



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
Ященко Іван Васильович
Перевірів: Павловський В.І.

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

Загальне завдання роботи полягає у наступному:

1. Реалізувати функції внесення, редагування та видалення 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 введення/редагування/видалення даних у таблицях бази даних з можливістю контролю відповідності типів даних атрибутів таблиць (рядків, чисел, дати/часу). Для контролю пропонується два варіанти:

- a. контроль при введенні - валідація даних;
- b. 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 **видалення (ON DELETE)** рядків за умови наявності даних у підлеглий 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.

2. 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.

3. 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 введення 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.

4. Програмний код організувати згідно шаблону Model-View-Controller (MVC). Приклад організації коду згідно шаблону доступний [за даним посиланням](#). 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. Суттєвих змін, в порівнянні з 1 Лабораторною роботою не відбулося, єдине що змінювалося це режими ON DELETE, ON UPDATE для дослідження в останньому пункті

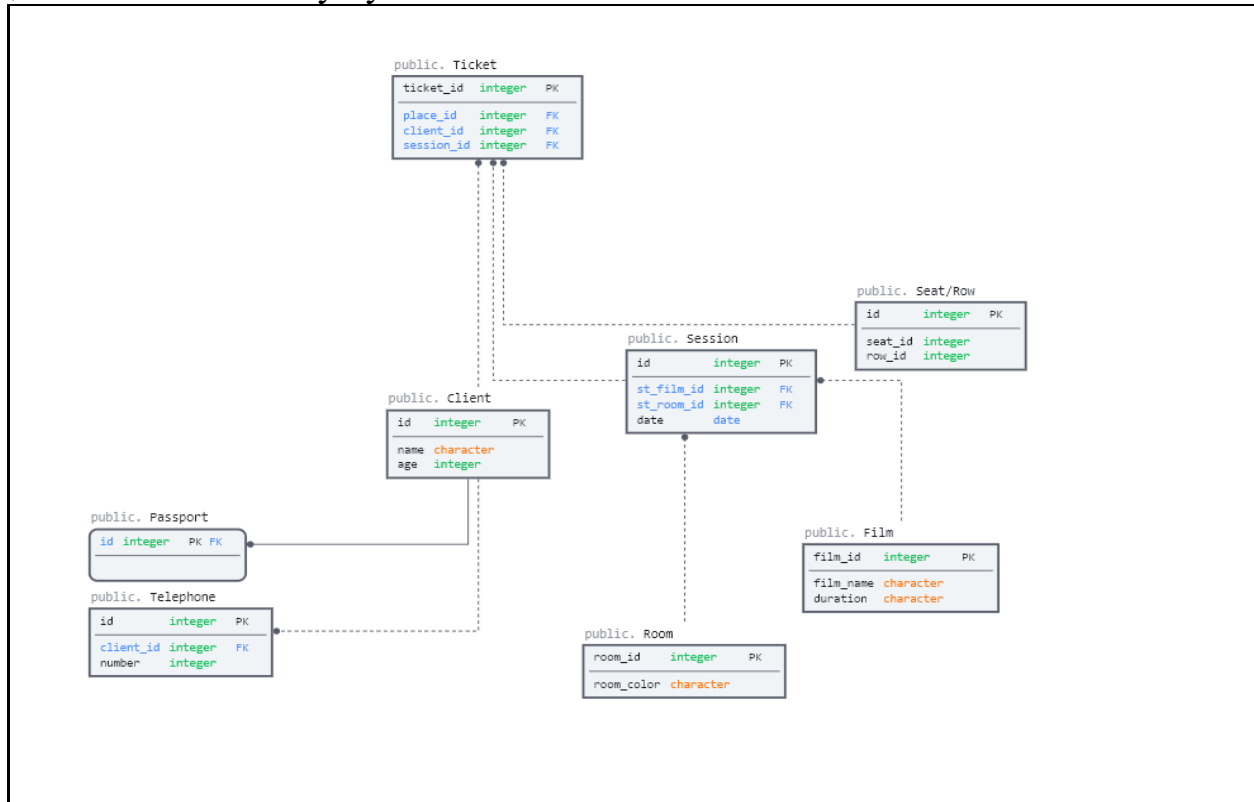


Рис. 1 – Логічна модель (Структура) БД “ Сервіс продажу квитків кіно ” (засобами SqlDMB)

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

A screenshot of a code editor showing C# code for connecting to a PostgreSQL database. The code is enclosed in a `static void Main(string[] args)` method. It defines a connection string `cs` with host, username, password, and database name. It then creates a `NpgsqlConnection` object, opens it, calls `Controller.Menu(con)`, and finally closes the connection.

```
static void Main(string[] args)
{
    var cs = "Host=127.0.0.1;Username=postgres;Password=qwerty;Database=Cinema";
    var con = new NpgsqlConnection(cs);
    con.Open();
    Controller.Menu(con);
    con.Close();
}
```

Рис 2. - Фотографія екрану, де описано підключення до БД засобами Npgsql

Структура програми

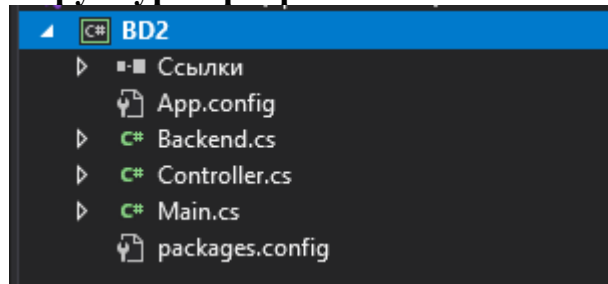
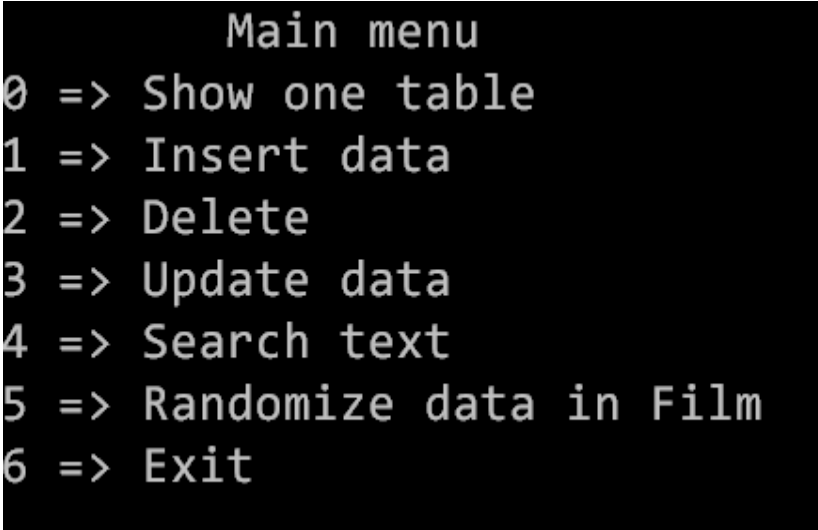


Рис 3. – Фотографія екрану, де описана структура програми

Опис структури програми. Програма містить 3 модуля, серед яких Backend – модуль, де описані усі взаємодії програми із базою даних (усі sql запити і її реалізація), Controller – модуль, де описана робота меню програми, Main – модуль, де розташована main функція, і де розпочинається робота програми.

Структура меню програми

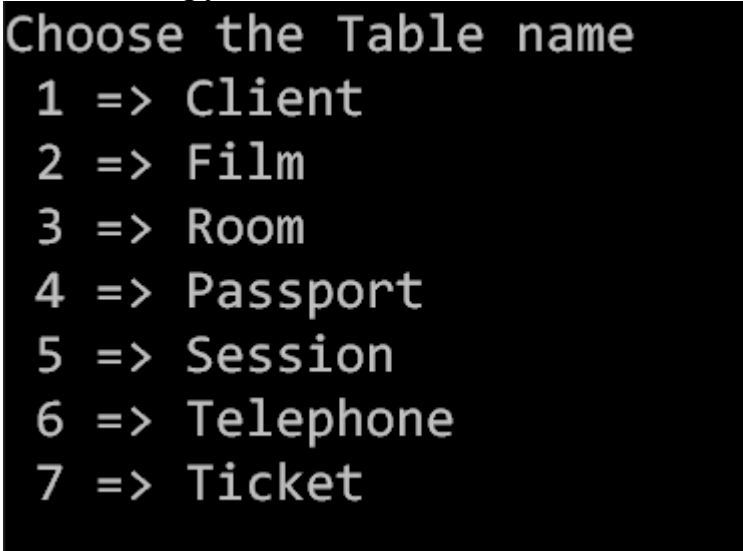
Головне меню



```
      Main menu
0 => Show one table
1 => Insert data
2 => Delete
3 => Update data
4 => Search text
5 => Randomize data in Film
6 => Exit
```

Рис 4. – Фотографія головного меню програми

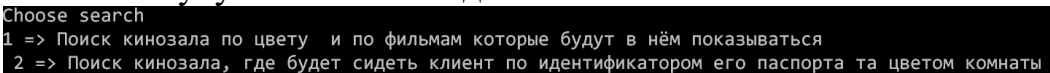
Меню вибору таблиць



```
Choose the Table name
1 => Client
2 => Film
3 => Room
4 => Passport
5 => Session
6 => Telephone
7 => Ticket
```

Рис 5. – Фотографія меню, яке з'являється якщо обрати пункти 0,1,2,3

Меню пошуку заготовлених динамічних запитів



```
Choose search
1 => Поиск кинозала по цвету и по фильмам которые будут в нём показываться
2 => Поиск кинозала, где будет сидеть клиент по идентификатором его паспорта та цветом комнаты
```

Рис 6. – Фотографія меню, яке з'являється якщо обрати пункт 4

Меню рандомізації даних

```
Main menu
0 => Show one table
1 => Insert data
2 => Delete
3 => Update data
4 => Search text
5 => Randomize data in Film
6 => Exit
5
Num of randomized rows:
```

Рис 7. – Фотографія меню, яке з'являється якщо обрати пункт 5

Загальна структура меню



Рис 8. – Структура меню програми

<u>Лістинги програми з директивами внесення, редагування та вилучення даних у базі даних та результати виконання цих директив</u>	8
<u>Функції для внесення даних</u>	8
<u>Функції для оновлення даних</u>	14
<u>Функції для вилучення даних</u>	18
<u>Лістинги програми з директивами внесення рандомізованих даних і виконання динамічних запитів у базі даних та результати виконання цих директив</u>	24
<u>Рандомізоване внесення даних до таблиці «Film»</u>	24
<u>Виконання динамічних запитів бази даних</u>	27
<u>Обробка виняткових ситуацій (помилки) при введенні/вилученні та валідації даних</u>	29
<u>Приклади перевірки на валідність вхідних даних</u>	29
<u>Дослідження режимів обмеження ON DELETE</u>	29
<u>Ілюстрації програмного коду на Github</u>	33

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 та Passport




```
static public void AddClient(NpgsqlConnection con, int id, int age, string name)//add client and passport
{
    var sql = "insert into \"Client\"(id,age,name) VALUES(@id, @age, @name)";
    var cmd = new NpgsqlCommand(sql, con);
    cmd.Parameters.AddWithValue("id", id);
    cmd.Parameters.AddWithValue("age", age);
    cmd.Parameters.AddWithValue("name", name);
    cmd.Prepare();

    Execute(cmd);
    sql = "insert into \"Passport\"(id) VALUES(@id)";
    cmd = new NpgsqlCommand(sql, con);
    cmd.Parameters.AddWithValue("id", id);
    cmd.Prepare();

    Execute(cmd);
    Console.WriteLine("row inserted");
}
```

Рис 9. – Функція додавання даних до таблиці Client та Passport в файлі Backend.cs

Початкова таблиця:

	 id [PK] integer	 age integer	 name character varying
1	0	54	Ivan
2	1	55	Koban

Запит у меню:





```

Choose the Table name
1 => Client
2 => Film
3 => Room
4 => Passport
5 => Session
6 => Telephone
7 => Ticket

1
Input data:
Id:
3
Age:
25
Name:
Victor
row inserted
To proceed press Enter

```

Таблиця після запиту:

	 id [PK] integer	 age integer	 name character varying
1	0	54	Ivan
2	1	55	Koban
3	3	25	Victor

Функція внесення даних до таблиці Room

```
static public void AddRoom(NpgsqlConnection con, int room_id, string room_colour)//add room
{
    var sql = "insert into \"Room\"(room_id,room_colour) VALUES(@room_id, @room_colour)";
    var cmd = new NpgsqlCommand(sql, con);
    cmd.Parameters.AddWithValue("room_id", room_id);
    cmd.Parameters.AddWithValue("room_colour", room_colour);
    cmd.Prepare();

    Execute(cmd);
}
```

Рис 10. – Функція додавання даних до таблиці Room в файлі Backend.cs

Початкова таблиця:

	room_id [PK] integer	room_colour character varying
1	0	red
2	1	black
3	2	green
4	3	pink

Запит у меню:

```
Choose the Table name
1 => Client
2 => Film
3 => Room
4 => Passport
5 => Session
6 => Telephone
7 => Ticket

3
Input data:
Room Id:
4
Room colour:
lemon
To proceed press Enter
```

Таблиця після запиту:

	room_id [PK] integer	room_colour character varying
1	0	red
2	1	black
3	2	green
4	3	pink
5	4	lemon

Функція внесення даних до таблиці Film

```
static public void AddFilm(NpgsqlConnection con, int film_id, string film_name, string duration)
{
    var sql = "insert into \"Film\"(film_id,film_name,duration) VALUES(@film_id, @film_name,@duration)";
    var cmd = new NpgsqlCommand(sql, con);
    cmd.Parameters.AddWithValue("film_id", film_id);
    cmd.Parameters.AddWithValue("film_name", film_name);
    cmd.Parameters.AddWithValue("duration", duration);
    cmd.Prepare();

    Execute(cmd);
}
```

Рис 11. – Функція додавання даних до таблиці Room в файлі Backend.cs

Початкова таблиця:

	film_id [PK] integer	film_name character varying	duration character varying
1	0	Moon	1 год. 30 хв.
2	1	Rose Tatoo	2 год.0 хв.

Запит у меню:

```
Choose the Table name
1 => Client
2 => Film
3 => Room
4 => Passport
5 => Session
6 => Telephone
7 => Ticket

2
Input data:
Film Id:
3
Name:
My mother told me...
Duration:
1 год 30 хв
To proceed press Enter
```

Таблиця після запиту:

	film_id [PK] integer	film_name character varying	duration character varying
1	0	Moon	1 год. 30 хв.
2	1	Rose Tatoo	2 год.0 хв.
3	3	My mother told me...	1 год 30 хв





Функція внесення даних до таблиці Session

```
static public void AddSession(NpgsqlConnection con, int session_id, int film_id, int room_id, DateTime date)
{
    var sql = "insert into \"Session\"(id,st_film_id,st_room_id,date) VALUES(@id,@st_film_id, @st_room_id,@date)";
    var cmd = new NpgsqlCommand(sql, con);
    cmd.Parameters.AddWithValue("id", session_id);
    cmd.Parameters.AddWithValue("st_film_id", film_id);
    cmd.Parameters.AddWithValue("st_room_id", room_id);
    cmd.Parameters.AddWithValue("date", date);
    cmd.Prepare();

    Execute(cmd);
}
```

Рис 12. – Функція додавання даних до таблиці Session в файлі Backend.cs

Початкова таблиця:





	 id [PK] integer	 st_film_id integer	 st_room_id integer	 date date
1	0	1	0	2020-09-...
2	1	0	1	2020-09-...

Запит у меню

```
Choose the Table name
1 => Client
2 => Film
3 => Room
4 => Passport
5 => Session
6 => Telephone
7 => Ticket

5
Input data:
Session id:
3
Film id:
1
Room id:
3
Date:
2020-11-09
To proceed press Enter
```

Таблиця після запиту:

	 id [PK] integer	 st_film_id integer	 st_room_id integer	 date date
1	0	1	0	2020-09-...
2	1	0	1	2020-09-...
3	3	1	3	2020-11-...

Аналогічно виглядають функції додавання для усіх інших таблиць

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

```
static public void UpdateClient(NpgsqlConnection con, int id, int age, string name)
{
    var sql = $"UPDATE \"Client\" SET age={age}, name='{name}' WHERE id = {id} ";
    var cmd = new NpgsqlCommand(sql, con);
    Execute(cmd);
}
```

Рис 13. – Функція для оновлення даних до таблиці Client в файлі Backend.cs

Початкова таблиця

	 id [PK] integer	 age integer	 name character varying	
1		0	54	Ivan
2		1	55	Koban
3		3	25	Victor

Запит у меню

```
Choose the Table name
1 => Client
2 => Film
3 => Room
4 => Passport
5 => Session
6 => Telephone
7 => Ticket

1
Input data:
Id:
3
Age:
44
Name:
Nate
To proceed press Enter
```

Таблиця після запиту:

	id [PK] integer	age integer	name character varying
1	0	54	Ivan
2	1	55	Koban
3	3	44	Nate

Функція оновлення даних до таблиці Film

```
static public void UpdateFilm(NpgsqlConnection con, int id, string film_name)
{
    var sql = $"UPDATE \"Film\" SET film_name = '{film_name}' WHERE film_id = {id} ";
    var cmd = new NpgsqlCommand(sql, con);
    Execute(cmd);
}
```

Рис 14. – Функція оновлення даних до таблиці Film в файлі Backend.cs

Початкова таблиця

	film_id [PK] integer	film_name character varying	duration character varying
1	0	Moon	1 год. 30 хв.
2	1	Rose Tatoo	2 год.0 хв.
3	3	My mother told me...	1 год 30 хв

Запит у меню

```
Choose the Table name
1 => Client
2 => Film
3 => Room
4 => Passport
5 => Session
6 => Telephone
7 => Ticket

2
Input data:
Film Id:
1
Name:
Vikings
To proceed press Enter
```

Таблиця після запиту

	film_id [PK] integer	film_name character varying	duration character varying
1	0	Moon	1 год. 30 хв.
2	1	Vikings	2 год.0 хв.
3	3	My mother told me...	1 год 30 хв

Функція оновлення даних до таблиці Room

```
static public void UpdateRoom(NpgsqlConnection con, int room_id, string room_colour)
{
    var sql = $"UPDATE \"Room\" SET room_colour = '{room_colour}' WHERE room_id = {room_id} ";
    var cmd = new NpgsqlCommand(sql, con);
    Execute(cmd);
}
```

Рис 15. – Функція оновлення даних до таблиці Room в файлі Backend.cs

Початкова таблиця

	room_id [PK] integer	room_colour character varying
1	0	red
2	1	black
3	2	green
4	3	pink
5	4	lemon

Запит у меню

```
Choose the Table name
1 => Client
2 => Film
3 => Room
4 =>Passport
5 => Session
6 => Telephone
7 => Ticket

3
Input data:
Room Id:
4
Room colour:
brown
To proceed press Enter
```


Таблиця після запиту

	room_id [PK] integer 	room_colour character varying 
1	0	red
2	1	black
3	2	green
4	3	pink
5	4	brown

Аналогічно виглядають функції оновлення для усіх інших таблиць

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

```
static public void DeleteClient(NpgsqlConnection con, int id)//using on delete cascade
{
    var sql = $"DELETE FROM \"Client\" WHERE id=" + id;
    var cmd = new NpgsqlCommand(sql, con);
    Execute(cmd);
}
```

Рис 16. – Функція вилучення даних з таблиці Client в файлі Backend.cs

Початкова таблиця

	id [PK] integer	age integer	name character varying
1	0	54	Ivan
2	1	55	Koban
3	3	44	Nate

Запит у меню

Choose the Table name

1 => Client

2 => Film

3 => Room

4 => Session

5 => Passport

6 => Telephone

7 => Ticket

1

Input data:

Id:

3

To proceed press Enter

Таблиця після запиту

	id [PK] integer	age integer	name character varying
1	0	54	Ivan
2	1	55	Koban

Функція вилучення даних з таблиці Film

```
static public void DeleteFilm(NpgsqlConnection con, int id)
{
    var sql = $"DELETE FROM \"Film\" WHERE film_id=" + id;
    var cmd = new NpgsqlCommand(sql, con);
    Execute(cmd);
}
```

Рис 17. – Функція вилучення даних з таблиці Film в файлі Backend.cs

Початкова таблиця

	film_id [PK] integer	film_name character varying	duration character varying
1	0	Moon	1 год. 30 хв.
2	1	Vikings	2 год.0 хв.
3	3	My mother told me...	1 год 30 хв

Запит у меню

```
Choose the Table name
1 => Client
2 => Film
3 => Room
4 => Session
5 => Passport
6 => Telephone
7 => Ticket

2
Input data:
Film Id:
3
To proceed press Enter
```

Таблиця після запиту

	film_id [PK] integer	film_name character varying	duration character varying
1	0	Moon	1 год. 30 хв.
2	1	Vikings	2 год.0 хв.

Функція вилучення даних з таблиці Room

```
static public void DeleteRoom(NpgsqlConnection con, int id)
{
    var sql = $"DELETE FROM \"Room\" WHERE room_id=" + id;
    var cmd = new NpgsqlCommand(sql, con);
    Execute(cmd);
}
```

Рис 18. – Функція вилучення даних з таблиці Room в файлі Backend.cs

Початкова таблиця




	room_id [PK] integer	room_colour character varying
1	0	red
2	1	black
3	2	green
4	3	pink
5	4	brown

Запит у меню

```
Choose the Table name
1 => Client
2 => Film
3 => Room
4 => Session
5 => Passport
6 => Telephone
7 => Ticket

3
Input data:
Room Id:
4
To proceed press Enter
```

Таблиця після запиту

	 room_id [PK] integer	 room_colour character varying	
1	0	red	
2	1	black	
3	2	green	
4	3	pink	

Функція вилучення даних з таблиці Session

```
static public void DeleteSession(NpgsqlConnection con, int id)
{
    var sql = $"DELETE FROM \"Session\" WHERE id=" + id;
    var cmd = new NpgsqlCommand(sql, con);
    Execute(cmd);
}
```

Рис 19. – Функція вилучення даних з таблиці Session в файлі Backend.cs

Початкова таблиця

	id [PK] integer	st_film_id integer	st_room_id integer	date date
1	0	1	0	2020-09-...
2	1	0	1	2020-09-...
3	3	1	3	2020-11-...

Запит у меню

```
Choose the Table name
1 => Client
2 => Film
3 => Room
4 => Session
5 => Passport
6 => Telephone
7 => Ticket

4
Input data:
Session Id:
3
To proceed press Enter
```

Таблиця після запиту

	id [PK] integer	st_film_id integer	st_room_id integer	date date
1	0	1	0	2020-09-...
2	1	0	1	2020-09-...

Аналогічно виглядають функції вилучення даних для усіх інших таблиць



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

Рандомізоване внесення даних до таблиці «Film»

```
static public void RandomFilm(NpgsqlConnection con)
{
    Console.WriteLine("Num of randomized rows: ");
    int num = Convert.ToInt32(Console.ReadLine());
    //var rand_char = "chr(trunc(65+random()*25)::int)";
    string[] masstr = new string[num];
    for (int j = 0; j < num; j++) masstr[j] += " chr(trunc(65+random()*25)::int)";
    string rand_char = String.Join("||",masstr);
    for(int i=0; i<num;i++)
    {
        var sql = $"insert into \"Film\"(film_id,film_name,duration) VALUES((random()*10000)::int , {rand_char} ,{rand_char})";
        var cmd = new NpgsqlCommand(sql, con);
        Execute(cmd);
    }
}
```

Рис 20. – Фотографія коду програми для рандомізованого внесення даних в файлі Backend.cs

Початкова таблиця

	 film_id [PK] integer	 film_name character varying	 duration character varying
1	0	Moon	1 год. 30 хв.
2	1	Vikings	2 год.0 хв.

Запит у меню

```
Main menu
0 => Show one table
1 => Insert data
2 => Delete
3 => Update data
4 => Search text
5 => Randomize data in Film
6 => Exit
5
Num of randomized rows:
1
To proceed press Enter
```

Таблиця після запиту

	film_id [PK] integer	film_name character varying	duration character varying
1	0	Moon	1 год. 30 хв.
2	1	Vikings	2 год.0 хв.
3	8530	G	W

Початкова таблиця

```

Main menu
0 => Show one table
1 => Insert data
2 => Delete
3 => Update data
4 => Search text
5 => Randomize data in Film
6 => Exit
5
Num of randomized rows:
15
To proceed press Enter

```

Таблиця після запиту

	film_id [PK] integer	film_name character varying	duration character varying
1	0	Moon	1 год. 30 хв.
2	1	Vikings	2 год.0 хв.
3	571	IIKLGKOECDWLTQ	DXBYFNIBMTEMELM
4	1064	AKHWHHOSEOIXUDK	AQGIBDLKFQORSBO
5	1572	YHTEBDSOASMTORV	IHMVDXTYNJFBTBL
6	2717	PSUBMHMWCWBGNY	SRNSFLTMVOICUDM
7	2814	AFMG SXOHVOFBRJN	HCMQTRLYANCRGVA
8	3698	CHRWSGEDKDRHRIV	LYJWWGXPWSPMUOG
9	3872	ORIEQIXPGHFMSRW	MUFIALPFBGYQMFL
10	3881	DYQEVBUVXRVTMXU	QOWNPVFVDBUEUHK
11	4552	RVXTEVXAVVFKIIG	KOAOSQRAYIVNAOT
12	4568	RJLOJMBRJGVNHND	VQBBWWALRPXBIWT
13	5814	XBLGNYSXIPFUCV	EIWNMQVXRAWXGAQ
14	5872	NMUTOFPBVEBSMCX	KCTSTOXGSDTRRKA
15	6193	EEGNVGFQEENQSFH	OXVQICGQCXXWYLS
16	6666	BTOXKQGIUTBBBLK	VGVELVCSHSQGSRQ
17	7256	WUMDENSDTDKEWUW	XVYPCXOXQUEAVLR
18	8530	G	W

Виконання динамічних запитів бази даних

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

```
Choose search
1 => Поиск кинозала по цвету и по фильмам которые будут в нём показываться
2 => Поиск кинозала, где будет сидеть клиент по идентификатором его паспорта та цветом комнаты
```

Рис 21. – Меню програми де ми можемо побачити типи доступних динамічних запитів

```
static public void DynamicSearch1(NpgsqlConnection con)
{
    Console.WriteLine("Поиск кинозала по цвету и по фильмам которые будут в нём показываться");
    string room_colour, film_name;
    Console.WriteLine("Цвет комнаты: ");
    room_colour = Console.ReadLine();
    Console.WriteLine("Название фильма: ");
    film_name = Console.ReadLine();

    var sql = $"select room_id,room_colour,film_id,duration from \"Room\" " +
        $"inner join \"Session\" on \"Room\".room_id = \"Session\".st_room_id " +
        $"inner join \"Film\" on \"Session\".st_film_id = \"Film\".film_id " +
        $" WHERE room_colour = '{room_colour}' and film_name = '{film_name}'";
    var cmd = new NpgsqlCommand(sql, con);
    Execute(cmd);

    NpgsqlDataReader rdr = cmd.ExecuteReader();
    Console.WriteLine($"{rdr.GetName(0),-4}\t {rdr.GetName(1),-4}\t {rdr.GetName(2),10}\t {rdr.GetName(3),10}");
    while (rdr.Read())
    {
        Console.WriteLine($"{rdr.GetInt32(0),-4} \t {rdr.GetString(1),-3}\t {rdr.GetInt32(2),10} \t {rdr.GetString(3),10}");
    }
    rdr.Close();
}
```

Рис 22. – Фотографія коду програми, 1-го динамічного запиту, що знаходиться в файлі Backend.cs

Запит в меню

```
Choose search
1 => Поиск кинозала по цвету и по фильмам которые будут в нём показываться
2 => Поиск кинозала, где будет сидеть клиент по идентификатором его паспорта та цветом комнаты
1
Поиск кинозала по цвету и по фильмам которые будут в нём показываться
Цвет комнаты:
red
Название фильма:
Vikings
room_id room_colour          film_id      duration
0       red                  1            2 год.0 хв.
To proceed press Enter
```

```

static public void DynamicSearch2(NpgsqlConnection con)
{
    Console.WriteLine("Поиск кинозала, где будет сидеть клиент по идентификатором его паспорта та цветом комнаты");
    int pass_id;
    string room_colour;
    Console.WriteLine("Идентификатор паспорта: ");
    pass_id = Convert.ToInt32(Console.ReadLine());
    Console.WriteLine("Цвет комнаты: ");
    room_colour = Console.ReadLine();

    var sql = $"select \"Room\".room_id,\"Room\".room_colour, \"Passport\".id, \"Client\".name " +
    "from \"Passport\" " +
    "inner join \"Client\" on \"Passport\".id = \"Client\".id " +
    "inner join \"Ticket\" on \"Ticket\".client_id = \"Client\".id " +
    "inner join \"Session\" on \"Ticket\".session_id = \"Session\".id " +
    "inner join \"Room\" on \"Session\".st_room_id = \"Room\".room_id " +
    $"where \"Passport\".id = {pass_id} and \"Room\".room_colour = '{room_colour}'";

    var cmd = new NpgsqlCommand(sql, con);
    Execute(cmd);

    NpgsqlDataReader rdr = cmd.ExecuteReader();
    Console.WriteLine($"{rdr.GetName(0),-4}\t {rdr.GetName(1),-4}\t {rdr.GetName(2),10}\t {rdr.GetName(3),10}");
    while (rdr.Read())
    {
        Console.WriteLine($"{rdr.GetInt32(0),-4} \t {rdr.GetString(1),-3}\t\t {rdr.GetInt32(2),10} \t {rdr.GetString(3),10}");
    }
    rdr.Close();
}

```

Рис 23. – Фотографія коду програми, 2-го динамічного запиту, що знаходиться в файлі Backend.cs

Запит в меню

```

Choose search
1 => Поиск кинозала по цвету и по фильмам которые будут в нём показываться
2 => Поиск кинозала, где будет сидеть клиент по идентификатором его паспорта та цветом комнаты
2
Поиск кинозала, где будет сидеть клиент по идентификатором его паспорта та цветом комнаты
Идентификатор паспорта:
0
Цвет комнаты:
red
room_id room_colour id name
0 red 0 Ivan
0 red 0 Ivan
To proceed press Enter

```

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

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

```
public static void ExecuteQuery(NpgsqlCommand _cmd)
{
    try
    {
        _cmd.ExecuteNonQuery();
    }
    catch(Exception ex)
    {
        Console.WriteLine("Error" + ex.Message);
    }
}
```

Рис 24. – Фотографія коду програми, де зображений механізм try-catch помилок, що знаходиться у файлі Backend.cs

Приклади перевірки на валідність вхідних даних

```
Choose the Table name
1 => Client
2 => Film
3 => Room
4 => Passport
5 => Session
6 => Telephone
7 => Ticket

1
Input data:
Id:
0
Age:
43
Name:
889
Error23505: повторяющееся значение ключа нарушает ограничение уникальности "Client_pkey"
Error23505: повторяющееся значение ключа нарушает ограничение уникальности "Passport_pkey"
row inserted
To proceed press Enter
```

Дослідження режимів обмеження ON DELETE

Режими обмеження ми будемо досліджувати на прикладі таблиць “Client” – батьківська таблиця та “Passport” - як дочірня.

Початкові таблиці

	id [PK] integer	age integer	name character varying
1	0	54	Ivan
2	1	55	Koban

	id [PK] integer
1	0
2	1

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

NO ACTION

Запит меню

Choose the Table name

1 => Client
2 => Film
3 => Room
4 => Session
5 => Passport
6 => Telephone
7 => Ticket

1

Input data:

Id:

1

Error23503: UPDATE или DELETE в таблице "Client" нарушает ограничение внешнего ключа "id" таблицы "Passport"
To proceed press Enter

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

SET NULL

Запит меню

```
Choose the Table name
1 => Client
2 => Film
3 => Room
4 => Session
5 => Passport
6 => Telephone
7 => Ticket

1
Input data:
Id:
0
Error23502: нулевое значение в столбце "id" нарушает ограничение NOT NULL
To proceed press Enter
```

SET DEFAULT

Запит меню

```
Choose the Table name
1 => Client
2 => Film
3 => Room
4 => Session
5 => Passport
6 => Telephone
7 => Ticket

1
Input data:
Id:
0
Error23502: нулевое значение в столбце "id" нарушает ограничение NOT NULL
To proceed press Enter
```

RESTRICT

Запит меню

```
Choose the Table name
1 => Client
2 => Film
3 => Room
4 => Session
5 => Passport
6 => Telephone
7 => Ticket

1
Input data:
Id:
0
Error23503: UPDATE или DELETE в таблице "Client" нарушает ограничение внешнего ключа "id" таблицы "Passport"
To proceed press Enter
```

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

```
Choose the Table name
1 => Client
2 => Film
3 => Room
4 => Session
5 => Passport
6 => Telephone
7 => Ticket

1
Input data:
Id:
1
To proceed press Enter
```

Таблиці після запиту

	id [PK] integer	age integer	name character varying
1	0	54	Ivan

	id [PK] integer
1	0


Ілюстрації програмного коду на Github

master ▾

[DataBase](#) / [Labs](#) / [Lab2](#) /

Go to file

Add file ▾



iVanyaVas Lab2 done

5e988ec 23 seconds ago

History

..

Code	Lab2 done	23 seconds ago
readme.txt	Lab2 done	23 seconds ago

readme.txt

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

Загальне завдання роботи полягає у наступному:

1. Реалізувати функції внесення, редагування та виучення 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 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).

Деталізоване завдання:

1. Забезпечити можливість введення/редагування/виучення даних у таблицях бази даних з можливістю контролю відповідності типів даних атрибутів таблиць (рядків, чисел, дати/часу). Для контролю пропонується два варіанти:
 - a. контроль при введенні - валідація даних;
 - b. перехоплення помилок (try...except) від сервера PostgreSQL при виконанні відповідної команди SQL.Особливу увагу варто звернути на дані таблиці, що мають зв'язок 1:N. З боку батьківської таблиці необхідно контролювати виучення (ON DELETE) рядків за умови наявності даних у підлеглий 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.
2. 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.
3. 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 введення конкретних значень констант для фільтрації з клавіатури користувачем. Після виведення даних вивести час виконання запиту у мілісекундах.
4. Перевірити швидкодію роботи запитів на попередньо згенерованих даних.


Програмний код організувати згідно шаблону Model-ViewController (MVC). Приклад організації коду згідно шаблону доступний за даними посиланням. Модель, подання (представлення) та контролер мають бути реалізовані у окремих файлах. Для доступу до бази даних використовувати лише мову SQL (без ORM).

master ▾

[DataBase](#) / [Labs](#) / [Lab2](#) / [Code](#) /

Go to file

Add file ▾



iVanyaVas Lab2 done

5e988ec 1 minute ago

History

..

Backend.cs	Lab2 done	1 minute ago
Controller.cs	Lab2 done	1 minute ago
Main.cs	Lab2 done	1 minute ago