tovar
10  from Category_tovarov import Ui_Category_tovarov
11  from Postavshik import Ui_Postavshik
12  from Zakaz import Ui_Zakaz
13  from Clienti import Ui_Clienti
14  from Prodazhi import Ui_Prodazhi
```

Рисунок 2.6 Импорт библиотек и классов

Создадим еще одну БД для сохранения пользователей и паролей. Затем мы создаем класс, где будем присваивать кнопкам нужные функции см. (Рис 2.7)

```
db = sqlite3.connect('database.db')
cursor = db.cursor()

cursor.execute('''CREATE TABLE IF NOT EXISTS users(login TEXT, password TEXT)''')
db.commit()

class Registration(QtWidgets.QMainWindow, login.Ui_login): #регистрация
    def __init__(self):
        super(Registration, self).__init__()
        self.setupUi(self)
        self.lineEdit.setPlaceholderText('Введите логин')
        self.lineEdit_2.setPlaceholderText('Введите пароль')
        self.pushButton.pressed.connect(self.reg) #  регитрация
        self.pushButton_2.pressed.connect(self.login) #переход на вход
```

Рисунок 2.7 Создание класса

Создадим функцию которая регистрирует пользователя см. (рис 2.8)

```
def reg(self): #регистрация
    user_login = self.lineEdit.text()
    user_password = self.lineEdit_2.text()

    if len(user_login) == 0:
        return
    if len(user_password) == 0:
        return
    cursor.execute(f'SELECT login FROM users WHERE login = "{user_login}" ')
    if cursor.fetchone() is None:
        cursor.execute(f'INSERT INTO users VALUES("{user_login}","{user_password}")')
        self.label_2.setText(f'Аккаунт {user_login} успешно зарегистрирован')
        db.commit()
    else:
        self.label_2.setText('Такая запись уже имеется')
```

Рисунок 2.8 Создание функции для регистрации

Создадим функцию чтобы при нажатии на кнопку вход открывалась форма вход см. (Рис 2.9)

```
def login(self): #показ класса логин (вход)
    self.login = Login()
    self.login.show()
    self.hide()
```

Рисунок 2.9 Функция открытия формы “Вход”

Далее создаем функцию для формы “Вход” так, чтобы при успешной авторизации открывалась форма см. (Рис 2.10)

```
def login(self):
    try:
        user_login = self.lineEdit.text()
        user_password = self.lineEdit_2.text()

        if len(user_login) == 0:
            return
        if len(user_password) == 0:
            return

        cursor.execute(f'SELECT password FROM users WHERE login = "{user_login}"')
        check_pass = cursor.fetchall()

        cursor.execute(f'SELECT login FROM users WHERE login = "{user_login}"')
        check_login = cursor.fetchall()
        print (check_login)
        print(check_pass)

        if check_pass[0][0] == user_password and check_login[0][0] == user_login:
            self.label_2.setText(f'Успешная авториазация')
            self.Sotrydniki = Form_Sotrydniki()
            self.Sotrydniki.show()
            self.hide()
        else:
            self.label_2.setText(f'Ошибка авторизации')
    except Exception as e:
        self.label_2.setText(f'Ошибка авторизации')
```

Рисунок 2.10 Функция для формы “Вход”

Создадим функцию для кнопки открытия базы данных в форме см. (Рис 2.11)

```
def open_Sotrydniki(self): #кнопка открыть сотрудники
    try:
        self.conn = sqlite3.connect('Store.db')
        cur = self.conn.cursor()
        data = cur.execute("select * from Sotrydniki;")
        col_name = [i[0] for i in data.description]
        data_rows = data.fetchall()

    except Exception as e:
        print ("Ошибки с подключением к БД")
        return e
    self.twSotrydniki.setColumnCount(len(col_name))
    self.twSotrydniki.setHorizontalHeaderLabels(col_name)
    self.twSotrydniki.setRowCount(0)
    for i, row in enumerate(data_rows):
        self.twSotrydniki.setRowCount(self.twSotrydniki.rowCount()+1)
        for j, elen in enumerate(row):
            self.twSotrydniki.setItem(i, j,QTableWidgetItem(str(elen)))
    self.twSotrydniki.resizeColumnsToContents()
```

Рисунок 2.11 Функция для открытия базы в форме

Затем создадим функцию для кнопки добавления, с помощью которой мы сможем добавлять данные в базу см. (Рис 2.12)

```
def insert_Sotrydniki(self): #кнопка добавить
    row = [self.lineName.text(), self.lineFamilia.text(), self.cbPost.itemText(self.cbPost.currentIndex()),
           self.lineDate.text(), self.linePrice.text(), 'м' if self.rbMan.isChecked() else 'ж',
           self.lineSerias.text(), self.lineNamber.text()]
    try:
        cur = self.conn.cursor()
        cur.execute(f"""insert into Sotrydniki(Name, Familia, Dolzenost,
        data_priema_na_raboty, Price, Pol, Seria, Nomer)
        values('{row[0]}','{row[1]}','{row[2]}','{row[3]}','{row[4]}','{row[5]}','{row[6]}','{row[7]}')""" )
        self.conn.commit()
        cur.close()
    except Exception as r:
        print("Не смогли добавить запись")
        return r
    self.update()#обращаемся к update чтобы сразу увидеть изменения в БД
```

Рисунок 2.12 Функция на добавление записей

Создадим функцию обновления, чтобы после применения каких-либо функций к базе, мы могли сразу увидеть изменения см. (Рис 2.13)

```
def update(self, query="select * from Sotrydniki"): #после добавление сразу видно запись
    try:
        cur = self.conn.cursor()
        data = cur.execute(query).fetchall()
    except Exception as d:
        print(f"Проблемы с подкл {d}")
        return d
    self.twSotrydniki.setRowCount(0) #обнулмяем все данные из таблцы
    #заносим по новой
    for i, row in enumerate(data):
        self.twSotrydniki.setRowCount(self.twSotrydniki.rowCount() + 1)
        for j, elen in enumerate(row):
            self.twSotrydniki.setItem(i, j, QTableWidgetItem(str(elen)))
    self.twSotrydniki.resizeColumnsToContents()
```

Рисунок 2.13 Функция на обновления данных

Создадим функцию для удаления записей по главному ID см. (Рис 2.14)

```
def delete_Sotrydniki(self):
    id = self.lineID.text()
    conn = sqlite3.connect('Store.db')
    c = conn.cursor()
    c.execute("DELETE FROM Sotrydniki WHERE ID_sotrydniki=?", (id,))
    conn.commit()
    conn.close()
    self.update()
```

Рисунок 2.14 Функция удаления записей

Создадим функцию поиска по выбранным столбцам выбранных в ComboBox см. (Рис 2.15)

```
def search_Sotrydniki(self):
    column_name = self.cbFind_4.currentText()
    column_index = self.twSotrydniki.horizontalHeaderItem(self.twSotrydniki.currentColumn())
    search_text = self.lineFind_4.text()
    query = f"select * from Sotrydniki where {column_name} like '%{search_text}%'"
    self.update(query)
```

Рисунок 2.15 Функция поиска по столбцам

Создадим функцию для изменения ранее добавленных записей по главному ID см. (Рис 2.16)

```python
def update_record(self):#изменение
    # Открываем соединение с базой данных
    conn = sqlite3.connect('Store.db')
    cursor = conn.cursor()

    # Получаем данные из полей ввода и метки с фото
    Name = self.lineName.text()
    Familia = self.lineFamilia.text()
    Dolzenost = self.cbPost.itemText(self.cbPost.currentIndex())
    Data_priema_na_raboty = self.lineDate.text()
    Price = self.linePrice.text()
    Pol = 'м' if self.rbMan.isChecked() else 'ж', self.lineSerias.text()
    Seria = self.lineSerias.text()
    Nomer = self.lineNamber.text()

    # Получаем ID_emploee из поля ввода
    ID_sotrydniki = self.lineChange_4.text()

    # Обновляем запись в таблице Emploee
    try:
        cursor.execute(
            """UPDATE Sotrydniki SET Name=?, Familia=?, Dolzenost=?, Data_priema_na_raboty=?,
               Price=?, Pol=?, Seria=?, Nomer=? WHERE ID_sotrydniki=?""",
            (Name, Familia, Dolzenost, Data_priema_na_raboty, Price, Pol, Seria, Nomer, ID_sotrydniki))
        conn.commit()
    except Exception as e:
        print("Ошибка при обновлении записи в таблице Emploee:", e)
    finally:
        cursor.close()
        conn.close()

    self.update()# Обновляем данные в таблице на форме
```


Рисунок 2.16 Функция изменения записей

Создадим функции для открытия других выбранных форм см. (Рис 2.17)

```
def show_Tovar(self):
    self.SH_can = Form_Tovar()
    self.SH_can.show()

def show_Category_tovarov(self):
    self.SH_can = Form_Category()
    self.SH_can.show()

def show_Postavshik(self):
    self.SH_can = Form_Postavshik()
    self.SH_can.show()

def show_Zakaz(self):
    self.SH_can = Form_Zakaz()
    self.SH_can.show()

def show_Clienti(self):
    self.SH_can = Form_Clienti()
    self.SH_can.show()

def show_Prodazhi(self):
    self.SH_can = Form_Prodazhi()
    self.SH_can.show()
```

Рисунок 2.17 Функция для открытия других форм

Все формы работают по такому же принципу, который показан выше.

ЗАКЛЮЧЕНИЕ

Целью курсовой работы являлась создание базы данных организации автоматизации учёта товаров организации оптовой или розничной торговли.

Для достижения поставленной цель были выполнены следующие задачи:

- Была создана ER-диаграмма (в Draw SQL);

- Была создана база данных по ER-диаграмме (в SqLite);

- База данных была заполнена тестовыми данными;

- Были разработаны формы для автоматизированного заполнения данными БД (в

Qt5 Designer);

- Были разработаны формы для входа и регистрации (в Qt5 Designer);

- Подключены формы к базе данных (в VSCode);

СПИСОК ИСТОЧНИКОВ

1) “Курс по Data Mining” - Чубукова И. А. https://portal.tpu.ru/departments/kafedra/vt/Disciplines_VT/Data_storehouses/FilesTab/Tab/lections%20data%20mining.pdf
2) “ООП в картинках” - Хабр https://habr.com/ru/articles/463125/
3) “Архитектура СУБД” https://books.ifmo.ru/file/pdf/677.pdf
4) “Основные принципы их создания и функционирования” http://inf.vspu.ac.ru/umm_chul/files/pido/lection2.pdf