Федеральное государственное бюджетное образовательное учреждение высшего образования «Национальный исследовательский университет «МЭИ»

Институт информационных и вычислительных технологий

Кафедра Управления и интеллектуальных технологий

**Научно-исследовательская работа**

**по теме «Основы работы с базами данных»**

Выполнил студент группы А-03-19

Баснак. Е. А.

Проверил

Фомин Г.А.

Москва 2021 г.

Содержание

[1. Работа с данными SQLite из программ на языке Python 4](#_Toc87758757)

[2. Часть 2 НИР 15](#_Toc87758758)

[2.1. Структура БД 15](#_Toc87758759)

[2.2. Описание алгоритма 17](#_Toc87758760)

[2.3. Результаты тестирования 42](#_Toc87758761)

[3. Часть 3 НИР 44](#_Toc87758762)

# Работа с данными SQLite из программ на языке Python

Средства для работы с БД SQLite содержатся в среде Python сразу после ее инсталляции.

Это надо было проверить!

Доказательством этого служит наличие библиотеки sqlite3.dll, находящейся по месту установки Python в каталоге DLLs:



И наличие подкаталога sqlite в каталоге Lib:



Подготовка к работе:

>>> import os

>>> os.chdir('C:\\PO\_AS\\NIR') ### установка рабочего каталога

>>> import sqlite3 ### импорт модуля sqlite3 для работы с SQLite

**Создание БД с 2 таблицами**

#### Устанавливаем соединение с существующей БД или создаем новую БД

>>> con = sqlite3.connect('bd1.sqlite')

#### Создаем переменную-курсор

>>> cur = con.cursor()

#### Создаем символьную переменную с SQL-командами создания новых

### таблиц в БД с заданными полями

>>> sql = """\

CREATE TABLE stud (nom INTEGER PRIMARY KEY AUTOINCREMENT,

fio TEXT,

email TEXT);

CREATE TABLE sessija (id SECONDARY KEY,

oz\_PO\_AS INTEGER,

oz\_NIR INTEGER,

oz\_TAU INTEGER)

"""

#### Выполняем совокупность SQL-команд (скрипт)

>>> cur.executescript(sql)

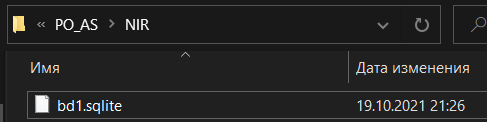
<sqlite3.Cursor object at 0x000001AA9345E5E0>

#### Закрываем курсор и БД

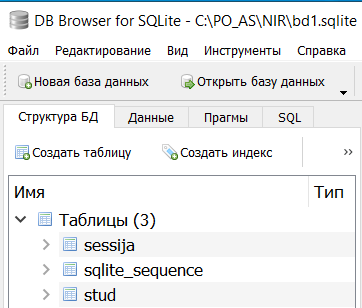
>>> cur.close()

>>> con.close()

В результате, в рабочем каталоге появляется файл bd1.sqlite.

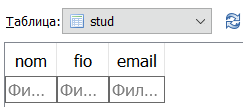
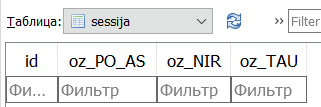


Используя сторонний софт, открываем созданную БД и проверяем содержимое:



На скриншоте видно 3 таблицы: две созданы нами (sessija и stud) и одна внутренняя таблица SQL (sqlite\_sequence), необходимая для реализации **AUTOINCREMENT**.

Если открыть содержимое, можно увидеть, что таблицы пока что пустые:



**Запись данных в таблицы**

>>> con = sqlite3.connect('bd1.sqlite')

>>> cur = con.cursor()

#### Создаем переменную с SQL-командой вставки записи в таблицу stud

>>> sql = """\

INSERT INTO stud (nom, fio, email) VALUES (3, 'СИДОРОВ С.С.', 'sidorovss@mpei.ru')

"""

#### Выполняем отдельную SQL-команду

>>> cur.execute(sql)

<sqlite3.Cursor object at 0x000001AA9345E7A0>

#### Фиксируем выполнение команды

>>> con.commit()

#### Аналогичные операции для вставки еще нескольких записей

>>> sql = """\

INSERT INTO stud (nom, fio, email) VALUES (2, 'ПЕТРОВ П.П.', 'petrovpp@mpei.ru')

"""

>>> cur.execute(sql)

<sqlite3.Cursor object at 0x000001AA9345E7A0>

>>> con.commit()

>>> sql = """\

INSERT INTO stud (nom, fio, email) VALUES (4, 'БАСНАК Е.А.', 'basnakya@mpei.ru')

"""

>>> cur.execute(sql)

<sqlite3.Cursor object at 0x000001AA9345E810>

>>> con.commit()

>>> sql = """\

INSERT INTO stud (nom, fio, email) VALUES (5, 'КОНОНЕНКО А. В.', 'kononenkoav@mpei.ru')

"""

>>> cur.execute(sql)

<sqlite3.Cursor object at 0x000001AA9345E7A0>

>>> con.commit()

>>> sql = """\

INSERT INTO stud (nom, fio, email) VALUES (1, 'КАПКИН Е.В.', 'kapkinyv@mpei.ru')

"""

>>> cur.execute(sql)

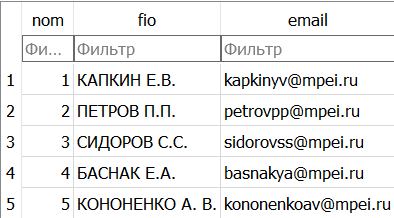
<sqlite3.Cursor object at 0x000001CAA81CE650>

>>> con.commit()

>>> cur.close()

>>> con.close()

После выполнения всех вышеуказанных действий таблица stud в БД bd1.sqlite имеет следующее содержимое:



**Множественная вставка данных в табл. БД**

# Создаем список с элементами-кортежами, содержащими значения полей таблицы

#### Создаем список с 2-мя кортежами – будущими записями таблицы

>>> arr = [(1, 4, 3, 3), (2, 5, 5, 4), (3, 5, 5, 5), (4, 4, 4, 5), (5, 3, 5, 4)]

В списке – студенты с номерами 2,3,4,5, а оценки – для студентов с номерами 1 и 5? Студента с номером 1 вообще нет!

>>> con = sqlite3.connect('bd1.sqlite')

>>> cur = con.cursor()

>>> sql = """\

INSERT INTO sessija (id, oz\_PO\_AS, oz\_NIR, oz\_TAU)

VALUES (?, ?, ?, ?)

"""

#### Выполняем «множественную» команду вставки строк из arr

>>> cur.executemany(sql, arr)

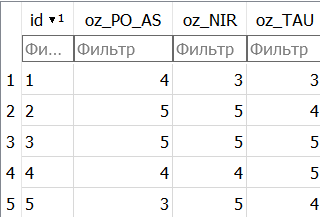
<sqlite3.Cursor object at 0x000001AA9345E810>

>>> con.commit()

>>> cur.close()

>>> con.close()

В результате таблица sessija заполнена следующими значениями:



#### Создаем и выполняем программу test3.py, позволяющую увидеть содержимое указанной таблицы в заданной БД

import os, sqlite3

def select\_cmd(): # просмотр таблицы

sql = 'SELECT \* FROM{}'.format(tblname)

with con:

data = cur.execute(sql).fetchall()

return (data)

dbname = input('Укажите имя файла SQLite:') # ввод имени файла SQLite

if (os.path.isfile(dbname) != True):

print('Нет такого файла!')

else:

tblname = input('Укажите имя таблицы:') # ввод имени таблицы

con = sqlite3.connect(dbname)

cur = con.cursor()

dan = select\_cmd()

nzap = len(dan)

print('Таблица: ', tblname, ' из БД', dbname)

for i in range(nzap):

print(dan[i])

cur.close()

con.close()

#### Запуск на выполнение программы test3:

Укажите имя файла SQLite:bd1.sqlite

Укажите имя таблицы:sessija

Таблица: sessija из БД bd1.sqlite

(1, 4, 3, 3)

(2, 5, 5, 4)

(3, 5, 5, 5)

(4, 4, 4, 5)

(5, 3, 5, 4)

**Чтение данных из таблицы БД**

Пусть имеется БД bd1.sqlite, содержащая таблицу stud. Требуется получить содержимое всех записей этой таблицы.

#### Открыть БД и создать курсор:

>>> con = sqlite3.connect('bd1.sqlite')

>>> cur = con.cursor()

#### Выполнить SQL-запрос для чтения всего содержимого указанной таблицы из БД в курсор:

>>> cur.execute("SELECT \* FROM stud")

<sqlite3.Cursor object at 0x000001FEF38CCB20>

#### Переписать содержимое курсора в список с указанным именем (элементы списка – кортежи со значениями полей в записях):

>>> ar = cur.fetchall()

#### Не забыть закрыть курсор и соединение с БД:

>>> cur.close()

>>> con.close()

>>> print(ar)

[(1, 'КАПКИН Е.В.', 'kapkinyv@mpei.ru'), (2, 'ПЕТРОВ П.П.', 'petrovpp@mpei.ru'), (3, 'СИДОРОВ С.С.', 'sidorovss@mpei.ru'), (4, 'БАСНАК Е.А.', 'basnakya@mpei.ru'), (5, 'КОНОНЕНКО А. В.', 'kononenkoav@mpei.ru')]

**Получить имена полей в таблице**

Пусть имеется БД bd1.sqlite, содержащая таблицу stud. Требуется получить список всех полей таблицы stud.

**Способ 1**

>>> con=sqlite3.connect('bd1.sqlite') ###Открываем БД

>>> def my\_factory(c,r):

### Функция извлечения имен полей

d={}

for i,name in enumerate(c.description):

d[name[0]]=r[i]

d[i]=r[i]

return(d)

>>> con.row\_factory=my\_factory

>>> cur=con.cursor()

>>> cur.execute('SELECT \* FROM stud')

<sqlite3.Cursor object at 0x00000254679BE340>

>>> ar=cur.fetchall()

#### Как только БД нам больше не нужна – закрываем!

>>> cur.close()

>>> con.close()

>>> print(ar)

[{'nom': 1, 0: 1, 'fio': 'КАПКИН Е.В.', 1: 'КАПКИН Е.В.', 'email': 'kapkinyv@mpei.ru', 2: 'kapkinyv@mpei.ru'}, {'nom': 2, 0: 2, 'fio': 'ПЕТРОВ П.П.', 1: 'ПЕТРОВ П.П.', 'email': 'petrovpp@mpei.ru', 2: 'petrovpp@mpei.ru'}, {'nom': 3, 0: 3, 'fio': 'СИДОРОВ С.С.', 1: 'СИДОРОВ С.С.', 'email': 'sidorovss@mpei.ru', 2: 'sidorovss@mpei.ru'}, {'nom': 4, 0: 4, 'fio': 'БАСНАК Е.А.', 1: 'БАСНАК Е.А.', 'email': 'basnakya@mpei.ru', 2: 'basnakya@mpei.ru'}, {'nom': 5, 0: 5, 'fio': 'КОНОНЕНКО А. В.', 1: 'КОНОНЕНКО А. В.', 'email': 'kononenkoav@mpei.ru', 2: 'kononenkoav@mpei.ru'}]

#### Теперь ar содержит словари, в ключах сохраняющие имена полей

>>> sp\_pol = list(ar[0].keys())[0::2] ### Распакуем имена полей в список

>>> print(sp\_pol)

['nom', 'fio', 'email']

**Способ 2**

>>> con=sqlite3.connect("bd1.sqlite")

>>> cur=con.cursor()

#### Формируем строку с SQL-запросом

>>> sql="""\

SELECT \* FROM stud"""

>>> cur.execute(sql) ###Исполнение запроса

<sqlite3.Cursor object at 0x00000254679BE7A0>

>>> pola= cur.description

#### Как только БД нам больше не нужна – закрываем!

>>> cur.close()

>>> con.close()

>>> pola

(('nom', None, None, None, None, None, None), ('fio', None, None, None, None, None, None), ('email', None, None, None, None, None, None))

>>> sp\_pol = [nam[0] for nam in pola] ### Извлекаем имена полей из кортежа

>>> print(sp\_pol)

['nom', 'fio', 'email']

При работе с данными, имеющими смысл даты или времени в программе перед записью в БД или после считывания из нее данных необходимо производить соответствующие преобразования.

Пример ввода с клавиатуры по запросу данных и занесение их в таблицу.

>>> con=sqlite3.connect('bd1.sqlite')

>>> cur=con.cursor()

>>> ar = []

>>> vv = []

>>> vv.append(int(input('Номер студента = ')))

Номер студента = 6

>>> vv.append(input('ФИО студента = '))

ФИО студента = Антимонов М.Е.

>>> vv.append(input('e-mail студента = '))

e-mail студента = antimonovm@mpei.ru

>>> vv

[6, 'Антимонов М.Е.', 'antimonovm@mpei.ru']

>>> ar.append(tuple(vv))

>>> ar

[(6, 'Антимонов М.Е.', 'antimonovm@mpei.ru')]

>>> sql = 'INSERT INTO stud (nom, fio, email) VALUES (?, ?, ?)'

>>> cur.executemany(sql, ar)

<sqlite3.Cursor object at 0x00000254679BE810>

>>> con.commit()

>>> cur.execute('SELECT \* FROM stud')

<sqlite3.Cursor object at 0x00000254679BE810>

>>> ar1 = cur.fetchall()

>>> print(ar1)

[(2, 'ПЕТРОВ П.П.', 'petrovpp@mpei.ru'), (3, 'СИДОРОВ С.С.', 'sidorovss@mpei.ru'), (4, 'БАСНАК Е.А.', 'basnakya@mpei.ru'), (5, 'КОНОНЕНКО А. В.', 'kononenkoav@mpei.ru'), (6, 'Антимонов М.Е.', 'antimonovm@mpei.ru')]

>>> ss=[]

>>> mr = []

>>> mr.append(vv[0])

>>> mr.append(int(input('Оценка студента за ПО АС = ')))

Оценка студента за ПО АС = 5

>>> mr.append(int(input('Оценка студента за НИР = ')))

Оценка студента за НИР = 4

>>> mr.append(int(input('Оценка студента за ТАУ = ')))

Оценка студента за ТАУ = 3

>>> mr

[6, 5, 4, 3]

>>> ss.append(tuple(mr))

>>> ss

[(6, 5, 4, 3)]

>>> sql = 'INSERT INTO sessija (id, oz\_PO\_AS, oz\_NIR, oz\_TAU) VALUES (?, ?, ?, ?)'

>>> cur.executemany(sql, ss)

<sqlite3.Cursor object at 0x000001FEF389E7A0>

>>> con.commit()

>>> cur.execute('SELECT FROM \* sessija')

Traceback (most recent call last):

File "<pyshell#75>", line 1, in <module>

cur.execute('SELECT FROM \* sessija')

sqlite3.OperationalError: near "FROM": syntax error

>>> cur.execute('SELECT \* FROM sessija')

<sqlite3.Cursor object at 0x000001FEF389E7A0>

>>> ar2 = cur.fetchall()

>>> print(ar2)

[(1, 4, 3, 3), (2, 5, 5, 4), (3, 5, 5, 5), (4, 4, 4, 5), (5, 3, 5, 4), (6, 5, 4, 3)]

>>> cur.close()

>>> con.close()

Также для проверки содержимого таблицы sessija запустим программу test3.py:

Укажите имя файла SQLite:bd1.sqlite

Укажите имя таблицы:sessija

Таблица: sessija из БД bd1.sqlite

(1, 4, 3, 3)

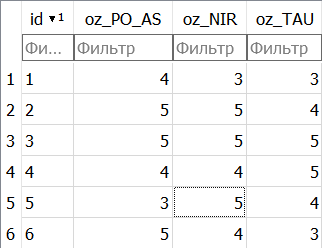
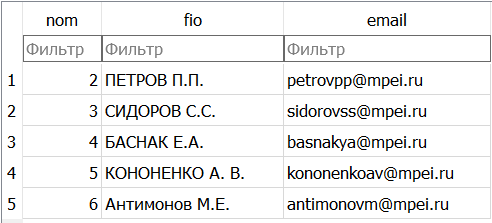
(2, 5, 5, 4)

(3, 5, 5, 5)

(4, 4, 4, 5)

(5, 3, 5, 4)

(6, 5, 4, 3)



Антимонова в список студентов добавили, а его оценки в сессию?

# Часть 2 НИР

## Структура БД

Цель этой части научно-исследовательской работы – получение первичного опыта работы с базой данных с использованием программной среды Python.

В результате выполнения этой части должна появиться БД, включающая одну таблицу со следующей структурой:

|  |  |
| --- | --- |
| **Содержание поля** | **Тип информации в поле** |
| Код дисциплины по учебному плану | текст |
| Название дисциплины | текст |
| Номер семестра с аттестацией по дисциплине | целое число |
| Тип аттестации (экзамен/зачет) | текст |
| Дата аттестации | дата |
| ФИО преподавателя, проводившего аттестацию | текст |
| Должность преподавателя | текст |
| Полученная оценка | целое число |
| Дата занесения/обновления записи | дата |

На примере этой БД необходимо освоить программирование работы с данными в среде Python. При выполнении этой части работы необходимо выполнить следующие операции:

* + 1. Создать базу данных, содержащую таблицу с заданной структурой. Имена БД, таблицы и полей выбрать самостоятельно.
    2. Разработать программу на языке Python, обеспечивающую открытие созданной БД и выполнение следующих функций по выбору пользователя:
* отображение текущего содержимого БД на экране в виде таблицы;
* сохранение таблицы в текстовый файл с задаваемым именем
* выбор пользователем имени одного из полей БД и задание условия по значениям этого (логического выражение). Отображение подмножества строк, удовлетворяющих заданному условию;
* выбор операции с подмножеством строк: удаление из БД, замена значений на заданное;
* добавление строки в БД;
* завершение работы с программой.
  + 1. Используя разработанную программу наполнить БД записями, взяв исходную информацию из своей зачетной книжки.

## Описание алгоритма

Ссылка на github: <https://github.com/exxtie/nir2>

**Разбор отдельных функций программы.**

**Вспомогательные функции:**

* **frontSpaceDel(strk)**

Удаление пробелов в начале строки.

Функция получает на вход строку, перебирает её посимвольно до нахождения первой буквы или символа подчеркивания «\_» или цифры и при этом удаляет все встреченные пробелы.

Возвращаемое значение: строка без пробелов в начале.

* **backSpaceDel(strk)**

Удаление пробелов в конце строки.

Функция получает на вход строку, перебирает её посимвольно, начиная с конца и идя в начало, до нахождения первой буквы или символа подчеркивания «\_» или цифры и при этом удаляет все встреченные пробелы.

Возвращаемое значение: строка без пробелов в конце.

* **fullSpaceDel(strk)**

Удаление пробелов в начале и конце строки.

Функция запускает функции frontSpaceDel и backSpaceDel, которые удаляют пробелы в начале и конце строки.

Возвращаемое значение: строка без пробелов в начале и конце

* **checkField(dbName, str\_field)**

Проверка введенного имени поля на соответствие полям таблицы.

Функция получает на вход имя БД и строку, первым словом которой является имя поля таблицы БД.

Сначала из заданной строки удаляются пробелы в начале и конце с помощью функции fullSpaceDel. Затем идет получение списка имен полей из БД с помощью функции fieldNames. После этого перебираются символы строки до достижения первого не являющегося буквой или символом нижнего подчеркивания «\_». Эти символы записываются в новую строку. В конце проверяется вхождение данной строки в список имен полей.

Возвращаемое значение: True, если заданное поле имеется в таблице, иначе False.

* **showFields(fields)**

Вывод имен указанных полей на экран.

На вход подаётся список имен полей. Функция выводит их на экран в виде шапки таблицы.

* **showData(data)**

Вывод содержимого таблицы на экран.

На вход подаётся список, состоящий из списков-строк таблицы БД. Функция выводит на экран содержимое данного списка в виде таблицы.

* **fieldNames(dbName)**

Получение списка имен полей. На вход подаётся имя БД.

Возвращаемое значение: список имен полей.

* **dbData(dbName)**

Получение списка содержимого таблицы БД. На вход подаётся имя БД.

Возвращаемое значение: список, состоящий из списков-строк таблицы БД.

**Основные функции:**

* **dbCreate(dbName)**

Создание БД.

На вход подаётся имя БД. Функция создаёт БД с одной таблицей TBL\_NAME (строка, имеющая константное значение которой задаётся в самом начале файла программы), содержащей следующие поля:

|  |  |  |
| --- | --- | --- |
| **Содержание поля** | **Имя поля** | **Тип информации в поле** |
| Код дисциплины по учебному плану | code | текст |
| Название дисциплины | subject | текст |
| Номер семестра с аттестацией по дисциплине | sem\_number | целое число |
| Тип аттестации (экзамен/зачет) | type\_of\_cert | текст |
| Дата аттестации | date\_of\_cert | дата |
| ФИО преподавателя, проводившего аттестацию | prof\_fio | текст |
| Должность преподавателя | prof\_pos | текст |
| Полученная оценка | mark | целое число |
| Дата занесения/обновления записи | date\_of\_update | дата |

* **menu()**

Вывод меню и выбор действия.

Функция выводит на экран меню и предлагает пользователю выбрать действие.

* **submenu()**

Вывод второго уровня меню (работа с подмножеством строк) и выбор действия.

Функция выводит на экран меню работы с подмножеством строк и предлагает пользователю выбрать действие.

* **tableDisplay(dbName)**

Вывод таблицы БД на экран.

На вход подаётся имя БД. Функция использует вспомогательные функции fieldNames, dbData, showFields и showData для вывода таблицы БД на экран.

* **tableFile(dbName)**

Сохранение таблицы БД в файл.

На вход подаётся имя БД. Функция перенаправляет поток вывода с экрана на файл, после чего используется функция tableDisplay для вывода таблицы. После этого поток вывода восстанавливается.

* **addRow(dbName)**

Добавление строки в БД.

На вход подаётся имя БД. Пользователь вводит 8 значений, соответствующих определённым полям записи таблицы. В качестве значения для поля «дата занесения/обновления записи» берётся дата из текущего времени в ОС. Все полученные значения заносятся в кортеж и записываются в таблицу БД.

* **condEnter(dbName)**

Ввод условия отбора подмножества.

На вход подаётся имя БД.

Возвращаемое значение: строка, содержащая условие отбора подмножества таблицы БД, которую ввёл пользователь.

* **showSubTable(dbName, cond)**

Отображение подмножества строк, удовлетворяющих заданному условию.

На вход подаётся имя БД и условие. Функция получает записи таблицы, удовлетворяющие заданному условию, после чего выводит их на экран в виде таблицы с помощью функций fieldNames, showFields и showData. Если в таблице нет записей, удовлетворяющих заданному условию, на экран будет выведено сообщение «В таблице нет строк, удовлетворяющих заданному условию. Попробуйте еще раз.» после чего пользователь будет возвращен в главное меню.

Если условие – пустая строка, на экран будет выведено сообщение «Условие отсутствует! Введите условие и повторите попытку.» после чего пользователь будет возвращен в главное меню.

* **updateValues(dbName, cond)**

Замена значений подмножества.

На вход подаётся имя БД и условие для отбора подмножества строк таблицы. Пользователю предлагается ввести имя поля и новое значение для данного поля, после чего это значение будет подставлено в указанное поле всех записей, удовлетворяющих заданному условию. После этого на экран будет выведено подмножество строк, удовлетворяющих условию с помощью функции showSubTable.

Если условие – пустая строка, на экран будет выведено сообщение «Условие отсутствует! Введите условие и повторите попытку.» после чего пользователь будет возвращен в главное меню.

* **delRows(dbName, cond)**

Удаление подмножества строк из БД.

На вход подаётся имя БД и условие для отбора подмножества строк таблицы. Функция удаляет из таблицы все записи, удовлетворяющие заданному условию.

Если условие – пустая строка, на экран будет выведено сообщение «Условие отсутствует! Введите условие и повторите попытку.» после чего пользователь будет возвращен в главное меню.

**Основная часть программы.**

Первым делом на экран выводится приветствие и информация о БД, с которой предстоит работать.

Пользователю предлагается создать новую БД или использовать уже имеющуюся.

После этого на экран выводится меню и программа ожидает выбора пользователя. Меню реализовано в виде словаря с ключами – целыми значениями номерами действий, и значениями – строками, содержащими инструкции, которые надо выполнить при выборе данного действия в меню.

Далее запускается бесконечный цикл, в котором находится условная конструкция, отвечающая за работу меню. Если в меню был выбран 0, происходит выход из цикла и программа завершает работу. Если пользователь выбрал действие в пределах 1-3, вызывается функция exec, которая выполняет инструкции из словаря-меню. Если в меню был выбран вариант 4, открывается меню второго уровня для работы с подмножеством строк, которое работает по такому же принципу, как и главное меню.

**Код программы:**

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# ФГБОУ ВО "НИУ "МЭИ"" \*

# Институт информационных и вычислительных технологий \*

# Кафедра Управления и интеллектуальных технологий \*

# \*

# Научно-исследовательская работа по теме \*

# "Основы работы с базами данных" \*

# \*

# Выполнил: студент группы А-03-19 \*

# Баснак Е.А. \*

# Проверил: Фомин Г.А. \*

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

import sqlite3

import sys

import time

import os

os.chdir('C:\\PO\_AS\\NIR\\part2')

# Название таблицы

tblname = 'stud'

# Вспомогательные функции

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

def checkField(dbName, str\_field):

"""Проверка введенного имени поля на соответствие полям таблицы

Входные аргументы: имя БД; имя поля для проверки

Возвращаемый результат функции: значение True, если указанное поле имеется в таблице иначе значение False"""

# Удаление начальных пробелов (подготовка к проверке на соответствие поля)

str\_field = fullSpaceDel(str\_field)

# Получение имен полей таблицы

right\_fields = fieldNames(dbName)

# Выделение имени поля из условия

field = ''

for ch in str\_field:

if (ch.isalpha() or ch == '\_'):

field = field + ch

else:

break

return field in right\_fields #True если принадлежит

def frontSpaceDel(strk):

"""Удаление пробелов в начале строки

Входной аргумент: строка, из которой надо удалить начальные пробелы

Возвращаемый результат функции: исходная строка без начальных пробелов"""

# Удаление начальных пробелов

for ch in strk:

if (ch == ' '):

strk = strk[1:]

if (ch.isalpha() or ch == '\_'):

break

return strk

def backSpaceDel(strk):

"""Удаление пробелов в конце строки

Входной аргумент: строка, из которой надо удалить конечные пробелы

Возвращаемый результат функции: исходная строка без конечных пробелов"""

# Удаление заключительных пробелов

for ch in reversed(strk):

if (ch == ' '):

strk = strk[:-1]

if (ch.isalpha() or ch == '\_'):

break

return strk

def fullSpaceDel(strk):

"""Удаление пробелов в начале и конце строки

Входной аргумент: строка, из которой надо удалить начальные и конечные пробелы

Возвращаемый результат функции: исходная строка без начальных и конечных пробелов"""

strk = frontSpaceDel(strk)

strk = backSpaceDel(strk)

return strk

def showFields(fields):

"""Вывод имен указанных полей на экран

Входной аргумент: список имен полей для вывода

В результате работы функции на экран выводится заголовок таблицы БД"""

print('{:^10} | {:^30} | {:^10} | {:^12} | {:^12} | {:^30} | {:^15} | {:^4} | {:^15}'.format(fields[0], fields[1], fields[2], fields[3], fields[4], fields[5], fields[6], fields[7], fields[8]))

for i in range(23): # В сумме 161

print('-------', end = '')

print('')

def showData(data):

"""Вывод содержимого таблицы БД на экран

Входной аргумент: список, состоящий из списков - строк таблицы БД

В результате работы функции на экран выводится содержимое таблицы БД"""

for di in data:

print('{:^10} | {:^30} | {:^10} | {:^12} | {:^12} | {:^30} | {:^15} | {:^4} | {:^15}'.format(di[0], di[1], di[2], di[3], di[4], di[5], di[6], di[7], di[8]))

print('')

def fieldNames(dbName):

"""Получение списка имен полей

Входной аргумент: имя БД

Возвращаемый результат функции: список, содержащий имена полей указанной БД"""

# Открытие БД

con = sqlite3.connect(dbName)

# Создание курсора

cur = con.cursor()

# Формирование строки с SQL-запросомом

sql = """\

SELECT \* FROM {}""".format(tblname)

# Исполнение запроса

cur.execute(sql)

# Запись имен полей

fields = cur.description

# Закрытие БД

cur.close()

con.close()

# Извлечение имен полей из кортежа

return [nam[0] for nam in fields]

def dbData(dbName):

"""Получение списка содержимого таблицы БД

Входной аргумент: имя БД

Возвращаемый результат функции: список, содержащий списки-строки таблицы БД"""

# Открытие БД

con = sqlite3.connect(dbName)

#Создание курсора

cur = con.cursor()

# Получание содержимого таблицы

sql = 'SELECT \* FROM {}'.format(tblname)

data = cur.execute(sql).fetchall()

# Закрытие БД

cur.close()

con.close()

return data

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# Конец вспомогательных функций

# Основные функции

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

### Создание БД

def dbCreate(dbName):

"""Создание БД

Входной аргумент: имя БД

В результате работы функции в рабочем каталоге создаётся файл БД с указанным именем"""

# Создание файла БД

con = sqlite3.connect(dbName)

# Создание курсора

cur = con.cursor()

# Формирование SQL-запроса

sql = """\

CREATE TABLE stud (code TEXT,

subject TEXT,

sem\_number INTEGER,

type\_of\_cert TEXT,

date\_of\_cert DATE,

prof\_fio TEXT,

prof\_pos TEXT,

mark INTEGER,

date\_of\_update DATE);

"""

# Выполнение запроса

cur.executescript(sql)

# Закрытие БД

cur.close()

con.close()

def menu():

"""Вывод меню и выбор действия

Возвращаемый результат функции: выбранное пользователем действие"""

print('')

print('Меню:')

print('1) Отображение текущего содержимого БД на экране в виде таблицы.')

print('2) Сохранение таблицы в текстовый файл с задаваемым именем.')

print('3) Добавление строки в БД.')

print('4) Работа с подмножеством строк.')

print('0) Завершение работы с программой.')

res = input('Выберите действие: ')

# Удаление пробелов в начале и конце строки

res = fullSpaceDel(res)

print('')

return res

def subMenu():

"""Вывод второго уровня меню и выбор действия

Возвращаемый результат функции: выбранное пользователем действие в данном меню"""

print('')

print(Меню работы с подмножеством строк:)

print('1) Ввод условия')

print('2) Просмотр подмножества')

print('3) Замена значений подмножества')

print('4) Удаление подмножества')

print('0) Вернуться в главное меню')

res = input('Выберите действие: ')

# Удаление пробелов в начале и конце строки

res = fullSpaceDel(res)

print('')

return res

def tableDisplay(dbName):

"""Вывод таблицы БД

Входной аргумент: имя БД

В результате работы функции на экран выводится таблица БД"""

fields = fieldNames(dbName) # получение имен полей

data = dbData(dbName) # получение содержимого таблицы

showFields(fields) # вывод полей

showData(data) # вывод содержимого

def tableFile(dbName):

"""Сохранение таблицы БД в файл

Входной аргумент: имя БД

В результате работы функции в рабочем каталоге создаётся или изменяется файл с указанным именем, в который записывается таблица БД"""

# Ввод имени файла

fileName = input('Введите имя файла для вывода:')

# Запись текущего потока вывода

vr\_out = sys.stdout

# Открытие файла вывода

fc = open(fileName, 'w')

# Перенацеливание стандартного потока вывода на файл

sys.stdout = fc

# Вывод таблицы

tableDisplay(dbName)

# Восстановление текущего потока

sys.stdout = vr\_out

# Закрытие файла

fc.close()

print('Таблица записана в файл.')

def addRow(dbName):

"""Добавление строки в БД

Входной аргумент: имя БД

В результате работы функции в таблицу указнной БД добавляется новая строка"""

# Ввод необходимых полей

code = input('Введите код дисциплины по учебному плану: ')

code = fullSpaceDel(code)

subject = input('Введите название дисциплины: ')

subject = fullSpaceDel(subject)

sem\_number = int(input('Введите номер семестра с аттестацией по дисциплине: '))

type\_of\_cert = input('Введите тип аттестации (экзамен/зачет): ')

type\_of\_cert = fullSpaceDel(type\_of\_cert)

date\_of\_cert = input('Введите дату аттестации (в формате ДД-ММ-ГГГГ): ')

date\_of\_cert = fullSpaceDel(date\_of\_cert)

prof\_fio = input('Введите ФИО преподавателя, проводившего аттестацию: ')

prof\_fio = fullSpaceDel(prof\_fio)

prof\_pos = input('Введите должность преподавателя: ')

prof\_pos = fullSpaceDel(prof\_pos)

while True:

mark = int(input('Введите полученную оценку: '))

if (mark < 0 or mark > 5):

print('Неверная оценка! Оценка должна находиться в диапазоне от 0 до 5!')

else:

break

# Получение даты занесения записи

dat = time.localtime()

date\_of\_update = '{:0>2}-{:0>2}-{:0>4}'.format(str(dat.tm\_mday), str(dat.tm\_mon), str(dat.tm\_year))

# Создание кортежа с данными для занесения в таблицу

new\_data = (code, subject, sem\_number, type\_of\_cert, date\_of\_cert, prof\_fio, prof\_pos, mark, date\_of\_update)

# Открытие БД

con = sqlite3.connect(dbName)

# Создание курсора

cur = con.cursor()

# Формирование строки с SQL-запросом

sql = """\

INSERT INTO {}

(code, subject, sem\_number, type\_of\_cert, date\_of\_cert, prof\_fio, prof\_pos, mark, date\_of\_update)

VALUES (?,?,?,?,?,?,?,?,?)

""".format(tblname)

# Занесение полученного кортежа в таблицу

cur.execute(sql, new\_data)

# Сохранение изменений

con.commit()

# Закрытие БД

cur.close()

con.close()

def condEnter(dbName):

"""Ввод условия отбора подмножества

Входной аргумент: имя БД

Возвращаемый результат функции: строка, содержащая условие отбора подмножества таблицы БД"""

while True:

# Ввод условия

res = input('Введите условие для отбора подмножества: ')

# Проверка имени поля из условия

if checkField(dbName, res):

break

else:

print('Несуществующее поле БД! Попробуйте еще раз.')

print('Условие сохранено')

return res

def showSubTable(dbName, cond):

""" Отображение подмножества строк, удовлетворяющих заданному условию

Входные аргументы: имя БД; условие для отбора подмножества строк таблицы

В результате работы функции на экран будет выведена шапка таблицы и подмножество строк, удовлетворяющих заданному условию"""

# Проверка наличия условия

if (cond == ''):

print('Условие отсутствует! Введите условие и повторите попытку.')

return

# Открытие БД

con = sqlite3.connect(dbName)

cur = con.cursor()

# Создание SQL-запроса

sql = 'SELECT \* FROM {0} WHERE {1}'.format(tblname, cond)

# Получение данных из таблицы

data = cur.execute(sql).fetchall()

# Закрытие БД

cur.close()

con.close()

# Проверка наличия данных по указанному условию и вывод таблицы, если данные не пусты

if (data):

print('')

# Получение имен полей таблицы

fields = fieldNames(dbName)

showFields(fields)

showData(data)

else:

print('В таблице нет строк, удовлетворяющих заданному условию.')

print('Попробуйте еще раз.')

def updateValues(dbName, cond):

"""Замена значений подмножества

Входные аргументы: имя БД; условие для отбора подмножества строк таблицы

В результате работы функции в таблице указанной БД обновится значение определенного поля всех строк, удовлетворяющих заданному условию"""

# Проверка наличия условия

if (cond == ''):

print('Условие отсутствует! Введите условие и повторите попытку.')

return

while True:

new\_val = input('Введите имя поля и новое значение (<имя поля> = <новое значение>): ')

# Проверка имени поля из условия

if checkField(dbName, new\_val):

break

else:

print('Несуществующее поле БД! Попробуйте еще раз.')

# Открытие БД

con = sqlite3.connect(dbName)

cur = con.cursor()

# Создание SQL-запроса

sql = 'UPDATE {0} SET {1} WHERE {2}'.format(tblname, new\_val, cond)

cur.execute(sql)

# Получение даты обновления записи

dat = time.localtime()

date\_of\_update = '{:0>2}-{:0>2}-{:0>4}'.format(str(dat.tm\_mday), str(dat.tm\_mon), str(dat.tm\_year))

# Создание SQL-запроса

sql = 'UPDATE {0} SET date\_of\_update="{1}" WHERE {2}'.format(tblname, date\_of\_update, cond)

cur.execute(sql)

# Сохранение изменений

con.commit()

# Сохранение изменений

cur.close()

con.close()

print('Значения подмножества изменены')

tableDisplay(dbName)

def delRows(dbName, cond):

"""Удаление подмножества строк из БД

Входные аргументы: имя БД; условие для отбора подмножества строк таблицы

В результате работы функции из таблицы указанной БД будут удалены строки, удовлетворяющие заданному условию"""

# Проверка наличия условия

if (cond == ''):

print('Условие отсутствует! Введите условие и повторите попытку.')

return

# Открытие БД

con = sqlite3.connect(dbName)

cur = con.cursor()

# Создание SQL-запроса

sql = 'DELETE FROM {0} WHERE {1}'.format(tblname, cond)

cur.execute(sql)

# Сохранение изменений

con.commit()

# Закрытие БД

cur.close()

con.close()

print('Подмножество удалено')

tableDisplay(dbName)

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# Конец основных функций

# Основная часть программы

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

print('Здравствуйте! Данная программа позволяет работать с БД, включающей одну таблицу (stud) со следующими полями:')

print('Код дисциплины по учебному предмету - code;')

print('Название дисциплины - subject;')

print('Номер семестра с аттестацией по дисциплине - sem\_number;')

print('Тип аттестации (экзамен/зачет) - type\_of\_cert;')

print('Дата аттестации - date\_of\_cert;')

print('ФИО преподавателя, проводившего аттестацию - prof\_fio;')

print('Должность преподавателя - prof\_pos;')

print('Полученная оценка - mark;')

print('Дата занесения/обновления записи - date\_of\_update.')

print('')

# Задание имени БД

while True:

dbcrt = input('Хотите создать новую БД? (Y - да; N - нет): ')

# Удаление пробелов в начале и конце строки

dbcrt = fullSpaceDel(dbcrt)

if (dbcrt.upper() == 'Y' or dbcrt.upper() == 'N'):

break

elif (dbcrt == 'exit'):

print('Благодарим за использование данной программы. До свидания!')

sys.exit()

else:

print('Неверный ответ! Попробуйте еще раз.')

print('Для завершения работы программы введите "exit".')

if (dbcrt.upper() == 'Y'):

my\_dbName = input('Введите имя новой БД: ')

# Удаление пробелов в начале и конце строки

my\_dbName = fullSpaceDel(my\_dbName)

dbCreate(my\_dbName)

print('БД "{}" создана.'.format(my\_dbName))

else:

while True:

my\_dbName = input('Введите имя существующей БД: ')

# Удаление пробелов в начале и конце строки

my\_dbName = fullSpaceDel(my\_dbName)

if (os.path.isfile(my\_dbName)):

break

elif (my\_dbName == 'exit'):

print('Благодарим за использование данной программы. До свидания!')

sys.exit()

else:

print('Неверный файл! Попробуйте еще раз')

print('Для завершения работы программы введите "exit".')

# Словарь с элементами меню

menu\_dict = {

1:'tableDisplay(my\_dbName)',

2:'tableFile(my\_dbName)',

3:'addRow(my\_dbName)',

}

# Условие отбора подмножества

cond = ''

# Словарь с элементами меню второго уровня

subMenu\_dict = {

1:'cond = condEnter(my\_dbName)',

2:'showSubTable(my\_dbName, cond)',

3:'updateValues(my\_dbName, cond)',

4:'delRows(my\_dbName, cond)'

}

# Бесконечный цикл для работы программы (выход из цикла происходит при вводе 0 в меню)

while True:

choice = menu()

if (choice == '0'):

break

elif (choice > '0' and choice < '4'):

exec(menu\_dict[int(choice)])

elif (choice == '4'):

sub\_choice = subMenu()

if (sub\_choice == '0'):

continue

if (sub\_choice < '0' or sub\_choice > '4'):

print('Такого варианта нет! Возврат в главное меню! Попробуйте еще раз.')

continue

exec(subMenu\_dict[int(sub\_choice)])

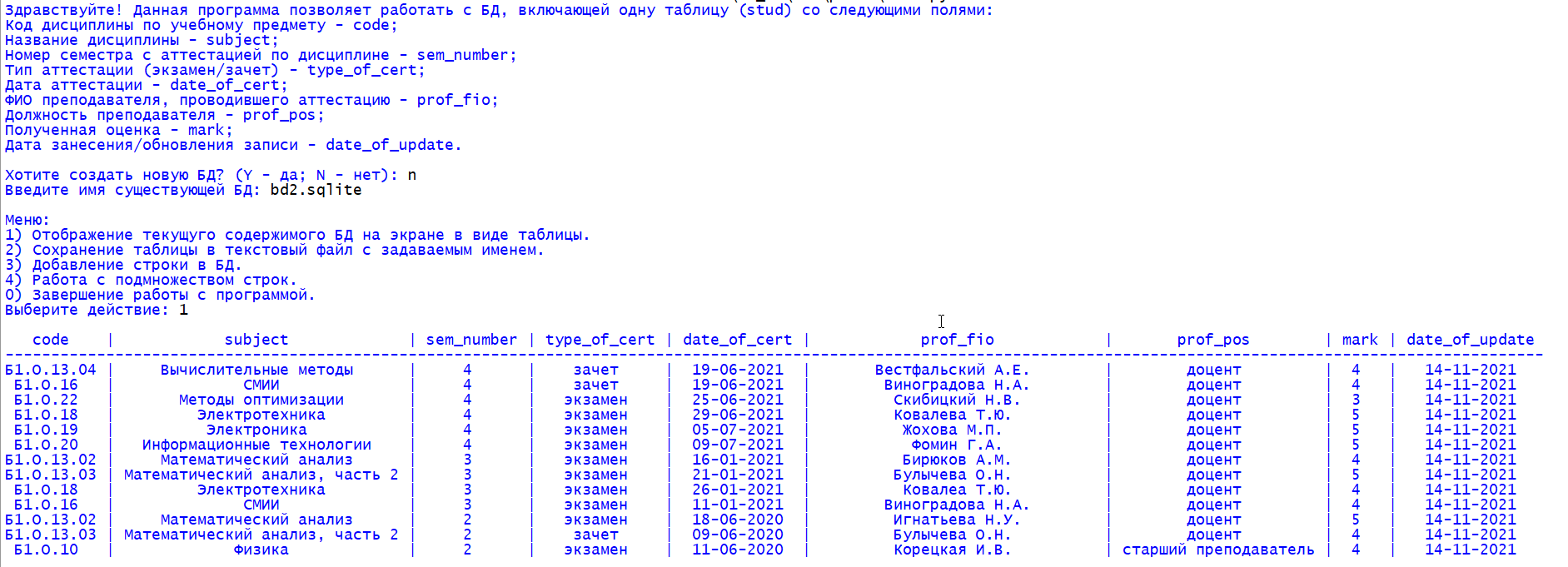
else:

print('Такого варианта нет! Попробуйте еще раз.')

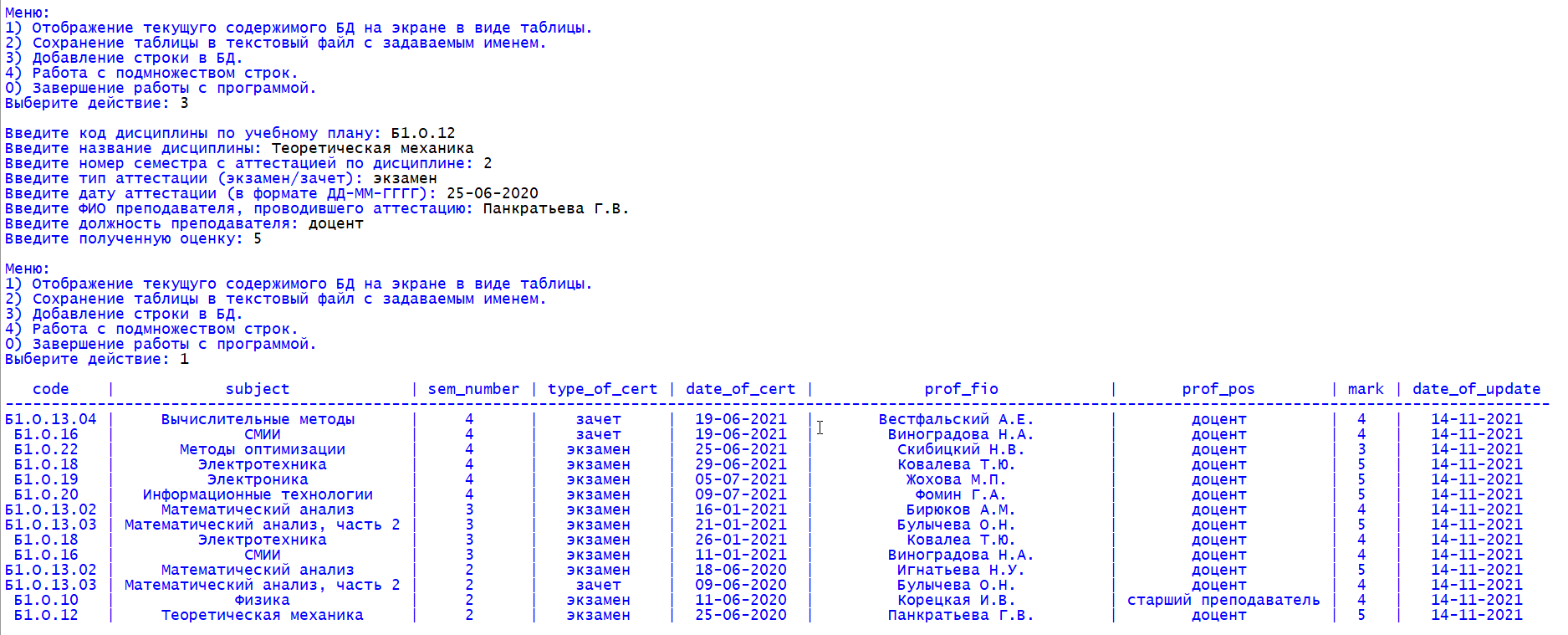
print('Благодарим за использование данной программы. До свидания!')

## Результаты тестирования

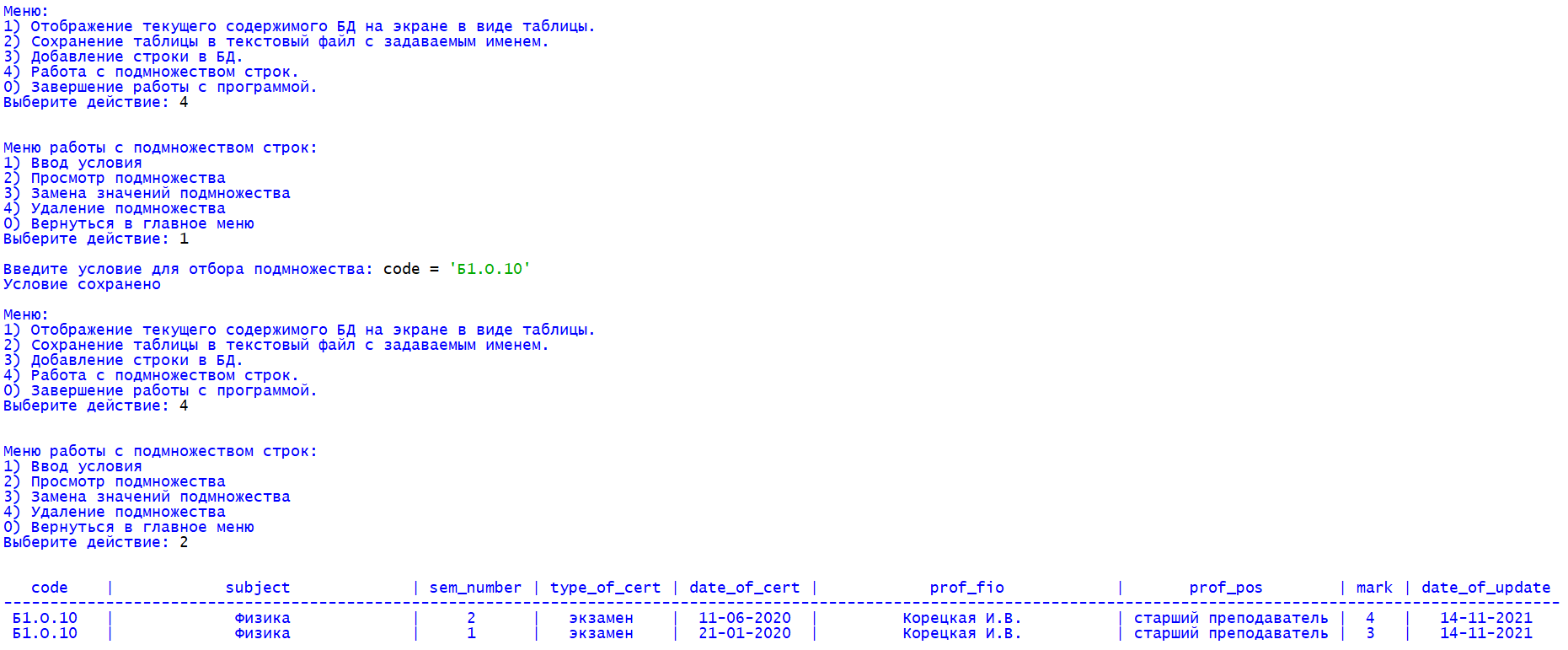
Запуск программы и вывод таблицы на экран.



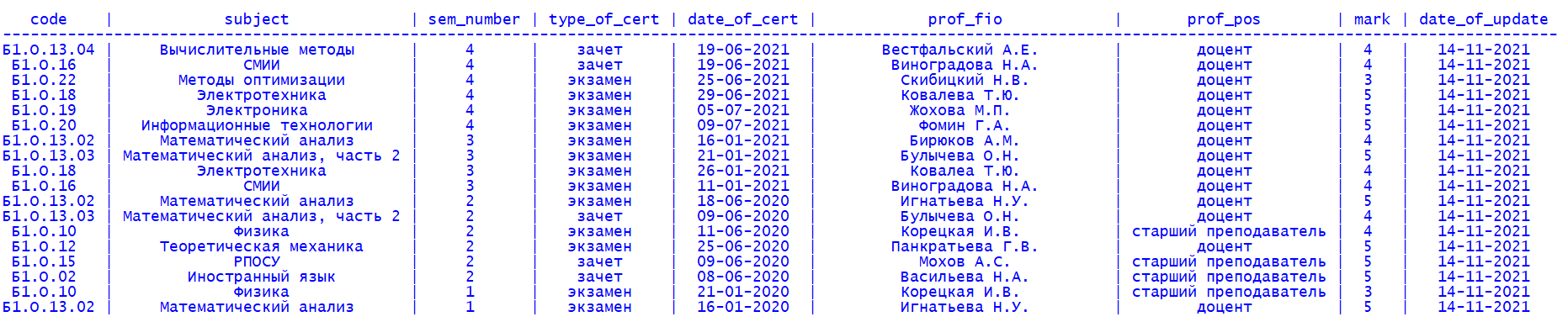
Добавление новой записи.



Ввод условия и просмотр подмножества.



Финальная версия БД



# Часть 3 НИР