Солтисюк Дмитро, студент ТУ-12-21-ІПЗ

Лабораторна робота №4.2

Тема

Цілісність даних.

Мета

Розробити програму для багатократного завантаження інформації про результати сесії у базу даних, створену в рамках лабораторної роботи 4.1.

Вступ

Цей звіт описує програму "SessionResultsUploader", яка розроблена для багатократного завантаження інформації про результати сесії у базу даних SQLite. Програма дозволяє користувачу створювати студентів, предмети та вводити результати екзаменів.

Функціонал програми

Програма "SessionResultsUploader" має такий функціонал:

- Створення студентів: користувач може ввести повне ім’я студента, і програма додає цього студента до бази даних.
- Створення предметів: користувач вводить назву предмету та дату екзамену, і програма додає цей предмет до бази даних.
- Введення результатів екзаменів: користувач може ввести ID студента, ID предмету та оцінку за екзамен, і програма додає цей результат до бази даних.
- Імпорт даних з CSV файлу: програма може імпортувати дані з CSV файлу, що містить інформацію про студентів, предмети та результати екзаменів.

Процес розробки

Для розробки програми "SessionResultsUploader" були використані такі технології та інструменти:

- Мова програмування C#: програма розроблена мовою програмування C# з використанням технології .NET Core.
- База даних SQLite: для зберігання даних було використано легковагову базу даних SQLite.
- Бібліотека Microsoft.Data.Sqlite: для взаємодії з базою даних SQLite було використано бібліотеку Microsoft.Data.Sqlite.
- Консольний інтерфейс: програма має консольний інтерфейс користувача для взаємодії з програмою та введення даних.

Додаткові особливості

Під час розробки програми "SessionResultsUploader" було приділено увагу таким аспектам:

- Обробка помилок: програма має механізм обробки помилок, який дозволяє коректно впоратися з непередбаченими ситуаціями та повідомляти користувача про помилки.
- Код якості: під час написання коду було дотримано принципів чистого коду та кращих практик програмування для забезпечення читабельності та підтримки програми у майбутньому.
- Тестування: програма була піддана тестуванню для перевірки правильності її роботи та виявлення можливих помилок.

Огляд

```
using System;
using System.IO;
using Microsoft.Data.Sqlite;

namespace SessionResultsUploader
{
    class Program
    {
        static void Main(string[] args)
        {
            string dbConnectionString = "Data Source=session_results.db";

            try
            {
                using (SqliteConnection connection = new
SqliteConnection(dbConnectionString))
                {
                    connection.Open();
                    Console.WriteLine("З'єднання з базою даних SQLite встановлено
успішно");

                    // Створення таблиць, якщо вони ще не існують
                    CreateTables(connection);

                    // Логіка вибору дій користувача та виконання відповідних операцій
                    while (true)
                    {
                        Console.WriteLine("Оберіть дію:");
                        Console.WriteLine("1. Створити студента");
                        Console.WriteLine("2. Створити предмет");
                        Console.WriteLine("3. Ввести результат екзамену");
                        Console.WriteLine("4. Переглянути всі дані");
                        Console.WriteLine("5. Імпортувати дані з CSV файлу");
```

```
                        Console.WriteLine("0. Вийти");

                        int choice = Convert.ToInt32(Console.ReadLine());

                        switch (choice)
                        {
                            case 1:
                                CreateStudent(connection);
                                break;
                            case 2:
                                CreateSubject(connection);
                                break;
                            case 3:
                                EnterExamResult(connection);
                                break;
                            case 4:
                                ViewAllData(connection);
                                break;
                            case 5:
                                ImportDataFromCSV(connection);
                                break;
                            case 0:
                                return;
                            default:
                                Console.WriteLine("Невірний вибір. Будь ласка,
спробуйте ще раз.");
                                break;
                        }
                    }
                }
            }
            catch (Exception ex)
            {
                Console.WriteLine("Помилка: " + ex.Message);
            }
        }

        static void CreateTables(SqliteConnection connection)
        {
            string createStudentsTable = @"
                CREATE TABLE IF NOT EXISTS Students (
                  student_id INTEGER PRIMARY KEY AUTOINCREMENT,
                  full_name VARCHAR(255) UNIQUE NOT NULL
                )";

            string createSubjectsTable = @"
                CREATE TABLE IF NOT EXISTS Subjects (
                  subject_id INTEGER PRIMARY KEY AUTOINCREMENT,
                  name VARCHAR(255) NOT NULL,
                  exam_date DATE NOT NULL,
                  CHECK (exam_date > '2000-01-01')
```

```
                )";

            string createExamsTable = @"
                CREATE TABLE IF NOT EXISTS Exams (
                  exam_id INTEGER PRIMARY KEY AUTOINCREMENT,
                  student_id INTEGER NOT NULL,
                  subject_id INTEGER NOT NULL,
                  score INTEGER CHECK (score >= 0 AND score <= 100),
                  FOREIGN KEY (student_id) REFERENCES Students(student_id),
                  FOREIGN KEY (subject_id) REFERENCES Subjects(subject_id)
                )";

            ExecuteNonQuery(connection, createStudentsTable);
            ExecuteNonQuery(connection, createSubjectsTable);
            ExecuteNonQuery(connection, createExamsTable);

            Console.WriteLine("Таблиці успішно створено або вже існують.");
        }

        static void ExecuteNonQuery(SqliteConnection connection, string query)
        {
            using (SqliteCommand command = new SqliteCommand(query, connection))
            {
                command.ExecuteNonQuery();
            }
        }

        static void CreateStudent(SqliteConnection connection)
        {
            Console.WriteLine("Введіть повне ім'я студента: ");
            string fullName = Console.ReadLine();

            string query = "INSERT INTO Students (full_name) VALUES (@fullName)";

            using (SqliteCommand command = new SqliteCommand(query, connection))
            {
                command.Parameters.AddWithValue("@fullName", fullName);

                int rowsAffected = command.ExecuteNonQuery();
                Console.WriteLine($"{rowsAffected} рядків додано.");
            }
        }

        static void CreateSubject(SqliteConnection connection)
        {
            Console.WriteLine("Введіть назву предмету: ");
            string name = Console.ReadLine();

            Console.WriteLine("Введіть дату екзамену (у форматі 'рік-місяць-день',
наприклад, '2024-04-11'): ");
            string examDateStr = Console.ReadLine();
```

```
            DateTime examDate;
            while (!DateTime.TryParse(examDateStr, out examDate))
            {
                Console.WriteLine("Неправильний формат дати. Введіть ще раз:");
                examDateStr = Console.ReadLine();
            }

            string query = "INSERT INTO Subjects (name, exam_date) VALUES (@name,
@examDate)";

            using (SqliteCommand command = new SqliteCommand(query, connection))
            {
                command.Parameters.AddWithValue("@name", name);
                command.Parameters.AddWithValue("@examDate", examDate);

                int rowsAffected = command.ExecuteNonQuery();
                Console.WriteLine($"{rowsAffected} рядків додано.");
            }
        }

        static void EnterExamResult(SqliteConnection connection)
        {
            Console.WriteLine("Введіть ID студента: ");
            int studentId = Convert.ToInt32(Console.ReadLine());

            Console.WriteLine("Введіть ID предмету: ");
            int subjectId = Convert.ToInt32(Console.ReadLine());

            Console.WriteLine("Введіть оцінку за екзамен: ");
            int score = Convert.ToInt32(Console.ReadLine());

            string query = "INSERT INTO Exams (student_id, subject_id, score) VALUES
(@studentId, @subjectId, @score)";

            using (SqliteCommand command = new SqliteCommand(query, connection))
            {
                command.Parameters.AddWithValue("@studentId", studentId);
                command.Parameters.AddWithValue("@subjectId", subjectId);
                command.Parameters.AddWithValue("@score", score);

                int rowsAffected = command.ExecuteNonQuery();
                Console.WriteLine($"{rowsAffected} рядків додано.");
            }
        }

        static void ViewAllData(SqliteConnection connection)
        {
            string query = @"
                SELECT s.full_name, subj.name, subj.exam_date, ex.score
                FROM Exams ex
                JOIN Students s ON ex.student_id = s.student_id
```

```
                JOIN Subjects subj ON ex.subject_id = subj.subject_id";

            using (SqliteCommand command = new SqliteCommand(query, connection))
            {
                using (SqliteDataReader reader = command.ExecuteReader())
                {
                    Console.WriteLine("Повні дані про екзамени:");

                    while (reader.Read())
                    {
                        Console.WriteLine($"Студент: {reader.GetString(0)}, Предмет:
{reader.GetString(1)}, Дата екзамену: {reader.GetDateTime(2)}, Оцінка:
{reader.GetInt32(3)}");
                    }
                }
            }
        }

    static void ImportDataFromCSV(SqliteConnection connection)
    {
        Console.WriteLine("Введіть шлях до CSV файлу:");
        string filePath = Console.ReadLine();

        try
        {
            using (StreamReader reader = new StreamReader(filePath))
            {
                while (!reader.EndOfStream)
                {
                    string line = reader.ReadLine();
                    string[] values = line.Split(',');

                    if (values.Length == 4)
                    {
                        string fullName = values[0];
                        string subjectName = values[1];
                        DateTime examDate = DateTime.Parse(values[2]);
                        int score = int.Parse(values[3]);

                        // Створення студента, якщо він ще не існує
                        int studentId = GetOrCreateStudent(connection, fullName);

                        // Створення предмету, якщо він ще не існує
                        int subjectId = GetOrCreateSubject(connection, subjectName,
examDate);

                        // Вставка результатів екзамену
                        string query = "INSERT INTO Exams (student_id, subject_id,
score) VALUES (@studentId, @subjectId, @score)";

                        using (SqliteCommand command = new SqliteCommand(query,
```

```
connection))
                        {
                            command.Parameters.AddWithValue("@studentId", studentId);
                            command.Parameters.AddWithValue("@subjectId", subjectId);
                            command.Parameters.AddWithValue("@score", score);

                            command.ExecuteNonQuery();
                        }
                    }
                }

                Console.WriteLine("Дані імпортовано успішно.");
            }
        }
        catch (Exception ex)
        {
            Console.WriteLine("Помилка під час імпорту: " + ex.Message);
        }
    }

    static int GetOrCreateStudent(SqliteConnection connection, string fullName)
    {
        // Перевірка чи студент вже існує
        string query = "SELECT student_id FROM Students WHERE full_name = @fullName";

        using (SqliteCommand command = new SqliteCommand(query, connection))
        {
            command.Parameters.AddWithValue("@fullName", fullName);

            object result = command.ExecuteScalar();
            if (result != null)
            {
                return Convert.ToInt32(result);
            }
        }

        // Якщо студент не існує, створити нового студента
        string insertQuery = "INSERT INTO Students (full_name) VALUES (@fullName);
SELECT last_insert_rowid();";

        using (SqliteCommand insertCommand = new SqliteCommand(insertQuery,
connection))
        {
            insertCommand.Parameters.AddWithValue("@fullName", fullName);

            return Convert.ToInt32(insertCommand.ExecuteScalar());
        }
    }

    static int GetOrCreateSubject(SqliteConnection connection, string subjectName,
DateTime examDate)
```

```
    {
        // Перевірка чи предмет вже існує
        string query = "SELECT subject_id FROM Subjects WHERE name = @subjectName AND
exam_date = @examDate";

        using (SqliteCommand command = new SqliteCommand(query, connection))
        {
            command.Parameters.AddWithValue("@subjectName", subjectName);
            command.Parameters.AddWithValue("@examDate", examDate);

            object result = command.ExecuteScalar();
            if (result != null)
            {
                return Convert.ToInt32(result);
            }
        }

        // Якщо предмет не існує, створити новий предмет
        string insertQuery = "INSERT INTO Subjects (name, exam_date) VALUES
(@subjectName, @examDate); SELECT last_insert_rowid();";

        using (SqliteCommand insertCommand = new SqliteCommand(insertQuery,
connection))
        {
            insertCommand.Parameters.AddWithValue("@subjectName", subjectName);
            insertCommand.Parameters.AddWithValue("@examDate", examDate);

            return Convert.ToInt32(insertCommand.ExecuteScalar());
        }
    }
  }
}
```

```
MSBuild version 17.9.8+b34f75857 for .NET
  Determining projects to restore...
  All projects are up-to-date for restore.
  4.2 -> /Users/dmytro.soltusyuk/Work/university/Бази_даних/additional_2/bin/Debug/net8.0/4.2.dll

Build succeeded.
    0 Warning(s)
    0 Error(s)

Time Elapsed 00:00:00.47
З'єднання з базою даних SQLite встановлено успішно
Таблиці успішно створено або вже існують.
Оберіть дію:
1. Створити студента
2. Створити предмет
3. Ввести результат екзамену
4. Переглянути всі дані
5. Імпортувати дані з CSV файлу
0. Вийти
1
Введіть повне ім'я студента:
Дмитро Солтисюк
1 рядків додано.
Оберіть дію:
1. Створити студента
2. Створити предмет
3. Ввести результат екзамену
4. Переглянути всі дані
5. Імпортувати дані з CSV файлу
0. Вийти
2
Введіть назву предмету:
Програмування
Введіть дату екзамену (у форматі 'рік-місяць-день', наприклад, '2024-04-11'):
2024-04-11
1 рядків додано.
Оберіть дію:
1. Створити студента
2. Створити предмет
3. Ввести результат екзамену
4. Переглянути всі дані
5. Імпортувати дані з CSV файлу
0. Вийти
3
Введіть ID студента:
1
Введіть ID предмету:
1
Введіть оцінку за екзамен:
95
1 рядків додано.
Оберіть дію:
1. Створити студента
2. Створити предмет
3. Ввести результат екзамену
4. Переглянути всі дані
5. Імпортувати дані з CSV файлу
0. Вийти
4
Повні дані про екзамени:
Студент: Дмитро Солтисюк, Предмет: Програмування, Дата екзамену: 4/11/2024 12:00:00 AM, Оцінка: 95
Оберіть дію:
1. Створити студента
2. Створити предмет
3. Ввести результат екзамену
4. Переглянути всі дані
5. Імпортувати дані з CSV файлу
0. Вийти
5
Введіть шлях до CSV файлу:
учфple1.csv
Дані імпортовано успішно.
Оберіть дію:
1. Створити студента
2. Створити предмет
3. Ввести результат екзамену
4. Переглянути всі дані
5. Імпортувати дані з CSV файлу
0. Вийти
4
Повні дані про екзамени:
Студент: Дмитро Солтисюк, Предмет: Програмування, Дата екзамену: 4/11/2024 12:00:00 AM, Оцінка: 95
Студент: Тимко Б.А., Предмет: Математика, Дата екзамену: 4/10/2024 12:00:00 AM, Оцінка: 85
Студент: Ющенко М.О., Предмет: Фізика, Дата екзамену: 4/11/2024 12:00:00 AM, Оцінка: 78
Студент: Ковальчук О.В., Предмет: Хімія, Дата екзамену: 4/12/2024 12:00:00 AM, Оцінка: 92
Оберіть дію:
1. Створити студента
2. Створити предмет
3. Ввести результат екзамену
4. Переглянути всі дані
5. Імпортувати дані з CSV файлу
0. Вийти
0
```

Висновок

Програма "SessionResultsUploader" є потужним інструментом для керування даними про результати сесії. Вона надає користувачам зручний спосіб створення та управління інформацією про студентів, предмети та результати екзаменів. Завдяки своїй простоті та функціональності, програма може бути використана в навчальних закладах та інших сферах, де потрібно ефективно керувати даними про успішність студентів.