Лабораторна робота № 10

“Створення об’єктноорієнтованого інтерфейсу до бази даних”

з курсу “Організація баз даних та знань”

Виконав:

Студент групи ФЕС-21с

Шавало А. А.

Викладач: асист. Галяткін О. О.

Львів 2024

Лабораторна робота №10. Створення об’єктноорієнтованого інтерфейсу до бази даних

Мета роботи: навчитись створювати об’єктно-орієнтовані інтерфейси до баз даних та керувати базами даних за допомогою програмних засобів.

Хід роботи

1. Написати інтерфейс доступу до бази даних використовуючи об’єктно-орієнтовану мову програмування, що реалізує один з шаблонів доступу до бази даних — Active Record або DAO (Data Access Object) і містить наступний функціонал:

a) Додавання нового запису (рядка) в таблицю бази даних;

```
def add_lawyer(self, lawyer):
    cursor = self.connection.cursor()
    query = """INSERT INTO lawyers (name, birth_date,
address, phone, education, position, experience_years)
               VALUES (%s, %s, %s, %s, %s, %s, %s)"""
    cursor.execute(query, (lawyer.name, lawyer.birth_date,
lawyer.address, lawyer.phone,
                           lawyer.education,
lawyer.position, lawyer.experience_years))
    self.connection.commit()
```

b) Отримання одного (унікального або першого) запису з бази даних за допомогою певного фільтра;

```
def find_by_id(self, lawyer_id):
    cursor = self.connection.cursor()
    query = "SELECT * FROM lawyers WHERE id = %s"
    cursor.execute(query, (lawyer_id,))
    result = cursor.fetchone()
    if result:
        return Lawyer(result[1], result[2], result[3], result[4],
result[5], result[6], result[7], result[0])
    return None
```

c) Оновлення наявного запису в базі даних;

```
def update_lawyer(self, lawyer):
    cursor = self.connection.cursor()
    query = """UPDATE lawyers SET name = %s, birth_date = %s,
address = %s, phone = %s,
             education = %s, position = %s, experience_years = %s
WHERE id = %s"""
    cursor.execute(query, (lawyer.name, lawyer.birth_date,
lawyer.address, lawyer.phone, lawyer.education,
                           lawyer.position, lawyer.experience_years,
lawyer.id))
    self.connection.commit()
```

d) Видалення запису з бази даних.

```
def delete_lawyer(self, lawyer_id):
    cursor = self.connection.cursor()
    lawyer = self.find_by_id(lawyer_id)  # Отримати юриста
для виводу
    if lawyer:  # Якщо юриста знайдено
        query = "DELETE FROM lawyers WHERE id = %s"
        cursor.execute(query, (lawyer_id,))
        self.connection.commit()
        print(f"Видалено юриста з ID {lawyer_id}:\n"
              f"Ім'я: {lawyer.name}, Дата народження:
{lawyer.birth_date}, Адреса: {lawyer.address}, "
              f"Телефон: {lawyer.phone}, Освіта:
{lawyer.education}, Посада: {lawyer.position}, "
              f"Кількість років досвіду:
{lawyer.experience_years}")
    else:
        print(f"Юриста з ID {lawyer_id} не знайдено.")
```

2. У основній частині програми здійснити:

a) Підключення до бази даних;

```
dao = LawyerDAOMySQL(host="localhost", user="root", password="", database="lab5")
```

b) Створення нового запису, використовуючи створений інтерфейс;

```
Додано юриста: Ім'я: Тест, Дата народження: 1985-05-15, Адреса: вул. Грушевського 2, Телефон: 380671234567,
```

c) Отримання створеного запису з таблиці бази даних, використовуючи створений інтерфейс;

```
Знайдено юриста: Ім'я: Петров Петро Петрович, Дата народження: 1985-07-20, Адреса: вул. Шевченка 10, Телефон: 380987654321,
```

d) Оновлення створеного запису з таблиці бази даних, використовуючи створений інтерфейс;

```
Оновлено юриста з ID 1:
Старі значення: Телефон: 380671234567, Адреса: вул. Грушевського 2, Освіта: Юридична академія, Посада: Партнер, Кількість років досвіду: 15
Нові значення: Телефон: 123123123, Адреса: вул. Грушевського 2, Освіта: Юридична академія, Посада: Партнер, Кількість років досвіду: 15
```

e) Видалення створеного запису з таблиці бази даних, використовуючи створений інтерфейс.

```
Видалено юриста з ID 33:
Ім'я: Тест, Дата народження: 1985-05-15, Адреса: вул. Грушевського 2, Телефон: 380671234567,
```

Лістинінг коду :

```
import mysql.connector
from mysql.connector import Error


class Lawyer:
    def __init__(self, name, birth_date, address, phone, education,
position, experience_years, lawyer_id=None):
        self.id = lawyer_id
        self.name = name
        self.birth_date = birth_date
        self.address = address
        self.phone = phone
        self.education = education
        self.position = position
        self.experience_years = experience_years


class ILawyerDAO:
    def add_lawyer(self, lawyer):
        pass

    def find_by_id(self, lawyer_id):
        pass

    def find_by_name(self, name):
        pass

    def update_lawyer(self, lawyer):
        pass

    def delete_lawyer(self, lawyer_id):
        pass


class LawyerDAOMySQL(ILawyerDAO):
    def __init__(self, host, user, password, database):
        self.connection = mysql.connector.connect(
            host=host,
```

```
            user=user,
            password=password,
            database=database
        )

    def add_lawyer(self, lawyer):
        cursor = self.connection.cursor()
        query = """INSERT INTO lawyers (name, birth_date, address,
phone, education, position, experience_years)
                   VALUES (%s, %s, %s, %s, %s, %s, %s)"""
        cursor.execute(query, (lawyer.name, lawyer.birth_date,
lawyer.address, lawyer.phone,
                               lawyer.education, lawyer.position,
lawyer.experience_years))
        self.connection.commit()

    def find_by_id(self, lawyer_id):
        cursor = self.connection.cursor()
        query = "SELECT * FROM lawyers WHERE id = %s"
        cursor.execute(query, (lawyer_id,))
        result = cursor.fetchone()
        if result:
            return Lawyer(result[1], result[2], result[3],
result[4], result[5], result[6], result[7], result[0])
        return None

    def find_by_name(self, name):
        cursor = self.connection.cursor()
        query = "SELECT * FROM lawyers WHERE name = %s"
        cursor.execute(query, (name,))
        result = cursor.fetchone()
        if result:
            return Lawyer(result[1], result[2], result[3],
result[4], result[5], result[6], result[7], result[0])
        return None

    def update_lawyer(self, lawyer):
        cursor = self.connection.cursor()
        query = """UPDATE lawyers SET name = %s, birth_date = %s,
address = %s, phone = %s,
                   education = %s, position = %s, experience_years =
%s WHERE id = %s"""
```

```
        cursor.execute(query, (lawyer.name, lawyer.birth_date,
lawyer.address, lawyer.phone, lawyer.education,
                               lawyer.position,
lawyer.experience_years, lawyer.id))
        self.connection.commit()

    def delete_lawyer(self, lawyer_id):
        cursor = self.connection.cursor()
        lawyer = self.find_by_id(lawyer_id)  # Отримати юриста для
виводу
        if lawyer:  # Якщо юриста знайдено
            query = "DELETE FROM lawyers WHERE id = %s"
            cursor.execute(query, (lawyer_id,))
            self.connection.commit()
            print(f"Видалено юриста з ID {lawyer_id}:\n"
                  f"Ім'я: {lawyer.name}, Дата народження:
{lawyer.birth_date}, Адреса: {lawyer.address}, "
                  f"Телефон: {lawyer.phone}, Освіта:
{lawyer.education}, Посада: {lawyer.position}, "
                  f"Кількість років досвіду:
{lawyer.experience_years}")
        else:
            print(f"Юриста з ID {lawyer_id} не знайдено.")


def add_new_lawyer(dao, name, birth_date, address, phone, education,
position, experience_years):
    new_lawyer = Lawyer(name=name, birth_date=birth_date,
                        address=address, phone=phone,
                        education=education, position=position,
experience_years=experience_years)
    try:
        dao.add_lawyer(new_lawyer)
        print(
            f"Додано юриста: Ім'я: {new_lawyer.name}, Дата
народження: {new_lawyer.birth_date}, Адреса: {new_lawyer.address},
Телефон: {new_lawyer.phone}, Освіта: {new_lawyer.education}, Посада:
{new_lawyer.position}, Кількість років досвіду:
{new_lawyer.experience_years}")
    except Error as e:
        print(f"Помилка при додаванні юриста: {e}")
```

```python
def get_lawyer_by_id(dao, lawyer_id):
    try:
        lawyer = dao.find_by_id(lawyer_id)
        if lawyer:
            print(
                f"Знайдено юриста: Ім'я: {lawyer.name}, Дата
народження: {lawyer.birth_date}, Адреса: {lawyer.address}, Телефон:
{lawyer.phone}, Освіта: {lawyer.education}, Посада:
{lawyer.position}, Кількість років досвіду:
{lawyer.experience_years}")
        else:
            print(f"Юриста з ID {lawyer_id} не знайдено.")
        return lawyer
    except Error as e:
        print(f"Помилка при отриманні юриста: {e}")


def update_lawyer_info(dao, lawyer_id, phone=None, address=None,
education=None, position=None, experience_years=None):
    try:
        lawyer = dao.find_by_id(lawyer_id)
        if lawyer:
            # Зберегти старі значення
            old_info = {
                'phone': lawyer.phone,
                'address': lawyer.address,
                'education': lawyer.education,
                'position': lawyer.position,
                'experience_years': lawyer.experience_years,
            }

            # Оновити значення
            if phone:
                lawyer.phone = phone
            if address:
                lawyer.address = address
            if education:
                lawyer.education = education
            if position:
                lawyer.position = position
            if experience_years is not None:
```

```
                lawyer.experience_years = experience_years

            # Зберегти зміни
            dao.update_lawyer(lawyer)

            # Вивести старі та нові значення
            print(f"Оновлено юриста з ID {lawyer_id}:\n"
                  f"Старі значення: Телефон: {old_info['phone']},
Адреса: {old_info['address']}, "
                  f"Освіта: {old_info['education']}, Посада:
{old_info['position']}, "
                  f"Кількість років досвіду:
{old_info['experience_years']}\n"
                  f"Нові значення: Телефон: {lawyer.phone}, Адреса:
{lawyer.address}, "
                  f"Освіта: {lawyer.education}, Посада:
{lawyer.position}, "
                  f"Кількість років досвіду:
{lawyer.experience_years}")
        else:
            print(f"Юриста з ID {lawyer_id} не знайдено.")
    except Error as e:
        print(f"Помилка при оновленні юриста: {e}")


def delete_lawyer(dao, lawyer_id):
    dao.delete_lawyer(lawyer_id)


def main():
    dao = LawyerDAOMySQL(host="localhost", user="root", password="",
database="lab5")
    add_new_lawyer(dao,
                   "Тест",
                   "1985-05-15",
                   "вул. Грушевського 2",
                   "380671234567",
                   "Академія правосуддя",
                   "Юрист",
                   10)
    lawyer = get_lawyer_by_id(dao, 2)
    update_lawyer_info(dao, 1, phone="123123123", address="вул.
```

```
Грушевського 2")
    delete_lawyer(dao, 33)


if __name__ == "__main__":
    main()
```

Висновок: Нормалізація бази даних є важливим етапом проектування, що забезпечує її ефективність, зменшуючи ризики аномалій при оновленні даних. Однак, важливо враховувати, що надмірна нормалізація може призводити до зниження швидкодії при виконанні запитів. Тому досягнення оптимального балансу між нормалізацією і денормалізацією є критично важливим для забезпечення ефективної роботи бази даних у реальних умовах.