****

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 3**

Выполнил студент: Михайловский Михаил Юрьевич

Группа: А-03-21

Проверил: Фомин Геннадий Александрович

**Москва 2023**

Оглавление

[Этап 3 3](#_Toc152601351)

[Структура базы данных 3](#_Toc152601352)

[Создание базы данных 3](#_Toc152601353)

[Описание программы 4](#_Toc152601354)

[Модуль Database.py 4](#_Toc152601355)

[Модуль IO\_funcs.py 4](#_Toc152601356)

[Модуль prog.py 5](#_Toc152601357)

[Примеры работы с программой 5](#_Toc152601358)

[Открытие базы данных 5](#_Toc152601359)

[Отображение содержимого таблиц 6](#_Toc152601360)

[Сохрание данных из таблицы в файл 8](#_Toc152601361)

[Внесение новых строк 8](#_Toc152601362)

[Изменение и удаление строк по фильтру 9](#_Toc152601363)

[Приложение 10](#_Toc152601364)

[Модуль DataBase.py 10](#_Toc152601365)

[Модуль IO\_funcs.py 15](#_Toc152601366)

[Модуль prog.py 18](#_Toc152601367)

Этап 3

Структура базы данных

|  |  |  |
| --- | --- | --- |
| **Содержание поля** | **Название столбца** | **Тип информации в поле** |
| Код дисциплины по учебному плану | disz\_code | Текст |
| Название дисциплины | disz\_name | Текст |
| Номер семестра с аттестацией по дисциплине | semester | целое число |
| Тип аттестации (экзамен/зачет) | attestation\_type | Текст |
| Дата аттестации | attestation\_date | Дата |
| ФИО преподавателя, проводившего аттестацию | prepod\_fio | Текст |
| Должность преподавателя | prepod\_dolzh | Текст |
| Полученная оценка | mark | целое число |
| Дата занесения/обновления записи | record\_date | Дата |

Таблица 1. Структура базы данных attestat.sql

Создание базы данных

import sqlite3

>>> con = sqlite3.connect('attestat.sqlite')

>>> cur = con.cursor()

>>> sql = '''

... CREATE TABLE attestat (id INTEGER PRIMARY KEY,

... disz\_code TEXT, disz\_name TEXT, semester INTEGER,

... attestation\_type TEXT, attestation\_date TEXT,

... prepod\_fio TEXT, prepod\_dolzh TEXT, mark INTEGER,

... record\_date TEXT)'''

>>> cur.executescript(sql)

<sqlite3.Cursor object at 0x000001B3FA4801C0>

>>> cur.close()

>>> con.close()

Описание программы

Программа написана на языке программирования python 3.11.5. В ней содержится три модуля: Database.py, IO\_funcs.py, prog.py. Программа рассчитана на работу с произвольными sqlite базами данных (БД). Программа поддерживает работу с БД содержащими одну или более таблиц.

Список используемых модулей: os, pandas, sqlite3.

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.

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) – Печать данных в виде таблицы с указанными столбцами.

data – список со списками, являющимися строками в таблице.

columns – список с названиями столбцов.

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 prog.py

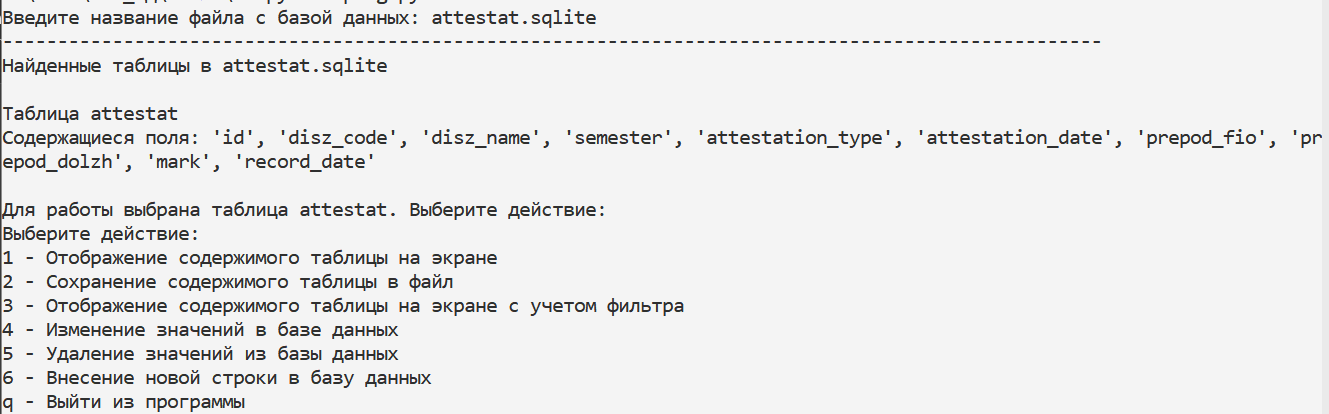
Основной модуль, задающий порядок выполнения программы. Порядок работы:

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.

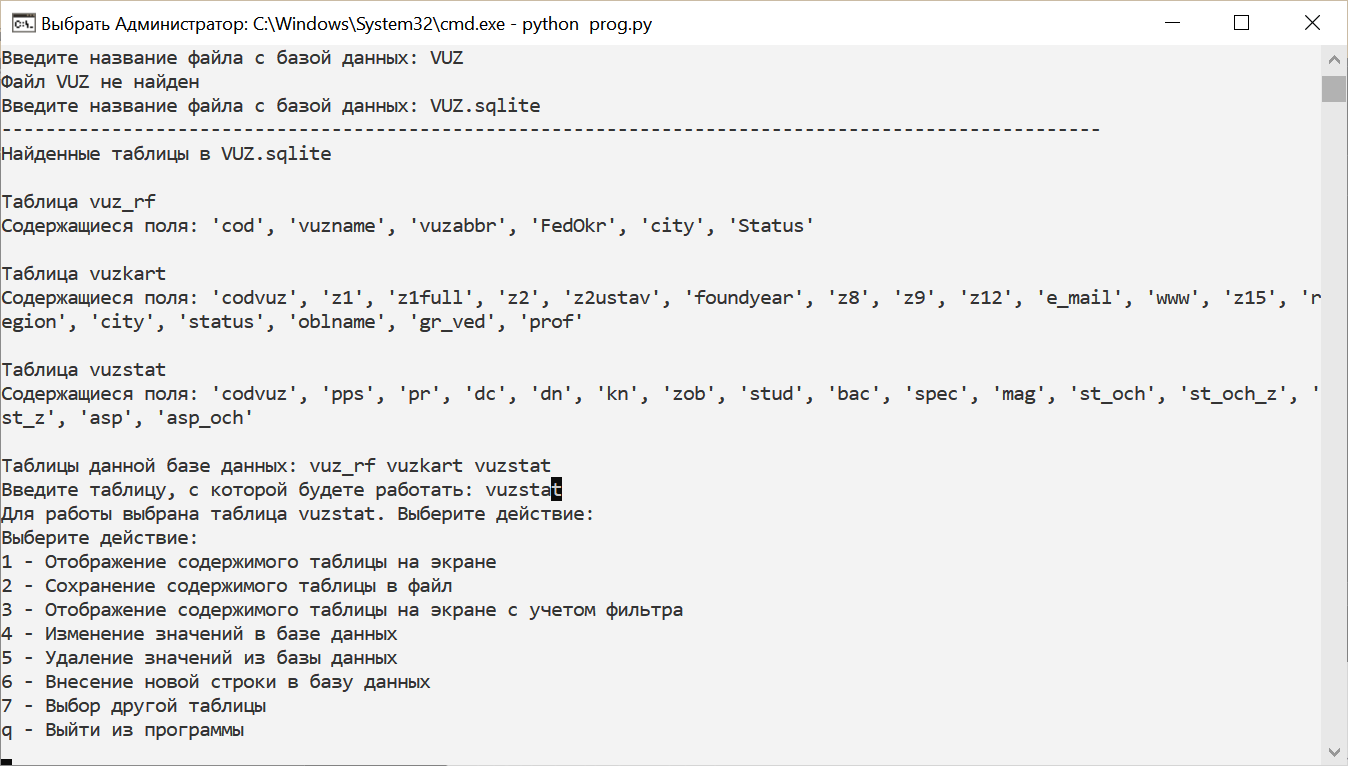
Примеры работы с программой

Открытие базы данных

При запуске программы необходимо ввести название файла с базой данной sqlite. При корректном вводе имени файла база данных открывается и программа проверяет то, какие таблицы в ней содержатся. Когда таблица одна, то она используется для дальнейшей работы (рис. 1). Если их более одной, то пользователь должен выбрать таблицу, с которой будет вести работу (рис. 2).



**Рис. 1. Открытие базы данных с одной таблицей**

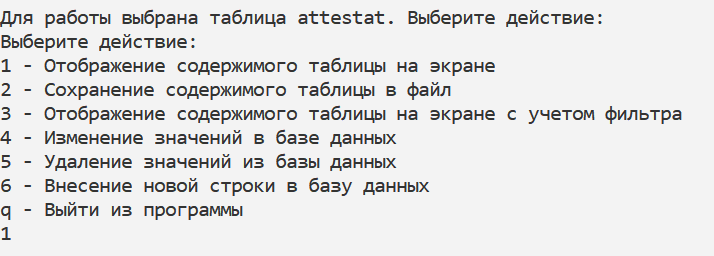


**Рис. 2. Открытие базы данных с одной таблицей**

Если для открытия будет введён несуществующий файл, то программа сообщит об этом пользователю, и попросит ввести существующий файл (рис. 2).

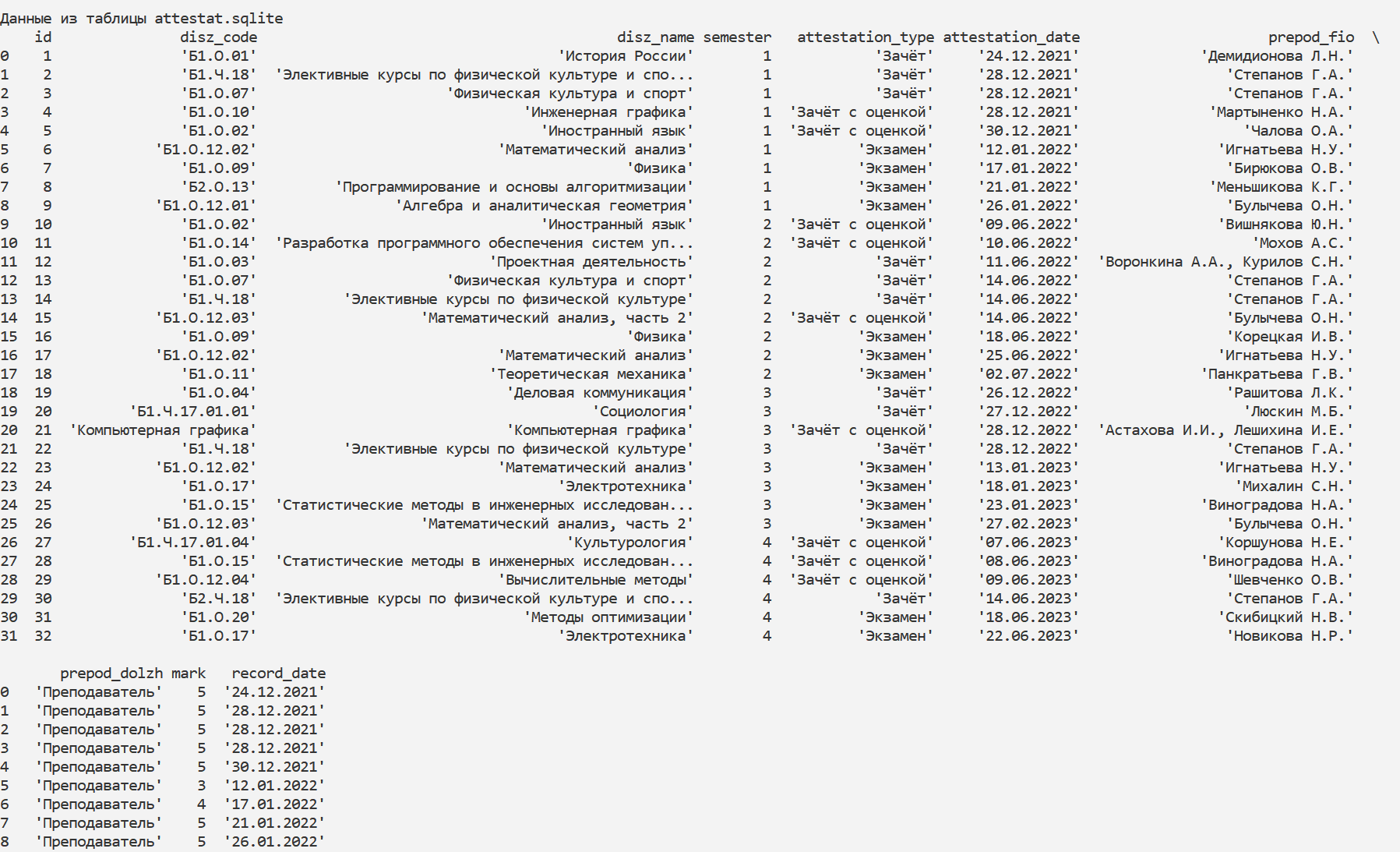
Отображение содержимого таблиц

После открытия базы данных и выбора таблицы можно отобразить её содержимое. Для этого нужно выбрать вариант 1 (рис. 3).



**Рис. 3. Выбор действия отображения содержимого таблиц**

После выбора данного действия на экране будет выведена таблица с данными из БД (рис. 4). В данном случае для поскольку в таблице столбцов много, часть столбцов выводятся отдельно.

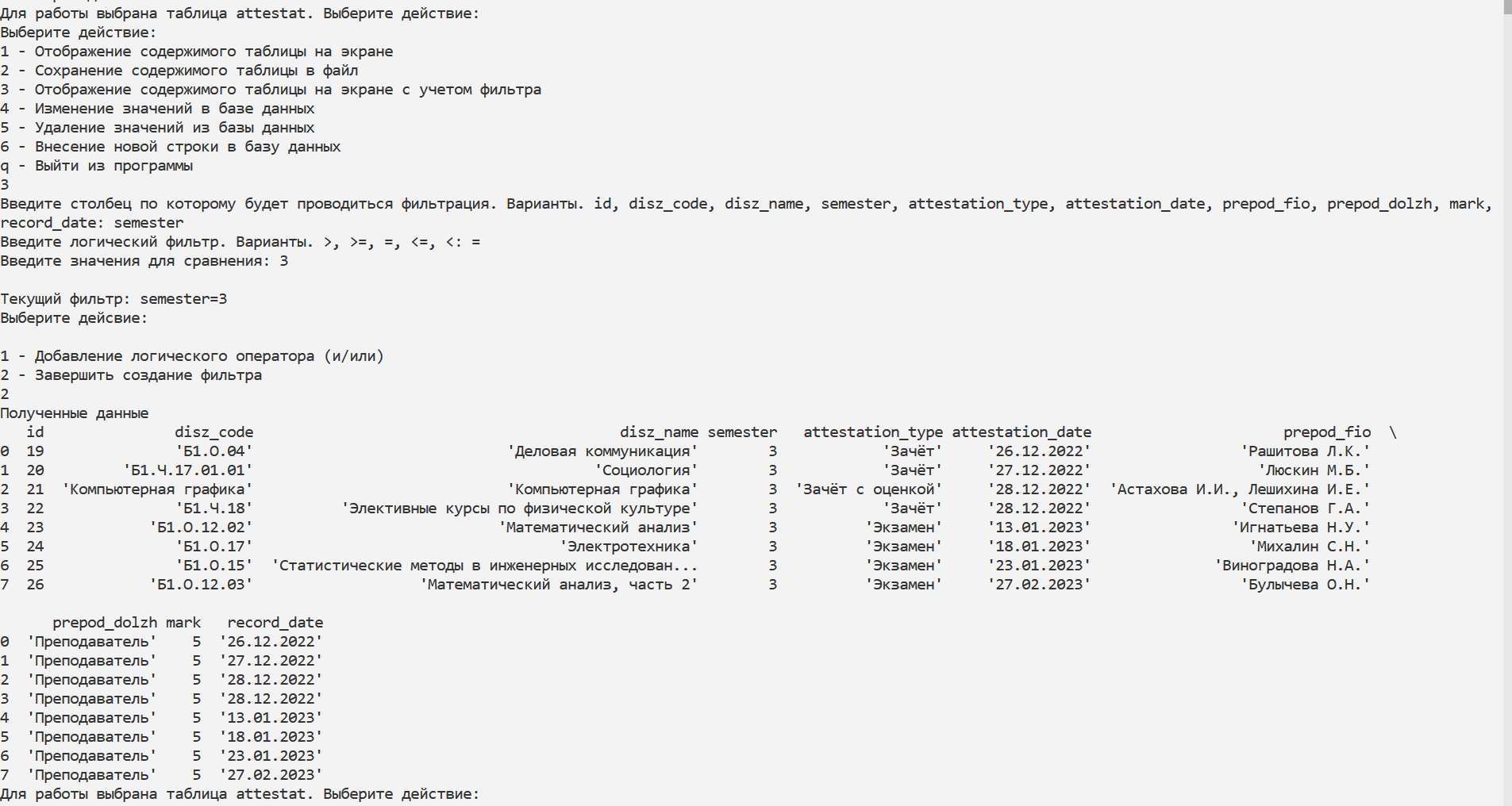


**Рис. 4. Результат отображения содержимого таблицы**

Для получения данных с учётом фильтра нужно использовать действие 3. Процедура создания фильтра следующая:

1. Выбор столбца, по которому будет вестись фильтрация;
2. Выбор логического фильтра (>, >=, =, <=, <);
3. Выбор значения для сравнения. Если это строка, она должны быть обёрнута в кавычки;
4. Составленный фильтр выводится на экран. Его можно дополнить фильтром такого же вида, связав их через логический оператор и/или. Если пользователь выбирает дополнить то переходим на 1 шаг. Иначе завершается создание фильтра.

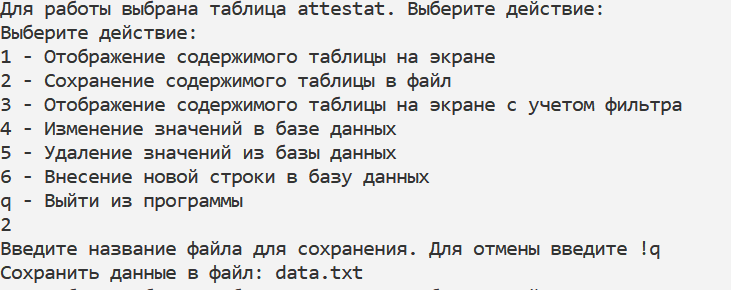
В результате выводятся данные на экран с учётом фильтра



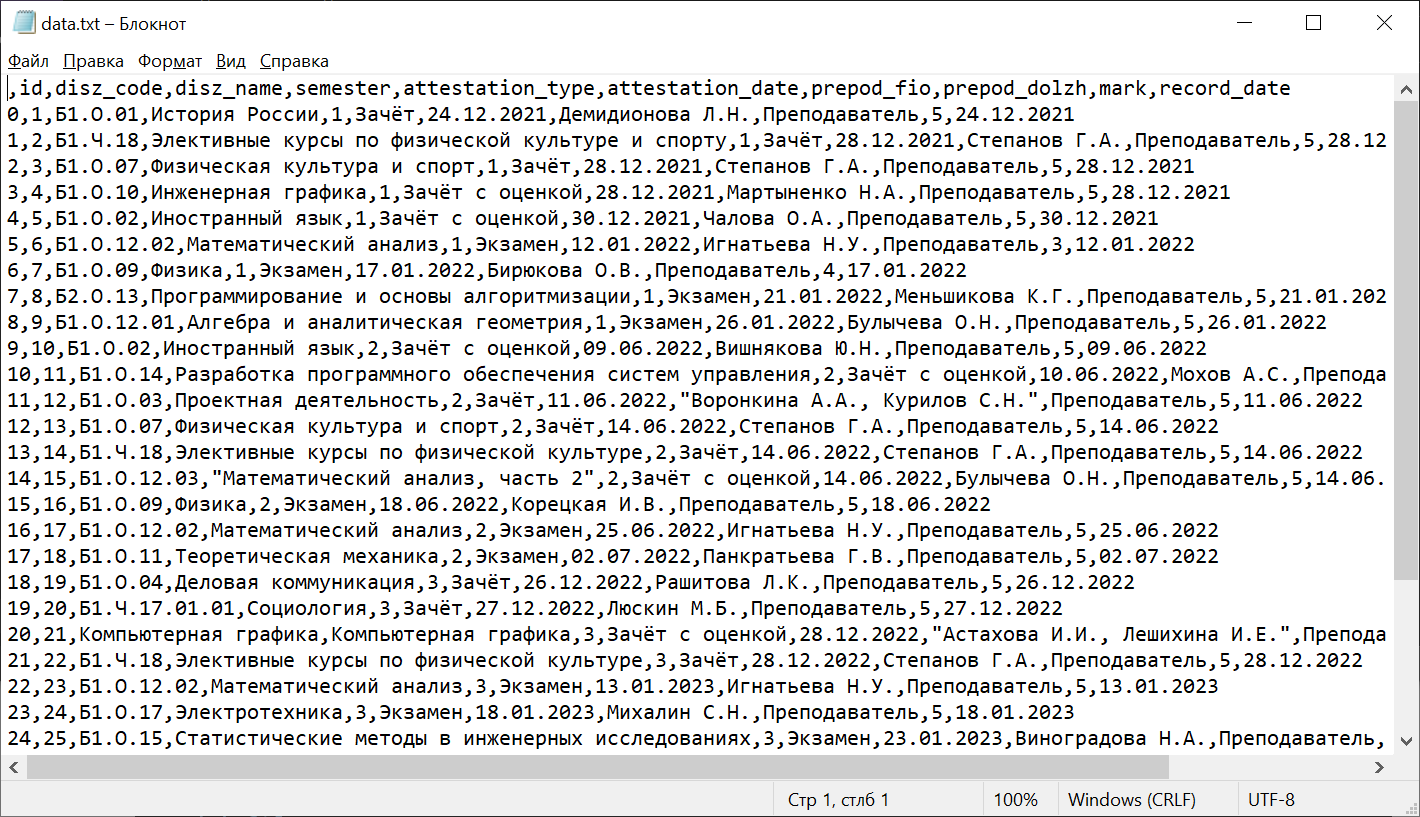
**Рис. 5. Получения данных с учётом фильтра**

Сохрание данных из таблицы в файл

Для сохранения данных в файл нужно выбрать действие под номером 2. Далее выбирается название файла (рис. 6). Если файл существует требуется подтверждение перезаписи данных. После чего данные записываются в выбранный файл (рис. 7).



**Рис. 6. Сохранение данных из таблицы в файл**



**Рис. 7. Результат сохранения данных в файл**

**Внесение новых строк**

Для внесения новых данных в таблицу необходимо выбрать действие под номером 6. Затем вводится количество строк, которое необходимо добавить. После чего по очереди вводятся данные для каждой строки, при чём, текстовые оборачивать в кавычки не нужно (рис. 8).

В нашем случае после добавления проверяем наличие новых добавленных (рис. 9).

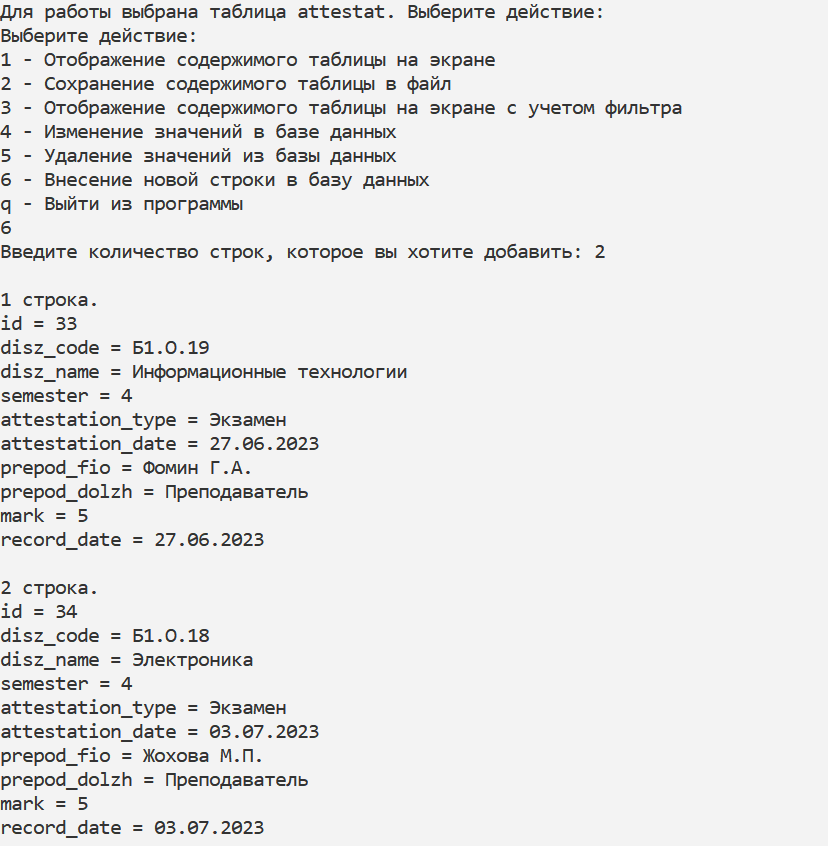
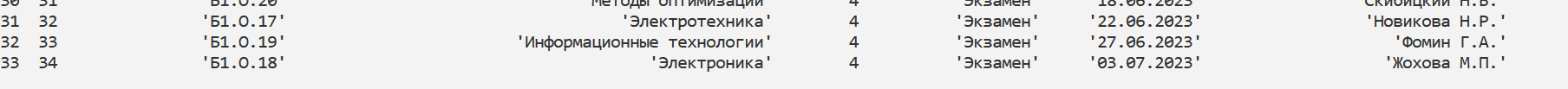


Рис. 8. Внесение данных в БД

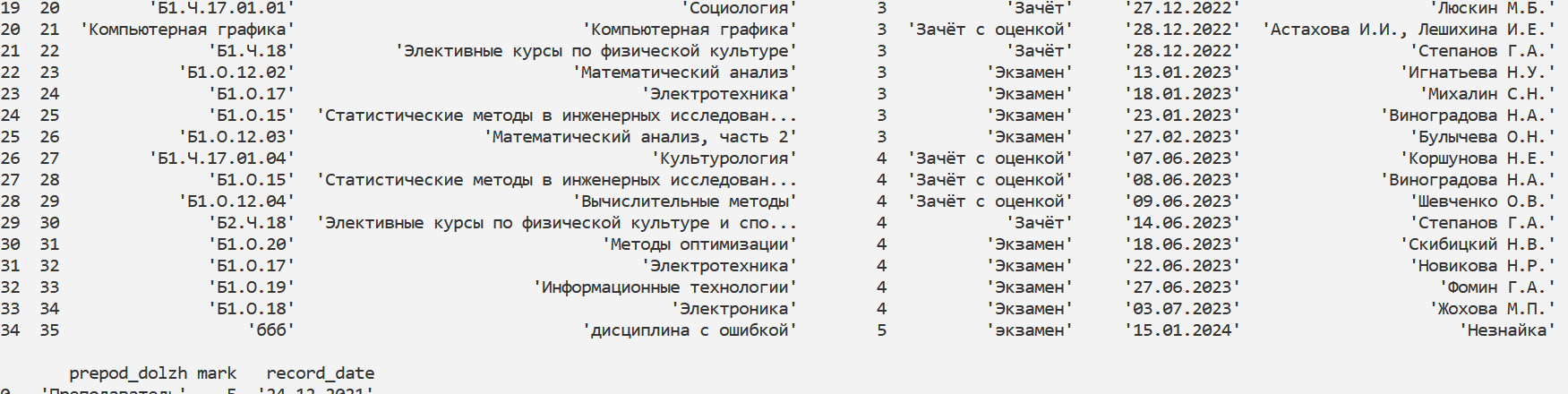


**Рис. 9. Проверка наличия введенных данных в БД**

Изменение и удаление строк по фильтру

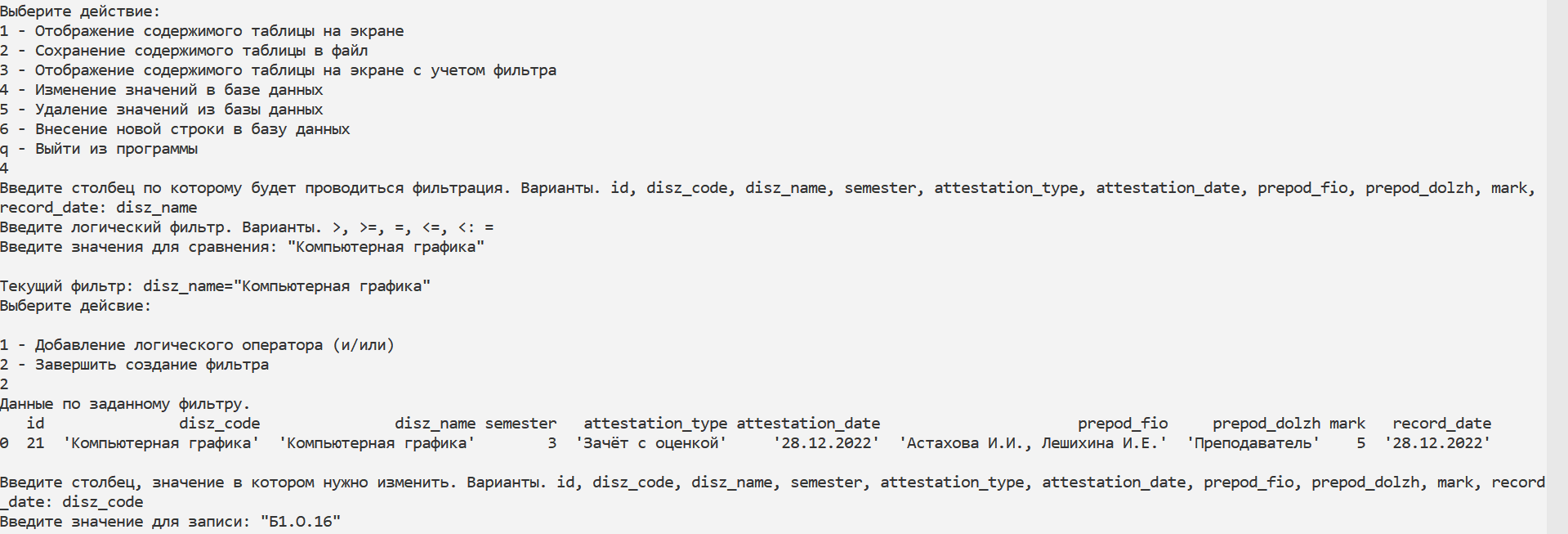
Пусть таблица в БД имеет следующий вид (рис. 10). Мы хотим исправить ошибку во внесенном коде дисциплины компьютерная графика и удалить последнюю строчку.





**Рис. 10. Исходная таблица**

Для изменения данных используется действие под номером 4 (рис. 11). Далее пользователем вводится фильтр для выборки изменяемых строк и строки, соответствующие ему отображаются на экране. Затем выбирается изменяемое поле и вводится новое значение. Если новое значение является текстовым, то оно должно быть обёрнуто в кавычки.

****

**Рис. 11. Изменение строк по фильтру**

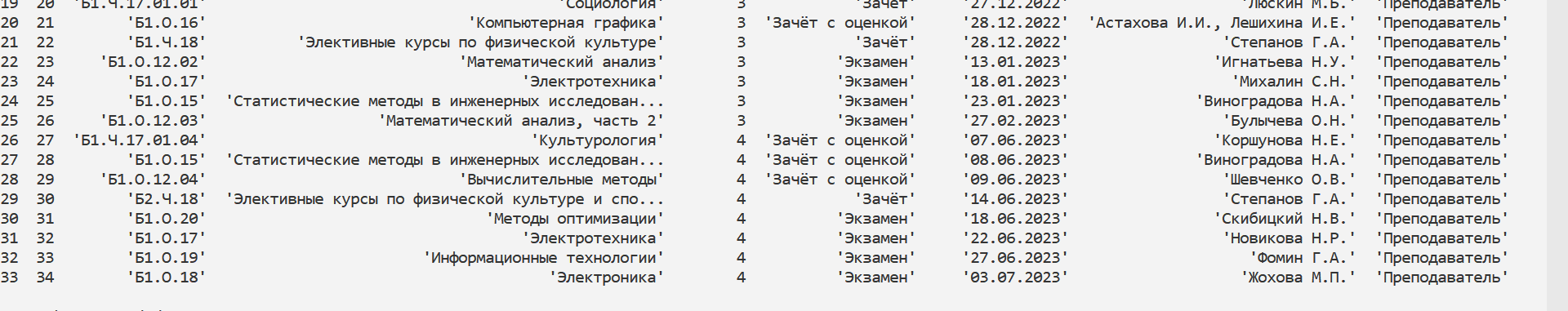
Для удаления строк выбирается действие под номером 5 (рис. 12). Далее вводится фильтр и ему строки отображаются на экране и их количество. Далее требуется подтверждение для удаления.



**Рис. 12. Удаление строк по фильтру**

Проверим теперь содержание таблицы (рис. 13). Как видно, теперь код дисциплины компьютерная графика указан верно, а последняя строка отсутствует.



****

**Рис. 13. Таблица БД после изменения и удаления строк**

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):

    if len(data) == 0:

        print('В таблице нет данных')

        return

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

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

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

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

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

    #Используем repr для отображения кавычек для текстовых элементов

    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 prog.py

#prog.py

import os

import DataBase as db

import IO\_funcs as io

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,

        }

    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',

            '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()