



НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ  
СІКОРСЬКОГО»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

**Кафедра системного програмування та спеціалізованих  
комп'ютерних систем**

**Розрахунково-графічна робота**

з дисципліни **Бази даних і засоби управління**

*на тему: “ Створення додатку бази даних, орієнтованого на взаємодію з  
СУБД PostgreSQL ”*

Виконав: студент III курсу

ФПМ групи КВ-23

Глухоман Андрій

Перевірив:

Київ – 2024

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 двох та більше сутностей одночасно: для числових атрибутів – у рамках діапазону, для рядкових – як шаблон функції LIKE оператора SELECT SQL, для логічного типу – значення True/False, для дат – у рамках діапазону дат.
4. Програмний код виконати згідно шаблону MVC (модель-подання-контролер).

## Виконання роботи

Логічну модель (схему бази даних) наведено на рисунку 1.

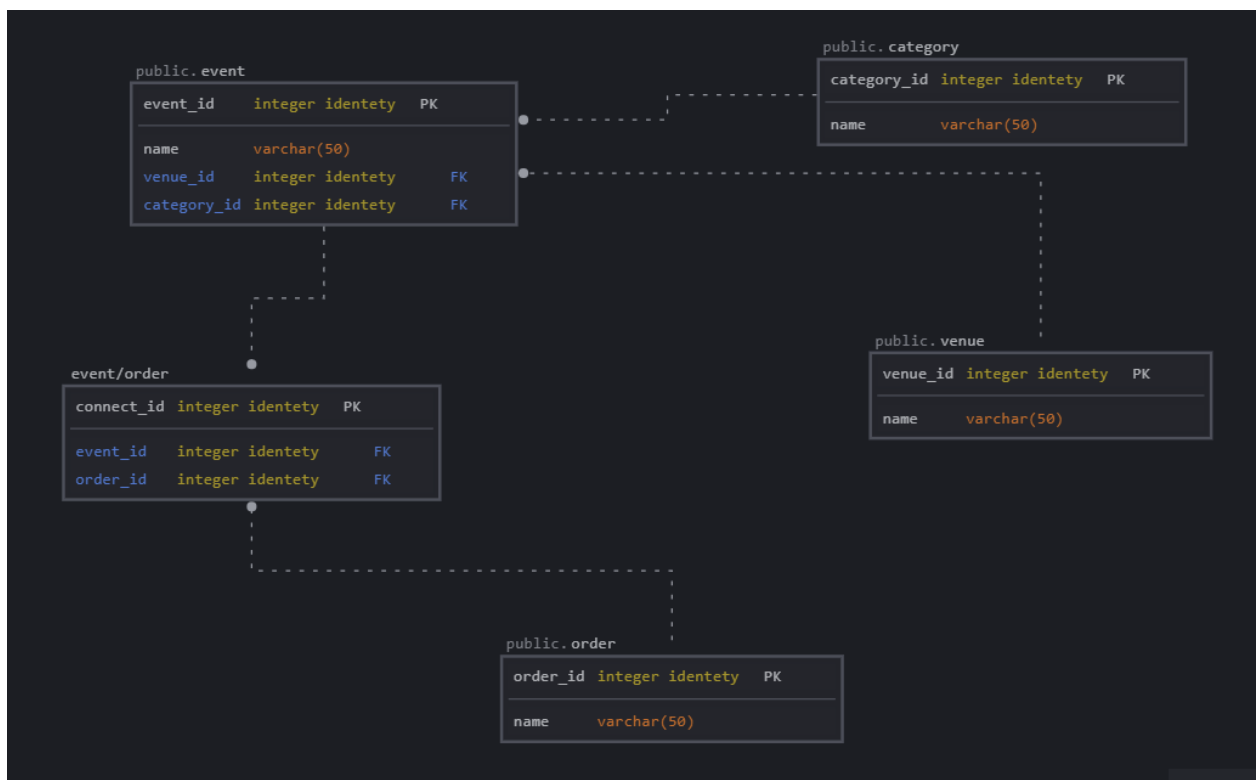


Рисунок 1 -Логічна модель бази даних

Зміни у порівнянні з першою лабораторною роботою відсутні.

## Середовище та компоненти розробки

Для розробки використовувалась мова програмування C#, середовище розробки Visual Studio, а також стороння бібліотека, що надає API для доступу до PostgreSQL – Npgsql.

## Шаблон проектування

MVC - шаблон проектування, який використаний у програмі.

Model – представляє клас, що описує логіку використовуваних даних. Згідно компоненту моделі, у моїй програмі відповідають всі компоненти які знаходяться у папці Models.

View – в нашому випадку консольний інтерфейс з яким буде взаємодіяти наш користувач. Згідно компоненту представлення, то їй відповідають такі компоненти, згідно яким користувач бачить необхідні дані, що є представленням даних у вигляді консольного інтерфейсу.

Controller – представляє клас, що забезпечує зв'язок між користувачем і системою, поданням і сховищем даних. Він отримує вводяться користувачем дані і обробляє їх. І в залежності від результатів обробки відправляє користувачеві певний висновок, наприклад, у вигляді подання.

## Структура програми та її опис

Програма умовно розділена на 4 модулі: файл DatabaseController.cs, файл DatabaseModel.cs та файл DatabaseView.cs та головний файл Program.cs

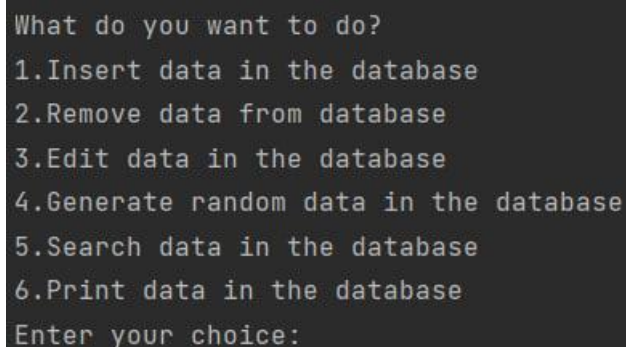
Класи, як видно з їх назв, повністю відповідають використаному патерну MVC.

У файлі DatabaseModel описаний клас моделі, що займається регулюванням підключення до бази даних, та виконанням запитів до неї.

У файлі DatabaseController описаний інтерфейс взаємодії з користувачем, запит бажаної дії, виконання пошуку, тощо.

У файлі DatabaseView описаний клас, що виводить результати виконання тієї чи іншої дії на екран консолі.

## Структура меню програми



```
What do you want to do?  
1.Insert data in the database  
2.Remove data from database  
3.Edit data in the database  
4.Generate random data in the database  
5.Search data in the database  
6.Print data in the database  
Enter your choice:
```

Рисунок 3 - Меню програми

Меню користувача складається з шести пунктів (Рисунок 3).

Перший пункт пропонує введення даних в таблицю

Другий пункт пропонує видалення даних з таблиці

Третій пункт пропонує оновлення даних в таблиці

Четвертий пункт пропонує генерування даних в таблиці

П'ятий пункт пропонує пошук даних у таблиці

Шостий пункт пропонує отримання імен та типів стовпчиків таблиці

## **Фрагменти програм внесення, редагування та видалення даних у базі даних**

Фрагмент програми для введення даних в таблицю:

```
public void Insert() {
    List<string> tables = GetTables("insert");
    int selected_index = int.Parse(Console.ReadLine()) - 1;
    List<string> columns = GetColumns(tables[selected_index]);
    List<string> values = new List<string>();
    foreach (string column in columns) {
        Console.WriteLine("Enter value for column - " + column + " - ");
        values.Add(Console.ReadLine());
    }
    string insert_query = "INSERT INTO " + tables[selected_index] + " (" +
ListToString(columns, false) + ") VALUES (" + ListToString(values, true) + ")";
    NpgsqlCommand insert_command = new NpgsqlCommand(insert_query, connection);
    Console.WriteLine(insert_query);
    insert_command.ExecuteNonQuery();
    Console.WriteLine("Data succesfull added!");
}
```

Фрагмент програми для видалення даних з таблиці:

```
public void Delete() {
    List<string> tables = GetTables("delete");
    int selected_index = int.Parse(Console.ReadLine()) - 1;
    Console.WriteLine("Input row id for delete");
    int row_id = int.Parse(Console.ReadLine());
    NpgsqlCommand delete_command = new NpgsqlCommand("DELETE FROM " +
tables[selected_index] + " WHERE " + tables[selected_index] + "_id = " + row_id,
connection);
    delete_command.ExecuteNonQuery();
    Console.WriteLine("Data succesfull deleted!");
}
```

Фрагмент програми для оновлення даних в таблиці:

```
public void Update() {
    List<string> tables = GetTables("update");
    int selected_index = int.Parse(Console.ReadLine()) - 1;
    List<string> columns = GetColumns(tables[selected_index]);
    List<string> values = new List<string>();
    Console.WriteLine("Input row id for update");
    int row_id = int.Parse(Console.ReadLine());
    foreach (string column in columns) {
        Console.WriteLine("Enter new value for column - " + column + " - ");
        values.Add(Console.ReadLine());
    }
    string update_query = "UPDATE " + tables[selected_index] + " SET " +
UpdatePartialString(columns, values) + " WHERE " + tables[selected_index] + "_id = " +
row_id;
    NpgsqlCommand update_command = new NpgsqlCommand(update_query, connection);
    update_command.ExecuteNonQuery();
    Console.WriteLine("Data succesfull updated! " + update_query);
}
```

```
}
```

Фрагмент програми для генерування даних в таблиці:

```
public void Generate() {
    List<string> tables = GetTables("generate");
    int selected_index = int.Parse(Console.ReadLine()) - 1;
    Console.WriteLine("Input rows count for generate");
    int rows_count = int.Parse(Console.ReadLine());
    List<string> columns = GetColumns(tables[selected_index]);
    Random random = new Random();
    for (int i = 0; i < rows_count; i++) {
        while (true) {
            try {
                List<string> values = new List<string>();
                foreach (string column in columns) {
                    values.Add(random.Next(0, 10).ToString());
                }
                //values[values.Count - 1] = "1";
                string insert_query = "INSERT INTO " + tables[selected_index] + " ("
+ ListToString(columns, false) + ") VALUES (" + ListToString(values, true) + ")";
                NpgsqlCommand insert_command = new NpgsqlCommand(insert_query,
connection);

                insert_command.ExecuteNonQuery();
                break;
            }
            catch (Exception ex) {
                //Console.WriteLine(ex.Message);
            }
        }
    }
    Console.WriteLine("Generated was succesfull!");
}
```

Фрагмент програми для пошуку даних у таблиці:

```
public void Search() {
    Console.WriteLine("Choose what to search:");
    Console.WriteLine("1 - select event by venue_id");
    Console.WriteLine("2 - select event by category_id");
    Console.WriteLine("3 - select name by venue_id");
    Console.Write("Your choice - ");
    int search = int.Parse(Console.ReadLine());
    switch (search) {
        case 1:
            Console.Write("Input venue_id - ");
            string venue_id = Console.ReadLine();
            NpgsqlCommand command1 = new NpgsqlCommand("SELECT name FROM category
WHERE department_id = " + venue_id, connection);
            NpgsqlDataReader reader1 = command1.ExecuteReader();
            List<string> data1 = new List<string>();
            while (reader1.Read()) {
                data1.Add(reader1.GetValue(0).ToString());
            }
            PrintRow(data1);
            reader1.Close();
            break;
        case 2:
            Console.Write("Input category_id - ");
            string category_id = Console.ReadLine();
            NpgsqlCommand command2 = new NpgsqlCommand("SELECT name FROM event WHERE
category_id = " + category_id, connection);
            NpgsqlDataReader reader2 = command2.ExecuteReader();
            List<string> data2 = new List<string>();
            while (reader2.Read()) {
                data2.Add(reader2.GetValue(0).ToString());
            }
            PrintRow(data2);
            reader2.Close();
        }
    }
}
```

```

        break;
    case 3:
        Console.WriteLine("Input venue_id - ");
        string venue_id3 = Console.ReadLine();
        NpgsqlCommand command3 = new NpgsqlCommand("SELECT name FROM department
WHERE department_id = " + venue_id3, connection);
        NpgsqlDataReader reader3 = command3.ExecuteReader();
        List<string> data3 = new List<string>();
        while (reader3.Read()) {
            data3.Add(reader3.GetValue(0).ToString());
        }
        PrintRow(data3);
        reader3.Close();
        break;
    }
}

```

Фрагмент програми для отримання імен та типів стовпчиків таблиці:



```

public void Print() {
    List<string> tables = GetTables("print");
    int selected_index = int.Parse(Console.ReadLine()) - 1;
    List<string> columns = GetColumns(tables[selected_index]);
    PrintRow(columns);
    NpgsqlCommand command = new NpgsqlCommand("SELECT * FROM " +
tables[selected_index], connection);
    NpgsqlDataReader reader = command.ExecuteReader();
    List<string> data = new List<string>();
    while (reader.Read()) {
        for (int i = 0; i < columns.Count; i++) {
            data.Add(reader.GetValue(i).ToString());
        }
        PrintRow(data);
        data.Clear();
    }
    reader.Close();
}

```

Дані фрагменти програми, які наведені вище, відповідають за функціонал додання даних, редагування та видалення даних у базі даних. Взаємодія відбувається через клас Model, який займається підключенням до БД, а самі функції знаходяться у файлі Controller.

### Скріншоти результатів виконання операції вставки запису в дочірню таблицю

	category_id [PK] integer 	name character varying (50) 
1	1	Football
2	2	Theater
3	3	Concert



```

What do you want to do?
1.Insert data in the database
2.Remove data from database
3.Edit data in the database
4.Generate random data in the database
5.Search data in the database
6.Print data in the database
Enter your choice: 1



Choose table to insert data
1.event
2.category
3.venue
4.order
5.event/order
2

Enter value for column - category_id - 4
Enter value for column - name - cinema
INSERT INTO category (name) VALUES ('cinema')
Data succesfull added!

```

	category_id [PK] integer 	name character varying (50) 
1	1	Football
2	2	Theater
3	3	NEW concert
4	4	cinema

**Скріншоти результатів виконання операції видалення даних з таблиці**

	category_id [PK] integer 	name character varying (50) 
1	1	Football
2	2	Theater
3	3	NEW concert
4	4	cinema

```

What do you want to do?
1.Insert data in the database
2.Remove data from database
3.Edit data in the database
4.Generate random data in the database
5.Search data in the database
6.Print data in the database
Enter your choice: 2

```

```

Choose table to delete data

```

```

1.event
2.category
3.venue
4.order
5.event/order
2

```

```



Input row id for delete

```

```

4
Data succesfull deleted!

```

	category_id [PK] integer 	name character varying (50) 
1	1	Football
2	2	Theater
3	3	Concert



## Скріншоти результатів виконання операції оновлення даних в таблиці

	category_id [PK] integer	name character varying (50)
1	1	Football
2	2	Theater
3	3	Concert

```
What do you want to do?
1.Insert data in the database
2.Remove data from database
3.Edit data in the database
4.Generate random data in the database
5.Search data in the database
6.Print data in the database
Enter your choice: 3

Choose table to update data
1.event
2.category
3.venue
4.order
5.event/order
2

Input row id for update
3

Enter new value for column - category_id - 3
Enter new value for column - name - NEW concert
Data succesfull updated! UPDATE category SET name = 'NEW concert' WHERE category_id = 3
```

	category_id [PK] integer	name character varying (50)
1	1	Football
2	2	Theater
3	3	NEW concert

## Скріншоти результатів виконання операції генерування даних в таблиці

	venue_id [PK] integer	name character varying (50)
1	1	NSK Olimpic
2	2	Frank's theater
3	3	Sports palace
4	4	Donbas arena

```

What do you want to do?
1.Insert data in the database
2.Remove data from database
3.Edit data in the database
4.Generate random data in the database
5.Search data in the database
6.Print data in the database
Enter your choice: 4

Choosetable to generate data
1.event
2.category
3.venue
4.order
5.event/order
3

Input row id for generate
3
Generated was succesfull!

```

	venue_id [PK] integer	name character varying (50)
1	1	NSK Olimpic
2	2	Frank's theater
3	3	Sports palace
4	4	Donbas arena
5	8	7
6	14	9
7	17	0

**Скріншоти результатів виконання операції пошуку даних у таблиці**

```
What do you want to do?
1.Insert data in the database
2.Remove data from database
3.Edit data in the database
4.Generate random data in the database
5.Search data in the database
6.Print data in the database
Enter your choice: 5

Choose what to search
1 - select event by venue_id
2 - select event by category_id
3 - select name by venue_id
Your choice - 3

Input venue_id - 1
NSK Olympic
```

**Скріншоти результатів виконання операції отримання імен та типів стовпчиків таблиці**

```
What do you want to do?
1.Insert data in the database
2.Remove data from database
3.Edit data in the database
4.Generate random data in the database
5.Search data in the database
6.Print data in the database
Enter your choice: 6

Choose table to print data
1.event
2.category
3.venue
4.order
5.event/order
2

category_id      name
1.Football
2.Theater
3.NEW concert
```

## Текст програми:

### *Program.cs*

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Npgsql;

namespace PostgreConsoleInteractorCS {
    class Program {
        static void Main(string[] args) {
            DatabaseView view = new DatabaseView();
            view.Process();
        }
    }
}
```

### *DatabaseView.cs*

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace PostgreConsoleInteractorCS {
    public class DatabaseView {
        DatabaseController controller;

        public void Process() {
            string connection_string = "Server=localhost;Port=5432;User
ID=postgres;Password=123;Database=shop;";
            controller = new DatabaseController(connection_string);
            while (true) {
                int action = controller.ShowActionList();
                controller.Execute(action);
            }
        }
    }
}
```

### *DatabaseController.cs*

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Npgsql;

namespace PostgreConsoleInteractorCS {
    public class DatabaseController {
        DatabaseModel db = null;
    }
}
```



```

        Console.WriteLine("Connection was opened succesfull!");
    }

    public void Insert() {
        List<string> tables = GetTables("insert");
        int selected_index = int.Parse(Console.ReadLine()) - 1;
        List<string> columns = GetColumns(tables[selected_index]);
        List<string> values = new List<string>();
        foreach (string column in columns) {
            Console.Write("Enter value for column - " + column + " - ");
            values.Add(Console.ReadLine());
        }
        string insert_query = "INSERT INTO " + tables[selected_index] + " (" +
ListToString(columns, false) + ") VALUES (" + ListToString(values, true) + ")";
        NpgsqlCommand insert_command = new NpgsqlCommand(insert_query, connection);
        Console.WriteLine(insert_query);
        insert_command.ExecuteNonQuery();
        Console.WriteLine("Data succesfull added!");
    }

    public void Delete() {
        List<string> tables = GetTables("delete");
        int selected_index = int.Parse(Console.ReadLine()) - 1;
        Console.WriteLine("Input row id for delete");
        int row_id = int.Parse(Console.ReadLine());
        NpgsqlCommand delete_command = new NpgsqlCommand("DELETE FROM " +
tables[selected_index] + " WHERE " + tables[selected_index] + "_id = " + row_id,
connection);
        delete_command.ExecuteNonQuery();
        Console.WriteLine("Data succesfull deleted!");
    }

    public void Update() {
        List<string> tables = GetTables("update");
        int selected_index = int.Parse(Console.ReadLine()) - 1;
        List<string> columns = GetColumns(tables[selected_index]);
        List<string> values = new List<string>();
        Console.WriteLine("Input row id for update");
        int row_id = int.Parse(Console.ReadLine());
        foreach (string column in columns) {
            Console.Write("Enter new value for column - " + column + " - ");
            values.Add(Console.ReadLine());
        }
        string update_query = "UPDATE " + tables[selected_index] + " SET " +
UpdatePartialString(columns, values) + " WHERE " + tables[selected_index] + "_id = " +
row_id;
        NpgsqlCommand update_command = new NpgsqlCommand(update_query, connection);
        update_command.ExecuteNonQuery();
        Console.WriteLine("Data succesfull updated! " + update_query);
    }

    public void Generate() {
        List<string> tables = GetTables("generate");
        int selected_index = int.Parse(Console.ReadLine()) - 1;
        Console.WriteLine("Input rows count for generate");
        int rows_count = int.Parse(Console.ReadLine());
        List<string> columns = GetColumns(tables[selected_index]);
        Random random = new Random();
        for (int i = 0; i < rows_count; i++) {
            while (true) {
                try {
                    List<string> values = new List<string>();
                    foreach (string column in columns) {
                        values.Add(random.Next(0, 10).ToString());
                    }
                    //values[values.Count - 1] = "1";
                    string insert_query = "INSERT INTO " + tables[selected_index] + " ("
+ ListToString(columns, false) + ") VALUES (" + ListToString(values, true) + ")";

```

```

        NpgsqlCommand insert_command = new NpgsqlCommand(insert_query,
connection);

        insert_command.ExecuteNonQuery();
        break;
    }
    catch (Exception ex) {
        //Console.WriteLine(ex.Message);
    }
}

Console.WriteLine("Generated was succesfull!");
}

public void Search() {
    Console.WriteLine("Choose what to search:");
    Console.WriteLine("1 - select event by venue_id");
    Console.WriteLine("2 - select event by category_id");
    Console.WriteLine("3 - select name by venue_id");
    Console.Write("Your choice - ");
    int search = int.Parse(Console.ReadLine());
    switch (search) {
        case 1:
            Console.Write("Input venue_id - ");
            string venue_id = Console.ReadLine();
            NpgsqlCommand command1 = new NpgsqlCommand("SELECT name FROM category
WHERE department_id = " + venue_id, connection);
            NpgsqlDataReader reader1 = command1.ExecuteReader();
            List<string> data1 = new List<string>();
            while (reader1.Read()) {
                data1.Add(reader1.GetValue(0).ToString());
            }
            PrintRow(data1);
            reader1.Close();
            break;
        case 2:
            Console.Write("Input category_id - ");
            string category_id = Console.ReadLine();
            NpgsqlCommand command2 = new NpgsqlCommand("SELECT name FROM event WHERE
category_id = " + category_id, connection);
            NpgsqlDataReader reader2 = command2.ExecuteReader();
            List<string> data2 = new List<string>();
            while (reader2.Read()) {
                data2.Add(reader2.GetValue(0).ToString());
            }
            PrintRow(data2);
            reader2.Close();
            break;
        case 3:
            Console.Write("Input venue_id - ");
            string venue_id3 = Console.ReadLine();
            NpgsqlCommand command3 = new NpgsqlCommand("SELECT name FROM department
WHERE department_id = " + venue_id3, connection);
            NpgsqlDataReader reader3 = command3.ExecuteReader();
            List<string> data3 = new List<string>();
            while (reader3.Read()) {
                data3.Add(reader3.GetValue(0).ToString());
            }
            PrintRow(data3);
            reader3.Close();
            break;
    }
}

public void Print() {
    List<string> tables = GetTables("print");
    int selected_index = int.Parse(Console.ReadLine()) - 1;
    List<string> columns = GetColumns(tables[selected_index]);
    PrintRow(columns);
}

```

```

        NpgsqlCommand command = new NpgsqlCommand("SELECT * FROM " +
tables[selected_index], connection);
        NpgsqlDataReader reader = command.ExecuteReader();
        List<string> data = new List<string>();
        while (reader.Read()) {
            for (int i = 0; i < columns.Count; i++) {
                data.Add(reader.GetValue(i).ToString());
            }
            PrintRow(data);
            data.Clear();
        }
        reader.Close();
    }

    private List<string> GetTables(string operation) {
        Console.WriteLine("Choose table to " + operation + " data");
        NpgsqlCommand command = new NpgsqlCommand("SELECT * FROM
information_schema.tables WHERE table_schema = 'public';", connection);
        NpgsqlDataReader reader = command.ExecuteReader();
        List<string> tables = new List<string>();
        while (reader.Read()) {
            tables.Add(reader.GetString(2));
        }
        reader.Close();
        int index = 1;
        foreach (string table in tables) {
            Console.WriteLine(index.ToString() + '.' + table);
            index++;
        }
        return tables;
    }

    private List<string> GetColumns(string table) {
        string query = "SELECT * FROM information_schema.columns WHERE table_schema =
'public' AND table_name = '" + table + "'";
        NpgsqlCommand command = new NpgsqlCommand(query, connection);
        NpgsqlDataReader reader = command.ExecuteReader();
        List<string> columns = new List<string>();
        while (reader.Read()) {
            columns.Add(reader.GetString(3));
        }
        reader.Close();
        return columns;
    }

    private string ListToString(List<string> list, bool is_value) {
        string result = string.Empty;
        for (int i = 1; i < list.Count; i++) {
            if (is_value) result += "'" + list[i] + "',";
            else result += list[i] + ',';
        }
        result = result.Remove(result.Length - 1, 1);
        return result;
    }

    private void PrintRow(List<string> list) {
        foreach (string s in list) {
            Console.Write(s + "    ");
        }
        Console.WriteLine('\n');
    }

    private string UpdatePartialString(List<string> columns, List<string> values) {
        string partial = string.Empty;
        for (int i = 1; i < columns.Count; i++) {
            partial += columns[i] + " = '" + values[i] + "'";
            if (i < columns.Count - 1) partial += ", ";
        }
    }

```



```
        return partial;  
    }  
}  
}
```

Посилання на репозиторій:

[https://github.com/AndriiHlukhoman/KS\\_Hlukhoman\\_Andrii](https://github.com/AndriiHlukhoman/KS_Hlukhoman_Andrii)