

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Львівський національний університет імені Івана Франка
Факультет електроніки та комп'ютерних технологій

ЗВІТ

про виконання ЛАБОРАТОРНОЇ РОБОТИ № 9

з дисципліни “Бази даних”

На тему: “Робота з базами даних у Docker-контейнерах. Реалізація доступу до даних через шаблони Active Record та DAO.”

Виконав

студент 2 курсу

групи ФЕП-23

Чепара Станіслав Богданович

Перевірив доцент:

Анохін Володимир Євгенович

Львів 2025

Мета роботи

Навчитися працювати з контейнеризованими базами даних, перемикатися між локальними та Docker-БД, застосовувати шаблони Active Record і Data Access Object (DAO).

Обладнання та ПЗ

- Python 3.x
- PostgreSQL (локальний)
- Docker + Docker Compose
- Модуль psycopg2-binary
- PostgreSQL контейнер з образом postgres:16

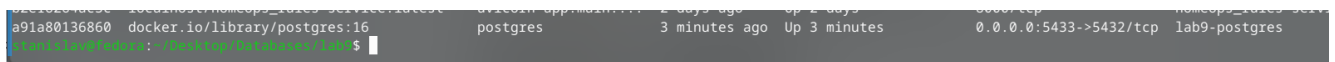
Теоретичні відомості

У роботі розглядаються два підходи до доступу до даних:

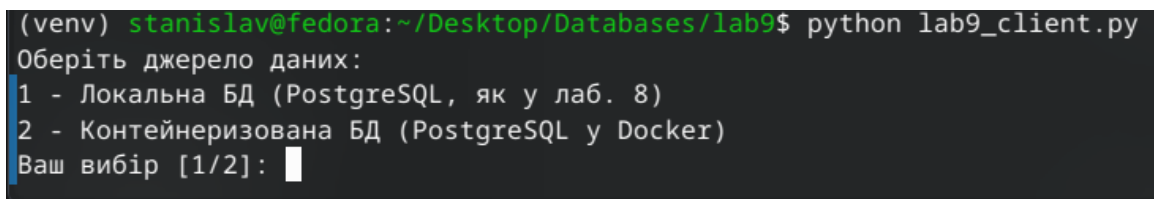
1. Active Record — модель відповідає таблиці БД, об'єкт — рядку, а методи класу виконують CRUD-операції.
2. DAO (Data Access Object) — виділяє спеціальний клас для роботи з SQL, забезпечує кращу структурованість та відокремлення бізнес-логіки.

Хід роботи

1. Встановлено Docker.
2. Підготовлено docker-compose.yml для розгортання PostgreSQL у контейнері.
3. Запущено контейнер командою `docker-compose up -d`.



4. Створено Python-клієнт, який може перемикаватися між локальною БД та контейнерною.



5. Реалізовано Active Record: клас ClientRecord із методами save(), delete(), all(), find().

```
2 - Контейнеризована БД (PostgreSQL у Docker)
Ваш вибір [1/2]: 2
Використовується контейнеризована БД.
Таблиця clients готова до роботи.
```

```
=== Головне меню (Lab 9) ===
```

```
1. Показати всіх клієнтів (Active Record)
2. Додати клієнта (Active Record)
3. Оновити клієнта (Active Record)
4. Видалити клієнта (Active Record)
```

```
5. Показати всіх клієнтів (DAO)
6. Додати клієнта (DAO)
7. Оновити клієнта (DAO)
8. Видалити клієнта (DAO)
```

```
9. Виміряти час SELECT для обох БД
0. Вихід
```

```
Оберіть пункт меню: 2
Ім'я: Ivan
Email: ivan@gmail.com
Вік (ціле число): 67
Створено клієнта з id=1
```

```
Оберіть пункт меню: 1
ClientRecord(id=1, name='Ivan', email='ivan@gmail.com', age=67, created_at=datetime.datetime(2025, 12, 3, 9, 53, 31, 350608))
```

```
Оберіть пункт меню: 3
ID клієнта для оновлення: 1
Поточні дані: ClientRecord(id=1, name='Ivan', email='ivan@gmail.com', age=67, created_at=datetime.datetime(2025, 12, 3, 9, 53, 31, 350608))
Нове ім'я [Ivan]: Oleksandr
Новий email [ivan@gmail.com]:
Новий вік [67]:
Оновлено клієнта з id=1
```

```
Оберіть пункт меню: 4
ID клієнта для видалення: 1
Клієнта з id=1 видалено.
```

6. Реалізовано DAO: клас ClientDAO з методами insert(), update(), delete(), get_all().

```
Оберіть пункт меню: 6
Ім'я: Maria
Email: maria@gmail.com
Вік (ціле число): 20
Створено клієнта DAO з id=2
```

```
Оберіть пункт меню: 7
ID клієнта для оновлення: 2
Поточні дані: ClientRecord(id=2, name='Maria', email='maria@gmail.com', age=20, created_at=datetime.datetime(2025, 12, 3, 9, 57, 48, 326666))
Нове ім'я [Maria]: Stefania
Новий email [maria@gmail.com]:
Новий вік [20]: 34
Клієнта оновлено.
```

```
Оберіть пункт меню: 5
ClientRecord(id=2, name='Stefania', email='maria@gmail.com', age=34, created_at=datetime.datetime(2025, 12, 3, 9, 57, 48, 326666))

=== Головне меню (Lab 9) ===
1. Показати всіх клієнтів (Active Record)
2. Додати клієнта (Active Record)
3. Оновити клієнта (Active Record)
4. Видалити клієнта (Active Record)

5. Показати всіх клієнтів (DAO)
6. Додати клієнта (DAO)
7. Оновити клієнта (DAO)
8. Видалити клієнта (DAO)

9. Виміряти час SELECT для обох БД
0. Вихід

Оберіть пункт меню: 8
ID клієнта для видалення: 2
Клієнта видалено (DAO).
```

7. Додано вимірювання часу SELECT-запитів для обох БД.

```
9. Виміряти час SELECT для обох БД
0. Вихід

Оберіть пункт меню: 9

Вимірювання часу SELECT для: Локальна БД
Таблиця clients готова до роботи.
Час виконання SELECT: 0.85 мс

Вимірювання часу SELECT для: Контейнерна БД
Таблиця clients готова до роботи.
Час виконання SELECT: 1.10 мс
```

Висновки

У ході роботи було вивчено роботу з Docker-контейнерами для розгортання СУБД PostgreSQL, реалізовано два підходи доступу до даних — Active Record та DAO, а також проведено порівняння часу виконання запитів у локальній та контейнерній БД.