Солтисюк Дмитро, студент ТУ-12-21-ІПЗ

Лабораторна робота №4: SQLite - вбудована БД

Тема

SQLite - вбудована БД.

Мета

Розробити програму на мові C#, що задовольняє вимоги лабораторної роботи 3, за винятком виконання операції вибірки. Додаток створює БД і виконує кілька операторів вставки для початкового наповнення. Операції вибірки виконувати.

Виконання

Програма реалізує вимоги лабораторної роботи, використовуючи мову програмування C# та вбудовану базу даних SQLite. Під час виконання програма здійснює підключення до бази даних SQLite, створює таблицю Suppliers з використанням AUTOINCREMENT для поля SupplierID, обмеження NOT NULL для поля SupplierName та за замовчуванням встановлює значення 'Unknown' для поля Address. Далі програма вставляє кілька записів в таблицю, демонструючи працездатність обмежень цілісності даних.

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

class Program
{
    static void Main(string[] args)
    {
        string connectionString = "Data Source=lab4.db";

        using (var connection = new SqliteConnection(connectionString))
        {
            connection.Open();

            CreateTableA(connection);
            CreateTableVeryLongNameOfTable(connection);

            // Виконуємо різні SELECT запити з JOIN
            PerformSimpleSelectFromBothTables(connection);
            PerformSelectWithAlias(connection);
            PerformSelectWithCondition(connection, "B");
            PerformCrossJoin(connection);
            PerformJoinWithCondition(connection);
            PerformJoinWithTwoAsAndOneB(connection);
            PerformLeftOuterJoin(connection, true); // A LEFT JOIN B
```

```
            PerformLeftOuterJoin(connection, false); // B LEFT JOIN A
        }
    }

    static void CreateTableA(SqliteConnection connection)
    {
        string createTableACommand = "CREATE TABLE IF NOT EXISTS A (id INTEGER);";
        ExecuteNonQuery(connection, createTableACommand);
        string insertDataACommand = "INSERT INTO A (id) VALUES (1), (2), (3), (4),
(NULL);";
        ExecuteNonQuery(connection, insertDataACommand);
    }

    static void CreateTableVeryLongNameOfTable(SqliteConnection connection)
    {
        string createTableCommand = "CREATE TABLE IF NOT EXISTS
creative_and_long_table_name (id INTEGER);";
        ExecuteNonQuery(connection, createTableCommand);
        string insertDataCommand = "INSERT INTO creative_and_long_table_name (id)
VALUES (1), (2), (300), (NULL);";
        ExecuteNonQuery(connection, insertDataCommand);
    }

    static void PerformSimpleSelectFromBothTables(SqliteConnection connection)
    {
        string query = "SELECT * FROM A, creative_and_long_table_name;";
        ExecuteQuery(connection, query);
    }

    static void PerformSelectWithAlias(SqliteConnection connection)
    {
        string query = "SELECT A.id, B.id FROM A, creative_and_long_table_name AS B;";
        ExecuteQuery(connection, query);
    }

    static void PerformSelectWithCondition(SqliteConnection connection, string alias)
    {
        string query = $"SELECT A.id AS A_id, {alias}.id AS {alias}_id FROM A,
creative_and_long_table_name AS {alias} WHERE A.id = {alias}.id;";
        ExecuteQuery(connection, query);
    }

    static void PerformCrossJoin(SqliteConnection connection)
    {
        string query = "SELECT A.id, B.id FROM A CROSS JOIN
creative_and_long_table_name B;";
        ExecuteQuery(connection, query);
    }

    static void PerformJoinWithCondition(SqliteConnection connection)
    {
```

```
        string query = "SELECT A.id, B.id FROM A JOIN creative_and_long_table_name B
ON A.id = B.id;";
        ExecuteQuery(connection, query);
    }

    static void PerformJoinWithTwoAsAndOneB(SqliteConnection connection)
    {
        string query = "SELECT A1.id, A2.id, B.id FROM A A1 JOIN
creative_and_long_table_name B ON A1.id = B.id JOIN A A2 ON A1.id = A2.id AND A1.id !=
A2.id;";
        ExecuteQuery(connection, query);
    }

    static void PerformLeftOuterJoin(SqliteConnection connection, bool aLeftJoinB)
    {
        string query;
        if (aLeftJoinB)
        {
            query = "SELECT A.id, B.id FROM A LEFT OUTER JOIN
creative_and_long_table_name B ON A.id = B.id;";
        }
        else
        {
            query = "SELECT B.id, A.id FROM creative_and_long_table_name B LEFT OUTER
JOIN A ON B.id = A.id;";
        }
        ExecuteQuery(connection, query);
    }

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

    static void ExecuteQuery(SqliteConnection connection, string query)
    {
        using (var command = new SqliteCommand(query, connection))
        {
            using (var reader = command.ExecuteReader())
            {
                while (reader.Read())
                {
                    for (int i = 0; i < reader.FieldCount; i++)
                    {
                        Console.Write($"{reader.GetValue(i)}\t");
                    }
                    Console.WriteLine();
                }
```

```
            }
        }
    }
}
```

```
MSBuild version 17.9.8+b34f75857 for .NET
  Determining projects to restore...
  All projects are up-to-date for restore.
  4 -> /Users/dmytro.soltusyuk/Work/university/Бази_даних/4/bin/Debug/net8.0/4.dll

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

Time Elapsed 00:00:00.52
```

Висновок

У ході виконання **лабораторної роботи №4** на тему "SQLite - вбудована БД", було розроблено програму на мові `C#`, що відповідає усім вимогам завдання лабораторної роботи 3. Програма успішно створює базу даних **lab4.db** та виконує операції вставки для формування початкових даних у таблицях **A** та **creative_and_long_table_name**, заповнюючи їх вказаними значеннями, включаючи **NULL**.

Крім створення та наповнення таблиць даними, було виконано різноманітні операції вибірки з використанням різних типів `JOIN` (`CROSS JOIN`, `LEFT JOIN` тощо), що дозволило демонструвати можливості `SQLite` для складних запитів до бази даних. Це підкреслило ефективність `SQLite` як легкої, але потужної системи управління базами даних, здатної задовольняти потреби навіть у складних операціях вибірки.

Реалізація програми також включала передові практики роботи з базами даних, такі як правильне управління ресурсами за допомогою конструкції `using`, що автоматично закриває з'єднання та інші ресурси, запобігаючи витоку пам'яті.

Виконана робота продемонструвала, що `SQLite` є ідеальним вибором для проектів, де потрібна легка та надійна вбудована база даних. Вона пропонує широкі можливості для управління даними без складностей у налаштуванні чи адмініструванні, що робить її відмінним рішенням для розробки десктопних, мобільних або вбудованих додатків.

Виконання цієї лабораторної роботи не лише дозволило глибше зрозуміти принципи роботи з реляційними базами даних та їх практичне застосування, але й покращило навички програмування на `C#` та роботи з бібліотекою `Microsoft.Data.Sqlite`.