**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»**

Виконав: студент ІII курсу

ФПМ групи КВ-13

Чоловенко Дмитро Володимирович

Telegram: @D\_4ubaka

GitHub: https://github.com/D-Cholik/DB\_lab-RGR-2.git

Перевірив:

**Київ – 2023**

**Створення додатку бази даних, орієнтованого на взаємодію з СУБД PostgreSQL**

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

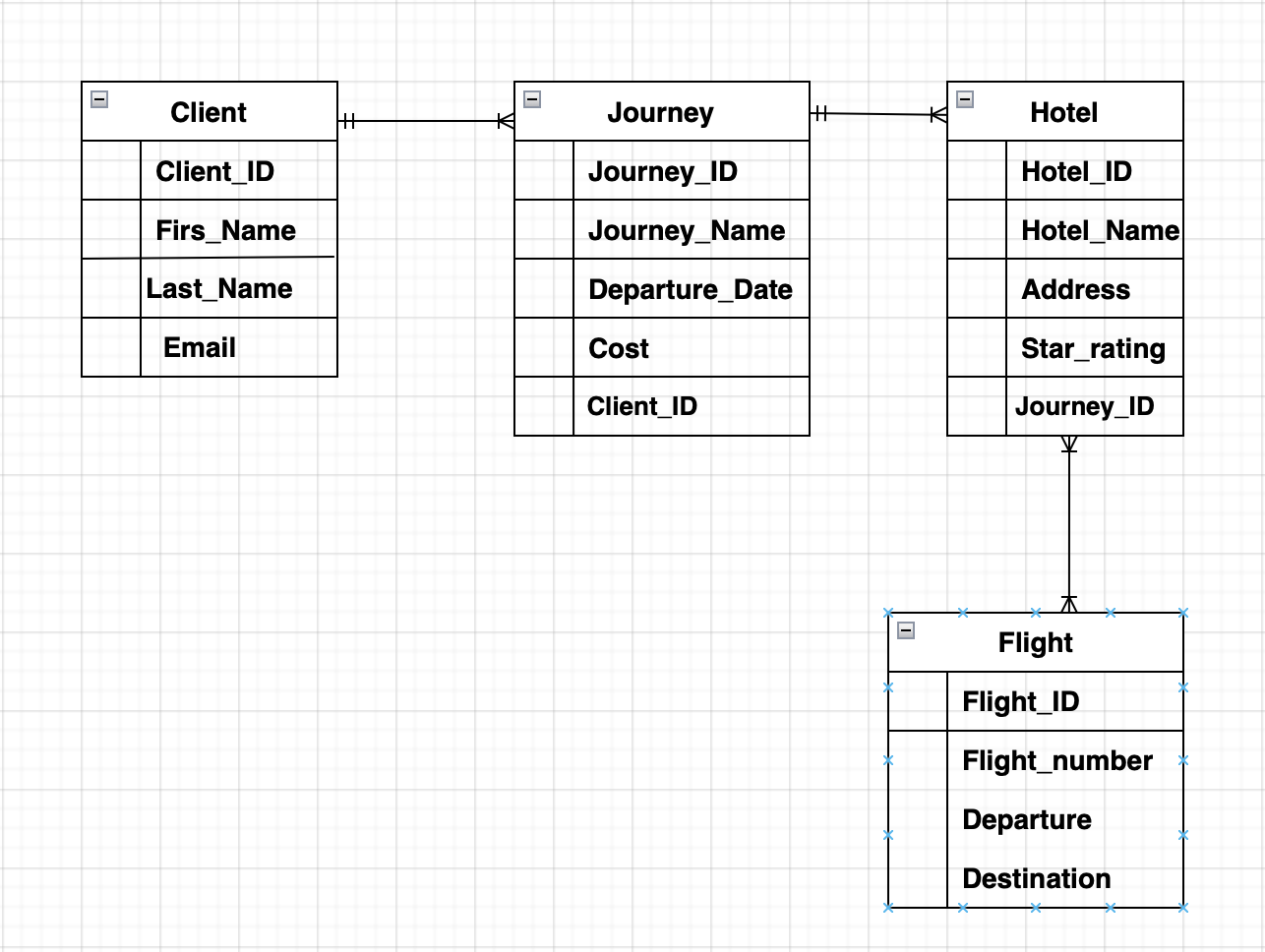
*Загальне* *завдання* роботи полягає у наступному:

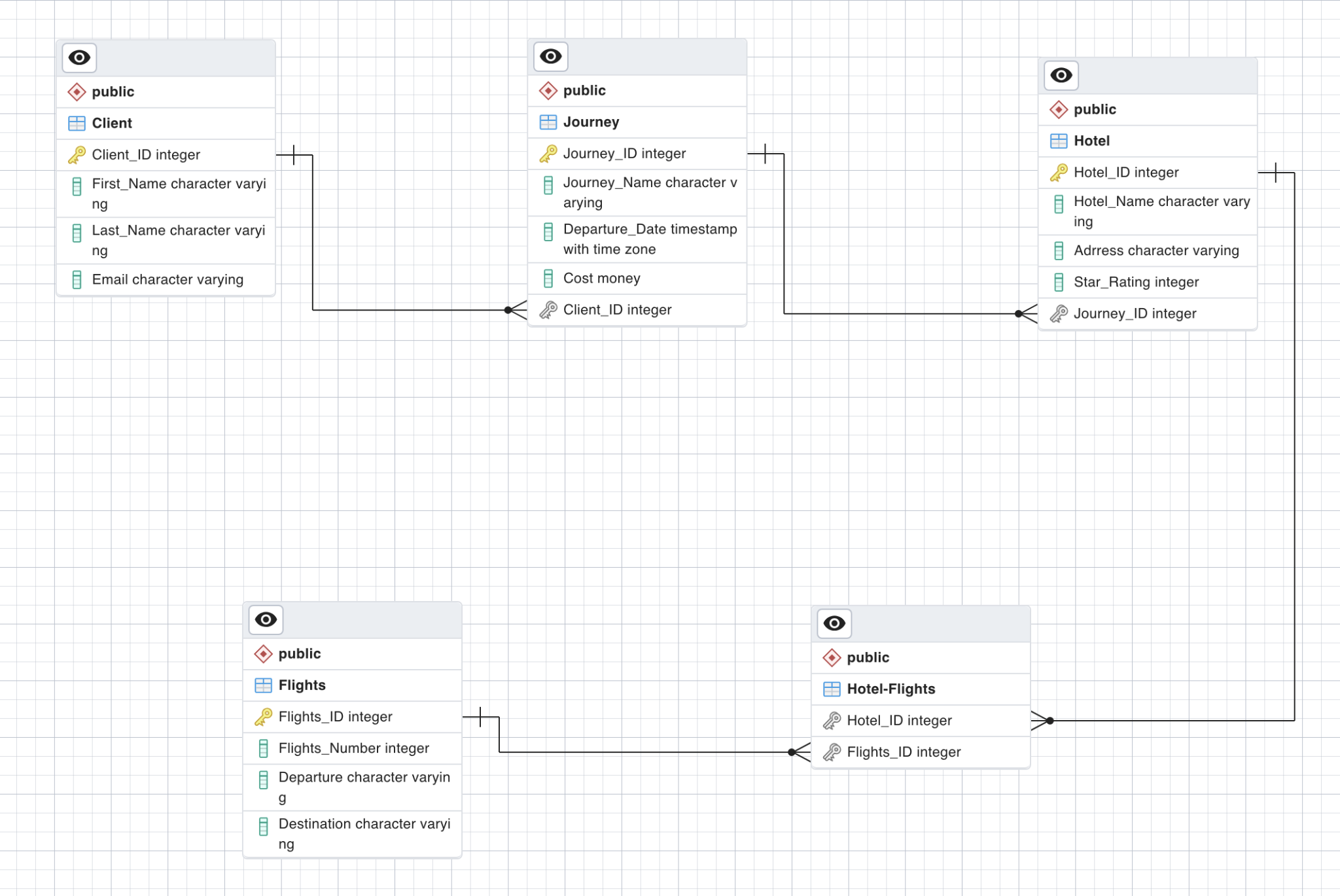
1. Реалізувати функції перегляду, внесення, редагування та вилучення даних у таблицях бази даних, створених у лабораторній роботі №1, засобами консольного інтерфейсу.
2. Передбачити автоматичне пакетне генерування «рандомізованих» даних у базі.
3. Забезпечити реалізацію пошуку за декількома атрибутами 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).

**Короткий опис бази даних**

Між Готелями і Рейсами існує зв'язок "багато до багатьох" (M:N). Один готель може відповідати багатьом рейсам, і навпаки, один рейс може мати відношення до багатьох готелів. Для вираження цього зв'язку в базі даних потрібно створити додаткову таблицю яка буде містити зв'язки між готелями і рейсами за допомогою зовнішніх ключів, що посилаються на ID готелю і ID рейсу відповідно.

**Діаграма сутність-зв’язок та структура бази даних**

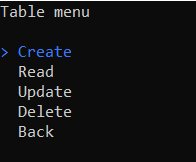




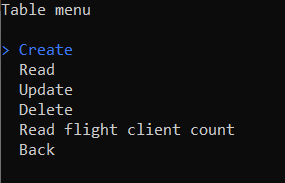
**Схема меню користувача**



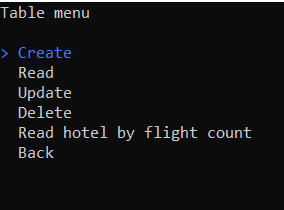
Clients:



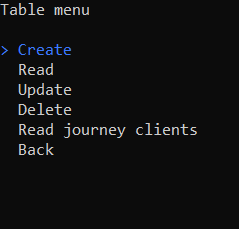
Flights:



Hotels:



Journeys:

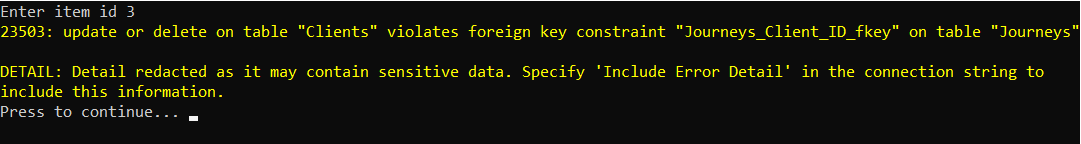


**Використані технології**

* .NET
  + ADO.NET (Npgsql)
  + Spectre.Console

**Видалення батьківських таблиць**

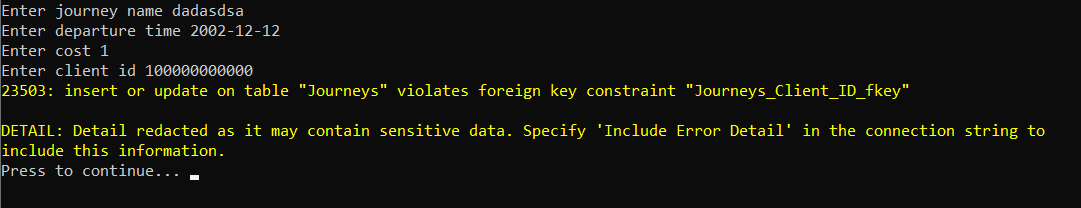
Спробуємо видалити запис із таблиці Clients



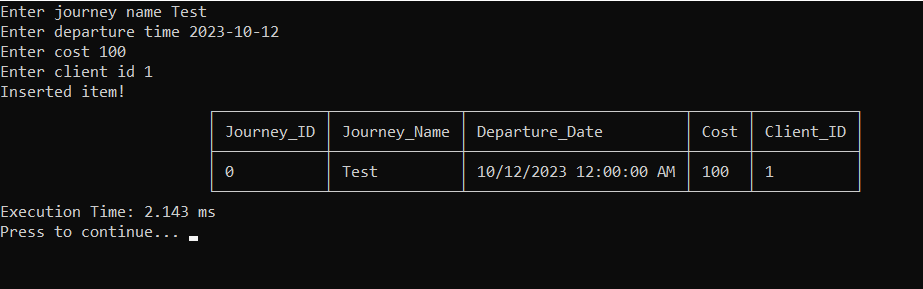
Як бачимо, ми не можемо видалити через зовнішній ключ. Програма працює правильно.

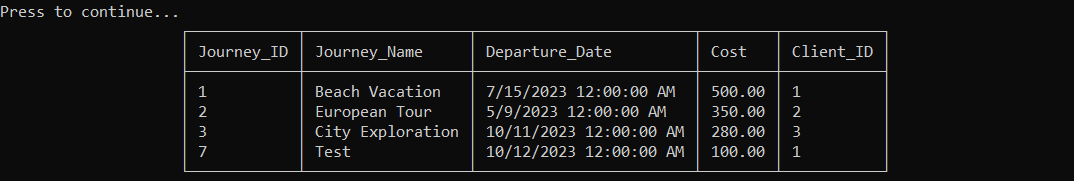
**Вставка дочірніх таблиць**

Спробуємо створити запис у таблицю Journeys з невірним значенням ключа Client\_ID



Як бачимо, програма працює правильно. Спробуємо вставити значення з існуючим ключем

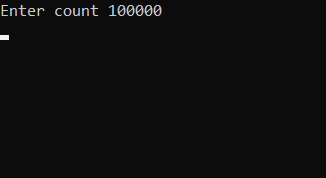


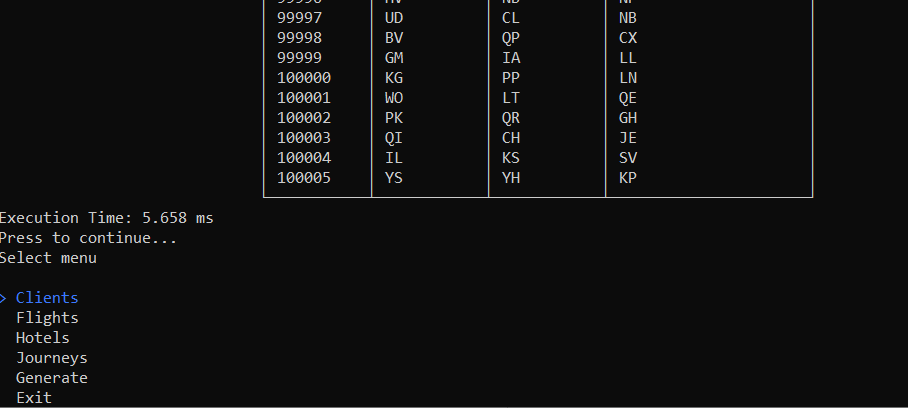


Як бачимо все пройшло успішно

**Генерація рандомізованих даних**

Згенеруємо 1000 записів для всіх таблиць, щоб побачити коректність складних запитів. Потім для тестування генерування великої кількості даних згенеруємо 100000 для таблиці Clients. Робимо це через пункт Generate





**Запити для генерації даних:**

**Clients:**

insert into "Clients"("First\_Name", "Last\_Name", "Email") values (chr(trunc(65 + random() \* 25)::int) || chr(trunc(65 + random() \* 25)::int), chr(trunc(65 + random() \* 25)::int) || chr(trunc(65 + random() \* 25)::int), chr(trunc(65 + random() \* 25)::int) || chr(trunc(65 + random() \* 25)::int));

**Flights:**

insert into "Flights"("Flight\_Number", "Departure", "Destination") values (trunc(random() \* 10000)::int, chr(trunc(65 + random() \* 25)::int) || chr(trunc(65 + random() \* 25)::int), chr(trunc(65 + random() \* 25)::int) || chr(trunc(65 + random() \* 25)::int));

**Hotels:**

insert into "Hotels"("Hotel\_Name", "Address", "Star\_Rating", "Journey\_ID")

values

(

chr(trunc(65 + random() \* 25)::int) || chr(trunc(65 + random() \* 25)::int),

chr(trunc(65 + random() \* 25)::int) || chr(trunc(65 + random() \* 25)::int),

trunc(1 + random() \* 5)::int,

(select f\_id from trunc(1 + random() \* (select max("Journey\_ID") from "Journeys")) as f\_id inner join "Journeys" on f\_id = "Journeys"."Journey\_ID")::bigint

);

**Journeys:**

insert into "Journeys"("Journey\_Name", "Departure\_Date", "Cost", "Client\_ID")

values

(

chr(trunc(65 + random() \* 25)::int) || chr(trunc(65 + random() \* 25)::int),

(select timestamp '2020-01-10 20:00:00' + random() \* (timestamp '2023-01-10 10:00:00' - timestamp '2020-01-10 20:00:00'))::timestamp,

trunc(1 + random() \* 2000)::int,

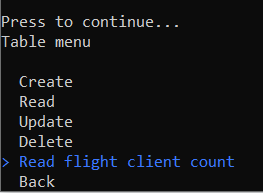
(select f\_id from trunc(1 + random() \* (select max("Client\_ID") from "Clients")) as f\_id inner join "Clients" on f\_id = "Clients"."Client\_ID")::bigint

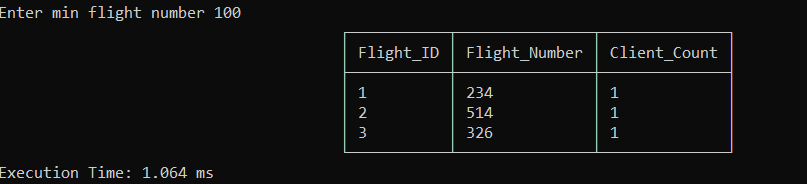
);

**Ілюстрації уведення пошукового запиту та результатів виконання запитів**

Read flight client count:

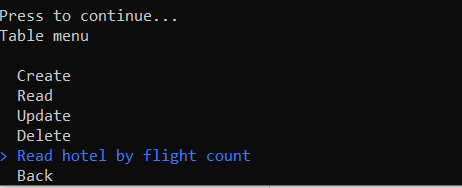
elect "Flights"."Flight\_ID", "Flights"."Flight\_Number", count("Client\_ID") as u\_c from "Flights" inner join "Hotel-Flights" using ("Flight\_ID") inner join "Hotels" using ("Hotel\_ID") inner join "Journeys" using ("Journey\_ID") inner join "Clients" using ("Client\_ID") where "Flights"."Flight\_Number" > ***{filter}*** group by "Flights"."Flight\_ID", "Flights"."Flight\_Number", "Flights"."Departure", "Flights"."Destination" Order by "Flights"."Flight\_ID";





Read hotel by flight count:

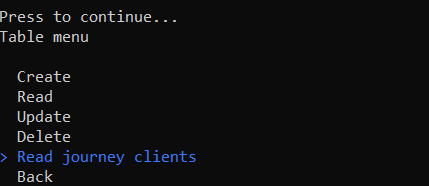
select "Hotels"."Hotel\_ID", "Hotels"."Hotel\_Name", count("Flights"."Flight\_ID") as "Flight\_Count"from "Hotels"inner join "Hotel-Flights" using ("Hotel\_ID")inner join "Flights" using ("Flight\_ID")group by "Hotels"."Hotel\_ID", "Hotels"."Hotel\_Name", "Hotels"."Star\_Rating"having count("Flights"."Flight\_ID") between {filterF} and {filterS}Order by "Hotels"."Hotel\_ID";





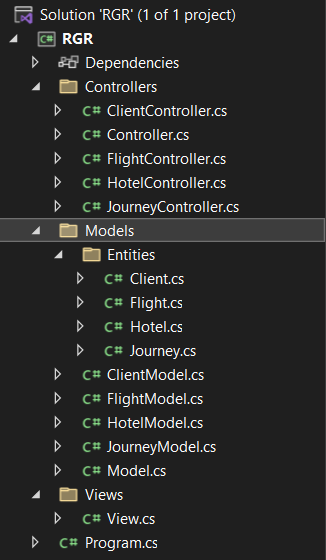
Read journey clients:

insert into "Journeys"("Journey\_Name", "Departure\_Date", "Cost", "Client\_ID")values (chr(trunc(65 + random() \* 25)::int) || chr(trunc(65 + random() \* 25)::int),(select timestamp '2020-01-10 20:00:00' + random() \* (timestamp '2023-01-10 10:00:00' - timestamp '2020-01-10 20:00:00'))::timestamp, trunc(1 + random() \* 2000)::int,(select f\_id from trunc(1 + random() \* (select max("Client\_ID") from "Clients")) as f\_id inner join "Clients" on f\_id = "Clients"."Client\_ID")::bigint);





**Ілюстрації програмного коду модуля “Model”**



**ClientModel.cs**

using Npgsql;

using NpgsqlTypes;

using RGR.Models.Entities;

using System.Collections.Generic;

using System.Reflection.PortableExecutable;

namespace RGR.Models;

public class ClientModel : Model<Client>

{

public ClientModel(NpgsqlConnection connection) : base(connection) {}

public override void CreateItem(Client item)

{

using NpgsqlCommand command = new NpgsqlCommand();

command.Connection = Connection;

command.Parameters.Add(new NpgsqlParameter("@param\_first\_name", NpgsqlDbType.Varchar, 100) { Value = (object?)item.First\_Name });

command.Parameters.Add(new NpgsqlParameter("@param\_last\_name", NpgsqlDbType.Varchar, 100) { Value = (object?)item.Last\_Name });

command.Parameters.Add(new NpgsqlParameter("@param\_mail", NpgsqlDbType.Varchar, 256) { Value = (object?)item.Email });

command.CommandText = $"explain analyze insert into \"Clients\"(\"First\_Name\", \"Last\_Name\", \"Email\") values(@param\_first\_name, @param\_last\_name, @param\_mail);";

try

{

Connection.Open();

using NpgsqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

LastQueryExecutionTime = (string)reader.GetValue(0);

}

}

finally

{

Connection.Close();

}

}

public override void DeleteItem(long id)

{

using NpgsqlCommand command = new NpgsqlCommand();

command.Connection = Connection;

command.CommandText = $"delete from \"Clients\" where \"Client\_ID\" = {id}";

try

{

Connection.Open();

using NpgsqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

LastQueryExecutionTime = (string)reader.GetValue(0);

}

}

finally

{

Connection.Close();

}

}

public override IEnumerable<Client> ReadItems()

{

using NpgsqlCommand command = new NpgsqlCommand();

command.Connection = Connection;

command.CommandText = $"select \* from \"Clients\"";

IEnumerable<Client> items = new List<Client>();

try

{

Connection.Open();

using NpgsqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

Client item = new Client();

item.Client\_ID = (long)reader["Client\_ID"];

item.First\_Name = (string)reader["First\_Name"];

item.Last\_Name = (string)reader["Last\_Name"];

item.Email = (string)reader["Email"];

((List<Client>)items).Add(item);

}

}

finally

{

Connection.Close();

}

using var explain = command.Clone();

explain.CommandText = "explain analyze " + command.CommandText;

try

{

Connection.Open();

using NpgsqlDataReader explainReader = explain.ExecuteReader();

while (explainReader.Read())

{

LastQueryExecutionTime = (string)explainReader.GetValue(0);

}

}

finally

{

Connection.Close();

}

return items;

}

public override void UpdateItem(long id, Client newItem)

{

using NpgsqlCommand command = new NpgsqlCommand();

command.Connection = Connection;

command.Parameters.Add(new NpgsqlParameter("@param\_first\_name", NpgsqlDbType.Varchar, 100) { Value = (object?)newItem.First\_Name });

command.Parameters.Add(new NpgsqlParameter("@param\_last\_name", NpgsqlDbType.Varchar, 100) { Value = (object?)newItem.Last\_Name });

command.Parameters.Add(new NpgsqlParameter("@param\_mail", NpgsqlDbType.Varchar, 256) { Value = (object?)newItem.Email });

command.CommandText = $"explain analyze update \"Clients\" set \"Client\_ID\" = {id}, \"First\_Name\" = @param\_first\_name, \"Last\_Name\" = @param\_last\_name, \"Email\" = @param\_mail where \"Client\_ID\" = {id};";

try

{

Connection.Open();

using NpgsqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

LastQueryExecutionTime = (string)reader.GetValue(0);

}

}

finally

{

Connection.Close();

}

}

public override void GenerateItems(long count)

{

using NpgsqlCommand command = new NpgsqlCommand();

command.Connection = Connection;

string query = $"insert into \"Clients\"(\"First\_Name\", \"Last\_Name\", \"Email\") values\r\n(chr(trunc(65 + random() \* 25)::int) || chr(trunc(65 + random() \* 25)::int),\r\nchr(trunc(65 + random() \* 25)::int) || chr(trunc(65 + random() \* 25)::int),\r\nchr(trunc(65 + random() \* 25)::int) || chr(trunc(65 + random() \* 25)::int));";

command.CommandText = "";

for (int i = 0; i < count; i++)

command.CommandText += query;

try

{

Connection.Open();

using NpgsqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

LastQueryExecutionTime = (string)reader.GetValue(0);

}

}

finally

{

Connection.Close();

}

}

}

**FlightModel.cs**

using Npgsql;

using NpgsqlTypes;

using RGR.Models.Entities;

using System.Collections.Generic;

namespace RGR.Models;

public class FlightModel : Model<Flight>

{

public FlightModel(NpgsqlConnection connection) : base(connection) { }

public override void CreateItem(Flight item)

{

using NpgsqlCommand command = new NpgsqlCommand();

command.Connection = Connection;

command.Parameters.Add(new NpgsqlParameter("@param\_number", NpgsqlDbType.Integer) { Value = (object?)item.Flight\_Number });

command.Parameters.Add(new NpgsqlParameter("@param\_departure", NpgsqlDbType.Varchar, 100) { Value = (object?)item.Departure });

command.Parameters.Add(new NpgsqlParameter("@param\_destination", NpgsqlDbType.Varchar, 256) { Value = (object?)item.Destination });

command.CommandText = $"explain analyze insert into \"Flights\"(\"Flight\_Number\", \"Departure\", \"Destination\") values(@param\_number, @param\_departure, @param\_destination);";

try

{

Connection.Open();

using NpgsqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

LastQueryExecutionTime = (string)reader.GetValue(0);

}

}

finally

{

Connection.Close();

}

}

public override void DeleteItem(long id)

{

using NpgsqlCommand command = new NpgsqlCommand();

command.Connection = Connection;

command.CommandText = $"delete from \"Flights\" where \"Flight\_ID\" = {id}";

try

{

Connection.Open();

using NpgsqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

LastQueryExecutionTime = (string)reader.GetValue(0);

}

}

finally

{

Connection.Close();

}

}

public override void GenerateItems(long count)

{

using NpgsqlCommand command = new NpgsqlCommand();

command.Connection = Connection;

string query = $"insert into \"Flights\"(\"Flight\_Number\", \"Departure\", \"Destination\") values \r\n(trunc(random() \* 10000)::int, \r\nchr(trunc(65 + random() \* 25)::int) || chr(trunc(65 + random() \* 25)::int), \r\nchr(trunc(65 + random() \* 25)::int) || chr(trunc(65 + random() \* 25)::int));";

command.CommandText = "";

for (int i = 0; i < count; i++)

command.CommandText += query;

try

{

Connection.Open();

using NpgsqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

LastQueryExecutionTime = (string)reader.GetValue(0);

}

}

finally

{

Connection.Close();

}

}

public override IEnumerable<Flight> ReadItems()

{

using NpgsqlCommand command = new NpgsqlCommand();

command.Connection = Connection;

command.CommandText = $"select \* from \"Flights\"";

IEnumerable<Flight> items = new List<Flight>();

try

{

Connection.Open();

using NpgsqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

Flight item = new Flight();

item.Flight\_ID = (long)reader["Flight\_ID"];

item.Flight\_Number = (int)reader["Flight\_Number"];

item.Departure = (string)reader["Departure"];

item.Destination = (string)reader["Destination"];

((List<Flight>)items).Add(item);

}

}

finally

{

Connection.Close();

}

using var explain = command.Clone();

explain.CommandText = "explain analyze " + command.CommandText;

try

{

Connection.Open();

using NpgsqlDataReader explainReader = explain.ExecuteReader();

while (explainReader.Read())

{

LastQueryExecutionTime = (string)explainReader.GetValue(0);

}

}

finally

{

Connection.Close();

}

return items;

}

public override void UpdateItem(long id, Flight newItem)

{

using NpgsqlCommand command = new NpgsqlCommand();

command.Connection = Connection;

command.Parameters.Add(new NpgsqlParameter("@param\_number", NpgsqlDbType.Integer) { Value = (object?)newItem.Flight\_Number });

command.Parameters.Add(new NpgsqlParameter("@param\_departure", NpgsqlDbType.Varchar, 100) { Value = (object?)newItem.Departure });

command.Parameters.Add(new NpgsqlParameter("@param\_destination", NpgsqlDbType.Varchar, 256) { Value = (object?)newItem.Destination });

command.CommandText = $"explain analyze update \"Flights\" set \"Flight\_ID\" = {id}, \"Flight\_Number\" = @param\_number, \"Departure\" = @param\_departure, \"Destination\" = @param\_destination where \"Flight\_ID\" = {id};";

try

{

Connection.Open();

using NpgsqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

LastQueryExecutionTime = (string)reader.GetValue(0);

}

}

finally

{

Connection.Close();

}

}

public IEnumerable<object> ReadFlightClientCount(long filter)

{

using NpgsqlCommand command = new NpgsqlCommand();

command.Connection = Connection;

command.CommandText = $"select \"Flights\".\"Flight\_ID\", \"Flights\".\"Flight\_Number\", count(\"Client\_ID\") as u\_c\r\nfrom \"Flights\"\r\ninner join \"Hotel-Flights\" using (\"Flight\_ID\")\r\ninner join \"Hotels\" using (\"Hotel\_ID\")\r\ninner join \"Journeys\" using (\"Journey\_ID\")\r\ninner join \"Clients\" using (\"Client\_ID\")\r\nwhere \"Flights\".\"Flight\_Number\" > {filter}\r\ngroup by \"Flights\".\"Flight\_ID\", \"Flights\".\"Flight\_Number\", \"Flights\".\"Departure\", \"Flights\".\"Destination\"\r\nOrder by \"Flights\".\"Flight\_ID\";";

IEnumerable<object> items = new List<object>();

try

{

Connection.Open();

using NpgsqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

((List<object>)items).Add(new

{

Flight\_ID = (long)reader["Flight\_ID"],

Flight\_Number = (int)reader["Flight\_Number"],

Client\_Count = (long)reader["u\_c"]

});

}

}

finally

{

Connection.Close();

}

using var explain = command.Clone();

explain.CommandText = "explain analyze " + command.CommandText;

try

{

Connection.Open();

using NpgsqlDataReader explainReader = explain.ExecuteReader();

while (explainReader.Read())

{

LastQueryExecutionTime = (string)explainReader.GetValue(0);

}

}

finally

{

Connection.Close();

}

return items;

}

}

**HotelModel.cs**

using Npgsql;

using NpgsqlTypes;

using RGR.Models.Entities;

using System.Collections.Generic;

namespace RGR.Models;

public class HotelModel : Model<Hotel>

{

public HotelModel(NpgsqlConnection connection) : base(connection) { }

public override void CreateItem(Hotel item)

{

using NpgsqlCommand command = new NpgsqlCommand();

command.Connection = Connection;

command.Parameters.Add(new NpgsqlParameter("@param\_name", NpgsqlDbType.Varchar, 100) { Value = (object?)item.Hotel\_Name });

command.Parameters.Add(new NpgsqlParameter("@param\_address", NpgsqlDbType.Varchar, 100) { Value = (object?)item.Address });

command.Parameters.Add(new NpgsqlParameter("@param\_rating", NpgsqlDbType.Integer) { Value = (object?)item.Star\_Rating });

command.Parameters.Add(new NpgsqlParameter("@param\_journey", NpgsqlDbType.Bigint) { Value = (object?)item.Journey\_ID });

command.CommandText = $"explain analyze insert into \"Hotels\"(\"Hotel\_Name\", \"Address\", \"Star\_Rating\", \"Journey\_ID\") values(@param\_name, @param\_address, @param\_rating, @param\_journey);";

try

{

Connection.Open();

using NpgsqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

LastQueryExecutionTime = (string)reader.GetValue(0);

}

}

finally

{

Connection.Close();

}

}

public override void DeleteItem(long id)

{

using NpgsqlCommand command = new NpgsqlCommand();

command.Connection = Connection;

command.CommandText = $"explain analyze delete from \"Hotels\" where \"Hotel\_ID\" = {id}";

try

{

Connection.Open();

using NpgsqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

LastQueryExecutionTime = (string)reader.GetValue(0);

}

}

finally

{

Connection.Close();

}

}

public override void GenerateItems(long count)

{

using NpgsqlCommand command = new NpgsqlCommand();

command.Connection = Connection;

string query = $"insert into \"Hotels\"(\"Hotel\_Name\", \"Address\", \"Star\_Rating\", \"Journey\_ID\") \r\nvalues \r\n(\r\n\tchr(trunc(65 + random() \* 25)::int) || chr(trunc(65 + random() \* 25)::int), \r\n\tchr(trunc(65 + random() \* 25)::int) || chr(trunc(65 + random() \* 25)::int), \r\n\ttrunc(1 + random() \* 5)::int, \r\n\t(select f\_id from trunc(1 + random() \* (select max(\"Journey\_ID\") from \"Journeys\")) as f\_id inner join \"Journeys\" on f\_id = \"Journeys\".\"Journey\_ID\")::bigint\r\n);";

command.CommandText = "";

for (int i = 0; i < count; i++)

command.CommandText += query;

try

{

Connection.Open();

using NpgsqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

LastQueryExecutionTime = (string)reader.GetValue(0);

}

}

finally

{

Connection.Close();

}

}

public override IEnumerable<Hotel> ReadItems()

{

using NpgsqlCommand command = new NpgsqlCommand();

command.Connection = Connection;

command.CommandText = $"select \* from \"Hotels\"";

IEnumerable<Hotel> items = new List<Hotel>();

try

{

Connection.Open();

using NpgsqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

Hotel item = new Hotel();

item.Hotel\_ID = (long)reader["Hotel\_ID"];

item.Hotel\_Name = (string)reader["Hotel\_Name"];

item.Address = (string)reader["Address"];

item.Star\_Rating = (int)reader["Star\_Rating"];

item.Journey\_ID = (long)reader["Journey\_ID"];

((List<Hotel>)items).Add(item);

}

}

finally

{

Connection.Close();

}

using var explain = command.Clone();

explain.CommandText = "explain analyze " + command.CommandText;

try

{

Connection.Open();

using NpgsqlDataReader explainReader = explain.ExecuteReader();

while (explainReader.Read())

{

LastQueryExecutionTime = (string)explainReader.GetValue(0);

}

}

finally

{

Connection.Close();

}

return items;

}

public override void UpdateItem(long id, Hotel newItem)

{

using NpgsqlCommand command = new NpgsqlCommand();

command.Connection = Connection;

command.Parameters.Add(new NpgsqlParameter("@param\_name", NpgsqlDbType.Varchar, 100) { Value = (object?)newItem.Hotel\_Name });

command.Parameters.Add(new NpgsqlParameter("@param\_address", NpgsqlDbType.Varchar, 100) { Value = (object?)newItem.Address });

command.Parameters.Add(new NpgsqlParameter("@param\_rating", NpgsqlDbType.Integer) { Value = (object?)newItem.Star\_Rating });

command.Parameters.Add(new NpgsqlParameter("@param\_journey", NpgsqlDbType.Bigint) { Value = (object?)newItem.Journey\_ID });

command.CommandText = $"explain analyze update \"Hotels\" set \"Hotel\_ID\" = {id}, \"Hotel\_Name\" = @param\_name, \"Address\" = @param\_address, \"Star\_Rating\" = @param\_rating, \"Journey\_ID\" = @param\_journey where \"Hotel\_ID\" = {id};";

try

{

Connection.Open();

using NpgsqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

LastQueryExecutionTime = (string)reader.GetValue(0);

}

}

finally

{

Connection.Close();

}

}

public IEnumerable<object> ReadHotelFlightCount(long filterF, long filterS)

{

using NpgsqlCommand command = new NpgsqlCommand();

command.Connection = Connection;

command.CommandText = $"select \"Hotels\".\"Hotel\_ID\", \"Hotels\".\"Hotel\_Name\", count(\"Flights\".\"Flight\_ID\") as \"Flight\_Count\"\r\nfrom \"Hotels\"\r\ninner join \"Hotel-Flights\" using (\"Hotel\_ID\")\r\ninner join \"Flights\" using (\"Flight\_ID\")\r\ngroup by \"Hotels\".\"Hotel\_ID\", \"Hotels\".\"Hotel\_Name\", \"Hotels\".\"Star\_Rating\"\r\nhaving count(\"Flights\".\"Flight\_ID\") between {filterF} and {filterS}\r\nOrder by \"Hotels\".\"Hotel\_ID\";";

IEnumerable<object> items = new List<object>();

try

{

Connection.Open();

using NpgsqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

((List<object>)items).Add(new

{

Hotel\_ID = (long)reader["Hotel\_ID"],

Hotel\_Name = (string)reader["Hotel\_Name"],

Flight\_Count = (long)reader["Flight\_Count"]

});

}

}

finally

{

Connection.Close();

}

using var explain = command.Clone();

explain.CommandText = "explain analyze " + command.CommandText;

try

{

Connection.Open();

using NpgsqlDataReader explainReader = explain.ExecuteReader();

while (explainReader.Read())

{

LastQueryExecutionTime = (string)explainReader.GetValue(0);

}

}

finally

{

Connection.Close();

}

return items;

}

}

**JourneyMode.cs**

using Npgsql;

using NpgsqlTypes;

using RGR.Models.Entities;

namespace RGR.Models;

public class JourneyModel : Model<Journey>

{

public JourneyModel(NpgsqlConnection connection) : base(connection) { }

public override void CreateItem(Journey item)

{

using NpgsqlCommand command = new NpgsqlCommand();

command.Connection = Connection;

command.Parameters.Add(new NpgsqlParameter("@param\_name", NpgsqlDbType.Varchar, 100) { Value = (object?)item.Journey\_Name });

command.Parameters.Add(new NpgsqlParameter("@param\_date", NpgsqlDbType.Timestamp) { Value = (object?)item.Departure\_Date });

command.Parameters.Add(new NpgsqlParameter("@param\_cost", NpgsqlDbType.Money) { Value = (object?)item.Cost });

command.Parameters.Add(new NpgsqlParameter("@param\_client", NpgsqlDbType.Bigint) { Value = (object?)item.Client\_ID });

command.CommandText = $"explain analyze insert into \"Journeys\"(\"Journey\_Name\", \"Departure\_Date\", \"Cost\", \"Client\_ID\") values(@param\_name, @param\_date, @param\_cost, @param\_client);";

try

{

Connection.Open();

using NpgsqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

LastQueryExecutionTime = (string)reader.GetValue(0);

}

}

finally

{

Connection.Close();

}

}

public override void DeleteItem(long id)

{

using NpgsqlCommand command = new NpgsqlCommand();

command.Connection = Connection;

command.CommandText = $"explain analyze delete from \"Journeys\" where \"Journey\_ID\" = {id}";

try

{

Connection.Open();

using NpgsqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

LastQueryExecutionTime = (string)reader.GetValue(0);

}

}

finally

{

Connection.Close();

}

}

public override void GenerateItems(long count)

{

using NpgsqlCommand command = new NpgsqlCommand();

command.Connection = Connection;

string query = $"insert into \"Journeys\"(\"Journey\_Name\", \"Departure\_Date\", \"Cost\", \"Client\_ID\")\r\nvalues \r\n(\r\n\tchr(trunc(65 + random() \* 25)::int) || chr(trunc(65 + random() \* 25)::int),\r\n\t(select timestamp '2020-01-10 20:00:00' + random() \* (timestamp '2023-01-10 10:00:00' - timestamp '2020-01-10 20:00:00'))::timestamp, \r\n\ttrunc(1 + random() \* 2000)::int,\r\n\t(select f\_id from trunc(1 + random() \* (select max(\"Client\_ID\") from \"Clients\")) as f\_id inner join \"Clients\" on f\_id = \"Clients\".\"Client\_ID\")::bigint\r\n);";

command.CommandText = "";

for (int i = 0; i < count; i++)

command.CommandText += query;

try

{

Connection.Open();

using NpgsqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

LastQueryExecutionTime = (string)reader.GetValue(0);

}

}

finally

{

Connection.Close();

}

}

public override IEnumerable<Journey> ReadItems()

{

using NpgsqlCommand command = new NpgsqlCommand();

command.Connection = Connection;

command.CommandText = $"select \* from \"Journeys\"";

IEnumerable<Journey> items = new List<Journey>();

try

{

Connection.Open();

using NpgsqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

Journey item = new Journey();

item.Journey\_ID = (long)reader["Journey\_ID"];

item.Journey\_Name = (string)reader["Journey\_Name"];

item.Departure\_Date = (DateTime)reader["Departure\_Date"];

item.Cost = (decimal)reader["Cost"];

item.Client\_ID = (long)reader["Client\_ID"];

((List<Journey>)items).Add(item);

}

}

finally

{

Connection.Close();

}

using var explain = command.Clone();

explain.CommandText = "explain analyze " + command.CommandText;

try

{

Connection.Open();

using NpgsqlDataReader explainReader = explain.ExecuteReader();

while (explainReader.Read())

{

LastQueryExecutionTime = (string)explainReader.GetValue(0);

}

}

finally

{

Connection.Close();

}

return items;

}

public override void UpdateItem(long id, Journey newItem)

{

using NpgsqlCommand command = new NpgsqlCommand();

command.Connection = Connection;

command.Parameters.Add(new NpgsqlParameter("@param\_name", NpgsqlDbType.Varchar, 100) { Value = (object?)newItem.Journey\_Name });

command.Parameters.Add(new NpgsqlParameter("@param\_date", NpgsqlDbType.Timestamp) { Value = (object?)newItem.Departure\_Date });

command.Parameters.Add(new NpgsqlParameter("@param\_cost", NpgsqlDbType.Money) { Value = (object?)newItem.Cost });

command.Parameters.Add(new NpgsqlParameter("@param\_client", NpgsqlDbType.Bigint) { Value = (object?)newItem.Client\_ID });

command.CommandText = $"explain analyze update \"Journeys\" set \"Journey\_ID\" = {id}, \"Journey\_Name\" = @param\_name, \"Departure\_Date\" = @param\_date, \"Cost\" = @param\_cost, \"Client\_ID\" = @param\_client where \"Journey\_ID\" = {id};";

try

{

Connection.Open();

using NpgsqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

LastQueryExecutionTime = (string)reader.GetValue(0);

}

}

finally

{

Connection.Close();

}

}

public IEnumerable<object> ReadJourneyClients(long filter)

{

using NpgsqlCommand command = new NpgsqlCommand();

command.Connection = Connection;

command.CommandText = $"select \"Journey\_ID\", \"Journey\_Name\", \"Client\_ID\", \"First\_Name\", \"Last\_Name\" \r\nfrom \"Journeys\"\r\ninner join \"Clients\" using (\"Client\_ID\")\r\nwhere \"Journey\_ID\" > {filter}\r\nOrder by \"Journey\_ID\";";

IEnumerable<object> items = new List<object>();

try

{

Connection.Open();

using NpgsqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

((List<object>)items).Add(new

{

Journey\_ID = (long)reader["Journey\_ID"],

Journey\_Name = (string)reader["Journey\_Name"],

Client\_ID = (long)reader["Client\_ID"],

First\_Name = (string)reader["First\_Name"],

Last\_Name = (string)reader["Last\_Name"],

});

}

}

finally

{

Connection.Close();

}

using var explain = command.Clone();

explain.CommandText = "explain analyze " + command.CommandText;

try

{

Connection.Open();

using NpgsqlDataReader explainReader = explain.ExecuteReader();

while (explainReader.Read())

{

LastQueryExecutionTime = (string)explainReader.GetValue(0);

}

}

finally

{

Connection.Close();

}

return items;

}

}

**Model.cs**

using Npgsql;

namespace RGR.Models;

public abstract class Model<T> where T : class, new()

{

public NpgsqlConnection Connection { get; private set; }

public Model(NpgsqlConnection connection)

{

Connection = connection;

}

public abstract void CreateItem(T item);

public abstract IEnumerable<T> ReadItems();

public abstract void UpdateItem(long id, T newItem);

public abstract void DeleteItem(long id);

public abstract void GenerateItems(long count);

public string LastQueryExecutionTime { get; protected set; }

}