МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО”

Факультет прикладної математики
Кафедра програмного забезпечення комп’ютерних систем

Лабораторна робота № 2

з дисципліни “Основи програмування”
тема “Бази даних та СКБД”

Виконав
студент І курсу
групи КП-01
Пецеля Артем Володимирович
(*прізвище, ім’я, по батькові*)

Перевірив
“____” “_____” 20___ р.
викладач
Гадиняк Руслан Анатолійович
(*прізвище, ім’я, по батькові*)

варіант №12

Київ 2020

Мета роботи

Навчитись керувати даними за допомогою СКБД SQLite.
Використати пагінацію для перебору великої кількості даних.
Створити простий консольний інтерфейс користувача для взаємодії з базою даних.

Постановка завдання

Створити консольну програму для взаємодії базою даних сутностей за варіантом (з попередньої лабораторної роботи).

Користувач має можливість вводити команди для:

- **getById** - отримання сутності за ідентифікатором
- **deleteById** - видалення сутності за ідентифікатором
- **insert** - додавання нової сутності
- **getTotalPages** - отримання кількості сторінок з сутностями
- **getPage** - отримання сторінки з сутностями за номером сторінки (нумерація з 1)
- **export** - експорт обраних сутностей у CSV файл
- **exit** - вихід з програми.

Після введення команди користувачу показується її результат або повідомлення про помилку. Після цього користувач може ввести наступну команду.

Формат команд:

Назва	Формат	Приклад
`getById`	`getById {idInteger}`	`getById 13`
`deleteById`	`deleteById {idInteger}`	`deleteById 40`
`insert`	`insert {field1},{field2},..,{fieldN}`	`insert New book,Author 1,1997`

`getTotalPages`	`getTotalPages`	`getTotalPages`
`getPage`	`getPage {pageNumberInteger}`	`getPage 2`
`export`	`export {valueX}`	`export 1990`
`exit`	`exit`	`exit`

Зауваження:

- **getById** виводить всі дані сутності, або повідомлення про те, що сутність з таким ідентифікатором не було знайдено
- **deleteById** - виводить статус виконання операції (видалено чи ні)
- При додаванні нової сутності (**insert**) користувач не задає ідентифікатор. Ідентифікатор визначає сама база даних. Дані сутності, що додається, можна задавати через коми (як у прикладі) або придумати інший формат, наприклад: `insert authors=Author 1&title=New book`. Якщо не було задано всі дані можна або використати для них деякі значення за замовчуванням або виводити користувачу повідомлення про помилку. Якщо введені значення невірні (наприклад, замість числа було введено літери) - виводити повідомлення про помилку.
- **getTotalPages** - виводить число сторінок. Розмір однієї сторінки - **10 елементів**.
- **getPage** - отримує за номером сторінки **10 елементів**, що належать цій сторінці (менше, якщо остання сторінка неповна). ***Нумерацію сторінок починати з 1***. Якщо номер сторінки введений невірно (від’ємне число, нуль, або такої сторінки не існує) - вивести повідомлення про помилку.
- **export** - отримати з БД сутності, що відповідають умові за варіантом (див. ***таблицю А в додатках***) зі значенням **{valueX}**, яке вводить користувач. Отримані сутності записати у файл `**export.csv**` у

форматі CSV. Сповістити користувача про назву файлу, у який було записано дані та кількість експортованих рядків даних.

Тексти коду програм

Program.cs

```
using System;
using System.Linq;
using System.Collections.Generic;
using static System.Console;
using Microsoft.Data.Sqlite;
using System.IO;

namespace lab2
{
    class Capital
    {
        public string name;
        public string country;
        public int population;
        public double area;
        public DateTime createdAt;
        public Capital()
        {

        }
        public Capital(string name, string country, int population, double area)
        {
            this.name = name;
            this.country = country;
            this.population = population;
            this.area = area;
        }
        public override string ToString()
        {
            return string.Format("Name: {0} - Country: {1} - Population: {2} - Area: {3} - Created at:
{4}",name, country, population, area, createdAt.ToString("o"));
        }
    }
    class ListCapital
    {
        private Capital[] _items;
        private int _size;
        public ListCapital()
        {
            this._items = new Capital[16];
            this._size = 0;
        }
        public void Add(Capital newCapital)
        {
            if (_size == _items.Length)
            {
                EnsureCapasity();
            }
            this._items[this._size] = newCapital;
            this._size += 1;
        }
        public void Insert(int index, Capital newCapital)
        {
            if (_size == _items.Length)
            {
                EnsureCapasity();
            }
            for (int i = _size - 1; i >= index; i--)
            {
                _items[i + 1] = _items[i];
            }
            _items[index] = newCapital;
            _size += 1;
        }
        public bool Remove(Capital newCapital)
        {
```

```
            for (int i = 0; i < _size; i++)
            {
                if (_items[i] == newCapital)
                {
                    for (int j = i; j < _size - 1; j++)
                    {
                        _items[j] = _items[j + 1];
                    }
                    _size -= 1;
                    _items[_size] = null;
                    return true;
                }
            }
            return false;
        }
        public void RemoveAt(int index)
        {
            if (index >= _size)
            {
                throw new IndexOutOfRangeException();
            }
            if (index != _size - 1)
            {
                for (int i = index; i < _size - 1; i++)
                {
                    _items[i] = _items[i + 1];
                }
            }
            _size -= 1;
            _items[_size] = null;
        }
        public void Clear()
        {
            int capacity = _items.Length;
            _items = new Capital[capacity];
            _size = 0;
        }
        public int Count
        {
            get
            {
                return _size;
            }
        }
        public int Capacity
        {
            get
            {
                return _items.Length;
            }
        }
        public Capital this[int index]
        {
            get
            {
                if (index >= _size)
                {
                    throw new IndexOutOfRangeException();
                }
                return _items[index];
            }
            set
            {
                if (index >= _size)
                {
                    throw new IndexOutOfRangeException();
                }
                _items[index] = value;
            }
        }

        public IEnumerator<Capital> GetEnumerator()
        {
```

```
            return this._items.Take(this._size).GetEnumerator();
        }

        private void EnsureCapasity()
        {
            int oldCapacity = _items.Length;
            Capital[] oldArray = _items;
            _items = new Capital[oldCapacity * 2];
            Array.Copy(oldArray, _items, oldCapacity);
        }
    }
    class CapitalRepository
    {
        private SqliteConnection connection;
        public CapitalRepository(SqliteConnection connection)
        {
            this.connection = connection;
        }

        public Capital GetById(int id)
        {
            SqliteCommand command = connection.CreateCommand();
            command.CommandText = "SELECT * FROM capitals WHERE id = $id";
            command.Parameters.AddWithValue("$id", id);
            SqliteDataReader reader = command.ExecuteReader();

            Capital capital = new Capital();
            if (reader.Read())
            {
                capital.name = reader.GetString(1);
                capital.country = reader.GetString(2);
                capital.population = reader.GetInt32(3);
                capital.area = reader.GetFloat(4);
                capital.createdAt = DateTime.Parse(reader.GetString(5));
            }
            reader.Close();

            return capital;
        }
        public int DeleteById(int id)
        {
            SqliteCommand command = connection.CreateCommand();
            command.CommandText = "DELETE FROM capitals WHERE id = $id";
            command.Parameters.AddWithValue("$id", id);
            int nChanged = command.ExecuteNonQuery();
            return nChanged;
        }
        public int Insert(Capital capital)
        {
            SqliteCommand command = connection.CreateCommand();
            command.CommandText =
                @"
                    INSERT INTO capitals (name, country, population, area, createdAt)
                    VALUES ($name, $country, $population, $area, $createdAt);

                    SELECT last_insert_rowid();
                ";
            command.Parameters.AddWithValue("$name", capital.name);
            command.Parameters.AddWithValue("$country", capital.country);
            command.Parameters.AddWithValue("$population", capital.population);
            command.Parameters.AddWithValue("$area", capital.area);
            command.Parameters.AddWithValue("$createdAt", capital.createdAt.ToString("o"));

            long newId = (long)command.ExecuteScalar();
            return Convert.ToInt32(newId);
        }
        public int GetTotalPages()
        {
            const int pageSize = 10;

            SqliteCommand command = connection.CreateCommand();
            command.CommandText = @"SELECT COUNT(*) FROM capitals";
            long count = (long)command.ExecuteScalar();
```

```
            return (int)Math.Ceiling(count / (double)pageSize);
        }
        public ListCapital GetPage(int pageNumber)
        {
            const int pageSize = 10;
            int numberOfPages = GetTotalPages();
            if (pageNumber > numberOfPages || pageNumber <= 0)
            {
                throw new Exception("Page out of number of pages");
            }
            SqliteCommand command = connection.CreateCommand();
            command.CommandText = "SELECT * FROM capitals LIMIT $limit OFFSET $offset";
            command.Parameters.AddWithValue("$limit", pageSize);
            command.Parameters.AddWithValue("$offset", (pageNumber - 1) * pageSize);
            SqliteDataReader reader = command.ExecuteReader();

            ListCapital page = new ListCapital();
            while (reader.Read())
            {
                Capital capital = new Capital
                {
                    name = reader.GetString(1),
                    country = reader.GetString(2),
                    population = reader.GetInt32(3),
                    area = reader.GetFloat(4),
                    createdAt = DateTime.Parse(reader.GetString(5))
                };

                page.Add(capital);
            }
            reader.Close();

            return page;
        }

        public ListCapital GetExport(DateTime date)
        {
            SqliteCommand command = connection.CreateCommand();
            command.CommandText = @"SELECT * FROM capitals WHERE createdAt < $time";
            command.Parameters.AddWithValue("$time", date.ToString("o"));

            SqliteDataReader reader = command.ExecuteReader();
            ListCapital list = new ListCapital();
            while(reader.Read())
            {
                Capital capital = new Capital
                {
                    name = reader.GetString(1),
                    country = reader.GetString(2),
                    population = reader.GetInt32(3),
                    area = reader.GetFloat(4),
                    createdAt = DateTime.Parse(reader.GetString(5))
                };

                list.Add(capital);
            }
            reader.Close();

            return list;
        }
    }
    class Program
    {
        static void GetHelp()
        {
            string[] commands = new string[] { "getById {idInteger}", "deleteById { idInteger }", "insert
{ name }&{ country }&{ population }&{ area }&{ createdAt }", "getTotalPages", "getPage
{pageNumberInteger}", "export {date and time}", "exit"};
            WriteLine("Commands:");
            foreach (string command in commands)
            {
                WriteLine(command);
            }
```

```
        }
        static void WriteAllCapitals(ListCapital capitals)
        {
            const string outputPath = @".\export.csv";
            StreamWriter sw = new StreamWriter(outputPath);
            foreach (Capital capital in capitals)
            {
                sw.WriteLine(capital.name + "," + capital.country + "," + capital.population + "," +
capital.area + "," + capital.createdAt.ToString("o"));
            }
            sw.Close();
        }
        static void Main()
        {
            string databaseFilePath = @"D:\progbase\labs\second_sem\lab2\capitals.db";
            SqliteConnection connection = new SqliteConnection($"Data Source = {databaseFilePath}");
            connection.Open();
            CapitalRepository rep = new CapitalRepository(connection);
            WriteLine("Type help for commands");
            bool exit = false;
            while (!exit)
            {
                Write("> ");
                string input = ReadLine();
                if (input == "help")
                {
                    GetHelp();
                }
                else if (input.StartsWith("getById") && input.Split(" ").Length == 2)
                {
                    if (!int.TryParse(input.Split(" ")[1], out int id))
                    {
                        WriteLine("Wrong id");
                        continue;
                    }
                    Capital capital = rep.GetById(id);
                    if (capital.name != null)
                    {
                        WriteLine(capital);
                    }
                    else
                    {
                        WriteLine("Such capital was not found");
                    }
                }
                else if (input.StartsWith("deleteById") && input.Split(" ").Length == 2)
                {
                    if (!int.TryParse(input.Split(" ")[1], out int id))
                    {
                        WriteLine("Wrong id");
                        continue;
                    }
                    int status = rep.DeleteById(id);
                    if (status == 0)
                    {
                        WriteLine("Capital was not deleted");
                    }
                    else
                    {
                        WriteLine("Capital was deleted");
                    }
                }
                else if (input.StartsWith("insert"))
                {
                    Capital capital = new Capital();
                    string[] args = input.Substring(7).Split("&");
                    if (args.Length != 5)
                    {
                        WriteLine("Unknown command");
                        continue;
                    }
                    if (!(int.TryParse(args[2], out capital.population) && double.TryParse(args[3], out
capital.area) && DateTime.TryParse(args[4], out capital.createdAt)))
```

```
                    {
                        WriteLine("Incorrect format");
                        continue;
                    }
                    capital.name = args[0];
                    capital.country = args[1];
                    int index = rep.Insert(capital);
                    if (index != 0)
                    {
                        WriteLine($"Capital was added with index {index}");
                    }
                    else
                    {
                        WriteLine("Capital was not added");
                    }
                }
                else if (input == "getTotalPages")
                {
                    WriteLine($"Pages: {rep.GetTotalPages()}");
                }
                else if (input.StartsWith("getPage") && input.Split(" ").Length == 2)
                {
                    if (!int.TryParse(input.Split(" ")[1], out int pageNumber))
                    {
                        WriteLine("Wrong page number");
                        continue;
                    }
                    try
                    {
                        ListCapital page = rep.GetPage(pageNumber);
                        foreach (Capital capital in page)
                        {
                            WriteLine(capital);
                        }
                    }
                    catch
                    {
                        WriteLine("Wrong page");
                    }
                }
                else if (input.StartsWith("export") && input.Split(" ").Length == 2)
                {
                    if (!DateTime.TryParse(input.Split(" ")[1], out DateTime date))
                    {
                        WriteLine("Incorrect date format");
                        continue;
                    }
                    ListCapital capitals = rep.GetExport(date);
                    WriteAllCapitals(capitals);
                    WriteLine($"{capitals.Count} capitals was written to \"export.csv\"");
                }
                else if (input == "exit")
                {
                    exit = true;
                }
                else
                {
                    WriteLine("Unknown command");
                }
            }
            connection.Close();
        }
    }
}
```

Приклади результатів

Приклад 1. Команда getById.

```
> getById 5
Name: Manama - Country: Benin - Population: 695539 - Area: 219,7969970703125 - Created at:
2021-03-01T00:00:00.0000000
```

Приклад 2. Команда deleteById.

```
> deleteById 5
Capital was deleted
```

Приклад 3. Команда insert.

```
> insert Manama&Benin&695539&219,796&2010-12-31
Capital was added with index 20004
```

Приклад 4. Команда getTotalPages.

```
> getTotalPages
Pages: 2001
```

Приклад 5. Команда getPage.

```
> getPage 2001
Name:  New Dehli - Country: India - Population: 200 - Area: 321,12298583984375 - Created at:
1323-05-10T00:00:00.0000000
Name: Manama - Country: Benin - Population: 695539 - Area: 219,79600524902344 - Created at:
2010-12-31T00:00:00.0000000
```

Приклад 6. Команда export.

```
> export 2020-12-31
5 capitals was written to "export.csv"
```

export.csv

```
Jerusalem,Brazil,807879,379,11700439453125,2020-03-01T00:00:00.0000000
West Island,Cape Verde,885730,20,94700050354004,1980-03-01T00:00:00.0000000
Kyiv,Ukraine,3000000,124,22200012207031,0500-04-01T00:00:00.0000000
New Dehli,India,200,321,12298583984375,1323-05-10T00:00:00.0000000
Manama,Benin,695539,219,79600524902344,2010-12-31T00:00:00.0000000
```

Висновки

Виконавши дану лабораторну роботу було створено консольну утиліту, яка обробляє команди користувача та реагує на них відповідним чином. Під час виконання програма доступається до бази даних SQLite за допомогою SQL команд. Доступ до бази даних в коді зроблений через окремий клас CapitalRepository.

Программа компілювалася за допомогою платформи dotnet.