



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: “Проектування бази даних та ознайомлення з базовими
операціями СУБД PostgreSQL”*

Виконав: студент 3 курсу

ФПМ групи КВ-23

Акімов Олександр Олександрович

Перевірив: Петрашенко А.В.

GitHub repository: <https://github.com/ex4mple13/Databases>

Київ – 2024

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

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

Графічне представлення моделі «сутність-зв'язок»

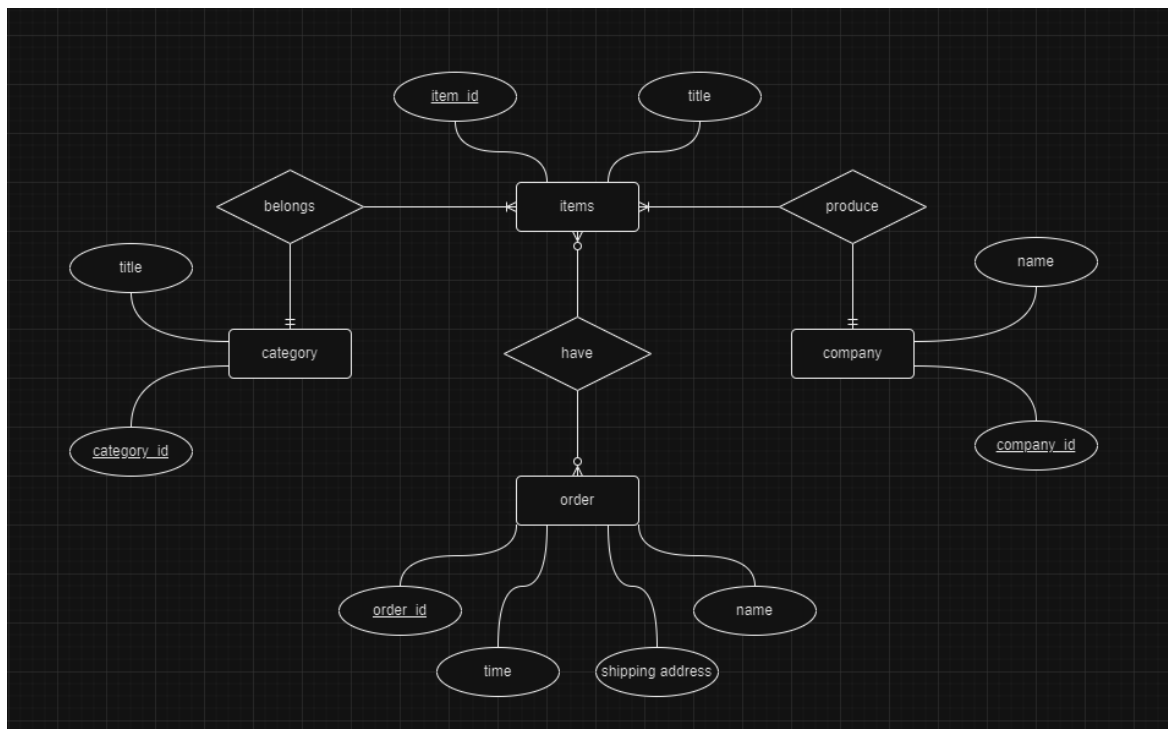


Рисунок 1 - ER-діаграма, побудована за нотацією Чена (draw.io)

Схема бази даних у графічному вигляді

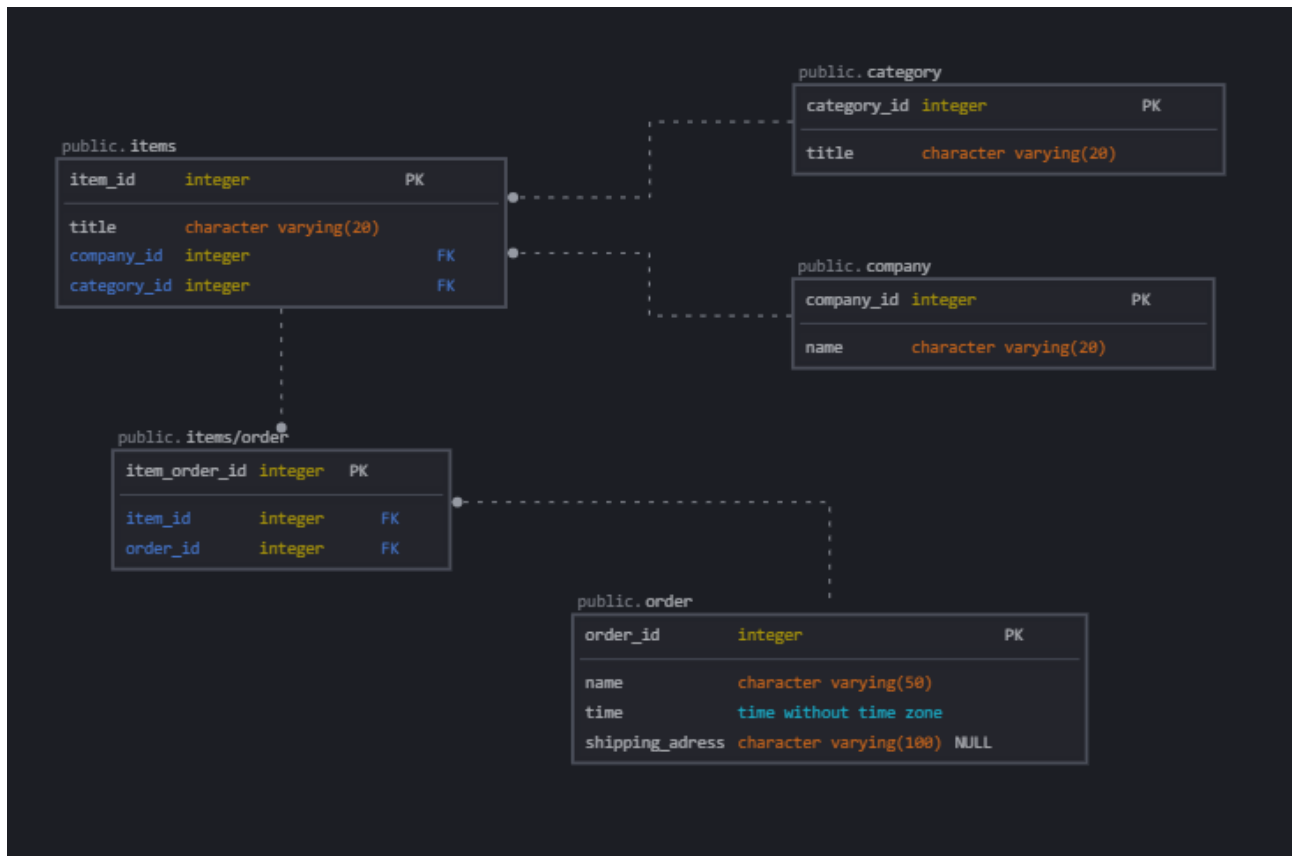


Рисунок 2 - Схема бази даних (інструмент: sqldbm.com)

Середовище та компоненти розробки

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

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: _
```

Меню користувача складається з шести пунктів.

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

Скріншоти результатів виконання операції вставки запису в дочірню таблицю

	category_id [PK] integer	title character varying (20)
1	1	phone
2	2	laptop
3	3	headphones

```
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.company
2.items
3.category
4.items/order
5.order
3
Enter value for column - category_id - 4
Enter value for column - title - New category
INSERT INTO category (title) VALUES ('New category')
```

	category_id [PK] integer	title character varying (20)
1	1	phone
2	2	laptop
3	3	headphones
4	4	New category



Скріншоти результатів виконання операції видалення даних з таблиці

	category_id [PK] integer 	title character varying (20) 
1	1	phone
2	2	laptop
3	3	headphones
4	4	New category



```

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.company
2.items
3.category
4.items/order
5.order
3
Input row id for delete
4

```

	category_id [PK] integer 	title character varying (20) 
1	1	phone
2	2	laptop
3	3	headphones



Скріншоти результатів виконання операції оновлення даних в таблиці

	category_id [PK] integer 	title character varying (20) 
1	1	phone
2	2	laptop
3	3	headphones



```

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.company
2.items
3.category
4.items/order
5.order
3
Input row id for update
2
Enter new value for column - category_id - 2
Enter new value for column - title - Update
UPDATE category SET title = 'Update' WHERE category_id = 2

```

	category_id [PK] integer 	title character varying (20) 
1	1	phone
2	2	Update
3	3	headphones

Скріншоти результатів виконання операції генерування даних в таблиці

	company_id [PK] integer 	name character varying (20) 
1	1	Apple
2	2	Asus
3	3	Razer

```

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
Choose table to generate data
1.company
2.items
3.category
4.items/order
5.order
1
Input rows count for generate
10

```

	company_id [PK] integer	name character varying (20)
1	1	Apple
2	2	Asus
3	3	Razer
4	7	9
5	11	108
6	15	2
7	17	32
8	23	57
9	47	13
10	78	94
11	102	122
12	135	83
13	141	62

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 name by company_id
2 - select items by company_id
3 - select items by category_id
Your choice - 3
Input category_id - 1
Iphone 15

```

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.company
2.items
3.category
4.items/order
5.order
2
item_id      title      company_id  category_id
3      Blackshark v2 pro      3      3
2      Zephyrus G16      2      2
1      Iphone 15      1      1
```

Програмний код модуля “DatabaseModel”

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

namespace PostgreConsoleInteractorCS
{
    public class DatabaseModel
    {
        NpgsqlConnection connection;
        public DatabaseModel(string connection_string)
        {
            connection = new NpgsqlConnection(connection_string);
            connection.Open();
        }

        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();
    }

    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();
    }

    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(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 { }
            }
        }
    }

```

```

    }
    catch (Exception ex)
    {
        Console.WriteLine(ex.Message);
    }
}
}

public void Search()
{
    Console.WriteLine("Choose what to search:");
    Console.WriteLine("1 - select name by company_id");
    Console.WriteLine("2 - select items by company_id");
    Console.WriteLine("3 - select items by category_id");
    Console.WriteLine("Your choice - ");
    int search = int.Parse(Console.ReadLine());
    switch (search)
    {
        case 1:
            Console.WriteLine("Input company_id - ");
            string trademark_id = Console.ReadLine();
            NpgsqlCommand command1 = new NpgsqlCommand("SELECT name FROM company WHERE company_id = " + trademark_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.WriteLine("Input company_id - ");
            string category_id = Console.ReadLine();
            NpgsqlCommand command2 = new NpgsqlCommand("SELECT title FROM items WHERE company_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 category_id - ");
            string category_id3 = Console.ReadLine();
            NpgsqlCommand command3 = new NpgsqlCommand("SELECT title FROM items WHERE category_id = " + category_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();
}

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;
}
}
}

```