Министерство образования и науки Российской Федерации Федеральное агентство по образованию Федеральное государственное бюджетное образовательное учреждение высшего образования «Вятский государственный университет»

Факультет автоматики и вычислительной техники

Кафедра электронных вычислительных машин

Лабораторная работа №5

по курсу «Базы данных»

Выполнил студент группы ИВТ-31\_\_\_\_\_\_\_\_\_\_\_\_\_\_/Птахова А.М/

Проверил доцент кафедры ЭВМ \_\_\_\_\_\_\_\_\_\_\_\_\_\_/ Клюкин В.Л./

Киров 2023

1. Задание

- разработать приложение на C# для работы с БД

- приложение должно выполнять следующие действия: отображение БД, добавление записи, удаление записи, сортировка БД

2. ER – диаграмма для разработанной структуры БД

ER – диаграмма для разработанной структуры БД представлена на рисунке 1.

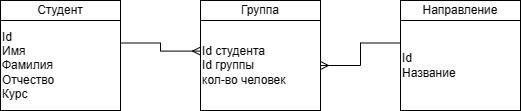


Рисунок 1 – ER- диаграмма

3. Листинг кода

import tkinter as tk

from tkinter import \*

from tkinter import ttk

from db import database

from tkinter.messagebox import showwarning

class App(tk.Tk):

def \_\_init\_\_(self):

super().\_\_init\_\_()

opts={'ipadx':5, 'ipady':5, 'sticky':'nswe'}

label\_table = tk.Label(self, text="Таблица успеваемости студентов")

label\_table.grid(row=0,column=1, \*\*opts) #название таблицы

group = ["ИВТб-3301-04-00", "ИВТб-3302-04-00"]

columns = ("id","name","surname","mid\_name", "id\_group", "addres")

opts={'ipadx':5, 'ipady':5, 'sticky':'nswe'}

#таблица - создание

tree = ttk.Treeview(column=columns, show="headings")

label\_tab = tk.Label(self, text="")

#таблица - описание

label\_tab.grid(row=4, column=0, columnspan=5, \*\*opts)

tree.grid(row=6, column=0,columnspan=5, sticky="nsew")

tree.heading("id", text="№")

tree.heading("name", text="Имя")

tree.heading("surname", text="Фамилия")

tree.heading("mid\_name", text="Отчество")

tree.heading("id\_group", text="№ группы")

tree.heading("addres", text="Адрес")

#настройка выборки - функция

def add\_string():

for i in tree.get\_children():

tree.delete(i)

#edit\_surname.delete(0, 'end')

selection = combobox.current()+1

#edit\_surname.insert(0, selection)

dtbs = database()

dtbs.openConnection()

s = dtbs.chooseGroup(selection)

dtbs.closeConnection()

for person in s:

tree.insert("", END, values=person)

return

#настройка выборки - создание

label\_choose = tk.Label(self, text="Настройка выборки")

#label\_surname = tk.Label(self, text="Фамилия")

label\_name\_group = tk.Label(self, text="Группа")

#edit\_surname = ttk.Entry()

combobox = ttk.Combobox(values=group)

btn\_choose= ttk.Button(text="Применить", command=add\_string)

#настройка выборки - описание

#label\_surname.grid(row=2, column=0, \*\*opts)

label\_name\_group.grid(row=2, column=1, \*\*opts)

#edit\_surname.grid(row=3, column=0, \*\*opts)

combobox.grid(row=3, column=1, \*\*opts)

btn\_choose.grid(row =3, column=2, \*\*opts)

label\_choose.grid(row=1,column=0, \*\*opts)#настройка выборки

#добавление записи - создание

label\_tab2 = tk.Label(self, text="")

label\_add = tk.Label(self, text="Добавление записи")

label\_add\_name = tk.Label(self, text="Имя")

label\_add\_surname = tk.Label(self, text="Фамилия")

label\_add\_mid\_name = tk.Label(self, text="Отчество")

label\_add\_adress = tk.Label(self, text="Адрес")

label\_add\_group = tk.Label(self, text="Группа")

label\_tab3 = tk.Label(self, text="")

edit\_add\_name = ttk.Entry()

edit\_add\_surname = ttk.Entry()

edit\_add\_mid\_name = ttk.Entry()

edit\_add\_adress = ttk.Entry()

combobox2=ttk.Combobox(values=group)

def add\_str():

name = edit\_add\_name.get()

surname = edit\_add\_surname.get()

mid\_name = edit\_add\_mid\_name.get()

adres = edit\_add\_adress.get()

selection = combobox2.current()+1

dtbs = database()

dtbs.openConnection()

dtbs.addPerson(name, surname, mid\_name, selection, adres, 2)

dtbs.closeConnection()

for i in tree.get\_children():

tree.delete(i)

dtbs = database()

dtbs.openConnection()

s = dtbs.getGroup()

dtbs.closeConnection()

for person in s:

tree.insert("", END, values=person)

btn\_add=ttk.Button(text="добавить", command = add\_str)

opts={'ipadx':5, 'ipady':5, 'sticky':'nswe'}

#добавление записи - описание

label\_tab2.grid(row=7, column=0, columnspan=4, \*\*opts)

label\_add.grid(row=8, column =0, \*\*opts)

label\_add\_name.grid(row=9, column=0, \*\*opts)

label\_add\_surname.grid(row=9, column=1, \*\*opts)

label\_add\_mid\_name.grid(row=9, column=2, \*\*opts)

label\_add\_adress.grid(row=9, column=3, \*\*opts)

label\_add\_group.grid(row=9, column=4, \*\*opts)

edit\_add\_name.grid(row=10,column=0, \*\*opts)

edit\_add\_surname.grid(row=10, column=1, \*\*opts)

edit\_add\_mid\_name.grid(row=10, column=2, \*\*opts)

edit\_add\_adress.grid(row=10,column=3, \*\*opts)

combobox2.grid(row=10, column=4, \*\*opts)

btn\_add.grid(row=10, column=5, \*\*opts)

label\_tab3.grid(row=11, column =0, columnspan=5, \*\*opts)

def del\_str():

showwarning(title="?", message = "Вы хотите удалить строку?")

selected=tree.item(tree.selection())

person = selected['values']

s = person[0]

dtbs = database()

dtbs.openConnection()

dtbs.delPerson(s)

dtbs.closeConnection()

edit\_add\_surname.insert(0, s)

for i in tree.get\_children():

tree.delete(i)

dtbs = database()

dtbs.openConnection()

s = dtbs.getGroup()

dtbs.closeConnection()

for persons in s:

tree.insert("", END, values=persons)

#удаление - создание

btn\_del\_str = ttk.Button(text="Удалить строку", command = del\_str)

btn\_del\_str.grid(row=8, column =4, \*\*opts)

#начальный вывод

dtbs = database()

dtbs.openConnection()

s = dtbs.getGroup()

dtbs.closeConnection()

for person in s:

tree.insert("", END, values=person)

if \_\_name\_\_ == "\_\_main\_\_":

app = App()

app.mainloop()

**db.py**

import postgresql

from postgresql import exceptions

class database:

\_connection = None

\_db\_url = "pq://postgres:74771@localhost:5432/lab"

def openConnection(self):

try:

self.\_connection = postgresql.open(self.\_db\_url)

print("Подключено")

except postgresql.exceptions.AuthenticationMethodError: # неподдерживаемый тип авторизации

print("Ошибка! Неподдерживаемый тип авторизации.")

except postgresql.exceptions.ClientCannotConnectError: # клиенту не удалось установить соединение с сервером

print("Ошибка! Клиенту не удалось установить соединение с сервером.")

except postgresql.exceptions.ConnectTimeoutError: # клиенту не удалось установить соединение в заданное время

print("Ошибка! Клиенту не удалось установить соединение в заданное время.")

except postgresql.exceptions.DriverError: # Ошибка в реализации драйвера

print("Ошибка в реализации драйвера")

def getGroup(self):

try:

return self.\_connection.query("SELECT \* FROM public.student")

except postgresql.exceptions.ConnectionDoesNotExistError:

print("Error")

except:

print("Error1")

def chooseGroup(self, number):

try:

sql = "SELECT \* FROM public.student WHERE id\_group ="+str(number)

ps = self.\_connection.query(sql)

return ps

except:

print("Error!")

def addPerson(self, name, surname, mid\_name, id\_group, adres, course):

try:

val\_str = "('"+name + "','"+surname+"','"+mid\_name+"',"+str(id\_group)+",'"+adres+"',"+str(course)+")"

self.\_connection.query("INSERT INTO student(name, surname, middle\_name, id\_group, address, cource) VALUES"+val\_str)

except:

print("Error")

def delPerson(self, id\_person):

try:

self.\_connection.query("DELETE FROM student where id="+str(id\_person))

except:

print("Error")

def closeConnection(self):

self.\_connection.close()

4. Вывод

В ходе выполнения лабораторной работы были получены навыки разработки приложения на Python для взаимодействия с разработанной БД.