Лабораторна робота №3

Студента групи МІТ-31

Заїчка Івана

Мета: cтворити REST сервіс для роботи з базою даних (виконання CRUD операцій), розробленої у попередніх лабораторних роботах.

Для виконання даної лабораторної роботи використаємо технології СУБД: PostgreSQL, та Python.

-- Active: 1714365464599@@127.0.0.1@5432@users

-- 1. Процедура для додавання нового листа:

CREATE OR REPLACE PROCEDURE add\_email(IN subject VARCHAR(255), IN message TEXT, IN sender\_id INT, IN receiver\_id INT, IN amount INT)

AS

$$

BEGIN

    INSERT INTO emails (subject, message, sender\_id, receiver\_id, amount) VALUES (subject, message, sender\_id, receiver\_id, amount);

END;

$$

LANGUAGE plpgsql;

SELECT \* FROM emails;

CALL add\_email('тема п', 'Нове повідомлення з процедури', 1, 2, 200);

-- 2. Процедура для оновлення існуючого листа:

CREATE OR REPLACE PROCEDURE update\_email(IN p\_email\_id INT, IN new\_subject VARCHAR(255), IN new\_message TEXT, IN new\_amount INT)

AS

$$

BEGIN

    UPDATE emails

    SET subject = new\_subject, message = new\_message, amount = new\_amount

    WHERE email\_id = p\_email\_id;

END;

$$

LANGUAGE plpgsql;

SELECT \* FROM emails;

CALL update\_email(1, 'Оновлена тема з процедури', 'Процедура процедура', 300);

-- 3. Архівування листа у поштовій скриньці:

CREATE OR REPLACE PROCEDURE archive\_email(

    IN p\_mailbox\_id INT

)

AS $$

BEGIN

    UPDATE mailbox SET is\_archived = TRUE WHERE mailbox\_id = p\_mailbox\_id;

END;

$$ LANGUAGE plpgsql;

CALL archive\_email(1);

SELECT \* FROM mailbox;

-- функція для підрахунку кількості листів, отриманих конкретним користувачем

CREATE FUNCTION count\_received\_emails(user\_id INT)

RETURNS INT AS $$

DECLARE

    received\_count INT;

BEGIN

    SELECT COUNT(\*)

    INTO received\_count

    FROM emails

    WHERE receiver\_id = user\_id;

    RETURN received\_count;

END;

$$ LANGUAGE plpgsql;

SELECT count\_received\_emails(1);

-- функція для підрахунку загальної суми символів, отриманих конкретним користувачем через листи

CREATE FUNCTION sum\_received\_amount(user\_id INT)

RETURNS INT AS $$

DECLARE

    total\_amount INT;

BEGIN

    SELECT COALESCE(SUM(amount), 0)

    INTO total\_amount

    FROM emails

    WHERE receiver\_id = user\_id;

    RETURN total\_amount;

END;

$$ LANGUAGE plpgsql;

SELECT sum\_received\_amount(1);

-- функція для підрахунку кількості листів, надісланих конкретним користувачем за певний період часу

CREATE FUNCTION count\_sent\_emails(user\_id INT, start\_date TIMESTAMP, end\_date TIMESTAMP)

RETURNS INT AS $$

DECLARE

    sent\_count INT;

BEGIN

    SELECT COUNT(\*)

    INTO sent\_count

    FROM emails

    WHERE sender\_id = user\_id AND sent\_at BETWEEN start\_date AND end\_date;

    RETURN sent\_count;

END;

$$ LANGUAGE plpgsql;

SELECT count\_sent\_emails(1, '2024-01-01 00:00:00', '2024-05-19 23:59:59');

-- тригер для автоматичного додавання листів у папку "inbox" після їх створення:

CREATE OR REPLACE FUNCTION add\_email\_to\_inbox()

RETURNS TRIGGER AS $$

BEGIN

    INSERT INTO mailbox (email\_id, id\_folder, is\_archived)

    VALUES (NEW.email\_id, (SELECT id\_folder FROM folder\_directory WHERE folder\_name = 'inbox'), FALSE);

    RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER trg\_add\_email\_to\_inbox

AFTER INSERT ON emails

FOR EACH ROW

EXECUTE FUNCTION add\_email\_to\_inbox();

INSERT INTO emails (subject, message, sender\_id, receiver\_id, amount)

VALUES ('Тестовий 1', 'Це тестовий лист для перевірки тригера', 1, 2, 100);

SELECT

    e.email\_id,

    e.subject,

    e.message,

    fd.folder\_name

FROM

    emails e

JOIN

    mailbox mb ON e.email\_id = mb.email\_id

JOIN

    folder\_directory fd ON mb.id\_folder = fd.id\_folder;

-- тригер для оновлення поля sent\_at при кожному оновленні листа:

CREATE OR REPLACE FUNCTION update\_sent\_at()

RETURNS TRIGGER AS $$

BEGIN

    NEW.sent\_at := CURRENT\_TIMESTAMP;

    RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER trg\_update\_sent\_at

BEFORE UPDATE ON emails

FOR EACH ROW

EXECUTE FUNCTION update\_sent\_at();

UPDATE emails

SET message = 'Оновлене повідомлення 3x'

WHERE email\_id = (SELECT MAX(email\_id) FROM emails);

SELECT email\_id, subject, message, sent\_at FROM emails WHERE email\_id = (SELECT MAX(email\_id) FROM emails);

-- тригер, який автоматично додає лист у поштову скриньку користувача, якщо він адресований йому

CREATE OR REPLACE FUNCTION add\_to\_mailbox()

RETURNS TRIGGER AS $$

BEGIN

    INSERT INTO mailbox (email\_id, id\_folder, is\_archived) VALUES (NEW.email\_id, 2, false); -- Інакше це лист отриманий

    RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER add\_to\_mailbox\_trigger

AFTER INSERT ON emails

FOR EACH ROW EXECUTE FUNCTION add\_to\_mailbox();

INSERT INTO emails (subject, message, sender\_id, receiver\_id, amount)

VALUES ('Лист3', 'Це тестовий лист для перевірки тригера 3', 1, 2, 100);

DROP TRIGGER add\_to\_mailbox\_trigger ON emails;

SELECT

    e.email\_id,

    e.subject,

    e.message,

    fd.folder\_name

FROM

    emails e

JOIN

    mailbox mb ON e.email\_id = mb.email\_id

JOIN

    folder\_directory fd ON mb.id\_folder = fd.id\_folder;

-- транзакції

-- транзакція 1 оновлення теми листа та переміщення до папки "drafts"

BEGIN;

-- Оновлення теми листа з email\_id 5

UPDATE emails

SET subject = 'Updated Subject'

WHERE email\_id = 1;

-- Переміщення листа з email\_id 5 до папки "drafts"

UPDATE mailbox

SET id\_folder = (SELECT id\_folder FROM folder\_directory WHERE folder\_name = 'drafts')

WHERE email\_id = 1;

COMMIT;

SELECT \* FROM emails;

SELECT \* FROM mailbox;

-- Транзакція 2: переміщення листа до папки "trash" і архівування

BEGIN;

-- Переміщення листа з id 3 до папки "trash"

UPDATE mailbox

SET id\_folder = (SELECT id\_folder FROM folder\_directory WHERE folder\_name = 'trash')

WHERE email\_id = 3;

-- Архівування листа з id 3

UPDATE mailbox

SET is\_archived = true

WHERE email\_id = 3;

COMMIT;

SELECT \* FROM mailbox WHERE email\_id = 3;

-- Транзакція 3: Зміна пароля користувача та видалення листа

BEGIN;

-- Зміна пароля для користувача з user\_id 2

UPDATE users

SET password = 'newpassword2'

WHERE user\_id = 2;

-- Видалення запису з таблиці mailbox, що відноситься до видаленого листа

DELETE FROM mailbox

WHERE email\_id = 4;

-- Видалення листа з id 4 з таблиці emails

DELETE FROM emails

WHERE email\_id = 4;

COMMIT;

SELECT \* FROM users;

SELECT \* FROM emails;

ROLLBACK;

Рест-сервіс:

from flask import Flask, request, jsonify

import psycopg2

from psycopg2.extras import RealDictCursor

app = Flask(\_\_name\_\_)

*def* get\_db\_connection():

    conn = psycopg2.connect(

*dbname*='users',

*user*='postgres',

*password*='admin',

*host*='localhost',

*port*='5432'

    )

    return conn

# CRUD операції для таблиці листів

@app.route('/emails', *methods*=['POST'])

*def* create\_email():

    data = request.get\_json()

    conn = get\_db\_connection()

    cur = conn.cursor()

    cur.execute(

        "INSERT INTO emails (subject, message, sender\_id, receiver\_id, amount) VALUES (%s, %s, %s, %s, %s) RETURNING email\_id;",

        (data['subject'], data['message'], data['sender\_id'], data['receiver\_id'], data['amount'])

    )

    email\_id = cur.fetchone()[0]

    conn.commit()

    cur.close()

    conn.close()

    return jsonify({'email\_id': email\_id}), 201

@app.route('/emails/<int:email\_id>', *methods*=['GET'])

*def* get\_email(*email\_id*):

    conn = get\_db\_connection()

    cur = conn.cursor(*cursor\_factory*=RealDictCursor)

    cur.execute("SELECT \* FROM emails WHERE email\_id = %s;", (email\_id,))

    email = cur.fetchone()

    cur.close()

    conn.close()

    return jsonify(email)

@app.route('/emails', *methods*=['GET'])

*def* get\_emails():

    conn = get\_db\_connection()

    cur = conn.cursor(*cursor\_factory*=RealDictCursor)

    cur.execute("SELECT \* FROM emails;")

    emails = cur.fetchall()

    cur.close()

    conn.close()

    return jsonify(emails)

@app.route('/emails/<int:email\_id>', *methods*=['PUT'])

*def* update\_email(*email\_id*):

    data = request.get\_json()

    conn = get\_db\_connection()

    cur = conn.cursor()

    cur.execute(

        "UPDATE emails SET subject = %s, message = %s, sender\_id = %s, receiver\_id = %s, amount = %s WHERE email\_id = %s;",

        (data['subject'], data['message'], data['sender\_id'], data['receiver\_id'], data['amount'], email\_id)

    )

    conn.commit()

    cur.close()

    conn.close()

    return '', 204

@app.route('/emails/<int:email\_id>', *methods*=['DELETE'])

*def* delete\_email(*email\_id*):

    conn = get\_db\_connection()

    cur = conn.cursor()

    cur.execute("DELETE FROM emails WHERE email\_id = %s;", (email\_id,))

    conn.commit()

    cur.close()

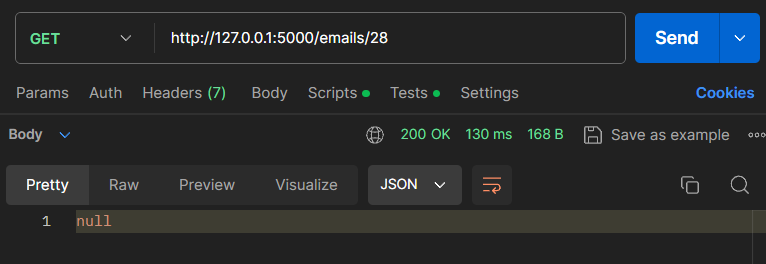
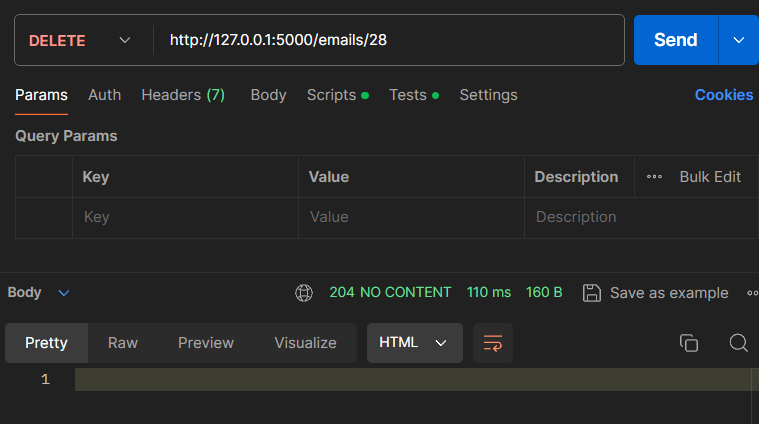
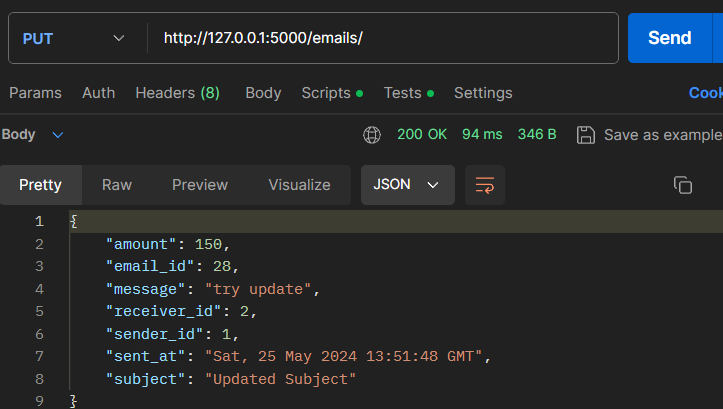
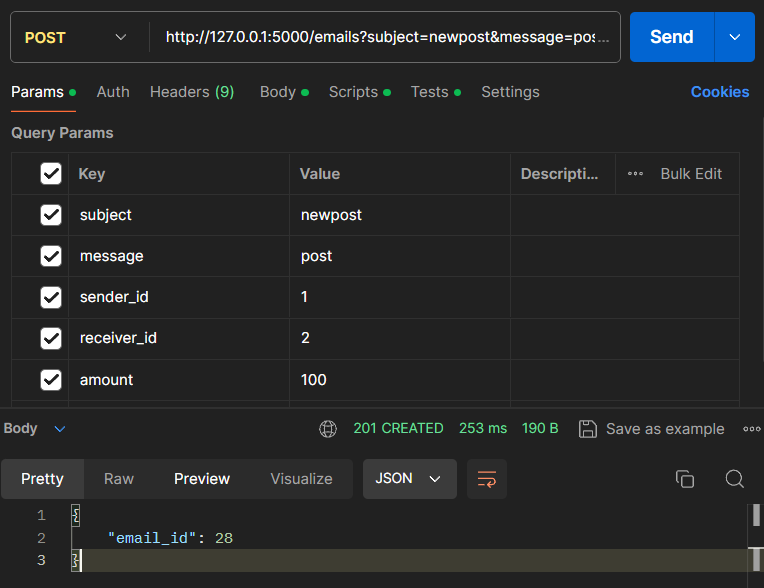
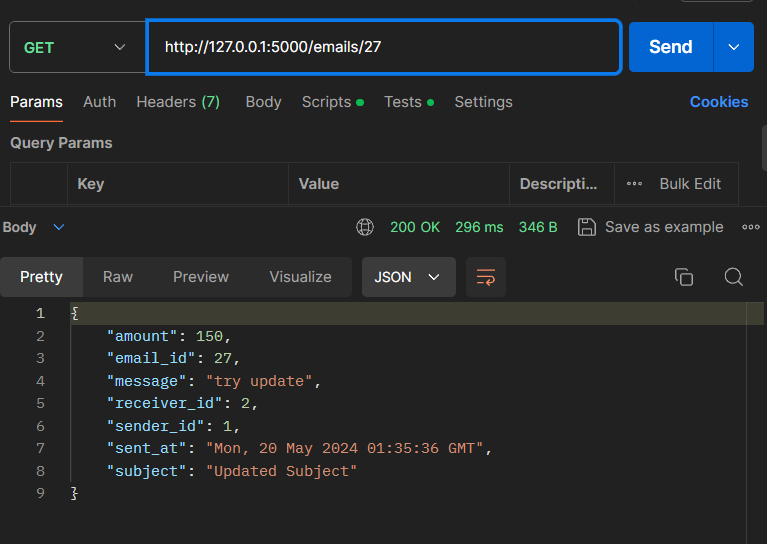
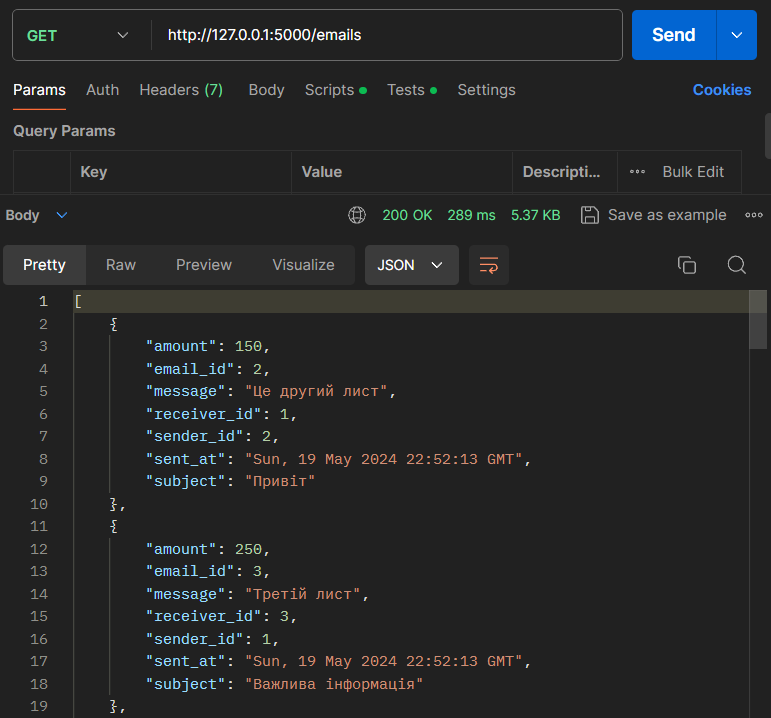
    conn.close()

    return '', 204

# Запуск сервера

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

    app.run(*debug*=True)



Висновки: у ході лабораторної роботи було досліджено роботу з базами даних, створюючи REST сервіс, використовуючи технології СУБД, процедури, функції, тригери та індекси для виконання CRUD операцій.

Додатки

1. lab3.sql
2. lab3.py