****

Институт информационных и вычислительных технологий

Кафедра управления и интеллектуальных технологий

**Отчёт по научно-исследовательской работе**

**По теме «Основы работы с реляционными базами данных с использованием языка Python 2023»**

**Этап 4**

Выполнил студент: Михайловский Михаил Юрьевич

Группа: А-03-21

Вариант: 30

Проверил: Фомин Геннадий Александрович

**Москва 2023**

Оглавление

[Этап 4 3](#_Toc153042917)

[Постановка задания 3](#_Toc153042918)

[Структура БД 3](#_Toc153042919)

[Описание программы 5](#_Toc153042920)

[Модуль Database.py 5](#_Toc153042921)

[Модуль IO\_funcs.py 6](#_Toc153042922)

[Модуль E4\_funcs.py 6](#_Toc153042923)

[Модуль E4.py 7](#_Toc153042924)

[Примеры работы с программой 7](#_Toc153042925)

[Открытыие базы данных 7](#_Toc153042926)

[Получение информации о ВУЗах по статусу 7](#_Toc153042927)

[Получение информации о преподавательских составах в федеральных университетах 8](#_Toc153042928)

[Приложение 8](#_Toc153042929)

[Модуль DataBase.py 8](#_Toc153042930)

[Модуль IO\_funcs.py 13](#_Toc153042931)

[Модуль E4.py 16](#_Toc153042932)

[Модуль E4\_funcs.py 17](#_Toc153042933)

Этап 4

Постановка задания

1. Обеспечить выбор из списка статуса вуза, интересующего пользователя. Составить и отобразить на экране перечень полных наименований вузов, имеющих выбранный статус и у которых в БД отсутствуют ФИО ректора и справочный телефон вуза.

2. Рассчитать и представить в виде таблицы распределения процента преподавателей, имеющих ученые степени кандидата и доктора наук, для Федеральных университетов. Таблица должна иметь поля: порядковый номер, Федеральный университет, количество преподавателей, количество преподавателей с учеными степенями доктора и кандидата наук в данном Федеральном университете, процентное отношение преподавателей со степенями к общему числу преподавателей в вузе. В шапке таблицы указываются имена полей. Нижняя строка таблицы – итоговая: с суммарными значениями количества преподавателей в Федеральных университетах и количества «остепененных» преподавателей, а также общее процентное отношение этих значений.

Структура БД

1. Таблица "Картотека вузов" - **vuzkart** с записями, содержащими сведения о вузах России.

Структура таблицы **vuzkart**

|  |  |  |  |
| --- | --- | --- | --- |
| Ключ | Имя поля | Тип данных | Описание |
| V | Codvuz | char(6) | Код вуза |
|  | z1 | char(200) | Полное наименование вуза |
|  | z1full | char(250) | Полное юридическое наименование вуза |
|  | z2 | char(12) | Сокращённое наименование |
|  | z2ustav | char(100) | Сокращённое наименование по уставу |
|  | foundyear | integer(4) | Год основания вуза |
|  | z8 | char(110) | Юридический адрес вуза |
|  | z9 | char(40) | Справочный телефон вуза |
|  | z12 | char(20) | Телефон руководителя вуза |
|  | e\_mail | char(30) | Адрес электронной почты |
|  | www | char(30) | Адрес официального сайта вуза |
|  | z15 | char(40) | Ректор вуза |
|  | region | char(17) | Федеральный округ |
|  | city | char(20) | Город |
|  | status | char(15) | Статус вуза: университет, академия, институт |
|  | oblname | char(40) | Субъект федерации по расположению вуза |
|  | gr\_ved | char(3) | Категория вуза - принадлежность к ведущим вузам (ФУ – федеральный университет, НИУ – национальный исследовательский университет) |
|  | prof | char(2) | Профиль вуза: ИТ – инженерно-технический, КЛ – классический университет, ГП – гуманитарно-педагогический, МП – многопрофильный |

1. Таблица **vuzstat**, содержащая статистические данные по вузам.

Структура таблицы **vuzstat**

|  |  |  |  |
| --- | --- | --- | --- |
| Ключ | Имя поля | Тип данных | Описание |
| V | codvuz | char(6) | Код вуза (вторичный) |
|  | PPS | integer(4) | Численность профессорско-преподавательского состава (ППС) |
|  | PR | integer(4) | ППС – профессоры |
|  | DC | integer(4) | ППС - доценты |
|  | DN | integer(4) | ППС - доктора наук |
|  | KN | integer(4) | ППС - кандидаты наук |
|  | ZOB | integer(4) | Количество зданий общежитий |
|  | STUD | integer(4) | Количество студентов, всего |
|  | BAC | integer(4) | Обучается по программам бакалавриата |
|  | SPEC | integer(4) | Обучается по программам подготовки специалистов |
|  | MAG | integer(4) | Обучается по программам магистратуры |
|  | ST\_OCH | integer(4) | Студентов по очной форме обучения |
|  | ST\_OCH\_Z | integer(4) | Студентов по очно-заочной форме обучения |
|  | ST\_Z | integer(4) | Студентов по заочной форме обучения |
|  | ASP | integer(4) | Аспирантов всего |
|  | ASP\_OCH | integer(4) | Аспирантов по очной форме обучения |

Описание программы

Программа написана на языке программирования python 3.11.5. В ней содержится четыре модуля: Database.py, IO\_funcs.py, E4.py, E4\_funcs.py. Список используемых стандартных модулей: os, pandas, sqlite3.

Программа разработана как модификация к программе написанной на этапе 3 данной научно-исследовательской работы. Модуль **E4.py** представляет собой модифицированную версию **prog.py** в которой добавлены варианты действий с БД для получения информации в соответствии с постановкой задачи.

Предназначение программы: работа с базой данных соответствующей структуре из раздела **структура бд.**

Возможности: отображение содержимого таблицы; отображение содержимого таблицы, с учетом фильтра; сохранение данных из таблицы в файл; удаление строк по фильтру; изменение значений столбцов для записей по фильтру; внесение новых строк в БД; получение данных по вузам с заданным статусом (их количество, количество без данных о ректоре, количество без данных о справочном телефоне) и вывод списка вузов без данных о ректоре и справочном телефоне; получение информации о распределении преподавателей с и без учёных степеней в федеральных университетах.

Полный код программы размещён в приложении. Здесь опишем только структуру программы и функции модулей.

**Модуль Database.py**

Содержит класс *DataBase*, предназначенный для работы с базой данных. Этот модуль связан со следующим модулем **IO\_funcs.py**.

Для всех функций в этом модуле, если table\_name равен None, то этот параметр принимается равным self.selected\_table. Если другие параметры по умолчанию равны None, то они вводятся пользователем.

Функции:

**get\_tables\_names(self) –** получение имён таблиц содержащихся в базе данных, путь к которой сохранён в self.bd\_file.

**select\_**table(**self, table\_name = None)** – изменение выбранной таблицы. Выбранная таблица хранится в self.selected\_table*.*

get\_column\_names(self, table\_name = None)– получение кортежа имён столбцов в выбранной таблице*.*

get\_table\_contents(self, table\_name = None) – получение содержимого выбранной таблицы.

show\_table(self, table\_name = None) – печать таблицы на экран.

save\_table(self, table\_name = None) – Сохранение содержимого таблицы в файл, с названием, которое введёт пользователь.

get\_filtered(self, table\_name= None, filter=None, prompt='Полученные данные', print\_data=True) – Получение данных с учётом фильтра, передаваемового в SQL запросе после ключевого слова WHERE.

filter – SQL фильтр. Если равен None, то фильтр вводит пользователь.

prompt – Строка, которая выводится перед печатью полученных данных

print\_data – Если True, то полученные данные будут распечатаны.

change\_values(self, table\_name = None, filter = None) – изменение значений в определённом столбце для строк, удовлетворяющих фильтру.

delete\_values(self, table\_name = None, filter = None) – Удаление строк выделенных фильтром.

insert\_values(self, table\_name = None) – Ввод и вставка новых строк в таблицу.

Модуль IO\_funcs.py

Данный модуль содержит функции для ввода данных от пользователя и вывода данных на экран или в файл.

Функции:

user\_select\_table(DataBase) – Выбор пользователем таблицы из имеющихся в переданной базе данных таблиц.

user\_select\_from\_list(choose\_from, prompt='Выберите действие:', compact\_form = False) – служебная функция. С помощью неё пользователь выбирает один из имеющихся вариантов.

choose\_from – Набор вариантов. Если compact\_form = False, то это словарь, ключи которого, это варианты для выбора, а значения это описания к вариантам, которые будут выведены пользователю. Если compact\_form = True, то это просто список вариантов.

prompt – Сообщение, которое будет выведено пользователю, перед выведением вариантов выбора.

compact\_form – определяет, будет ли использоваться компактное представление вариантов к выбору.

user\_get\_save\_file\_name() – функция, для выбора пользователем файла, куда сохранять данные. Если пользователь указывает существующий файл, запрашивается дополнительное подтверждение на перезапись файла.

print\_table(data, columns=None, is\_dataframe=False) – Печать данных в виде таблицы с указанными столбцами.

data – список со списками, являющимися строками в таблице.

columns – список с названиями столбцов.

is\_dataframe – флаг указывающий на тип data. Если False, то ожидается data в виде вложенных списков/кортежей. Иначе ожидается data в виде датафрейма

user\_constuct\_simplest\_filter(DataBase, table\_name = None) – ввод пользователем простейшего фильтра вида “<имя стобца> <отношение> <значение>”

<имя столбца>: выбирается из столбцов в выбранной таблице.

<отношение>: выбирается из вариантов >=, >, =, <, <=

<значение>: это может быть число, название столбца или строка. Строчные данные небходимо оборачивать в кавычки.

user\_construct\_filter(DataBase, table\_name = None) – ввод пользователем фильтра вида “<простейший фильтр> <и/или> <простейший фильтр> <и/или> …”

Модуль E4\_funcs.py

Данный модуль содержит функции для решения задач, поставленных в этапе 4 данной НИР.

task1(DataBase: db.DataBase) - вывод данных по вузам с заданным статусом (их количество, количество без данных о ректоре, количество без данных о справочном телефоне) и вывод списка вузов без данных о ректоре и справочном телефоне.

task2(DataBase: db.DataBase) - вывод информации о распределении преподавателей с и без учёных степеней в федеральных университетах.

Модуль E4.py

Основной модуль, задающий порядок выполнения программы. Порядок работы:

1. Пользователем выбирается файл базы данных;
2. Создаётся объект класса DataBase из *DataBase.py*, который открывает заданную базу данных и получает данные о таблицах, хранящихся в ней;
3. Если таблиц больше одной, пользователю показываются найденные таблицы и содержащиеся в них столбцы и он выбирает таблицу, с которой будет работать;
4. Пользователю представляются на выбор все действия, которые можно совершать с выбранной таблицей. Действия выбираются и выполняются до тех пор, пока пользователь не выберет пукт выхода из программы.

Примеры работы с программой

Открытыие базы данных

При запуске программы необходимо указать путь к файлу с базой данных (рис. 1). После открытия его программа выведет информацию о найденных таблицах в базе данных. После чего пользователь выбирает таблицу, с которй можно будет проводить основные действия (отображения содержимого, сохранения данных в файл и т.д.).

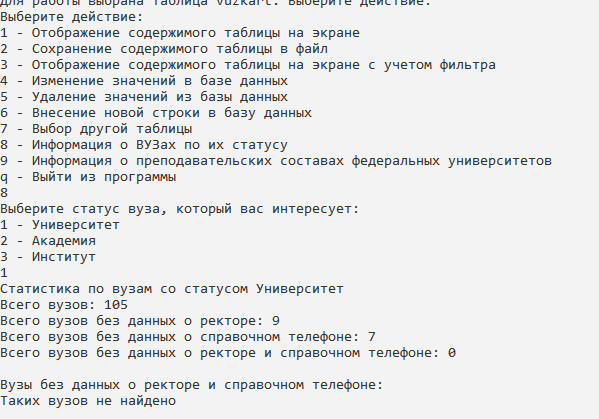


**Рис. 1. Открытыие базы данных**

Получение информации о ВУЗах по статусу

Для получения информации о ВУЗах по статусу в основном меню выбирается действие под номером 8 (рис. 2). Затем выбирается статус, из списка найденных статусов в базе данных. После чего выводятся данные об общем количестве соответствующих ВУЗов, и количестве ВУЗов с пропусками.

В результате будет выведен список ВУЗов без данных о ректоре и справочном телефоне.

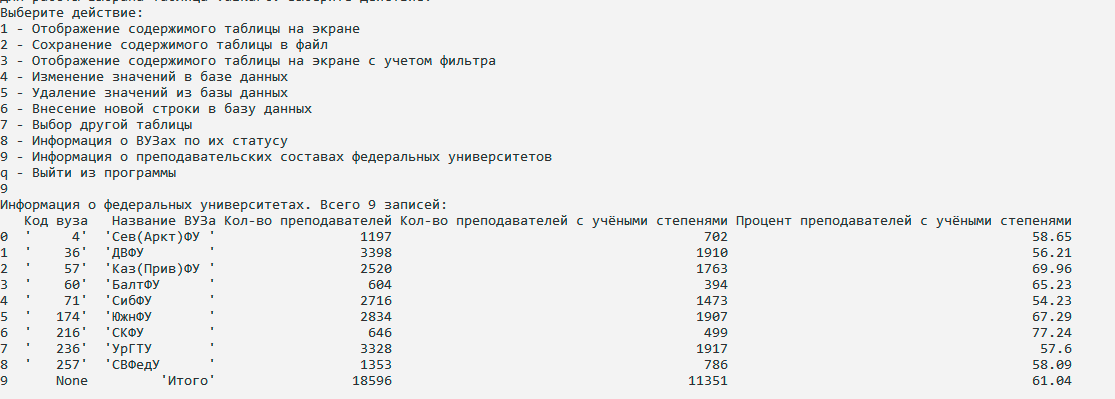


**Рис. 2. Информация о ВУЗах по статусу**

Получение информации о преподавательских составах в федеральных университетах

Для получения информации о преподавательских составах федеральных используется действие под номером 9.

В результате получаем таблицу с федеральными университетами, общим количеством преподавателей, количеством преподавателей с учёными степенями и процентнтым отношением преподавателей с учёными степенями к общему количеству преподавателей, а также общей статистикой по всем этим записям (рис. 3).



**Рис. 3. Полученная инфорамция о преподавательских составах**

Приложение

Модуль DataBase.py

#DataBase.py

import sqlite3

import IO\_funcs as io

import pandas as pd

class DataBase:

    bd\_file = None

    tables = None

    selected\_table = None

    columns = None

    data\_buff = None

    def \_\_init\_\_(self, bd\_file: str):

        self.bd\_file = bd\_file

        self.tables = self.get\_tables\_names()

        if self.tables:

            self.selected\_table = self.tables[0]

            self.columns = self.get\_column\_names(self.selected\_table)

    def get\_tables\_names(self):

        '''Получение названий таблиц в БД bd\_file'''

        con = sqlite3.connect(self.bd\_file)

        cur = con.cursor()

        tables\_sql = '''SELECT name FROM sqlite\_master

        WHERE type IN ('table','view') AND name NOT LIKE 'sqlite\_%'

        UNION ALL SELECT name FROM sqlite\_temp\_master WHERE type IN ('table','view') ORDER BY 1;'''

        cur.execute(tables\_sql)

        data = cur.fetchall()

        tables = [ data[i][0] for i in range(len(data))]

        cur.close()

        con.close()

        return tables

    def select\_table(self, table\_name = None):

        '''Изменить выбранную для работы таблицу'''

        if len(self.tables) == 1:

            return

        if table\_name is None:

            self.selected\_table = io.user\_select\_table(self)

            self.columns = self.get\_column\_names(self.selected\_table)

            return

        assert table\_name in self.tables

        self.selected\_table = table\_name

        self.columns = self.get\_column\_names(table\_name)

    def get\_column\_names(self, table\_name = None):

        '''Получение списка названий колонок таблицы table\_name из БД bd\_file'''

        if table\_name is None and self.columns is not None:

            #Возвращение столбцов выбранной таблицы

            return self.columns

        con = sqlite3.connect(self.bd\_file)

        cur = con.cursor()

        cur.execute(f'PRAGMA table\_info({table\_name})')

        columns = [i[1] for i in cur.fetchall()]

        cur.close()

        con.close()

        return columns

    def get\_table\_contents(self, table\_name = None):

        '''Получение содержимого таблицы из БД'''

        if table\_name is None:

            table\_name = self.selected\_table

        con = sqlite3.connect(self.bd\_file)

        cur = con.cursor()

        cur.execute(f'SELECT \* FROM {table\_name}')

        data = cur.fetchall()

        cur.close()

        con.close()

        return data

    def show\_table(self, table\_name = None):

        '''Напечатать таблицу'''

        if table\_name is None:

            table\_name = self.selected\_table

        data = self.get\_table\_contents(table\_name)

        print('\nДанные из таблицы', self.bd\_file)

        io.print\_table(data, self.get\_column\_names(table\_name))

    def save\_table(self, table\_name = None):

        '''Сохранение таблицы из БД в файл'''

        if table\_name is None:

            table\_name = self.selected\_table

        data = self.get\_table\_contents(table\_name)

        columns = self.get\_column\_names(table\_name)

        file\_to\_save = io.user\_get\_save\_file\_name()

        if file\_to\_save:

            data\_pd = pd.DataFrame(data, columns=columns)

            data\_pd.to\_csv(file\_to\_save)

    def get\_filtered(self, table\_name= None, filter=None, prompt='Полученные данные', print\_data=True):

        '''Получение данных из заданной таблицы с учётом фильтра передаваемого после ключевого слова WHERE в SQL запросе.

        table\_name - имя таблицы из которой получить данные. Если None, то используется выбранная таблица, хранящаяся в self.selected\_table

        filter - фильтр, который будет указан после WHERE в SQL запросе.

        prompt - Сообщение, которое будет выведено перед выводом полученных данных.

        print\_data - нужно ли печатать таблицу полученных данных. Значение bool.

        '''

        if table\_name is None:

            table\_name = self.selected\_table

        if filter is None:

            filter = io.user\_construct\_filter(self)

        sql = f'SELECT \* FROM {table\_name} WHERE {filter}'

        con = sqlite3.connect(self.bd\_file)

        cur = con.cursor()

        cur.execute(sql)

        data = cur.fetchall()

        cur.close()

        con.close()

        if prompt:

            print(prompt)

        if print\_data:

            io.print\_table(data, self.get\_column\_names(table\_name))

        return data

    def change\_values(self, table\_name = None, filter = None):

        if table\_name is None:

            table\_name = self.selected\_table

        if filter is None:

            filter = io.user\_construct\_filter(self)

        self.get\_filtered(table\_name, filter=filter, prompt='Данные по заданному фильтру.', print\_data=True)

        print()

        columns = self.get\_column\_names(table\_name)

        selected\_column = io.user\_select\_from\_list(columns, 'Введите столбец, значение в котором нужно изменить. ', compact\_form=True)

        while True:

            value = input('Введите значение для записи: ')

            if value.isnumeric():

                break

            if value[0] == value[-1] and value[0] in ['"', "'"]:

                break

            print(f'Значение должно быть числом или обёрнуто в кавычки. Введено {value}')

        con = sqlite3.connect(self.bd\_file)

        cur = con.cursor()

        cur.execute(f'UPDATE {table\_name} SET {selected\_column}={value} WHERE {filter}')

        con.commit()

        cur.close()

        con.close()

    def delete\_values(self, table\_name = None, filter = None):

        if table\_name is None:

            table\_name = self.selected\_table

        if filter is None:

            filter = io.user\_construct\_filter(self)

        data = self.get\_filtered(table\_name, filter=filter, prompt='Данные по заданному фильтру.', print\_data=True)

        ans = input(f'Будет удалено {len(data)} строк. Продолжить? y/n: ')

        if ans != 'y':

            return

        print()

        if filter == '':

            ans = input(f'Удаление данных с заданным фильтром {filter} сотрёт всё содержимое таблицы. Всё равно удалить? y/n: ')

            if ans != 'y':

                return

        con = sqlite3.connect(self.bd\_file)

        cur = con.cursor()

        cur.execute(f'DELETE FROM {table\_name} WHERE {filter}')

        con.commit()

        cur.close()

        con.close()

    def insert\_values(self, table\_name = None):

        if table\_name is None:

            table\_name = self.selected\_table

        columns = self.get\_column\_names(table\_name)

        while True:

            rows = input('Введите количество строк, которое вы хотите добавить: ')

            try:

                rows = int(rows)

                break

            except ValueError:

                print(f'Введите целое значение. Получено {rows}')

        print()

        rows\_data = []

        for i in range(rows):

            print(f'{i+1} строка.')

            row = []

            for column in columns:

                value = input(f'{column} = ')

                row.append(value)

            rows\_data.append(tuple(row))

            print()

        values\_format = repr(tuple(['?',]\*len(columns))).replace("'", '')

        con = sqlite3.connect(self.bd\_file)

        cur = con.cursor()

        cur.executemany(f'INSERT INTO {table\_name} {repr(tuple(columns))} VALUES {values\_format}', rows\_data)

        con.commit()

        cur.close()

        con.close()

Модуль IO\_funcs.py

#IO\_funcs.py

import os

import pandas as pd

import DataBase as db

def user\_select\_table(DataBase):

    while True:

            print('Таблицы данной базе данных:', \*DataBase.tables)

            selected\_table = input('Введите таблицу, с которой будете работать: ')

            if selected\_table in DataBase.tables:

                break

            print('В данной базе данных таблицы', selected\_table, 'нет')

    return selected\_table

def user\_select\_from\_list(choose\_from, prompt='Выберите действие:', compact\_form = False):

    '''Выбор пользователем варианта из предоставленных в списке

    choose\_from - словарь, с вариантами выбора в качестве ключей и их описаниями в качестве значений. При заданном compact\_form=True choose\_from это список вариантов

    prompt - приглашение к вводу перед перечислением вариантов выбора.

    compact\_form - Если True то вывод происходит в компактном виде. Если False, то на каждый вариант используется целая строка и соответствующее ей описание'''

    while True:

        if prompt:

            print(prompt, end='')

        if compact\_form:

            str\_variants = ', '.join(choose\_from)

            ans = input(f'Варианты. {str\_variants}: ')

            if ans not in choose\_from:

                print(f'Варианта {ans} в списке действий нет.')

                continue

            return ans

        print()

        for choose in sorted(choose\_from.keys()):

            print(f'{choose} - {choose\_from[choose]}')

        ans = input()

        if ans not in choose\_from.keys():

            print(f'Варианта {ans} в списке действий нет.')

            continue

        return ans

def user\_get\_save\_file\_name():

    '''Запрос и получение названия файла для записи данных. Для отмены используется комбманция !q. Если пользователь вводит

    существующий файл, то требуется дополнительное подтверждение для перезаписи файла. '''

    ans = ''

    while True:

        print('Введите название файла для сохранения. Для отмены введите !q')

        ans = input('Сохранить данные в файл: ')

        if ans == '!q':

            return None

        if os.path.isfile(ans):

            print('Такой файл существует. Уверены, что хотите его перезаписать?\ny - перезаписать, n - не перезаписывать')

            ans2 = input()

            if ans2.lower() != 'y':

                continue

        return ans

def print\_table(data, columns=None, is\_dataframe=False):

    if not is\_dataframe:

        if len(data) == 0:

            print('В таблице нет данных')

            return

        assert len(data[0]) == len(columns)

        data\_pd = pd.DataFrame(data, columns=columns)

    else:

        data\_pd = data

    #Используем repr для отображения кавычек для текстовых элементов

    pd.set\_option('display.max\_rows', None)

    pd.set\_option('display.width', os.get\_terminal\_size()[0])

    pd.set\_option('display.max\_columns', None)

    print(data\_pd.apply(lambda row: [repr(x) for x in row]))

    pd.reset\_option('display.max\_rows')

    pd.reset\_option('display.width')

    pd.reset\_option('display.max\_columns')

def user\_constuct\_simplest\_filter(DataBase, table\_name = None):

        '''Пользовательский ввод простейшего фильтра вида:

        <столбец> <сравнение> <значение/столбец>'''

        if table\_name is None:

            table\_name = DataBase.selected\_table

        column\_names = DataBase.get\_column\_names(table\_name)

        selected\_column = user\_select\_from\_list(column\_names, 'Введите столбец по которому будет проводиться фильтрация. ', compact\_form=True)

        relation = user\_select\_from\_list(['>', '>=', '=', '<=', '<'], 'Введите логический фильтр. ', compact\_form=True)

        while True:

            value = input('Введите значения для сравнения: ')

            if value.isnumeric():

                break

            if value in column\_names:

                break

            if value[0] == value[-1] and (value[0] == '"' or value[0] == "'"):

                break

            print(f'Значение для сравнения должно быть числом, названием одного из столбцов или обернуто в кавычки. Получено {value}')

        return selected\_column + relation + value

def user\_construct\_filter(DataBase, table\_name = None):

    '''Пользовательский ввод фильтра вида:

    <простейший фильтр> <и/или> <простейший фильтр> <и/или> ...'''

    if table\_name is None:

        table\_name = DataBase.selected\_table

    constructing = True

    filter = ''

    while constructing:

        filter += user\_constuct\_simplest\_filter(DataBase, table\_name)

        while True:

            print(f'\nТекущий фильтр: {filter}')

            print('Выберите дейсвие:\n')

            print('1 - Добавление логического оператора (и/или)')

            print('2 - Завершить создание фильтра')

            ans = input()

            if ans not in ['1', '2']:

                print(f'Варианта {ans} среди данных действий нет')

                continue

            if ans == '2':

                constructing = False

                break

            if ans == '1':

                log\_dict = ['or', 'and']

                logical = user\_select\_from\_list(log\_dict, prompt='Введите логический оператор. ', compact\_form=True)

                filter += f' {logical} '

                filter += user\_constuct\_simplest\_filter(DataBase, table\_name)

    return filter

Модуль E4.py

import os

import DataBase as db

import IO\_funcs as io

import E4\_funcs as action\_funcs

def main():

    #Выбор и открытие базы данных

    while True:

        bd\_file = input('Введите название файла с базой данных: ')

        if os.path.isfile(bd\_file):

            break

        print('Файл', bd\_file, 'не найден')

    DataBase = db.DataBase(bd\_file)

    #Получение информации о таблицах

    if not DataBase.tables:

        print('В данном файле не найдено таблиц')

        exit()

    print('-'\*100)

    print('Найденные таблицы в', bd\_file, '\n')

    for table in DataBase.tables:

        print(f'Таблица {table}')

        print('Содержащиеся поля:', str(DataBase.get\_column\_names(table))[1:-1], '\n')

    #Выбор таблицы с которой будет проводиться работа. По умолчанию первая из DataBase.tables

    if len(DataBase.tables) > 1:

        DataBase.select\_table()

    #Основная работа с базой данных

    actions = {

        '1': db.DataBase.show\_table,

        '2': db.DataBase.save\_table,

        '3': db.DataBase.get\_filtered,

        '4': db.DataBase.change\_values,

        '5': db.DataBase.delete\_values,

        '6': db.DataBase.insert\_values,

        '7': db.DataBase.select\_table,

        '8': action\_funcs.task1,

        '9': action\_funcs.task2,

        }

    close\_program = False

    while not close\_program:

        print(f'''Для работы выбрана таблица {DataBase.selected\_table}. Выберите действие:''')

        actions\_descr = {

            '1': 'Отображение содержимого таблицы на экране',

            '2': 'Сохранение содержимого таблицы в файл',

            '3': 'Отображение содержимого таблицы на экране с учетом фильтра',

            '4': 'Изменение значений в базе данных',

            '5': 'Удаление значений из базы данных',

            '6': 'Внесение новой строки в базу данных',

            '8': 'Информация о ВУЗах по их статусу',

            '9': 'Информация о преподавательских составах федеральных университетов',

            'q': 'Выйти из программы'

        }

        if len(DataBase.tables) > 1:

            actions\_descr['7'] = 'Выбор другой таблицы'

        ans = io.user\_select\_from\_list(actions\_descr)

        if ans == 'q':

            break

        if ans in actions.keys():

            actions[ans](DataBase)

            continue

        print('Опции', ans, 'в списке нет. Введите цифру соответствующую действию или напишите q, чтобы выйти')

if \_\_name\_\_ == '\_\_main\_\_':

    main()

Модуль E4\_funcs.py

import DataBase as db

import IO\_funcs as io

import sqlite3

import pandas as pd

def task1(DataBase: db.DataBase):

    #Получение статусов

    con = sqlite3.connect(DataBase.bd\_file)

    cur = con.cursor()

    cur.execute('SELECT DISTINCT status FROM vuzkart')

    status\_list = [ row[0] for row in cur.fetchall() ]

    cur.close()

    con.close()

    #Выбор статуса

    status\_dict = dict(zip(map(str, range(1, len(status\_list) + 1)), status\_list))

    selected\_status = status\_dict[io.user\_select\_from\_list(status\_dict, 'Выберите статус вуза, который вас интересует:')]

    #Получение данных

    con = sqlite3.connect(DataBase.bd\_file)

    cur = con.cursor()

    cur.execute(f'SELECT z1 FROM vuzkart WHERE status="{selected\_status}"')

    data = [ row[0] for row in cur.fetchall() ]

    cur.execute(f'SELECT z1 FROM vuzkart WHERE status="{selected\_status}" AND TRIM(z15) = ""')

    data\_pr\_null = [ row[0] for row in cur.fetchall() ]

    cur.execute(f'SELECT z1 FROM vuzkart WHERE status="{selected\_status}" AND TRIM(z9) = ""')

    data\_tel\_null = [ row[0] for row in cur.fetchall() ]

    cur.execute(f'SELECT z1 FROM vuzkart WHERE status="{selected\_status}" AND TRIM(z9) = "" AND TRIM(z15) = ""')

    data\_pr\_tel\_null = [ row[0] for row in cur.fetchall() ]

    cur.close()

    con.close()

    print(f'Статистика по вузам со статусом {selected\_status}')

    print(f'Всего вузов: {len(data)}')

    print(f'Всего вузов без данных о ректоре: {len(data\_pr\_null)}')

    print(f'Всего вузов без данных о справочном телефоне: {len(data\_tel\_null)}')

    print(f'Всего вузов без данных о ректоре и справочном телефоне: {len(data\_pr\_tel\_null)}')

    print(f'\nВузы без данных о ректоре и справочном телефоне:')

    if data\_pr\_tel\_null:

        print(pd.Series(data\_pr\_tel\_null), '\n')

    else:

        print('Таких вузов не найдено\n')

def task2(DataBase : db.DataBase):

    con = sqlite3.connect(DataBase.bd\_file)

    cur = con.cursor()

    cur.execute('''select vuzstat.codvuz, z2, PPS, DN+KN, ROUND(CAST(DN+KN AS float)/CAST(PPS AS float)\*100, 2)

from vuzstat join vuzkart on vuzstat.codvuz = vuzkart.codvuz

where vuzkart.gr\_ved = "ФУ "; ''')

    data = cur.fetchall()

    cur.close()

    con.close()

    data\_pd = pd.DataFrame(data, columns=['Код вуза', 'Название ВУЗа', 'Кол-во преподавателей', 'Кол-во преподавателей с учёными степенями', 'Процент преподавателей с учёными степенями'])

    summary = data\_pd.sum()

    summary.iloc[0] = None

    summary.iloc[1] = 'Итого'

    summary.iloc[4] = round((summary.iloc[3] / summary.iloc[2])\*100, 2)

    data\_pd.loc[len(data\_pd)] = summary

    print(f'Информация о федеральных университетах. Всего {len(data)} записей:')

    io.print\_table(data\_pd, is\_dataframe=True)

    print()