

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені Ігоря Сікорського»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра системного програмування та спеціалізованих комп’ютерних систем

**Лабораторна робота №2**

з дисципліни

**«Бази даних і засоби управління»**

Тема: «Створення додатку бази даних, орієнтованого на взаємодію з СУБД PostgreSQL»

Виконав: студент 3 курсу

ФПМ групи КВ-82

Варган Олексій Ігорович

Перевірив: Павловський В.І.

Київ – 2020

**Метою роботи** є здобуття вмінь програмування прикладних додатків баз даних PostgreSQL.

**Загальне завдання** роботи полягає у наступному:

1. Реалізувати функції внесення, редагування та вилучення даних у таблицях бази даних, створених у лабораторній роботі №1, засобами консольного інтерфейсу.
2. Передбачити автоматичне пакетне генерування «рандомізованих» даних у базі.
3. Забезпечити реалізацію пошуку за декількома атрибутами з 2-х та більше сутностей одночасно: для числових атрибутів – у рамок діапазону, для рядкових – як шаблон функції LIKE оператора SELECT SQL, для логічного типу – значення True/False, для дат – у рамок діапазону дат.
4. Програмний код виконати згідно шаблону MVC (модель-подання контролер).

**Деталізоване завдання**:

1. Забезпечити можливість уведення/редагування/вилучення даних у таблицях бази даних з можливістю контролю відповідності типів даних атрибутів таблиць (рядків, чисел, дати/часу). Для контролю пропонується два варіанти:
   1. контроль при введенні - валідація даних;
   2. перехоплення помилок **(try...except)** від сервера PostgreSQL при виконанні відповідної команди SQL.

Особливу увагу варто звернути на дані таблиць, що мають зв’язок 1:N. З боку батьківської таблиці необхідно контролювати **вилучення (ON DELETE)** рядків за умови наявності даних у підлеглій таблиці. З боку підлеглої таблиці варто контролювати наявність відповідного рядка у батьківській таблиці при виконанні **внесення** до неї нових даних. Унеможливити виведення програмою на екрані системних помилок PostgreSQL шляхом їх перехоплення і адекватної обробки. Внесення даних виконується користувачем у консольному вікні програми.

1. Забезпечити можливість автоматичної генерації великої кількості даних у таблицях за допомогою вбудованих у PostgreSQL функцій роботи з псевдовипадковими числами. Дані мають бути згенерованими **не програмою**, а відповідним **SQL-запитом**! Кількість даних для генерування має вводити користувач з клавіатури.
2. Для реалізації багатокритеріального пошуку необхідно підготувати 3 запити, що включають дані з декількох таблиць і фільтрують рядки за 3-4 атрибутами цих таблиць. Забезпечити можливість уведення конкретних значень констант для фільтрації з клавіатури користувачем. Після виведення даних вивести час виконання запиту у мілісекундах. Перевірити швидкодію роботи запитів на попередньо згенерованих даних.
3. Програмний код організувати згідно шаблону Model-ViewController (MVC). Приклад організації коду згідно шаблону доступний [за даним посиланням](https://www.giacomodebidda.com/mvc-pattern-in-python-introduction-and-basicmodel/). Модель, подання (представлення) та контролер мають бути реалізовані у окремих файлах. Для доступу до бази даних використовувати **лише мову SQL** (без ORM).

**Модель бази даних**

На рис. 1 наведено логічну структуру бази даних яка використовується в даній роботі. В порівнянні з 1-ою лабораторною роботою суттєвих змін не відбулося.

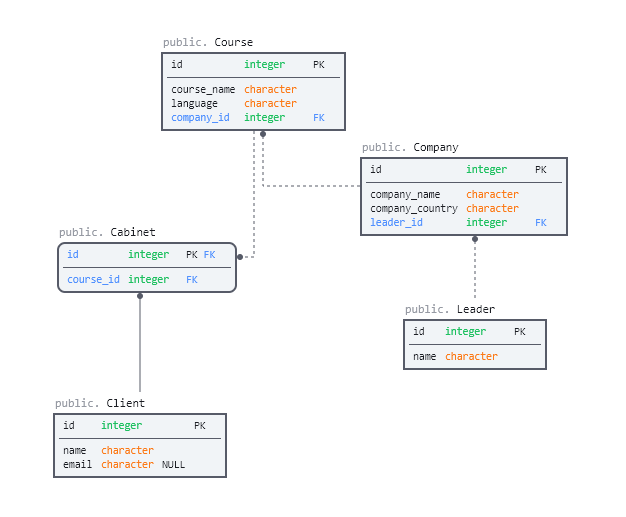


Рис. 1 – Логічна модель БД

**Середовище розробки та налаштування підключення до бази даних**

Для виконання лабораторної роботи використовувалась мова програмування C# та IDE Visual Studio 2019. Для підключення до серверу бази даних PostgreSQL використовувався пакет Npgsql.

**Підключення до БД**

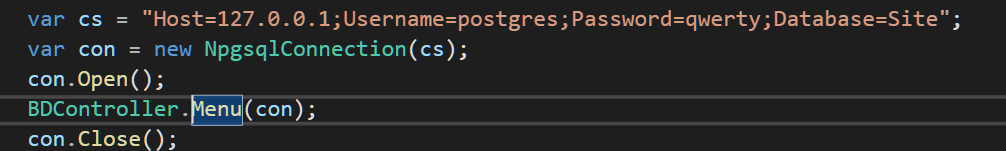


Рис. 2 ‑ Підключення до БД засобами Npgsql

**Структура програми**

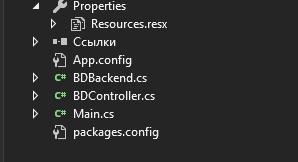


Рис. 3 – Структура програми

Опис структури програми. Програма містить 3 модуля, серед яких BDBackend – модуль, де описані усі sql запити та їх реалізація,

BDController – модуль, де описана робота меню програми,

Main – модуль, де розташована main функція.

**Структура меню програми**

Головне меню

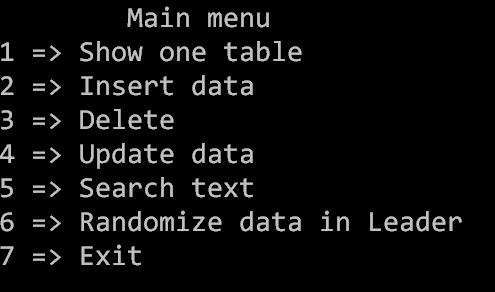


Рис 4. – Фотографія головного меню програми

Меню вибору таблиць (пункти 1, 2, 3, 4)

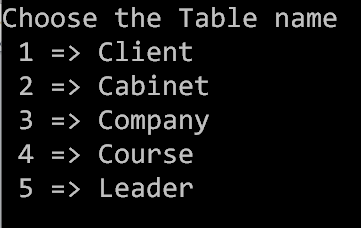


Рис 5. – Фотографія меню, яке з’являється якщо обрати пункти 1, 2, 3, 4

Меню пошуку заготовлених динамічних запитів (пункт 5)

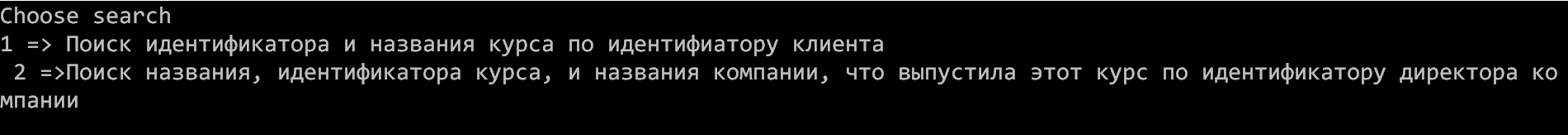


Рис 6. – Пункт 5 головного меню

Меню рандомізації даних (пункт 6)

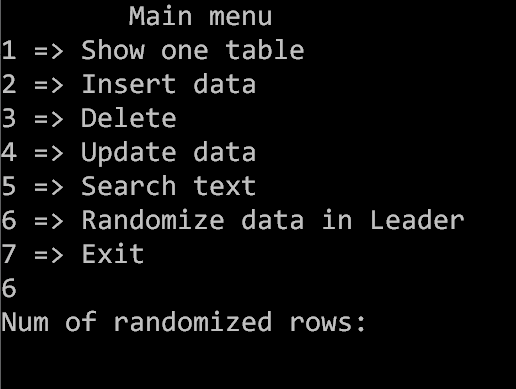


Рис 7. – Пункт 5 головного меню

[**Середовище розробки та налаштування підключення до бази даних** 4](#_Toc57754681)

[**Структура меню програми** 5](#_Toc57754682)

[Лістинги програми з директивами внесення, редагування та вилучення даних у базі даних та результати виконання цих директив 8](#_Toc57754683)

[Функції внесення даних 8](#_Toc57754684)

[Функції оновлення даних 14](#_Toc57754685)

[Функції вилучення даних 19](#_Toc57754686)

[Лістинги програми з директивами внесення рандомізованих даних і виконання динамічних запитів у базі даних та результати виконання цих директив 23](#_Toc57754687)

[Виконання динамічних запитів бази даних 25](#_Toc57754688)

[Обробка виняткових ситуацій (помилок) при введенні/вилученні та валідації даних 27](#_Toc57754689)

[Дослідження режимів обмеження ON DELETE 28](#_Toc57754690)

[Ілюстрації програмного коду на Github 31](#_Toc57754691)

# Лістинги програми з директивами внесення, редагування та вилучення даних у базі даних та результати виконання цих директив

## Функції внесення даних

Для кожної таблиці створений окремий метод, що приймає дані, що потрібні для внесення, і додає нові рядки у відповідну таблицю.

Функція внесення даних до таблиці Client

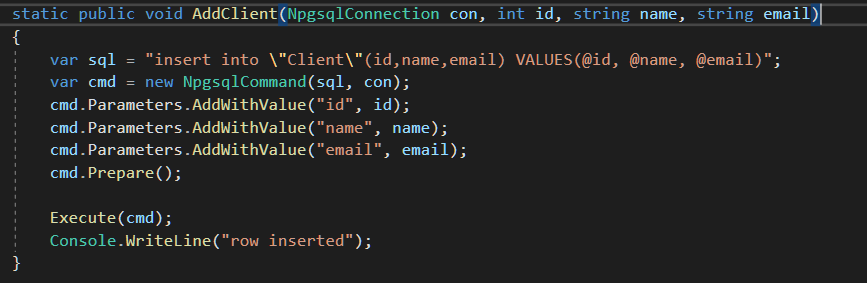
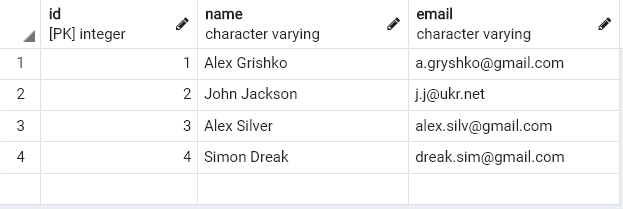
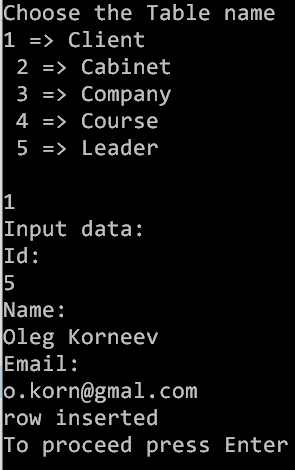


Рис 9. – Функція додавання даних до таблиці Client

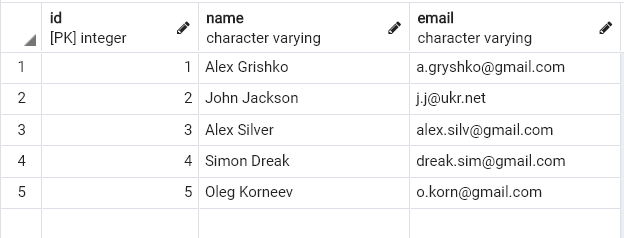
Початкова таблиця:



Запит у меню:



Таблиця після запиту:



Функція внесення даних до таблиці Course

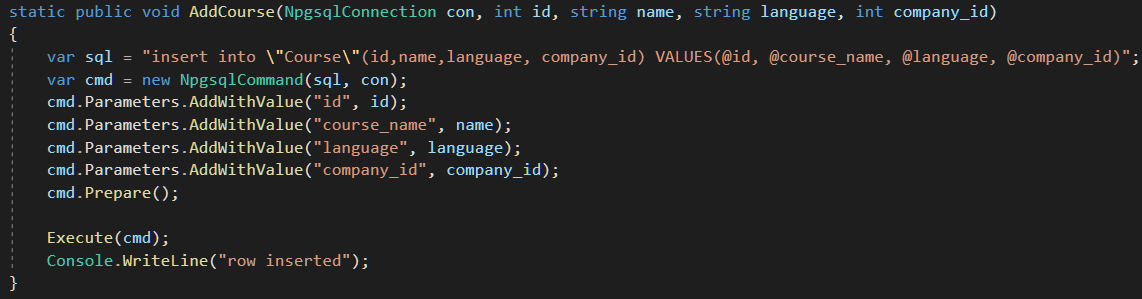
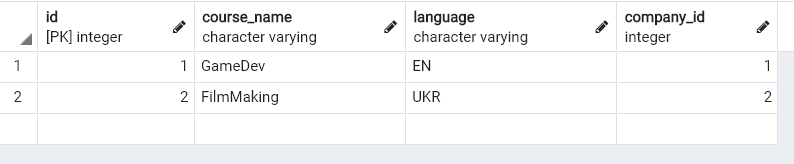
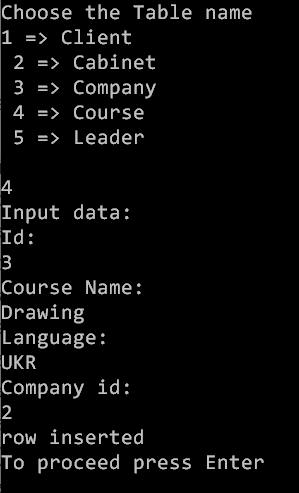


Рис 11. – Функція додавання даних до таблиці Course

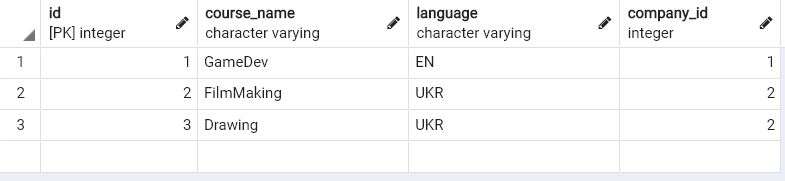
Початкова таблиця:



Запит у меню:



Таблиця після запиту:



Функція внесення даних до таблиці Company

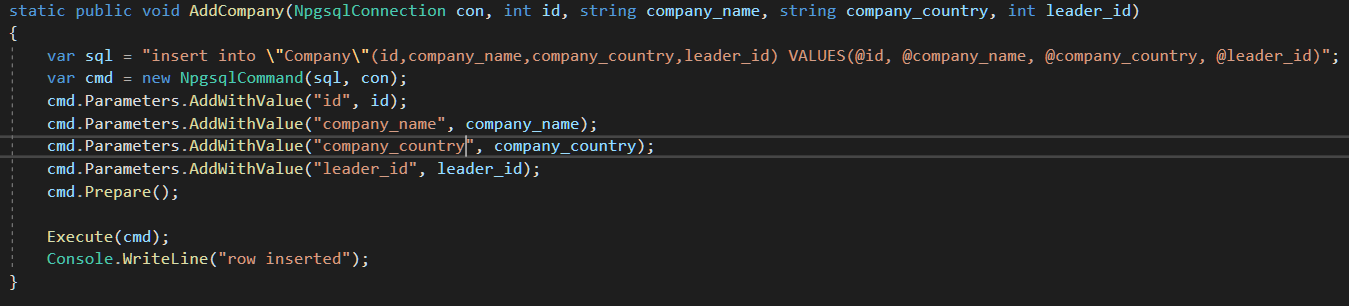
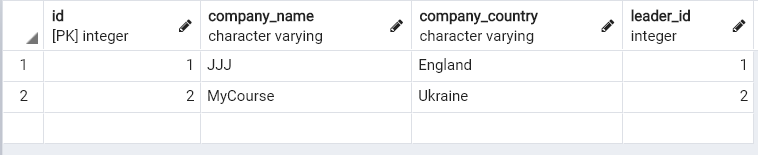


Рис 12. – Функція додавання даних до таблиці Company

Початкова таблиця:



Запит у меню:



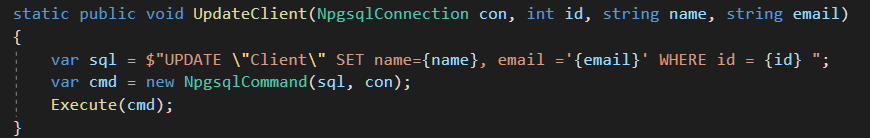
Таблиця після запиту:



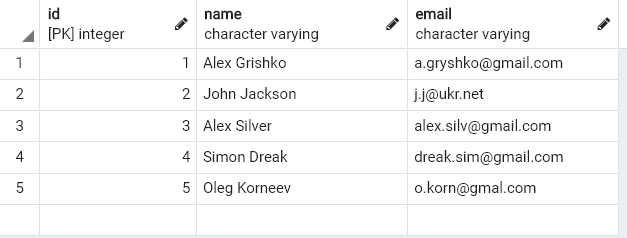
## Функції оновлення даних

Для кожної таблиці створений окремий метод, що приймає дані, що потрібні для внесення, і вносить зміни до бази даних.

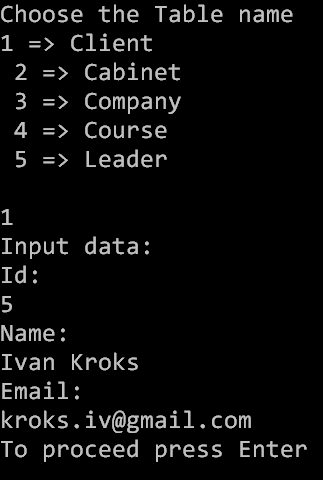
Функція для оновлення даних до таблиці Client

Рис 13. – Функція оновлення даних до таблиці Client

Початкова таблиця:



Запит у меню:



Таблиця після запиту:



Функція для оновлення даних до таблиці Course

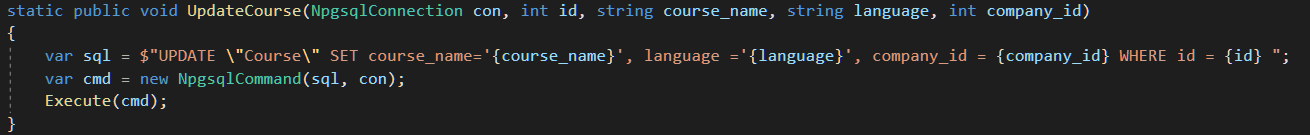
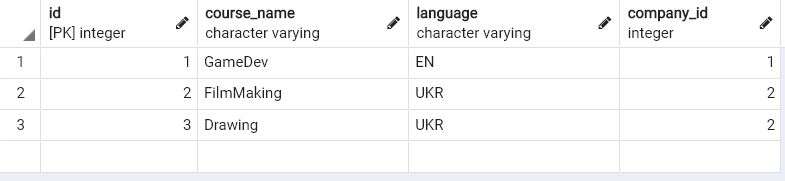
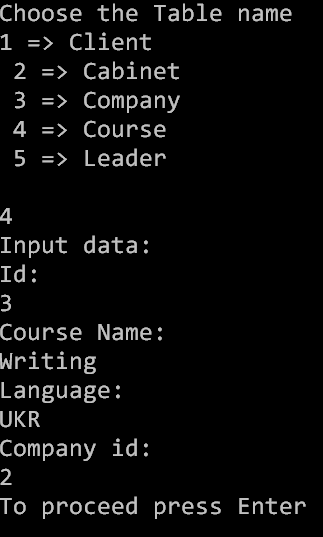


Рис 14. – Функція оновлення даних до таблиці Course

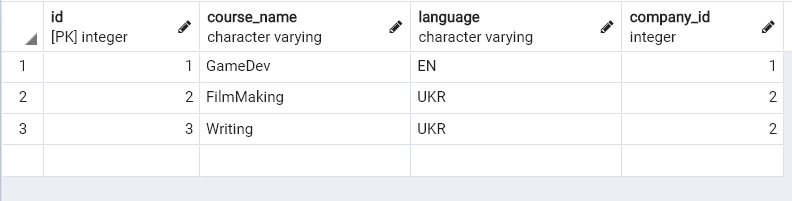
Початкова таблиця:



Запит у меню:



Таблиця після запиту:



Функція для оновлення даних до таблиці Company

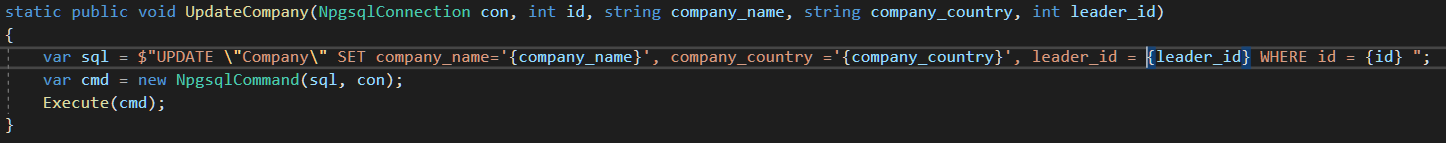
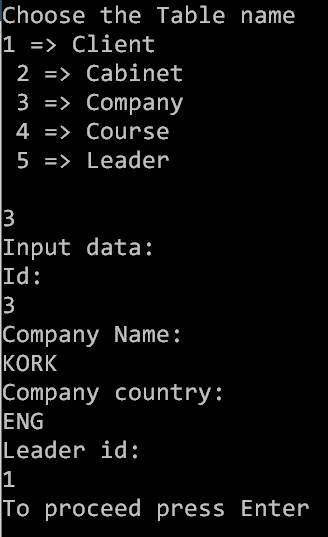


Рис 15. – Функція оновлення даних до таблиці Company

Початкова таблиця:



Запит у меню:



Таблиця після запиту:



## Функції вилучення даних

Для кожної таблиці створений окремий метод, що приймає дані, що потрібні для видалення (id), і вносить зміни до бази даних.

Функція для вилучення даних з таблиці Client, Course, Company відповідно.

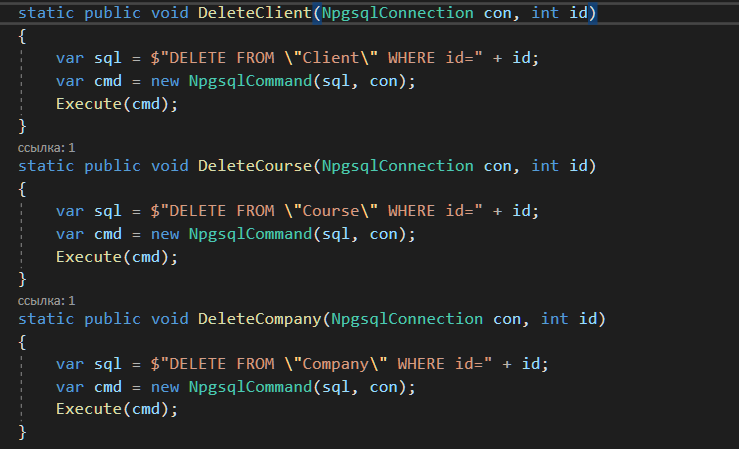
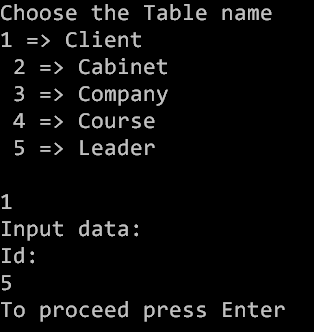


Рис 16. – Вилучення даних

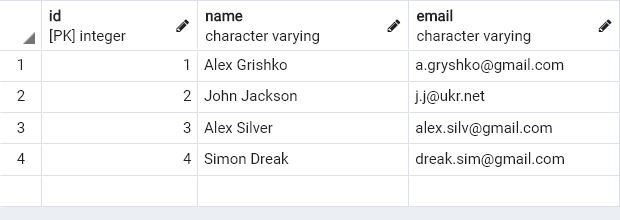
Початкова таблиця Client:



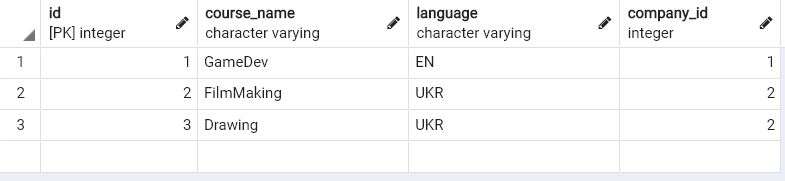
Запит у меню:



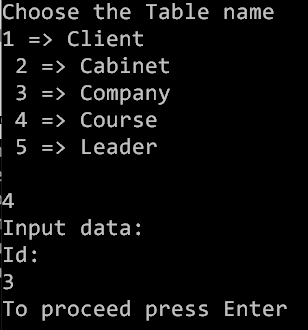
Таблиця Client після запиту:



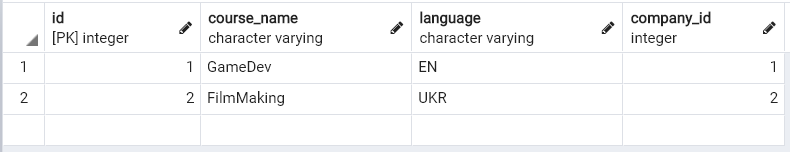
Початкова таблиця Course:



Запит у меню:



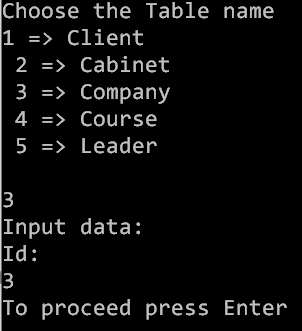
Таблиця Course після запиту:



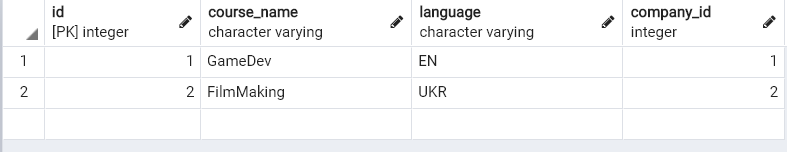
Початкова таблиця Company:



Запит у меню:



Таблиця Company після запиту:

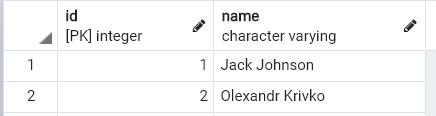


# Лістинги програми з директивами внесення рандомізованих даних і виконання динамічних запитів у базі даних та результати виконання цих директив

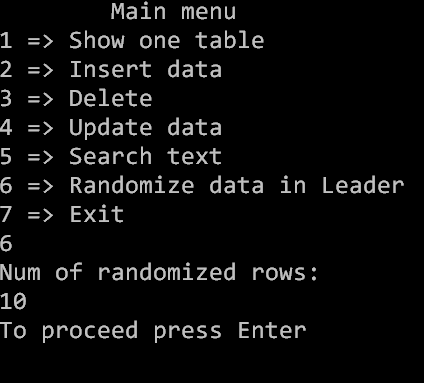
**Рандомізоване внесення даних до таблиці “Leader”**

Рис 17. – Рандомізування даних

Початкова таблиця



Запит у меню

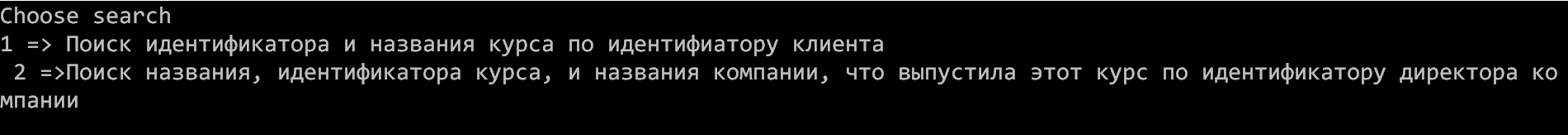


Таблиця після запиту:



# Виконання динамічних запитів бази даних

В даній програмі визначені типи динамічних запитів, які визначені в меню:

Рис 17. – Динамічні запити

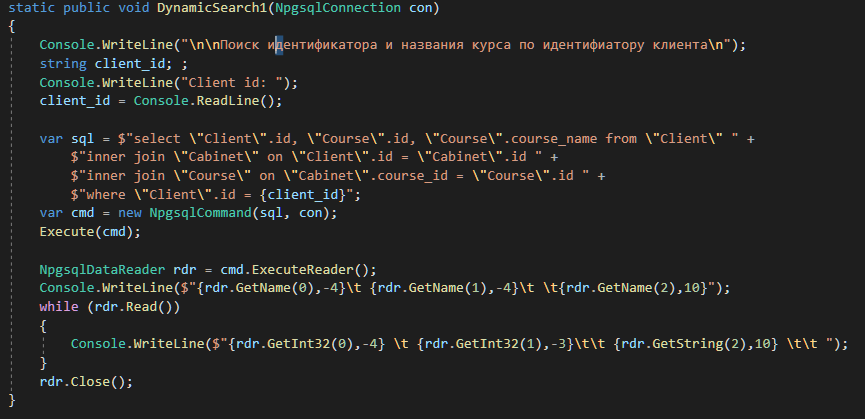
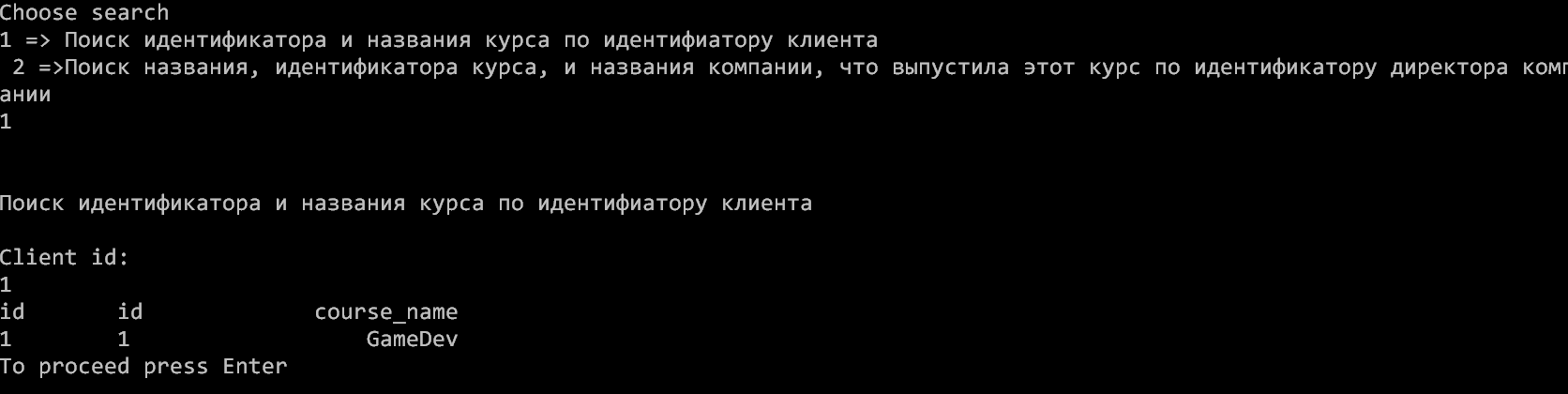


Рис 18. – Динамічний запити №1

Запит в меню:



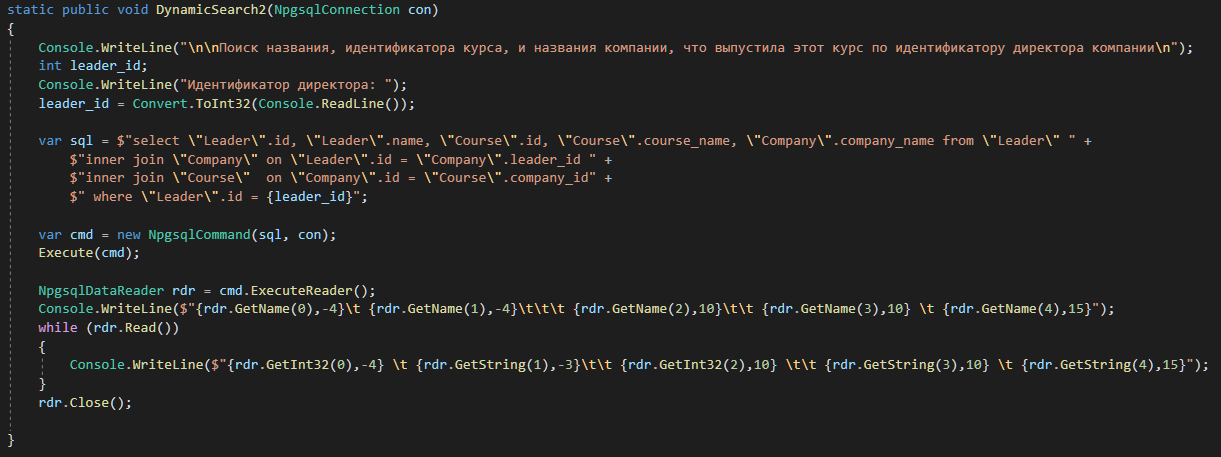
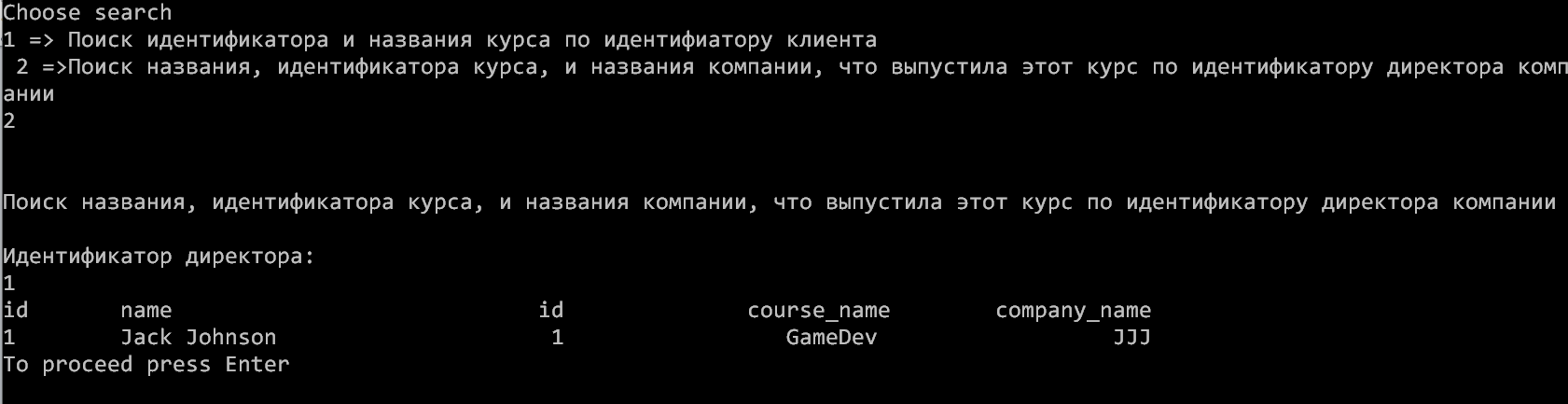


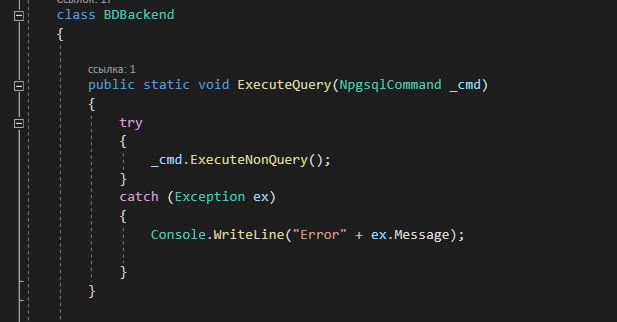
Рис 19. – Динамічний запити №1

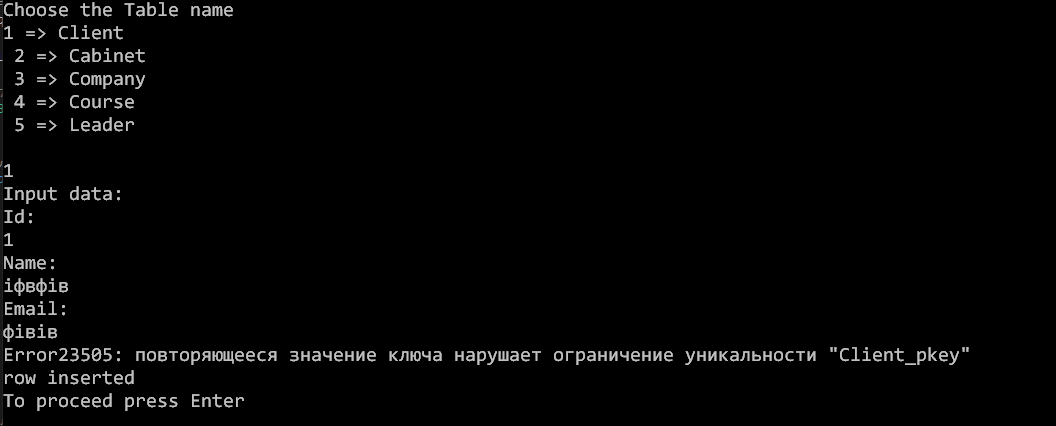
Запит в меню:



# Обробка виняткових ситуацій (помилок) при введенні/вилученні та валідації даних

Обробка виняткових при виконанні будь-якого запиту виконується за допомогою блоку і механізму try-catch у модулі BDBackend. При введенні помилкових даних помилка серверу SQL не зупинить роботу програми, а в меню користувача з’явиться відповідне повідомлення про помилку.

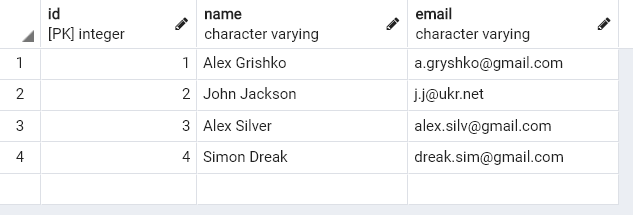


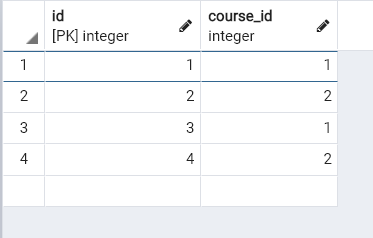
Приклади перевірки на валідність вхідних даних:

# Дослідження режимів обмеження ON DELETE

Режими обмеження ми будемо досліджувати на прикладі таблиць

Початкові таблиці

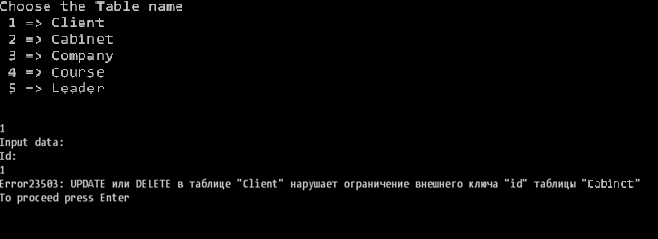




Режим **NO ACTION**, **SET NULL** (за умови що id має обмеження NOT NULL), **SET DEFAULT**, **RESTRICT**

NO ACTION

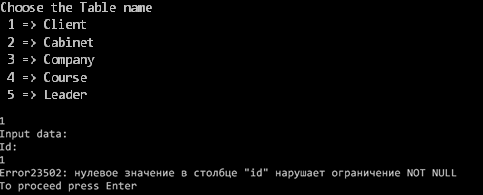
Запит меню



При видаленні запису з батьківської таблиці ,ми можемо побачити помилку щодо порушення зовнішнього ключа.

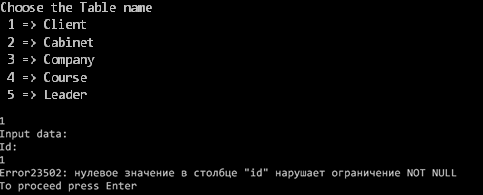
SET NULL

Запит меню



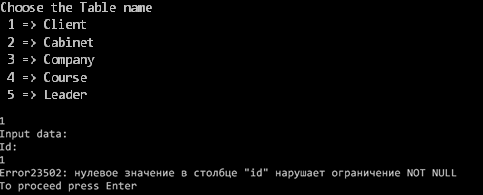
SET DEFAULT

Запит меню



RESTRICT

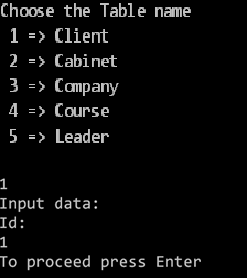
Запит меню



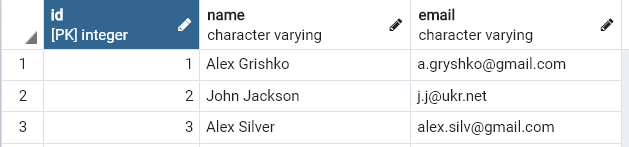
Кожний з вказаних режимів повертає одну й ту саму помилку, через особливості роботи кожного з них. Наприклад, **SET NULL** та **RESTRICT** ,не дозволяє видалити запис, через обмеження NOT NULL у полях запису, **NO ACTION** та **RESTRICT** просто забороняють видалення значень з батьківської таблиці, при наявності зовнішніх залежностей.

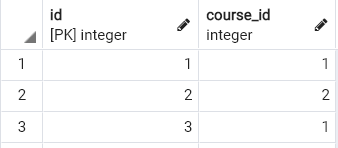
CASCADE

Запит меню



Таблиці після запиту





# Ілюстрації програмного коду на Github