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	Перевірив
студент І курсу	“____” “_____” 20___ р.
групи КП-01	викладач
Беліцький Олександр Сергійович	Гадиняк Руслан Анатолійович
(*прізвище, ім’я, по батькові*)	(*прізвище, ім’я, по батькові*)

варіант № 3

Київ 2021

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 вчителів

Користувач має можливість вводити команди для:
getById - отримання вчителя за ідентифікатором
deleteById - видалення вчителя за ідентифікатором
insert - додавання нової вчителя
getTotalPages - отримання кількості сторінок з вчителями
getPage - отримання сторінки з вчителями за номером сторінки (нумерація з 1)
export - експорт обраних вчителів 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.

Аналіз вимог і проектування

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, задавати через коми. Якщо не було задано всі дані 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 вчителя, У полі ***fullname*** шукати всі сутності, у яких значення по цьому полю містить підрядок зі значенням {**valueX**}. 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 static System.Console;
using Microsoft.Data.Sqlite;

namespace lab_2
{
    class Teacher
    {
        public int id;
        public string fullname;
        public string subject;
        public int age;

        public Teacher()
        {
            id = 0;
            fullname = "";
            subject = "";
            age = 0;
        }

        public Teacher(int id, string fullname, string subject, int age)
        {
            this.id = id;
            this.fullname = fullname;
            this.subject = subject;
            this.age = age;
        }

        public override string ToString()
        {
            return $"{id,-8} {fullname,20} - {subject,-10} | {age,3}";
        }
    }

    class ListTeachers
    {
        private Teacher[] _items;
        private int _size;

        public ListTeachers()
        {
            _items = new Teacher[16];
            _size = 0;
        }

        public void Add(Teacher newTeacher)
        {
            if (this._size == this._items.Length)
            {
                Expand();
            }
            this._items[this._size] = newTeacher;
```

```
        this._size += 1;
    }

    private void Expand()
    {
        int oldCapacity = this._items.Length;
        Teacher[] oldArray = this._items;
        this._items = new Teacher[oldCapacity * 2];
        System.Array.Copy(oldArray, this._items, oldCapacity);
    }

    public void Insert(int index, Teacher teacher)
    {
        if ((index > (_size)) || (index < 0))
        {
            WriteLine("Error: Index does not exist");
            Environment.Exit(0);
        }
        if (this._size == this._items.Length)
        {
            Expand();
        }
        for (int i = _size; i >= index; i--)
        {
            _items[i] = _items[i - 1];
        }
        _items[index] = teacher;
        _size += 1;
    }

    public bool Remove(Teacher teacher)
    {
        for (int i = 0; i <= _size; i++)
        {
            if (_items[i] == teacher)
            {
                _size -= 1;
                for (int j = i; j < _size; j++)
                {
                    _items[j] = _items[j + 1];
                }
                return true;
            }
        }
        return false;
    }


    public int GetCount()
    {
        return _size;
    }

    public int GetCapacity()
    {
        return _items.Length;
    }
```

```
        public Teacher GetAt(int index)
        {
            if ((index > (_size - 1)) || (index < 0))
            {
                WriteLine("Error: Teacher under this index does not exist");
                Environment.Exit(0);
            }
            return _items[index];
        }

        public void SetAt(int index, Teacher teacher)
        {
            if ((index > (_size - 1)) || (index < 0))
            {
                WriteLine("Error: Teacher under this index does not exist");
                Environment.Exit(0);
            }
            _items[index] = teacher;
        }

        public double AverageAge(ListTeachers list)
        {
            int sum = 0;
            for (int i = 0; i < list._size; i++)
            {
                sum += list._items[i].age;
            }
            double avg = sum / list._size;
            return avg;
        }

        public ListTeachers ClearList()
        {
            return new ListTeachers();
        }
    }

    class TeacherRepository
    {
        private SqliteConnection connection;
        public TeacherRepository(SqliteConnection connection)
        {
            this.connection = connection;
        }

        public Teacher GetById(int id)
        {
            connection.Open();

            SqliteCommand command = connection.CreateCommand();
            command.CommandText = @"SELECT * FROM teachers WHERE id = $id";
            command.Parameters.AddWithValue("$id", id);

            SqliteDataReader reader = command.ExecuteReader();

            Teacher teacher = new Teacher();

            if (reader.Read())
```

```
            {
                teacher = GetTeacher(reader);
            }

            reader.Close();

            connection.Close();

            return teacher;
        }
        public int DeleteById(int id)
        {
            connection.Open();

            SqliteCommand command = connection.CreateCommand();
            command.CommandText = @"DELETE FROM teachers WHERE id = $id";
            command.Parameters.AddWithValue("$id", id);

            int nChanged = command.ExecuteNonQuery();

            connection.Close();
            return nChanged;
        }
        public long Insert(Teacher teacher)
        {
            connection.Open();

            SqliteCommand command = connection.CreateCommand();
            command.CommandText =
            @"
            INSERT INTO teachers (fullname, subject, age)
            VALUES ($fullname, $subject, $age);

            SELECT last_insert_rowid();
            ";
            command.Parameters.AddWithValue("$fullname", teacher.fullname);
            command.Parameters.AddWithValue("$subject", teacher.subject);
            command.Parameters.AddWithValue("$age", teacher.age);

            long newId = (long)command.ExecuteScalar();

            connection.Close();
            return newId;
        }
        public int GetTotalPages()
        {
            const int pageSize = 10;
            return (int)Math.Ceiling(this.GetCount() / (double)pageSize);
        }

        private long GetCount() //for GetTotalPages
        {
            connection.Open();

            SqliteCommand command = connection.CreateCommand();
            command.CommandText = @"SELECT COUNT(*) FROM teachers";

            long count = (long)command.ExecuteScalar();
```

```
            return count;
        }

        public ListTeachers GetPage(int pageNumber)
        {
            connection.Open();
            SqliteCommand command = connection.CreateCommand();
            command.CommandText = @"SELECT * FROM teachers LIMIT 10 OFFSET
10*($pageNumber-1)";
            command.Parameters.AddWithValue("$pageNumber", pageNumber);
            SqliteDataReader reader = command.ExecuteReader();
            ListTeachers teachers = new ListTeachers();
            while (reader.Read())
            {
                teachers.Add(GetTeacher(reader));
            }
            reader.Close();
            connection.Close();
            return teachers;
        }

        public ListTeachers GetExport(string valueX)
        {
            connection.Open();

            SqliteCommand command = connection.CreateCommand();
            command.CommandText = @"SELECT * FROM teachers WHERE fullname
LIKE '%' || $str || '%'";
            command.Parameters.AddWithValue("$str", valueX);

            SqliteDataReader reader = command.ExecuteReader();

            ListTeachers teachers = new ListTeachers();
            while (reader.Read())
            {
                teachers.Add(GetTeacher(reader));
            }

            reader.Close();
            connection.Close();
            Export(teachers);
            return teachers;
        }

        public bool Export(ListTeachers teachers)
        {
            System.IO.File.WriteAllText("./export.csv", String.Empty);
            System.IO.StreamWriter sw = new
System.IO.StreamWriter("./export.csv");
            string s = "";
            int i = 1;
            while (true)
            {
                string[] str = new string[] {
teachers.GetAt(i).id.ToString(), teachers.GetAt(i).fullname,
teachers.GetAt(i).subject, teachers.GetAt(i).age.ToString() };
                s = string.Join(',', str);
                if (s == null)
```

```
                {
                    break;
                }
                sw.WriteLine(s);
                i++;
                if (i == teachers.GetCount())
                {
                    break;
                }
            }
            sw.Close();
            if (i == 0)
            {
                return false;
            }
            else
            {
                return true;
            }

        }

        static Teacher GetTeacher(SqliteDataReader reader)
        {
            Teacher teacher = new Teacher();
            teacher.id = int.Parse(reader.GetString(0));
            teacher.fullname = reader.GetString(1);
            teacher.subject = reader.GetString(2);
            teacher.age = int.Parse(reader.GetString(3));

            return teacher;
        }

    }

    class Program
    {
        static void Main(string[] args)
        {
            string databaseFileName = "teachersdb";
            SqliteConnection connection = new SqliteConnection($"Data
Source={databaseFileName}");
            TeacherRepository repository = new TeacherRepository(connection);
            PrintCommands();
            Console.ForegroundColor = ConsoleColor.Green;
            string command = ReadLine();
            Console.ResetColor();
            while (true)
            {
                string[] arrayCommand = command.Split(' ');
                arrayCommand[0] = arrayCommand[0].ToLower();
                Console.ForegroundColor = ConsoleColor.Yellow;

                if (ChooseProccess( arrayCommand, repository))
                {
                    break;
                }
```

```
                Console.ResetColor();
                PrintCommands();
                Console.ForegroundColor = ConsoleColor.Green;
                command = ReadLine();
                Console.ResetColor();

            }
        }

        static void PrintTeachers(ListTeachers teachers)
        {
            for (int i = 0; i < teachers.GetCount(); i++)
            {
                WriteLine(teachers.GetAt(i));
            }
        }

        static void PrintCommands()
        {
            WriteLine(@"
Available commands to work with the database:

/c o m m a n d/   /f o r m a t/                              /e x e c u t i o
n/
+-------------------------------------------------------------------------
------------------------+
getById         - getById {idInteger}                    -  obtaining the
teacher by ID
deleteById      - deleteById {idInteger}                 -  deleting the
teacher by ID
insert          - insert {fullname},{subject},{age}      -  adding a new
teacher
getTotalPages   - getTotalPages                          -  getting the number
of pages with teachers
getPage         - getPage {pageNumberInteger}            -  getting a page
with entities by page number
export          - export {substringOfFullname}           -  export selected
teachers to a CSV file
exit            - exit                                   -  exit the program
+-------------------------------------------------------------------------
------------------------+
            ");
        }

        static void GetByIdProccessing(string[] arrayCommand,
TeacherRepository repository)
        {
            int id;
            if (arrayCommand.Length != 2)
            {
                WriteLine("Error: check correctness of input data");
                return;
            }
            else if (int.TryParse(arrayCommand[1], out id))
            {
                if (id <= 0)
                {
                    WriteLine("Error: Id must be positive number");
```

```
                    return;
                }
                else
                {
                    Teacher teacher = repository.GetById(id);
                    Teacher nullTeacher = new Teacher();
                    if (teacher.id != 0)
                    {
                        WriteLine("Teacher found: " + teacher);
                    }
                    else
                    {
                        Console.WriteLine("Teacher NOT found.");
                    }

                }
            }
            else
            {
                WriteLine("Error: Id must be a number");
                return;
            }
        }
        static void DeleteByIdProccessing(string[] arrayCommand,
TeacherRepository repository)
        {
            int id;
            if (arrayCommand.Length != 2)
            {
                WriteLine("Error: check correctness of input data");
                return;
            }
            else if (int.TryParse(arrayCommand[1], out id))
            {
                if (id <= 0)
                {
                    WriteLine("Error: Id must be positive number");
                    return;
                }
                else
                {
                    int deleted = repository.DeleteById(id);
                    if (deleted == 0)
                    {
                        Console.WriteLine("Teacher NOT deleted.");
                    }
                    else
                    {
                        Console.WriteLine("Teacher deleted.");
                    }


                }
            }
            else
            {
                WriteLine("Error: Id must be a number");
                return;
```

```
            }
        }

        static void InsertProccessing(string[] arrayCommand,
TeacherRepository repository)
        {
            if (arrayCommand.Length != 2)
            {
                WriteLine("Error: Check correctness of input data");
                return;
            }
            else
            {
                string[] data = arrayCommand[1].Split(',');
                if (data.Length != 3)
                {
                    WriteLine("Error: Check correctness of input data of
teacher");
                    return;
                }
                else
                {
                    if (int.TryParse(data[0], out int id))
                    {
                        WriteLine("Error: Check correctness of input data of
teacher: do not point id");
                        return;
                    }
                    else
                    {
                        if (int.TryParse(data[2], out int age))
                        {
                            if (data[0] != "")
                            {
                                if (data[1] != "")
                                {
                                    Teacher teacher = new Teacher();
                                    teacher.fullname = data[0];
                                    teacher.subject = data[1];
                                    teacher.age = age;

                                    long newId = repository.Insert(teacher);
                                    if (newId == 0)
                                    {
                                        Console.WriteLine("Teacher NOT
added.");
                                    }
                                    else
                                    {
                                        Console.WriteLine("Teacher added. New
id is: " + newId);
                                    }
                                }
                                else
                                {
                                    WriteLine("Error: Check correctness of
input data of teacher: enter subject of teaching");
                                    return;
```

```
                                }
                            }
                            else
                            {
                                WriteLine("Error: Check correctness of input
data of teacher: enter teacher name");
                                return;
                            }
                        }
                        else
                        {
                            WriteLine("Error: Check correctness of input data
of teacher: age must be a number");
                            return;
                        }
                    }
                }
            }

        }

        static void GetTotalPageProccessing(string[] arrayCommand,
TeacherRepository repository)
        {
            if (arrayCommand.Length != 1)
            {
                WriteLine("Error: Do not enter anything else only command");
            }
            else
            {
                WriteLine("Number of pages: " + repository.GetTotalPages());
            }
        }

        static void GetPageProccessing(string[] arrayCommand,
TeacherRepository repository)
        {
            int page;
            if (arrayCommand.Length != 2)
            {
                WriteLine("Error: check correctness of input data");
                return;
            }
            else if (int.TryParse(arrayCommand[1], out page))
            {
                if (page <= 0)
                {
                    WriteLine("Error: Id must be positive number");
                    return;
                }
                else
                {
                    ListTeachers teachers = repository.GetPage(page);
                    PrintTeachers(teachers);
                }
            }
            else
            {
```

```
                WriteLine("Error: Id must be a number");
                return;
            }
        }

        static void ExportProccessing(string[] arrayCommand,
TeacherRepository repository)
        {
            if (arrayCommand.Length != 2)
            {
                WriteLine("Error: check correctness of input data");
                return;
            }
            else
            {
                ListTeachers teachers =
repository.GetExport(arrayCommand[1]);
                WriteLine("Tip: Data was exported to export.csv\r\nTeachers
recorded: " + teachers.GetCount());
            }
        }

        static bool ChooseProccess(string[] arrayCommand, TeacherRepository
repository)
        {
            if (arrayCommand[0] == "getbyid")
            {
                GetByIdProccessing(arrayCommand, repository);
            }
            else if (arrayCommand[0] == "deletebyid")
            {
                DeleteByIdProccessing(arrayCommand, repository);
            }
            else if (arrayCommand[0] == "insert")
            {
                InsertProccessing(arrayCommand, repository);
            }
            else if (arrayCommand[0] == "gettotalpages")
            {
                GetTotalPageProccessing(arrayCommand, repository);
            }
            else if (arrayCommand[0] == "getpage")
            {
                GetPageProccessing(arrayCommand, repository);
            }
            else if (arrayCommand[0] == "export")
            {
                ExportProccessing(arrayCommand, repository);
            }
            else if (arrayCommand[0] == "exit")
            {
                if (arrayCommand.Length != 1)
                {
                    WriteLine("Error: Do not enter anything else only
command");
                }
                else
                {
```

```
                    WriteLine("Ending poccessing...");
                    return true;
                }
            }
            else
            {
                WriteLine("Error: check correctness of input data");
            }
            return false;
        }
    }
}
```

Приклади результатів

*Всі команди можна писати назалежно від регістру.

- неправильні вхідні дані
- правильні вхідні дані
- початок групи команд
- команда не передбачена для виконання

getbyid incorrect input data:

getbyid

```
getbyid
Error: Id must be a number
```

getbyid *yy*

```
getbyid yy
Error: Id must be a number
```

getbyid *66 y*

```
getbyid 66 y
Error: check correctness of input data
```

getbyid *n (n <=0)*

```
getbyid -1
Error: Id must be positive number
```

getbyid *66*

```
getbyid 66
Teacher found: 66          Prokhorov Ludwig - History    |  66
```

deletebyid incorrect input data:

deletebyid

```
deletebyid
Error: Id must be a number
```

deletebyid *yy*

```
deletebyid yy
Error: Id must be a number
```

deletebyid *66 y*

```
deletebyid 66 y
Error: check correctness of input data
```

deletebyid *n (n <=0)*

```
deletebyid -1
Error: Id must be positive number
```

deletebyid *20003*

```
deletebyid 20003
Teacher deleted.
```

insert incorrect input data:

insert *55,sss,ddd,55*

```
insert 55,sss,ddd,55
Error: Check correctness of input data of teacher
```

insert *55,sss,sss*

```
insert 55,sss,sss
Error: Check correctness of input data of teacher: do not point id
```

insert *sss,sss,sss*

```
insert sss,sss,sss
Error: Check correctness of input data of teacher: age must be a number
```

insert *sss,sss,55*

```
insert sss,sss,55
Teacher added. New id is: 20004
```

gettotalpages incorrect input data:

gettotalpages *smth*

```
gettotalpages smth
Error: Do not enter anything else only command
```

gettotalpages

```
gettotalpages
Number of pages: 2001
```

getpage incorrect input data:

getpage

```
getpage
Error: Id must be a number
```

getpage *yy*

```
getpage yy
Error: Id must be a number
```

getpage *n (n <=0)*

```
getpage 0
Error: Id must be positive number
```

getpage 5

```
getpage 5
41          Prokhorov Ludwig - Chemistry  |  78
42              Rusakov Yuri - Biology    |  78
43              Rusakov Yuri - Health     |  45
44            Ilyin Mechislav - Art       |  23
45          Bespalov Mitrofan - Geometry  |  78
46            Stepanov Gordey - Art       |  78
47            Ilyin Mechislav - Geography |  23
48          Bespalov Mitrofan - History   |  33
49          Bespalov Mitrofan - Geography |  78
50           Prokhorov Ludwig - Geography |  33
```

export incorrect input data:

export

```
export
Error: check correctness of input data
```

export *g*

```
export g
Tip: Data was exported to export.csv
Teachers recorded: 3872
```

exit incorrect input data:

exit *smth*

```
exit smth
Error: Do not enter anything else only command
```

exit

```
exit
Ending poccessing...
```

unknowncommand

```
unknowncommand
Error: check correctness of input data
```

Висновки

Виконавши дану лабораторну роботу були вивчені основи мови (анг. structured query language), що застосовується для формування запитів, оновлення і керування БД.

Виконуючи дану лабораторну роботу було ознайомлено з СКБД "DB Browser for SQLite". Головні функції СКБД – це визначення даних, обробка даних і керування даними.

Будь-яка СКБД дозволяє виконувати чотири найпростіші операції з даними:

- додавати в таблицю один або кілька записів;
- видаляти з таблиці один або кілька записів;
- обновляти значення деяких полів в одному або декількох записах;
- знаходити одну або кілька записів, що задовольняють заданій умові.

Було досліджено метод пагінації для швидшої обробки та виведення даних. Також було покращено навички з розробки простого консольного інтерфейсу користувача для взаємодії з базою даних.

Для роботи із СКБД було використано пакет Microsoft.Data.Sqlite. Компіляція всього коду відбувалася за допомогою утиліти dotnet.