МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ ІМЕНІ СЕМЕНА КУЗНЕЦЯ

В. В. Федько

ТЕХНОЛОГІЇ БАЗ ДАНИХ

Лабораторний практикум

Харків
ХНЕУ ім. С. Кузнеця
2020

УДК 004.65(076)
Ф32

Рецензенти: професор кафедри інформатики Харківського національного педагогічного університету імені Г. С. Сковороди, д-р пед. наук, канд. техн. наук *О. Г. Колгатін*; професор кафедри системотехніки Харківського національного університету радіоелектроніки, канд. техн. наук *Д. Е. Ситніков.*

Рекомендовано до видання рішенням ученої ради Харківського національного економічного університету імені Семена Кузнеця.
Протокол № 5 від 26.12.2019 р.

Самостійне електронне текстове мережеве видання

Федько В. В.

Ф32 Технології баз даних [Електронний ресурс] : лабораторний практикум / В. В. Федько. – Харків : ХНЕУ ім. С. Кузнеця, 2020. – 344 с.
ISBN 978-966-676-792-2

Наведено навчальний матеріал для виконання лабораторних робіт під час вивчення навчальної дисципліни. Розглянуто класичні та сучасні засоби доступу до даних, що зберігають у базах даних, технологію розроблення застосунків для виконання CRUD-операцій із базами даних та аналізу даних. Для поглиблення вмінь та навичок запропоновано значну кількість завдань для самостійного виконання. Завдання ґрунтуються на одній предметній області й охоплюють широкий спектр засобів розроблення застосунків із базами даних MS SQL Server.

Рекомендовано для студентів галузі знань 12 "Інформаційні технології" першого (бакалаврського) рівня всіх форм навчання.

УДК 004.65(076)

ISBN 978-966-676-792-2

Вступ

Абсолютна більшість сучасних програмних продуктів використовують дані, що зберігають у базах даних. Тому оволодіння технологіями доступу до даних стали невід'ємним напрямом у підготовці сучасних фахівців у галузі комп'ютерних наук. Цей лабораторний практикум призначено, насамперед, студентам напряму підготовки в галузі знань 12 "Інформаційні технології" першого (бакалаврського) рівня всіх форм навчання.

Лабораторний практикум призначено для формування у студентів компетентностей, які забезпечено здобуттям концептуальних сучасних знань і умінь виконувати складні непередбачувані завдання, створюванням у тих, хто навчається, спроможності донесення до фахівців власного досвіду, а також відповідальності за ухваленням рішень у різноманітних умовах.

Практикум написано, відповідно до програми навчальної дисципліни "Технології баз даних" [3], він містить опис таких лабораторних робіт:

Лабораторна робота 1. Розробка програм виконання операцій у з'єднаному середовищі.

Лабораторна робота 2. Розробка програм виконання операцій у роз'єднаному середовищі.

Лабораторна робота 3. Розробка програм на основі інтерфейсу JDBC.

Лабораторна робота 4. Розробка програм із використанням типізованих наборів даних.

Лабораторна робота 5. Розробка програм із використанням технології LINQ to DataSet.

Лабораторна робота 6. Реалізація доступу до даних на основі технології Code First.

Лабораторна робота 7. Реалізація звітів у бізнес-застосунках.

Теоретичний матеріал до лабораторних робіт подано у друкованих навчальних посібниках [1; 2] та в електронному мультимедійному навчальному посібнику [4].

У процесі виконання лабораторних робіт студенти оволодіють такими професійними компетентностями:

здатність використовувати режими з'єднаного та роз'єднаного середовища для доступу до даних у бізнес-застосунках;

здатність застосовувати сучасні теорії організації баз даних, методи та технології їхнього розроблення.

У межах зазначених компетентностей виконання завдань лабораторних робіт дозволяє студентам досягти таких результатів навчання:

оволодіти принципами побудови архітектури сучасних бізнес-застосунків;

застосовувати основні концепції організації з'єднань зі сховищами даних у програмних продуктах;

вибирати найбільш доцільне середовище організації доступу до даних;

використовувати у проєктах основні компоненти ADO.NET та JDBC;

будувати алгоритми створення з'єднання із джерелом даних у програмних проєктах;

проєктувати засоби керування з'єднанням зі сховищем даних у програмних продуктах;

організовувати автономну роботу з базою даних;

створювати програми для виконання операцій із даними в з'єднаному та роз'єднаному середовищі;

прив'язувати дані до інтерфейсу користувача у програмних продуктах;

уміти використовувати переваги типізованих наборів даних під час програмування бізнес-застосунків;

застосовувати технологію LINQ у завданнях аналізу даних;

визначати найбільш раціональні сценарії використання платформи Entity Framework;

будувати моделі даних засобами технології Code First;

виконувати налаштування сутностей у сценарії Code First;

модифікувати моделі даних засобами міграцій;

програмувати застосунки Windows Forms на основі технології Code First;

проєктувати та створювати документи на основі даних, що зберігають у базі;

розробляти програмні засоби бізнес-аналізу даних.

Вивчення матеріалу навчального посібника має відбуватися в такому порядку: спочатку прочитайте матеріал, який подано в навчальних посібниках [1; 2; 4], спробуйте відповісти на запитання і виконайте завдання, розміщені в кінці відповідних параграфів. Після знайомства з теоретичним матеріалом розділу виконайте лабораторну роботу. Під час

її виконання слід звернути увагу на добір завдань. Він має бути посильним для розуміння студентом. Із цією метою кожний студент самостійно вибирає рівень складності з-поміж таких: 1) фронтальний; 2) індивідуальний; 3) компетентнісний.

Якщо вибрано фронтальний рівень, то студент виконує завдання базового рівня, детально описаних в інструкції. За їхнє виконання студент набирає 4 бали за 12-бальною системою оцінювання.

Із метою випробування своїх сил і підвищення оцінки, студент може самостійно виконати ще кілька завдань, частина з яких репродуктивного, а інші – креативного типу. За правильне їхнє виконання додають ще до трьох балів. До набраної суми балів студент може додати ще два бали, якщо самостійно запропонує і виконає оригінальне завдання за темою, що вивчають. Це завдання має бути із предметної області особистих зацікавлень студента (індивідуальне завдання). Загальна оцінка за цим рівнем не перевищує 10 балів.

У разі вибору індивідуального рівня студент ознайомлюється з інструкцією щодо виконання завдання базового рівня і виконує аналогічне завдання із предметної області, визначену індивідуальним завданням. За виконання такого індивідуального завдання студент набирає сім балів. Ще два бали він може набрати, якщо адаптує до предметної області вибраного варіанта завдання, подані в інструкції, і виконає їх. Подібно до фронтального рівня студент може додати ще два бали до набраної суми балів, якщо сформулює і виконає оригінальне завдання за темою, що вивчає. Загальна оцінка за цим рівнем не перевищує 11 балів.

На компетентнісному рівні студент демонструє можливість самостійно ставити та виконувати завдання за темою, що вивчають, із предметної області індивідуального завдання. Спочатку він формулює і виконує завдання аналогічне базовому (вісім балів), потім – аналогічні додатковим завданням (ще два бали) і на сам кінець – оригінальне завдання (до двох балів). Загальна оцінка за цим рівнем може досягати 12 балів.

Переважну більшість робіт практикуму виконують у середовищі Visual Studio. Його можна безкоштовно завантажити із сайту компанії Microsoft, вибравши версію Community [5]. У цьому разі застосунки розробляють мовою C# на основі шаблону Windows Forms. Він дозволяє достатньо просто реалізувати інтерфейс користувача. Оволодівши принципами розроблення застосунків у Windows Forms, можна достатньо

просто перейти до WPF чи вебпроєктів. Як СУБД у цих роботах практикуму використовують SQL Server як одну з найбільш поширених і потужних систем. Перевагою цієї СУБД є також те, що її не потрібно додатково встановлювати. Найпростіший, але достатній для виконання лабораторних робіт сервер встановлюється автоматично під час розгортання Visual Studio на комп'ютері.

Водночас потрібно звернути увагу на ту обставину, що ім'я встановленого сервера визначено версією Visual Studio. Залежно від версії, ім'я сервера (параметр **Data Source**) має такі значення:

для Visual Studio 2010 – **.\SQLExpress**;

для Visual Studio 2012, 2013 – **(localDB)\v11.0**;

для вищих версій – **(localDB)\MSSQLLocalDB**.

Потрібне значення вказують у параметр **Data Source** рядка з'єднання ConnectionString.

У зв'язку зі значними адміністративними обмеженнями комп'ютерів в аудиторіях університету для Visual Studio 2010 (сервер **.\SQLExpress**), слід вибирати екземпляр користувача сервера, додатково вказавши в рядку з'єднання параметр **User Instance=True**.

Для забезпечення високої мобільності проєктів, що створюють під час виконання лабораторних робіт, краще користуватися не базою даних, постійно під'єднаної до SQL Server, а окремим mdf-файлом. Його зберігають у тій самій папці, що й проєкт. Такий підхід дозволяє без додаткових витрат часу продовжувати роботу над проєктом удома і, навпаки, демонструвати на заняттях результати, визначені на домашньому комп'ютері під час самостійної роботи студента.

Третю лабораторну роботу, присвячену вивченню технології JDBC у застосунках, що розробляють мовою Java, виконують у середовищі NetBeans. Його можна безкоштовно завантажити із сайту [6].

Загалом лабораторний практикум є самодостатнім для оволодіння засобами доступу до даних із застосунків. Виконання описаних у ньому лабораторних робіт дозволить набути професійних навичок розробника програмних продуктів.

Розділ 1
Класичні засоби доступу до даних

Лабораторна робота 1
Розробка програм виконання операцій у з'єднаному середовищі

Цілі лабораторної роботи:

1. Набуття практичних навичок у створенні й запуску командних об'єктів.

2. Набуття практичних навичок у використанні класів провайдерів даних ADO.NET.

3. Набуття практичних навичок у програмній реалізації CRUD-операцій із базою даних у з'єднаному середовищі.

4. Удосконалювання навичок у роботі з інтегрованим середовищем розроблення Visual C# і довідковою системою Microsoft Developer Network (MSDN).

Перед виконанням роботи студент має знати:

основи використання бібліотеки Windows Forms;
організацію системи створення й керування з'єднанням в ADO.NET;
основні команди мови SQL;
методику створення функцій-оброблювачів подій;
принципи оброблення винятків у C#-програмі.

Після виконання роботи студент має вміти:

самостійно розробляти прості C#-програми із графічним інтерфейсом користувача для встановлення з'єднання із сервером даних;

самостійно створювати бази даних і таблиці в них, використовуючи засоби C#;

забезпечувати програмну реалізацію виконання CRUD-операцій із базою даних у з'єднаному середовищі;

використовувати основні бібліотеки. Net Framework під час розроблення програм.

Підготовча частина

Хід роботи

1. Створення кнопкової форми застосунку.
2. Операції з базою даних загалом.
3. Створення і заповнення таблиці ***Товари***.
4. Зміна даних таблиці ***Товари***.
5. Виконання CRUD-операцій з іншими таблицями (самостійно).

Форма звітності

За результатами виконання роботи необхідно оформити електронний звіт засобами Word. За кожним завданням слід зробити скриншоти з результатами виконання, а також подати код оброблювача події. У кінці деяких завдань зазначено, що слід надати пояснення. Аналогічним чином оформити звіт із виконання кожного завдання з п. "Завдання для самостійного виконання".

Критерії оцінювання

У 12-бальній системі оцінку результату захисту лабораторної роботи формують за такими правилами:

1. За кожний пункт практичної частини може бути виставлено від 0 до 1 бала.

2. За виконання завдань п. "Завдання для самостійного виконання" може бути виставлено:

від 0 до 1 бала за кожне завдання з діапазону 1 – 3;

від 0 до 2 балів за кожне завдання з діапазону 4 – 6;

від 0 до 4 балів за завдання 7;

від 0 до 3 балів за завдання 8;

від 0 до 6 балів за завдання 9.

3. За кілька варіантів виконання одного із завдань додають 1 бал.

4. За вибір варіанта, який із кількох варіантів виконання є оптимальним, та обґрунтування вибору додають 1 бал.

5. За виконання кожного завдання в іншій СУБД (Oracle, DB2, MySQL, PostgreSQL тощо) додають по 1 балу.

6. За запізнення із захистом лабораторної роботи знімають 2 бали за кожний тиждень.

Набрану кількість балів із відповідей на кожне завдання лабораторної роботи підсумовують. У результаті такого підрахунку студент може набрати від 0 до 12 балів.

У разі запізнення із захистом лабораторної роботи понад три тижні студент виконує завдання роботи на основі індивідуальної бази даних. Роботу оцінюють за описаними раніше критеріями.

Практична частина

Постановка загального завдання

Вивчення засобів роботи з базами даних у з'єднаному середовищі здійснюють шляхом створення застосунку ***з'єднанийХліб***. Його призначено для роботи з базою даних ***Хліб*** у СУБД SQL Server.

У базі даних ***Хліб*** зберігають дані про роботу кіоску із продажу хлібобулочних виробів – отримання товарів і їхній продаж. База даних має схему, подану на рис. 1.1.

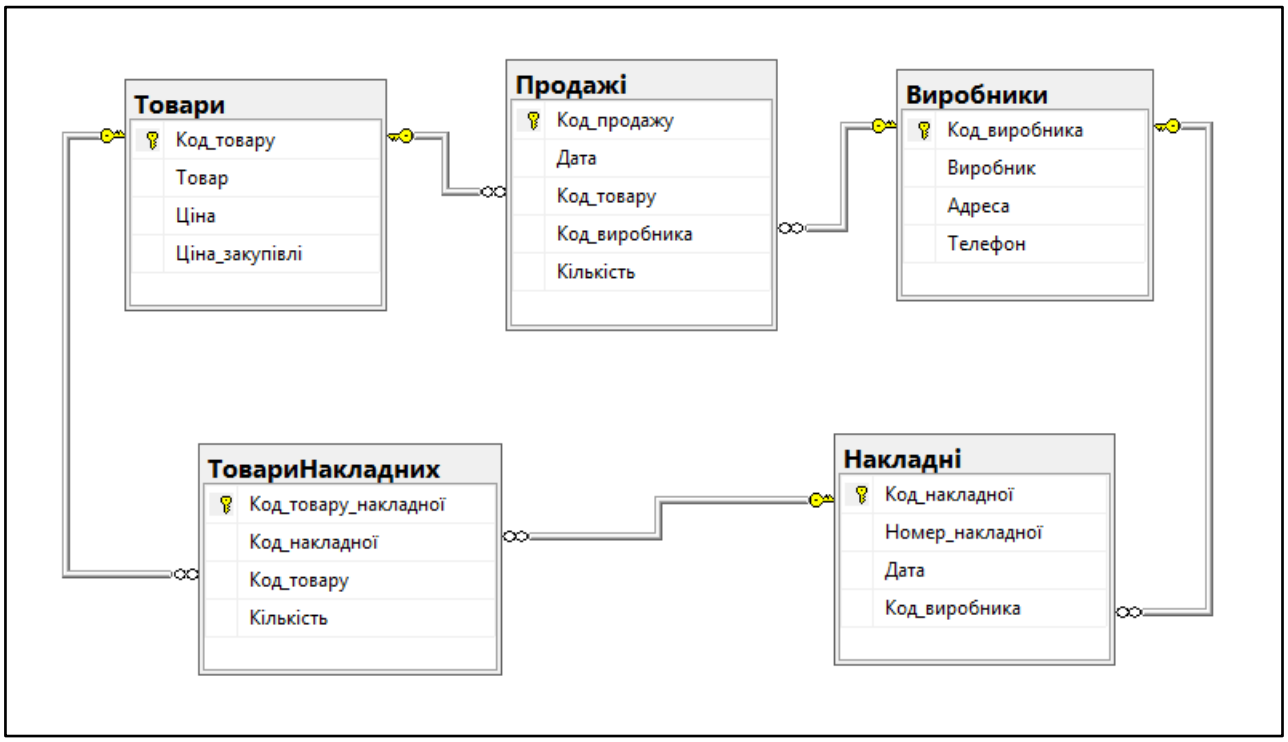


Рис. 1.1. **Схема бази даних *Хліб***

До складу бази даних входять таблиці, які мають такі типи даних полів:

Товари (
Код_товару int,
Товар nvarchar(25),
Ціна Money,
Ціна_закупівлі Money);

Продажі (
Код_продажу int,
Дата datetime,
Код_товару int,
Код_виробника int,
Кількість smallint);

ТовариНакладних (
Код_товару_накладної int,
Код_накладної int,
Код_товару int,
Кількість smallint).

Виробники (
Код_виробника int,
Виробник nvarchar(20),
Адреса nvarchar(30),
Телефон nvarchar(15));

Накладні (
Код_накладної int,
Номер_накладної nvarchar(6),
Дата datetime,
Код_виробника int);

Батьківські таблиці ***Товари*** та ***Виробники*** є довідниковими. У таблиці ***Продажі*** фіксують результати щоденних продажів. За допомогою таблиць ***Накладні*** та ***ТовариНакладних*** подають дані накладних, за якими кіоск отримує товари.

Управління застосунком здійснюють за допомогою кнопкової форми ***Хліб*** (рис. 1.2). На початку вона має кнопки, призначені для виконання операцій загалом з базою даних, а також із таблицею ***Товари***.

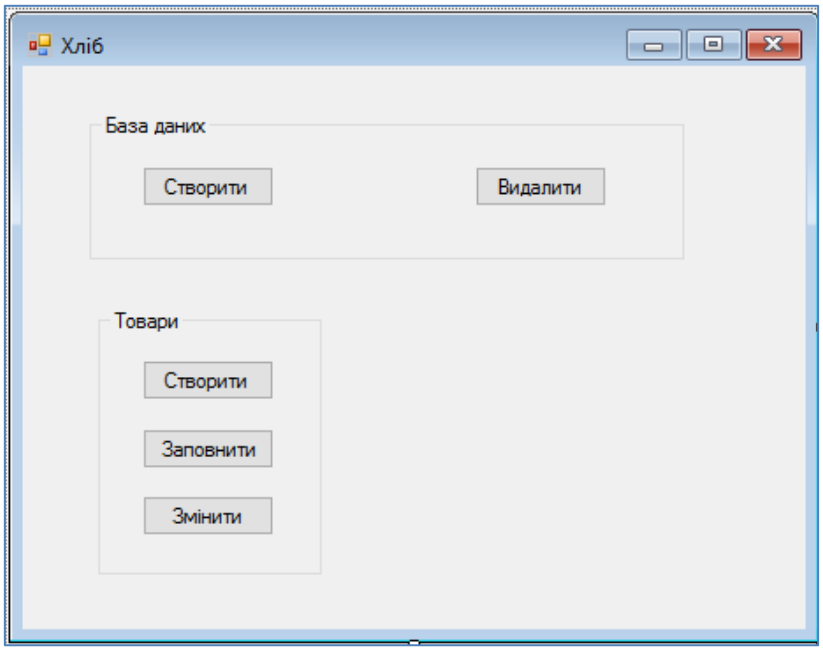


Рис. 1.2. **Кнопкова форма**

У групі кнопок ***База даних*** за допомогою кнопки ***Створити*** викликають скрипт для створення бази даних ***Хліб***, а за допомогою кнопки ***Видалити*** викликається скрипт для видалення цієї бази.

Групу кнопок ***Товари*** призначено для виконання операцій з однойменною таблицею. За допомогою кнопки ***Створити*** викликають скрипт для створення таблиці, а за допомогою кнопки ***Заповнити*** викликають скрипт для заповнення її даними. Клацання кнопки ***Змінити*** відкриває форму ***Товари***. На ній відображено дані однойменної таблиці – у списку всі записи, а в текстовому полі під ним – середня ціна всіх товарів (рис. 1.3).

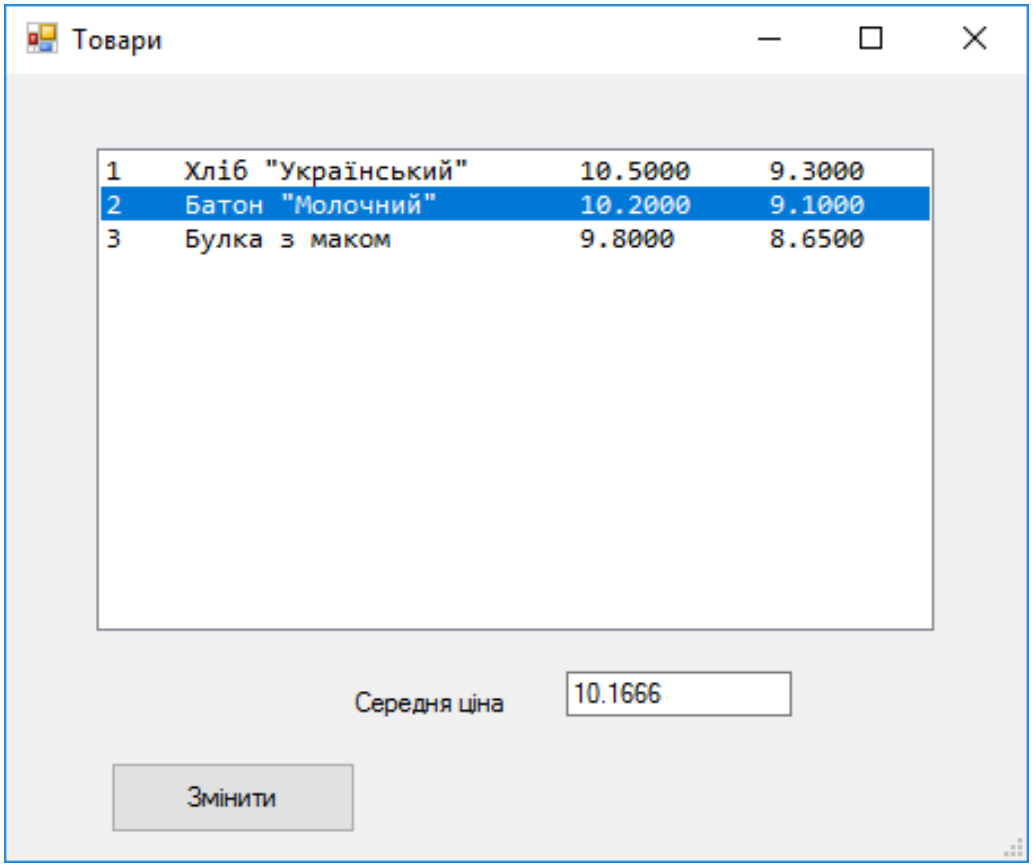


Рис. 1.3. **Форма *Товари***

Щоб змінити дані про який-небудь товар, його вибирають у списку та клацають кнопку ***Змінити***. Відкривається форма ***Товар***, у якій відображено дані з вибраного товару (рис. 1.4). Тут їх можна змінити. Щоб зафіксувати в базі даних зроблені зміни, клацають кнопку ***Зберегти***.

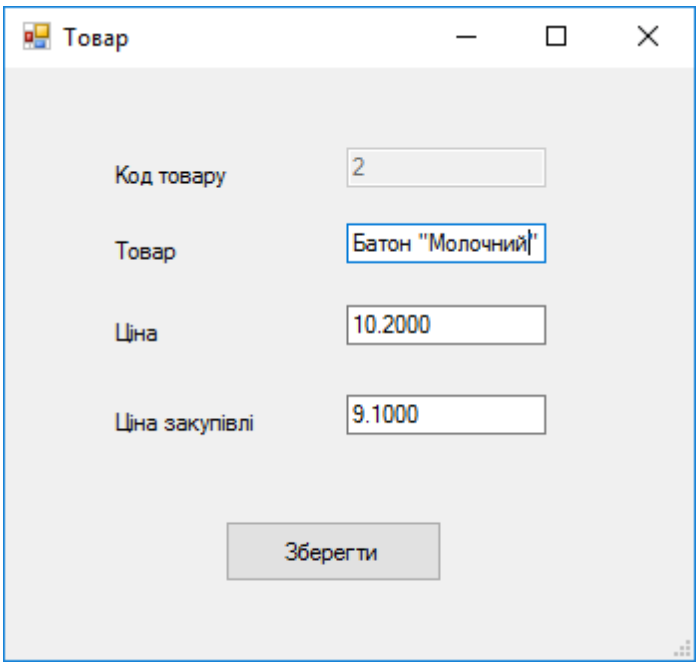


Рис. 1.4. **Форма *Товар***

1. Створення кнопкової форми застосунку

Завдання

Створіть застосунок і побудуйте в ньому кнопкову форму (рис. 1.5).

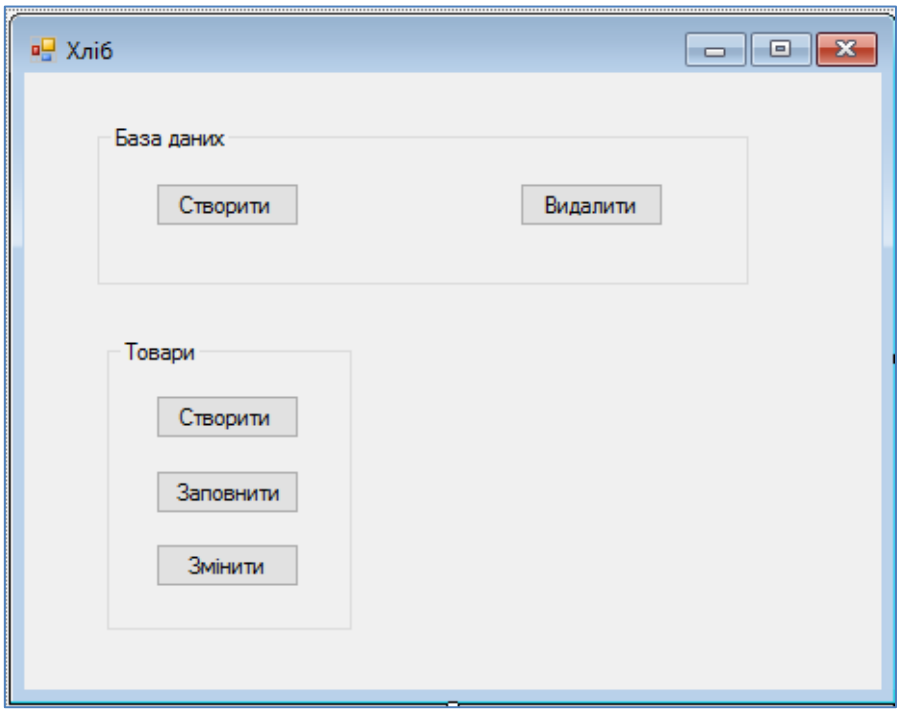


Рис. 1.5. **Кнопкова форма**

Виконання

1. Відкрийте Visual Studio.

2. Створіть проєкт Windows Forms мовою C# з ім'ям ***з'єднанийХліб*** і збережіть його у своїй папці із проєктами (рис. 1.6).

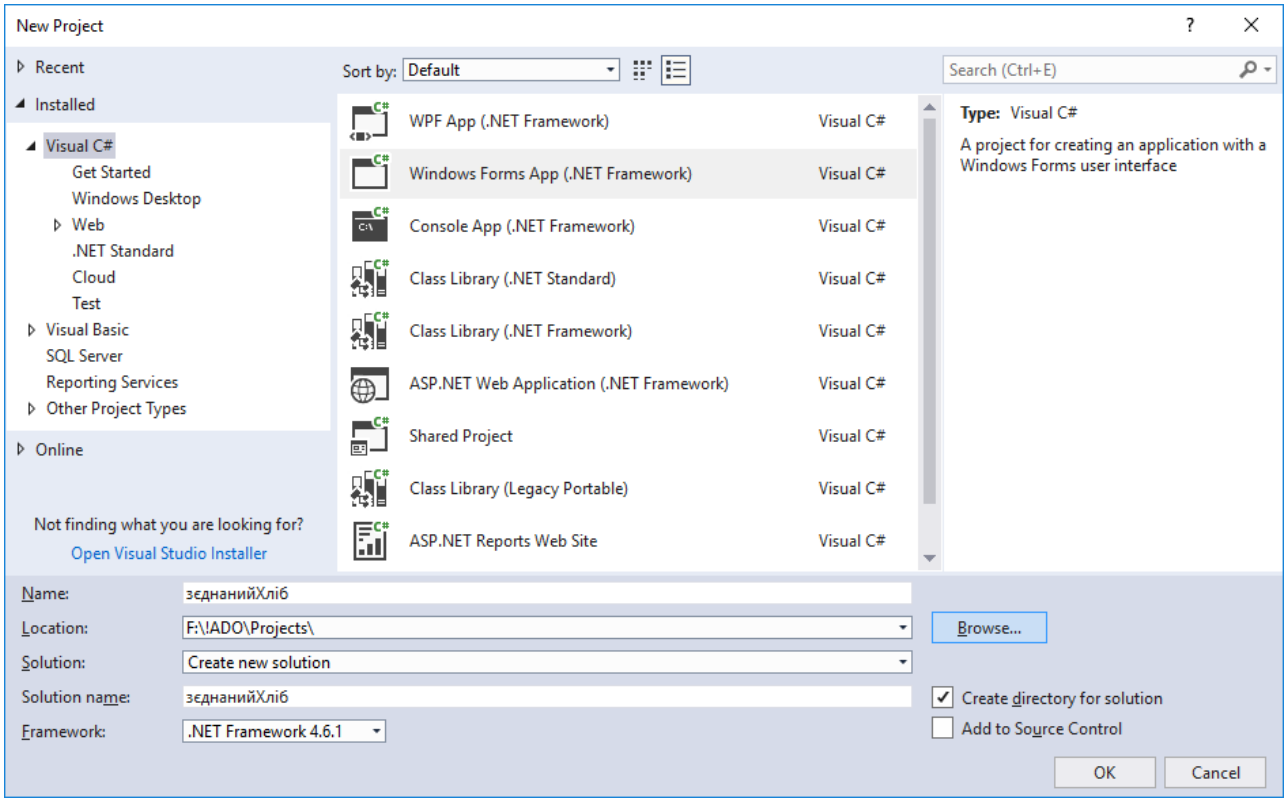


Рис. 1.6. **Створення проєкту Windows Forms**

3. У вікні **Solution Explorer** виділіть значок ***Form1*** і для файлу ***Form1.cs*** у вікні властивостей задайте ім'я файлу ***formХліб.cs***.

4. Клацніть на формі у вікні конструктора і для властивості **Text** форми задайте значення ***ХлібПрізвище***.

Примітка. Замість слова ***Прізвище*** у значенні властивості **Text** форми вставте своє прізвище. Тоді форма буде мати відповідний заголовок, наприклад, ***ХлібПетренко***.

5. Додайте на форму елемент керування **GroupBox** із панелі **Toolbox** і задайте значення ***База даних*** для його властивості **Text**.

6. Додайте дві кнопки всередину елемента ***База даних*** та установіть для них значення властивостей, наведених у табл. 1.1.

Таблиця 1.1

Властивості кнопок групи *База даних*

Кнопки	Властивості	Значення
1	Text	Створити
	Name	buttonСтворитиБД
2	Text	Видалити
	Name	buttonВидалитиБД

7. Повторіть п. 5 і 6 для групи кнопок ***Товари***, установивши значення властивостей, наведених у табл. 1.2.

Таблиця 1.2

Властивості кнопок групи *База даних*

Кнопки	Властивості	Значення
1	Text	Створити
	Name	buttonСтворитиТовари
2	Text	Заповнити
	Name	buttonЗаповнитиТовари
3	Text	Змінити
	Name	buttonЗмінитиТовари

8. Збережіть зміни, зроблені у проєкті.

У звіті з лабораторної роботи помістіть скриншот форми, аналогічний рис. 1.5.

2. Операції з базою даних загалом

Постановка завдання

Потрібно створити код, що реалізує створення і видалення бази даних ***ХлібПрізвище*** в СУБД MS SQL Server у вигляді mdf-файлу.

Примітки:

1) замість слова ***Прізвище*** у значенні імені бази даних потрібно вставити своє прізвище. Тоді база даних буде мати відповідне ім'я, наприклад, ***ХлібПетренко***;

2) в описах подальших завдань база даних буде мати узагальнене ім'я ***Хліб***.

Ідеї виконання

Створення і видалення бази даних здійснюють за допомогою відповідних SQL-скриптів. У з'єднаному середовищі їх виконують методом **ExecuteNonQuery()** класу **Command**.

Перед викликанням методу **ExecuteNonQuery()** задають рядок з'єднання, у якому вказують тип і місце розташування бази даних. Водночас в операціях із базою даних загалом (створення і видалення) у рядку з'єднання вказують тільки сервер, а в операціях із таблицями – адресу конкретної бази даних. Для доступу до цих рядків у різних оброблювачах подій їх розміщують в окремому класі ***myPublic***.

2.1. Побудова рядків з'єднання

Завдання

Створіть клас ***myPublic*** для зберігання величин, які є спільними для всього проєкту.

Виконання

1. Виконайте команду **Project – Add Class**.

2. Виберіть значок **Class** у вікні **Add New Item**, уведіть ім'я ***myPublic.cs*** і клацніть кнопку **Add**.

3. У вікні коду, що відкрилося, уведіть код тіла класу, щоб став таким:

```
class myPublic
{
  //Рядок з'єднання із сервером
  public static string stringConnServer =
    @"Data Source=(localDB)\MSSQLlocalDB;Integrated Security=True";

  //Рядок з'єднання з базою даних
  public static string stringConnDB =
    @"Data Source = (localDB)\MSSQLlocalDB; " +
    @"AttachDbFilename=Шлях\ХлібПрізвище.mdf; Integrated Security=True";
}
```

Примітки:

1) замість слова ***Шлях*** у значенні параметра **AttachDbFilename** потрібно вказати шлях до папки, у якій зберігають проєкт, наприклад,

F:\!ADO\Projects\з'єднанийХліб\з'єднанийХліб.

Після цього рядок коду для наведеного прикладу буде мати такий вигляд:

```
@" AttachDbFilename = 'F:\!ADO\Projects\з'єднанийХліб\з'єднанийХліб\Хліб.mdf')";
```

2) замість слова ***Прізвище*** у значенні параметра **AttachDbFilename** потрібно вставити своє прізвище. Тоді база даних буде мати відповідне ім'я, наприклад, ***ХлібПетренко***;

3) в описах подальших завдань база даних буде мати узагальнене ім'я ***Хліб***;

4) залежно від версії Visual Studio, ім'я сервера (параметр **Data Source**) має такі значення:

для Visual Studio 2010 – **.\SQLExpress**;

для Visual Studio 2012, 2013 – **(localDB)\v11.0**;

для вищих версій – **(localDB)\MSSQLLocalDB**.

5) у зв'язку зі значними адміністративними обмеженнями комп'ютерів в аудиторіях університету для Visual Studio 2010 (сервер **.\SQLExpress**), слід вибирати екземпляр користувача сервера, указавши параметр **User Instance=True**. У цьому разі class ***myPublic*** буде мати такий формат:

```
class myPublic
{
  //Рядок з'єднання із сервером
  public static string stringConnServer =
    @"Data Source=.\SQLExpress;Integrated Security=True" +
    "User Instance=True";

  //Рядок з'єднання з базою даних
  public static string stringConnDB =
    @"Data Source = .\SQLExpress; " +
    @"AttachDbFilename=Шлях\ХлібПрізвище.mdf; Integrated Security=True;"+
    "User Instance=True";
}
```

2.2. Створення бази даних

Завдання

Створіть базу даних ***ХлібПрізвище*** в СУБД MS SQL Server у вигляді mdf-файлу.

Виконання

1. Перейдіть у вікно конструктора форми ***Хліб***.

2. Додайте код оброблювача події "Клацання кнопки buttonСтворитиБД", щоб запустити на виконання код для створення бази даних. Для цього:

2.1. Двічі клацніть кнопку ***Створити*** у групі ***База даних***.

2.2. Установіть посилання на потрібні простори імен:

```
using System;
using System.Data;
using System.Text;
using System.Windows.Forms;
// Бібліотека постачальника даних SQL Server
using System.Data.SqlClient;
```

2.3. Уведіть оператори тіла функції-оброблювача події "Клацання кнопки buttonСтворитиБД". Вона має такий вигляд:

```
private void buttonСтворитиБД_Click(object sender, EventArgs e)
{
    // З'єднання із сервером
    SqlConnection conn = new SqlConnection();
    conn.ConnectionString = myPublic.stringConnServer;
    conn.Open();

    // Командний об'єкт створює базу даних
    SqlCommand cmd = new SqlCommand();
    cmd.Connection = conn;

    // !!!! ВАЖЛИВО !!!
    // У параметрі Filename указати повний шлях до mdf-файлу,
    // у якому зберігають базу даних
    cmd.CommandText = @"CREATE DATABASE Хліб On (NAME = Хліб_dat, " +
        @"Filename='Шлях\Хліб.mdf')";
    cmd.ExecuteNonQuery();
    MessageBox.Show("Базу даних Хліб створено", "База даних");

    conn.Close();
}
```

Примітка. Замість слова ***Шлях*** у значенні параметра **Filename** потрібно вказати шлях до папки, у якій зберігають проєкт, наприклад, ***F:\!ADO\Projects\з'єднанийХліб\з'єднанийХліб***. Після цього рядок коду буде мати такий вигляд:

```
@"Filename = 'F:\!ADO\Projects\з'єднанийХліб\з'єднанийХліб\Хліб.mdf')";
```

3. Запустіть програму на виконання, клацніть кнопку ***Створити*** у групі ***База даних*** і дочекайтеся поки з'явиться вікно повідомлення про створення бази даних (рис. 1.7). Клацніть кнопку **ОК**, а потім закрийте кнопкову форму.

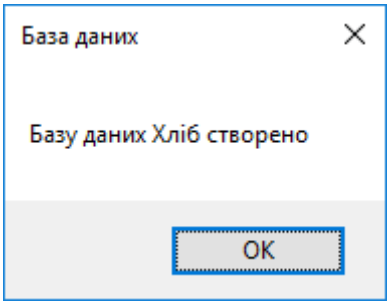


Рис. 1.7. **Вікно повідомлення про створення бази даних**

4. Перейдіть у вікно папки проєкту і переконайтеся, що там з'явилися значки двох файлів – ***Хліб.mdf*** та ***Хліб_log.ldf*** (рис. 1.8).

У звіті з лабораторної роботи подайте код оброблювача події "Клацання кнопки buttonСтворитиБД" та скриншот папки проєкту, аналогічний рис. 1.8.

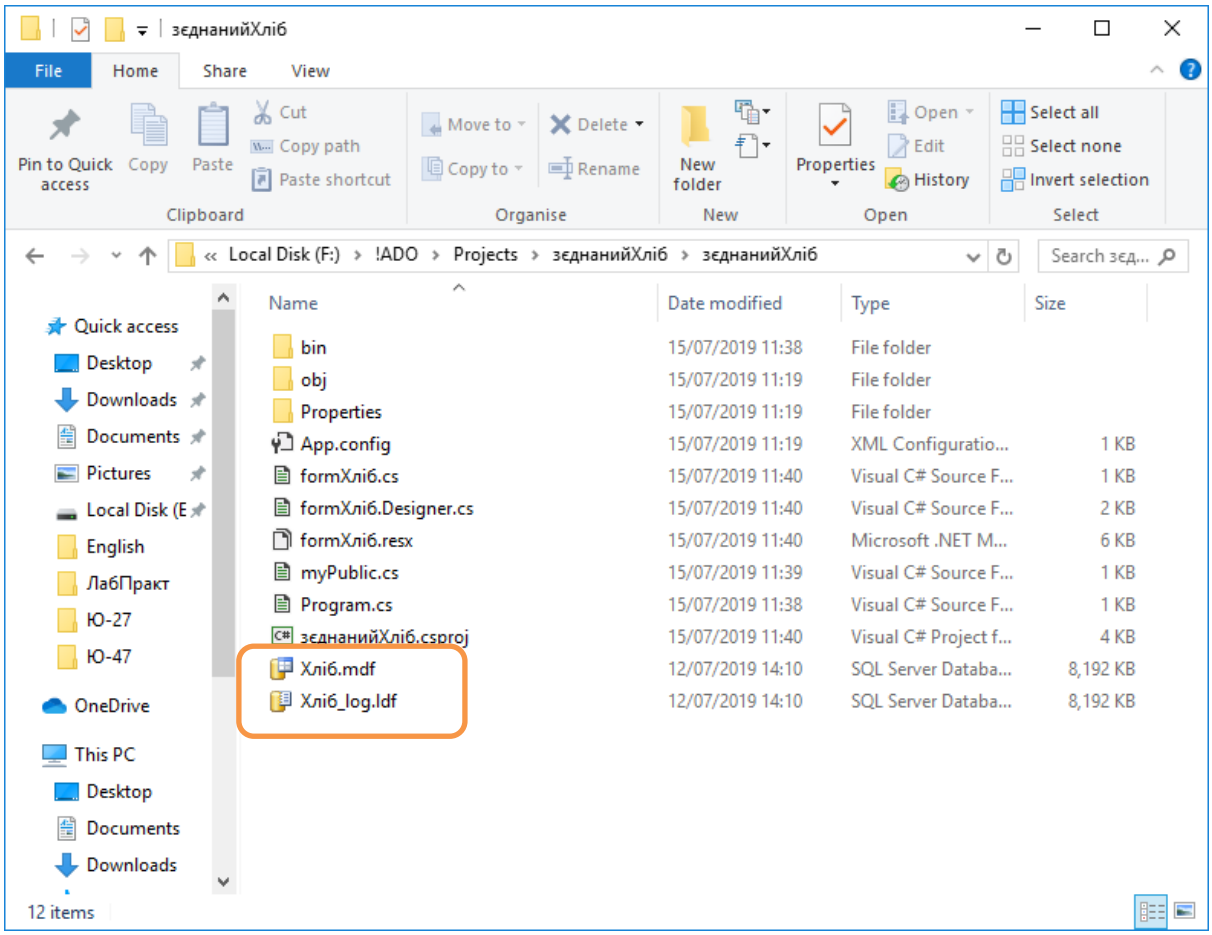


Рис. 1.8. **Файли бази даних у вікні папки проєкту**

2.3. Видалення бази даних

Завдання

Видаліть базу даних ***Хліб*** у СУБД MS SQL Server, що була створена в завд. 2.

Виконання

1. Перейдіть у вікно конструктора форми ***Хліб***.

2. Додайте код оброблювача події "Клацання кнопки buttonВидалитиБД", щоб запустити на виконання код для видалення бази даних. Для цього:

2.1. Двічі клацніть кнопку ***Видалити*** у групі ***База даних***.

2.3. Уведіть оператори тіла функції-оброблювача події "Клацання кнопки buttonВидалитиБД". Вона має такий вигляд:

```
private void buttonВидалитиБД_Click(object sender, EventArgs e)
{
    // Перед видаленням бази даних необхідно перевірити
    // у вікні Server Explorer чи встановлено підключення бази даних
    // до сервера. Якщо так, то видалити підключення
    // за допомогою контекстового меню

    // З'єднання
    SqlConnection conn = new SqlConnection();
    // Указуємо в рядку з'єднання лише сервер
    conn.ConnectionString = myPublic.stringConnServer;
    conn.Open();
----
    // Командний об'єкт видаляє базу даних
    SqlCommand cmd = new SqlCommand();
    cmd.Connection = conn;
    cmd.CommandText = @"DROP DATABASE Хліб";
    cmd.ExecuteNonQuery();
    MessageBox.Show("Базу даних Хліб видалено", "База даних");

    conn.Close();
}
```

3. Запустіть програму на виконання, клацніть кнопку ***Видалити*** у групі ***База даних*** і дочекайтеся поки з'явиться вікно повідомлення про видалення бази даних. Клацніть кнопку **ОК**.

4. Перейдіть у вікно папки проєкту і переконайтеся, що там зникли значки двох файлів – ***Хліб.mdf*** та ***Хліб_log.ldf***.

5. Повторно створіть базу даних ***Хліб***, клацнувши кнопку ***Створити*** у групі ***База даних*** програми, що перебуває в режимі виконання.

6. Зупиніть виконання програми.

У звіті з лабораторної роботи подайте код оброблювача події "Клацання кнопки buttonВидалитиБД".

2.4. З'єднання з базою даних візуальними засобами

Завдання

Використовуючи візуальні засоби Visual Studio, установіть з'єднання бази даних ***Хліб*** із СУБД MS SQL Server.

Виконання

1. Клацніть правою клавішею мишки на значок **Data Connections** у вікні **Server Explorer** і виберіть команду **Add Connection** у контекстовому меню (рис. 1.9).

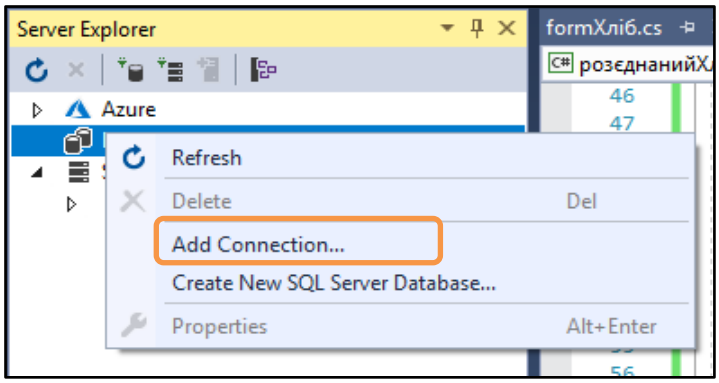


Рис. 1.9. **Команда Add Connection у вікні Server Explorer**

2. Виконайте такі операції у вікні **Add Connection**.

2.1. Перевірте, чи встановлено значення **Microsoft SQL Server Database File (SqlClient)** у полі **Data source**. Якщо ні, то скористайтеся кнопкою **Change** (рис. 1.10).

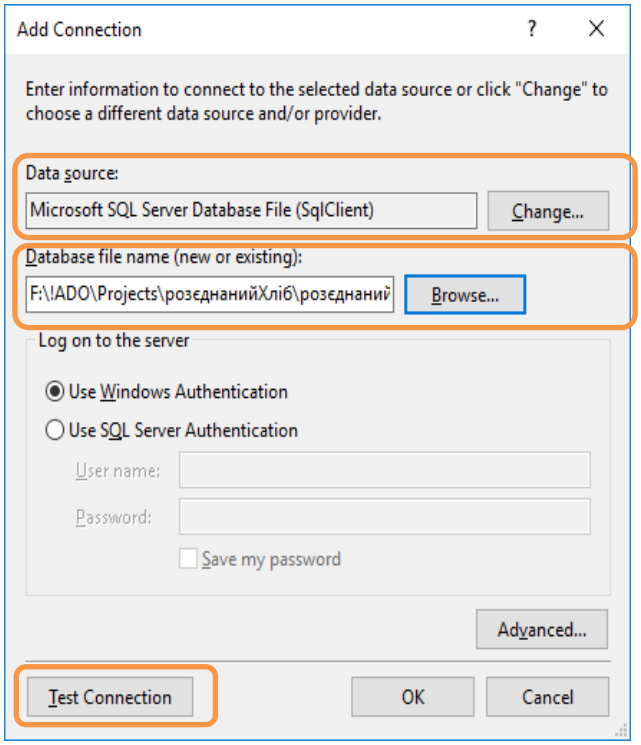


Рис. 1.10. **Вікно Add Connection**

2.2. Установіть шлях до файлу бази даних ***Хліб.mdf***, що перебуває в папці проєкту. Для цього скористайтеся кнопкою **Browse** (див. рис. 1.10).

2.3. Перевірте з'єднання за допомогою кнопки **Test Connection** (див. рис. 1.10).

2.4. Клацніть кнопку **ОК**.

У вікні **Server Explorer** з'явився значок бази даних ***Хліб***. Переконайтеся, що в базі даних ще немає жодної таблиці, розкривши вузли **Хліб – Tables** (рис. 1.11).

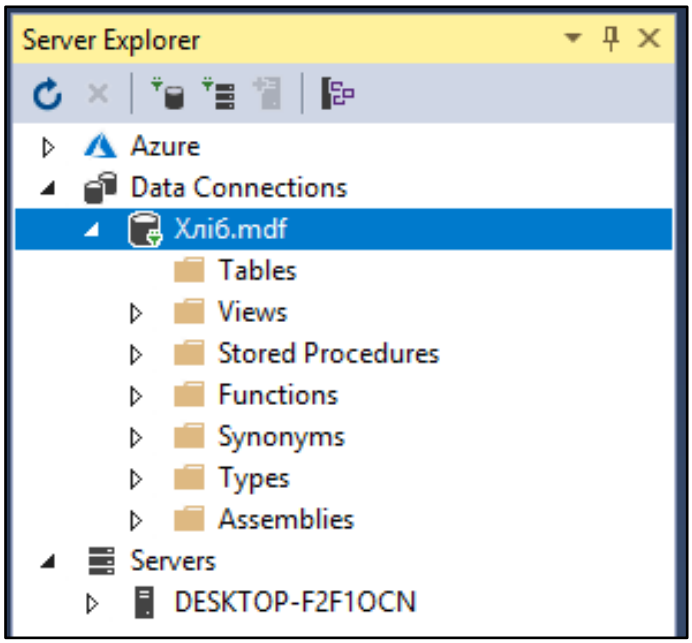


Рис. 1.11. **Вузли Хліб – Tables у вікні Server Explorer**

У звіті з лабораторної роботи подайте скриншот порожньої бази даних, аналогічний рис. 1.11.

3. Створення і заповнення таблиці *Товари*

Постановка завдання

Потрібно побудувати код, що реалізує створення і заповнення таблиці ***Товари*** в базі даних ***Хліб***.

Ідеї виконання

Створення таблиці бази даних реалізують оператором SQL CREATE TABLE, а заповнення даними – оператором INSERT. У з'єднаному середовищі їх виконують методом **ExecuteNonQuery()** класу **Command**.

Перед викликанням методу **ExecuteNonQuery()** задають рядок з'єднання, у якому вказують тип і місце розташування бази даних. Він перебуває у класі ***myPublic***.

3.1. Створення таблиці програмними засобами

Завдання

Побудуйте код, що реалізує створення таблиці ***Товари***.

Виконання

1. Перейдіть у вікно конструктора форми ***Хліб***.

2. Додайте код оброблювача події "Клацання кнопки buttonСтворитиТовари" у групі ***Товари***, щоб запустити на виконання код для створення таблиці ***Товари***:

```
private void buttonСтворитиТовари_Click(object sender, EventArgs e)
{
    // З'єднання
    SqlConnection conn = new SqlConnection();
    conn.ConnectionString = myPublic.stringConnDB;
    conn.Open();

    //Командний об'єкт створює таблицю Товари
    SqlCommand cmd = new SqlCommand();
    cmd.Connection = conn;
    cmd.CommandText = "CREATE TABLE Товари " +
        "(Код_товару int IDENTITY (1,1) Primary Key, " +
        "Товар NVarChar(25), " +
        "Ціна Money, " +
        "Ціна_закупівлі Money)";
    cmd.ExecuteNonQuery();
    MessageBox.Show("Таблицю Товари створено", "Таблиця Товари");

    conn.Close();
}
```

3. Запустіть програму на виконання, клацніть кнопку ***Створити*** у групі ***Товари*** й дочекайтеся поки з'явиться вікно повідомлення про створення таблиці ***Товари***. Клацніть кнопку **ОК**, а потім закрийте кнопкову форму.

4. Перейдіть у вікно **Server Explorer**, обновіть зображення бази даних ***Хліб*** і переконайтеся, що в базі даних з'явилася таблиця ***Товари***, розкривши вузли **Хліб – Tables – Товари** (рис. 1.12).

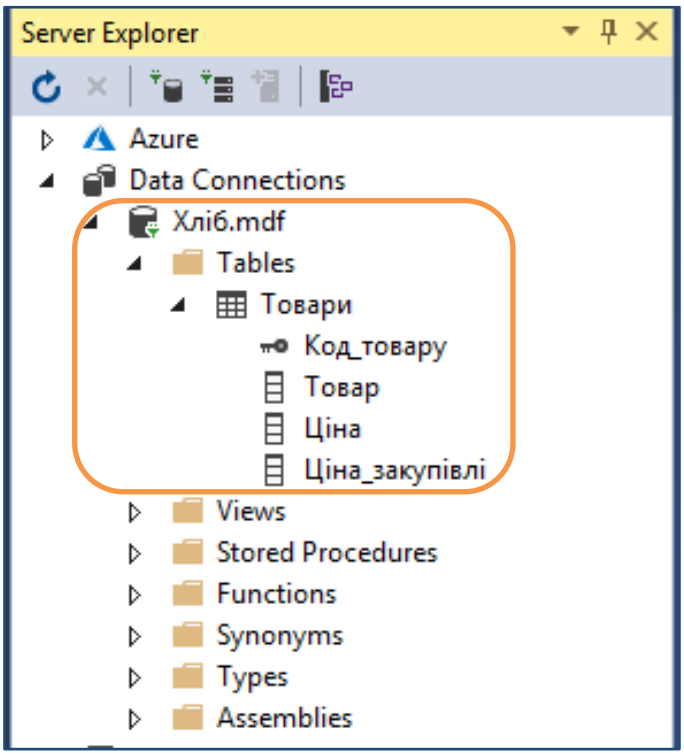


Рис. 1.12. **Таблиця *Товари* в базі даних *Хліб***

У звіті з лабораторної роботи подайте код оброблювача події "Клацання кнопки buttonСтворитиТовари" та скриншот бази даних із таблицею ***Товари***, аналогічний рис. 1.12.

3.2. Заповнення таблиці програмними засобами

Завдання

Побудуйте код, що реалізує заповнення даними таблиці ***Товари***.

Виконання

1. Перейдіть у вікно конструктора форми ***Хліб***.

2. Додайте код оброблювача події "Клацання кнопки buttonЗаповнитиТовари" у групі ***Товари***, щоб запустити на виконання код для заповнення даними таблиці ***Товари***:

```
private void buttonЗаповнитиТовари_Click(object sender, EventArgs e)
{
    // З'єднання
    SqlConnection conn = new SqlConnection();
    conn.ConnectionString = myPublic.stringConnDB;
    conn.Open();

    // Командний об'єкт заповнює таблицю Товари
    SqlCommand cmd = new SqlCommand();
    cmd.Connection = conn;
    cmd.CommandText = @"INSERT Товари VALUES" +
        @"(N'Хліб ""Український""',10.50,9.30), " +
        @"(N'Батон ""Молочний""',10.20,9.10), " +
        @"(N'Булка з маком',9.80,8.65)";
    cmd.ExecuteNonQuery();
    MessageBox.Show("Дані додано ", "Таблиця Товари");

    conn.Close();
}
```

3. Запустіть програму на виконання, клацніть кнопку ***Заповнити*** у групі ***Товари*** й дочекайтеся поки з'явиться вікно повідомлення про додавання даних до таблиці ***Товари***. Клацніть кнопку **ОК**, а потім закрийте кнопкову форму.

4. Перейдіть у вікно **Server Explorer**, обновіть зображення бази даних ***Хліб*** і переконайтеся, що в базі даних з'явилися дані в таблиці ***Товари***, вибравши з контекстового меню таблиці команду **Show Table Data** (рис. 1.13).

	Код_товару	Товар	Ціна	Ціна_закупівлі
▶	1	Хліб "Український"	10.5000	9.3000
	2	Батон "Молочний"	10.2000	9.1000
	3	Булка з маком	9.8000	8.6500
*	NULL	NULL	NULL	NULL

Рис. 1.13. **Дані таблиці *Товари***

У звіті з лабораторної роботи подайте код оброблювача події "Клацання кнопки buttonЗаповнитиТовари" та скриншот таблиці ***Товари*** з даними, аналогічний рис. 1.13.

4. Зміна даних таблиці *Товари*

Постановка завдання

Зміну значень записів таблиці ***Товари*** можна здійснити за допомогою двох форм (рис. 1.14 і 1.15). На першій формі виводять усі дані таблиці ***Товари*** та вибирають запис, у якому потрібно виконати зміни. Після клацання кнопки ***Змінити*** відкривають форму ***Товар***. На ній змінюють дані й дають команду на збереження зміненого запису в базі даних. Після повернення на форму ***Товари*** в ній відображають попередні значення даних. Цей недолік ліквідують під час виконання завд. 2 з п. "Завдання для самостійного виконання".

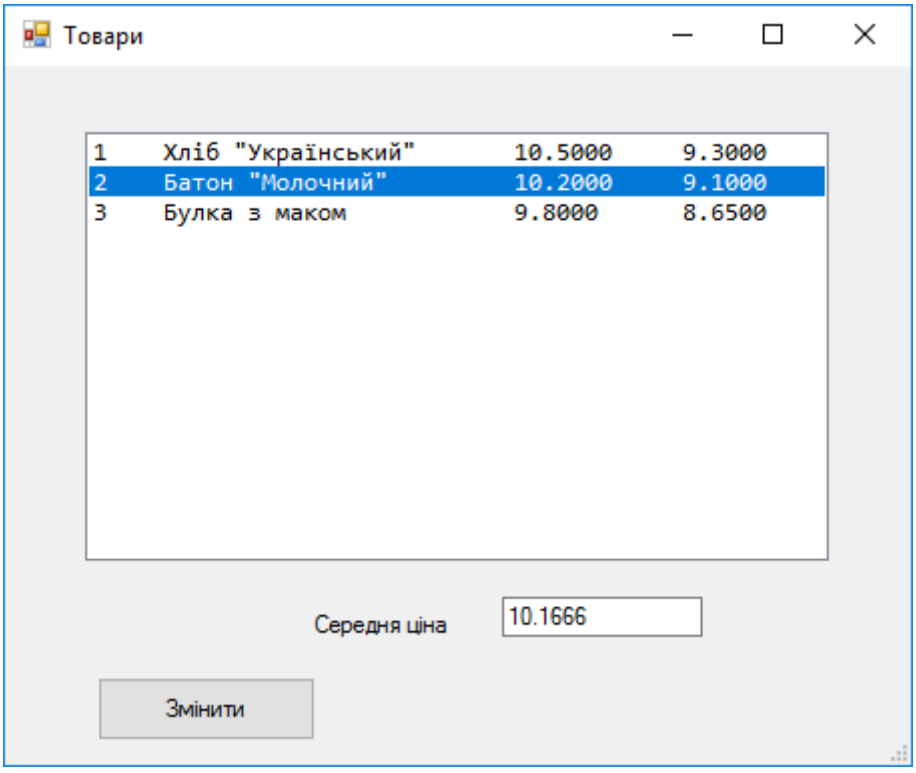


Рис. 1.14. **Форма для вибору запису, який треба змінити**

Завдання виконують у 2 етапи:

1. Вибір даних із бази.
2. Зміна даних у базі.

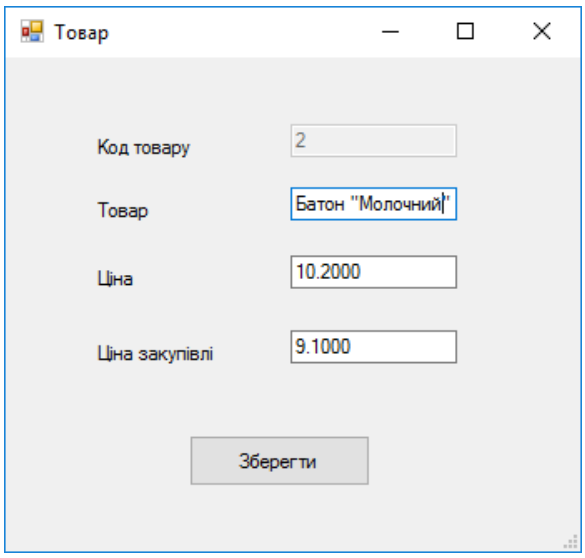


Рис. 1.15. **Форма зміни даних у записі**

4.1. Вибір даних із бази

Порядок виконання

На цьому етапі виконують такі операції:

1) отримання набору даних методом ExecuteReader;

2) отримання підсумкових даних методом ExecuteScalar.

Завдання 1

Створіть форму ***Товари***, на якій відображено дані однойменної таблиці (рис. 1.16).

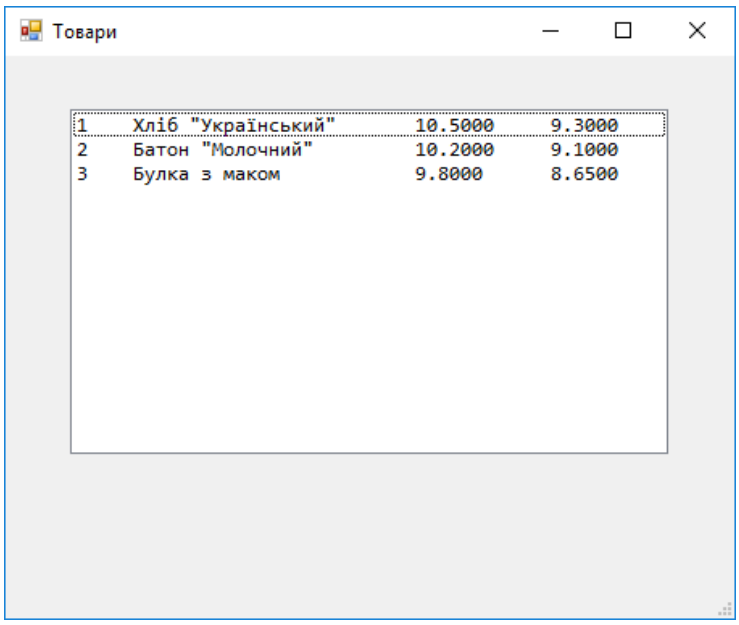


Рис. 1.16. **Список із записами таблиці *Товари***

Ідеї виконання

Для відображення даних однієї таблиці доцільно використати елемент керування *Список* (*ListBox*). У ньому відразу відображено багато записів на відміну від елемента *Поле зі списком* (*ComboBox*). За розмірами список має більш компактний вигляд, ніж елемент керування *DataGridView*.

Отримання даних таблиці ***Товари*** з бази даних у мережевий кеш комп'ютера клієнта здійснюють методом **ExecuteReader()**. Потім у циклі обробляють кожен запис кеша в такій послідовності:

1) вибирають черговий запис, починаючи з першого;

2) об'єднують значення окремих полів в один рядок;

3) додають рядок як елемент у список.

Оскільки у списку відображено дані в чотирьох стовпцях, тому, щоб забезпечити чітке подання стовпців, використовують моноширинний шрифт. У цьому разі встановлюють шрифт Consolas.

Виконання

1. Додайте форму ***Товари*** до проєкту. Для цього:

1.1. Виконайте команду **Project – Add Windows Form**.

1.2. Виберіть значок **Windows Form** у вікні **Add New Item**, уведіть ім'я ***formТовари*** і клацніть кнопку **Add**.

1.3. Для властивості **Text** форми задайте значення ***Товари***.

1.4. Перейдіть у вікно коду форми й установіть посилання на потрібні простори імен:

```
using System;
using System.Data;
using System.Text;
using System.Windows.Forms;
using System.Data.SqlClient;
```

2. Поверніться у вікно конструктора форми, додайте елемент управління **ListBox** із панелі елементів **Toolbox** й установіть значення властивостей для елемента **ListBox** (табл. 1.3). Форму з новими значеннями властивостей елемента **ListBox** подано на рис. 1.17.

Таблиця 1.3

Властивості елемента ListBox

Властивості	Значення
Name	ListBoxТовари
Font.Name	Consolas
Font.Size	10

3. Додайте код оброблювача події "Завантаження форми", щоб відображати у списку всі записи таблиці ***Товари*** під час відкриття форми. Для цього виконайте таке:

3.1. Двічі клацніть у вільному місці форми.

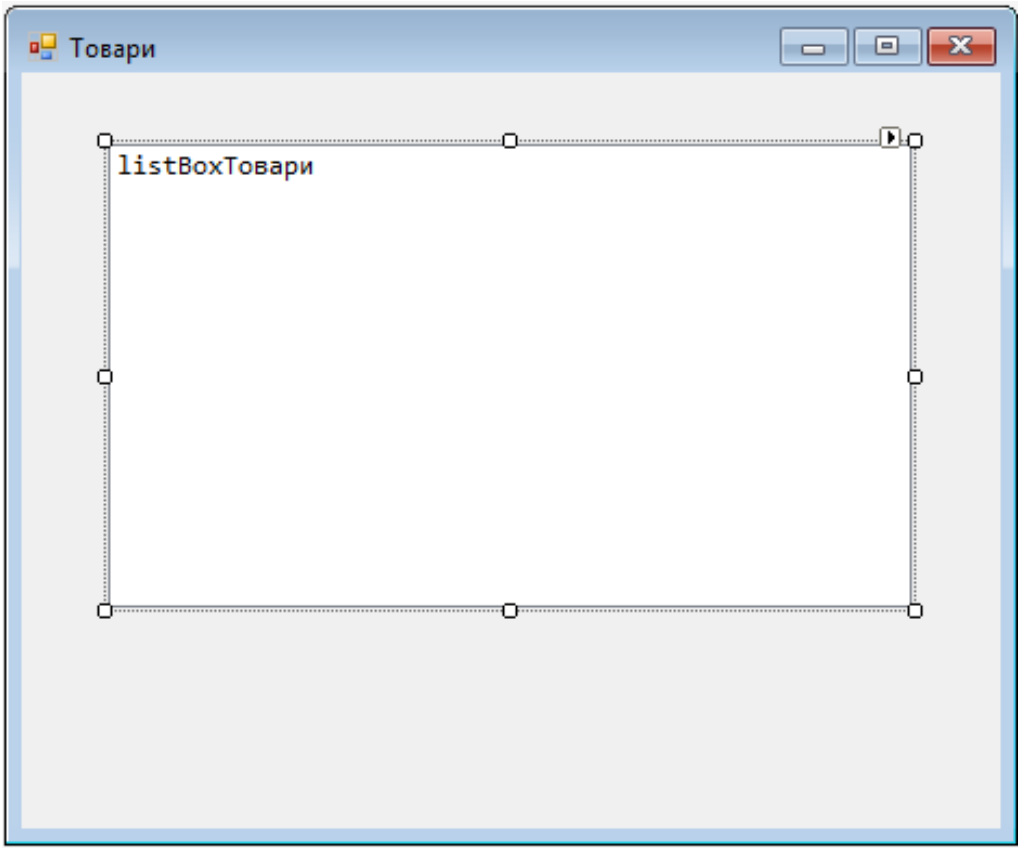


Рис. 1.17. **Нові значення властивостей елемента ListBox**

3.2. Уведіть оператори тіла функції-оброблювача події. Вона має такий вигляд:

```
private void formТовари_Load(object sender, EventArgs e)
{
    // З'єднання
    SqlConnection conn = new SqlConnection();
    conn.ConnectionString = myPublic.stringConnDB;
    conn.Open();

    // Командний об'єкт
    SqlCommand cmd = new SqlCommand();
    cmd.Connection = conn;

    // Список товарів

    // Запит для отримання набору товарів
    cmd.CommandText = "SELECT * FROM Товари";

    //Створюємо DataReader
    SqlDataReader dr = cmd.ExecuteReader();

    //Виводимо набір записів у список
    int i; //Номер стовпця
    int[] len = new int[4] { 5, 25, 12, 12 }; //Ширина стовпців

    //Проходимо по записах
    while (dr.Read())
    {
        string stringList = "";
        for (i = 0; i < dr.FieldCount; i++)
        {
            // Накопичуємо запис із полів
            // Поле відповідної довжини притискаємо вліво
            stringList += dr[i].ToString().PadRight(len[i]);
        }
        listBoxТовари.Items.Add(stringList);
    }

    dr.Close();
}
```

4. Перейдіть на форму ***Хліб***, двічі клацніть кнопку ***Змінити*** у групі ***Товари*** та введіть оператори тіла функції-оброблювача події "Клацання кнопки buttonЗмінитиТовари". Вона має такий вигляд:

```
private void buttonТовари_Click(object sender, EventArgs e)
{
    formТовари вікноТовари = new formТовари();
    вікноТовари.ShowDialog();
}
```

5. Збережіть зміни у проєкті й запустіть програму на виконання та перевірте функціональність форми ***Товари***.

У звіті з лабораторної роботи подайте код оброблювача події "Завантаження форми Товари" та скриншот форми ***Товари*** в режимі виконання програми, аналогічний рис. 1.16.

Завдання 2

Додайте на форму напис і текстове поле, у якому відображено середню ціну всіх товарів (рис. 1.18).

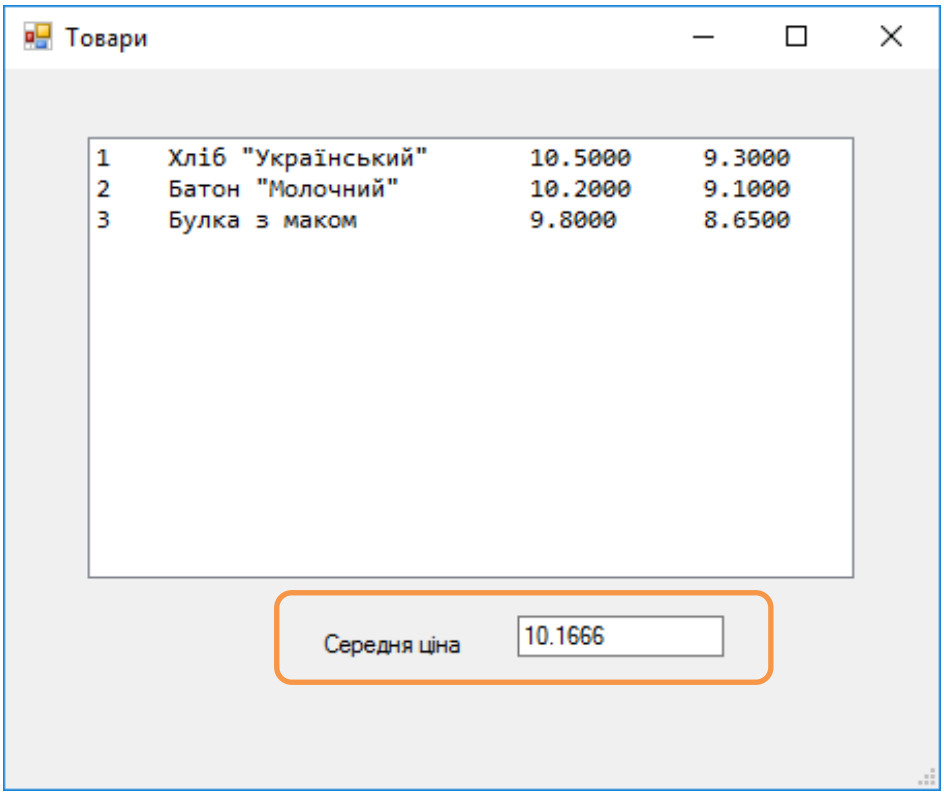


Рис. 1.18. **Середня ціна на формі *Товари***

Виконання

1. Додайте на форму елементи керування Label і TextBox, розмістивши їх під списком.

2. Установіть значення властивостей для нових елементів керування, наведених у табл. 1.4.

Таблиця 1.4

Властивості нових елементів керування

Елементи	Властивості	Значення
Label	Text	Середня ціна
TextBox	Name	textBoxСередня_ціна

3. Щоб відображати в текстовому полі середню ціну всіх товарів, додайте в код оброблювача події "Завантаження форми" такі оператори, розмістивши їх між викликом методу **dr.Close();** і закриваючою фігурною дужкою **}**.

```
    // Середня ціна

    // Запит для обчислення середньої ціни
   cmd.CommandText  = "SELECT AVG(Ціна) FROM Товари";
    // Отримуємо й відображаємо середню ціну
textBoxСередня_ціна.Text = cmd.ExecuteScalar().ToString();
    //Закриваємо з'єднання
    conn.Close();
```

4. Збережіть зміни у проєкті й запустіть програму на виконання та перевірте функціональність форми ***Товари***.

У звіті з лабораторної роботи подайте скриншот форми ***Товари*** в режимі виконання програми, аналогічний рис. 1.18. Опишіть, у чому полягає відмінність в отриманні даних із бази в завд. 1 і 2.

4.2. Зміни даних у базі

Завдання та порядок виконання

У наявний застосунок додайте функціональність для зміни даних із вибраного товару.

Вибір запису роблять у списку на формі ***Товари***, а потім клацають кнопку ***Змінити***, щоб викликати форму ***Товар***.

Під час завантаження форми ***Товар*** вона заповнюється даними з вибраного у списку товару (рис. 1.19). Тут їх можна змінити.

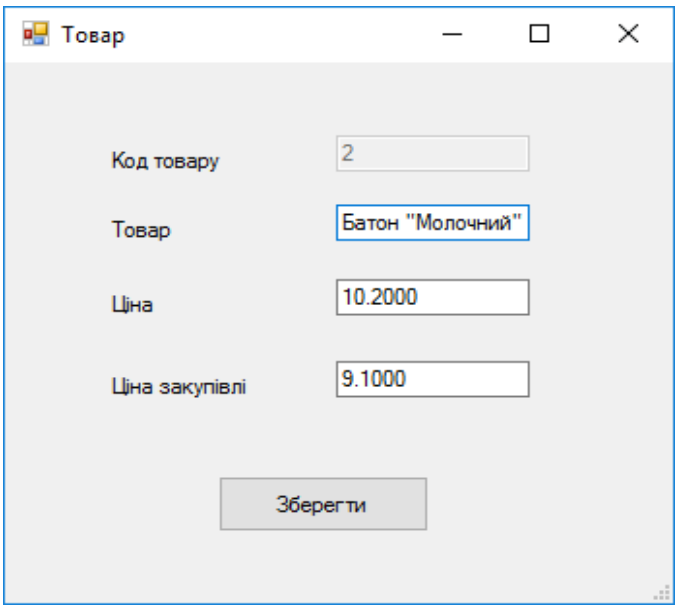


Рис. 1.19. **Форма *Товар* зі змінюваними даними**

Оновлення даних у базі відбувається після клацання кнопки ***Зберегти***. На цьому етапі виконують такі операції:

1) вибір змінюваного запису;
2) зміну даних на формі;
3) оновлення даних у базі.

Завдання 1

Додайте в застосунок форму ***Товар***, у якій будуть змінюватися дані з вибраного товару (рис. 1.20).

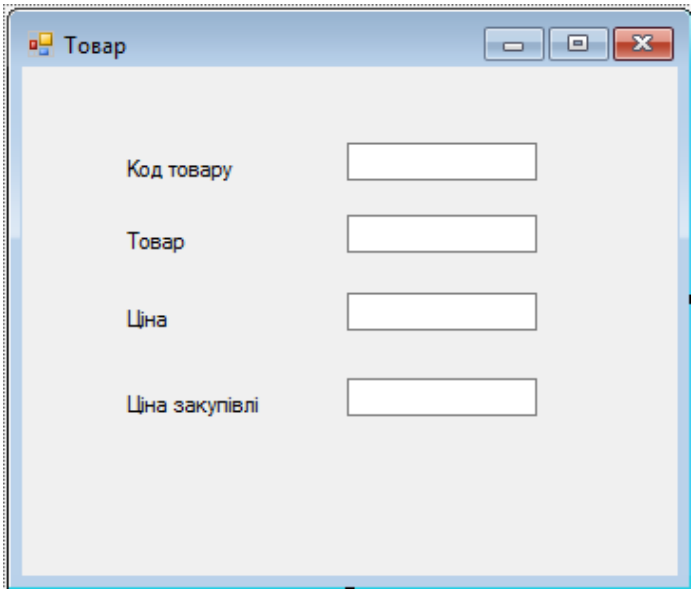


Рис. 1.20. **Форма *Товар* для зміни даних**

Виконання

1. Додайте в застосунок нову форму з ім'ям ***formТовар*** і заголовком ***Товар***.

2. Додайте чотири написи й текстові поля на форму для відображення значень полів вибраного запису (рис. 1.21).

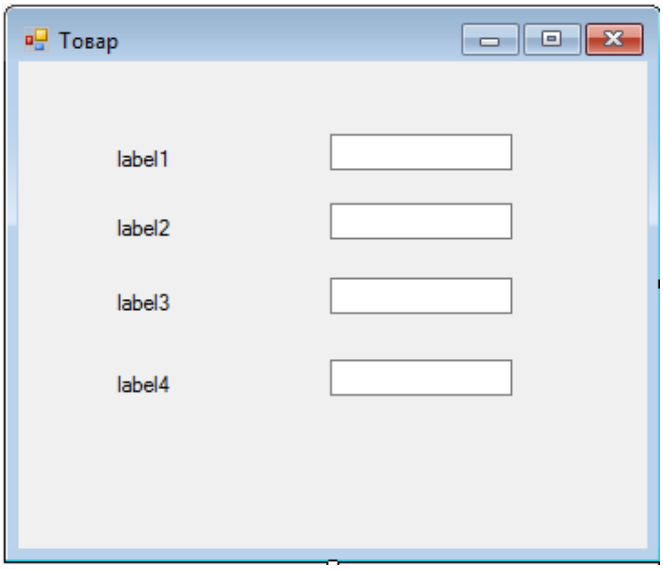


Рис. 1.21. **Початковий вигляд форми *Товар***

3. Установіть для елементів керування значення властивостей, наведених у табл. 1.5.

Таблиця 1.5

Властивості елементів керування

№ п/п	Написи (властивості Text)	Текстові поля (властивості Name)
1	Код товару	textBoxКод_товару
2	Товар	textBoxТовар
3	Ціна	textBoxЦіна
4	Ціна закупівлі	textBoxЦіна_закупівлі

4. У текстовому полі ***textBoxКод_товару*** додатково установіть значення ***False*** для властивості **Enabled**.

5. Збережіть зміни, зроблені у проєкті.

Завдання 2

Додайте на форму ***Товари*** кнопку ***Змінити***, яка викликає форму ***Товар*** (рис. 1.22).

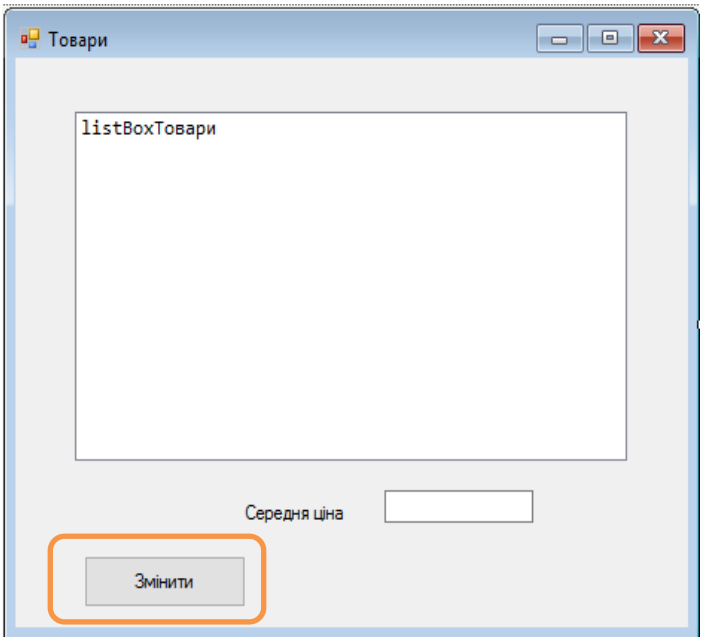


Рис. 1.22. **Кнопка *Змінити* на формі *Товари***

Виконання

1. Поверніться у вікно конструктора форми ***Товари***, додайте на неї кнопку й установіть значення властивостей, наведених у табл. 1.6.

Таблиця 1.6

Властивості елементів керування

Властивості	Значення
Text	Змінити
Name	buttonЗмінити

2. Додайте код оброблювача події "Клацання кнопки Змінити".

```
private void buttonЗмінити_Click(object sender, EventArgs e)
{
    //Формуємо ключ вибраного товару й відкриваємо форму для змін
    if (listBoxТовари.SelectedItem != null) // Чи вибрано товар?
    {
        formТовар вікноТовар = new formТовар();
        вікноТовар.ShowDialog();
    }
    else //Якщо не вибрано товар
    {
        MessageBox.Show("Виберіть товар", "Зміни",
            MessageBoxButtons.OK, MessageBoxIcon.Stop);
    }
}
```

3. Збережіть зміни, зроблені у проєкті.

4. Перевірте функціональність кнопки ***Змінити*** – спочатку без вибору товару у списку, а потім – із його вибором. В останньому разі має відкритися порожня форма ***Товар***.

Завдання 3

Відобразіть на формі ***Товар*** дані про товар, який вибрано у списку на формі ***Товари***.

Ідеї виконання

Код товару, вибраного на формі ***Товари***, передають на форму ***Товар***. Тут за цим кодом читають дані з бази та відображають на формі.

Щоб передати значення коду товару між формами, його записують у змінну з атрибутами **public static** у класі ***myPublic***, тобто він є доступним на обох формах.

Виконання

1. Щоб можна було передати код вибраного товару з форми ***Товари*** на форму ***Товар***, відкрийте вікно класу ***myPublic*** і додайте опис змінної ***Код_товару***. Після цього код класу буде таким:

```
class myPublic
{
  // Рядок з'єднання із сервером
  public static string stringConnServer =
    @"Data Source=(localDB)\MSSQLlocalDB;Integrated Security=True";

  // Рядок з'єднання з базою даних
  public static string stringConnDB =
  @"Data Source = @"Data Source=(localDB)\MSSQLlocalDB; " +
  @"AttachDbFilename=Шлях\ХлібПрізвище.mdf; Integrated Security=True";

  // Код вибраного товару
  public static string Код_товару;
}
```

Примітки:

1) код вибраного товару в цьому разі має тип string, оскільки значення змінної використовують для формування рядка запиту на формі ***Товар***;

2) для комп'ютерів в аудиторіях університету з Visual Studio 2010 (сервер **.\SQLExpress**) class ***myPublic*** буде мати такий формат:

```
class myPublic
{
  //Рядок з'єднання із сервером
  public static string stringConnServer =
    @"Data Source=.\SQLExpress;Integrated Security=True" +
    "User Instance=True";

  //Рядок з'єднання з базою даних
  public static string stringConnDB =
    @"Data Source = @"Data Source=.\SQLExpress; " +
    @"AttachDbFilename=Шлях\ХлібПрізвище.mdf; Integrated Security=True;"+
    "User Instance=True";

  // Код вибраного товару
  public static string Код_товару;
}
```

2. Для отримання значення коду вибраного товару використовують такий оператор:

```
// Отримуємо код товару, вибраного у списку
myPublic.Код_товару =
    listBoxТовари.SelectedItem.ToString().Substring(0, 3).Trim();
```

Уставте його як перший рядок гілки if в оброблювачі події "Клацання кнопки Змінити" у формі ***Товари***. Після цього код оброблювача події буде таким:

```
private void buttonЗмінити_Click(object sender, EventArgs e)
{

  if (listBoxТовари.SelectedItem != null) // Чи вибрано товар?
  {
    // Дії, якщо товар вибрано:

    //Формуємо ключ вибраного товару й відкриваємо форму для змін
    myPublic.Код_товару =
    listBoxТовари.SelectedItem.ToString().Substring(0, 3).Trim();

    // Відкриваємо вікно Товар (дані тимчасово не відображаємо)
    formТовар вікноТовар = new formТовар();
    вікноТовар.ShowDialog();
  }
  else //Якщо не вибрано товар
  {
    // Виводимо нагадування про необхідність вибрати товар у списку
    MessageBox.Show("Виберіть товар", "Зміни",
        MessageBoxButtons.OK, MessageBoxIcon.Stop);
  }
}
```

3. Щоб відобразити на формі ***Товар*** дані про товар, який вибрано у списку на формі ***Товари***, виконайте таке:

3.1. Перейдіть у вікно коду форми ***Товар*** й установіть посилання на потрібні простори імен:

```
using System;
using System.Data;
using System.Text;
using System.Windows.Forms;
using System.Data.SqlClient;
```

3.2. Додайте код оброблювача події "Завантаження форми":

```
private void formTовар_Load(object sender, EventArgs e)
{
    // Установлюємо з'єднання з базою даних
    SqlConnection conn = new SqlConnection();
    conn.ConnectionString = myPublic.stringConnDB;
    conn.Open();

    // Створюємо командний об'єкт
    // і налаштовуємо на з'єднання
    SqlCommand cmd = new SqlCommand();
    cmd.Connection = conn;

    // Дані про вибраний товар

    // Запит для отримання даних про вибраний товар
    cmd.CommandText = "SELECT * FROM Товари Where Код_товару=" +
        myPublic.Код_товару;

    // Створюємо DataReader
    // Після закриття dr автоматично закриється conn
    SqlDataReader dr =
    cmd.ExecuteReader(CommandBehavior.CloseConnection);

    // Читаємо вибраний запис
    dr.Read();

    // Відображаємо поля запису на формі
    textBoxКод_товару.Text = dr[0].ToString();
    textBoxТовар.Text = dr[1].ToString();
    textBoxЦіна.Text = dr[2].ToString();
    textBoxЦіна_закупівлі.Text = dr[3].ToString();

    // Закриваємо DataReader і з'єднання
    dr.Close();
}
```

4. Збережіть зміни, зроблені у проєкті, та перевірте функціональність кнопки ***Змінити***.

У звіті з лабораторної роботи подайте скриншоти форм ***Товари*** й ***Товар*** із даними в режимі виконання програми. Опишіть механізм передавання коду вибраного товару з форми ***Товари*** на форму ***Товар*** та які класи використовують для реалізації цієї операції. Опишіть також, які ще механізми обміну даними між формами вам відомі. Для цього доцільно скористатися пошуковою системою Google. Порівняйте їх за складністю реалізації й ефективністю виконання.

Завдання 4

Обновіть дані в базі, відповідно до змін, зроблених на формі ***Товар*** (рис. 1.23).

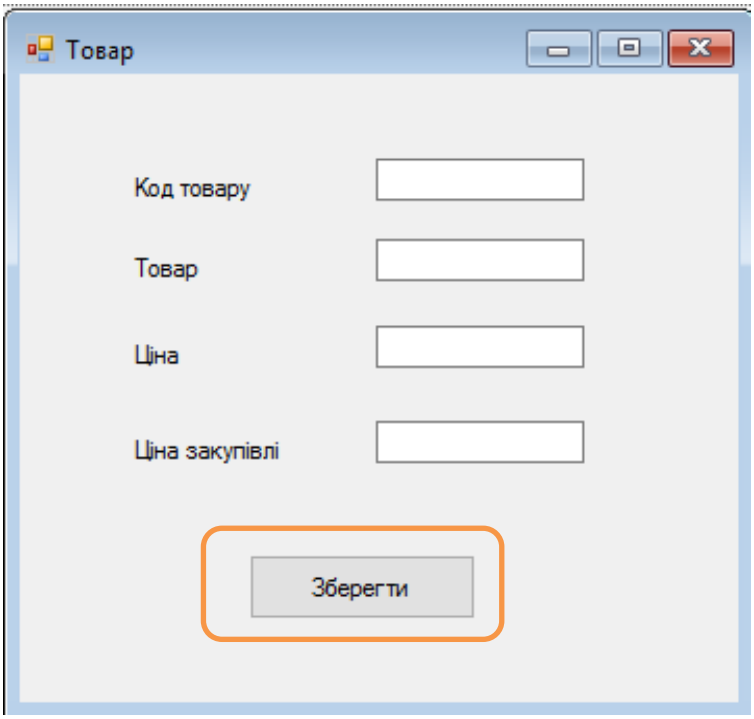


Рис. 1.23. **Кнопка *Зберегти* на формі *Товар***

Виконання

1. Перейдіть на форму ***Товар***, додайте на неї кнопку й установіть значення властивостей, наведених у табл. 1.7.

Таблиця 1.7

Властивості кнопки

Властивості	Значення
Text	Зберегти
Name	buttonЗберегти

2. Додайте код оброблювача події "Клацання кнопки ***Зберегти***".

```
private void buttonЗберегти_Click(object sender, EventArgs e)
{
    // Установлюємо з'єднання з базою даних
    SqlConnection conn = new SqlConnection();
    conn.ConnectionString = myPublic.connString;
    conn.Open();

    // Створюємо командний об'єкт
    SqlCommand cmd = new SqlCommand();
    cmd.Connection = conn;
    cmd.CommandType = CommandType.Text; // За замовчуванням

    // Формуємо запит на зміну даних у базі
    cmd.CommandText  = "UPDATE Товари SET Товар = '" +
        textBoxТовар.Text + "'" +
        ", Ціна = " + textBoxЦіна.Text.Replace(",", ".") +
        ", Ціна_закупівлі = " + textBoxЦіна_закупівлі.Text.Replace(",",
        ".")+
        " WHERE Код_товару=" + textBoxКод_товару.Text;

    // Оновлюємо дані в базі
    int КількістьЗаписів = cmd.ExecuteNonQuery();

    // Виводимо повідомлення про результат
    MessageBox.Show(КількістьЗаписів.ToString(), "Змінено записів");

    //Закриваємо з'єднання
    conn.Close();
}
```

3. Збережіть зміни, зроблені у проєкті.

4. Перевірте функціональність кнопки ***Зберегти***.

У звіті з лабораторної роботи подайте скриншоти форм ***Товар*** і ***Товари*** зі зміненими даними після того, як натиснуто кнопку ***Зберегти***.

Для контролю правильності виконання операцій додайте ще скриншоти таблиці **Товари** в режимі відображення даних на початковому етапі й після збереження змінених даних. Поясніть, чому не змінилися дані на формі ***Товари***.

Подайте свої міркування щодо можливості об'єднання двох форм в одну. Зобразіть ескіз нової форми й опишіть процес виконання операцій із даними на рівні кінцевого користувача.

Завдання для самостійного виконання

1. Розширте дію групи кнопок ***База даних***, щоб під час створення і видалення бази даних можна було вказувати її ім'я.

2. Забезпечте оновлення даних на формі ***Товари*** після того, як їх змінили на формі ***Товар***.

3. На формі ***Товари*** перед першим рядком із даними у списку товарів помістіть рядок, у якому відобразіть назви стовпців списку. Для цього використайте метод отримання імені кожного стовпця таблиці.

4. Додайте функції додавання і видалення вибраному запису на формі ***Товари***, додавши відповідні кнопки в однойменну групу на формі ***Хліб***.

5. Додайте групу кнопок ***Виробники*** на форму ***Хліб***, що є аналогічною групі ***Товари***, і забезпечте їхню функціональність.

6. Додайте групу ***Продажі*** на форму ***Хліб***, що є аналогічними групі ***Товари***, і забезпечте їхню функціональність.

7. Повторіть п. 6, надавши можливість користувачу під час виконання операцій уживати назви товарів і виробників, замість їхніх кодів. Для реалізації завдання доцільно використайте елемент керування ComboBox.

8. Додайте групу ***Накладні*** на форму ***Хліб***, що є аналогічними групі ***Товари***, і забезпечте їхню функціональність щодо таблиць ***Накладні*** та ***ТовариНакладних***.

9. Повторіть п. 8, надавши можливість користувачу під час виконання операцій одночасно відображати на формі дані таблиць ***Накладні*** та ***ТовариНакладних*** та вживати назви товарів і виробників, замість їхніх кодів.

Лабораторна робота 2
Розробка програм виконання операцій у роз'єднаному середовищі

Цілі лабораторної роботи:

1. Набуття практичних навичок у програмній реалізації CRUD-операцій із базою даних у роз'єднаному середовищі.

2. Набуття практичних навичок у створенні наборів даних.

3. Набуття практичних навичок у створенні локальних таблиць і виконанні операцій із ними.

4. Набуття практичних навичок у використанні класів адаптерів даних ADO.NET.

5. Набуття практичних навичок у відображенні даних у застосунках і виконання операцій ведення бази даних.

Перед виконанням роботи студент має знати:

основи використання бібліотеки Windows Forms;

організацію системи створення й керування з'єднанням в ADO.NET у роз'єднаному середовищі;

основи доступу до даних у роз'єднаному середовищі;

базові елементи схеми таблиць бази даних;

організацію системи встановлення зв'язків між таблицями;

алгоритми оновлення ієрархічних даних;

основи архітектури класу DataSet;

основні методи класу DataAdapter;

основи прив'язування даних до елементів інтерфейсу.

Після виконання роботи студент має уміти:

самостійно розробляти прості C#-програми із графічним інтерфейсом користувача для автономної роботи користувача;

створювати локальні таблиці, використовуючи програмні засоби C#;

забезпечувати програмну реалізацію виконання CRUD-операцій із базою даних у роз'єднаному середовищі;

використовувати засоби прив'язування даних під час розроблення інтерфейсу користувача.

Підготовча частина

Хід роботи

1. Створення проєкту та кнопкової форми.

2. Побудова схеми локальної таблиці й адаптера даних (таблиця *Товари*).

3. Використання класу CommandBuilder (таблиця *Виробники*).

4. Завантаження схем таблиць із бази даних (таблиця *Продажі*).

5. Спільне ведення батьківської й дочірньої таблиць (таблиці *Накладні* та *ТовариНакладних*).

Форма звітності

За результатами виконання роботи необхідно оформити електронний звіт засобами Word. За кожним завданням слід зробити скриншоти з результатами виконання, а також подати код оброблювача події. У кінці деяких завдань зазначено, що слід надати пояснення. Аналогічним чином оформити звіт із виконання кожного завдання з п. "Завдання для самостійного виконання".

Критерії оцінювання

У 12-бальній системі оцінку результату захисту лабораторної роботи формують за такими правилами:

1. За кожний пункт практичної частини може бути виставлено від 0 до 1 бала.

2. За виконання завдань п. "Завдання для самостійного виконання" може бути виставлено

від 0 до 1 бала за кожне завдання з діапазону 1 – 5;

від 0 до 2 балів за завдання 6.1;

від 0 до 3 балів за завдання 6.2;

від 0 до 6 балів за завдання 7;

від 0 до 4 балів за завдання 8;

від 0 до 3 балів за завдання 9.

3. За кілька варіантів виконання одного із завдань додають 1 бал.

4. За вибір варіанта, який із кількох варіантів виконання є оптимальним, та обґрунтування вибору додають 1 бал.

5. За виконання кожного завдання в іншій СУБД (Oracle, DB2, MySQL, PostgreSQL тощо) додають по 1 балу.

6. За запізнення із захистом лабораторної роботи знімають 2 бали за кожний тиждень.

Набрану кількість балів із відповідей на кожне завдання лабораторної роботи підсумовують. У результаті такого підрахунку студент може набрати від 0 до 12 балів.

У разі запізнення із захистом лабораторної роботи понад три тижні студент виконує завдання роботи на основі індивідуальної бази даних. Роботу оцінюють за описаними раніше критеріями.

Практична частина

Постановка загального завдання

Вивчення засобів роботи з базою даних у роз'єднаному середовищі здійснюють шляхом створення застосунку ***роз'єднанийХліб***. Його призначено для роботи з базою даних ***Хліб*** у СУБД SQL Server.

У базі даних ***Хліб*** зберігають дані про роботу кіоску із продажу хлібобулочних виробів – отримання товарів і їхній продаж. База даних має схему, подану на рис. 1.1 під час опису лабораторної роботи 1.

Керування роботою застосунку здійснюють за допомогою кнопкової форми ***Хліб*** (рис. 2.1). Кнопки призначено для виклику відповідних функціональних форм.

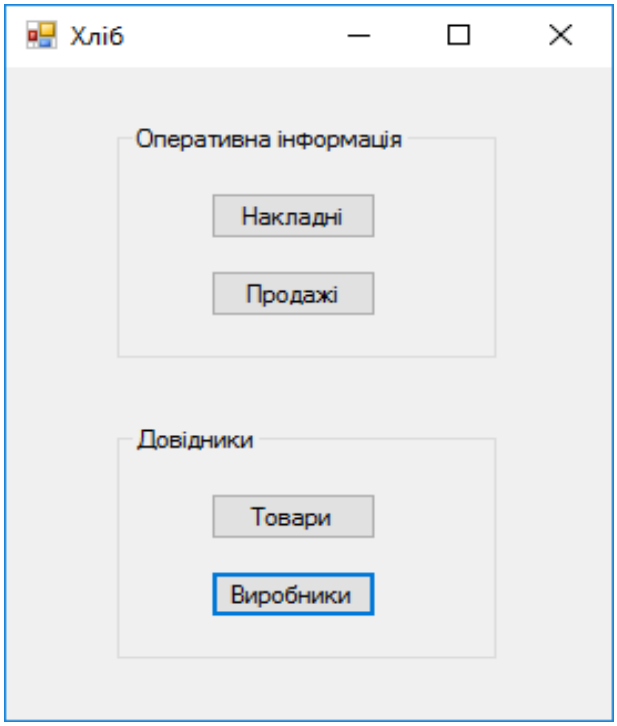


Рис. 2.1. **Кнопкова форма *Хліб***

На рис. 2.2 – 2.5 подано функціональні форми застосунку. Із їхньою допомогою можна переглядати й змінювати дані (обновляти, додавати та видаляти) у відповідних таблицях бази даних ***Хліб***.

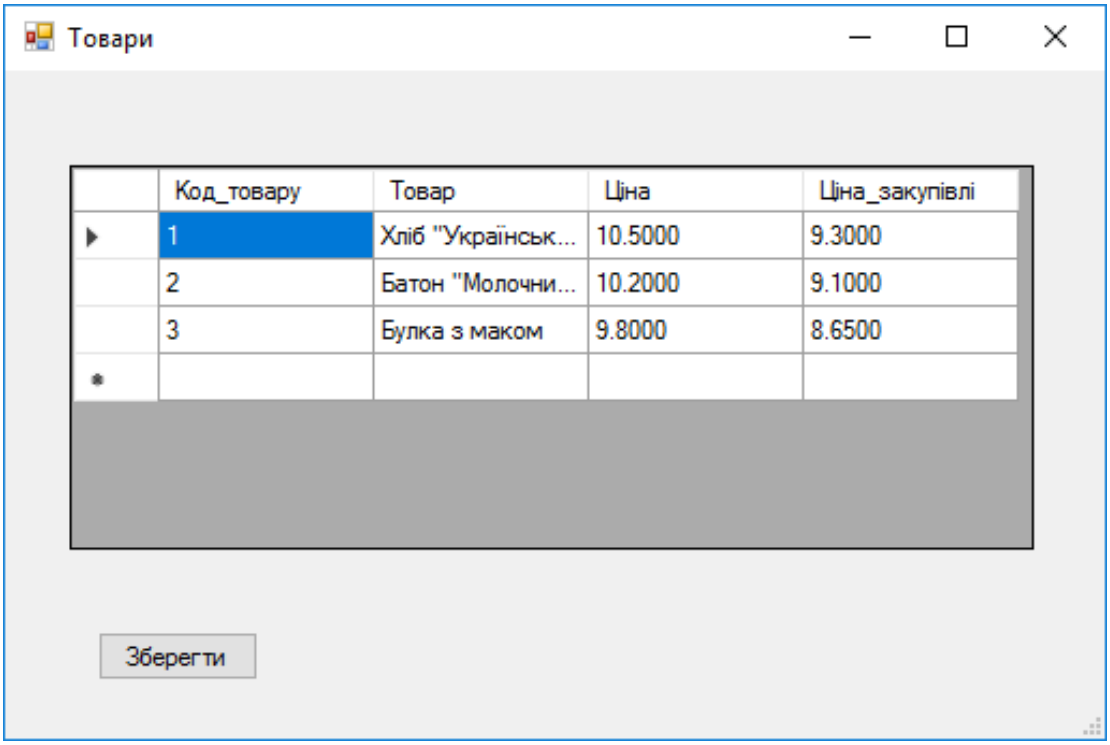


Рис. 2.2. **Форма для ведення таблиці *Товари***

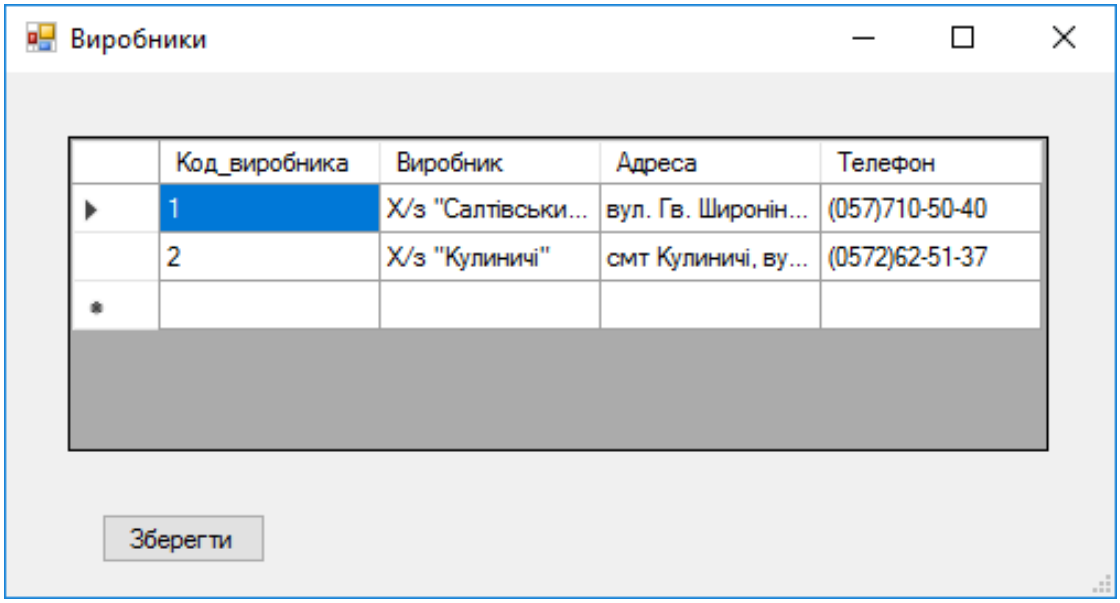


Рис. 2.3. **Форма для ведення таблиці *Виробники***

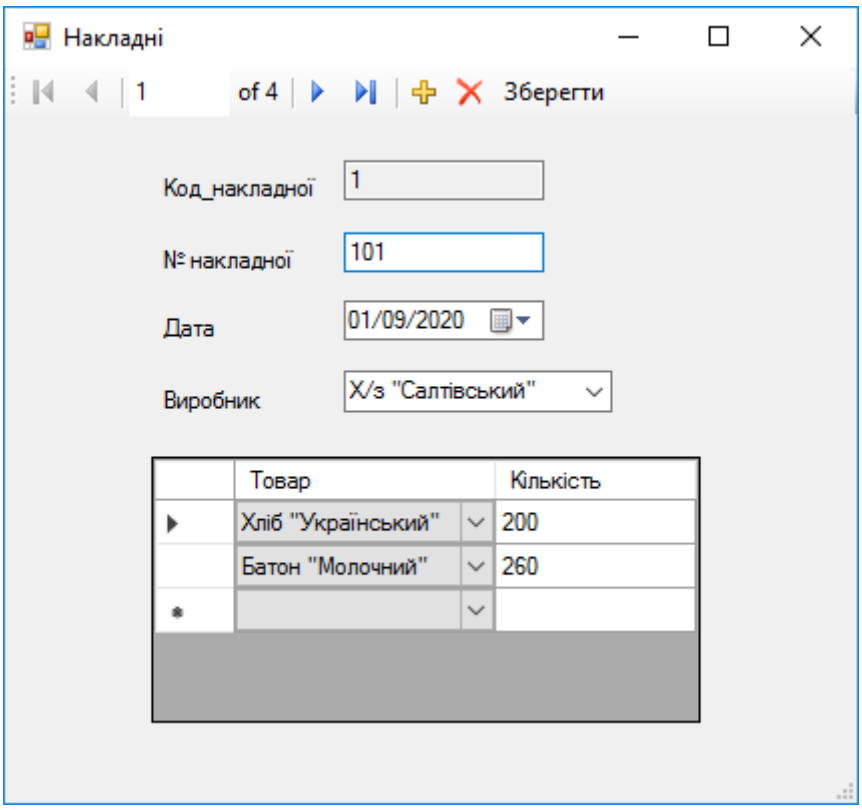


Рис. 2.4. **Форма для ведення таблиць *Накладні* й *ТовариНакладних***

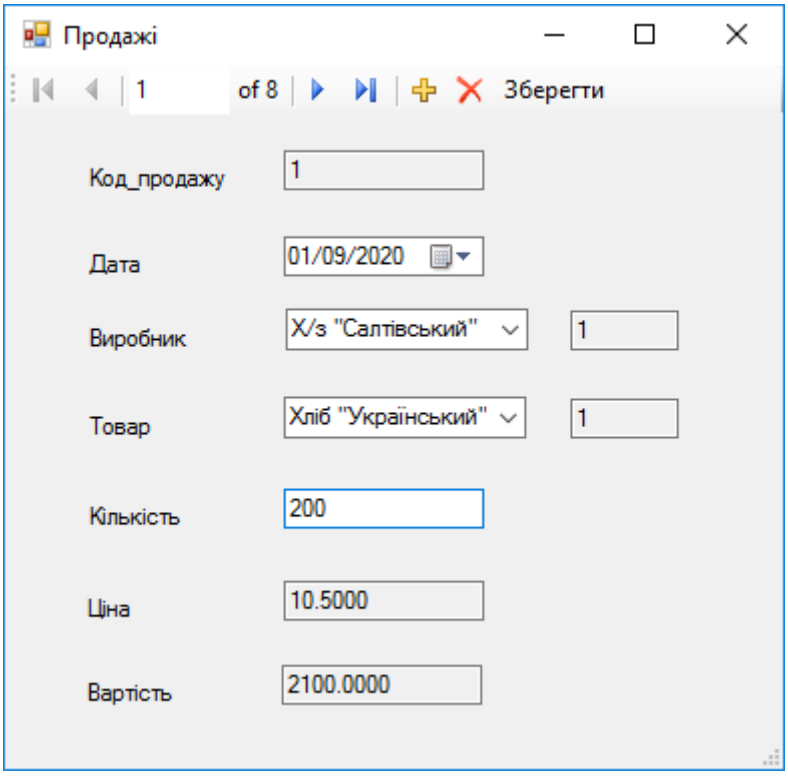


Рис. 2.5. **Форма для ведення таблиці *Продажі***

Побудову функціональних форм спрямовано на оволодіння такими вміннями та навичками в роботі з об'єктами класу DataSet:

форма ***Товари*** – побудова повної схеми таблиці класу DataTable, що відображає в DataSet відповідну таблицю бази даних;

форма ***Виробники*** – використання класу CommandBuilder для спрощення побудови об'єкта класу DataAdapter (позбавляє необхідності задавати властивості InsertCommand, DeleteCommand та UpdateCommand для виконання операцій збереження даних);

форма ***Продажі*** – завантаження схем таблиць із бази даних, окрім зв'язків між ними, які будують у коді;

форма ***Накладні*** – виконання операцій CRUD із таблицями Накладні і ТовариНакладних, пов'язані відношенням "один-до-багатьох" на одній формі.

Для відображення даних на формах у проєкті використовують клас BindingSource як з'єднувач між об'єктами класу DataSet та елементами форми.

1. Створення проєкту та кнопкової форми

1.1. Додавання бази даних до нового проєкту

Завдання

Створіть проєкт і додайте базу даних до нього.

Виконання

1. Відкрийте Visual Studio.

2. Створіть проєкт Windows Forms мовою C# з ім'ям ***роз'єднанийХліб*** і збережіть його у своїй папці із проєктами.

3. Додайте до проєкту файл бази даних ***ХлібПрізвище.mdf***, який було створено під час виконання лабораторної роботи 1. Для цього:

3.1. Перегляньте вікно **Server Explorer**. Якщо в ньому є значок бази даних ***ХлібПрізвище***, видаліть його, вибравши з його контекстового меню команду **Delete**.

Примітки:

1) виконання цієї команди призводить тільки до видалення під'єднання бази до сервера. Сам файл бази даних залишається і в разі потреби його можна повторно під'єднати, як це описано в п. 2 завд. 4 лабораторної роботи 1;

2) замість слова ***Прізвище*** в імені файлу бази даних має бути прізвище студента, наприклад, ***ХлібПетренко.mdf***. У подальших описах база даних буде мати узагальнене ім'я ***Хліб***.

3.2. Виберіть команду **Project – Existing Item** в меню **Visual Studio**.

3.3. Установіть фільтр **All Files** у вікні **Add New Item**, виберіть файл ***Хліб.mdf*** і клацніть кнопку **Add**.

У вікнах **Solution Explorer** і **Server Explorer** з'явилися значки бази даних ***Хліб.mdf*** (рис. 2.6).

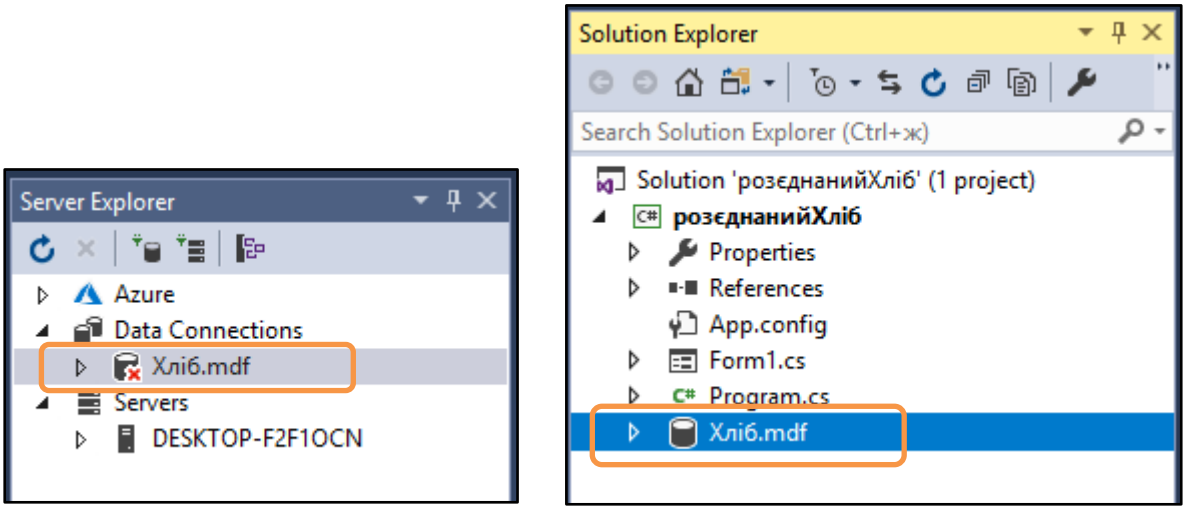


Рис. 2.6. **Значки бази даних *Хліб.mdf* у вікнах Server Explorer і Solution Explorer**

4. Перегляньте склад бази даних ***Хліб***. Зверніть увагу на те, щоб вона містила п'ять таких таблиць: ***Товари***, ***Виробники***, ***Продажі***, ***Накладні*** та ***ТовариНакладних***, а в них зберігали дані про результати надходження та продажів товарів, принаймні, за два дні від двох виробників.

У разі потреби, скористайтеся візуальними засобами **Visual Studio** для створення та заповнення таблиць. Схема бази даних і типи даних полів таблиць описано в п. "Постановка загального завдання" лабораторної роботи 1.

Примітки:

1) таблицю можна створити командою **Add New Table** з контекстового меню папки **Tables** бази даних у вікні **Server Explorer**;

2) переглянути дані таблиці й доповнити її новими можна командою **Show Table Data** з контекстового меню таблиці у вікні **Server Explorer**;

3) у таблицях бази даних можуть зберігати дані, наведені в табл. 2.1 – 2.5:

Таблиця 2.1

Дані таблиці *Товари*

Товари	Ціни	Ціни_закупівель
Хліб "Український"	10.50	9.30
Батон "Молочний"	10.20	9.10
Булка з маком	9.80	8.65

Таблиця 2.2

Дані таблиці *Виробники*

Виробники	Адреси	Телефони
Х/з "Салтівський"	вул. Гв. Широнінців, 1	(057)710-50-40
Х/з "Кулиничі"	смт Кулиничі, вул. Шкільна, 18	(0572)62-51-37

Таблиця 2.3

Дані таблиці *Продажі*

Дати	Коди_виробників	Коди_товарів	Кількість
01.09.2020 р.	1	1	200
01.09.2020 р.	1	2	250
01.09.2020 р.	2	1	150
01.09.2020 р.	2	3	180
02.09.2020 р.	1	1	220
02.09.2020 р.	1	3	170
02.09.2020 р.	2	1	200
02.09.2020 р.	2	2	100

Таблиця 2.4

Дані таблиці *Накладні*

Номери_накладних	Дати	Коди_виробників
101	01.09.2020 р.	1
201	01.09.2020 р.	2
102	02.09.2020 р.	1
202	02.09.2020 р.	2

Таблиця 2.5

Дані таблиці *ТовариНакладних*

Коди_накладних	Коди_товарів	Кількість
1	1	200
1	2	260
2	1	150
2	3	200
3	1	230
3	3	170
4	1	200
4	2	120

1.2. Формування рядка з'єднання

Завдання

Створіть клас ***myPublic*** для зберігання рядка з'єднання з базою даних, який є спільним для всього проєкту.

Ідеї виконання

Щоб уникнути переналаштування рядка з'єднання в разі перенесення проєкту в іншу папку чи на інший комп'ютер, скористайтеся визначенням шляху до папки, у якій зберігають проєкт за допомогою методу **Directory.GetCurrentDirectory()**. Цей метод повертає шлях до папки **bin\Debug**, із якої запускається exe-файл проєкту під час налаштування. Шлях до папки самого проєкту, у якій зберігають базу даних, є на два рівні коротшим. Тому використовують такий вираз:

```
Directory.GetParent(Directory.GetCurrentDirectory()).Parent.FullName
```

Клас **Directory** перебуває у просторі імен **`System.IO`**.

Виконання

1. Виконайте команду **Project – Add Class**.

2. Виберіть значок **Class** у вікні **Add New Item**, уведіть ім'я ***myPublic.cs*** і клацніть кнопку **Add**.

3. У вікні коду, що відкрилося, додайте до директив using ще й таку:

```
using System.IO;
```

4. Уведіть код тіла класу, щоб опис класу став таким:

```
class myPublic
{
      // Шлях до папки проєкту
      public static string projectPath =
Directory.GetParent(Directory.GetCurrentDirectory()).Parent.FullName;

      //Рядок з'єднання із БД
      public static string connString =
          @"Data Source = (LocalDB)\MSSQLLocalDB;" +
          "AttachDbFilename=" + projectPath + @"\Хліб.mdf;" +
          "Integrated Security = True";
}
```

Примітки:

1) залежно від версії Visual Studio, ім'я сервера (параметр **Data Source**) має такі значення:

для Visual Studio 2010 – **.\SQLExpress**;

для Visual Studio 2012, 2013 – **(localDB)\v11.0**;

для вищих версій – **(localDB)\MSSQLLocalDB;**

2) у зв'язку зі значними адміністративними обмеженнями комп'ютерів в аудиторіях університету для Visual Studio 2010 (сервер **.\SQLExpress**), слід вибирати екземпляр користувача сервера, указавши параметр **User Instance=True**. У цьому разі class **myPublic** буде мати такий формат:

```
class myPublic
{
      // Шлях до папки проєкту
      public static string projectPath =
Directory.GetParent(Directory.GetCurrentDirectory()).Parent.FullName;

      //Рядок з'єднання із БД
      public static string connString =
          @"Data Source = (LocalDB)\MSSQLLocalDB;" +
          "AttachDbFilename=" + projectPath + @"\Хліб.mdf;" +
          "Integrated Security = True;" +
          "User Instance=True";
}
```

1.3. Побудова кнопкової форми

Завдання

Побудуйте у проєкті кнопкову форму (рис. 2.7).

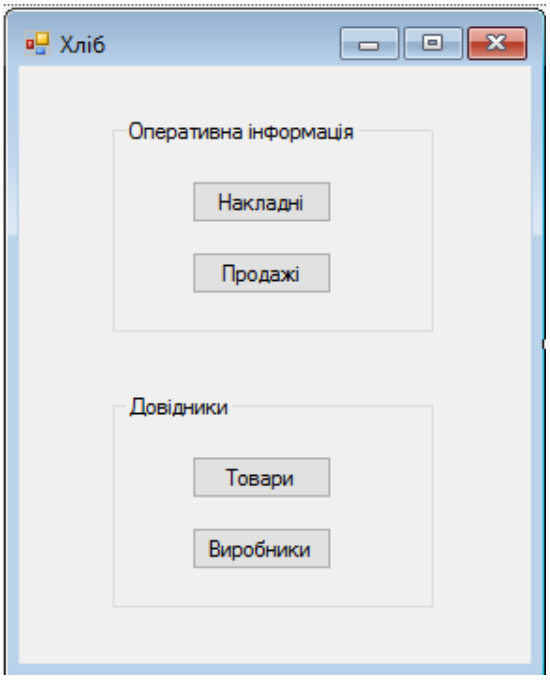


Рис. 2.7. **Кнопкова форма**

Виконання

1. У вікні **Solution Explorer** виділіть значок ***Form1*** і для файлу ***Form1.cs*** у вікні властивостей задайте ім'я ***formХліб.cs***.

2. Для властивості **Text** форми задайте значення ***Хліб***.

3. Додайте на форму елемент управління **GroupBox** і задайте значення ***Оперативна інформація*** для його властивості **Text**.

4. Додайте дві кнопки всередину елемента ***Оперативна інформація*** й установіть для них значення властивостей, наведених у табл. 2.6.

Таблиця 2.6

Властивості кнопок

Кнопки	Властивості	Значення
1	Text	Накладні
	Name	buttonНакладні
2	Text	Продажі
	Name	buttonПродажі

5. Повторіть п. 3 і 4 для групи кнопок ***Довідники***, установивши такі значення імен кнопок: ***buttonТовари*** та ***buttonВиробники***.

6. Збережіть зміни, зроблені у проєкті.

У звіті з лабораторної роботи помістіть скриншот форми, аналогічний рис. 2.7.

2. Побудова схеми локальної таблиці й адаптера даних (таблиця *Товари*)

Ідеї виконання

Для роботи з даними таблиці ***Товари*** побудуйте відповідну локальну таблицю в DataSet. Для цього задайте всі її стовпці із зазначенням типів даних, а також первинний ключ таблиці.

Щоб у локальній таблиці краще відрізняти додані рядки від уже наявних, установіть такі властивості первинного ключа:

```
column.AutoIncrementSeed = -1;
column.AutoIncrementStep = -1;
```

У результаті цього нові записи будуть мати від'ємні значення, хоча після збереження їх у базі даних СУБД надасть їм наступних за порядком додатних значень.

Для завантаження даних із бази в локальну таблицю використайте метод **Fill()** адаптера даних. Для цього задайте властивість **SelectCommand**, у якій зазначте речення **SELECT** для відбору всіх записів таблиці ***Товари***. Оскільки надалі потрібно буде задавати й інші властивості адаптера даних, помістіть ці операції у функцію користувача ***CreateТовариAdapter***.

У другому завданні для виконання збереження даних використайте метод **Update()** адаптера даних. Для цього задайте властивості **DeleteCommand**, **InsertCommand** та **UpdateCommand** у функції ***CreateТовариAdapter***. Оскільки операції видалення, додавання та редагування даних виконуються окремо з кожним записом, задайте ці властивості, використавши параметри адаптера даних. Для провайдера даних SqlClient параметри адаптера даних задають за синтаксисом, який подібний до збережених процедур. Більш детальну інформацію можна знайти за такою

адресою: https://docs.microsoft.com/en-us/dotnet/framework/data/adonet/dataadapter-parameters.

2.1. Створення локальної таблиці та адаптера даних

Завдання

Додайте в застосунок форму ***Товари***, у якій буде відображено дані однойменної локальної таблиці (рис. 2.8).

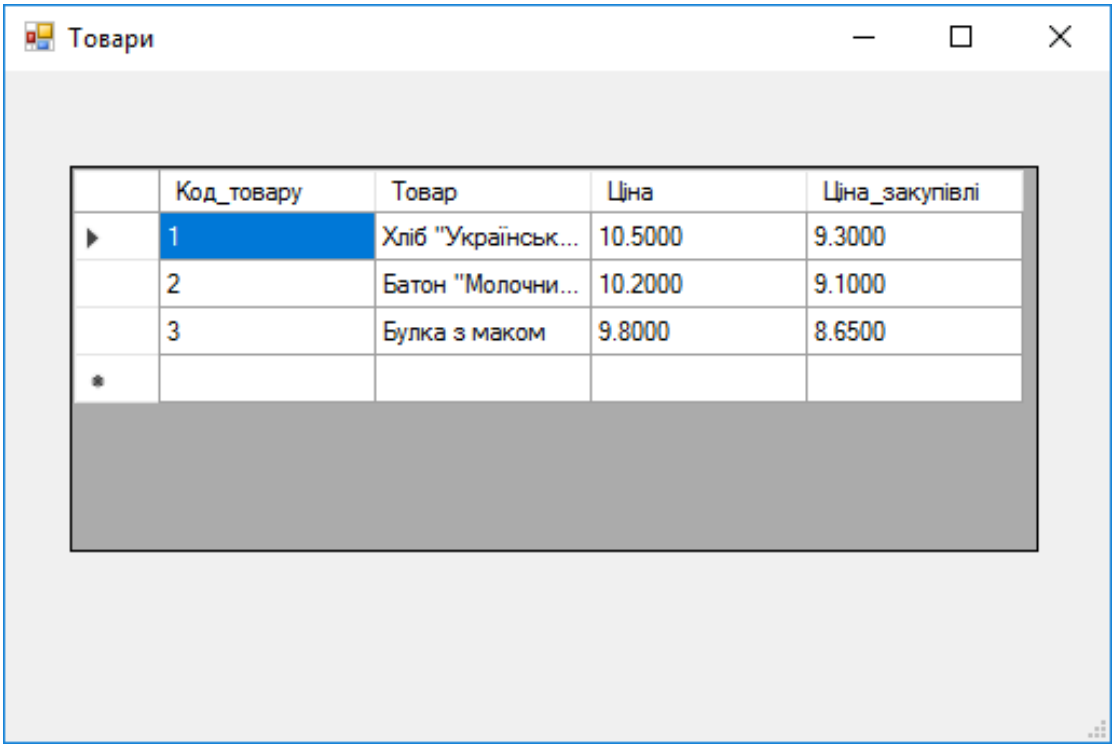


Рис. 2.8. **Форма *Товари***

Виконання

1. Додайте в застосунок нову форму. Для цього виконайте таке:

1.1. Виконайте команду **Project – Add Windows Form**.

1.2. У вікні **Add New Item** уведіть ім'я ***formТовари.cs*** і клацніть кнопку **Add**.

1.3. У вікні властивостей установіть значення ***Товари*** для властивості **Text**.

2. Додайте елемент **DataGridView** на форму для відображення записів таблиці ***Товари*** (рис. 2.9). Клацніть у вільному місці форми, щоб закрити вікно прив'язування даних.

3. Установіть в елементі керування DataGridView значення ***dataGrid-ViewТовари*** для властивості **Name**.

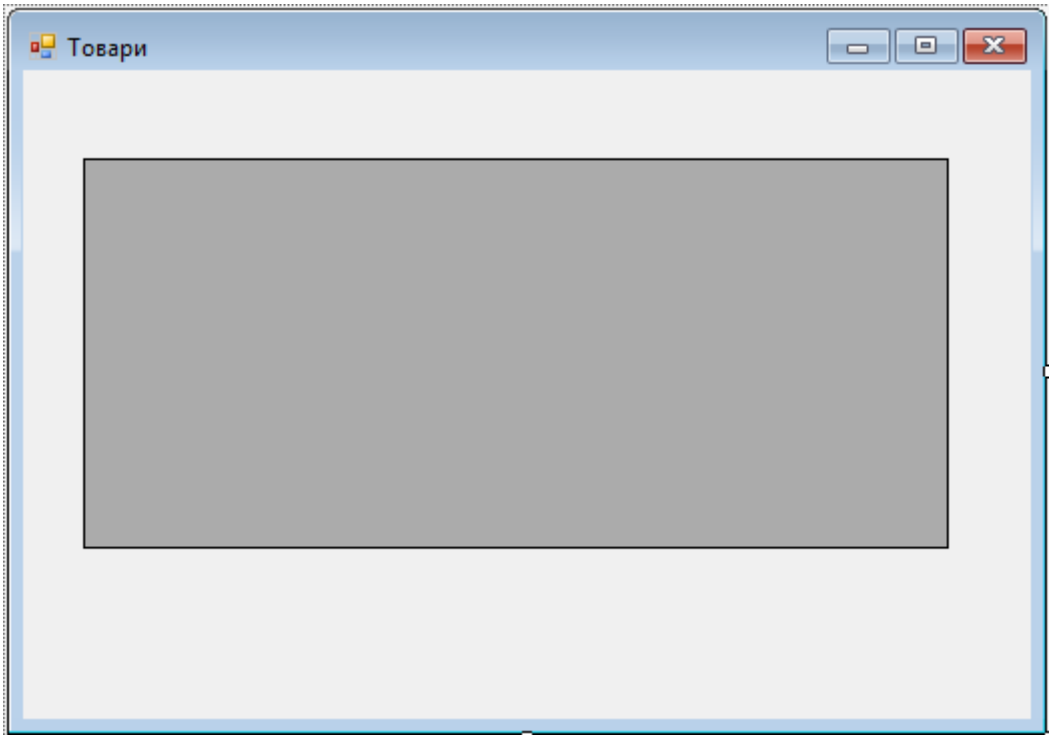


Рис. 2.9. **Елемент DataGridView на формі *Товари***

4. Перейдіть у вікно коду форми ***Товари*** й установіть посилання на потрібні простори імен:

```
using System;
using System.Data;
using System.Text;
using System.Windows.Forms;
using System.Data.SqlClient;
```

5. Уведіть у вікні коду форми ***Товари*** перед кодом конструктора `public formТовари()` такий опис спільних об'єктів:

```
// Спільні об'єкти
SqlConnection conn;     // З'єднання
DataSet ds;             // Набір даних
DataTable tableТовари;  // Локальна таблиця Товари
SqlDataAdapter da;      // Адаптер даних
BindingSource bindingТовари;//З'єднувач локальної таблиці з DataGridView
```

6. Перейдіть у вікно конструктора форми ***Товари***, двічі клацніть у вільному місці форми ***Товари*** та введіть оператори тіла оброблювача події ***formТовари_Load***. Він має такий вигляд:

```
private void formТовари_Load(object sender, EventArgs e)
{
  // З'єднання SqlConnection
  conn = new SqlConnection();
  conn.ConnectionString = myPublic.connString;

  // Створюємо DataSet
  ds = new DataSet("dsХліб");

  // Будуємо повну схему локальної таблиці Товари
  tableТовари = new DataTable("Товари");
  DataColumnCollection cols = tableТовари.Columns;
  // Ключовий стовпець
  DataColumn column = cols.Add("Код_товару",typeof(System.Int32));
  column.AutoIncrement = true;
  column.AutoIncrementSeed = -1;
  column.AutoIncrementStep = -1;
  // Інші стовпці
  cols.Add("Товар", typeof(System.String)).MaxLength = 25;
  cols.Add("Ціна", typeof(System.Decimal));
  cols.Add("Ціна_закупівлі", typeof(System.Decimal));

  // Задаємо первинний ключ
  tableТовари.PrimaryKey = new DataColumn[] { cols["Код_товару"]};

  // Додаємо таблицю Товари в DataSet
  ds.Tables.Add(tableТовари);

  // Створюємо SqlDataAdapter
  da = CreateТовариAdapter(conn);

  // Заповнюємо таблицю Товари в ds даними з бази
  da.Fill(tableТовари);
 //Відображаємо таблицю на формі, прив'язуючи її до dataGridViewТовари
  bindingТовари = new BindingSource();
  bindingТовари.DataSource = tableТовари;
  dataGridViewТовари.DataSource = bindingТовари;
}
```

7. Додайте першу частину коду функції користувача ***CreateТовари-Adapter***, розмістивши його під кодом оброблювача події formТовари_Load. Він має такий вигляд:

```
public static SqlDataAdapter CreateТовариAdapter(
    SqlConnection connection)
{
    SqlDataAdapter dataAdapter = new SqlDataAdapter();
    SqlCommand command;
    SqlParameter parameter;

    // Властивість SelectCommand
    command = new SqlCommand("SELECT * FROM Товари", connection);
    dataAdapter.SelectCommand = command;

    return dataAdapter;
}
```

8. Перейдіть у вікно конструктора форми ***Хліб***, двічі клацніть кнопку ***Товари*** та введіть оператори тіла оброблювача події ***buttonТовари_Click***. Він має такий вигляд:

```
private void buttonТовари_Click(object sender, EventArgs e)
{
    formТовари вікноТовари = new formТовари();
    вікноТовари.ShowDialog();
}
```

9. Перевірте функціональність форми. Після її завантаження мають відображати всі записи таблиці ***Товари***.

10. У разі потреби, зробіть налаштування розмірів елемента DataGridView.

11. Збережіть зміни, зроблені у проєкті.

2.2. Опис властивостей DeleteCommand, InsertCommand та UpdateCommand адаптера даних

Завдання

Додайте на форму ***Товари*** кнопку ***Зберегти***, призначену для збереження в базі даних усіх змін, і забезпечте її функціональність (рис. 2.10).

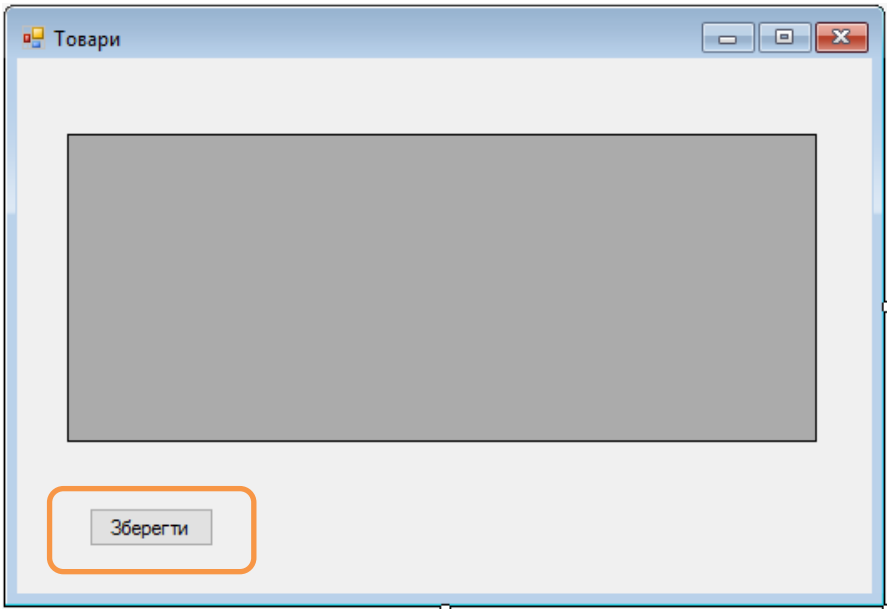


Рис. 2.10. **Кнопка *Зберегти* на формі *Товари***

Виконання

1. Поверніться на форму ***Товари***, додайте кнопку в лівому нижньому кутку форми й установіть значення властивостей, наведених у табл. 2.7.

Таблиця 2.7

Властивості кнопки

Властивості	Значення
Text	Зберегти
Name	buttonЗберегти

2. Додайте код оброблювача події "Клацання кнопки Зберегти".

```
private void buttonЗберегти_Click(object sender, EventArgs e)
{
  // Завершуємо процес редагування і змінюємо дані у БД
  bindingТовари.EndEdit();
  da.Update(ds, "Товари");
}
```

3. Додайте опис властивостей **DeleteCommand**, **InsertCommand** та **UpdateCommand** у функцію ***CreateТоварuAdapter()***, розмістивши його перед оператором:

```
return dataAdapter;
```

Після цього код функції ***CreateТовариAdapter()*** набуде такого вигляду:

```
public static SqlDataAdapter CreateТовариAdapter(
  SqlConnection connection)
{
  SqlDataAdapter dataAdapter = new SqlDataAdapter();
  SqlCommand command;
  SqlParameter parameter;

  // Властивість SelectCommand
  command = new SqlCommand("SELECT * FROM Товари", connection);
  dataAdapter.SelectCommand = command;

  // Властивість DeleteCommand
  command = new SqlCommand(
  "DELETE FROM Товари WHERE Код_товару = @Код_товару",connection);
  parameter = command.Parameters.Add("@Код_товару",SqlDbType.Int);
  parameter.SourceColumn = "Код_товару";
  parameter.SourceVersion = DataRowVersion.Original;
  dataAdapter.DeleteCommand = command;

  // Властивість InsertCommand
  command = new SqlCommand(
    "INSERT INTO Товари (Товар, Ціна, Ціна_закупівлі) " +
    "VALUES (@Товар, @Ціна, @Ціна_закупівлі)", connection);
  command.Parameters.Add("@Товар",SqlDbType.NVarChar, 25,"Товар");
  command.Parameters.Add("@Ціна", SqlDbType.Money, 5,"Ціна");
  command.Parameters.Add("@Ціна_закупівлі", SqlDbType.Money, 5,
  "Ціна_закупівлі");
  dataAdapter.InsertCommand = command;

  // Властивість UpdateCommand
  command = new SqlCommand(
    "UPDATE Товари SET Товар = @Товар, Ціна = @Ціна, " +
    "Ціна_закупівлі = @Ціна_закупівлі" +
    " WHERE Код_товару = @Код_товару", connection);
  command.Parameters.Add("@Товар", SqlDbType.NVarChar,25,"Товар");
  command.Parameters.Add("@Ціна", SqlDbType.Decimal, 5, "Ціна");
  command.Parameters.Add("@Ціна_закупівлі", SqlDbType.Decimal, 5,
    "Ціна_закупівлі");
  parameter = command.Parameters.Add("@Код_товару",SqlDbType.Int);
  parameter.SourceColumn = "Код_товару";
  parameter.SourceVersion = DataRowVersion.Original;
  dataAdapter.UpdateCommand = command;

  return dataAdapter;
}
```

4. Перевірте функціональність кнопки ***Зберегти***, виконуючи операції вилучення, додавання та редагування даних в елементі DataGridView із наступним збереженням змін у базі даних.

5. Збережіть зміни, зроблені у проєкті.

6. Додайте в оброблювач події ***buttonЗберегти_Click*** код, що реалізує повідомлення про успішне збереження змінених даних (рис. 2.11).

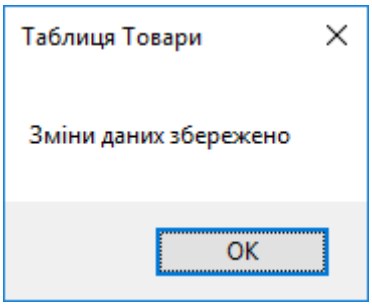


Рис. 2.11. **Повідомлення про успішне збереження змінених даних**

У звіті з лабораторної роботи подайте скриншот форми ***Товари*** зі зміненими даними після того, як натиснуто кнопку ***Зберегти***, а також код оброблювачів подій ***formТовари_Load*** та ***buttonЗберегти_Click***. Поясніть, чому в кінці кожного коду відсутнє закривання з'єднання, як це було в лабораторній роботі 1.

3. Використання класу CommandBuilder (таблиця *Виробники*)

Ідеї виконання

Перегляд даних таблиці ***Виробники*** реалізують аналогічно до описаного раніше для таблиці ***Товари***. Для реалізації інших CRUD-операцій можна скоротити код застосунку, використавши клас **CommandBuilder**. Тому єдиною відмінністю від попереднього завдання є створення об'єкта класу **CommandBuilder**. Він позбавляє від необхідності задавати властивості **DeleteCommand**, **InsertCommand** та **UpdateCommand** у функції ***CreateТовариAdapter()*** для виконання операцій видалення, додавання та редагування даних у базі даних. Їх автоматично створює об'єкт класу **CommandBuilder**, використовуючи властивість **SelectCommand**.

3.1. Створення локальної таблиці та об'єкта класу CommandBuilder

Завдання

Додайте в застосунок форму ***Виробники***, яка буде відображати дані однойменної таблиці (рис. 2.12).

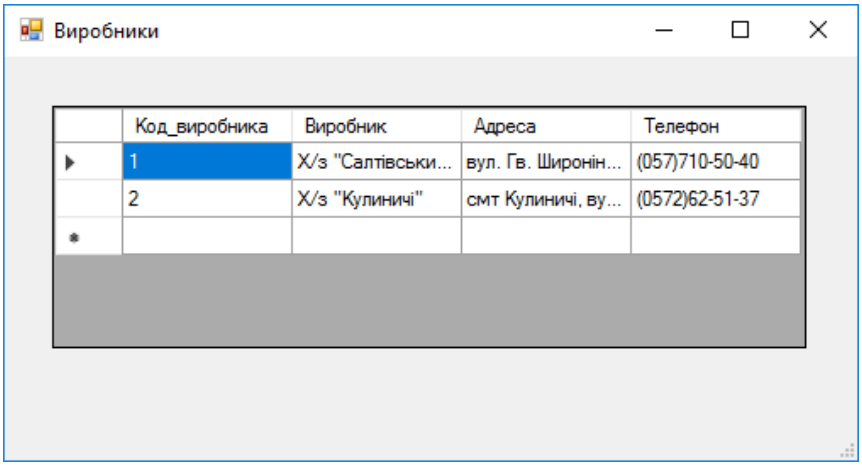


Рис. 2.12. **Форма *Виробники***

Виконання

1. Додайте в застосунок нову форму з ім'ям ***formВиробники*** заголовком ***Виробники***.

2. Додайте елемент **DataGridView** на форму для відображення записів таблиці ***Виробники*** (рис. 2.13). Клацніть у вільному місці форми, щоб закрити вікно прив'язування даних.

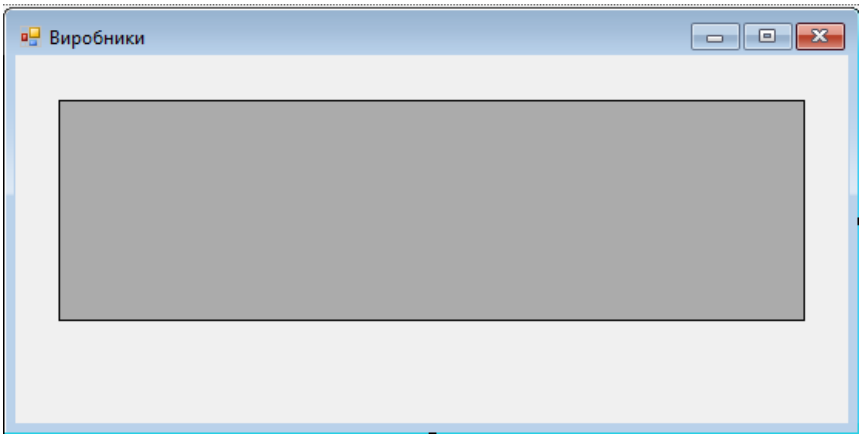


Рис. 2.13. **Елемент DataGridView на формі *Виробники***

3. Установіть в елементі керування DataGridView значення ***dataGridViewВиробники*** для властивості **Name**.

4. Перейдіть у вікно коду форми ***Виробники*** й установіть посилання на потрібні простори імен:

```
using System;
using System.Data;
using System.Text;
using System.Windows.Forms;
using System.Data.SqlClient;
```

5. Уведіть у вікні коду форми ***Виробники*** перед кодом конструктора `public formВиробники()` такий опис спільних об'єктів:

```
// Спільні об'єкти
SqlConnection conn;       //З'єднання
DataSet ds;               //Набір даних
DataTable tableВиробники; //Локальна таблиця Виробники
SqlDataAdapter da;        //Адаптер даних
//З'єднувач локальної таблиці з DataGridView
BindingSource bindingВиробники;
// Будівник запитів на зміну,
// додавання і видалення для адаптера
SqlCommandBuilder cb;
```

6. Перейдіть у вікно конструктора форми ***Виробники***, двічі клацніть у вільному місці форми ***Виробники*** та введіть оператори тіла оброблювача події ***formВиробники_Load***. Він має такий вигляд:

```
private void formВиробники_Load(object sender, EventArgs e)
{
  // З'єднання SqlConnection
  conn = new SqlConnection();
  conn.ConnectionString = myPublic.connString;
  // Створюємо DataSet
  ds = new DataSet("dsХліб");

  // Будуємо повну схему локальної таблиці Виробники
  tableВиробники = new DataTable("Виробники");
  DataColumnCollection cols = tableВиробники.Columns;
  // Ключовий стовпець
  DataColumn column=cols.Add("Код_виробника",typeof(System.Int32));
  column.AutoIncrement = true;
  column.AutoIncrementSeed = -1;
  column.AutoIncrementStep = -1;
  //Інші стовпці
  cols.Add("Виробник", typeof(System.String)).MaxLength = 20;
  cols.Add("Адреса", typeof(System.String)).MaxLength = 30;
  cols.Add("Телефон", typeof(System.String)).MaxLength = 15;
```

```
  // Задаємо первинний ключ
  tableВиробники.PrimaryKey = new DataColumn[] {
cols["Код_виробника"]};

  // Додаємо таблицю Виробники в DataSet
  ds.Tables.Add(tableВиробники);

  // Створюємо SqlDataAdapter
  da = CreateВиробникиAdapter(conn);

  //Створюємо CommandBuilder
  cb = new SqlCommandBuilder(da);

  // Заповнюємо таблицю Виробники у ds даними з бази
  da.Fill(tableВиробники);

  // Відображаємо таблицю на формі, прив'язуючи її
  // до dataGridViewВиробники
  bindingВиробники = new BindingSource();
  bindingВиробники.DataSource = tableВиробники;
  dataGridViewВиробники.DataSource = bindingВиробники;
}
```

7. Додайте код функції користувача ***CreateВиробникиAdapter***, розмістивши його під кодом оброблювача події ***formВиробники_Load***. Він має такий вигляд:

```
public static SqlDataAdapter CreateВиробникиAdapter(
    SqlConnection connection)
{
  SqlDataAdapter dataAdapter = new SqlDataAdapter();
  SqlCommand command;

  // Властивість SelectCommand
  command = new SqlCommand("SELECT * FROM Виробники", connection);
  dataAdapter.SelectCommand = command;
  return dataAdapter;
}
```

8. Перейдіть у вікно конструктора форми ***Хліб***, двічі клацніть кнопку ***Виробники*** та введіть оператори тіла оброблювача події ***buttonВиробники_Click***. Він має такий вигляд:

```
private void buttonВиробники_Click(object sender, EventArgs e)
{
    formВиробники вікноВиробники = new formВиробники();
    вікноВиробники.ShowDialog();
}
```

9. Перевірте функціональність форми. Після її завантаження мають відображати всі записи таблиці ***Виробники***.

10. У разі потреби, зробіть налаштування розмірів елемента DataGridView.

11. Збережіть зміни, зроблені у проєкті.

3.2. Збереження змін даних із використанням класу CommandBuilder

Завдання

Додайте на форму ***Виробники*** кнопку ***Зберегти***, призначену для збереження в базі даних усіх змін, і забезпечте її функціональність (рис. 2.14).

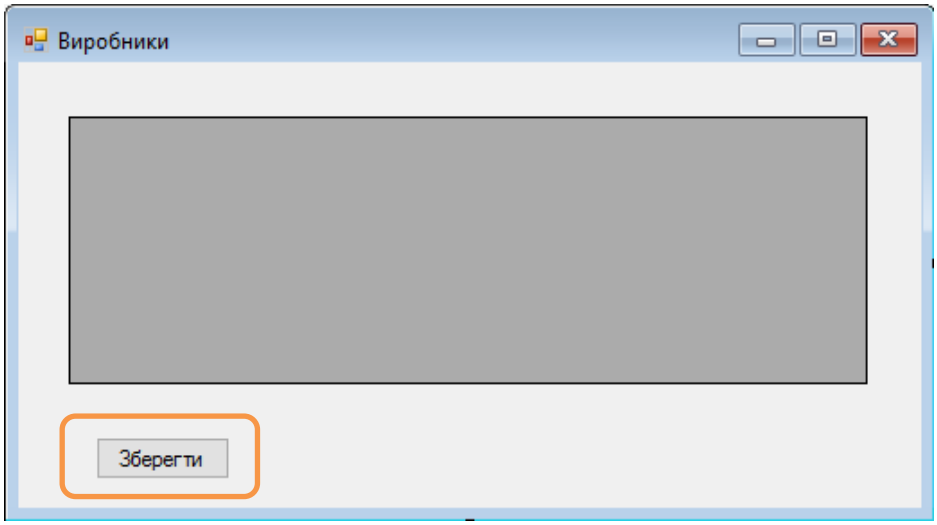


Рис. 2.14. **Кнопка *Зберегти* на формі *Виробники***

Виконання

1. Поверніться на форму ***Виробники***, додайте кнопку в лівому нижньому кутку форми й установіть такі значення властивостей, наведених у табл. 2.8:

Таблиця 2.8

Властивості кнопки

Властивості	Значення
Text	Зберегти
Name	buttonЗберегти

2. Додайте код оброблювача події "Клацання кнопки Зберегти".

```
private void buttonЗберегти_Click(object sender, EventArgs e)
{
  // Завершуємо процес редагування і змінюємо дані у БД
  bindingВиробники.EndEdit();
  da.Update(ds, "Виробники");
  MessageBox.Show("Зміни даних збережено", "Таблиця Виробники");
}
```

3. Перевірте функціональність кнопки ***Зберегти***, виконуючи операції вилучення, додавання та редагування даних в елементі DataGridView із наступним збереженням змін у базі даних.

4. Збережіть зміни, зроблені у проєкті.

У звіті з лабораторної роботи подайте скриншот форми ***Виробники*** зі зміненими даними після того, як натиснуто кнопку ***Зберегти***, а також код оброблювачів подій ***formВиробники_Load*** та ***buttonЗберегти_Click***. Поясніть, чому код функції ***CreateВиробникиAdapter()*** значно коротший, ніж аналогічний ***CreateТовариAdapter()*** у п. 2.2.

4. Завантаження схем таблиць із бази даних (таблиця *Продажі*)

Ідеї виконання

Для роботи з даними таблиці ***Продажі*** створимо порожню локальну таблицю в DataSet, не задаючи її схеми. Вона автоматично завантажиться в локальну таблицю під час першого завантаження даних із бази методом **Fill()**.

Для зручності перегляду та оновлення даних таблиці ***Продажі*** використайте дані з батьківських таблиць ***Товари*** та ***Виробники***. Відповідні локальні таблиці побудуйте за тим самим алгоритмом – створіть порожні локальні таблиці, а їхні схеми автоматично завантажаться з бази даних під час першого використання методу **Fill()**.

Зв'язки між локальними батьківськими таблицями ***Товари*** та ***Виробники*** й дочірньою ***Продажі*** установіть за допомогою класу **DataRelation**.

Для кожної локальної таблиці, дані якої використовують на формі, потрібно створити свій об'єкт класу **DataAdapter**. Властивість **SelectCommand** адаптерів даних різних таблиць відрізняється тільки назвою таблиці. Тому для всіх таблиць побудуйте спільну функцію користувача ***Cre-***

ateTableAdapter(String table, SqlConnection connection), у якій перший параметр указує назву таблиці в базі даних.

Оскільки на формі ***Продажі*** можна змінювати тільки дані однойменної таблиці, об'єкт класу **CommandBuilder** створіть лише для неї.

4.1. Відображення даних без побудови схеми таблиці

Завдання

Додайте в застосунок форму ***Продажі***, у якій будуть відображати та змінювати дані однойменної таблиці (рис. 2.15).

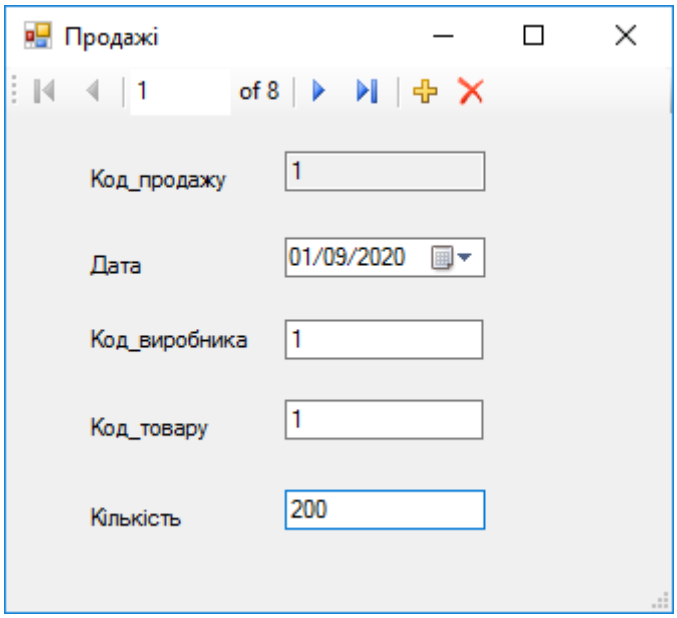


Рис. 2.15. **Форма *Продажі* з елементами, що відображають дані однойменної таблиці**

Виконання

1. Додайте в застосунок нову форму з ім'ям ***formПродажі*** заголовком ***Продажі***.

2. Для відображення значень полів одного запису таблиці ***Продажі*** додайте на форму такі елементи керування:

2.1. П'ять написів (елемент керування Label) у стовпчик один під одним з лівого боку форми.

2.2. Праворуч від написів додайте елементи керування для відображення даних у такому порядку: TextBox, DateTimePicker, TextBox, TextBox, TextBox (рис. 2.16).

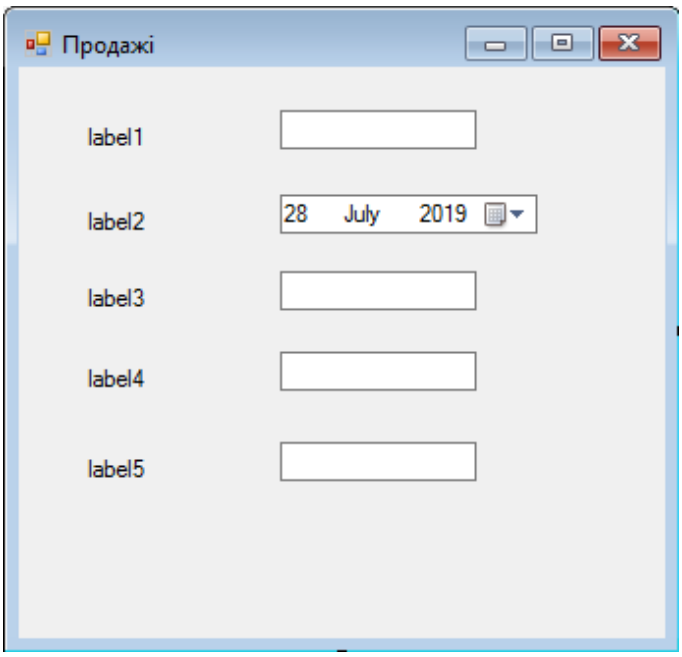


Рис. 2.16. **Початковий вигляд форми *Продажі***

3. Установіть для елементів керування такі значення властивостей, наведених у табл. 2.9.

Таблиця 2.9

Властивості елементів керування

№ п/п	Написи (властивості Text)	Елементи керування для відображення даних (властивості Name)
1	Код_продажу	textBoxКод_продажу
2	Дата	dateTimePickerДата
3	Код_виробника	textBoxКод_виробника
4	Код_товару	textBoxКод_товару
5	Кількість	textBoxКількість

4. У текстовому полі ***textBoxКод_продажу*** додатково установіть значення ***True*** для властивості **ReadOnly**.

5. В елементі ***dateTimePickerДата*** установіть значення ***Short*** для властивості **Format** і зменште його ширину, зробивши такою, як і в текстових полях.

6. Додайте на форму елемент керування **BindingNavigator** (розміщають у категорії **Data**) і встановіть значення ***bindingNavigatorПродажі*** для властивості **Name** (рис. 2.17).

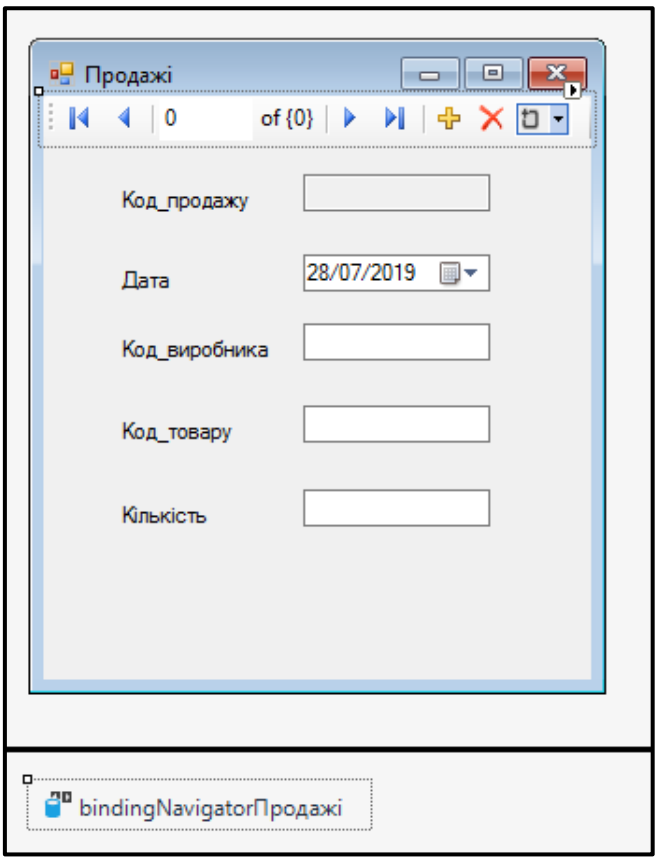


Рис. 2.17. **Форма *Продажі* з елементом BindingNavigator**

7. Перейдіть у вікно коду форми й установіть посилання на потрібні простори імен:

```
using System;
using System.Data;
using System.Text;
using System.Windows.Forms;
using System.Data.SqlClient;
```

8. Уведіть у вікні коду форми перед кодом конструктора public formПродажі() такий опис спільних об'єктів:

```
// Спільні об'єкти
SqlConnection conn;   //З'єднання
DataSet ds;             //Набір даних
DataTable tableПродажі; //Локальна таблиця Продажі
SqlDataAdapter daПродажі;    //Адаптер даних для таблиці Продажі
SqlCommandBuilder cbПродажі; //Будівник запитів для адаптера
BindingSource bindingПродажі;  //З'єднувач локальної таблиці
                               //з елементами форми
```

9. Перейдіть у вікно конструктора форми ***Продажі***, двічі клацніть у вільному місці форми та введіть оператори тіла оброблювача події ***formПродажі_Load***. Він має такий вигляд:

```
private void formПродажі_Load(object sender, EventArgs e)
{
    // З'єднання SqlConnection
    conn = new SqlConnection();
    conn.ConnectionString = myPublic.connString;

    // Створюємо DataSet
    ds = new DataSet("dsХліб");

    // Будуємо таблицю Продажі
    tableПродажі = new DataTable("Продажі");

    // Додаємо таблицю Продажі в DataSet
    ds.Tables.Add(tableПродажі);

    // Створюємо DataAdapter для таблиці Продажі
    daПродажі = CreateTableAdapter("Продажі", conn);

    // Створюємо CommandBuilder
    cbПродажі = new SqlCommandBuilder(daПродажі);

    //Заповнюємо таблицю Продажі у ds даними з бази
    daПродажі.Fill(ds.Tables["Продажі"]);

    // Створюємо з'єднувача локальної таблиці з елементами форми
    bindingПродажі = new BindingSource();

    // Установлюємо прив'язку до таблиці
    bindingПродажі.DataSource = tableПродажі;

    // Налаштовуємо на прив'язку елементи на формі
    textBoxКод_продажу.DataBindings.Add("Text",
        bindingПродажі, "Код_продажу");
    dateTimePickerДата.DataBindings.Add("Value", bindingПродажі,
        "Дата", true);
    textBoxКод_товару.DataBindings.Add("Text", bindingПродажі,
        "Код_товару");
    textBoxКод_виробника.DataBindings.Add
        ("Text", bindingПродажі, "Код_виробника");
    textBoxКількість.DataBindings.Add("Text", bindingПродажі,
        "Кількість");

    // Налаштовуємо навігатор записами на з'єднувача
    bindingNavigatorПродажі.BindingSource = bindingПродажі;

}
```

10. Додайте код функції користувача ***CreateTableAdapter***, розмістивши його під кодом оброблювача події formПродажі_Load. Він має такий вигляд:

```
public static SqlDataAdapter CreateTableAdapter(
    String table,
    SqlConnection connection)
{
  SqlDataAdapter dataAdapter = new SqlDataAdapter();
  SqlCommand command;

  // Властивість SelectCommand
  command = new SqlCommand("SELECT * FROM " + table, connection);
  dataAdapter.SelectCommand = command;

  return dataAdapter;
}
```

11. Додайте код оброблювача події "Клацання кнопки Продажі".

```
private void buttonПродажі_Click(object sender, EventArgs e)
{
    formПродажі вікноПродажі = new formПродажі();
    вікноПродажі.ShowDialog();
}
```

12. Перевірте функціональність форми ***Продажі***. Після її завантаження має відображатися перший запис таблиці ***Продажі***, а за допомогою навігатора можна переглянути всі інші.

13. Перевірте функціональність кнопки **Delete**, розміщеної на панелі навігатора. Поточний запис видаляється з перегляду (у DataSet його позначено як вилучений).

14. Перевірте функціональність кнопки **Add new**, розміщеної на панелі навігатора, увівши дані про продажі 100 батонів "Молочних" із хлібозаводу "Кулиничі" за поточний день. Водночас виникли труднощі через необхідність знати коди товарів і виробників. Їх усувають у наступному завданні.

15. Збережіть зміни, зроблені у проєкті.

4.2. Установлення зв'язків між локальними таблицями

Завдання

Додайте на форму ***Продажі*** поле зі списком ***Товар*** (рис. 2.18). Його використовують для введення в таблицю ***Продажі*** кодів товарів. Під час

вибору із цього списку назви товару в таблицю ***Продажі*** автоматично запишеться його код.

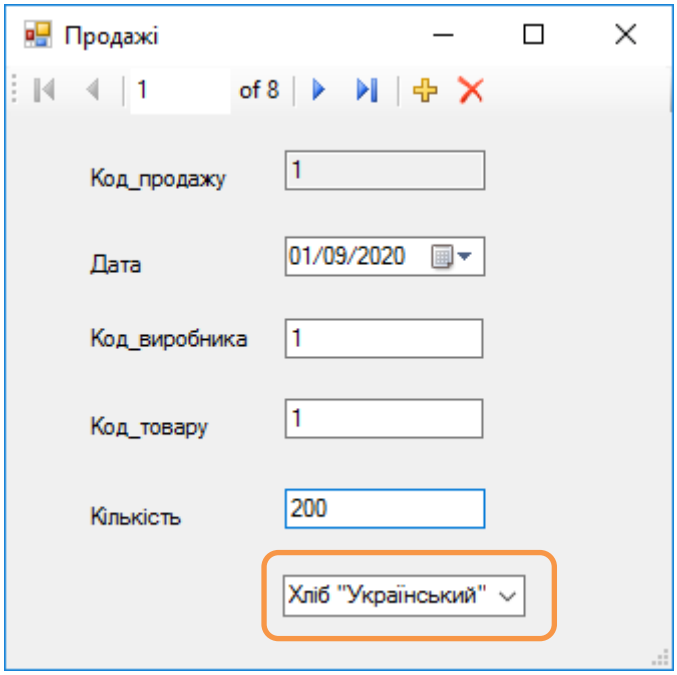


Рис. 2.18. **Форма *Продажі* з полем зі списком *Товар***

Виконання

1. Додайте на форму поле зі списком (елемент керування ComboBox), помістивши його під останнім полем (рис. 2.19).

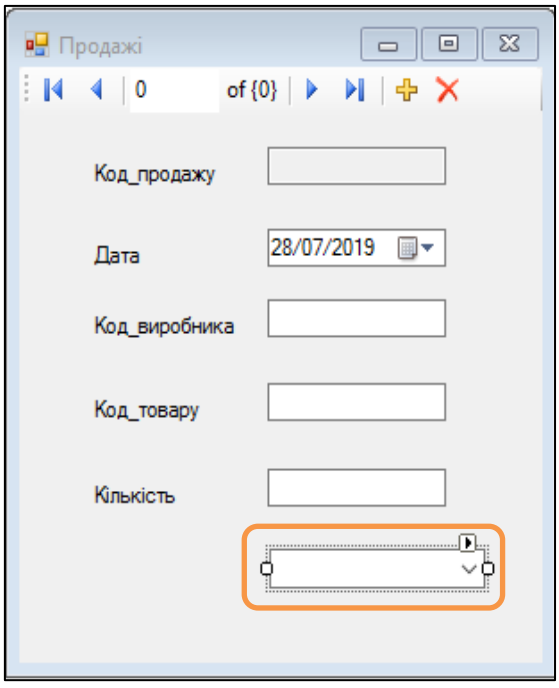


Рис. 2.19. **Поле зі списком на формі *Продажі***

2. Для нового елемента встановіть значення ***comboBoxТовар*** для властивості **Name**.

3. Після останнього оператора тіла оброблювача події ***formПродажі_Load*** (перед закриваючою дужкою) додайте оператори, які дають функціональність полю зі списком ***comboBoxТовар***. Вони мають такий вигляд:

```
//*******************************
//*   Робота з таблицею Товари   *
//*******************************

//Створюємо таблицю Товари
DataTable tableТовари = new DataTable("Товари");

//Додаємо таблицю Товари в DataSet
ds.Tables.Add(tableТовари);

//Створюємо SqlDataAdapter для таблиці Товари
SqlDataAdapter daТовари = CreateTableAdapter("Товари", conn);

//Заповнюємо таблицю Товари у ds даними з бази
daТовари.Fill(ds.Tables["Товари"]);

//Створюємо прив'язку до таблиці Товари у ds
BindingSource bindingТовари = new BindingSource();
bindingТовари.DataSource = tableТовари;

//Установлюємо зв'язок між таблицями Товари й Продажі
DataRelation relationТовариПродажі =
  new DataRelation("ТовариПродажі",
     new DataColumn[] { tableТовари.Columns["Код_товару"] },
     new DataColumn[] { tableПродажі.Columns["Код_товару"] });
ds.Relations.Add(relationТовариПродажі);

//Прив'язуємо поле зі списком до таблиці підстановки Товари
comboBoxТовар.DataSource = bindingТовари;
comboBoxТовар.DisplayMember = "Товар";
comboBoxТовар.ValueMember = "Код_товару";
comboBoxТовар.DataBindings.Add(
  "SelectedValue",bindingПродажі,"Код_товару");
```

4. Запустіть програму на виконання й перевірте функціональність поля зі списком ***comboBoxТовар***. Спочатку перегляньте всі записи таблиці ***Продажі***, а потім додайте новий. Код товару потрібно вводити шляхом вибору назви товару в полі зі списком ***comboBoxТовар***.

5. Збережіть зміни, зроблені у проєкті.

4.3. Використання даних із батьківської таблиці

Завдання

Додайте на форму ***Продажі*** текстове поле ***Ціна*** разом із написом (рис. 2.20). У ньому відображено ціну товару, код якого зазначено в таблиці ***Продажі***. Цю величину вибирають із батьківської таблиці ***Товари***. Її використовують тільки для перегляду і не можна тут змінити.

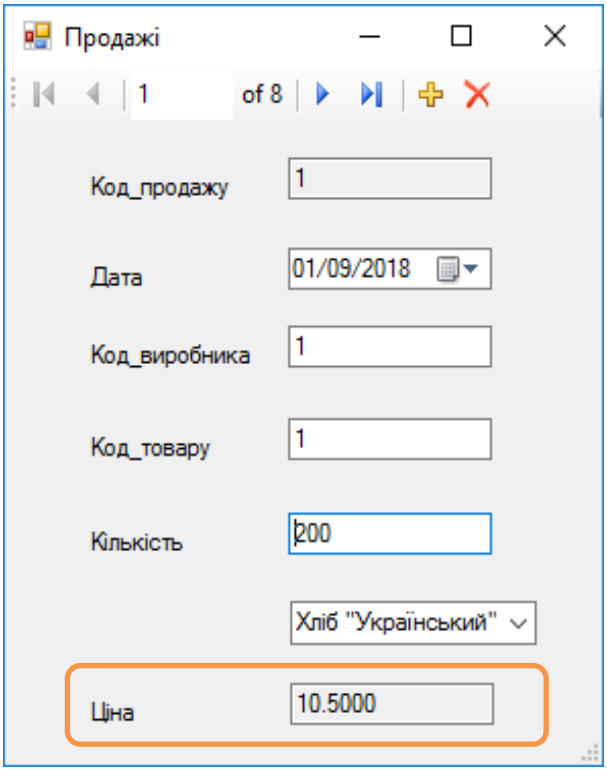


Рис. 2.20. **Форма *Продажі* з полем *Ціна***

Виконання

1. Додайте на форму напис і текстове поле (елемент керування TextBox), помістивши їх під полем зі списком (рис. 2.21).

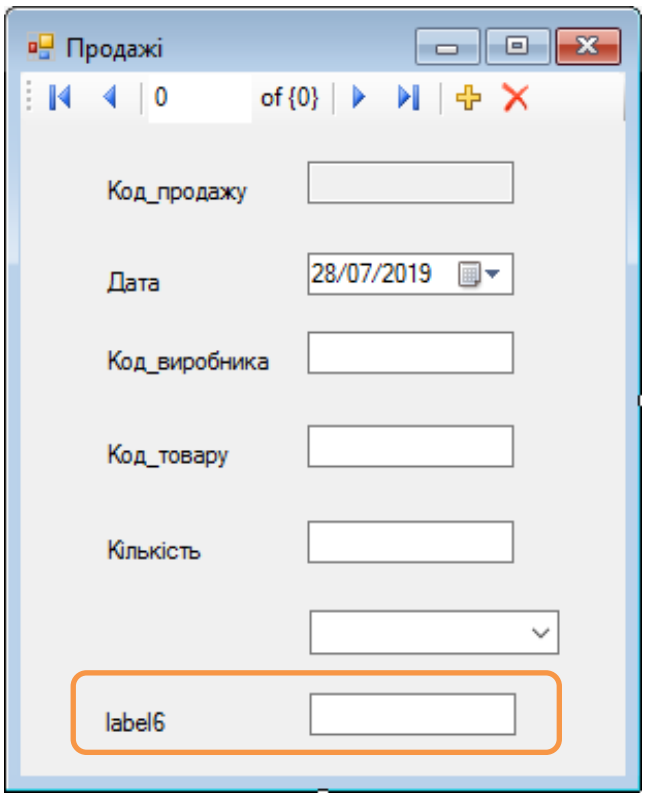


Рис. 2.21. **Напис і текстове поле на формі *Продажі***

2. Для нових елементів установіть значення властивостей, наведених у табл. 2.10.

Таблиця 2.10

Властивості нових елементів

Елементи	Властивості	Значення
Label	Text	Ціна
TextBox	Name	textBoxЦіна
	ReadOnly	True

3. Після останнього оператора тіла оброблювача події ***formПродажі_Load*** (перед закриваючою дужкою) додайте оператор, який дає функціональність текстовому полю ***Ціна***. Він має такий вигляд:

```
//Відображаємо ціну
textBoxЦіна.DataBindings.Add("Text", bindingТовари, "Ціна");
```

4. Запустіть програму на виконання й перевірте функціональність текстового поля ***Ціна***. Спочатку перегляньте кілька записів, а потім змініть код товару в будь-якому записі, вибираючи назву товару в полі зі списком ***comboBoxТовар***. Простежте за тим, як змінюється ціна.

5. Збережіть зміни, зроблені у проєкті.

4.4. Додавання обчислюваного стовпця в локальну таблицю

Завдання

Додайте на форму ***Продажі*** текстове поле ***Вартість*** разом із написом (рис. 2.22). У ньому відображено вартість товару. Її обчислюють шляхом множення ціни товару на його кількість. Ціну вибирають із батьківської таблиці ***Товари***, а кількість – із дочірньої. Вартість використовують тільки для перегляду та вона змінюється тут, якщо змінюється кількість проданого товару або його код.

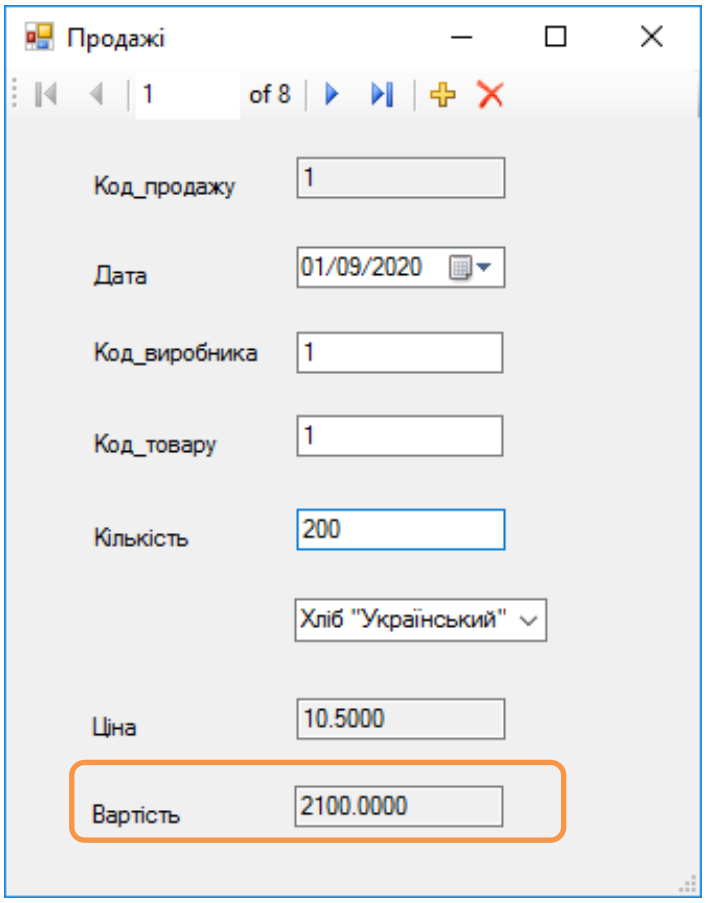


Рис. 2.22. **Форма *Продажі* з полем *Вартість***

Виконання

1. Додайте на форму напис і текстове поле, помістивши їх під ціною (рис. 2.23).

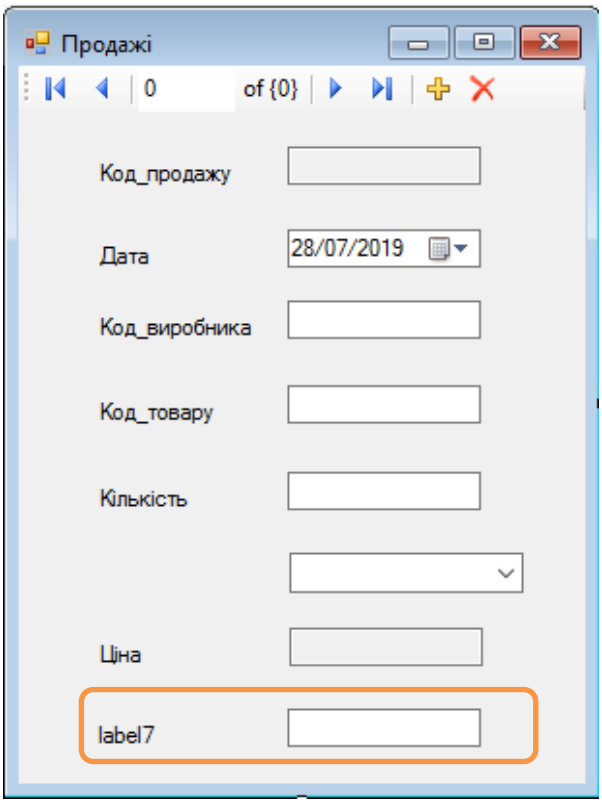


Рис. 2.23. **Напис і текстове поле для відображення вартості на формі *Продажі***

2. Для нових елементів установіть значення властивостей, наведених у табл. 2.11.

Таблиця 2.11

Властивості нових елементів управління

Елементи	Властивості	Значення
Label	Text	Вартість
TextBox	Name	textBoxВартість
	ReadOnly	True

3. Після останнього оператора тіла оброблювача події ***formПродажі_Load*** (перед закриваючою дужкою) додайте оператори, які дають функціональність текстовому полю ***Вартість***. Вони мають такий вигляд:

```
// Обчислюємо Вартість
DataColumnCollection cols = tableПродажі.Columns;
DataColumn column = cols.Add("Вартість",
  typeof(System.Decimal));
column.Expression = "Parent(ТовариПродажі).Ціна*Кількість";
textBoxВартість.DataBindings.Add("Text",bindingПродажі,
  "Вартість");
```

4. Запустіть програму на виконання й перевірте функціональність текстового поля ***Вартість***. Спочатку перегляньте кілька записів. Після цього змініть кількість товару в будь-якому записі, перейдіть на інший запис, а потім поверніться на змінений.

5. Збережіть зміни, зроблені у проєкті.

4.5. Використання поля зі списком Виробник

Завдання

Додайте на форму ***Продажі*** поле зі списком ***Виробник*** (рис. 2.24). Його використовують для введення в таблицю ***Продажі*** кодів виробників. Під час вибору із цього списку назви виробника в таблицю ***Продажі*** автоматично запишеться його код.

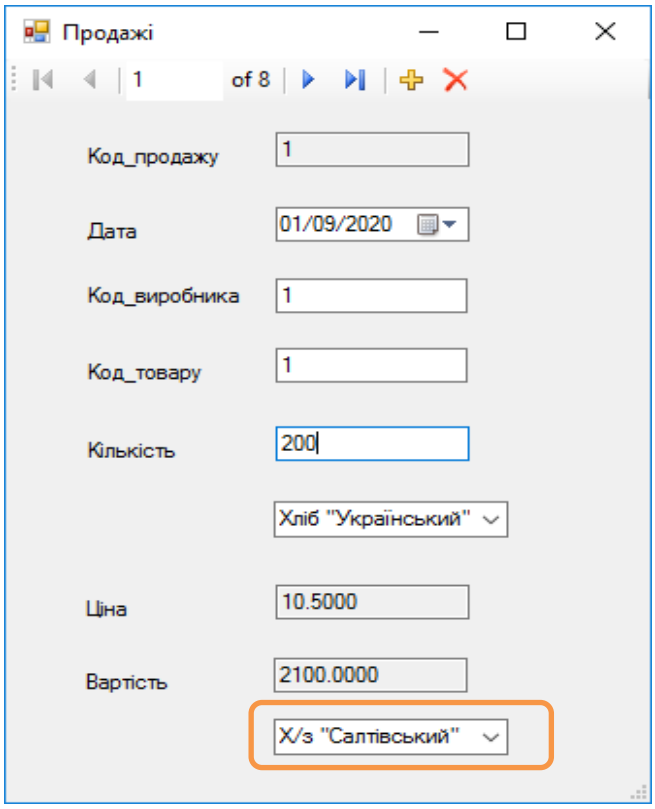


Рис. 2.24. **Форма *Продажі* з полем зі списком *Виробник***

Виконання

Виконайте дії, подібні до описаних у п. 4.2.

4.6. Доопрацювання форми

Завдання

Перемістіть елементи керування на формі ***Продажі*** так, щоб вони відображалися в логічно правильному порядку (рис. 2.25).

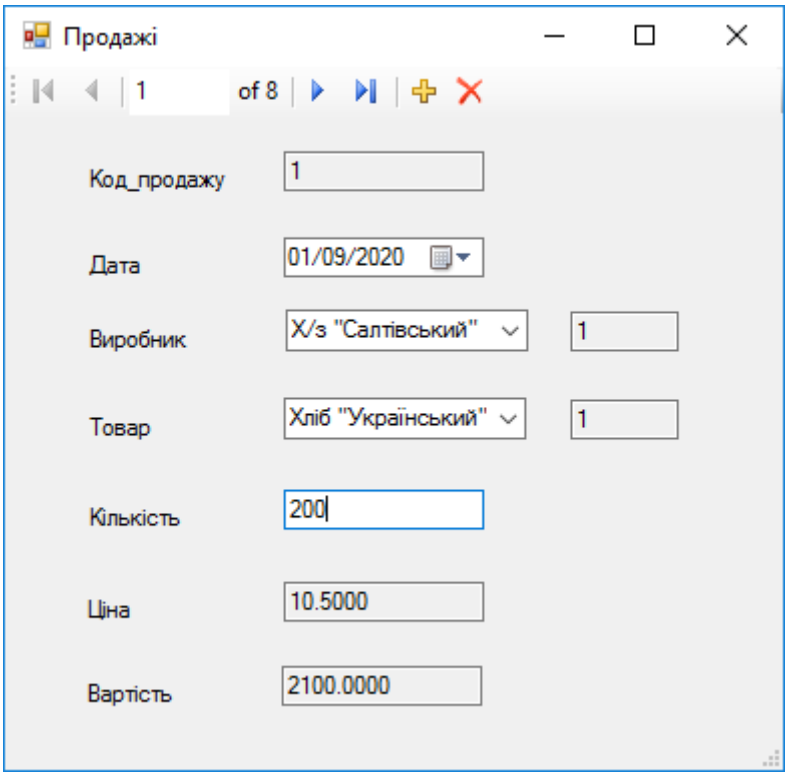


Рис. 2.25. **Форма *Продажі* з упорядкованими елементами**

Виконання

1. Змініть текст у написах ***Код_товару*** й ***Код_виробника***, відповідно, на ***Товар*** і ***Виробник***.

2. Установіть значення **True** для властивості **ReadOnly** в текстових полях ***textBoxКод_товару*** й ***textBoxКод_виробника***, а потім зменште їхню ширину.

3. Перемістіть елементи керування на формі, щоб їх відображали так, як зазначено на рис. 2.25.

4. Установіть однакову ширину в елементах одного стовпця.

5. Запустіть програму на виконання й перевірте функціональність усіх елементів керування на формі.

6. Збережіть зміни, зроблені у проєкті.

7. Відкрийте ще раз форму ***Продажі*** й переконайтеся в тому, що зміни даних не збереглися в базі даних. Вони відбуваються тільки в DataSet. Цю проблему вирішують у двох наступних завданнях.

4.7. Збереження даних

Завдання

Додайте на панель навігатора кнопку ***Зберегти***, яка зберігає в базі даних усі зміни даних (рис. 2.26).

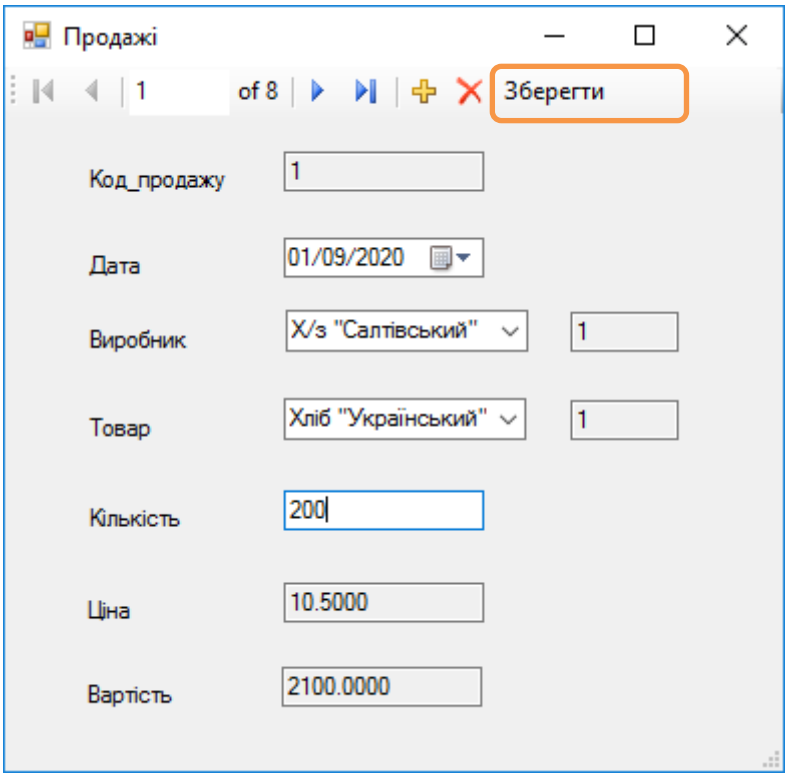


Рис. 2.26. **Кнопка *Зберегти* на формі *Продажі***

Виконання

1. Перебуваючи у вікні конструктора форми ***Продажі***, клацніть у вільному місці панелі навігатора і клацніть кнопку, що з'явилася праворуч.

2. Виберіть команду **Button** із меню значка, що з'явився (рис. 2.27).

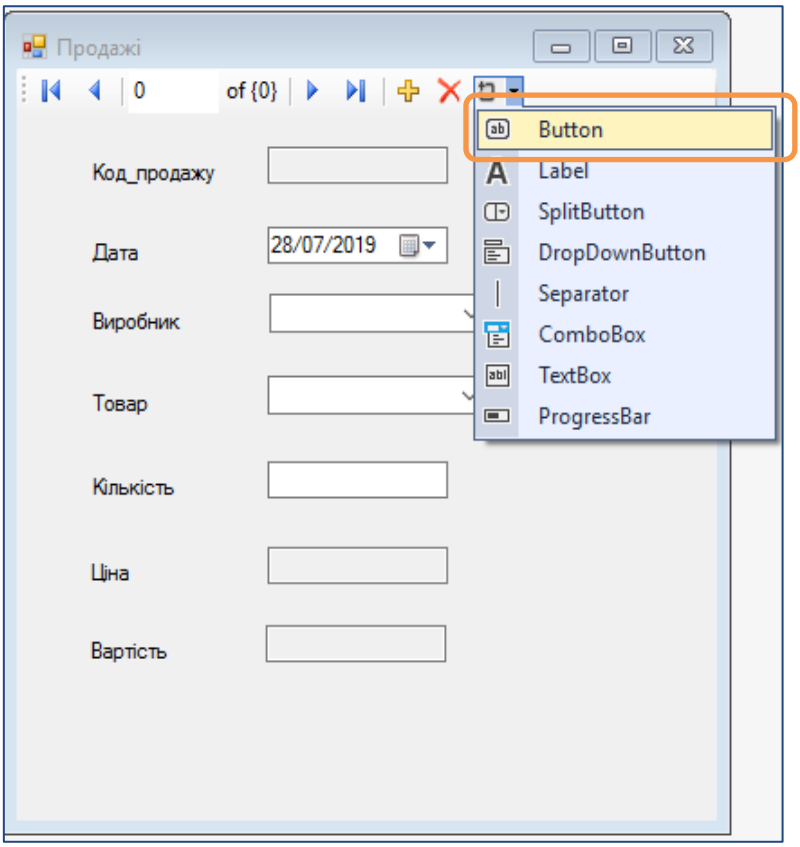


Рис. 2.27. **Команда Button у меню значка кнопки панелі навігатора**

3. Для нової кнопки на навігаторі встановіть значення властивостей, наведених у табл. 2.12.

Таблиця 2.12

Властивості нової кнопки

Властивості	Значення
DisplayStyle	Text
Text	Зберегти
Name	toolStripButtonЗберегти

5. Додайте код оброблювача події "Клацання кнопки Зберегти".

```
private void toolStripButtonЗберегти_Click(object sender, EventArgs e)
{
    // Завершуємо редагування й передаємо у БД всі зміни
    // у локальній таблиці
    bindingПродажі.EndEdit();
    daПродажі.Update(ds, "Продажі");
    MessageBox.Show("Зміни даних збережено", "Таблиця Продажі");
}
```

4. Перевірте функціональність кнопки ***Зберегти***.

5. Збережіть зміни, зроблені у проєкті.

4.8. Нагадування про необхідність у збереженні даних

Завдання

Додайте оброблювача події ***formПродажі_FormClosing***, щоб нагадати користувачеві про незбережені дані в базі.

Виконання

1. Клацніть у будь-якому вільному місці форми ***Продажі***.

2. У вікні властивостей знайдіть подію **FormClosing** і двічі клацніть на ній.

3. У вікні коду введіть оператори тіла оброблювача події ***formПродажі_FormClosing***. Він має такий вигляд:

```
private void formПродажі_FormClosing(object sender,
    FormClosingEventArgs e)
{
  //Завершуємо редагування
  bindingПродажі.EndEdit();

  //Перевіряємо, чи є незбережені дані
  if (ds.HasChanges())
  {
    //Запит на підтвердження збереження змін
    DialogResult result =
    MessageBox.Show("Зберегти усі зміни в таблиці?", "Продажі",
        MessageBoxButtons.YesNo, MessageBoxIcon.Warning,
        MessageBoxDefaultButton.Button2);

    if (result.ToString() == "Yes")
    {
      toolStripButtonЗберегти_Click(sender, e);
    }
    else
    {
      //Відмовляємося від видалення запису
      ds.Tables["Продажі"].RejectChanges();
    }
  }
}
```

4. Перевірте функціональність оброблювача події закривання форми ***Продажі***.

5. Збережіть зміни, зроблені у проєкті.

У звіті з лабораторної роботи подайте скриншот форми ***Продажі*** з даними, а також код оброблювача події ***formПродажі_Load***. Поясніть, чому в коді форми описано тільки одного будівника запитів для адаптера ***cbПродажі,*** тоді як використовують дані трьох таблиць – ***Товари***, ***Виробники*** та ***Продажі***.

5. Спільне ведення батьківської й дочірньої таблиць (таблиці Накладні та ТовариНакладних)

Ідеї та кроки виконання завдання

Вивчення засобів ADO.NET для спільного ведення зв'язаних таблиць здійснюють на прикладі документа ***Накладна***. Він супроводжує постачання товарами від виробника кіоск. Документ складається із двох частин – загальної та подробиць.

У загальній частині постачальник (виробник) указує номер накладної, дату, назву виробника й отримувача товарів. У подробицях записують дані про ті товарів, які доставив виробник у черговій партії (назву товару, ціну, кількість і вартість).

У базі даних документи ***Накладні*** зберігають у двох таблицях – ***Накладні*** й ***ТовариНакладних***. Вони зв'язані відношенням "один-до-багатьох". Оскільки ці дані становлять один документ, то в інтерфейсі користувача їх відображають на одній формі ***Накладні***. Усі операції з даними (перегляд, зміна, додавання та видалення) мають виконувати одночасно у двох таблицях. Розроблення застосунку із забезпечення цих операцій складається з таких кроків:

1. Створення форми *Накладні*.

2. Створення DataSet.

3. Прив'язування DataSet до елементів керування форми.

4. Доопрацювання елемента керування dataGridViewТовариНакладних.

5. Збереження змін.

5.1. Створення форми Накладні

Завдання

Додайте в застосунок форму ***Накладні***, у якій будуть відображати та змінювати дані таблиць ***Накладні*** й ***ТовариНакладних*** (рис. 2.28).

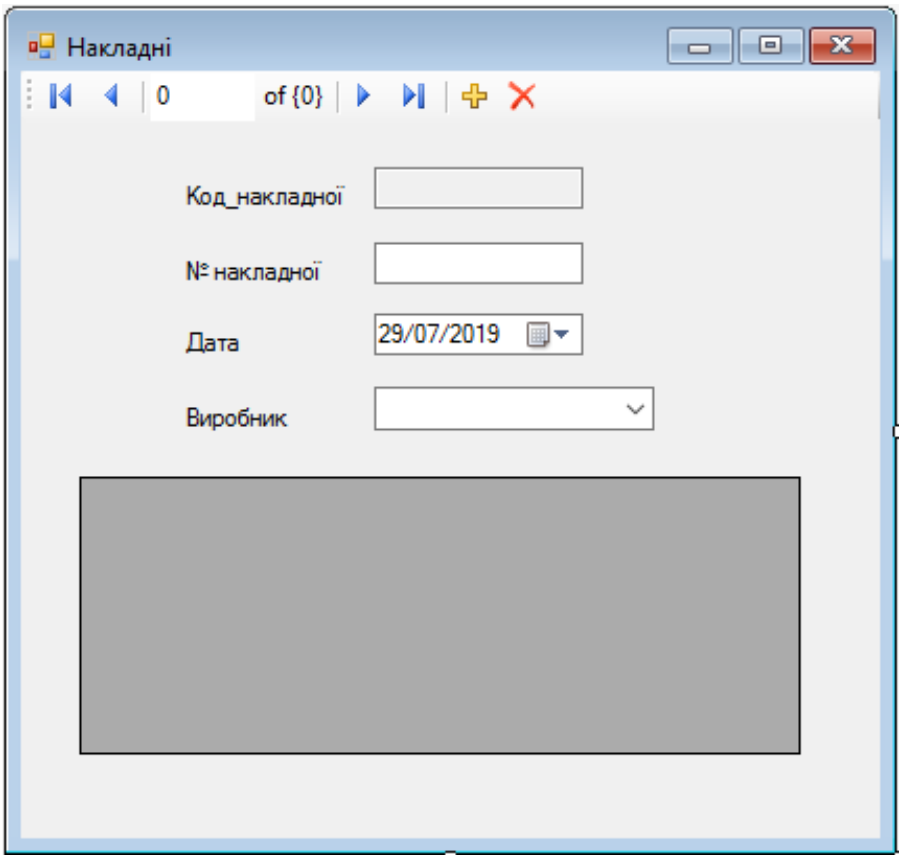


Рис. 2.28. **Форма *Накладні***

Виконання

1. Додайте в застосунок нову форму з ім'ям ***formНакладні*** й заголовком ***Накладні***.

2. Для відображення значень полів одного запису таблиці ***Накладні*** додайте на форму такі елементи керування: чотири написи (елемент керування Label) у стовпчик один під одним з лівого боку форми, а праворуч від написів додайте елементи керування для відображення даних у такому порядку: TextBox, TextBox, DateTimePicker, ComboBox (рис. 2.29).

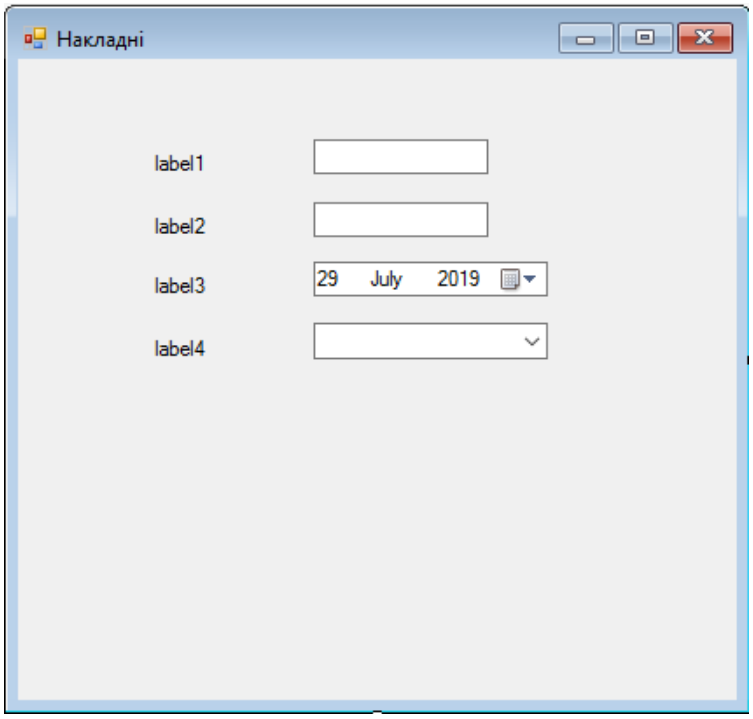


Рис. 2.29. **Початковий вигляд форми *Накладні***

3. Для відображення значень полів відповідних записів таблиці ***ТовариНакладних*** додайте на форму елемент керування DataGridView, помістивши його під раніше доданими елементами (рис. 2.30).

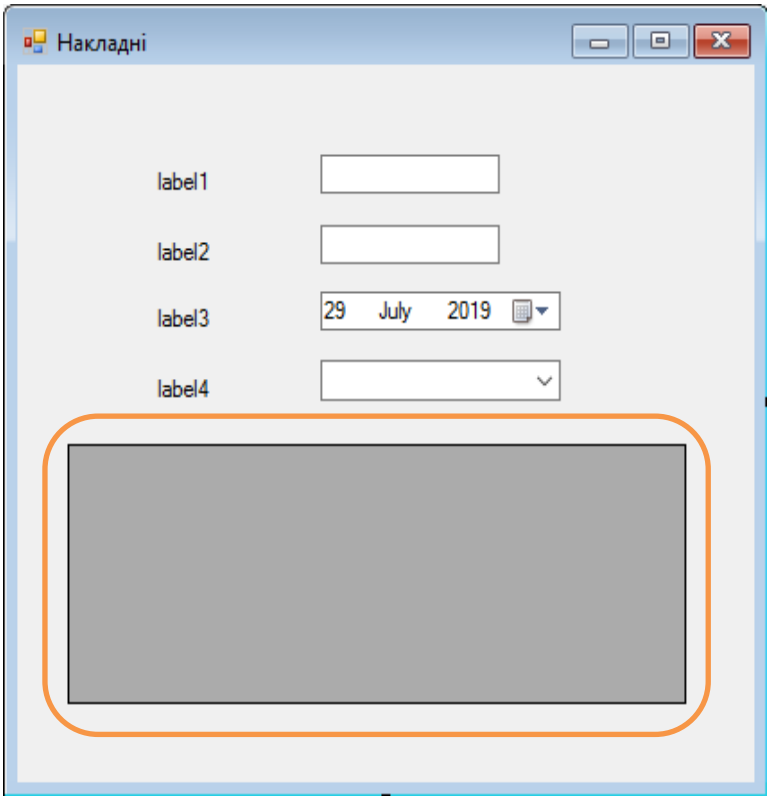


Рис. 2.30. **Елемент керування DataGridView на формі *Накладні***

4. Установіть для елементів керування значення властивостей, наведених у табл. 2.13.

Таблиця 2.13

Властивості елементів керування

№ п/п	Написи (властивості Text)	Елементи керування для відображення даних (властивості Name)
1	Код накладної	textBoxКод_накладної
2	№ накладної	textBoxНомер_накладної
3	Дата	dateTimePickerДата
4	Виробник	comboBoxВиробник
5	–	dataGridViewТовариНакладних

5. У текстовому полі ***textBoxКод_накладної*** додатково установіть значення **True** для властивості **ReadOnly**.

6. В елементі ***dateTimePickerДата*** встановіть значення **Short** для властивості **Format** і зменште його ширину, зробивши такою, як і в текстових полях.

7. Додайте на форму елемент керування **BindingNavigator** (розміщено в категорії **Data**) і встановіть значення ***bindingNavigatorНакладні*** для властивості **Name**.

8. Додайте код оброблювача події "Клацання кнопки Накладні" на формі ***Хліб***.

```
private void buttonНакладні_Click(object sender, EventArgs e)
{
    formНакладні вікноНакладні = new formНакладні();
    вікноНакладні.ShowDialog();
}
```

9. Перевірте функціональність кнопки ***Накладні*** на формі ***Хліб***.

10. Збережіть зміни, зроблені у проєкті.

5.2. Створення DataSet

Завдання

Створіть об'єкт DataSet, який забезпечує даними роботу форми ***Накладні***.

Ідеї виконання

Для роботи з документом ***Накладна*** на формі ***Накладні*** використовують дані двох таблиць ***Накладні*** й ***ТовариНакладних***. Їх потрібно завантажити в DataSet. Крім цього, для створення зручного інтерфейсу користувача знадобляться дані таблиці ***Виробники*** (джерело даних для елемента керування ***comboBoxВиробник***) і ***Товари*** (в елементі керування ***dataGridViewТовариНакладних*** як джерело даних для поля зі списком ***Товар***, поля ***Ціна***, обчислюваного поля ***Вартість***). Тому об'єкт DataSet має містити дані таблиць ***Накладні***, ***ТовариНакладних***, ***Виробники*** й ***Товари***.

Дані перших двох таблиць можуть змінюватися на формі. Об'єкти, пов'язані з ними, мають бути доступними в різних функціях класу ***formНакладні***. Тому їх описують у спільній частині класу.

Дані з останніх двох таблиць мають допоміжний характер і на формі змінюватися не будуть. Їх використовують тільки в оброблювачі події "Завантаження форми", тому описують у ньому.

Виконання

1. Перейдіть у вікно коду форми й установіть посилання на потрібні простори імен:

```
using System;
using System.Data;
using System.Text;
using System.Windows.Forms;
using System.Data.SqlClient;
```

2. Уведіть у вікні коду форми перед кодом конструктора public formНакладні() такий опис спільних об'єктів:

```
// Спільні об'єкти
SqlConnection conn;   //З'єднання
DataSet ds;             //Набір даних

// Таблиця Накладні
DataTable tableНакладні;      //Локальна таблиця
SqlDataAdapter daНакладні;    //Адаптер даних
SqlCommandBuilder cbНакладні; //Будівник запитів для адаптера

// Таблиця ТовариНакладних
DataTable tableТовариНакладних;      //Локальна таблиця
SqlDataAdapter daТовариНакладних;    //Адаптер даних
SqlCommandBuilder cbТовариНакладних; // Будівник для адаптера
```

3. Перейдіть у вікно конструктора форми ***Накладні*** та двічі клацніть у вільному місці форми.

4. Уведіть оператори тіла оброблювача події ***formНакладні_Load***. Він має такий вигляд:

```
private void formНакладні_Load(object sender, EventArgs e)
{
  // З'єднання SqlConnection
  conn = new SqlConnection();
  conn.ConnectionString = myPublic.connString;

  // Створюємо DataSet
  ds = new DataSet("dsХліб");

  //************************************
  //*    Батьківська таблиця Накладні    *
  //************************************

  // Створюємо таблицю Накладні
  tableНакладні = new DataTable("Накладні");

  //Додаємо таблицю Накладні в DataSet
  ds.Tables.Add(tableНакладні);

  //Створюємо SqlDataAdapter
  daНакладні = CreateTableAdapter("Накладні", conn);

  //Створюємо SqlCommandBuilder (надалі буде використовуватися
  // для збереження змін даних у батьківській таблиці)
  cbНакладні = new SqlCommandBuilder(daНакладні);

  //Заповнюємо таблицю Накладні у ds даними з бази
  daНакладні.Fill(tableНакладні);

  //****************************************
  //*    Дочірня таблиця ТовариНакладних   *
  //****************************************

  //Створюємо таблицю ТовариНакладних
  tableТовариНакладних = new DataTable("ТовариНакладних");

  //Додаємо таблицю ТовариНакладних у DataSet
  ds.Tables.Add(tableТовариНакладних);

  //Створюємо SqlDataAdapter
  daТовариНакладних = CreateTableAdapter("ТовариНакладних", conn);

  // Створюємо SqlCommandBuilder (надалі буде використовуватися
  // для збереження змін даних у дочірній таблиці)
  cbТовариНакладних = new SqlCommandBuilder(daТовариНакладних);
```

```
//Заповнюємо таблицю ТовариНакладних у ds даними з бази
daТовариНакладних.Fill(tableТовариНакладних);

//Установлюємо зв'язок між таблицями Накладні й ТовариНакладних
DataRelation relationНакладніТовариНакладних =
  new DataRelation("НакладніТовариНакладних",
  new DataColumn[] { tableНакладні.Columns["Код_накладної"] },
  new DataColumn[]{ tableТовариНакладних.Columns["Код_накладної"] },
  true);
ds.Relations.Add(relationНакладніТовариНакладних);

//**********************************
//*   Допоміжна таблиця Виробники   *
//**********************************

//Створюємо таблицю Виробники
DataTable tableВиробники = new DataTable("Виробники");

//Додаємо таблицю Виробники в DataSet
ds.Tables.Add(tableВиробники);

//Створюємо SqlDataAdapter
SqlDataAdapter daВиробники = CreateTableAdapter("Виробники", conn);

//Заповнюємо таблицю Виробники у ds даними з бази
daВиробники.Fill(tableВиробники);

//Установлюємо зв'язок між таблицями Виробники й Накладні
DataRelation relationВиробникиНакладні =
  new DataRelation("ВиробникиНакладні",
  new DataColumn[] { tableВиробники.Columns["Код_виробника"] },
  new DataColumn[] { tableНакладні.Columns["Код_виробника"] });
ds.Relations.Add(relationВиробникиНакладні);

//*******************************
//*   Допоміжна таблиця Товари   *
//*******************************

//Створюємо таблицю Товари
DataTable tableТовари = new DataTable("Товари");

//Додаємо таблицю Товари в DataSet
ds.Tables.Add(tableТовари);

//Створюємо SqlDataAdapter
SqlDataAdapter daТовари = CreateTableAdapter("Товари", conn);

//Заповнюємо таблицю Товари у ds даними з бази
daТовари.Fill(tableТовари);
```

```
  //Установлюємо зв'язок між таблицями Товари й ТовариНакладних
  DataRelation relationТовариТовариНакладних =
    new DataRelation("ТовариТовариНакладних",
    new DataColumn[] { tableТовари.Columns["Код_товару"] },
    new DataColumn[] { tableТовариНакладних.Columns["Код_товару"]});
  ds.Relations.Add(relationТовариТовариНакладних);
}
```

5. Додайте код функції користувача ***CreateTableAdapter***, розмістивши його під кодом оброблювача події formТовари_Load. Він має такий вигляд:

```
public static SqlDataAdapter CreateTableAdapter(
    String table,
    SqlConnection connection)
{
  SqlDataAdapter dataAdapter = new SqlDataAdapter();
  SqlCommand command;

  // Властивість SelectCommand
  command = new SqlCommand("SELECT * FROM " + table, connection);
  dataAdapter.SelectCommand = command;

  return dataAdapter;
}
```

6. Перевірте функціональність кнопки ***Накладні*** на формі ***Хліб*** та оброблювача події ***formНакладні_Load*** на відсутність синтаксичних помилок. Після завантаження форми ***Накладні*** не мають відображати дані й повідомлення про помилки.

7. Збережіть зміни, зроблені у проєкті.

5.3. Прив'язування DataSet до елементів керування форми

Завдання

Відобразіть дані з DataSet на формі ***Накладні*** (рис. 2.31).

Ідеї виконання

Щоб забезпечити користувача доступом до даних, які перебувають у DataSet, необхідно прив'язати ці дані до елементів форми. Прив'язування до елементів загальної частини форми здійснюють аналогічно формі ***Продажі*** в п. 4, а подробиць (елемент ***dataGridViewТовариНакладних***) – аналогічно формі ***Товари*** в п. 2.

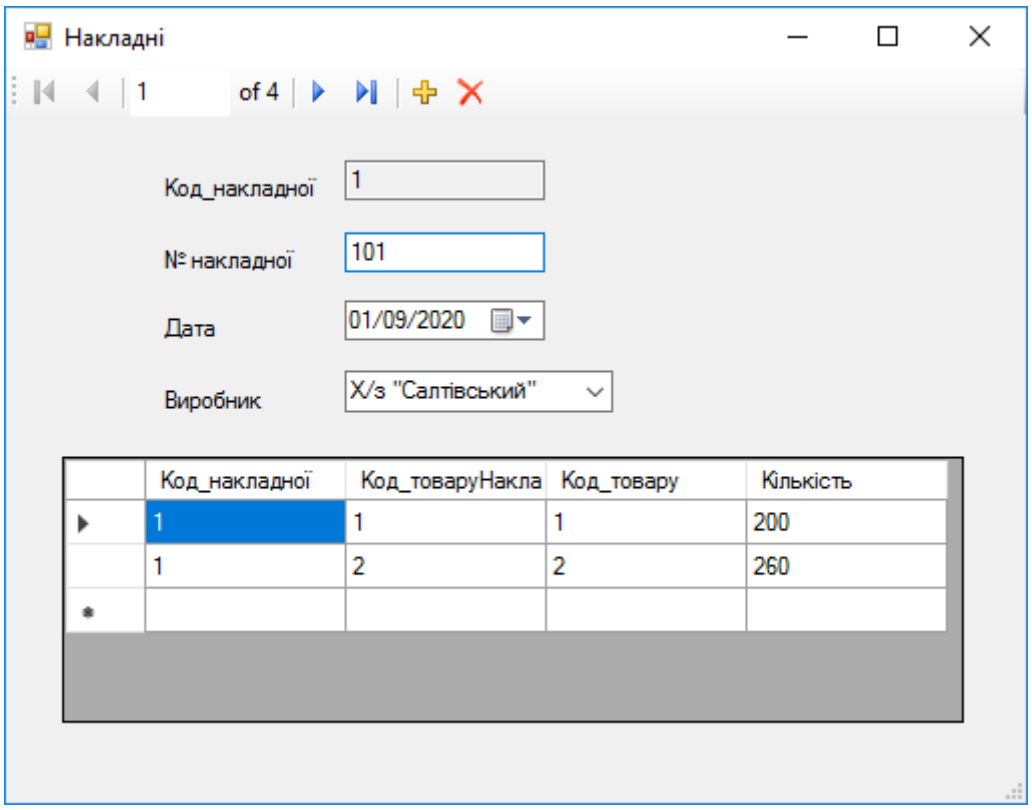


Рис. 2.31. **Форма *Накладні* з даними в початковому вигляді**

На формі постійно відображено один запис таблиці ***Накладні*** та відповідні йому записи з таблиці ***ТоваріНакладних***. Об'єкт ***bindingНакладні***, який забезпечує прив'язку DataSet до елементів керування, описують у загальній частині класу ***formНакладні***. Його використовують у декількох функціях (для відображення даних і збереження їхніх змін).

Щоб в елементі ***dataGridViewТоваріНакладних*** відображалися тільки ті дані, які відповідають вибраній накладній, досить у його властивості **DataMember** указати назву відношення між батьківською й дочірньою таблицями ***НакладніТоваріНакладних***.

Установлення прив'язування з об'єкта ***bindingНакладні*** до навігатора дає можливість переміщатися записам таблиці ***Накладні***, тобто по черзі переглядати відповідні документи.

Виконання

1. Уведіть у вікні коду форми перед кодом конструктора public formНакладні() такий опис об'єкта ***bindingНакладні***:

```
// З'єднувачі локальних таблиць з елементами форми
BindingSource bindingНакладні;
BindingSource bindingТоваріНакладних;
```

2. Уведіть оператори, які прив'язують дані з DataSet до елементів керування форми, додавши їх після останнього оператора тіла оброблювача події ***formНакладні_Load*** (перед закриваючою дужкою). Вони мають такий вигляд:

```
//*****************************************
//*   Прив'язування до елементів керування  *
//*****************************************

// Створюємо об'єкт
bindingНакладні = new BindingSource();
// Установлюємо прив'язку до таблиці
bindingНакладні.DataSource = tableНакладні;

// Налаштовуємо на прив'язку елементи на формі
textBoxКод_накладної.DataBindings.Add("Text", bindingНакладні,
     "Код_накладної");
textBoxНомер_накладної.DataBindings.Add("Text", bindingНакладні,
     "Номер_накладної");
dateTimePickerДата.DataBindings.Add("Value", bindingНакладні,
     "Дата", true);

//*** Прив'язування до comboBoxВиробник ***
// Створюємо прив'язку до таблиці Виробники у ds
BindingSource bindingВиробники = new BindingSource();
// Джерело даних прив'язування
bindingВиробники.DataSource = tableВиробники;
// Джерело рядків comboBox
comboBoxВиробник.DataSource = bindingВиробники;
// Відображуване поле з рядків джерела
comboBoxВиробник.DisplayMember = "Виробник";
// Указуємо, за значеннями якого поля даних виконують прив'язування
comboBoxВиробник.ValueMember = "Код_виробника";
// Додаємо в колекцію прив'язування, указавши, що результатом вибору
// у списку comboBoxВиробник буде  Код_виробника
comboBoxВиробник.DataBindings.Add("SelectedValue", bindingНакладні,
    "Код_виробника");

//*** Прив'язуємо dataGridViewТовариНакладних до відповідних
//    записів таблиці ТовариНакладних ***
bindingТовариНакладних = new BindingSource();
bindingТовариНакладних.DataSource = tableТовариНакладних;
// Джерело даних dataGridView
dataGridViewТовариНакладних.DataSource = bindingНакладні;
// Указуємо за значеннями якого зв'язку (відношення) виконують
//  прив'язування. Конкретні дані - тільки відповідні дочірні записи
dataGridViewТовариНакладних.DataMember =
    "НакладніТовариНакладних";

// Налаштовуємо на прив'язування навігатор записами
bindingNavigatorНакладні.BindingSource = bindingНакладні;
```

3. Перевірте функціональність форми ***Накладні***. Після її завантаження має відображати перший запис таблиці ***Накладні*** та відповідні записи таблиці ***ТовариНакладних***, а за допомогою навігатора можна переглянути всі інші.

4. Збережіть зміни, зроблені у проєкті.

5.4. Доопрацювання елемента керування dataGridViewТовариНакладних

Завдання

Додайте в елемент ***dataGridViewТовариНакладних*** стовпець поле зі списком ***Товар***, яке спрощує введення коду товару, і вилучіть стовпці ***Код_товару***, ***Код_товару_накладної*** й ***Код_накладної*** (рис. 2.32).

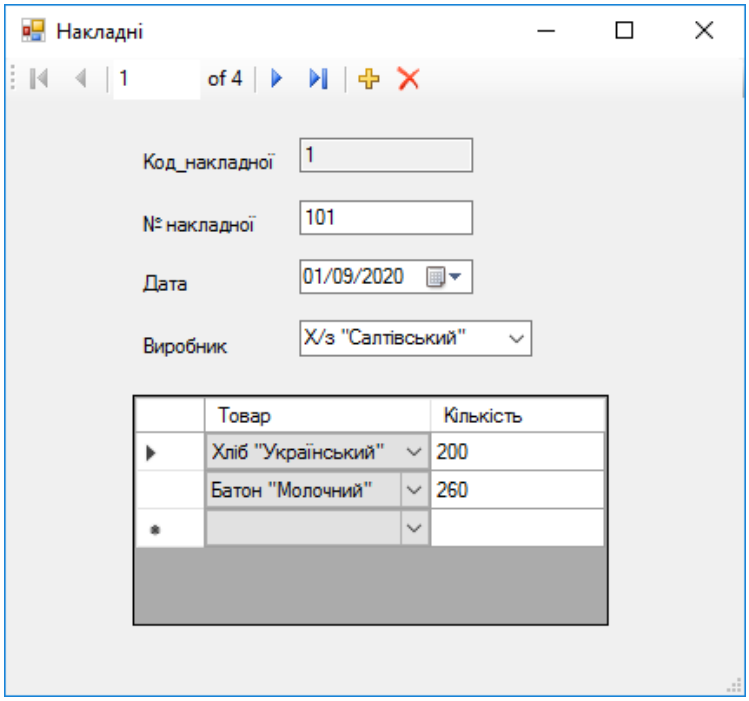


Рис. 2.32. **Доопрацьована форма *Накладні***

Ідеї виконання

Стовпець полів зі списком ***Товар*** реалізують через клас DataGridViewComboBoxColumn. Він є стовпцем об'єктів DataGridViewComboBoxCell (клітинка типу ComboBox у кожному рядку). Стовпець списків, що розкриваються, можна зв'язати зі своїм власним джерелом даних (таблиця ***Товари***), задавши властивість **DataSource**.

Властивість **DisplayMember** визначає поле бази даних, відображене в ComboBox (поле ***Товар***).

Властивість **ValueMember** визначає поле, яке є його значенням (поле ***Код_товару***).

Властивість **DataPropertyName** відповідає за прив'язування до джерела даних (поле ***Код_товару***).

Стовпець DataGridViewComboBoxColumn можна додати в кінці елемента DataGridView за допомогою методу **Add** або вставити на певне місце за допомогою методу **Insert** (нумерацію починають із нуля).

Текстові поля в елементі ***dataGridViewТовариНакладних*** видаляють методом **Remove** з колекції **Columns**.

Виконання

1. Збільште ширину форми ***Накладні*** й елемента ***dataGridViewТовариНакладних***, щоб відображався новий стовпець.

2. Уведіть оператори, які створюють стовпець полів зі списком ***Товар*** і прив'язують до поля ***Код_товару*** локальної таблиці ***ТовариНакладних***. Для цього додайте такі оператори після останнього оператора тіла оброблювача події ***formНакладні_Load*** (перед закриваючою дужкою):

```
//*************************************************************
//  Доопрацювання елемента керування dataGridViewТовариНакладних  *
//*************************************************************

// Створюємо стовпець полів зі списком Товар
DataGridViewComboBoxColumn comboColumn =
    new DataGridViewComboBoxColumn();
comboColumn.DataPropertyName = "Код_товару";
comboColumn.HeaderText = "Товар";
comboColumn.Width = 130;
comboColumn.DataSource = tableТовари;
comboColumn.ValueMember = "Код_товару";
comboColumn.DisplayMember = "Товар";
//Уставляємо стовпець у dataGridViewТовариНакладних
dataGridViewТовариНакладних.Columns.Insert(3,comboColumn);
```

3. Перевірте функціональність елемента ***dataGridViewТовариНакладних*** на формі ***Накладні***. Після її завантаження має відображати перший запис таблиці ***Накладні*** та відповідні записи таблиці ***ТовариНакладних***. Зверніть увагу на те, що назви товарів відповідають кодам. Спробуйте змінити назву товару в полі зі списком і простежте за зміною коду. Зробіть висновок про надлишковість стовпця ***Код_товару***. Про його вилучення розказано в наступних пунктах.

4. Щоб вилучити стовпці ***Код_товару***, ***Код_товаруНакладної*** й ***Код_накладної*** в елементі ***dataGridViewТовари Накладних***, додайте

такі оператори після останнього оператора тіла оброблювача події ***formНакладні_Load*** (перед закриваючою дужкою):

```
// Видаляємо стовпці Код_товару, Код_товаруНакладної
// й Код_накладної
dataGridViewТовариНакладних.Columns.Remove("Код_товару");
dataGridViewТовариНакладних.Columns.Remove("Код_товаруНакладної");
dataGridViewТовариНакладних.Columns.Remove("Код_накладної");
```

5. Перевірте функціональність елемента ***dataGridViewТовариНакладних*** на формі ***Накладні***.

6. У разі потреби зменште ширину форми ***Накладні*** й елемента ***dataGridViewТовариНакладних***.

7. Збережіть зміни, зроблені у проєкті.

5.5. Збереження змін

Завдання

Забезпечте збереження всіх змін даних в обох таблицях із додаванням кнопки ***Зберегти*** на панель навігатора (рис. 2.33).

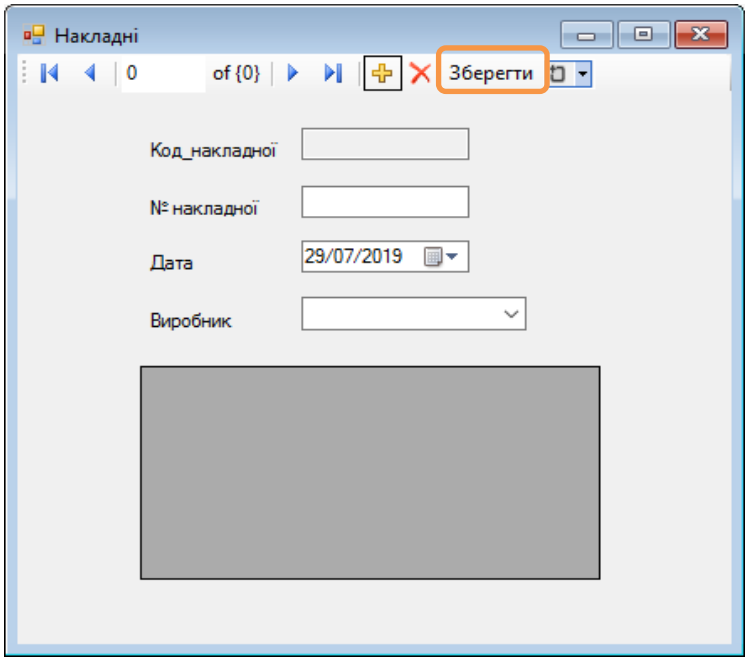


Рис. 2.33. **Кнопка *Зберегти* на формі *Накладні***

Ідеї виконання

Для збереження змін в ієрархічно пов'язаних таблицях збереження записів найчастіше виконують у такому порядку: спочатку зберігають дані в дочірній таблиці, а потім – у батьківській. Винятком є додавання в обох

таблицях. У такому разі спочатку зберігають дані в батьківській таблиці, а потім – у дочірній.

Для останнього скористайтеся таким алгоритмом. Після клацання кнопки **Add new** на навігаторі з'являється запит про те, що користувач дійсно хоче додати нову накладну. У разі позитивної відповіді запис із порожньою новою накладною зберігають у базі даних, де їй надано значення ключового поля ***Код_накладної***. Потім цей запис додають у локальну таблицю, а запис із порожнім значенням ключа видаляють із локальної таблиці. Нову накладну можна змінювати у спільній частині й додавати нові записи до детальної частини. Тобто ситуацію зведено до збереження змін спочатку в дочірній таблиці, а потім – у батьківській.

Виконання

1. Перейдіть у вікно конструктора форми ***Накладні***, клацніть у вільному місці панелі навігатора.

2. Із меню значка, що з'явився, виберіть команду **Button**.

3. Для нової кнопки на навігаторі установіть значення властивостей, наведених у табл. 2.14.

Таблиця 2.14

Властивості елементів керування

Властивості	Значення
DisplayStyle	Text
Text	Зберегти
Name	toolStripButtonЗберегти

4. Додайте код оброблювача події "Клацання кнопки Зберегти".

```
private void toolStripButtonЗберегти_Click(object sender,
    EventArgs e)
{
    // Завершуємо редагування й зберігаємо всі зміни у БД -
    // спочатку в батьківській таблиці, а потім - у дочірній

    bindingТовариНакладних.EndEdit();
    bindingНакладні.EndEdit();

    daТовариНакладних.Update(ds, "ТовариНакладних");
    daНакладні.Update(ds, "Накладні");
    MessageBox.Show("Зміни даних збережено",
        "Таблиці Накладні та ТовариНакладних");
}
```

5. Додайте код оброблювача події "Клацання кнопки AddNewItem" на навігаторі (кнопку позначено символом "**+**").

```
private void bindingNavigatorAddNewItem_Click(object sender,
    EventArgs e)
{
  // Запит на підтвердження збереження нової накладної
  DialogResult result =
    MessageBox.Show("Нова накладна?", "Накладні",
    MessageBoxButtons.YesNo, MessageBoxIcon.Warning,
    MessageBoxDefaultButton.Button2);

  if (result.ToString() == "Yes")
  {
    // Завершуємо редагування
    bindingНакладні.EndEdit();

    // Зберігаємо порожню нову накладну
    daНакладні.Update(ds, "Накладні");

    //*************************************************
    // Заміняємо в локальній таблиці нову накладну         *
    // з порожнім кодом на накладну із заповненим кодом    *
    //*************************************************

    // Додаємо в локальну таблицю  із БД порожню нову накладну.
    // У ній уже є код
    int numberAddRow = tableНакладні.Rows.Count - 1;
    daНакладні.Fill(ds, numberAddRow, numberAddRow, "Накладні");

    // Видаляємо з локальної таблиці нову накладну з порожнім кодом
    ds.Tables["Накладні"].Rows[numberAddRow].Delete();
    ds.Tables["Накладні"].AcceptChanges();
    }
}
```

6. Перевірте функціональність кнопки ***Зберегти***.

7. Збережіть зміни, зроблені у проєкті, і закрийте його.

У звіті з лабораторної роботи подайте скриншот форми ***Накладні*** з даними, а також код оброблювача події ***formНакладні _Load***. Опишіть

операції, які виконують у проєкті після ствердної відповіді на запитання "Нова накладна?".

Завдання для самостійного виконання

1. Перетворіть форму ***Товари*** на "розділену" форму (рис. 2.34).

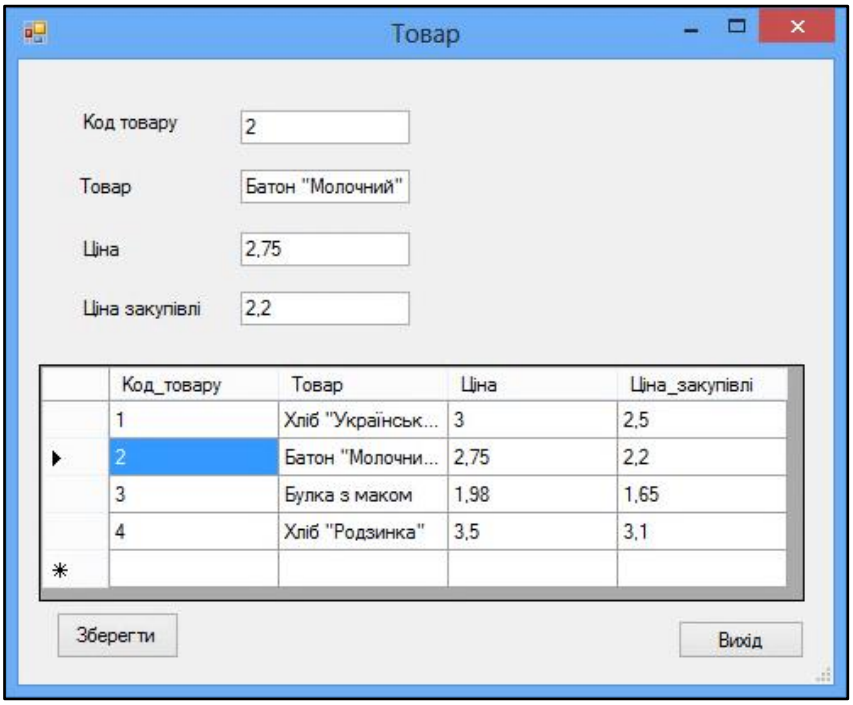


Рис. 2.34. "**Розділена" форма *Товари***

2. Додайте стовпець ***Продано*** на форму ***Товари***. У стовпці ***Продано*** відобразіть загальну кількість проданих товарів кожного виду (рис. 2.35).

Рис. 2.35. **Форма *Товари* зі стовпцем *Продано***

Ці дані вибирають із дочірньої таблиці ***Продажі***, а потім їх підсумовують.

3. Після виконання п. 2 на форму ***Товари*** ще додайте стовпці ***Отримано*** й ***Залишок***. У стовпці ***Отримано*** відобразіть загальну кількість отриманих за накладними товарів кожного виду, а ***Залишок*** – різницю значень у стовпцях ***Отримано*** та ***Продано***.

Примітка. Після виконання завд. 3 перевірте, чи зберігаються зміни (оновлення, додавання та видалення) в обох таблицях бази даних.

4. Перетворіть елемент ***dataGridViewТовариНакладних*** на формі ***Накладні***, щоб він більше відповідав частині подробиць документа ***Накладна***, додавши стовпці ***Ціна*** та ***Вартість*** (рис. 2.36).

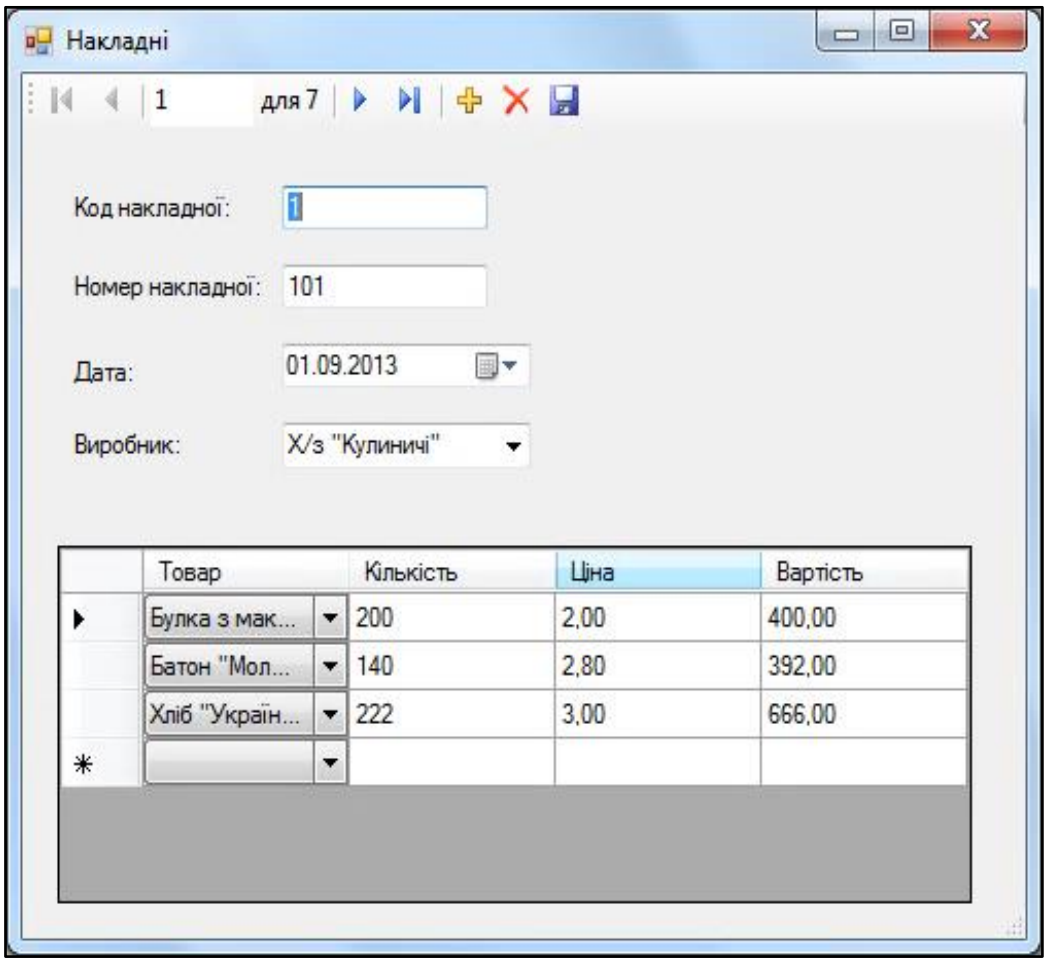


Рис. 2.36. **Стовпці *Ціна* та *Вартість* на формі *Товари***

5. Помістіть на формі ***Накладні*** поле, у якому відображено загальну суму вартостей товарів за накладною.

6. Попередньо додайте в базу даних таблицю ***Групи*** з полями ***Код_групи*** та ***Група*** і зв'язати її за зовнішнім ключем із таблицею ***Товари***.

6.1. Додайте поле зі списком ***Група*** на форму ***Продажі***, щоб після вибору групи у списку ***Група*** відображалися товари у списку ***Товар*** тільки вибраної групи.

6.2. Додайте поле зі списком ***Група*** в елемент ***dataGridView-ТовариНакладних*** на формі ***Накладні***, щоб після вибору групи у списку ***Група*** відображалися товари у списку ***Товар*** тільки вибраної групи.

7. У проєкті з індивідуальної бази даних застосуйте прийоми, розглянуті в базовій частині лабораторної роботи, а також у завданнях для самостійного виконання.

8. Створіть застосунок для роботи в роз'єднаному середовищі з індивідуальною базою даних, у якому довідкові таблиці перебувають в одній базі даних, а оперативні – в іншій.

9. Дослідіть експериментальним шляхом, від яких зв'язків між таблицями (об'єктів класу DataRelation) у коді форми ***Накладні*** можна відмовитися без утрати функціональності застосунку, а від яких – ні. Дайте відповідні пояснення.

Лабораторна робота 3
Розробка програм на основі інтерфейсу JDBC

Цілі лабораторної роботи:

1. Набуття практичних навичок у створенні з'єднань із базою даних мовою Java.
2. Освоєння сценаріїв виконання CRUD-операцій мовою Java.
3. Набуття практичних навичок в отриманні метаданих об'єкта ResultSet.
4. Оволодіння технологією пакетних і підготовлених запитів.
5. Набуття практичних навичок у роботі зі збереженими процедурами засобами мови Java.
6. Удосконалення навичок у роботі з реляційними базами даних.

Перед виконанням роботи студент має знати:

принципи проєктування реляційних баз даних;
основи побудови SQL-запитів;
навички роботи зі збереженими процедурами;
основи програмування мовою Java.

Після виконання роботи студент має вміти:

самостійно розробляти прості Java-застосунки з базами даних на основі інтерфейсу JDBC;
самостійно встановлювати з'єднання з базою даних у Java-застосунках;
використовувати збережені процедури під час розроблення програм.

Підготовча частина

Хід роботи

1. Створення бази даних.
2. Створення кнопкової форми застосунку.
3. Компоненти підключення до бази даних.
4. Використання об'єкта ResultSet.
5. Операції CRUD для довідкової таблиці.
6. Використання звичайних, підготовлених і пакетних запитів.
7. Робота зі збереженими процедурами.

Форма звітності

За результатами виконання роботи необхідно оформити електронний звіт засобами Word. За кожним завданням слід зробити скриншоти з результатами виконання, а також подати код, що реалізує операцію. У кінці деяких завдань зазначено, що слід надати пояснення. Аналогічним чином оформити звіт із виконання кожного завдання з п. "Завдання для самостійного виконання".

Критерії оцінювання

У 12-бальній системі оцінку результату захисту лабораторної роботи формують за такими правилами:

1. За всі п. 1 – 7 практичної частини може бути виставлено від 0 до 4 балів.

2. За виконання завдань п. "Завдання для самостійного виконання" може бути виставлено:

від 0 до 1 бала за кожне завдання 1, 2 та з діапазону 3.1 – 3.4;

від 0 до 2 балів за завдання 3.5;

від 0 до 7 балів за завдання 5;

від 0 до 3 балів за завдання 6.1;

від 0 до 5 балів за завдання 6.2.

3. За кілька варіантів виконання одного із завдань додають 1 бал.

4. За вибір варіанта, який із кількох варіантів виконання є оптимальним, та обґрунтування вибору додають 1 бал.

5. За виконання кожного завдання в іншій СУБД (Oracle, MySQL, DB2, PostgreSQL тощо) додають по 1 балу.

6. За запізнення із захистом лабораторної роботи знімають 2 бали за кожний тиждень.

Набрану кількість балів із відповідей на кожне завдання лабораторної роботи підсумовують. У результаті такого підрахунку студент може набрати від 0 до 12 балів.

У разі запізнення із захистом лабораторної роботи понад три тижні студент виконує завдання роботи на основі індивідуальної бази даних. Роботу оцінюють за описаними раніше критеріями.

Практична частина

Постановка загального завдання

Вивчення засобів роботи з даними мовою Java на основі інтерфейсу JDBC здійснюють шляхом створення проєкту ***jdbcMiniMarket***. Дані

зберігають у базі даних ***MiniMarket*** під керівництвом СУБД Java DB. Базу даних створюють у процесі виконання лабораторної роботи. Вона має ту саму схему, що й база даних ***Хліб***, яку використовували в попередніх лабораторних роботах.

Керування застосунком здійснюють за допомогою кнопкової форми ***MiniMarket*** (рис. 3.1).

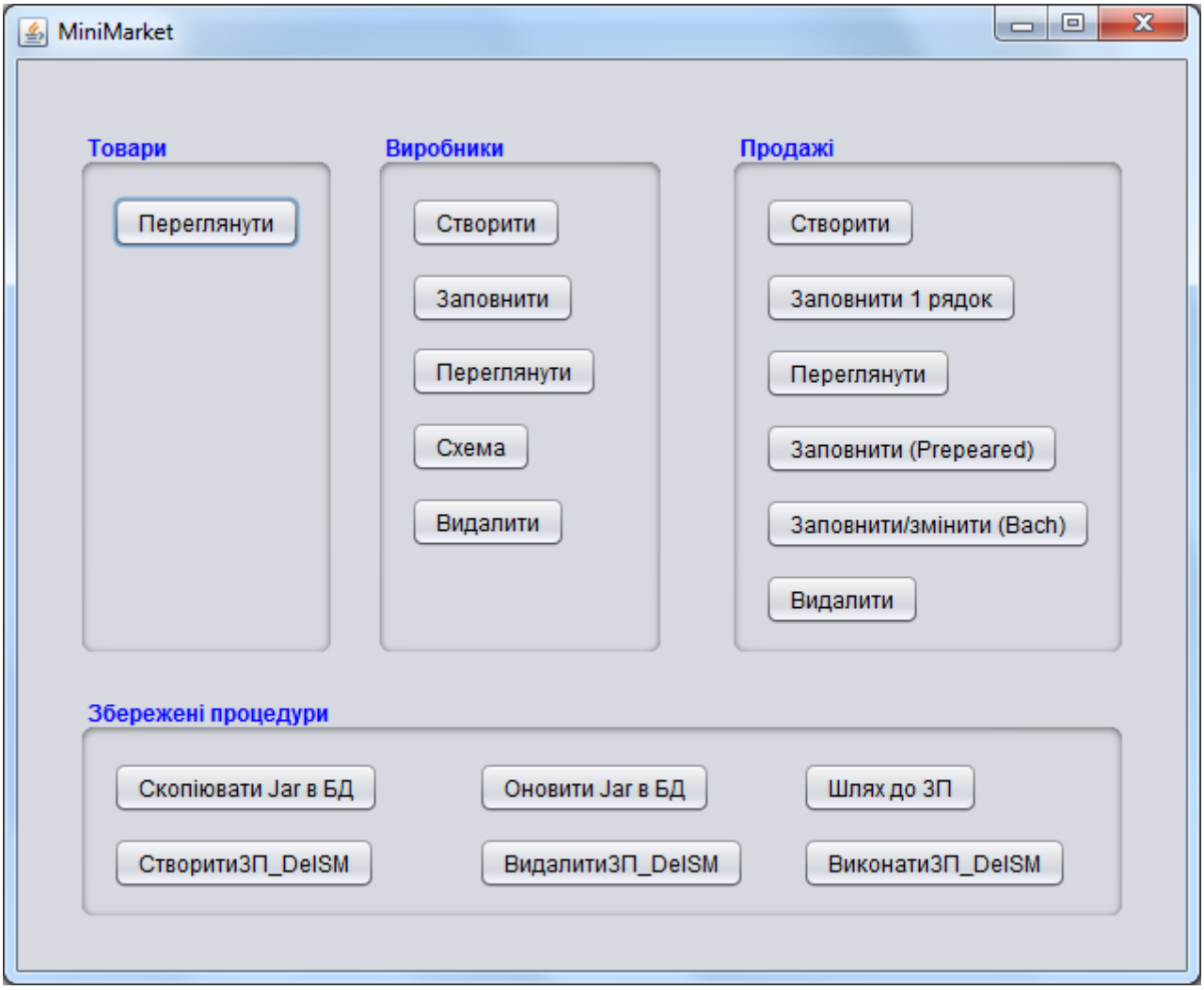


Рис. 3.1. **Кнопкова форма *MiniMarket***

Кнопки форми ***MiniMarket*** призначено для виконання операцій з однойменною базою даних. У заголовках панелей (Товари, Виробники та Продажі) зазначено таблиці, із якими виконують операції. Результати операцій подають у вікні виведення середовища NetBeans. У нижній панелі форми подано кнопки, що викликають операції під час роботи зі збереженою процедурою.

1. Створення бази даних

Завдання

У СУБД Java DB створіть базу даних ***MiniMarket***, а в ній створіть та заповніть таблицю ***Товари***.

Виконання

1. Відкрийте середовище розроблення застосунків NetBeans.
2. Створіть базу даних ***MiniMarket***. Для цього:

2.1. Перейдіть у вікно **Службы** й розкрийте вузол **Базы данных**.

2.2. Клацніть правою клавішею мишки на значок **Java DB** і з контекстового меню виберіть команду **Запустите сервер**.

2.3. Клацніть ще раз правою клавішею мишки на значок **Java DB** і з контекстового меню виберіть команду **Создать базу данных**.

2.4. Уведіть значення в текстові поля вікна **Создать базу данных Java DB**, наведені в табл. 3.1.

Таблиця 3.1

Значення текстових полів

Текстові поля	Значення
Имя базы данных	MiniMarket
Имя пользователя	student
Пароль	student
Подтверждение пароля	student

Потім клацніть кнопку **ОК** (рис. 3.2).

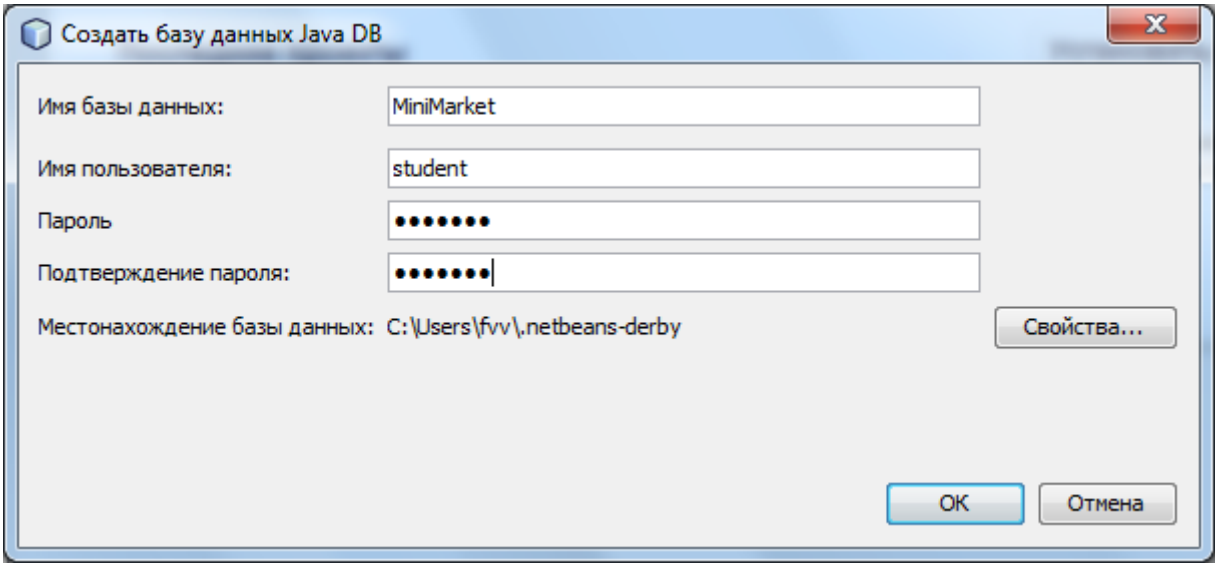


Рис. 3.2. **Вікно *Создать базу данных Java DB***

Примітка. Запам'ятайте ім'я папки, у якій зберігають базу даних. Воно вказано в написі **Местонахождение базы данных** (рис. 3.2).

У вузлі **Java DB** з'явився значок бази даних ***MiniMarket***, а нижче – значок підключення до бази даних (рис. 3.3).

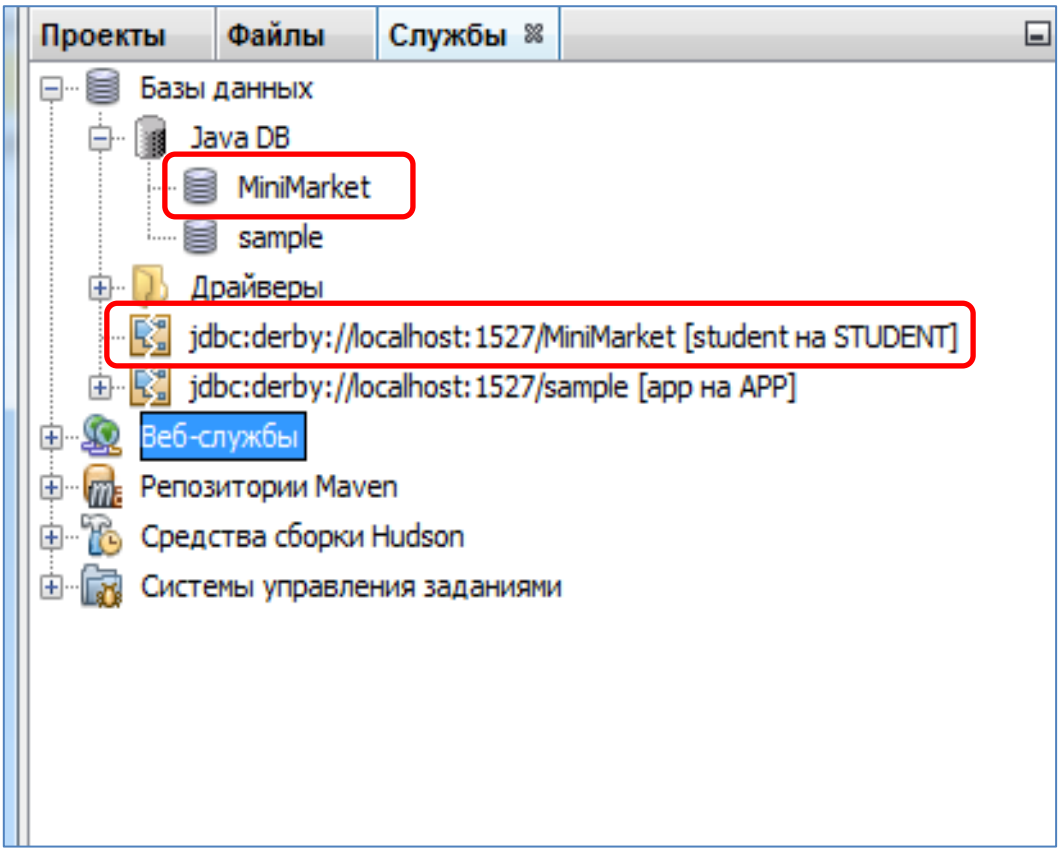


Рис. 3.3. **Значки бази даних *MiniMarket* і підключення у вузлі Java DB**

3. Створіть таблицю ***Товари*** в базі даних ***MiniMarket***. Для цього:

3.1. Клацніть правою клавішею мишки на значок підключення до бази даних ***MiniMarket*** і з контекстового меню виберіть команду **Установить соединение**.

3.2. Розкрийте вузол ***STUDENT*** і клацніть правою клавішею мишки на значок **Таблицы**, потім із контекстового меню виберіть команду **Создать таблицу**. З'явилося вікно **Создать таблицы**.

3.3. Уведіть ім'я таблиці ***Товари*** та, користуючись кнопкою **Добавить столбец**, укажіть властивості нових стовпців, наведені в табл. 3.2.

Таблиця 3.2

Властивості нових стовпців

Стовпці	Властивості	Значення
Код_товару	Имя	Код_товару
	Тип	INTEGER
	Первичный ключ	Так
Товар	Имя	Товар
	Тип	VARCHAR
	Размер	25
Ціна	Имя	Ціна
	Тип	DECIMAL
	Размер	5
	Масштаб	2
Ціна_закупівлі	Имя	Ціна_закупівлі
	Тип	DECIMAL
	Размер	5
	Масштаб	2

На рис. 3.4 подано вікно **Создать таблицы** після задавання властивостей. Клацніть кнопку **ОК**.

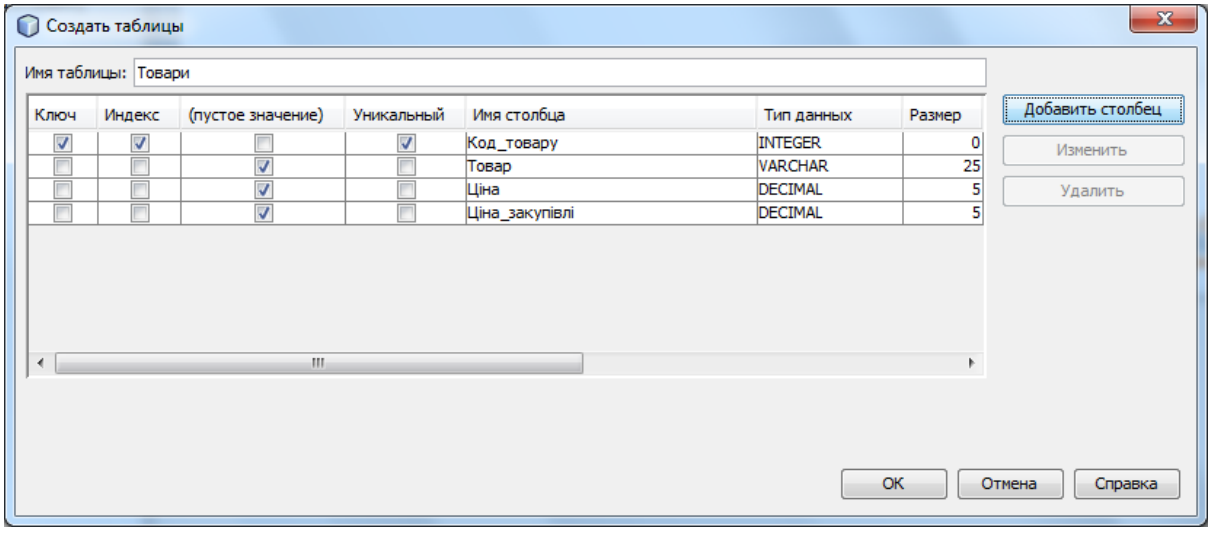


Рис. 3.4. **Вікно Создать таблицы**

4. Заповніть таблицю ***Товари*** даними. Для цього:

4.1. Клацніть правою клавішею мишки на значок таблиці ***Товари*** та з контекстового меню виберіть команду **Просмотр данных**. З'явилося вікно **Команда SQL** із порожньою таблицею в нижній частині.

4.2. Клацніть кнопку **Вставить записи**, що перебуває ліворуч у заголовку нижньої частини вікна **Команда SQL** (рис. 3.5).

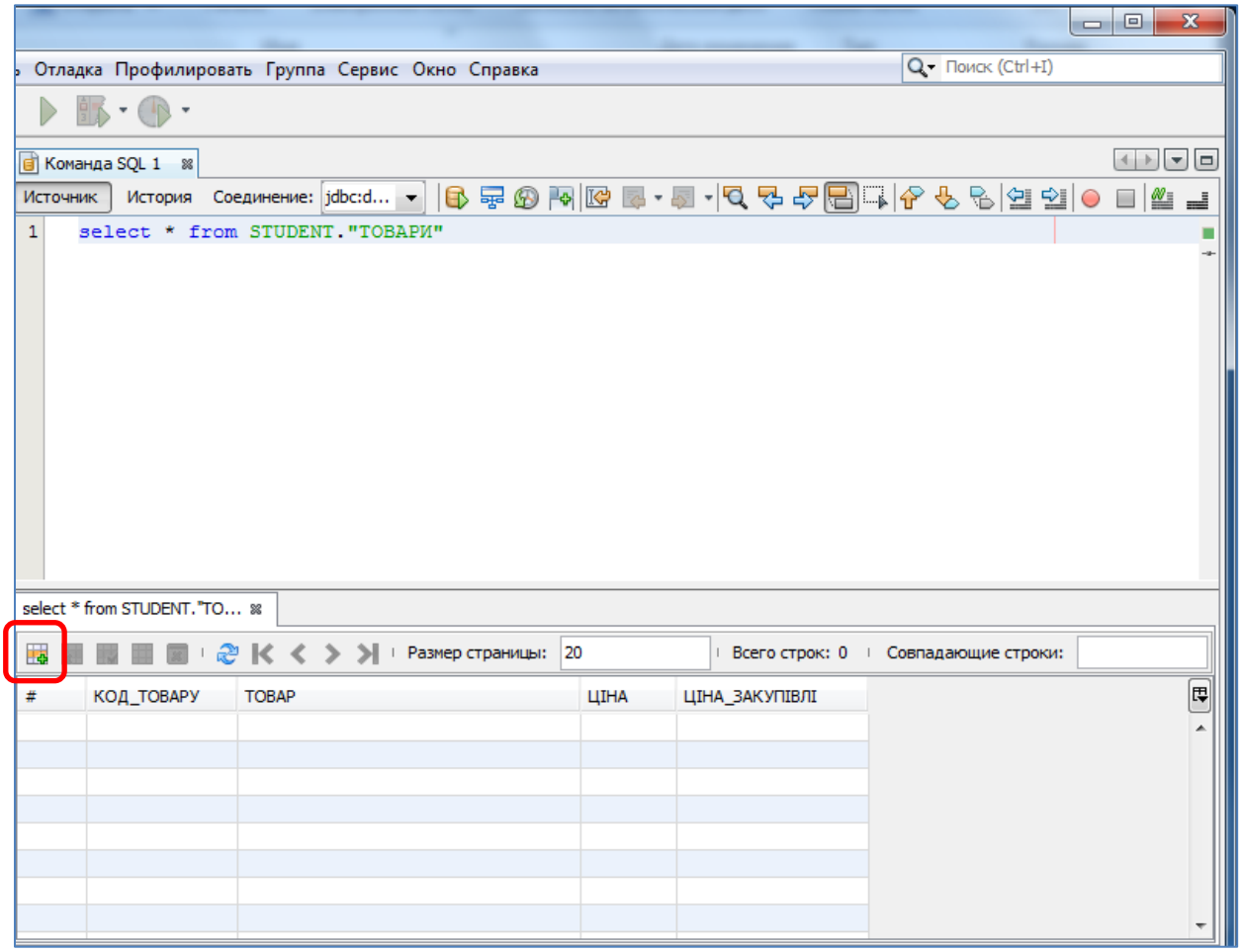


Рис. 3.5. **Кнопка *Вставить записи* у вікні *Команда SQL***

З'явилося вікно **Вставить записи**.

4.4. Уведіть дані про три товари, наведені в табл. 3.3.

Таблиця 3.3

Дані про товари

Код_товару	Товари	Ціни	Ціна_закупівлі
1	Хліб "Український"	10,50	9,30
2	Батон "Молочний"	10,20	9,10
3	Булка з маком	9,80	8,65

Примітка. Для додавання другого і наступних записів використовуйте кнопку **Добавить с…**

Завершіть уведення даних клацанням кнопки **ОК** (рис. 3.6).

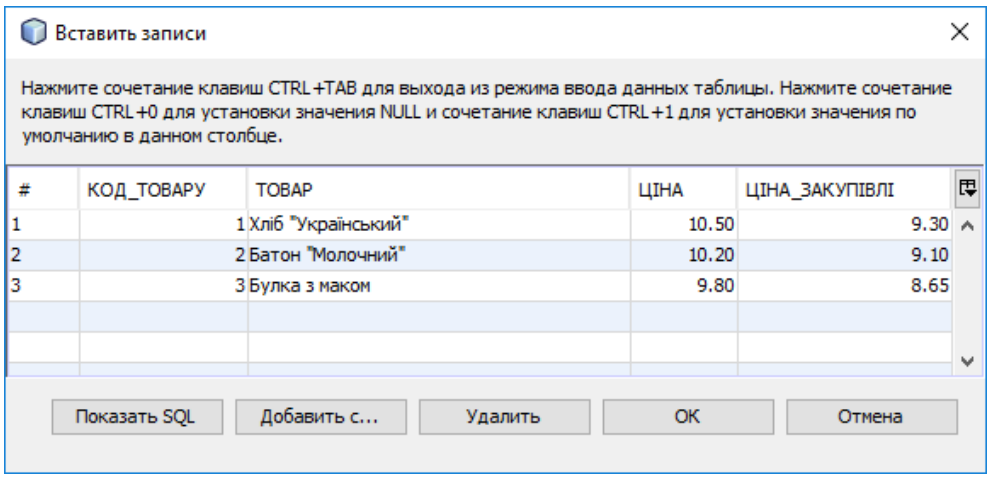


Рис. 3.6. **Вікно *Вставить записи* з даними**

У таблиці, розташованій у нижній частині вікна **Команда SQL**, з'явилися дані.

4.5. Закрийте вікно **Команда SQL**.

У звіті з лабораторної роботи помістіть скриншоти, аналогічні рис. 3.3, 3.4 та 3.6.

2. Створення кнопкової форми застосунку

Завдання

Створіть проєкт ***jdbcMiniMarket***, а в ньому – кнопкову форму ***MiniMarket*** (рис. 3.7).

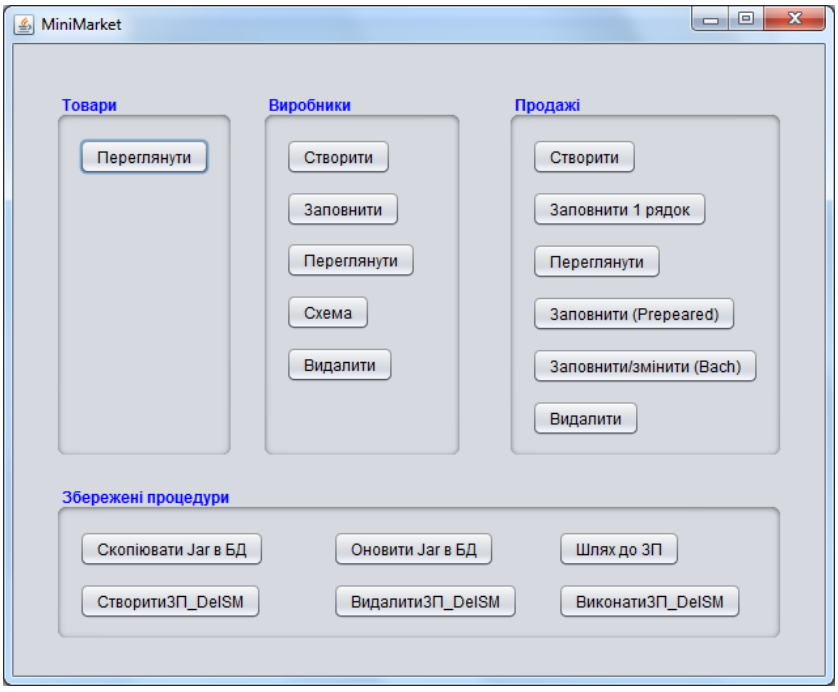


Рис. 3.7. **Кнопкова форма *MiniMarket***

Виконання

1. Створіть проєкт ***jdbcMiniMarket***. Для цього:

1.1. Перебуваючи в середовищі NetBeans, виконайте команду **Файл – Создать проект**. З'явилося вікно майстра **Создать проект**.

1.2. На першому кроці роботи майстра виберіть категорію **Java** і проєкт **Приложение Java**. Потім клацніть кнопку **Далее**.

1.3. На другому кроці роботи майстра введіть ім'я проєкту ***jdbcMiniMarket*** і приберіть прапорець **Создать главный класс** (рис. 3.8). Потім клацніть кнопку **Готово.**

У вікні **Проекты** з'явився значок нового проєкту ***jdbcMiniMarket***.

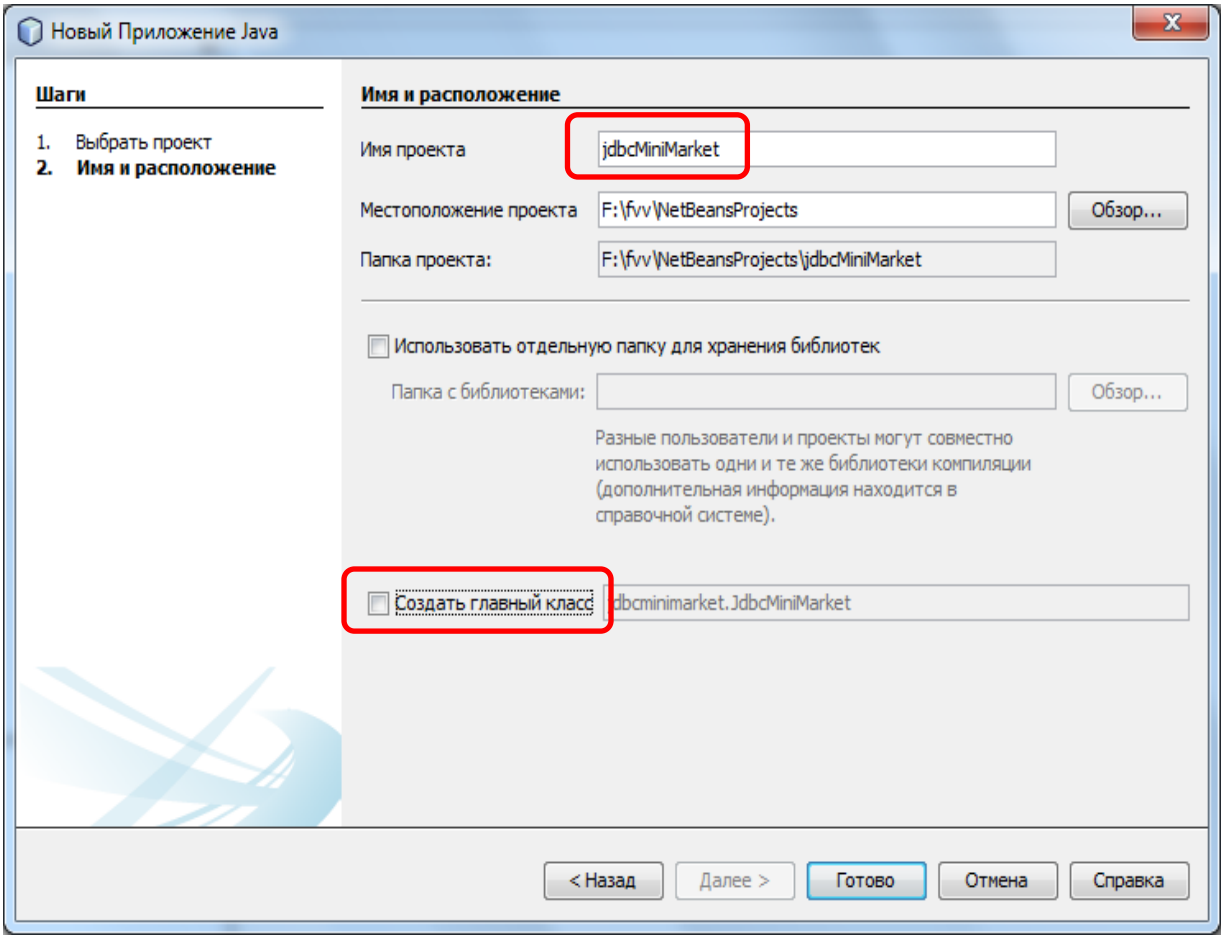


Рис. 3.8. **Другий крок роботи майстра**

2. Додайте форму ***MiniMarket*** до проєкту. Для цього:

2.1. Клацніть правою клавішею мишки на значок проєкту ***jdbcMini-Market*** і з контекстового меню виберіть команду **Новый – Форма JFrame**.

З'явилося вікно **New Форма JFrame** з порожньою таблицею в нижній частині.

2.2. Уведіть ім'я класу ***MiniMarket*** і пакета (прізвище студента латиницею), а потім клацніть кнопку **Готово** (рис. 3.9).

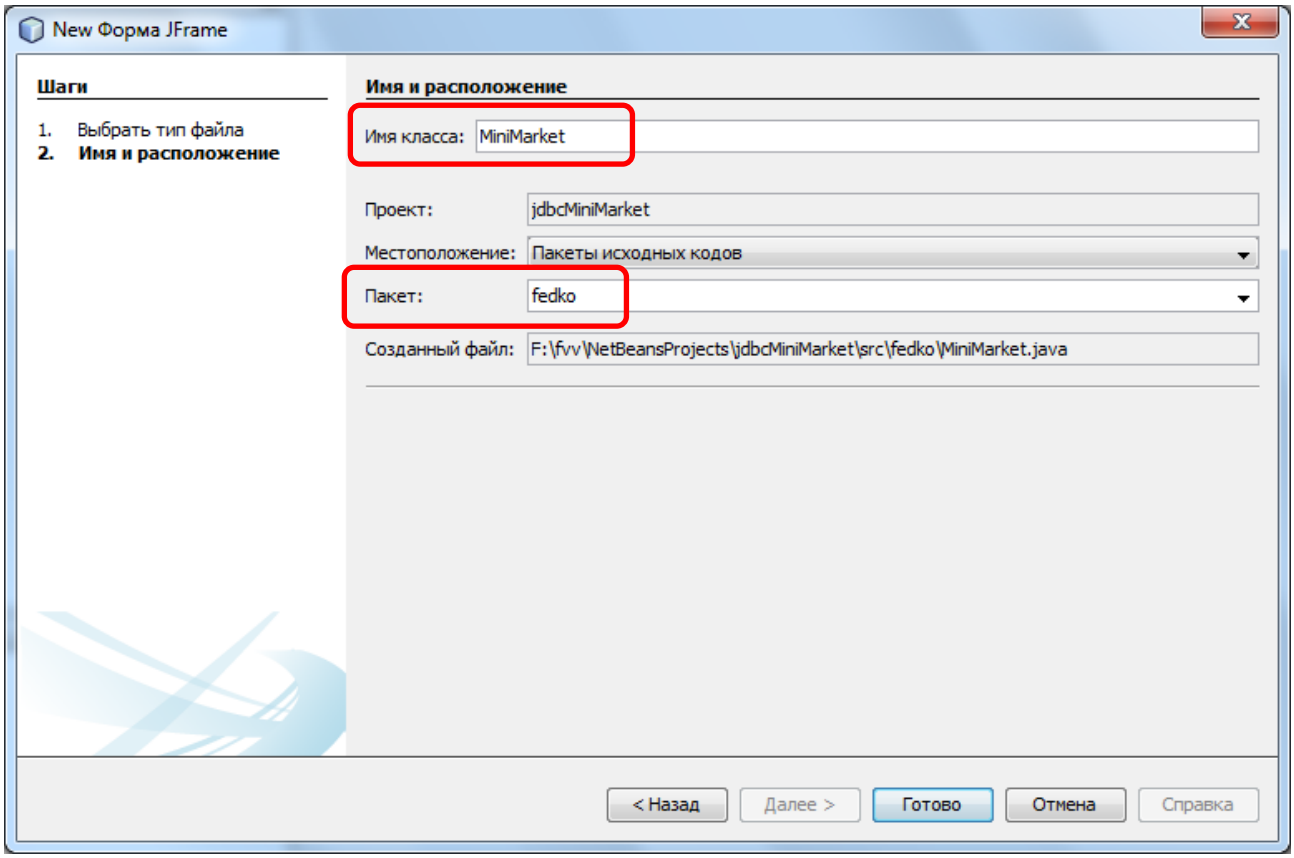


Рис. 3.9. **Задавання імені класу й пакета**

3. Додайте контейнер JPanel ***Товари*** на форму і кнопку ***Переглянути*** в ній. Для цього:

3.1. Перетягніть елемент **Панель** з вікна палітри на форму ***Mini-Market*** і встановіть потрібні розміри.

3.2. Виберіть властивість **border** у вікні властивостей і клацніть кнопку … в ній. З'явилося вікно **JPanel – border**.

3.3. Виберіть елемент **Рамка с надписью** у списку **Доступные границы**, для властивості **Заголовок** уведіть значення ***Товари***, для властивості **Цвет** виберіть у списку значення **Синий** (рис. 3.10). Потім клацніть кнопку **ОК**.

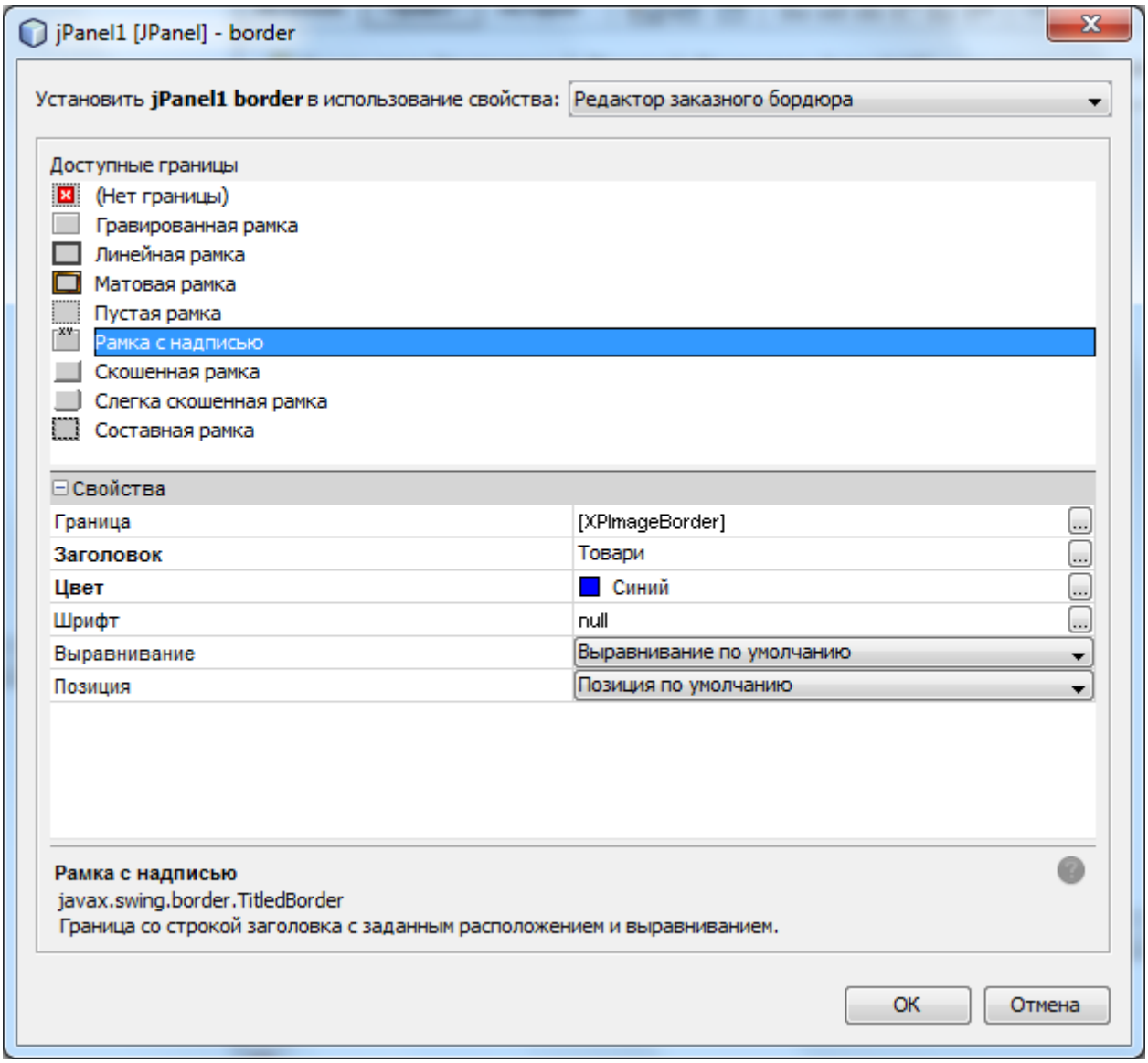


Рис. 3.10. **Установлення властивостей JPanel *Товари***

4. Додайте кнопку ***Переглянути*** в JPanel ***Товари***. Для цього:

4.1. Перетягніть елемент **Кнопка** з вікна палітри у верхню частину панелі ***Товари***.

4.2. Клацніть на кнопці правою клавішею мишки та з контекстового меню виберіть команду **Редактировать текст**, потім, замість попереднього напису, уведіть слово ***Переглянути***.

4.3. Клацніть ще раз на кнопці правою клавішею мишки та з контекстового меню виберіть команду **Изменить имя переменной**, потім, замість попереднього імені, уведіть таке: ***jButtonПереглянути***.

Повторіть п. 3 і 4 для панелей ***Виробники***, ***Продажі*** та ***Збережені процедури***. Для відповідних кнопок установіть імена, наведені в табл. 3.4.

Таблиця 3.4

Імена кнопок

Панелі	Кнопки	Імена
Виробники	Створити	jButtonСтворитиВиробники
	Заповнити	jButtonЗаповнитиВиробники
	Переглянути	jButtonПереглянутиВиробники
	Схема	jButtonСхемаВиробники
	Видалити	jButtonВидалитиВиробники
Продажі	Створити	jButtonСтворитиПродажі
	Заповнити 1 рядок	jButtonЗаповнити1рядокПродажі
	Переглянути	jButtonПереглянутиПродажі
	Заповнити (Prepared)	jButtonЗаповнитиPreparedПродажі
	Заповнити/змінити (Bach)	jButtonЗаповнитиBachПродажі
	Видалити	jButtonВидалитиПродажі
Збережені процедури	Скопіювати Jar у БД	jButtonСкопіюватиJar_у_БД
	Оновити Jar у БД	jButtonОновитиJar_у_БД
	Шлях до ЗП	jButtonШляхДоЗП
	СтворитиЗП_DelSM	jButtonСтворитиЗП_DelSM
	ВидалитиЗП_DelSM	jButtonВидалитиЗП_DelSM
	ВиконатиЗП_DelSM	jButtonВиконатиЗП_DelSM

5. Запустіть програму на виконання і перевірте зовнішній вигляд кнопкової форми ***MiniMarket***.

У звіті з лабораторної роботи помістіть скриншот форми, аналогічний рис. 3.10.

3. Компоненти підключення до бази даних

3.1. Бібліотеки для роботи з базою даних Java DB

Завдання

Додайте бібліотеки **derbyclient.jar** та **derbytools.jar** до проєкту ***jdbcMiniMarket***.

Перша із зазначених бібліотек надає клієнтський JDBC-драйвер і низку класів, що забезпечують JDBC-інтерфейс. Її використовують, якщо з'єднання з базою даних здійснюють за технологією "клієнт – сервер". Для використання бази даних як умонтованої в застосунок ще треба додати бібліотеку derby.jar.

Бібліотека derbytools.jar містить ряд утиліт для роботи з базою даних Java DB (Derby). Її будуть використовувати під час роботи зі збереженими процедурами.

Виконання

1. Клацніть правою клавішею мишки на значок **Библиотеки** у проєкті та з контекстового меню виберіть команду **Добавить JAR файл/папку**. З'явилося однойменне вікно.

2. Перейдіть у вікні **Добавить JAR файл/папку** в папку ***C:\Program Files\Java\jdk1.7.0_15\db\lib*** (чи подібну до неї), виберіть у ній файли ***derbyclient.jar*** та ***derbytools.jar*** і клацніть кнопку **Open**.

У вузлі **Библиотеки** з'явилися значки підключених бібліотек.

Примітка. У разі відсутності на комп'ютері файлів **derbyclient.jar** та **derbytools.jar** їх можна завантажити із сайту, який розміщений за адресою http://db.apache.org/derby/derby_downloads.html.

3.2. Клас підключення до бази даних

Завдання

Створіть клас ***Підключення***, а в ньому – спільні величини ***url***, ***user*** і ***password***. Їх будуть використовувати для встановлення з'єднань із базою даних під час реалізації операцій із базою даних.

Виконання

1. Клацніть правою клавішею мишки на значок проєкту і з контекстового меню виберіть команду **Новый – Класс Java**. З'явилося вікно **New Класс Java**.

2. Уведіть ім'я класу ***Підключення*** і пакета ***операції*** (рис. 3.11). Потім клацніть кнопку **Готово**. У вузлі проєкту з'явився підвузол пакета, а в ньому – клас, і відкрилося вікно коду класу.

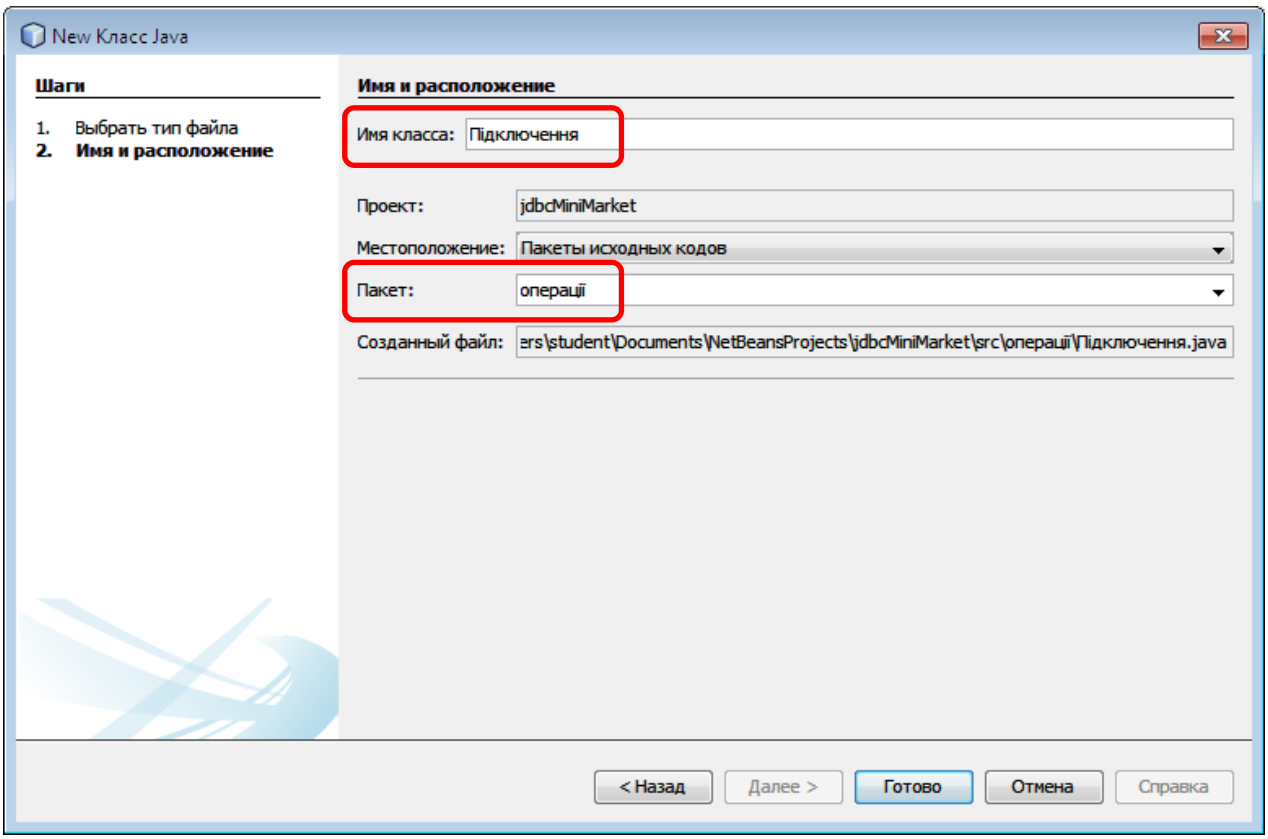


Рис. 3.11. **Задавання імен класу і пакета**

3. Перейдіть у вікно коду класу ***Підключення*** і перед закриваючою фігурною дужкою введіть опис спільних величин. Клас ***Підключення*** набуде такого вигляду:

```
public class Підключення {
     // Спільні величини
    public static String url =
          "jdbc:derby://localhost:1527/MiniMarket;create=true";
    public static String user="student";
    public static String password="student";
}
```

4. Збережіть зміни, зроблені у проєкті.

У звіті з лабораторної роботи помістіть текст класу ***Підключення*** і поясніть кожний елемент рядка **url**.

4. Використання об'єкта ResultSet (таблиця *Товари*)

Завдання

Виведіть на консоль дані, що зберігають у таблиці ***Товари***.

Ідеї виконання

Оскільки таблицю ***Товари*** вже створено і заповнено, перегляньте дані, що зберігають у ній. Для цього використайте об'єкт класу ResultSet, у який прочитайте дані з бази даних. Спочатку створіть клас ***Товари***, а в ньому – метод ***переглянутиТовари***. Потім викличте цей метод з оброблювача події "Клацання кнопки ***Переглянути*** на панелі ***Товари***".

Виконання

1. Клацніть правою клавішею мишки на значок ***операції*** у проєкті та з контекстового меню виберіть команду **Новый – Класс Java**. З'явилося вікно **New Класс Java**.

2. Уведіть ім'я класу ***Товари*** й пакета ***операції*** (див. рис. 3.11). Потім клацніть кнопку **Готово**.

3. Перейдіть у вікно коду класу ***Товари*** й під заголовком пакета операції введіть оператор імпорту пакета **java.sql**:

```
// Імпортуємо пакет класів доступу до баз даних
import java.sql.*;
```

4. Уведіть опис методу ***переглянутиТовари*** перед закриваючою фігурною дужкою класу ***Товари***:

```
public static void переглянутиТовари()
    throws SQLException {
    // Підключаємося до бази даних
    Connection conn = DriverManager.getConnection(
        Підключення.url,Підключення.user,Підключення.password);

    // Створюємо команду виконання SQL-запиту
    Statement st =conn.createStatement();
    String query = "Select * from Товари";

    // Виконуємо SQL-запит
    ResultSet rs= st.executeQuery(query);

    // Виводимо "шапку" таблиці
    System.out.println(String.format("%10s  %-25s %5s %5s",
        "Код_товару" , "Товар","Ціна", "Ціна_закупівлі"));

    // Виводимо дані
    while(rs.next()){
```

```
        int Код_товару = rs.getInt("Код_товару");
        String Товар = rs.getString("Товар");
        float Ціна = rs.getFloat("Ціна");
        float Ціна_закупівлі = rs.getFloat("Ціна_закупівлі");

        System.out.println(String.format("%10s  %-25s %5s %5s",
            Код_товару ,Товар, Ціна, Ціна_закупівлі));
    }

    //Закриваємо підключення
    conn.close();
}
```

5. Додайте код оброблювача події "Клацання кнопки ***Переглянути*** на панелі ***Товари***". Для цього:

5.1. Перейдіть у вікно класу ***MiniMarket*** і встановіть режим **Проект**.

5.2. Клацніть правою клавішею мишки кнопку ***Переглянути*** на панелі ***Товари*** та з контекстового меню виберіть команду **События – Action – actionPerformed**.

5.3. Замість рядка // TODO add your handling code here: уведіть тіло оброблювача ***jButtonПереглянутиActionPerformed***. Він набуде такого вигляду:

```
private void jButtonПереглянутиActionPerformed
    (java.awt.event.ActionEvent evt) {
    try{
          Товари.переглянутиТовари();
    }
    catch (SQLException e) {
          System.out.println("SQL Exception: "+ e.toString());
    }
}
```

Примітка. Погодьтеся з додаванням імпорту класів під час додавання коду тіла оброблювача.

6. Перевірте, чи запущено сервер Java DB, і запустіть на виконання проєкт ***jdbcMiniMarket***. Потім клацніть кнопку ***Переглянути*** на панелі ***Товари***.

У вікні виведення з'явилися дані таблиці ***Товари*** (рис. 3.12). Отже, метод ***переглянутиТовари*** створено правильно.

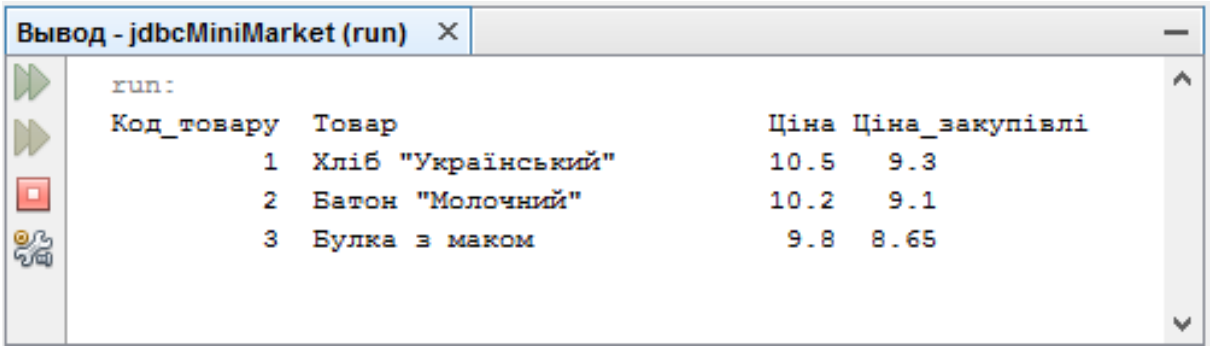


Рис. 3.12. **Дані таблиці *Товари***

7. Збережіть зміни, зроблені у проєкті.

У звіті з лабораторної роботи подайте код методу ***переглянути-Товари*** та поясніть, у чому полягає відмінність від коду методу ***form-Товари_Load***, що розглядали в п. 4.1 лабораторної роботи 1.

5. Операції CRUD для довідкової таблиці (таблиця *Виробники*)

Ідеї виконання

Виконання операцій створення та видалення таблиці, додавання та перегляд даних будуть вивчати на прикладі довідкової таблиці ***Виробники***. Операції оновлення і видалення даних винесено на самостійне виконання.

Для ознайомлення із засобами отримання метаданих виведіть назви полів цієї таблиці та їхні типи даних.

5.1. Створення таблиці програмними засобами

Завдання

Створіть таблицю ***Виробники***.

Виконання

1. Додайте до пакета ***операції*** клас ***Виробники***.

2. Перейдіть у вікно коду класу ***Виробники*** та під заголовком пакета операції введіть оператор імпорту пакета **java.sql**:

```
// Імпортуємо пакет класів доступу до баз даних
import java.sql.*;
```

3. Уведіть опис методу ***створитиВиробники*** перед закриваючою фігурною дужкою класу ***Виробники***:

```
public static void створитиВиробники()
    throws SQLException {
    // Підключаємося до бази даних
    Connection conn = DriverManager.getConnection(
        Підключення.url,Підключення.user,Підключення.password);

    // Створюємо таблицю Виробники
    Statement st =conn.createStatement();
    String query = "CREATE TABLE Виробники " +
        "(Код_виробника int Primary Key, " +
        "Виробник VarChar(20), " +
        "Адреса VarChar(30),  " +
        "Телефон VarChar(15))";
    st.executeUpdate(query);

    //Закриваємо підключення
    conn.close();
}
```

4. Додайте код оброблювача події "Клацання кнопки ***Створити*** на панелі ***Виробники***":

```
private void jButtonСтворитиВиробникиActionPerformed
    (java.awt.event.ActionEvent evt) {
    try{
            Виробники.створитиВиробники();
            System.out.println("Таблицю Виробники створено");
    }
    catch (SQLException e) {
            System.out.println("SQL Exception: "+ e.toString());
    }
}
```

Примітка. Погодьтеся з додаванням імпорту класів під час додавання коду тіла оброблювача.

5. Перевірте функціональність кнопки ***Створити*** на панелі ***Виробники***.

У вікні виведення з'явився текст "Таблицю Виробники створено", а у вікні **Службы** – таблиця ***Виробники*** із заданою схемою (рис. 3.13). Отже, метод ***створитиВиробники*** побудовано правильно.

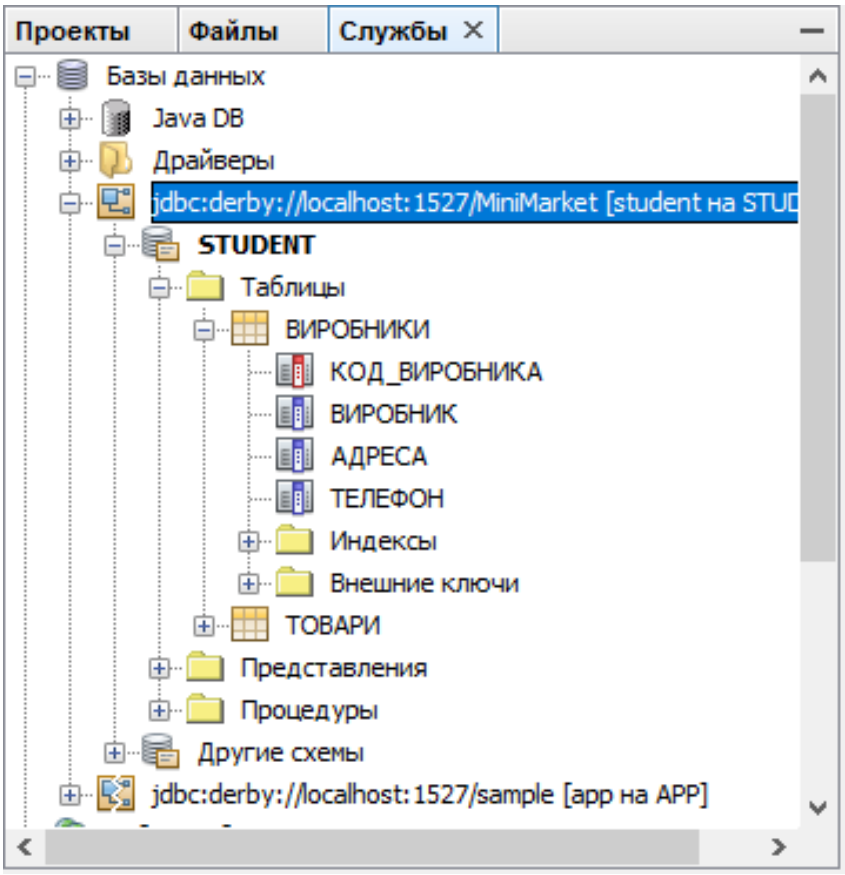


Рис. 3.13. **Таблиця *Виробники* у вікні *Службы***

6. Збережіть зміни, зроблені у проєкті.

5.2. Видалення таблиці програмними засобами

Завдання

Видаліть таблицю ***Виробники***.

Виконання

1. Перейдіть у вікно коду класу ***Виробники*** та введіть опис методу ***видалитиВиробники*** перед закриваючою фігурною дужкою класу ***Виробники***:

```
public static void видалитиВиробники()
    throws SQLException {
    // Підключаємося до бази даних
    Connection conn = DriverManager.getConnection(
        Підключення.url,Підключення.user,Підключення.password);

    // Видаляємо таблицю Виробники
    Statement st =conn.createStatement();
    String query = "DROP TABLE Виробники ";
    st.executeUpdate(query);

    //Закриваємо підключення
    conn.close();
}
```

2. Додайте код оброблювача події "Клацання кнопки ***Видалити*** на панелі ***Виробники***":

```
private void jButtonВидалитиВиробникиActionPerformed
    (java.awt.event.ActionEvent evt) {
    try{
            Виробники.видалитиВиробники();
            System.out.println("Таблицю Виробники видалено");
    }
    catch (SQLException e) {
            System.out.println("SQL Exception: "+ e.toString());
    }
}
```

3. Перевірте функціональність кнопки ***Видалити*** на панелі ***Виробники***.

У вікні виведення з'явився текст "Таблицю Виробники видалено", а з вікна **Службы** зникла таблиця ***Виробники***. Отже, метод ***видалити-Виробники*** побудовано правильно.

4. Клацніть кнопку ***Створити*** на панелі ***Виробники***, щоб у вікні виведення з'явився текст "Таблицю Виробники створено", а у вікні **Службы** – таблиця ***Виробники***.

5. Збережіть зміни, зроблені у проєкті.

5.3. Заповнення таблиці програмними засобами

Завдання

Заповніть даними таблицю ***Виробники***.

Виконання

1. Перейдіть у вікно коду класу ***Виробники*** та введіть опис методу ***заповнитиВиробники*** перед закриваючою фігурною дужкою класу ***Виробники***:

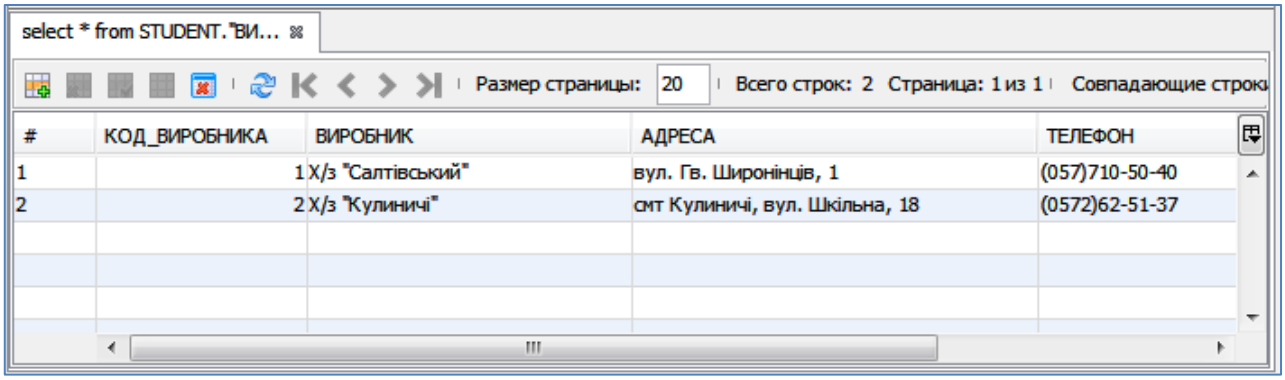
```
public static void заповнитиВиробники()
    throws SQLException {
    // Підключаємося до бази даних
    Connection conn = DriverManager.getConnection(
        Підключення.url,Підключення.user,Підключення.password);

    // Додаємо дані в таблицю Виробники
    Statement st =conn.createStatement();
    String query =
    "INSERT INTO Виробники (КОД_ВИРОБНИКА, ВИРОБНИК, "
        + "АДРЕСА, ТЕЛЕФОН) "
     + "VALUES(1, 'Х/з \"Салтівський\"', "
         + "'вул. Гв. Широнінців, 1', '(057)710-50-40')";
    st.executeUpdate(query);

    query = "INSERT INTO Виробники VALUES " +
        "(2, 'Х/з \"Кулиничі\"', "
        + "'смт Кулиничі, вул. Шкільна, 18', '(0572)62-51-37')";
    st.executeUpdate(query);

    // Закриваємо підключення
    conn.close();
}
```

2. Додайте код оброблювача події "Клацання кнопки ***Заповнити*** на панелі ***Виробники***":

```
private void jButtonЗаповнитиВиробникиActionPerformed
    (java.awt.event.ActionEvent evt) {
    try{
            // Заповнюємо таблицю Виробники
            Виробники.заповнитиВиробники();
            System.out.println("Таблицю Виробники заповнено");
    }
    catch (SQLException e) {
            // Виводимо повідомлення у разі помилки
            System.out.println("SQL Exception: "+ e.toString());
    }
}
```

3. Перевірте функціональність кнопки ***Заповнити*** на панелі ***Виробники***.

У вікні виведення з'явився текст "Таблицю Виробники заповнено". Скориставшись вікном **Службы** можна переглянути дані таблиці ***Виробники*** (рис. 3.14). Отже, метод ***заповнитиВиробники*** побудовано правильно.



#	КОД_ВИРОБНИКА	ВИРОБНИК	АДРЕСА	ТЕЛЕФОН
1	1	Х/з "Салтівський"	вул. Гв. Широнінців, 1	(057)710-50-40
2	2	Х/з "Кулиничі"	смт Кулиничі, вул. Шкільна, 18	(0572)62-51-37

Рис. 3.14. **Дані таблиці *Виробники* у вікні *Команда SQL***

4. Збережіть зміни, зроблені у проєкті.

5.4. Перегляд даних таблиці

Завдання

Виведіть на консоль дані, що зберігають у таблиці ***Виробники***.

Виконання

Подібно до описаного в п. 4 "Використання об'єкта ResultSet" створіть у класі ***Виробники*** метод ***переглянутиВиробники*** та викличте його з оброблювача події "Клацання кнопки ***Переглянути*** на панелі ***Виробники***". Визначений результат подано на рис. 3.15.

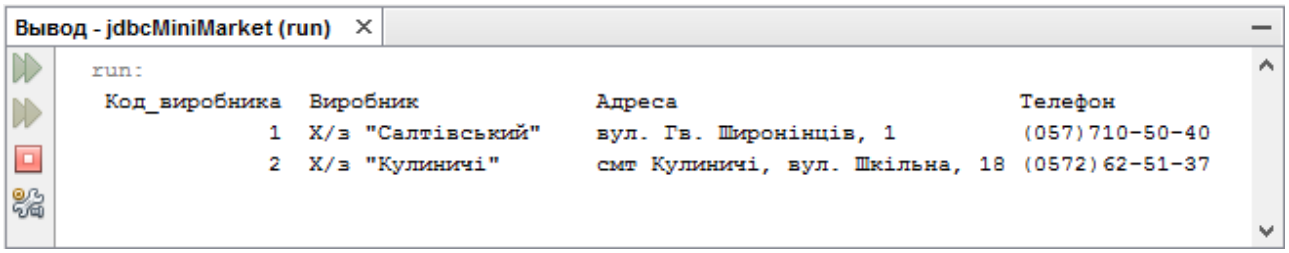


Рис. 3.15. **Дані таблиці *Виробники* у вікні виведення**

5.5. Отримання метаданих

Завдання

Виведіть на консоль імена стовпців таблиці ***Виробники*** та їхні типи даних.

Виконання

1. Перейдіть у вікно коду класу ***Виробники*** та введіть опис методу ***схемаВиробники*** перед закриваючою фігурною дужкою класу ***Виробники***:

```
public static void схемаВиробники()
    throws SQLException {
    //Підключаємося до бази даних
    Connection conn = DriverManager.getConnection(
        Підключення.url,Підключення.user,Підключення.password);

    //Вибираємо дані з таблиці
    Statement st = conn.createStatement();
    String query = "Select * from Виробники";
    ResultSet rs= st.executeQuery(query);

    // Отримуємо метадані запиту
    ResultSetMetaData rsmd=rs.getMetaData();

    //Знаходимо кількість стовпців
    int colCount = rsmd.getColumnCount();

    // Виводимо схему таблиці
    System.out.println("Схема таблиці Виробники");
    for (int i=1; i<= colCount; i++) {
        String colName = rsmd.getColumnName(i);     // Ім'я стовпця
        String colType = rsmd.getColumnTypeName(i);//Тип данних стовпця

        // Виводимо характеристики стовпця
      System.out.println(String.format("%-20s %-15s",colName,colType));
        }
           // Закриваємо підключення
    conn.close();
}
```

2. Додайте код оброблювача події "Клацання кнопки ***Схема*** на панелі ***Виробники***", у якому викликають метод ***схемаВиробники***.

3. Перевірте функціональність кнопки ***Схема*** на панелі ***Виробники***.

У вікні виведення з'явилися дані про схему таблиці ***Виробники*** (рис. 3.16). Отже, метод ***схемаВиробники*** створено правильно.

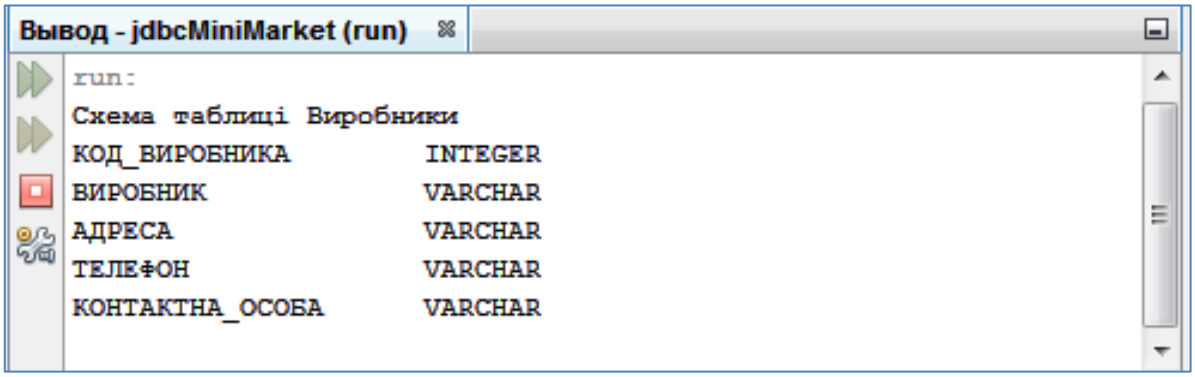


Рис. 3.16. **Дані про схему таблиці *Виробники***

4. Збережіть зміни, зроблені у проєкті.

У звіті з лабораторної роботи подайте код методів, які використовували для реалізації CRUD-операцій і поясніть, у чому полягає їхня відмінність від коду аналогічних методів, що розглядали в п. 4 лабораторної роботи 1.

6. Використання звичайних, підготовлених і пакетних запитів (таблиця *Продажі*)

6.1. Створення дочірньої таблиці

Завдання

Створіть таблицю ***Продажі***.

Виконання

1. Додайте до пакета ***операції*** клас ***Продажі***.

2. Перейдіть у вікно коду класу ***Продажі*** та під заголовком пакета операції введіть оператор імпорту пакета **java.sql**:

```
// Імпортуємо пакет класів доступу до баз даних
import java.sql.*;
```

3. Уведіть опис методу ***створитиПродажі*** перед закриваючою фігурною дужкою класу ***Продажі***:

```
public static void створитиПродажі()
    throws SQLException {
    // Підключаємося до бази даних
    Connection conn = DriverManager.getConnection(
        Підключення.url,Підключення.user,Підключення.password);

    // Створюємо таблицю Продажі
    Statement st =conn.createStatement();
    String query = "CREATE TABLE Продажі " +
        "(Код_продажу int Primary Key, " +
        "Дата date, " +
       "Код_виробника int REFERENCES Виробники(Код_виробника)," +
        "Код_товару int REFERENCES Товари(Код_товару), " +
        "Кількість smallint)";
    st.executeUpdate(query);
    //Закриваємо підключення
    conn.close();
}
```

4. Додайте код оброблювача події "Клацання кнопки Створити на панелі Продажі", у якому викликають метод ***створитиПродажі***.

5. Перевірте функціональність кнопки ***Створити*** на панелі ***Продажі***.

У вікні виведення з'явився текст "Таблицю Продажі створено", а у вікні **Службы** – таблиця ***Продажі*** із заданою схемою. Отже, метод ***створитиПродажі*** побудовано правильно.

6. Збережіть зміни, зроблені у проєкті.

6.2. Видалення дочірньої таблиці

Завдання

Видаліть таблицю ***Продажі.***

Виконання

Подібно до описаного в завд. 2 п. 5 "Операції CRUD для довідкової таблиці" створіть у класі ***Продажі*** метод ***видалитиПродажі*** та викличте його з оброблювача події "Клацання кнопки ***Видалити*** на панелі ***Продажі***". Потім знову створіть таблицю ***Продажі,*** клацнувши кнопку ***Створити*** на панелі ***Продажі***.

6.3. Уведення рядка даних

Завдання

Заповніть даними один рядок таблиці ***Продажі.*** Для цього скористайтеся виконанням звичайного запиту.

Виконання

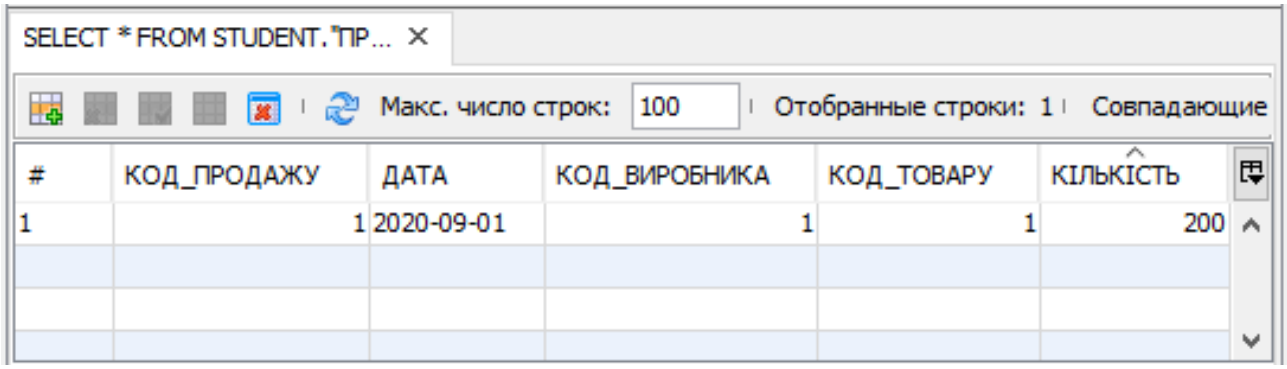
1. Перейдіть у вікно коду класу ***Продажі*** та введіть опис методу ***заповнити1Продажі*** перед закриваючою фігурною дужкою класу ***Продажі***:

```
public static void заповнити1Продажі()
    throws SQLException {
    // Підключаємося до бази даних
    Connection conn = DriverManager.getConnection(
        Підключення.url,Підключення.user,Підключення.password);
    // Додаємо дані в таблицю Продажі (1 рядок),
    // виконуючи звичайний запит
    Statement st =conn.createStatement();
    String query =
       "INSERT INTO Продажі (Код_продажу, Дата, Код_виробника, "
        + "Код_товару, Кількість) "
        + "VALUES(1, '01.09.2020', 1, 1, 200)";
    st.executeUpdate(query);
    //Закриваємо підключення
    conn.close();
}
```

2. Додайте код оброблювача події "Клацання кнопки ***Заповнити 1 рядок*** на панелі ***Продажі***", у якому викликають метод ***заповнити1-Продажі***.

3. Перевірте функціональність кнопки ***Заповнити 1 рядок*** на панелі ***Продажі***.

У вікні виведення з'явився текст "В таблицю Продажі додано 1 рядок". Скориставшись вікном **Службы** можна переглянути дані таблиці ***Продажі*** (рис. 3.17). Отже, метод ***заповнитиВиробники*** побудовано правильно.



#	КОД_ПРОДАЖУ	ДАТА	КОД_ВИРОБНИКА	КОД_ТОВАРУ	КІЛЬКІСТЬ
1	1	2020-09-01	1	1	200

Рис. 3.17. **Перший рядок таблиці *Продажі* у вікні *Команда SQL***

4. Збережіть зміни, зроблені у проєкті.

6.4. Об'єднане виведення даних кількох таблиць

Завдання

Виведіть на консоль дані, що зберігають у таблиці ***Продажі***, замінивши коди товару і виробника даними з відповідних таблиць, а також визначте вартість проданого товару.

Виконання

Скористайтеся даними із трьох таблиць ***Продажі***, ***Товари*** та ***Виробники***. Тому в запиті об'єднайте ці таблиці за значеннями зовнішніх ключів таблиці ***Продажі***. Для визначення вартості проданого товару додайте до запиту обчислюване поле. Запит набуде такого вигляду:

```
String query =
"SELECT Продажі.Код_продажу, Продажі.Дата,Виробники.Виробник, "
    + "Товари.Товар, Товари.Ціна, Продажі.Кількість, "
    + "Ціна*Кількість AS Вартість "
    + "FROM Товари INNER JOIN (Виробники INNER JOIN Продажі "
    + "ON Виробники.Код_виробника = Продажі.Код_виробника) "
    + "ON Товари.Код_товару = Продажі.Код_товару "
    + "Order By Код_продажу";
```

Решта дій подібна до описаних у п. 4 "Використання об'єкта ResultSet". Тобто створіть у класі ***Продажі*** метод ***переглянутиПродажі*** та викличте його з оброблювача події "Клацання кнопки ***Переглянути*** на панелі ***Продажі***". Визначений результат подано на рис. 3.18.

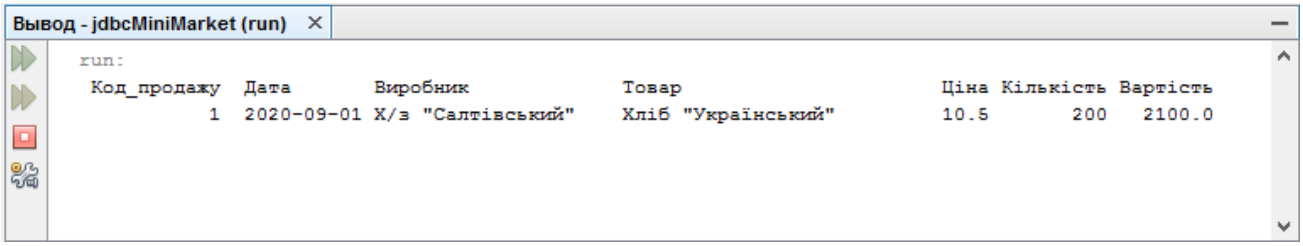


Рис. 3.18. **Дані про продаж товару (перший рядок)**

6.5. Використання підготовленого запиту

Завдання

Заповніть даними ще чотири рядки таблиці ***Продажі***. Для цього скористайтеся виконанням підготовленого (компільованого) запиту.

Ідеї виконання

Використання підготовленого запиту доцільне, якщо той самий запит потрібно виконувати кілька разів. Спочатку створіть метод ***додати1-Prepared***, який додає один рядок до таблиці ***Продажі*** за допомогою підготовленого запиту. Метод інкапсулює надання значень стовпцям цього рядка, щоб не записувати їх для кожного рядка.

Потім побудуйте метод ***заповнитиPreparedПродажі***, який компілює запит і передає його методу ***додати1Prepared***. Останній викликають сім разів (для кожного рядка, що додають).

Виконання

1. Перейдіть у вікно коду класу ***Продажі*** та введіть опис методу ***додати1Prepared*** перед закриваючою фігурною дужкою класу ***Продажі***:

```
static void додати1Prepared(PreparedStatement ps,
    int Код_продажу,
    String Дата,
    int Код_виробника,
    int Код_товару,
    int Кількість)throws SQLException {

    // Задаємо значення полів
    ps.setInt(1, Код_продажу);
    ps.setString(2, Дата);
    ps.setInt(3, Код_виробника);
    ps.setInt(4, Код_товару);
    ps.setInt(5, Кількість);

    // Виконуємо підготовлений запит
    ps.executeUpdate();
}
```

2. Уведіть опис методу ***заповнитиPreparedПродажі*** перед закриваючою фігурною дужкою класу ***Продажі***:

```
public static void заповнитиPreparedПродажі()
    throws SQLException {
    // Підключаємося до бази даних
    Connection conn = DriverManager.getConnection(
        Підключення.url,Підключення.user,Підключення.password);

       // Додаємо дані в таблицю
    Statement st =conn.createStatement();
    String query = "INSERT INTO Продажі (Код_продажу, Дата, "
        + " Код_виробника, Код_товару, Кількість) "
        + "VALUES(?, ?, ?, ?, ?)";
```

```
    // Готуємо (компілюємо) запит
    PreparedStatement ps = conn.prepareStatement(query);

    // Додаємо рядки
    додати1Prepared(ps, 2, "2020-09-01", 1, 2, 250);
    додати1Prepared(ps, 3, "2020-09-01", 2, 1, 150);
    додати1Prepared(ps, 4, "2020-09-01", 2, 3, 180);
    додати1Prepared(ps, 5, "2020-09-02", 1, 1, 220);

    // Закриваємо підключення
    conn.close();
}
```

3. Додайте код оброблювача події "Клацання кнопки ***Заповнити (Prepared)*** на панелі ***Продажі***", у якому викликають метод ***заповнитиPreparedПродажі***:

```
private void jButtonЗаповнитиPreparedПродажіActionPerformed
    (java.awt.event.ActionEvent evt) {
    try {
        Продажі.заповнитиPreparedПродажі();
        System.out.println("У таблицю Продажі додано 4 рядки");
    } catch (SQLException e) {
        System.out.println("SQL Exception: "+ e.toString());
    }
}
```

4. Перевірте функціональність кнопки ***Заповнити (Prepeared)*** на панелі ***Продажі***. Потім клацніть кнопку ***Переглянути*** на цій самій панелі.

У вікні виведення з'явився текст "У таблицю Продажі додано 4 рядки", а потім 8 рядків із даними про продажі (рис. 3.19). Отже, методи ***додати1Prepared*** та ***заповнитиPreparedПродажі*** побудовано правильно.

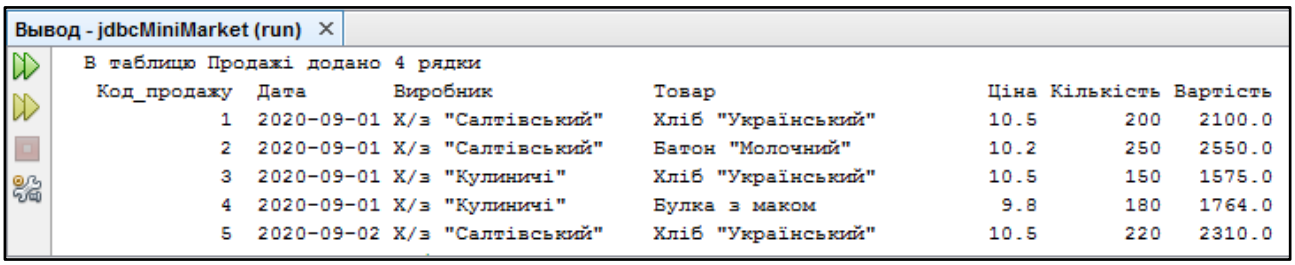


Рис. 3.19. **П'ять рядків із даними про продаж товарів**

5. Збережіть зміни, зроблені у проєкті.

6.6. Пакетне виконання запитів

Завдання

Додайте ще три рядки даних і змініть дані в одному рядку, що вже зберігають у таблиці ***Продажі***. Для цього скористайтеся пакетним виконанням запитів, тобто виконайте кілька запитів за одне звертання до сервера баз даних.

Ідеї виконання

Для пакетного виконання запитів спочатку додайте їх до пакета за допомогою методу **addBatch**, а потім одночасно виконайте всі запити пакета, викликавши метод **executeBatch**.

Виконання

1. Перейдіть у вікно коду класу ***Продажі*** та введіть опис методу ***заповнитиBachПродажі*** перед закриваючою фігурною дужкою класу ***Продажі***:

```
public static void заповнитиBachПродажі()
    throws SQLException {
}
```

2. Додайте код оброблювача події "Клацання кнопки Заповнити/змінити (Bach) на панелі Продажі", у якому викликають метод ***заповнитиBachПродажі***.

3. Перевірте функціональність кнопки ***Заповнити/змінити (Bach)*** на панелі ***Продажі***. Потім клацніть кнопку ***Переглянути*** на цій самій панелі.

У вікні виведення з'явився текст "У таблицю Продажі додано 3 рядки і змінено 1", а потім 8 рядків із даними про продажі, у першому з яких змінено кількість проданого товару (рис. 3.20) Отже, метод ***заповнитиBachПродажі*** побудовано правильно.

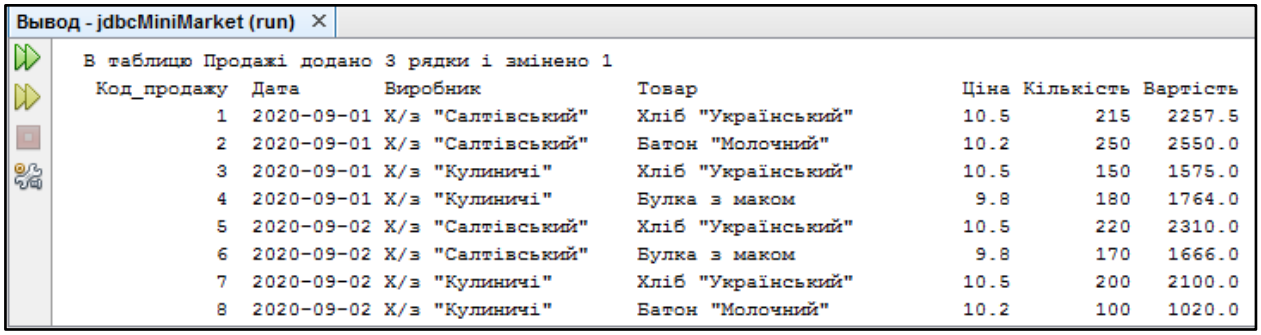


Рис. 3.20. **Вісім рядків із даними про продаж товарів**

4. Збережіть зміни, зроблені у проєкті.

У звіті з лабораторної роботи порівняйте засоби використання звичайних, підготовлених і пакетних запитів та опишіть сфери найкращого застосування кожного з них.

7. Робота зі збереженими процедурами

Ідеї та порядок виконання

Збережені процедури у Java DB подають як методи мовою Java, що зберігають у jar-файлах. У базу даних записують тільки заголовок збереженої процедури та місце розташування класу, у якому зберігають метод. Тому створення і використання збережених процедур у Java DB складається з таких кроків:

1. Створення Java-методу у класі та отримання jar-файлу.
2. Копіювання jar-файлу в базу даних і задавання шляху до нього.
3. Створення збереженої процедури для викликання Java-методу.
4. Виклик збереженої процедури.

Використання збережених процедур будуть вивчати на прикладі такого завдання: потрібно видалити дані про заданого виробника і продаж його товарів. Оскільки під час створення бази даних ***MiniMarket*** не встановлено обмеження каскадного видалення між таблицями ***Виробники*** та ***Продажі***, у збереженій процедурі спочатку видаліть дані з дочірньої таблиці, а потім – із батьківської. Також дізнайтеся, скільки рядків було видалено з таблиці ***Продажі***. Цю величину подайте як вихідний параметр збереженої процедури.

7.1. Створення методу реалізації завдання

Завдання

Створіть Java-метод ***del_SM*** у класі ***StoredProc*** та отримайте jar-файл.

Виконання

1. Створіть проєкт ***SP***. Для цього:

1.1. Перебуваючи в середовищі NetBeans, виконайте команду **Файл – Создать проект**. З'явилося вікно майстра **Создать проект**.

1.2. На першому кроці роботи майстра виберіть категорію **Java** і проєкт **Библиотека классов Java**. Потім клацніть кнопку **Далее**.

1.3. На другому кроці роботи майстра введіть ім'я проєкту ***SP***. Потім клацніть кнопку **Готово.**

У вікні **Проекты** з'явився значок нового проєкту ***SP***.

2. Додайте клас ***StoredProc*** до проєкту ***SP***. Для цього:

2.1. Клацніть правою клавішею мишки на значок проєкту ***SP*** і з контекстового меню виберіть команду **Новый – Класс Java**. З'явилося вікно **New Класс Java**.

2.2. Уведіть ім'я класу ***StoredProc*** і пакета ***packStoredProc***. Потім клацніть кнопку **Готово**.

3. Перейдіть у вікно коду класу ***StoredProc*** і під заголовком пакета операції введіть оператор імпорту пакета **java.sql**:

```
// Імпортуємо пакет класів доступу до баз даних
import java.sql.*;
```

4. Уведіть опис методу ***del_SM*** перед закриваючою фігурною дужкою класу ***StoredProc***:

```
public static void del_SM( int manufacturerID, int[] countRows)
    throws SQLException{
    // !!!
    // int[] countRows - вихідний параметр обов'язково масив

    // Підключаємося до бази даних (укладене підключення)
    Connection conn = DriverManager.getConnection(
        "jdbc:default:connection");

    // Видаляємо дані з таблиці Продажі
    // (дочірня таблиця)
    String query = "Delete From Продажі Where Код_виробника = ?";
    // Готуємо (компілюємо) запит
    PreparedStatement ps = conn.prepareStatement(query);
    ps.setInt(1, manufacturerID);
    // Виконуємо запит
    // Скалярний результат записуємо в перший елемент
    // вихідного параметра
    countRows[0] = ps.executeUpdate();

    // Видаляємо дані з таблиці Виробники
    // (батьківська таблиця)
    query = "Delete From Виробники Where Код_виробника = ?";
    ps = conn.prepareStatement(query);
    ps.setInt(1, manufacturerID);
    ps.executeUpdate();
}
```

5. Скомпілюйте і запакуйте метод ***del_SM*** у jar-файл. Для цього виконайте команду **Выполнить – Очистить и собрать проект (SP)**.

У папці ***dist*** проєкту ***SP*** з'явився файл ***SP.jar***.

7.2. Перенесення методу реалізації завдання в базу даних

Завдання

Скопіюйте jar-файл ***SP.jar*** у базу даних ***MiniMarket*** і задайте шлях до нього.

Виконання

1. Додайте до пакета ***операції*** клас ***ЗбереженіПроцедури***.

2. Перейдіть у вікно коду класу ***ЗбереженіПроцедури*** та під заголовком пакета операції введіть оператор імпорту пакета **java.sql**:

```
// Імпортуємо пакет класів доступу до баз даних
import java.sql.*;
```

3. Щоб скопіювати jar-файл ***SP.jar*** у базу даних ***MiniMarket***, виконайте таке:

3.1. Уведіть опис методу ***скопіюватиJar_у_БД*** перед закриваючою фігурною дужкою класу ***ЗбереженіПроцедури***:

```
public static void скопіюватиJar_у_БД()
    throws SQLException {

    //Підключаємося до бази даних
    Connection conn = DriverManager.getConnection(
        Підключення.url,Підключення.user,Підключення.password);

    // Копіюємо jar-файл із збереженими процедурами у БД
    Statement st =conn.createStatement();
    String query = "CALL sqlj.install_jar " +
       "('/Users/student/Documents/NetBeansProjects/SP/dist/SP.jar',"
       + "'student.StoredProc', 0)" ;
    st.executeUpdate(query);

    //Закриваємо підключення
    conn.close();
}
```

3.2. Додайте код оброблювача події "Клацання кнопки Скопіювати Jar у БД на панелі Збережені процедури", у якому викликають метод ***скопіюватиJar_в_БД***, і перевірте функціональність цієї кнопки.

4. Якщо змінено метод ***del_SM*** у jar-файлі ***SP.jar***, замініть файл у базі даних. Для цього:

4.1. Уведіть опис методу ***оновитиJar_у_БД*** перед закриваючою фігурною дужкою класу ***ЗбереженіПроцедури***:

```
public static void оновитиJar_у_БД()
    throws SQLException {

    //Підключаємося до бази даних
    Connection conn = DriverManager.getConnection(
        Підключення.url,Підключення.user,Підключення.password);

    // Замінюємо збережені процедури на нові(jar-файл)
    Statement st =conn.createStatement();
    String query = "CALL sqlj.replace_jar "
       + "('/Users/student/Documents/NetBeansProjects/SP/dist/SP.jar',"
       + "'student.StoredProc')";
    st.executeUpdate(query);

    //Закриваємо підключення
    conn.close();
}
```

4.2. Додайте код оброблювача події "Клацання кнопки Оновити Jar у БД на панелі Збережені процедури", у якому викликають метод ***оновитиJar_у_БД***, і перевірте функціональність цієї кнопки.

5. Щоб задати шлях до jar-файлу ***SP.jar*** у базі даних ***MiniMarket***, виконайте таке:

5.1. Уведіть опис методу ***шляхДоЗП*** перед закриваючою фігурною дужкою класу ***ЗбереженіПроцедури***:

```
public static void шляхДоЗП()
    throws SQLException {

    //Підключаємося до бази даних
    Connection conn = DriverManager.getConnection(
        Підключення.url,Підключення.user,Підключення.password);

    // Установлюємо шлях до збережених процедур
    Statement st =conn.createStatement();
    String query = "CALL SYSCS_UTIL.SYSCS_SET_DATABASE_PROPERTY " +
        "('derby.database.classpath', 'student.StoredProc')";
    st.executeUpdate(query);

    //Закриваємо підключення
    conn.close();
}
```

5.2. Додайте код оброблювача події "Клацання кнопки ***Шлях до ЗП*** на панелі ***Збережені процедури***", у якому викликають метод ***шляхДоЗП***, і перевірте функціональність цієї кнопки.

6. Збережіть зміни, зроблені у проєкті.

7.3. Створення збереженої процедури

Завдання

Створіть збережену процедуру ***del_SM*** для викликання однойменного Java-методу, а в разі необхідності видаліть її.

Виконання

1. Щоб створити збережену процедуру ***del_SM***, виконайте таке:

1.1. Перейдіть у вікно коду класу ***ЗбереженіПроцедури*** та введіть опис методу ***створитиЗП_DelSM*** перед закриваючою фігурною дужкою класу ***ЗбереженіПроцедури***:

```
public static void створитиЗП_DelSM()
    throws SQLException {
    //Підключаємося до бази даних
    Connection conn = DriverManager.getConnection(
        Підключення.url,Підключення.user,Підключення.password);
    // Додаємо збережену процедуру в БД
    Statement st =conn.createStatement();
    String query =
        "CREATE PROCEDURE Del_SM"
        + "(IN manufacturerID int, OUT countRows int ) "
        + "parameter style java "
        + "modifies sql data "
        + "language java "
        + "external name 'packStoredProc.StoredProc.del_SM'";
    st.executeUpdate(query);
    //Закриваємо підключення
    conn.close();
}
```

1.2. Додайте код оброблювача події "Клацання кнопки ***СтворитиЗП_DelSM*** на панелі ***Збережені процедури***", у якому викликають метод ***створитиЗП_DelSM***, і перевірте функціональність цієї кнопки.

У вікні виведення з'явився текст "Створено збережену процедуру DelSM", а у вікні **Службы** – збережена процедура ***DEL_SM*** (рис. 3.21).

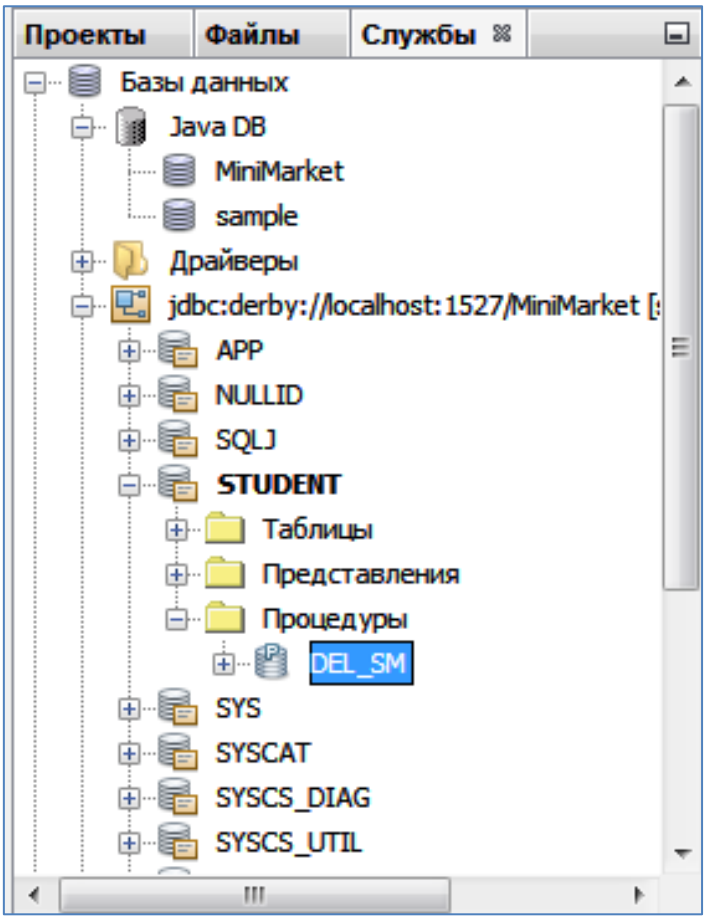


Рис. 3.21. **Збережена процедура *DEL_SM* у вікні Служби**

2. Щоб надати можливість видалити ***del_SM***, виконайте таке:

2.1. Уведіть опис методу ***видалитиЗП_DelSM*** перед закриваючою фігурною дужкою класу ***ЗбереженіПроцедури***:

```
public static void видалитиЗП_DelSM()
    throws SQLException {
    //Підключаємося до бази даних
    Connection conn = DriverManager.getConnection(
        Підключення.url,Підключення.user,Підключення.password);

    // Видаляємо збережену процедуру із БД
    Statement st =conn.createStatement();
    String query = "DROP PROCEDURE Del_SM";
    st.executeUpdate(query);

    //Закриваємо підключення
    conn.close();
}
```

2.2. Додайте код оброблювача події "Клацання кнопки ***ВидалитиЗП_DelSM*** на панелі ***Збережені процедури***", у якому викликають метод ***видалитиЗП_DelSM***, і перевірте функціональність цієї кнопки. У вікні виведення з'явився текст "Видалено збережену процедуру Del_SM", а у вікні **Службы** зникне збережена процедура ***DEL_SM***.

2.3. Повторно клацніть кнопку ***СтворитиЗП_DelSM***, щоб ще раз створити збережену процедуру.

7.4. Використання збереженої процедури

Завдання

Викличте збережену процедуру.

Виконання

1. Перейдіть у вікно коду класу ***ЗбереженіПроцедури*** та введіть опис методу ***виконатиЗП_DelSM*** перед закриваючою фігурною дужкою класу ***ЗбереженіПроцедури***:

```
public static void виконатиЗП_DelSM()
    throws SQLException {
    //Підключаємося до бази даних
    Connection conn = DriverManager.getConnection(
    Підключення.url,Підключення.user,Підключення.password);

    // Готуємося до виклику збереженої процедури
    String query = "{call Del_SM(?, ?)}";
    CallableStatement cs = conn.prepareCall(query);
    cs.setInt(1,2);
    // Реєструємо вихідний параметр
    cs.registerOutParameter(2,Types.INTEGER);

    // Виконуємо збережену процедуру
    cs.execute();

    // Отримуємо значення вихідного параметра
    int кількістьВидаленихРядків = cs.getInt(2);
    System.out.println("З таблиці Продажі видалено рядків: "
        + кількістьВидаленихРядків);

    //Закриваємо підключення
    conn.close();
}
```

2. Додайте код оброблювача події "Клацання кнопки Виконати-ЗП_DelSM на панелі Збережені процедури", у якому викликають метод ***виконатиЗП_DelSM***, і перевірте функціональність цієї кнопки.

У вікні виведення з'явився текст "З таблиці Продажі видалено рядків: 3. Виконано збережену процедуру Del_SM".

3. Перевірте дані, що зберігають у таблицях ***Виробники*** та ***Продажі***, клацнувши кнопки ***Переглянути*** на відповідних панелях. Упевніться в тому, що із цих таблиць видалено дані про другого виробника і продаж його товарів.

4. Щоб відновити видалені дані, скористайтеся кнопками форми ***MiniMarket***,– видаліть таблиці ***Продажі*** та ***Виробники***, а потім створіть і заповніть їх знову.

У звіті з лабораторної роботи порівняйте збережені процедури у SQL Server і Java DB. Укажіть переваги й недоліки кожного виду процедур.

Завдання для самостійного виконання

1. Додайте на панель ***Виробники*** кнопки ***Видалити рядок*** та ***Змінити рядок***, забезпечивши їхню функціональність.

2. Повторіть на панелі ***Товари*** всі кнопки, які є на панелі ***Виробники***, забезпечивши їхню функціональність.

3. Додайте на кнопкову форму панель ***Накладні*** для виконання операцій із таблицями ***Накладні*** та ***ТовариНакладних***. Розмістіть у ній такі кнопки та забезпечте їхню функціональність:

3.1. ***Створити*** – створює в базі даних порожні таблиці ***Накладні*** та ***ТовариНакладних***. Їхні схеми подано в п. "Постановка загального завдання" опису лабораторної роботи 1.

3.2. ***Видалити*** – видаляє з бази даних таблиці ***Накладні*** та ***ТовариНакладних*** (спочатку дочірню, а потім – батьківську).

3.3. ***Заповнити*** – додає дані в таблиці ***Накладні*** та ***ТовариНакладних***. Їхні дані подано в п. 1.1 опису лабораторної роботи 2.

3.4. ***Переглянути*** – виводить на консоль дані, що зберігають у таблицях ***Накладні*** та ***ТовариНакладних*** (спочатку в таблиці ***Накладні***, а потім – у ***ТовариНакладних***).

3.5. ***Переглянути за накладними*** – на відміну від завдання 3.4, виводить на консоль дані про першу накладну (із таблиць ***Накладні***

та ***ТоварНакладних***), потім – про другу накладну тощо. Замість кодів виробника і товару використайте відповідні назви з таблиць ***Виробники*** та ***Товари,*** як це було зроблено для кнопки ***Переглянути*** панелі ***Продажі***.

4. Повторіть п. 6.6 із використанням підготовлених запитів.

5. Виконайте п.п. 1 – 7 ходу роботи для індивідуальної бази даних.

6. Додайте на кнопкову форму панель ***Накладні SQL Server*** для виконання операцій із таблицями ***Накладні*** та ***ТоварНакладних***, які зберігають у базі даних ***Хліб.mdf*** (можна взяти із проєкту лабораторної роботи 2). Розмістіть у ній такі кнопки та забезпечте їхню функціональність:

6.1. ***Переглянути*** – виводить на консоль дані, що зберігають у таблицях ***Накладні*** та ***ТоварНакладних*** (спочатку у таблиці ***Накладні***, а потім – у ***ТоварНакладних***).

6.2. ***Переглянути за накладними*** – на відміну від завдання 6.1, виводить на консоль дані про першу накладну (із таблиць ***Накладні*** та ***ТоварНакладних***), потім – про другу накладну тощо. Замість кодів виробника і товару, використайте відповідні назви з таблиць ***Виробники*** та ***Товари,*** як це було зроблено для кнопки ***Переглянути*** панелі ***Продажі***. Таблиці ***Накладні*** та ***ТоварНакладних*** перебувають у базі даних ***Хліб.mdf*** (SQL Server)***,*** а ***Виробники*** та ***Товари*** – у базі даних ***MiniMarket*** (Java DB).

Додаток

Копіювання проєкту і бази даних

Ідеї виконання

Під час виконання лабораторної роботи 3 виникає необхідність скопіювати проєкт і базу даних з університетського комп'ютера на домашній і навпаки. Такий сценарій розглядають на прикладі переміщення в папку ***Лаб3***, розташовану на *Робочому столі*.

Д.1. Копіювання проєкту

Завдання

Скопіюйте проєкт у папку ***Лаб3***, розташовану на *Робочому столі*.

Виконання

1. Створіть на *Робочому столі* папку ***Лаб3***.

2. Поверніться в середовище NetBeans і дізнайтеся адресу папку, у якій зберігають проєкт. Для цього виконайте таке:

2.1. Виберіть із контекстового меню значка проєкту команду **Свойства**.

2.2. Виберіть елемент **Исходные файлы** в лівій панелі вікна **Свойства проекта**, потім із поля **Папка проекта** скопіюйте в буфер обміну шлях до папки. Після цього клацніть кнопку **ОК** (рис. 3.22).

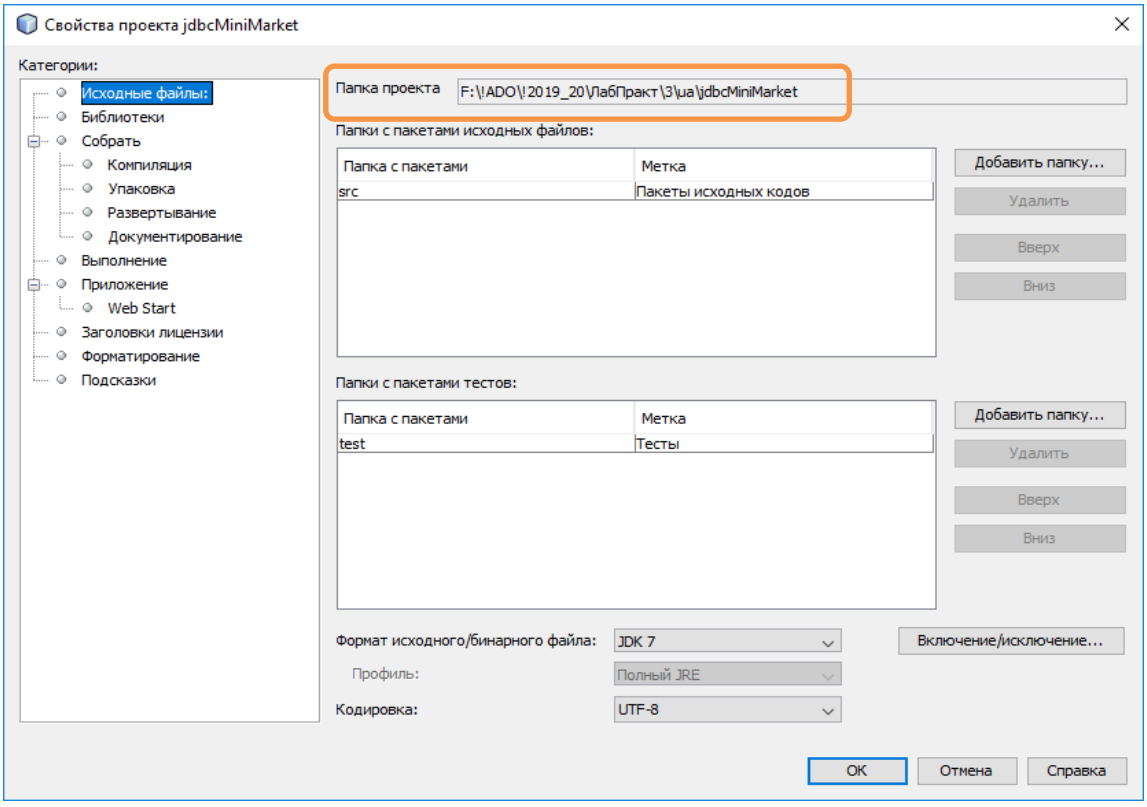


Рис. 3.22. **Адреса папки проєкту у вікні *Свойства проекта***

3. Вийдіть із середовища NetBeans і відкрийте вікно будь-якої папки, а потім уставте в адресний рядок уміст буфера.

4. Перейдіть на один рівень раніше у вікні папки проєкту, скопіюйте папку проєкту в буфер, а потім вставте її в папку ***Лаб3***, розташовану на *Робочому столі.*

Примітка. У разі відкривання проєкту на іншому комп'ютері може виникнути проблема з місцезнаходженням бібліотек **derbyclient.jar** та **derbytools.jar**. Для її вирішення повторно виконайте п. 3.1.

Д.2. Копіювання бази даних

Завдання

Скопіюйте базу даних у папку ***Лаб3***, розташовану на *Робочому столі*.

Виконання

1. Поверніться в середовище NetBeans, перейдіть у вікно **Службы** та за допомогою контекстового меню значка **Java DB** зупиніть сервер.

2. Дізнайтеся адресу папки, у якій зберігають базу даних. Для цього виконайте таке:

2.1. Виберіть команду **Свойства** з контекстового меню значка **Java DB**.

2.2. Скопіюйте в буфер обміну шлях до папки з поля **Местонахождение базы данных** вікна **Свойства Java DB**. Після цього клацніть кнопку **ОК** (рис. 3.23).

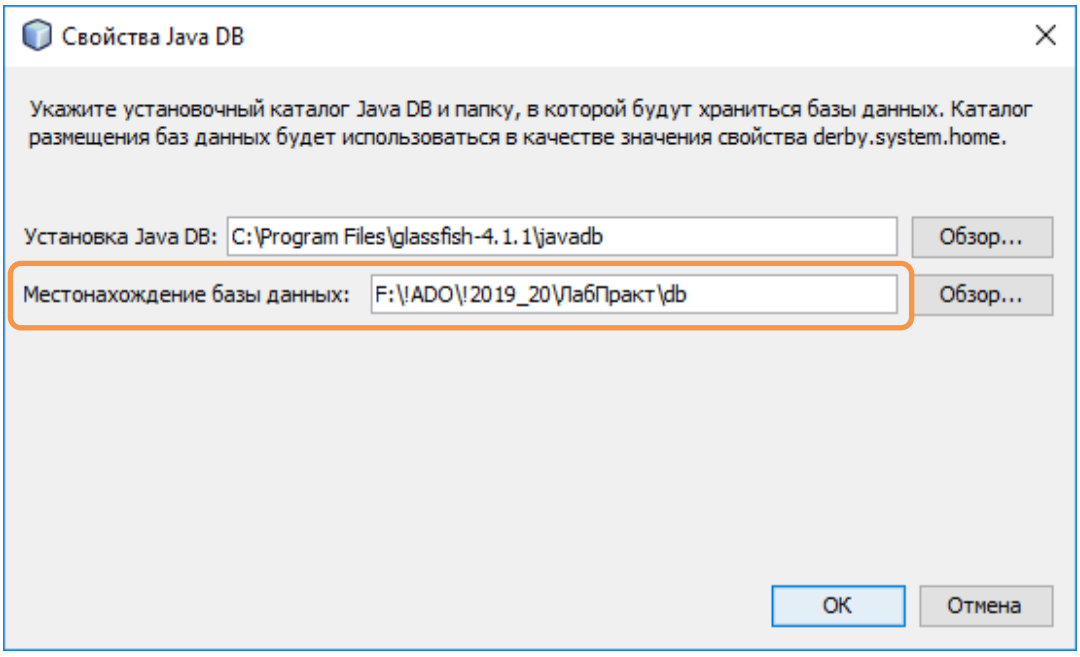


Рис. 3.23. **Адреса папки, у якій зберігають базу даних**

3. Вийдіть із середовища NetBeans і відкрийте вікно будь-якої папки, а потім уставте в адресний рядок уміст буфера.

4. Скопіюйте папку, у якій зберігають базу даних, у буфер і вставте її у папку ***Лаб3***, розташовану на *Робочому столі*.

Примітка. У разі відкривання проєкту на іншому комп'ютері може виникнути проблема з місцезнаходженням сервера. Для її вирішення виберіть команду **Свойства** з контекстового меню значка **Java DB** у вікні **Службы**. Потім у полі **Установка Java DB** вікна **Свойства Java DB** укажіть шлях до підпапки **javadb** папки сервера **glassfish**. Для встановлення адреси скористайтеся першою кнопкою **Обзор**. Після цього клацніть кнопку **ОК** (рис. 3.24).

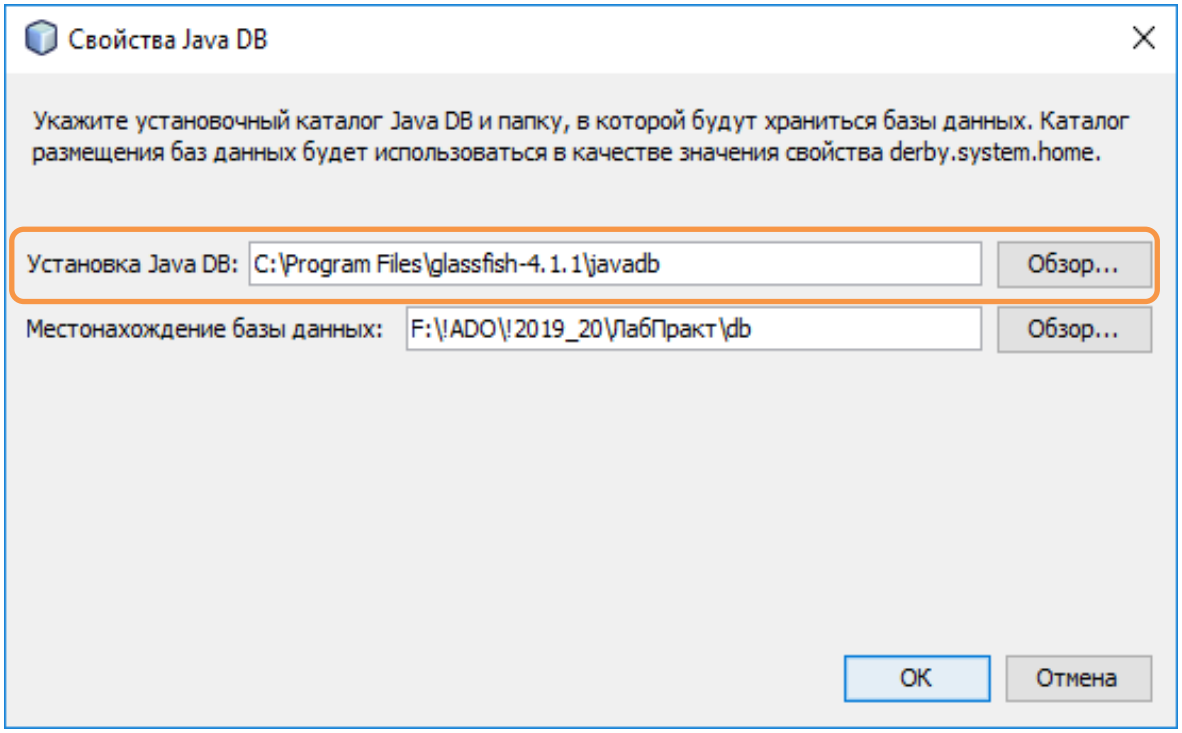


Рис. 3.24. **Адреса папки, у якій розміщено сервер бази даних**

Розділ 2
Сучасні засоби доступу до даних

Лабораторна робота 4
Розробка програм із використанням типізованих наборів даних

Цілі лабораторної роботи:

1. Набуття практичних навичок у створенні типізованих наборів даних.
2. Вироблення вмінь зміни властивостей об'єктів у типізованих наборах даних.
3. Набуття практичних навичок у використанні адаптерів таблиць.
4. Набуття практичних навичок у відображенні даних у застосунках на основі типізованих наборів даних.
5. Удосконалення навичок у роботі з інтегрованим середовищем розроблення Visual C# і довідковою системою Microsoft Developer Network (MSDN).

Перед виконанням роботи студент має знати:

основи використання бібліотеки Windows Forms;
структурні елементи бази даних і їхні властивості;
основи побудови SQL-запитів;
організацію системи створення з'єднання в ADO.NET;
принципи оброблення подій у C#-програмі.

Після виконання роботи студент має вміти:

самостійно розробляти C#-застосунки із графічним інтерфейсом користувача для роботи з базою даних, використовуючи візуальні засоби;
створювати локальні таблиці, використовуючи програмні засоби C#;
забезпечувати програмну реалізацію виконання CRUD-операцій із базою даних на основі типізованих наборів даних;
використовувати візуальні засоби прив'язування даних під час розроблення інтерфейсу користувача.

Підготовча частина

Хід роботи

1. Побудова кнопкової форми застосунку.
2. Створення типізованого набору даних.
3. Реалізація форм для ведення довідкових таблиць.
4. Розроблення форми з деталізованим поданням даних.
5. Формування інтерфейсу для ведення ієрархічних даних.
6. Додавання адаптера таблиці з агрегованою функцією.

Форма звітності

За результатами виконання роботи необхідно оформити електронний звіт засобами Word. За кожним завданням слід зробити скриншоти з результатами виконання, а також подати код оброблювача події. У кінці завдань зазначено, що саме слід подати у звіті.

За результатами кожного виконаного завдання з п. "Завдання для самостійного виконання" слід подати постановку завдання, скриншот форми та відповідний програмний код.

Критерії оцінювання

У 12-бальній системі оцінку результату захисту лабораторної роботи формують за такими правилами:

1. За виконання всіх пунктів практичної частини може бути виставлено від 0 до 4 балів.

2. За виконання завдань п. "Завдання для самостійного виконання" може бути виставлено:

від 0 до 1 бала за кожне завдання 1, 2, 4 – 6;

від 0 до 2 балів за кожне завдання 3, 7;

від 0 до 3 балів за завдання 8;

від 0 до 2 балів за завдання 9.

3. За кілька варіантів виконання одного із завдань додають 1 бал.

4. За вибір варіанта, який із кількох варіантів виконання є оптимальним, та обґрунтування вибору додають 1 бал.

5. За виконання кожного завдання в іншій СУБД (Oracle, MySQL, DB2, PostgreSQL тощо) додають по 1 балу.

6. За запізнення із захистом лабораторної роботи знімають 2 бали за кожний тиждень.

Набрану кількість балів із відповідей на кожне завдання лабораторної роботи підсумовують. У результаті такого підрахунку студент може набрати від 0 до 12 балів.

У разі запізнення із захистом лабораторної роботи понад три тижні студент виконує завдання роботи на основі індивідуальної бази даних. Роботу оцінюють за описаними раніше критеріями.

Практична частина

Постановка загального завдання

Вивчення засобів роботи з типізованими наборами даних здійснюють шляхом створення застосунку ***типізованийХліб***.

Керування роботою застосунку здійснюють за допомогою кнопкової форми ***Хліб*** (рис. 4.1). Кнопки призначено для виклику відповідних функціональних форм.

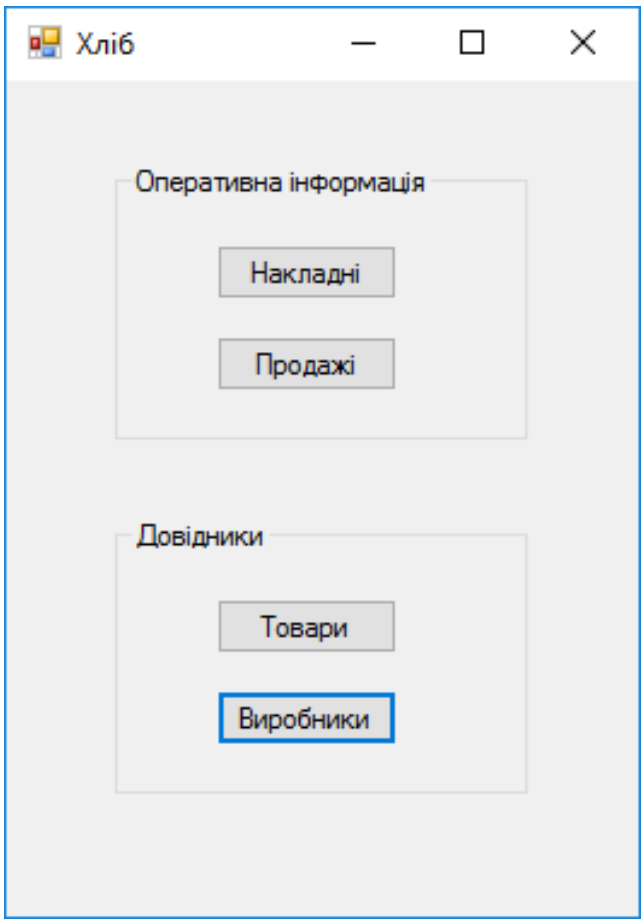


Рис. 4.1. **Кнопкова форма *Хліб***

На рис. 4.2 – 4.5 подано функціональні форми застосунку. Із їхньою допомогою можна переглядати й змінювати дані (обновляти, додавати та видаляти) у відповідних таблицях бази даних ***Хліб***.

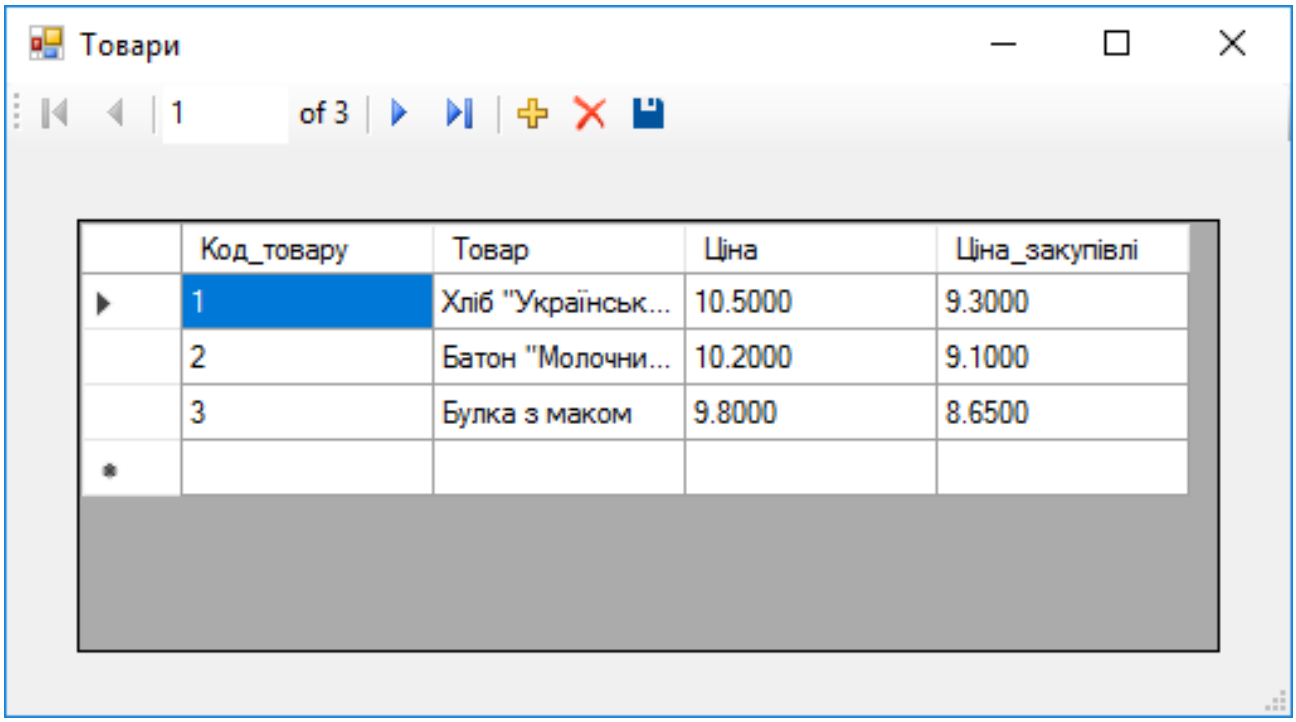


Рис. 4.2. **Форма для ведення таблиці *Товари***

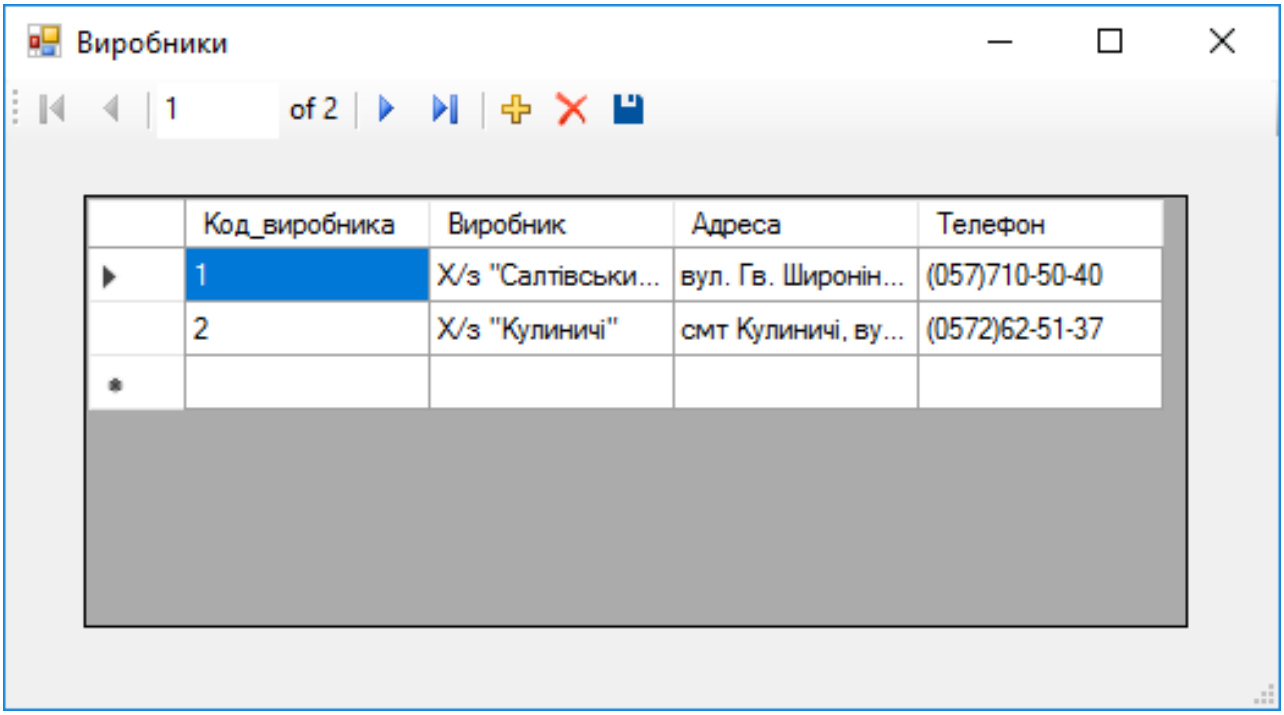


Рис. 4.3. **Форма для ведення таблиці *Виробники***

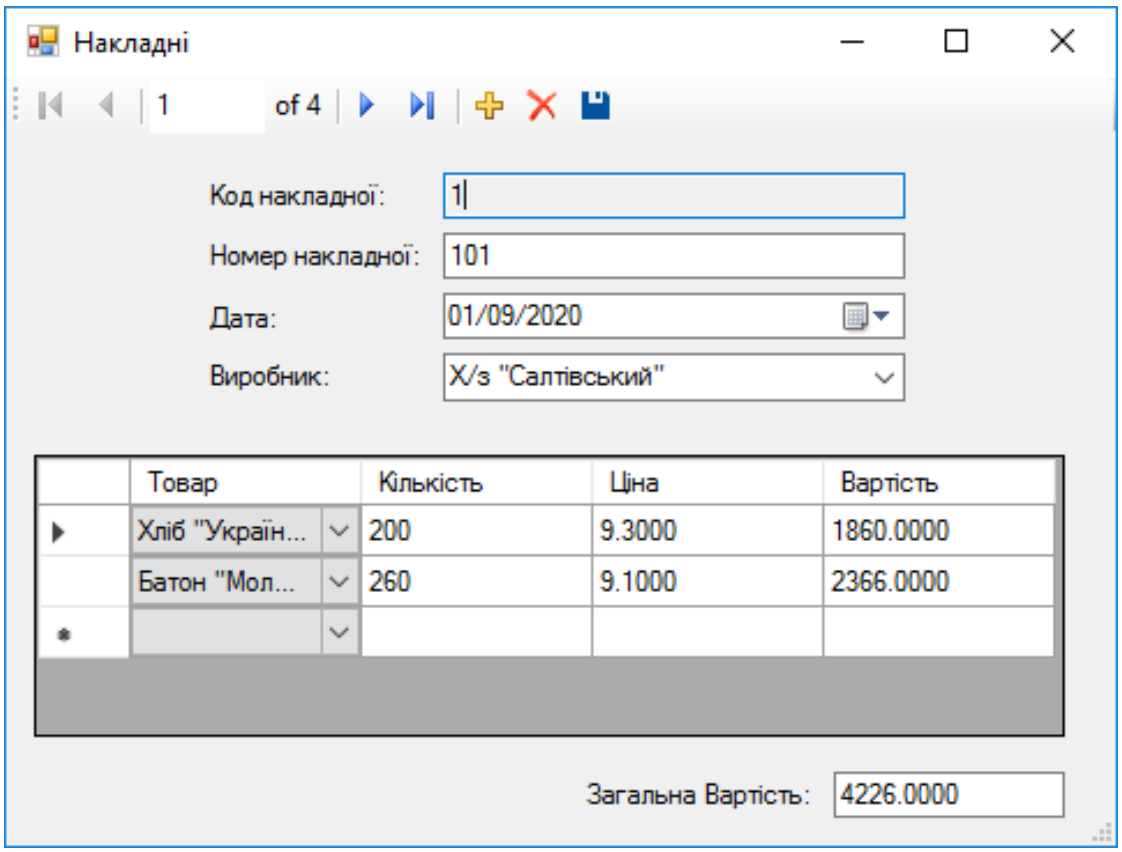


Рис. 4.4. **Форма для ведення таблиць *Накладні* й *Товари_накладних***

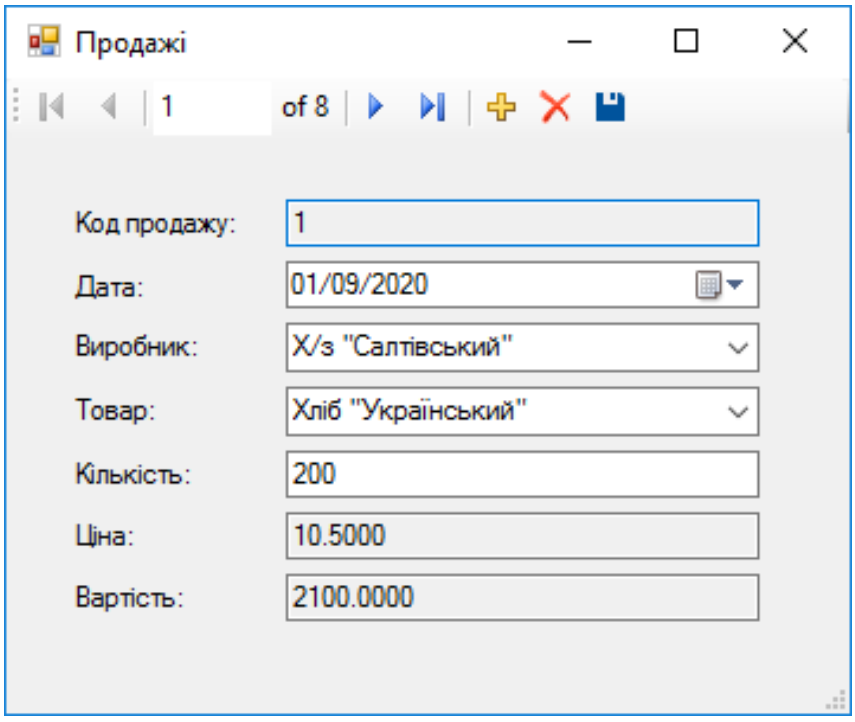


Рис. 4.5. **Форма для ведення таблиці *Продажі***

1. Побудова кнопкової форми застосунку

Завдання

Створіть застосунок і побудуйте в ньому кнопкову форму (рис. 4.6).

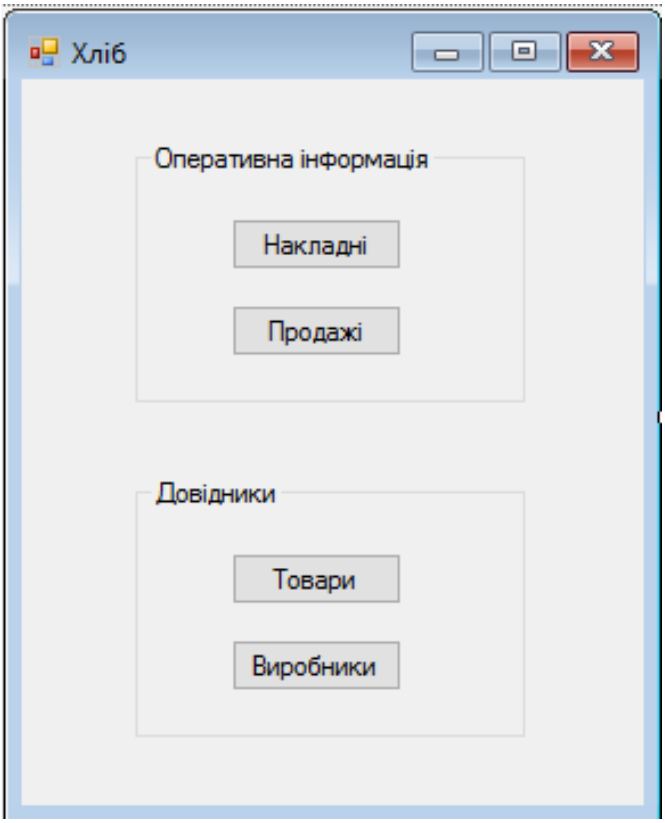


Рис. 4.6. **Кнопкова форма**

Ідеї виконання

Оскільки кнопкова форма Хліб повністю збігається з однойменною формою у проєкті ***роз'єднанийХліб***, який створено в лабораторній роботі 2, можна скопіювати елементи керування зі старої форми на нову. У разі відсутності проєкту ***роз'єднанийХліб*** слід повторити дії, описані в завд. 1.3 лабораторної роботи 2.

Виконання

1. Відкрийте Visual Studio.

2. Створіть проєкт Windows Forms мовою C# з ім'ям ***типізованийХліб*** і збережіть його.

3. У вікні **Solution Explorer** виділіть значок ***Form1*** і для файлу ***Form1.cs*** у вікні властивостей задайте ім'я ***formХліб.cs***.

4. Для властивості **Text** форми задайте значення ***Хліб***.

5. Скопіюйте всі елементи керування із кнопкової форми ***Хліб*** проєкт ***роз'єднанийХліб***, який створено в лабораторній роботі 2 на однойменну форму в поточному проєкті ***типізованийХліб***. Для цього виконайте таке:

5.1. Відкрийте проєкт ***роз'єднанийХліб***, а в ньому – кнопкову форму ***Хліб*** у режимі конструктора.

5.2. Виділіть всі елементи керування на формі ***Хліб*** проєкту ***роз'єднанийХліб*** і скопіюйте їх у буфер обміну.

5.3. Поверніться у проєкт ***типізованийХліб*** і вставте з буфера елементи керування на форму ***Хліб***.

5.4. У разі потреби змініть розміри форми ***Хліб*** у проєкті ***роз'єднанийХліб***.

5.5. Закрийте проєкт ***роз'єднанийХліб***.

6. Збережіть зміни, зроблені у проєкті.

2. Створення типізованого набору даних

2.1. Додавання бази даних до нового проєкту

Завдання

Скопіюйте базу даних ***ХлібПрізвище.mdf***, яку використовували під час виконання лабораторної роботи 2.

Виконання

1. Перегляньте вікно **Server Explorer**. Якщо в ньому є значок бази даних ***ХлібПрізвище***, видаліть його, вибравши з його контекстового меню команду **Delete**.

Примітка. Замість слова ***Прізвище***, в імені файлу бази даних має бути прізвище студента, наприклад, ***ХлібПетренко.mdf***. У подальших описах база даних буде мати узагальнене ім'я ***Хліб***.

2. Виберіть команду **Project – Existing Item** у меню Visual Studio.

3. Установіть фільтр **All Files** у вікні **Add New Item**, перейдіть у папку проєкту ***роз'єднанийХліб***, виберіть файл ***Хліб.mdf*** і клацніть кнопку **Add**.

У вікнах **Solution Explorer** і **Server Explorer** з'явилися значки бази даних ***Хліб.mdf***.

4. Перегляньте склад бази даних ***Хліб***. Зверніть увагу на те, щоб вона містила п'ять таких таблиць: ***Товари***, ***Виробники***, ***Продажі***, ***Накладні*** та ***ТовариНакладних***, а в них зберігали дані про результати надходження та продажів товарів принаймні за два дні від двох виробників.

5. Видаліть з'єднання з базою даних ***Хліб.mdf***, вибравши з контекстового меню її значка у вікні **Server Explorer** команду **Delete**.

2.2. Побудова типізованого набору даних

Завдання

Створіть типізований набір даних ***ХлібDataSet***, використовуючи майстра налаштування джерела даних.

Виконання

1. Виберіть команду **Add New Data Source** в меню **Project** (у Visual Studio 2010 – у меню **Data**).

2. Виберіть значок **Database** як джерело даних для застосунку в першому вікні майстра й клацніть кнопку **Next**.

3. Виберіть значок **Dataset** як модель бази даних для застосунку у другому вікні майстра й клацніть кнопку **Next**.

4. Клацніть кнопку **New connection** у наступному вікні майстра.

5. Створіть нове з'єднання з базою даних, що зберігають у файлі ***Хліб.mdf***. Для цього виконайте такі операції у вікні **Add Connection**.

5.1. Перевірте, чи встановлено значення **Microsoft SQL Server Database File (SqlClient)** у полі **Data source**. Якщо ні, то скористайтеся кнопкою **Change**.

5.2. Установіть шлях до файлу бази даних ***Хліб.mdf***, розміщений у папці проєкту ***типізованийХліб***. Для цього скористайтеся кнопкою **Browse**.

5.3. Перевірте з'єднання за допомогою кнопки **Test Connection**.

5.4. Клацніть кнопку **ОК**.

6. Після повернення у вікно **Choose Your Data Connection** клацніть кнопку **Next**.

7. Клацніть кнопку **Next** в наступному вікні майстра, погодившись із тим, що рядок з'єднання буде зберігатися у файлі конфігурації застосунку.

8. Установіть прапорець у вузлі **Tables** у наступному вікні майстра, указавши, що всі таблиці бази даних будуть відображати в наборі даних, а потім клацніть кнопку **Finish**.

У вікні **Solution Explorer** з'явився вузол ***ХлібDataSet.xsd***, що подає типізований набір даних (рис. 4.7).

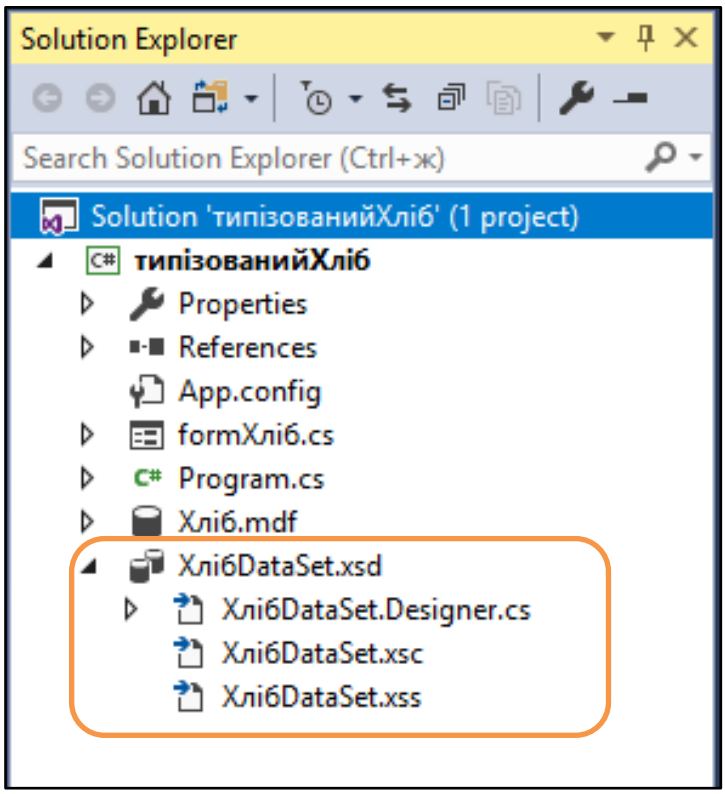


Рис. 4.7. **Вузол *ХлібDataSet.xsd***

9. Збережіть зміни, зроблені у проєкті.

2.3. Дослідження структури типізованого набору даних

Завдання

Визначте нові об'єкти, які з'явилися під час створення типізованого набору даних ***ХлібDataSet***.

Виконання

1. Двічі клацніть значок вузла типізованого набору даних ***ХлібDataSet.xsd***. З'явилося вікно конструктора набору даних (рис. 4.8). У ньому подано зображення таблиць набору, які повторюють аналогічні таблиці бази даних. У нижній частині кожної таблиці розташовано їхні адаптери. Таблиці пов'язано відношеннями.

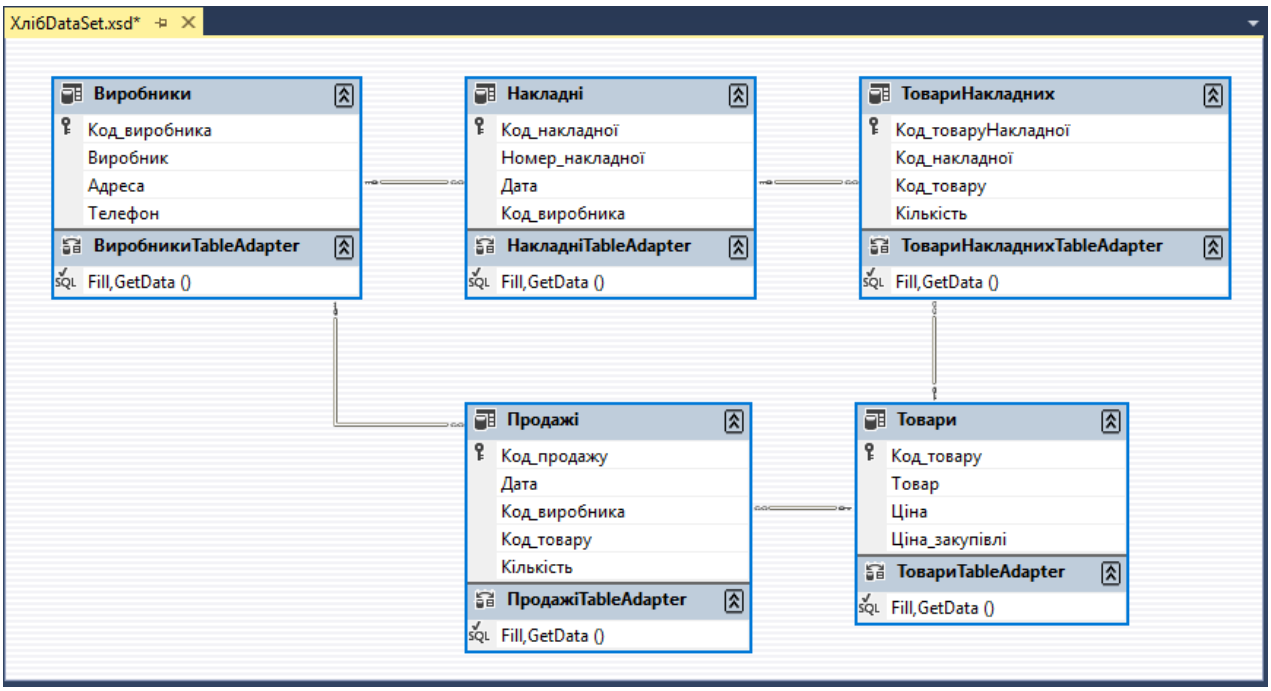


Рис. 4.8. **Набір даних у вікні конструктора**

2. Двічі клацніть перший значок підвузла ***ХлібDataSet.Designer.cs***, розміщений у вузлі ***ХлібDataSet.xsd***. З'явилося вікно коду, який згенеровано системою Visual Studio під час створення типізованого набору даних. У ньому зберігають описи класів, їхніх методів тощо. Із назвами нових класів можна ознайомитися в середньому списку, що розкривається, розташованому у верхній частині вікна (рис. 4.9). Ознайомившись з умістом вікна, закрийте його.

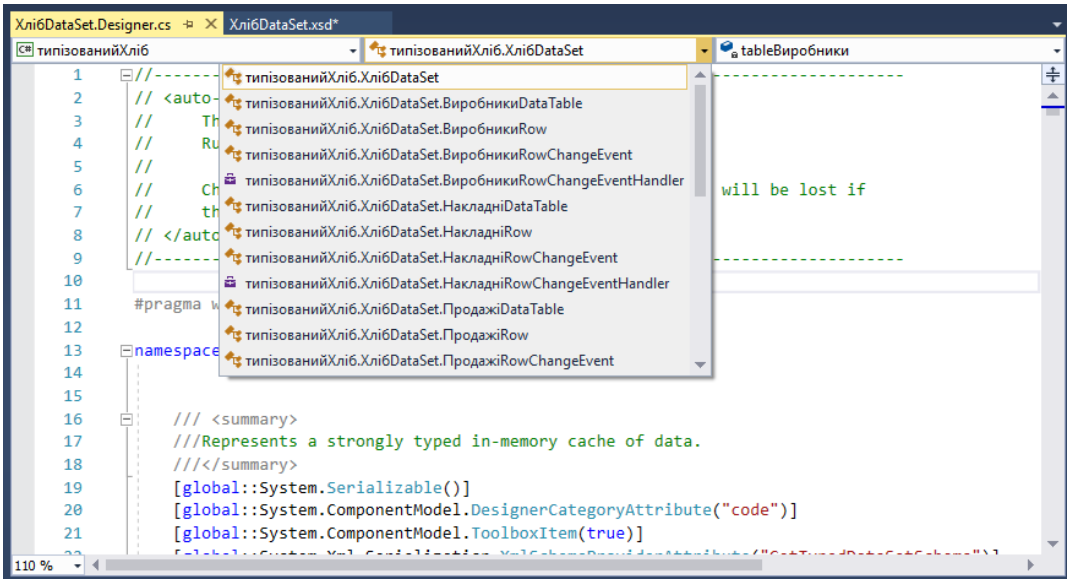


Рис. 4.9. **Вікно коду типізованого набору даних**

3. Клацніть правою клавішею мишки на значок вузла типізованого набору даних ***ХлібDataSet.xsd*** і в контекстовому меню виберіть команду **View Code**. З'явилося вікно коду часткового класу (того, що може описуватися в кількох файлах, англ. *partial*) типізованого набору даних (рис. 4.10). У ньому можна додавати код, який не видаляється в разі повторного генерування набору даних. Ознайомившись з умістом вікна, закрийте його. Після закривання вікна у вузлі набору даних ***ХлібDataSet.xsd*** з'явився значок файлу ***ХлібDataSet.cs***.

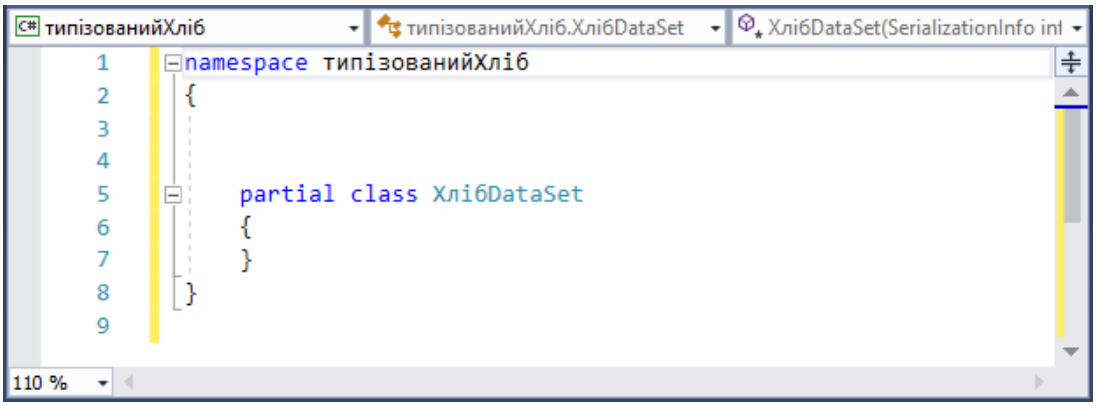


Рис. 4.10. **Вікно коду часткового класу типізованого набору даних**

4. Перейдіть у вікно **Data Sources**. У ньому відображено значок типізованого набору даних ***ХлібDataSet*** у вигляді вузла. Його підвузлами є таблиці. Якщо розкрити підвузол будь-якої таблиці, відображаються її поля. Якщо таблиця має дочірню, то під полями відображаються дочірні підтаблиці (рис. 4.11). Ознайомтеся з умістом кожного вузла таблиці.

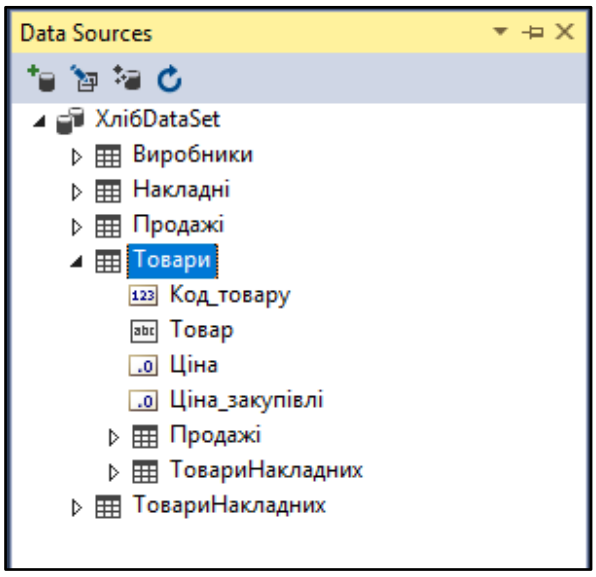


Рис. 4.11. **Поля й пов'язані дочірні таблиці у вузлі таблиці *Товари***

У звіті з лабораторної роботи подайте скриншоти набору даних у вікні конструктора і вікні **Data Source** з розкритим вузлом ***Накладні***, а також тег **connectionStrings** із файлу **App.config**, що відображено у вікні **Solution Explorer**. Опишіть призначення кожного параметра тегу **connectionStrings**.

3. Реалізація форм для ведення довідкових таблиць

3.1. Форма Товари

Завдання

У наявний застосунок додайте форму ***Товари***, за допомогою якої можна переглядати й змінювати дані (обновляти, додавати та видаляти) в однойменній таблиці бази даних (рис. 4.12).

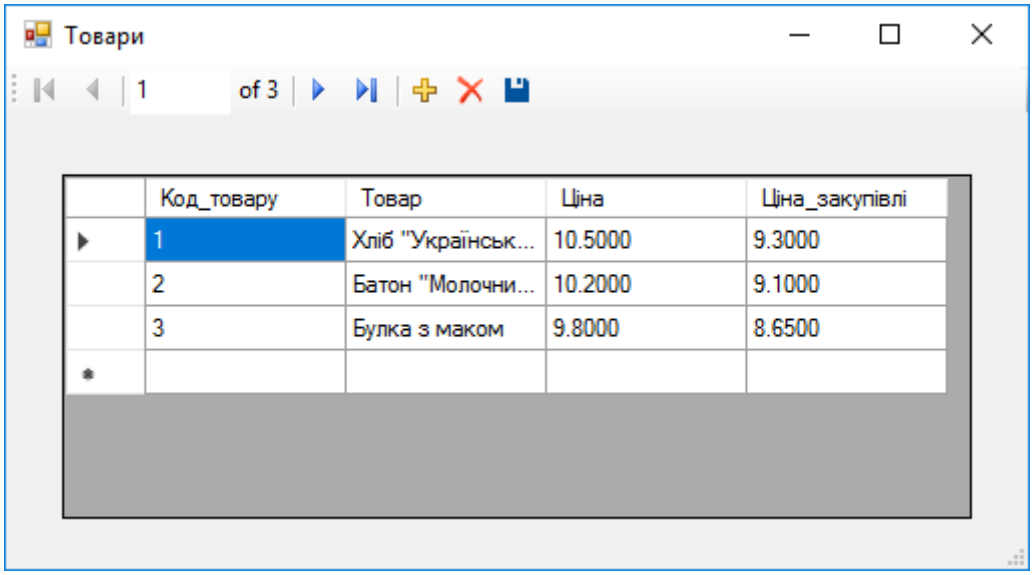


Рис. 4.12. **Форма *Товари***

Виконання

1. Додайте в застосунок нову форму з ім'ям ***formТовари*** та заголовком ***Товари***.

2. Перетягніть вузол таблиці ***Товари*** з вікна **Data Sources** на форму ***Товари***. На формі, крім елемента керування DataGridView із заголовком даних таблиці ***Товари***, з'явилася панель навігатора, а в області компонентів – ряд об'єктів, які забезпечують роботу з даними таблиці в типізованому наборі даних (рис. 4.13). Ознайомтеся з ними та визначте призначення кожного.

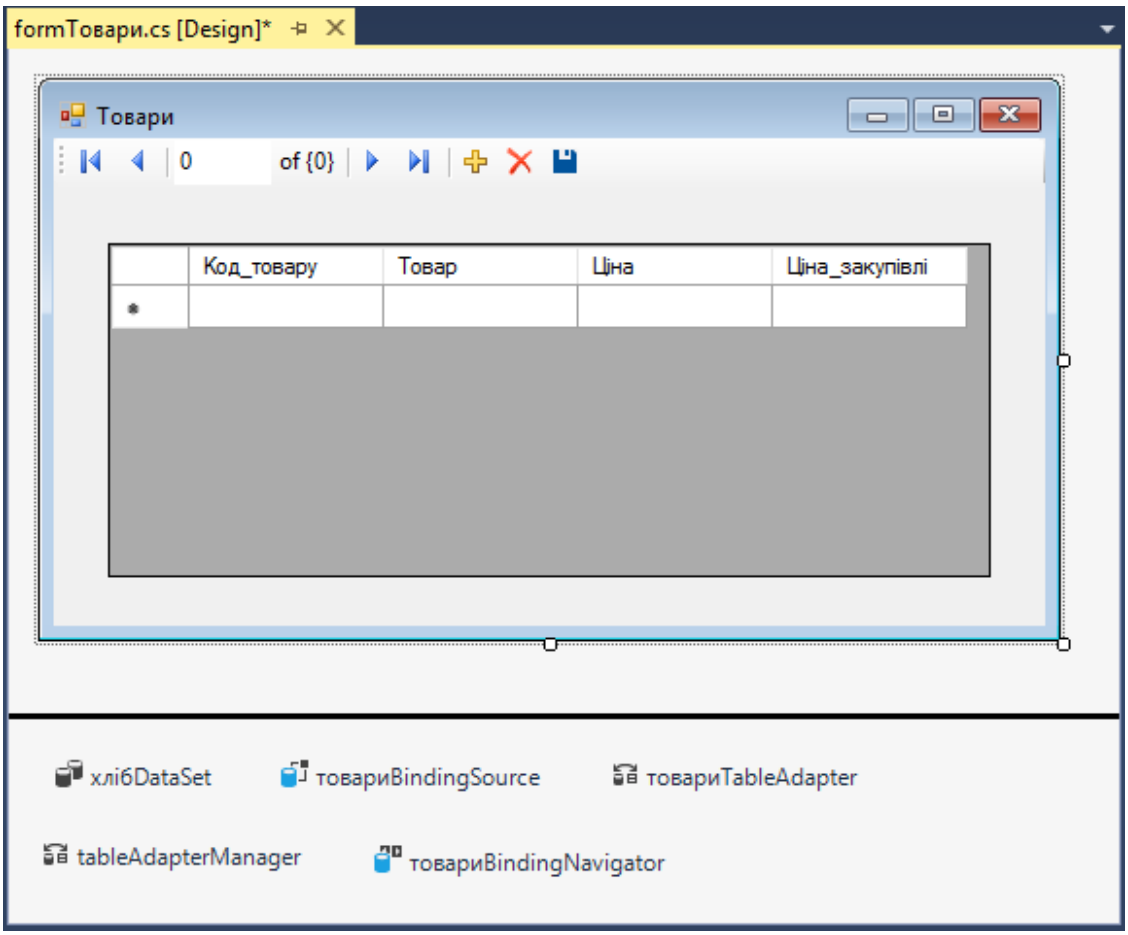


Рис. 4.13. **Вікно форми *Товари* після додавання однойменної таблиці**

3. Установіть потрібну ширину форми й елемента DataGridView, щоб дані таблиці відображалися повністю.

4. Перейдіть у вікно конструктора форми ***Хліб*** і додайте код оброблювача події "Клацання кнопки Товари".

```
private void buttonТовари_Click(object sender, EventArgs e)
        {
            formТовари вікноТовари = new formТовари();
            вікноТовари.ShowDialog();
        }
```

5. Запустіть програму на виконання й перевірте функціональність форми ***Товари*** шляхом переміщення записами, зміни, додавання і видалення записів. Спочатку виконайте операції без збереження, а потім зі збереженням у базі даних зроблених змін.

6. Закрийте форми ***Товари*** та ***Хліб***.

7. Збережіть зміни, зроблені у проєкті.

У звіті з лабораторної роботи подайте скриншот форми ***Товари***, а також код оброблювачів подій ***formТовари_Load*** та ***товаруBinding-NavigatorSaveItem_Click***. Поясніть призначення операторів, що містяться в тілі цих оброблювачів.

3.2. Форма Виробники

Завдання

У застосунок ***типізованийХліб*** додайте форму ***Виробники***, за допомогою якої можна переглядати та змінювати дані в однойменній таблиці бази даних (рис. 4.14).

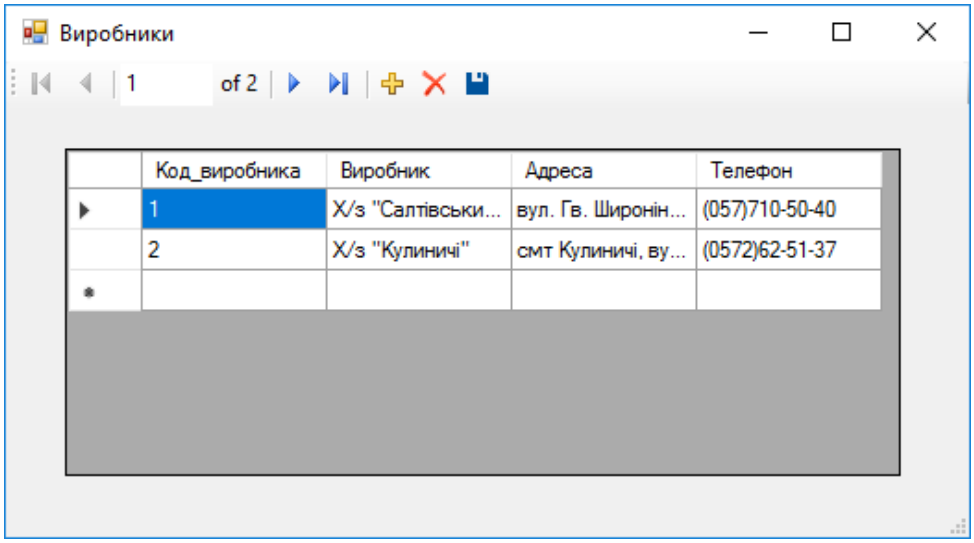


Рис. 4.14. **Форма *Виробники***

Виконання

Виконайте дії аналогічні тим, які описано для форми ***Товари***.

У звіті з лабораторної роботи подайте скриншот форми ***Виробники***, а також код оброблювача події ***formВиробники_Load***. Поясніть різницю між цим кодом і кодом однойменного оброблювача події в лабораторній роботі 2.

4. Розроблення форми з деталізованим поданням даних

Ідеї та кроки виконання

Вивчення візуальних засобів налаштування інтерфейсу користувача здійснюють на прикладі форми ***Продажі***. На ній подають дані таблиці ***Продажі***. Оскільки виробник і товар у цій таблиці подають кодами, для зручності користувачу бажано відображати їхні назви, а також ціну

та вартість товару. Назву і ціну зберігають у довідкових таблицях, а вартість – обчислювана величина.

Щоб додавати нові дані про продажі товарів, потрібно надати можливість вибирати назви виробників і товарів зі списків. На формі такі списки найкраще подавати елементом поле зі списком (comboBox). Джерелом даних для них є відповідна довідкова таблиця. У полі зі списком вибирають назву (товару чи виробника), а відповідний код записують у базу даних. Аналогічним чином використовують поле зі списком для зміни даних.

Ціну товару і його вартість найлегше подати на формі, якщо попередньо додати ці поля в локальну таблицю. Таке додавання виконують у вікні конструктора типізованого набору даних.

Оскільки форма ***Продажі*** має містити різноманітні елементи керування для виконання CRUD-операцій, її краще подавати в деталізованому вигляді. У цьому разі користувач одночасно має справу тільки з одним записом таблиці ***Продажі***, на відміну від елемента DataGridView, у якому одночасно відображено кілька записів, як це використовували для простих таблиць ***Товари*** та ***Виробники***.

Розроблення форми ***Продажі*** складається з таких кроків:

1. Налаштування типізованого набору даних.
2. Створення форми.
3. Налаштування форми.

4.1. Налаштування типізованого набору даних

Завдання

Додайте стовпці ***Ціна*** та ***Вартість*** у локальну таблицю ***Продажі*** типізованого набору даних (рис. 4.15).

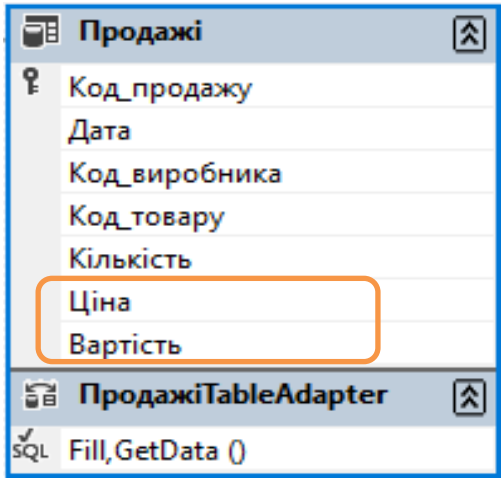


Рис. 4.15. **Локальна таблиця *Продажі***

Виконання

1. Перейдіть у вікно конструктора типізованого набору даних, двічі клацнувши значок вузла набору даних ***ХлібDataSet.xsd*** у вікні **Solution Explorer**.

2. Клацніть лінію зв'язку між таблицями ***Товари*** та ***Продажі*** у вікні конструктора. Потім у вікні властивостей уведіть нове значення ***Товари-Продажі*** для властивості **Name**.

3. Клацніть правою клавішею мишки назву таблиці ***Продажі*** та виберіть із контекстового меню команду **Add – Column** і у вікні властивостей установіть значення, наведені в табл. 4.1.

Таблиця 4.1

Значення властивостей нового стовпця

Властивості	Значення
Name	Ціна
DataType	System.Decimal
Expression	Parent(ТовариПродажі).Ціна

4. Аналогічним чином додайте стовпець ***Вартість*** у таблицю ***Продажі*** типізованого набору даних, установивши у вікні властивостей значення, наведені в табл. 4.2.

Таблиця 4.2

Значення властивостей стовпця Вартість

Властивості	Значення
Name	Вартість
DataType	System.Decimal
Expression	Parent(ТовариПродажі).Ціна*Кількість

5. Закрийте вікно конструктора типізованого набору даних зі збереженням зроблених змін.

4.2. Створення форми

Завдання

Побудуйте форму ***Продажі*** для виконання CRUD-операцій із таблицею ***Продажі*** (рис. 4.16).

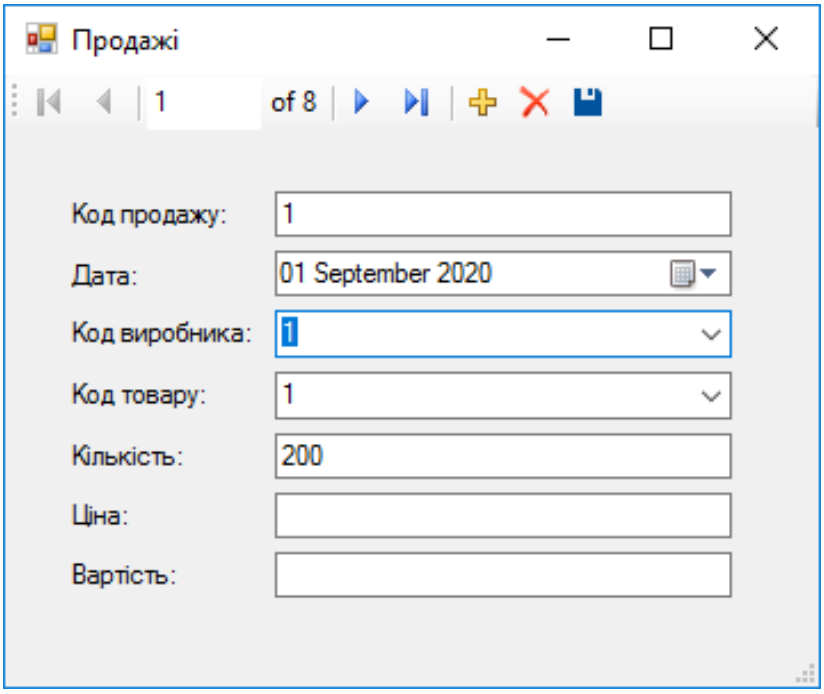


Рис. 4.16. **Початковий вигляд форми *Продажі***

Виконання

1. Додайте у проєкт нову форму, давши їй ім'я ***formПродажі*** та заголовок ***Продажі***.

2. Клацніть вузол таблиці ***Продажі*** у вікні **Data Sources** і з її списку, що розкривається, виберіть подання **Details** (**Таблиця**).

3. Розкрийте вузол таблиці ***Продажі*** та для стовпця ***Код_товару*** виберіть у списку, що розкривається, подання **ComboBox**.

4. Повторіть п. 3 для стовпця ***Код_виробника***.

5. Перетягніть вузол таблиці ***Продажі*** з вікна **Data Sources** на форму ***Продажі***. На формі з'явилися елементи керування для відображення даних таблиці й панель навігатора, а в області компонентів – ряд об'єктів, які забезпечують роботу з даними таблиці в типізованому наборі даних (рис. 4.17). Ознайомтеся з ними та визначте призначення кожного.

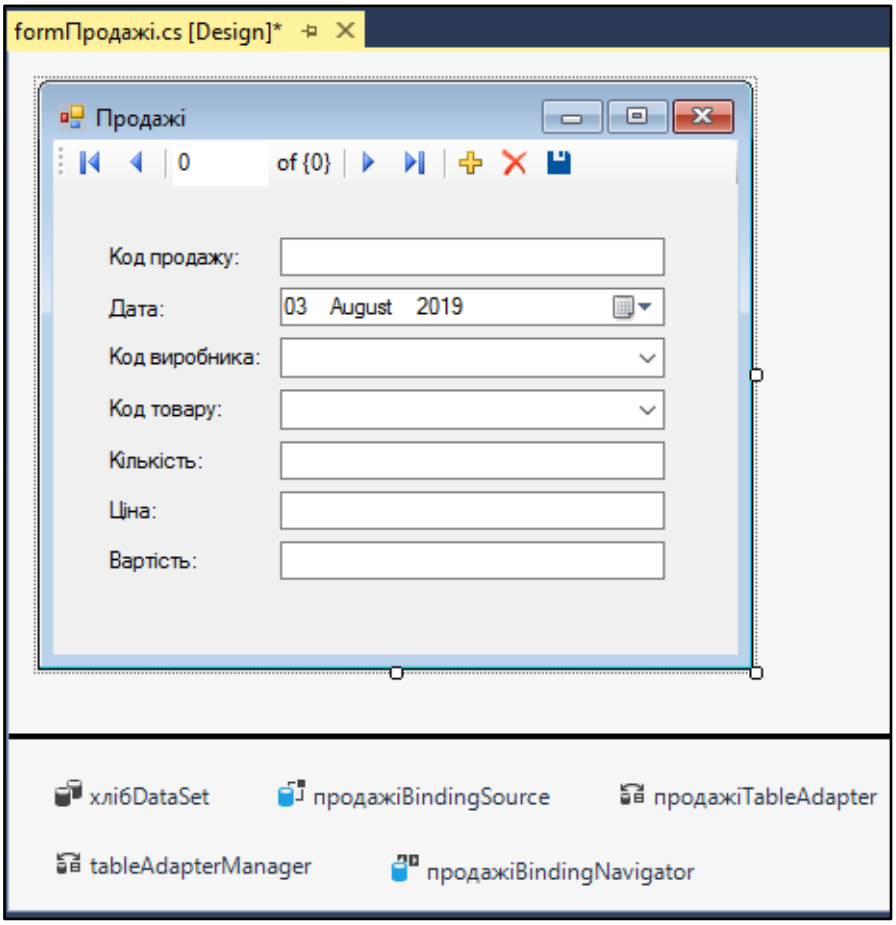


Рис. 4.17. **Вікно форми *Продажі* після додавання однойменної таблиці**

6. Перейдіть у вікно конструктора форми ***Хліб*** і додайте код оброблювача події "Клацання кнопки Продажі".

```
private void buttonПродажі_Click(object sender, EventArgs e)
{
    formПродажі вікноПродажі = new formПродажі();
    вікноПродажі.ShowDialog();
}
```

7. Запустіть програму на виконання й перевірте функціональність форми ***Продажі*** шляхом переміщення, зміни, додавання та видалення записів.

Форма Продажі має ряд недоліків (див. рис. 4.16). У полях ***Ціна*** та ***Вартість*** не відображаються дані. У полях зі списком ***Код_товару*** і ***Код_виробника*** відображаються тільки поточні числові значення, а не назви та їхні переліки під час відкривання списків. Ці недоліки усувають у наступному завданні.

8. Закрийте форми ***Продажі*** та ***Хліб***.

4.3. Налаштування форми

Завдання

Удоскональте форму ***Продажі*** для більш зручного виконання CRUD-операцій із таблицею ***Продажі*** (рис. 4.18).

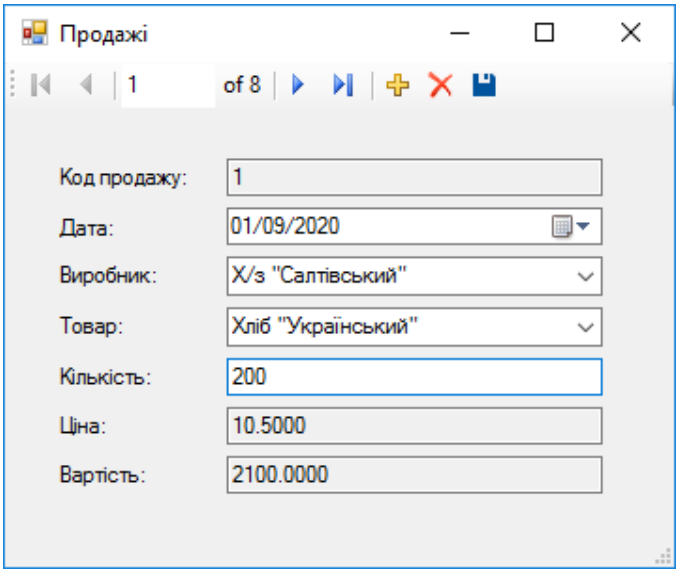


Рис. 4.18. **Удосконалений вигляд форми *Продажі***

Виконання

1. Перетягніть вузол таблиці ***Товари*** з вікна **Data Sources** на поле зі списком ***Код_товару***.

2. Повторіть п. 2 для поля зі списком ***Код_виробника***.

3. Змініть значення властивості Text для написів ***Код_товару*** та ***Код_виробника*** на ***Товар:*** та ***Виробник:***, відповідно.

4. В елементі ***damaDateTimePicker*** установіть значення ***Short*** для властивості **Format**.

5. Текстовим полям ***код_продажуTextBox***, ***цінаTextBox*** та ***вартістьTextBox*** установіть значення ***True*** для властивості **ReadOnly**.

6. Запустіть програму на виконання й перевірте функціональність форми ***Продажі*** шляхом переміщення, зміни, додавання і видалення записів. Переконайтеся у правильному функціонуванні полів зі списком ***Виробник*** і ***Товар***, а також текстових полів ***Ціна*** та ***Вартість***.

У звіті з лабораторної роботи подайте скриншот форми ***Продажі*** в режимі конструктора разом з областю компонентів. Опишіть призначення кожного об'єкта, розміщеного в цій області.

5. Формування інтерфейсу для ведення ієрархічних даних

Ідеї та кроки виконання

Вивчення візуальних засобів налаштування інтерфейсу користувача виду **master-detail** здійснюють на прикладі форми ***Накладні***. На ній подають дані таблиць ***Накладні*** та ***ТовариНакладних***, пов'язані відношенням "один-до-багатьох". Щоб форма мала максимально наближений вигляд до документа Накладна, на ній одночасно відображено один запис із таблиці ***Накладні*** та кілька записів із таблиці ***ТовариНакладних***. Тому для відображення записів із таблиці ***Накладні*** вибирають форму в деталізованому вигляді. Записи з таблиці ***ТовариНакладних*** на ній подають за допомогою елемента DataGridView.

Оскільки з таблиці ***ТовариНакладних*** мають відображати саме ті записи, що стосуються поточного запису з таблиці ***Накладні***, під час побудови форми з використанням вікна **Data Source** слід вибирати значок таблиці ***ТовариНакладних***, що є підвузлом таблиці ***Накладні***, а не окремий вузол. Інакше в елементі DataGridView буде відображено відразу всі записи таблиці ***ТовариНакладних***, а не тільки ті, що відповідають поточному запису з таблиці ***Накладні***.

Для удосконалення форми слід окремо налаштувати її загальну частину, що відображає дані таблиці ***Накладні*** та подробиці (елемент DataGridView), у якому подають дані таблиці ***ТовариНакладних***. Дії в першій із них аналогічні описаним під час побудови форми ***Продажі***. У другій частині потрібно виконати операції зі стовпцями елемент DataGridView, а також забезпечити доступ до даних таблиці ***Товари***.

Розроблення форми ***Накладні*** складається з таких кроків:

1. Налаштування типізованого набору даних.
2. Створення форми.
3. Налаштування елемента DataGridView.

5.1. Налаштування типізованого набору даних

Завдання

Додайте стовпці ***Ціна*** та ***Вартість*** у локальну таблицю ***ТовариНакладних*** типізованого набору даних (рис. 4.19).

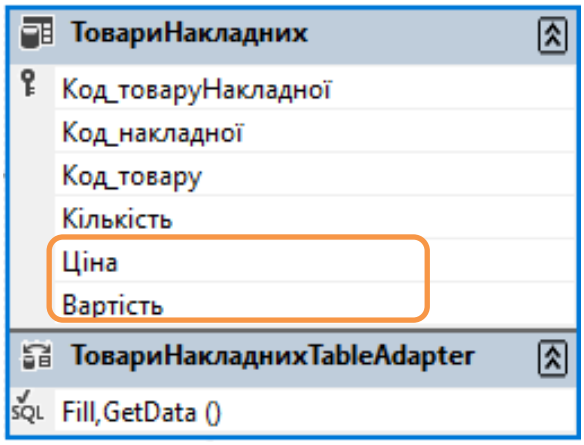


Рис. 4.19. **Локальна таблиця *ТовариНакладних***

Виконання

1. Перейдіть у вікно конструктора типізованого набору даних, двічі клацнувши значок вузла набору даних ***ХлібDataSet.xsd*** у вікні **Solution Explorer**.

2. Клацніть лінію зв'язку між таблицями ***Товари*** й ***ТовариНакладних*** у вікні конструктора. Потім у вікні властивостей уведіть нове значення ***ТовариТовариНакладних*** для властивості **Name**.

3. Клацніть правою клавішею мишки назву таблиці ***ТовариНакладних*** і виберіть із контекстового меню команду **Add – Column** та у вікні властивостей установіть значення, наведені в табл. 4.3.

Таблиця 4.3

Властивості нового стовпця

Властивості	Значення
Name	Ціна
DataType	System.Decimal
Expression	Parent(ТовариТовариНакладних).Ціна_закупівлі

4. Аналогічним чином додайте стовпець ***Вартість*** у таблицю ***ТовариНакладних*** типізованого набору даних, установивши у вікні властивостей значення, наведені в табл. 4.4.

Таблиця 4.4

Властивості стовпця Вартість

Властивості	Значення
Name	Вартість
DataType	System.Decimal
Expression	Parent(ТовариТовариНакладних).Ціна_закупівлі*Кількість

5. Закрийте вікно конструктора типізованого набору даних зі збереженням зроблених змін.

5.2. Створення форми

Завдання

Побудуйте форму ***Накладні*** для виконання CRUD-операцій із таблицями ***Накладні*** та ***ТовариНакладних*** (рис. 4.20).

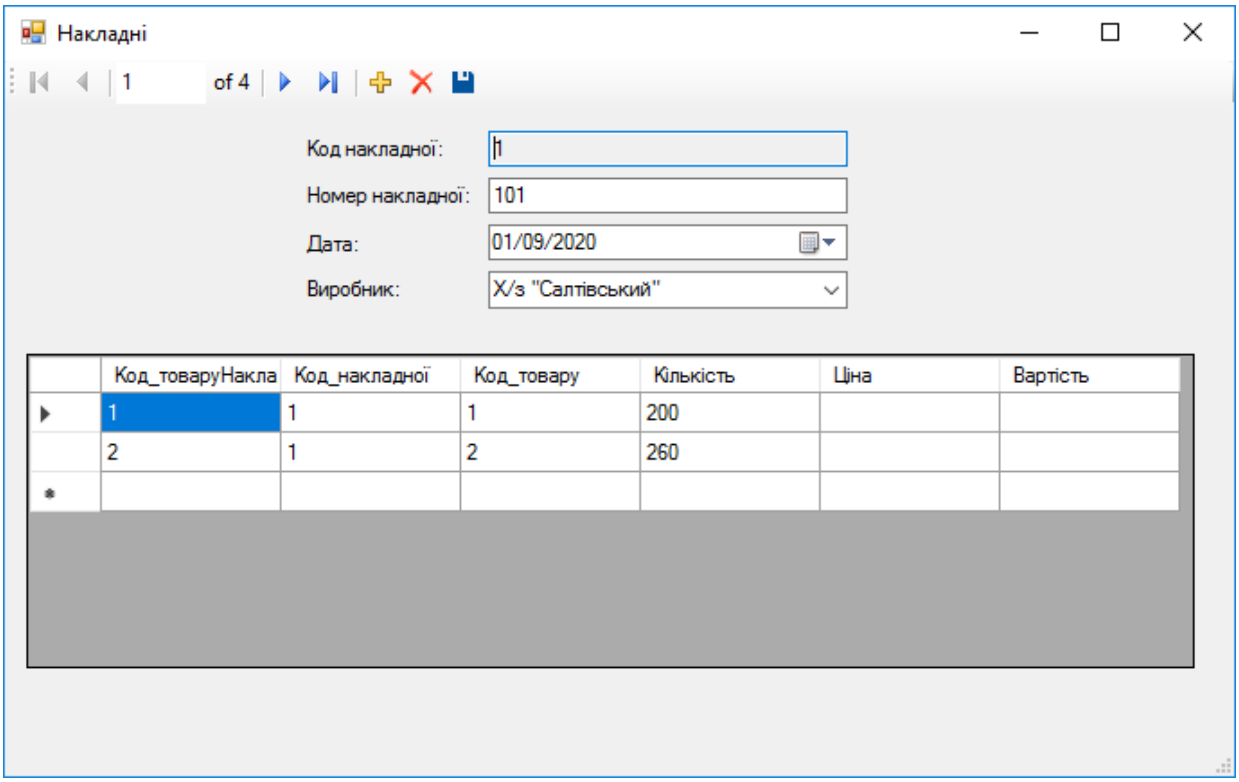


Рис. 4.20. **Початковий вигляд форми *Накладні***

Виконання

1. Додайте в застосунок нову форму, давши їй ім'я ***formНакладні*** та заголовок ***Накладні***.

2. Клацніть вузол таблиці ***Накладні*** у вікні **Data Sources** і з її списку, що розкривається, виберіть подання **Details** (**Таблиця**).

3. Розкрийте вузол таблиці ***Накладні*** й для стовпця ***Код_виробника*** виберіть у списку, що розкривається, подання **ComboBox**.

4. Перетягніть вузол таблиці ***Накладні*** з вікна **Data Sources** на форму ***Накладні***. На формі з'явилися елементи керування для відображення даних таблиці ***Накладні*** й панель навігатора, а в області компонентів – ряд об'єктів, які забезпечують роботу з даними таблиці в типізованому наборі даних.

5. Змініть властивість **Text** для напису ***Код_виробника***, установивши нове значення ***Виробник***. В елементі ***damaDateTimePicker*** установіть значення ***Short*** для властивості **Format**. Текстовому полю ***код_накладноїTextBox*** установіть значення ***True*** для властивості **ReadOnly**.

6. Щоб в елементі ComboBox для виробника відображалися назви виробників, перетягніть вузол таблиці ***Виробники*** з вікна **Data Sources** на цей ComboBox і відпустіть.

7. Розкрийте підвузол ***ТовариНакладних***, який перебуває у нижній частині вузла ***Накладні*** у вікні **Data Sources**. Потім змініть подання стовпця ***Код_товару*** на **ComboBox**.

8. Перетягніть підвузол таблиці ***ТовариНакладних***, який перебуває у нижній частині вузла ***Накладні***, із вікна **Data Sources** на форму ***Накладні***, помістивши його під наявними там елементами керування. На формі з'явився елемент керування DataGridView для відображення даних таблиці ***ТовариНакладних***, а в області компонентів – об'єкти BindingSource і TableAdapter, які забезпечують роботу з даними таблиці ***ТовариНакладних*** (рис. 4.21).

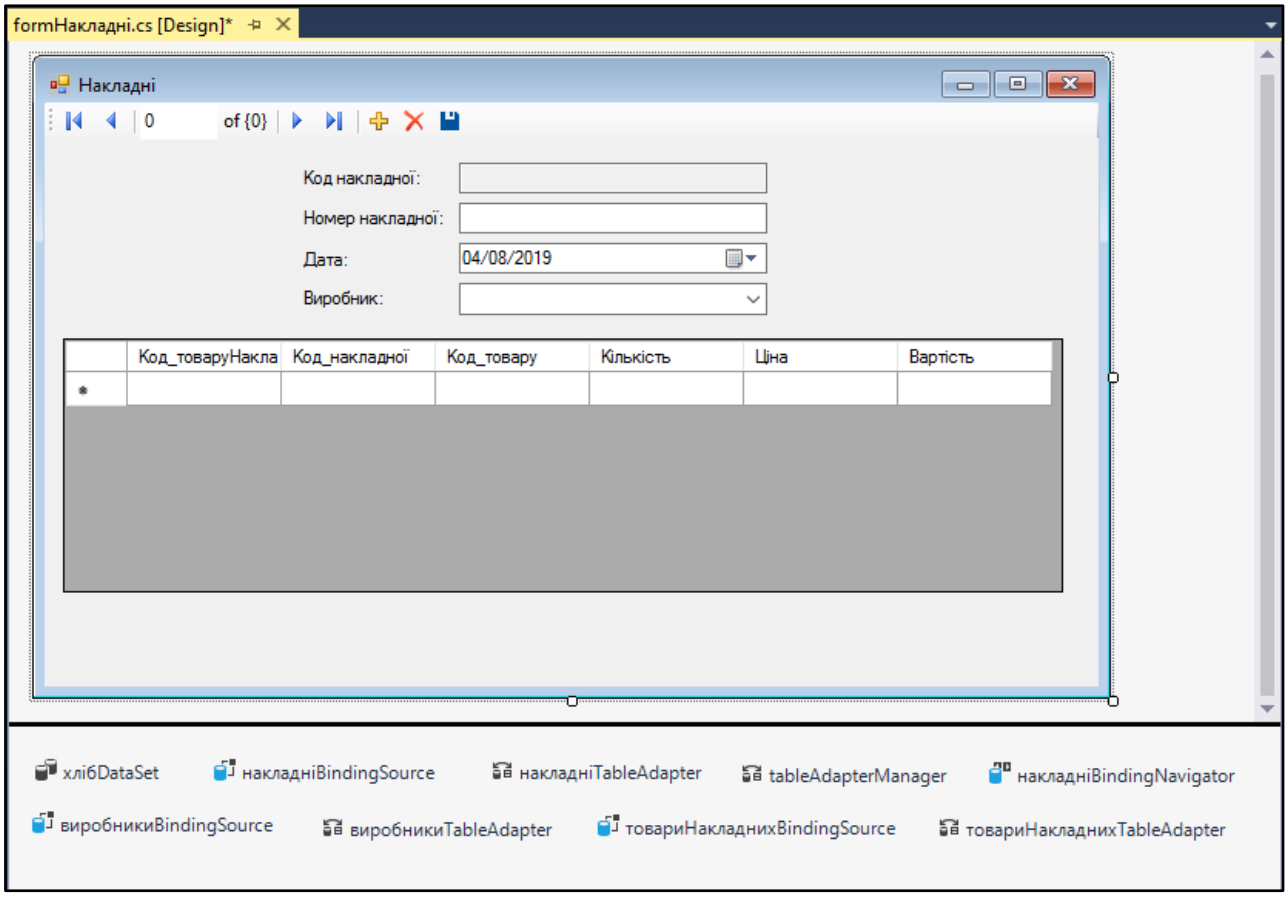


Рис. 4.21. **Вікно форми *Накладні* після налаштування загальної частини**

9. Перейдіть у вікно конструктора форми ***Хліб*** і додайте код оброблювача події "Клацання кнопки Накладні".

```
private void buttonНакладні_Click(object sender, EventArgs e)
        {
            formНакладні вікноНакладні = new formНакладні();
            вікноНакладні.ShowDialog();
        }
```

10. Запустіть програму на виконання й перевірте функціональність форми ***Накладні*** шляхом переміщення, зміни, додавання і видалення записів. Спочатку виконайте операції без збереження, а потім зі збереженням у базі даних зроблених змін. Стовпці ***Ціна*** і ***Вартість*** в елементі DataGridView є порожніми, тому що серед компонентів відсутні об'єкти для роботи з даними таблиці ***Товари***. Ці недоліки усувають в наступному завданні.

11. Якщо в елементі DataGridView відображаються всі записи таблиці ***ТовариНакладних***, а не тільки ті, що стосуються поточної накладної, завершіть виконання застосунку і перейдіть у вікно форми ***Накладні*** в режимі конструктора. Там клацніть елемент DataGridView і з його тегу налаштування виберіть зв'язок за зовнішнім ключем у полі зі списком **Choose Data Source** (рис. 4.22).

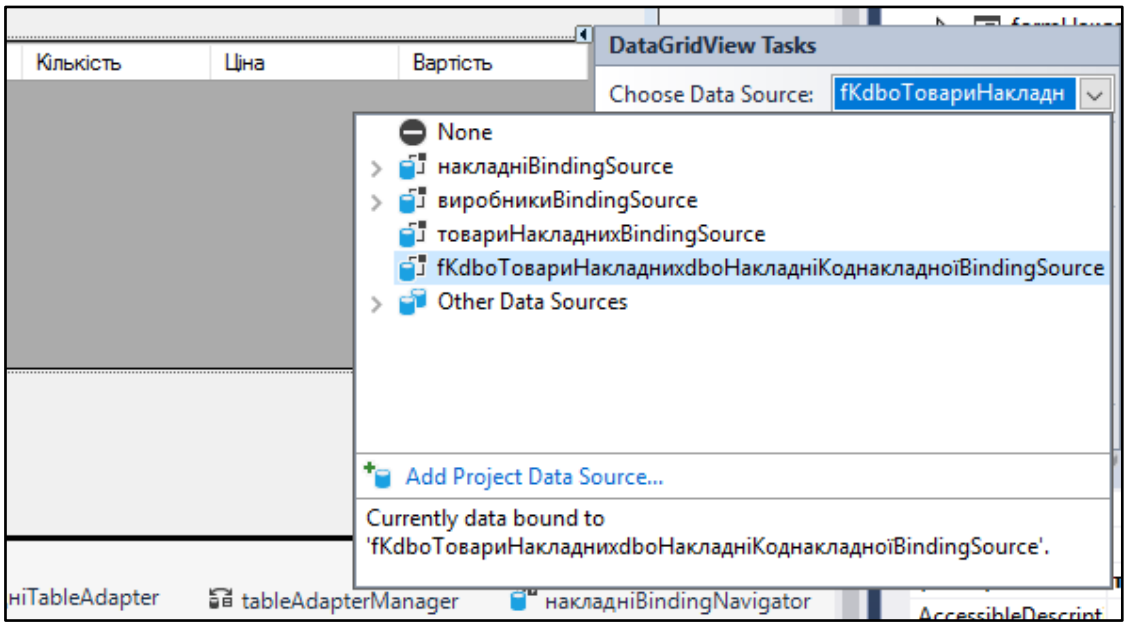


Рис. 4.22. **Вибір зв'язку як джерела даних**

12. Перевірте ще раз функціональність форми. В елементі DataGrid View мають відображатися тільки ті рядки з даними про товари, що належать до поточної накладної.

13. Збережіть зміни, зроблені у проєкті.

5.3. Налаштування елемента DataGridView

Завдання

Удоскональте елемент DataGridView на форму ***Накладні*** для більш зручної роботи з таблицею ***ТовариНакладних*** (рис. 4.23).

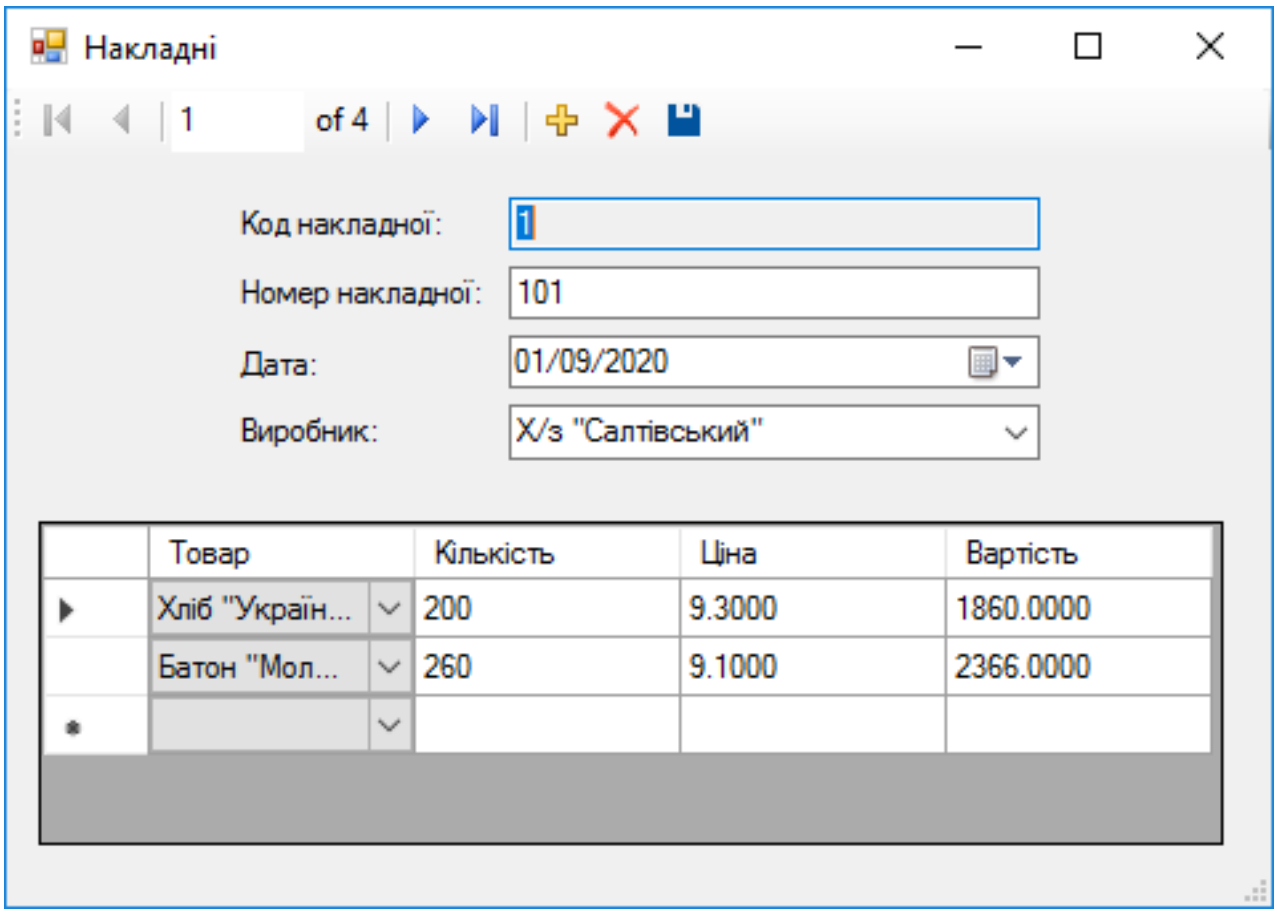


Рис. 4.23. **Удосконалений вигляд форми *Накладні***

Виконання

1. Щоб в елементі DataGridView відображалися дані, пов'язані з таблицею ***Товари*** (стовпці ***Товар***, ***Ціна*** та ***Вартість***), потрібно додати

об'єкти BindingSource і TableAdapter. Для цього перетягніть будь-який стовпець (наприклад, ***Товар***) із вузла ***Товари*** на вільне місце форми. Потім вилучіть із форми елемент керування, що з'явився на ній. Як результат в області компонентів з'являться компоненти **товариBindingSource** та **товариTableAdapter**.

2. Запустіть програму на виконання й перевірте функціональність форми ***Накладні***. Стовпець ***Код_товару*** відображено у вигляді елемента TextBox, а не ComboBox (рис. 4.24). Зупиніть виконання застосунку.

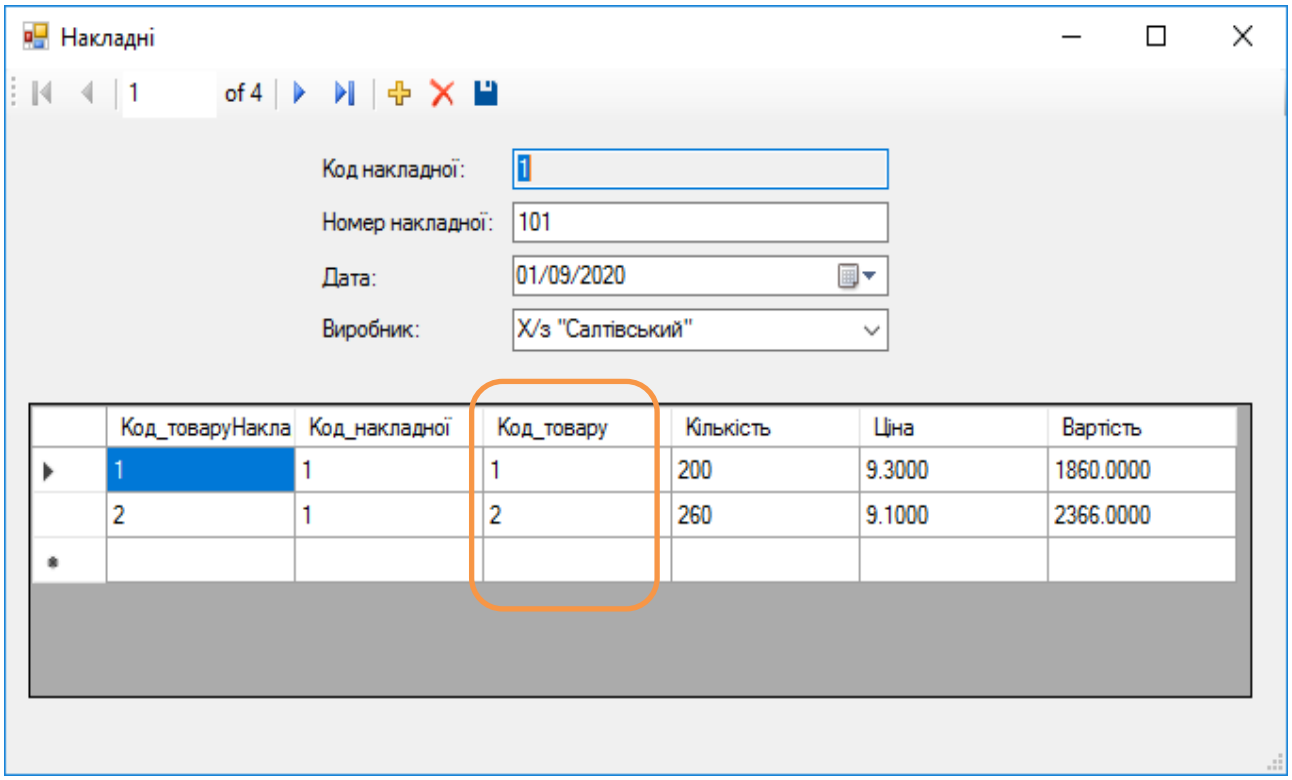


Рис. 4.24. **Стовпець *Код_товару* у вигляді елемента TextBox у вікні форми *Накладні***

3. Щоб стовпець ***Код_товару*** відображався у вигляді елемента ComboBox, виконайте таке:

3.1. Відкрийте вікно форми ***Накладні*** в режимі конструктора.

3.2. Виділіть елемент керування DataGridView.

3.3. Клацніть кнопку ... у властивості **Columns** вікна властивостей. З'явиться вікно **Edit Columns** (рис. 4.25).

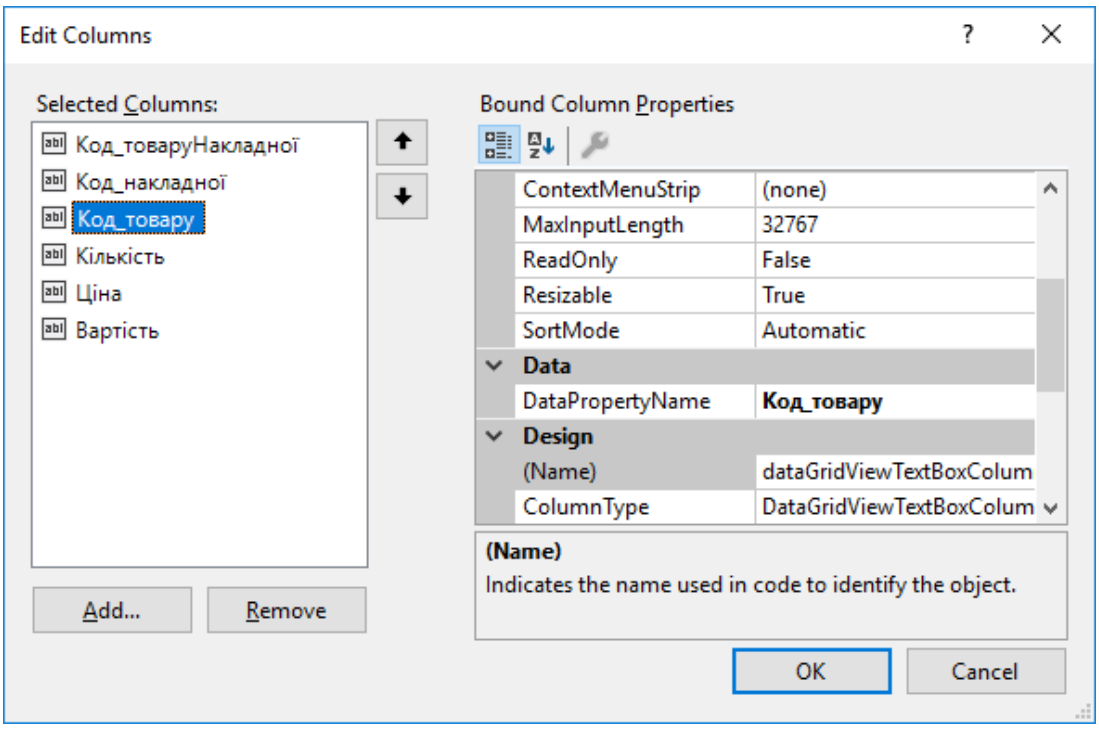


Рис. 4.25. **Вікно Edit Columns**

3.4. Виберіть елемент ***Код_товару*** у списку **Selected Columns** вікна **Edit Columns**, а в таблиці **Bound Column Properties** установіть значення властивостей, наведених у табл. 4.5.

Таблиця 4.5

Властивості елемента Код_товару

Властивості	Значення
ColumnType	DataGridViewComboBoxColumn
HeaderText	Товар
DataPropertyName	Код_товару
DataSource	товариBindingSource
DisplayMember	Товар
ValueMember	Код_товару

3.5. Клацніть кнопку **ОК**.

3.6. Запустіть програму на виконання й перевірте функціональність форми ***Накладні***. Стовпець ***Код_товару*** відображено у вигляді ComboBox. Бажано не відображати стовпці ***Код_товару_накладної*** та ***Код_накладної***, які відіграють службову роль і для кінцевого користувача не цікаві.

4. Зробіть невидимими стовпці ***Код_товару_накладної*** та ***Код_накладної*** в елементі керування DataGridView. Для цього виділіть елемент

керування DataGridView, у вікні властивостей клацніть кнопку ... у властивості **Columns**. З'явиться вікно **Edit Columns**. У цьому вікні установіть значення **False** для властивості **Visible** у стовпцях ***Код_товару_накладної*** та ***Код_накладної***.

5. Установіть потрібну ширину форми й елемента DataGridView (рис. 4.26).

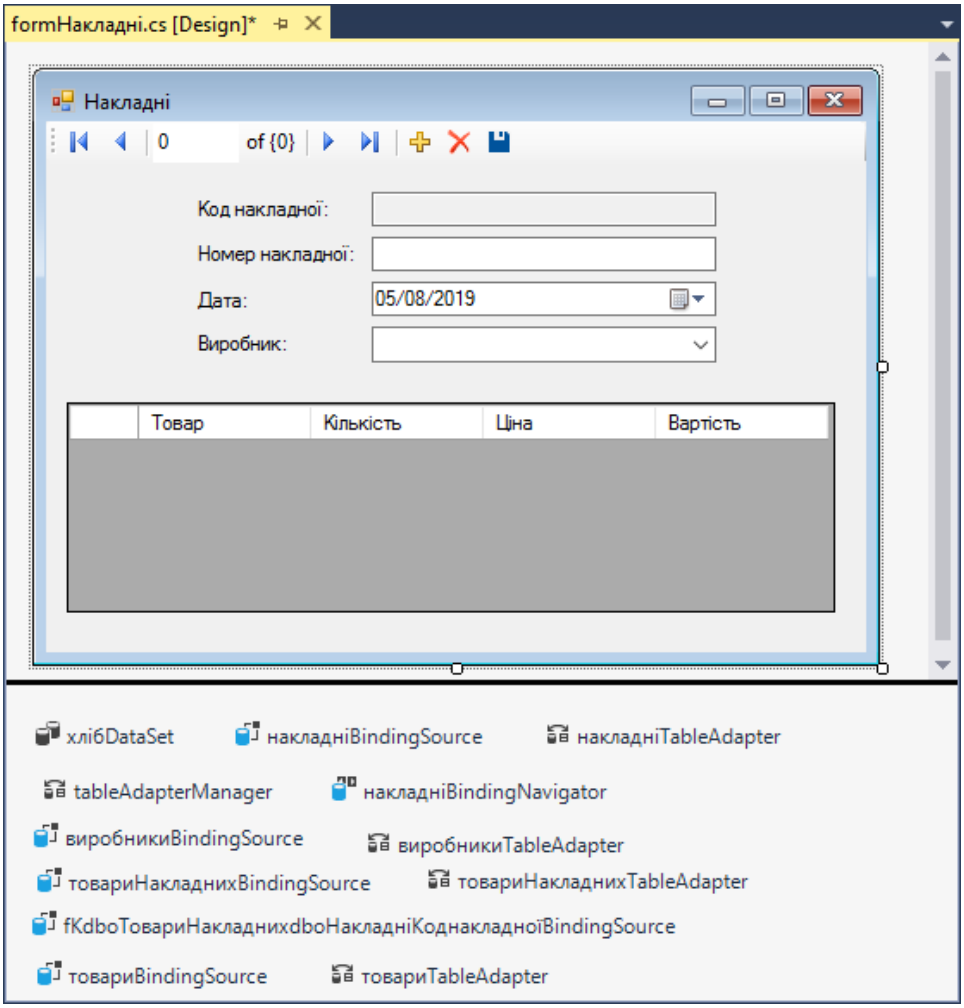


Рис. 4.26. **Вікно форми *Накладні* в режимі конструктора**

6. Запустіть програму на виконання й перевірте функціональність форми ***Накладні*** шляхом переміщення, зміни, додавання і видалення записів. Спочатку виконайте операції без збереження, а потім зі збереженням у базі даних зроблених змін. Усі зміни даних зберігають коректно, за винятком додавання нової накладної. Проблему пов'язано з автоінкрементними значеннями ключа таблиці ***Накладні*** та ієрархічно пов'язаними даними. Її вирішують за тим самим алгоритмом, що й у лабораторній роботі 2. У поточній роботі її винесено в завдання для самостійного виконання.

7. Закрийте форми ***Накладні*** та ***Хліб***.

У звіті з лабораторної роботи подайте скриншот форми ***Накладні*** в режимі конструктора разом з областю компонентів, а також код оброблювача ***formНакладні_Load***. Опишіть, як оператори коду оброблювача пов'язані з об'єктами, розміщеними в області компонентів.

6. Додавання адаптера таблиці з агрегованою функцією

Завдання

Додайте на форму ***Накладні*** загальну суму вартостей товарів за накладною (рис. 4.27).

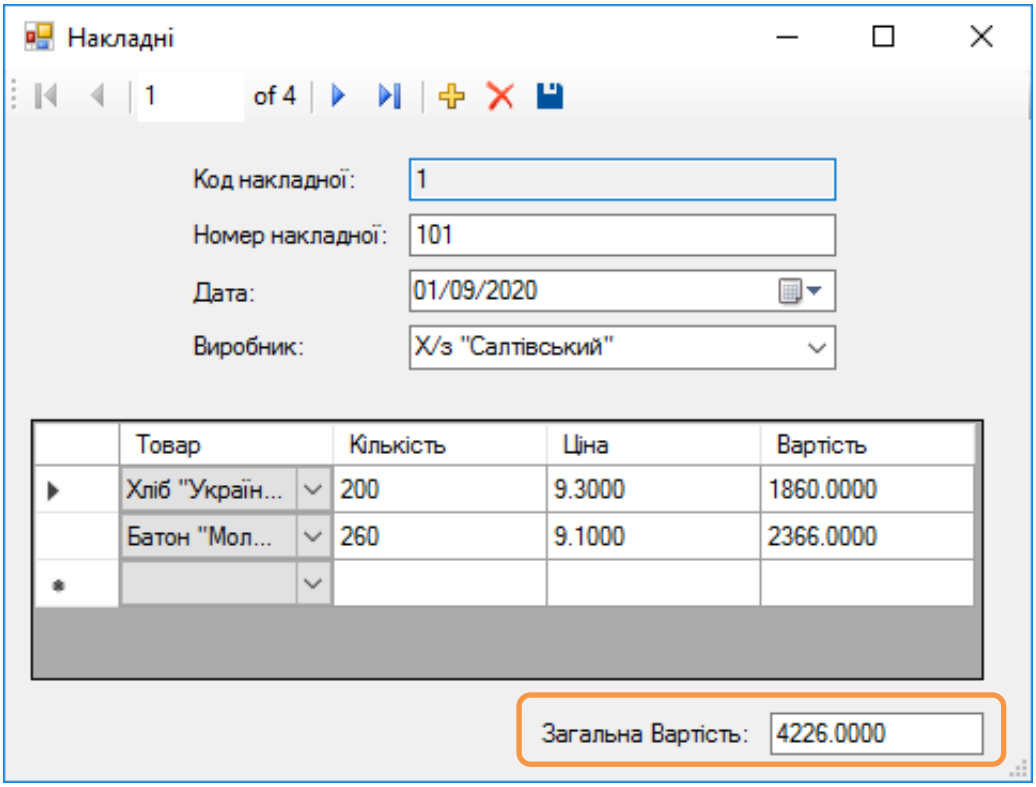


Рис. 4.27. **Поле *Загальна Вартість* на формі *Накладні***

Виконання

1. Додайте в типізований набір даних адаптер таблиці ***СумиНакладних***, який містить запит. У ньому є два стовпці – ***Код_накладної*** та ***ЗагальнаВартість***, що є сумою вартостей товарів за відповідною накладною. Для цього виконайте таке:

1.1. Перейдіть у вікно конструктора набору даних.

1.2. Клацніть правою клавішею мишки у вільному місці вікна конструктора й виберіть команду **Add – Table Adapter**.

1.3. У першому вікні майстра погодьтеся з попереднім рядком підключення, клацнувши кнопку **Next**.

1.4. У другому вікні майстра погодьтеся з тим, що адаптер буде використовувати інструкції SQL, клацнувши кнопку **Next**.

1.5. У третьому вікні майстра введіть інструкцію SQL

```
SELECT [ТовариНакладних].[Код_накладної],
    SUM(Товари.Ціна_закупівлі * [ТовариНакладних].Кількість) AS Загаль-
наВартість
FROM (Товари INNER JOIN [ТовариНакладних] ON
    Товари.[Код_товару] = [ТовариНакладних].[Код_товару])
GROUP BY [ТовариНакладних].[Код_накладної]
```

і клацніть кнопку **Next**.

Примітка. Інструкцію SQL можна побудувати за допомогою візуальних засобів Visual Studio, клацнувши кнопку **Query Builder** у цьому вікні.

1.6. У четвертому вікні майстра клацніть кнопку **Next**, погодившись з іменами методів, що створюють.

1.7. У п'ятому вікні майстра ознайомтеся зі списком виконаних дій і клацніть кнопку **Finish**. У вікні конструктора набору даних з'явилася нова таблиця з незручним ім'ям ***ТовариНакладних1***. Вона пов'язана з таблицею ***Накладні***.

1.8. Клацніть ім'я нової таблиці й у вікні властивостей уведіть більш зручне, наприклад, ***СумиНакладних***. На рис. 4.28 подано вікно конструктора набору даних з адаптером таблиці ***СумиНакладних***.

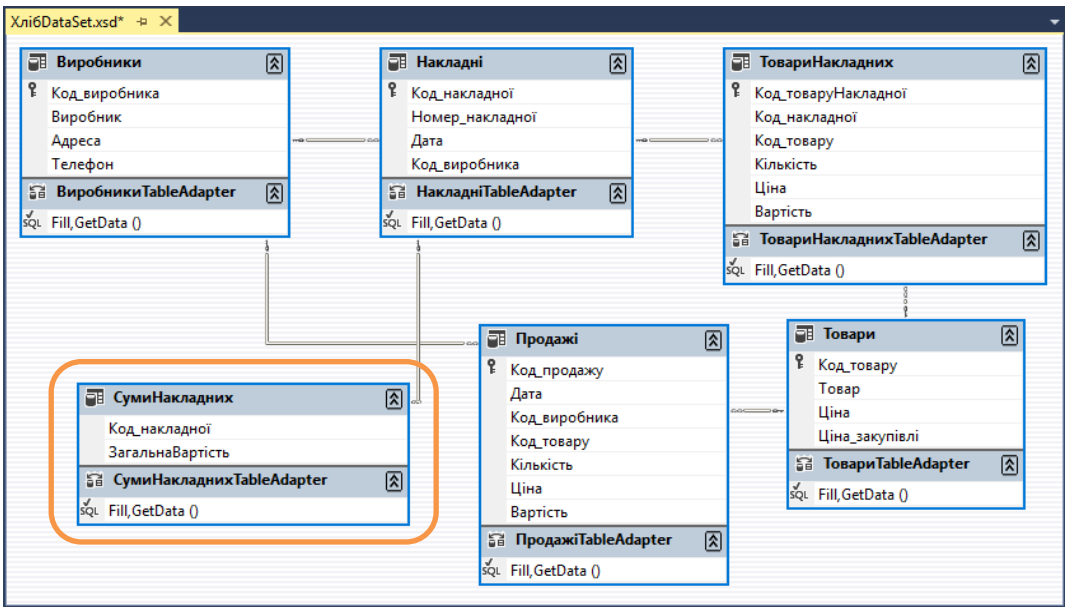


Рис. 4.28. **Вікно конструктора набору даних з адаптером таблиці *СумиНакладних***

1.9. Двічі клацніть на лінії зв'язку таблиць ***Накладні*** та ***СумиНакладних*** і у вікні **Relation** задайте для властивості **Name** значення ***Накладні-СумиНакладних***.

1.10. Закрийте вікно конструктора набору даних зі збереженням зроблених змін.

2. Перейдіть у вікно конструктора форми ***Накладні*** та збільште висоту форми приблизно на 1 см.

3. Розкрийте вузол таблиці ***Накладні*** у вікні **Data Sources** і в ньому – підвузол таблиці ***СумиНакладних***, потім перетягніть стовпець ***ЗагальнаВартість*** на форму ***Накладні***, помістивши його під елементом DataGrid View. Там з'явиться текстове поле з написом ***Загальна Вартість*** (рис. 4.29).

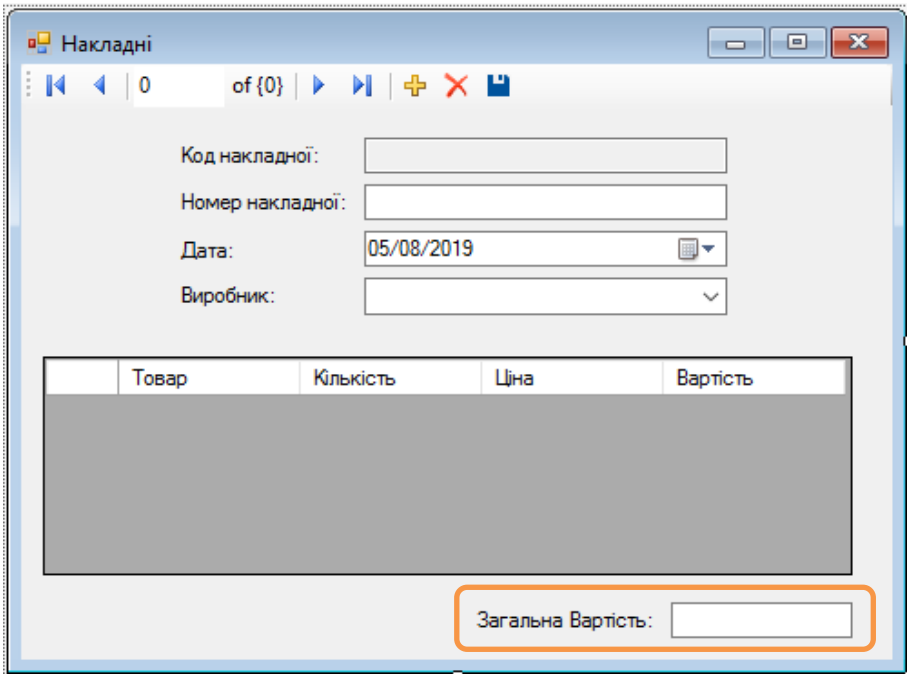


Рис. 4.29. **Вікно форми *Накладні* з полем *Загальна Вартість* у режимі конструктора**

4. Запустіть програму на виконання й перевірте функціональність форми ***Накладні*** шляхом переміщення записів. Простежте за зміною значень у полі ***Загальна Вартість***.

У звіті з лабораторної роботи подайте скриншот форми ***Накладні*** в режимі конструктора разом з областю компонентів. Поясніть, завдяки чому реалізують синхронізацію між поточною накладною, відображеною на формі, і значенням у текстовому полі ***Загальна Вартість***.

Завдання для самостійного виконання

1. Обновіть поле ***Код_продажу*** на формі ***Продажі*** після додавання нового запису.

2. Перетворіть форму ***Товари*** в "розділену", подібно до того, як показано на рис. 2.34 опису лабораторної роботи 2.

3. Додайте на форму ***Товари*** стовпці ***Отримано***, ***Продано*** й ***Залишок***. Для товарів кожного виду дані у цих стовпцях обчислюють так:

Продано – сума значень поля ***Кількість*** у таблиці ***Продажі***;

Отримано – сума значень поля ***Кількість*** у таблиці ***Товари Накладних***;

Залишок – різниця значень у стовпцях ***Отримано*** та ***Продано***.

4. Додайте на форму ***Виробники*** стовпець ***Внесок***. У ньому подають відсотковий внесок кожного виробника в загальну вартість проданих товарів.

5. Додайте на форму ***Продажі*** поле ***Прибуток***, яке обчислюють за формулою:

$$Прибуток = (Ціна - Ціна\_закупівлі) \times Кількість.$$

6. Додайте на форму ***Накладні*** середню ціну товарів за накладною.

7. Забезпечте збереження нової накладної разом із її товарами на формі ***Накладні***.

8. Відформатуйте елементи керування на формах, дотримуючись таких правил:

текстові дані притискають до лівого краю елемента, а числові – до правого;

усі числа в одному стовпці відображають з однаковою кількістю цифр у дробовій частині;

грошові величини (ціна, вартість тощо) відображають із двома цифрами у дробовій частині.

9. Виконайте завд. 1 – 6 ходу роботи для індивідуальної бази даних.

Лабораторна робота 5
Розробка програм із використанням технології LINQ to DataSet

Цілі лабораторної роботи:

1. Набуття практичних навичок у формуванні LINQ-запитів до DataSet.
2. Вироблення вмінь і навичок у групуванні даних та обчислення агрегованих величин у технології LINQ to DataSet.
3. Опанування операцій об'єднання даних кількох таблиць і створення підзапитів.
4. Набуття практичних навичок у побудові діаграм із використанням технології LINQ to DataSet.
5. Оволодіння основними підходами в розробленні застосунків для спеціалістів з аналізу даних.

Перед виконанням роботи студент має знати:

основи використання бібліотеки Windows Form;.
основи побудови запитів мовою SQL;
структуру DataSet і принципи роботи з його таблицями;
принципи прив'язування даних до елементів інтерфейсу.

Після виконання роботи студент має вміти:

самостійно розробляти прості застосунки для роботи з даними на основі технології LINQ to DataSet;
використовувати основні бібліотеки. Net Framework під час розроблення програм;
виконувати прості завдання аналізу даних;
подавати дані, що містяться в DataSet, у вигляді діаграм;
будувати інтерфейс користувача для спеціаліста з аналізу даних.

Підготовча частина

Хід роботи

1. Підготовка застосунку до використання технології LINQ to DataSet.
2. Сортування і фільтрація даних.
3. Групування даних. Обчислення агрегованих величин.

4. Об'єднання даних таблиць. Укладені запити.

5. Аналіз даних із діаграмною візуалізацією.

Форма звітності

За результатами виконання роботи необхідно оформити електронний звіт засобами Word. За кожним завданням слід зробити скриншоти з результатами виконання, а також подати код оброблювача події. У кінці завдань зазначено, що саме слід подати у звіті.

За результатами кожного виконаного завдання з п. "Завдання для самостійного виконання" слід подати постановку завдання, скриншот форми та відповідний програмний код.

Критерії оцінювання

У 12-бальній системі оцінку результату захисту лабораторної роботи формують за такими правилами:

1. За виконання кожного пункту практичної частини може бути виставлено від 0 до 1 бала.

2. За виконання завдань п. "Завдання для самостійного виконання" може бути виставлено:

від 0 до 1 бала за кожне завдання 1, 2;

від 0 до 2 балів за завдання 3;

від 0 до 3 балів за завдання 4;

від 0 до 1 бала за кожне завдання 5;

від 0 до 2 балів за завдання 6;

від 0 до 3 балів за кожне завдання 7;

на 1 бал більше, ніж за відповідні завдання на основі бази даних ***Хліб*** за завдання 8.

3. За кілька варіантів виконання одного із завдань додають 1 бал.

4. За вибір варіанта, який із кількох варіантів виконання є оптимальним, та обґрунтування вибору додають 1 бал.

5. За виконання кожного завдання в іншій СУБД (Oracle, MySQL, DB2, PostgreSQL тощо) додають по 1 балу.

6. За запізнення із захистом лабораторної роботи знімають 2 бали за кожний тиждень.

Набрану кількість балів із відповідей на кожне завдання лабораторної роботи підсумовують. У результаті такого підрахунку студент може набрати від 0 до 12 балів.

У разі запізнення із захистом лабораторної роботи понад три тижні студент виконує завдання роботи на основі індивідуальної бази даних. Роботу оцінюють за описаними раніше критеріями.

Практична частина

Постановка загального завдання

Вивчення засобів технології LINQ to DataSet здійснюють шляхом розширення застосунку ***типізованийХліб***, створеного в попередній лабораторній роботі. До нього додають функції аналізу даних на основі операцій сортування, фільтрування, групування та об'єднання. Результати подають у табличному вигляді та у вигляді діаграми.

Керування роботою застосунку здійснюють за допомогою групи кнопок ***Аналіз***, що додають на форму ***Хліб*** (рис. 5.1). Кнопки призначено для виклику відповідних функціональних форм.

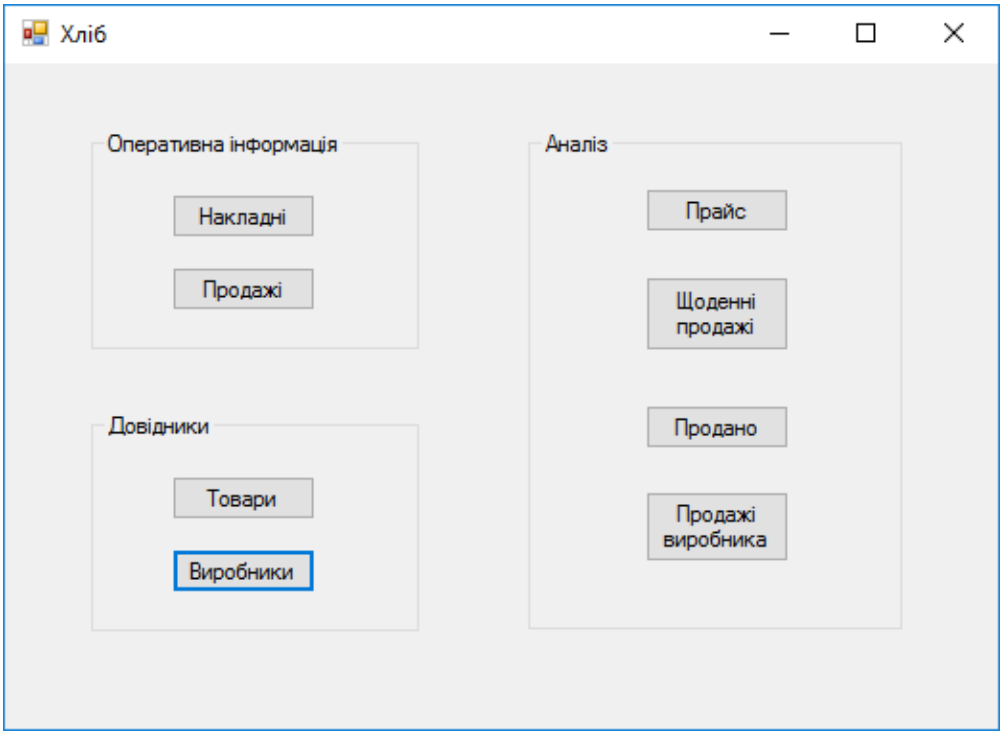


Рис. 5.1. **Кнопкова форма *Хліб* із групою кнопок *Аналіз***

На рис. 5.2 – 5.5 подано нові функціональні форми застосунку. Із їхньою допомогою можна переглядати дані, що містяться в таблицях бази даних ***Хліб***. Далі розгляньте призначення цих форм.

Форма ***Прайс*** слугує для перегляду цін товарів. Причому можна відфільтрувати дані про товари, задавши діапазон цін (див. рис. 5.2).

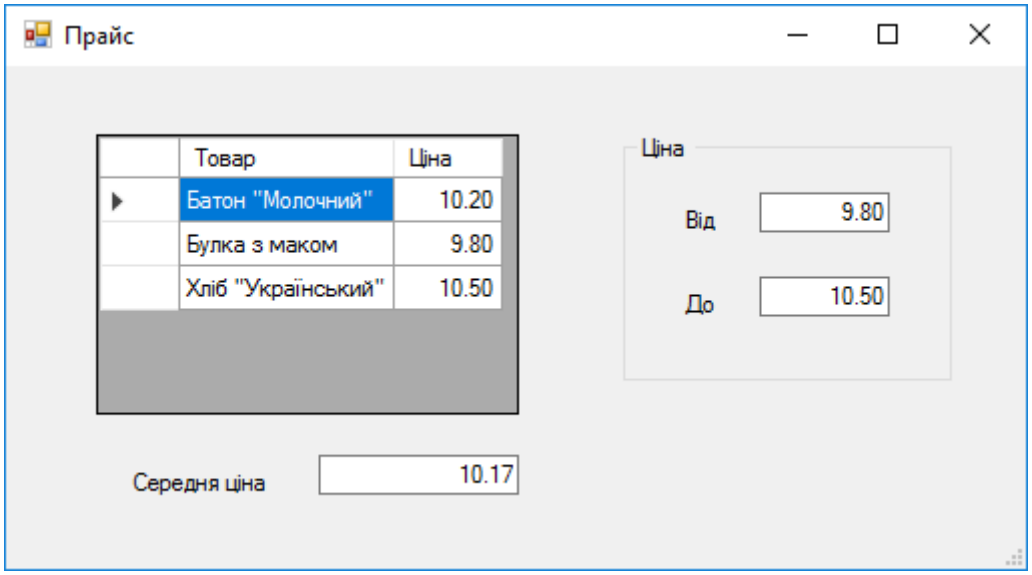


Рис. 5.2. **Форма *Прайс***

На формі ***Щоденні продажі*** подають кількість одиниць усіх товарів, проданих за відповідний день (див. рис. 5.3).

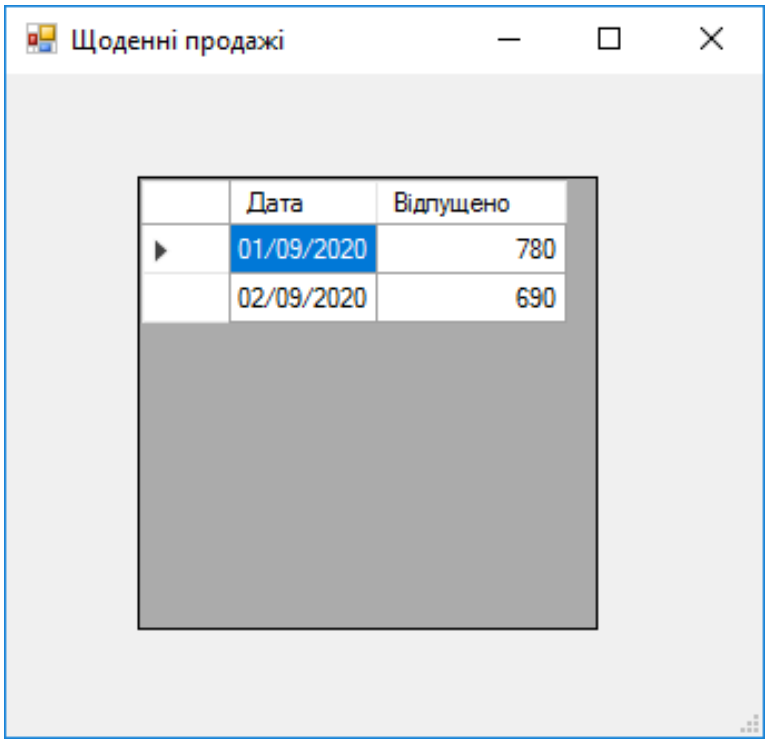


Рис. 5.3. **Форма *Щоденні продажі***

Форму ***Продано*** використовують для аналізу того, як продавали товари різних видів (див. рис. 5.4). Ліворуч подано список товарів, у яких величина ***Кількість*** більша за нуль, а праворуч – усіх товарів. В останньому видно, які товари зовсім не продавали (***Кількість = 0***).

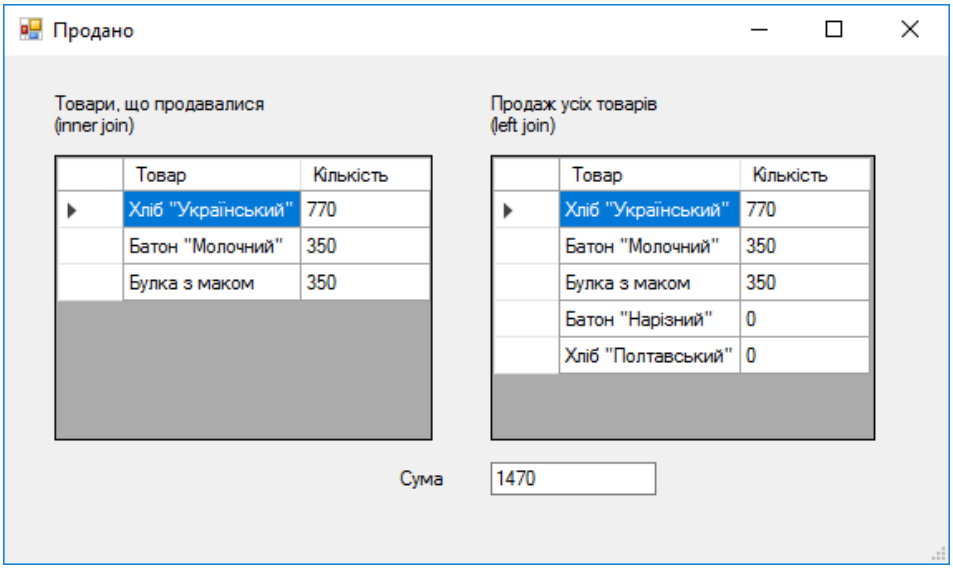


Рис. 5.4. **Форма *Продано***

Форму ***Продажі виробника*** застосовують для аналізу продажів щодо вибраного виробника. Тут, окрім табличного подання даних, використовують графічне. Із кругової діаграми можна дізнатися, яку частину займає кожний товар у загальній сумі продажів вибраного виробника (див. рис. 5.5).

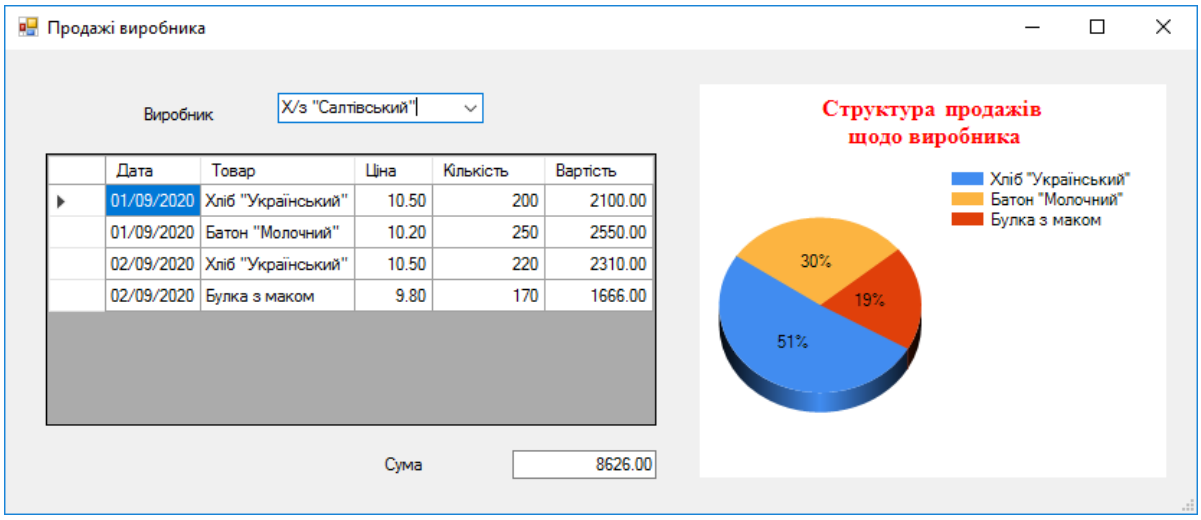


Рис. 5.5. **Форма *Продажі виробника***

1. Підготовка застосунку до використання технології LINQ to DataSet

Завдання

Скопіюйте застосунок для подальшої роботи й додайте на кнопкову форму в ньому групу кнопок ***Аналіз*** (рис. 5.6).

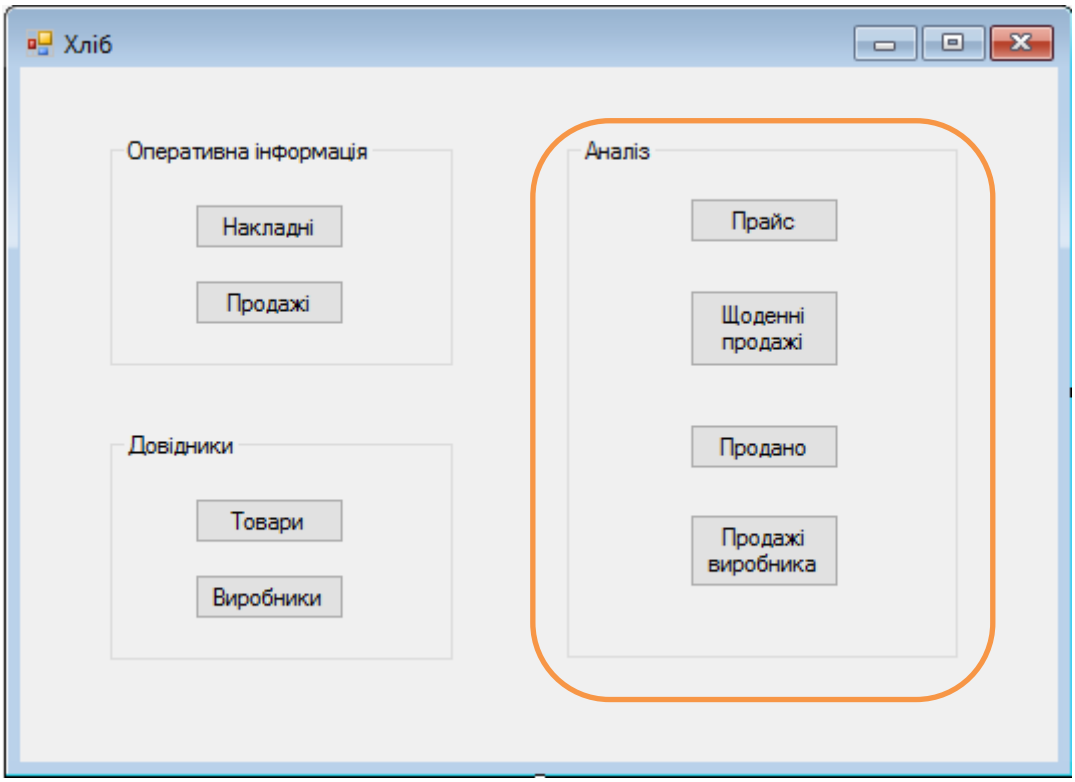


Рис. 5.6. **Група кнопок *Аналіз* на кнопковій формі *Хліб***

Виконання

1. Створіть нову папку з ім'ям ***linqХліб*** і скопіюйте в неї проєкт ***типізованийХліб***, створений у лабораторній роботі 4.

2. Відкрийте проєкт ***типізованийХліб***, розміщений у папці ***linqХліб***. Запустіть його на виконання і перевірте функціональність кнопок, що містяться на формі ***Хліб***. Потім зніміть застосунок із виконання.

3. Перейдіть у вікно конструктора форми ***Хліб***.

4. Додайте на форму елемент керування **GroupBox** і задайте значення ***Аналіз*** для його властивості **Text**.

5. Додайте чотири кнопки всередину елемента ***Аналіз*** та установіть для них значення властивостей, наведені в табл. 5.1.

Таблиця 5.1

Властивості нових кнопок

Кнопки	Властивості	Значення
1	Text	Прайс
	Name	buttonПрайс
2	Text	Щоденні продажі
	Name	buttonЩоденніПродажі
3	Text	Продано
	Name	buttonПродано
4	Text	Продажі виробника
	Name	buttonПродажіВиробника

2. Сортування і фільтрація даних (форма *Прайс*)

Ідеї та кроки виконання

Вивчення засобів сортування та фільтрації даних як елемента аналізу здійснюють на прикладі форми ***Прайс***. Вона слугує для перегляду цін на товари (рис. 5.7).

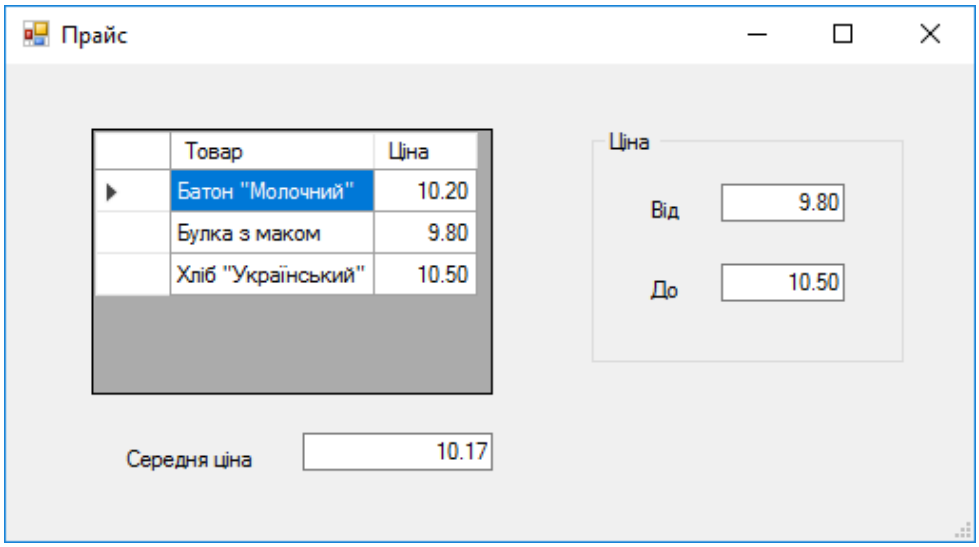


Рис. 5.7. **Форма *Прайс***

В елементі DataGridView відображають два стовпці даних із таблиці ***Товари***. Причому дані відсортовано за полем ***Товар***. Нижче в елементі TextBox відображено результат обчислення середньої ціни. Праворуч у групі ***Ціна*** подано мінімальну й максимальну ціну. Їх використовують як значення за замовчуванням, щоб під час завантаження форми відображали дані

про всі товари. Тут можна задати нові межі діапазону цін, відповідно до яких будуть відбирати товари в елементі DataGridView (фільтрація).

Розроблення форми ***Прайс*** складається з таких кроків:

1. Побудова форми.
2. Додавання функції фільтрування даних.

2.1. Побудова форми

Завдання

Додайте в застосунок форму ***Прайс***, у якій будуть відображати назви й ціни товарів (рис. 5.8).

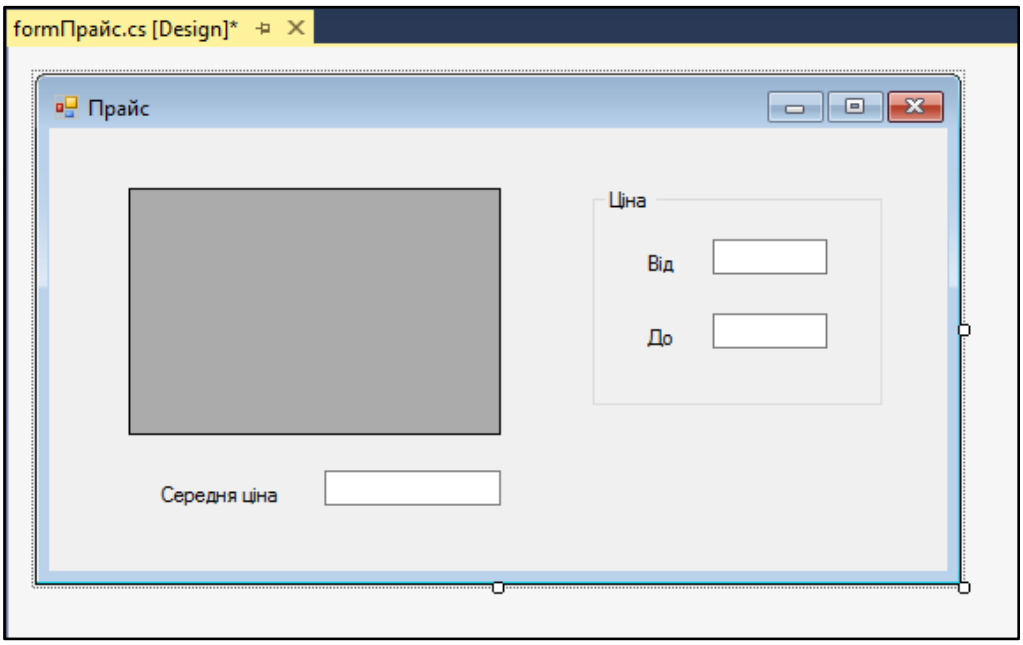


Рис. 5.8. **Форма *Прайс* у вікні конструктора**

Виконання

1. Додайте в застосунок нову форму з ім'ям файлу ***formПрайс.cs*** і значенням ***Прайс*** властивості **Text**.

2. Для відображення даних про товари додайте на форму елемент **DataGridView** зі значенням ***dataGridViewПрайс*** властивості **Name**.

3. Для подання середньої ціни товарів під елементом DataGridView розмістіть елементи **Label** і **TextBox**. Для властивості **Text** елемента **Label** задайте значення ***Середня ціна***, а для властивості **Name** елемента **TextBox** – значення ***textBoxСередняЦіна***.

4. Для встановлення меж діапазону цін праворуч від елемента DataGridView додайте елемент **GroupBox** зі значенням ***Ціна*** для властивості **Text**.

5. Усередині елемента GroupBox розмістіть два елементи **Label** зі значеннями ***Від*** і ***До*** для властивості **Text**, а праворуч – по одному елементу **TextBox** зі значеннями ***textBoxВід*** та ***textBoxДо*** для властивості **Name**, відповідно.

6. Двічі клацніть у вільному місці форми ***Прайс*** та уведіть оператори тіла оброблювача події ***formПрайс_Load***. Він має такий вигляд:

```
private void formПрайс_Load(object sender, EventArgs e)
{
    // Створюємо типізований набір даних
    ХлібDataSet хлібDataSet = new ХлібDataSet();

    // Заповнюємо таблицю Товари
    ХлібDataSetTableAdapters.ТовариTableAdapter
    товариTableAdapter =
        new ХлібDataSetTableAdapters.ТовариTableAdapter();
    товариTableAdapter.Fill(хлібDataSet.Товари);

    // Задаємо значення за замовчуванням
    if (textBoxВід.Text == "")
    {
        decimal Від =
            (from товар in хлібDataSet.Товари.AsEnumerable()
             select товар.Ціна).Min();
        textBoxВід.Text = Від.ToString("0.00");
        textBoxВід.TextAlign = HorizontalAlignment.Right;
    }

    if (textBoxДо.Text == "")
    {
        decimal До =
            (from товар in хлібDataSet.Товари.AsEnumerable()
             select товар.Ціна).Max();
        textBoxДо.Text = До.ToString("0.00");
        textBoxДо.TextAlign = HorizontalAlignment.Right;
    }

    // Готуємо дані для DataGridView і серед. ціни
    var query =
        from товар in хлібDataSet.Товари.AsEnumerable()
        let price = товар.Ціна
        where (price >= decimal.Parse(textBoxВід.Text)) &&
              (price <= decimal.Parse(textBoxДо.Text))
        orderby товар.Товар
        select new
        {
            Товар = товар.Товар,
            Ціна = товар.Ціна
        };
```

```
    // Відображаємо дані
    BindingSource bindingТовари = new BindingSource();
    bindingТовари.DataSource = query;
    dataGridViewПрайс.DataSource = bindingТовари;
    // Форматування виведених даних
    dataGridViewПрайс.AutoResizeColumns();
    dataGridViewПрайс.Columns["Ціна"].DefaultCellStyle.Format="0.00";
    dataGridViewПрайс.Columns["Ціна"].DefaultCellStyle.Alignment=
        DataGridViewContentAlignment.MiddleRight;
    // Середня ціна
    decimal Середня_ціна = query.Average(t => t.Ціна);
    textBoxСередняЦіна.Text = Середня_ціна.ToString("0.00");
    textBoxСередняЦіна.TextAlign = HorizontalAlignment.Right;
}
```

7. Перейдіть у вікно конструктора форми ***Хліб***, двічі клацніть кнопку ***Прайс*** і додайте код оброблювача події "Клацання на кнопці Прайс".

```
private void buttonПрайс_Click(object sender, EventArgs e)
{
    formПрайс вікноПрайс = new formПрайс();
    вікноПрайс.ShowDialog();
}
```

8. Перевірте функціональність форми ***Прайс***.

9. Збережіть зміни, зроблені у проєкті.

2.2. Додавання функції фільтрування даних

Завдання

Додайте функцію фільтрування даних у разі зміни значень в елементах ***Від*** і ***До***.

Ідеї виконання

Оскільки оброблення події "Зміна значення" призведе до багаторазового виклику оброблювача після додавання кожного символу, більш раціонально обробляти подію "Втрата фокуса". Вона виникає тільки один раз після закінчення зміни значення в елементі TextBox. Як оброблювач можна використати наявний оброблювач події "Завантаження форми" (функція formПрайс_Load).

Виконання

1. Перейдіть у вікно конструктора форми ***Прайс*** і клацніть поле ***Від*** (елемент ***textBoxВід***).

2. У вікні властивостей відобразіть перелік подій, клацнувши кнопку **Events**  .

3. Установіть значення ***formПрайс_Load*** для властивості **Leave**, вибравши його зі списку, що розкривається.

4. Повторіть п. 1 – 3 для поля ***До*** (елемент ***textBoxДо***).

5. Збережіть зміни, зроблені у проєкті.

6. Запустіть програму на виконання й перевірте функціональність полів ***Від*** і ***До***.

У звіті з лабораторної роботи подайте скриншот форми ***Прайс*** у режимі виконання, а також код оброблювача ***formПрайс_Load***. Поясніть роботу кожного запиту LINQ.

3. Групування даних. Обчислення агрегованих величин (форма *Щоденні продажі*)

Завдання

Додайте в застосунок форму ***Щоденні продажі***, у якій будуть відображати дати й кількість одиниць товарів, проданих у відповідний день (рис. 5.9).

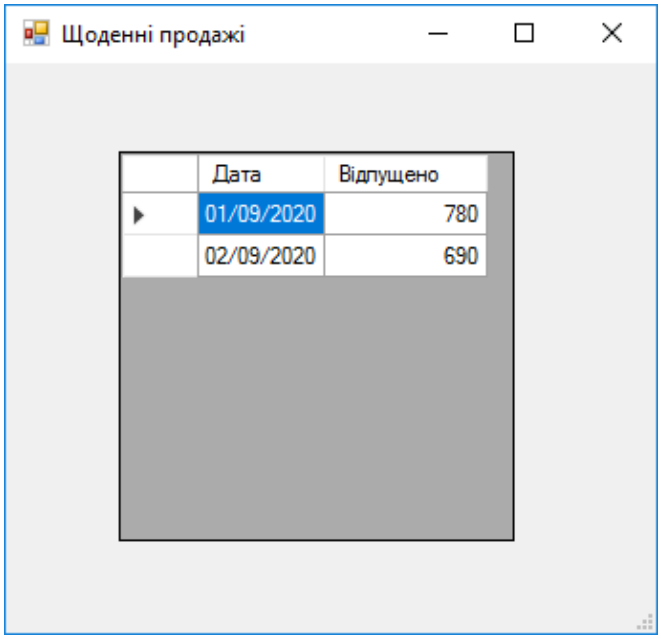


Рис. 5.9. **Форма *Щоденні продажі***

Ідеї виконання

Вивчення засобів відбору даних у групи за значеннями певного поля і виконання обчислень у кожній групі здійснюють на прикладі форми ***Щоденні продажі***. Вона слугує для перегляду кількості одиниць усіх товарів, проданих за відповідний день.

В елементі DataGridView відображають два стовпці даних із таблиці ***Продажі*** (див. рис. 5.9). У першому стовпці відображають дати, у які здійснювали продаж товарів, а в другому – кількість одиниць товарів, відпущених у відповідний день. Для отримання таких даних будують запит LINQ. У ньому результати продажів збирають у групи за значеннями поля ***Дата*** і в кожній групі виконують обчислення суми значень поля ***Кількість***.

Виконання

1. Додайте в застосунок нову форму з ім'ям файлу ***formЩоденніПродажі.cs*** і значенням ***Щоденні продажі*** для властивості **Text**.

2. Для відображення даних про товари додайте на форму елемент **DataGridView** зі значенням ***dataGridViewЩоденніПродажі*** властивості **Name**.

3. Двічі клацніть у вільному місці форми ***Щоденні продажі*** й уведіть оператори тіла оброблювача події ***formЩоденніПродажі_Load***. Оброблювач має такий вигляд:

```
private void formЩоденніПродажі_Load(object sender, EventArgs e)
{
    // Створюємо типізований набір даних
    Хліб1DataSet хліб1DataSet = new Хліб1DataSet();
    // Заповнюємо таблицю Продажі
    Хліб1DataSetTableAdapters.ПродажіTableAdapter
        продажіTableAdapter =
        new Хліб1DataSetTableAdapters.ПродажіTableAdapter();
    продажіTableAdapter.Fill(хліб1DataSet.Продажі);
    // Готуємо дані для DataGridView
    var query =
        from продаж in хліб1DataSet.Продажі.AsEnumerable()
        group продаж by продаж.Дата into Дні
        select new
        {
            Дата = Дні.Key,
            Відпущено = (from день in Дні
            select (int)день.Кількість).Sum()
        };
    // Відображаємо дані
    BindingSource bindingПродажі = new BindingSource();
    bindingПродажі.DataSource = query;
    dataGridViewЩоденніПродажі.DataSource = bindingПродажі;
    dataGridViewЩоденніПродажі.AutoResizeColumns();
    dataGridViewЩоденніПродажі.Columns["Відпущено"].
        DefaultCellStyle.Alignment =
        DataGridViewContentAlignment.MiddleRight;
}
```

4. Перейдіть у вікно конструктора форми ***Хліб***, двічі клацніть кнопку ***Щоденні продажі*** й додайте код оброблювача події "Клацання на кнопці Щоденні продажі".

```
private void buttonЩоденніПродажі_Click(object sender, EventArgs e)
{
    formЩоденніПродажі вікноЩоденніПродажі= new
        formЩоденніПродажі();
    вікноЩоденніПродажі.ShowDialog();
}
```

5. Збережіть зміни, зроблені у проєкті.

6. Перевірте функціональність форми ***Щоденні продажі***.

У звіті з лабораторної роботи подайте скриншот форми ***Щоденні продажі*** в режимі виконання, а також код оброблювача ***formЩоденніПродажі_Load***. Поясніть призначення кожного речення запиту LINQ.

4. Об'єднання даних таблиць. Укладені запити (форма *Продано*)

Завдання

Додайте в застосунок форму ***Продано***. У ній будуть відображати дані про кількості продажів тільки тих товарів, які продавали, а також усіх товарів, якими можуть постачати кіоск (рис. 5.10).

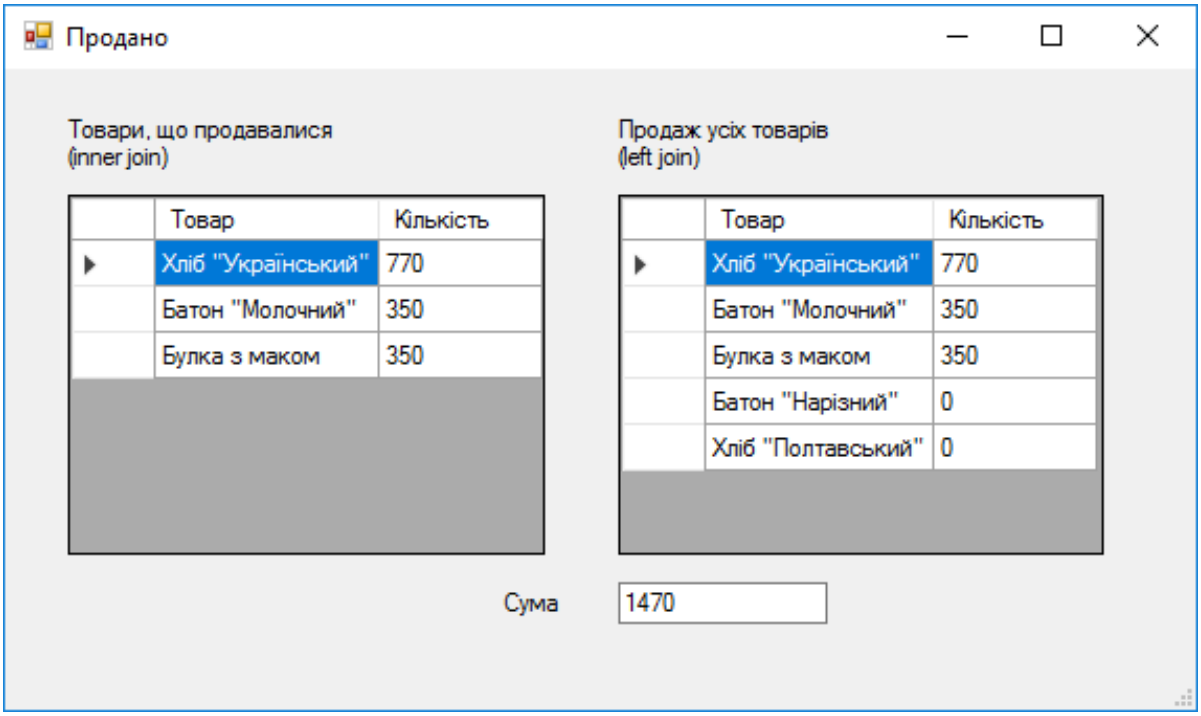


Рис. 5.10. **Форма *Продано***

Ідеї виконання

Спочатку слід створити перший запит (підсумковий), у якому обчислюють кількість проданих товарів кожного виду. Його будують на основі даних таблиці ***Продажі***. Цей запит складається із двох полів – ***Код_товару*** й ***Кількість***.

Щоб на формі відображати не коди товарів, а їхні назви, треба побудувати другий запит. У ньому об'єднують дані таблиці ***Товари*** й результати першого запиту за полем ***Код_товару***. Із таблиці ***Товари*** беруть поле ***Товар***, а з першого запиту – ***Кількість***.

Об'єднання слід зробити двома способами. У першому способі в результат другого запиту потрапляють тільки ті записи, у яких збігаються значення полів ***Код_товару***. Тому стануть відомі результати продажів тих товарів, про які є дані в таблиці ***Продажі***. Якщо ж за якимись товарами не було продажів, вони не потраплять у підсумковий список. Тобто реалізують внутрішнє об'єднання таблиць (реляційна операція inner join).

У другому способі об'єднання в результат другого запиту потрапляють усі записи з таблиці ***Товари***, навіть ті, які не продавали (реляційна операція left join).

Для обчислення загальної суми кількості проданих товарів достатньо результатів першого запиту.

Виконання

1. Додайте в застосунок нову форму з ім'ям файлу ***formПродано.cs*** і значенням ***Продано*** властивості **Text**.

2. Для відображення даних про кількість проданих товарів додайте на форму два елементи **DataGridView**. Першому дайте ім'я ***dataGridViewПродано***, а другому – ***dataGridViewПродано2***.

3. Над кожним елементом DataGridView помістіть напис (елемент **Label**). У властивість **Text** першого напису введіть значення ***Товари, що продавали (inner join),*** а другого – ***Продаж усіх товарів (left join)***.

4. Для подання загальної суми кількості проданих товарів під елементами DataGridView розмістіть елементи **Label** і **TextBox**. Для властивості **Text** елемента **Label** задайте значення ***Сума***, а для властивості ***Name*** елемента **TextBox** – ***textBoxСума***.

5. Двічі клацніть у вільному місці форми ***Продано*** й уведіть оператори тіла оброблювача події ***formПродано_Load***. Оброблювач має такий вигляд:

```
private void formПродано_Load(object sender, EventArgs e)
{
    // Створюємо типізований набір даних
    ХлібDataSet хлібDataSet = new ХлібDataSet();

    // Заповнюємо таблицю Товари
    ХлібDataSetTableAdapters.ТовариTableAdapter товариTableAdapter
        =new ХлібDataSetTableAdapters.ТовариTableAdapter();
    товариTableAdapter.Fill(хлібDataSet.Товари);

    // Заповнюємо таблицю Продажі
    ХлібDataSetTableAdapters.ПродажіTableAdapter продажіTableAdapter
        = new ХлібDataSetTableAdapters.ПродажіTableAdapter();
    продажіTableAdapter.Fill(хлібDataSet.Продажі);

    // Обчислюємо сумарну кількість продажів
    // по кожному виду товарів (через Код_товару)

    var продажі = from продаж in хлібDataSet.Продажі.AsEnumerable()
       group продаж by продаж.Код_товару into g
       select new
       {
           Код_товару = g.Key,
           Кількість = g.Sum(p => p.Кількість)
       };

    // Додаємо назви товарів для кожної кількості (inner join)
    var query = from товар in хлібDataSet.Товари.AsEnumerable()
    join продаж in продажі
        on товар.Код_товару equals продаж.Код_товару
        select new
        {
            Товар = товар.Товар,
            Кількість = продаж.Кількість
        };

    // Відображаємо дані
    BindingSource bindingТовари = new BindingSource();
    bindingТовари.DataSource = query;
```

```
    dataGridViewПродано.DataSource = bindingТовари;
    dataGridViewПродано.AutoResizeColumns();

    // Додаємо назви товарів для кожної кількості (left join)
    var query2 = from товар in хліб DataSet.Товари.AsEnumerable()
         join продаж in продажі
         on товар.Код_товару equals продаж.Код_товару into pg
        from t in pg.DefaultIfEmpty(new
            { Код_товару = 0, Кількість= 0 })
         select new
         {
             Товар = товар.Товар,
             Кількість = t.Кількість
         };

    // Відображаємо дані
    BindingSource bindingТовари2 = new BindingSource();
    bindingТовари2.DataSource = query2;
    dataGridViewПродано2.DataSource = bindingТовари2;
    dataGridViewПродано2.AutoResizeColumns();

    //Сума
    int Сума = продажі.Sum(p => p.Кількість);
    textBoxСума.Text = Сума.ToString();
}
```

6. Перейдіть у вікно конструктора форми ***Хліб***, двічі клацніть кнопку ***Продано*** і у вікні коду додайте код оброблювача події "Клацання на кнопці Продано".

```
private void buttonПродано_Click(object sender, EventArgs e)
{
    formПродано вікноПродано = new formПродано();
    вікноПродано.ShowDialog();
}
```

7. Збережіть зміни, зроблені у проєкті.

8. Перевірте функціональність форми ***Продано***. Для цього:

8.1. Запустіть на виконання застосунок.

8.2. Викличте форму ***Товари***, уведіть дані про два нові товари і збережіть їх у базі даних. Потім закрийте форму ***Товари***.

8.3. Викличте форму ***Продано*** і переконайтеся в тому, що у правому списку з'явилися назви доданих товарів із нульовими значеннями у стовпці ***Кількість***.

У звіті з лабораторної роботи подайте скриншот форми ***Продано*** в режимі виконання, а також код оброблювача ***formПродано_Load***. Поясніть роботу кожного запиту LINQ.

5. Аналіз даних із діаграмною візуалізацією (форма *Продажі виробника*)

Ідеї та кроки виконання

Вивчення запитів для інтерактивного аналізу даних, у яких відображають дані про вибраного виробника, здійснюють на прикладі форми ***Продажі виробника***. У ній відображають дані в табличній формі й у вигляді об'ємної кругової діаграми (рис. 5.11).

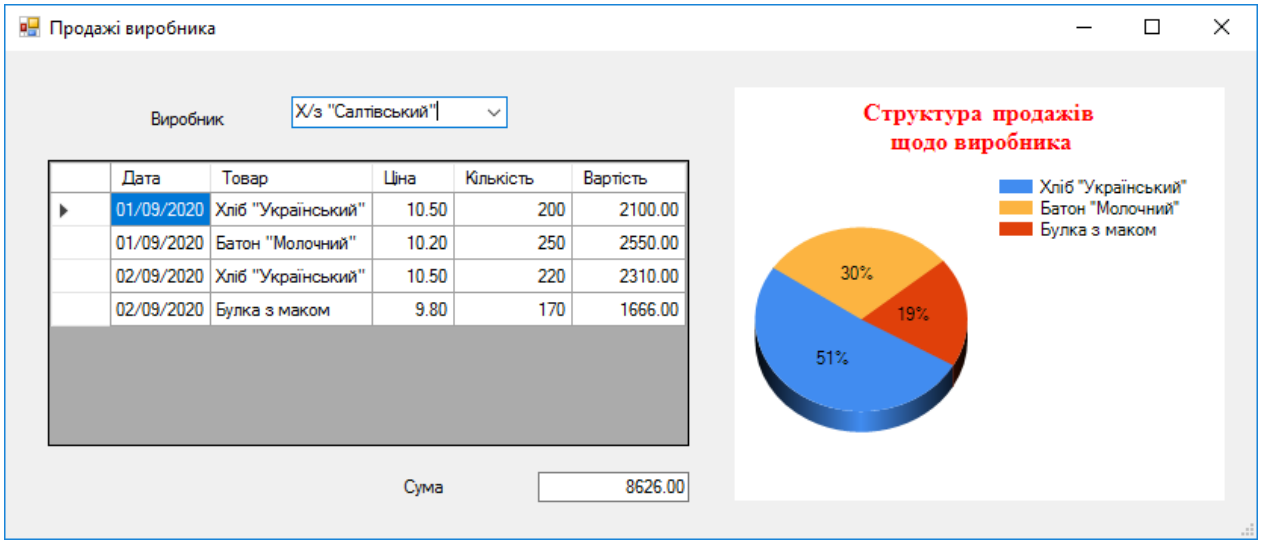


Рис. 5.11. **Форма *Продажі виробника***

Інтерактивність аналізу даних здійснюють за таким сценарієм. Спочатку відображають текстові та графічні дані для першого виробника. Після вибору в полі зі списком ***Виробник*** іншого виробника відбувається зміна детальних даних у таблиці, підсумкових – у текстовому полі ***Сума*** та графічних даних на круговій діаграмі.

Розроблення форми слід здійснити в такому порядку. Спочатку потрібно створити поле зі списком ***Виробник***. Вибране в ньому значення

використовують як умову відбору даних у запиті. Його дані будуть відображати в елементах DataGridView і Chart. На основі цього запиту також обчислюють загальну суму вартостей проданих товарів вибраного виробника.

Для забезпечення інтерактивності обробляють подію **SelectedIndexChanged** для поля зі списком ***Виробник***.

Після введення коду обох оброблювачів подій слід протестувати форму, спостерігаючи за синхронним відображенням даних в елементах DataGridView та Chart

Отже, розроблення форми ***Продажі виробника*** складається з таких кроків:

1. Створення форми для відображення текстових даних.
2. Забезпечення інтерактивності форми.
3. Побудова діаграми.

5.1. Створення форми для відображення текстових даних

Завдання

Додайте в застосунок форму ***Продажі виробника***. У ній будуть відображати текстові дані про продажі товарів вибраного виробника (рис. 5.12).

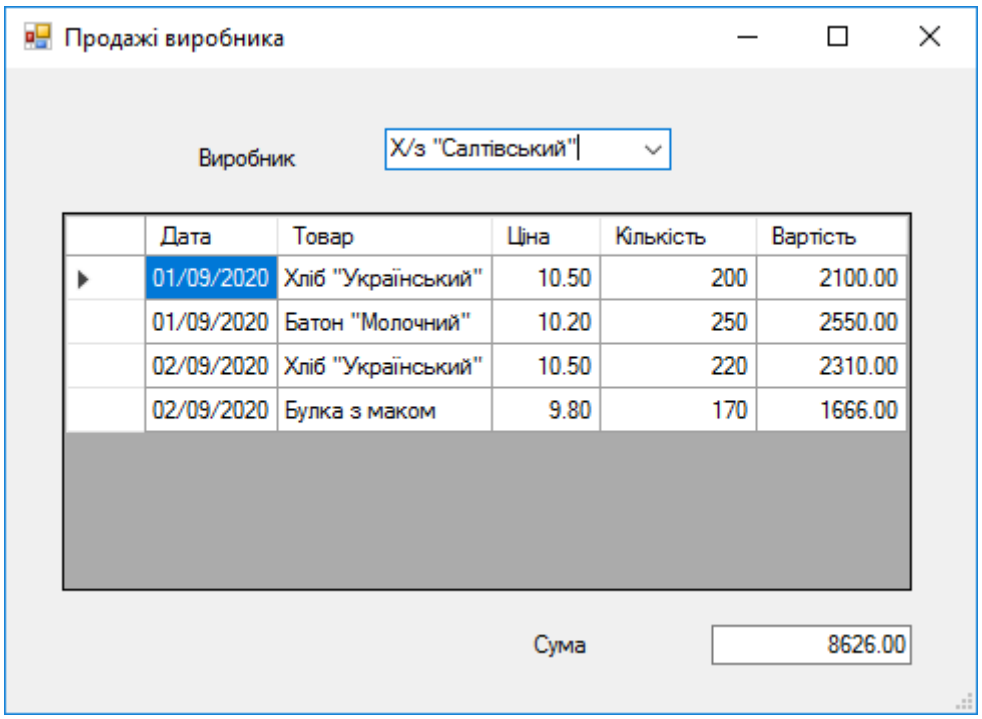


Рис. 5.12. **Форма *Продажі виробника* з текстовими даними**

Виконання

1. Додайте в застосунок нову форму з ім'ям файлу ***formПродажіВиробника.cs*** і значенням ***Продажі виробника*** властивості **Text**.

2. Для відображення списку виробників додайте на форму елемент ComboBox, а ліворуч від нього – елемент Label. Для властивості **Text** елемента **Label** задайте значення ***Виробник***, а для властивості **Name** елемента **ComboBox** – значення ***comboBoxВиробник***.

3. Для відображення результатів продажів про товари вибраного виробника додайте на форму елемент **DataGridView** з ім'ям ***dataGridViewПродажіВиробника***.

4. Для подання загальної суми вартостей проданих товарів вибраного виробника під елементом DataGridView розмістіть елементи **Label** і **TextBox**. Для властивості **Text** елемента **Label** задайте значення ***Сума***, а для властивості **Name** елемента **TextBox** – значення ***textBoxСума***.

5. Двічі клацніть у вільному місці форми ***Продажі виробника*** й уведіть оператори тіла оброблювача події ***formПродажіВиробника_Load***. Він має такий вигляд:

```
// Спільні об'єкти
ХлібDataSet хлібDataSet = new ХлібDataSet();
BindingSource bindingВиробники = new BindingSource();
BindingSource bindingПродажіВиробника = new BindingSource();
BindingSource bindingСтруктура = new BindingSource();

private void formПродажіВиробника_Load(object sender, EventArgs e)
{
    //******************
    // comboBoxВиробник *
    //******************

    // Заповнюємо таблицю Виробники
    ХлібDataSetTableAdapters.ВиробникиTableAdapter
        виробникиTableAdapter = new
        ХлібDataSetTableAdapters.ВиробникиTableAdapter();
    виробникиTableAdapter.Fill(хлібDataSet.Виробники);

    // Відображаємо дані
    bindingВиробники.DataSource = хлібDataSet.Виробники;
    comboBoxВиробник.DataSource = bindingВиробники;
```

```
comboBoxВиробник.DisplayMember = "Виробник";
comboBoxВиробник.ValueMember = "Код_виробника";

//*****************************
// dataGridViewПродажіВиробника *
//*****************************

// Заповнюємо таблицю Товари
ХлібDataSetTableAdapters.ТовариTableAdapter
    товариTableAdapter = new
    ХлібDataSetTableAdapters.ТовариTableAdapter();
товариTableAdapter.Fill(хлібDataSet.Товари);

// Заповнюємо таблицю Продажі
ХлібDataSetTableAdapters.ПродажіTableAdapter
    продажіTableAdapter = new
    ХлібDataSetTableAdapters.ПродажіTableAdapter();
продажіTableAdapter.Fill(хлібDataSet.Продажі);

// Будуємо запит заданими таблиць Продажі
// і Товари для вибраного виробника
var queryПродажіВиробника =
    from продаж in хлібDataSet.Продажі.AsEnumerable()
    join товар in хлібDataSet.Товари.AsEnumerable()
    on продаж.Код_товару equals товар.Код_товару
    where продаж.Код_виробника ==
        (int)comboBoxВиробник.SelectedValue
    select new
    {
        Дата = продаж.Дата,
        Товар = товар.Товар,
        Ціна = товар.Ціна,
        Кількість = продаж.Кількість,
        Вартість = товар.Ціна * продаж.Кількість
    };

// Відображаємо дані
bindingПродажіВиробника.DataSource = queryПродажіВиробника;
dataGridViewПродажіВиробника.DataSource = bindingПродажіВиробника;
dataGridViewПродажіВиробника.AutoResizeColumns();

// Форматуємо виведені дані (грошовий стиль)
dataGridViewПродажіВиробника.Columns["Ціна"].DefaultCellStyle.Format
    = "0.00";
dataGridViewПродажіВиробника.Columns["Ціна"].DefaultCellStyle.
```

```
    Alignment= DataGridViewContentAlignment.MiddleRight;
    dataGridViewПродажіВиробника.Columns["Вартість"].DefaultCellStyle.Format
        = "0.00";
    dataGridViewПродажіВиробника.Columns["Вартість"].DefaultCellStyle.
    Alignment = DataGridViewContentAlignment.MiddleRight;
    // Форматуємо виведені дані стовпця Кількість (ціле число)
    dataGridViewПродажіВиробника.Columns["Кількість"].
    DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleRight;

    //Сума
    Decimal Сума = queryПродажіВиробника.Sum(p =>p.Вартість);
    textBoxСума.Text = Сума.ToString("0.00");
    textBoxСума.TextAlign = HorizontalAlignment.Right;
}
```

6. Перейдіть у вікно конструктора форми ***Хліб***, двічі клацніть кнопку ***Продажі виробника*** і додайте код оброблювача події "Клацання кнопки Продажі виробника".

```
private void buttonПродажіВиробника_Click(object sender, EventArgs e)
{
    formПродажіВиробника вікноПродажіВиробника = new
        formПродажіВиробника();
    вікноПродажіВиробника.ShowDialog();
}
```

7. Перевірте функціональність форми ***Продажі виробника*** і збережіть зміни, зроблені у проєкті.

5.2. Забезпечення інтерактивності форми

Завдання

Додайте інтерактивність формі, щоб під час вибору в полі зі списком іншого виробника, на формі відображалися відповідні дані.

Ідеї виконання

Щоб у разі зміни виробника відображалися нові дані, слід додати оброблювача події **SelectedIndexChanged** для поля зі списком ***Виробник***.

В оброблювачі потрібно перечитати запит з урахуванням нового значення в умові відбору й обновити прив'язування даних. Таке оброблення буде здійснюватися після повного завантаження форми, якщо виконується

умова **if (this.CanFocus)**. Інакше оброблювач буде викликатися ще під час завантаження форми, що призведе до помилки.

Виконання

1. Перейдіть у вікно конструктора форми ***Продажі виробника*** та двічі клацніть поле зі списком ***Виробник***.

2. Уведіть оператори тіла оброблювача події ***comboBoxВиробник_SelectedIndexChanged***. Він має такий вигляд:

```
private void comboBoxВиробник_SelectedIndexChanged(object sender,
    EventArgs e)
{
    // Після завантаження
    if (this.CanFocus)
    {
        var queryПродажіВиробника =
        from продаж in хліб DataSet.Продажі.AsEnumerable()
        join товар in хліб DataSet.Товари.AsEnumerable()
        on продаж.Код_товару equals товар.Код_товару
        where продаж.Код_виробника ==
           (int)comboBoxВиробник.SelectedValue
        select new
        {
            Дата = продаж.Дата,
            Товар = товар.Товар,
            Ціна = товар.Ціна,
            Кількість = продаж.Кількість,
            Вартість = товар.Ціна * продаж.Кількість
        };
        // Відображаємо дані
        bindingПродажіВиробника.DataSource =
            queryПродажіВиробника;
        //Сума
        Decimal Сума= queryПродажіВиробника.Sum(p => p.Вартість);
        textBoxСума.Text = Сума.ToString("0.00");
    }
}
```

3. Збережіть зміни, зроблені у проєкті.

4 Перевірте функціональність поля зі списком ***Виробник*** на формі ***Продажі виробника***. Після зміни виробника мають змінюватися дані в елементах DataGridView і TextBox.

5.3. Побудова діаграми

Завдання

Додайте на форму ***Продажі виробника*** об'ємну кругову діаграму (рис. 5.13). У ній відображено структуру продажів товарів вибраного виробника (яку частину займає кожний вид товару в загальній сумі продажів його товарів).

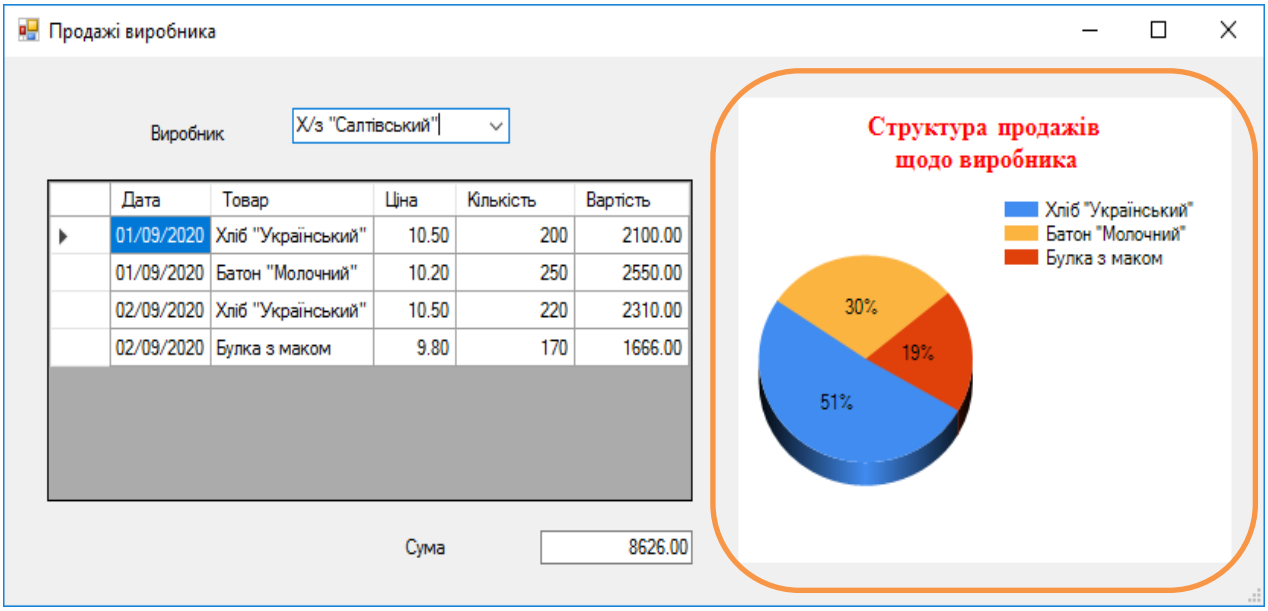


Рис. 5.13. **Кругова діаграма на формі *Продажі виробника***

Виконання

1. Перейдіть у вікно конструктора форми ***Продажі виробника*** та збільште її ширину приблизно на 12 см.

2. Додайте на форму елемент **Chart** і дайте йому ім'я ***chartСтруктура***.

3. Перейдіть у вікно коду форми й додайте таке посилання на простір імен:

```
//Для діаграми
using System.Windows.Forms.DataVisualization.Charting;
```

4. Перебуваючи у вікні коду форми, додайте в оброблювач події ***formПродажіВиробника_Load*** такий код, помістивши його перед фігурною дужкою оброблювача, яка закривається:

```
//************
// Діаграма *
//************

// Запит
var queryСтруктура =
from товар inqueryПродажіВиробника
group товар by товар.Товар into t
    select new
    {
        Товар = t.Key,
        Вартість = t.Sum(p => p.Вартість)
    };
    // Дані для відображення
    bindingСтруктура.DataSource = queryСтруктура;
    chartСтруктура.DataSource = bindingСтруктура;
    chartСтруктура.Series["Series1"].XValueMember = "Товар";
    chartСтруктура.Series["Series1"].YValueMembers = "Вартість";

// Тип діаграми
chartСтруктура.Series["Series1"].ChartType = SeriesChartType.Pie;

// Підписи на діаграмі
chartСтруктура.Series["Series1"].Label = "#PERCENT{P0}";
//"#VALX\n#PERCENT";

// Підписи як виноски
// this.chartСтруктура.Series["Series1"]["PieLabelStyle"]="Outside";

// Об'ємний варіант (3D)
this.chartСтруктура.ChartAreas[0].Area3DStyle.Enable3D = true;

// Заголовок
chartСтруктура.Titles.Add("Заголовок");
chartСтруктура.Titles[0].Text=
    "Структура продажів\n щодо виробника";
chartСтруктура.Titles[0].Font
= new Font("Times New Roman", 12, FontStyle.Bold);
    chartСтруктура.Titles[0].ForeColor = Color.Red;

// Легенда
chartСтруктура.Series["Series1"].IsVisibleInLegend = true;
chartСтруктура.Series["Series1"].LegendText = "#VALX";//"#AXISLABEL";
```

Примітка. Деякі конструкції коду закоментовано. Вони можуть знадобитися під час побудови інших діаграм.

5. Перевірте можливість відображати діаграму після завантаження форми.

6. У вікні коду форми додайте такий код в оброблювач події **comboBoxВиробник_*SelectedIndexChanged***, помістивши його перед фігурною дужкою оператора **if**, яка закривається:

```
//***********
// Діаграма *
//***********

// Запит
var queryСтруктура =
from товар in queryПродажіВиробника
    group товар by товар.Товар into t
    select new
    {
        Товар = t.Key,
        Вартість = t.Sum(p => p.Вартість)
    };

// Дані для відображення
bindingСтруктура.DataSource = queryСтруктура;
if (queryСтруктура.Count() > 0)
{
    chartСтруктура.DataSource = bindingСтруктура;
    chartСтруктура.Series["Series1"].XValueMember = "Товар";
    chartСтруктура.Series["Series1"].YValueMembers = "Вартість";
    chartСтруктура.Series["Series1"].ChartType = SeriesChartType.Pie;
}
```

7. Збережіть зміни, зроблені у проєкті.

8. Перевірте функціональність поля зі списком ***Виробник*** щодо можливості змінювати діаграму. Простежте за тим, щоб на діаграмі відображали дані, які відповідають тим, що подають в елементі управління DataGridView.

У звіті з лабораторної роботи подайте скриншот форми ***Продажі виробника*** в режимі виконання, а також код оброблювача ***formПродажі-Виробника_Load*** та ***comboBoxВиробник_SelectedIndex Changed***. Поясніть, чи був сенс використовувати технологію LINQ to DataSet в оброблювачах подій форми ***Продажі виробника***, чи можна це було зробити з використанням класичної технології.

Завдання для самостійного виконання

1. За аналогією з формою ***Продано*** побудуйте форму ***Поставки виробників***. У ній відобразіть дані про вартість усіх товарів, які надійшли від кожного виробника (inner join та left join).

2. Додайте стовпчикову діаграму на форму ***Продажі виробників***. На ній подайте вартості проданих товарів вибраного виробника за датами (на осі X). Бажано додати лінії тренда за кожним товаром.

3. Попередньо додайте в базу даних таблицю ***Групи*** з полями ***Код_групи*** та ***Група*** й пов'яжіть її за зовнішнім ключем із таблицею ***Товари***. Після цього виконайте такі завдання:

3.1. Додайте поле зі списком ***Група*** на форму ***Прайс***. Із його допомогою будуть відбирати товари заданої групи.

3.2. Додайте поле зі списком ***Група*** на форму ***Продажі***, щоб після вибору групи у списку ***Товар*** відображали товари тільки вибраної групи.

4. Додайте на форму ***Накладні*** список ***Журнал накладних***, який дозволяє швидко знайти потрібну накладну. У списку передбачте такі стовпці: ***Код_накладної***, ***Дата*** відпуску товарів за накладною, ***Виробник***, від якого отримано товари та ***Сума*** вартостей усіх товарів за накладною. Значенням вибраного елемента списку є ***Код_накладної***.

5. На формі ***Продажі виробників*** додайте дані про найкращий товар (топтовар) за результатами продажів за такими показниками:

5.1. За кількістю проданих одиниць.

5.2. За загальною вартістю.

5.3. За отриманим прибутком.

6. До елементів поля зі списком ***Виробник*** на формі ***Продажі виробників*** додайте елемент ***Усі***. У разі його вибору відображають дані про продаж товарів усіма виробниками.

7. На формі ***Продажі виробників*** додатково забезпечте такі види фільтрації:

7.1. За видом товарів (за допомогою елемента ComboBox);

7.2. За діапазоном дат (за допомогою двох елементів TextBox або DateTimePicker).

Примітка. Передбачте ситуацію, коли кожен фільтр не діє, тобто відображають усі дані.

8. У проєкті з індивідуальною базою даних застосуйте прийоми, які було розглянуто в базовій частині лабораторної роботи, а також у завданнях для самостійного виконання.

Лабораторна робота 6
Реалізація доступу до даних на основі технології Code First

Цілі лабораторної роботи:

1. Набуття практичних навичок у побудові класів сутностей та класу DbContext.
2. Освоєння засобів Data Annotations для налаштування моделей.
3. Оволодіння засобами вдосконалення моделі на основі міграцій.
4. Набуття практичних навичок у побудові застосунків Windows Forms на основі технології Code First.
5. Оволодіння практичними навичками в розробленні інтерфейсу користувача в разі ієрархічних даних.
6. Удосконалення навичок у роботі з інтегрованим середовищем розроблення Visual C# і довідковою системою Microsoft Developer Network (MSDN).

Перед виконанням роботи студент має знати:

основи створення POCO-класів;
базові домовленості в Code First;
основи проєктування реляційних баз даних;
структурні елементи бази даних і їхні властивості;
основи побудови SQL-запитів;
основи технології LINQ;
принципи прив'язування даних до елементів інтерфейсу;
основи роботи з ієрархічними даними;
принципи оброблення подій у C#-програмі.

Після виконання роботи студент має вміти:

створювати класи сутностей і DbContext;
налаштовувати сутності засобами Data Annotations;
супроводжувати бази даних засобами міграцій;
самостійно розробляти C#-застосунок на основі технології Code First.

Підготовча частина

Хід роботи

1. Побудова початкової моделі.
2. Налаштування моделі.
3. Розвиток моделі (міграції).
4. Побудова застосунку.

Форма звітності

За результатами виконання роботи необхідно оформити електронний звіт засобами Word. За кожним завданням слід зробити скриншоти з результатами виконання, а також подати код оброблювача події. У кінці завдань зазначено, що саме слід подати у звіті.

За результатами кожного виконаного завдання з п. "Завдання для самостійного виконання" слід подати постановку завдання, скриншот форми та відповідний програмний код.

Критерії оцінювання

У 12-бальній системі оцінку результату захисту лабораторної роботи формують за такими правилами:

1. За виконання кожного пункту практичної частини може бути виставлено від 0 до 1 бала.

2. За виконання завдань п. "Завдання для самостійного виконання" може бути виставлено:

від 0 до 1 бала за кожне завдання 1, 2;

від 0 до 2 балів за завдання 3;

від 0 до 1 бала за завдання 4;

від 0 до 2 балів за завдання 5;

від 0 до 1 бала за завдання 6;

від 0 до 1 бала за кожне завдання 7, 8;

від 0 до 2 балів за кожне завдання 9;

від 0 до 2 балів за завдання 10 для кожного розділу 1 – 4.

3. За кілька варіантів виконання одного із завдань додають 1 бал.

4. За вибір варіанта, який із кількох варіантів виконання є оптимальним, та обґрунтування вибору додають 1 бал.

5. За виконання кожного завдання в іншій СУБД (Oracle, MySQL, DB2, PostgreSQL тощо) додають по 1 балу.

6. За запізнення із захистом лабораторної роботи знімають 2 бали за кожний тиждень.

Набрану кількість балів із відповідей на кожне завдання лабораторної роботи підсумовують. У результаті такого підрахунку студент може набрати від 0 до 12 балів.

У разі запізнення із захистом лабораторної роботи понад три тижні студент виконує завдання роботи на основі індивідуальної бази даних. Роботу оцінюють за описаними раніше критеріями.

Практична частина

Постановка загального завдання

Вивчення засобів роботи з даними на основі технології Code First здійснюють шляхом створення рішення ***codeFirstХліб***, яке складається із двох проєктів – ***моделювання*** та ***winForms***.

Проєкт ***моделювання*** є консольним, а ***winForms*** – проєктом Windows Forms. У проєкті ***моделювання*** створюють класи сутностей та клас DbContext. На їхній основі будують базу даних, яку потім налаштовують засобами Data Annotations. Після цього її розвивають із використанням засобів міграцій. Щоб під час захисту лабораторної роботи можна було продемонструвати обидва підходи до налаштування моделей, перед початком налаштувань зберігають попередній варіант (без налаштувань) і виконують налаштування з його копії.

Проєкт ***winForms*** призначено для побудови інтерфейсу користувача з даними, що зберігають у базі даних. Під час його розроблення використовують засоби, аналогічні до тих, що використовували під час виконання попередньої лабораторної роботи. Але водночас вони мають свою специфіку. Слід звернути увагу на особливості роботи з ієрархічними сутностями. Під час реалізації застосунку, що працює з такими сутностями, необхідно розширити клас Observable Collection.

Отже, консольний проєкт використовують для виконання п. 1 – 3 ходу роботи, а проєкт Windows Forms – п. 4. Обидва проєкти взаємопов'язані між собою – другий проєкт використовує ті класи, що створені в першому. У такий спосіб реалізують багатошарову архітектуру застосунків.

1. Побудова початкової моделі

1.1. Установлення пакета EntityFramework

1. Створіть нове рішення ***codeFirstХліб***, а в ньому консольний проєкт ***моделювання***. Для цього:

1.1. Відкрийте Visual Studio.

1.2. Виконайте команду **File – New – Project**.

1.3. Виберіть вид застосунку **Console Application** у вікні **New – Project**, а потім уведіть ім'я проєкту ***моделювання*** в текстовому полі **Name** та ім'я рішення ***codeFirstХліб*** у текстовому полі **Solution Name** (рис. 6.1).

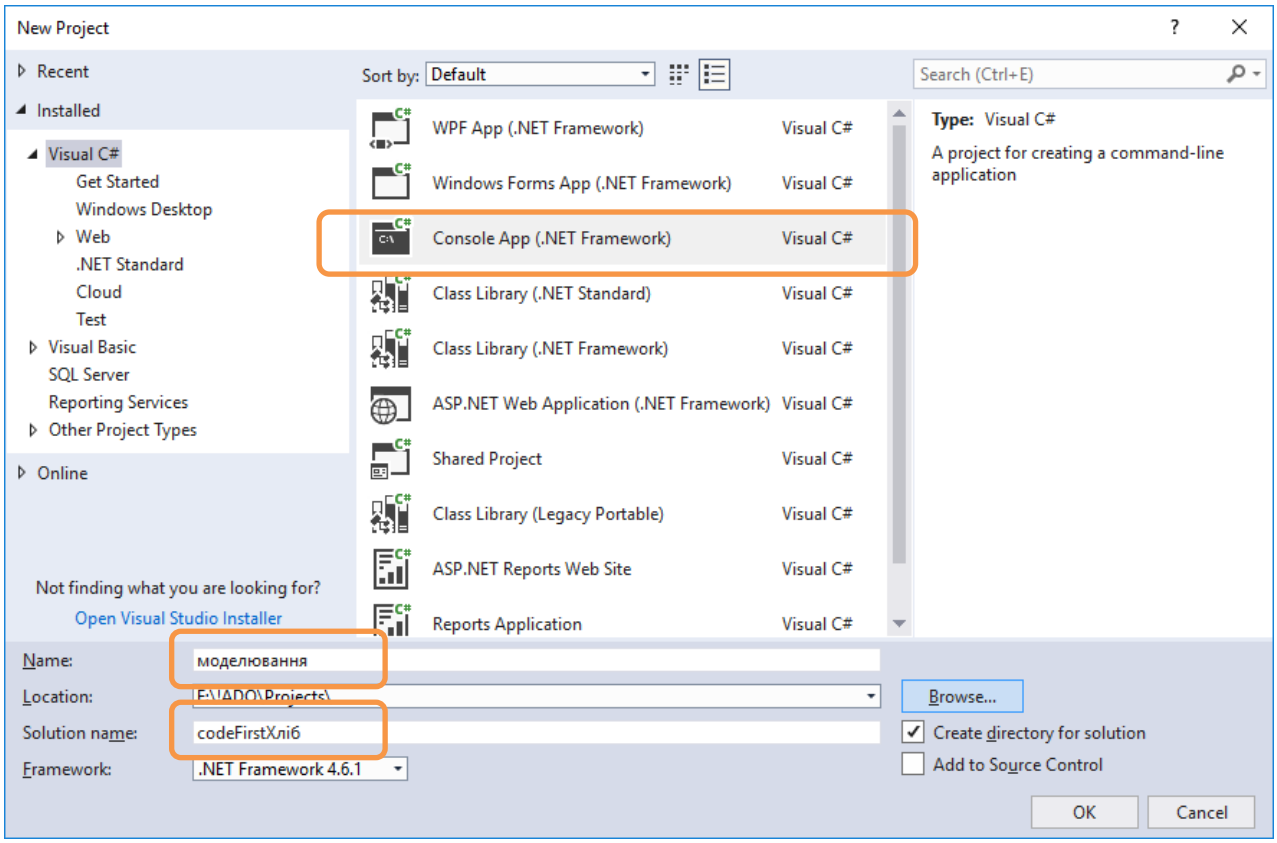


Рис. 6.1. **Побудова рішення *codeFirstХліб* із консольним проєктом *моделювання***

2. Додайте останню версію пакета EntityFramework за допомогою засобу NuGet. Для цього:

2.1. Виконайте команду **Project – Manage NuGet Packages**. З'явилося вікно Manage NuGet **Packages**.

2.2. Виберіть елемент **Browse** у верхній частині вікна.

2.3. Виберіть елемент **EntityFramework** у центральному списку і клацніть кнопку **Install** (рис. 6.2).

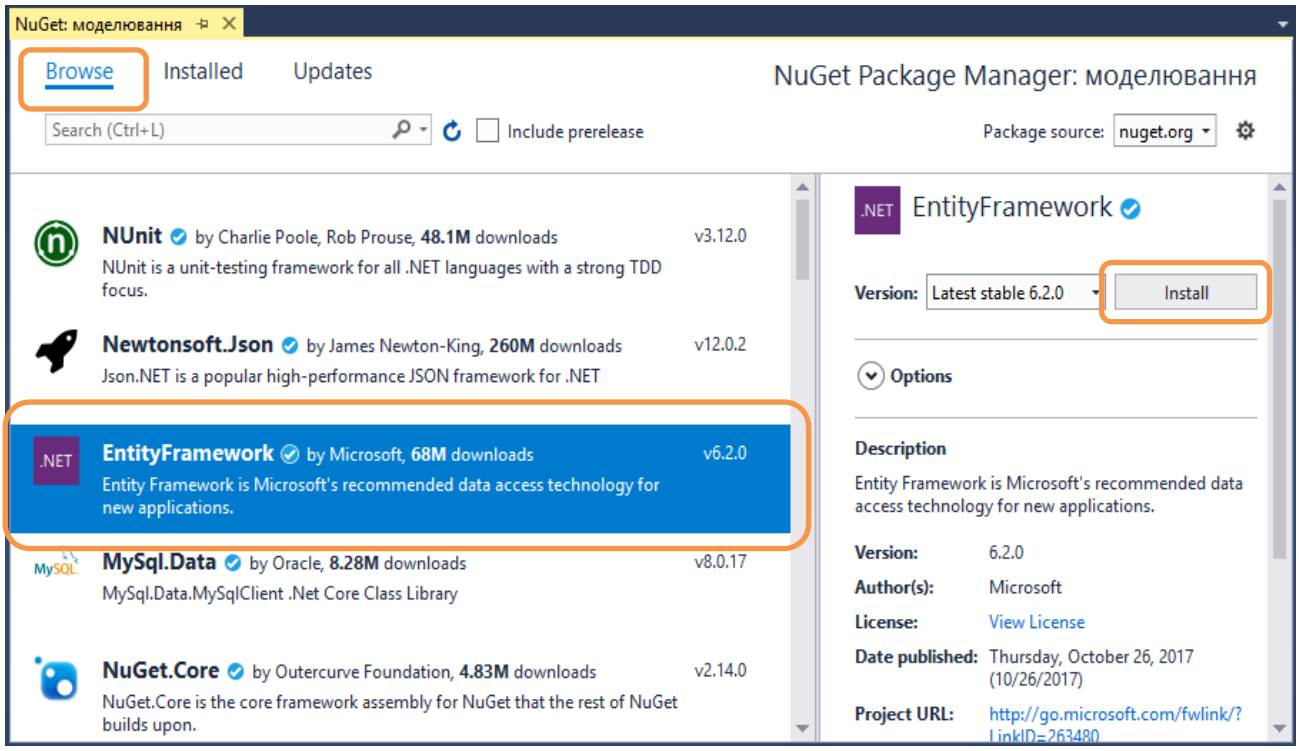


Рис. 6.2. **Додавання останньої версії пакета EntityFramework**

Дочекайтеся, поки завершиться встановлення пакета EntityFramework, клацнувши кнопку **I Accept**. У папці ***моделюванняХліб*** разом із файлом рішення з'явилася папка **packages**. Перевірте її вміст.

Примітка. Якщо лабораторну роботу виконують у середовищі Visual Studio, у якій ще не встановлено засіб **NuGet** (у меню **Project** відсутня команда **Manage NuGet Packages**), то перед виконанням п. 2 встановіть його, виконавши таке:

1. Перебуваючи в середовищі Visual Studio, виконайте команду **Tools – Extension Manager**. З'явилося вікно **Extension Manager**.

2. Виберіть елемент **Online Gallery** в лівому списку.

3. Виберіть елемент **NuGet Package Manager** у центральному списку і клацніть кнопку **Download**.

4. Далі відповідайте на запити завантажувача, щоб установити пакет **NuGet**.

Пакет **NuGet** можна також завантажити на свій комп'ютер із-поза меж Visual Studio, перейшовши на сайт http://visualstudiogallery.msdn.microsoft.com/27077b70-9dad-4c64-adcf-c7cf6bc9970c.

На комп'ютерах університету таке завантаження виконують системні адміністратори.

1.2. Побудова класів сутностей

Завдання

Побудуйте класи сутностей, за якими в подальшому буде створено базу даних ***Хліб***. У базі даних будуть зберігатися таблиці, аналогічні таблицям ***Товари***, ***Виробники*** та ***Продажі***, які розглядали в лабораторних роботах до вивчення платформи Entity Framework.

Ідеї виконання

Клас сутності є відображенням рядка відповідної таблиці бази даних. Тому в ньому як властивості класу вказують поля таблиці. Типам даних властивостей класу відповідають типи полів у таблиці.

Щоб спростити створення бази даних, класам і таблицям слід дати імена англійською мовою. Причому імена класів будуть мати форму однини (***Product***, ***Manufacturer*** і ***Sale***), тоді імена таблиць автоматично дістануть форму множини (***Products***, ***Manufacturers*** та ***Sales***).

Майже всім властивостям сутностей можна дати ті самі імена українською мовою, що збігаються з іменами полів у таблицях, за винятком ключових полів (первинних і зовнішніх ключів). Імена первинних ключів будуть утворювати за таким шаблоном: `Ім'я_сутностіID`, наприклад, ***ProductID***, ***ManufacturerID***.

Для встановлення зв'язків між таблицями варто зазначити властивості навігації у класах сутностей. Причому в батьківській сутності треба вказати колекцію сутностей дочірньої сутності (наприклад, у сутності ***Product***):

```
public virtual ICollection<Sale> Sales { get; set; },
```

а в дочірній сутності, навпаки, – один екземпляр батьківської сутності (наприклад, у сутності ***Sale***):

```
public virtual Product Product { get; set; }.
```

Примітка. Відсутність однієї із двох властивостей навігації не призводить до втрати відношення між таблицями в базі даних, але ускладнює побудову запитів LINQ у коді.

Виконання

1. Перейдіть у вікно **Solution Explorer** і додайте до проєкту ***моделювання*** нову папку ***Models***.

2. Додайте в папку ***Models*** новий клас ***Product*** і введіть його опис:

```
public class Product
{
    public int ProductID { get; set; }
    public string Товар { get; set; }
    public Decimal Ціна { get; set; }
    public Decimal Ціна_закупівлі { get; set; }

    // Властивості навігації
    public virtual ICollection<Sale> Sales { get; set; }
}
```

3. Додайте в папку ***Models*** новий клас ***Manufacturer*** і введіть його опис:

```
public class Manufacturer
{
    public int ManufacturerID { get; set; }
    public string Виробник { get; set; }
    public string Адреса { get; set; }
    public string Телефон { get; set; }

    // Властивості навігації
    public virtual ICollection<Sale> Sales { get; set; }
}
```

4. Додайте в папку ***Models*** новий клас ***Sale*** і введіть його опис:

```
public class Sale
{
    public int SaleID { get; set; }
    public DateTime Дата { get; set; }
    public int ManufacturerID { get; set; }
    public int ProductID { get; set; }
    public Int16 Кількість { get; set; }

    // Властивості навігації
    public virtual Product Product { get; set; }
    public virtual Manufacturer Manufacturer { get; set; }
}
```

5. Збережіть зміни, зроблені у проєкті.

1.3. Побудова класу DbContext

Завдання

Побудуйте клас DbContext, за яким буде створено таблиці бази даних ***Хліб*** і в подальшому будуть здійснювати доступ до них із застосунку через створені раніше класи сутностей. У базі даних будуть міститися таблиці, аналогічні таблицям ***Товари***, ***Виробники*** та ***Продажі***, які розглядали в попередніх лабораторних роботах.

Ідеї виконання

У базі даних будуть міститися таблиці, що є аналогічними до таблиць ***Товари***, ***Виробники*** та ***Продажі***. Для доступу до них слід створити клас, у якому вказати як властивості колекції сутностей, описані у створених раніше у класах. Для спрощення відображення цих колекцій у таблиці бази даних треба дати їм такі самі імена, як для таблиць, наприклад, опис властивості

```
public DbSet<Product> Products { get; set; }
```

слугує для відображення колекції ***Products*** сутностей ***Product*** у таблицю ***Products***.

Виконання

1. Додайте до проєкту ***моделювання*** нову папку ***DataAccess***.

2. Додайте в папку ***DataAccess*** новий клас ***BreadContext*** і введіть його опис:

```
public class BreadContext : DbContext
{
    public DbSet<Product> Products { get; set; }
    public DbSet<Manufacturer> Manufacturers { get; set; }
    public DbSet<Sale> Sales { get; set; }
}
```

3. Щоб забезпечити доступ до опису класів, додайте посилання на простори імен:

```
using System.Data.Entity;
using моделювання.Models;
```

1.4. Заповнення даними

Завдання

Створіть базу даних ***Хліб*** і заповніть її тестовими даними.

Ідеї виконання

Щоб створити базу даних, відповідно до вказаних описів класів, потрібно запустити застосунок на виконання і здійснити будь-яку операцію з її даними. Тому заповнення даними бази попередньо викличе її створення.

Для розвантаження методу ***Main*** у класі ***Program*** слід винести процес заповнення даними в метод ***FillDB*** окремого класу ***DataAdd***.

Заповнення кожної таблиці бази даних засноване на такому алгоритмі. Спочатку створюють екземпляри сутностей. Потім їхнім властивостям надають певні значення. Після цього сутності додаються до колекцій. Усі ці дії виконуються безпосередньо в пам'яті. Для збереження даних у базі викликають метод **SaveChanges** класу **DbContext**.

Щоб під час повторного запуску застосунку на виконання не було проблем із повторенням даних, слід дотримуватися стратегії видалення вже наявної бази даних і створення нової. Для цього вказують метод **DropCreateDatabaseAlways** в ініціалізаторі бази даних.

Якщо явно не задано ім'я бази даних через рядок підключення, платформа Entity Framework створить базу даних з ім'ям ***моделювання.DataAccess.BreadContext*** на сервері ./SQLEXPRESS для Visual Studio 2010 (чи (localDB)/v.11 або (localDB)/MSSQLLocalDB, залежно від версії Visual Studio). Тому в разі обмеження прав користувача може з'явитися виключення про неможливість створення бази даних. Цю проблему вирішено в завданні 2.1.

Виконання

1. Додайте до проєкту ***моделювання*** нову папку ***Fill***.
2. Додайте в папку ***Fill*** новий клас ***DataAdd*** і введіть його опис:

```
public class DataAdd
{
    public static void FillDB(out int nProduct,
         out int nManufacturer,
         out int nSale)
```

```
{
    var products = new List<Product>
    {
        new Product { Товар = @"Хліб ""Український""", Ціна = 10.50M,
            Ціна_закупівлі =9.30M},
        new Product { Товар = @"Батон ""Молочний""", Ціна = 10.20M,
            Ціна_закупівлі =9.10M},
        new Product { Товар = @"Булка з маком", Ціна = 9.80M,
            Ціна_закупівлі = 8.65M}
    };
    using (var context = new BreadContext())
    {
        products.ForEach(p => context.Products.Add(p));
        context.SaveChanges();
        // Визначаємо кількість доданих рядків
        nProduct = context.Products.Count();
    }

    var manufacturers = new List<Manufacturer>
    {
        new Manufacturer { Виробник = @"Х/з ""Салтівський""",
            Адреса = "вул. Гв. Широнінців, 1",
            Телефон ="(057)710-50-40"},
        new Manufacturer { Виробник = @"Х/з ""Кулиничі""",
            Адреса = "смт Кулиничі, вул. Шкільна, 18",
            Телефон ="(0572)62-51-37"}
    };
    using (var context = new BreadContext())
    {
        manufacturers.ForEach(m => context.Manufacturers.Add(m));
        context.SaveChanges();
        // Визначаємо кількість доданих рядків
        nManufacturer = context.Manufacturers.Count();
    }

    var sales = new List<Sale>
    {
        new Sale { Дата = DateTime.Parse("01.09.2020"),
            ManufacturerID =1,
            ProductID = 1,
            Кількість=200 },
        new Sale { Дата = DateTime.Parse("01.09.2020"),
            ManufacturerID =1,
            ProductID = 2,
            Кількість=250 },
        new Sale { Дата = DateTime.Parse("01.09.2020"),
            ManufacturerID =2,
            ProductID = 1,
            Кількість=150 }
    };
```

```
    using (var context = new BreadContext())
    {
        sales.ForEach(s => context.Sales.Add(s));
        context.SaveChanges();

        // Визначаємо кількість доданих рядків
        nSale = context.Sales.Count();
    }
}
}
```

3. Щоб забезпечити доступ до опису класів, додайте посилання на простори імен:

```
using моделювання.Models;
using моделювання.DataAccess;
```

4. Перейдіть у вікно класу ***Program*** і додайте опис методу ***Main***:

```
class Program
{
    static void Main(string[] args)
    {
        try
        {
            // Стратегія роботи з базою даних
            Database.SetInitializer(
                new DropCreateDatabaseAlways<BreadContext>());

            // Описуємо фактичні параметри методу FillDB
            int nProduct;
            int nManufacturer;
            int nSale;

            // Викликаємо метод
            DataAdd.FillDB(out nProduct, out nManufacturer,
                out nSale);

            // Виводимо результат
            Console.WriteLine(
                "Базу даних на SQL Server створено і заповнено.\n"
                + "У таблиці записано таку кількість рядків:\n"
                + "Products - " + nProduct
                + ", Manufacturers - " + nManufacturer
                + ",  Sales - " + nSale
                + ".\n Перевірте!!!");
        }
        catch (System.Exception ex)
```

```
        {
           // Виводимо повідомлення про помилку
           Console.WriteLine("Базу даних не створено. \n Помилка:\n "
           + ex.ToString());
        }

       // Виводимо повідомлення про подальші дії
       Console.WriteLine("Натисніть будь-яку клавішу, щоб вийти...");
       Console.ReadKey();
    }
}
```

5. Щоб забезпечити доступ до опису класів, додайте посилання на простори імен:

```
using System.Data.Entity;
using моделювання.DataAccess;
using моделювання.Fill;
```

6. Запустіть програму на виконання і дочекайтеся, поки у вікні консолі з'явиться повідомлення.

Примітка. Якщо виконання п. 6 завершилося повідомленням про помилку, пов'язану з обмеженням прав користувача, відразу переходьте до виконання п. 2.1 "Установлення імені бази даних".

7. Знайдіть нову базу даних ***моделюванняХліб.DataAccess. BreadContext*** на сервері SQL Server. Ознайомтеся із властивостями стовпців її таблиць і даними, що містяться в таблицях. Вони відповідають тим, що вказані в методі ***FillDB***. Щоб знайти нову базу даних, створіть нове з'єднання з нею. Для цього:

7.1. Виконайте команду **Add Connection** із контекстового меню значка **Data Connections**, розташованого у вікні **Server Explorer**.

7.2. Перевірте, чи встановлено значення **Microsoft SQL Server (SqlClient)** у полі **Data source**. Якщо ні, то скористайтеся кнопкою **Change**.

7.3. Уведіть ім'я сервера в полі **Server name**. Залежно від версії Visual Studio, ім'я сервера має такі значення:

для Visual Studio 2010 –**.\SQLExpress**;

для Visual Studio 2012, 2013 – **(localDB)\v11.0**;

для вищих версій – **(localDB)\MSSQLLocalDB**.

7.4. Виберіть у полі зі списком **Select or enter a database name** виберіть ім'я бази даних ***моделювання.DataAccess.BreadContext*** (див. рис. 6.3).

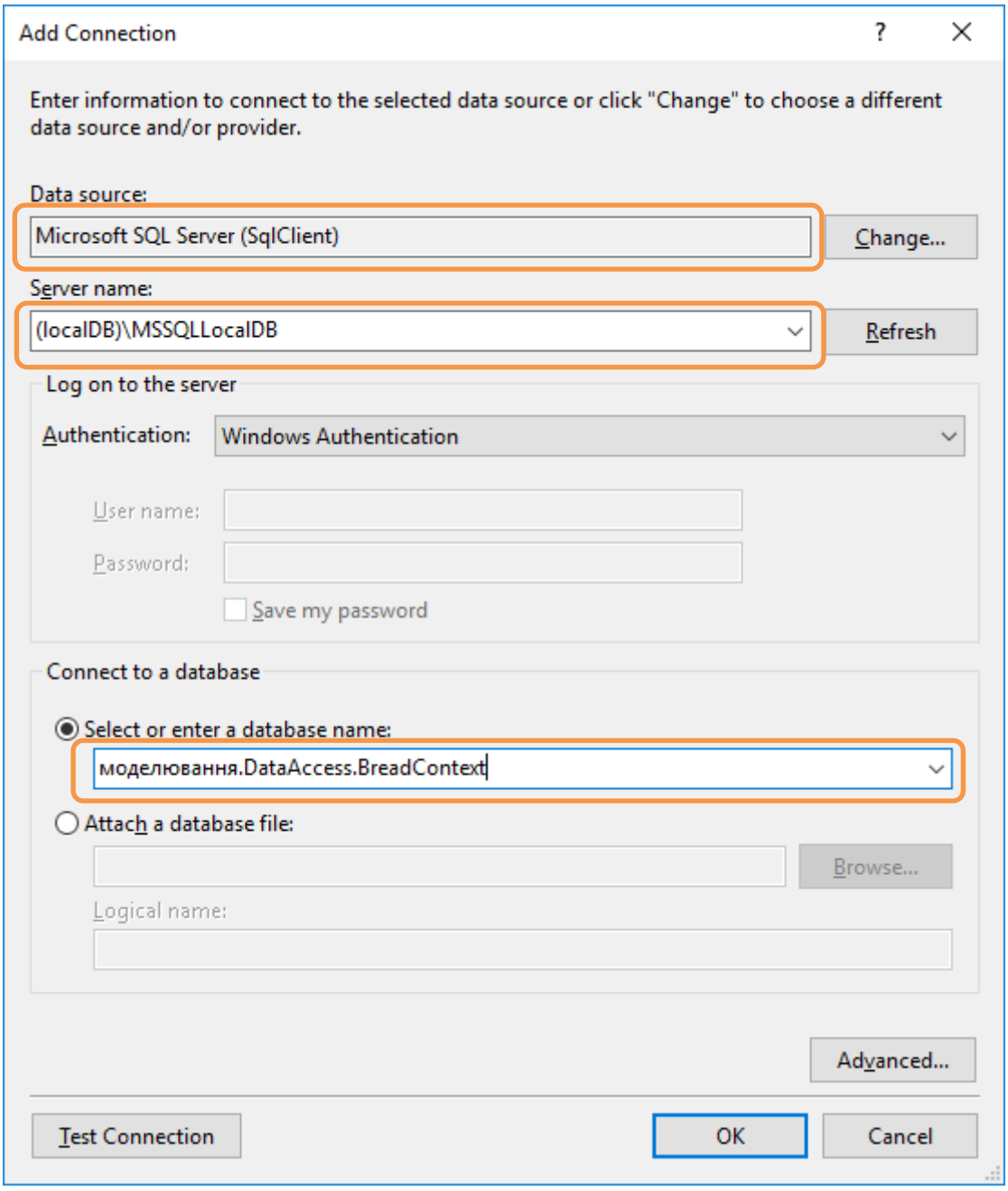


Рис. 6.3. **Установлення нового з'єднання з базою даних**

Ім'я бази даних збігається з іменем класу **DbContext**. Далі слід потурбуватися про те, щоб під час створення база даних діставала ім'я, яке вказав користувач.

У звіті з лабораторної роботи подайте код моделей сутностей та опишіть домовленості, прийняті в технології Code First, які використовували.

Наведіть результат виконання п. 6 завдання підрозділу 1.4 і поясніть причину саме такого результату. Якщо виконання п. 6 завершили успішно, подайте скриншот структури нової бази даних у вікні **Server Explorer** із полями усіх таблиць.

2. Налаштування моделі

2.1. Установлення імені бази даних

Завдання

Створіть базу даних з ім'ям ***Хліб*** і заповніть її даними.

Ідеї виконання

Відповідно до домовленостей Code First, треба вибрати ім'я бази даних із рядка під'єднання, який у файлі конфігурації має ім'я класу **DbContext**, тобто ***BreadContext***.

Виконання

1. Якщо в результаті виконання попереднього завдання успішно створено базу даних ***моделювання.DataAccess.BreadContext***, видаліть її із сервера SQL Server, наприклад за допомогою запиту:

```
DROP DATABASE [моделювання.DataAccess.BreadContext]
```

Примітка. Перед виконанням запиту закрийте з'єднання з базою даних у вікні **Server Explorer**.

2. Відкрийте файл конфігурації **App.config** і додайте тег рядка з'єднання, помістивши його перед останнім закриваючим тегом конфігурації. Якщо лабораторну роботу виконують на власному комп'ютері, він має такий вигляд:

```
<connectionStrings>
<add name="BreadContext"
    connectionString="Data Source=Сервер;
        AttachDbFilename=Шлях\Хліб Прізвище.mdf;
         Integrated Security=True"
    providerName="System.Data.SqlClient" />
</connectionStrings>
```

Примітки:

1) залежно від версії Visual Studio величина **Сервер** має такі значення:

для Visual Studio 2010 –. **\SQLExpress**;

для Visual Studio 2012, 2013 – **(localDB)\v11.0**;

для вищих версій – **(localDB)\MSSQLLocalDB**;

2) замість слова **Шлях** у значенні параметра **AttachDbFilename** потрібно вказати шлях до папки, у якій зберігають проєкт, наприклад,

F:\!ADO\Projects\codeFirstХліб\моделювання.

Після цього рядок коду для наведеного прикладу буде мати такий вигляд:

```
AttachDbFilename=F:\!ADO\Projects\codeFirstХліб\моделювання\Хліб***Прізвище***.mdf;;
```

3) замість слова ***Прізвище*** у значенні параметра **AttachDbFilename** потрібно вставити своє прізвище. Тоді база даних буде мати відповідне ім'я, наприклад, ***Хліб-Петренко***;

4) в описах подальших завдань база даних буде мати узагальнене ім'я ***Хліб***.

Якщо лабораторну роботу виконують на комп'ютері університету в середовищі Visual Studio 2010 (сервер **.\SQLExpress**), слід додатково вказати екземпляр користувача сервера, зазначивши параметр **User Instance=True**. У цьому разі тег **connectionStrings** має такий вигляд:

```
<connectionStrings>
<add name="BreadContext"
    connectionString="Data Source=.\SQLExpress;
        AttachDbFilename=Шлях\Хліб***Прізвище***.mdf;
        Integrated Security=True;
        User Instance=True"
    providerName="System.Data.SqlClient" />
</connectionStrings>
```

3. Запустіть програму на виконання і дочекайтеся, поки у вікні консолі з'явиться повідомлення.

4. Переконайтеся, що в папці проєкту з'явилися два нові файли (база даних і її журнал).

5. Установіть під'єднання файлу нової бази даних ***Хліб***, розміщеного в папці проєкту ***моделювання***. Для цього скористайтеся командою **Add Connection** із контекстового меню значка **Data Connections**, розташованого у вікні **Server Explorer,** і перевірте, чи встановлено значення **Microsoft SQL Server Database File (SqlClient)** у полі **Data source** вікна **Add Connection**. Ознайомтеся із властивостями стовпців її таблиць і даними,

що містяться в таблицях (рис. 6.4). Вони відповідають тим, що вказані в методі ***FillDB***.

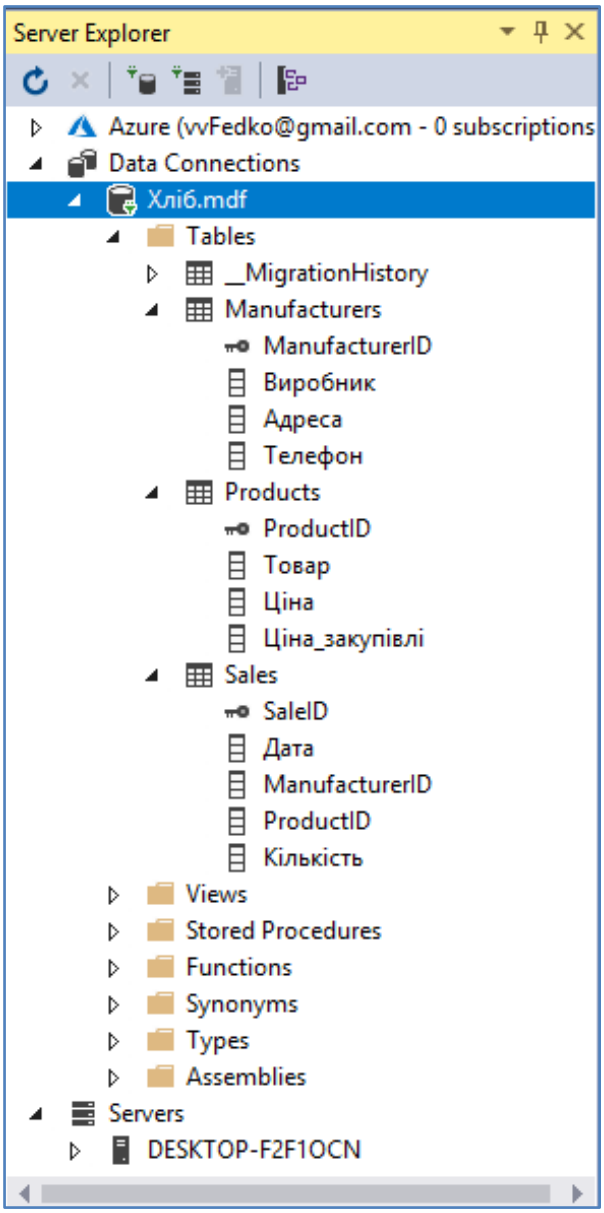


Рис. 6.4. **База даних *Хліб***

6. Закрийте вікно проєкту зі збереженням зроблених змін.

2.2. Налаштування сутностей із використанням Data Annotations

Завдання 1

Налаштуйте властивості сутності ***Product***, щоб вони відповідали властивостям полів таблиці ***Товари*** в попередніх проєктах. Тобто властивості мають мати атрибути, наведені в табл. 6.1.

Таблиця 6.1

Атрибути властивостей сутності Product

Властивості	Атрибути
ProductID	Ключ
Товар	Обов'язкова, довжина не перевищує 25 символів
Ціна	Необов'язкова
Ціна_закупівлі	Обов'язкова

Ідеї виконання

Для встановлення відповідних атрибутів перед властивостями слід зазначити анотації даних, якщо атрибут не відповідає домовленостям, прийнятим у Code First.

Оскільки ім'я властивості ***ProductID*** складається з імені сутності й символів ***ID***, тому за домовленостями Code First вона є ключем сутності та використання анотації [Key] у цьому разі є надлишковим.

Щоб установити атрибути ***обов'язкова*** і ***довжина не перевищує 25 символів*** для властивості ***Товар***, треба скористатися анотаціями ***Required*** та ***MaxLength***.

Оскільки властивість ***Ціна*** є необов'язковою, варто зазначити, що вона належить до класу **Nullable**, тобто може мати значення ***null***. Якщо цю обставину пропустити, то в разі, коли не задано значення для властивості ***Ціна***, вона отримає значення ***0***, а це не те саме, що ***null***.

Для властивості ***Ціна_закупівлі*** слід зазначити анотацію ***Required***, яка забезпечить атрибут ***обов'язкова***.

Виконання

1. Скопіюйте папку, у якій міститься рішення ***codeFirstХліб*** і новій папці дайте ім'я ***codeFirstХліб_Data_Annotations***. Рішення ***codeFirstХліб*** ще знадобиться під час захисту лабораторної роботи, щоб можна було продемонструвати початковий варіант проєкту і подальше його вдосконалення.

2. Відкрийте рішення ***codeFirstХліб***, що міститься в папці рішення ***codeFirstХліб_Data_Annotations***.

3. Переналаштуйте шлях до бази даних, розміщеної в папці проєкту **моделювання** скопійованого рішення. Для цього скоригуйте шлях до бази даних у параметрі **AttachDbFilename**, що міститься в тегу **connectionStrings** файлу конфігурації **App.config**.

4. Змініть також шлях до бази даних у з'єднанні, що перебуває у вікні **Server Explorer**. Для цього відкрийте вікно **Modify Connection**, скориставшись однойменною командою з контекстового меню значка бази даних, розташованого у вікні **Server Explorer**, і встановіть потрібне значення в полі **Database file name (new or existing)** за допомогою кнопки **Browse**. Потім перевірте з'єднання за допомогою кнопки **Test Connection** (рис. 6.5).

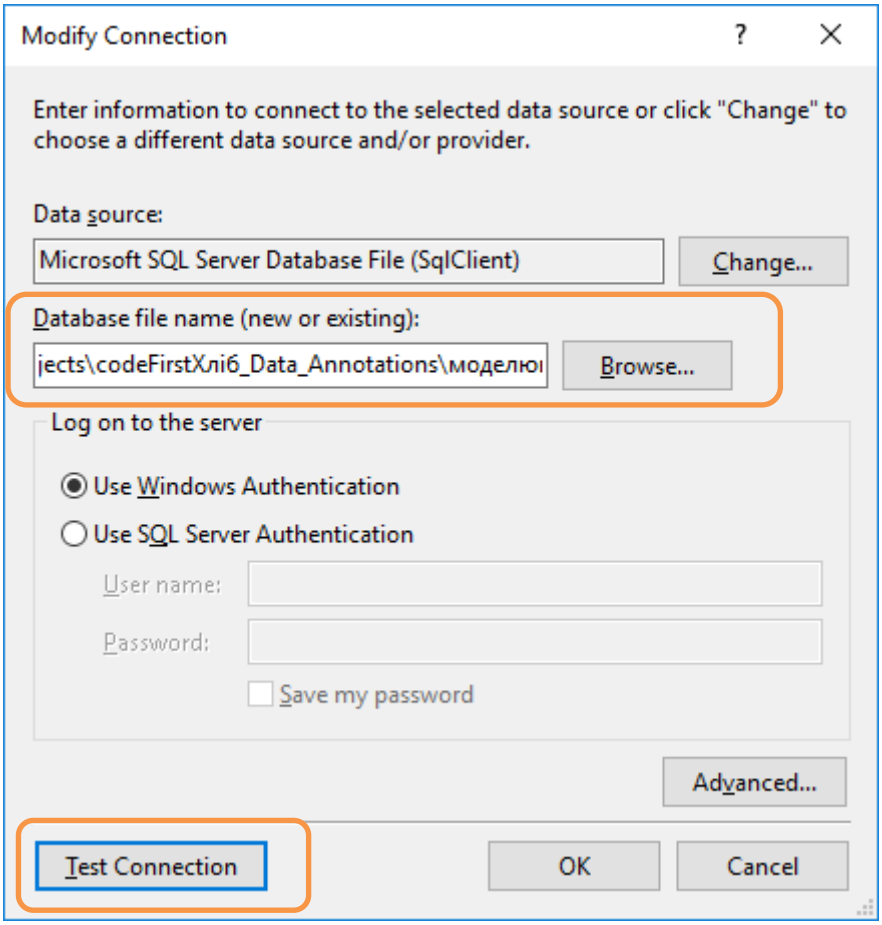


Рис. 6.5. **Зміна шляху до бази даних у з'єднанні**

5. Відкрийте визначення полів таблиці ***Products***, двічі клацнувши на її значок у вікні **Server Explorer**. На рис. 6.6 подано властивості полів таблиці ***Products***.

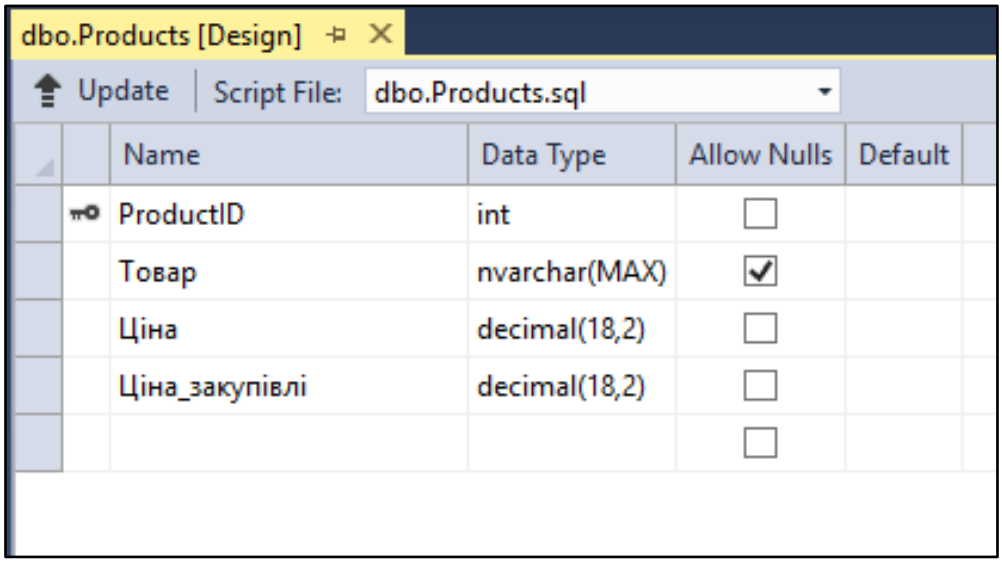


Рис. 6.6. **Властивості полів таблиці *Products***

6. Щоб змінити властивості полів таблиці, відкрийте код класу ***Product*** і вставте анотації перед полями класу. Код класу має такий вигляд:

```
public class Product
{
    public int ProductID { get; set; }
    [Required]
    [MaxLength(25)]
    public string Товар { get; set; }
    public Nullable<Decimal> Ціна { get; set; }
    [Required]
    public Decimal Ціна_закупівлі { get; set; }

    // Властивості навігації
    public virtual ICollection<Sale> Sales { get; set; }
}
```

Щоб забезпечити доступ до опису класів, додайте посилання на простір імен:

```
using System.ComponentModel.DataAnnotations;
```

7. Запустіть програму на виконання і дочекайтеся, поки у вікні консолі з'явиться повідомлення. Обновіть вікно **Server Explorer** і перегляньте властивості полів таблиці ***Products*** (рис. 6.7).

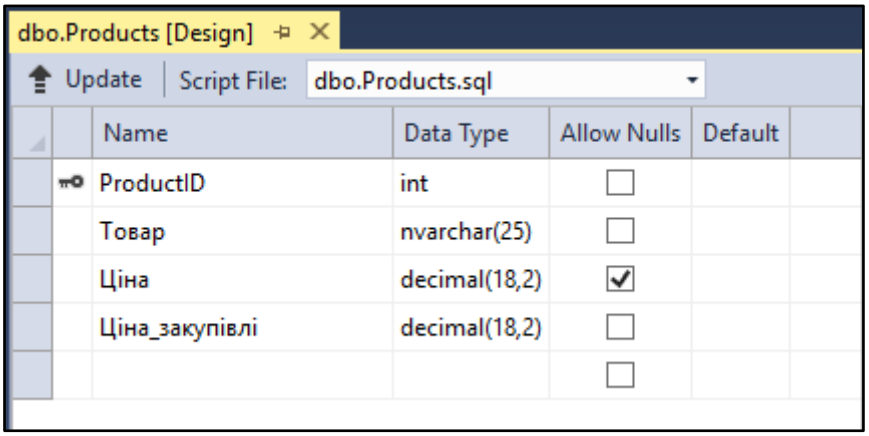


Рис. 6.7. **Властивості полів таблиці *Products* після налаштування**

Примітка. Оскільки під час виконання програми видаляють базу даних і створюють нову, стежте за тим, щоб з'єднання з нею було закрите. Це можна зробити контекстною командою бази даних **Close Connection** у вікні **Server Explorer**.

Завдання 2

Налаштуйте властивості сутності ***Manufacturer***, щоб вони відповідали властивостям полів таблиці ***Виробники*** в попередніх проєктах. Тобто властивості мають мати атрибути, наведені в табл. 6.2.

Таблиця 6.2

Атрибути властивостей сутності Manufacturer

Властивості	Атрибути
ManufacturerID	Ключ
Виробник	Обов'язкова, довжина не перевищує 20 символів
Адреса	Обов'язкова, довжина не перевищує 30 символів
Телефон	Обов'язкова, довжина не перевищує 15 символів

Ідеї виконання

Установлення атрибутів властивостей сутності виконують аналогічно описаному для сутності ***Product***.

Виконання

1. Відкрийте визначення полів таблиці ***Manufacturers***, двічі клацнувши на її значок у вікні **Server Explorer**. На рис. 6.8 подано властивості полів таблиці ***Manufacturers***.

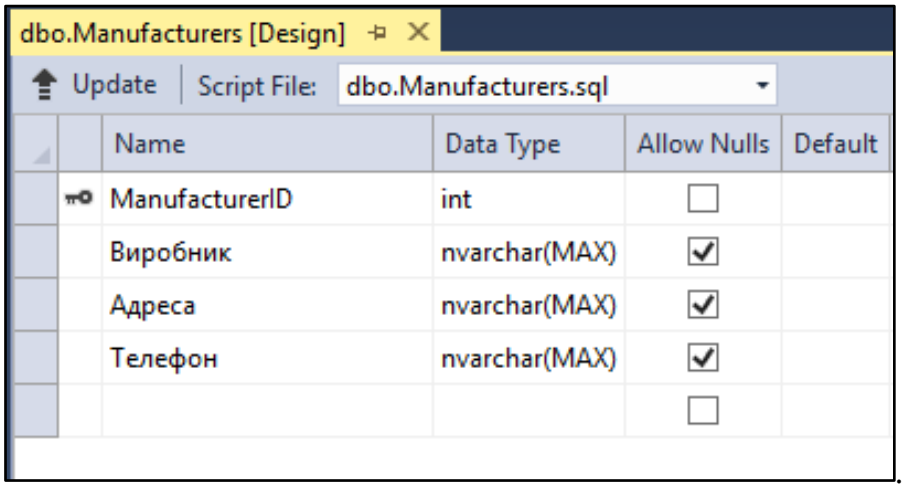


Рис. 6.8. **Властивості полів таблиці *Manufacturers***

2. Щоб змінити властивості полів таблиці, відкрийте код класу ***Manufacturer*** і вставте анотації перед полями класу. Код класу має такий вигляд:

```
public class Manufacturer
{
    public int ManufacturerID { get; set; }
    [Required]
    [MaxLength(20)]
    public string Виробник { get; set; }
    [Required]
    [MaxLength(30)]
    public string Адреса { get; set; }
    [Required]
    [MaxLength(15)]
    public string Телефон { get; set; }

    // Властивості навігації
    public virtual ICollection<Sale> Sales { get; set; }
}
```

Щоб забезпечити доступ до опису класів, додайте посилання на простір імен:

```
using System.ComponentModel.DataAnnotations;
```

3. Запустіть програму на виконання і дочекайтеся, поки у вікні консолі з'явиться повідомлення. Обновіть вікно **Server Explorer** і перегляньте властивості полів таблиці ***Manufacturers*** (рис. 6.9).

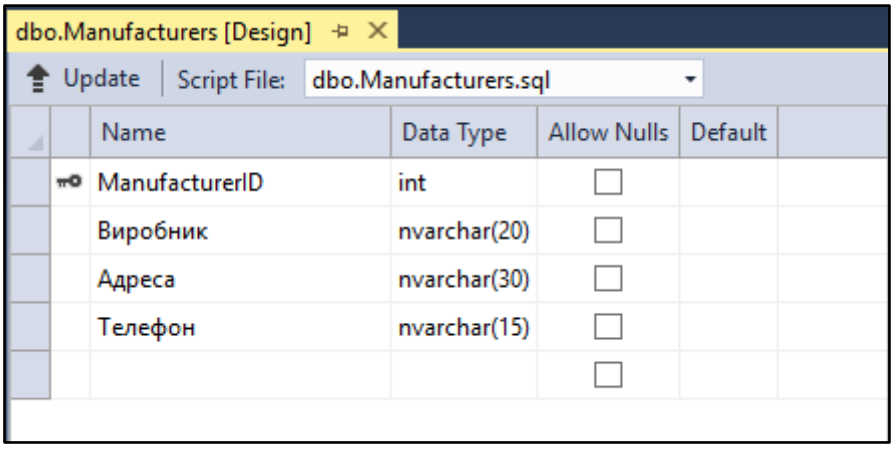


Рис. 6.9. **Властивості полів таблиці *Manufacturers* після налаштування**

Завдання 3

Налаштуйте властивості сутності ***Sale***, щоб вони відповідали властивостям полів таблиці ***Продажі*** в попередніх проєктах. Тобто властивості мають мати атрибути, наведені в табл. 6.3.

Таблиця 6.3

Атрибути властивостей сутності Sale

Властивості	Атрибути
SaleID	Ключ
Дата	Обов'язкова, відображено у форматі dd.MM.yyyy
ManufacturerID	Зовнішній ключ для зв'язку із сутністю Manufacturer
ProductID	Зовнішній ключ для зв'язку із сутністю Product
Кількість	Обов'язкова, ціла, змінюється в межах від 0 до 10 000

А також до сутності ***Продажі*** додайте обчислювану властивість ***Вартість***, що визначають за такою формулою:

$$Вартість = Ціна \times Кількість,$$

але в базі даних не зберігають.

Ідеї виконання

Установлення атрибутів властивостей сутності виконують аналогічно до описаного для сутності ***Product*** та ***Manufacturer***.

Оскільки імена властивостей ***ProductID*** та ***ManufacturerID*** складається з імен сутностей, із якими встановлюється зв'язок, і символів ***ID***, тому за домовленостями CodeFirst вона є зовнішніми ключами і використання анотації [ForeignKey("Product")] чи [ForeignKey("Manufacturer")] у цьому разі є надлишковим.

Щоб властивість ***Вартість*** не зберігалася в базі даних, треба використати анотацію [NotMapped].

Виконання

1. Відкрийте визначення полів таблиці ***Sales***, двічі клацнувши на її значок у вікні **Server Explorer**. На рис. 6.10 подано властивості полів таблиці ***Sales***.

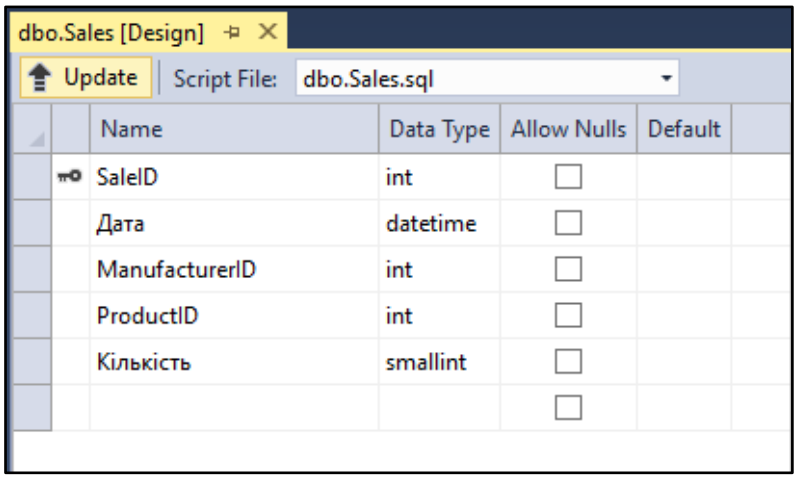


Рис. 6.10. **Властивості полів таблиці *Sales***

2. Щоб змінити властивості полів таблиці, відкрийте код класу ***Sale*** і вставте анотації перед полями класу, а також додайте опис властивості ***Вартість***. Код класу має такий вигляд:

```
public class Sale
{
    public int SaleID { get; set; }
    [Required]
    [DisplayFormat(DataFormatString = "{0:dd.MM.yyyy}",
        ApplyFormatInEditMode = true)]
    public DateTime Дата { get; set; }
    public int ManufacturerID { get; set; }
```

```
    public int ProductID { get; set; }
    [Required]
    public Int16 Кількість { get; set; }

    // Властивості навігації
    public virtual Product Product { get; set; }
    public virtual Manufacturer Manufacturer { get; set; }

    // Обчислювана властивість
    [NotMapped]
    public decimal Вартість
    {
        get
        {
            if (Product != null)
                return (Decimal)(Product.Ціна * Кількість);
            else
                return 0;
        }
    }
}
```

Щоб забезпечити доступ до опису класів, додайте посилання на простір імен:

```
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
```

3. Запустіть програму на виконання і дочекайтеся, поки у вікні консолі з'явиться повідомлення. Обновіть вікно **Server Explorer** і перегляньте властивості полів таблиці ***Sales*** (рис. 6.11).

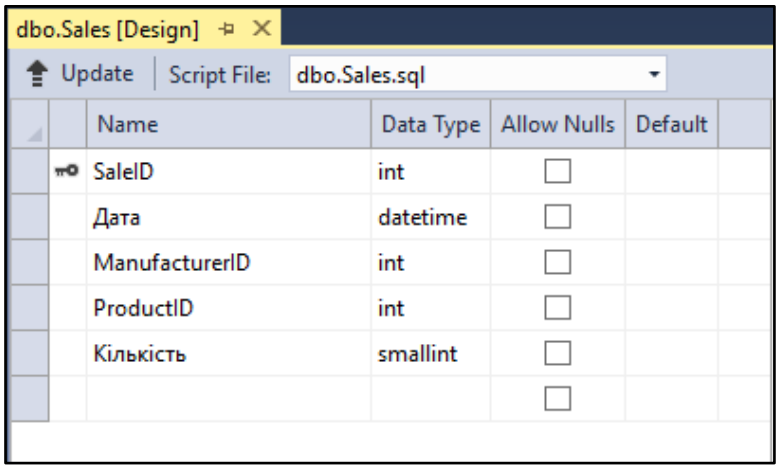


Рис. 6.11. **Властивості полів таблиці *Sales* після налаштування**

4. Закрийте вікно проєкту зі збереженням зроблених змін.

У звіті з лабораторної роботи подайте засоби налаштування, які використовували для бази даних загалом і для кожної сутності зокрема під час виконання завдань розд. 2 та опишіть визначені результати.

3. Розвиток моделі (міграції)

3.1. Умикання міграцій

Завдання

Додайте засоби розвитку проєкту з можливостями відкочування до певної версії.

Ідеї виконання

Засоби Data Annotation та Fluent API дозволяють тільки налаштувати сутності. Застосовуючи описані засоби, спочатку видаляли базу даних і, замість неї, створювали нову. Такі самі кардинальні операції з базою даних виконують у разі, коли потрібно додавати нові сутності або змінювати вже наявні. Водночас значно ускладнено можливість повернутися до однієї з попередніх версій моделі та, відповідно, бази даних. Для вирішення цієї проблеми застосовують міграції.

Для опанування засобів міграцій слід використати рішення, що міститься в папці ***codeFirstХліб_Data_Annotation***.

Виконання

1. Скопіюйте папку ***codeFirstХліб_Data_Annotation*** і новій папці дайте ім'я ***codeFirstХліб_Migration***, щоб під час захисту лабораторної роботи можна було продемонструвати попередній варіант проєкту та подальший його розвиток.

2. Відкрийте проєкт ***моделювання***, що міститься в папці ***codeFirstХліб_Migration***.

3. Переналаштуйте шлях до бази даних, розміщеної в папці проєкту ***моделювання*** скопійованого рішення як це було описано в п. 3, 4 завд. 1 розд 2.2. "Налаштування сутностей із використанням Data Annotations".

4. Відкрийте вікно **Package Manager Console** за допомогою однойменної команди, що міститься у списку команди **View – OtherWindows**.

5. Уведіть команду **Enable-Migrations** у вікні **Package Manager Console** і натисніть клавішу **Enter**.

У вікні **Package Manager Console** з'являється повідомлення про доступність міграцій у проєкті (рис. 6.12).

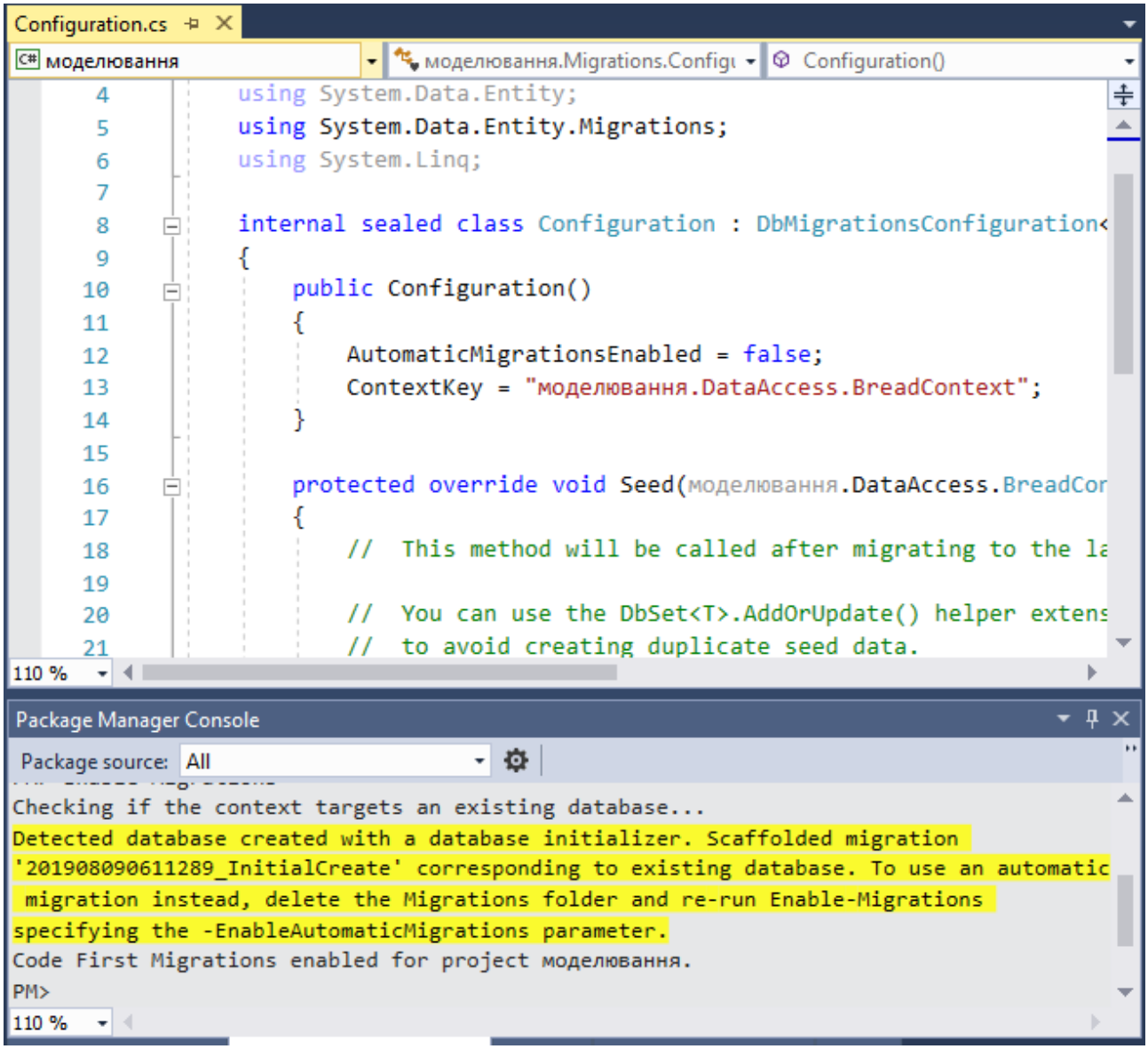


Рис. 6.12. **Вікна *Configuration.cs* та *Package Manager Console***

У вікні **Solution Explorer** з'явилася папка **Migrations** із двома файлами: ***xxx_InitialCreate.cs*** та ***Configuration.cs***. Причому останній відкрито. Він описує клас конфігурації й надає можливості налаштування її поведінки. У цьому класі також міститься метод ***Seed***, який викликають під час виконання кожної міграції для заповнення таблиць бази даних.

6. Відкрийте файл ***xxx_InitialCreate.cs***, познайомтеся з його вмістом і визначте призначення методів ***Up*** та ***Down*** класу ***InitialCreate***.

3.2. Формування методу Seed

Завдання

Забезпечте можливість заповнення даними таблиць ***Products***, ***Manufacturers*** та ***Sales***.

Ідеї виконання

Оскільки під час міграцій базу даних не створюють повторно, а метод **Seed** викликають кожний раз під час виконання міграції, то до таблиць можна повторно додавати ті самі дані. Щоб запобігти цьому, потрібно перевіряти дані, що додають, на збіг із тими, що вже зберігають у таблицях.

Найпростіше це робити в довідниках. Там перевірку можна виконувати разом із додаванням до колекції сутностей, які потрібно зберегти в базі даних. Для цього використовують метод ***AddOrUpdate***. У ньому перший параметр указує назву властивості сутності, за якою виконують перевірку на неповторюваність. Наприклад, для сутності ***Product*** можна застосувати такий алгоритм формування колекції нових даних:

```
products.ForEach(p => context.Products.AddOrUpdate(t => t.Товар, p))
```

У ньому новими вважають дані, що мають оригінальну назву товару.

Якщо ж перевірку потрібно виконати за даними кількох властивостей, алгоритм ускладнюється. Наприклад, для сутності ***Sale*** можна застосувати такий алгоритм формування колекції нових даних:

```
foreach (Sale s in sales)
{
    var saleInDB = context.Sales.Where(
        p.ManufacturerID == s.ManufacturerID &&
        p => p.ProductID == s.ProductID &&
        p.Дата == s.Дата).SingleOrDefault();

    if (saleInDB == null)
    {
        context.Sales.Add(s);
    }
}
```

Виконання

1. Відкрийте файл ***DataAdd.cs***, що перебуває у папці ***Fill***, і скопіюйте з нього тіло методу ***FillDB***, а потім уставте його як тіло методу ***Seed*** у класі ***Configuration***.

2. Відкоригуйте тіло методу ***Seed***, щоб позбавитися дублювання даних під час формування колекції нових даних. Після цього метод ***Seed*** буде мати такий вигляд:

```
protected override void Seed(моделювання.DataAccess.BreadContext context)
{
    var products = new List<Product>
    {
         new Product
             { Товар = @"Хліб ""Український""",
               Ціна = 10.50M,
               Ціна_закупівлі =9.30M},
         new Product
             { Товар = @"Батон ""Молочний""",
               Ціна =10.20M,
               Ціна_закупівлі =9.10M},
         new Product
             { Товар = @"Булка з маком",
               Ціна = 9.80M,
               Ціна_закупівлі =8.65M}
    };

    // Формуємо колекцію нових товарів без повторювань
    products.ForEach(p => context.Products.AddOrUpdate(t =>
        t.Товар, p));
    context.SaveChanges();
    var manufacturers = new List<Manufacturer>
    {
        new Manufacturer
            { Виробник = @"Х/з ""Салтівський""",
              Адреса = "вул. Гв. Широнінців, 1",
              Телефон ="(057)710-50-40"
},
        new Manufacturer
            { Виробник = @"Х /з ""Кулиничі""",
              Адреса = "смт Кулиничі, вул. Шкільна, 18",
              Телефон ="(0572)62-51-37"
}
    };

    // Формуємо колекцію нових виробників без повторювань
    manufacturers.ForEach(m => context.Manufacturers.AddOrUpdate
```

```
        (v => v.Виробник, m));
    context.SaveChanges();

    var sales = new List<Sale>
    {
        new Sale
            { Дата = DateTime.Parse("01.09.2020"),
              ManufacturerID =1,
              ProductID = 1,
              Кількість=200 },
        new Sale
            { Дата = DateTime.Parse("01.09.2020"),
              ManufacturerID =1,
              ProductID = 2,
              Кількість=250 },
        new Sale
            { Дата = DateTime.Parse("01.09.2020"),
              ManufacturerID =2,
              ProductID = 1,
              Кількість=150 }
    };

    // Формуємо колекцію нових даних про продажі без повторювань
    foreach (Sale s in sales)
    {
        var saleInDB = context.Sales.Where(
            p => p.ProductID == s.ProductID &&
            p.ManufacturerID == s.ManufacturerID &&
            p.Дата == s.Дата).SingleOrDefault();
        if (saleInDB == null)
        {
            context.Sales.Add(s);
        }
    }
    context.SaveChanges();

}
```

Щоб забезпечити доступ до опису класів, додайте посилання на простори імен:

```
using System.Collections.Generic;
using моделювання.Models;
```

3. Щоб перевірити, чи метод ***Seed*** може додавати нові дані без повторювань рядків, спочатку видаліть усі таблиці з бази, потім створіть їх знову і додайте в них дані. Після цього спробуйте повторно додати

ті самі дані. Усе це зробіть за допомогою міграцій. Для цього виконайте таке:

3.1. Зробіть відкочування до порожньої бази даних за допомогою команди **Update-Database – TargetMigration: $InitialDatabase.**

Перевірте, що в базі даних не залишилося жодної таблиці.

3.2. Виконайте всі відкладені міграції (у цьому разі ***InitialDatabase***), щоб повернутися до того стану, у якому була база даних до відкочування. Для цього виконайте команду **Update-Database.**

Зверніть увагу на повідомлення про те, що під час виконання міграції InitialCreate викликався метод Seed. Перевірте, що в базі даних з'явилися всі заповнені таблиці.

3.3. Додайте порожню міграцію, виконання якої буде викликати тільки метод **Seed**. Для цього виконайте команду **Add-Migration TestSeed.**

3.4. Виконайте останню міграцію за допомогою команди **Update-Database.**

Зверніть увагу на повідомлення про те, що під час виконання міграції ***TestSeed*** викликався метод **Seed**. Перевірте вміст таблиць бази даних. Повторне виконання методу **Seed** не призвело до появи однакових рядків.

Отже, метод Seed працює правильно. Після виконання нової міграції він робить спробу додати в таблиці бази даних тільки ті дані, яких ще там не було.

3.3. Додавання нової сутності Invoice

Завдання

Додайте до моделі даних нову сутність ***Invoice***. У базі даних вона реалізують у вигляді таблиці ***Invoices***, що відповідає таблиці ***Накладні*** в попередніх роботах.

Ідеї виконання

У папку ***Models*** потрібно додайте новий клас ***Invoice***, а до класу ***BreadContext*** – інформацію про відповідний об'єкт класу DbSet. На основі цієї інформації Code First створить таблицю ***Invoices***.

Для виконання операції створення додайте два рядки до таблиці ***Invoices***. Причому в першому рядку задайте значення всіх полів, а у другому – можна пропустити значення поля ***Номер_накладної***, щоб перевірити реакцію на значення Null. У подальшому це знадобиться для демонстрування інших операцій у міграціях.

Щоб не зачіпати об'єкти бази даних, що були створені раніше, для додавання нової сутності слід скористатися механізмом міграцій. Тому додавання рядків до таблиці ***Invoices*** треба вказати в методі ***Seed***.

Виконання

1. Додайте в папку ***Models*** новий клас ***Invoice*** та введіть його опис:

```
public class Invoice
{
    public int InvoiceID { get; set; }
    [MaxLength(6)]
    public string Номер_накладної { get; set; }
    public DateTime Дата { get; set; }
    public int ManufacturerID { get; set; }

    // Властивості навігації
    public virtual Manufacturer Manufacturer { get; set; }
}
```

Щоб забезпечити доступ до опису класів, додайте посилання на простір імен:

```
using System.ComponentModel.DataAnnotations;
```

2. Додайте у клас ***BreadContext***, що перебуває в папці ***DataAccess***, властивість колекції нових сутностей, розташувавши її в кінці класу:

```
public DbSet<Invoice> Invoices { get; set; }
```

3. Додайте до методу ***Seed***, що перебуває у класі ***Configuration***, код для додавання двох рядків до таблиці ***Invoices***, розташувавши його в кінці методу:

```
    //***********
    // Inovices *
    //***********
    // Нові сутності
    var invoices = new List<Invoice>
    {
        new Invoice
        { Номер_накладної ="101",
          Дата=DateTime.Parse("01.09.2020"),
          ManufacturerID=1 },
        new Invoice
```

```
        { Дата=DateTime.Parse("01.09.2020"),
          ManufacturerID=2 }
    };

    // Перевіряємо рядки на дублювання накладних
    foreach (Invoice i in invoices)
    {
        var invoiceInDB = context.Invoices.Where(
            n => n.Дата == i.Дата &&
            n.ManufacturerID == i.ManufacturerID
            ).FirstOrDefault();
        if (invoiceInDB == null)
        {
            context.Invoices.Add(i);
        }
    }

    context.SaveChanges();
```

4. Щоб створити шаблон для міграції, яка реалізує додавання нової сутності, уведіть команду **Add-Migration AddInvoice** у вікні **Package Manager Console**.

У папці ***Migrations*** з'явився новий клас ***AddInvoice***. Він містить метод ***Up*** для створення таблиці ***Invoices***, а також метод ***Down*** для її видалення.

5. Щоб виконати міграцію і додати дані в нову таблицю, уведіть команду **Update-Database – Verbose** у вікні **Package Manager Console**. Ключ **Verbose** забезпечує виведення відповідних SQL-команд.

6. Перейдіть у вікно **Server Explorer** і перегляньте схему таблиці ***Invoices*** та дані, що зберігають у ній (рис. 6.13 і 6.14).

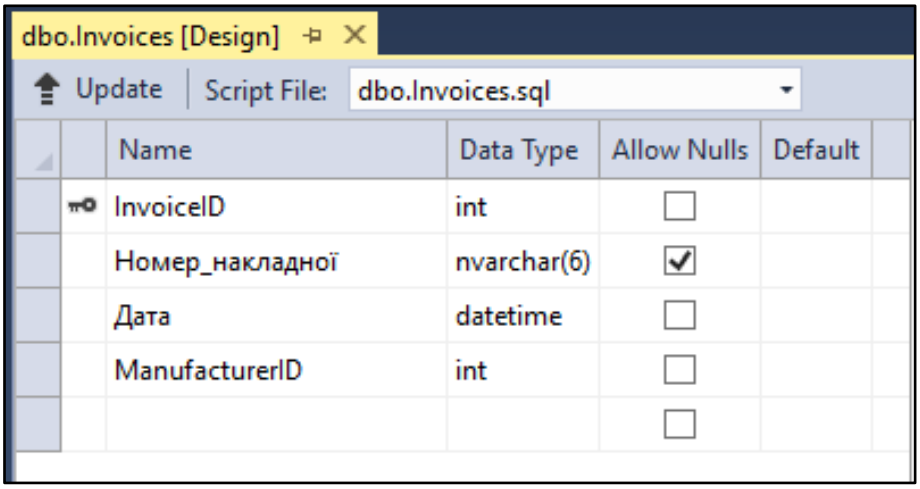


Рис. 6.13. **Схема таблиці *Invoices***

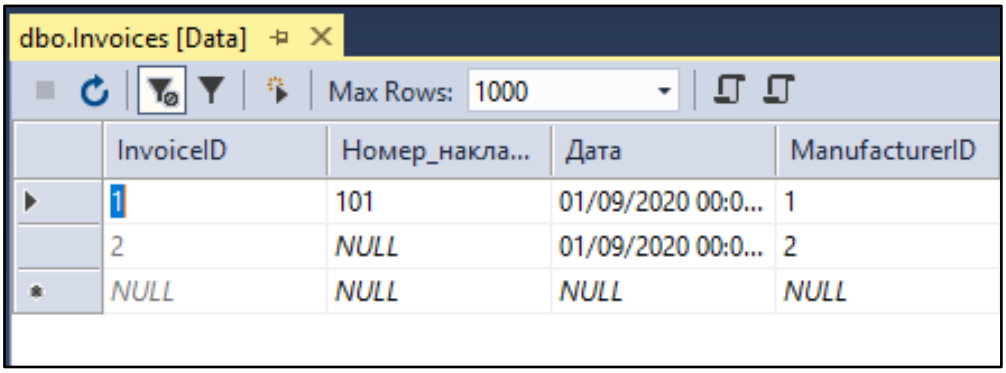


Рис. 6.14. **Дані таблиці *Invoices***

3.4. Зміни атрибутів властивості. Використання SQL

Завдання

Замініть порожні значення поля ***Номер_накладної*** у всіх рядках таблиці ***Invoices*** на значення, що обчислюють за таким виразом:

ID + "–" + YYYYMMDD,

використавши значення полів того самого рядка. У виразі прийнято такі скорочення:

ID – значення поля ***ManufacturerID***;

YYYYMMDD – рік, місяць і день у полі ***Дата***.

Зазначені величини розділено символом "мінус".

Ідеї виконання

Заміну значень можна виконати за допомогою запиту SQL, що містить такий оператор UPDATE:

```
UPDATE Invoices SET Номер_накладної =CONVERT(varchar(3),
ManufacturerID) + '-'  + CONVERT(varchar(8), Дата, 112)  WHERE
Номер_накладної IS NULL
```

В операторі використано функцію CONVERT(varchar(3), ManufacturerID), з огляду на те, що кількість виробників, які постачають товарами кіоск, менша за 1 000, чого більше ніж достатньо.

Проаналізувавши вираз, можна зробити висновок, що максимальна довжина результату може досягати 12 символів (3 символи для ***ID***, 1 символ для "мінуса" і 8 символів для дати). Тому в міграції перед викликом SQL-запиту слід змінити розмір поля ***Номер_накладної***.

Щоб міграція могла мати зворотну дію, потрібно в методі ***Down*** уставити запит:

```
UPDATE Invoices SET Номер_накладної = NULL WHERE
Номер_накладної=CONVERT(varchar(3), ManufacturerID) + '-'  +
CONVERT(varchar(8), Дата, 112)
```

який повертає попереднє значення поля ***Номер_накладної***.

Виконання

1. Відкрийте файл ***Invoice.cs***, розміщений у папці ***Models***.

2. Змініть анотацію даних, розташовану перед властивістю ***Номер_ накладної*** з ***MaxLength(6)*** на ***MaxLength(12)***. Після цього опис класу набуде такого вигляду:

```
public class Invoice
{
    public int InvoiceID { get; set; }
    [MaxLength(12)]
    public string Номер_накладної { get; set; }
    public DateTime Дата { get; set; }
    public int ManufacturerID { get; set; }
    // Властивості навігації
    public virtual Manufacturer Manufacturer { get; set; }
}
```

3. Щоб створити шаблон для міграції, яка реалізує зміни поля ***Номер_накладної***, уведіть у вікні **Package Manager Console** команду **Add-Migration ChangeInvoiceNumber.**

У папці ***Migrations*** з'явився новий клас ***ChangeInvoiceNumber***. Він містить метод ***Up*** для змінення властивості поля **Номер_накладної** в таблиці ***Invoices***, а також метод ***Down*** для відновлення попереднього значення цієї властивості.

4. Щоб замінити порожні значення поля ***Номер_накладної*** у всіх рядках таблиці ***Invoices***, додайте звертання до методу ***SQL***, розмістивши його в кінці методу ***Up*** класу ***ChangeInvoiceNumber***. Метод ***Up*** набуде такого вигляду:

```
public override void Up()
{
    AlterColumn("dbo.Invoices", "Номер_накладної", c =>
c.String(maxLength: 12));
    Sql("UPDATE Invoices SET Номер_накладної =CONVERT(varchar(3),
ManufacturerID) + '-'  + CONVERT(varchar(8), Дата, 112)  WHERE
Номер_накладної IS NULL");
}
```

Зворотну дію забезпечено методом ***Down***, який після додавання запиту набуде такого вигляду:

```
public override void Down()
{
    Sql("UPDATE Invoices SET Номер_накладної = NULL WHERE
       Номер_накладної=CONVERT(varchar(3), ManufacturerID) + '-'
       +CONVERT(varchar(8), Дата, 112)");
    AlterColumn("dbo.Invoices", "Номер_накладної", c =>
        c.String(maxLength: 6));
}
```

5. Щоб виконати міграцію і додати дані у нову таблицю, уведіть у вікні **Package Manager Console** команду **Update-Database – Verbose.**

6. Перейдіть у вікно **Server Explorer** і перегляньте схему таблиці ***Invoices*** та дані, що зберігають у ній (рис. 6.15 і 6.16).

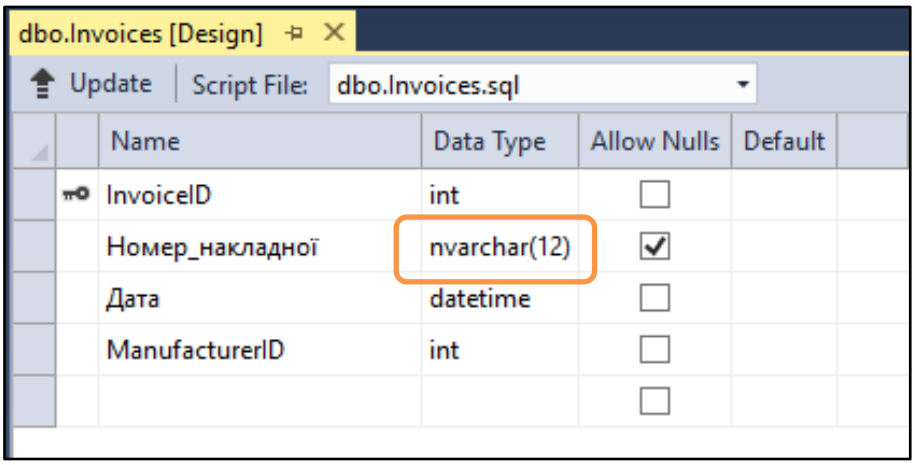


Рис. 6.15. **Схема таблиці *Invoices* зі зміненою властивістю поля *Номер_накладної***

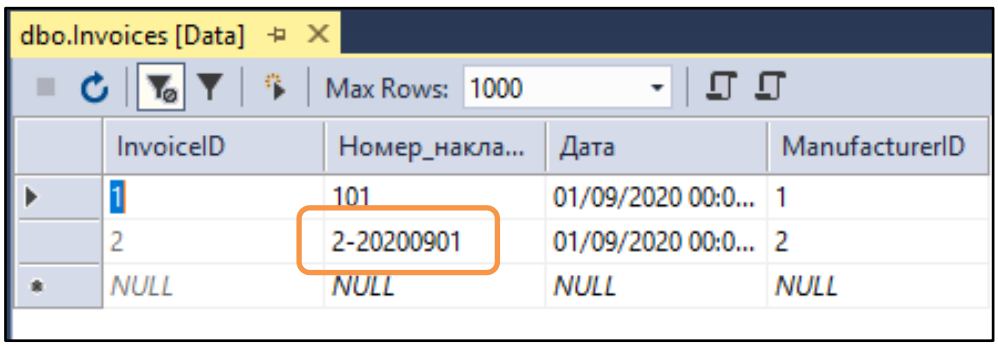


Рис. 6.16. **Дані таблиці *Invoices* зі зміненим значенням поля *Номер_накладної***

3.5. Додавання дочірньої сутності

Завдання

Додайте до моделі даних нову сутність ***InvoiceProduct,*** що є дочірньою до сутності ***Invoice***. У базі даних її реалізують у вигляді таблиці ***InvoiceProducts***, що відповідає таблиці ***ТовариНакладних*** у попередніх роботах.

Ідеї виконання

У папку ***Models*** слід додати новий клас ***InvoiceProduct***, до класів ***Invoice*** та ***Product*** – властивості навігації, а до класу ***BreadContext*** – інформацію про відповідний об'єкт класу **DbSet**. На основі цієї інформації Code First створить таблицю ***InvoiceProducts*** і встановить зв'язок із таблицями ***Invoices*** та ***Products***.

Для виконання операції створення таблиці ***InvoiceProducts*** додайте дані про товари до накладної, отримані за накладною з номером ***101*** від виробника з кодом ***1*** (сама накладна вже міститься в таблиці ***Invoices***), а також нову накладну з товарами в ній. Для цього використайте метод ***Seed***.

Виконання

1. Додайте у папку ***Models*** новий клас ***InvoiceProduct*** і введіть його опис:

```
public class InvoiceProduct
{
    public int InvoiceProductID { get; set; }
    public int InvoiceID { get; set; }
    public int ProductID { get; set; }
    public Int16 Кількість { get; set; }
    // Властивості навігації
    public virtual Product Product { get; set; }
    public virtual Invoice Invoice { get; set; }

    // Обчислювана властивість
    [NotMapped]
    public decimal Вартість
    {
        get
        {
            if (Product != null)
                return (Decimal)(Product.Ціна * Кількість);
            else
                return 0;
        }
    }
}
```

Щоб забезпечити доступ до опису класів, додайте посилання на простір імен:

```
using System.ComponentModel.DataAnnotations.Schema;
```

2. Додайте до класу ***Invoice*** властивість навігації:

```
public virtual ICollection<InvoiceProduct> InvoiceProducts {get; set;}
```

3. Повторіть п. 2 для класу ***Product***.

4. Додайте у клас ***BreadContext***, що перебуває в папці ***DataAccess***, властивість колекції нових сутностей, розташувавши її в кінці класу:

```
public DbSet<InvoiceProduct> InvoiceProducts { get; set; }
```

5. Додайте до методу ***Seed***, що перебуває у класі ***Configuration***, код для додавання даних про товари до накладних, що вже містяться в таблиці ***Invoices***, а також нову накладну з товарами в ній, розташувавши його в кінці методу:

```
// 1. Додавання товарів до накладної, отримані за накладною
// з номером 101 від виробника з кодом 1

// Дізнаємося InoviceID накладної, до якої потрібно додати товари
int inoviceID = context.Invoices.Where(
    n => n.Номер_накладної == "101" && n.ManufacturerID == 1
    ).SingleOrDefault().InvoiceID;
// Додаємотовари
var invoiceProducts = new List<InvoiceProduct>
{
// Товари накладних через їхні ID
    new InvoiceProduct
    { InvoiceID=inoviceID,
      ProductID=1, Кількість=200},
    new InvoiceProduct
    { InvoiceID=inoviceID,
      ProductID=2, Кількість=260},
};

// Перевіряємо рядки на дублювання товарів у накладній
foreach (InvoiceProduct ip in invoiceProducts)
{
```

```
    var invoiceProductInDB = context.InvoiceProducts.Where(
        nt => nt.InvoiceID == ip.InvoiceID &&
        nt.ProductID == ip.ProductID).FirstOrDefault();

    if (invoiceProductInDB == null)
    {
        context.InvoiceProducts.Add(ip);
    }
}
context.SaveChanges();

// 2. Додавання товарів до нової накладної, отримані
// від виробника з кодом 2.

invoices = new List<Invoice>
{
    new Invoice
    { Номер_накладної="102",
      Дата=DateTime.Parse("01.09.2020"),
      ManufacturerID=1,
      InvoiceProducts=new List<InvoiceProduct>
    // Товари накладних через їхні назви (LINQ)
    {
        new InvoiceProduct
        { ProductID=products.Single( p => p.Товар ==
            @"Хліб ""Український""").ProductID,
            Кількість=230},
        }
    }
};
// Перевіряємо рядки на дублювання накладних
foreach (Invoice i in invoices)
{
    var invoiceInDB = context.Invoices.Where(
        n => n.Дата== i.Дата && n.ManufacturerID == i.ManufacturerID
        ).FirstOrDefault();

    if (invoiceInDB == null)
    {
        context.Invoices.Add(i);
    }
}
context.SaveChanges();
```

6. Щоб створити шаблон для міграції, яка реалізує додавання нової сутності, уведіть у вікні **Package Manager Console** команду **Add-Migration AddInvoiceProduct.**

У папці ***Migrations*** з'явився новий клас ***AddInvoiceProduct***. Він містить метод ***Up*** для створення таблиці ***AddInvoiceProducts***, а також метод ***Down*** для її видалення.

7. Оскільки під час виконання міграції ***AddInvoiceProduct*** у методі ***Seed*** додано новий рядок до таблиці ***Invoice***, щоб забезпечити зворотність цієї операції, додайте в кінець методу ***Down*** цієї міграції оператор, що видаляє доданий рядок:

```
Sql("DELETE FROM Invoices WHERE Номер_накладної= N'102' AND Manufac-
turerID=1");
```

8. Щоб виконати міграцію і додати дані в нову таблицю, уведіть команду **Update-Database – Verbose** у вікні **Package Manager Console**. Ключ **Verbose** забезпечує виведення відповідних SQL-команд.

9. Перейдіть у вікно **Server Explorer** і перегляньте схему таблиці ***InvoiceProducts*** та дані, що зберігають у ній (рис. 6.17, 6.18).

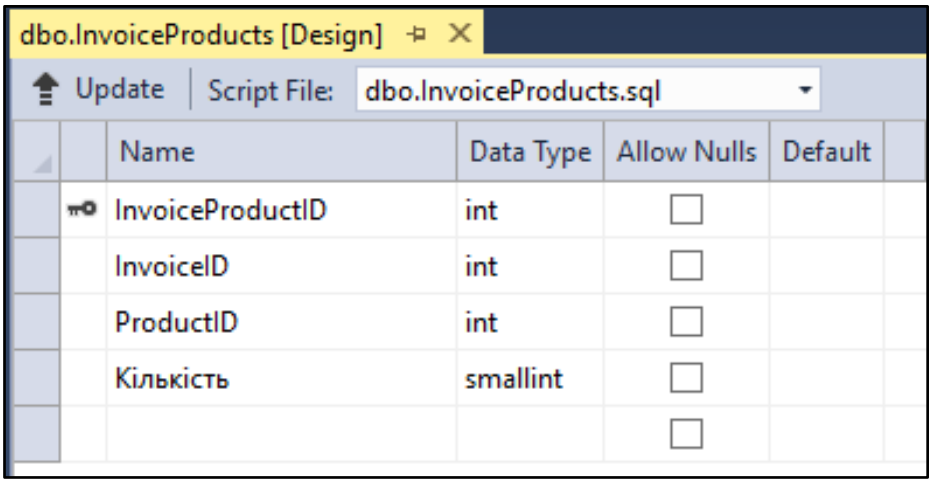


Рис. 6.17**. Схема таблиці *InvoiceProducts***

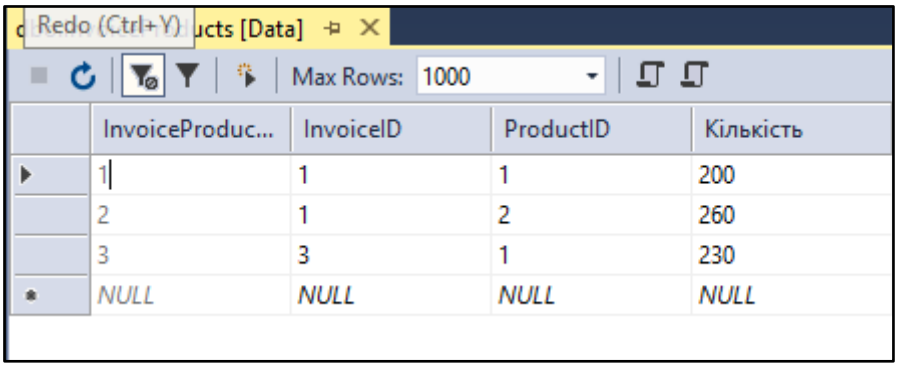


Рис. 6.18. **Дані таблиці *InvoiceProducts***

10. Із метою закріплення навичок у роботі з ієрархічними даними додайте дані про товари до накладної, отримані за накладною від виробника з кодом 2 за 01.09.2020 р. (сама накладна вже міститься в таблиці ***Invoices***). Для цього використайте метод ***Seed***.

3.6. Відкочування до заданої міграції

Завдання

Видаліть із бази даних таблицю ***InvoiceProducts***, а потім поверніть до останньої міграції.

Ідеї виконання

Оскільки таблицю ***InvoiceProducts*** додано в базу даних за допомогою міграції ***AddInvoiceProduct***, потрібно відкотити базу даних до стану, отриманого за допомогою попередньої міграції, тобто ***ChangeInvoiceNumber***. Це можна виконати за допомогою команди **Update-Database – TargetMigration: ChangeInvoiceNumber**.

Виконання

1. Перейдіть у вікно **Package Manager Console** та введіть команду **Update-Database – TargetMigration: ChangeInvoiceNumber.**
2. Перейдіть у вікно **Server Explorer**, обновіть таблиці бази даних і впевніться, що з бази даних зникла таблиця ***InvoiceProducts***.
3. Щоб повернутися до останньої міграції, уведіть у вікні **Package Manager Console** команду **Update-Database – TargetMigration: AddInvoiceProduct.**
4. Перейдіть у вікно **Server Explorer**, обновіть таблиці бази даних і впевніться, що в базі даних знову з'явилася таблиця ***InvoiceProducts*** із даними.

3.7. Отримання скрипту SQL

Завдання

Створіть SQL-скрипт, за допомогою якого можна побудувати нову базу даних ***Хліб*** на іншому комп'ютері.

Ідеї виконання

Code First дозволяє будувати SQL-скрипти, які реалізують зміни в базі даних, що виконують послідовністю міграцій. Для цього використовують

команду **Update-Database** з параметром – **Script**, указавши початкову та кінцеву міграції за допомогою параметрів – **SourceMigration** та – **TargetMigration**, відповідно.

Виконання

1. Перейдіть у вікно **Package Manager Console** і введіть команду **Update-Database – Script – SourceMigration: $InitialDatabase – TargetMigration: AddInvoiceProduct.**

2. Перейдіть у вікно нового запиту і видаліть всі оператори SQL, що стосуються ***MigrationHistory***. Отримаємо SQL-скрипт, за допомогою якого можна побудувати нову БД ***Хліб***:

```
    CREATE TABLE [dbo].[Manufacturers] (
        [ManufacturerID] [int] NOT NULL IDENTITY,
        [Виробник] [nvarchar](20) NOT NULL,
        [Адреса] [nvarchar](30) NOT NULL,
        [Телефон] [nvarchar](15) NOT NULL,
        [Контактна_особа] [nvarchar](20) NOT NULL,
        CONSTRAINT [PK_dbo.Manufacturers] PRIMARY KEY
([ManufacturerID])
    )
    CREATE TABLE [dbo].[Sales] (
        [SaleID] [int] NOT NULL IDENTITY,
        [Дата] [datetime] NOT NULL,
        [ProductID] [int] NOT NULL,
        [ManufacturerID] [int] NOT NULL,
        [Кількість] [smallint] NOT NULL,
        CONSTRAINT [PK_dbo.Sales] PRIMARY KEY ([SaleID])
    )
    CREATE INDEX [IX_ManufacturerID] ON [dbo].[Sales]([ManufacturerID])
    CREATE INDEX [IX_ProductID] ON [dbo].[Sales]([ProductID])
    CREATE TABLE [dbo].[Products] (
        [ProductID] [int] NOT NULL IDENTITY,
        [Товар] [nvarchar](25) NOT NULL,
        [Ціна] [decimal](18, 2),
        [Ціна_закупівлі] [decimal](18, 2) NOT NULL,
        CONSTRAINT [PK_dbo.Products] PRIMARY KEY ([ProductID])
    )
    ALTER TABLE [dbo].[Sales] ADD CONSTRAINT
[FK_dbo.Sales_dbo.Manufacturers_ManufacturerID] FOREIGN KEY ([Manufac-
turerID]) REFERENCES [dbo].[Manufacturers] ([ManufacturerID]) ON DELETE
CASCADE
    ALTER TABLE [dbo].[Sales] ADD CONSTRAINT
[FK_dbo.Sales_dbo.Products_ProductID] FOREIGN KEY ([ProductID])
REFERENCES [dbo].[Products] ([ProductID]) ON DELETE CASCADE
    CREATE TABLE [dbo].[Invoices] (
```

```
        [InvoiceID] [int] NOT NULL IDENTITY,
        [Номер_накладної] [nvarchar](6),
        [Дата] [datetime] NOT NULL,
        [ManufacturerID] [int] NOT NULL,
        CONSTRAINT [PK_dbo.Invoices] PRIMARY KEY ([InvoiceID])
    )
    CREATE INDEX [IX_ManufacturerID] ON
[dbo].[Invoices]([ManufacturerID])
    ALTER TABLE [dbo].[Invoices] ADD CONSTRAINT
[FK_dbo.Invoices_dbo.Manufacturers_ManufacturerID] FOREIGN KEY ([Manu-
facturerID]) REFERENCES [dbo].[Manufacturers] ([ManufacturerID]) ON
DELETE CASCADE
    ALTER TABLE [dbo].[Invoices] ALTER COLUMN [Номер_накладної] [nvar-
char](12)
    UPDATE Invoices SET Номер_накладної =CONVERT(varchar(3), Manufac-
turerID) +         '-'  + CONVERT(varchar(8), Дата, 112)  WHERE Но-
мер_накладної IS NULL
    CREATE TABLE [dbo].[InvoiceProducts] (
        [InvoiceProductID] [int] NOT NULL IDENTITY,
        [InvoiceID] [int] NOT NULL,
        [ProductID] [int] NOT NULL,
        [Кількість] [smallint] NOT NULL,
        CONSTRAINT [PK_dbo.InvoiceProducts] PRIMARY KEY ([InvoicePro-
ductID])
    )
    CREATE INDEX [IX_InvoiceID] ON [dbo].[InvoiceProducts]([InvoiceID])
    CREATE INDEX [IX_ProductID] ON [dbo].[InvoiceProducts]([ProductID])
    ALTER TABLE [dbo].[InvoiceProducts] ADD CONSTRAINT
[FK_dbo.InvoiceProducts_dbo.Invoices_InvoiceID] FOREIGN KEY
([InvoiceID]) REFERENCES [dbo].[Invoices] ([InvoiceID]) ON DELETE
CASCADE
    ALTER TABLE [dbo].[InvoiceProducts] ADD CONSTRAINT
[FK_dbo.InvoiceProducts_dbo.Products_ProductID] FOREIGN KEY
([ProductID]) REFERENCES [dbo].[Products] ([ProductID]) ON DELETE
CASCADE
```

3. Збережіть SQL-скрипт у файлі ***SqlQuery_Хліб*** у папці рішення.

3.8. Автоматичне оновлення бази даних під час запуску застосунку

Завдання

Створіть нову базу даних з урахуванням усіх міграцій.

Ідеї виконання

За допомогою тих міграцій, що містяться в застосунку, можна створити нову базу даних на рівні останньої міграції. Для цього потрібно задати

нове ім'я бази даних у рядку ConnectionString, потім зареєструвати ініціалізатор бази даних **MigrateDatabaseToLatestVersion** і запустити застосунок на виконання. Водночас у застосунку має бути хоч одна операція з базою даних, наприклад, читання даних.

Виконання

1. Відкрийте файл **App.config** і замініть значення параметра **Initial Catalog** у тегу **ConnectionStrings** із ***ХлібПрізвище.mdf*** на ***ХлібПрізвище_1.mdf***. Потім закрийте файл **App.config** зі збереженням зроблених змін.

2. Відкрийте файл **Program.cs** і в методі **Main** закоментуйте всі оператори, що входять до тіла гілки **try**, а під ними вставте такі:

```
// Реєстрування ініціалізатора бази даних
Database.SetInitializer(new
    MigrateDatabaseToLatestVersion<BreadContext, Configuration>());
using (var context = new BreadContext())
{
    foreach (var p in context.Products)
    {
        Console.WriteLine(p.Товар);
    }
}
```

У результаті метод **Main** набуде такого вигляду:

```
static void Main(string[] args)
{
    try
    {
        /*
        // Стратегія роботи з базою даних
        Database.SetInitializer(
                new DropCreateDatabaseAlways<BreadContext>());
        int nProduct;
        int nManufacturer;
        int nSale;
        DataAdd.FillDB(out nProduct, out nManufacturer,
        out nSale);
        Console.WriteLine(
            "Базу даних на SQL Server створено і заповнено.\n"
            + "У таблиці записано таку кількість рядків:\n"
```

```
            + "Products - " + nProduct
            + ", Manufacturers - " + nManufacturer
            + ",  Sales - " + nSale
            + ".\n Перевірте!!!");
        */
        // Реєстрування ініціалізатора бази даних
        Database.SetInitializer(new
      MigrateDatabaseToLatestVersion<BreadContext, Configuration>());
        using (var context = new BreadContext())
        {
            foreach (var p in context.Products)
            {
                Console.WriteLine(p.Товар);
            }
        }
    }
    catch (System.Exception ex)
    {
        Console.WriteLine("Базу даних не створено. \n Помилка:\n "
            + ex.ToString());
    }
    Console.WriteLine("Натисніть будь-яку клавішу, щоб вийти...");
    Console.ReadKey();
}
```

3. Щоб забезпечити доступ до опису класів, додайте посилання на простори імен:

```
using моделювання.Migrations;
using моделювання.Models;
```

4. Запустіть проєкт на виконання. Дочекайтеся закінчення операцій із базою даних і завершіть виконання проєкту.

5. Переконайтеся, що в папці проєкту ***моделювання*** з'явилися два нові файли – ***ХлібПрізвище_1.mdf*** (база даних) і ***ХлібПрізвище_1_log.ldf*** (журнал бази даних).

6. Установіть під'єднання файлу нової бази даних ***ХлібПрізвище_1***, що перебуває у папці проєкту ***моделювання***. Для цього скористайтеся командою **Add Connection** із контекстового меню значка **Data Connections**, розташованого у вікні **Server Explorer**, і перевірте, чи встановлено значення **Microsoft SQL Server Database File (SqlClient)** у полі **Data source**

вікна **Add Connection**. Ознайомтеся із властивостями стовпців її таблиць і даними, що містяться в таблицях.

7. Закрийте вікно проєкту зі збереженням зроблених змін.

У звіті з лабораторної роботи подайте команди, які виконували для кожного завдання з розд. 3, та опишіть визначені результати після їхнього виконання.

4. Побудова застосунку

4.1. Додавання проєкту Windows Forms

Завдання

Додайте до рішення ***codeFirstХліб*** новий проєкт Windows Forms з ім'ям ***winForms*** і створіть у ньому кнопкову форму для керування основними функціями проєкту.

Виконання

1. Скопіюйте папку ***codeFirstХліб_Migration*** і новій папці дайте ім'я ***codeFirstХліб_App***, щоб під час захисту лабораторної роботи можна було продемонструвати попередній варіант проєкту та подальший його розвиток.

2. Відкрийте рішення ***codeFirstХліб***, додайте до нього новий проєкт Windows Forms з ім'ям ***winForms***.

3. Установіть покажчик мишки на імені нового проєкту і з контекстового меню виберіть команду **Set as StartUp Project**.

4. Дізнайтеся шлях до бібліотеки **EntityFramework.dll**. Для цього розкрийте вузол **References** проєкту ***моделювання*** і запам'ятайте значення властивості **Path** для елемента **EntityFramework.dll**. Він знадобиться під час виконання наступного пункту. Особливу увагу зверніть на те, яку папку зазначено всередині папки **lib** – **net40** чи **net45**. Саме її слід вибрати в наступному пункті.

5. Установіть покажчик мишки на значок **References** нового проєкту у вікні **Solution Explorer**, із контекстового меню виберіть команду **Add Reference**, а потім додайте такі посилання (рис. 6.19):

проєкт ***моделювання*** (вкладка **Projects**);

System.Data.Entity (вкладка **NET**, для Visual Studio 2017 – **Assemblies**);

EntityFramework, **EntityFramework.SqlServer** (вкладка **Browse**, шлях \packages\EntityFramework.***x.x.x***\lib\net4**Y** у папці рішення, де **Y** – 0 чи 5, залежно від знайденого значення в п. 4). Для встановлення посилань використайте кнопку **Browse**.

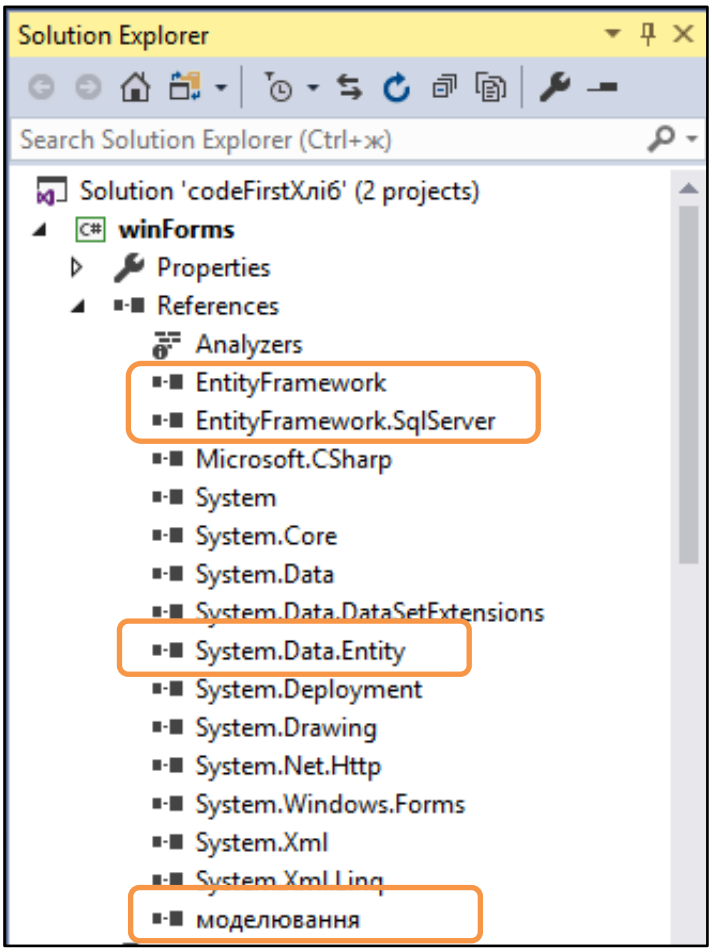


Рис. 6.19. **Нові посилання у проєкті *winForms***

6. Скопіюйте файл **App.config** із проєкту ***моделювання*** у ***winForms***. Для цього перетягніть його значок із першого проєкту у другий. Відкрийте цей файл і приберіть символи ***_1*** в імені файлу, який перебуває в параметрі **connectionString** тегу **connectionStrings**, щоб повернутися до попередньої бази даних.

7. Перейменуйте файл ***Form1.cs*** у проєкті ***winForms*** на ***formХліб.cs*** і для властивості **Text** цієї форми задайте значення ***Хліб***.

8. Додайте елементи керування на форму ***Хліб***, щоб вона набула вигляду, поданого на рис. 6.20. Водночас установіть значення властивостей кнопок, наведені в табл. 6.4.

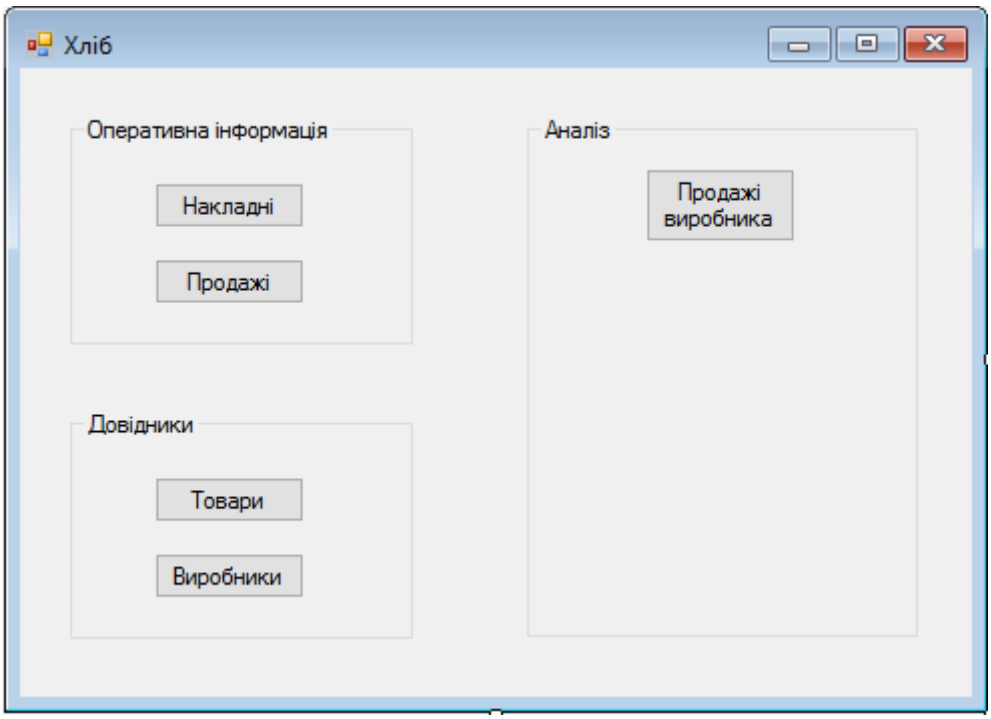


Рис. 6.20. **Форма *Хліб***

Таблиця 6.4

Властивості кнопок

Групи	Кнопки	Властивості	Значення
Оперативна інформація	1	Text	Накладні
		Name	buttonНакладні
	2	Text	Продажі
		Name	buttonПродажі
Довідники	1	Text	Товари
		Name	buttonТовари
	2	Text	Виробники
		Name	buttonВиробники
Аналіз	1	Text	Продажі виробника
		Name	buttonПродажіВиробника

9. Збережіть зміни, зроблені в рішенні.

4.2. Безпосереднє прив'язування сутностей

Завдання 1

Створіть форму ***Товари*** для виконання операцій CRUD із таблицею ***Products*** у базі даних ***Хліб*** (рис. 6.21).

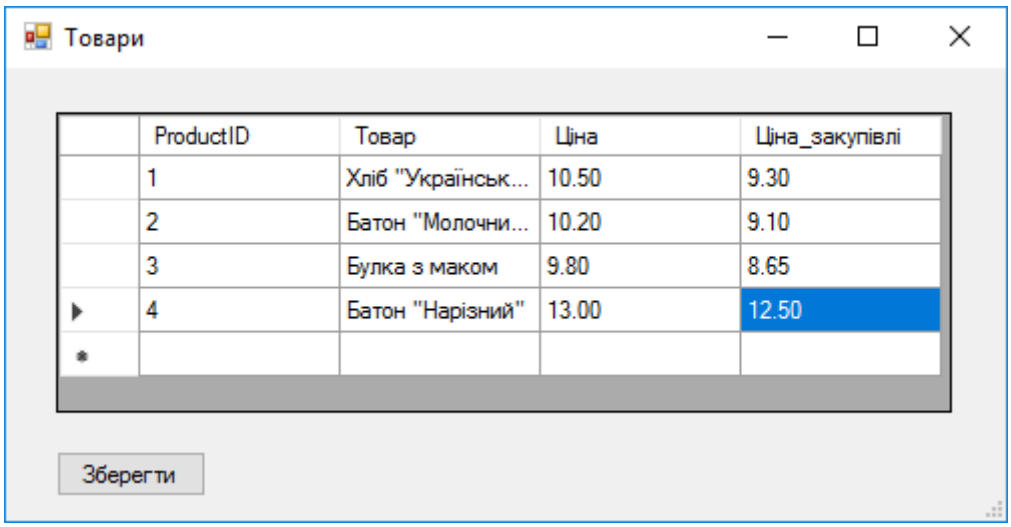


Рис. 6.21. **Форма *Товари***

Ідеї виконання

Оскільки таблиця ***Products*** є батьківською і довідниковою, для виконання операцій CRUD із нею достатньо використати елемент керування DataGridView, а для фіксування зроблених змін – кнопку ***Зберегти***.

Для відображення даних у DataGridView потрібно використати такі методи та властивості:

Load – для додавання сутності до контексту;

Local – для подання всіх доданих, модифікованих і незмінених об'єктів у локальному наборі;

ToBindingList – для створення колекції рядків, що прив'язують до інтерфейсу, які синхронізуються з даними Observable Collection.

Для фіксування змін із даними, виконаними користувачем, застосовують такі методи:

SaveChanges – для збереження в базі даних усіх змін, зроблених у контексті;

Refresh – для оновлення даних, що відображають в елементі DataGridView.

Виконання

1. Додайте до проєкту ***winForms*** нову форму з ім'ям файлу ***formТовари.cs*** і значенням ***Товари*** для властивості **Text**.

2. Додайте на форму елемент **DataGridView** з ім'ям ***gvТовари*** для відображення набору сутностей ***Products*** (рис. 6.22).

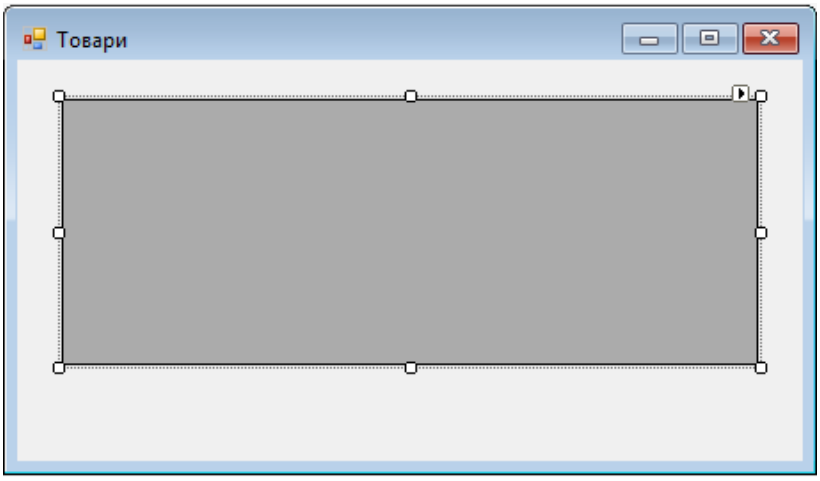


Рис. 6.22. **Елемент DataGridView на формі *Товари***

3. Двічі клацніть у вільному місці форми ***Товари*** й у вікні коду форми введіть оператори тіла оброблювача події ***formТовари_Load***. Він має такий вигляд:

```
BreadContext context;
private void formТовари_Load(object sender, EventArgs e)
{
    // Створюємо екземпляр класу DbContext
    context = new BreadContext();
    // Додаємо колекцію сутностей до контексту
    context.Products.Load();
    // Прив'язуємо набір сутностей до елемента DataGridView
    gvТовари.DataSource = context.Products.Local.ToBindingList();
    //Вилучаємо властивості навігації з інтерфейсу
    gvТовари.Columns.Remove("InvoiceProducts");
    gvТовари.Columns.Remove("Sales");
}
```

У розділ опису просторів імен додайте ще й такі:

```
using System.Data.Entity;
using моделювання.DataAccess;
```

4. Перейдіть у вікно конструктора форми ***formХліб***, двічі клацніть кнопку ***Товари*** й у вікні коду введіть код оброблювача події "Клацання кнопки Товари".

```
private void buttonТовари_Click(object sender, EventArgs e)
{
    formТовари вікноТовари = new formТовари();
    вікноТовари.ShowDialog();
}
```

5. Перевірте функціональність форми. Після її завантаження має бути відображено всі дані таблиці ***Products***.

6. Для збереження змін у базі даних додайте на форму ***formТовари*** кнопку ***Зберегти*** та код її оброблювача.

```
private void buttonЗберегти_Click(object sender, EventArgs e)
{
    context.SaveChanges();
    gvТовари.Refresh();
}
```

7. Перевірте функціональність кнопки ***Зберегти***, додавши дані про товар та зберігши їх у базі даних. Потрібно ввести такі дані, що наведені в табл. 6.5.

Таблиця 6.5

Дані про новий товар

Товар	Ціна	Ціна_закупівлі
Батон "Нарізний"	13.00	12.50

Поясніть дію методу **Refresh** на цій формі.

8. Збережіть зміни, зроблені у проєкті.

Завдання 2

Додайте до проєкту ***winForms*** форму ***Виробники***, у якій будуть відображати й змінювати дані таблиці ***Manufacturers*** (рис. 6.23).

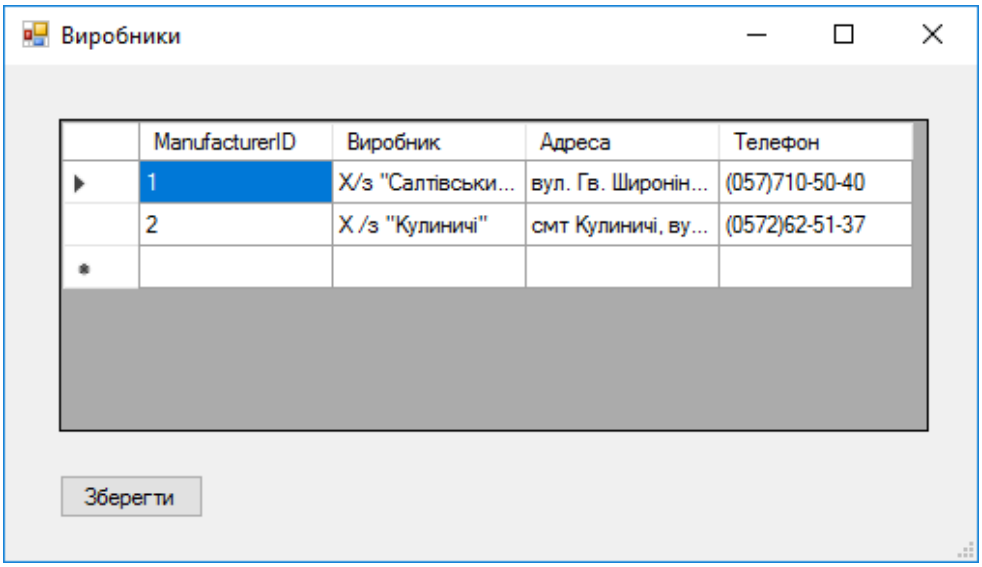


Рис. 6.23. **Форма *Виробники***

Виконання

Оскільки форму ***Виробники*** будують аналогічно формі ***Товари***, створіть її самостійно.

4.3. Візуальні засоби побудови інтерфейсу

Завдання

Створіть форму ***Продажі*** для виконання операцій CRUD із таблицею ***Sales*** у базі даних ***Хліб*** (рис. 6.24).

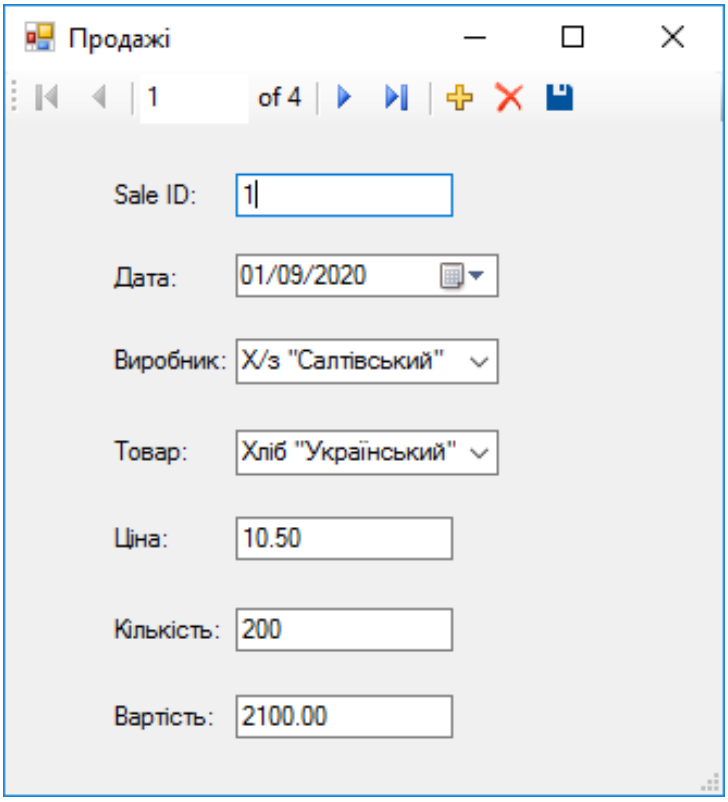


Рис. 6.24. **Форма *Продажі***

Ідеї виконання

Для побудови форми слід застосувати візуальні засоби на основі вікна **Data Sources**. Побудову форми виконують у 2 етапи:

1. Додавання джерела даних на основі об'єкта.
2. Створення форми візуальними засобами.

У результаті виконання цих етапів буде отримано форму ***Продажі***, у якій можна виконувати CRUD-операції з даними сутності ***Sale***.

Оскільки таблиця ***Sales*** містить поля ***ProductID*** та ***ManufacturerID*** для зв'язку з відповідними батьківськими таблицями, на формі ***Продажі*** слід використати поля підстановки у вигляді елементів ComboBox.

Завдання 1

Додайте в застосунок джерела даних на основі моделі даних.

Виконання

1. Викличте майстра налаштування джерела даних, вибравши команду **Add New Data Source**, що перебуває в меню **Data** у Visual Studio 2010, а в старших версіях – у меню **Project**.

2. Виберіть значок **Object** у вікні **Choose a Data Source Type** і клацніть кнопку **Next** (рис. 6.25).

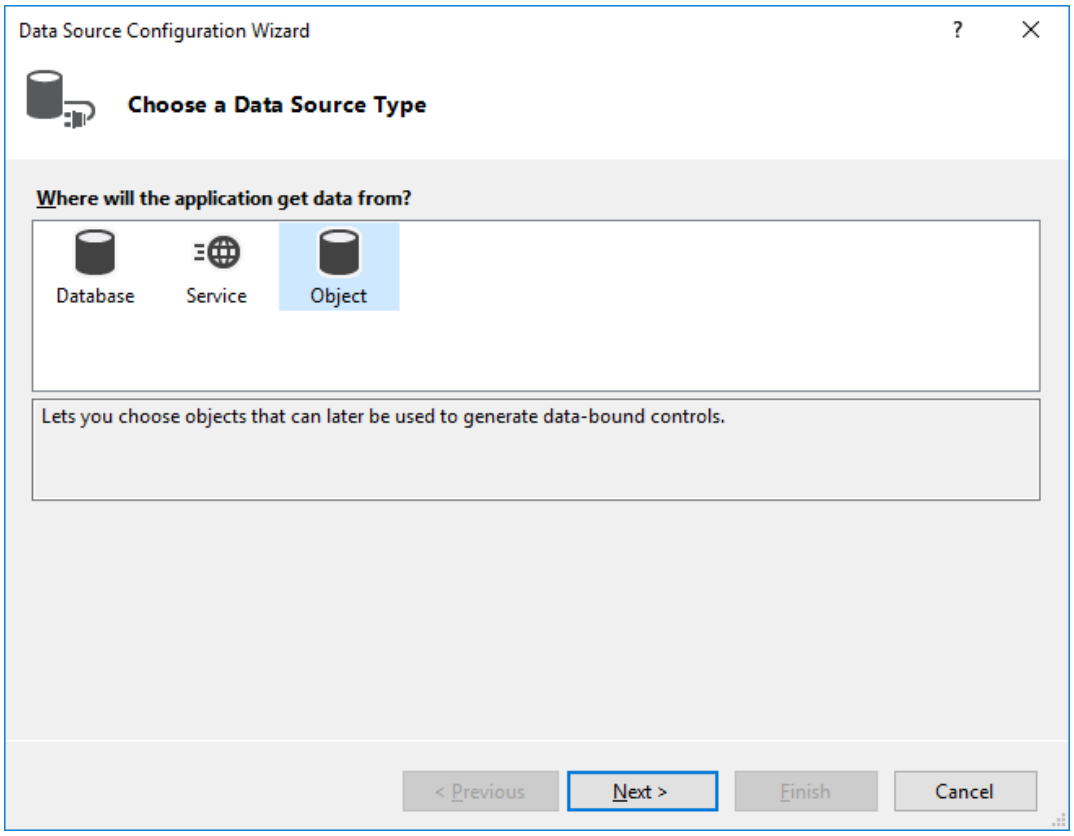


Рис. 6.25. **Вікно *Choose a Data Source Type***

3. Розкрийте вузол ***моделювання*** та виберіть сутності ***Invoice***, ***InvoiceProduct***, ***Manufacturer***, ***Product*** і ***Sale*** з підвузла ***моделювання.Models***, дані яких знадобляться для відображення на формах, у вікні **Select the Data Objects** та клацніть кнопку **Finish** (рис. 6.26).

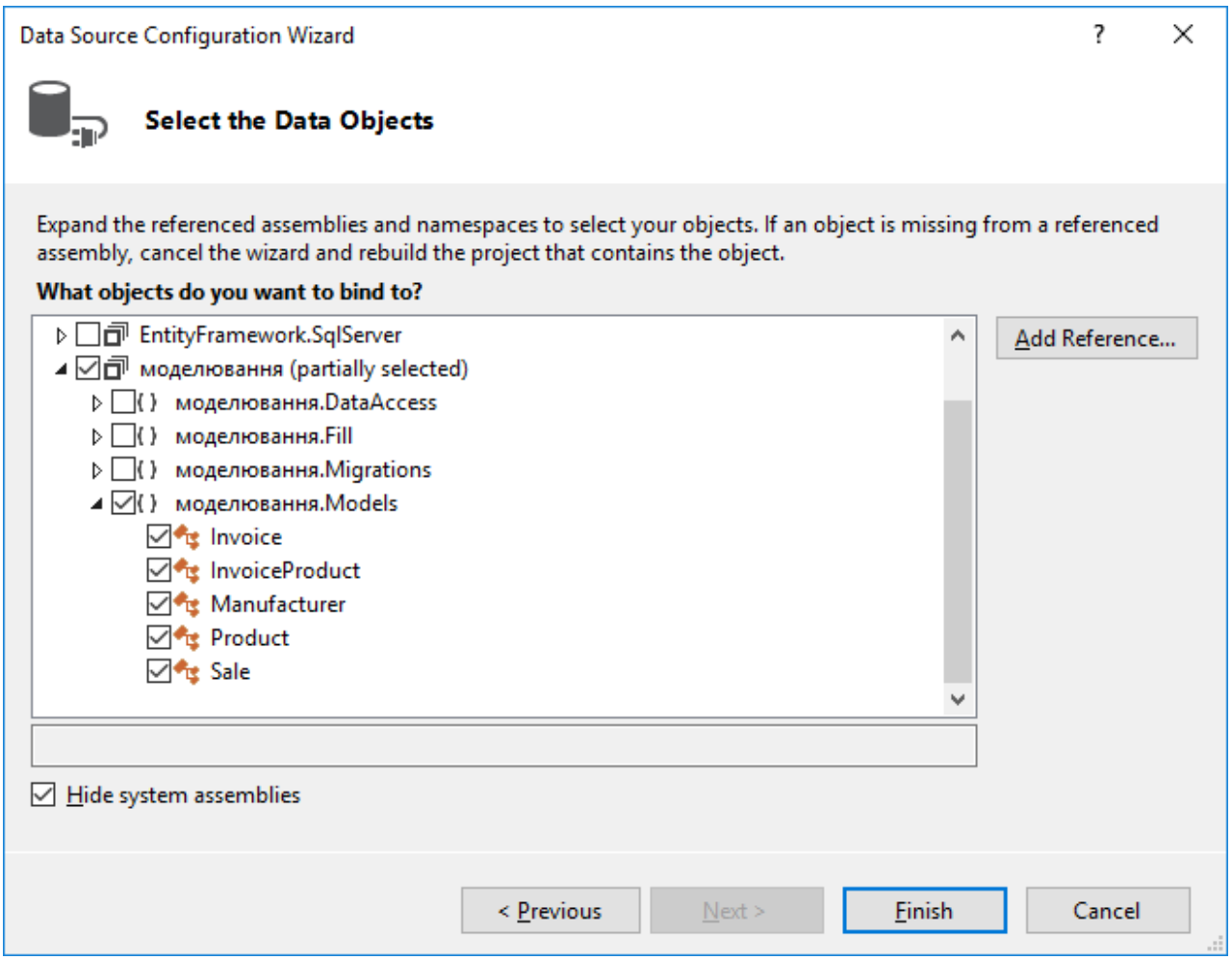


Рис. 6.26. **Вікно *Select the DataObjects***

4. Відкрийте вікно **DataSources** командою **Data – Show DataSources** у Visual Studio 2010, а у старших версіях – у меню **View – Other Windows – Data Sources**.

У результаті роботи майстра у вікні **Data Sources** з'явилося зображення вибраних сутностей. Кожну сутність тут подано вузлом у вигляді значка. Елементами вузла є імена властивостей. Якщо сутність має асоціацію з іншою сутністю, то останню відображено у вигляді підвузла. На рис. 6.27 у вузлі сутності ***Sale*** відображено її властивості та дві батьківські сутності – ***Manufacturer*** і ***Product***. Сутність ***Product*** будуть використовувати під час побудови форми для відображення ціни товару. Сутності ***Manufacturer*** і ***Product***, подані вузлами першого рівня, будуть використовувати під час побудови відповідних елементів ComboBox для заповнення їх даними.

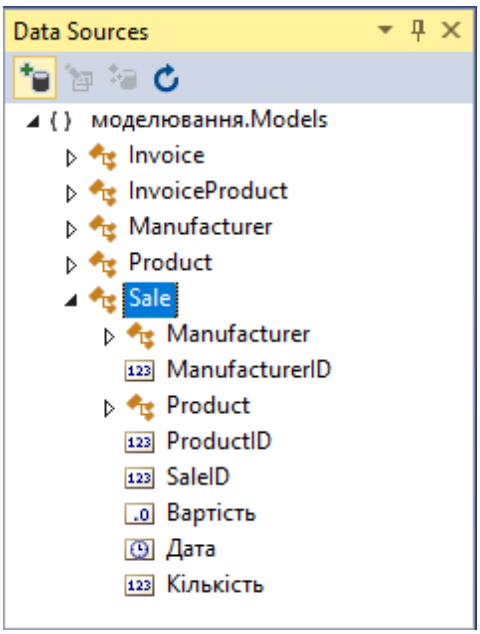


Рис. 6.27. **Вікно *Data Sources***

Завдання 2

Створіть форму ***Продажі*** з використанням візуальних засобів (рис. 6.28).

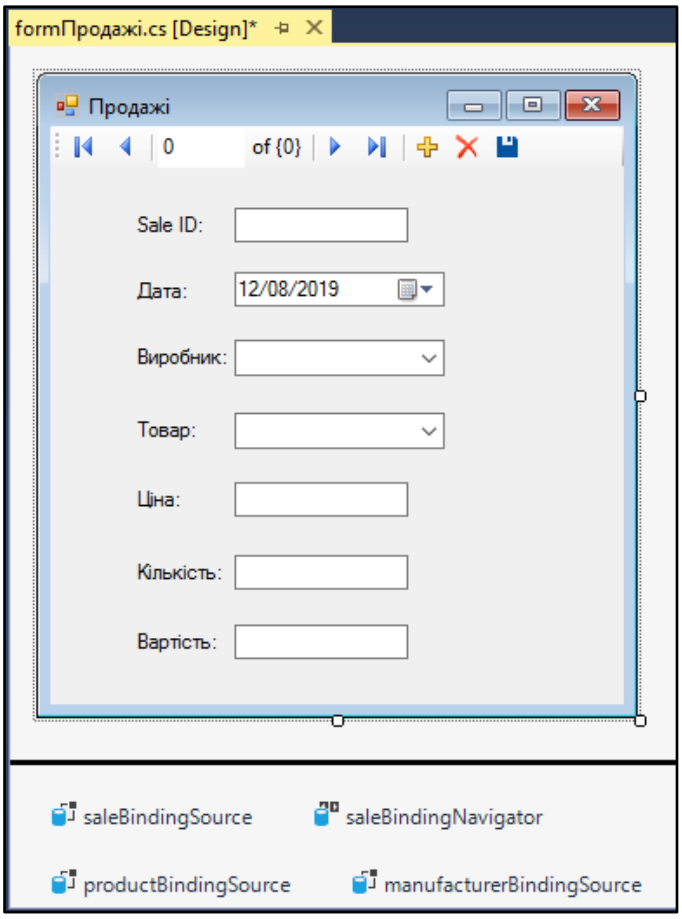


Рис. 6.28. **Форма *Продажі* в конструкторі**

Виконання

1. Додайте до проєкту ***winForms*** нову форму з ім'ям файлу ***formПродажі.cs*** і значенням ***Продажі*** для властивості **Text**.

2. Клацніть вузол сутності ***Sale*** у вікні **Data Sources** і з її списку, що розкривається, виберіть подання **Details**.

3. Виберіть подання **ComboBox** у вузлі ***Sale*** для властивості ***ManufacturerID*** у списку, що розкривається.

4. Повторіть п. 3 для властивості ***ProductID***.

5. Перетягніть із вузла ***Sale*** на форму ***Продажі*** по черзі такі властивості: SaleID, Дата, ManufacturerID, ProductID, Product.Ціна, Кількість, Вартість.

На формі з'явилися елементи керування для відображення даних і панель навігатора, а в області компонентів – ряд об'єктів, які забезпечують роботу з даними сутності (рис. 6.29). Ознайомтеся з ними та визначте призначення кожного.

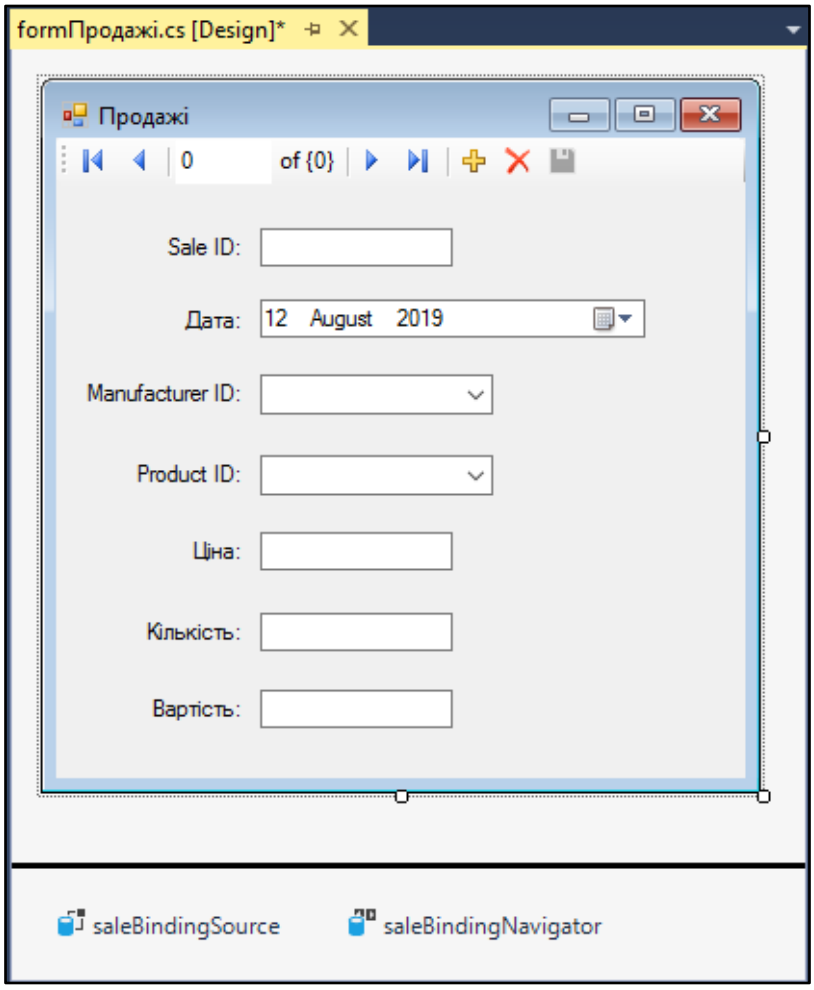


Рис. 6.29. **Вікно форми *Продажі* після додавання сутності *Sale***

6. Клацніть елемент DateTimePicker і для його властивості **Format** установіть значення ***Short***.

7. Змініть властивість **Text** для написів ***ProductID*** і ***ManufacturerID***, установивши нові значення ***Товар*** і ***Виробник***, відповідно. Потім вирівняйте всі елементи Label по лівому краю.

8. Щоб в елементі ComboBox для товару відображалися назви товарів, перетягніть вузол сутності ***Product*** із вікна ***Data Sources*** на цей ComboBox і відпустіть. Потім для елемента ComboBox ***Товар*** установіть значення властивостей, наведені в табл. 6.6.

Таблиця 6.6

Властивості елемента *Товар*

Властивості	Значення
DataSource	productBindingSource
DisplayMember	Товар
ValueMember	ProductID
SelectedValue	saleBindingSource – ProductID

Примітка. Значення цих чотирьох властивостей краще встановлювати у вікні завдань ComboBox (рис. 6.30).

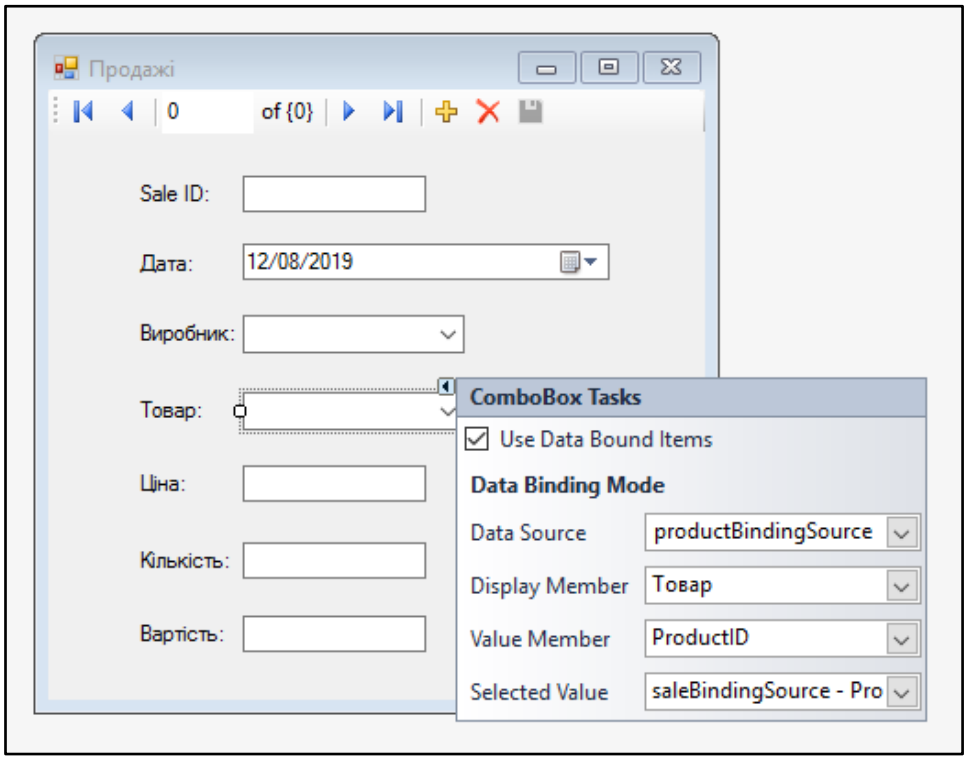


Рис. 6.30**. Установлення значень властивостей прив'язування для елемента ComboBox**

9. Повторіть п. 8 для ComboBox ***Виробник***.

Примітка. В область компонентів форми автоматично додалися компоненти BindingSource для сутностей ***Product*** і ***Manufacturer***.

10. Двічі клацніть у вільному місці форми ***Продажі*** й у вікні коду форми введіть оператори тіла оброблювача події ***formПродажі_Load***. Він має такий вигляд:

```
BreadContext context;

private void formПродажі_Load(object sender, EventArgs e)
{
    // Створюємо екземпляр класу DbContext
    context = new BreadContext();

    // Завантажуємо дані для productBindingSource
    // та manufacturerBindingSource
    productBindingSource.DataSource = context.Products.ToList();
    manufacturerBindingSource.DataSource =
        context.Manufacturers.ToList();

    // Завантажуємо дані для saleBindingSource
    context.Sales.Load();

    saleBindingSource.DataSource =
        context.Sales.Local.ToBindingList();
}
```

У розділ опису просторів імен додайте ще й такі:

```
using System.Data.Entity;
using моделювання.DataAccess;
```

11. Перейдіть у вікно конструктора форми ***Продажі***, клацніть правою клавішею мишки на кнопці ***Зберегти***, розташованій в елементі ***saleBindingNavigator***, і виберіть значення **Enable** з контекстового меню. Потім додайте код оброблювача події "Клацання кнопки Зберегти".

```
private void saleBindingNavigatorSaveItem_Click(object sender,
    EventArgs e)
{
    saleBindingSource.EndEdit();
    context.SaveChanges();
}
```

12. Перейдіть у вікно конструктора форми ***Хліб*** і додайте код оброблювача події "Клацання кнопки Продажі":

```
private void buttonПродажі_Click(object sender, EventArgs e)
{
    formПродажі вікноПродажі = new formПродажі();
    вікноПродажі.ShowDialog();
}
```

13. Запустіть програму на виконання й перевірте функціональність форми ***Продажі***, додавши дані про продаж одного товару та зберігши їх (рис. 6.31).

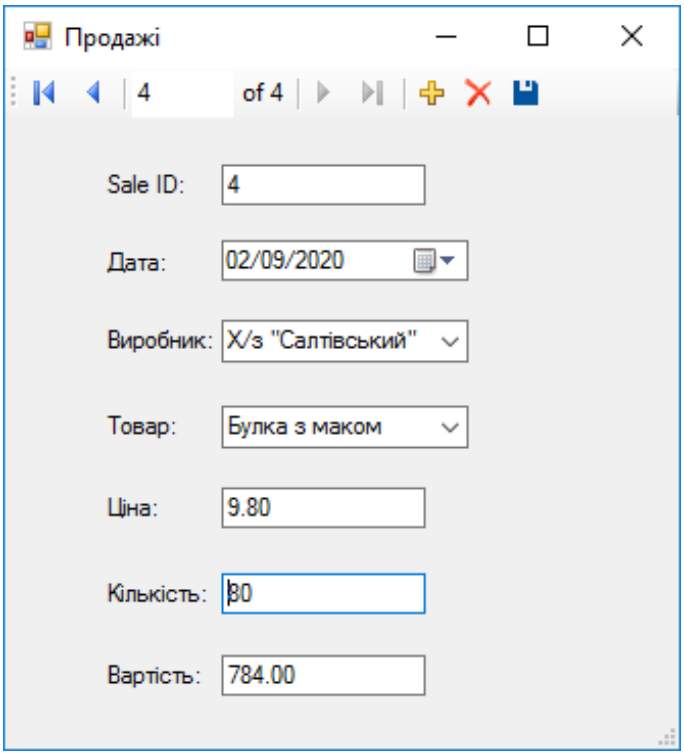


Рис. 6.31. **Додавання даних про продаж одного товару**

14. Закрийте форми ***Продажі*** та ***Хліб*** і збережіть зміни, зроблені у проєкті.

4.4. Робота з ієрархічними сутностями

Завдання

Додайте в застосунок форму ***Накладні***, у якій будуть відображати й змінюватися дані сутностей ***Invoice*** та ***InvoiceProduct***, пов'язані відношенням "один-до-багатьох" (рис. 6.32).

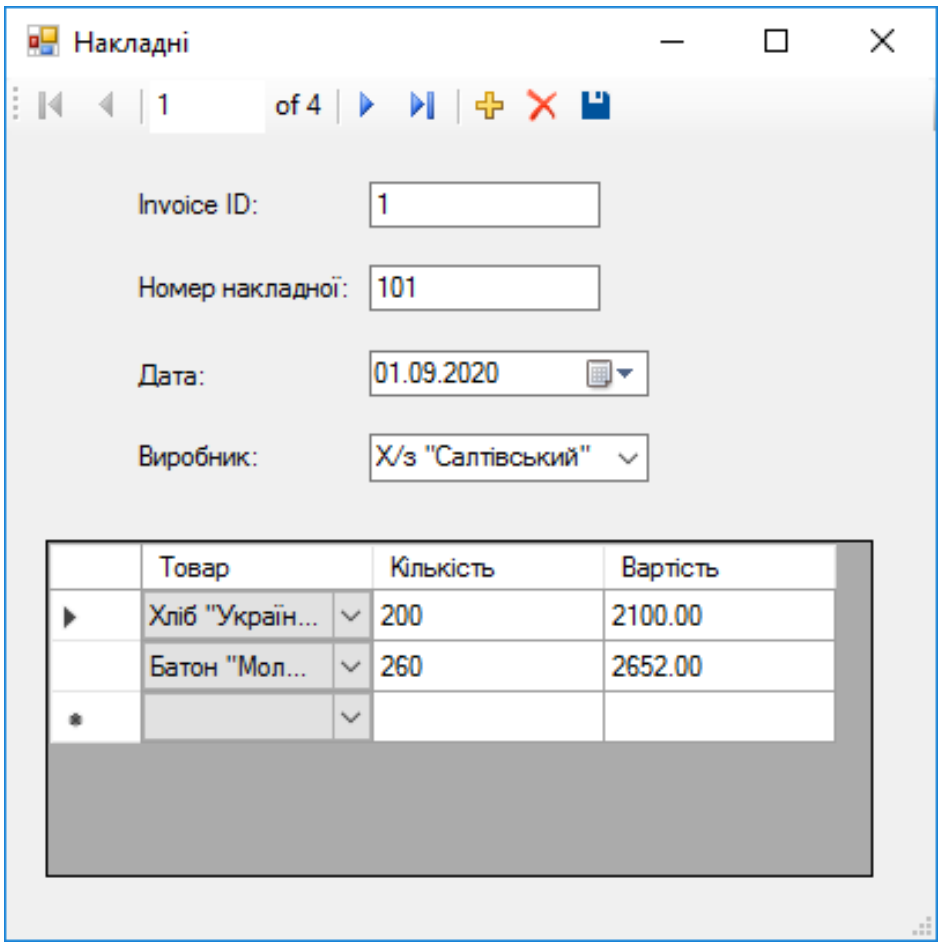


Рис. 6.32. **Форма *Накладні***

Ідеї виконання

Оскільки сутності ***Invoice*** та ***InvoiceProduct*** пов'язано асоціацією "один-до-багатьох", то, із метою забезпечення двостороннього прив'язування та сортування, необхідно розширити Observable Collection для додавання функціональності IListSource. Варто нагадати, що інтерфейс IListSource надає об'єкту функціональні можливості, що дозволяють повертати список, який може бути пов'язаним із джерелом даних. Це потребує внесення змін до батьківського класу ***Invoice*** для опису пов'язаної колекції сутностей ***InvoiceProducts***. Тому виконання завдання складається із двох етапів:

1. Реалізація IListSource для колекцій.
2. Створення форми.

У результаті виконання цих етапів буде отримано форму ***Накладні***, у якій можна переглядати, додавати, видаляти та змінювати дані ієрархічних сутностей ***Invoice*** та ***InvoiceProduct***.

Завдання 1

Реалізуйте інтерфейс IListSource для колекцій сутностей ***Invoices***.

Виконання

1. Додайте в папку ***Models*** проєкту ***моделювання*** новий клас ***ObservableListSource*** і введіть його опис:

```
using System.Collections;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.ComponentModel;
using System.Diagnostics.CodeAnalysis;
using System.Data.Entity;

namespace моделювання.Models
{
    public class ObservableListSource<T> : ObservableCollection<T>,
        IListSource where T : class
    {
        private IBindingList _bindingList;

        bool IListSource.ContainsListCollection {get {return false; }}

        IList IListSource.GetList()
        {
           return _bindingList ?? (_bindingList=this.ToBindingList());
        }
    }
}
```

2. Відкрийте клас ***Invoice***, що перебуває в папці ***Models*** проєкту ***моделювання***, закоментуйте останній рядок опису класу, що стосується колекції ***InvoiceProducts***, і додайте після нього такі рядки:

```
private readonly ObservableListSource<InvoiceProduct>
    invoiceProducts = new ObservableListSource<InvoiceProduct>();
public virtual ObservableListSource<InvoiceProduct>
    InvoiceProducts { get { return invoiceProducts; } }
```

Після цього опис класу набуде такого вигляду:

```
public class Invoice
{
    public int InvoiceID { get; set; }
    [MaxLength(12)]
    public string Номер_накладної { get; set; }
    public DateTime Дата { get; set; }
    public int ManufacturerID { get; set; }

    // Властивості навігації
    public virtual Manufacturer Manufacturer { get; set; }
  //public virtual ICollection<InvoiceProduct> InvoiceProducts {get; set;}

    private readonly ObservableListSource<InvoiceProduct>
        invoiceProducts = new ObservableListSource<InvoiceProduct>();
    public virtual ObservableListSource<InvoiceProduct>
        InvoiceProducts { get { return invoiceProducts; } }
}
```

3. Щоб позбавитися від помилок, пов'язаних із додаванням даних у методі ***Seed***, відкрийте файл ***Configuration.cs***, що міститься в папці ***Migrations*** проєкту ***моделювання***, і закоментуйте рядки методу ***Seed***, які стосуються додавання до нової накладної товарів, отриманих від виробника з кодом 2 (позначених коментарем //2. Додавання товарів …). Ці рядки можна відновити, якщо знову знадобляться міграції.

4. Перевірте рішення на відсутність помилок, виконавши команду **Build – Rebuild Solution**.

Завдання 2

Створіть форму ***Накладні*** з використанням візуальних засобів (див. рис. 6.32).

Виконання

1. Додайте до проєкту ***winForms*** нову форму, давши їй ім'я ***form-Накладні***.

2. Клацніть вузол сутності ***Invoice*** у вікні **Data Sources** і з її списку, що розкривається, виберіть подання **Details**.

3. Виберіть подання **ComboBox** у вузлі ***Invoice*** для властивості ***ManufacturerID*** у списку, що розкривається.

4. Перетягніть з вузла ***Invoice*** на форму ***Накладні*** по черзі такі властивості: InvoiceID, Номер_накладної, Дата, ManufacturerID.

На формі з'явилися елементи керування для відображення даних і панель навігатора, а в області компонентів – об'єкти, які забезпечують роботу з даними сутності (рис. 6.33).

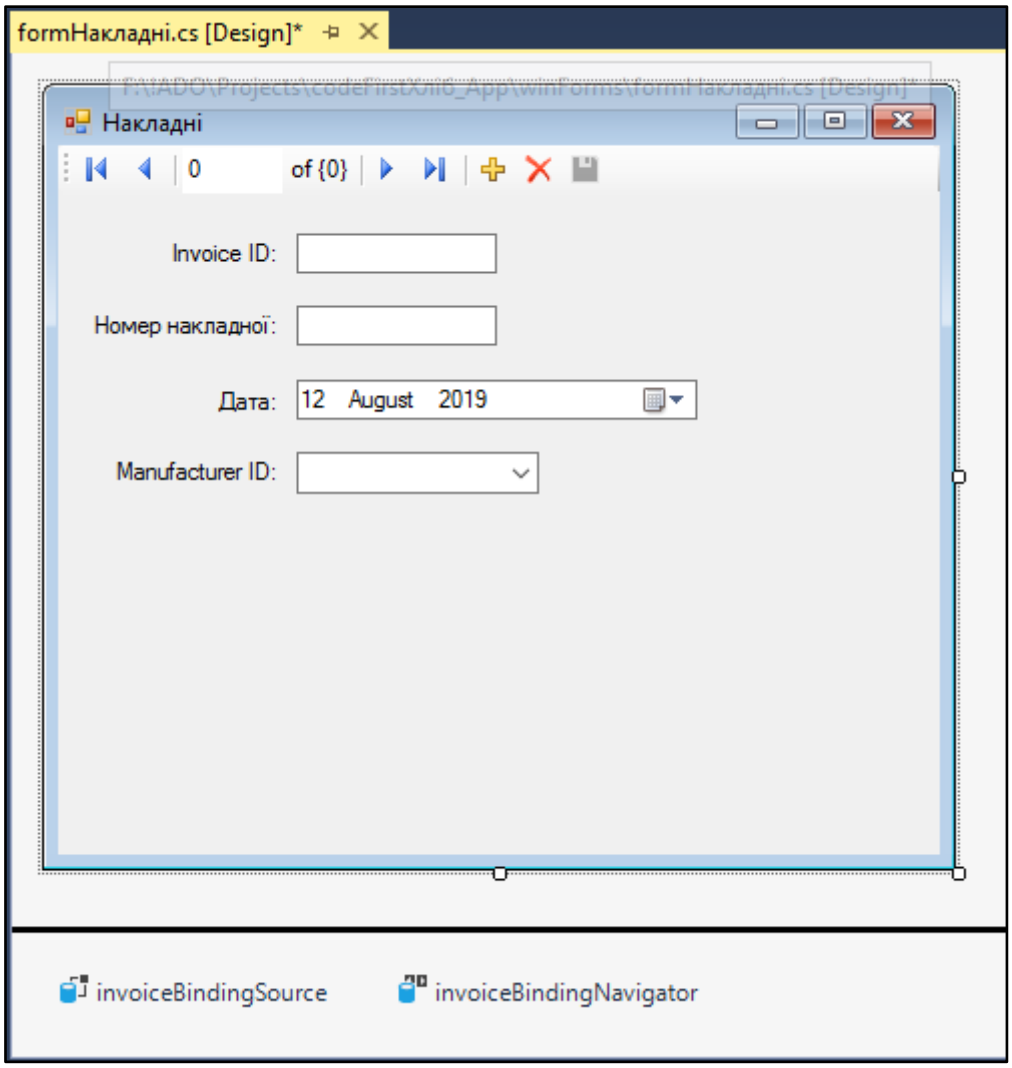


Рис. 6.33. **Вікно форми *Накладні* після додавання сутності *Invoice***

5. Налаштуйте елемент DateTimePicker, установивши для його властивості **Format** значення ***Short***, а також змініть значення властивості **Text** для напису ***ManufacturerID***, установивши нове значення ***Виробник***.

6. Щоб в елементі ComboBox відображалися назви виробників, перетягніть вузол сутності ***Manufacturer*** із вікна **Data Sources** на цей ComboBox і відпустіть. Потім для елемента ComboBox ***Виробник*** установіть такі значення властивостей, наведені в табл. 6.7.

Таблиця 6.7

Властивості елемента *Виробник*

Властивості	Значення
DataSource	manufacturerBindingSource
DisplayMember	Виробник
ValueMember	ManufacturerID
SelectedValue	invoiceBindingSource – ManufacturerID

7. Двічі клацніть у вільному місці форми ***Накладні*** й у вікні коду форми уведіть оператори тіла оброблювача події ***formНакладні_Load***. Він має такий вигляд:

```
BreadContext context;

private void formНакладні_Load(object sender, EventArgs e)
{
    // Створюємо екземпляр класу DbContext
    context = new BreadContext();

    // Завантажуємо дані для manufacturerBindingSource
    manufacturerBindingSource.DataSource =
        context.Manufacturers.ToList();

    // Завантажуємо дані для invoiceBindingSource
    context.Invoices.Load();

    invoiceBindingSource.DataSource =
        context.Invoices.Local.ToBindingList();
}
```

У розділ опису просторів імен додайте ще й такі:

```
using System.Data.Entity;
using моделювання.DataAccess;
```

8. Перейдіть у вікно конструктора форми ***Накладні***, клацніть правою клавішею мишки на кнопці ***Зберегти***, розташованої в елементі ***invoiceBindingNavigator***, і виберіть значення **Enable** з контекстового меню. Потім додайте код оброблювача події "Клацання кнопки Зберегти".

```
private void invoiceBindingNavigatorSaveItem_Click(object sender,
    EventArgs e)
{
    invoiceBindingSource.EndEdit();
    context.SaveChanges();
}
```

9. Перейдіть у вікно конструктора форми ***Хліб*** і додайте код оброблювача події "Клацання кнопки Накладні":

```
private void buttonНакладні_Click(object sender, EventArgs e)
{
    formНакладні вікноНакладні = new formНакладні();
    вікноНакладні.ShowDialog();
}
```

10. Запустіть програму на виконання й перевірте функціональність форми ***Накладні***, додавши дані про нову накладну і зберігши їх (рис. 6.34).

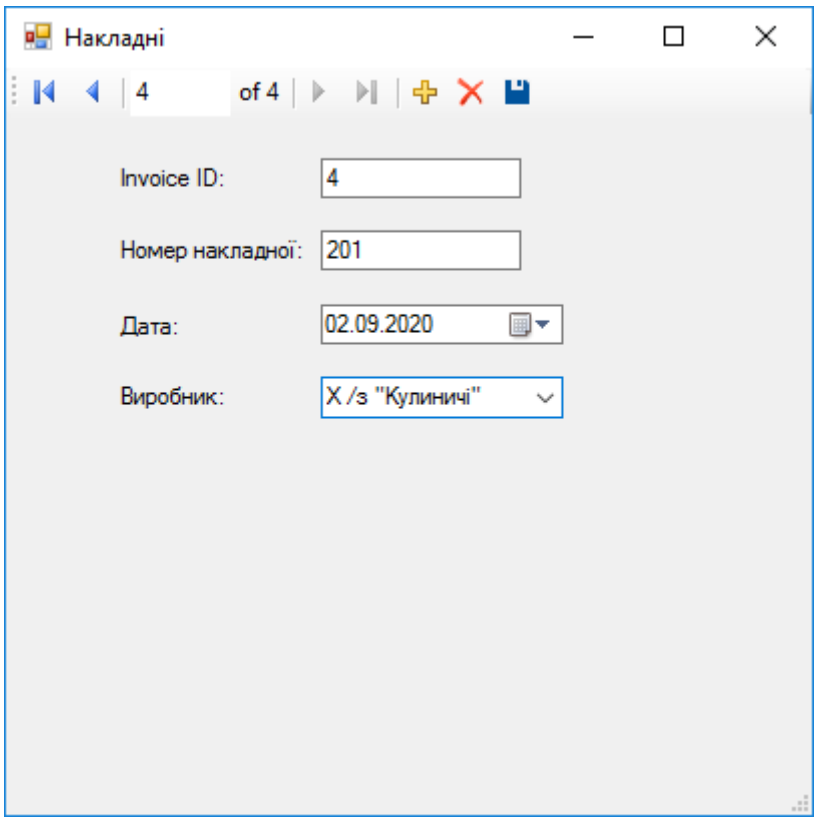


Рис. 6.34. **Додавання даних про накладну**

11. Закрийте форми ***Накладні*** та ***Хліб*** і збережіть зміни, зроблені у проєкті.

12. Перетягніть підвузол ***InvoiceProducts***, що перебуває у вузлі ***Invoice*** вікна **DataSources**, у нижню частину форми ***Накладні***. Перевірте функціональність форми ***Накладні*** (рис. 6.35).

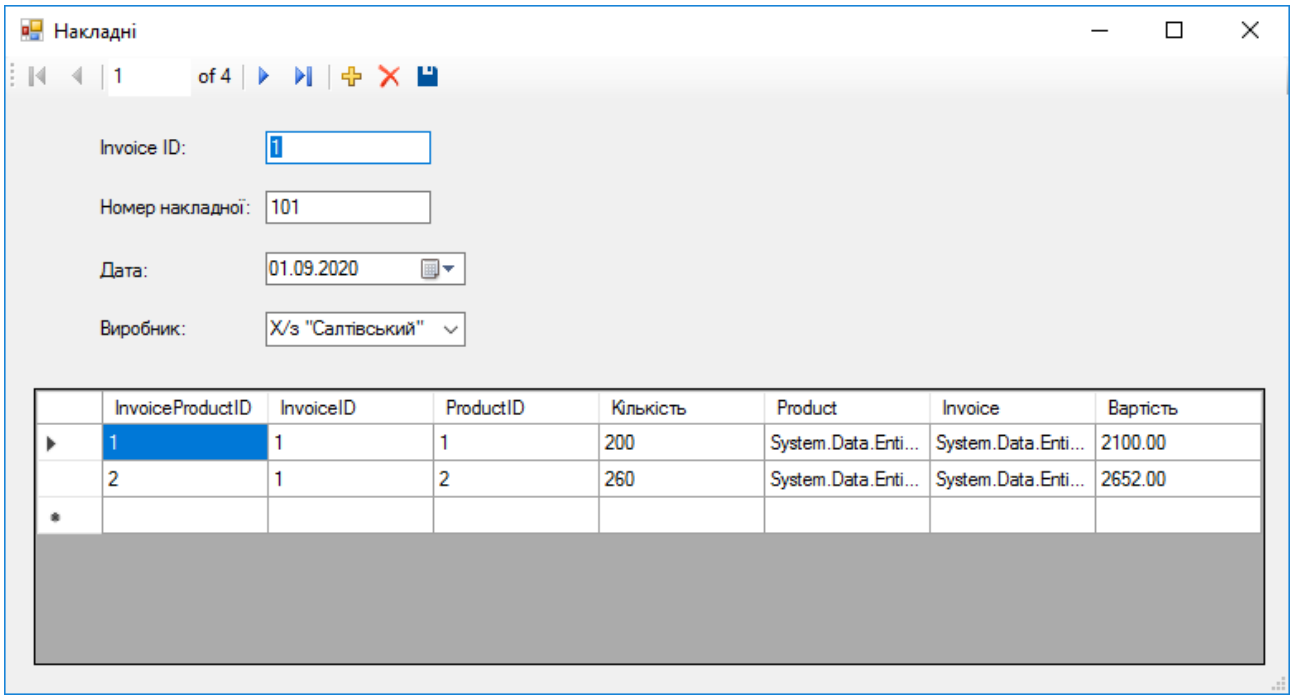


Рис. 6.35. **Форма *Накладні* з доданим підвузол *InvoiceProducts***

13. Налаштуйте відображення елемента ***invoiceProductsDataGridView***, зробивши невидимими стовпці ***InvoiceProductID*** та ***InvoiceID*** і видаливши стовпці ***Product*** та ***Invoice***. Для цього можна використати властивість **Columns**.

14. Налаштуйте властивість ***ProductID*** у вигляді елемента ComboBox. Для цього:

14.1. Перетягніть на форму яке-небудь поле (наприклад, ***Товар***) із вузла ***Products*** вікна **Data Sources** у вільне місце форми та відразу вилучіть його. У нижній частині вікна конструктора залишиться елемент ***productBindingSource***.

14.2. Перейдіть у вікно **Edit Columns** за допомогою властивості **Columns** для елемента ***invoiceProductsDataGridView***.

14.3. Змініть тип стовпця ***ProductID*** на **DataGridViewComboBoxColumn**, дайте йому ім'я ***productIDDataGridViewComboBoxColumn*** і заголовок ***Товар***. Тут також установіть значення властивостей у групі **Data**, наведені в табл. 6.8.

Таблиця 6.8

Властивості у групі *Data*

Властивості	Значення
DataPropertyName	ProductID
DataSource	productBindingSource
DisplayMember	Товар
ValueMember	ProductID

Вікно **Edit Columns** зі встановленими значеннями властивостей у групі **Data** подано на рис. 6.36.

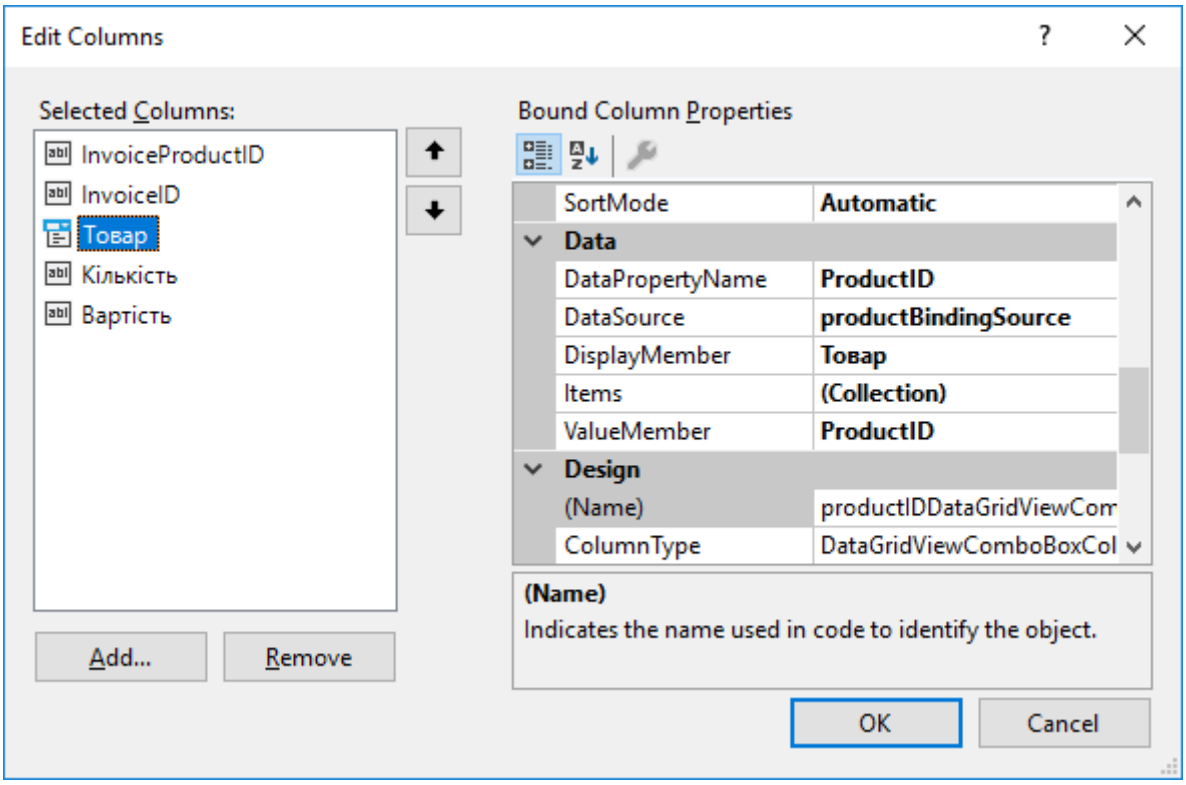


Рис. 6.36. **Установлення значень властивостей у групі *Data***

14.4. Перейдіть у вікно коду форми ***Накладні*** й додайте оператор для завантаження даних для ***productBindingSource*** в тілі оброблювача події ***formНакладні_Load***:

```
// Завантажуємо дані для productBindingSource
productBindingSource.DataSource = context.Products;
```

Після цього оброблювач набуде такого вигляду:

```
private void formНакладні_Load(object sender, EventArgs e)
{
    // Створюємо екземпляр класу DbContext
    context = new BreadContext();
    // Завантажуємо дані для manufacturerBindingSource
    manufacturerBindingSource.DataSource =
        context.Manufacturers.ToList();
    // Завантажуємо дані для productBindingSource
    productBindingSource.DataSource = context.Products.ToList();

    // Завантажуємо дані для invoiceBindingSource
    context.Invoices.Load();
    invoiceBindingSource.DataSource =
        context.Invoices.Local.ToBindingList();
}
```

14.5. Додайте оператор завершення редагування ***invoiceProducts-BindingSource*** в тілі оброблювача події ***invoiceBindingNavigatorSaveI-tem_Click***:

```
invoiceProductsBindingSource.EndEdit();
```

та переміщення записами, щоб забезпечити оновлення даних на формі.

Після цього оброблювач набуде такого вигляду:

```
private void invoiceBindingNavigatorSaveItem_Click(object sender, EventArgs e)
{
    invoiceProductsBindingSource.EndEdit();
    invoiceBindingSource.EndEdit();
    context.SaveChanges();

    // Оновлення даних на формі
    invoiceBindingSource.MovePrevious();
    invoiceBindingSource.MoveNext();
}
```

15. Установіть значення за замовчуванням для стовпця ***Товар***. Для цього:

15.1. Виділіть елемент invoiceProductsDataGridView.

15.2. Перейдіть у вікно властивостей і встановіть режим відобра-

ження подій, клацнувши кнопку **Events**  .

15.3. Двічі клацніть подію **DefaultValuesNeeded**.

15.4. Уведіть тіло оброблювача події **DefaultValuesNeeded**. Він набуде такого вигляду:

```
private void invoiceProductsDataGridView_DefaultValuesNeeded(object
sender, DataGridViewRowEventArgs e)
{
    //Значення за замовчуванням для стовпця Товар
    e.Row.Cells["productIDDataGridViewComboBoxColumn"].Value =
        context.Products.Min(t => t.ProductID);
}
```

16. Запустіть програму на виконання й перевірте функціональність форми ***Накладні***, додавши дані про надходження кількох товарів в останню накладну і зберігши їх (рис. 6.37).

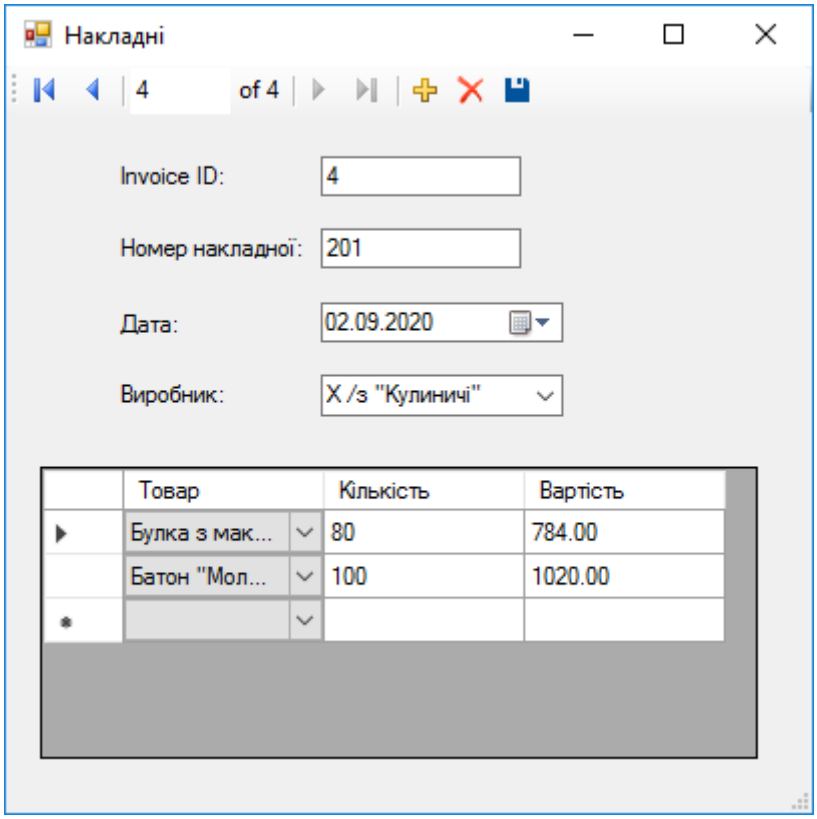


Рис. 6.37. **Форма *Накладні* з відображенням властивості *ProductID* у вигляді ComboBox**

17. Закрийте форми ***Накладні*** та ***Хліб***.

18. Збережіть зміни, зроблені в рішенні.

4.5. Використання технології **LINQ to Entity** *в завданнях аналізу даних*

Завдання

Додайте в застосунок форму ***Продажі виробника***. У ній відображають дані про результати продажів товарів вибраного виробника в табличній формі та у вигляді об'ємної кругової діаграми (рис. 6.38).

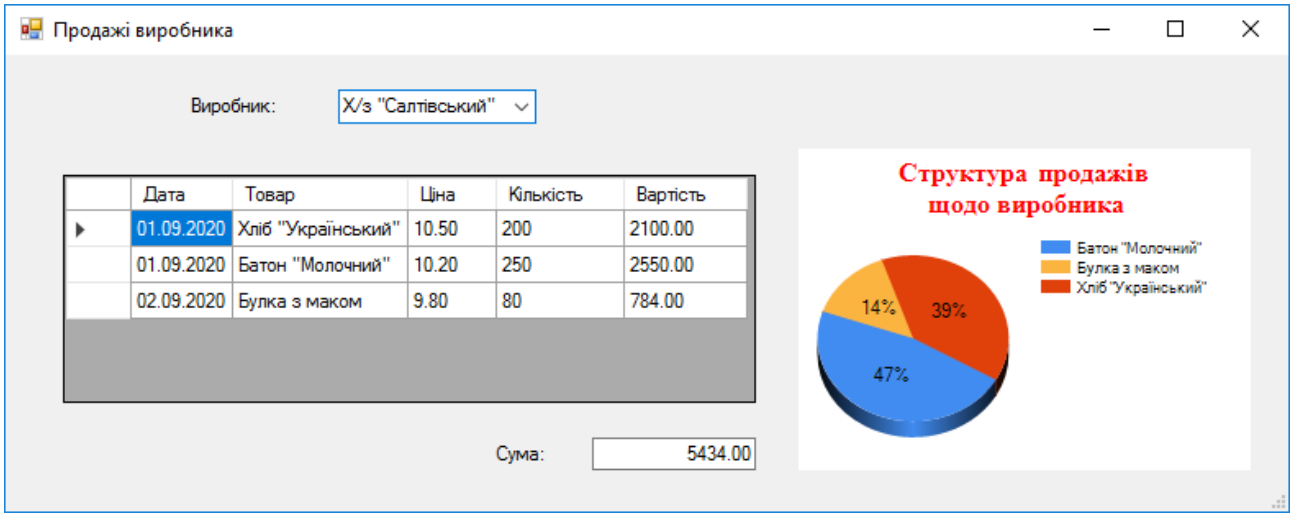


Рис. 6.38. **Форма *Продажі виробника***

Ідеї виконання

Дані, що відображають в елементі DataGridView, отримують із сутностей ***Sale*** та ***Product***, пов'язаних між собою асоціацією. Тому для відбору таких комбінацій даних доцільно використати запит LINQ.

Більш того, набір сутностей, що відображають, залежить від вибору виробника в елементі ComboBox. Такий відбір даних у LINQ-запиті реалізують за допомогою речення Where.

Для відображення загальної вартості проданих товарів вибраного виробника слід скористатися агрегатною функцією Sum у LINQ-запиті.

Дані, що використовують для побудови діаграми, також отримані з LINQ-запиту, до якого додано речення group. Воно забезпечує групування даних за назвами товарів з обчисленням суми вартостей у кожній групі.

Для реакції на зміну виробника в ComboBox треба використати оброблювача події із цим елементом. Він містить аналогічні запити.

Виконання

1. Додайте в застосунок нову форму, давши їй ім'я ***formПродажіВиробника***, а для властивості **Text** установіть значення ***Продажі виробника***.

2. Додайте на форму поле зі списком ***Виробник***. Для цього:

2.1. Виберіть у вікні **Data Sources** подання **ComboBox** для властивості ***ManufacturerID*** у вузлі ***Manufacturer***.

2.2. Перетягніть властивість ***ManufacturerID*** із вузла ***Manufacturer*** у вікні ***Data Sources*** на форму ***Продажі виробника***.

На формі з'явився елемент керування ComboBox і панель навігатора, а в області компонентів – елемент ***manufacturerBindingSource***.

2.3. Видаліть панель навігатора з форми.

2.4. Змініть властивість **Text** для напису ***ManufacturerID***, установивши нове значення ***Виробник***.

2.5. Установіть у вікні властивостей значення для елемента ComboBox ***Виробник***, наведені в табл. 6.9.

Таблиця 6.9

Властивості елемента Виробник

Властивості	Значення
DataSource	manufacturerBindingSource
DisplayMember	Виробник
ValueMember	ManufacturerID
SelectedValue	(none)

2.6. Двічі клацніть у вільному місці форми ***Продажі виробника*** й у вікні коду форми введіть оператори тіла оброблювача події ***form-ПродажіВиробника_Load***. Він має такий вигляд:

```
// Спільні об'єкти
BreadContext context;

private void formПродажіВиробника_Load(object sender, EventArgs e)
{
    // Створюємо екземпляр класу DbContext
    context = new BreadContext();

    // Завантажуємо дані для manufacturerBindingSource
    manufacturerBindingSource.DataSource =
        context.Manufacturers.ToList();
}
```

У розділ опису просторів імен додайте ще й такі:

```
using System.Data.Entity;
using моделювання.DataAccess;
```

2.7. Перейдіть у вікно конструктора форми ***Хліб*** і додайте код оброблювача події "Клацання кнопки Продажі виробника":

```
private void buttonПродажіВиробника_Click(object sender, EventArgs e)
{
    formПродажіВиробника вікноПродажіВиробника =
        new formПродажіВиробника();
    вікноПродажіВиробника.ShowDialog();
}
```

2.8. Запустіть програму на виконання й перевірте функціональність поля зі списком ***Виробник*** на формі ***Продажі виробника*** (рис. 6.39).

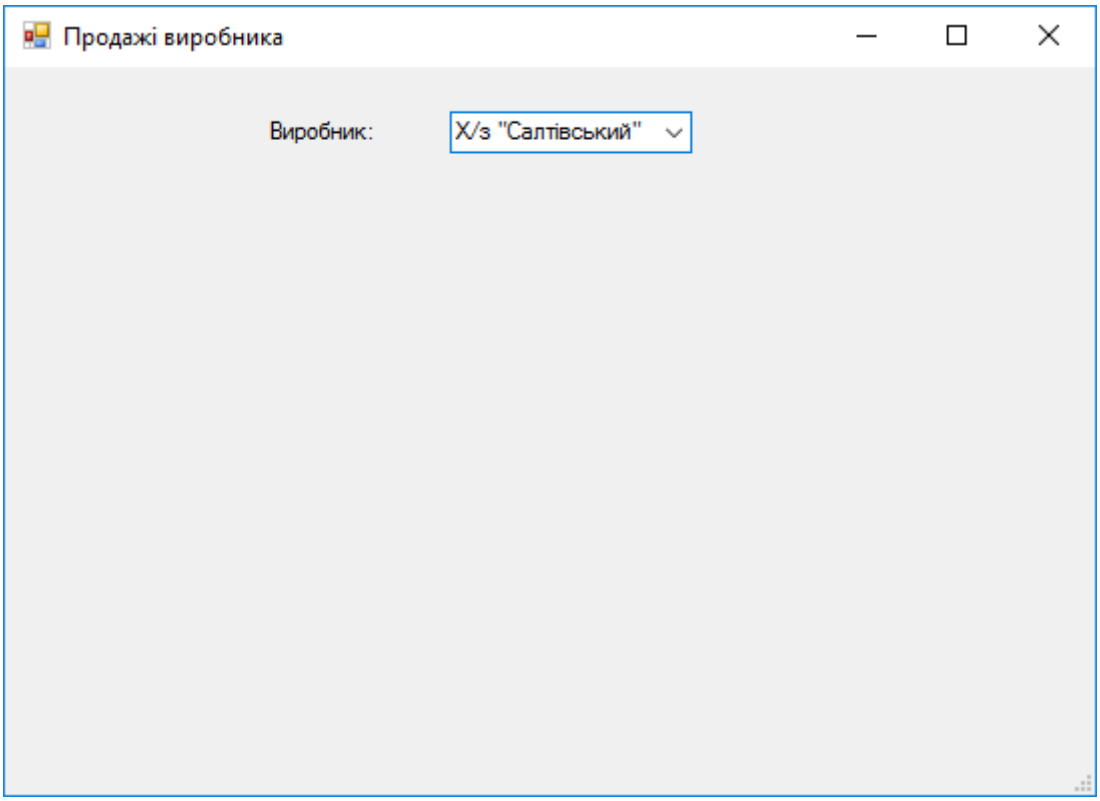


Рис. 6.39. **Форма *Продажі виробника* з полем зі списком *Виробник***

2.9. Закрийте форми ***Продажі виробника*** та ***Хліб*** і збережіть зміни, зроблені у проєкті.

3. Для відображення результатів продажів товарів вибраного виробника в табличній формі виконайте таке:

3.1. Додайте на форму елемент **DataGridView** з ім'ям ***gvПродажіВиробника***.

3.2. Двічі клацніть у вільному місці форми ***Продажі виробника*** й у вікні коду форми введіть у розділ спільних об'єктів класу такий оператор:

```
BindingSource bindingПродажіВиробника = new BindingSource();
```

,

а в тіло оброблювача події ***formПродажіВиробника_Load*** додайте такі оператори:

```
// Запит для gvПродажіВиробника
var queryПродажіВиробника = from продаж in context.Sales
    where продаж.ManufacturerID ==
(int)manufacturerIDComboBox.SelectedValue
select new
    {
        Дата = продаж.Дата,
        Товар = продаж.Product.Товар,
        Ціна = продаж.Product.Ціна,
        Кількість = продаж.Кількість,
        Вартість = продаж.Product.Ціна * продаж.Кількість
    };

// Відображаємо дані
bindingПродажіВиробника.DataSource = queryПродажіВиробника.ToList();
gvПродажіВиробника.DataSource = bindingПродажіВиробника;
gvПродажіВиробника.AutoResizeColumns();
```

Тепер ці дві частини коду мають такий вигляд:

```
// Спільні об'єкти
BreadContext context;
BindingSource bindingПродажіВиробника = new BindingSource();

private void formПродажіВиробника_Load(object sender, EventArgs e)
{
    // Створюємо екземпляр класу DbContext
    context = new BreadContext();

    // Завантажуємо дані для manufacturerBindingSource
    manufacturerBindingSource.DataSource =
        context.Manufacturers.ToList();
```

```
    // Запит для gvПродажіВиробника
    var queryПродажіВиробника = from продаж in context.Sales
        where продаж.ManufacturerID ==
        (int)manufacturerIDComboBox.SelectedValue
        select new
        {
            Дата = продаж.Дата,
            Товар = продаж.Product.Товар,
            Ціна = продаж.Product.Ціна,
            Кількість = продаж.Кількість,
            Вартість = продаж.Product.Ціна * продаж.Кількість
        };

    // Відображаємо дані
    bindingПродажіВиробника.DataSource =
        queryПродажіВиробника.ToList();
    gvПродажіВиробника.DataSource = bindingПродажіВиробника;
    gvПродажіВиробника.AutoResizeColumns();
}
```

3.3. Запустіть програму на виконання й перевірте функціональність поля зі списком ***Виробник*** та елемента DataGridView на формі ***Продажі виробника***. В останньому відображають результати продажів товарів виробника, указаного в полі зі списком (рис. 6.40). Але в разі зміни виробника, відповідні дані не з'являються. Цей недолік ліквідують у подальшому за допомогою оброблювача події.

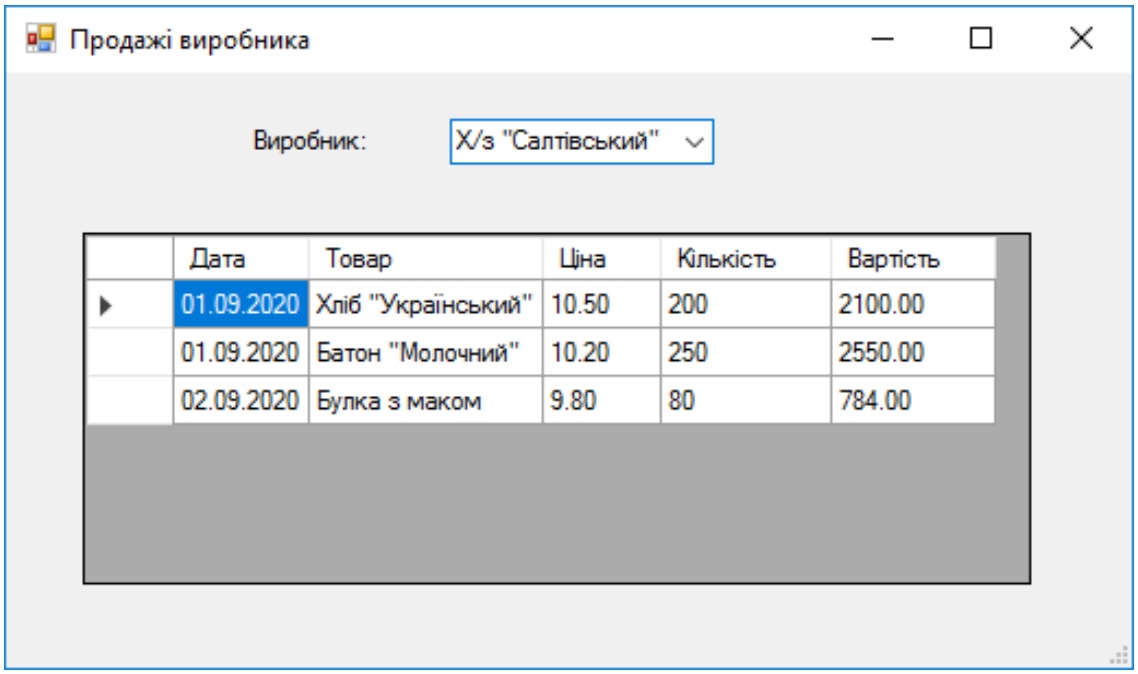


Рис. 6.40. **Форма *Продажі виробника* з результатами продажів товарів виробника**

3.4. Закрийте форми ***Продажі виробника*** та ***Хліб*** і збережіть зміни, зроблені у проєкті.

4. Для відображення сумарної вартості продажів товарів вибраного виробника виконайте таке:

4.1. Додайте на форму елементи напис і текстове поле.

4.2. Для властивості **Text** напису встановіть значення ***Сума***, а для властивості **Name** текстового поля – значення ***textBoxСума***.

4.3. Двічі клацніть у вільному місці форми ***Продажі виробника*** й у вікні коду форми додайте в кінець коду оброблювача події ***formПродажіВиробника*** такі оператори:

```
//Сума
decimal Сума= (decimal)queryПродажіВиробника.Sum(p => p.Вартість);
textBoxСума.Text = Сума.ToString("0.00");
textBoxСума.TextAlign = HorizontalAlignment.Right;
```

4.4. Запустіть програму на виконання й перевірте функціональність текстового поля ***Сума*** на формі ***Продажі виробника***. У ньому відображено сумарну вартість продажів товарів виробника, указаного в полі зі списком (рис. 6.41).

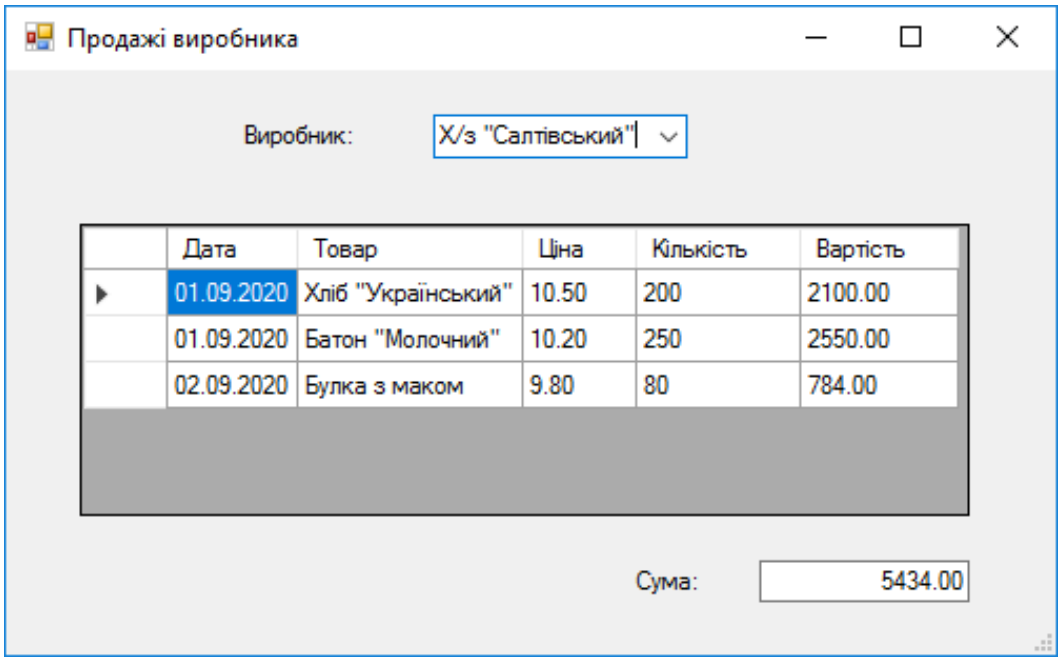


Рис. 6.41. **Форма *Продажі виробника* з полем *Сума***

4.5. Закрийте форми ***Продажі виробника*** та ***Хліб*** і збережіть зміни, зроблені у проєкті.

5. Для відображення структури продажів товарів вибраного виробника у вигляді об'ємної кругової діаграми виконайте таке:

5.1. Додайте на форму елемент **Chart** з ім'ям ***chartСтруктура***.

5.2. Двічі клацніть у вільному місці форми ***Продажі виробника*** й у вікні коду форми додайте у розділ просторів імен такий оператор:

```
using System.Windows.Forms.DataVisualization.Charting;//Для діаграми
```

,

у розділ спільних об'єктів класу такий оператор:

```
BindingSource bindingСтруктура = new BindingSource();
```

,

а в кінець коду тіла оброблювача події ***formПродажіВиробника_Load*** такі оператори:

```
//***********
// Діаграма *
//***********

//Запит
var queryСтруктура = from товар in queryПродажіВиробника
group товар by товар.Товар into t
select new
{
    Товар = t.Key,
    Вартість = t.Sum(p => p.Вартість)
};

//Дані для відображення
bindingСтруктура.DataSource = queryСтруктура.ToList();
chartСтруктура.DataSource = bindingСтруктура;
chartСтруктура.Series["Series1"].XValueMember = "Товар";
chartСтруктура.Series["Series1"].YValueMembers = "Вартість";

//Тип діаграми
chartСтруктура.Series["Series1"].ChartType = SeriesChartType.Pie;
```

```
//Підписи на діаграмі
chartСтруктура.Series["Series1"].Label = "#PERCENT{P0}";

//Об'ємний варіант (3D)
this.chartСтруктура.ChartAreas[0].Area3DStyle.Enable3D = true;

//Заголовок
chartСтруктура.Titles.Add("Заголовок");
chartСтруктура.Titles[0].Text = "Структура продажів\n щодо виробника";
chartСтруктура.Titles[0].Font = new Font("Times New Roman", 12,
    FontStyle.Bold);
chartСтруктура.Titles[0].ForeColor = Color.Red;

//Легенда
chartСтруктура.Series["Series1"].IsVisibleInLegend = true;
chartСтруктура.Series["Series1"].LegendText = "#VALX";
```

5.3. Запустіть програму на виконання й перевірте функціональність діаграми на формі ***Продажі виробника***. У ній відображено структуру продажів товарів вибраного виробника у вигляді об'ємної кругової діаграми (рис. 6.42).

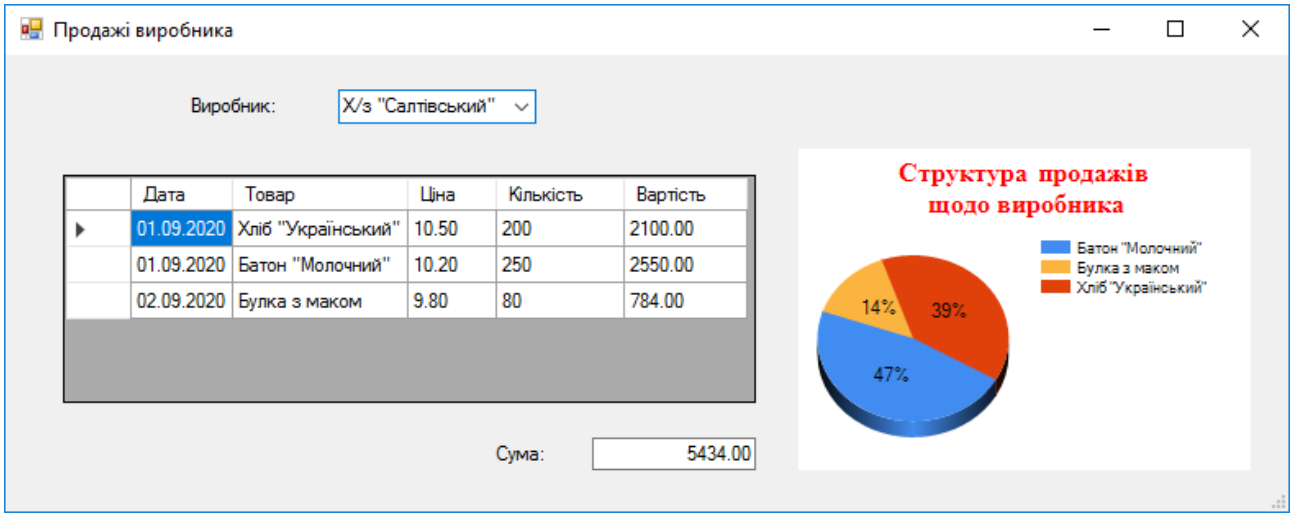


Рис. 6.42. **Форма *Продажі виробника* з діаграмою**

5.4. Закрийте форми ***Продажі виробника*** та ***Хліб*** і збережіть зміни, зроблені у проєкті.

6. Для відображення результатів продажів товарів вибраного виробника в разі його зміни виконайте таке:

6.1. Перейдіть у вікно конструктора форми ***Продажі виробника***.

6.2. Двічі клацніть на полі зі списком ***Виробник*** і у вікні коду форми введіть оператори тіла оброблювача події ***ManufacturerIDComboBox_SelectedIndexChanged***. Він має такий вигляд:

```
private void manufacturerIDComboBox_SelectedIndexChanged(object sender,
        EventArgs e)
{
     if (this.CanFocus) //Після завантаження
     {
         var queryПродажіВиробника =
             from продаж in context.Sales
             where продаж.ManufacturerID ==
                 (int)manufacturerIDComboBox.SelectedValue
             select new
             {
               Дата = продаж.Дата,
               Товар = продаж.Product.Товар,
               Ціна = продаж.Product.Ціна,
               Кількість = продаж.Кількість,
               Вартість = продаж.Product.Ціна * продаж.Кількість
             };

         // Відображаємо дані
         bindingПродажіВиробника.DataSource =
             queryПродажіВиробника.ToList();

         //Сума
         decimal Сума = (decimal)queryПродажіВиробника.Sum(p =>
             p.Вартість);
         textBoxСума.Text = Сума.ToString("0.00");

         //***********
         // Діаграма *
         //***********

         //Запит
         var queryСтруктура =
             from товар in queryПродажіВиробника
             group товар by товар.Товар into t
             select new
```

```
            {
                Товар = t.Key,
                Вартість = t.Sum(p => p.Вартість)
            };
        //Дані для відображення
        bindingСтруктура.DataSource = queryСтруктура.ToList();
        if (queryСтруктура.Count() > 0)
        {
           chartСтруктура.DataSource = bindingСтруктура;
           chartСтруктура.Series["Series1"].XValueMember = "Товар";
           chartСтруктура.Series["Series1"].YValueMembers = "Вартість";
           chartСтруктура.Series["Series1"].ChartType = SeriesChartType.Pie;
        }
    }
}
```

6.3. Запустіть програму на виконання й перевірте функціональність поля зі списком ***Виробник***. У разі зміни виробника на формі відображено результати продажів товарів виробника, указаного в полі зі списком.

6.4. Закрийте форми ***Продажі виробника*** та ***Хліб*** і збережіть зміни, зроблені у проєкті.

У звіті з лабораторної роботи подайте скриншот форми ***Продажі виробника*** в режимі виконання, а також код оброблювача ***formПродажі-Виробника_Load*** та **comboBoxВиробник*_SelectedIndexChanged***. Поясніть, чим відрізняється побудова аналогічної форми в лабораторній роботі 5.

Завдання для самостійного виконання

1. Проаналізуйте проєкт ***моделювання*** після додавання проєкту ***winForms***. Які елементи з першого проєкту можна видалити без втрати функціонування другого проєкту? Перевірте експериментальним шляхом зроблені висновки.

2. Додайте в застосунок форму ***Виробники***, у якій будуть відображати й змінювати дані таблиці ***Manufacturers***.

3. Додайте в базу таблицю ***Групи_товарів*** із полями ***Група_товарівID*** та ***Група_товарів***, пов'язавши її з таблицею ***Товари*** відношенням "один-до-багатьох". Використайте для цього засоби міграції.

4. Додайте стовпець ***Ціна*** в елемент ***invoiceProductsDataGridView*** на формі ***Накладні***.

5. Додайте у проєкт ***winForms*** такі форми для завдань аналізу даних ***Прайс*** та ***Продано*** (див. лабораторну роботу 5) .

6. Додайте на форму ***Продажі виробника*** стовпчикову діаграму для візуального аналізу динаміки продажів товарів кожного виробника. На горизонтальній осі подайте дати продажів, а на вертикальній – вартості.

7. На формі ***Продажі виробників*** додайте дані про найкращий товар (топтовар) за результатами продажів за такими показниками:

7.1. За кількістю проданих одиниць.

7.2. За загальною вартістю.

7.3. За отриманим прибутком.

8. До елементів поля зі списком ***Виробник*** на формі ***Продажі виробників*** додайте елемент ***Усі***. У разі його вибору відображають дані про продаж товарів усіма виробниками.

9. На формі ***Продажі виробників*** додатково забезпечте такі види фільтрації:

9.1. За видом товарів (за допомогою елемента ComboBox).

9.2. За діапазоном дат (за допомогою двох елементів TextBox або DateTimePicker).

Примітка. Передбачте ситуацію, коли кожен фільтр не діє, тобто на формі відображають усі дані.

10. Виконайте п. 1 – 4 ходу роботи для індивідуальної бази даних.

Лабораторна робота 7
Реалізація звітів у бізнес-застосунках

Цілі лабораторної роботи:

1. Набуття практичних навичок у створенні документів на основі даних, що зберігають у базі.

2. Освоєння операції фільтрування даних, що відображають у звітах.

3. Набуття практичних навичок у побудові зведених таблиць із різними рівнями вкладеності.

4. Освоєння засобів візуалізації даних та їхньої інтерактивності під час бізнес-аналізу даних.

Перед виконанням роботи студент має знати:

основи побудови SQL-запитів;
основи технології LINQ;
структурні елементи бази даних та їхні властивості;
основи роботи з типізованими наборами даних;
принципи прив'язування даних до елементів інтерфейсу;
основи роботи з ієрархічними даними;
принципи оброблення подій у C#-програмі.

Після виконання роботи студент має вміти:

створювати документи на основі даних, що зберігають у базі;

самостійно розробляти C#-проєкти для друкування документів і виконання бізнес-аналізу даних;

створювати панелі керування даними під час їхнього бізнес-аналізу;

використовувати основні бібліотеки .Net Framework під час розроблення програм.

Підготовча частина

Хід роботи

1. Підготовчий етап.
2. Побудова документів.
3. Фільтрація.
4. Аналіз.

Форма звітності

За результатами виконання роботи необхідно оформити електронний звіт засобами Word. За результатами кожного виконаного завдання з п. "Завдання для самостійного виконання" слід подати постановку завдання, скриншот форми, звіту та відповідний програмний код.

Критерії оцінювання

У 12-бальній системі оцінку результату захисту лабораторної роботи формують за такими правилами:

1. За виконання кожного пункту практичної частини може бути виставлено від 0 до 1 бала.

2. За виконання завдань п. "Завдання для самостійного виконання" може бути виставлено:

від 0 до 1 бала за кожне завдання з діапазону 1 – 5;

від 0 до 0,5 балів за кожне підзавдання із завдання 6;

від 0 до 8 балів за завдання 7;

від 0 до 10 балів за завдання 8.

3. За кілька варіантів виконання одного із завдань додають 1 бал.

4. За вибір варіанта, який із кількох варіантів виконання є оптимальним, та обґрунтування вибору додають 1 бал.

5. За виконання кожного завдання в іншій СУБД (Oracle, MySQL, DB2, PostgreSQL тощо) додають по 1 балу.

6. За запізнення із захистом лабораторної роботи знімають 2 бали за кожний тиждень.

Набрану кількість балів із відповідей на кожне завдання лабораторної роботи підсумовують. У результаті такого підрахунку студент може набрати від 0 до 12 балів.

У разі запізнення із захистом лабораторної роботи понад три тижні студент виконує завдання роботи на основі індивідуальної бази даних. Роботу оцінюють за описаними раніше критеріями.

Практична частина

Постановка загального завдання

Вивчення засобів створення друкованих документів та засобів аналізу даних здійснюють на основі використання елемента керування ReportViewer.

Примітка. Елемент керування ReportViewer поставляють у складі Visual Studio для версій до 2013 включно. Для старших версій Visual Studio його потрібно встановлювати додатково. У п. 1.4 подано такий опис для Visual Studio 2017.

Під час виконання лабораторної роботи будують проєкт ***rptХліб***, який можна використовувати як окремий модуль у застосунках, побудованих на попередніх роботах.

Дані зберігають у базі даних ***Хліб*** під керуванням СУБД SQL Server. На відміну від попередніх робіт, до бази додано таблицю ***Групи***, у якій містяться назви груп товарів (Хліб, Батони, Булки). Кожний товар належить до однієї із груп, тому дочірню таблицю ***Товари*** пов'язано через зовнішній ключ із батьківською таблицею ***Групи***. Додавання цієї таблиці розширить можливості вивчення засобів використання ієрархічних даних у звітах.

Керування проєктом здійснюють за допомогою кнопкової форми ***Звіти*** (рис. 7.1). Кнопки призначено для виклику форм, на яких містяться відповідні звіти.

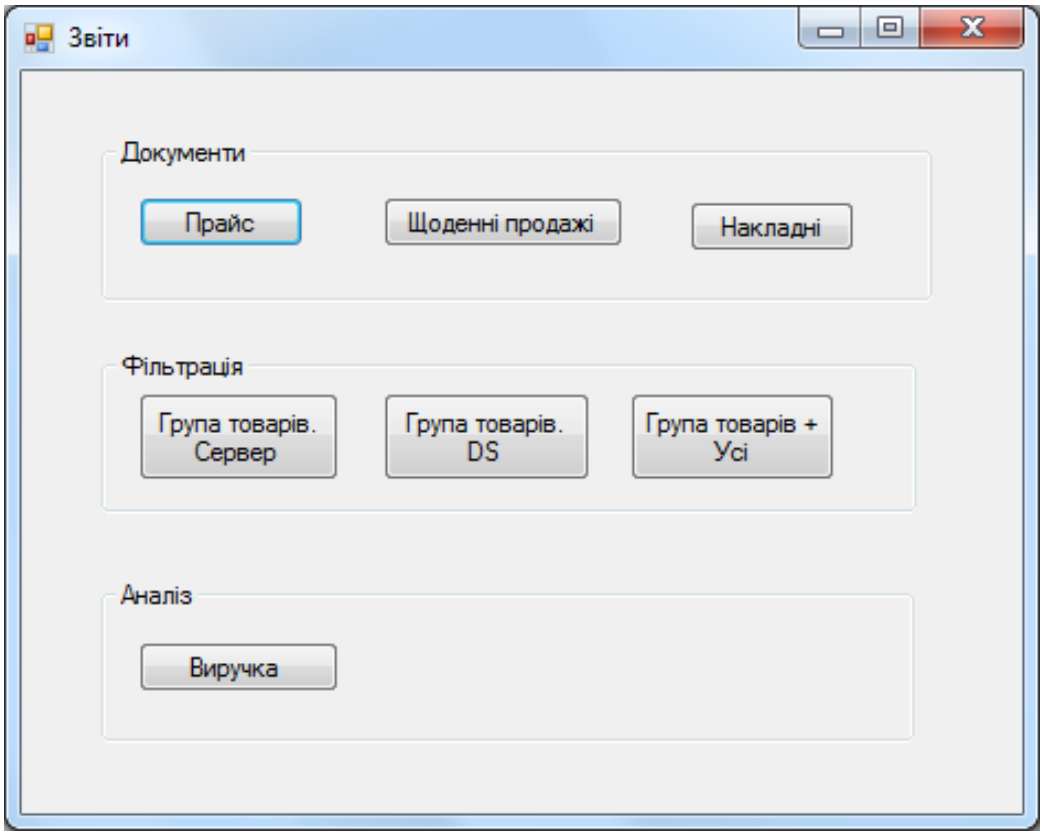


Рис. 7.1. **Кнопкова форма *Звіти***

На рис. 7.2 – 7.9 подано функціональні форми проєкту і відповідні звіти. Їх побудовано на даних, що зберігають у базі даних ***Хліб***.

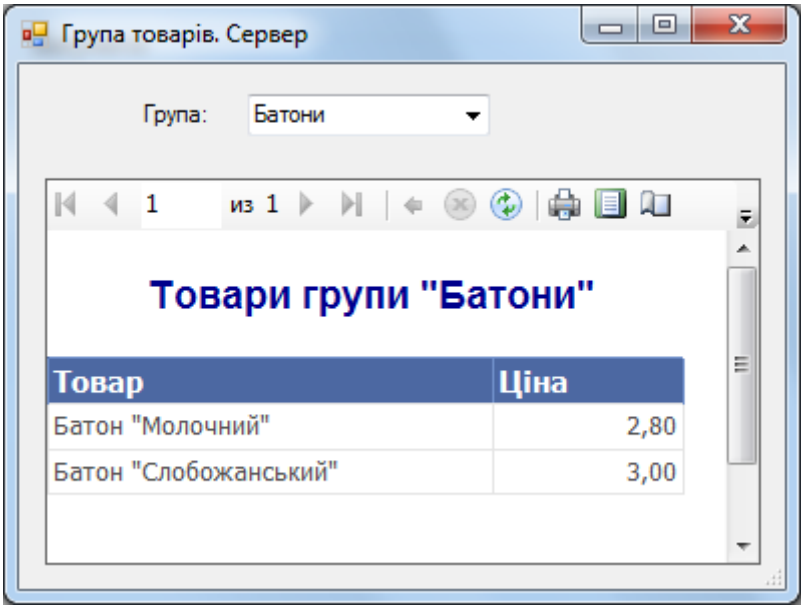


Рис. 7.2. **Відфільтровані на сервері товари вибраної групи**

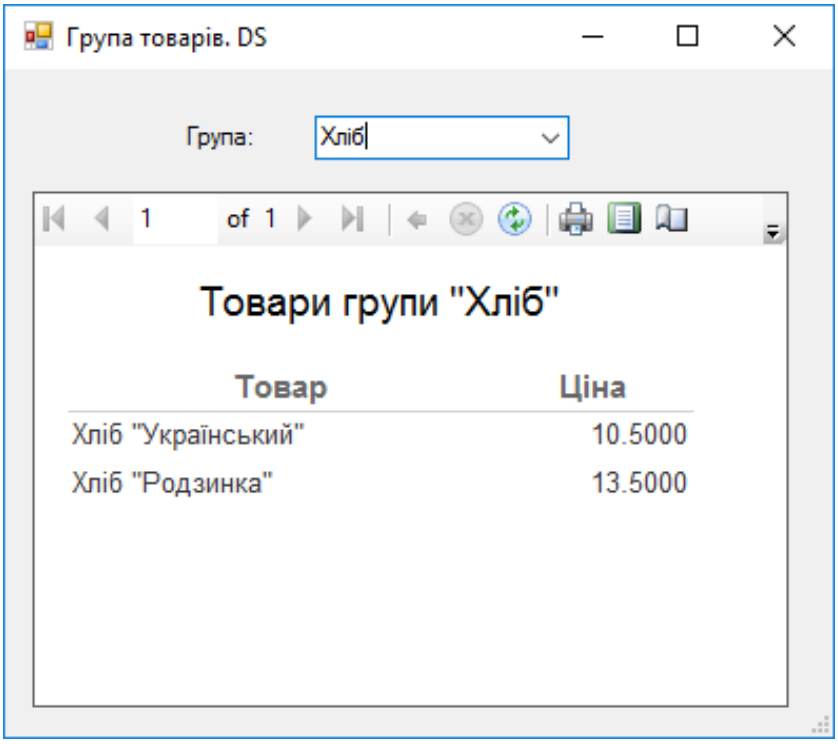


Рис. 7.3. **Відфільтровані на локальному комп'ютері товари вибраної групи**

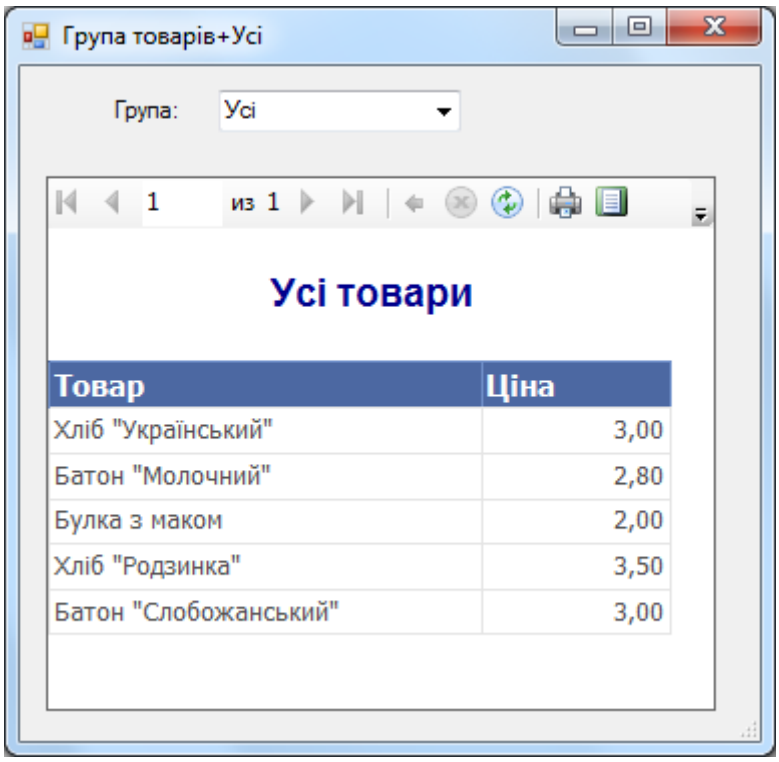


Рис. 7.4. **Список усіх товарів із можливістю відфільтрувати товари певної групи**

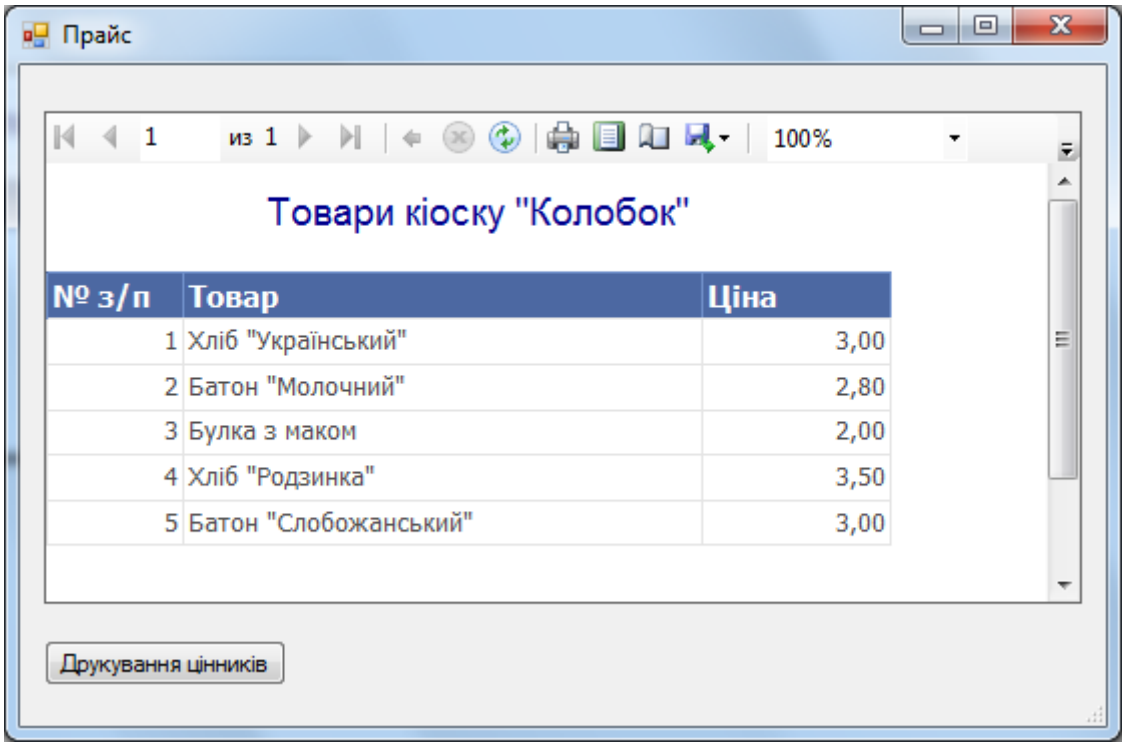


Рис. 7.5. **Список цін товарів на формі *Прайс***

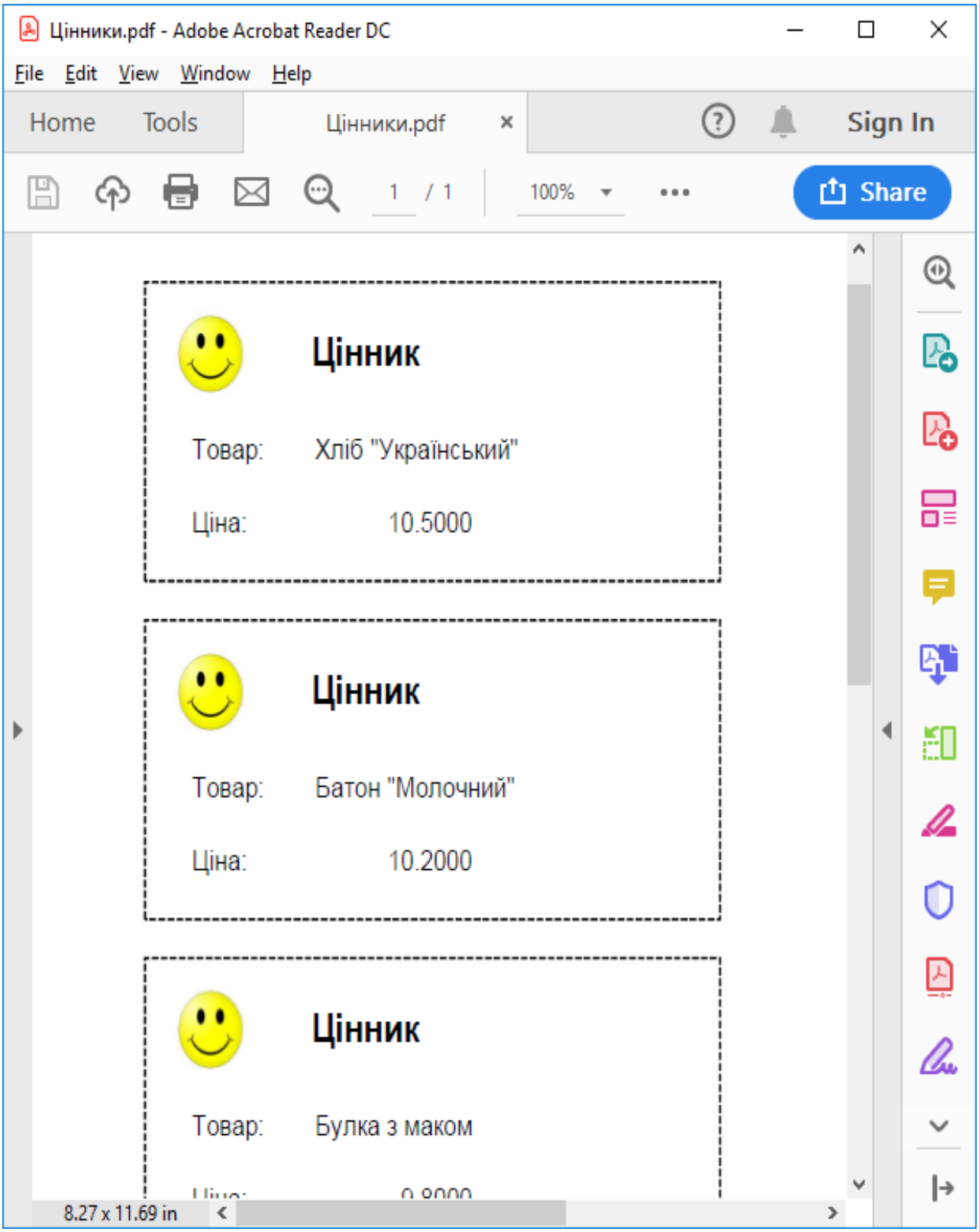


Рис. 7.6. **Цінники, що виводять клацанням кнопки *Друкування цінників* на формі *Прайс***

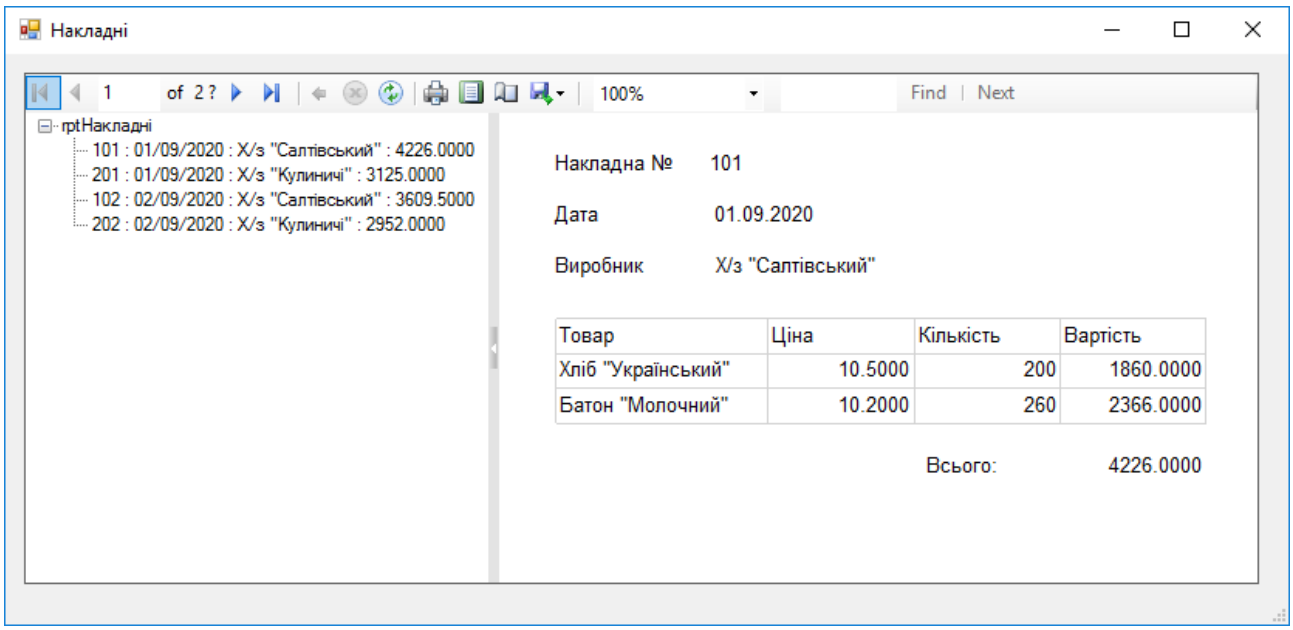


Рис. 7.7. **Документи *Накладні* на однойменній формі**

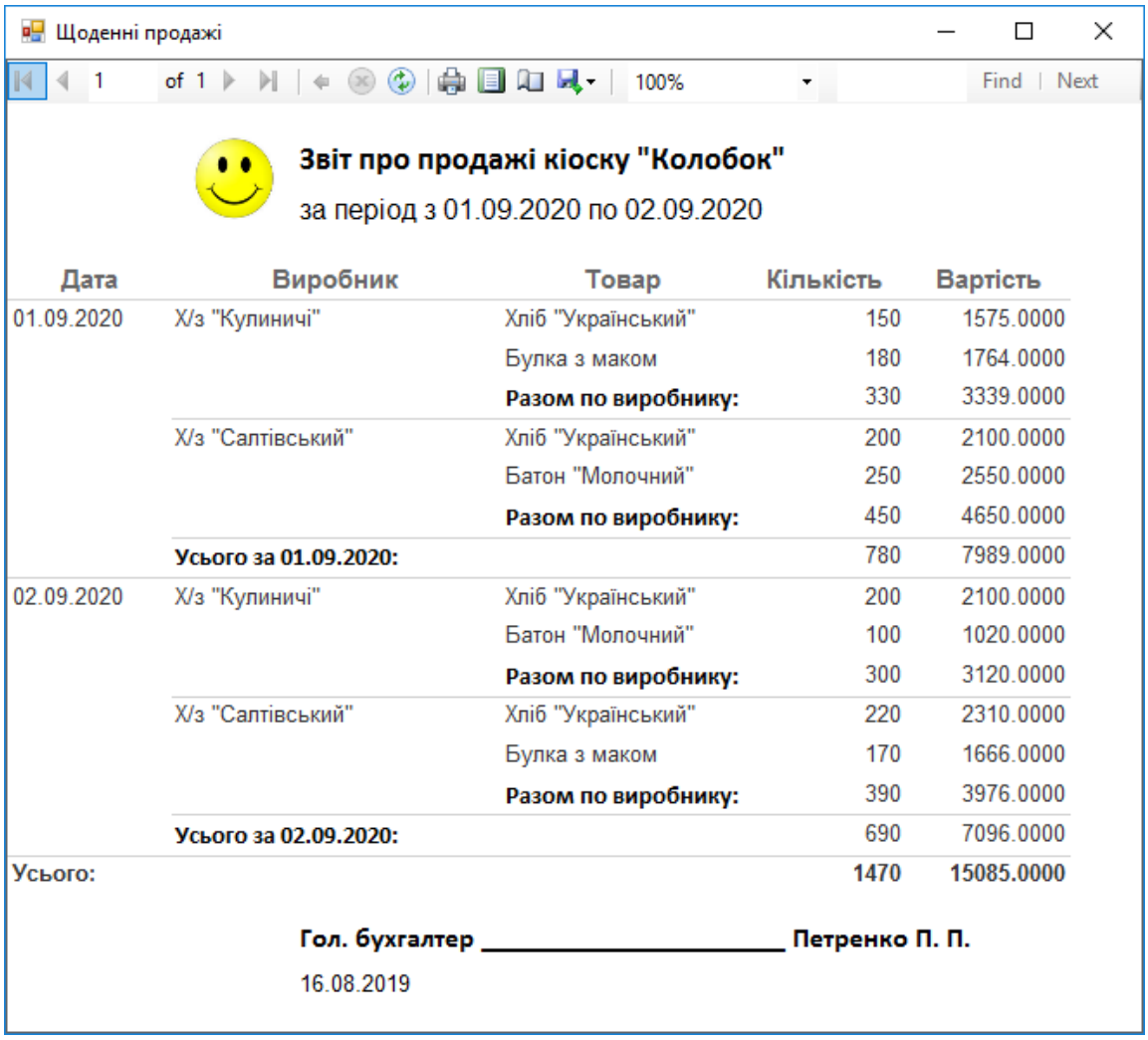
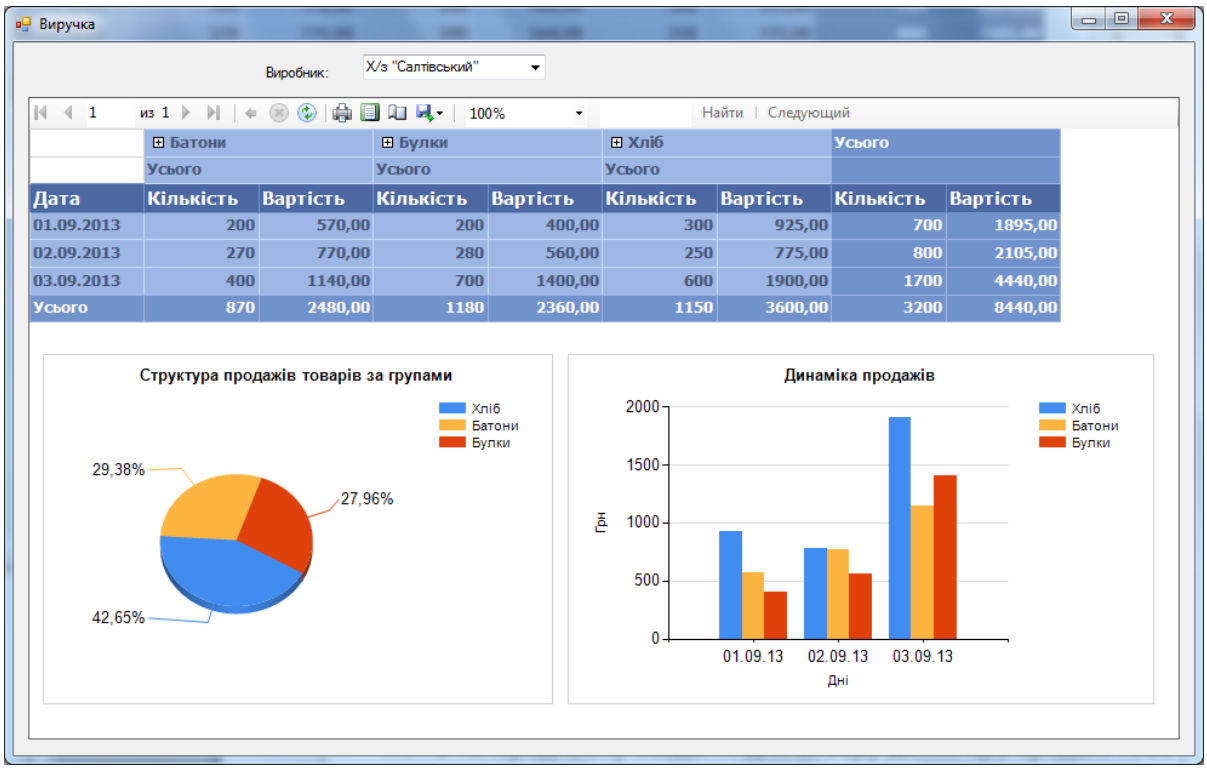


Рис. 7.8. **Звіт про продажі товарів на формі *Щоденні продажі***



	Батони		Булки		Хліб		Усього	
	Усього		Усього		Усього			
Дата	Кількість	Вартість	Кількість	Вартість	Кількість	Вартість	Кількість	Вартість
01.09.2013	200	570,00	200	400,00	300	925,00	700	1895,00
02.09.2013	270	770,00	280	560,00	250	775,00	800	2105,00
03.09.2013	400	1140,00	700	1400,00	600	1900,00	1700	4440,00
Усього	870	2480,00	1180	2360,00	1150	3600,00	3200	8440,00

Рис. 7.9. **Форма для аналізу результатів продажів товарів певного виробника**

1. Підготовчий етап

1.1. Побудова кнопкової форми застосунку

Завдання

Створіть проєкт і побудуйте в ньому кнопкову форму (рис. 7.10).

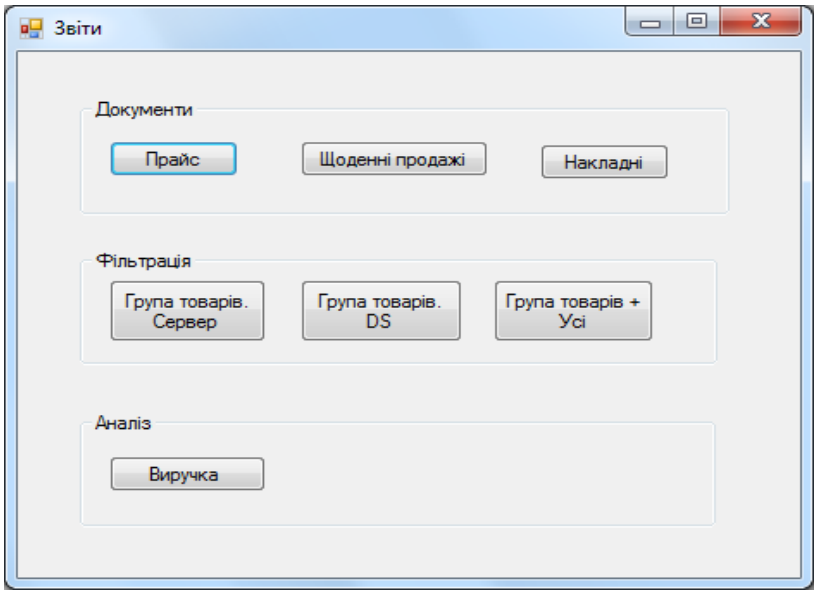


Рис. 7.10. **Кнопкова форма**

Виконання

1. Відкрийте Visual Studio.

2. Створіть проєкт Windows Forms мовою C# з ім'ям ***rptХліб*** і збережіть його.

3. У вікні **Solution Explorer** виділіть значок ***Form1*** і для файлу ***Form1.cs*** у вікні властивостей задайте ім'я ***formЗвіти.cs***.

4. Для властивості **Text** форми задайте значення ***Звіти***.

5. Додайте на форму елемент керування **GroupBox** і задайте значення ***Документи*** для його властивості **Text**.

6. Додайте три кнопки всередину елемента ***Документи*** й установіть для них значення властивостей, наведені в табл. 7.1.

Таблиця 7.1

Властивості кнопок

Кнопки	Властивості	Значення
1	Text	Прайс
	Name	buttonПрайс
2	Text	Щоденні продажі
	Name	buttonЩоденніПродажі
3	Text	Накладні
	Name	buttonНакладні

7. Повторіть п. 5 і 6 для групи кнопок ***Фільтрація***, установивши такі значення імен кнопок: ***buttonГрупаТоварів_Сервер***, ***buttonГрупаТоварів_DS*** та ***buttonГрупаТоварів_Усі***.

8. Повторіть ще раз п. 5 і 6 для групи ***Аналіз***, установивши значення імені кнопки ***buttonВиручка.*** У рамці групи залишіть вільне місце для кнопок викликання форм з іншими видами аналізу, які додадуть під час роботи над завданнями для самостійного виконання.

1.2. Додавання бази даних до нового проєкту

Завдання

Скопіюйте базу даних ***ХлібПрізвище.mdf***, яку використовували під час виконання лабораторної роботи 2.

Виконання

1. Перегляньте вікно **Server Explorer**. Якщо в ньому є значок бази даних ***ХлібПрізвище***, видаліть його, вибравши з його контекстового меню команду **Delete**.

Примітка. Замість слова ***Прізвище*** в імені файлу бази даних має бути прізвище студента, наприклад, ***ХлібПетренко.mdf***. У подальших описах база даних буде мати узагальнене ім'я ***Хліб***.

2. Виберіть команду **Project – Existing Item** у меню Visual Studio.

3. Установіть фільтр **All Files** у вікні **Add New Item**, перейдіть у папку проєкту ***роз'єднанийХліб***, виберіть файл ***Хліб.mdf*** і клацніть кнопку **Add**.

У вікнах **Solution Explorer** і **Server Explorer** з'явилися значки бази даних ***Хліб.mdf***.

4. Перегляньте склад бази даних ***Хліб***. Зверніть увагу на те, щоб вона містила п'ять таких таблиць: ***Товари***, ***Виробники***, ***Продажі***, ***Накладні*** та ***ТовариНакладних***, а в них зберігалися дані про результати надходження і продажів товарів принаймні за два дні від двох виробників.

5. Додайте таблицю ***Групи***, у якій містяться назви груп товарів, а кожний товар належить до однієї з груп, тому дочірню таблицю ***Товари*** пов'язано через зовнішній ключ із батьківською таблицею ***Групи***. Для цього виконайте таке:

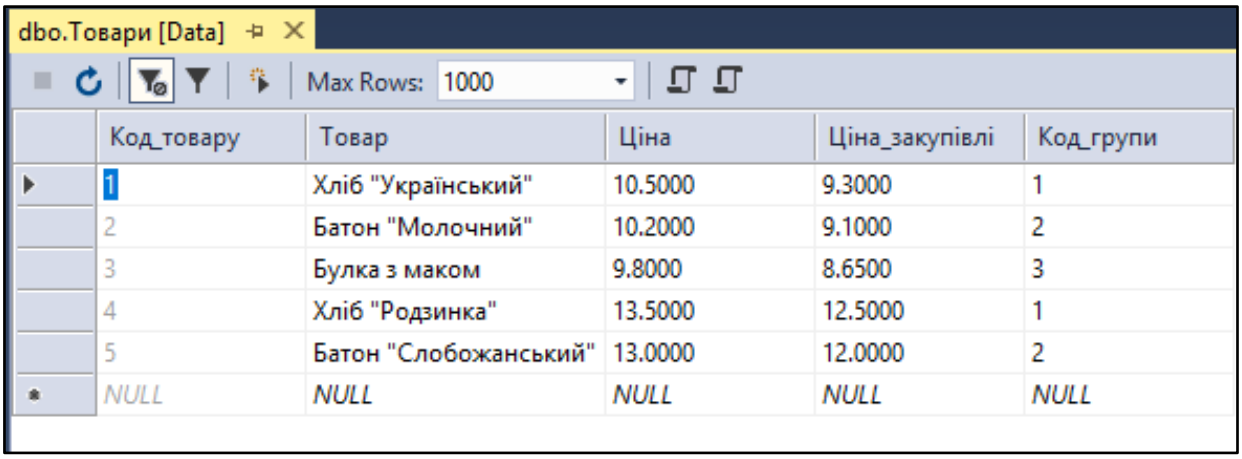
5.1. Для створення і заповнення таблиці ***Групи*** виберіть команду **New Query** з контекстового меню папки **Tables** бази даних у вікні **Server Explorer**, а потім уведіть і виконайте такі запити:

```
-- Створення таблиці Групи
CREATE TABLE [dbo].[Групи]
(
      [Код_групи] INT NOT NULL PRIMARY KEY IDENTITY,
    [Група] NVARCHAR(10) NOT NULL
);

-- Заповнення таблиці Групи
INSERT Групи VALUES
      (N'Хліб'),
      (N'Батон'),
      (N'Булки');
```

5.2. Для додавання зовнішнього ключа ***Код_групи*** в таблицю ***Товари*** введіть і виконайте такий запит:

```
-- Додавання поля Код_групи у таблицю Товари
ALTER TABLE Товари
    ADD Код_групи INT;
```

5.3. За допомогою команди **Show Table Data** з контекстового меню таблиці у вікні **Server Explorer** перегляньте і доповніть дані таблиці ***Товари*** (рис. 7.11).



Код_товару	Товар	Ціна	Ціна_закупівлі	Код_групи
1	Хліб "Український"	10.5000	9.3000	1
2	Батон "Молочний"	10.2000	9.1000	2
3	Булка з маком	9.8000	8.6500	3
4	Хліб "Родзинка"	13.5000	12.5000	1
5	Батон "Слобожанський"	13.0000	12.0000	2
NULL	NULL	NULL	NULL	NULL

Рис. 7.11. **Дані таблиці *Товари***

5.4. Для встановлення за полями ***Код_групи*** зв'язку між таблицею ***Групи*** й таблицею ***Товари*** введіть і виконайте такий запит:

```
-- Установлення зв'язку між таблицями Групи й  Товари
ALTER TABLE Товари
    ADD CONSTRAINT [FK_ГрупиТовари] FOREIGN KEY (Код_групи) REFERENCES
Групи
```

6. Видаліть з'єднання з базою даних ***Хліб.mdf***, вибравши з контекстового меню її значка у вікні **Server Explorer** команду **Delete**.

1.3. Побудова типізованого набору даних

Завдання

Створіть типізований набір даних ***ХлібDataSet***, використовуючи майстра налаштування джерела даних (рис. 7.12).

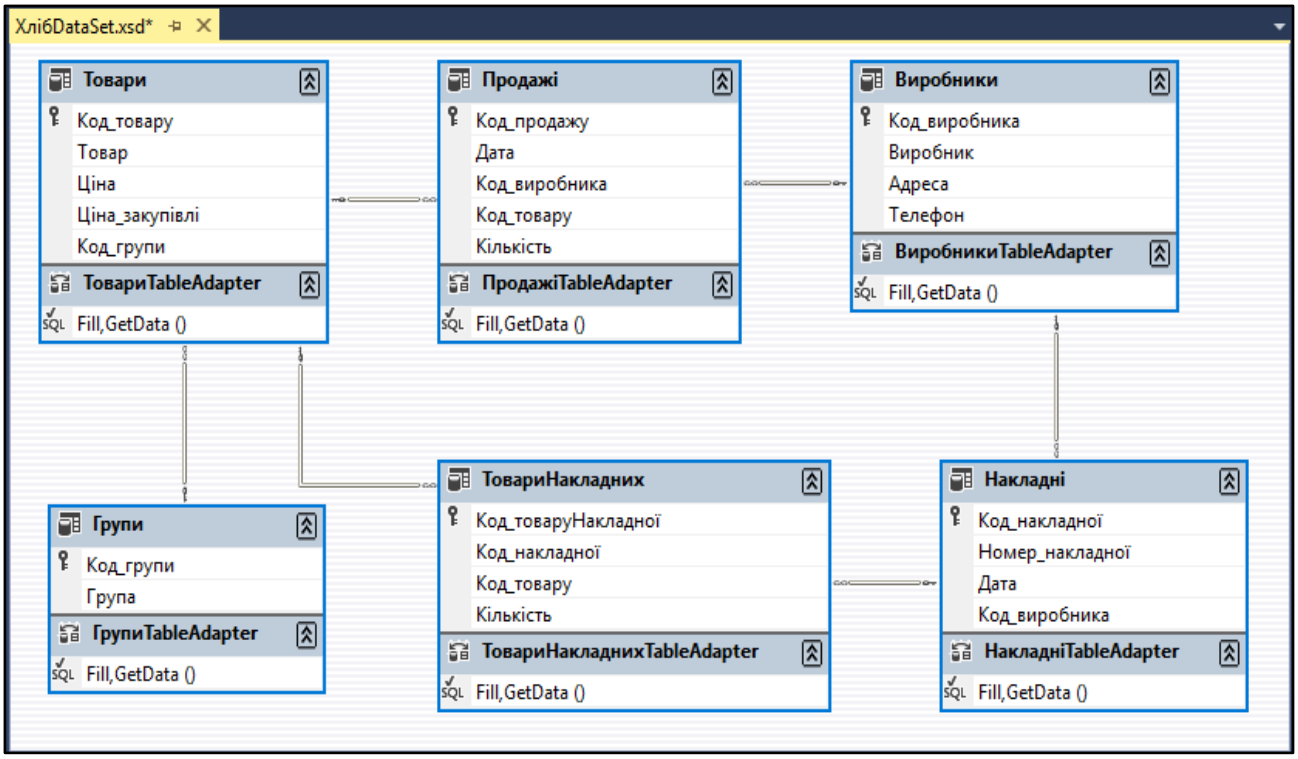


Рис. 7.12. **Схема типізованого DataSet**

Виконання

1. Виберіть команду **Add New Data Source** в меню **Project** (у Visual Studio 2010 – у меню **Data**).

2. Погодьтеся з вибором значків **Database** та **Dataset**, двічі клацнувши кнопку **Next**.

3. Клацніть кнопку **New connection** у вікні **Choose Your Data Connection**.

4. Створіть нове з'єднання з базою даних, що зберігають у файлі ***Хліб.mdf***. Для цього виконайте такі операції у вікні **Add Connection**.

4.1. Перевірте, чи встановлено значення **Microsoft SQL Server Database File (SqlClient)** у полі **Data source**. Якщо ні, то скористайтеся кнопкою **Change**.

4.2. Установіть шлях до файлу бази даних ***Хліб.mdf***, що перебуває в папці проєкту ***rptХліб***. Для цього скористайтеся кнопкою **Browse**.

5. Після повернення у вікно **Choose Your Data Connection** клацніть кнопку **Next**, а потім ще раз у наступному вікні майстра.

6. Установіть прапорець у вузлі **Tables** у вікні **Choose Your Data Objects**, а потім клацніть кнопку **Finish**.

У вікні **Solution Explorer** з'явився вузол ***ХлібDataSet.xsd***, що становить типізований набір даних

7. Збережіть зміни, зроблені у проєкті.

1.4. Додавання засобів роботи зі звітами (для Visual Studio 2017)

Завдання

Додайте в середовище Visual Studio 2017 засоби для побудови та відображення звітів.

Ідеї та кроки виконання

У разі першого встановлення Visual Studio 2017 відсутні засоби роботи зі звітом. Необхідно додати конструктора і майстра для побудови звітів, а також елемент керування для перегляду звітів. Ці роботи виконують у два етапи:

1. Додавання конструктора і майстра побудови звітів.
2. Додавання елемента керування ReportViewer.

Виконання

1. Додайте в середовище Visual Studio 2017 конструктора і майстра для побудови звітів. Для цього виконайте такі дії:

1.1. Виберіть команду **Extensions and Updates** у меню **Tools**.

1.2. Виберіть елемент **Online** в лівому списку вікна **Extensions and Updates** і введіть значення ***rdlc*** у поле **Search**. Потім виділіть елемент **Microsoft Rdlc Report Designer for Visual Studio** і клацніть кнопку **Download** у центральній частині вікна. Далі дотримуйтеся команд майстра встановлення.

1.3. Перевірте наявність конструктора і майстра для побудови звітів, вибравши в контекстному меню проєкту команду **Add – New Item**, а потім – елемент **Visual C# Item** у лівому списку вікна **Add New Item** та знайдіть елементи **Report** і **Report Wizard** у центральній частині вікна (рис. 7.13).

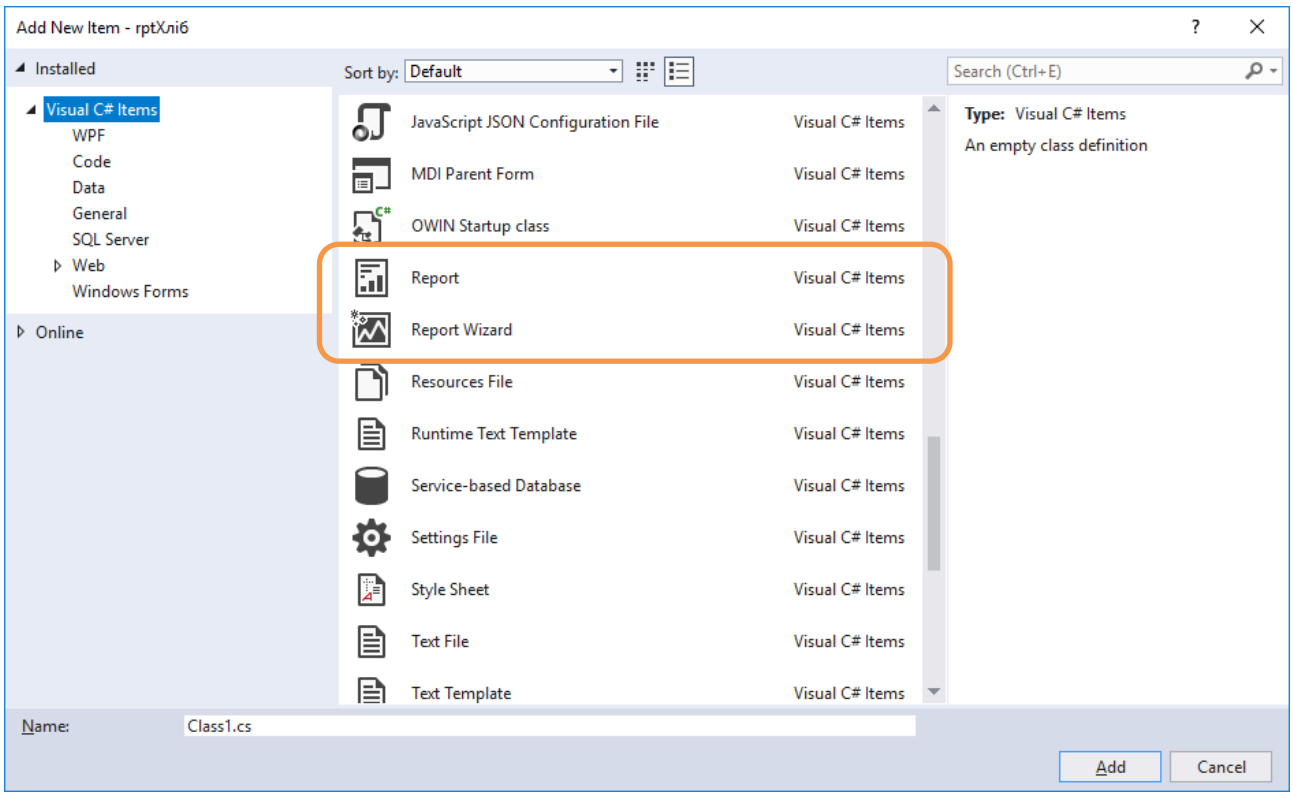


Рис. 7.13. **Елементи *Report* і *Report Wizard* у вікні *Add New Item***

1.4. Закрийте вікно **Add New Item**, клацнувши кнопку **Cancel**.

2. Додайте на панель **Toolbox** елемент керування **ReportViewer**. Для цього виконайте такі дії:

2.1. Клацніть правою клавішею мишки на елементі **References** у вікні **Solution Explorer** і в контекстному меню виберіть команду **Manage NuGet Packages** (рис. 7.14).

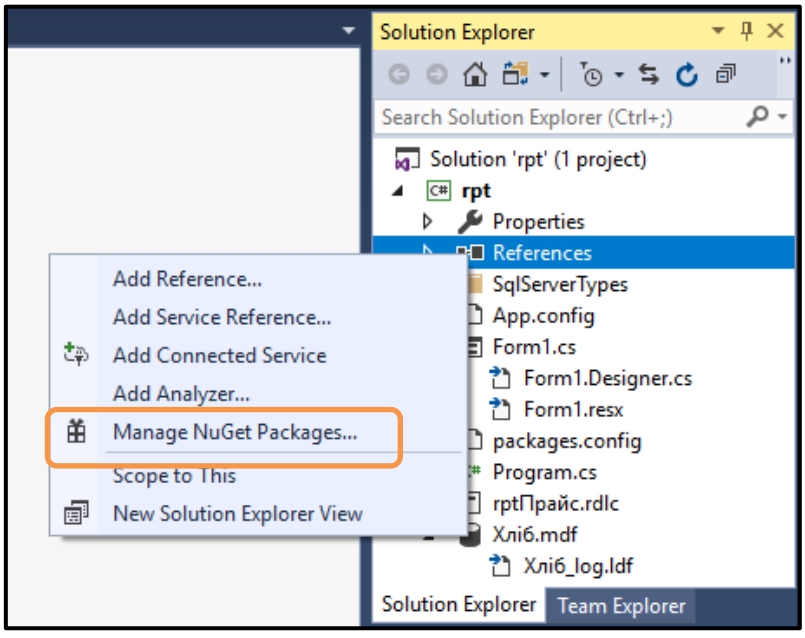


Рис. 7.14. **Команда *Manage NuGet Packages***

2.2. Уведіть значення ***Reportviewercontrol*** у поле **Search** вікна **NuGet Package Manager** і клацніть кнопку **Browse**. Потім виберіть елемент **ReportViewerControl.Winforms** і клацніть кнопку **Install** (рис. 7.15).

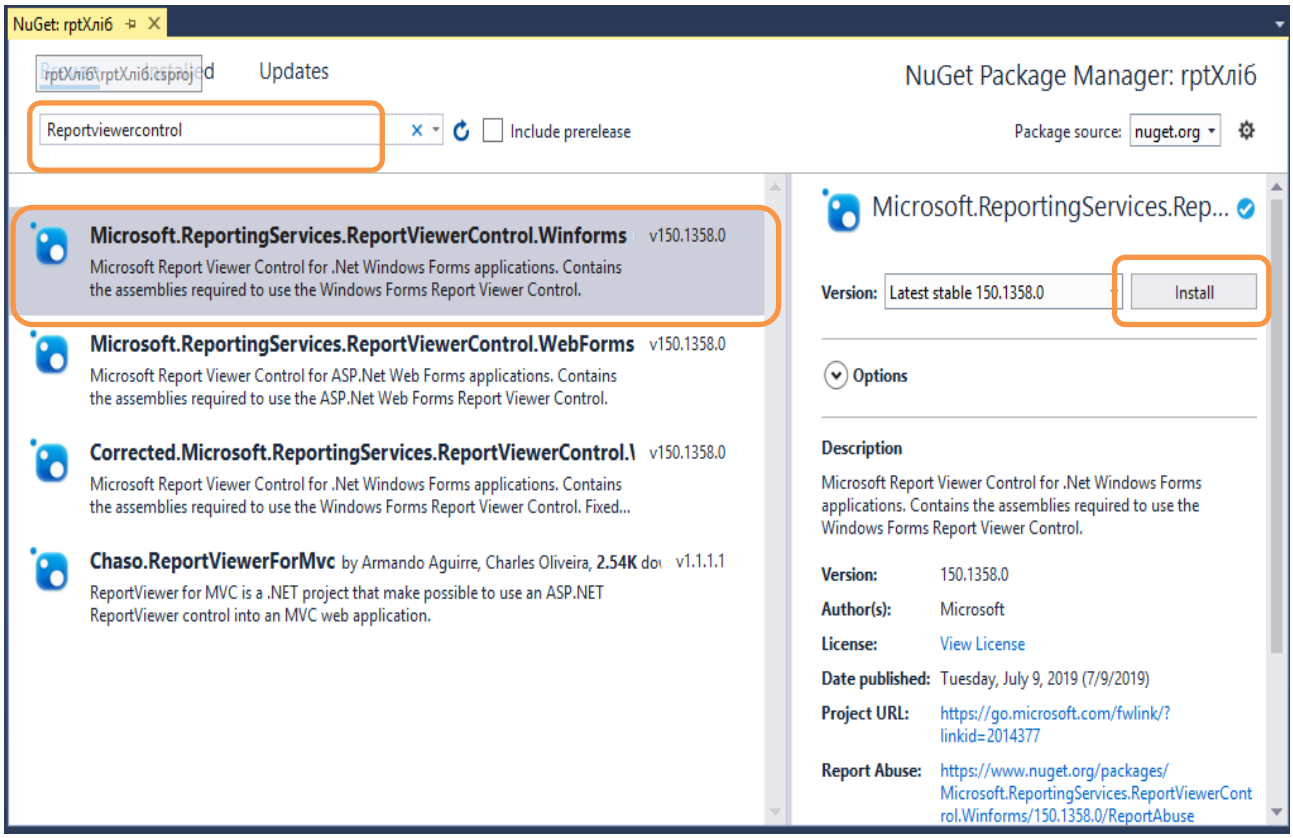


Рис. 7.15. **Вибір пакета *ReportViewerControl.Winforms***

2.3. Далі дотримуйтеся команд майстра встановлення, а потім закрийте вікно **NuGet Package Manager**.

2.4. У разі відкритого вікна конструктора форм перейдіть на панель елементів керування **Toolbox**, викличте контекстове меню для елемента **General** і виберіть команду **Add Tab**. Уведіть назву нового розділу ***Reporting***.

2.5. Викличте контекстне меню для елемента **Reporting** і виберіть команду **Choose Items**.

2.6. Натисніть кнопку **Browse** у вікні **Choose Toolbox Items**. Перейдіть у папку рішення, у якому ви зараз перебуваєте, у ній – у папку **packages**, потім – у папку **Microsoft.ReportingServices.ReportViewerControl.Winforms**, після цього – у папку **lib**, у ній – у папку **net40**, а в останній

виділіть файл **Microsoft.ReportViewer.WinForms.dll** і клацніть кнопку **Open**, а потім – кнопку **ОК**. У групі **Reporting** з'явився елемент **Report-Viewer**.

Примітка. У разі використання елемента **ReportViewer** у Visual Studio 2017 у його смарт-тегу відсутня можливість створення звіту. Тут новий звіт створюють тільки командою **Add – New Item**, вибравши її в контекстному меню проєкту.

У звіті з лабораторної роботи подайте скриншот схеми типізованого набору даних. Поясніть призначення нових елементів.

2. Побудова документів

2.1. Простий табличний звіт і друкований список

Завдання

На основі табличного звіту побудуйте список, у якому містяться всі товари кіоску "Колобок" із їхніми цінами (рис. 7.16), а також передбачте друкування даних про кожний товар у вигляді цінника (рис. 7.17). Цінник подати за допомогою таблікса список.

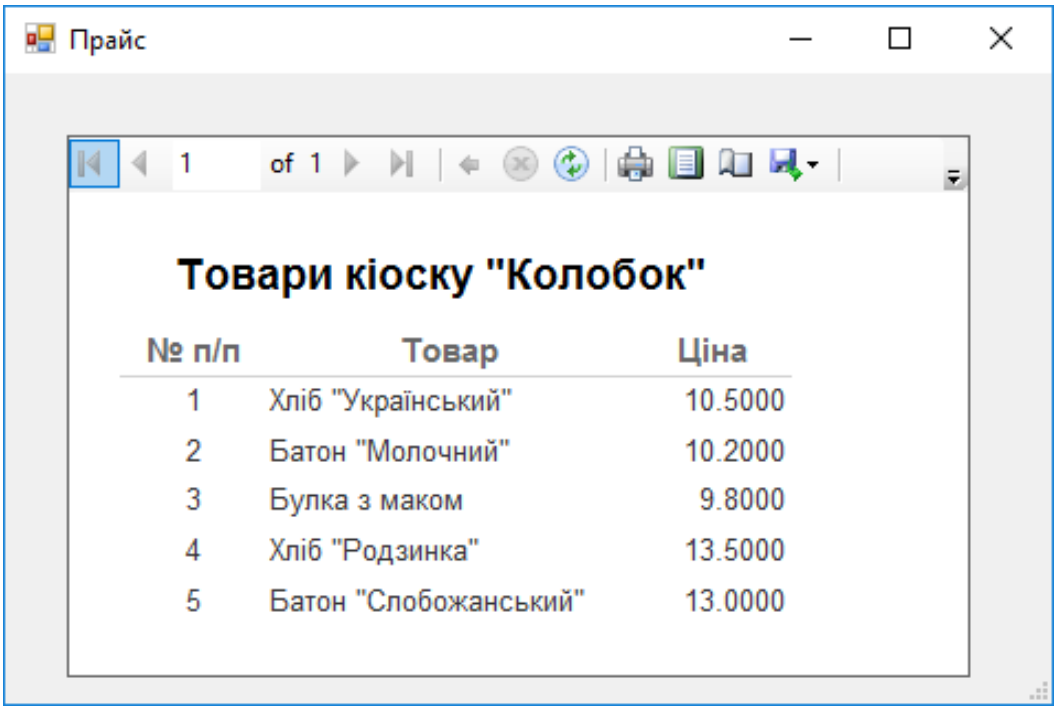


Рис. 7.16. **Список цін товарів**

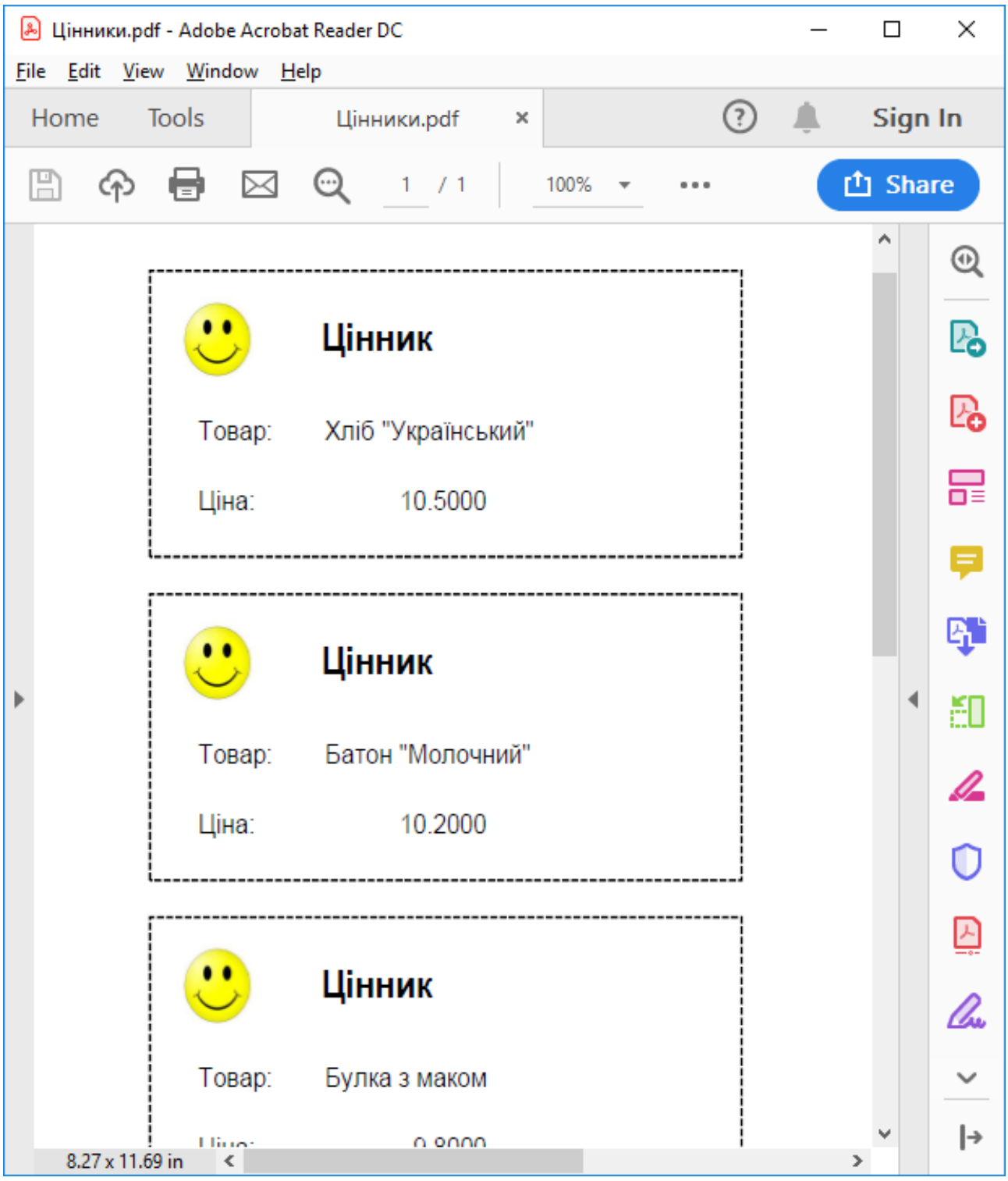


Рис. 7.17. **Цінники на товари кіоску "Колобок"**

Ідеї виконання

Для відображення списку товарів і виведення цінників побудуйте звіти ***rptПрайс*** та ***rptЦінник***. Перший звіт подайте у вигляді таблиці,

а другий – у вигляді списку. Звіт ***rptПрайс*** будуть відображати в елементі ReportViewer, а звіт ***rptЦінник*** – виводити безпосередньо на друк.

Обидва звіти побудуйте на даних, що зберігають у таблиці ***Товари***, яку вже подано в наборі даних ***ХлібDataSet***. Тому немає необхідності будувати окреме джерело даних.

Для створення звіту ***rptПрайс*** використайте майстра звітів, а звіт ***rptЦінник*** побудуйте вручну.

Виконання

1. Створіть звіт, у якому містяться всі товари з їхніми цінами. Для цього:

1.1. Виконайте команду **Project – Add New Item**.

1.2. Виберіть елемент **Visual C# Items** у лівому списку вікна **Add New Item,** виділіть елемент **Report Wizard** у центральній частині вікна, уведіть ім'я ***rptПрайс*** і клацніть кнопку **Add** (рис. 7.18). З'явилося перше вікно майстра звітів **Report Wizard**.

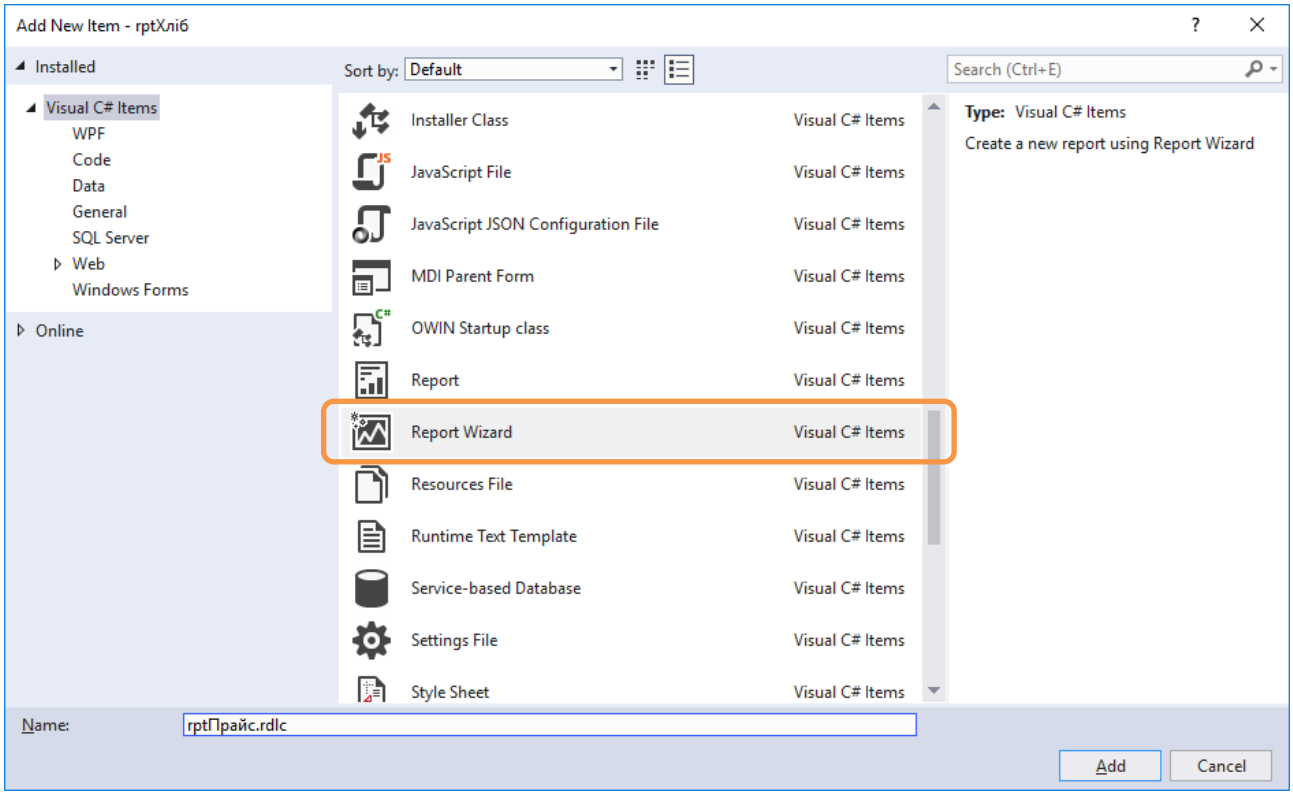


Рис. 7.18. **Елемент *Report Wizard* у вікні *Add New Item***

1.3. Установіть значення елементів у першому вікні **Report Wizard** (рис. 7.19), наведені в табл. 7.2.

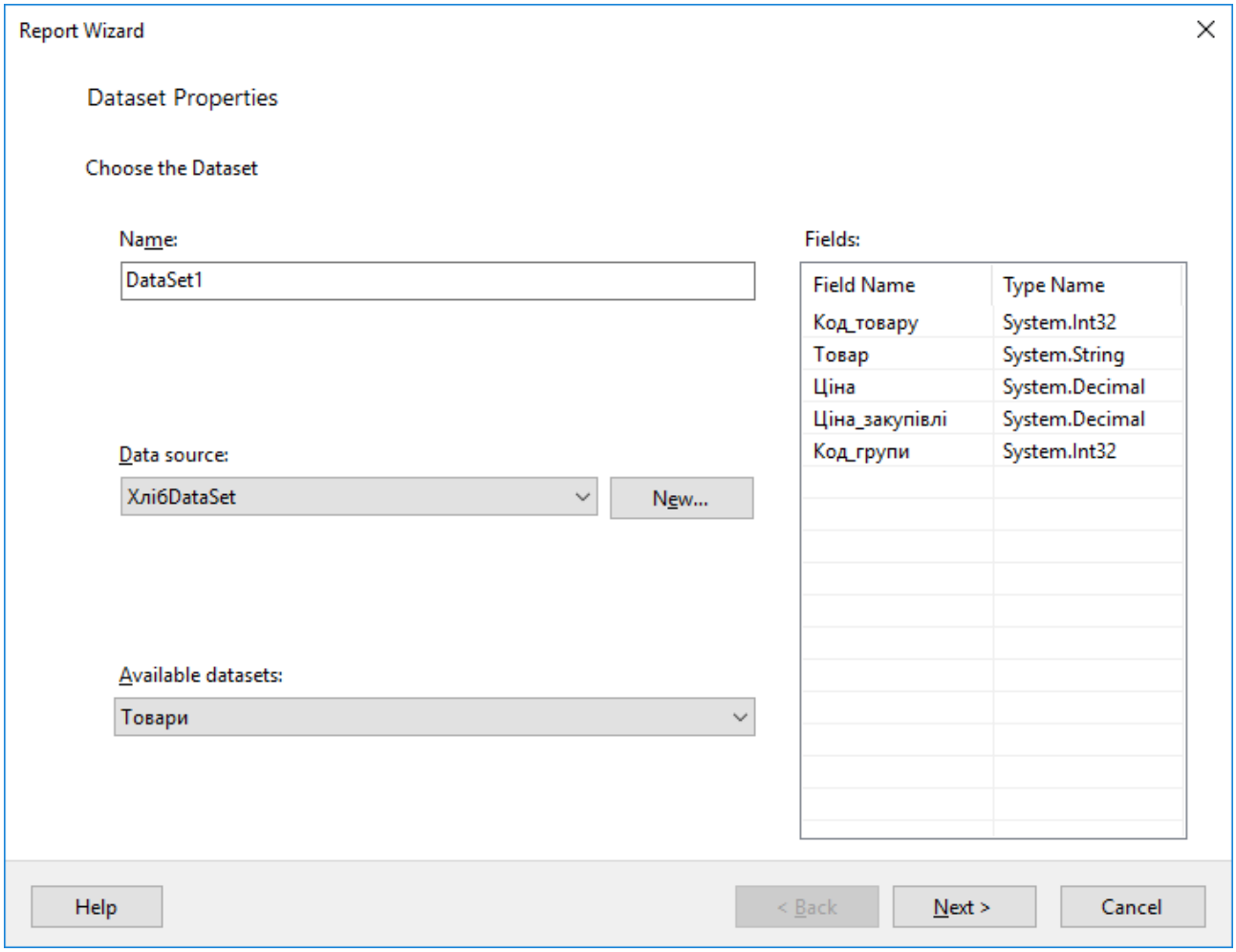


Рис. 7.19. **Перше вікно *Report Wizard***

Таблиця 7.2

Значення елементів

Елементи	Значення
Datasource	хлібDataSet
Available datasets	Товари

1.4. Перетягніть поля ***Товар*** і ***Ціна*** зі списку **Available fields** у список **Values** у другому вікні майстра (рис. 7.20).

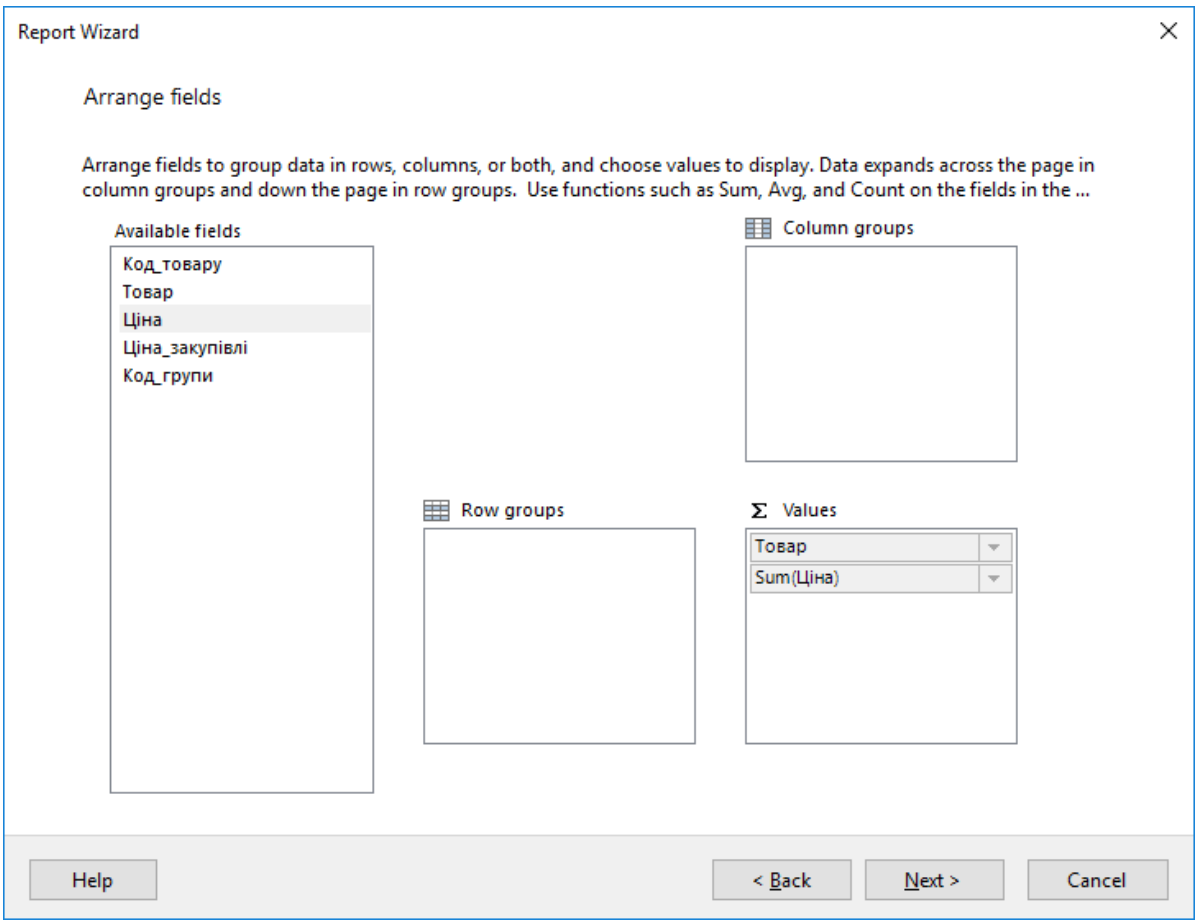


Рис. 7.20. **Друге вікно *Report Wizard***

1.5. Клацніть кнопку **Next** у третьому вікні майстра, а в четвертому – кнопку **Finish**. У вікні **Solution Explorer** з'явився значок звіту (файл з ім'ям ***rptПрайс.rdlc***).

1.6. Змініть значення ***[Sum(Ціна)]*** на ***[Ціна]*** у вікні конструктора звіту (рис. 7.21).

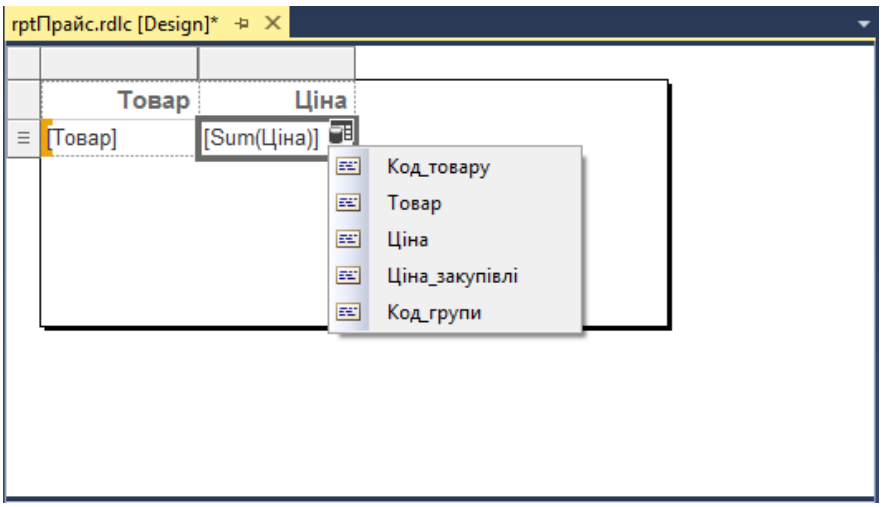


Рис. 7.21. **Зміна значення *[Sum(Ціна)]* на *[Ціна]***

1.7. Розширте стовпець ***Товар***, щоб назви товарів у ньому відображали в одному рядку, скориставшись лінією розділу між стовпцями ***Товар*** і ***Ціна*** у верхній частині таблікса.

2. Додайте ще один стовпець до таблікса, у якому будуть відображатися номери товарів. Для цього:

2.1. Клацніть правою клавішею мишки на будь-якій клітинці стовпця ***Товар*** і виберіть із контекстового меню команду **Insert Column – Left**.

2.2. Уведіть текст ***№ п/п*** у заголовковій клітинці нового стовпця.

2.3. Клацніть правою клавішею мишки на клітинці для даних стовпця ***№ п/п*** і виберіть із контекстового меню команду **Expression**.

2.4. Уведіть значення ***=RowNumber(Nothing)***.

Примітки:

1) функція ***RowNumber*** міститься в категорії **Common Functions – Miscellaneous**;

2) значення аргументу ***Nothing*** означає, що нумерацію виконують підряд для всіх рядків. Якщо дані зібрано у групи рядків, то в кожній групі можна починати свою нумерацію. Для цього вказують ім'я групи як значення аргументу функції ***RowNumber***.

2.5. Розмістіть значення стовпця ***№ п/п*** по центру, указавши значення **Center** для його властивості **TextAlign**. Таке саме значення встановіть для назв стовпців.

3. Додайте заголовок у звіт. Для цього:

3.1. Перетягніть таблікс нижче, щоб звільнити місце для заголовка.

3.2. Додайте до звіту з панелі **Toolbox** елемент **Text Box**, розмістивши його над табліксом.

3.3. Уведіть текст заголовка ***Товари кіоску "Колобок"*** у нове текстове поле.

3.4. Розташуйте заголовок по центру над табліксом і встановіть розмір шрифту 14 пт та напівжирного накреслення (рис. 7.22).

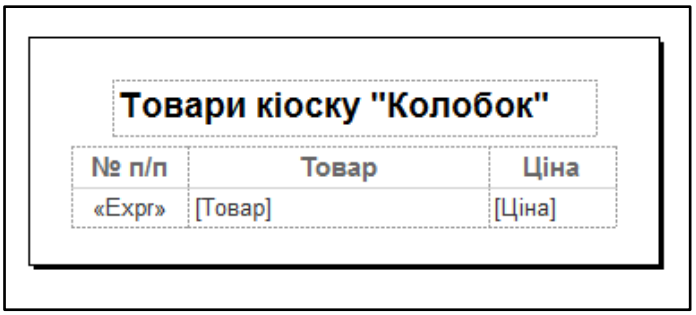


Рис. 7.22. **Звіт *rptПрайс* у вікні конструктора**

3.5. Збережіть зміни у звіті та закрийте його вікно.

4. Додайте у проєкт нову форму з ім'ям файлу ***formПрайс.cs*** і значенням ***Прайс*** властивості **Text**.

5. Для відображення даних про товари додайте на форму елемент **ReportViewer**, що міститься на панелі елементів у групі **Reporting**. За замочуванням він дістане ім'я ***reportViewer1***.

Примітка. Якщо після додавання елемента **ReportViewer** на форму з'являється тільки компонент під формою, а сам елемент не відображено на формі, спробуйте більш стабільну версію елемента. Щоб установити її у проєкті, уведіть у вікні **Package Manager Console** і виконайте таку команду:

```
Install-Package Microsoft.ReportingServices.ReportViewerControl.Winforms -Version 140.340.80
```

Після цього виконайте п. 2.5 і 2.6 підрозділу 1.4. Потім повторіть спробу додавання на форму елемента **ReportViewer**.

6. Перейдіть у вікно конструктора форми ***Прайс*** та у смарт-тегу **ReportViewer Tasks** виберіть значення ***rptХліб.rptПрайс.rdlc*** у полі зі списком **Choose Report**.

7. Перейдіть у вікно конструктора форми ***Звіти***, двічі клацніть кнопку ***Прайс*** і додайте код оброблювача події "Клацання на кнопці Прайс".

```
private void buttonПрайс_Click(object sender, EventArgs e)
{
    formПрайс вікноПрайс = new formПрайс();
    вікноПрайс.ShowDialog();
}
```

8. Перевірте функціональність форми ***Прайс***. Має з'явитися форма, подана на рис. 7.23.

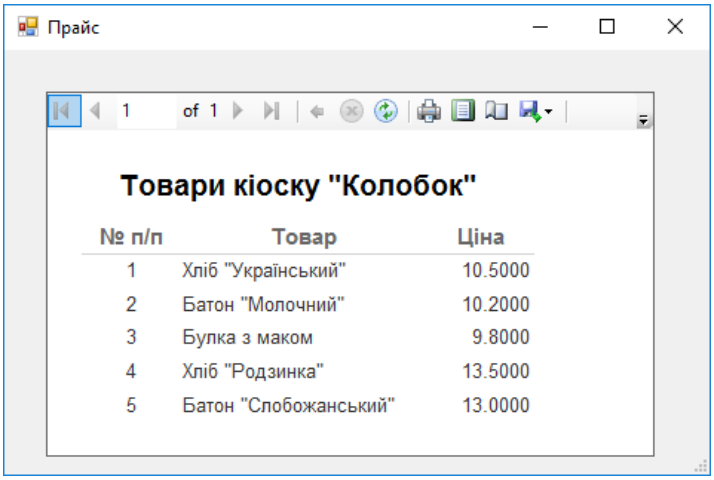


Рис. 7.23. **Звіт *rptПрайс* у вікні форми *Прайс***

9. Додайте до проєкту звіт ***rptЦінник***, призначений для друкування даних про кожний товар у вигляді цінника (рис. 7.24).

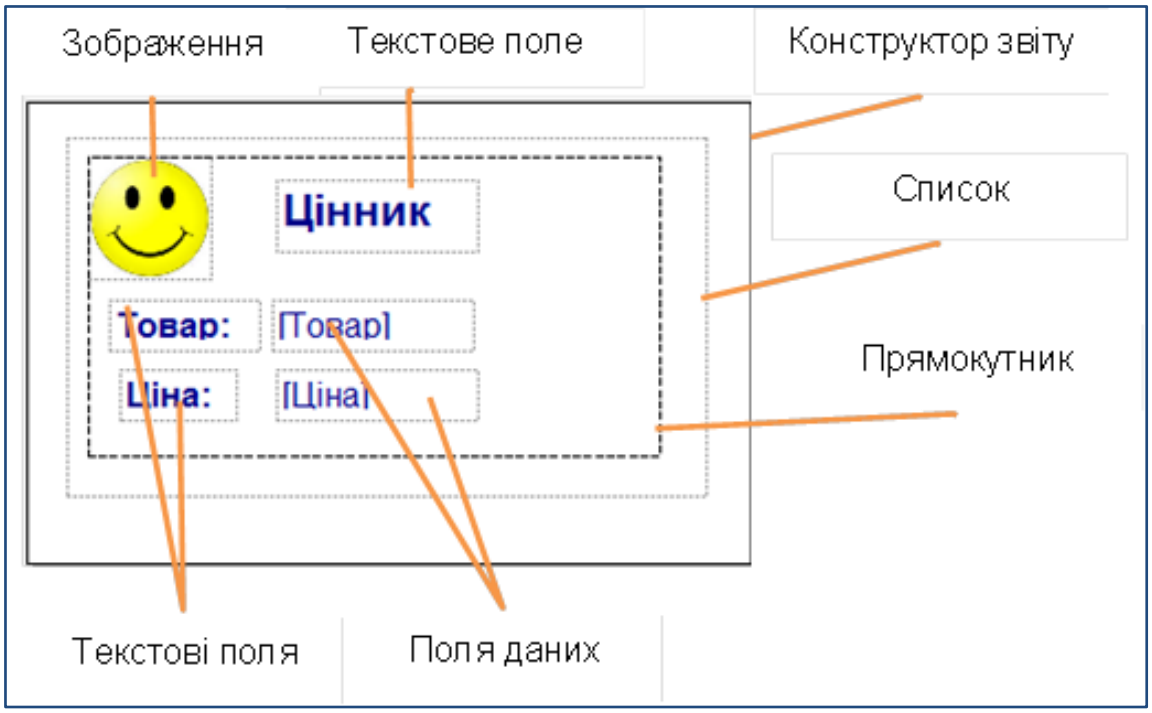


Рис. 7.24. **Елементи звіту *rptЦінник***

Для цього:

9.1. Виконайте команду **Project – Add New Item**.

9.2. Виберіть елемент **Visual C# Items** у лівому списку вікна **Add New Item,** виділіть елемент **Report** у центральній частині вікна, уведіть ім'я ***rptЦінник*** і клацніть кнопку **Add**. З'явилося порожнє вікно конструктора звітів.

9.3. Перетягніть елемент **List** із панелі елементів в область конструктора звіту й у вікні **Dataset Properties** установіть значення елементів, наведених у табл. 7.3.

Таблиця 7.3

Значення елементів

Елементи	Значення
Datasource	хлібDataSet
Available datasets	Товари

9.4. Додайте елемент **Rectangle** всередину елемента **List**. Він буде обмежувати кожен цінник. Установіть значення ***Dashed*** для властивості прямокутника **BorderStyle – Default**.

9.5. Додайте логотип кіоску і заголовок ***Цінник*** у верхню частину прямокутника, скориставшись елементами **Image** і **Text Box**.

9.6. Відкрийте вікно **Данные отчета** (можна скористатися комбінацією клавіш **Ctrl+Alt+D**) і перетягніть поля ***Товар*** і ***Ціна***. Перед ними поставте відповідні написи за допомогою елементів **Text Box**.

Примітка. Під час побудови звіту скористайтеся панеллю **Report Formatting Toolbar Options** для надання звіту потрібного вигляду.

10. Додайте на форму ***formПрайс*** кнопку ***Друкування цінників*** і код оброблювача події клацання на ній, а також код, що забезпечує друкування даних про кожний товар у вигляді цінника:

```
private void buttonДрукування_цінників_Click(object sender, EventArgs e)
{
    PrintPriceList();
}

// Routine to provide to the report renderer, in order to
//    save an image for each page of the report.
private Stream CreateStream(string name,
    string fileNameExtension, Encoding encoding,
    string mimeType, bool willSeek)
{
    Stream stream = new FileStream(@"..\..\" + name +
    "." + fileNameExtension, FileMode.Create);
    m_streams.Add(stream);
    return stream;
}
// Export the given report as an EMF (Enhanced Metafile) file.
private void Export(LocalReport report)
{
    string deviceInfo =
        "<DeviceInfo>" +
        "  <OutputFormat>EMF</OutputFormat>" +
        "  <PageWidth>8.5in</PageWidth>" +
        "  <PageHeight>11in</PageHeight>" +
        "  <MarginTop>0.25in</MarginTop>" +
        "  <MarginLeft>0.25in</MarginLeft>" +
        "  <MarginRight>0.25in</MarginRight>" +
        "  <MarginBottom>0.25in</MarginBottom>" +
        "</DeviceInfo>";
```

```
    Warning[] warnings;
    m_streams = new List<Stream>();
    report.Render("Image", deviceInfo, CreateStream, out warnings);
    foreach (Stream stream in m_streams)
        stream.Position = 0;
}

// Handler for PrintPageEvents
private void PrintPage(object sender, PrintPageEventArgs ev)
{
    Metafile pageImage = new
    Metafile(m_streams[m_currentPageIndex]);

    // Adjust rectangular area with printer margins.
    Rectangle adjustedRect = new Rectangle(
        ev.PageBounds.Left - (int)ev.PageSettings.HardMarginX,
        ev.PageBounds.Top - (int)ev.PageSettings.HardMarginY,
        ev.PageBounds.Width,
        ev.PageBounds.Height);

    // Draw a white background for the report
    ev.Graphics.FillRectangle(Brushes.White, adjustedRect);

    // Draw the report content
    ev.Graphics.DrawImage(pageImage, adjustedRect);

    // Prepare for the next page. Make sure we haven't hit the end.
    m_currentPageIndex++;
    ev.HasMorePages = (m_currentPageIndex < m_streams.Count);
}

private int m_currentPageIndex;
private IList<Stream> m_streams;

private void Print()
{
    if (m_streams == null || m_streams.Count == 0)
        throw new Exception("Error: no stream to print.");
    PrintDocument printDoc = new PrintDocument();
    if (!printDoc.PrinterSettings.IsValid)
    {
       throw new Exception("Error: cannot find the default printer.");
    }
    else
    {
        printDoc.PrintPage += new PrintPageEventHandler(PrintPage);
        m_currentPageIndex = 0;
        printDoc.Print();
    }
```

```
}
// Create a local report for Report.rdlc, load the data,
//    export the report to an .emf file, and print it.
private void PrintPriceList()
{
    LocalReport report = new LocalReport();
    report.ReportPath = @"..\..\rptЦінник.rdlc";
    report.DataSources.Add(
        new ReportDataSource("DataSet1", ТовариBindingSource));
    Export(report);
    m_currentPageIndex = 0;
    Print();
}

public void Dispose()
{
    if (m_streams != null)
    {
        foreach (Stream stream in m_streams)
            stream.Close();
        m_streams = null;
    }
}
```

Примітки:

1. Для забезпечення роботи об'єктів коду додайте в описи просторів імен ще й такі:

```
//Для друкування звіту
using Microsoft.Reporting.WinForms;
using System.IO;
using System.Drawing.Imaging;
using System.Drawing.Printing;
```

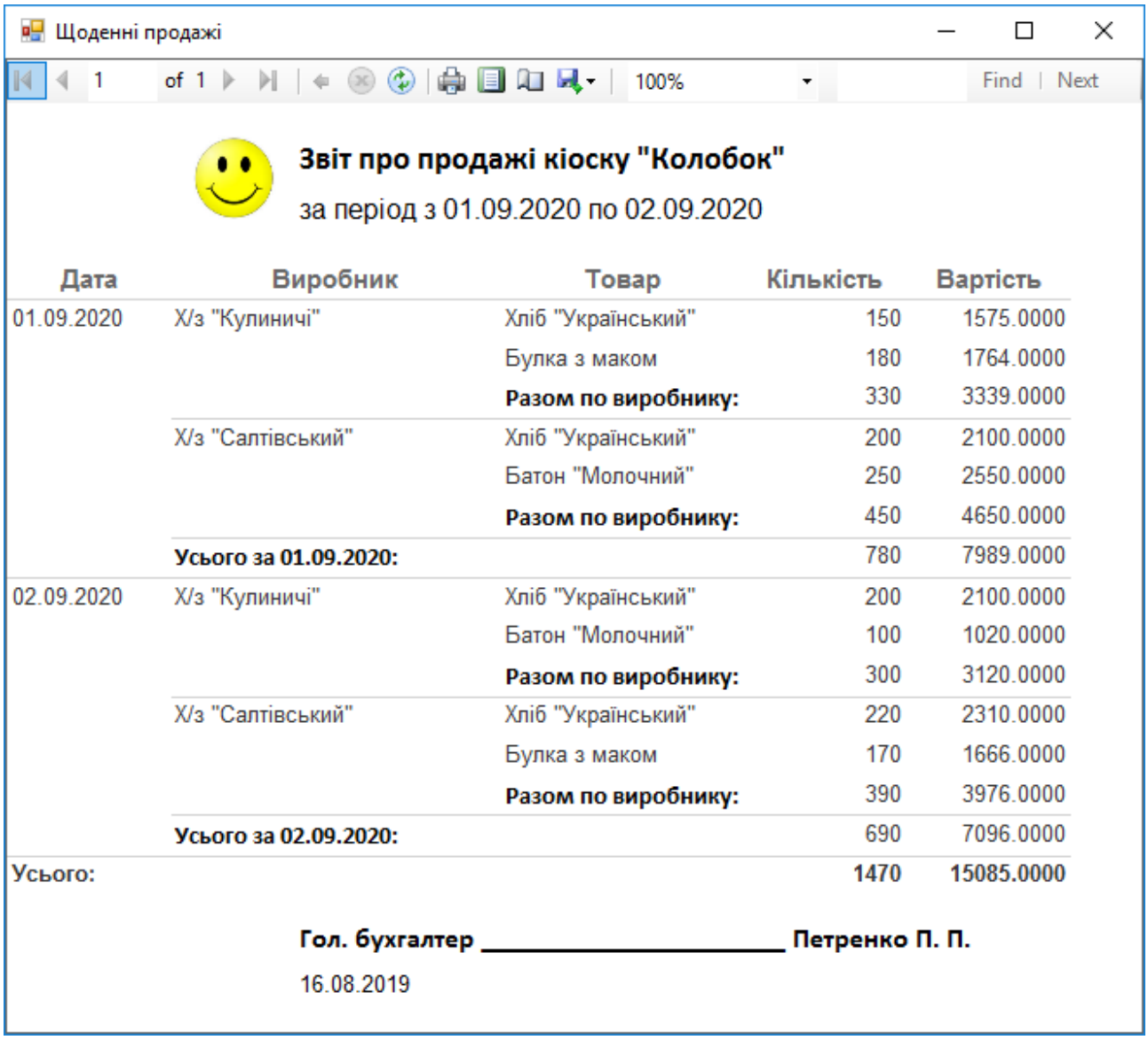
2. Текст коду взято із MSDN, тому подано без перекладу.

11. Перевірте функціональність кнопки ***Друкування цінників***. Якщо на комп'ютері для друкування за замовчуванням установлено якийсь застосунок (OneNote, Microsoft XPS Document Writer тощо), укажіть ім'я файлу, у якому збережуть видані дані. Потім їх можна буде надрукувати. На рис. 7.14 показано вміст файлу в Adobe Acrobat Reader, збереженому з розширенням *.pdf.

2.2. Багаторівневий звіт

Завдання

Побудуйте багаторівневий звіт ***rptЩоденні_продажі***, у якому містяться результати продажів товарів за увесь час роботи кіоску "Колобок" (рис. 7.25).



Дата	Виробник	Товар	Кількість	Вартість
01.09.2020	Х/з "Кулиничі"	Хліб "Український"	150	1575.0000
		Булка з маком	180	1764.0000
		Разом по виробнику:	330	3339.0000
	Х/з "Салтівський"	Хліб "Український"	200	2100.0000
		Батон "Молочний"	250	2550.0000
		Разом по виробнику:	450	4650.0000
	Усього за 01.09.2020:		780	7989.0000
02.09.2020	Х/з "Кулиничі"	Хліб "Український"	200	2100.0000
		Батон "Молочний"	100	1020.0000
		Разом по виробнику:	300	3120.0000
	Х/з "Салтівський"	Хліб "Український"	220	2310.0000
		Булка з маком	170	1666.0000
		Разом по виробнику:	390	3976.0000
	Усього за 02.09.2020:		690	7096.0000
Усього:			**1470**	**15085.0000**

Рис. 7.25. **Звіт *rptЩоденніПродажі***

Ідеї виконання

У звіті ***rptЩоденні_продажі*** використовують дані з таблиць ***Продажі***, ***Товари*** та ***Виробники***. Тому до набору даних ***ХлібDataSet*** попередньо додайте таблицю ***ЩоденніПродажі***, у якій зібрано дані.

Для відображення кінцевих дат періоду, за який подано дані, використайте функції **Min** і **Max** для поля ***Дата***, а для зазначення дати складання звіту – функцію **Today**.

Щоб звіт відразу відображався в розгорнутому вигляді, зніміть прапорець **Expand/collapse groups** під час побудови його майстром. Якщо цей момент пропущено, можна скористатися властивостями відповідної підгрупи. Для цього потрібно з контекстового меню підгрупи, розташованої під вікном конструктора звіту, вибрати команду **Group Properties** і в категорії **Visibility** зняти прапорець **Display can be toggled by this report item**, а також вибрати перемикач **Show** у групі **When the report initially run** (рис. 7.26).

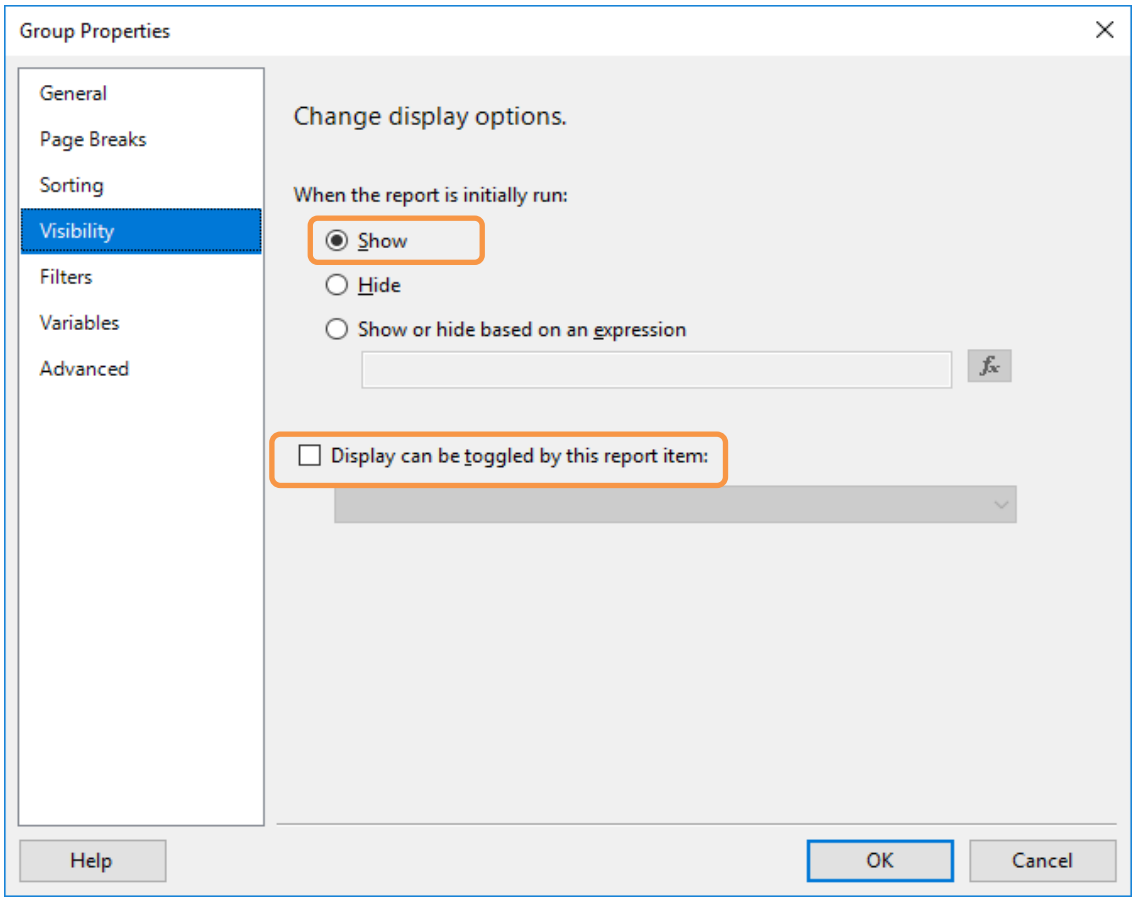


Рис. 7.26. **Вікно налаштування властивості розгорнутого вигляду групи**

Оскільки звіт має форму таблиці, почніть його будувати за допомогою майстра.

Зовнішній вигляд звіту ***rptЩоденніПродажі*** в конструкторі подано на рис. 7.27.

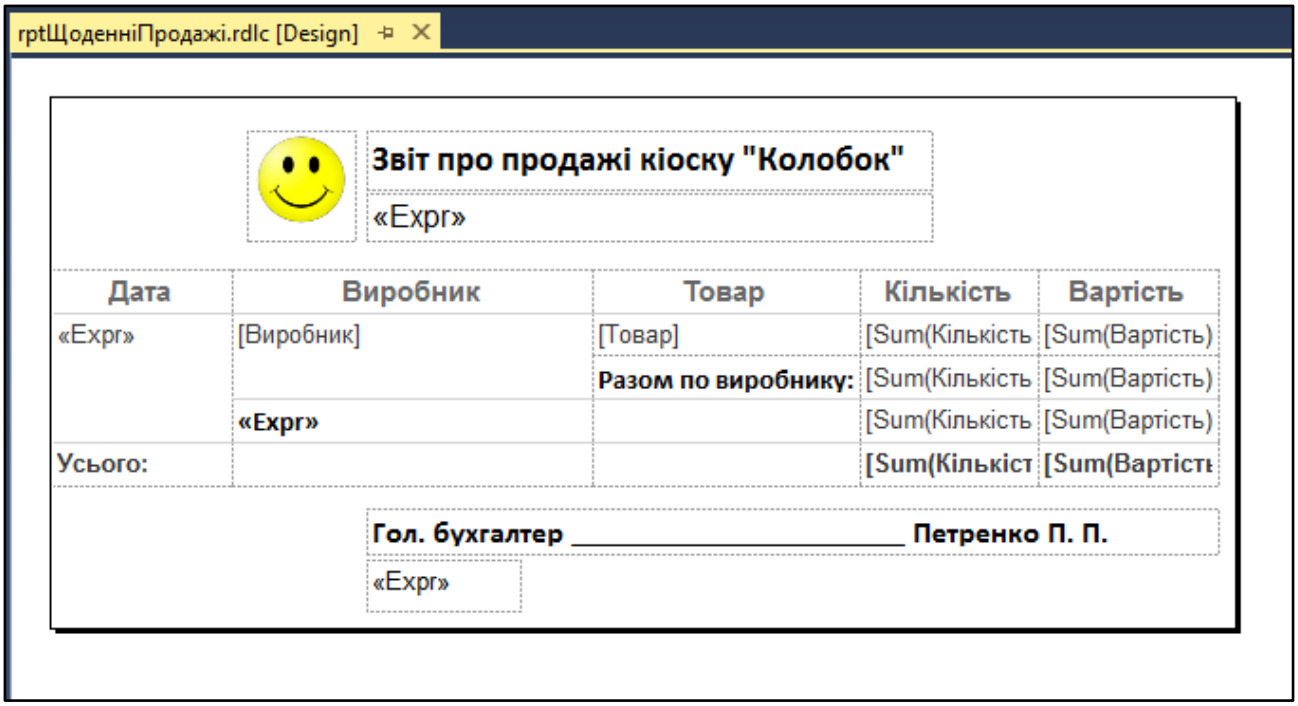


Рис. 7.27. **Звіт *rptЩоденніПродажі* в конструкторі**

Виконання

1. Відкрийте в конструкторі набір даних ***ХлібDataSet*** і додайте до нього таблицю ***ЩоденніПродажі***. Для цього:

1.1. Перетягніть елемент **TableAdapter** із панелі елементів **Toolbox** у вільне місце конструктора набору даних.

1.2. У першому вікні майстра налаштування адаптера таблиці погодьтеся з рядком підключення ***ХлібConnectionString***, клацнувши кнопку **Next**. Клацніть цю саму кнопку та у другому вікні майстра.

1.3. Клацніть кнопку **Query Builder** у третьому вікні майстра, додайте таблиці ***Продажі***, ***Товари*** та ***Виробники*** й побудуйте такий запит:

```
SELECT Продажі.Дата, Виробники.Виробник, Товари.Товар, Товари.Ціна,
    Продажі.Кількість, Товари.Ціна * Продажі.Кількість AS Вартість
FROM Виробники INNER JOIN
       Продажі ON Виробники.Код_виробника = Продажі.Код_виробника
       INNER JOIN Товари ON Продажі.Код_товару = Товари.Код_товару
```

У вікні **Query Builder** можна перевірити правильність побудови запиту, клацнувши кнопку **Execute Query** (рис. 7.28).

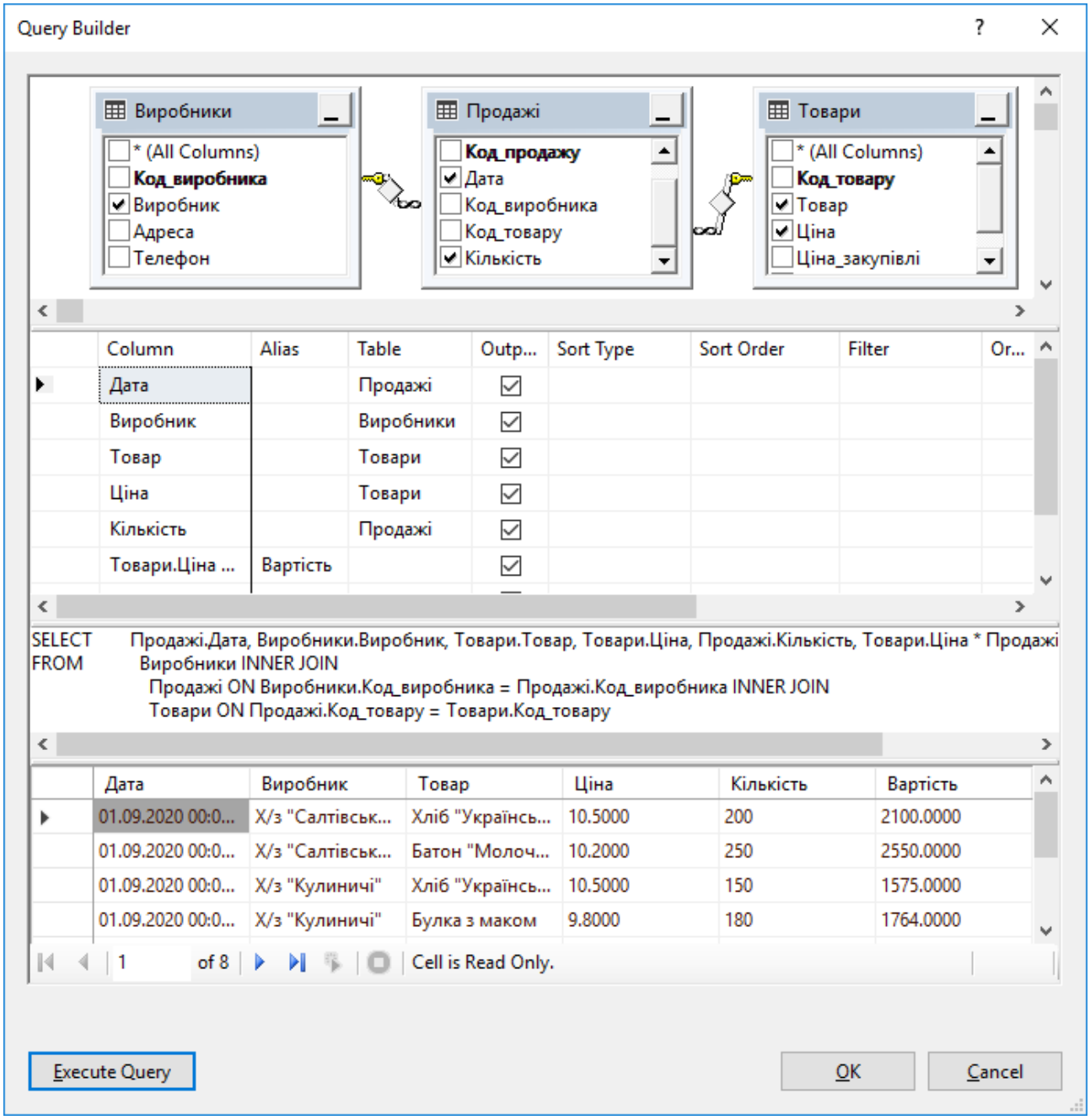


Рис. 7.28. **Вікно *Execute Query***

1.4. Повернувшись у вікно майстра, клацніть кнопку **Finish**.

1.5. Повернувшись у вікно набору даних, змініть ім'я таблиці ***DataTable1*** на ***ЩоденніПродажі*** та закрийте це вікно зі збереженням змін.

2. Створіть звіт, у якому містяться дані про щоденні продажі. Для цього:

2.1. Виконайте команду **Project – Add New Item**.

2.2. Виберіть елемент **Visual C# Items** у лівому списку вікна **Add New Item,** виділіть елемент **Report Wizard** у центральній частині вікна, уведіть ім'я ***rptЩоденніПродажі*** та клацніть кнопку **Add**. З'явилося перше вікно майстра звітів **Report Wizard**.

2.3. Установіть значення елементів у першому вікні **Report Wizard**, наведені в табл. 7.4.

Таблиця 7.4

Значення елементів

Елементи	Значення
Datasource	хлібDataSet
Available datasets	ЩоденніПродажі

2.4. Перетягніть поля ***Дата*** і ***Виробник*** зі списку **Available fields** у список **Row groups**, а поля ***Товар***, ***Кількість*** і ***Вартість*** – у список **Values** у другому вікні майстра (рис. 7.29).

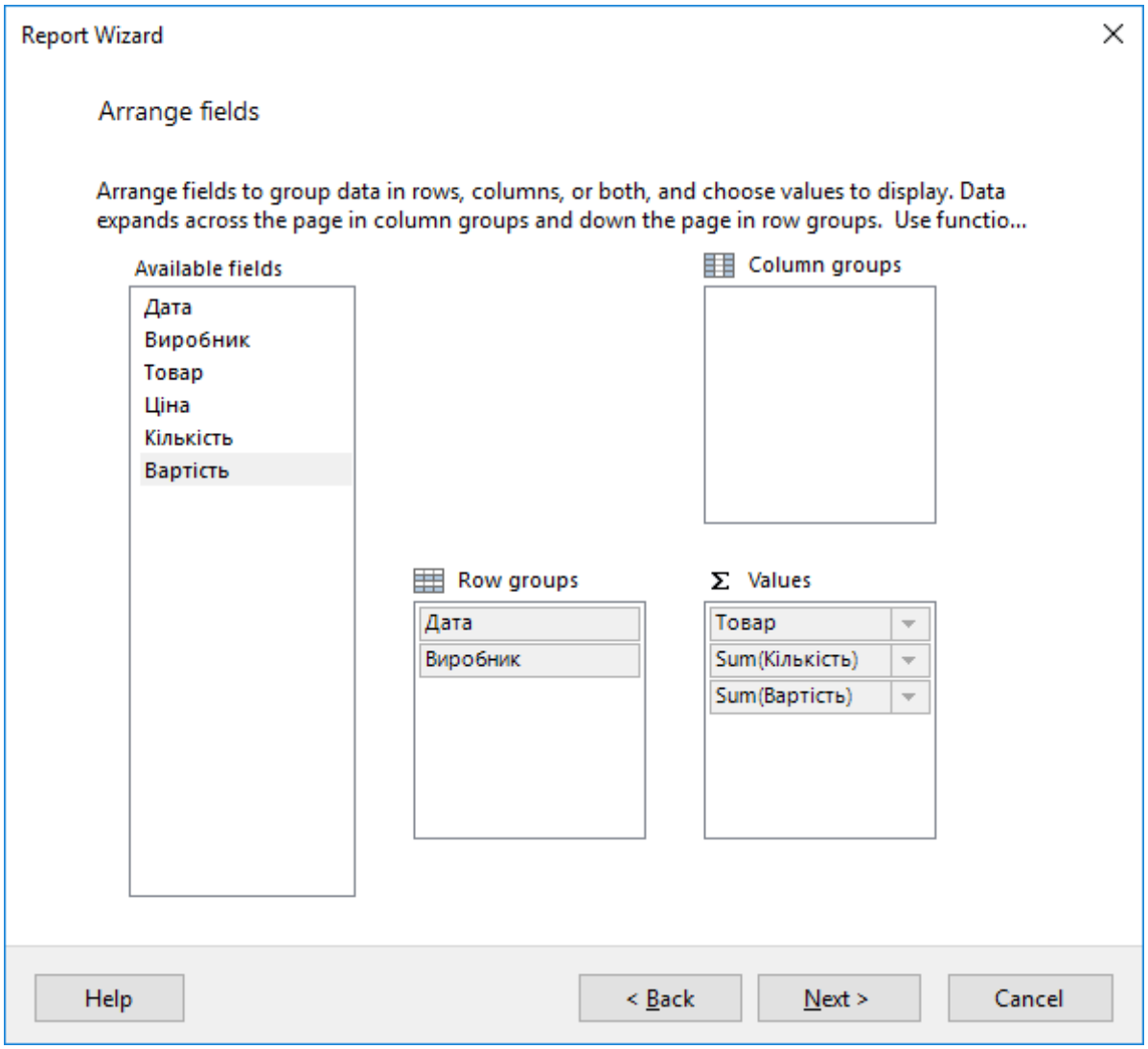


Рис. 7.29. **Друге вікно *Report Wizard***

2.5. Приберіть прапорець **Expand/collapse groups** і клацніть кнопку **Next** у третьому вікні майстра (рис. 7.30).

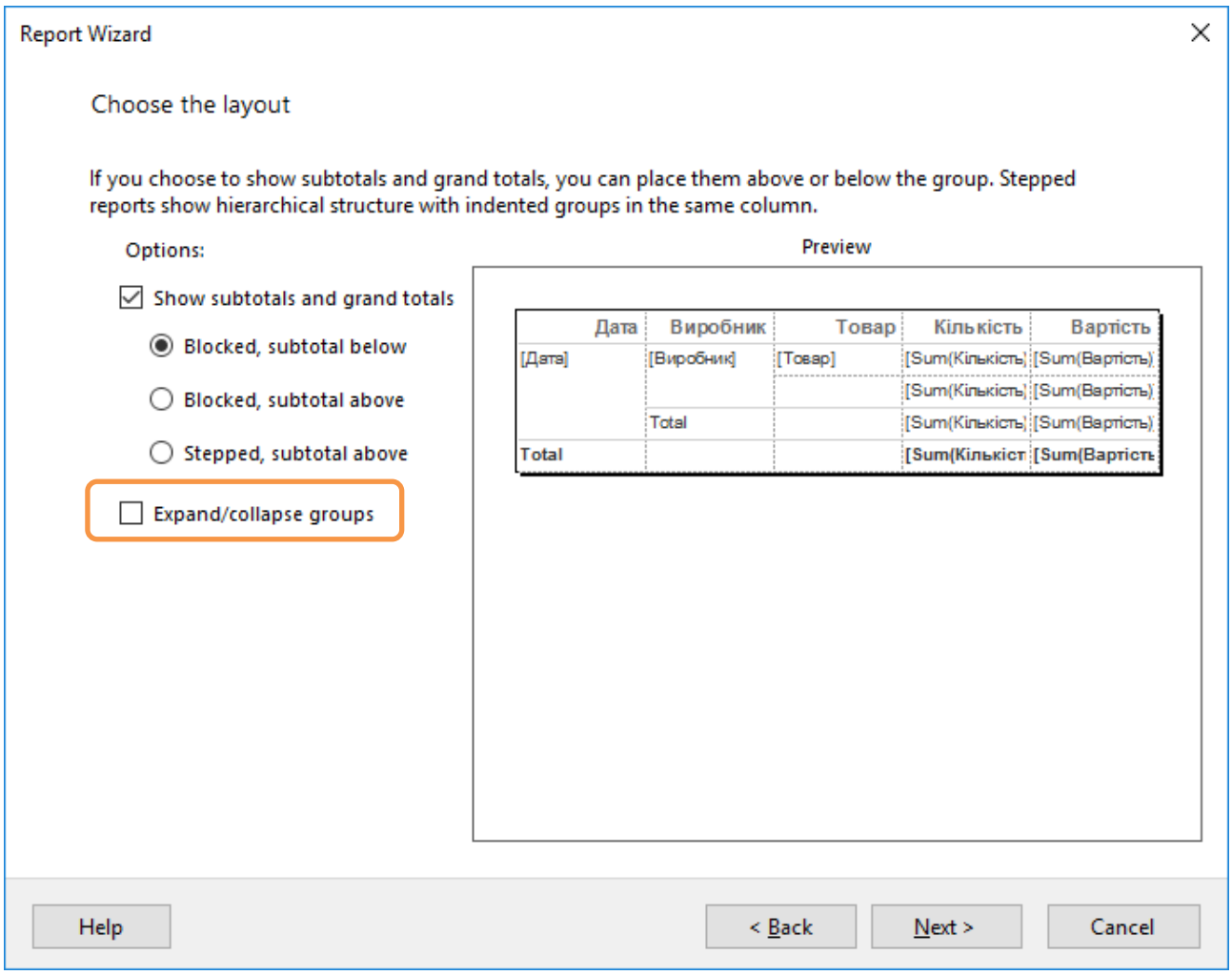


Рис. 7.30. **Прибраний прапорець *Expand/collapse groups***

2.6. Клацніть кнопку **Finish** у четвертому вікні майстра.

2.7. У вікні **Solution Explorer** з'явився значок звіту (файл з ім'ям ***rptЩоденніПродажі.rdlc***).

3. Відредагуйте звіт ***rptЩоденніПродажі*** (рис. 7.31).

Дата	Виробник	Товар	Кількість	Вартість
«Expr»	[Виробник]	[Товар]	[Sum(Кількість	[Sum(Вартість)
		Разом по виробнику:	[Sum(Кількість	[Sum(Вартість)
	«Expr»		[Sum(Кількість	[Sum(Вартість)
Усього:			**[Sum(Кількіст**	**[Sum(Вартість**

Рис. 7.31. **Таблікс звіту *rptЩоденніПродажі* у вікні конструктора**

3.1. Розширте стовпці ***Виробник*** і ***Товар***, щоб відповідні назви в них відображалися в одному рядку.

3.2. Розташуйте назви стовпців по центру, указавши значення **Center** для їхньої властивості **TextAlign**.

3.3. Відформатуйте виведення даних поля ***Дата***. Для цього виберіть із його контекстового меню команду **Expression** і введіть такий вираз:

```
=Format(Fields!Дата.Value, "dd.MM.yyyy")
```

3.4. Відформатуйте виведення даних поля ***Дата***. Для цього виберіть із його контекстового меню команду **Expression** і введіть такий вираз:

```
=Format(Fields!Дата.Value, "dd.MM.yyyy")
```

3.5. У підсумковому рядку, що перебуває у стовпці ***Товар*** (третій рядок таблікса), уведіть текст ***Разом по виробнику:***.

3.6. У підсумковому рядку, що перебуває у стовпці ***Виробник*** (четвертий рядок таблікса), замініть текст ***Total*** на вираз:

```
="Усього за " & Format(Fields!Дата.Value, "dd.MM.yyyy") & ":"
```

Для цього виберіть команду **Expression** із контекстового меню клітинки.

Примітка. Вирази в елементі ReportViewer будують за правилами мови Visual Basic. Зокрема, конкатенацію рядків позначають символом **&**, який оточують із кожного боку пробілом.

3.7. У підсумковому рядку, що перебуває у стовпці ***Дата*** (останній рядок таблікса), замініть текст ***Total*** на ***Усього:***.

4. Перетягніть таблікс нижче, щоб звільнити місце для заголовка і додайте логотип кіоску та заголовок ***Звіт про продажі кіоску "Колобок"*** у верхню частину звіту, скориставшись елементами **Image** і **Text Box**. Потім додайте ще одне текстове поле, розмістивши його під попереднім, і введіть у нього вираз:

```
= "за період з " & Format(Min(Fields!Дата.Value,"DataSet1"), "dd.MM.yyyy")
& " по " & Format(Max(Fields!Дата.Value, "DataSet1"), "dd.MM.yyyy")
```

Примітка. Тут і далі вираз, що починається символом **=** потрібно вводити за допомогою команди **Expression** із контекстового меню елемента **Text Box**.

5. Додайте текстове поле, розмістивши його під табліксом, і введіть у нього текст:

Гол. бухгалтер _____ П. П. Петренко

Потім додайте ще одне текстове поле, розмістивши його під попереднім, і введіть у нього вираз:

```
=Format(Today(),"dd.MM.yyyy")
```

6. Додайте у проєкт нову форму з ім'ям файлу ***formЩоденніПродажі.cs*** і значенням ***Щоденні продажі*** властивості **Text**.

7. Додайте на форму ***Щоденні продажі*** елемент **ReportViewer** і в його смарт-тегу **ReportViewer Tasks** виберіть значення ***rptХліб.rptЩоденніПродажі.rdlc*** у полі зі списком **Choose Report**, а також клацніть гіперпосилання **Dock in Parent Container**, щоб звіт зайняв усе місце на формі.

8. Перейдіть у вікно конструктора форми ***Звіти***, двічі клацніть кнопку ***Щоденні продажі*** та додайте код оброблювача події "Клацання на кнопці ***Щоденні продажі***".

```
private void buttonЩоденніПродажі_Click(object sender, EventArgs e)
{
  formЩоденніПродажі вікноЩоденніПродажі = new formЩоденніПродажі();
  вікноЩоденніПродажі.ShowDialog();
}
```

9. Перевірте функціональність форми ***Щоденні продажі***. Має з'явитися форма, подана на рис. 7.25.

2.3. Схема документа як засіб навігації документами

Завдання

Побудуйте звіт ***rptНакладні***, у якому містяться прибуткові накладні, за якими кіоск "Колобок" отримував товари. Кожну накладну розмістіть

на окремій сторінці, забезпечивши засобами навігації для швидкого переходу до потрібної накладної (рис. 7.32).

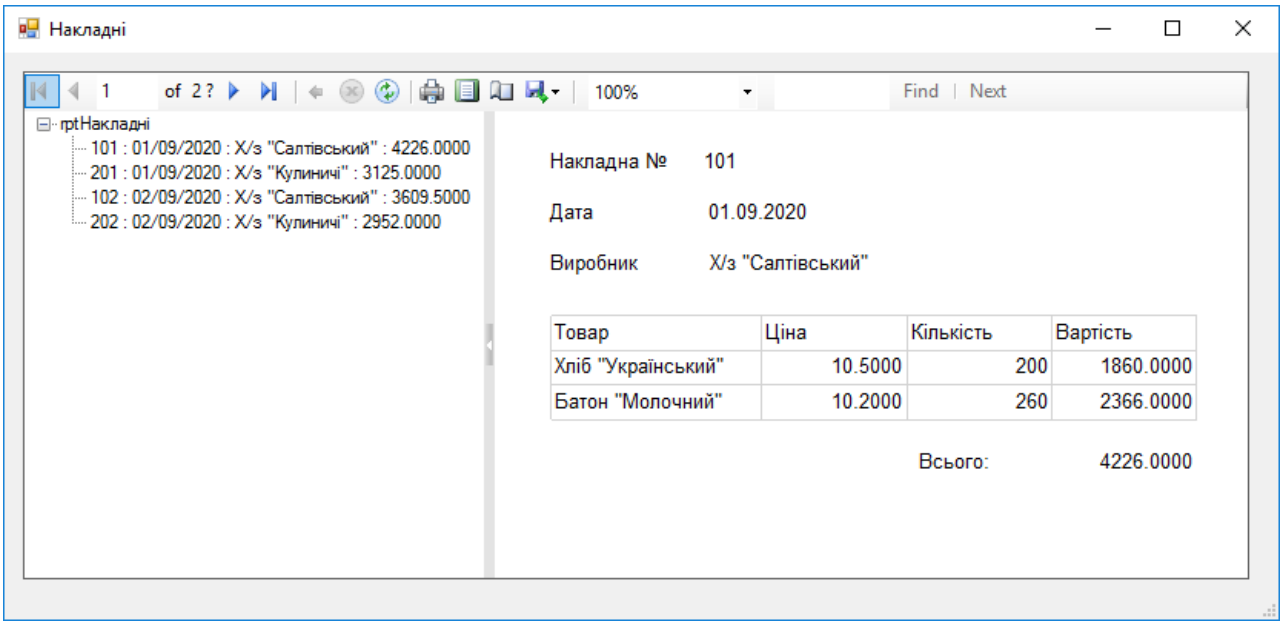


Рис. 7.32. **Звіт *rptНакладні***

Ідеї виконання

Документ "Накладна" складається із двох частин – спільної та детальної. У спільній частині відображають загальні дані накладної (номер накладної, дата її видачі та назва виробника, від якого її отримано), а в детальній частині – дані про товари, отримані за накладною. У базі даних ці дані містяться в основних таблицях ***Invoices*** та ***Invoice Products***, пов'язані відношенням "один-до-багатьох", а також у допоміжних таблицях ***Products*** та ***Manufacturers***. Тому до набору даних ***ХлібDataSet*** попередньо додайте таблицю ***НакладніЗагальна***, у якій зібрано потрібні дані.

Відношення "один-до-багатьох" реалізуйте у звіті так: дані спільної частини звіту подайте у вигляді списку, а детальної – у вигляді таблиці.

Для забезпечення навігації накладними відобразіть область звіту **Document map**, яку додають до звіту через властивості груп. Для цього кожну накладу оформіть як окрему групу записів.

У схемі документа подайте дані, за якими найлегше ідентифікувати потрібну накладну (номер накладної, дата її видачі, назва виробника та загальна вартість товарів за нею).

Оскільки звіт має комбіновану форму, слід будувати його вручну.

Зовнішній вигляд звіту ***rptНакладні*** в конструкторі подано на рис. 7.33.

Рис. 7.33. **Звіт *rptНакладні* в конструкторі**

Виконання

1. Відкрийте в конструкторі набір даних ***ХлібDataSet*** і додайте до нього таблицю ***НакладніЗагальна*** з полями ***Номер_накладної***, ***Дата***, ***Виробник***, ***Товар***, ***Ціна***, ***Кількість*** і ***Вартість***. Для цього в елементі **TableAdapter** побудуйте запит:

```
SELECT Накладні.Номер_накладної, Накладні.Дата,
    Виробники.Виробник, Товари.Товар, Товари.Ціна,
    ТовариНакладних.Кількість,
    Товари.Ціна_закупівлі * ТовариНакладних.Кількість AS Вартість,
    Накладні.Код_накладної
FROM ТовариНакладних INNER JOIN Накладні ON
    ТовариНакладних.Код_накладної = Накладні.Код_накладної
    INNER JOIN Виробники ON Накладні.Код_виробника =
    Виробники.Код_виробника
    INNER JOIN Товари ON ТовариНакладних.Код_товару =
    Товари.Код_товару
```

Примітка. Поле ***Invoices.Код_накладної*** потрібне буде для групування даних звіту за накладними.

Приберіть ключ із поля ***Код_накладної*** за допомогою контекстового меню.

2. Додайте до проєкту звіт ***rptНакладні***, виконавши команду **Project – Add New Item**, а у вікні **Add New Item**, вибравши тип **Report**.

3. Перетягніть елемент **List** із панелі елементів в область конструктора звіту й у вікні **Dataset Properties** установіть значення елементів, наведені в табл. 7.5:

Таблиця 7.5

Значення елементів

Елементи	Значення
Datasource	хлібDataSet
Available datasets	НакладніЗагальна

4. Відкрийте вікно **Данные отчета** (можна скористатися комбінацією клавіш **Ctrl+Alt+D**) і перетягніть поля ***Номер_накладної***, ***Дата*** і ***Виробник***. Перед ними поставте відповідні написи за допомогою елементів **Text Box**. У полі ***Дата*** замініть значення на вираз:

```
=Format(Fields!Дата.Value,"dd/MM/yyyy")
```

5. Додайте до звіту елемент **Table**, розмістивши його під текстовими полями в рамці списку. Потім за допомогою контекстового меню будь-якої клітинки таблиці вставте ще один стовпець.

6. Клацаючи клітинки другого рядка таблиці, уставте в них такі поля: ***Товар***, ***Ціна***, ***Кількість*** і ***Вартість***. У верхньому рядку з'являться їхні назви.

7. Додайте до звіту два текстових поля, розмістивши їх під стовпцями ***Кількість*** і ***Вартість*** у рамці списку. У перше з них уведіть текст У***сього:***, а у друге – вираз:

```
=Sum(Fields!Вартість.Value)
```

8. Згрупуйте рядки звіту для кожної накладної та встановіть область **Document map**. Для цього виберіть команду **Group Properties** із контекстового меню підгрупи **Details**, розташованої під вікном конструктора звіту. З'явилося однойменне вікно. У ньому виконайте такі дії:

8.1. У категорії **General** клацніть кнопку **Add** і в полі зі списком **Group on** установіть значення ***Код_накладної***.

8.2. У категорії **Page Breaks** установіть прапорець ***Between each instance of a group***.

8.3. У категорії **Advanced** клацніть кнопку виразу для схеми документа (**Document map**) і введіть такий вираз:

```
=Fields!Номер_накладної.Value & " : " & Fields!Дата.Value &
" : " & Fields!Виробник.Value & " : " & Sum(Fields!Вартість.Value)
```

Його значення будуть відображатися в області **Document map**.

9. Перейдіть у вікно конструктора форми ***Накладні***, додайте елемент **ReportViewer** і у його смарт-тегу **ReportViewer Tasks** виберіть значення ***rptХліб.rptНакладні.rdlc*** у полі зі списком **Choose Report**. Потім встановіть значення ***300*** для властивості **DocumentMapWidth** елемента **ReportViewer**, щоб повністю відображалися дані в області **Document map**.

10. Додайте у проєкт нову форму з ім'ям файлу ***formНакладні.cs*** та значенням ***Накладні*** властивості **Text**.

11. Додайте на форму ***Накладні*** елемент **ReportViewer** і у його смарт-тегу **ReportViewer Tasks** виберіть значення ***rptХліб.rptНакладні.rdlc*** у полі зі списком **Choose Report**.

12. Перейдіть у вікно конструктора форми ***Звіти***, двічі клацніть кнопку ***Накладні*** та додайте код оброблювача події "Клацання на кнопці ***Накладні***".

```
private void buttonНакладні_Click(object sender, EventArgs e)
{
    formНакладні вікноНакладні = new formНакладні();
    вікноНакладні.ShowDialog();
}
```

13. Перевірте функціональність форми ***Накладні***. Має з'явитися форма, подана на рис. 7.31.

У звіті з лабораторної роботи подайте скриншоти форм, у яких відображають звіти ***rptПрайс***, ***rptЩоденніПродажі*** та ***rptНакладні*** в режимі виконання.

3. Фільтрація

3.1. Фільтрація на сервері

Завдання

Побудуйте список, у якому містяться дані про ціни на товари вибраної групи (рис. 7.34). Відбір даних зазначеної групи виконайте в базі даних на сервері.

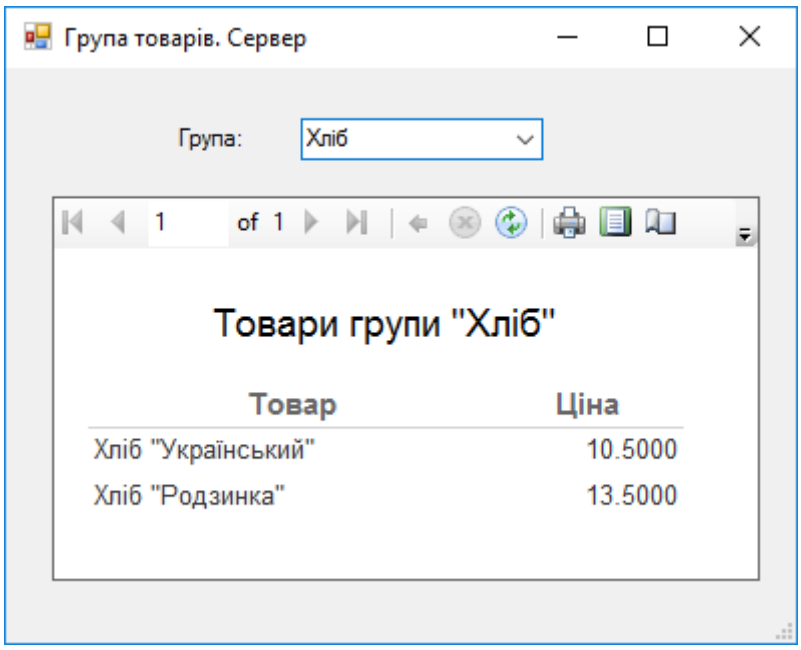


Рис. 7.34. **Ціни на товари вибраної групи (фільтрація на сервері)**

Ідеї виконання

Для зазначення потрібної групи товарів використайте поле зі списком ***Група***. У ньому відображають назви груп товарів, а значенням є код вибраної групи.

Як джерело даних для списку додайте в набір даних таблицю ***ГрупаТоварів***. Її побудуйте на основі запиту, що має параметр ***Код_групи*** в реченні Where. Завдяки цьому з бази даних завантажуються тільки відомості про товари, що мають указаний код групи (тобто відбір даних виконують на сервері). Як значення параметра запиту вкажіть значення поля зі списком ***Група***.

Щоб дані у звіті змінювалися після того, як у полі зі списком зміниться значення, додайте до форми оброблювача відповідної події.

Необхідність у додаванні до набору даних нової таблиці ***Група_товарів*** зумовлено тим, що у звіті відображають дані двох таблиць: ***Групи*** (у заголовку) і ***Товари*** (у таблиці). Якби використовували дані тільки однієї таблиці, то запит можна було б додати до вже наявної таблиці як метод ***FillBy***.

Оскільки звіт має форму таблиці, будуть будувати його за допомогою майстра.

Зовнішній вигляд звіту ***rptГрупаТоварів_Сервер*** у конструкторі подано на рис. 7.35.

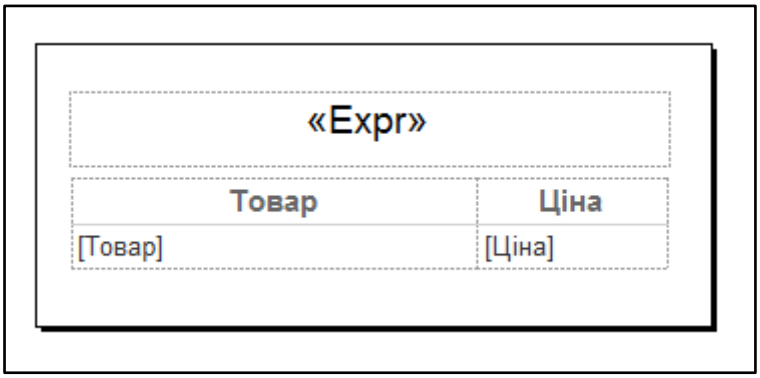


Рис. 7.35. **Звіт *rptГрупаТоварів_Сервер* у конструкторі**

Виконання

1. Додайте у проєкт нову форму з ім'ям файлу ***formГрупаТоварів_Сервер.cs*** і значенням ***Група товарів. Сервер*** властивості **Text**.

2. Помістіть на формі поле зі списком ***Група***. Для цього:

2.1. Перетягніть елемент ComboBox із панелі елементів на форму, давши йому ім'я ***comboBoxГрупа***. Перед ним додайте напис ***Група***.

2.2. Відкрийте вікно **Data Sources** (можна скористатися комбінацією клавіш **Shift+Alt+D**) і перетягніть вузол ***Групи*** на поле зі списком ***Група***.

3. Відкрийте в конструкторі набір даних ***ХлібDataSet*** і додайте до нього таблицю ***ГрупаТоварів*** із полями ***Група***, ***Товар*** і ***Ціна***. Для цього в елементі **TableAdapter** побудуйте запит:

```
SELECT Групи.Група, Товари.Товар, Товари.Ціна
FROM   Групи INNER JOIN
       Товари ON Групи.Код_групи = Товари.Код_групи
WHERE  (Товари.Код_групи = @Код_групи)
```

4. Додайте до проєкту звіт ***rptГрупаТоварів_Сервер*** за допомогою майстра. Для цього:

4.1. Виконайте команду **Project – Add New Item**.

4.2. Виберіть елемент **Visual C# Items** у лівому списку вікна **Add New Item,** виділіть елемент **Report Wizard** у центральній частині вікна, уведіть ім'я ***rptГрупаТоварів_Сервер*** і клацніть кнопку **Add**. З'явилося перше вікно майстра звітів **Report Wizard**.

4.3. Установіть значення елементів у першому вікні **Report Wizard**, наведені в табл. 7.6.

Таблиця 7.6

Значення елементів

Елементи	Значення
Datasource	хлібDataSet
Available datasets	ГрупаТоварів

4.4. Перетягніть поля ***Товар*** і ***Ціна*** зі списку **Available fields** у список **Values** у другому вікні майстра.

4.5. Клацніть кнопку **Next** у третьому вікні майстра, а в четвертому – кнопку **Finish**. У вікні **Solution Explorer** з'явився значок звіту (файл з ім'ям ***rptГрупаТоварів_Сервер.rdlc***).

4.6. Змініть значення ***[Sum(Ціна)]*** на ***[Ціна]*** у вікні конструктора звіту.

5. Перетягніть таблікс нижче, щоб звільнити місце для заголовка, і додайте для нього елемент **Text Box**, у який уведіть вираз:

```
="Товари групи """ & First(Fields!Група.Value, "DataSet1") & """"
```

Розташуйте заголовок по центру над табліксом і встановіть розмір шрифту 14 пт та напівжирного накреслення.

6. Додайте на форму ***Група товарів. Сервер*** елемент **ReportViewer** і у його смарт-тегу **ReportViewer Tasks** виберіть значення ***rptХліб.rptЩоденніПродажі.rdlc*** у полі зі списком **Choose Report**.

7. Перейдіть у вікно коду форми ***Група товарів. Сервер*** і змініть код оброблювача події завантаження форми, щоб він набув такого вигляду:

```
private void formГрупа_товарів_Сервер_Load(object sender,
    EventArgs e)
{
    // Завантажуємо дані для comboBoxГрупа
    this.групиTableAdapter.Fill(this.хліbDataSet.Групи);
    // Дізнаємося код вибраної групи товарів
    int k = (int)comboBoxГрупа.SelectedValue;

    // Завантажуємо із сервера дані про товари вибраної групи
    this.ГрупаТоварівTableAdapter.Fill(
    this.хліbDataSet.ГрупаТоварів, k);

    this.reportViewer1.RefreshReport();
}
```

8. Перейдіть у вікно конструктора форми ***Група товарів. Сервер*** і двічі клацніть поле зі списком ***Група***. Потім додайте код оброблювача події "Зміна вибраного елемента".

```
private void comboBoxГрупа_SelectedIndexChanged(object sender, Even-
tArgs e)
{
    // Дізнаємося код вибраної групи товарів
    int k = (int)comboBoxГрупа.SelectedValue;
    // Завантажуємо із сервера дані про товари вибраної групи
    this.ГрупаТоварівTableAdapter.Fill(
        this.хліб DataSet.ГрупаТоварів, k);

     this.reportViewer1.RefreshReport();
}
```

9. Перейдіть у вікно конструктора форми ***Звіти***, двічі клацніть кнопку ***Група товарів. Сервер*** і додайте код оброблювача події "Клацання на кнопці ***Група товарів. Сервер***".

```
private void buttonrptГрупа_товарів_Сервер_Click(object sender, Even-
tArgs e)
{
    formГрупаТоварів_Сервер вікноГрупаТоварів_Сервер =
        new formГрупаТоварів_Сервер();
    вікноГрупаТоварів_Сервер.ShowDialog();
}
```

10. Перевірте функціональність форми ***Група товарів. Сервер***. Має з'явитися форма, подана на рис. 7.34.

3.2. Фільтрація в DataSet

Завдання

Побудуйте список, у якому містяться дані про ціни на товари вибраної групи (рис. 7.36). Відбір даних зазначеної групи виконайте на локальному комп'ютері в типізованому наборі даних.

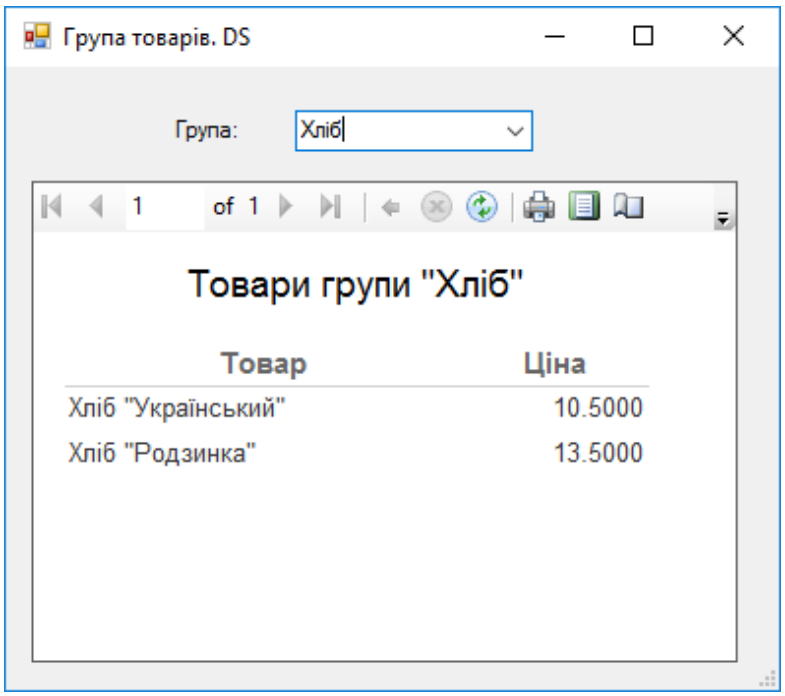


Рис. 7.36. **Ціни на товари вибраної групи (фільтрація в наборі даних)**

Ідеї виконання

Форма буде мати такий самий вигляд, що й у попередньому завданні, але джерело даних формують інакше. У набір даних відразу завантажте всі дані із сервера в локальну таблицю ***ГрупаТоварів***, а потім на локальному комп'ютері виконайте фільтрацію, відповідно до вибраного значення в полі зі списком ***Група***. Такий підхід скорочує кількість звернень до сервера під час перегляду даних про товари по багатьох групах.

Як джерело даних використайте локальну таблицю ***ГрупаТоварів***, до якої додайте запит. У ньому відсутнє речення Where (тобто відразу завантажте всі дані), але додано поле ***Код_групи*** з таблиці ***Products***. Його використовують у фільтрації даних. Цей запит будуть викликати через метод ***FillByУсі***.

Оскільки звіт має ту саму структуру, що й у попередньому завданні, використайте його. Дані, що відображають у звіті, відфільтруйте в коді форми.

Виконання

1. Додайте у проєкт нову форму з ім'ям файлу ***formГрупаТоварів_DS.cs*** і значенням ***Група товарів. DS*** властивості **Text**.

2. Помістіть на формі поле зі списком ***Група*** так само, як у попередньому завданні.

3. Відкрийте в конструкторі набір даних ***ХлібDataSet*** і додайте запит до таблиці ***ГрупаТоварів_Сервер***. Для цього:

3.1. Виберіть команду **Add – Query** з контекстового меню таблиці.

3.2. Клацніть кнопку **Next** у першому й другому вікнах майстра налаштування адаптера таблиці, а у третьому – спочатку вилучіть речення **Where** із запиту і додайте поле ***Код_групи*** з таблиці ***Групи***, а потім клацніть кнопку **Next**.

Запит буде мати такий вигляд:

```
SELECT Групи.Група, Товари.Товар, Товари.Ціна,
       Групи.Код_групи
FROM   Групи INNER JOIN
       Товари ON Групи.Код_групи = Товар.Код_групи
```

3.3. Дайте імена методам ***FillByУсі*** та ***GetDataByУсі*** в четвертому вікні майстра і клацніть кнопку **Finish**.

4. Перейдіть у вікно конструктора форми ***Група товарів. DS***, додайте на форму елемент **ReportViewer** і у смарт-тегу **ReportViewer Tasks** елемента **ReportViewer** виберіть значення ***rptХліб.rptГрупаТоварів_Сервер.rdlc*** у полі зі списком **Choose Report**.

5. Перейдіть у вікно коду форми ***Група товарів. DS*** і змініть код оброблювача події завантаження форми, щоб він набув такого вигляду:

```
private void formГрупаТоварів_DS_Load(object sender, EventArgs e)
{
    // Завантажуємо дані для comboBoxГрупа
    this.групиTableAdapter.Fill(this.хліbDataSet.Групи);

    // Завантажуємо  всі дані для звіту в локальну таблицю
    this.ГрупаТоварівTableAdapter.FillByУсі(
        this.хліbDataSet.ГрупаТоварів);

    // Дізнаємося код вибраної групи товарів
    int k = (int)comboBoxГрупа.SelectedValue;

    // Відбираємо в DataSet потрібні дані
    ГрупаТоварівBindingSource.Filter =
        "Код_групи = " + k.ToString();
    this.reportViewer1.RefreshReport();
}
```

6. Перейдіть у вікно конструктора форми ***Група товарів. DS*** і двічі клацніть поле зі списком ***Група***. Потім додайте код оброблювача події "Змінено вибраний елемент".

```
private void comboBoxГрупа_SelectedIndexChanged(object sender, Even-
tArgs e)
{
    // Дізнаємося код вибраної групи товарів
    int k = (int)comboBoxГрупа.SelectedValue;

    // Відбираємо в DataSet потрібні дані
    ГрупаТоварівBindingSource.Filter =
        "Код_групи = " + k.ToString();

    this.reportViewer1.RefreshReport();
}
```

7. Перейдіть у вікно конструктора форми ***Звіти***, двічі клацніть кнопку ***Група товарів. DS*** і додайте код оброблювача події "Клацання на кнопці ***Група товарів. DS***".

```
private void buttonrptГрупа_товарів_DS_Click(object sender,
    EventArgs e)
{
    formГрупаТоварів_DS вікноГрупаТоварів_DS =
        new formГрупаТоварів_DS();
    вікноГрупаТоварів_DS.ShowDialog();
}
```

8. Перевірте функціональність форми ***Група товарів. DS***. Має з'явитися форма, подана на рис. 7.36.

3.3. Фільтрація з можливістю відмови від неї. Параметр звіту

Завдання

Побудуйте список, у якому містяться дані про ціни на товари вибраної групи, а також є можливість переглянути дані про всі товари (рис. 7.37). Відбір даних зазначеної групи виконайте в типізованому наборі даних.

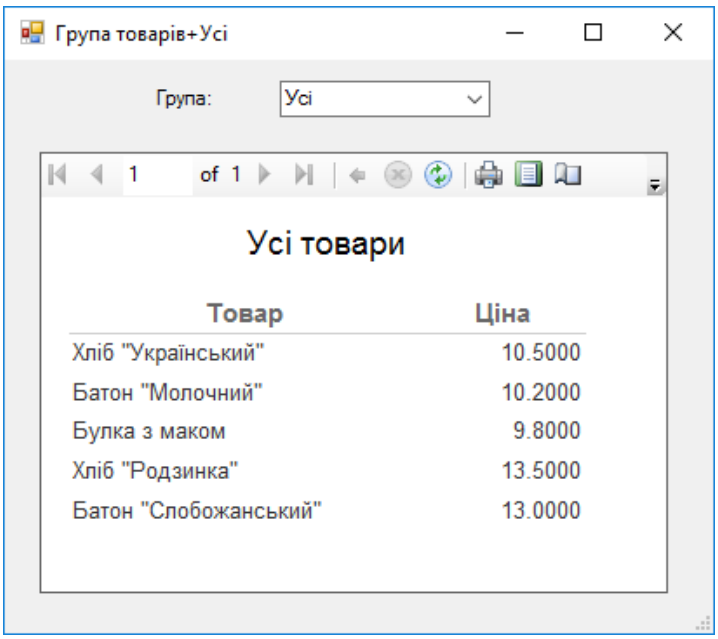


Рис. 7.37. **Ціни на всі товари з можливістю вибору групи**

Ідеї виконання

Форма буде мати такий самий вигляд як і у двох попередніх завданнях, але в поле зі списком додано ще один елемент – ***Усі***. Залежно від вибору елемента в полі зі списком у звіті відображають або дані про всі товари, або тільки вибраної групи.

Для реалізації такої можливості як джерело даних використайте таблицю ***ГрупаТоварів*** у наборі даних і проаналізуйте вибраний елемент. Якщо вибрано елемент ***Усі***, заповніть таблицю за допомогою методу ***FillByУсі***, а якщо якийсь інший – за допомогою методу ***Fill***.

Від вибраного в полі зі списком елемента залежить також заголовок. У ньому відображено або текст ***Усі товари***, або текст ***Товари групи "xxx"***, де ***xxx*** – назва групи. Текст заголовка буде передано у звіт як параметр.

Щоб у полі зі списком з'явився елемент ***Усі***, додайте рядок у локальну таблицю ***Групи*** під час завантаження форми. У цьому рядку поля ***Група*** надайте значення ***Усі***, а поле ***Код_групи*** набуде за замовчуванням значення ***-1***. Після сортування таблиці ***Групи*** за зростанням значень поля ***Код_групи*** елемент ***Усі*** буде відображати на першому місці в полі зі списком.

Звіт має ту саму структуру, що й у двох попередніх завданнях, але заголовок формують по-різному, залежно від того, яке значення вибрано в полі зі списком ***Група***. Тому скопіюйте звіт ***rptХліб.rptГрупаТоварів_Сервер*** і модифікуйте його копію – для відображення заголовка використайте параметр звіту.

Виконання

1. Додайте у проєкт нову форму з ім'ям файлу ***formГрупаТоварів_Усі.cs*** і значенням ***Група товарів + Усі*** властивості **Text**.

2. Помістіть на формі поле зі списком ***Група*** так само, як у попередньому завданні.

3. Створіть попередній варіант звіту ***rptГрупа_товарів_Усі*** шляхом модифікації звіту ***rptГрупаТоварів_Сервер***. Для цього:

3.1. Виділіть значок звіту ***rptГрупаТоварів_Сервер*** і скопіюйте його в буфер за допомогою комбінації клавіш **Ctrl+C**. Потім виділіть значок проєкту і вставте з буфера звіт за допомогою комбінації клавіш **Ctrl+V**.

3.2. Перейменуйте новий звіт на ***rptГрупаТоварів_Усі***.

4. Перейдіть у вікно конструктора форми ***Група товарів + Усі***, додайте на форму елемент **ReportViewer** і у смарт-тегу **ReportViewer Задачи** елемента **ReportViewer** виберіть значення ***rptХліб.rptГрупаТоварів_Усі.rdlc*** у полі зі списком **Choose Report**.

5. Перейдіть у вікно коду форми ***Група товарів + Усі*** і змініть код оброблювача події завантаження форми, щоб він набув такого вигляду:

```
private void formГрупаТоварів_Усі_Load(object sender, EventArgs e)
{
    // Завантажуємо дані для comboBoxГрупа
    this.групиTableAdapter.Fill(this.хліб1DataSet.Групи);

    // Додаємо у список груп елемент Усі
    this.хліб1DataSet.Групи.AddГрупиRow("Усі");

    // Виконуємо сортування списку, щоб елемент Усі став першим
    групиBindingSource.DataSource =
        this.хліб1DataSet.Групи.OrderBy(g => g.Код_групи);

    // Завантажуємо дані про товари всіх груп
    // для звіту в локальну таблицю
    this.ГрупаТоварівTableAdapter.FillByУсі(
        this.хліб1DataSet.ГрупаТоварів);

    this.reportViewer1.RefreshReport();
}
```

6. Перейдіть у вікно конструктора форми ***Звіти***, двічі клацніть кнопку ***Група товарів + Усі*** і додайте код оброблювача події "Клацання на кнопці ***Група товарів + Усі***".

```
private void buttonГрупаТоварів_Усі_Click(object sender,
    EventArgs e)
{
    formГрупаТоварів_Усі вікноГрупаТоварів_Усі =
        new formГрупаТоварів_Усі();
    вікноГрупаТоварів_Усі.ShowDialog();
}
```

7. Перевірте функціональність форми ***Група товарів + Усі***. Має з'явитися форма, подана на рис. 7.37.

8. Щоб, залежно від вибору елемента у списку (елемент ***Усі*** чи певна група товарів), відповідний заголовок відображався у звіті, додайте параметр до звіту. Для цього виконайте таке:

8.1. Відкрийте звіт ***rptГрупаТоварів_Усі*** та перейдіть у вікно **Report Data**. Із контекстового меню папки **Parameters** виберіть команду **Add Parameter**.

8.2. Уведіть значення ***Заголовок*** у поле **Name**, що перебуває в категорії **General** (рис. 7.38).

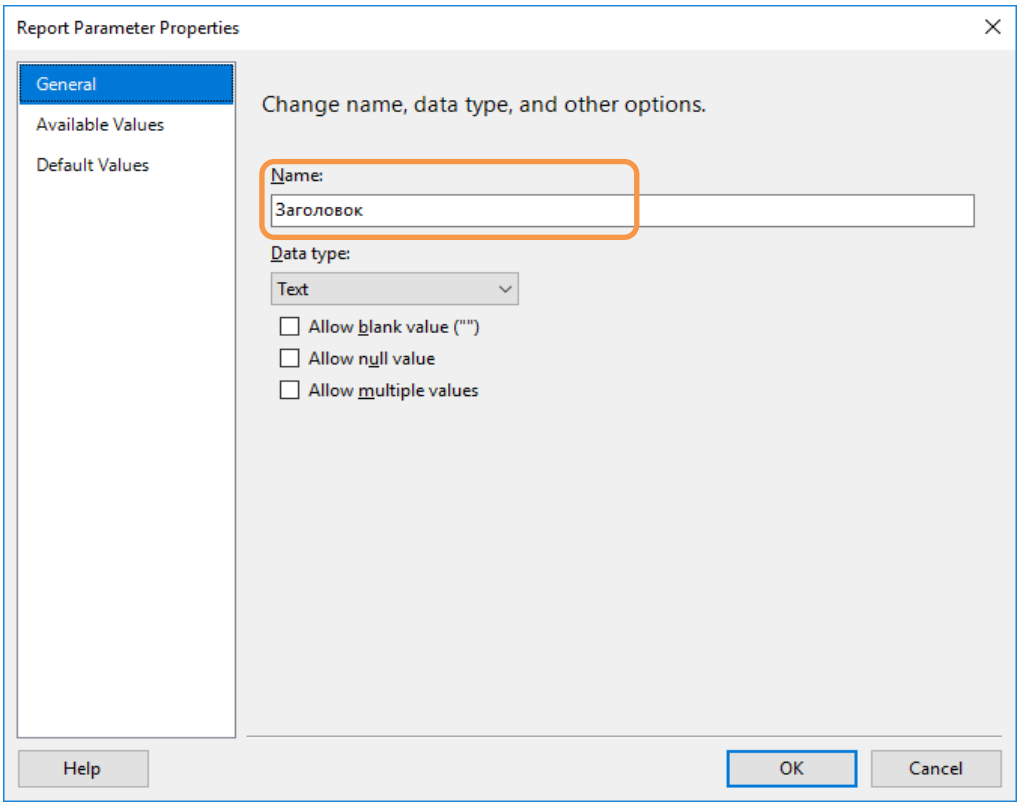


Рис. 7.38. **Задавання імені параметра**

8.3. Перейдіть у категорію **Default Values**, виберіть перемикач **Specify values**, клацніть кнопку **Add** і введіть значення ***Усі товари*** (рис. 7.39). Потім клацніть кнопку **OK**.

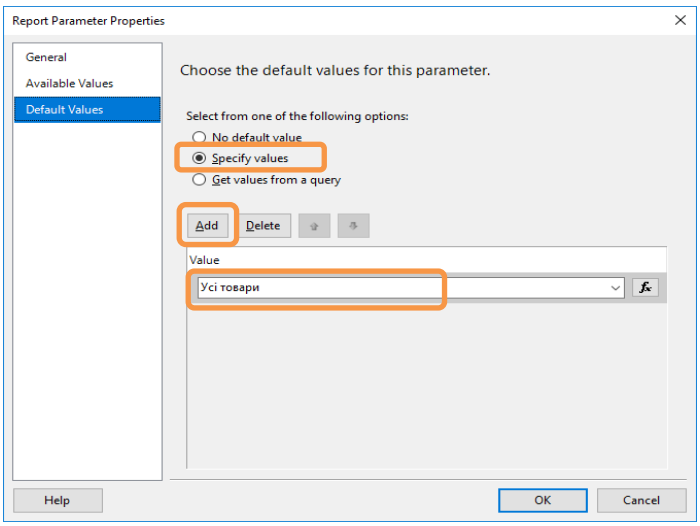


Рис. 7.39. **Задавання значення за замовчуванням**

8.4. Перейдіть у вікно звіту ***rptГрупаТоварів_Усі*** та з контекстового меню текстового поля заголовка виберіть команду **Expression**. Потім виберіть категорію **Parameters**, а в ній значення ***Заголовок*** і замініть ним попередній вираз, двічі клацнувши на значенні ***Заголовок*** (рис. 7.40).

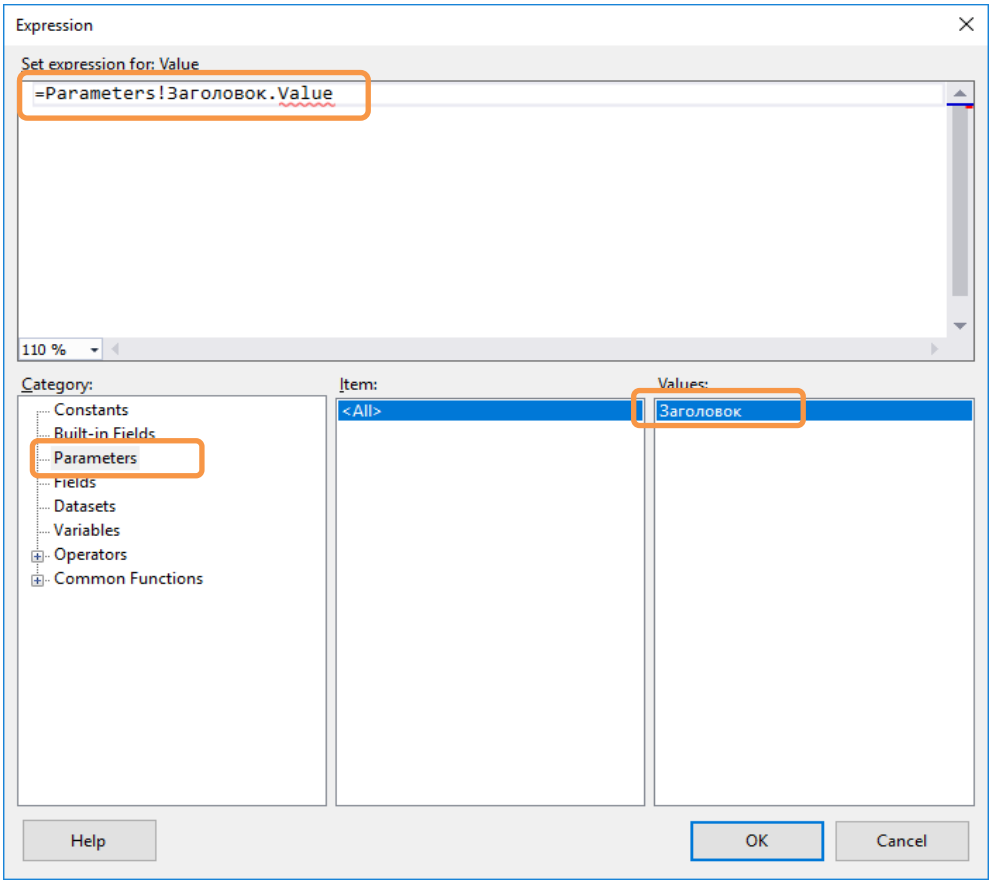


Рис. 7.40. **Задавання параметра як виразу для текстового поля заголовка**

Зовнішній вигляд звіту ***rptГрупаТоварів_Усі*** в конструкторі подано на рис. 7.41.

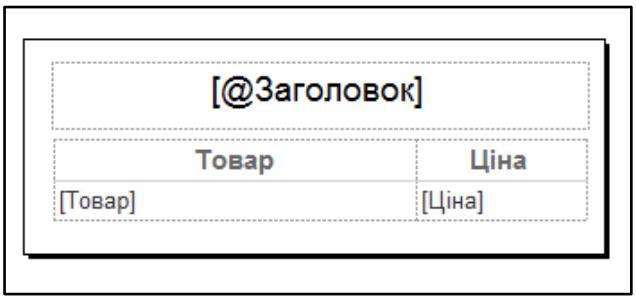


Рис. 7.41. **Звіт *rptГрупаТоварів_Усі* в конструкторі**

9. Перейдіть у вікно конструктора форми ***Група товарів + Усі*** та двічі клацніть поле зі списком ***Група***. Потім додайте код оброблювача події "Змінено вибраний елемент":

```
private void comboBoxГрупа_SelectedIndexChanged(object sender, Even-
tArgs e)
{
    // Дізнаємося код вибраного елемента у списку груп
    int k = (int)comboBoxГрупа.SelectedValue;

    // Описуємо параметр звіту
    ReportParameter paramЗаголовок;

    // Залежно від вибраного елемента списку
    // формуємо значення параметра звіту
    if (k < 0)
    {
        this.ГрупаТоварівTableAdapter.FillByУсі(
            this.хліб1DataSet.ГрупаТоварів);
        paramЗаголовок = new
            ReportParameter("Заголовок", "Усі товари");
    }
    else
    {
        this.ГрупаТоварівTableAdapter.Fill(
        this.хліб1DataSet.ГрупаТоварів, k);
        paramЗаголовок = new ReportParameter("Заголовок",
            "Товари групи: " + comboBoxГрупа.Text);
    };

    // Передаємо значення параметра у звіт
    this.reportViewer1.LocalReport.SetParameters(
        new ReportParameter[] { paramЗаголовок });

    this.reportViewer1.RefreshReport();
}
```

Примітка. Щоб забезпечити роботу з параметрами звіту, додайте до описів просторів імен ще й такий:

```
// Для використання параметрів
using Microsoft.Reporting.WinForms;
```

10. Збережіть зміни, зроблені у проєкті.

11. Перевірте функціональність форми ***Група товарів + Усі***. Спочатку має з'явитися форма, подана на рис. 7.37. Виконайте перевірку для різних значень поля зі списком ***Група***.

У звіті з лабораторної роботи подайте коди опису функцій, що забезпечують фільтрацію на сервері та в DataSet, а також із можливістю відмови від неї. Поясніть різницю між ними.

4. Аналіз

4.1. Зведена таблиця

Завдання

Побудуйте зведену таблицю для аналізу результатів продажів товарів кожного виробника (кількість і вартість) за кожний день із можливістю підбиття підсумків за групами товарів (рис. 7.42).

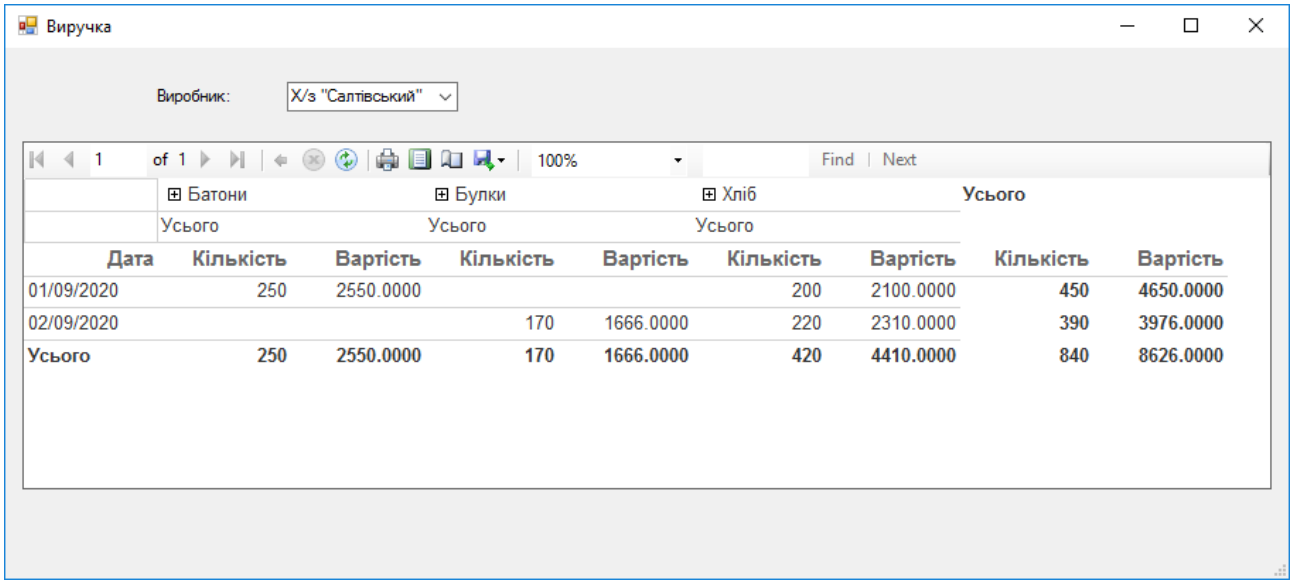


Рис. 7.42. **Зведена таблиця на формі *Виручка***

Ідеї виконання

Для відбору даних про продажі товарів потрібного виробника використайте поле зі списком ***Виробник***. У ньому відображають назви виробників, а значенням є код вибраного виробника.

У назвах рядків таблиці помістіть дати, у які здійснювали продажі товарів, у назвах стовпців – назви груп (верхній рівень) і товарів (нижній рівень), а на їхньому перетині – відповідні результати продажів (сумарні значення кількості та вартості). Тому для побудови зведеної таблиці знадобляться такі дані: ***Дата***, ***Кількість*** і ***Код_виробника*** з таблиці ***Продажі*** (перші два поля для відображення, а останнє – для фільтрації), ***Товар*** – із таблиці ***Товари***, ***Група*** – із таблиці ***Групи***. Значення поля ***Вартість*** обчислюють на основі даних полів ***Кількість*** із таблиці ***Продажі*** та ***Ціна*** з таблиці ***Товари***.

Як джерело даних для зведеної таблиці додайте в набір даних таблицю ***Виручка***, у якій зібрано потрібні дані. Її побудуйте на основі запиту. Відразу завантажте всі дані, а фільтрацію виконайте в локальній таблиці ***Виручка***. Це позбавить користувача від очікувань під час виконання бізнес-аналізу під час переходу до перегляду результатів продажів товарів іншого виробника.

Оскільки звіт має форму матриці, побудуйте його за допомогою майстра.

Зовнішній вигляд звіту ***rptВиручка*** в конструкторі подано на рис. 7.43.

	[Група]				Усього	
	[Товар]		Усього			
Дата	Кількість	Вартість	Кількість	Вартість	Кількість	Вартість
«Expr»	[Sum(Кількіс	[Sum(Вартість)	[Sum(Кількість	[Sum(Вартість)	[Sum(Кількіст	[Sum(Вартість
Усього	[Sum(Кількіст	[Sum(Вартість	[Sum(Кількіст	[Sum(Вартість	[Sum(Кількіст	[Sum(Вартість

Рис. 7.43. **Звіт *rptВиручка* в конструкторі**

Виконання

1. Додайте у проєкт нову форму з ім'ям файлу ***formВиручка.cs*** і значенням ***Виручка*** властивості **Text**.
2. Помістіть на формі поле зі списком ***Виробник***. Для цього:

2.1. Перетягніть елемент ComboBox із панелі елементів на форму, давши йому ім'я ***comboBoxВиробник***. Перед ним додайте напис ***Виробник***.

2.2. Відкрийте вікно **Data Sources** (можна скористатися комбінацією клавіш **Shift+Alt+D**) і перетягніть вузол ***Виробники*** на поле зі списком ***Виробник***.

3. Відкрийте в конструкторі набір даних ***ХлібDataSet*** і додайте до нього таблицю ***Виручка*** з полями ***Дата***, ***Кількість*** і ***Код_виробника*** із таблиці ***Продажі***, ***Товар*** – із таблиці ***Товари***, ***Група*** – із таблиці ***Групи***, а також обчисліть значення поля ***Вартість*** на основі даних полів ***Кількість*** із таблиці ***Продажі*** та ***Ціна*** з таблиці ***Товари***. Для цього в елементі **TableAdapter** побудуйте такий запит:

```
SELECT Продажі.Код_виробника, Продажі.Дата, Продажі.Кількість,
       Товари.Товар, Групи.Група,
       Товари.Ціна * Продажі.Кількість AS Вартість
FROM Групи INNER JOIN
    Товари ON Групи.Код_групи = Товари.Код_групи INNER JOIN
    Продажі ON Товари.Код_товару = Продажі.Код_товару
```

4. Створіть звіт, у якому містяться дані про результати продажу товарів вибраного виробника. Для цього:

4.1. Виконайте команду **Project – Add New Item**.

4.2. Виберіть елемент **Visual C# Items** у лівому списку вікна **Add New Item,** виділіть елемент **Report Wizard** у центральній частині вікна, уведіть ім'я ***rptВиручка*** і клацніть кнопку **Add**. З'явилося перше вікно майстра звітів **Report Wizard**.

4.3. Установіть значення елементів у першому вікні **Report Wizard**, наведені в табл. 7.7.

Таблиця 7.7

Значення елементів

Елементи	Значення
Datasource	хлібDataSet
Available datasets	Виручка

4.4. Перетягніть зі списку **Available fields** у другому вікні майстра поле ***Дата*** у список **Row groups**, поля ***Група*** і ***Товар*** – у список **Column groups**, а поля ***Кількість*** і ***Вартість*** – у список **Values** (рис. 7.44).

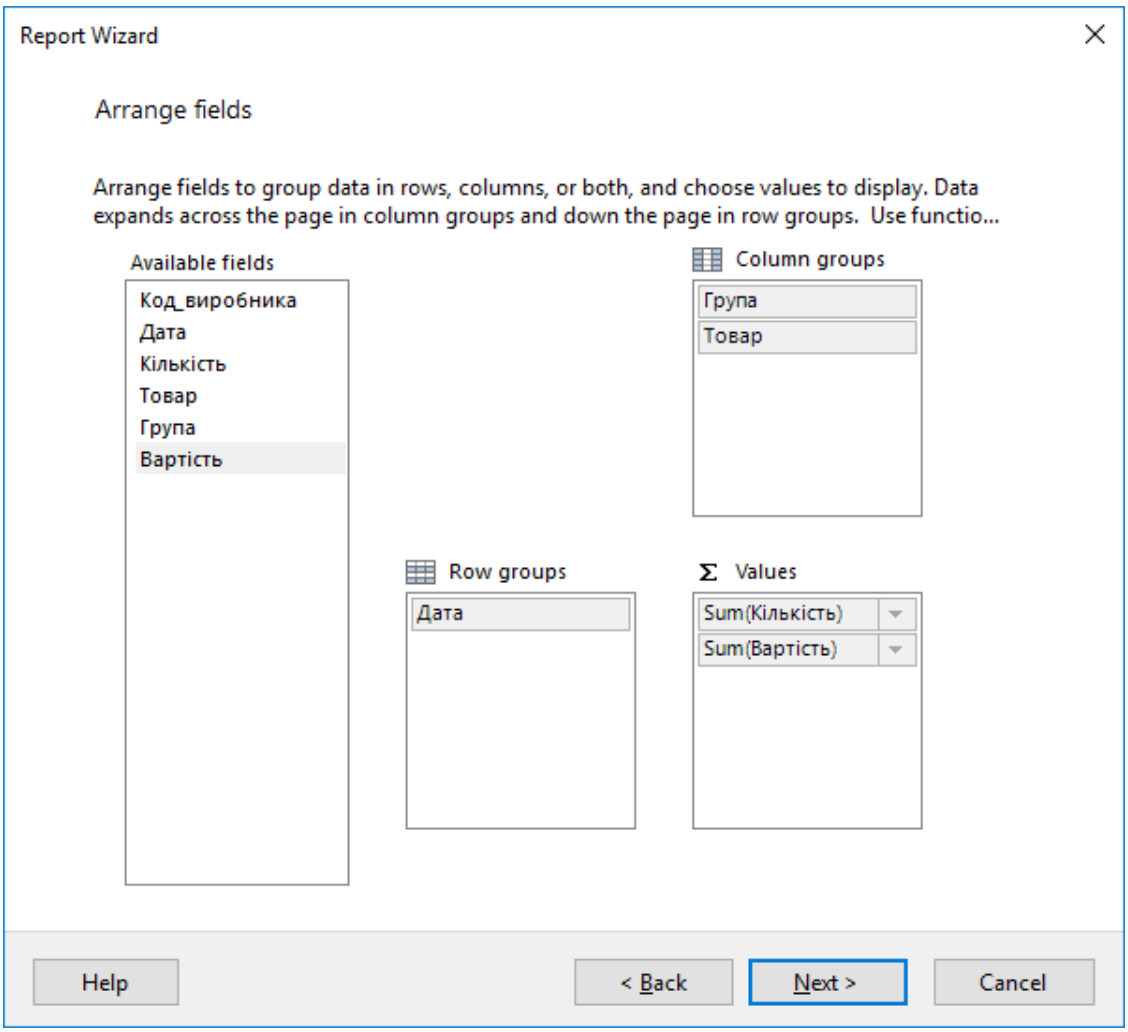


Рис. 7.44. **Друге вікно майстра**

4.5. Клацніть кнопку **Next** у третьому вікні майстра, а в четвертому – кнопку **Finish**. У вікні **Solution Explorer** з'явився значок звіту (файл з ім'ям ***rptВиручка.rdlc***).

4.6. Замініть в усіх текстових полях звіту слово ***Total*** на ***Усього***, а у другому рядку стовпця ***Дата*** замініть значення поля ***Дата*** на такий вираз:

```
=Format(Fields!Дата.Value,"dd/MM/yyyy")
```

Отримують звіт ***rptВиручка***, зовнішній вигляд якого в конструкторі подано на рис. 7.44.

5. Додайте на форму ***Виручка*** елемент **ReportViewer** і у його смарт-тегу **ReportViewer Tasks** виберіть значення ***rptВиручка.rdlc*** у полі зі списком **Choose Report**.

6. Перейдіть у вікно коду форми ***Виручка*** і змініть код оброблювача події завантаження форми, щоб він набув такого вигляду:

```
private void formВиручка_Load(object sender, EventArgs e)
{
    // Завантажуємо дані для comboBoxВиробники
    this.виробникиTableAdapter.Fill(this.хлібDataSet.Виробники);

    // Завантажуємо  всі дані для звіту в локальну таблицю
    this.ВиручкаTableAdapter.Fill(this.хлібDataSet.Виручка);

    // Відбираємо дані про продаж товарів вибраного виробника
    ВиручкаBindingSource.Filter = "Код_виробника = "
        + comboBoxВиробник.SelectedValue;

    this.reportViewer1.RefreshReport();
}
```

7. Перейдіть у вікно конструктора форми ***Виручка*** і двічі клацніть поле зі списком ***Виробник***. Потім додайте код оброблювача події "Зміна вибраного елемента".

```
private void comboBoxВиробник_SelectedIndexChanged(object sender, Even-
tArgs e)
{
    // Відбираємо дані про продаж товарів вибраного виробника
    ВиручкаBindingSource.Filter = "Код_виробника = "
        + comboBoxВиробник.SelectedValue;
    this.reportViewer1.RefreshReport();
}
```

8. Перейдіть у вікно конструктора форми ***Звіти***, двічі клацніть кнопку ***Виручка*** і додайте код оброблювача події "Клацання на кнопці ***Виручка***".

```
private void buttonВиручка_Click(object sender, EventArgs e)
{
    formВиручка вікноВиручка = new formВиручка();
    вікноВиручка.ShowDialog();
}
```

9. Перевірте функціональність форми ***Виручка***. Має з'явитися форма, подана на рис. 7.42. Перегляньте дані для різних виробників і товарів у різних групах.

4.2. Кругова діаграма

Завдання

Побудуйте зведену діаграму для аналізу структури результатів продажів товарів кожного виробника за групами (рис. 7.45).

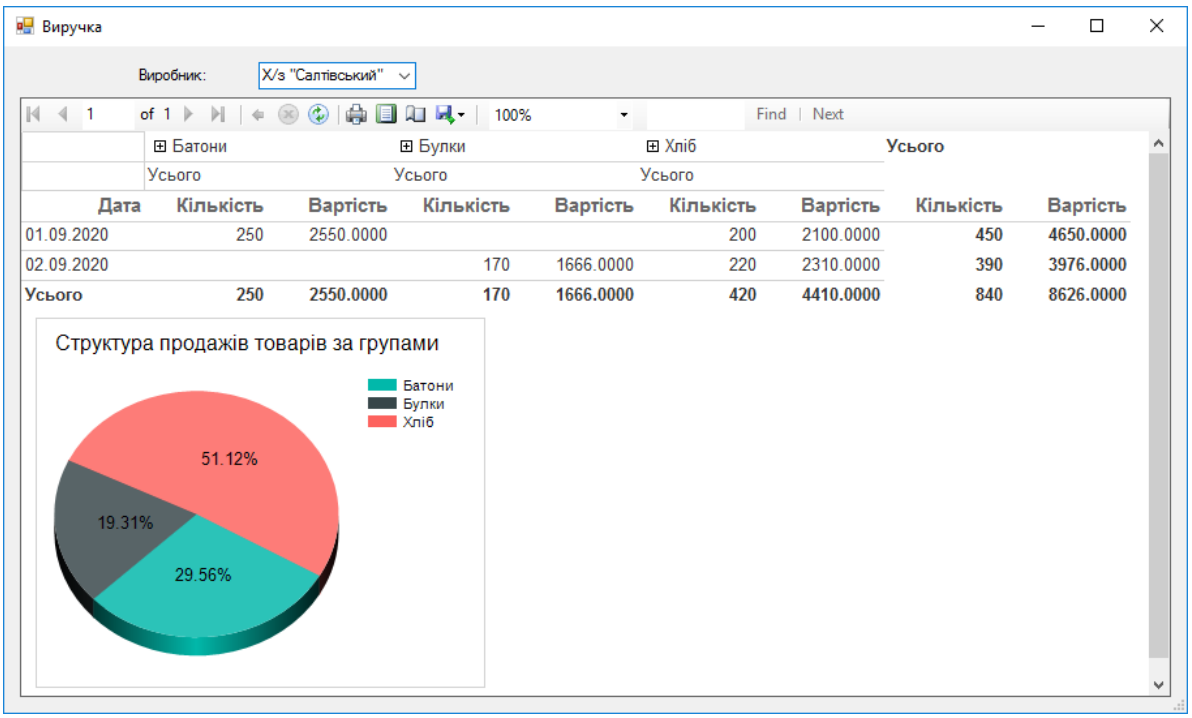


Рис. 7.45. **Зведена кругова діаграма *Структура продажів товарів за групами* на формі *Виручка***

Ідеї виконання

Як і в попередньому завданні для відбору даних про продажі товарів потрібного виробника використовуйте поле зі списком ***Виробник***.

Для відображення структури результатів продажів товарів вибраного виробника за групами використайте кругову діаграму. Її побудуйте на локальній таблиці ***Виручка***, із якої візьмемо поля ***Група*** і ***Вартість***.

Зовнішній вигляд кругової діаграми в конструкторі подано на рис. 7.46.

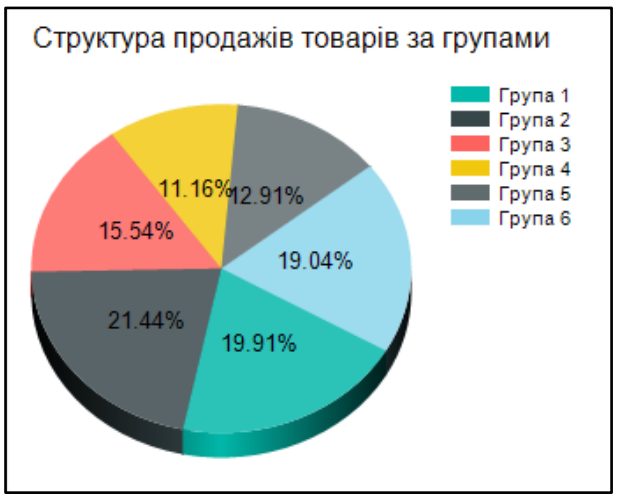


Рис. 7.46. **Кругова діаграма в конструкторі**

Виконання

1. Відкрийте в конструкторі звіт ***rptВиручка*** і збільште його висоту.

2. Перетягніть елемент **Chart** із панелі елементів на звіт, розмістивши його під зведеною таблицею.

3. Виберіть тип діаграми **3-D Pie** (тривимірна кругова діаграма) у вікні, що з'явилося. У звіті з'явився прототип майбутньої діаграми. Він має ім'я ***Chart1***.

4. Клацніть у вільному місці виділеної діаграми, щоб з'явилися області даних, відкрийте вікно **Report Data** (**Ctrl+Alt+D**) і за допомогою кнопок **+** додайте поле ***Вартість*** в область **Values**, а поле ***Група*** в область **Category Groups** (рис. 7.47).

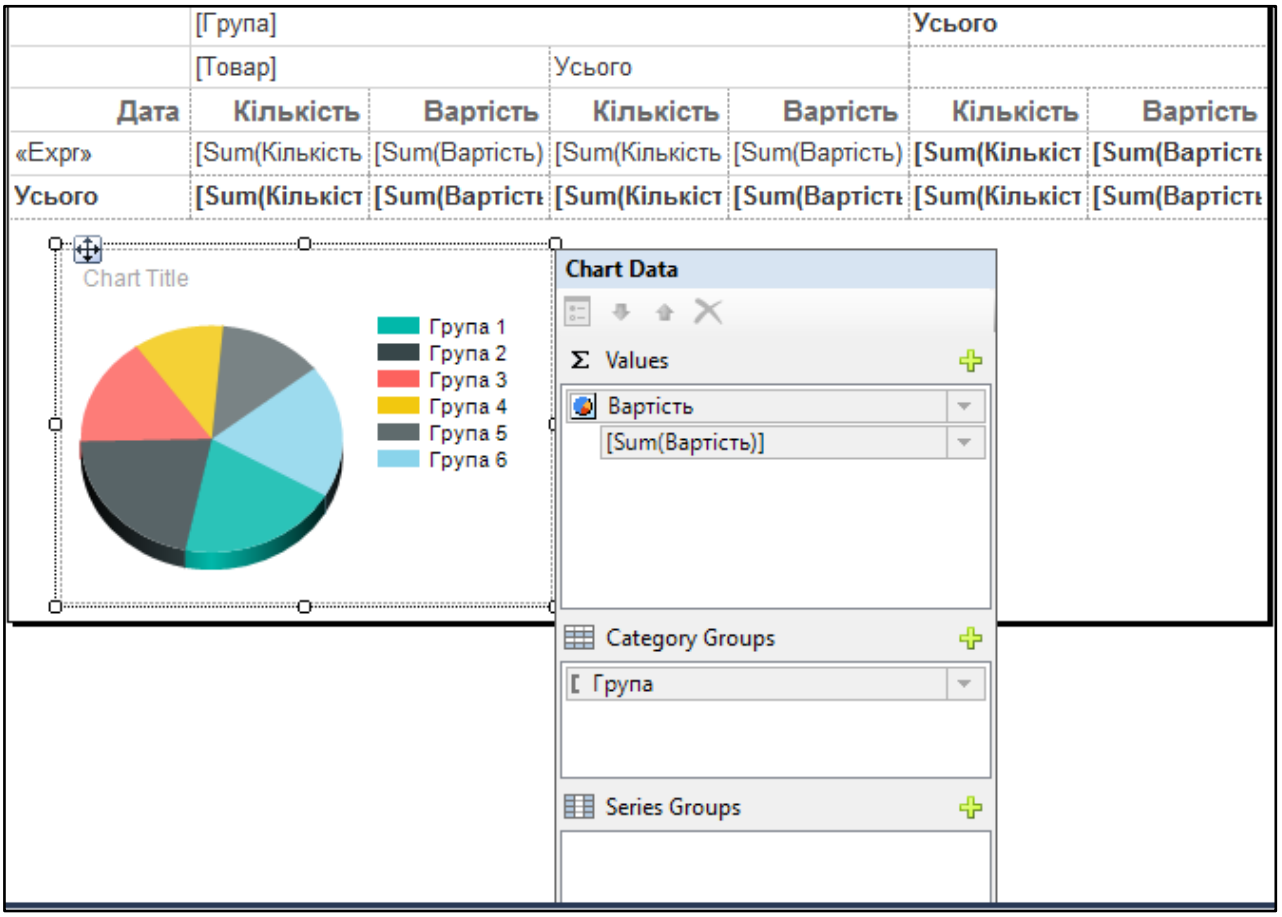


Рис. 7.47. **Дані кругової діаграми**

Примітка. Раніше описано задавання даних для кругової діаграми у Visual Studio 2017. У Visual Studio 2010 цю операцію здійснюють шляхом перетягування поля ***Група*** на область рядів (праворуч від діаграми), а поля ***Вартість*** – на область даних (над діаграмою).

5. Уведіть текст заголовка діаграми ***Структура продажів товарів за групами***. У вікні властивостей установіть значення, наведені в табл. 7.8.

Таблиця 7.8

Значення властивостей заголовка діаграми

Властивості	Значення
Color	Black
Font – FontSize	12pt

6. Клацніть саму діаграму, щоб у вікні властивостей відображалися властивості **Chart Series** (ряди діаграми), розкрийте вузол властивостей **Label** і встановіть значення властивостей значення, наведені в табл. 7.9.

Таблиця 7.9

Значення властивостей написів

Властивості	Значення
Label	#PERCENT
Visible	True

7. Для усієї діаграми встановіть значення **Solid** для властивості **BorderStyle – Default**.

8. Перевірте функціональність кругової діаграми. Після завантаження форми ***Виручка*** має з'явитися звіт з діаграмою, подана на рис. 7.45. Виконайте перевірку для різних значень поля зі списком ***Виробник***.

4.3. Стовпчикова діаграма

Завдання

Побудуйте зведену діаграму для візуального аналізу динаміки продажів товарів кожного виробника за групами (рис. 7.48).

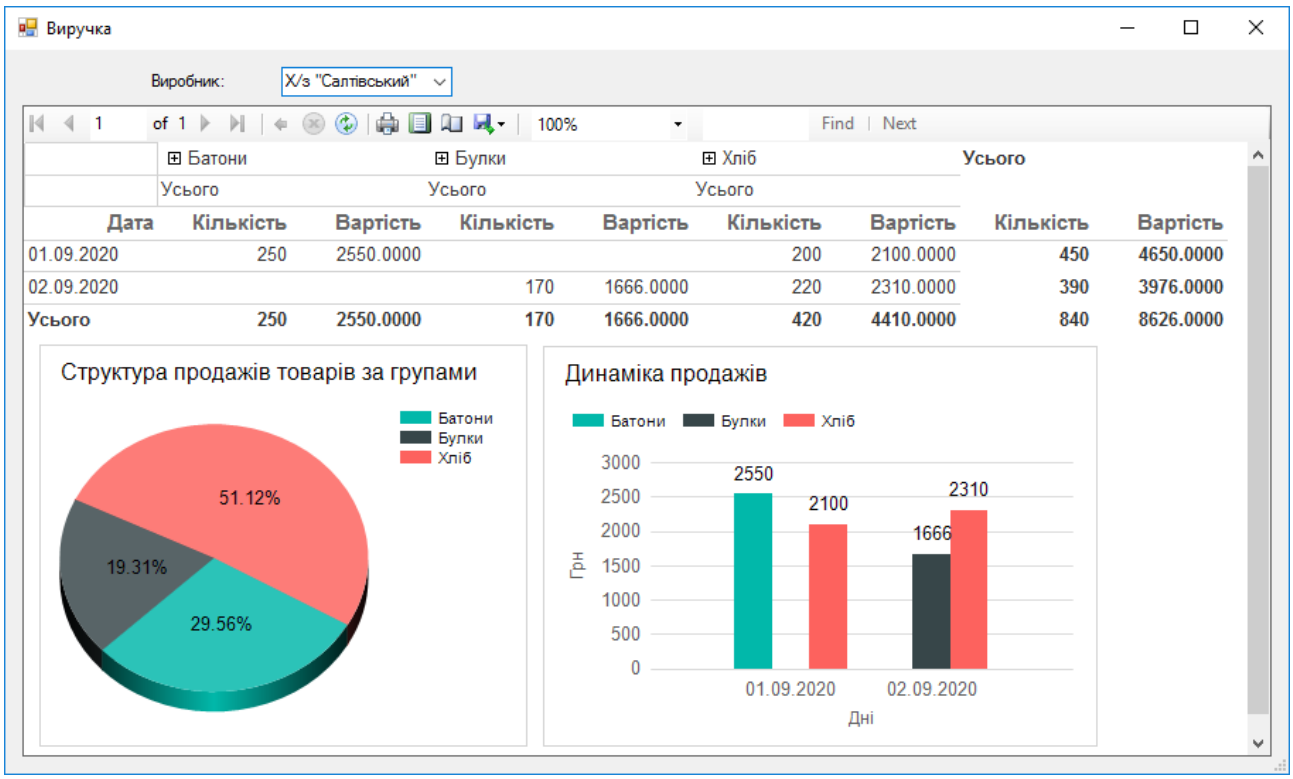


Рис. 7.48. **Зведена стовпчикова діаграма *Динаміка продажів* на формі *Виручка***

Ідеї виконання

Побудову стовпчикової діаграми здійснюють подібно до того, як це було зроблено в попередньому завданні.

Для відображення динаміки результатів продажів товарів візьміть із локальної таблиці ***Виручка*** поля ***Дата***, ***Група*** і ***Вартість***.

Зовнішній вигляд стовпчикової діаграми в конструкторі подано на рис. 7.49.

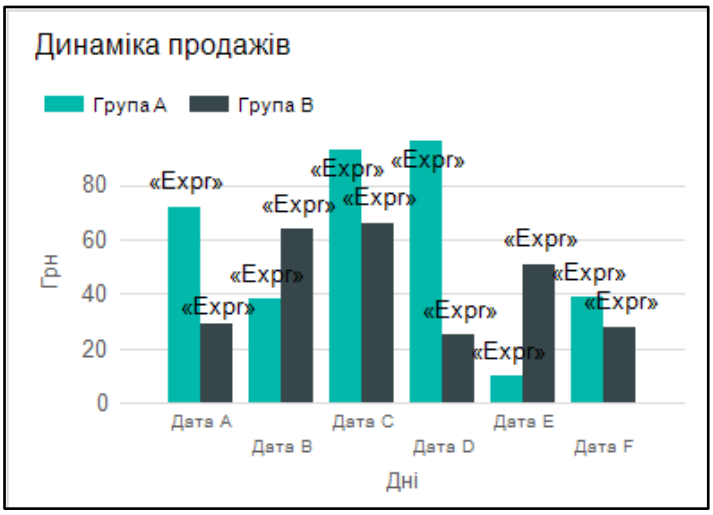


Рис. 7.49. **Стовпчикова діаграма в конструкторі**

Виконання

1. Відкрийте в конструкторі звіт ***rptВиручка*** і перетягніть елемент **Chart** із панелі елементів на звіт, розмістивши його праворуч від кругової діаграми.

2. Погодьтеся з тим, що будують стовпчикову діаграму, клацнувши кнопку **ОК** у вікні **Select Chart Type**.

3. Клацніть у вільному місці виділеної діаграми, щоб з'явилися області даних, відкрийте вікно **Report Data** (**Ctrl+Alt+D**) і за допомогою кнопок **+** додайте поле ***Вартість*** в область **Values**, поле ***Дата*** – в область **Category Groups**, а поле ***Група*** – в область **Series Groups** (рис. 7.50).

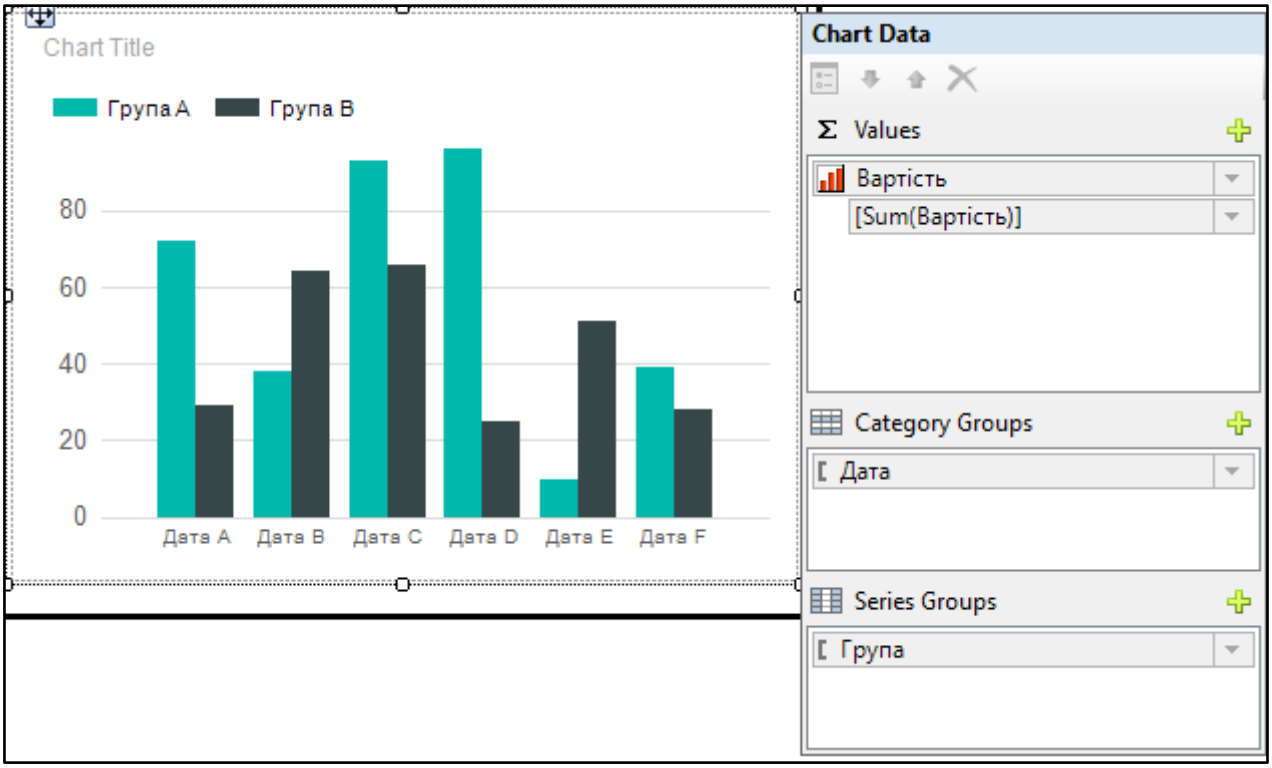


Рис. 7.50. **Дані кругової діаграми**

Примітка. Раніше описано задавання даних для стовпчикової діаграми у Visual Studio 2017. У Visual Studio 2010 цю операцію здійснюють шляхом перетягування поля ***Дата*** на область категорій (нижче від діаграми), поля ***Група*** – на область рядів (праворуч від діаграми), а поля ***Вартість*** – на область даних (над діаграмою).

4. Відформатуйте виведення дати на горизонтальній осі. Для цього клацніть поле ***Дата*** в області **Category Groups** та у вікні властивостей задайте для властивості **Label** таке значення:

```
=Format(Fields!Дата.Value,"dd/MM/yyyy")
```

5. Уведіть текст заголовка діаграми ***Динаміка продажів***. У вікні властивостей установіть значення, наведені в табл. 7.10.

Таблиця 7.10

Значення властивостей заголовка діаграми

Властивості	Значення
Color	Black
Font – FontSize	12pt

6. Уведіть заголовок горизонтальної осі ***Дні*** та вертикальної осі – ***Грн***.

7. Клацніть поле ***Вартість*** в області **Values**, у вікні властивостей розкрийте вузол **Label** і встановіть значення властивостей значення, наведені в табл. 7.11.

Таблиця 7.11

Значення властивостей поля Вартість

Властивості	Значення
Label	=Round(Fields!Вартість.Value)
Visible	True

8. Для усієї діаграми встановіть значення **Solid** для властивості **BorderStyle – Default**.

9. Перевірте функціональність стовпчикової діаграми. Після завантаження форми ***Виручка*** має з'явитися звіт із діаграмами, поданими на рис. 7.48. Виконайте перевірку для різних значень поля зі списком ***Виробник***.

У звіті з лабораторної роботи подайте скриншот форми ***Виручка*** з табличними даними та діаграмами в режимі виконання. Поясніть, за допомогою яких засобів забезпечено інтерактивність аналізу, залежно від вибраного виробника. Порівняйте трудомісткість розроблення форм для аналізу даних на основі технології LINQ, що використовували в лабораторній роботі 5 і поточній.

Завдання для самостійного виконання

1. Підготуйте листи кожному виробникові, у яких подайте діаграму ***Динаміка продажів товарів за групами*** у вигляді графіка з маркерами та поштову наклейку з адресою для конверта. Передбачте можливість друкувати наклейки відразу без попереднього перегляду.

2. Побудуйте багаторівневий звіт ***Щоденні залишки товарів***, у якому подайте кількість непроданих товарів кожного дня. Залишки обчисліть як різницю між отриманою кількістю товарів кожного виду за накладними та результатами продажів. За основу візьміть звіт ***rptЩоденні_продажі***.

3. Повторіть завд. 2, подавши залишки товарів у вигляді зведеної таблиці та діаграм.

4. Побудуйте зведену таблицю та діаграми для аналізу результатів продажів товарів вибраної групи (кількість і вартість) за кожний день. Дані про продаж товарів кожного виробника подайте окремими стовпцями.

5. Додайте до звіту ***Щоденні продажі*** можливість відбирати дані про певного виробника за заданий період. Відбір даних виконайте окремо на сервері та в наборі даних.

6. Додайте на форму ***Накладні*** можливість відбирати документи за такими критеріями: номер накладної, діапазон дат, виробник, діапазон сум.

7. Виконайте п. 1 – 4 ходу роботи для індивідуальної бази даних.

8. Виконайте п. 1 – 4 ходу роботи, використавши технологію Code First, а не типізовані набори даних, як це описано в поточній лабораторній роботі.

Рекомендована література

1. Федько В. В. Організація баз даних та знань : навч.-практ. посіб. для самост. підготовки студ. / В. В. Федько, О. В. Тарасов, М. Ю. Лосєв. – Харків : ХНЕУ, 2013. – 198 с.

2. Федько В. В. Сучасні засоби доступу до даних : навч. посіб. для самост. роб. студ. з навч. дисципліни "Організація баз даних та знань" для студентів напряму підготовки 6.050101 "Комп'ютерні науки" / В. В. Федько, О. В. Тарасов, М. Ю. Лосєв. – Харків : Вид. ХНЕУ ім. С. Кузнеця, 2014. – 328 с. ; [Електронний ресурс]. – Режим доступу : http://www.repository.hneu.edu.ua/jspui/handle/123456789/7724. – ISBN 978-966-676-531-7.

3. Технології баз даних [Електронний ресурс] : робоча програма навчальної дисципліни для студентів галузі знань 12 "Інформаційні технології" першого (бакалаврського) рівня / уклад. В. В. Федько, М. Ю. Лосєв. – Харків : ХНЕУ ім. С. Кузнеця, 2018. – 19 с. – Режим доступу : http://www.repository.hneu.edu.ua/handle/123456789/20089.

4. Федько В. В. Классические средства доступа к данным [Электронный ресурс] : учеб. пособ. по учеб. дисциплине "Базы данных" для иностр. студ. / В. В. Федько, А. В. Тарасов, М. Ю. Лосев. – Мультимедийное интерактивное электрон. изд. комбинированного использования (819 Мб). – Харьков : ХНЭУ им. С. Кузнеца, 2016. – 218 с. – Режим доступа : http://library.hneu.edu.ua/e-media. – Загл. с тит. экрана. – ISBN 978-966-676-663-5.

5. Microsoft Visual Studio: Downloads [Electronic resource]. – Access mode : https://visualstudio.microsoft.com/downloads.

6. NetBeans IDE 8.2 Download [Electronic resource]. – Access mode : https://netbeans.org/downloads/8.2.

Зміст

НАВЧАЛЬНЕ ВИДАННЯ

Федько Віктор Васильович

ТЕХНОЛОГІЇ БАЗ ДАНИХ

Лабораторний практикум

Самостійне електронне текстове мережеве видання

Відповідальний за видання *І. О. Ушакова*

Відповідальний редактор *М. М. Оленич*

Редактор *О. Г. Доценко*

Коректор *О. Г. Доценко*

План 2020 р. Поз. № 14-ЕНП. Обсяг 344 с.

Видавець і виготовлювач – ХНЕУ ім. С. Кузнеця, 61166, м. Харків, просп. Науки, 9-А

Свідоцтво про внесення суб'єкта видавничої справи до Державного реєстру
ДК № 4853 від 20.02.2015 р.