Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Факультет Автоматики і комп'ютеризованих технологій
(повна назва)

Кафедра Комп'ютерно-інтегрованих технологій, автоматизації та робототехніки
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

Розроблення програмного модуля для обліку комерційних операцій на виробничому підприємстві
(тема)

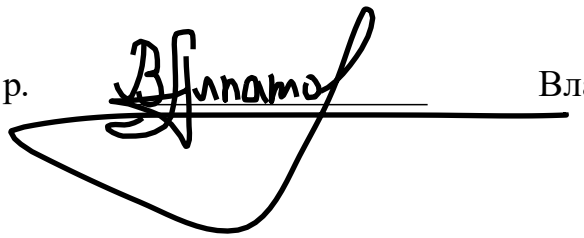
Виконав:
Здобувач 4 року навчання,
групи АКТСІ-21-1
Владислав АЛПАТОВ.
(власне ім'я прізвище)
Спеціальності 151 Автоматизація та комп`ютерно-інтегровані технології
(код і повна назва спеціальності)
Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)
Освітня програма Системна інженерія.
(повна назва освітньої програми)
Керівник доцент Олег ЗАМІРЕЦЬ
(посада, власне ім'я прізвище)

Допускається до захисту

Завідувач кафедри КІТАР _____ Ігор НЕВЛЮДОВ
(підпис) (власне ім'я прізвище)

2025р.

Я, Алпатов Владислав Михайлович, як здобувач вищої освіти ХНУРЕ, розумію та підтримую політику закладу з академічної доброчесності. Я не надавав і не одержував недозволену допомогу під час підготовки кваліфікаційної роботи. Я не використовував штучний інтелект для підготовки кваліфікаційної роботи. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

«22» червня 2025 р.  Владислав АЛПАТОВ

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
РАДІОЕЛЕКТРОНІКИ

Факультет Автоматики і комп'ютеризованих технологій
Кафедра Комп'ютерно-інтегрованих технологій, автоматизації та робототехніки
Рівень вищої освіти перший (бакалаврський)
Спеціальність 151 Автоматизація та комп`ютерно-інтегровані технології
Тип програми освітньо-професійна
Освітня програма Системна інженерія
(шифр і назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
« 28 » квітня 2025 р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Алпатову Владиславу Михайловичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розроблення програмного модуля для обліку комерційних операцій на виробничому підприємстві
затверджена наказом по університету від " 19 " травня 2025 р. № 388 Ст.
2. Термін подання здобувачем роботи " 24 " червня 2025 р.
3. Вихідні дані до роботи 3.1 Інтерфейс користувача – графичний;
3.2 Тип СУБД – SQL;
3.3 Функції програмного модуля – автоматизація створення структури БД, реалізація функцій додавання, редагування та видалення даних;
3.4 Операційна система – Windows;
3.5 Оформлення текстової документації – ДСТУ 3008-2015.

4. Перелік питань, що потрібно опрацювати в роботі 4.1 Вступ;
4.2 Аналіз технічного завдання та предметної області;
4.3 Розроблення структури програмного модуля, алгоритму його роботи та структури бази даних;
4.4 Розроблення програмного забезпечення;
4.5 Практична реалізація та випробування розробленого програмного модуля;
4.6 Висновки.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) Демонстраційний матеріал представлений у форматі презентації PowerPoint (*.ppt) – 16 с. формату А4 .

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Аналіз технічного завдання та предметної області	28.04 – 04.05.25	виконано
2	Розроблення структури програмного модуля, алгоритму його роботи та структури бази даних	05.05 – 14.05.25	виконано
3	Розроблення програмного забезпечення	15.05 – 28.05.25	виконано
4	Практична реалізація та випробування розробленого програмного модуля	29.05 – 10.06.25	виконано
5	Оформлення пояснювальної записки	11.06 – 14.06.25	виконано
6	Подання роботи на перевірку Інтернет-системою StrikePlagiarism	15.06 – 17.06.25	виконано
7	Подання роботи на рецензію	18.06 – 20.06.25	виконано
8	Подання роботи на підпис зав. кафедри	21.06 – 23.06.25	виконано
9	Подання кваліфікаційної роботи в ЕК	24.06.25	виконано

Дата видачі завдання 28.04.2025 р.

Здобувач _____ Владислав АЛПАТОВ
(підпис)

Керівник роботи _____ доцент Олег ЗАМІРЕЦЬ
(підпис) (посада, власне ім'я прізвище)

РЕФЕРАТ

Пояснювальна записка: 64 с., 6 табл., 35 рис., 2 дод., 16 джерел.

ПРОГРАМНИЙ МОДУЛЬ, БАЗА ДАНИХ, СУБД, ТАБЛИЦЯ, АЛГОРИТМ, ДІАЛОГ З КОРИСТУВАЧЕМ, C#, SQLITE, VISUAL STUDIO, ОБЛІК, КОМЕРЦІЙНІ ОПЕРАЦІЇ.

Мета роботи – розроблення програмного модуля для обліку комерційних операцій виробничого підприємства на основі бази даних SQLite.

Об’єкт розробки – процеси обліку комерційних операцій виробничого підприємства.

Предмет розробки – програмний модуль для обліку комерційних операцій виробничого підприємства.

В результаті виконання кваліфікаційної роботи була описана структура програмного модулю, що розроблюється, наведена структура файлової системи для зберігання опису таблиць. Розроблено алгоритми роботи програми, описані параметри вхідної інформації та розроблена структура бази даних. Для створення програмного модуля було використано мову програмування C# та середовище розробки VisualStudio. Описано інтерфейс користувача та програмні функції. Виконано тестовий приклад створення структури бази даних для зберігання інформації про замовлення виготовлення виробів на виробничому підприємстві. Працездатність програми була перевірена на практичному прикладі. Всі таблиці було створено без помилок. Було виконано додавання тестових даних, які потім можна було переглянути за допомогою вбудованого табличного інструменту.

Отримані результати роботи можна віднести до Цілі сталого розвитку 9 «Промисловість, інновації та інфраструктура», зокрема до пункту 9.4 «Розвиток високотехнологічного машинобудування».

ABSTRACT

Explanatory note: 54 pp., 6 tab., 35 figs., 2 appendices, 18 sources.

PROGRAM MODULE, DATABASE, SUBD, TABLE, ALGORITHM, USER DIALOG, C#, SQLITE, VISUAL STUDIO, ACCOUNTING, COMMERCIAL OPERATIONS.

The purpose of the work is to develop a program module for accounting for commercial operations of a manufacturing enterprise based on the SQLite database.

Object of development – automation of accounting of commercial operations of a manufacturing enterprise.

Subject of development – a software module for accounting for commercial operations of a manufacturing enterprise.

As a result of the qualification work, the structure of the program module under development was described, the structure of the file system for storing table descriptions was given. The algorithms of the program were developed, the parameters of the input information were described, and the structure of the database was developed. The C# programming language and the VisualStudio development environment were used to create the program module. The user interface and program functions are described. A test example of creating a database structure for storing information about orders for the manufacture of products at a manufacturing enterprise has been performed. The program's performance was tested on a practical example. All tables were created without errors. Test data was added, which could then be viewed using the built-in tabular tool.

The results of the work can be attributed to Sustainable Development Goal 9 “Industry, Innovation and Infrastructure”, in particular to paragraph 9.4 “Development of high-tech engineering”.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

БД – база даних;

ОС – операційна система;

ПЗ – програмне забезпечення;

ПК – персональний комп’ютер;

СУБД – система управління базою даних;

DLL– динамічно приєднувана бібліотека;

SQL – мова структурованих запитів.

ВСТУП

Для користувачів, які по роду діяльності працюють з даними, важливим є допоміжний засіб для швидкого створення інструменту зберігання і обробки даних в залежності від певних вимог. Насамперед, таким інструментом є простий, але ефективний движок для зберігання даних, яким є SQLite.

SQLite – це бібліотека для реалізації вбудованих автономних рішень, які можуть працювати без виділеного серверу, але реалізовувати весь спектр запитів в форматі СУБД SQL. Особливістю є те, що сам процес не треба настроювати в системі. Всі необхідні запити обробляються у нутрі бібліотеці.

SQLite не є автономним процесом, як інші бази даних (БД), тобто його можна поєднати статично або динамічно зі своєю програмою або сервісом відповідно до завдання. Програма з SQLite безпосередньо звертається до своїх файлів в яких зберігаються дані.

Актуальність розробки полягає в створенні універсального інструменту для управління даними за допомогою якого можна створити будь-яку структуру бази даних в залежності від потреб користувача. Особливістю програми є інтерфейс, який підстроюється під структуру даних, що була створена.

Мета роботи – розроблення програмного модуля для обліку комерційних операцій виробничого підприємства на основі бази даних SQLite.

Об’єкт розробки – процеси обліку комерційних операцій виробничого підприємства.

Предмет розробки – програмний модуль для обліку комерційних операцій виробничого підприємства.

Програмний модуль, що розроблюється призначений для автоматизації роботи за БД SQLite. Згідно з технічним завданням, програма повинна мати можливість працювати з різним набором та структурою даних. Також, даний інструмент повинен мати гнучкий алгоритм створення інтерфейсу діалогових вікон, щоб можна було відображати різні дані.

Для управління даними в структурі програми передбачені діалогові вікна. За допомогою них користувач може додавати, або редагувати дані в таблицях.

Для досягнення поставленої мети необхідно розробити програмний модуль який міг би автоматизувати такі функції оператора БД:

– створювати структуру БД;

– задавати різні поля таблиць даних;

– додавати дані в створені таблиці;

– відображати введені дані в табличній формі;

– редагувати, та видаляти дані.

Пояснювальну записку кваліфікаційної роботи оформлено згідно з ДСТУ 3008:2015 [1], а також з рекомендаціями з підготовки і оформлення кваліфікаційної роботи здобувачами першого (бакалаврського) рівня вищої освіти [2-3].

Отримані результати роботи можна віднести до Цілі сталого розвитку 9 «Промисловість, інновації та інфраструктура», зокрема до пункту 9.4 «Розвиток високотехнологічного машинобудування».

1 АНАЛІЗ ТЕХНІЧНОГО ЗАВДАННЯ ТА ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз вимог технічного завдання

Мета роботи: створення програмного модуля для автоматизації роботи з базою даних SQLite.

Для досягнення поставленої мети необхідно розробити програмний модуль який міг би автоматизувати такі функції оператора бази даних [1]:

– створювати структуру бази даних;

– задавати різні поля таблиць даних;

– додавати дані в створені таблиці;

– відображати введені дані в табличній формі;

– редагувати, та видаляти дані.

Програма повинна працювати під керуванням операційної системи Windows. Кожна операція повинна виконуватись з використанням діалогових вікон з відповідним інтерфейсом.

Актуальність розробки полягає в створенні універсального інструменту для управління даними за допомогою якого можна створити будь-яку структуру бази даних в залежності від потреб користувача. Особливістю програми є інтерфейс, який підстроюється під структуру даних, що була створена.

1.2 Опис особливостей використання СУБД SQLite

SQLite – це бібліотека для реалізації вбудованих автономних рішень, які можуть працювати без виділеного серверу, але реалізовувати весь спектр запитів в форматі СУБД SQL[3]. Особливістю є те, що сам процес не треба настроювати в системі. Всі необхідні запити обробляються у нутрі бібліотеці.

SQLite не є автономним процесом, як інші бази даних, тобто його можна поєднати статично або динамічно зі своєю програмою або сервісом відповідно до завдання. Програма з SQLite безпосередньо звертається до своїх файлів в яких зберігаються дані.

Особливості SQLite:

– SQLite не вимагає окремого процесу сервера або системи для роботи (без сервера);

– SQLite поставляється з нульовою конфігурацією, що означає відсутність необхідності в налаштуванні або адмініструванні;

– повна база даних SQLite зберігається в одному кросс-платформенномфайлі;

– SQLite займає мало місця, менше 400кб і при цьому є повністю конфігурований;

– SQLite є автономним, що означає відсутність зовнішніх залежностей;

– SQLite-транзакции повністю сумісні з ACID, забезпечуючи безпечний доступ до декількох процесів або потоків;

– SQLite підтримує більшість функцій мови запитів, знайдених в стандарті SQL92 (SQL2);

– SQLite написаний на ANSI-C і надає простий і простий у використанні API;

– SQLite доступний в UNIX (Linux, Mac OS - X, Android, iOS) і Windows (Win32, WinCE, WinRT).

Движок БД представляє бібліотеку, з якою програма компонується і SQLite стає складовою частиною програми. Уся БД зберігається в єдиному стандартному файлі на машині, на якій виконується програма.

Декілька процесів або потоків можуть одночасно без яких-небудь проблем читати дані з однієї бази. Запис у базу можна здійснити тільки у тому випадку, якщо ніяких інших запитів в даний момент не обслуговуються, інакше спроба запису закінчується невдачею, і в програму повертається код помилки. Іншим

варіантом розвитку подій є автоматичне повторення спроб запису впродовж заданого інтервалу часу.

У SQLite є декілька непідтримуваних функцій SQL92, вони наведені в таблиці 1.1.

Таблиця 1.1 – Функції, що не підтримуються SQLite

Назва	Опис функції
RIGHT OUTER JOIN	Реалізовано тільки LEFT OUTER JOIN
FULL OUTER JOIN	Реалізовано тільки LEFT OUTER JOIN
ALTER TABLE	Підтримуються варіанти RENAME TABLE і ADD COLUMN команди ALTER TABLE. DROP COLUMN, ALTER COLUMN, ADD CONSTRAINT не підтримуються.
Triggersupport	Для кожного тригера ROW підтримуються, але не для тригерів EACH STATEMENT.
VIEWs	VIEWs в SQLite доступні тільки для читання. Не маєможливості виконувати оператор DELETE, INSERT або UPDATE в уявленні.
GRANT та REVOKE	Єдиними правами доступу, які можуть бути застосовані, є звичайні дозволи доступу до файлу базової операційної системи.

Стандартні команди SQLite для взаємодії з реляційними БД аналогічні SQL це: CREATE, SELECT, INSERT, UPDATE, DELETE і DROP:

– CREATE – створення нової таблиці, представлення таблиці або інший об'єкт у БД;

– ALTER – змінює існуючий об'єкт БД, такий як таблиця;

– DROP – видаляє усю таблицю, представлення таблиці або іншого об'єкту у БД;

– INSERT – створює запис;

– UPDATE – змінює записи;

– DELETE – видаляє записи;

– SELECT – витягає певні записи з однієї або декількох таблиць.

SQLite використовує динамічну типізацію даних [4]. Це означає, що тип стовпця не визначає тип значення, що зберігається, в цьому полі запису. Таким чином, у будь-який стовпець можна занести будь-яке значення.

Тип стовпця визначає як порівнювати значення. Але не зобов'язує заносити значення саме такого типу в стовпець.

Припустимо, ми оголосили стовпець як «INTEGER». SQLite дозволяє занести в цей стовпець значення будь-якого типу (999, «abc», «123», 678.525). Якщо значення, що вставляється, не ціле, то SQLite намагається привести його до цілого. Тобто, рядок «123» перетвориться на ціле 123, а інші значення запишуться «як є».

Можливі такі типи полів: NULL, INTEGER, REAL, TEXT, BLOB.

SQLite виконує неявні перетворення типів «на льоту» в декількох місцях:

– при занесенні значення в стовпець (тип стовпця задає рекомендацію по перетворенню);

– при порівнянні значень між собою.

Стовпець може мати наступні рекомендації приведення типу: TEXT, NUMERIC, INTEGER, REAL, NONE.

Значення BLOB і NULL завжди заносяться у будь-який стовпець «як є».

У стовпець TEXT значення TEXT заносяться «як є», значення INTEGER і REAL стають рядками. У стовпець NUMERIC, INTEGER числа записуються «як є», а рядки стають числами, якщо можуть (тобто допустиме зворотне перетворення «без втрат»). Для стовпця REAL правила схожі на INTEGER(NUMERIC), але відмінність в тому, що усі числа представлені у форматі з плаваючою комою. У стовпець NONE значення заносяться «як є».

При порівнянні значень різного типу між собою може виконуватися додаткове перетворення типів.

При порівнянні числа з рядком, якщо рядок може бути перетворений в число «без втрат», вона стає числом.

У SQLite в унікальному індексі може бути скільки завгодно NULL значень.

SQLite ретельно дотримує цілісність даних у БД (ACID), реалізовуючи механізм зміни даних через транзакції. Коротко про транзакції: транзакція або повністю накочується, або повністю відкочується. Проміжних станів бути не може. Якщо транзакції не використовується явно (BEGIN; ...; COMMIT;), то завжди створюється неявна транзакція. Вона стартує перед виконанням команди і коммітится відразу після. SQLite може вставляти і до 50 тисяч записів в секунду, але фіксувати транзакцій він не може більше, ніж 50 в секунду. Саме тому, не виходить вставляти записи швидко, використовуючи неявну транзакцію.

При налаштуваннях за умовчанням SQLite гарантує цілісність БД навіть при відключенні живлення в процесі роботи. Досягається подібна поведінка веденням журналу (спеціального файлу) і механізмом синхронізації змін на диску. Оновлення даних у БД працює так:

– до будь-якої модифікації БД SQLite зберігає змінювані сторінки з БД в окремому файлі (журналі), тобто копіює їх в нього;

– переконавшись, що копія сторінок створена, SQLite починає міняти БД;

– переконавшись, що усі зміни у БД «дійшли до диска» і БД стала цілісною, SQLite стирає журнал.

Якщо SQLite відкриває з'єднання до БД і бачить, що журнал вже є, він робить висновок, що БД знаходиться в незавершеному стані і автоматично відкочує останню транзакцію. Тобто механізм відновлення БД після збоїв, фактично, вбудований в SQLite і працює непомітно для користувача. За умовчанням журнал ведеться в режимі DELETE.

Це означає, що файл журналу видаляється після завершення транзакції. Сам факт наявності файлу з журналом в цьому режимі означає для SQLite, що транзакція не була завершена, база потребує відновлення. Файл журналу має ім'я файлу БД, до якого додане "-journal".

У режимі TRUNCATE файл журналу обрізується до нуля (на деяких системах це працює швидше, ніж видалення файлу).

У режимі PERSIST початок файлу журналу забивається нулями (при цьому його розмір не міняється і він може займати купу місця).

У режимі MEMORY файл журналу ведеться в пам'яті і це працює швидко, але не гарантує відновлення бази при помилках, тому що на дику немає копії журналу.

Можна зовсім відключити журнал (PRAGMA journal_mode = OFF). У цій ситуації перестає працювати відкат транзакцій (команда ROLLBACK) і база, швидше за все, зіпсується, якщо програма буде завершена аварійно. Для бази даних в пам'яті режим журналу може бути тільки або MEMORY, або OFF.

Сучасні системи, в тому числі і SQLite використовують кешування для підвищення продуктивності і можуть відкладати запис на диск. PRAGMA synchronous задає спеціальнийрежим кешуванняSQLite.

Режим OFF означає, щоSQLite вважає, що дані фіксовані на диску відразу після того як він передав їх операційної системи (ОС) (тобто відразу після виклику відповідного API ОС). Це означає, що цілісність гарантована при аварії програми (оскільки ОС продовжує працювати), але не при аварії ОС або відключенні живлення.

Режим синхронізації NORMAL гарантує цілісність при аваріях ОС і майже при усіх відключеннях живлення. Існує ненульовий шанс, що при втраті живлення база зіпсується. Це деякий середній, компромісний режим по продуктивності і надійності.

Режим FULL гарантує цілісність завжди і скрізь і при будь-яких аваріях. Але працює, зрозуміло, повільніше, оскільки в певних місцях робляться паузи очікування. Це режим за умовчанням.

За умовчанням, режим журналу БД завжди «повертається» в DELETE. Припустимо, ми відкрили з'єднання до БД і встановили режим PERSIST. Змінили дані, закрили з'єднання. На диску залишився файл журналу (почало якого забито нулями). Відкриваємо з'єднання до БД знову. Якщо не задати режим журналу в цьому з'єднанні, він знову працюватиме в DELETE. Як тільки ми відновимо дані, механізм фіксації транзакцій зітре файл журналу.

Режим журналу WAL працює інакше – він «постійний». Як тільки ми перевели базу в режим WAL, вона залишиться в цьому режимі, поки їй явно не поміняють режим журналу на інший.

Спочатку SQLite проектувалася як вбудована БД. Архітектура розділення одночасного доступу до даних була влаштована примітивно: одночасно декілька з'єднань можуть читати БД, а ось записувати в даний момент часу може тільки одне з'єднання. Це, як мінімум, означає, що з'єднання, що пише, чекає «звільнення» БД від тих, що читають. При спробі записати в «зайняту» БД застосування отримує помилку SQLITE_BUSY (не плутати з SQLITE_LOCKED!). Досягається цей механізм розділення доступу через API блокування файлів.

У режимі WAL (Write – AheadLogging) «читачі» БД і «постачальники аних» у БД вже не заважають один одному, тобто допускається модифікація даних при одночасному читанні.

Але є і недоліки:

– БД займає декілька файлів (файли «XXX-wal» і «XXX-shm»);

– погано працює на великих транзакціях (умовно, якщо транзакція більше 50 Мбайт);

– не можна відкрити таку БД в режимі «тільки читання»;

– виникає додаткова операція checkpoint.

Фактично, в режимі WAL дані БД розділяються між БД і файлом журналу. Операція checkpointпереносить дані у БД. За умовчанням, це робиться автоматично, якщо журнал зайняв 1000 сторінок БД.

Незважаючи на мініатюрність, SQLite в реальності не накладає серйозних обмежень на розміри полів, таблиць або БД.

Граничні параметри БД SQLite:

– за умовчанням, BLOB або текстові значення можуть займати 1 Гбайт і це тільки обмеження розміру одного запису (можна підняти цей параметр до 2^{31}, параметр SQLITE_MAX_LENGTH);

– кількість стовпців: 2000 (можна підняти до 32767, SQLITE_MAX_COLUMN);

– розмір SQL оператора: 1 Мбайт (1073741824 байт, SQLITE_MAX_SQL_LENGTH);

– одночасний join: 64 таблиці;

– кількість приєднанихбаз до з'єднання: 10 (до 62, SQLITE_MAX_ATTACHED);

– максимальна кількість сторінок у БД: 1073741823 (до 2147483646, SQLITE_MAX_PAGE_COUNT);

– якщо задати розмір сторінки 65636 байт, то максимальний розмір БД буде приблизно 14 Терабайт;

– максимальне число записів в таблиці: 2^{64}, але на практиці, звичайно, обмеження розміру вступить раніше.

1.3 Аналіз аналогічних програмних засобів

1.3.1 SQLiteStudio

Менеджер SQLite баз даних, який дозволяє переглядати і редагувати SQLite 3, SQLite 2 і SQLCipher БД.

Особливості:

– портативний – немає необхідності установки або видалення. Досить викачати, розпакувати і запустити;

– інтуїтивно зрозумілий інтерфейс;

– потужний, але легкий і швидкий інструмент;

– усі функції SQLite3 і SQLite2 помістилися в простому графічному інтерфейсі;

– крос-платформений – працює на Windows 9x / 2k / XP / 2003 / Vista / 7, Linux, MacOS X і повинен працювати на інших Unix системах;

– експорт в різні формати (SQL – операторы, CSV, HTML, XML, PDF, JSON);

– імпорт даних з різних форматів (CSV, призначені для користувача текстові файли - регулярні вирази);

– численні невеликі доповнення, такі як форматування коду, історія запитів, що виконуються у вікнах редактора, перевірка синтаксису на льоту, і багато що інше;

– підтримка Unicode;

– кольори, що настроюються, шрифти і значки;

– відкритий початковий код, який опублікований під ліцензією GPLv3.

Зовнішній вигляд інтерфейсу програми показано на рисунку 1.1.

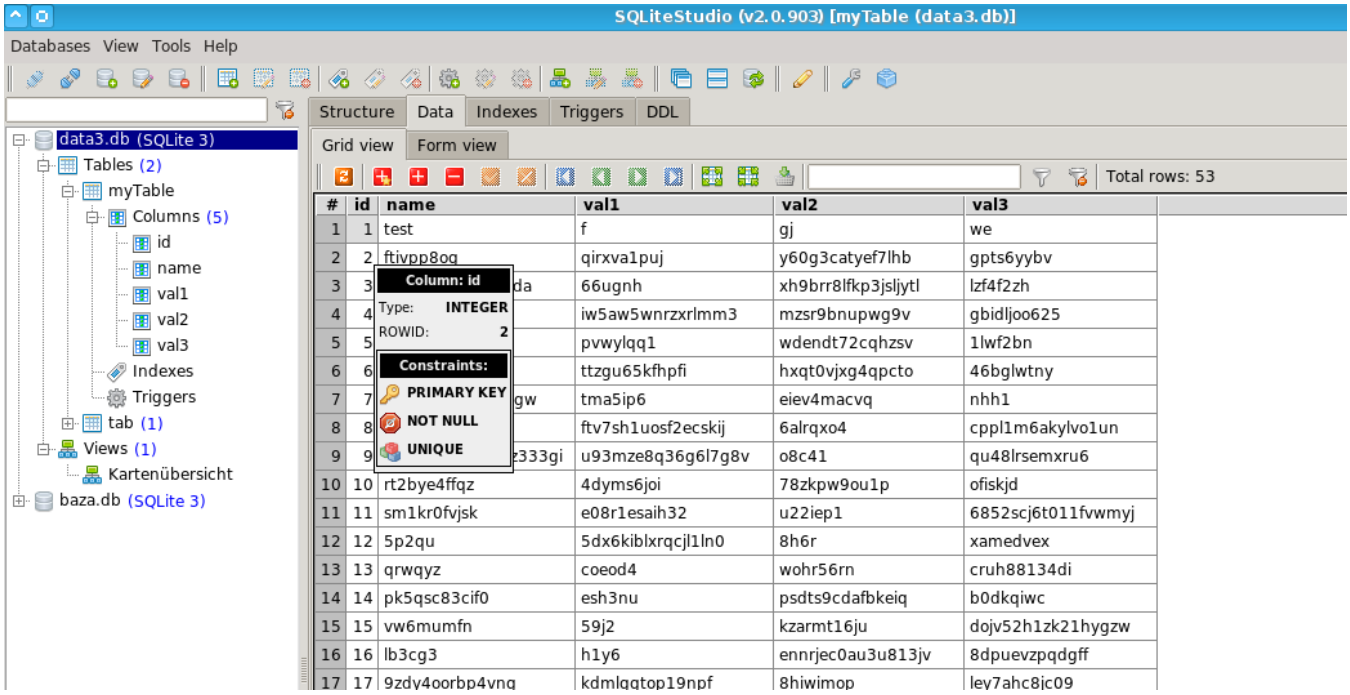


Рисунок 1.1 – Зовнішній вигляд інтерфейсу програмиSQLiteStudio

1.3.2 DB BrowserforSQLite

DB Browser for SQLite — це зручна програма з відкритим вихідним кодом, яка допоможе вам візуально створювати, проектувати та редагувати файли баз даних. Можливо, ви пам'ятаєте її як "SQLiteDatabaseBrowser" або "DatabaseBrowserforSQLite".

Це інструмент для користувачів і розробників, яким треба створювати бази даних, шукати і редагувати дані в них. Він використовує звичний табличний інтерфейс, тому не треба вивчати складні SQL-команди.

Зовнішній вигляд прогарамиDB BrowserforSQLite показано на рисунку 1.2.

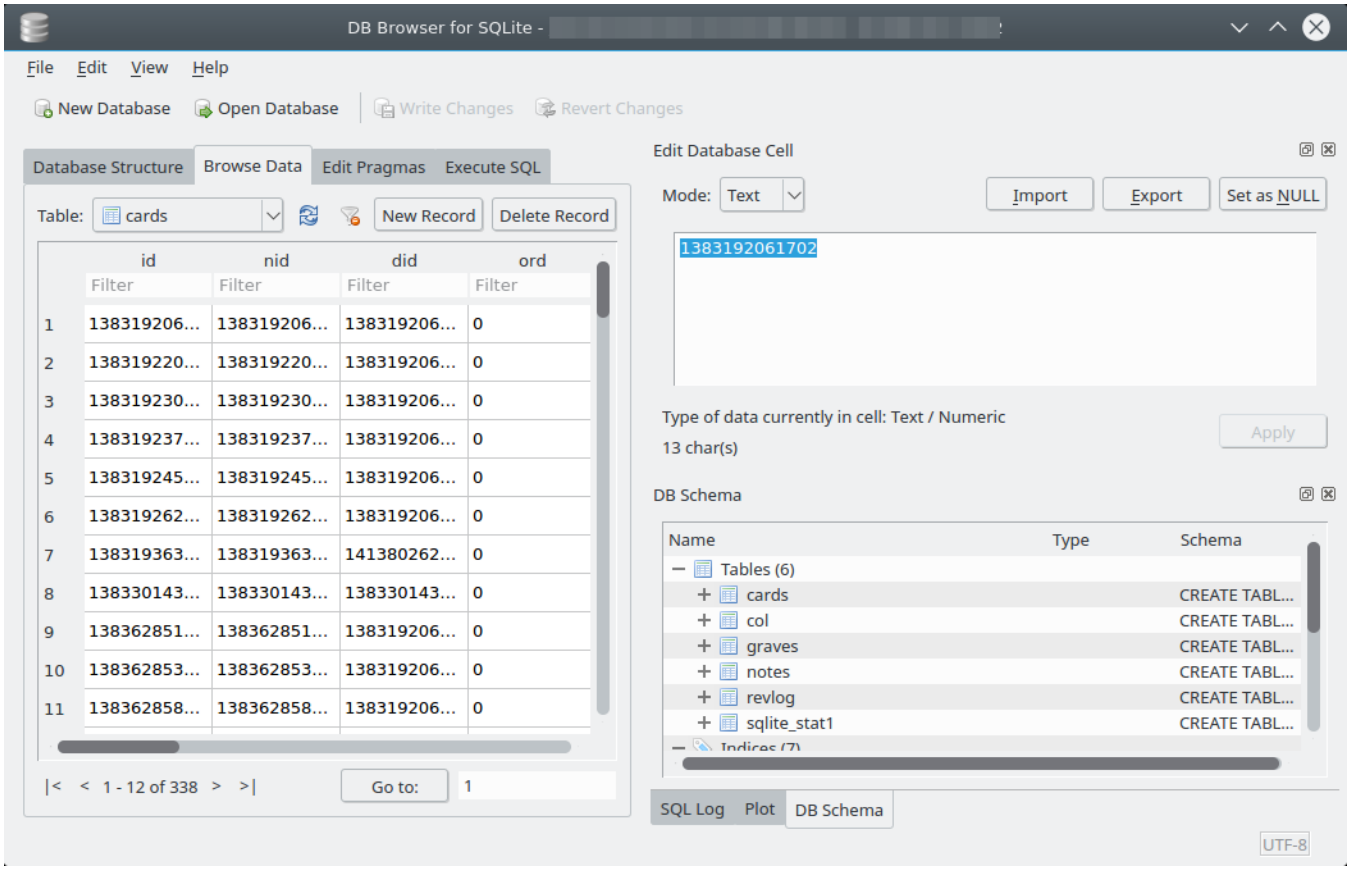


Рисунок 1.2 – Зовнішній вигляд прогарами DB BrowserforSQLite

Засоби управління даної програми дозволяють:

– створювати файли бази даних;

– створювати, визначати, змінювати і видаляти таблиці;

– створювати, визначати і видаляти індекси;

– переглядати, редагувати, додавати і видаляти записи;

– шукати запису;

– імпортувати і експортувати записи як текст;

– імпортувати і експортувати таблиці/у файли CSV;

– імпортувати і експортувати БД як SQL дамп;

– виконувати SQL-запросы і перевіряти результати;

– вивчати журнал усіх SQL команд.

1.3.3 DBeaverCommunityEdition

DBeaverCommunityEdition – це багатоплатформений інструмент БД для розробників, програмістів SQL, адміністраторів БД. З такими базами даних як MySQL, PostgreSQL, SQLite, Oracle, DB2, SQL Server, Sybase, MS Access, Teradata, Firebird, Apache Hive, Phoenix, Presto, etc[7].

DBeaver є комплексним програмним рішенням, орієнтованим на розробників і адміністраторів БД, яким необхідно управляти і організовувати таблиці, тригери, представлення і процедури, що зберігаються, з декількох БД.

Незалежно від бази даних, з якою треба працювати, посилаючись на MySQL, SQLite, PostgreSQL, Oracle, Microsoft SQL Server, IBM DB2 і Firebird, застосування може бути корисне для користувачів, яким необхідно обробляти декілька з'єднань, таким чином організовуючи і редагуючи різні об'єкти БД, включаючи індекси, облікові записи користувачів і скрипти.

Після першого запуску застосування воно автоматично сканує комп'ютер і відображає існуючі з'єднання (якщо вони доступні).

Якщо необхідно створити нове з'єднання, досить вибрати відповідну опцію, вибрати БД, що цікавить, і задати облікові дані для входу (необов'язково, можна вказати драйвери ODBC, якщо вони потрібні).

DBeaver відображає усі ідентифіковані бази даних і користувачів для поточного з'єднання. Нова вкладка під назвою «Проекти» стане доступної на лівій панелі програми, звідки можна легко отримати доступ до об'єкту БД, і подивитися його дані.

Якщо двічі клацнути по певній таблиці, права панель відобразить усю необхідну інформацію, таку як доступні стовпці, зумовлені обмеження і зовнішні ключі, посилання, тригери, індекси і відповідний оператор CREATE DDL.

Враховуючи основну мету, для якої інструмент був розроблений, DBeaver поставляється з редактором SQL, який дозволяє швидко редагувати і виконувати запити. Крім того, можна завантажити існуючий SQL -скрипт, а також перевірити або просто проаналізувати вибраний запит.

Зовнішній вигляд програми DBeaverCommunityEdition показано на рисунку 1.3.

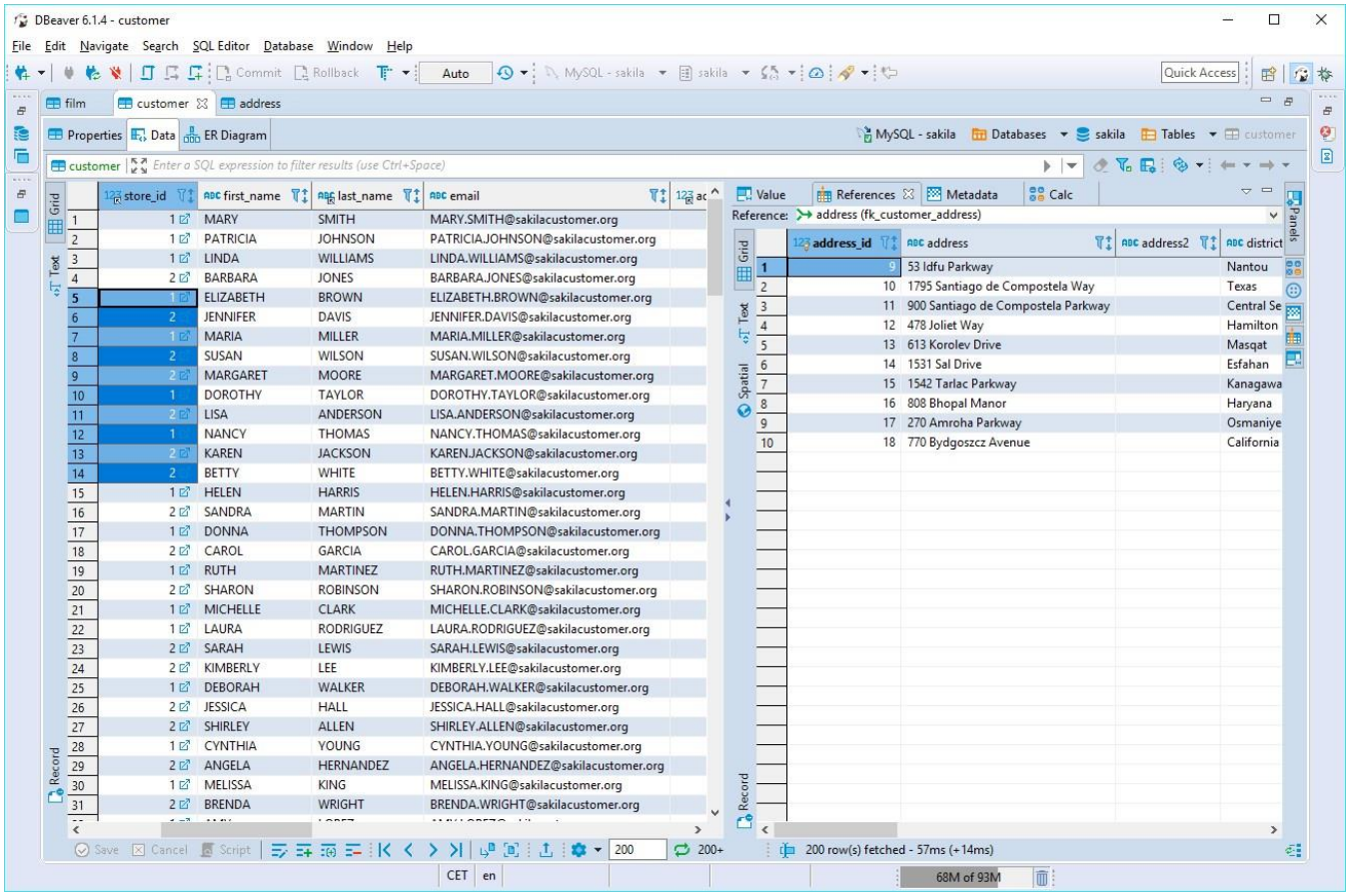


Рисунок 1.3 – Зовнішній вигляд програми DBeaverCommunityEdition

1.3.4 Kexi

Kexi – інструмент для створення візуальних застосувань для БД у візуальному режимі. KEXI Може використовуватися для проєктування баз даних, обробки даних, виконання запитів[8].

Поштовх для розвитку KEXI викликаний помітною відсутністю застосувань, що мають функції Microsoft Access, FoxPro, OracleForms або FileMaker, одночасно віг є потужними, недорогими, орієнтованими на відкриті стандарти і досить мобільним.

KEXI працює під Linux / Unix (FreeBSD, OpenBSD, NetBSD, Solaris) і операційними системами Microsoft Windows. Старі версії були доступні для Mac OS X (з використанням Homebrew). Версія macOS не була випущена, але може бути скомпільована.

Застосування KEXI доступний під LGPL. Документація для користувачів і розробників доступна під GFDL.

Розробка KEXI почалася в 2002 році. OpenOfficeSoftware сприяла цьому значною мірою в період з 2003 по 2008 рік.

Хоча KEXI був проектом KOffice із самого початку, перший випуск не залежав від KOffice. 22 січня 2004 року перша публічна бета-версія KEXI була версією beta 2. Перший стабільний релиз був 0,9 – випущений 31 травня 2005 року. Версія KEXI з нижчим номером версії – 0,8 – була відправлена пізніше з KOffice 1.4 21 червня 2005 року. Інші стабільні версії KEXI були випущені з KOffice 1.5 і 1.6 в 2006 році.

Після випуску KOffice 2.3 KEXI перейшов в CalligraSuite, і обоє були випущені як версія 2.4 11 квітня 2012 року. Найбільш видимою зміною було зміна назви головного призначеного для користувача інтерфейсу, що дістав назву ModernMenu.

Починаючи з версії 3.1 KEXI знову офіційно підтримує Microsoft Windows.

Користувачі KEXI можуть вибирати з різних движків бази даних, які можуть використовуватися для зберігання даних і проектування. За умовчанням використовується вбудований механізм БД SQLite 3, який усуває необхідність в установці і обслуговуванні сервера БД. Для більше професійного використання KEXI може використовувати сервери баз даних, такі як MySQL / MariaDB,

PostgreSQL і Microsoft SQL Server / Sybase. Сервери баз даних мають бути доступні користувачеві на тому ж або будь-якому іншому комп'ютері.

Усі об'єкти БД – таблиці, запити, форми і так далі – зберігаються в таблицях однієї БД (файл або сервер), що дозволяє легко обмінюватися даними і проектами.

Зовнішній вигляд програми KEXI показано на рисунку 1.4.

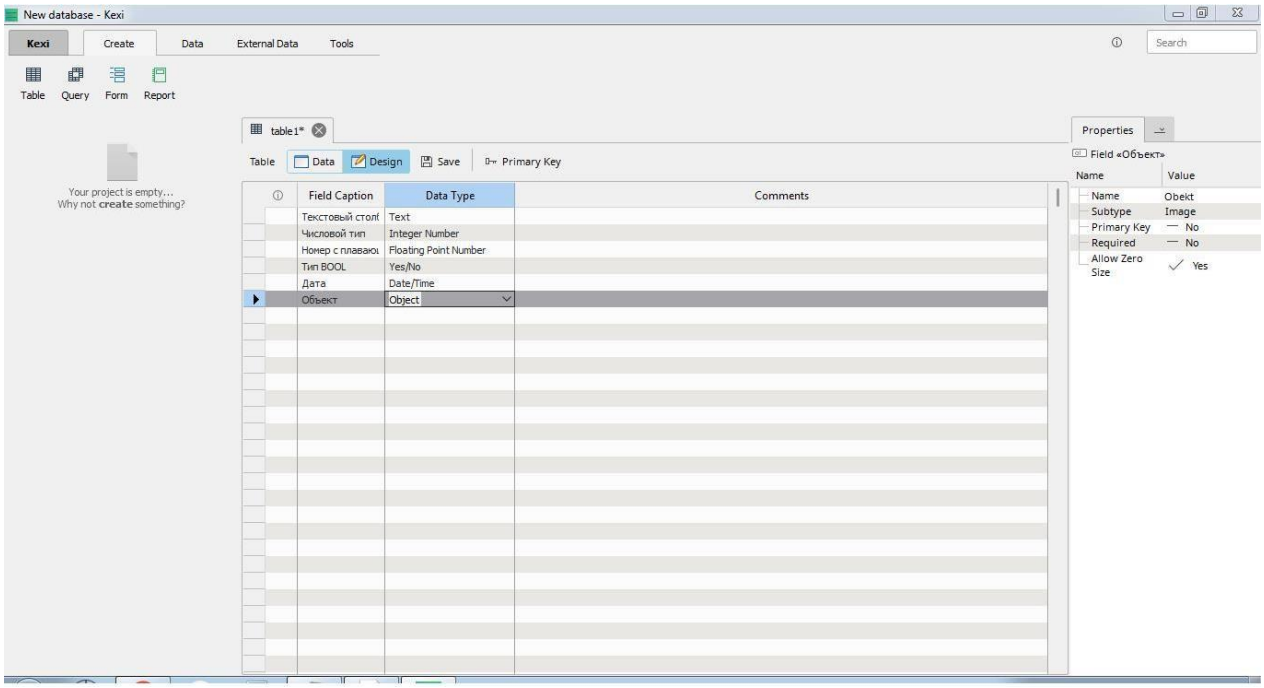


Рисунок 1.4 – Зовнішній вигляд програми KEXI

Дизайнер KEXI Table дозволяє користувачам створювати таблиці, які потім можуть бути створені і відкриті для введення, сортування і пошуку даних. Стандартні типи даних доступні для підтримуваних баз даних.

Запити можуть бути спроектовані візуально або вбудованим спеціалізованим редактором SQL, після чого вони можуть бути виконані. Існує підтримка запитів, що параметризуються, і пошук даних. Об'єм підтримуваного SQL обмежений в порівнянні з необробленими ядрами баз даних (MySQL і навіть SQLite), але діалект SQL, підтримуваний KEXI, є (по дизайну) загальним для усіх підтримуваних движків БД.

Форми можуть бути створені для надання призначеного для користувача інтерфейсу для даних. Існує конструктор і попередній перегляд, здатний до введення даних.

1.4 Постановка задачі

В результаті виконання аналізу завдання на кваліфікаційну роботу бакалавра були розглянуті сучасні програмні інструменти для роботи із БД SQLite.

В результаті проведеного аналізу були відібрані функції, які треба реалізувати в програмі, що розроблюється:

– програма повинна працювати під керуванням операційної системи Windows;

– кожна операція повинна виконуватись з використанням діалогових вікон з відповідним інтерфейсом.

У результаті виконання роботи необхідно створити універсальний інструмент для управління даними за допомогою якого можна будувати будь-яку структуру БД в залежності від потреб користувача. Також програма повинна мати можливість формувати діалогові вікна в залежності від структури даних, що була створена.

2 РОЗРОБЛЕННЯ СТРУКТУРИ ПРОГРАМНОГО МОДУЛЯ, АЛГОРИТМУ ЙОГО РОБОТИ ТА СТРУКТУРИ БАЗИ ДАНИХ

2.1 Опис структури програмного модуля

Програмний модуль, що розроблюється призначений для автоматизації роботи за базою даних SQLite. Згідно з технічним завданням, треба мати можливість працювати з різним набором даних та структурою даних. Аналіз літературних джерел показав, що такий інструмент повинен мати гнучкий алгоритм створення інтерфейсу діалогових вікон, щоб можна було відображати різні дані.

На рисунку 2.1 показана структура програмного модуля, що розробляється.

Структура БД задається у вигляді текстового шаблону в якому описані всі поля і їх назви. За допомогою розробленого програмного майстра оператор має можливість створювати структуру таблиць БД. Структура кожної таблиці зберігається в окремому файлі. На рисунку 2.2 показана структура файлової системи програмного модуля, що розробляється.

Кожна створена таблиця представлена окремим файлом з назвою, яка відповідає назві таблиці.

За допомогою розробленого генератора таблиць в базі даних створюється бажана структура бази даних.

Другий інструмент програми – це таблична форма, яка може відобразити будь-яку інформацію з згенерованих таблиць. Враховуючи, що це учбовий проект функції програми обмежені і не всі типи даних вона може оброблювати. На даний момент програма може працювати з текстовими та числовими полями.

Для управління даними в структурі програми передбачені діалогові вікна. За допомогою них користувач може додавати, або редагувати дані в таблицях.

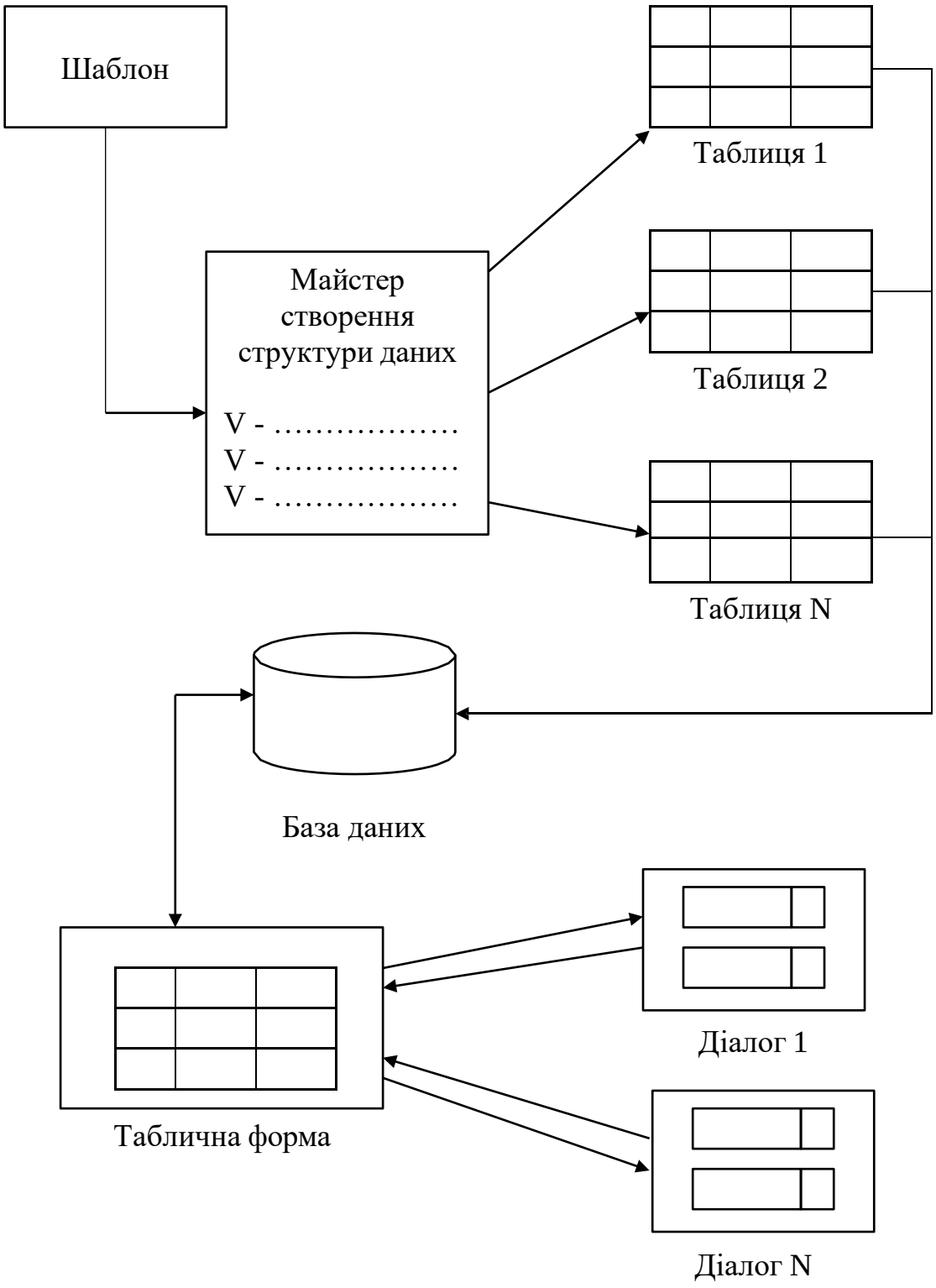


Рисунок 2.1 – Структура програмного модуля

Кожна форма створюється в процесі роботи, тобто на початку завантаження вона є пустою. Після завантаження структури даних, що необхідно відобразити, в реальному часі створюються необхідні компоненти.

Таким чином реалізована вимога створення універсального погромного засобу для роботи з різною структурою даних.

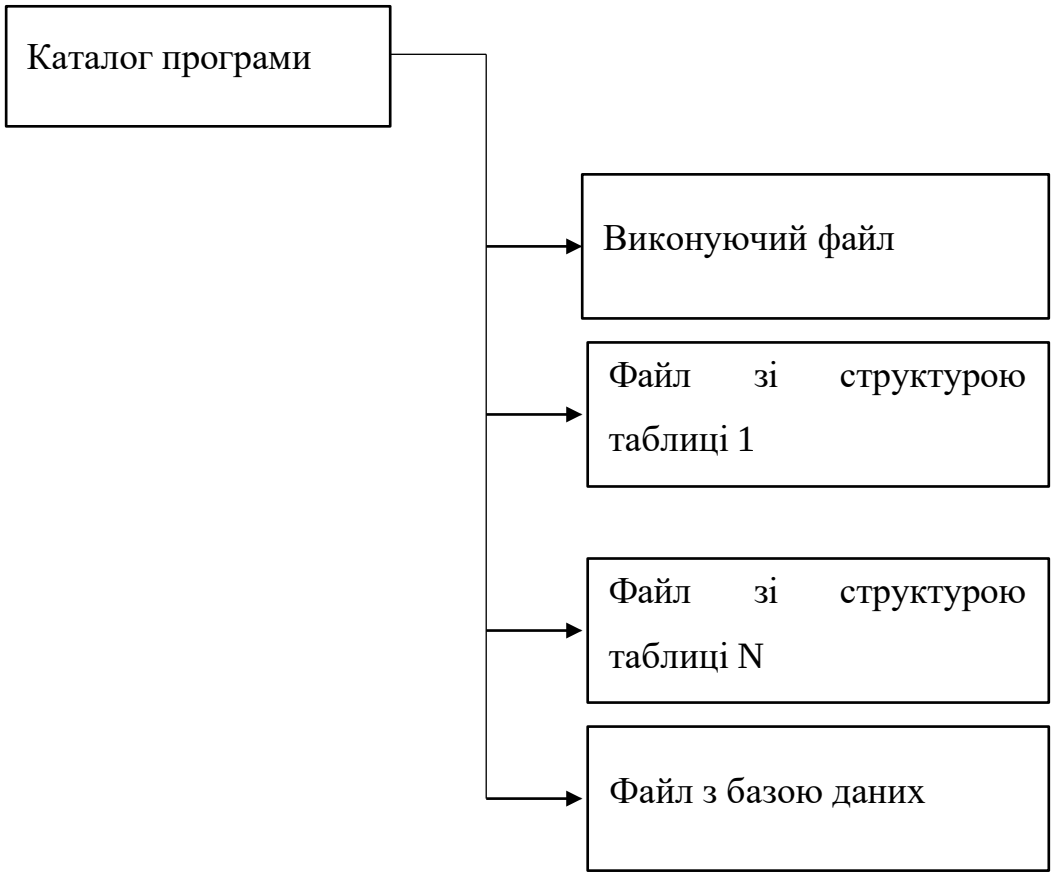


Рисунок 2.2 – Структура файлової системи програмного модуля

В каталозі з головною програмою знаходиться файл з БД. Після першого завантаження він пустий. Після того, як оператор створить структуру першої таблиці, в каталозі з’являються новий файл з назвою таблиці та розширенням “*.sdb”.

Для кожної нової таблиці створюється окремий файл з назвою відповідної таблиці.

На рисунку 2.3 наведено алгоритм роботи з програмою в режимі створення структури бази даних.

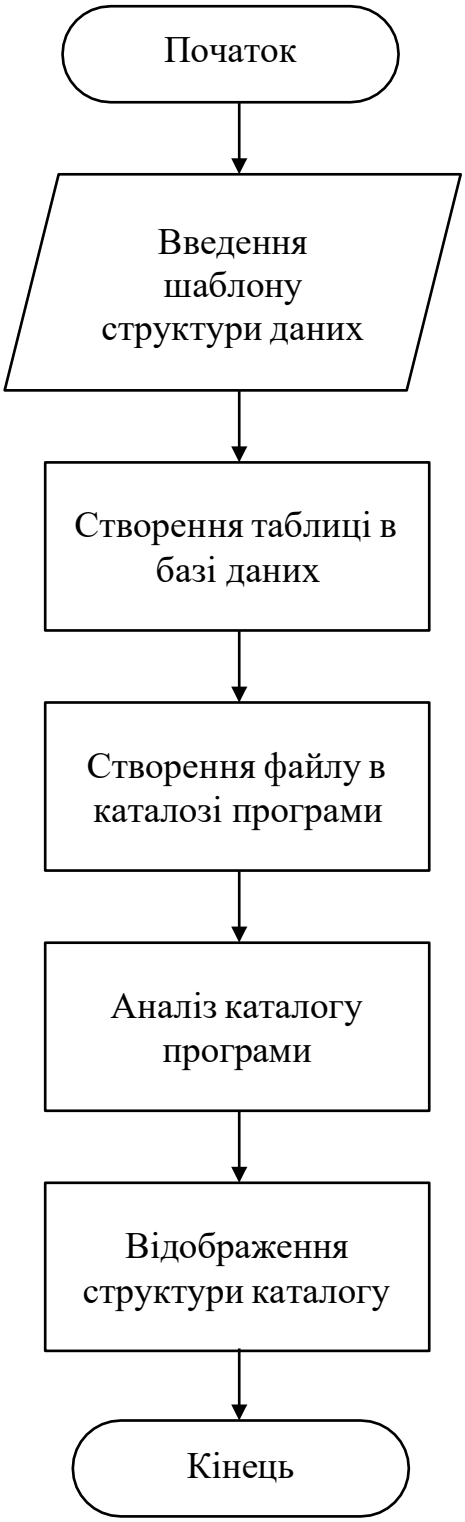


Рисунок 2.3 – Алгоритм роботи з програмою в режимі створення структури бази даних

Після створення структури бази, формується інтерфейс відображення табличної інформації. На рискну 2.4 показано алгоритм роботи з програмою в режимі інтерфейс відображення табличної інформації.

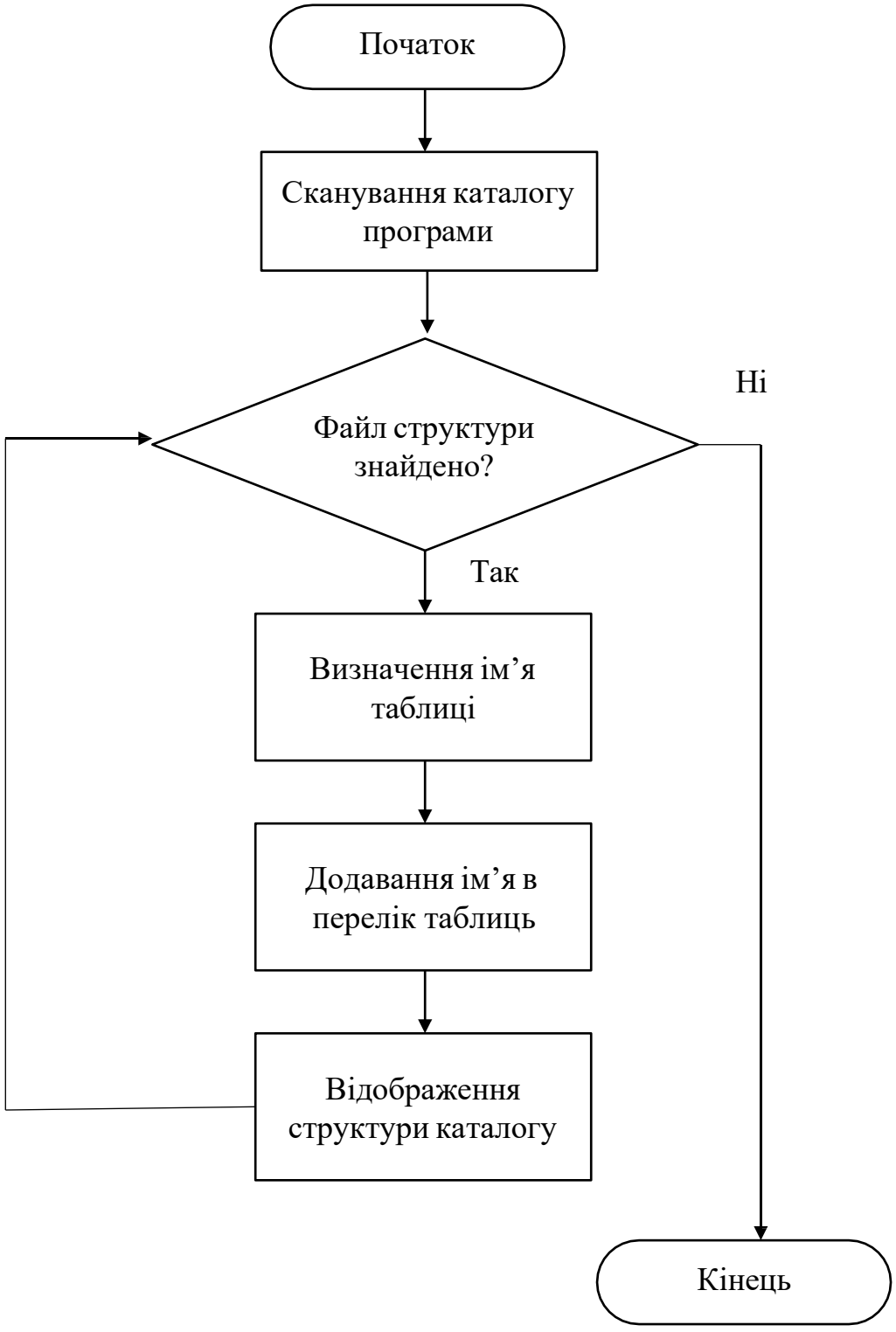


Рисунок 2.4 – Алгоритм роботи з програмою в режимі інтерфейс відображення табличної інформації

Як можна бачити з наведеного алгоритму, спочатку виконується сканування робочого каталогу на предмет знаходження там файлів з відповідним розширенням. За результатами сканування будується список файлів.

Далі запускається процес перевірки списку. Якщо знайдено ім’я файлу то воно додається до списку на екрані програми. Цей процес продовжується до того моменту, поки всі знайдені файли не будуть додані до списку.

На рисунку 2.5 показано алгоритм формування діалогового вікна.

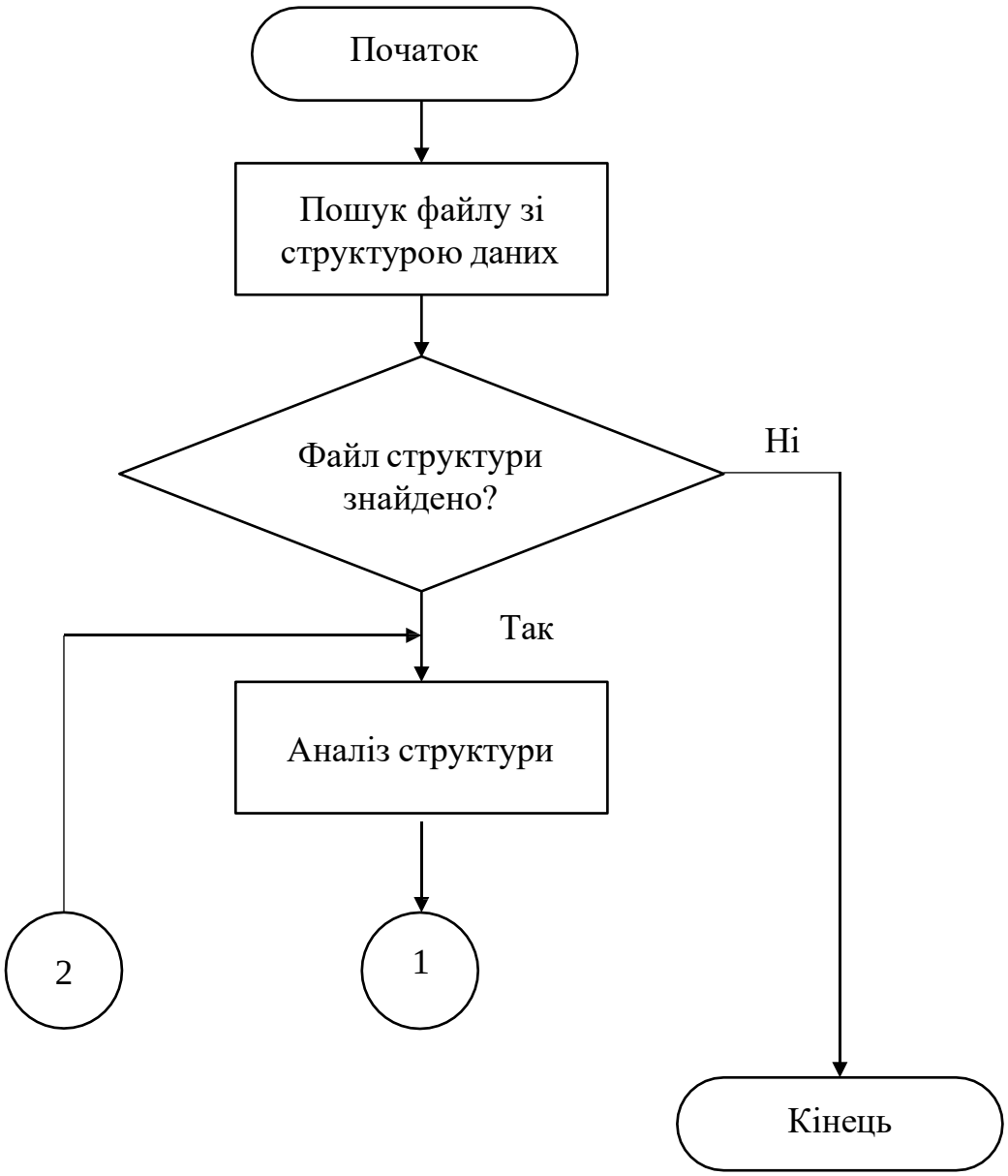


Рисунок 2.5 – Алгоритм формування діалогового вікна

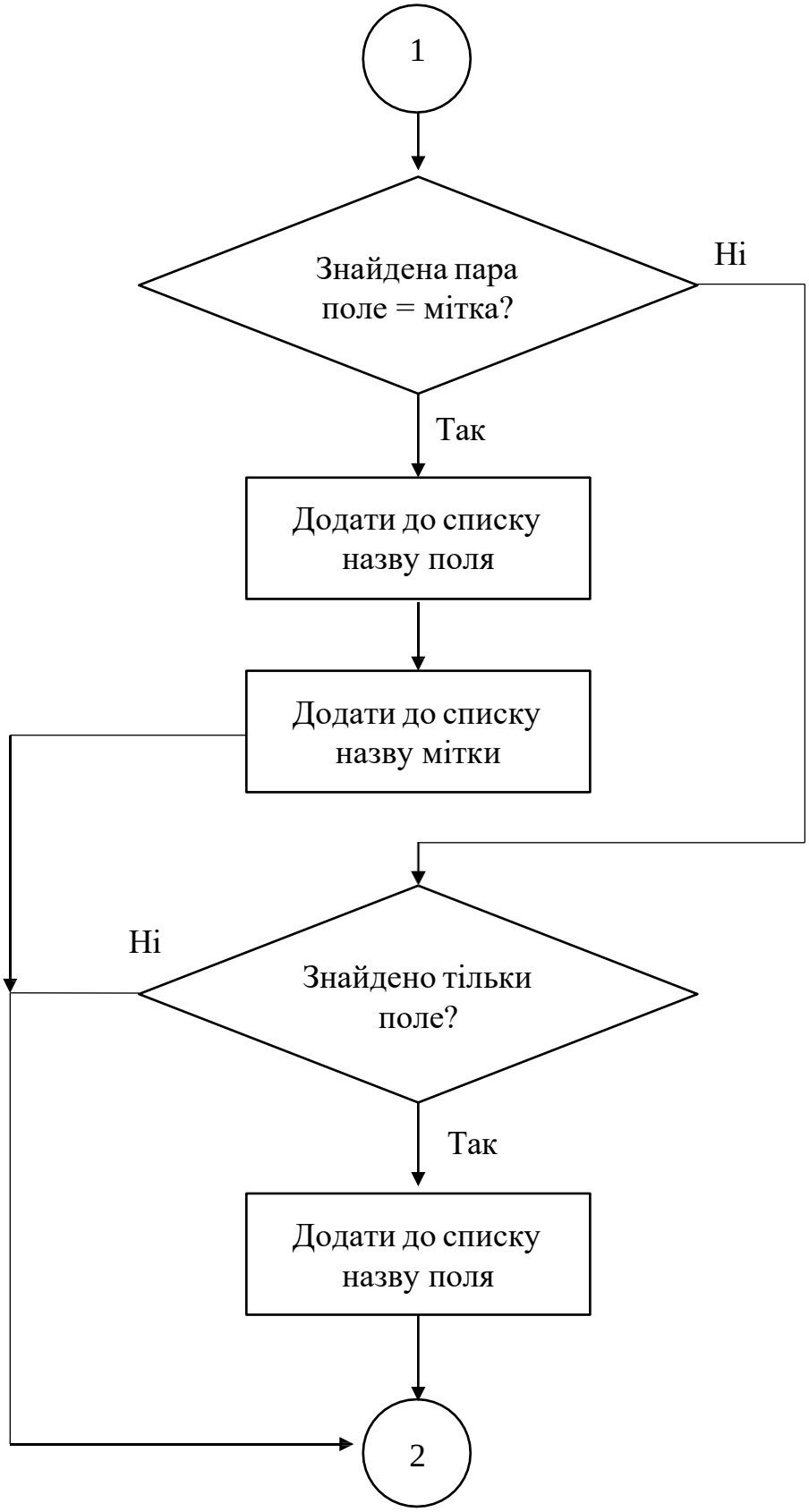


Рисунок 2.5, аркуш 2

Як можна бачити з наведеного алгоритму, після того, як файл зі структурою буде знайдено запускається підпрограма її аналізу.

Аналіз полягає в пошуку стрічки формату «поле = мітка». Поле, це назва поле даних, а мітка – це назва цього поля в інтерфейсі програми.

Якщо така пара буде знайдена, то вона ділиться на складові, та до відповідних списків додаються назва поля, та назва мітки. Далі ці списки будуть використовуватись для створення візуальних компонентів на формі діалогового вікна в режимі реального часу.

Якщо буде знайдена тільки назва поля, то це означає, що це ключове поле. Його відображати на екрані не потрібно, тому що дані такому полі генеруються автоматично.

Якщо всі строки файлу були оброблені, то після закінчення процедури аналізу на екрані формуються візуальні компоненти за визначеним списком. Мітці відповідає компонент Label, а полю – TextBox. В властивістьTag компоненту TextBox зберігається назва поля. Вона використовується при натисканні кнопки «ОК» та запуску процедури зберігання інформації.

2.2 Опис параметрів вхідної інформації

Розглянем приклад використання програми для створення структури бази даних зберігання інформації про замовлення виготовлення виробів на підприємстві.

Проаналізувавши задачу були створені такі інформаційні об’єкти:

– замовники;

– замовлення;

– вироби;

– категорії виробів.

Для того, щоб мати можливість зберігати інформацію про ці об'єкти у БД, проаналізуємо атрибути, які відповідають кожному об'єкту.

Об'єкт «Замовник» можна описати за допомогою наступних атрибутів:

– найменування організації (прізвище, ім'я і по батькові замовника);

– адреса підприємства (проживання замовника);

– контактний телефон;

– реєстраційний номер (номер паспорта);

– дата реєстрації.

Об'єкт «Замовлення» можна описати за допомогою наступних атрибутів:

– дата замовлення;

– дані про замовника;

– найменування замовлення;

– дані про вироби, які входять в це замовлення;

– кількість виробів;

– вартість замовлення.

Об'єкт «Виріб» можна описати за допомогою наступних атрибутів:

– найменування виробу;

– коротка характеристика виробу;

– посилання на категорію виробу для реалізації можливості розподілу пристроїв по теках.

Об'єкт «Категорія» можна описати за допомогою наступних атрибутів:

– найменування;

– посилання на категорію для реалізації можливості створення підкатегорій.

2.3 Розробка структури бази даних

Виходячи з розглянутого набору вхідних даних можемо розробити структуру бази даних.

В таблиці 2.1 наведена структура таблиці «Замовник».

Таблиця 2.1 – Структура таблиці «Замовник»

№ з/п	Найменування	Тип	Розмір	Індекс
1	ID_Zamovnyk	Integer		Первинний
2	Name	Text	32	
3	SName	Text	32	
4	FName	Text	32	
5	Address	Text	32	
6	PnoneNumber	Text	32	
7	Passport	Text	256	
8	DateReestr	Text	32	

Розглянемо структуру таблиці «Замовник»:

– ID_Zamovnyk – первинний ключ таблиці. Використовуються для посилання на замовника з боку інших таблиць;

– Name– ім’я замовника;

– SName– прізвища замовника;

– FName– по батькові замовника;

– Address – адреса проживання, або роботи замовника;

– PnoneNumber– контактний телефон замовника;

– Passport – паспортні дані замовника;

– DateReestr – дата реєстрації паспорта замовника.

Всі поля окрім ключового текстові.

В таблиці 2.2 наведена структура таблиці «Замовлення».

Таблиця 2.2 – Структура таблиці «Замовлення»

№ з/п	Найменування	Тип	Розмір	Індекс
1	ID_Zamovlennja	Integer		Первинний
2	ID_Zamovnyk	Integer		Зовнішній

Продовження таблиці 2.2

№ з/п	Найменування	Тип	Розмір	Індекс
3	Name	Text	32	
4	Zakaz	Text	256	
5	Kilkist	Text	32	
6	Price	Text	32	

Розглянемо структуру таблиці «Замовник»:

– ID_Zamovlennja – первинний ключ таблиці. Використовуються для посилання на замовлення з боку інших таблиць;

– ID_Zamovnuk – посилання на таблицю замовника для відображення повних даних. Це зовнішній ключ то тип даних Integer;

– Name – найменування заказу;

– Zakaz – тіло заказу. Тут приводиться перелік товарів або виробів, які замовляє замовник;

– Kilkist – кількість одиниць виробів в заказі;

– Price – ціна заказу в грн.

В таблиці 2.3 наведена структура таблиці «Виріб».

Таблиця 2.3 – Структура таблиці «Виріб»

№ з/п	Найменування	Тип	Розмір	Індекс
1	ID_Vyrib	Integer		Первинний
2	ID_Kategor	Integer		Зовнішній
3	Name	Text	32	
4	Notice	Text	256	

Розглянемо структуру таблиці «Виріб»:

– ID_Vyrib– первинний ключ таблиці. Використовуються для посилання на вироби з боку інших таблиць;

– ID_Kategor – посилання на таблицю з категоріями виробів для відображення даних про них;

– Name – назва виробу;

– Notice – короткий опис виробу.

В таблиці 2.4 наведена структура таблиці «Категорії».

Таблиця 2.4 – Структура сутності «Категорії»

№ з/п	Найменування	Тип	Розмір	Індекс
1	ID_Kategor	Integer		Первинний
2	Name_Kategor	Text	32	
3	Notice	Text	256	

Поле ID_Kategorвикористовуються в якості первинного ключа та необхідне для посилання на користувача з боку інших таблиць.

Поле Name_Kategorвикористовуються для зберігання назви категорії.

Поле Noticeвикористовуються для зберігання стислого опису категорії.

Отже, у результаті виконання другого розділу кваліфікаційної роботи була описана структура програмного модуля, що розроблюється, наведена структура файлової системи для зберігання опису таблиць. Розроблено алгоритми роботи програми, описані параметри вхідної інформації та розроблена структура БД.

3 РОЗРОБЛЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Опис інструментів розроблення програмного забезпечення

Для розробки ПЗ використовувались такі інструменти:

– інтегроване середовище розробки Visual Studio 2019 [9];

– інструмент для роботи з пакетами допоміжних програм NuGet [10];

– менеджер бази даних SQLite Expert Professional [11].

Visual Studio – середовище розробника програмного забезпечення, яке підтримує декілька мов програмування: C#, VisualBaisic, F#, ASP.NET, Java, JavaScript, Phyton та інші. Даний інструмент пропонує візуальні компоненти для створення графічного інтерфейсу користувач. За допомогою VisualStudio можна розробляти:

– програмне забезпечення для комп'ютера під керуванням операційної системи Windows;

– мобільні додатки (Windows, iOS, Android);

– Web-додатки;

– хмарні додатки;

– різні розширення для Office, SharePoint, а також створення власних розширень для VisualStudio;

– БД SQL Server і SQL Azure.

Для розробки даного програмного модуля використовується професійна версія IDE. VisualStudio Professional – містить професійні інструменти для розробки програм різної складності.

Важливим інструментом будь-якої сучасної платформи розробки є механізм, за допомогою якого розробники можуть створювати, ділитися та використовувати вихідний код. Часто такий код поєднується в "пакети", що містять складений код (як DLL) разом з іншим вмістом, необхідним в проектах, які споживають ці пакети.

Для .NET (включаючи .NET Core) механізмом спільного використання коду є NuGet, який визначає, як пакети для .NET створюються, розміщуються та використовуються, і надає інструменти для кожної з цих ролей.

NuGet – це єдиний ZIP-файл з розширенням .nupkg, який містить скомпільований код (DLL), інші файли, пов’язані з цим кодом, та описані в маніфесті, що включає інформацію, наприклад номер версії пакета. Розробники з кодом для спільного використання створюють пакети та публікують їх на публічному або приватному хості. Споживачі пакетів отримують ці пакети від відповідних постачальників, додають їх до своїх проектів, а потім викликають функціональність програм у коді проекту.

Оскільки NuGet підтримує приватні сховища поряд із загально доступним сховищами nuget.org, можно використовувати пакети NuGet для обміну кодом, який є ексклюзивним для організації або робочої групи. Також можна використовувати пакети NuGet як зручний спосіб зберігання власного коду для використання лише у власних проектах.

3.2 Опис інтерфейсу користувача та програмних функцій

Інтерфейс користувача розроблений за допомогою візуальних компонентів VisualStudio і наведено на рисунку 3.1.

Кнопки 1 та 2 перемикають режими роботи програми. При натисканні на кнопку 1 вмикається режим створення структури БД. Користувачу відкривається перша вкладка (рисунок 3.1) на якій знаходяться компоненти для опису таблиць бази даних та редагування їх структури.

Приклад програмної функції для включення першої вкладки показано на рисунку 3.2.

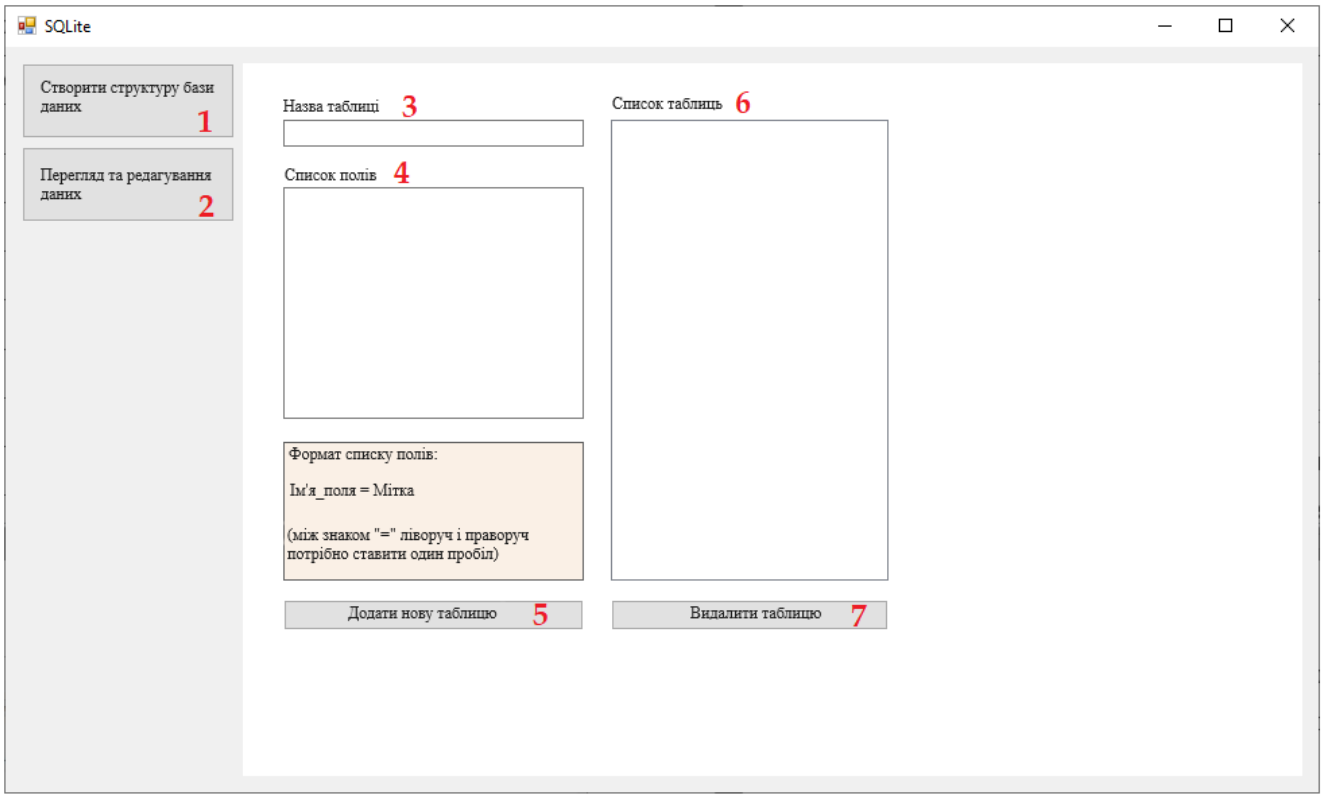


Рисунок 3.1 – Приклад розробленого інтерфейсу користувача

```
ссылка: 1
private void button2_Click(object sender, EventArgs e)
{
    tablessControl1.SelectedTab = tabPage1;
    updateListDB();
    textBox1.Text = "";
    textBox2.Text = "";
}
```

Рисунок 3.2 – Приклад програмної функції для включення першої вкладки

Розглянемо основні поля інтерфейсу користувача в режимі створення структури БД. Поле 3 призначене для введення назви таблиці. Це поле зроблено на основі компоненту textbox. Даний компонент дозволяє оперувати строчкою тексту. Все, що вводить користувач, може бути збережено при виконанні відповідної функції в програмі.

Поле 4 призначене для опису полів таблиці. Воно теж зроблено на основі компоненту textbox, але в ньому включено режим «MultiLine», як показано на рисунку 3.3.

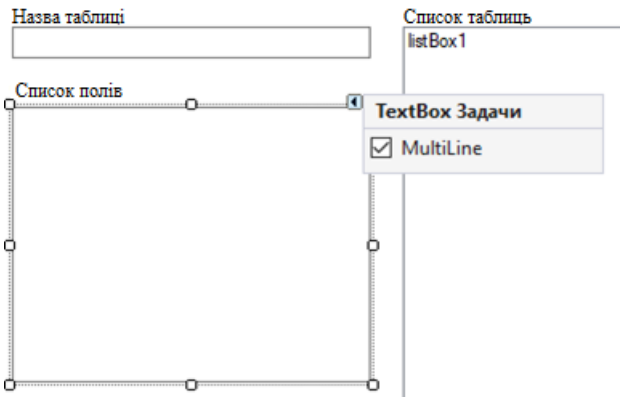


Рисунок 3.3 – Перетворення компоненту textbox
в простий редактор тексту

При обиранні даного режиму компонент перетворюється в простий текстовий редактор. В ньому можна редагувати текст, додавати нові рядки, видаляти та вставляти потрібний фрагмент тексту.

Відповідне поле в інтерфейсі програми – це підказка для оператора, щоб він правильно описував структуру таблиці. Формат списку полів таблиці такий:

Ім’яполя = Метка.

Між знаком "=" зліва та справа потрібно ставити один знак пробілу.

При натисканні на кнопку 5 викликається функція зберігання введених даних, та створення структури бази даних.

Текст функції наведено на рисунку 3.4. Для опису кожної таблиці на диску створюється відповідний файл зі структурою даних. Файл має розширення «.sdb». Далі файл використовується для програмного створення інтерфейсу діалогу, який відкривається користувачеві для введення даних.

```
private void button4_Click(object sender, EventArgs e)
{
    if (textBox1.Text != "")
    {
        System.IO.File.WriteAllText(mPath + textBox1.Text + ".sdb" , textBox2.Text);

        string sql = "Create table " + textBox1.Text + "(";
        string text = System.IO.File.ReadAllText(String.Format("{0}.sdb", mPath + textBox1.Text));
        string[] stringSeparators = new string[] { "\r\n" };
        string[] ravnoSeparators = new string[] { " = " };
        string[] field = text.Split(stringSeparators, StringSplitOptions.None);
        foreach (String s in field)
        {
            if (s != "") //пропускаем пустые строки
            {
                if (s.Contains("=")) //отделяем поля без названия
                {
                    string[] pair = s.Split(ravnoSeparators, StringSplitOptions.None);
                    if (pair.Length == 2)
                    {
                        sql = sql + String.Format(", {0} TEXT", pair[0]);
                    }
                }
                else
                {
                    sql = sql + String.Format("{0} integer PRIMARY KEY AUTOINCREMENT NOT NULL", s);
                }
            }
        }
        sql = sql + String.Format(");");
        DataBase.Exec_SQL(sql);
    }
    updateListDB();
    textBox1.Text = "";
    textBox2.Text = "";
}
```

Рисунок 3.4 – Функція зберігання введених даних, та створення структури бази даних

Поле 6 – це список вже створених таблиць, структура яких збережена на диску. Список оновлюється декілька раз в програмі. Перший раз – коли користувач відкриває вкладку «Створення структури БД», інші рази – кожен раз, коли створюється нова таблиця та зберігається її структура.

На рисунку 3.5 показана функція оновлення списку таблиць. З даного рисунку можна бачити, як відбувається читання файлів в відповідному каталозі, та відбираються тільки ті, які мають розширення «.sdb». Всі знайдені файли відображаються у вигляді списку за допомогою компоненту ListBox [10].

```
Ссылок: 4
private void updateListDB()
{
    List<string> filesDir = (from a in Directory.GetFiles(mPath, "*.sdb")
                             select Path.GetFileName(a)).ToList();
    listBox1.Items.Clear();
    foreach (string s in filesDir)
    {
        listBox1.Items.Add(s);
    }
}
```

Рисунок 3.5 – Функція оновлення списку таблиць

За допомогою кнопки 7 можна видили таблицю, ім'я якої виділено в компоненті ListBox. Таблиця видаляється, як з диску, так і зі структури бази даних.

На рисунку 3.6 показано інтерфейс вкладки 2, яка відкривається при обиранні режиму роботи «Перегляд та редагування даних».

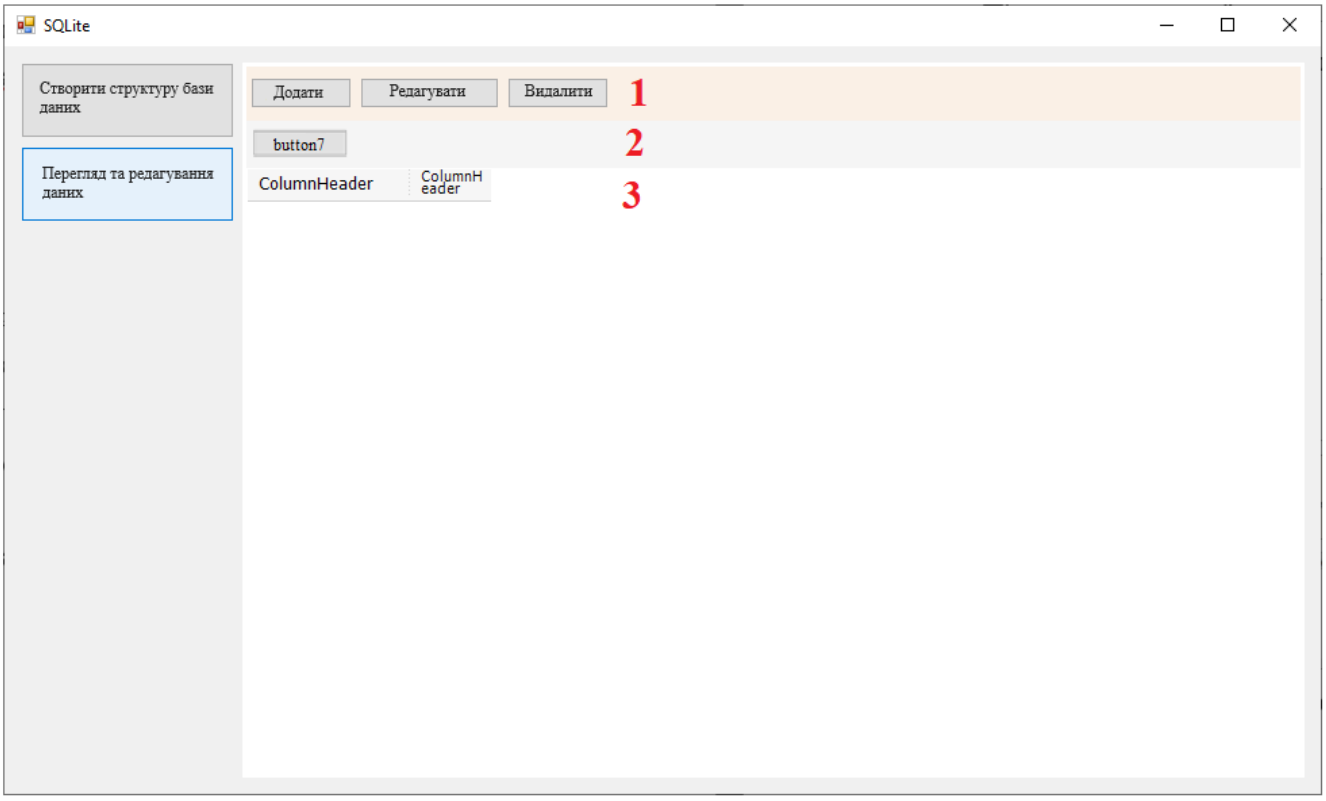


Рисунок 3.6 – Перегляд та редагування даних

Даний інтерфейс має три поля:

– поле 1 – кнопки керування даними;

– поле 2 – кнопки перемикання між таблицями;

– поле 3 – відображення структури даних.

Поле 1 містить три кнопки:

– «Додати»;

– «Редагувати»;

– «Видалити».

При натисканні на кнопку «Додати» викликається функція, текст якої показано на рисунку 3.7.

```
private void button6_Click(object sender, EventArgs e)
{
    DataDialog dd = new DataDialog();
    dd.dataName = selectTable;
    DialogResult res = dd.ShowDialog();
    if (res == DialogResult.Cancel) return;
    FillTable();
}
```

Рисунок 3.7 – Функція додавання даних

Як можна бачити, функція викликає діалог для введення даних в таблицю. Приклад діалогового вікна показано на рисунку 3.8.

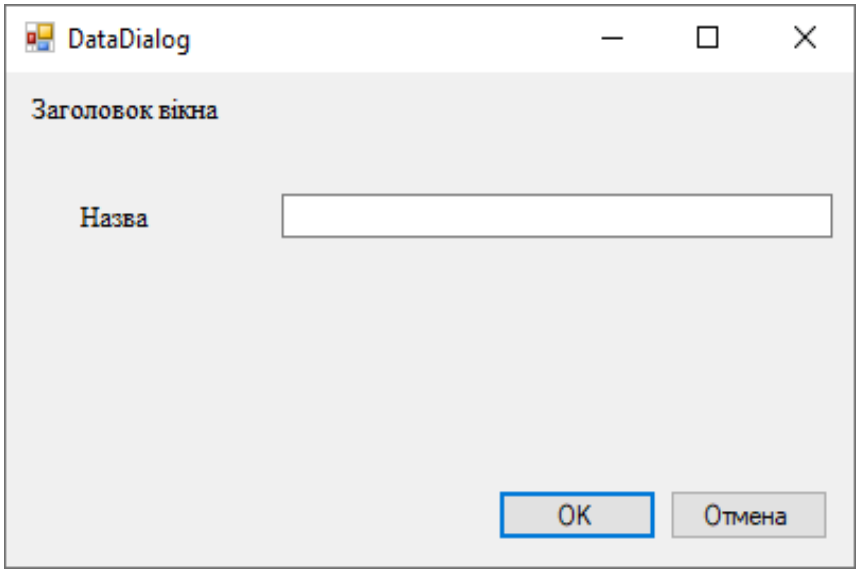


Рисунок 3.8 – Приклад вікна додавання даних в таблицю

Дане вікно формується програмно. Кількість полів вводу даних залежить від структури обраної таблиці.

Всі поля формуються динамічно за допомогою функції generateField, текст якої показано на рисунку 3.9.

```
ссылка: 1
public void geerateField()
{
    int x = 10;
    int y = 10;
    foreach (fieldMembers item in fieldName)
    {
        Label label = new Label();
        label.AutoSize = true;
        label.Left = x;
        label.Top = y;
        label.Text = item.Label;
        label.Parent = panel3;

        if (item.Name.Contains("ID"))
        {

        }
        else
        {
            TextBox tb = new TextBox();
            tb.Left = x + 120;
            tb.Top = y;
            tb.Tag = item.Name;
            tb.Width = 260;
            tb.Parent = panel3;
        }
        y = y + 24;
    }
    if (y + 24 > panel3.Height)
    {
        panel3.Height = y + 24;
    }
}
```

Рисунок 3.9 – Функція generateField

Кнопки в полі 2 також формуються програмно. Їх кількість залежить від кількості таблиць.

На рисунку 3.10 показано текст функції формування поля 2 на екрані.

```
ссылка: 1
private void LoadDataTableList()
{
    panel2.Controls.Clear();
    List<string> filesDir = (from a in Directory.GetFiles(mPath, "*.sdb") select Path.GetFileName(a)).ToList();
    listBox1.Items.Clear();
    int x = 10;
    int y = 5;
    foreach (string s in filesDir)
    {
        String name = Path.GetFileNameWithoutExtension(s);
        Button b = new Button();
        b.Top = y;
        b.Left = x;
        b.Text = name;
        b.AutoSize = true;
        b.Click += new EventHandler(button_Click);
        b.Parent = panel2;
        x = x + b.Width + 4;
    }
}
```

Рисунок 3.10 – Текст функції формування поля 2 на екрані

Поле 3 – це таблиці з даними. Вона заповнюється за допомогою функції FillTable (рисунок 3.11).

```
private void FillTable()
{
    Control_ViewTable cv = control_ViewTable1;
    string sql = String.Format("Select * from {0};", selectTable);
    cv.tableStructure.Clear();

    string text = System.IO.File.ReadAllText(String.Format("{0}.sdb", selectTable));
    string[] stringSeparators = new string[] { "\r\n" };
    string[] ravnoSeparators = new string[] { " = " };
    string[] field = text.Split(stringSeparators, StringSplitOptions.None);

    //разделяем на поля и названия
    int count = 0;
    foreach (String s in field)
    {
        if (s != "") //пропускаем пустые строки
        {
            if (s.Contains("=")) //отделяем поля без названия
            {
                string[] pair = s.Split(ravnoSeparators, StringSplitOptions.None);
                if (pair.Length == 2)
                {
                    if (count == 0) cv.tableStructure.Add(new Control_ViewTable.TableStructure(pair[0], "string", pair[1], 200, HorizontalAlignment.Left, 0, true));
                    else cv.tableStructure.Add(new Control_ViewTable.TableStructure(pair[0], "string", pair[1], 200, HorizontalAlignment.Left, 0, false));
                    count++;
                }
            }
            else
            {
                cv.ID_Field_Name = s;
            }
        }
    }

    cv.createStructure();

    cv.sql_d = sql;
    cv.FillDataList();
}
```

Рисунок 3.11 – Функція FillTable для заповнення поля 3

4 ПРАКТИЧНА РЕАЛІЗАЦІЯ ТА ВИПРОБУВАННЯ РОЗРОБЛЕНОГО ПРОГРАМНОГО МОДУЛЯ

4.1 Вимоги до апаратного забезпечення

Мінімальні вимоги до обладнання:

– процесор з тактовою частотою не нижче 1,8 ГГц. Рекомендується використовувати як мінімум двоядерний процесор;

– 2 ГБ оперативної пам'яті, рекомендується 8 ГБ (якщо встановлювати на віртуальну машину, то мінімум 2,5 ГБ);

– вільного місця на жорсткому диску повинно бути від 800 мегабайт до 210 гігабайт, залежно від встановлених компонентів. У більшості випадків потрібно як мінімум 30 гігабайт, наприклад, якщо встановлюється всього кілька компонентів, то це займає більше 20 ГБ місця на жорсткому диску. Також Microsoft рекомендує встановлювати VisualStudio на SSD диск;

– відеоадаптер з мінімальним дозволом 1280 на 720 пікселів (для оптимальної роботи VisualStudio рекомендується роздільна здатність 1366 на 768 пікселів і вище).

4.2 Опис тестового прикладу

Виконаємо тестовий приклад створення структури БД для зберігання інформації про замовлення виготовлення виробів на підприємстві. Структура бази даних була розроблена та описана в підрозділі 2.2.

Спочатку створюємо порожню БД за допомогою інструменту SQLite Expert.

На рисунку 4.1 показано приклад створення бази даних.

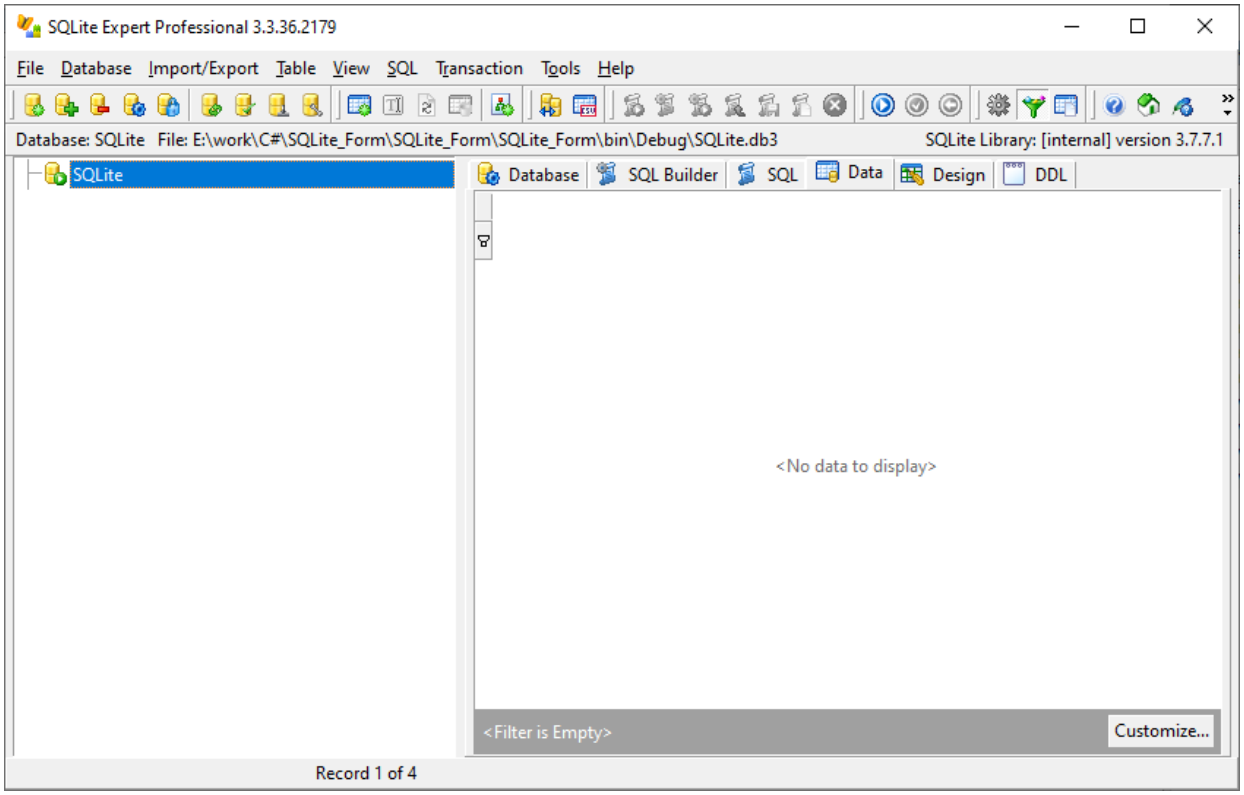


Рисунок 4.1 – Створення бази даних

Після запуску нашої програми ми бачимо порожній список доступних таблиць (рисунок 4.2) та порожній список кнопок обирання таблиці для перегляду (рисунок 4.3).

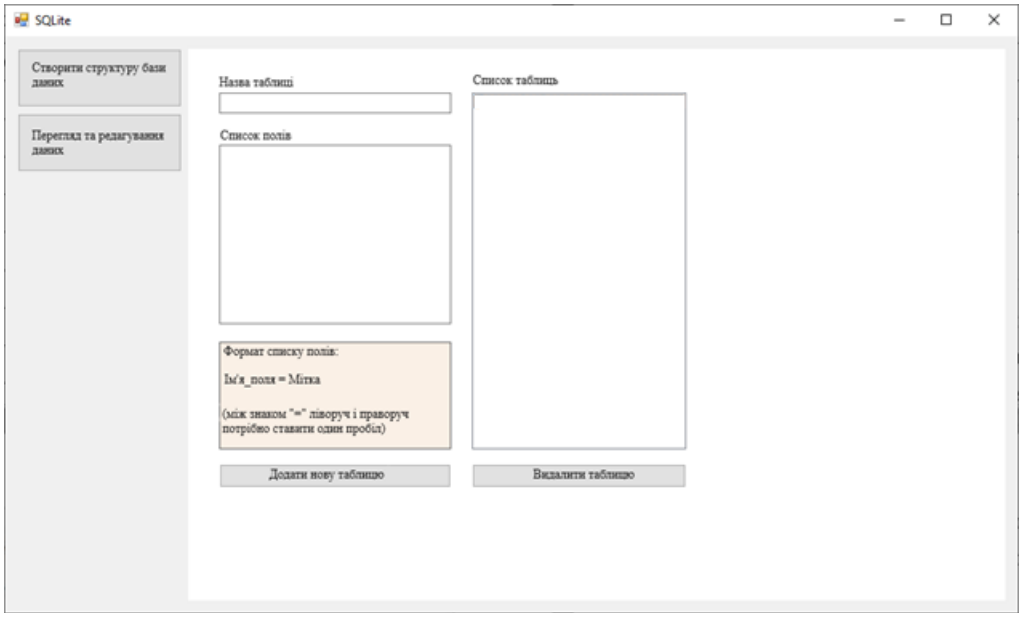


Рисунок 4.2 – Порожній список доступних таблиць

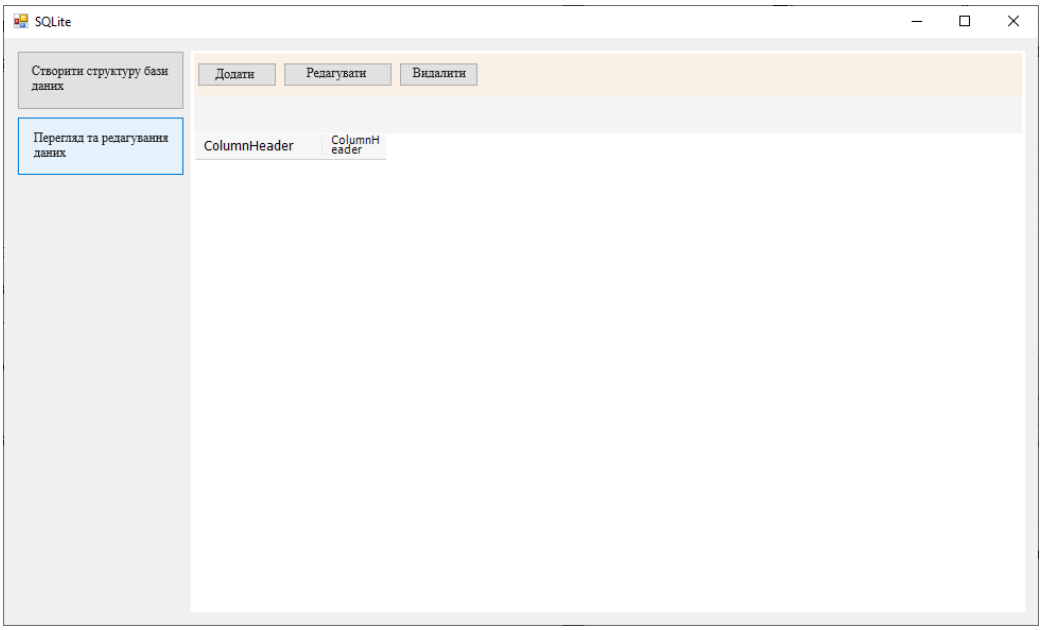


Рисунок 4.3 – порожній список кнопок обирання таблиці для перегляду

Першою створюємо таблицю «Замовники» згідно зі структурою, що наведена в таблиці 2.1.

Приклад створення структури таблиці показано на рисунку 4.4.

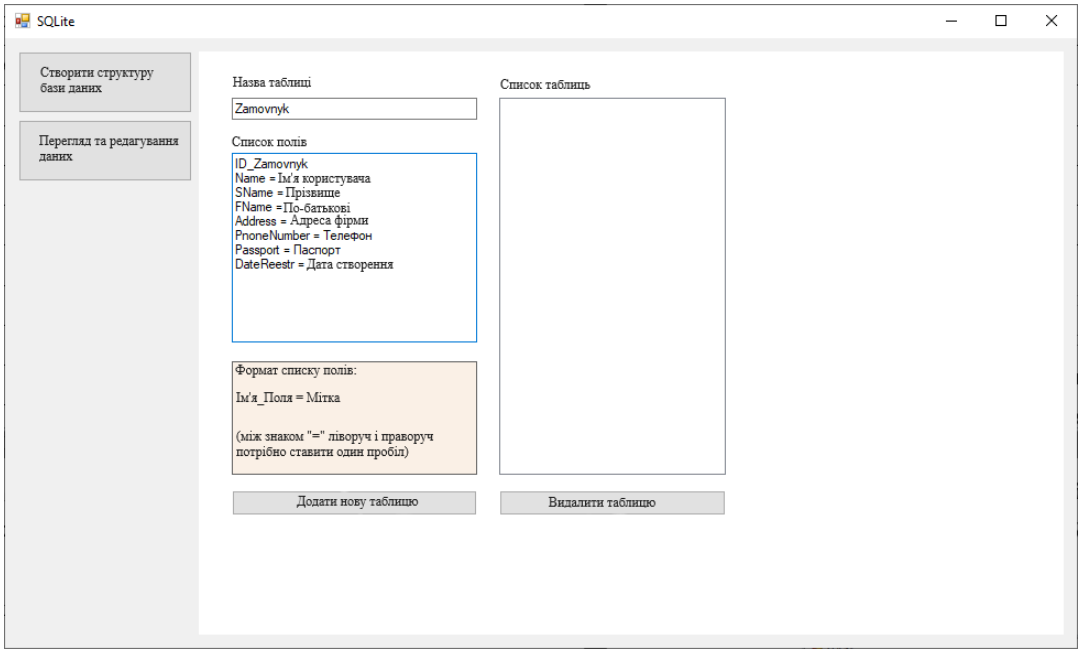


Рисунок 4.4 – Приклад створення структури таблиці «Замовники»

Кожну назву поля записуємо з нового рядку. Навпроти назви поля вказуємо назву, яка буде відображатися в діалоговому вікні. Назви розділяються знаком «=».

Поле «ID_Zamovnyk» не має назви, тому що воно не буде відображатись в діалоговому вікні. Це ключове поле та інформація в ньому не вводиться користувачем, а заноситься автоматично при додаванні кожного нового рідку до таблиці.

Після натискання на кнопку «Додати нову таблицю» створюється новий файл на диску та нова таблиця відображається в відповідному списку, як показано на рисунку 4.5.

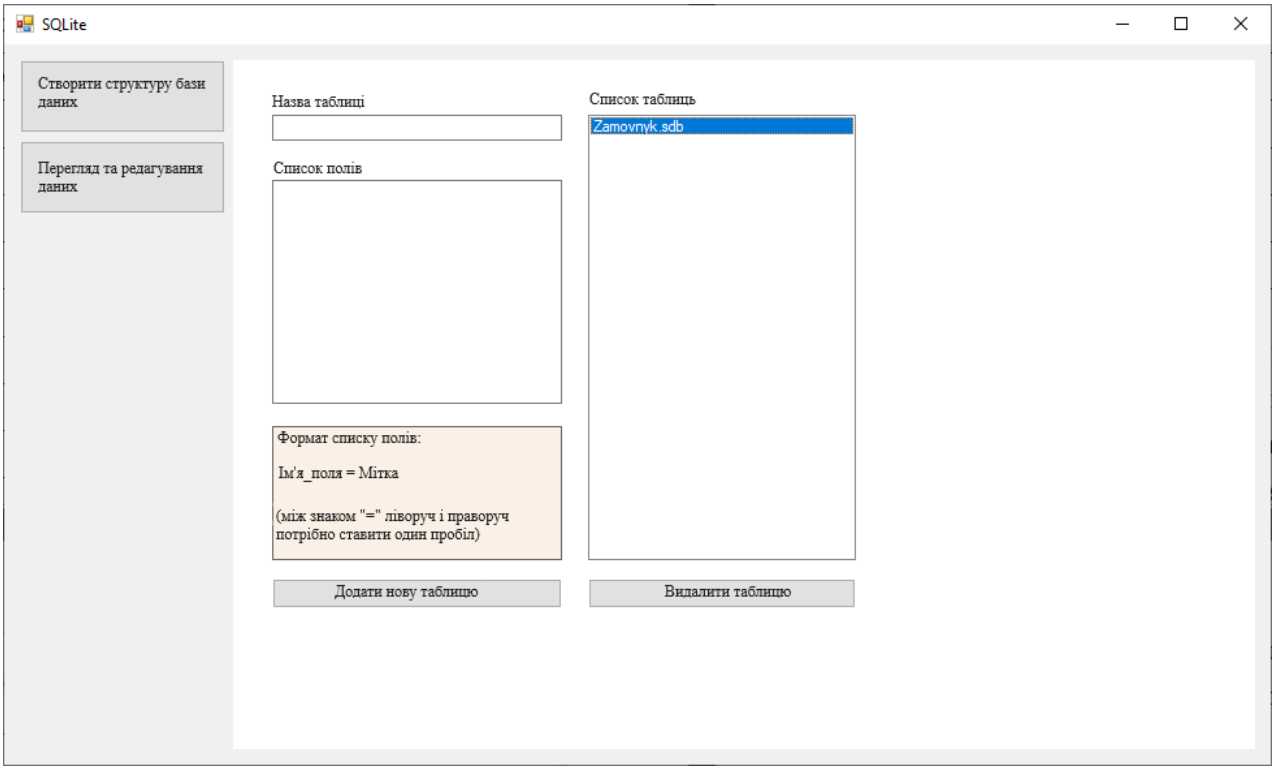
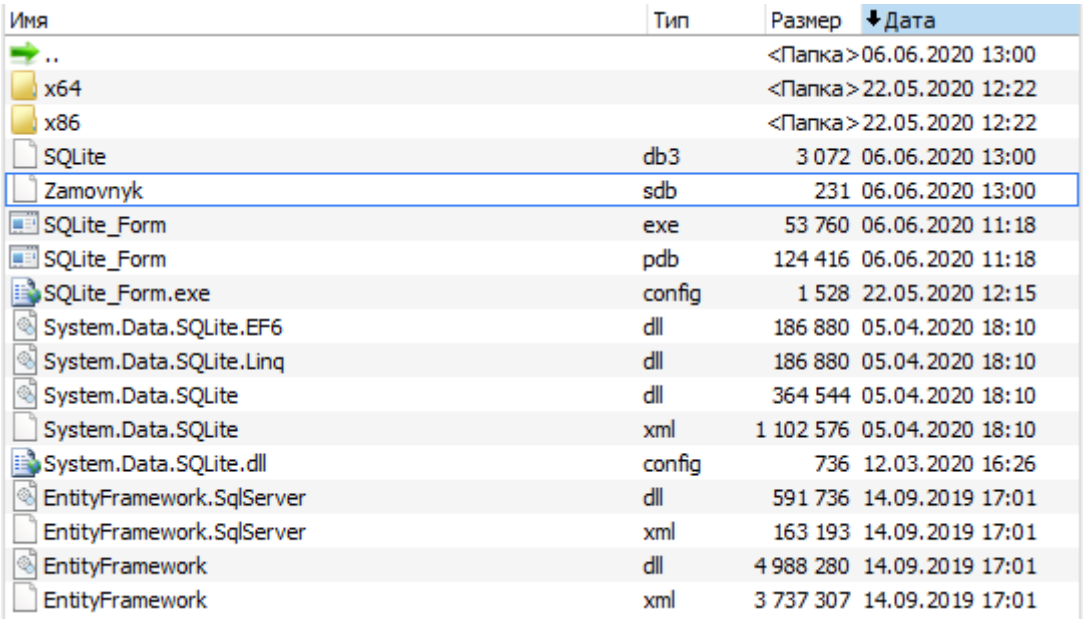


Рисунок 4.5 – Відображення нової таблиці в списку

На рисунку 4.6 показано структуру файлів на диску. Серед файлів можна бачити новий файл зі списком полів Zamovnyk.sdb.



Имя	Тип	Размер	Дата
..		<Папка>	06.06.2020 13:00
x64		<Папка>	22.05.2020 12:22
x86		<Папка>	22.05.2020 12:22
SQLite	db3	3 072	06.06.2020 13:00
Zamovnyk	sdb	231	06.06.2020 13:00
SQLite_Form	exe	53 760	06.06.2020 11:18
SQLite_Form	pdb	124 416	06.06.2020 11:18
SQLite_Form.exe	config	1 528	22.05.2020 12:15
System.Data.SQLite.EF6	dll	186 880	05.04.2020 18:10
System.Data.SQLite.Linq	dll	186 880	05.04.2020 18:10
System.Data.SQLite	dll	364 544	05.04.2020 18:10
System.Data.SQLite	xml	1 102 576	05.04.2020 18:10
System.Data.SQLite.dll	config	736	12.03.2020 16:26
EntityFramework.SqlServer	dll	591 736	14.09.2019 17:01
EntityFramework.SqlServer	xml	163 193	14.09.2019 17:01
EntityFramework	dll	4 988 280	14.09.2019 17:01
EntityFramework	xml	3 737 307	14.09.2019 17:01

Рисунок 4.6 – Структура файлів на диску

Відкривши менеджер бази даних можна переглянути структуру створеної таблиці за допомогою розробленої програми (рисунок 4.7).

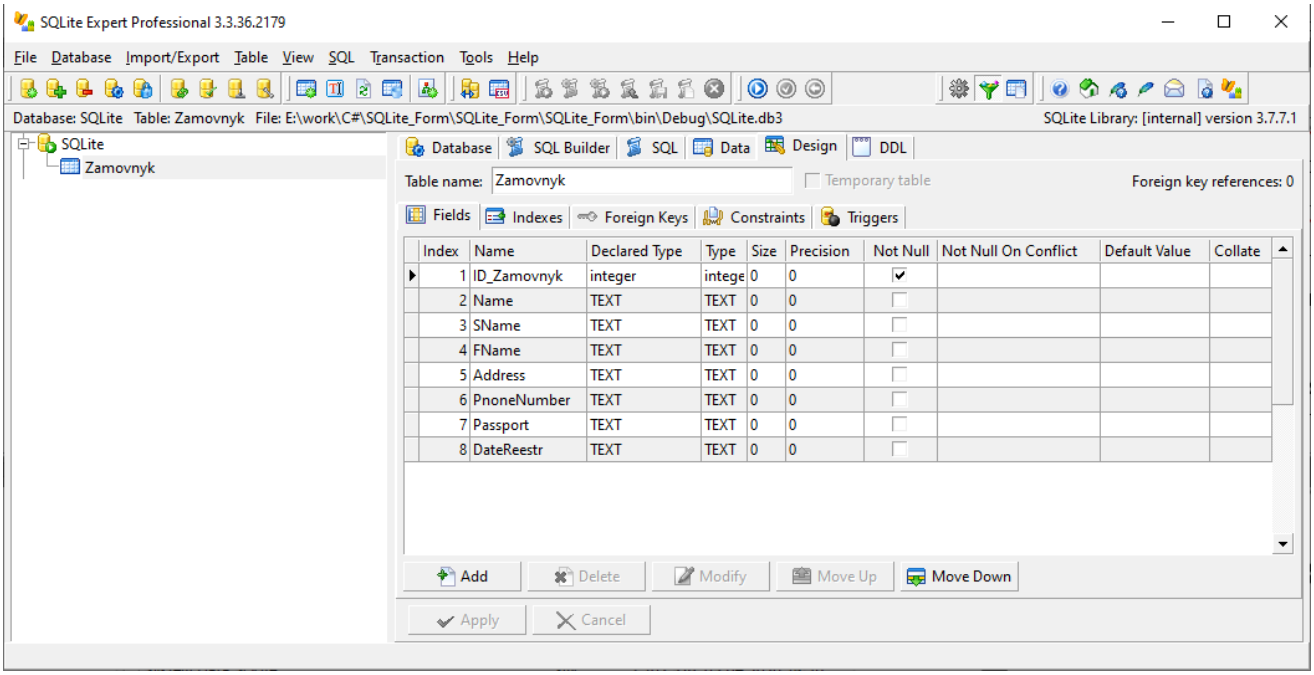


Рисунок 4.7 – Структура автоматично створеної таблиці Zamovnyk

Якщо в нашій програмі перейти на вкладку «Перегляд та редагування даних» то можна побачити, що з’явилась кнопка з назвою нової таблиці, як показано на рисунку 4.8.

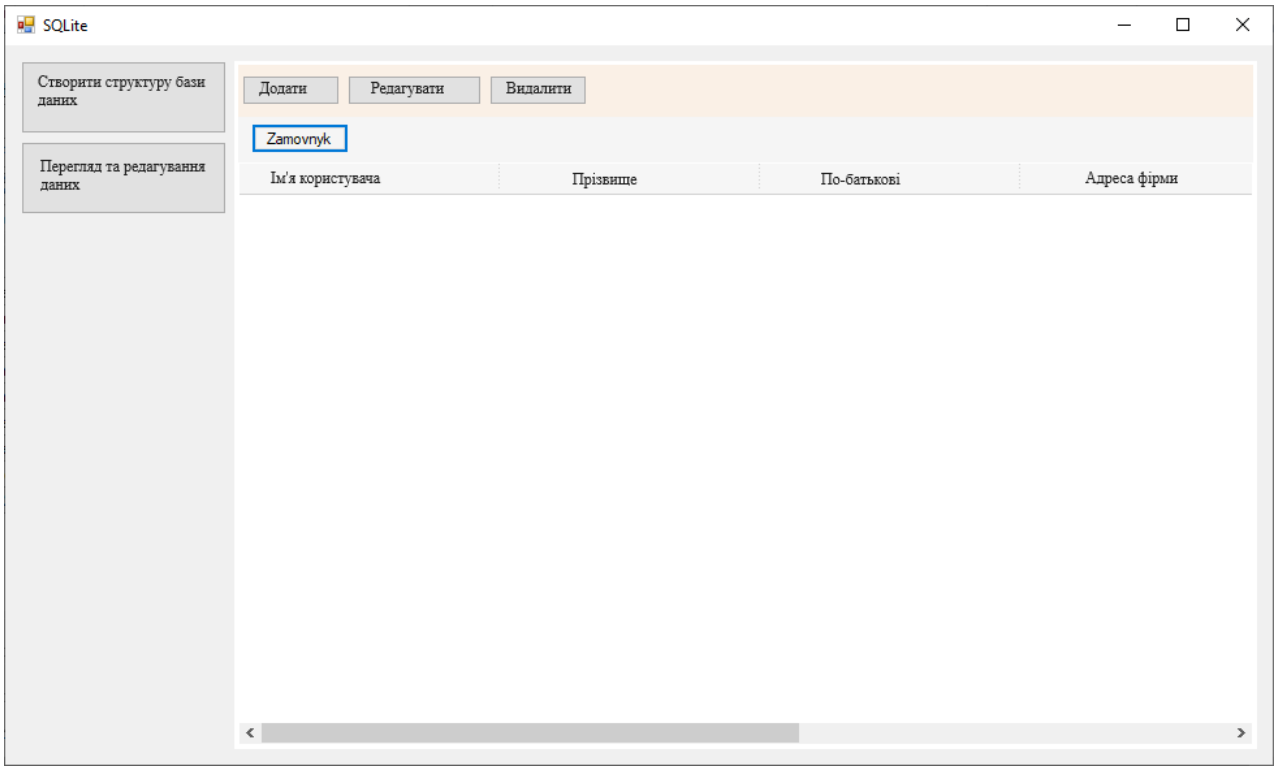


Рисунок 4.8 – Вкладка «Перегляд та редагування даних» з новою таблицею

На даному рисунку можна бачити всі створені поля та назви, які були задані на момент створення таблиці.

При натисканні на кнопку «Додати» автоматично створиться та відобразиться на екрані діалогове вікно зі всіма полями, що доступні для редагування.

На рисунку 4.9 показана приклад вікна додавання даних до таблиці «Замовники».

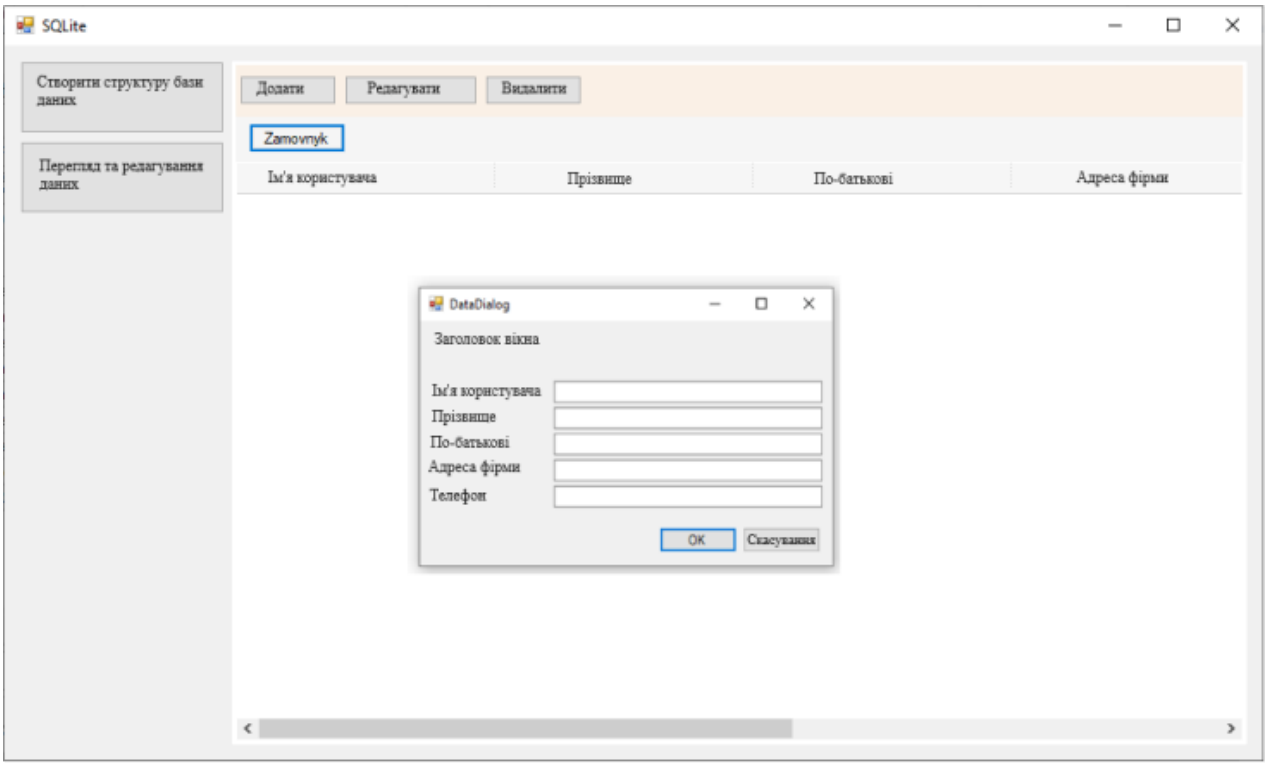


Рисунок 4.9 – Приклад вікна додавання даних до таблиці «Замовники».

На рисунку 4.10 наведено приклад заповнення полів форми.

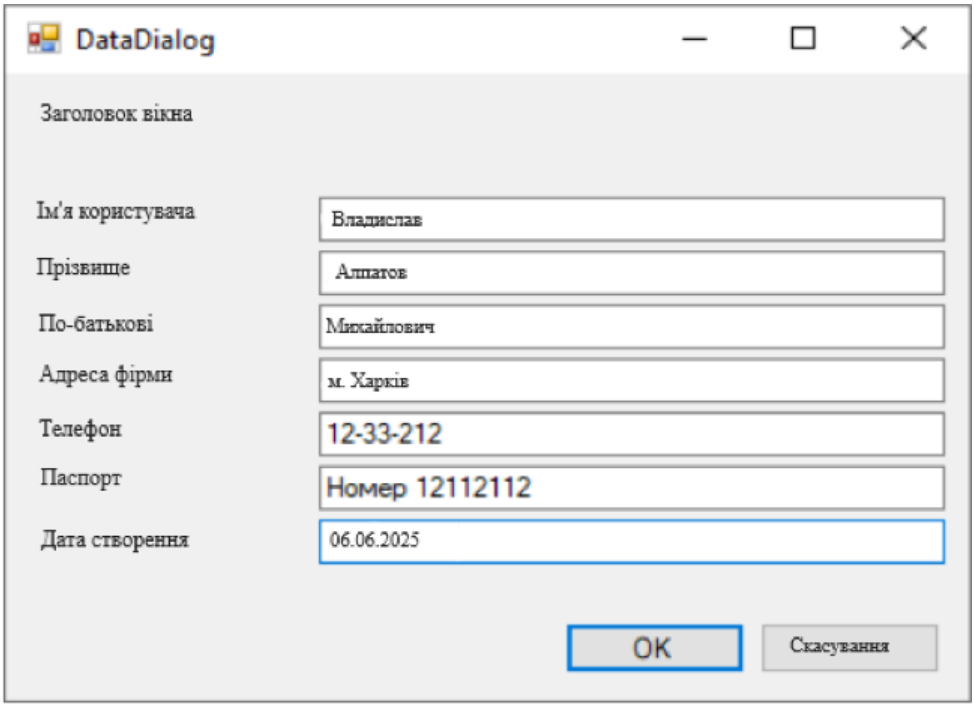


Рисунок 4.10 – Приклад заповнення полів форми

Після натискання на кнопку ОК (рисунок 4.10) введені дані попадуть в базу даних та їх можна переглянути за допомогою нашої програми, як показано на рисунку 4.11.

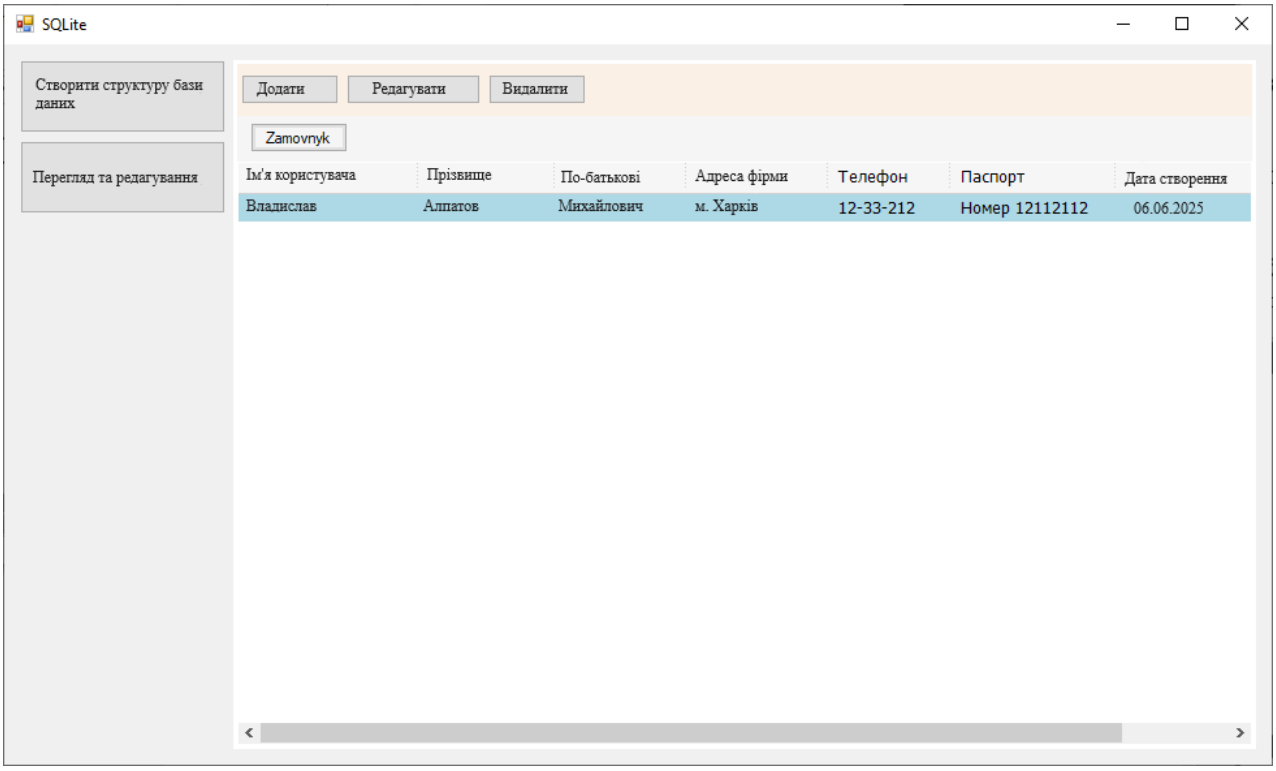


Рисунок 4.11 – Приклад введених даних

Аналогічним чином створимо таблиці «Замовлення», «Вироби» та «Категорії виробів».

Після створення всіх таблиць перша вкладка нашої програми буде мати такий вигляд, як показано на рисунку 4.12.

З наведеного рисунку можна бачити перелік всіх створених таблиць на диску (лівий список) та структуру таблиці «Виріб».

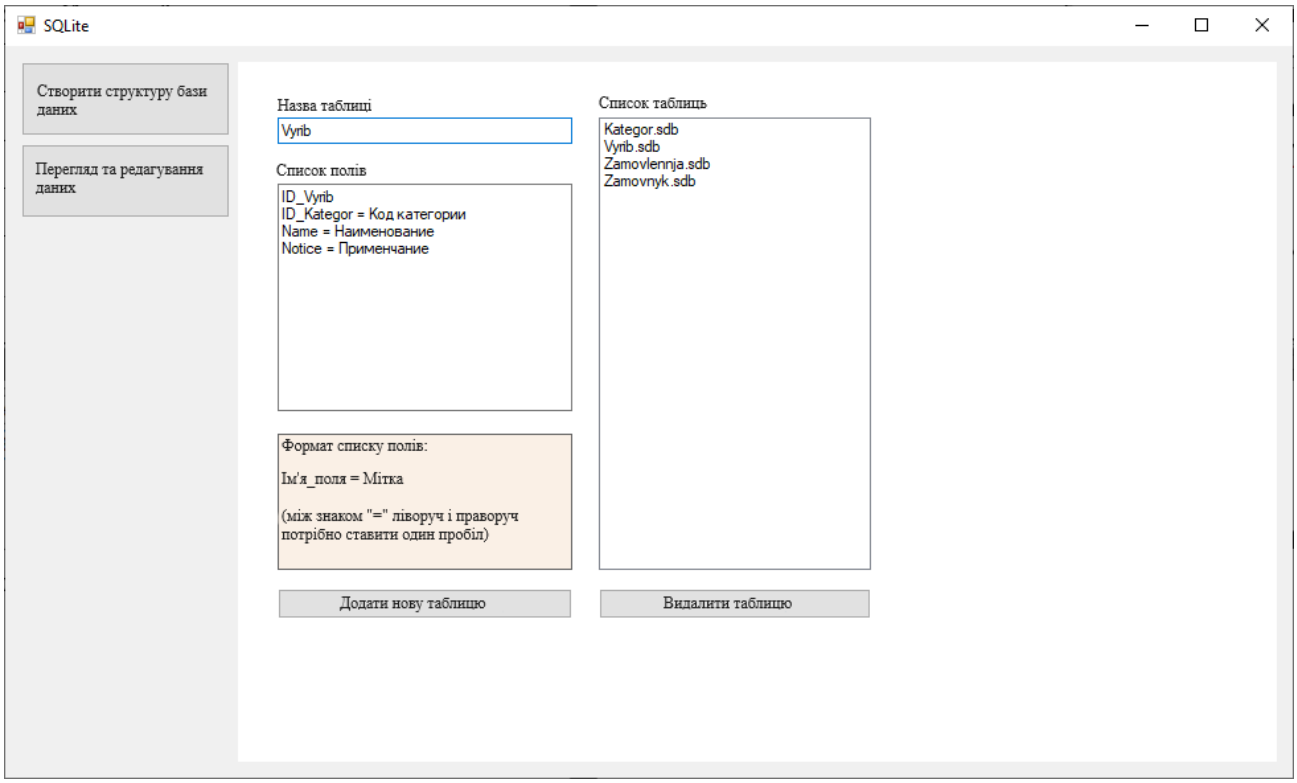


Рисунок 4.12 – Перелік всіх створених таблиць

На рисунку 4.13 показані нові файли автоматично створені на диску при створенні структури бази даних.

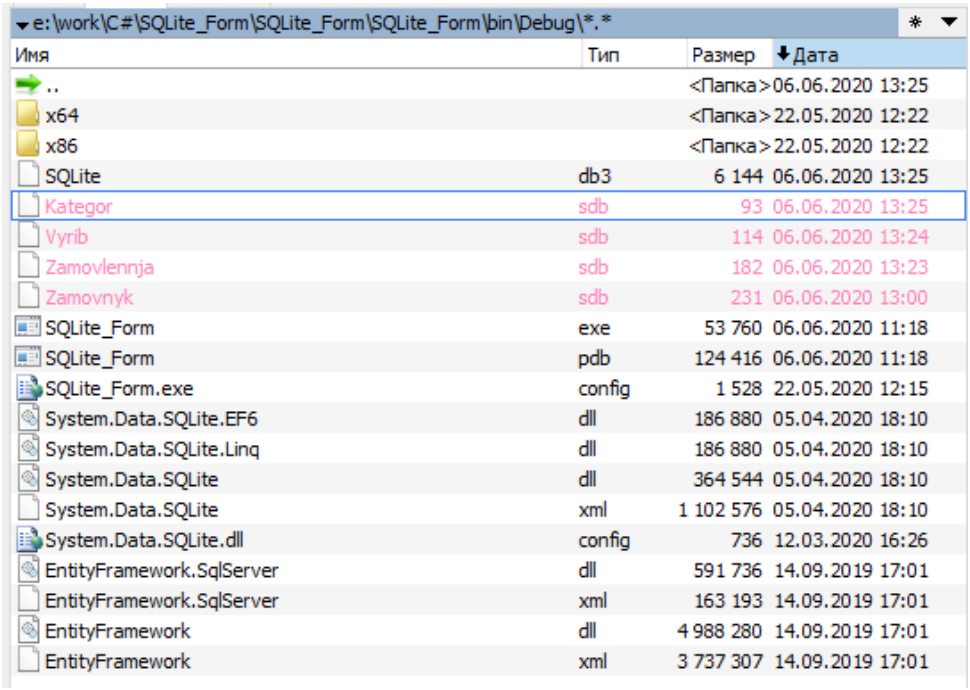


Рисунок 4.13 – Нові файли автоматично створені на диску

Якщо тепер перейти на другу вкладу програми, то можна бачити, що кількість кнопок в полі 2 збільшилося відповідно до кількості таблиць (рисунок 4.14).

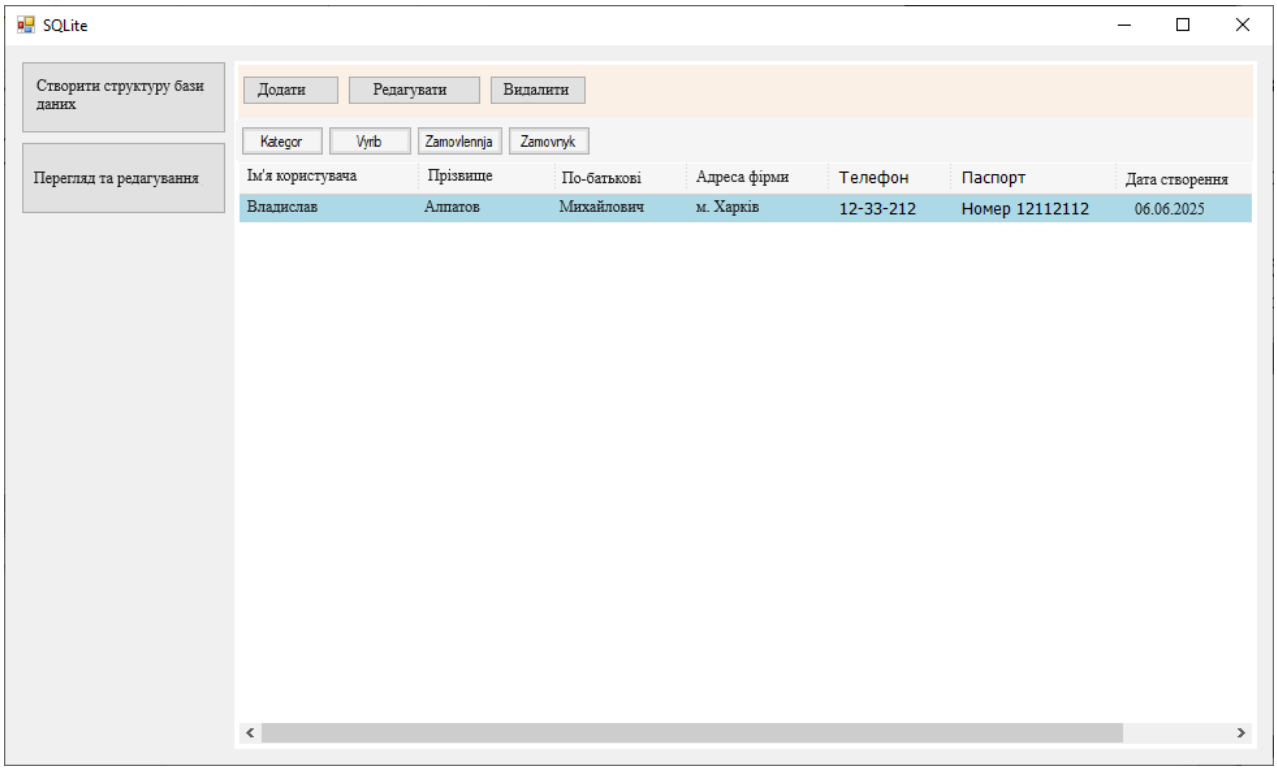


Рисунок 4.14 – Кнопки обирання таблиці для перегляду

На рисунку 4.15 показані всі діалогові вікна, які формуються нашою програмою відповідно до створеної структури таблиць.

На даному рисунку можна бачити вікна:

– вікно а – додавання даних про категорію;

– вікно б – додавання даних про виріб;

– вікно в – додавання даних про замовника;

– вікно г – додавання даних про заказ.

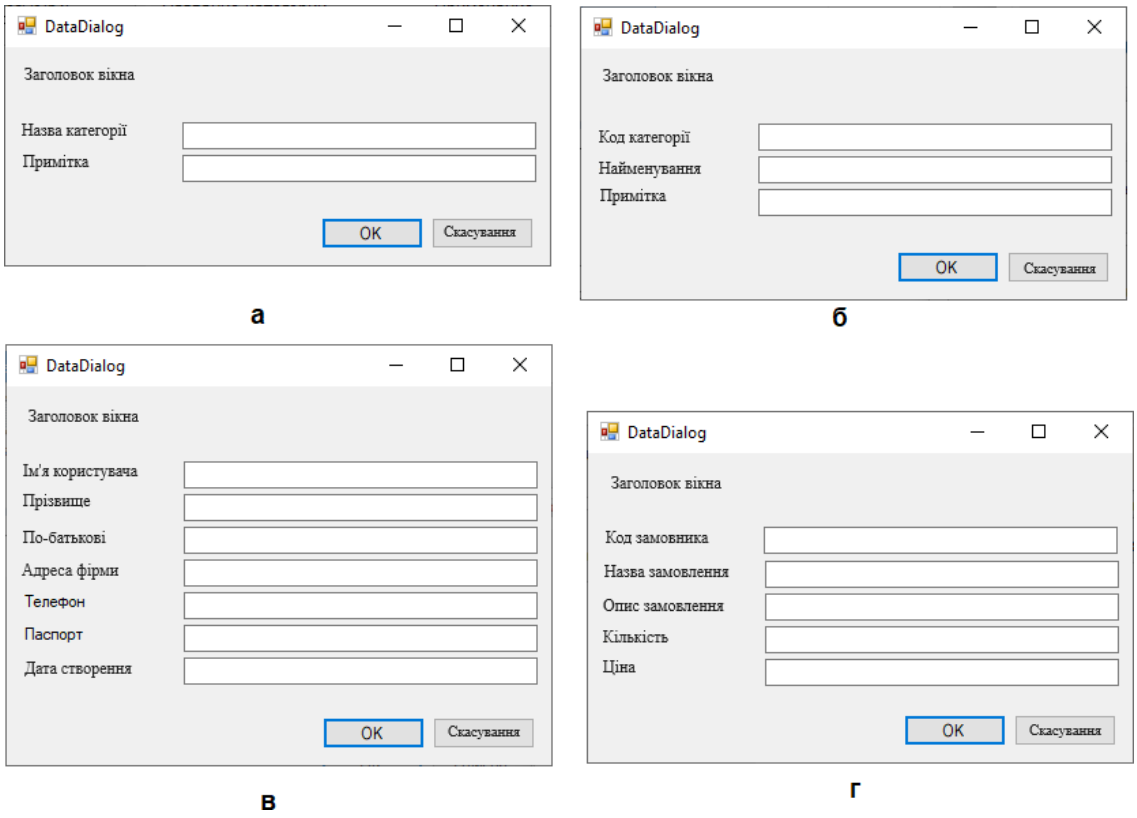


Рисунок 4.15 – Всі діалогові вікна, які формуються нашою програмою відповідно до створеної структури таблиць

Таким чином, було розроблено програмний модуль за допомогою якого можна автоматизовано створювати структуру будь-якої БД. Програма дозволяє автоматично генерувати діалогові вікна для додавання, редагування та видалення інформації відповідно до створеної структури БД.

Працездатність програми перевірена на практичному прикладі. Всі таблиці було створено без помилок. Було виконано додавання тестових даних, які потім можна було переглянути за допомогою вбудованого табличного інструменту.

4.3 Розрахунок штучного освітлення в дослідницькій лабораторії

Робота з розроблення автоматичної підсистеми відстеження поштових відправлень на основі технології Blockchain велася у дослідницькій лабораторії за робочим місцем користувача ПК. Її розміри становлять:

10 м × 6 м × 3,5 м. Площа – 60 м^2, обсяг – 210 м^3, кількість працюючих розробників – 8. Лабораторія розташована на 2 поверсі чотириповерхового залізобетонної будівлі. Устаткування, розміщене в приміщенні – 8 ПК.

Зорова робота користувача ПК є роботою високої точності, оскільки найменший розмір об'єкта розрізнення 0,3 – 0,5 мм, тому розряд зорової роботи – III. Згідно з вимогами ДБН В.25-28-2006 величина коефіцієнта природного освітлення повинна бути дорівнювати 2 %. Природне світло проникає в приміщення лабораторії через бічні світлові прорізи. Штучне освітлення слід виконати у вигляді безперервних або переривчастих ліній світильників, розташованих паралельно лінії зору користувачів. Освітленість на робочому столі повинна бути в межах 300 – 500 лк [12].

Необхідно виконати розрахунок загального рівномірного штучного освітлення і на підставі цього реконструювати систему штучного освітлення. Розрахунок буде проведено методом коефіцієнта використання світлового потоку.

Як джерело світла виберемо світильники ЛПО 12-2×40-904, лампи люмінесцентні 36 Вт, в одному світильнику 2 лампи (Philips TL-D 36W/54), оскільки вони володіють великою економічністю і світловіддачею, ніж лампи розжарювання. У зв'язку з цим найбільш доцільно вибрати систему загального освітлення [12].

Згідно ДБН В.25-28-2006 виконуються зорові роботи відносяться до III розряду зорових робіт (здатність розрізняти деталі від 0,3 – 0,5 мм).

Штучне освітлення нормується за ДБН В.25-28-2006, згідно з якими в приміщеннях з ЕОМ освітленість робочого місця повинна становити 300 – 500 лк. Розрахунок штучного освітлення виконується за методом використання світлового потоку [12].

Необхідне значення світлового потоку визначається як:

$$F = \frac{E \cdot S \cdot Z \cdot k}{\eta}, \qquad (4.10)$$

де S – площа приміщення;

E – нормоване значення освітленості;

Z – коефіцієнт нерівномірності (при розрахунку освітлення від світильників з люмінесцентними лампами Z = 1,1);

k – коефіцієнт запасу залежить від вмісту пилу в приміщенні, раз (приймається в межах від 1,3 – 2,0, в залежності від вмісту пилу у виробничих приміщеннях з урахуванням регулярного очищення світильників і виду джерела світла);

η – коефіцієнт використання світлового потоку, що залежить від індексу приміщення.

Для визначення коефіцієнта використання світлового потоку η знаходять індекс приміщення I і передбачувані коефіцієнти відображення поверхонь для світлих адміністративно конторських приміщень: стелі $r_{ст}$ = 70 %, стін r_c = 50 %, підлогі $r_п$ = 30 % [18].

У таблиці 4.1 наведено результати визначення коефіцієнта використання світлового потоку.

Таблиця 4.1 – Коефіцієнт використання світлового потоку

I	$r_{ст}$ = 70 % r_c = 50 % $r_п$ = 30 %	$r_{ст}$ = 50 % r_c = 30 % $r_п$ = 10 %	$r_{ст}$ = 30 % r_c = 10 % $r_п$ = 10 %
0,5	0,28	0,21	0,18
1,0	0,49	0,40	0,36
3,0	0,73	0,61	0,58
5,0	0,8	0,67	0,65

Обчислимо висоту підвісу світильників над робочою поверхнею:

$$h = H - h_1 - h_2, \qquad (4.11)$$

де H – висота приміщення, м;

h_1 – висота робочої поверхні, дорівнює 0,8 м;

h_2 – підвіс світильника, дорівнює 0 м.

Висота приміщення $H = 3$ м, висота робочої поверхні $h_1 = 0{,}8$ м, висота схилу для даного типу світильників $h_2 = 0$ м. Підставляючи дані величини в вираз (3.10), отримуємо:

$$h = 3{,}5 - 0 - 0{,}8 = 2{,}7 \text{ м}.$$

Обчислимо індекс приміщення:

$$I = \frac{A \cdot B}{h \cdot (A + B)}, \tag{4.12}$$

$$I = \frac{10 \cdot 6}{2{,}7 \cdot (6 + 10)} = 1{,}38.$$

Відстань між рядами світильників:

$$L - I_p \cdot h, \tag{4.13}$$

де I_p – характерне відстань між рядами (для даного світильника $I_p = 1{,}3$).

$$L = 1{,}3 \cdot 2{,}7 = 3{,}5 \text{ м}.$$

Число рядів світильників:

$$n = \frac{b}{L} = \frac{6}{3{,}5} = 1{,}8 \approx 2. \tag{4.14}$$

Світильники розміщуємо в два ряди. Відстань між стіною і крайніх рядів:

$$l=(0{,}3\ldots0{,}5)\cdot L=1{,}75 \text{ м}. \qquad (4.15)$$

Для ламп Philips TL-D 36W/54 номінальний світловий потік $F_{ламп.}=2500$ лк.

Число світильників в ряду:

$$N=\frac{E\cdot S\cdot Z\cdot k}{\eta\cdot n\cdot m\cdot F_{св}}, \qquad (4.16)$$

$$N=\frac{300\cdot 60\cdot 1{,}5\cdot 1{,}1}{0{,}54\cdot 2\cdot 2\cdot 2500}=5{,}5\approx 6.$$

де m – кількість ламп в світильнику.

При довжині одного світильника $L_{св}=1{,}235$ м, загальна довжина $L_{св}=1{,}235\cdot 6=8{,}1$ м. $L_{св}<A$, тобто світильники розташовуються уздовж A.

ВИСНОВКИ

В результаті виконання кваліфікаційної роботи розроблено програмний модуль для автоматизації роботи з базою даних SQLite.

Виконано аналіз технічного завдання, та розглянуті сучасні програмні інструменти для роботи за базою даних SQLite. Була описана структура програмного модуля, що розроблюється, наведена структура файлової системи для зберігання опису таблиць. Розроблено алгоритми роботи програми, описані параметри вхідної інформації та розроблена структура бази даних.

Для створення програмного модуля було використано мову програмування C# та середовище розробки VisualStudio. Описано інтерфейс користувача та програмні функції.

Таким чином, було розроблено програмний модуль за допомогою якого можна автоматизовано створювати структуру будь-якої бази даних. Програма дозволяє автоматично генерувати діалогові вікна для додавання, редагування та видалення інформації відповідно до створеної структури бази даних.

Виконано тестовий приклад створення структури бази даних для зберігання інформації про замовлення виготовлення виробів на підприємстві. Працездатність програми була перевірена на практичному прикладі. Всі таблиці було створено без помилок. Було виконано додавання тестових даних, які потім можна було переглянути за допомогою вбудованого табличного інструменту.

Проведено заходи і розрахунки для забезпечення безпечних умов роботи в лабораторії, де виконувалася кваліфікаційна робота.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. ДСТУ 3008:2015 Інформація та документація «Звіти у сфері науки і техніки». Структура та правила офомрлювання. / В. Земцева; Ю. Поліщук, канд. фіз.-мат. наук; Р. Санченко, канд. техн. наук; Л. Шрамко; А. Ямчук (науковий керівник) ДП «УкрНДНЦ» від 22 червня 2015р. № 61 з 2017- 07-01.

2. Методичні вказівки з підготовки кваліфікаційної роботи для здобувачів першого (бакалаврського) рівня вищої освіти денної і заочної форми навчання спеціальності 151 «Автоматизація та комп’ютерно-інтегровані технології» освітньої програми «Автоматизація та комп’ютерно-інтегровані технології» / Упоряд.: І. Ш. Невлюдов, О. І. Филипенко, О. В. Токарєва, С. П. Новоселов, О. В. Сичова. – Харків: ХНУРЕ, 2023. – 64 с.

3. Навчальний посібник з підготовки кваліфікаційної роботи бакалавра для здобувачів вищої освіти денної і заочної форм навчання спеціальності 151 «Автоматизація та комп’ютерно-інтегровані технології» освітньої програми «Автоматизація та комп’ютерно-інтегровані технології»: Навчальний посібник / І. Ш. Невлюдов, В.А. Андрусевич, О. В. Токарєва, С. П. Новоселов, О. В. Сичова. – Харків : Видавництво Іванченка І. С., 2022. – 151 с.

4. SQL[Електронний ресурс]. – Електрон. текстові дані. – Режим доступу:https://uk.wikipedia.org/wiki/SQL

5. Data types – Microsoft.Data.Sqlite [Електронний ресурс]. – Електрон. текстові дані. – Режим доступу:https://learn.microsoft.com/uk-ua/dotnet/standard/data/sqlite/types

6. User_Manual · pawelsalawa/sqlitestudio Wiki [Електронний ресурс]. – Електрон. текстові дані. – Режим доступу:https://github.com/pawelsalawa/sqlitestudio/wiki/User_Manual

7. Засіб для роботи з базами даних DBeaver [Електронний ресурс]. – Електрон. текстові дані. – Режим доступу:https://ukr-technologies.blogspot.com/2020/03/dbeaver.html

8. Kexi [Електронний ресурс]. – Електрон. текстові дані. – Режим доступу: https://uk.wikipedia.org/wiki/Kexi

9. Visual Studio 2019[Електронний ресурс]. – Електрон. текстові дані. – Режим доступу:https://visualstudio.microsoft.com/ru/vs/

10. NuGet [Електронний ресурс]. – Електрон. текстові дані. – Режим доступу:https://uk.wikipedia.org/wiki/NuGet

11. SQLiteExpert: The expert way to SQLite [Електронний ресурс]. – Електрон. текстові дані. – Режим доступу:http://www.sqliteexpert.com/

12. Комплекс навчально-методичного забезпечення навчальної дисципліни «Організація керування умовами праці» підготовки освітнього рівня бакалавр усіх спеціальностей та усіх напрямів університету [Електронний ресурс] / ХНУРЕ; розроб.: Т.Є. Стиценко, Г.В. Пронюк, Н.М. Сердюк. – Харків, 2017. – 108 с.