ЛАБОРАТОРНА РОБОТА 9

Тема: Створення " *.sdf" файлу локальної бази даних Microsoft SQL Server в системі Microsoft Visual Studio 2019

Мета роботи: набути практичних навиків створення бази даних за допомогою SQL Server Compact 4.0

Методичне, програмне забезпечення і ТЗН: Інструкція до лабораторної роботи №1, персональний комп'ютер, ОС Windows 7/8.1/10, Microsoft Office 2010-2019, Adobe Reader, Visual Studio 2019.

Короткі теоретичні відомості

На сьогоднішній день відомо більше двох десятків серверних СУБД, з яких найбільш популярними є Oracle, Microsoft SQL Server, Informix, DB2, Sybase, InterBase, MySQL. Для виконання лабораторної робооти буде використовуватися сервер "Microsoft SQL Server Compact 4.0". Microsoft® SQL Server ™ - це система аналізу та управління реляційними базами даних в рішеннях електронної комерції, виробничих галузей і сховищ даних. Microsoft SQL Server - система керування базами даних (СКБД), розроблена корпорацією Microsoft.

Основна використовувана мова запитів - Transact-SQL, створений спільно Microsoft та Sybase. Transact-SQL є реалізацією стандарту 40 ANSI / ISO по структурованого мови запитів (SQL) з розширеннями. використовується для роботи з базами даних розміром від персональних до великих баз даних масштабу підприємства; конкурує з іншими СУБД в цьому сегменті ринку.

Structured Query Language (SQL) – це непроцедурна мова, яка використовується для управління даними реляційних СУБД. Т

Термін "непроцедурна" означає, що мовою можна сформулювати те, що потрібно зробити з даними, але не можна проінструктувати, як його слід виконати. У стандарті мови SQL відсутні алгоритмічні конструкції такі, як мітки, оператори циклу, умовні переходи тощо. Хоча в діалектах мови, що створені різними розробниками (Microsoft, Oracle, IBM) вони є.

Мова SQL має п'ять основних типів команд, кожну з яких можна розглядати як окрему мову. До них входять такі:

DDL (Data Definition Language) – мова визначення даних. Призначена для створення, змінення та вилучення об'єктів;

DML (Data Manipulation Language) – мова маніпуляцій даними. Містить оператори, що дозволяють вибирати, додавати, видаляти та модифікувати дані;

DCL (Data Control Language) – мова управління даними. Застосовується для реалізації адміністративних функцій, які надають або скасовують право (привілей) використовувати базу даних, таблиці в базі даних, а також виконувати ті чи інші оператори SQL;

TCL (Transaction Control Language) – мова управління транзакціями. Дозволяє організовувати роботу транзакцій, що здійснюють зміни у базі даних за допомогою груп операторів DML;

CCL (Cursor Control Language) – мова управління курсором. Містить оператори визначення курсора, підготовки SQL-речень для виконання, а також для деяких інших операторів.

Для створення нової бази даних засобами мови SQL слід використовувати таку команду:

```
CREATE DATABASE Ім'я_бази_даних [Параметри];
```

Для видалення бази даних застосовується команда DROP DATABASE, яка має наступний синтаксис:

```
DROP DATABASE Ім'я_бази_даних
```

Для створення таблиць застосовується команда CREATE TABLE. З цією командою можна використовувати ряд операторів, які визначають стовпці таблиці і їх атрибути. І крім того, можна використовувати ряд операторів, які визначають властивості таблиці в цілому. Одна база даних може містити до 2 мільярдів таблиць.

Загальний синтаксис створення таблиці виглядає наступним чином:

```
CREATE TABLE назва_таблиці
(назва_стовбця1 тип_даних атрибути_стовбця1,
 назва_стовбця2 тип_даних атрибути_стовбця2,
 ..............................................
 назва_стовбцяN тип_даних атрибути_стовбцяN,
 атрибути_таблиці
)
```

Для видалення таблиць використовується команда DROP TABLE, яка має наступний синтаксис:

```
DROP TABLE table1 [, table2, ...]
```

При створенні таблиці для всіх її стовпців необхідно вказати певний тип даних. Тип даних визначає, які значення можуть зберігатися в стовпці, скільки вони будуть

займати місця в пам'яті. Мова T-SQL надає безліч різних типів. Залежно від характеру значень все їх можна розділити на групи.

Числові типи даних

– **BIT**: зберігає значення 0 або 1. Фактично є аналогом булевого типу в мовах програмування. Займає 1 байт.

– **TINYINT**: зберігає числа від 0 до 255. Займає 1 байт. Добре підходить для зберігання невеликих чисел.

– **SMALLINT**: зберігає числа від -32 768 до 32 767. Займає 2 байта

– **INT**: зберігає числа від -2 147 483 648 до 2 147 483 647. Займає 4 байта. Найбільш використовуваний тип для зберігання чисел.

– **BIGINT**: зберігає дуже великі числа від -9 223 372 036 854 775 808 до 9 223 372 036 854 775 807, які займають в пам'яті 8 байт.

– **DECIMAL**: зберігає числа с фіксованою точністю. Займає від 5 до 17 байт в залежності від кількості чисел після коми. Даний тип може приймати два параметри precision і scale: DECIMAL (precision, scale).

Параметр precision представляє максимальну кількість цифр, які може зберігати число. Це значення повинно знаходитися в діапазоні від 1 до 38. За замовчуванням воно дорівнює 18.

Параметр scale представляє максимальну кількість цифр, які може містити число після коми. Це значення повинно знаходитися в діапазоні від 0 до значення параметра precision. За замовчуванням воно дорівнює 0.

– **NUMERIC**: даний тип аналогічний типу DECIMAL.

– **SMALLMONEY**: зберігає дробові значення від -214 748.3648 до 214 748.3647. Призначено для зберігання грошових величин. Займає 4 байти. Еквівалентний типу DECIMAL (10,4).

– **MONEY**: зберігає дробові значення від -922 337 203 685 477.5808 до 922 337 203 685 477.5807. Являє грошові величини і займає 8 байт. Еквівалентний типу DECIMAL (19,4).

– **FLOAT**: зберігає числа від -1.79E + 308 до 1.79E + 308. Займає від 4 до 8 байт в залежності від дробової частини. Може мати форму опредеенія у вигляді FLOAT (n), де n представляє число біт, які використовуються для зберігання десяткової частини числа (мантиси). За замовчуванням n = 53.

– **REAL**: зберігає числа від -340E + 38 to 3.40E + 38. Займає 4 байти. Еквівалентний типу FLOAT (24).

Типи даних, що представляють дату і час

– **DATE**: зберігає дати від 0001-01-01 (1 січня 0001 року) до 9999-12-31 (31 грудня 9999 року). Займає 3 байта.

– **TIME**: зберігає час в діапазоні від 00: 00: 00.0000000 до 23: 59: 59.9999999. Займає від 3 до 5 байт. Може мати форму TIME (n), де n представляє кількість цифр від 0 до 7 в дробової частини секунд.

– **DATETIME**: зберігає дати і час від 01/01/1753 до 31/12/9999. Займає 8 байт.

– **DATETIME2**: зберігає дати і час в діапазоні від 01/01/0001 00: 00: 00.0000000 до 31/12/9999 23: 59: 59.9999999. Займає від 6 до 8 байт в залежності від точності часу. Може мати форму DATETIME2 (n), де n представляє кількість цифр від 0 до 7 в дробової частини секунд.

– **SMALLDATETIME**: зберігає дати і час в діапазоні від 01/01/1900 до 06/06/2079, тобто найближчі дати. Займає від 4 байта.

– **DATETIMEOFFSET**: зберігає дати і час в діапазоні від 0001-01-01 до 9999-12-31. Зберігає детальну інформацію про час з точністю до 100 наносекунд. Займає 10 байт.

Рядкові типи даних

– **CHAR**: зберігає рядок довжиною від 1 до 8 000 символів. На кожен символ виділяє по 1 байту. Не підходить для багатьох мов, так як зберігає символи не в кодуванні Unicode. Кількість символів, яке може зберігати стовпець, передається в дужках. Наприклад, для стовпця з типом CHAR (10) буде виділено 10 байт. І якщо ми збережемо в стовпці рядок менше 10 символів, то вона буде доповнена пробілами.

– **VARCHAR**: зберігає рядок. На кожен символ виділяється 1 байт. Можна вказати конкретну довжину для стовпця - від 1 до 8 000 символів, наприклад, VARCHAR (10). Якщо рядок повинен мати більше 8000 символів, то задається розмір MAX, а на зберігання рядка може виділятися до 2 Гб: VARCHAR (MAX). Не підходить для багатьох мов, так як зберігає символи не в кодуванні Unicode. На відміну від типу CHAR якщо в стовпець з типом VARCHAR (10) буде збережена рядок в 5 символів, то в Столц буде збережено саме п'ять символів.

– **NCHAR**: зберігає рядок в кодуванні Unicode завдовжки від 1 до 4 000 символів. На кожен символ виділяється 2 байти. Наприклад, NCHAR (15) NVARCHAR: зберігає рядок в кодуванні Unicode. На кожен символ виділяється 2 байта.Можно задати конкретний розмір від 1 до 4 000 символів:. Якщо рядок повинен мати більше 4000 символів, то задається розмір MAX, а на зберігання рядка може виділятися до 2 Гб.

Ще два типу **TEXT** і **NTEXT** є застарілими і тому їх не рекомендується використовувати. Замість них застосовуються VARCHAR і NVARCHAR відповідно.

Бінарні типи даних

– **BINARY**: зберігає бінарні дані у вигляді послідовності від 1 до 8 000 байт.

– **VARBINARY**: зберігає бінарні дані у вигляді послідовності від 1 до 8 000 байт, або до 2 ^ 31-1 байт при використанні значення MAX (VARBINARY (MAX)).

Ще один бінарний тип - тип **IMAGE** є застарілим, і замість нього рекомендується застосовувати тип VARBINARY.

Решта типів даних

– **UNIQUEIDENTIFIER**: унікальний ідентифікатор GUID (по суті рядок з унікальним значенням), який займає 16 байт.

– **TIMESTAMP**: певна кількість, яке зберігає номер версії рядки в таблиці. Займає 8 байт. CURSOR: представляє набір рядків.

– **HIERARCHYID**: представляє позицію в ієрархії.

– **SQL_VARIANT**: може зберігати дані будь-якого іншого типу даних T-SQL.

– **XML**: зберігає документи XML або фрагменти документів XML. Займає в пам'яті до 2 Гб.

– **TABLE**: представляє визначення таблиці.

– **GEOGRAPHY**: зберігає географічні дані, такі як широта і довгота.

– **GEOMETRY**: зберігає координати місцезнаходження на площині.

При створенні стовпців в T-SQL ми можемо використовувати ряд атрибутів, ряд яких є обмеженнями. Розглянемо ці атрибути.

PRIMARY KEY За допомогою виразу PRIMARY KEY стовпець можна зробити первинним ключем. Первинний ключ унікально ідентифікує рядок в таблиці. В якості первинного ключа необов'язково повинні виступати стовпці з типом int, вони можуть представляти будь-який інший тип.

IDENTITY Атрибут IDENTITY дозволяє зробити стовпець ідентифікатором. Цей атрибут може призначатися для стовпців числових типів INT, SMALLINT, BIGINT, TYNIINT, DECIMAL і NUMERIC. При додаванні нових даних в таблицю SQL Server буде інкрементувати на одиницю значення цього стовпця у останнього запису. Як правило, в ролі ідентифікатора виступає той же стовпець, який є первинним ключем, хоча в принципі це необов'язково.

NULL і **NOT NULL** Щоб вказати, чи може стовпець приймати значення NULL, при визначенні стовпця йому можна задати атрибут NULL або NOT NULL. Якщо цей атрибут явно не буде використаний, то за замовчуванням стовпець буде допускати значення NULL. Винятком є той випадок, коли стовпчик виступає в ролі первинного ключа - в цьому випадку за замовчуванням стовпець має значення NOT NULL.

DEFAULT Атрибут DEFAULT визначає значення за замовчуванням для стовпця. Якщо при додаванні даних для стовпця нічого очікувати передбачено значення, то для нього буде використовуватися значення за замовчуванням.

Обмеження можуть носити довільні назви, але, як правило, для застосовуються такі префікси:

- "PK_" - для PRIMARY KEY
- "FK_" - для FOREIGN KEY
- "DF_" - для DEFAULT

В принципі необов'язково ставити імена обмежень, при установці відповідних атрибутів SQL Server автоматично визначає їх імена. Але, знаючи ім'я обмеження, ми можемо до нього звертатися, наприклад, для його видалення.

Зовнішні ключі застосовуються для установки зв'язку між таблицями. Зовнішній ключ встановлюється для стовпців з залежною, підлеглої таблиці, і вказує на один із стовпців з головної таблиці. Хоча, як правило, зовнішній ключ вказує на первинний ключ з пов'язаної головною таблиці, але це не обов'язково має бути неодмінною умовою. Зовнішній ключ також може вказувати на якийсь інший стовпець, який має унікальне значення.

Загальний синтаксис установки зовнішнього ключа на рівні стовпця:

```
[FOREIGN KEY] REFERENCES головна_таблиця (стовпець_головної_таблиці)
    [ON DELETE {CASCADE|NO ACTION}]
    [ON UPDATE {CASCADE|NO ACTION}]
```

ON DELETE і ON UPDATE

За допомогою виразів ON DELETE і ON UPDATE можна встановити дії, які виконуватимуться відповідно при видаленні і зміні пов'язаної рядки з головної таблиці. І для визначення дії ми можемо використовувати такі опції:

- **CASCADE**: автоматично видаляє або змінює рядки з залежною таблиці при видаленні або зміні пов'язаних рядків в головній таблиці.
- **NO ACTION**: запобігає будь-які дії в залежною таблиці при видаленні або зміні пов'язаних рядків в головній таблиці. Тобто фактично будь-які дії відсутні.
- **SET NULL**: при видаленні пов'язаної рядки з головної таблиці встановлює для стовпця зовнішнього ключа значення NULL.
- **SET DEFAULT**: при видаленні пов'язаної рядки з головної таблиці встановлює для стовпця зовнішнього ключа значення за замовчуванням, яке задається за допомогою атрибути DEFAULT. Якщо для стовпця не задано значення за замовчуванням, то в якості нього застосовується значення NULL.

Порядок виконання роботи

1. Завантажити Microsoft Visual Studio 2019.
2. Створіть новий проект Windows Forms додатки (платформа .NET Framework)
3. Додати стартову форму, що буде завантажуватись першою при запуску програми: у вікні Solution explorer викликати контекстне меню для назви проекту і вибрати пункт **Добавить – Создать елемент – Форма Windows Forms** і ввести ім’я **MyForm**.
4. Створити інтерфейс головної форми, змінивши властивості text та image згідно рисунку 1 та розмістивши на ній наступні компоненти: button 1 і button2 для завершення та початку роботи програми відповідно, menuStrip1 для створення головного меню, toolStrip1 для створення панелі інструментів та toolTip1 для створення випливаючих підказок при наведенні на кнопки button1 і button2 (рисунок 1).

Рисунок 1 – Вигляд стартової форми у режимі конструктора

5. Активувати вікно SQLite/Server Explorer Compact Toolbox (рисунок2). Для цього, у MS Visual Studio потрібно викликати команду View→Other Windows→SQLite/Server Explorer Compact Toolbox.

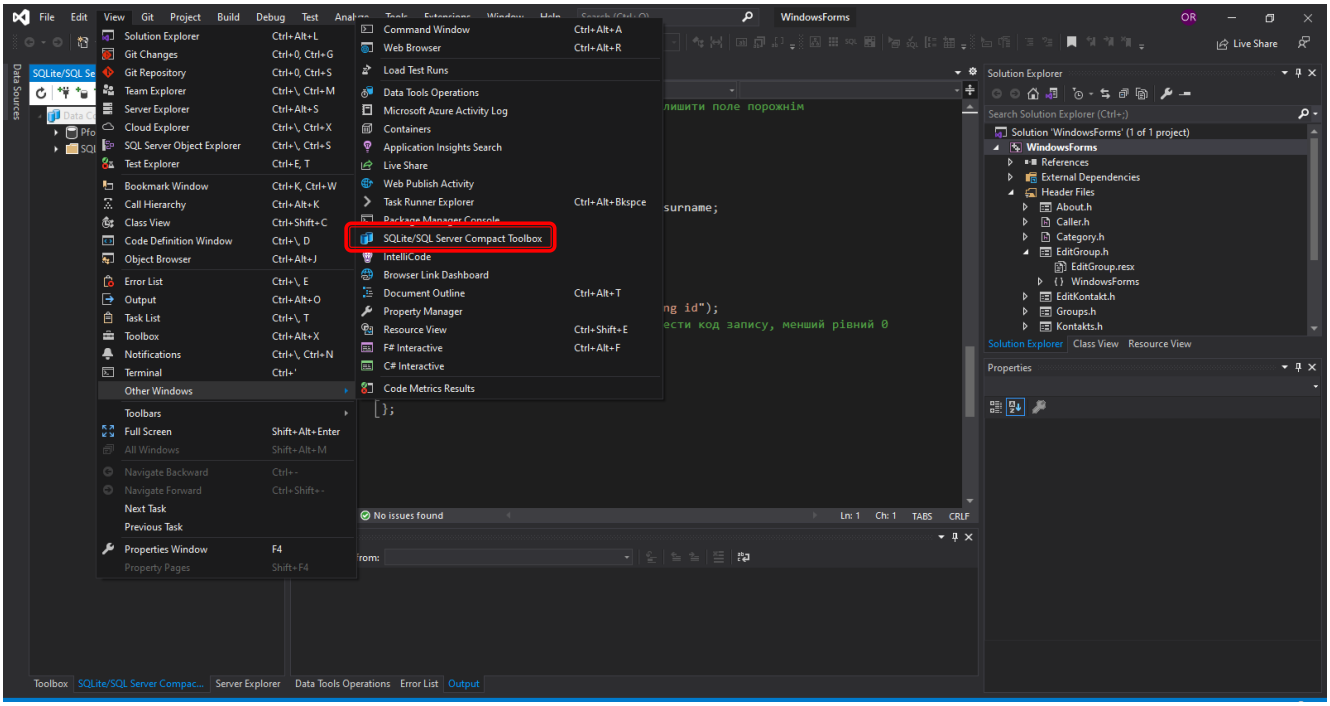

Рисунок 2 – Вигляд вікна активації «SQLite/Server Explorer Compact Toolbox»

6. Або Tools→SQLite/Server Explorer Compact Toolbox (рисунок3).

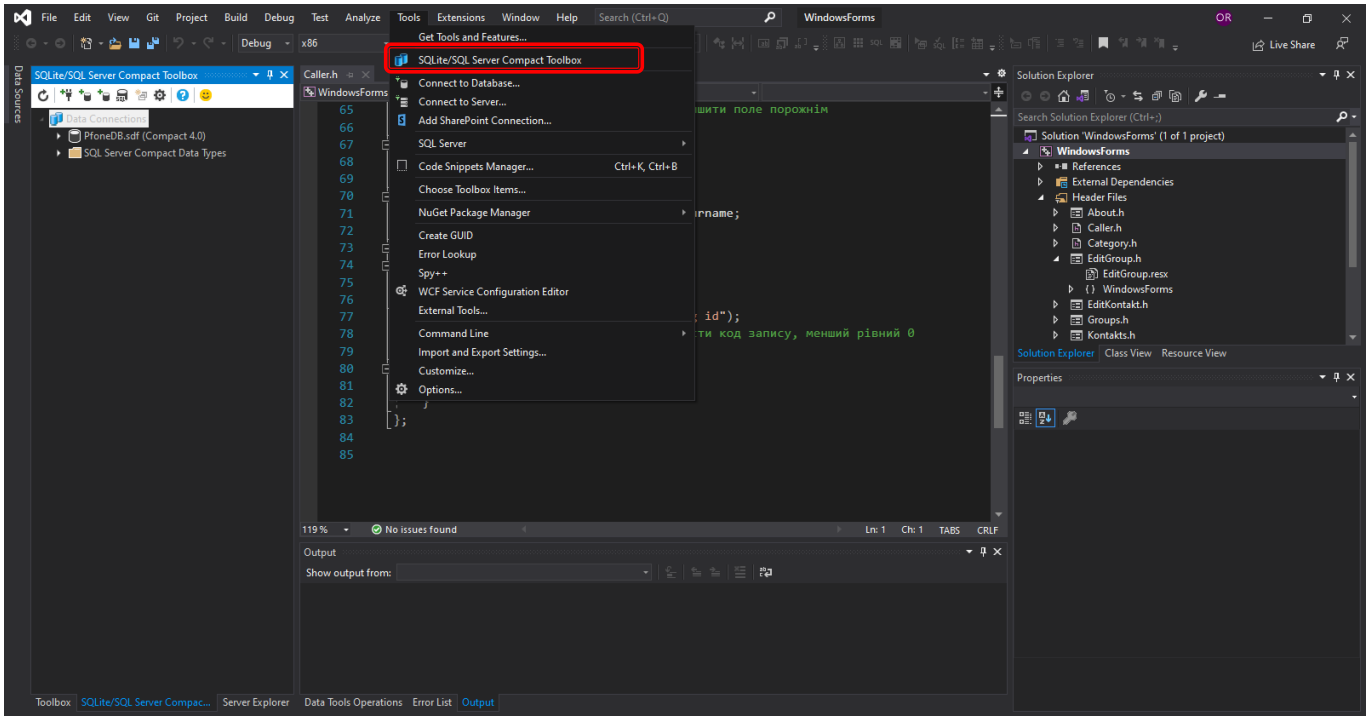

Рисунок 3 – Вигляд вікна активації «SQLite/Server Explorer Compact Toolbox»

7. **Створити демонстраційну базу даних Телефонний довідник, яка міститиме таблиці Абоненти та Категорії абонентів.** Щоб викликати вікно створення бази даних необхідно викликати команду **Add SQL CE 4.0 Connection**. (рисунок 4).

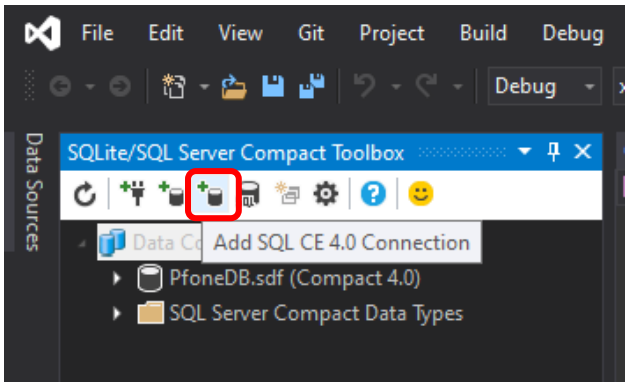

Рисунок 4 – Виклик команди Add SQL CE 4.0 Connection

8. Ознайомитись із вікном **«Add SQL Server Compact Connection».** У результаті виконання попередньої команди відкривається вікно **«Add SQL Server Compact Connection»** (рисунок 5). У цьому вікні користувач має можливість:

– створити новий або вибрати вже існуючий файл бази даних (Database file name);

– при необхідності задати пароль входу в базу даних;

– задати розмір бази в MB;

– перевірити зв’язок з базою даних (кнопка “Test Connection”).

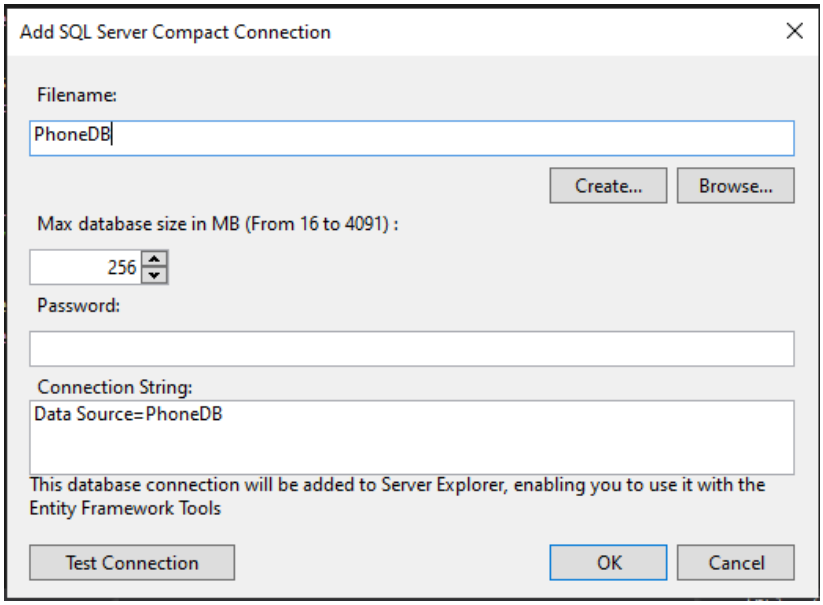

Рисунок 5 – Вікно «Add SQL Server Compact Connection»

9. У нашому випадку потрібно ввести назву бази даних **“PhoneDB”**, як показано на рисунку 5, та натиснути кнопку «**Create…**» та вказати папку де буде зберігатися БД (рисунок 6).

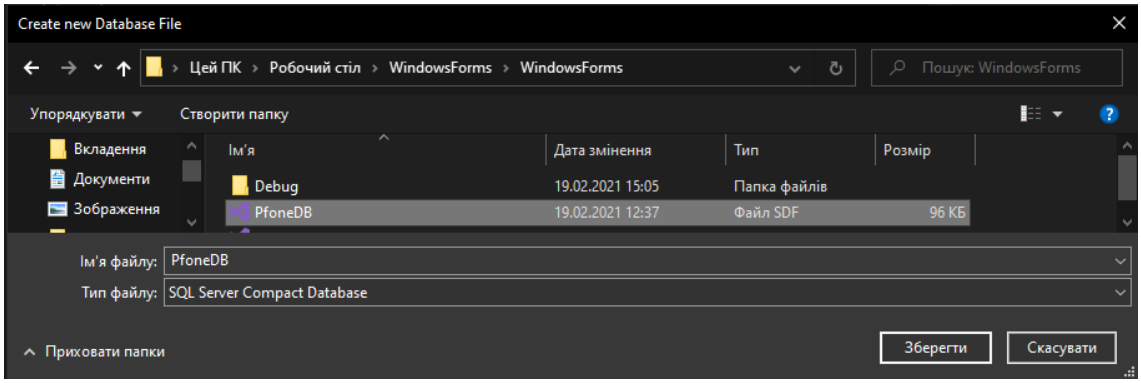

Рисунок 6 – Створення бази даних PhoneDB

10. Після успішного підтвердження створення бази даних слід натиснути на кнопку **“OK”** (рисунок 7), після чого базу даних **PhoneDB.sdf** буде додано до вікна SQLite/Server Explorer Compact Toolbox (рисунок 8).

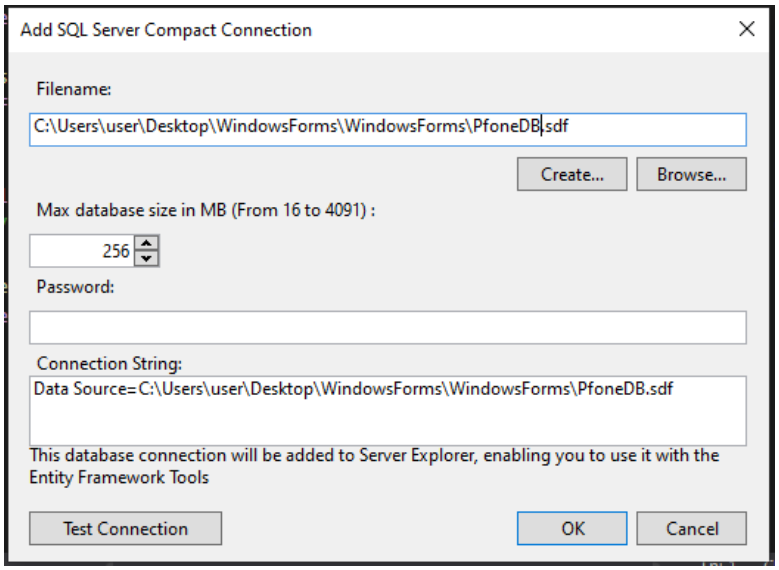

Рисунок 7 – Створення бази даних PhoneDB

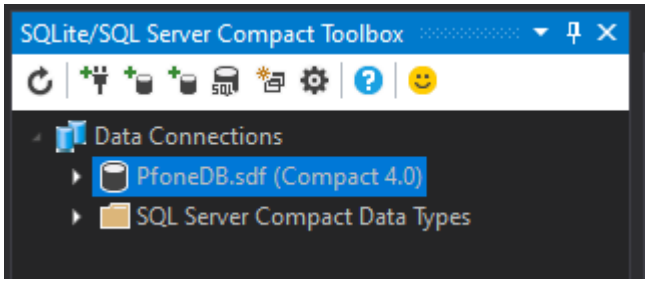

Рисунок 8 – Вікно SQLite/Server Explorer Compact Toolbox після створення бази даних «PhoneDB.sdf»

11. Створити таблицю **Categories** для збереження інформації про категорії абонентів з наступними полями:

– **id** (int) – поле-ідентифікатор записів, цілого типу, первинний ключ таблиці (Primary key);

– **category** (nvarchar) – поле текстового типу змінної довжини, що вміщує в собі назву категорії;

– **description** (nvarchar) - поле текстового типу змінної довжини, що вміщує в собі опис категорії.

12. Поки що в базі даних PhoneDB немає ніяких таблиць. Щоб створити таблицю, потрібно викликати контекстне меню (клік правою кнопкою мишки) і вибрати команду **«Build Table(beta)…»** (рисунок 9).

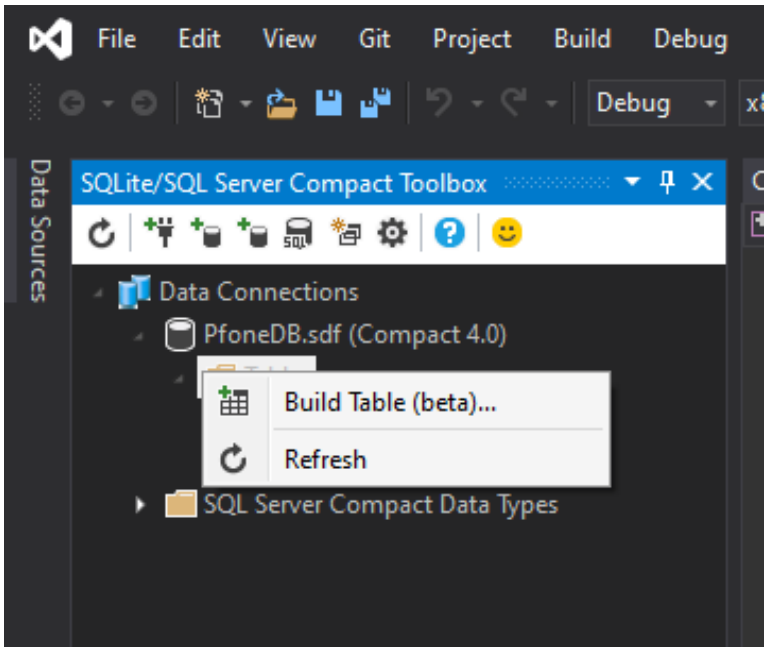

Рисунок 9 – Команда додавання нової таблиці

13. У результаті відкриється вікно додавання таблиці, яке містить наступні стовпці (рисунок 10): у першому стовпці **“Column Name”** потрібно ввести назву відповідного поля таблиці бази даних, у другому стовпці **“Data Type”** потрібно ввести тип даних цього поля, у третьому стовпці **“Length** задається розмір поля, у четвертому стовпці **“Allow Nulls”** вказується опція про можливість відсутності даних у полі, у п’ятому **“Primary key”** вказується опція про створення первинного ключа таблиці для ідентифікації записів, у шостому “**Default”** вказується опція про значення поля по замовчуванням і в сьомому **“Identity”** вказується опція для того, щоб дозволити даному полю заповнювати значення автоматично**.**

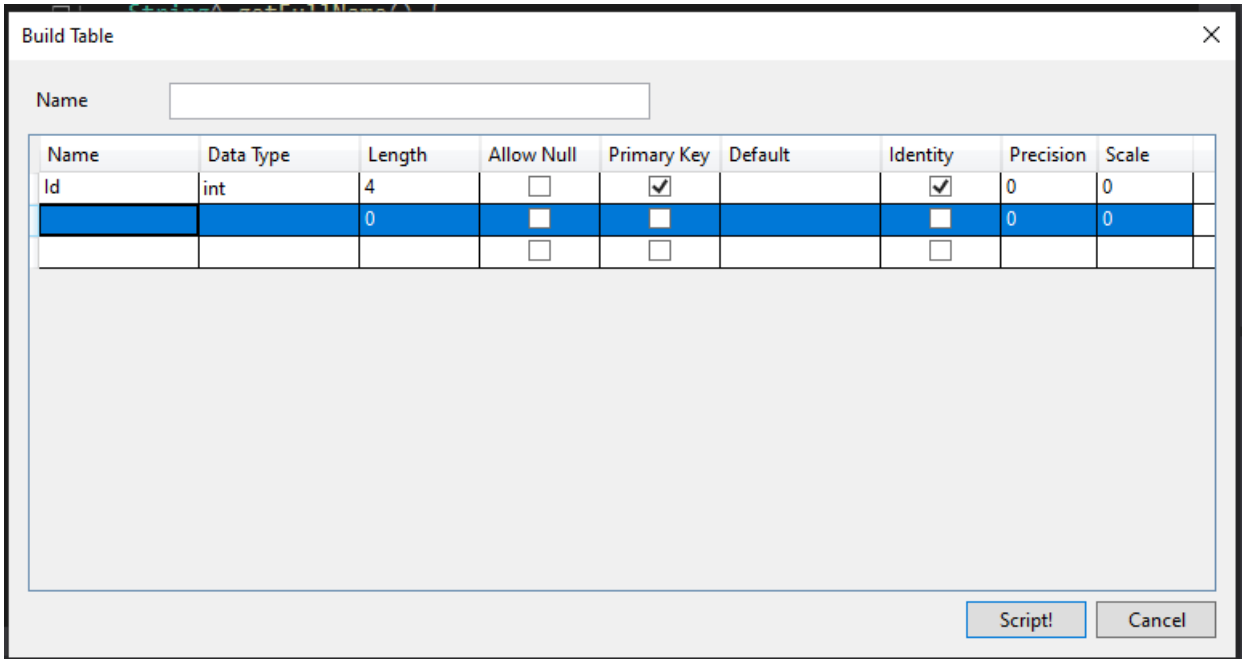

Рисунок 10 – Вікно створення нової таблиці

14. Таким чином, за допомогою даного візуального редактора таблиць потрібно сформувати таблицю **Categories** як показано на рисунку 11.

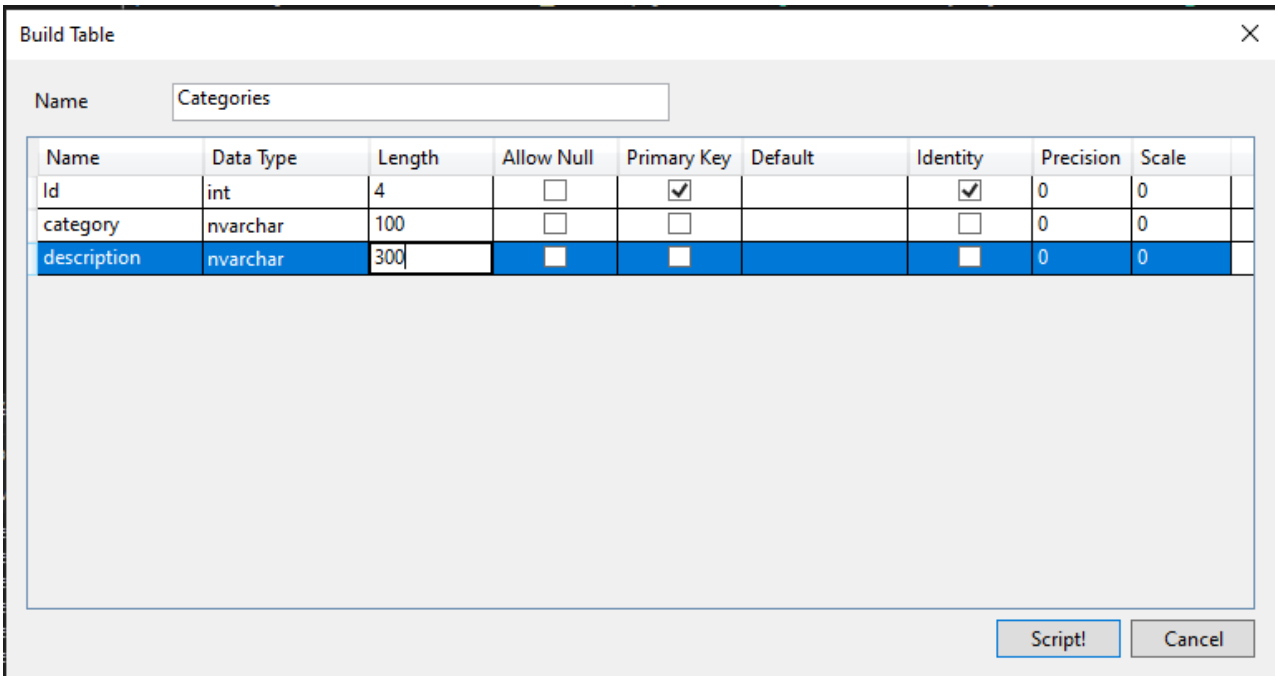

Рисунок 11 – Вигляд вікна створення таблиці Categories

15. Тиснемо на кнопку Script! (рисунок 12)

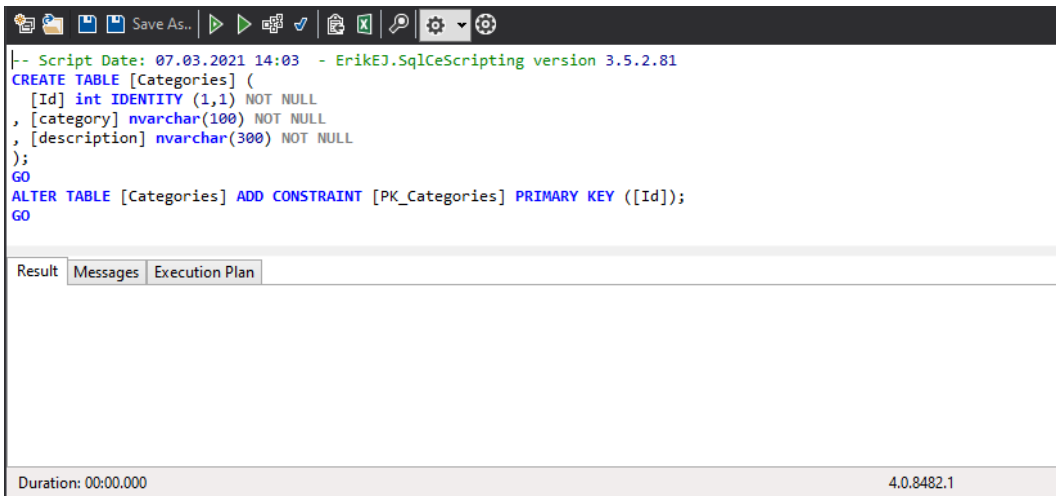

Рисунок 12 – Запит на створення таблиці Categories

16. Запускаємо запит на виконання, натиснувши кнопку  Execute(Ctrl+E).

17. Щоб вносити зміни в таблиці бази даних в MS Visual Studio, спочатку потрібно зняти опцію **“Prevent Saving changes that require table re-creation” (Не допускать изменений, требующих повторного создания таблиц)** як показано на рисунку 12. Інакше, MS Visual Studio буде блокувати внесення змін в раніше створену таблицю. Вікно Options (Параметры), яке показане на рисунку 13, викликається з меню Tools (Сервис) в такій послідовності:

Tools (Сервис)→Options (Параметры)→Database Tools (Инструменты базы данных) →Table and Database Designers (Параметры таблицы и базы данных):

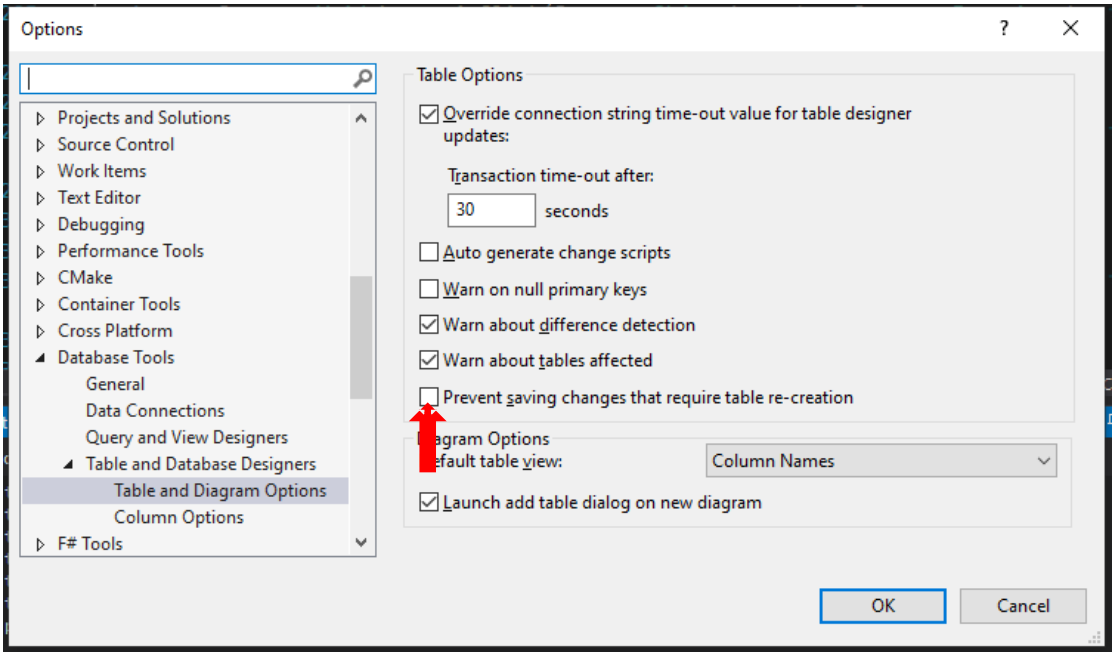

Рисунок 13 – Вікно виключення опції блокування зміни таблиці

18. **Створити за аналогією таблицю Callers для збереження інформації про абонентів з наступними полями:**

– **id** (int) – поле-ідентифікатор записів, цілого типу, первинний ключ таблиці;

– **FirstName** (nvarchar) – поле текстового типу змінної довжини, що вміщує в собі ім’я абонента;

– **LastName** (nvarchar) - поле текстового типу змінної довжини, що вміщує в собі прізвище абонента;

– **phone** (nvarchar) – поле текстового типу змінної довжини, що вміщує в собі номер телефону абонента;

– **categor** (int) – поле цілого типу, вторинний ключ (Foreign key), що вміщує в собі код категорії абонентів для зв’язку з таблицею categories.

19. Вигляд структури таблиці зі властвостями її полів наведено на рисунку 14

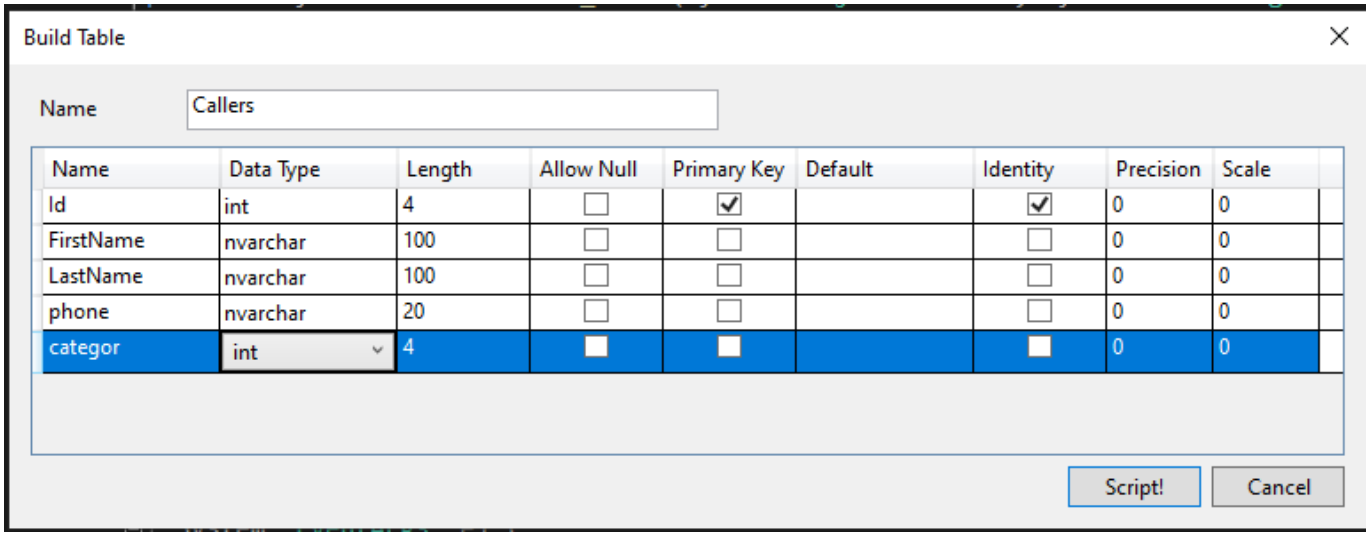

Рисунок 14– Вигляд таблиці Callers у редакторі таблиці

20. Тиснемо на кнопку Script! (рисунок 15)

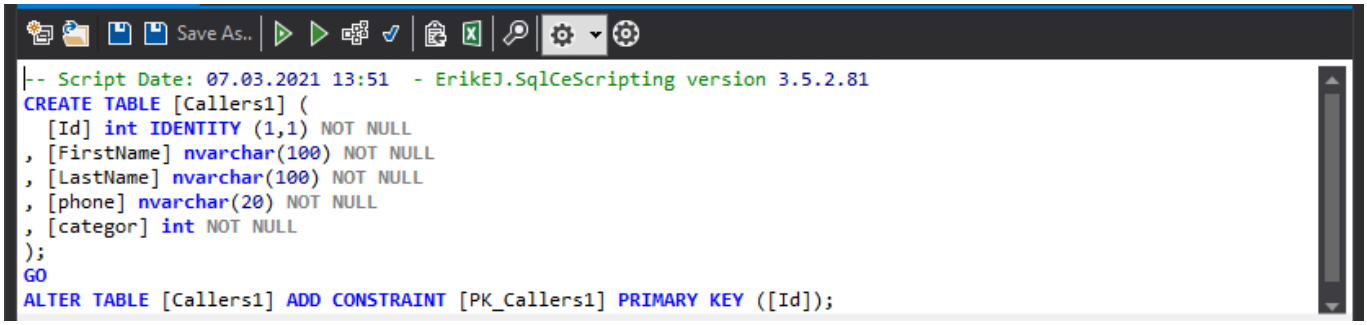

Рисунок 15 – Запит на створення таблиці Callers

21. Запускаємо запит на виконання, натиснувши кнопку  Execute(Ctrl+E).

22. Після виконаних дій, у вікні Server Explorer (Обозреватель серверов) буде відображатись дві таблиці **Categories** та **Callers**. Таким чином, в базу даних можна додавати будь-яку кількість таблиць.

23. **Створити зв'язок між таблицями Categories та Callers.** Згідно з умовою задачі, таблиці повинні бути зв'язані між собою за допомогою полів **Categories.id** та **Callers.categor**. Для цього потрібно обрати команду **Add foreign key…** з контекстного меню, викликаного для таблиці **Callers** (рисунок 16):

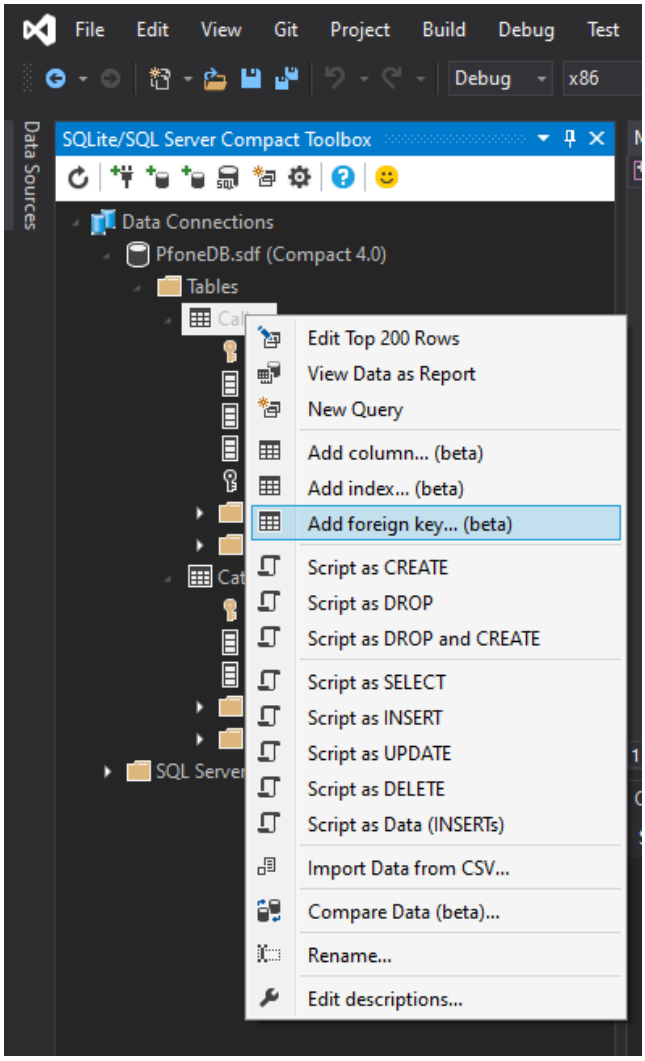

Рисунок 16 – Виклик команди додавання зовнішнього ключа

24. У вікні, що з'явилось, слід вказати ім'я зв'язку, далі обрати таблицю з первинним ключем **Categories(Id)**, потім обрати поле, призначене для зв'язку, **categor**. Вказати дію на каскадне оновлення і видалення, після чого вигляд вікна створення зв'язку між таблицями повинен набути вигляду, представленому на рисунку 17.

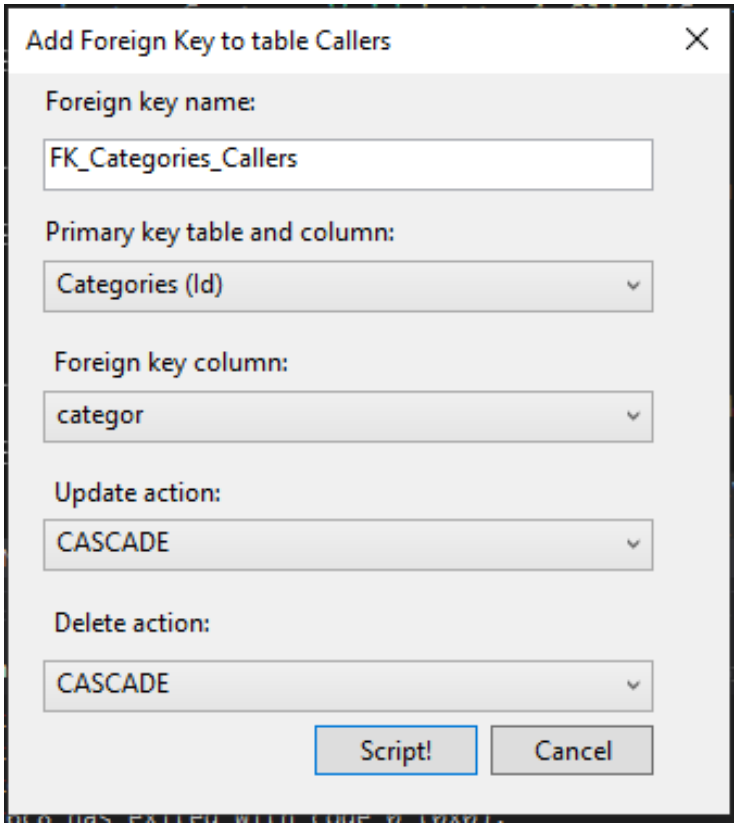

Рисунок 17 – Вікно створення зв’язку між таблицями

25. Тиснемо на кнопку Script! (рисунок 18)

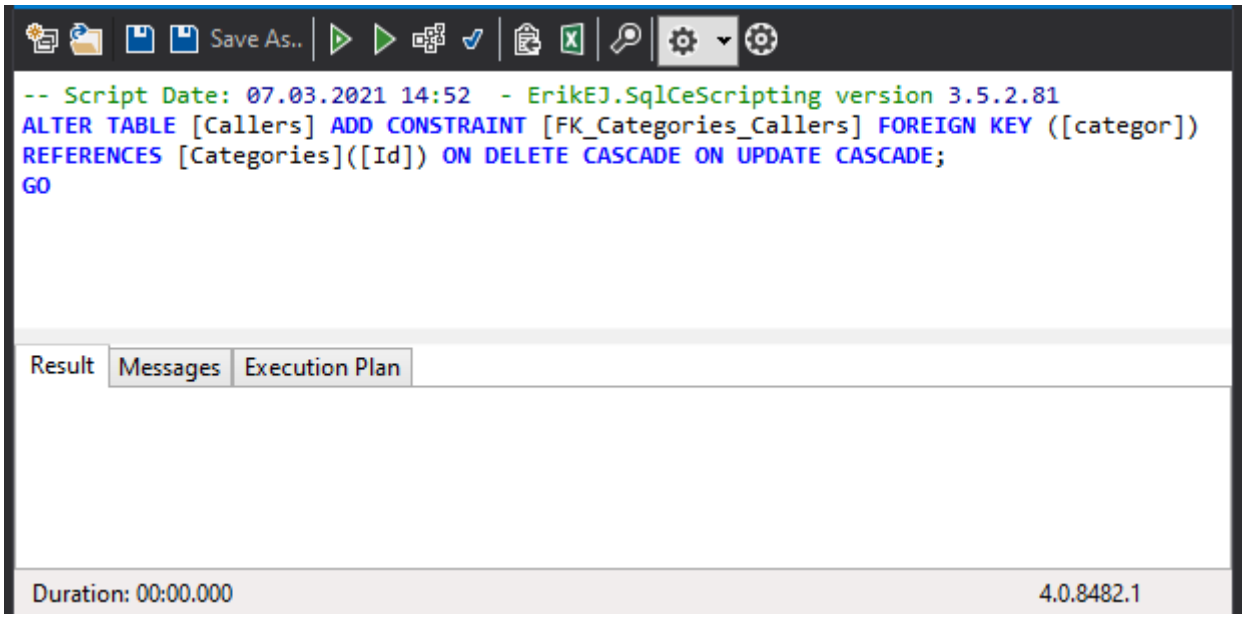

Рисунок 18 – Запит на створення вторинного ключа

26. Запускаємо запит на виконання, натиснувши кнопку  Execute(Ctrl+E).

27. **Внести дані до таблиць.** Система Microsoft Visual Studio дозволяє безпосередньо вносити дані в таблиці бази даних. У нашому випадку, при встановленні зв'язку **первинною (Primary Key Table and column)** обрано таблицю **Categories(Id)**. Тому, спочатку потрібно вносити дані в комірки саме цієї таблиці. Якщо спробувати спочатку внести дані до таблиці **Callers**, то система заблокує такий ввід з відповідним повідомленням.

28. Щоб викликати режим вводу даних в таблицю, потрібно викликати команду **Edit Top 200 Rows** з контекстного меню (клік правою кнопкою мишки) (рисунок 19).

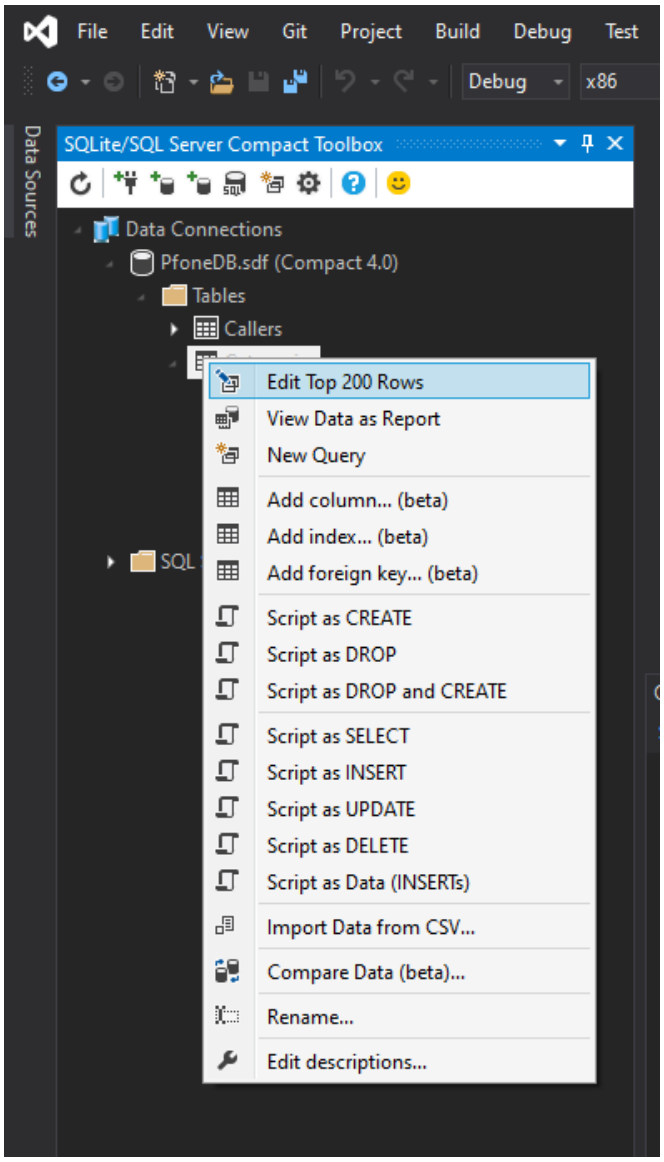

Рисунок 19 – Команда внесення даних у таблицю

29. Після цього відкривається вікно, в якому потрібно ввести вхідні дані (рисунок 20.

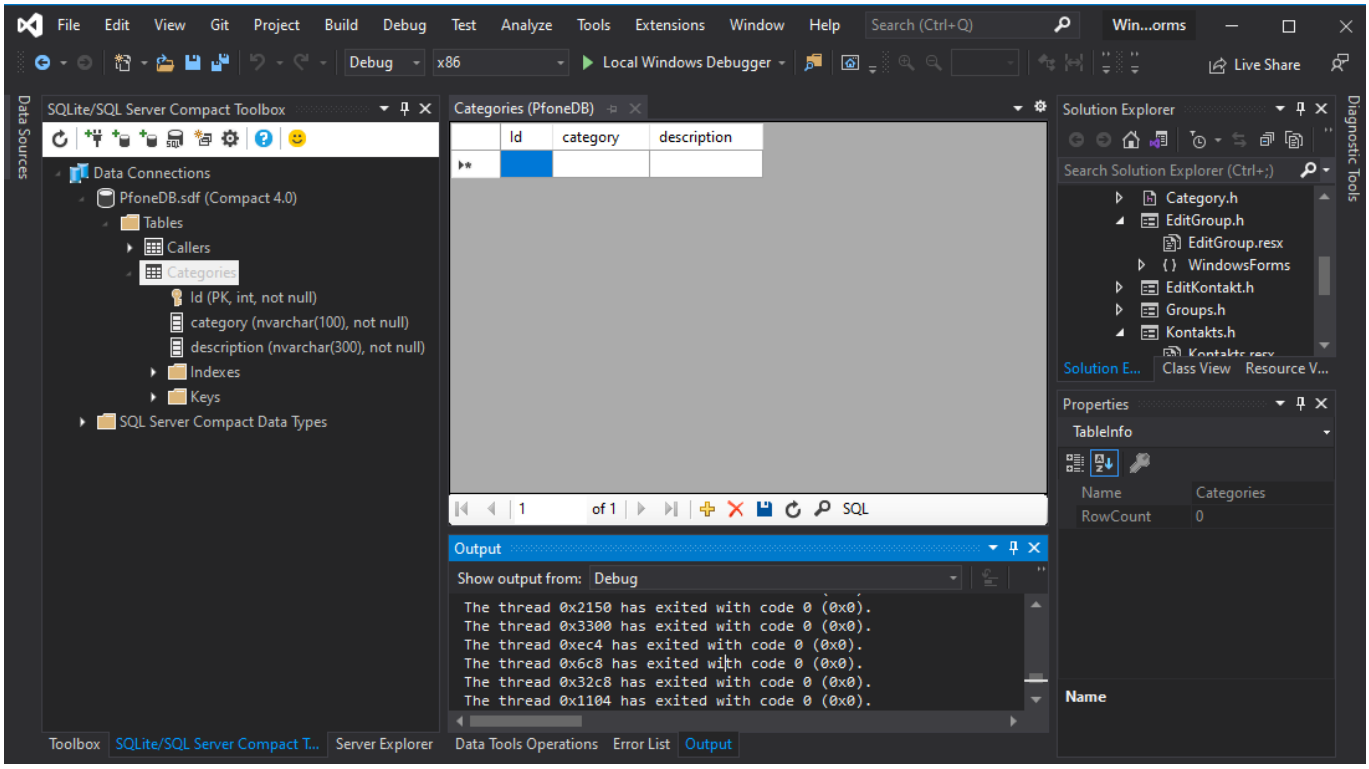

Рисунок 20 – Вікно вводу даних до таблиці Categories

30. Після внесення даних у таблицю categories потрібно внести дані у таблицю Callers.

31. При внесенні даних у поле **categor** таблиці **Callers** потрібно вводити такі самі дані, які є введені в полі **Id** таблиці **Categories**, тому що ці поля є зв’язаними між собою і повинні співпадати як за типом, так і за значеннями.

32. Тепер база даних готова до роботи. Ввід та обробку даних в таблицях можна реалізувати програмним шляхом.

Лабораторне завдання

1. У протоколі лабораторної роботи надати відповіді на контрольні питання.
2. У протоколі лабораторної роботи представити порядок створення БД(згідно обраної предметної області таблиця 1).
3. Створити на комп’ютері програмний проєкт у середовищі Visual C++ для роботи з БД.
4. В БД створити 2 взаємозв’язані таблиці.

Таблиця 1 – Рекомендований перелік предметних областей

№	Тематика БД	№	Тематика БД
1	Абітурієнт	37	Крилаті фрази
2	Автомобілі	38	Кухонні меблі
3	Автосалон	39	Лікарські рослини
4	Автостанція	40	Літаки
5	Адміністратор готелю	41	Мобільне покриття
6	Акваріумні рибки	42	Мобільні телефони
7	Анкета для ОВІРу	43	Монітори
8	Аптека	44	Мотоспорт
9	Аудіо-відео техніка	45	Нерухомість
10	Баскетбол	46	Обласні центри України
11	Бібліотека КЕПу	47	Облік кадрів
12	Біржа праці	48	Облік студентів
13	Бокс	49	Особиста бібліотека
14	Валюта	50	Ощадбанк
15	ВЗН	51	Перевезення
16	Відвідування студентів	52	Поліклініка
17	Відділ кадрів	53	Продаж білетів у кінотеатрі
18	ВРУ	54	Продаж книг
19	Довідник командира	55	Проєкти котеджів
20	Довідник любителя живопису	56	РАГС(весілля)
21	Довідник нумізмата	57	Рибалка
22	Довідник туриста	58	Рієлтерська контора (купівля-продаж житла)
23	Довідник філателіста	59	Склад
24	Домашні тварини	60	Стилі музики
25	Залізничний вокзал	61	Теніс
26	Замовлення мірроприємства	62	Транспортування
27	Записна книжка	63	Туристичне агентство
28	Зелене господарство	64	Туризм
29	Каса автовокзалу	65	Успішність студентів
30	Каталог запчастин автомобілів	66	Фітнес
31	Каталог квітів	67	Формула 1
32	Каталог радіодеталей (довідник телевізійного обладнання).	68	Фото альбоми
33	Кінофільми	69	Футбол
34	Комунальні послуги	70	Черга на житло
35	Космос	71	Щоденник(база намічуваних заходів)
36	Країни	72	Ювелірні вироби

Контрольні запитання

1. За допомогою якої команди можна створити базу даних
2. За допомогою якої команди можна створити таблицю бази даних
3. За допомогою якої команди можна видалити базу даних

4. За допомогою якої команди можна видалити таблицю з бази даних
5. Назвіть числові типи даних
6. Назвіть типи даних, що представляють дату і час
7. Назвіть рядкові типи даних
8. Назвіть бінарні типи даних
9. Для чого використовують PRIMARY KEY
10. Що визначає атрибут IDENTITY
11. Що визначає атрибут NULL і NOT NULL
12. Що визначає атрибут DEFAULT
13. Які дії можна встановити за допомогою виразів ON DELETE і ON UPDATE