Отчет: Создание базы данных
и интерфейса “Glazki save”

Государственное бюджетное профессиональное учреждение

Московской области Люберецкий техникум имени героя Советского Союза, летчика космонавта Юрия Алексеевича Гагарина

Отчет: Создание базы данных
и интерфейса “Glazki save”

Авторы:
Студенты специальности 09.02.07
“Информационные системы и программирование”,
2-ого курса, группы ИС-21
Барсегян Екатерина Геннадьевна,
Малиновская София Витальевна
Рецензент:
Тарджиманян Лия Николаевна
Оценка:_____

Дзержинский 2023 г

Содержание:

Введение

Цели и задачи

Цель отчета проектирования: закрепление теоретических знаний, а также навыков проектирования БД, полученных при изучении дисциплины «Базы данных».

Задачи:

- Разработать ER-диаграмму по предметной области
- Создать БД по ER-диаграмме
- Наполнить БД данными
- Создать формы для работы с БД
- Связать формы с БД с помощью программирования

Инструменты:

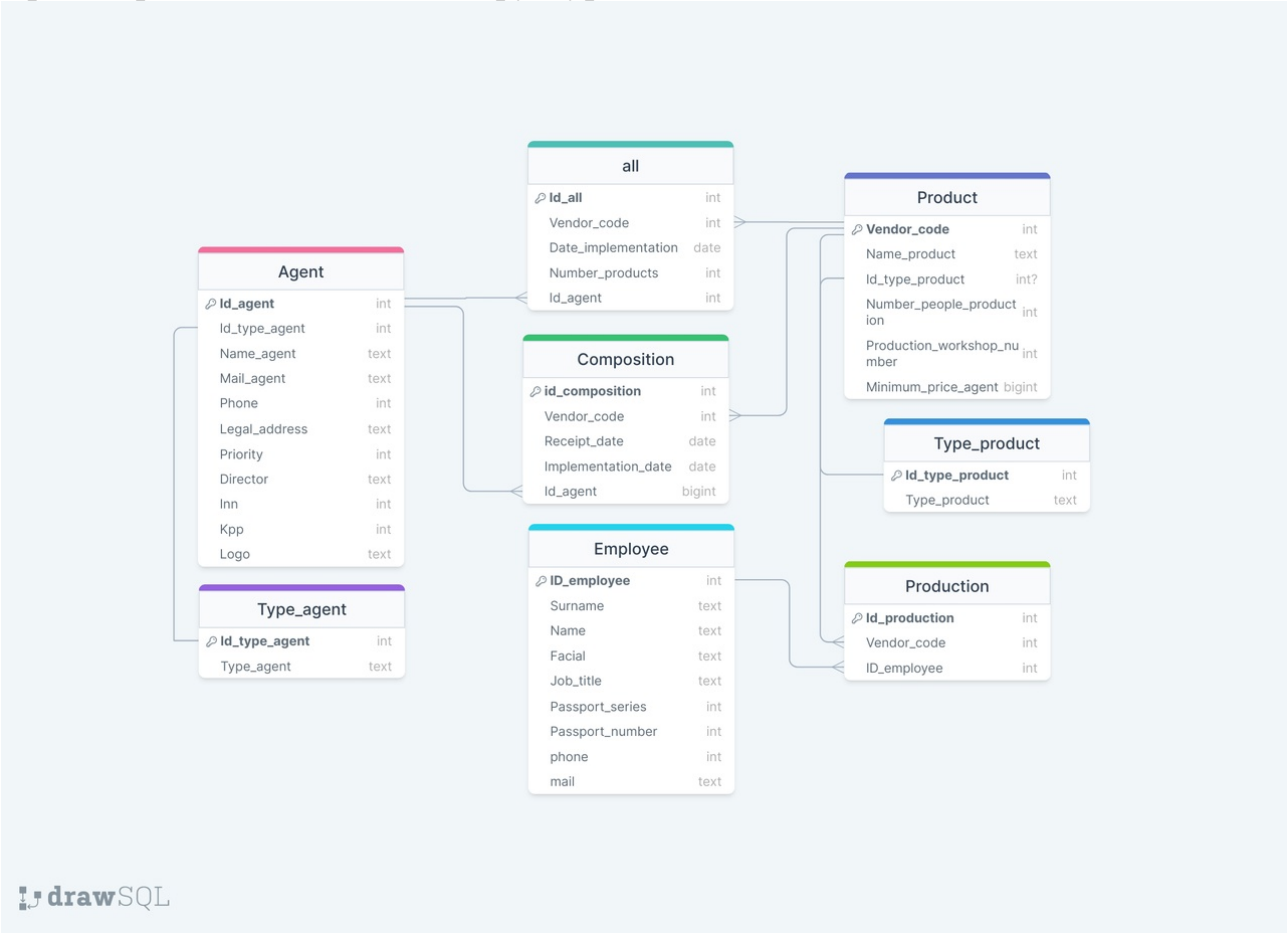
- SqlDraw(разработка ER-диаграммы)
- Sqlite (создание БД)
- Qt5 дизайнер (создание форм)
- PyCharm (связка форм и БД)

Выбор СУБД.

- Sqlite

Логическая структура

Проектирование логической структуры базы данных.



Физическая структура

Проектирование физической структуры базы данных.

На базе er-диаграммы создаём базу данных в Sqlite. У нас есть восемь сущностей: Agent, Composition, Employee, Product, Production, Type_product, prodano, type_agent

Таблица “Agent” имеет следующие атрибуты:

Id_agent
id_type_agent
Naimenovanie
Mail_agent
phone_agent
Adress
priority
director
inn
kpp

Таблица “Composition” имеет следующие атрибуты:

id_composition
vendor_code
receipt_date
implementatin_date
id_agent

Таблица “Employee” имеет следующие атрибуты:

id_emploee
surname
name
middle_name
job_title
passport_series
passpoet_number
phone
mail

Таблица “Product” имеет следующие атрибуты:

id
naimenovanie
id_type_product
vendor_code
Number_people_product_ion
Production_workschop_number
Min_price_agent

Таблица “Production” имеет следующие атрибуты:

id_production

Vendor_code
ID_emploee

Таблица “Type_product” имеет следующие атрибуты:
Id_type_product
Type_product

Таблица “prodano” имеет следующие атрибуты:
id
vendor_code
id_agent
date_implementation
number_products

Таблица “type_agent” имеет следующие атрибуты:
id_type_agent
type_agent

База данных “database.db”, была создана специально для регистрации и логирования пользователя, таблица “users” имеет следующие атрибуты:
login
password

Работа с данными
Организация ввода данных в БД.

Первичный ввод записей в базу данных будет производится при помощи импортирования данных из Excel. Далее ввод данных будет производится непосредственно с формы.

Инструкция пользования

Описание информационных потребностей пользователей и выбор способов их реализации.

У пользователя могут возникнуть следующие потребности пи пользовании интерфейсом:

- Регистрация на форме
- Авторизация на форме
- Добавление данных в базу данных
- Удаление данных из базы данных
- Поиск информации и ее вывод

Авторизация:

Если Вы уже зарегистрированы для работы с формами(интерфейсом), то Вам необходимо просто внести данные в соответствующие поля и Вас перенесет на страницу “Employee”

Но если Вы не зарегистрированы, то перед началом работы с формами(интерфейсом) нужно зарегистрироваться. Это можно сделать нажав на кнопку “регистрация” и ввести свой адрес электронной почты и пароль в соответствующие ячейки. После всех выполненных действий можете авторизоваться.

Работа с формами бд

Добавление новых записей

По завершению авторизации. если все предыдущие шаги были выполнены верно, Вы попадаете на новую форму “Employee”. Для того, чтобы начать работать с формой необходимо нажать на кнопку “Открыть”, после чего внизу должна открыться таблица базы данных. Через форму можно добавлять новые записи в базу данных. Для этого необходимо заполнить соответствующие пустые ячейки данными, которые необходимо внести в базу данных. После чего нажимаем на кнопку “Добавить”

Удаление записей

Если данные были введены неверно, то их можно удалить из базы данных. Для того, чтобы удалить данные из базы данных необходимо в нижнем окне, где отображается таблица, навести на номер записи, которую Вы хотите удалить и нажать на кнопку “Удалить”.

Поиск записей

Чтобы найти нужную запись, достаточно ввести любую информацию об этой записи и нажfnm кнопку “Найти”. В окошко, через которое отображается таблица, будут выделены необходимые данные.

Переключение между формами

Чтобы переключаться между формами предусмотрены кнопки, которые располагаются под блоком “Поиск”. Нажимая на одну из кнопок вы будете переходить на другую форму. Чтобы вернуться, достаточно нажать на крестик.

Разработка интерфейса.

Для начала в QT 5 дизайнере необходимо создать внешний вид форм. После создания внешнего вида формы нам необходимо добавить функции на кнопки и поля ввода, чтобы наш интерес исправно функционировал.
Назначение команд для функционирования форм

Импорт библиотек

```
import sqlite3
import sys
from PyQt5.QtWidgets import QApplication, QWidget, QTableWidgetItem
from PyQt5 import QtWidgets
```

Импорт форм

```
import login
import vxod
from glazkisave import Ui_glazkisave
from agent import Ui_Form
from composition import Ui_composition
from prodano import Ui_prodano
from product import Ui_product

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

Создание базы, где будут хранится логины и пароли зарегистрированных пользователей

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

Регистрирование пользователей. Связка кнопок и кода

```
class Registration(QtWidgets.QMainWindow, login.Ui_Form):
  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

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

```
        self.login.show()
        self.hide()


    def reg(self): #регитрация
        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')
```

Авторизация пользователей. Связка кнопок и кода

```
class Login(QtWidgets.QMainWindow, vxod.Ui_Form_1):# вход
    def __init__(self):
        super(Login,self).__init__()
        self.setupUi(self)
        self.lineEdit.setPlaceholderText('Введите логин')
        self.lineEdit_2.setPlaceholderText('Введите пароль')

        self.pushButton_2.pressed.connect(self.login)
        self.pushButton.pressed.connect(self.reg)



    def reg(self):
        self.reg=Registration()
        self.reg.show()
        self.hide()


    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.Emploe = Emploee1()
            self.Emploe.show()
            self.hide()
        else:
            self.label_2.setText(f'Ошибка авторизации')
    except Exception as e:
        self.label_2.setText(f'Ошибка авторизации')
```

Форма “Emploee”. Связка

```
STAFF_POSTS = ['Зам. директора', 'Директор', 'Менеджер', 'Аналитик',
'Консультант', 'Бухгалтер', 'Экономист' ]
#STAFF_POSTS - (константа) для выбора должности
class Emploee1 (QWidget, Ui_glazkisave):
  def __init__(self):
    super(Emploee1, self).__init__()
    self.setupUi(self)
    self.comboBox.addItems(STAFF_POSTS) # выбора должности сотрудники
    self.pushButton.clicked.connect(self.open) #открытие таблицы сотрудники
    self.pushButton_2.clicked.connect(self.insert) #добавить данные сотрудники
    self.pushButton_3.clicked.connect(self.deleteRow) #удаление
    self.pushButton_8.clicked.connect(self.show_2)#таблица агент
    self.pushButton_10.clicked.connect(self.serch)#поиск сотрудники
    self.pushButton_7.clicked.connect(self.show_Composition)#таблица композиция
    self.pushButton_5.clicked.connect(self.show_Prodano)#таблица продано
    self.pushButton_9.clicked.connect(self.show_Product)#таблица продукт
  def open(self): #кнопка добавить
```

```
    try:
        self.conn = sqlite3.connect('Glazki.db')
        cur = self.conn.cursor()
        data = cur.execute("select * from Emploee;")
        col_name = [i[0] for i in data.description]
        data_rows = data.fetchall()
    except Exception as e:
        print ("Ощибки с подключением к БД")
        return e
    self.tableWidget.setColumnCount(len(col_name))
    self.tableWidget.setHorizontalHeaderLabels(col_name)
    self.tableWidget.setRowCount(0)
    for i, row in enumerate(data_rows):
        self.tableWidget.setRowCount(self.tableWidget.rowCount()+1)
        for j, elen in enumerate(row):
            self.tableWidget.setItem(i, j,QTableWidgetItem(str(elen)))
    self.tableWidget.resizeColumnsToContents()

def update(self, query="select * from Emploee"): #обновление данных
    try:
        cur = self.conn.cursor()
        data = cur.execute(query).fetchall()
    except Exception as d:
        print(f"Проблемы с подкл {d}")
        return d
    self.tableWidget.setRowCount(0) #обнуляем все данные из таблицы
    #заносим по новой
    for i, row in enumerate(data):
        self.tableWidget.setRowCount(self.tableWidget.rowCount() + 1)
        for j, elen in enumerate(row):
            self.tableWidget.setItem(i, j, QTableWidgetItem(str(elen)))
    self.tableWidget.resizeColumnsToContents()


def insert(self): #кнопка добавить
        row = [self.lineEdit.text(), self.lineEdit_2.text(), self.lineEdit_3.text(),
self.comboBox.itemText(self.comboBox.currentIndex()), self.lineEdit_4.text(),
self.lineEdit_5.text(), self.lineEdit_7.text(), self.lineEdit_6.text()]
    try:
        cur = self.conn.cursor()
                cur.execute(f""" insert into
Emploee(surname,name,middle_name,job_title,passport_series,passpoet_number,pho
ne,mail)
         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


    def deleteRow(self): #удаление записей по id_emploee
        id = self.lineEdit_8.text()
        conn = sqlite3.connect('Glazki.db')
        c = conn.cursor()
        c.execute("DELETE FROM Emploee WHERE id_emploee=?", (id,))
        conn.commit()
        conn.close()
        self.update()

    def serch(self): #поиск по таблице
        search_text = self.lineEdit_9.text()
        # Проходим по всем строкам таблицы
        for row in range(self.tableWidget.rowCount()):
            # Проходим по всем ячейкам строки
            for column in range(self.tableWidget.columnCount()):
                # Получаем текст ячейки
                cell_text = self.tableWidget.item(row, column).text()
                if search_text.lower() in cell_text.lower():
                    self.tableWidget.selectRow(row)
                    return
```

Показ формы “Agent”

```
    def show_2(self):
        self.Agen = Agent1()
        self.Agen.show()
```

Показ формы “composition”

```
    def show_Composition(self):
        self.Comp = Composition1()
        self.Comp.show()
```

Показ формы “Prodano”

```
    def show_Prodano(self):
        self.prod = Prodano1()
        self.prod.show()
```

Показ формы “Product”

```
    def show_Product(self):
        self.produc=Product1()
        self.produc.show()
```

```
TYPE = ['1', '2', '3', '4', '5', '6']
```

Форма “Agent”

```
class Agent1 (QWidget, Ui_Form): #
  def __init__(self):
    super(Agent1, self).__init__()
    self.setupUi(self)
    self.pushButton.clicked.connect(self.open_1)#открыть таблицу агент
    self.comboBox.addItems(TYPE) #выбор
    self.pushButton_2.clicked.connect(self.insert_1) #добавить
    self.pushButton_3.clicked.connect(self.deleteRow_1)
    self.pushButton_10.clicked.connect(self.serch_1)



  def open_1(self): #открыть таблицу агент
      try:
          self.conn = sqlite3.connect('Glazki.db')
          cur = self.conn.cursor()
          data = cur.execute("select * from Agent;")
          col_name = [i[0] for i in data.description]
          data_rows = data.fetchall()
      except Exception as e:
          print ("Ощибки с подключением к БД")
          return e
      self.tableWidget.setColumnCount(len(col_name))
      self.tableWidget.setHorizontalHeaderLabels(col_name)
      self.tableWidget.setRowCount(0)
      for i, row in enumerate(data_rows):
          self.tableWidget.setRowCount(self.tableWidget.rowCount()+1)
          for j, elen in enumerate(row):
              self.tableWidget.setItem(i, j,QTableWidgetItem(str(elen)))
      self.tableWidget.resizeColumnsToContents()

   def update_1(self, query="select * from Agent"): #после добавление сразу видно
запись агент
      try:
          cur = self.conn.cursor()
          data = cur.execute(query).fetchall()
      except Exception as d:
          print(f"Проблемы с подкл {d}")
          return d
```

```
        self.tableWidget.setRowCount(0) #обнулмяем все данные из таблцы
        #заносим по новой
        for i, row in enumerate(data):
            self.tableWidget.setRowCount(self.tableWidget.rowCount() + 1)
            for j, elen in enumerate(row):
                self.tableWidget.setItem(i, j, QTableWidgetItem(str(elen)))
        self.tableWidget.resizeColumnsToContents()


    def insert_1(self): #кнопка добавить
                        row = [self.comboBox.itemText(self.comboBox.currentIndex()),
self.lineEdit.text(), self.lineEdit_2.text(), self.lineEdit_3.text(), self.lineEdit_4.text(),
self.lineEdit_5.text(), self.lineEdit_7.text(), self.lineEdit_6.text(),
self.lineEdit_16.text()]
        try:
            cur = self.conn.cursor()
                cur.execute(f"""insert into Agent(id_type_agent, Naimenovanie, Mail_agent,
phone_agent, Adress, priority, director, inn, kpp)
             values('{row[0]}','{row[1]}','{row[2]}','{row[3]}','{row[4]}','{row[5]}','{row[6
]}','{row[7]}', '{row[8]}')""")
            self.conn.commit()
            cur.close()
        except Exception as r:
            print("Не смогли добавить запись")
            return r
        self.update_1()#обращаемся к update чтобы сразу увидеть изменения в БД


    def deleteRow_1(self): #удалить агент
         id = self.lineEdit_8.text()
         conn = sqlite3.connect('Glazki.db')
         c = conn.cursor()
         c.execute("DELETE FROM Agent WHERE id_agent=?", (id,))
         conn.commit()
         conn.close()
         self.update_1()

    def serch_1(self): #поиск агент
        search_text = self.lineEdit_9.text()
        # Проходим по всем строкам таблицы
        for row in range(self.tableWidget.rowCount()):
            # Проходим по всем ячейкам строки
            for column in range(self.tableWidget.columnCount()):
                # Получаем текст ячейки
                cell_text = self.tableWidget.item(row, column).text()
```

```
            if search_text.lower() in cell_text.lower():
                self.tableWidget.selectRow(row)
                return
```

Форма “Composition”

```
class Composition1(QWidget, Ui_composition): #композиция
  def __init__(self):
    super(Composition1, self).__init__()
    self.setupUi(self)
    self.pushButton.clicked.connect(self.open_2)#открыть таблицу композиция
    self.pushButton_2.clicked.connect(self.insert_2) #добавить композиция
    self.pushButton_3.clicked.connect(self.deleteRow_2) #удалить композиция
    self.pushButton_10.clicked.connect(self.serch_2)#найти композиция

  def open_2(self): #открыть таблицу композиция
      try:
          self.conn = sqlite3.connect('Glazki.db')
          cur = self.conn.cursor()
          data = cur.execute("select * from Composition;")
          col_name = [i[0] for i in data.description]
          data_rows = data.fetchall()
      except Exception as e:
          print ("Ощибки с подключением к БД")
          return e
      self.tableWidget.setColumnCount(len(col_name))
      self.tableWidget.setHorizontalHeaderLabels(col_name)
      self.tableWidget.setRowCount(0)
      for i, row in enumerate(data_rows):
          self.tableWidget.setRowCount(self.tableWidget.rowCount()+1)
          for j, elen in enumerate(row):
              self.tableWidget.setItem(i, j,QTableWidgetItem(str(elen)))
      self.tableWidget.resizeColumnsToContents()

      def update_2(self, query="select * from Composition"): #после добавление
композиция
      try:
          cur = self.conn.cursor()
          data = cur.execute(query).fetchall()
      except Exception as d:
          print(f"Проблемы с подкл {d}")
          return d
      self.tableWidget.setRowCount(0) #обнулмяем все данные из таблцы
      #заносим по новой
      for i, row in enumerate(data):
          self.tableWidget.setRowCount(self.tableWidget.rowCount() + 1)
          for j, elen in enumerate(row):
```

```
            self.tableWidget.setItem(i, j, QTableWidgetItem(str(elen)))
        self.tableWidget.resizeColumnsToContents()


    def insert_2(self): #кнопка добавить композиция
                    row = [self.lineEdit.text(), self.lineEdit_2.text(), self.lineEdit_3.text(),
self.spinBox.text()]
        try:
            cur = self.conn.cursor()
                        cur.execute(f"""insert into Composition(vendor_code, receipt_date,
implementatin_date, id_agent )
            values('{row[0]}','{row[1]}','{row[2]}','{row[3]}')""")
            self.conn.commit()
            cur.close()
        except Exception as r:
            print("Не смогли добавить запись")
            return r
        self.update_2()#обращаемся к update чтобы сразу увидеть изменения в БД

    def deleteRow_2(self): #удалить агент
        id = self.lineEdit_8.text()
        conn = sqlite3.connect('Glazki.db')
        c = conn.cursor()
        c.execute("DELETE FROM Composition WHERE id_composition=?", (id,))
        conn.commit()
        conn.close()
        self.update_2()

    def serch_2(self): #поиск композиция
        search_text = self.lineEdit_9.text()
        # Проходим по всем строкам таблицы
        for row in range(self.tableWidget.rowCount()):
            # Проходим по всем ячейкам строки
            for column in range(self.tableWidget.columnCount()):
                # Получаем текст ячейки
                cell_text = self.tableWidget.item(row, column).text()
                if search_text.lower() in cell_text.lower():
                    self.tableWidget.selectRow(row)
                    return
```

Форма “Prodano”

```
class Prodano1 (QWidget, Ui_prodano): #продано
    def __init__(self):
        super(Prodano1, self).__init__()
```

```
        self.setupUi(self)
        self.pushButton.clicked.connect(self.open_3)#открыть таблицу композиция
        self.pushButton_2.clicked.connect(self.insert_3) #добавить
        self.pushButton_3.clicked.connect(self.deleteRow_3)#удалить продано
        self.pushButton_10.clicked.connect(self.serch_3)#найти

    def open_3(self): #открыть таблицу агент
        try:
            self.conn = sqlite3.connect('Glazki.db')
            cur = self.conn.cursor()
            data = cur.execute("select * from prodano;")
            col_name = [i[0] for i in data.description]
            data_rows = data.fetchall()
        except Exception as e:
            print ("Ощибки с подключением к БД")
            return e
        self.tableWidget.setColumnCount(len(col_name))
        self.tableWidget.setHorizontalHeaderLabels(col_name)
        self.tableWidget.setRowCount(0)
        for i, row in enumerate(data_rows):
            self.tableWidget.setRowCount(self.tableWidget.rowCount()+1)
            for j, elen in enumerate(row):
                self.tableWidget.setItem(i, j,QTableWidgetItem(str(elen)))
        self.tableWidget.resizeColumnsToContents()

    def update_3(self, query="select * from prodano"): #обнова продано
        try:
            cur = self.conn.cursor()
            data = cur.execute(query).fetchall()
        except Exception as d:
            print(f"Проблемы с подкл {d}")
            return d
        self.tableWidget.setRowCount(0) #обнулмяем все данные из таблцы
        #заносим по новой
        for i, row in enumerate(data):
            self.tableWidget.setRowCount(self.tableWidget.rowCount() + 1)
            for j, elen in enumerate(row):
                self.tableWidget.setItem(i, j, QTableWidgetItem(str(elen)))
        self.tableWidget.resizeColumnsToContents()


    def insert_3(self): #кнопка добавить продано
                row = [self.lineEdit.text(), self.lineEdit_3.text(), self.lineEdit_2.text(),
self.spinBox.text()]
        try:
```

```
            cur = self.conn.cursor()
                                cur.execute(f"""insert  into  prodano(vendor_code,  id_agent,
date_implementation, number_products )
            values('{row[0]}','{row[1]}','{row[2]}',{row[3]})""")
            self.conn.commit()
            cur.close()
        except Exception as r:
            print("Не смогли добавить запись")
            return r
        self.update_3()

    def deleteRow_3(self): #удалить продано
        id = self.lineEdit_8.text()
        conn = sqlite3.connect('Glazki.db')
        c = conn.cursor()
        c.execute("DELETE FROM prodano WHERE id=?", (id,))
        conn.commit()
        conn.close()
        self.update_3()

    def serch_3(self): #поиск продано
        search_text = self.lineEdit_9.text()
        # Проходим по всем строкам таблицы
        for row in range(self.tableWidget.rowCount()):
            # Проходим по всем ячейкам строки
            for column in range(self.tableWidget.columnCount()):
                # Получаем текст ячейки
                cell_text = self.tableWidget.item(row, column).text()
                if search_text.lower() in cell_text.lower():
                    self.tableWidget.selectRow(row)
                    return
```

Форма “Product”

```
class Product1(QWidget, Ui_product): #продукт
    def __init__(self):
        super(Product1, self).__init__()
        self.setupUi(self)
        self.pushButton.clicked.connect(self.open_4)#открыть продукт
        self.pushButton_2.clicked.connect(self.insert_4)#добавить продукт
        self.pushButton_3.clicked.connect(self.deleteRow_4)#удалить продукт
        self.pushButton_10.clicked.connect(self.serch_4)#найти продукт


    def open_4(self): #открыть таблицу продукт
        try:
```

```
        self.conn = sqlite3.connect('Glazki.db')
        cur = self.conn.cursor()
        data = cur.execute("select * from Product;")
        col_name = [i[0] for i in data.description]
        data_rows = data.fetchall()
    except Exception as e:
        print ("Ощибки с подключением к БД")
        return e
    self.tableWidget.setColumnCount(len(col_name))
    self.tableWidget.setHorizontalHeaderLabels(col_name)
    self.tableWidget.setRowCount(0)
    for i, row in enumerate(data_rows):
        self.tableWidget.setRowCount(self.tableWidget.rowCount()+1)
        for j, elen in enumerate(row):
            self.tableWidget.setItem(i, j,QTableWidgetItem(str(elen)))
    self.tableWidget.resizeColumnsToContents()

def update_4(self, query="select * from Product"): #обнова продукт
    try:
        cur = self.conn.cursor()
        data = cur.execute(query).fetchall()
    except Exception as d:
        print(f"Проблемы с подкл {d}")
        return d
    self.tableWidget.setRowCount(0) #обнулмяем все данные из таблцы
    #заносим по новой
    for i, row in enumerate(data):
        self.tableWidget.setRowCount(self.tableWidget.rowCount() + 1)
        for j, elen in enumerate(row):
            self.tableWidget.setItem(i, j, QTableWidgetItem(str(elen)))
    self.tableWidget.resizeColumnsToContents()


def insert_4(self): #кнопка добавить продукт
            row = [self.lineEdit.text(),    self.spinBox_3.text(), self.lineEdit_2.text(),
self.spinBox.text(), self.spinBox_2.text() , self.lineEdit_3.text()]
    try:
        cur = self.conn.cursor()
                cur.execute(f"""insert into Product(naimenovanie, id_type_product,
vendor_code, Number_people_product_ion, Production_workschop_number,
Min_price_agent )
        values('{row[0]}','{row[1]}','{row[2]}',{row[3]}, '{row[4]}', '{row[5]}')""")
        self.conn.commit()
        cur.close()
    except Exception as r:
```

```
            print("Не смогли добавить запись")
            return r
        self.update_4()

    def deleteRow_4(self): #удалить продукт
        id = self.lineEdit_8.text()
        conn = sqlite3.connect('Glazki.db')
        c = conn.cursor()
        c.execute("DELETE FROM Product WHERE id=?", (id,))
        conn.commit()
        conn.close()
        self.update_4()

    def serch_4(self): #поиск продукт
        search_text = self.lineEdit_9.text()
        # Проходим по всем строкам таблицы
        for row in range(self.tableWidget.rowCount()):
            # Проходим по всем ячейкам строки
            for column in range(self.tableWidget.columnCount()):
                # Получаем текст ячейки
                cell_text = self.tableWidget.item(row, column).text()
                if search_text.lower() in cell_text.lower():
                    self.tableWidget.selectRow(row)
                    return


App = QtWidgets.QApplication([])
window = Login()
window.show()
App.exec()
```

Заключение

В ходе проделанной работы были выполнены все поставленные задачи: была разработана er-диаграмма по предметной области, по er диаграмме, которую мы разработали была создана база данных, которая была заполнена данными, также был разработан интерфейс непосредственно под саму базу данных, был создан файл main.pu, в котором мы при помощи языка python связали формы и базу данных, назначили действия на кнопки и поля ввода, которые располагаются на форме.