МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний аерокосмічний університет ім. М.Є. Жуковського

“Харківський авіаційний інститут”

Кафедра комп’ютерних систем, мереж і кібербезпеки

Титульный лист заполняется на украинском языке

Остальная часть отчета может быть как на украинском, так и на русском

Лабораторна робота № 5

з дисципліни “Обробка даних засобами Python”

Робота с базами данных SQL в Python

Виконав студент гр. 555іМБ Литвинов.О.А Фамилия И.О.

(підпис, дата)

Перевірив к.т.н зав.кафедри каф. 301

(науковий ступінь, вчене звання, посада)

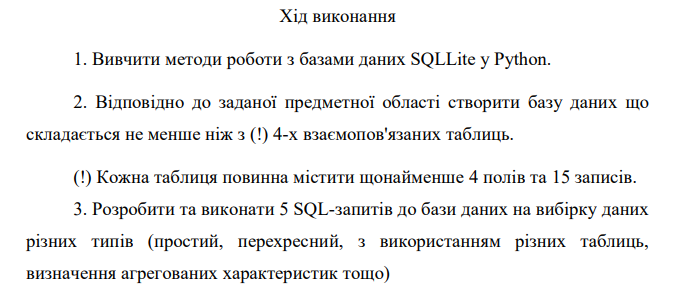
Дергачов.К.Ю

(підпис, дата) (П.І.Б.)

2023

Мета: Дослідження методів роботи з базами даних у Python

18. Система документообігу



main.py

import database

import test\_data\_for\_db

import test\_request

database.create\_db()

test\_data\_for\_db.create\_test\_data()

test\_request.test\_requests()

datebase.py

import sqlite3

def create\_db():

    conn = sqlite3.connect('database.db')

    cursor = conn.cursor()

    cursor.execute('''DROP TABLE IF EXISTS documents;''')

    cursor.execute('''DROP TABLE IF EXISTS users;''')

    cursor.execute('''DROP TABLE IF EXISTS departments;''')

    cursor.execute('''DROP TABLE IF EXISTS change\_history;''')

    cursor.execute('''

    CREATE TABLE IF NOT EXISTS documents (

            id\_document INT PRIMARY KEY,

            name\_document TEXT,

            create\_date DATE,

            user INT,

            FOREIGN KEY (user) REFERENCES users (id\_user)

    );

    ''')

    cursor.execute('''

    CREATE TABLE IF NOT EXISTS users (

            id\_user INT PRIMARY KEY,

            name TEXT,

            position TEXT,

            id\_department INT,

            FOREIGN KEY (id\_department) REFERENCES departments (id\_department)

    );

    ''')

    cursor.execute('''

    CREATE TABLE IF NOT EXISTS departments (

            id\_department INT PRIMARY KEY,

            name\_department TEXT,

            head\_of\_department INT,

            role\_department TEXT,

            FOREIGN KEY (head\_of\_department) REFERENCES users (id\_user)

    );

    ''')

    cursor.execute('''

    CREATE TABLE IF NOT EXISTS change\_history (

            id\_change INTEGER PRIMARY KEY,

            id\_document INT,

            change\_date DATE,

            text\_change TEXT,

            FOREIGN KEY (id\_document) REFERENCES documents (id\_document)

    );

    ''')

    conn.commit()

    conn.close()

test\_data\_for\_db.py

import sqlite3

from datetime import datetime, date, timedelta

from random import randint

def create\_test\_data():

    conn = sqlite3.connect('database.db')

    cursor = conn.cursor()

    documents\_data = [(i, f'document\_{i}', datetime.now().date(),

                       randint(1, 15)) for i in range(1, 16)]

    users\_data = [(i, f'user\_{i}', f'position\_{i + randint(1, 100)}',

                   randint(1, 15)) for i in range(1, 16)]

    departments\_data = [(i, f'department №{i}', f'user\_{16 - i}',

                         f'role\_department\_{16 - i}') for i in range(1, 16)]

    change\_history\_data = [(i, 16 - i, date(2023, 1, 1) + timedelta(days=i),

                            f'some change text №{i}') for i in range(1, 16)]

    cursor.executemany('INSERT INTO documents VALUES (?, ?, ?, ?)',

                       documents\_data)

    cursor.executemany(f'INSERT INTO users VALUES (?, ?, ?, ?)', users\_data)

    cursor.executemany(f'INSERT INTO departments VALUES (?, ?, ?, ?)',

                       departments\_data)

    cursor.executemany(f'INSERT INTO change\_history VALUES (?, ?, ?, ?)',

                       change\_history\_data)

    conn.commit()

test\_request.py

import sqlite3

class Colors:

    DEFAULT = '\033[0m'

    CHANGED = '\033[95m'

def test\_requests():

    conn = sqlite3.connect('database.db')

    cursor = conn.cursor()

    print(f'{Colors.CHANGED}Simple request:')

    simple\_response = cursor.execute('''

        SELECT name\_document, create\_date FROM documents;

    ''').fetchall()

    print\_response(simple\_response)

    print(f'{Colors.CHANGED}Cross join request:')

    cross\_join = cursor.execute("""

        SELECT documents.name\_document, users.name

        FROM documents INNER JOIN users

        ON documents.user = users.id\_user;

    """)

    print\_response(cross\_join)

    print(f'{Colors.CHANGED}Using different tables request:')

    using\_different\_tables = cursor.execute('''

        SELECT documents.name\_document,

        users.name, departments.name\_department,

        change\_history.change\_date

        FROM documents

        JOIN users ON documents.user = users.id\_user

        JOIN departments ON users.id\_department = departments.id\_department

        JOIN change\_history ON change\_history.id\_document = documents.id\_document;

    ''')

    print\_response(using\_different\_tables)

    print(f'{Colors.CHANGED}aggregated characteristics request:')

    aggregated\_characteristics = cursor.execute('''

    SELECT documents.name\_document, change\_history.change\_date

    FROM documents

    LEFT JOIN change\_history ON documents.id\_document = change\_history.id\_document

    WHERE change\_history.change\_date > '2023-01-05';

    ''')

    print\_response(aggregated\_characteristics)

    conn.close()

def print\_response(obj):

    for line in obj:

        print(f'{Colors.DEFAULT}',line)

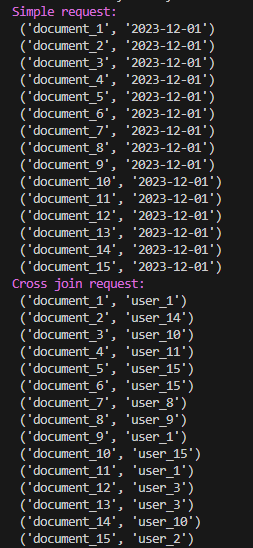


Рисунок 1 – Результати запитів до ДБ

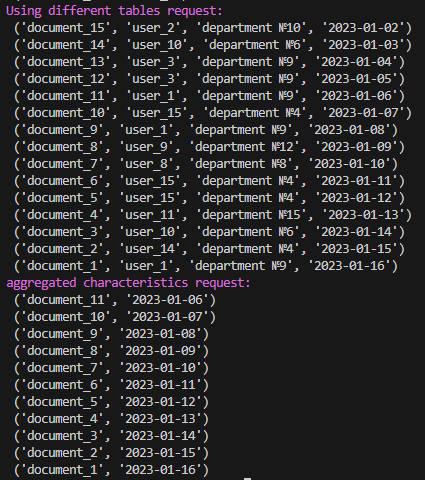


Рисунок 2 – Результати запитів до ДБ

Висновок:

Для виконання роботи було використано стандартну бібліотеку для python sqlite3. Було використано такі типи даних як INTEGER(цілі чисельні значення), та TEXT(текст). Для більшості дій достатньо команди cursor.execute, за допомогою якої робиться запит до сервера, також була використана cursor.executemany для масового запиту, такого як вставка масиву даних. Сама робота з бд для мене не є новою так як, я обширно використовую sqlite у мобільній розробці.

Є основні категорії запитів:

SELECT – для виборки, а точніше для перегляду інформації з БД;

CREATE TABLE – створення таблиці;

INSERT INTO – завантаження інформації до таблиці;

DELETE – видалити інформацію з таблиці;

DROP TABLE – видалити таблицю;