Лабораторна робота №1, етап 2

з дисципліни

«Інформаційні технології»

Виконав:

Гончар Михайло, група ТК-41

Київський національний університет імені Тараса Шевченка

Факультет комп'ютерних наук і кібернетики

Дата виконання:

03 жовтня 2025 р.

Мета роботи

Ознайомитися з принципами побудови систем управління табличними базами даних, створити частково реалізовану програмну систему для роботи з базами даних, таблицями та полями, реалізувати функції додавання, редагування, видалення, збереження та порівняння таблиць.

Завдання варіанту

- Підтримати типи даних: ціле, дійсне, символ, рядок, комплексне ціле, комплексне дійсне.
- Реалізувати функцію **різниця таблиць** — порівняння значень між двома таблицями за вибраними полями.

Хід виконання роботи

1. Розробка структури бази даних

Було створено базу даних із такими основними сутностями:

- **Dbase** — база даних;
- **Table** — таблиця;
- **Field** — поле таблиці;
- **Row** — рядок;
- **Value** — значення;
- **Type** — тип даних поля.

Всі зв'язки між таблицями реалізовано через ORM Entity Framework.

2. Інтерфейс користувача

Графічний інтерфейс створено на базі **Windows Forms**. Користувач має змогу:

- створювати нову базу даних;
- створювати таблиці в межах бази;
- додавати поля;
- додавати рядки та редагувати значення;
- видаляти бази, таблиці, поля, рядки;
- зберігати всі бази даних у JSON-файл;

- завантажувати дані з JSON-файлу;
- порівнювати дві таблиці (функція «Різниця таблиць»).

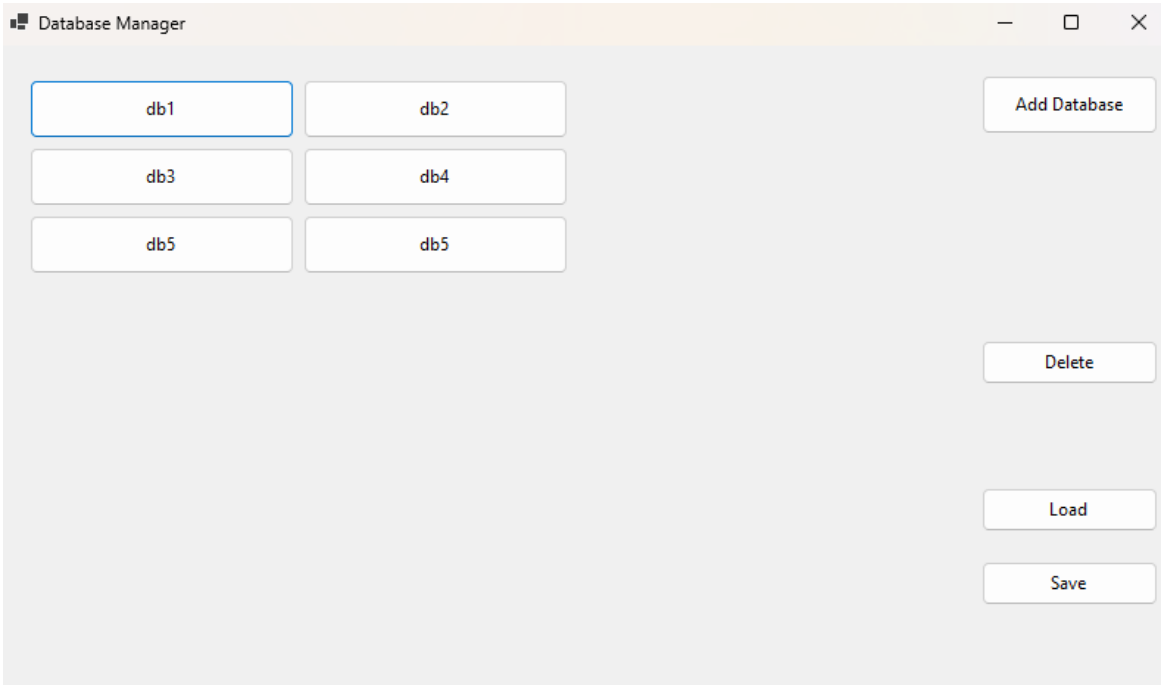

Рис. 1: Головне вікно програми з переліком баз даних

3. Реалізація функціональності

3.1. Створення та видалення об'єктів

У коді реалізовано керування станами:

- 0 — робота з базами;
- 1 — робота з таблицями;
- 2 — робота з полями та рядками.

3.2. Валідація введених значень

Для перевірки введених користувачем значень реалізовано метод:

```
public bool ValidateValue(string val, int typeId)
```

Перевірка комплексних типів:

```
case 5: return Regex.IsMatch(val, @"^[+-]?\d+([+-]\d+)i$"); // комплексне ціле
case 6: return Regex.IsMatch(val, @"^[+-]?(\d+(\.\d+)?)([+-](\d+(\.\d+)?))i$");
```

3.3. Збереження та завантаження баз даних

Кнопки **Save** і **Load** реалізують експорт та імпорт у форматі JSON:

```
File.WriteAllText("backup.json", json);
File.ReadAllText("backup.json");
```

3.4. Порівняння таблиць (різниця таблиць)

Функція викликається кнопкою **“Diff”**. Відкривається нове вікно, де користувач обирає поле з поточної таблиці, іншу таблицю та поле для порівняння. Програма відображає обидві таблиці поруч і виконує аналіз значень.

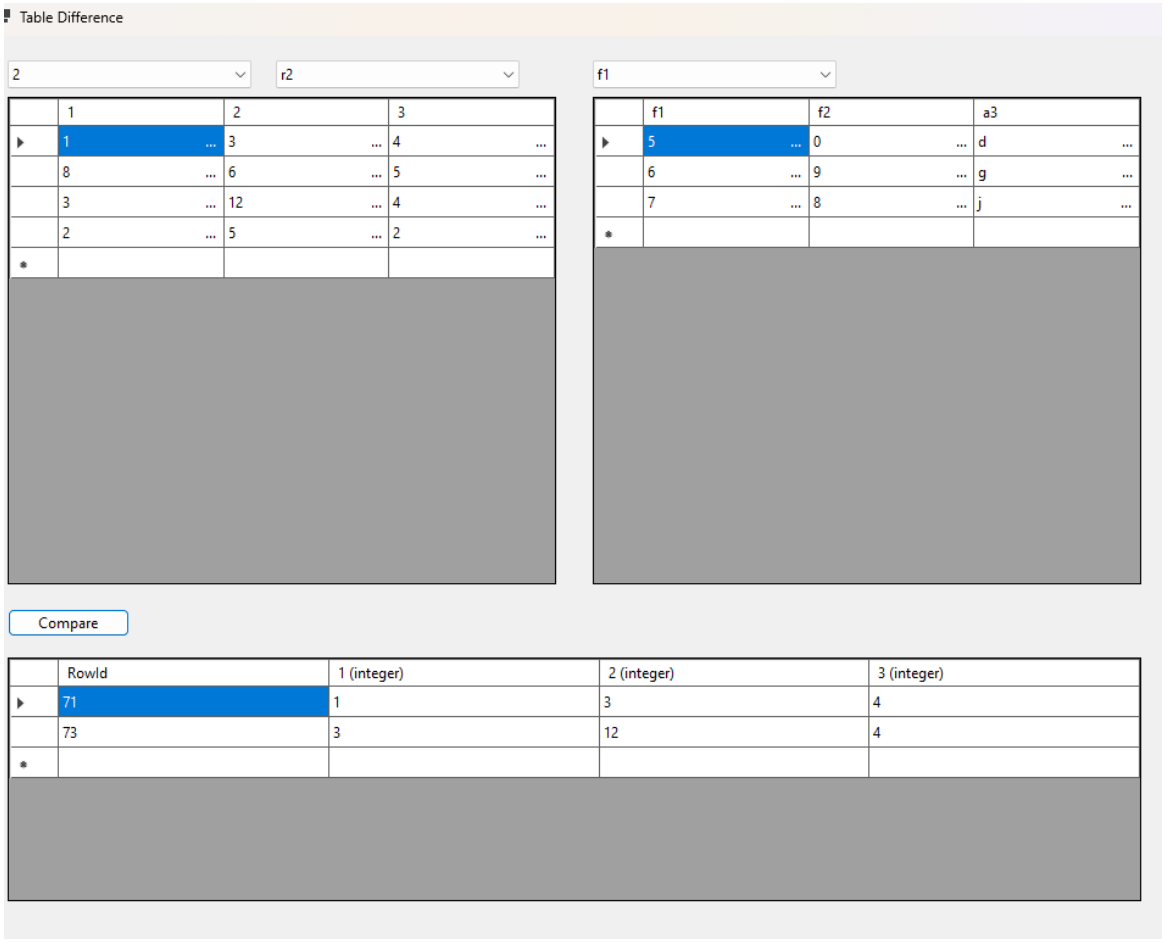

Рис. 2: Вікно “Table Difference” з двома таблицями для порівняння

4. Особливості реалізації

- ORM: **Entity Framework Core**;
- Формат резервного копіювання: **JSON (Newtonsoft.Json)**;
- Інтерфейс: **Windows Forms (C#)**;
- Повна підтримка CRUD-операцій.

5. Тестування програми (Unit-тести)

Для перевірки коректності роботи функцій було створено набір модульних тестів із використанням бібліотеки **xUnit**. Тестування охоплює:

- перевірку валідності введених значень;
- додавання нових рядків і створення пов'язаних значень;
- пошук відмінностей між таблицями.

5.1. Тести валідації

Мета тесту — переконатися, що метод `ValidateValue` коректно обробляє різні типи даних.

```
        [Theory]
        [InlineData("123", 1, true)]   // int
        [InlineData("abc", 1, false)]  // int
        [InlineData("12.5", 2, true)]  // double
        [InlineData("x", 3, true)]     // char
        [InlineData("xyz", 3, false)]  // char
        public void TestValidateValue(string val, int typeId, bool
           expected)
        {
            var form = new skbd.Form1();
            var result = form.ValidateValue(val, typeId);
            Assert.Equal(expected, result);
        }
```

Лістинг 1: ValidationTests

5.2. Тести для сервісу рядків

```
        [Fact]
        public void AddNewRow_ShouldCreateRowAndValues()
        {
            using var context = GetInMemoryContext();

            var table = new Table { Name = "TestTable" };
            context.Tables.Add(table);
            context.SaveChanges();

            context.Fields.Add(new Field { TableId = table.Id, Name = "
               Field1" });
            context.Fields.Add(new Field { TableId = table.Id, Name = "
               Field2" });
```

```
            context.SaveChanges();

            var service = new RowService(context);
            var row = service.AddNewRow(table.Id);

            Assert.NotNull(row);
            Assert.True(row.Id > 0);

            var values = context.Values.Where(v => v.RowId == row.Id).
                ToList();
            Assert.Equal(2, values.Count);
            Assert.All(values, v => Assert.Equal("", v.Val));
        }
```

Лістинг 2: RowServiceTests

5.3. Тести для порівняння таблиць

```
        [Fact]
        public void GetDifferentRows_ShouldReturnRowsWithoutMatches()
        {
            using var context = GetInMemoryContext();
            var service = new TableDiffService(context);

            var diff = service.GetDifferentRows(1, "Col1", 2, "ColX");

            Assert.Single(diff);
            Assert.Equal(2, diff[0].Id);
        }
```

Лістинг 3: TableDiffServiceTests

5.4. Результати тестів

Усі тести були виконані успішно. Це підтверджує, що всі основні модулі системи працюють коректно.

```
Test summary: total: 7, failed: 0, succeeded: 7, skipped: 0, duration: 2.8s
```

Рис. 3: Результати проходження unit-тестів у xUnit

Результати роботи

- Створено функціональну систему управління табличними базами даних;
- Реалізовано підтримку комплексних типів;

- Реалізовано операцію різниці таблиць;
- Додано збереження/відновлення у форматі JSON;
- Забезпечено зручний графічний інтерфейс.

Висновки

У ході виконання лабораторної роботи:

- засвоєно принципи побудови систем управління табличними базами даних;
- реалізовано програмну систему з використанням ORM Entity Framework;
- додано підтримку комплексних чисел;
- створено інструмент для порівняння таблиць;
- отримано практичні навички роботи з C#, WinForms, EF, JSON.