**МІНІСТЕРСТВО ОСВІТИ І НАУКИ,
МОЛОДІ ТА СПОРТУ УКРАЇНИ**

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ

Методичні рекомендації до виконання лабораторних робіт з навчальної дисципліни "ВИДАВНИЧІ БАЗИ ДАНИХ"

**для студентів галузі знань 0515
"Видавничо-поліграфічна справа"
всіх форм навчання**

Харків. Вид. ХНЕУ, 2012

Затверджено на засіданні кафедри комп’ютерних систем і технологій.
Протокол № 2 від 04.10.2011 р.

Укладач Пандорін О. К.

М54 Методичні рекомендації до виконання лабораторних робіт з навчальної дисципліни "Видавничі бази даних" для студентів галузі знань 0515 "Видавничо-поліграфічна справа" всіх форм навчання / укл. Пандорін О. К. – Х. : Вид. ХНЕУ, 2012. – 56 с. (Укр. мов.)

Подано методичні рекомендації до виконання лабораторних робіт за першими п’ятьма темами даної навчальної дисципліни.

Рекомендовано для студентів галузі знань 0515 "Видавничо-поліграфічна справа".

Вступ

Навчальна дисципліна "Видавничі бази даних" вивчається студентами галузі знань 0515 "Видавничо-поліграфічна справа" усіх форм навчання протягом десятого семестру і є методологічною і методичною основою для подальшого опанування студентами технологій і методів розробки програмної частини мультимедійних електронних видань; організації процесу проектування компонентів, а також отримання практичних навичок самостійного опрацювання мультимедійної інформації і подання її у вигляді компонентів мультимедійних електронних видань, наукових знань та технологій, що полягають у основі принципів дії систем управління видавничими підприємствами та продуктами, що проектуються або створено.

Навчальна дисципліна є методологічною, методичною та інструментальною основою для виконання аналітичної і практичної частин дисциплін, а також курсових і дипломних робіт.

Особливість даного лабораторного практикуму – орієнтація на різноманітне програмні технології, вивчення яких допомагає сформувати різнобічно підготовленого фахівця. У даному практикумі розглядаються на основі наскрізного розвитку одного додатка за індивідуального варіантом завдання методи використання різних програмних технологій розповсюдження та опрацювання контенту мультимедійних середовищ, які представлені різноманітними інформаційними технологіями.

У результаті виконання лабораторних робіт практикуму студенти повинні здобути навички проектування, розробки, впровадження та експлуатації програмних систем, що базуються на компонентному та об’єктно-орієнтованому підході.

Тексти програм визначаються за допомогою літер однакової ширини. Перехід до іншої строки у фрагментах коду, там, де він вимовлений лише можливостями цього видання, позначається символом. При використанні сучасних середовищ розробки значна частина коду має будуватися автоматично. Частини коду, які не додаються автоматично, за допомогою графічних засобів розробки, зображуються на сірому фоні.

Модуль 1. Базові концепції об’єктного та компонентного підходів до реалізації гнучких ІС мультимедійного контенту

Лабораторна робота 1. Відбудова концептуальної моделі ІС по фізичній

Мета роботи. Ознайомити студентів з основними прийомами і технологічними інструментами для організації ітеративного процесу розробки програмного продукту, який використовує абстракційні рівні концептуальної і логічної моделі даних.

Приведемо опис процесу зворотного проектування, тобто побудови концептуальної моделі по логічній.

При цьому використовуємо засоби DevArt Entity Developer, який є найбільш просунутим безкоштовним додатком відображення між концептуальною і логічною моделями.

Entity Developer це багатофункціональний інструмент для моделювання і генерації коду для технологій LinqConnect і ADO.NET Entity Framework, дозволяє згенерувати модель з нуля або здійснити реінжиніринг існуючої БД. Модель використовується для генерації C# або Visual Basic коду з використанням гнучких готових шаблонів.

Entity Developer дозволяє створити і редагувати структуру сутностей з використанням LinqConnect і LINQ to SQL візуально, без кодування у XML. Він підтримує створення всіх видів відображення як, наприклад, розбивання об'єкта на декілька таблиць, складених типів, ієрархії спадкоємства, створюючи об'єкти, інструкції SELECT, DELETE, UPDATE-реалізацію методів класу у вигляді коду SQL, і т. п. Генерація коду достатньо гнучка завдяки використанню шаблонів. Можливо створювати власні шаблони для інших мов програмування, використовувати як автономно, так і у складі Microsoft Visual Studio як будь-який інший редактор.

Даний редактор підтримує Entity Framework, LinqConnect, і LINQ to SQL. Entity Developer for SQL Server використовує стандартний SqlClient, і призначений для проектування LINQ в SQL і моделі Entity Framework. Entity Framework для dotConnect використовує провідники даних з

колекції dotConnect. Це дає можливість проектувати моделі Entity Framework як частину LinqConnect. Професійні видання dotConnect існують для Oracle, MYSQL, POSTGRESQL, і SQLite.

Використання Entity Developer продемонструємо на такому прикладі. Нехай у середовищі Microsoft SQL Server існує проста база даних магазину електронних і друкарських видань, яку можна описати такою схемою (рис. 1).

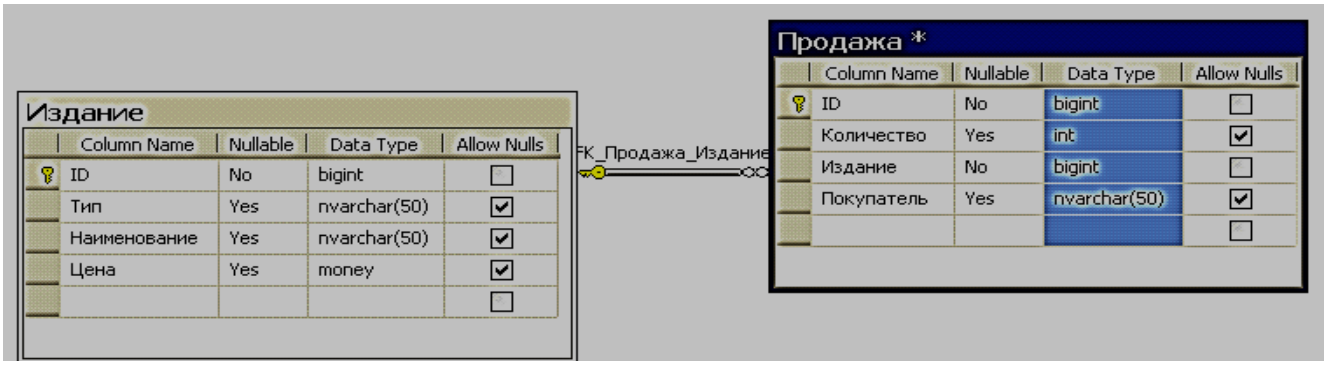

Рис. 1. **Схема БД**

Приклад заповнення таблиці Издание показано на рис. 2.

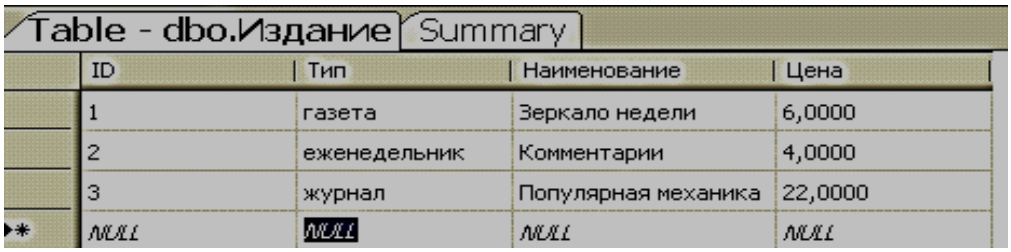

Рис. 2. **Приклад заповнення таблиці Издание**

Приклад заповнення таблиці Продажа показано на рис. 3.

Рис. 3. **Приклад заповнення таблиці Продажа**

Порядок виконання

1. Завантажити середовище DevArt Entity Developer.
2. Створити нову модель для Entity Framework. (рис. 4).

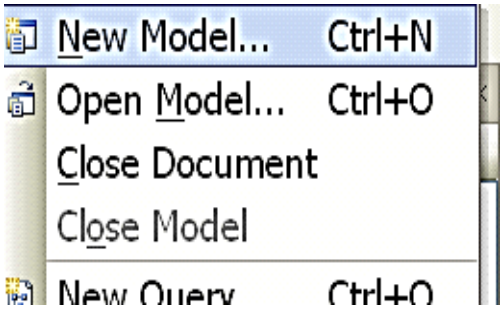

Рис. 4. **Створення нової моделі БД**

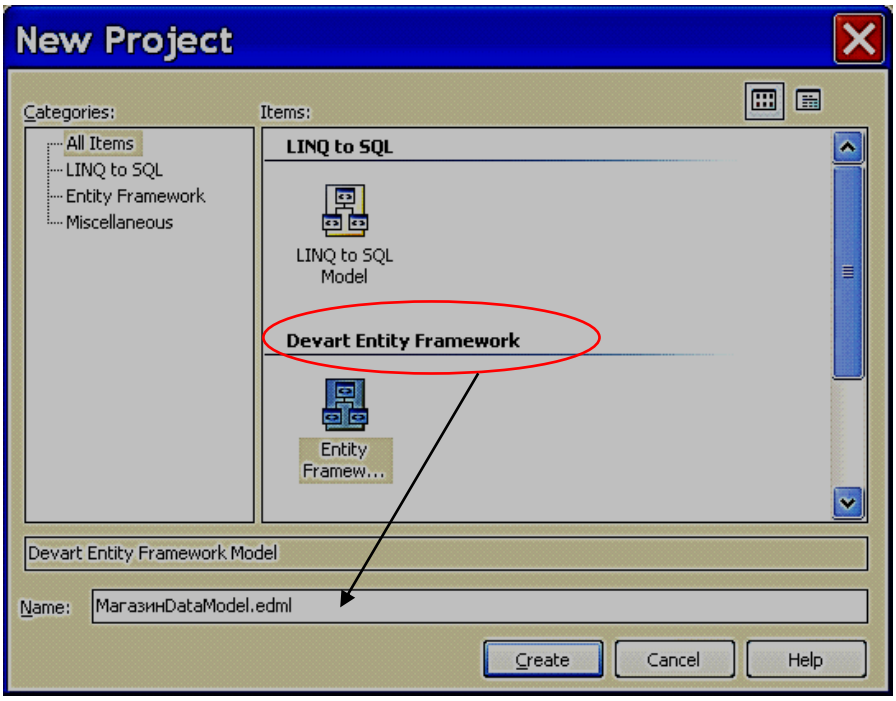

Рис. 5. **Вікно вибору типу моделі**

3. Створюємо модель за існуючою БД за допомогою майстра моделей. Майстер моделей запускається після натиснення на кнопку "Create". Показується перше вікно майстра створення моделі (рис. 6). Вказуємо, що модель створюватиметься за існуючою базою даних (рис. 5).

4. У другому вікні майстра створення моделі вибираємо сервер СКБД (рис. 7).

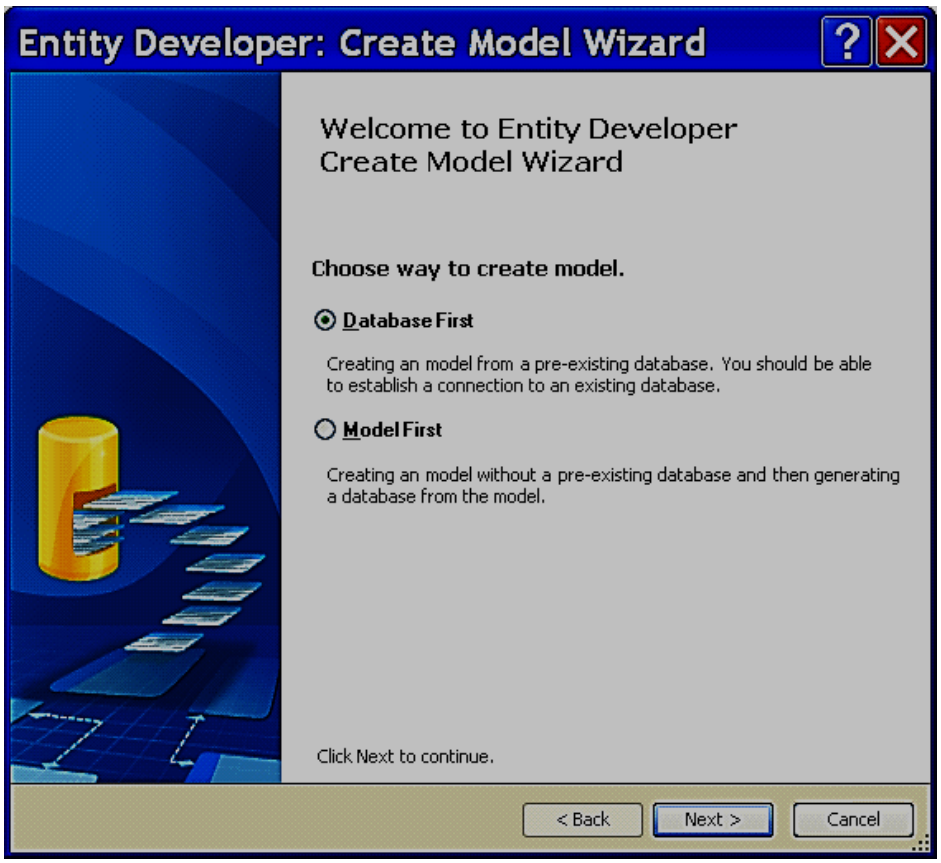

Рис. 6. **Перше вікно майстра створення моделі**

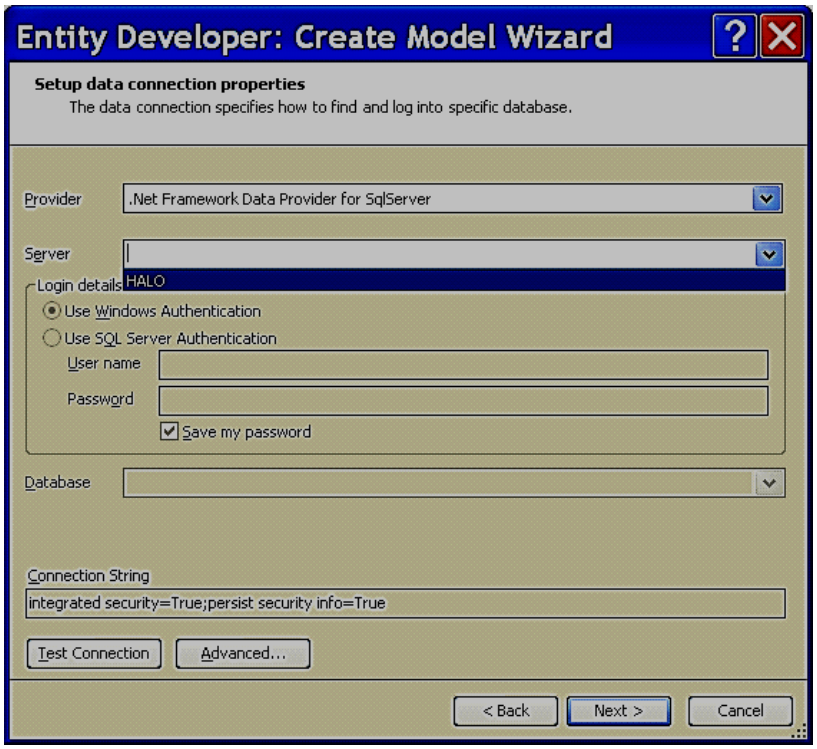

Рис. 7. **Друге вікно майстра створення моделі**

5. Виберемо базу із списку тих, що існують (рис. 8).
6. Генеруємо модель за БД (рис. 9).
7. Вибираємо всі таблиці БД (рис. 10).

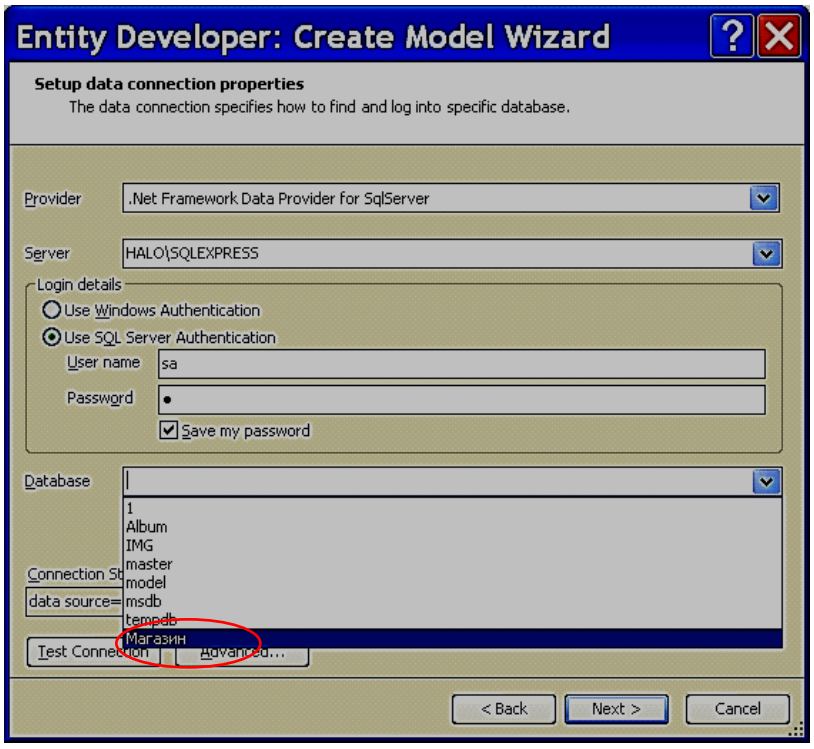

Рис. 8. **Третє вікно майстра створення моделі**

Рис. 9. **Четверте вікно майстра створення моделі**

8. Вважаючи, що при розробці БД ми керувалися розумними угодами про найменування, відмовляємося при виборі параметрів перетворення від всіх перейменувань БД (рис. 11).

9. Призначаємо мнемонічні імена для моделі і класу (рис. 12).

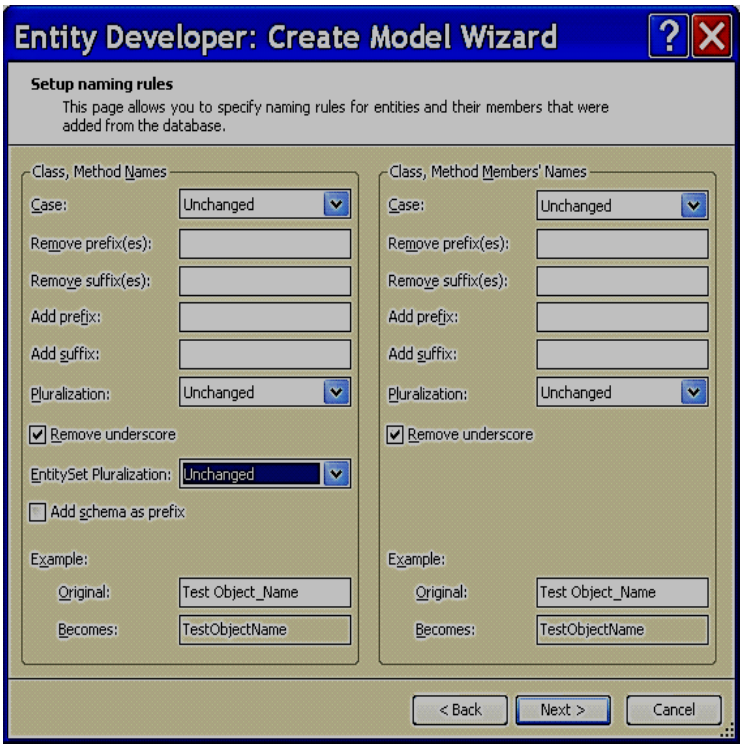

Рис.10. **П'яте вікно майстра створення моделі**

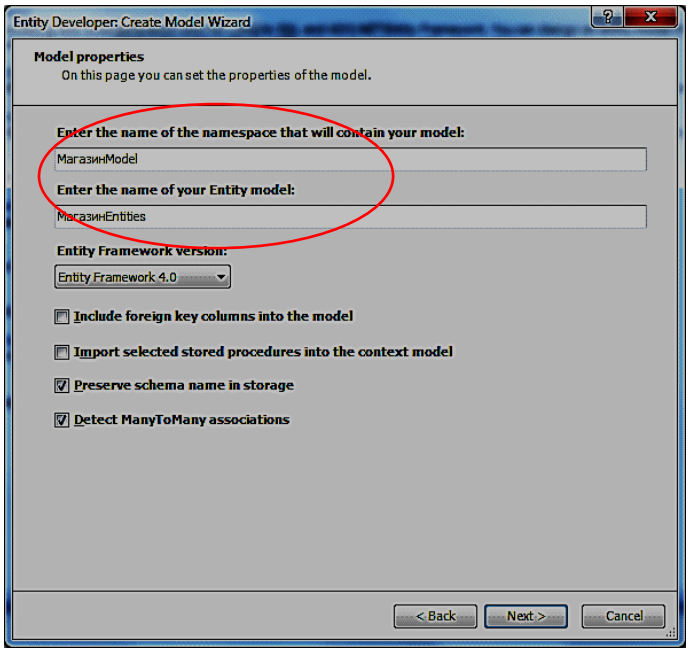

Рис. 11. **Шосте вікно майстра створення моделі**

10. Вмикаємо сутності для всіх таблиць і відміняємо генерацію навігаційних властивостей для Foreign Key (рис. 13).

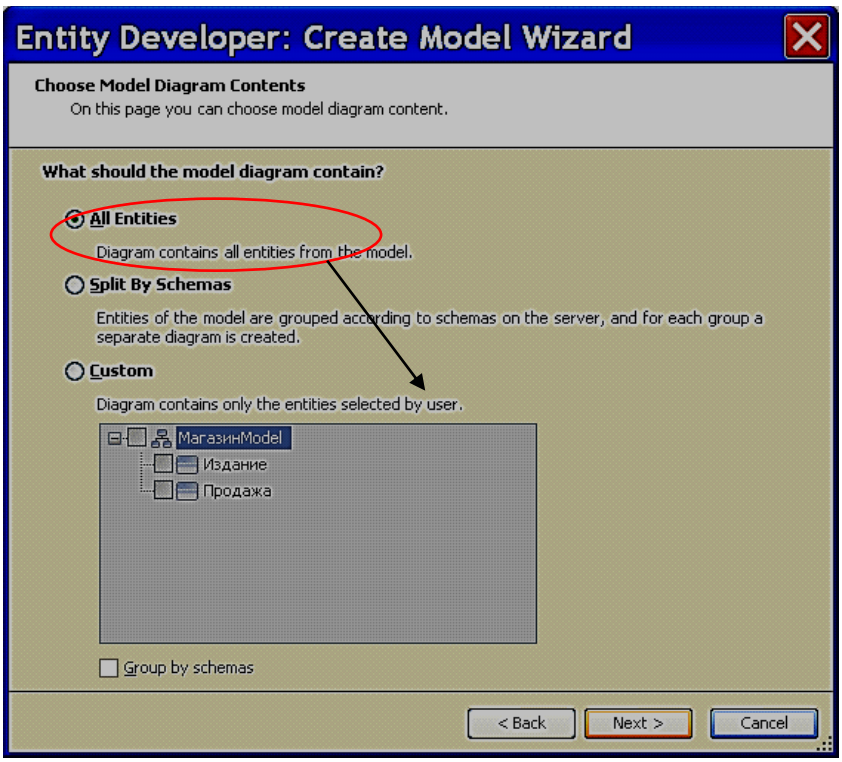

Рис.12. **Сьоме вікно майстра створення моделі**

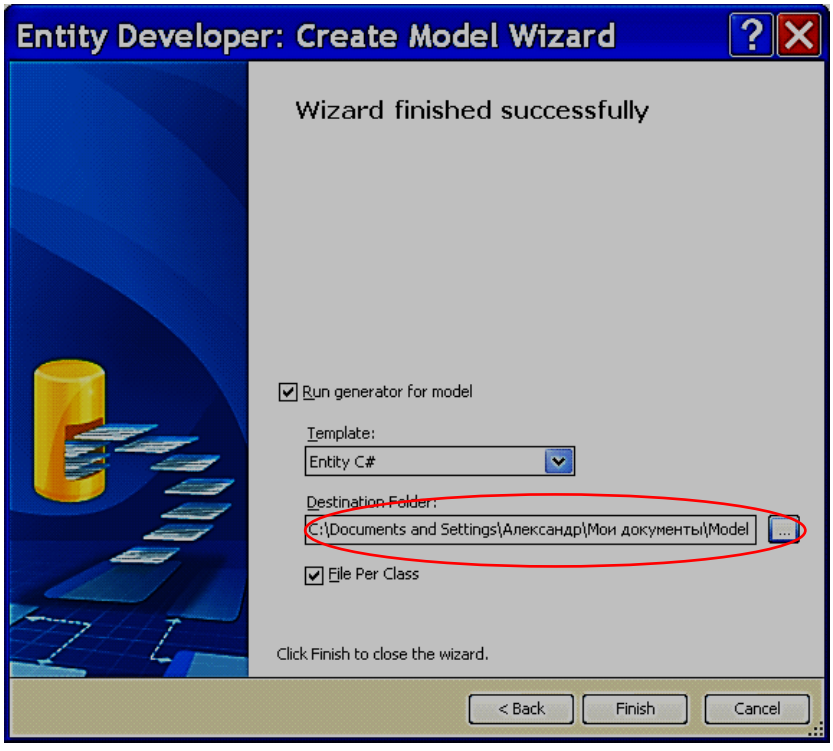

Рис.13. **Восьме вікно майстра створення моделі**

11. Вказуємо розташування файлів моделі (рис. 14).
12. Спостерігаємо за процесом генерації коду (рис. 15).

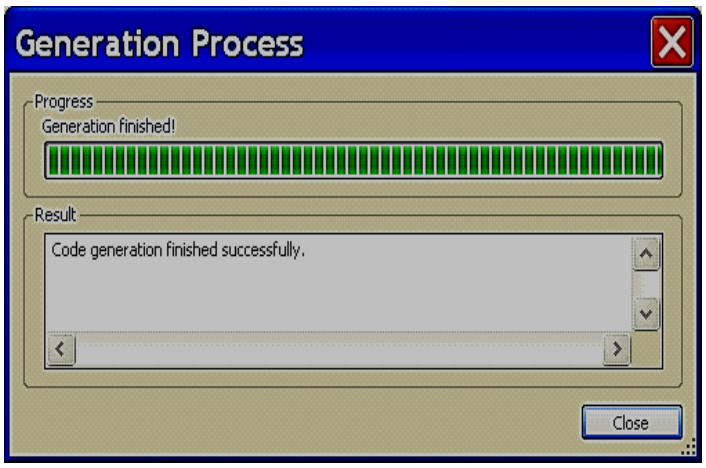

Рис. 14. **Дев'яте вікно майстра створення моделі**

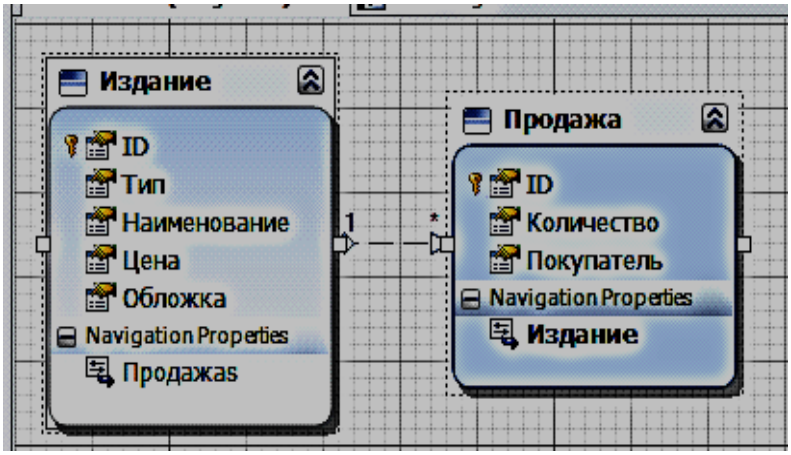

Рис. 15. **Вікно графічного представлення створеної моделі**

13. Зберігаємо модель (рис. 16, 17).
14. Надаємо моделі ім'я відповідно до семантики (рис. 18).
15. Переглядаємо подробиці відображення (рис. 19, 20)

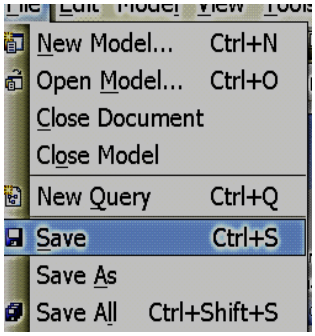

Рис. 16. **Збереження моделі**

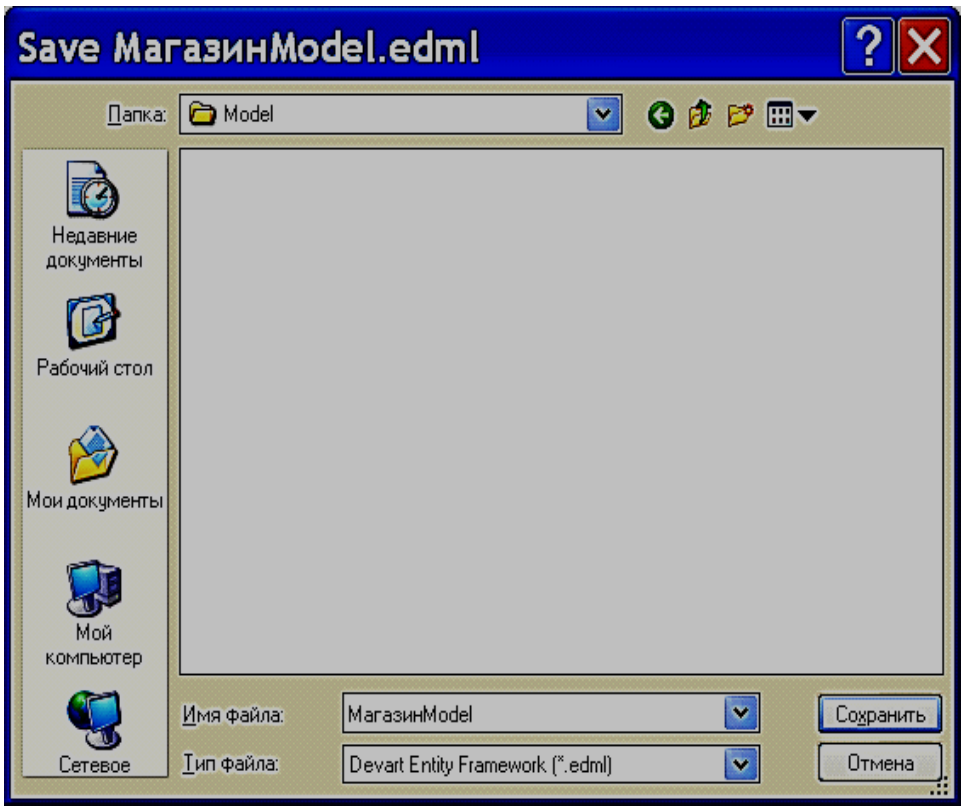

Рис. 17. **Вікно вибору місця зберігання та ім’я моделі**

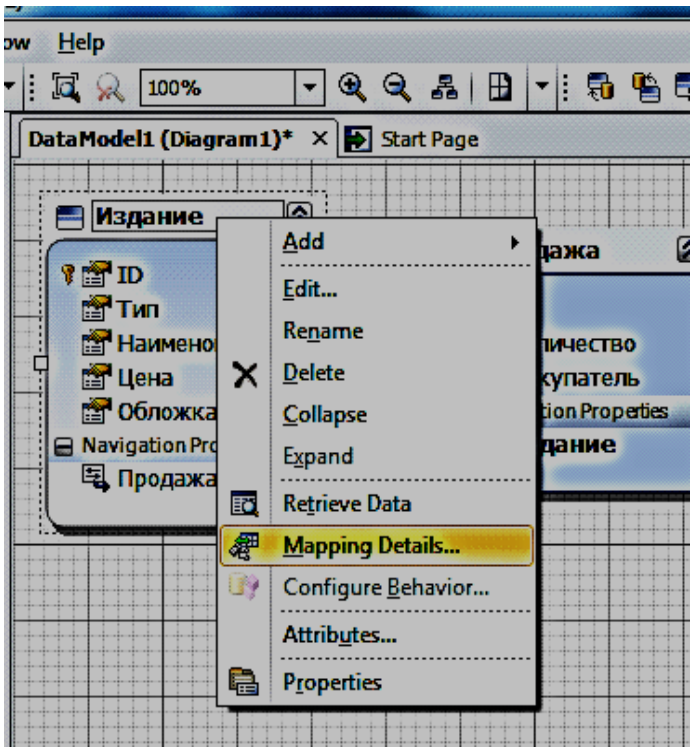

Рис. 18. **Можливі операції над сутністю**

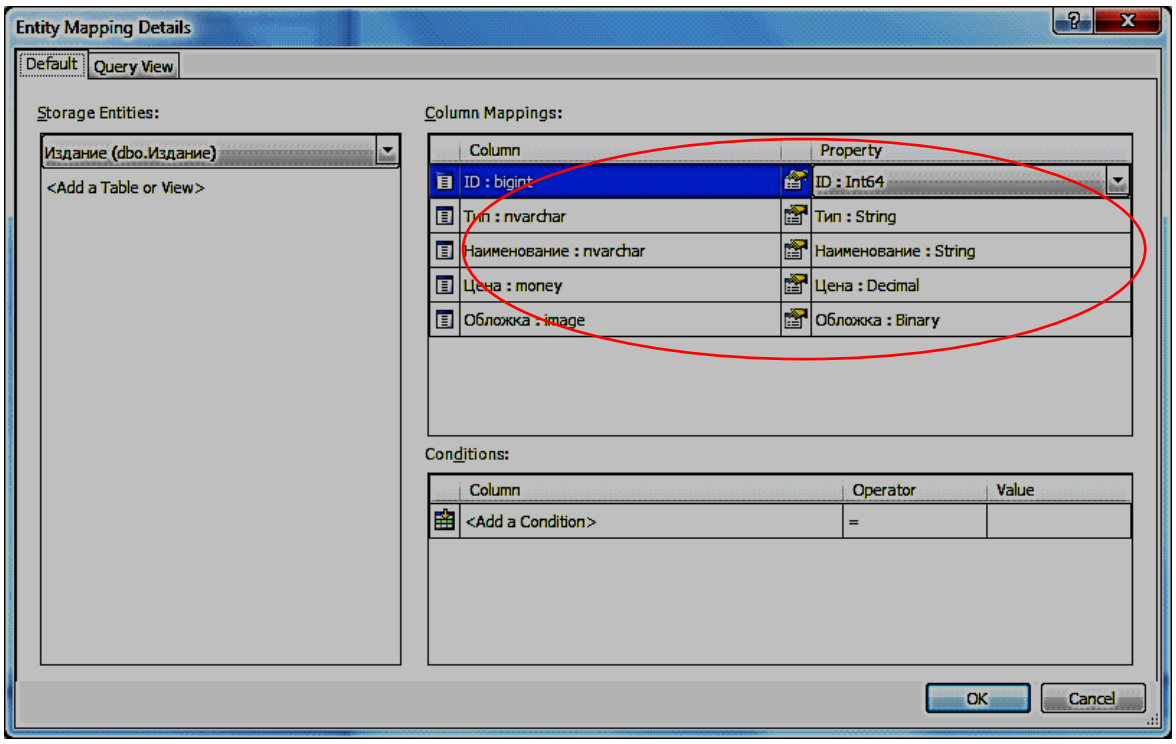

Рис. 19. **Вікно подробиць відображення**

Розглянемо файл опису сутностей, який було автоматично згенеровано, на мові C#.

Він складається з таких частин.

Для моделі додається новий простір імен.

```
namespace МагазинModel
```

Створюються класи для кожної сутності.

```
    public partial class Издание : EntityObject
```

Описується конструктор.

```
        public static Издание CreateИздание(long iD)
        {
            Издание издание = new Издание();
            издание.ID = iD;
            return издание;
        }
```

Властивості для кожного з атрибутів сутності з методами читання і запису.

```
        public long ID
        {get{long value = _ID;
                OnGetID(ref value);
                return value;
            }set{
                if (_ID != value){
```

```
                    OnIDChanging(ref value);
                    ReportPropertyChanging("ID");
                    _IDStructuralObject.
                    SetValidValue(value);
                    ReportPropertyChanged("ID");
                    OnIDChanged();
                }
            }
        }
```

Тепер у програмі на будь-якій з мов, яка підтримується .NET, можливо оперувати з об'єктами-екземплярами даного класу.

Контрольні питання

1. Навести та пояснити рівні абстракції даних при проектуванні БД.
2. Розкрити зміст поняття "концептуальна модель даних".
3. Навести та пояснити типи зв'язків між елементами даних.
4. Навести визначення й призначення понять: сутність, тип сутності, описові та ключові атрибути типу сутності.
5. Розкрити зміст понять "семантичне й об'єктне моделювання даних".

Лабораторна робота 2. Концептуальне проектування спадкоємця об’єктів відбудованої ІС

Мета роботи. Ознайомити студентів з основними прийомами і технологічними інструментами для організації спадкоємства у процесі розробки програмного продукту, який використовує базу даних як середовище зберігання стану екземплярів (тобто персістетності).

Розробники часто використовують механізми спадкоємства при роботі з однотипними об'єктами. Наприклад, можна описати клас **`Издание`**, який міститиме ключові атрибути будь-якого видання (найменування, тип, ціна) і тому подібне), а на його основі створити класи **`ЭлектронноеИздание`** і **`ПечатноеИздание`**, які міститимуть як успадковані ключові атрибути, так і додаткові атрибути, унікальні для даного типа об'єкта.

При роботі з даними, такий похід реалізувати досить складно, але модель Entity Data Model підтримує спадкоємство на рівні концептуальної моделі, що істотно спрощує роботу з даними.

Entity Data Model підтримує можливість відображення декількох таблиць на одну сутність. Таким чином, використовуючи концептуальну модель, а не логічну модель бази даних, додаток може працювати з повноцінною сутностю без необхідності у вказівці, де зберігаються окремі частини інформації.

У Entity Data Model підтримується три типи спадкоємства:

одна таблиця на ієрархію, де одна таблиця містить дані для всіх типів і колонка описує відмінності між ними;

одна таблиця на підклас, де одна таблиця містить базовий тип і окремі таблиці містять дані для підтипів;

одна таблиця на тип, де одна таблиця містить всі дані для підтипу, включаючи успадковані дані.

Ці можливості дозволяють розробникам працювати з сутностями так, як і із звичайними бізнес-об'єктами.

Сутність можуть мати простий тип даних – Binary, Boolean, Byte, DateTime, DateTimeOffset, Decimal, Double, Guid, Int16, Int32, Int64, SByte, Single, String, Time- або складний запис з полями чи екземпляр класу.

Продемонструємо процес побудови спадкоємців з використанням засобів моделювання, які вбудовані в середовище MS Visual Studio.

Порядок виконання

1. Завантажити модель Entity Framework. (рис. 20).

2. За допомогою контекстного меню (рис. 21) додати сутності-спадкоємці з іменами, що відображають семантику сутностей (рис. 22).

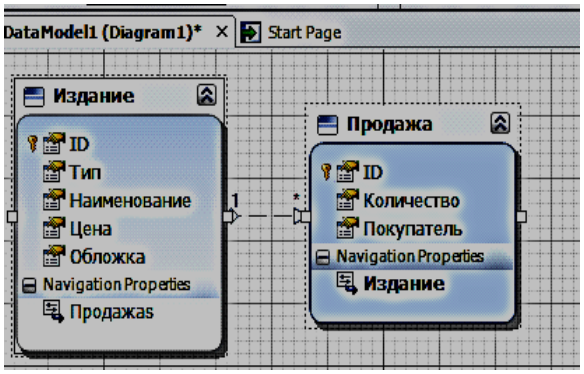

Рис. 20. **Вікно редагування моделі Entity Framework у середовищі MS Visual Studio**

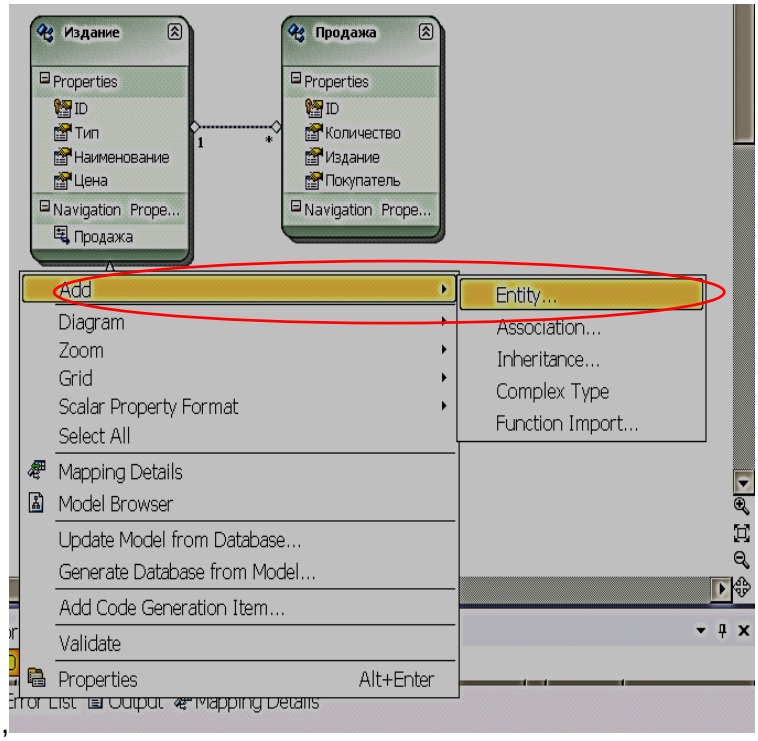

Рис. 21. **Контекстне меню операцій над моделлю**

3. За допомогою контекстного меню (рис. 23) додати властивості сутностей (рис. 22).

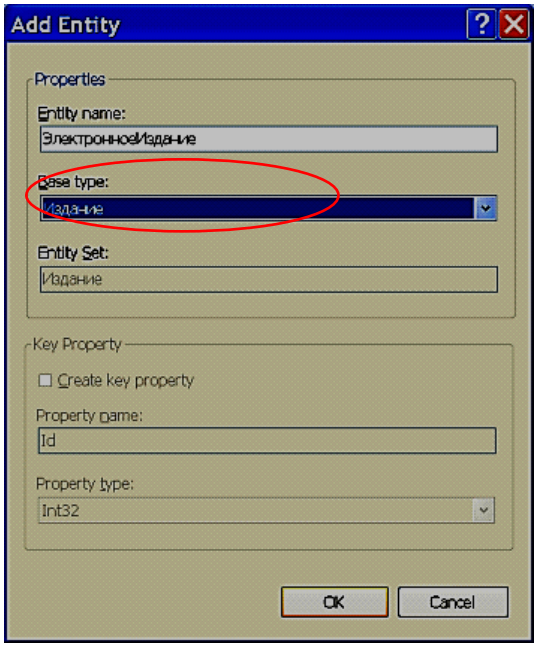

Рис. 22. **Вікно додання сутності**

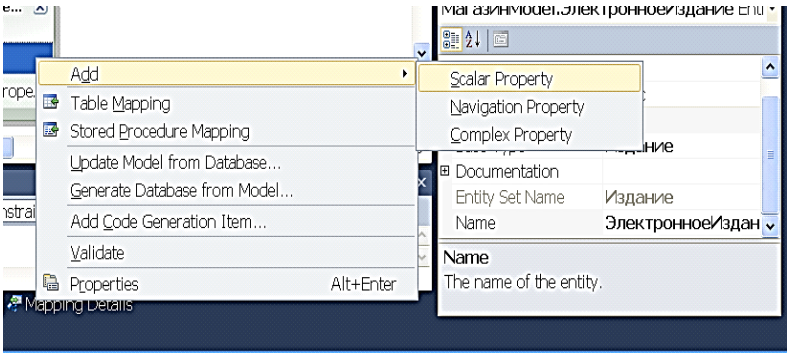

Рис. 23. **Контекстне меню операцій над сутністю**

4. Повторити крок 2 для кожної сутності-спадкоємця, а крок 3 – для кожної властивості доданих сутностей.

Для прикладу, що розглядається, після виконання кроку 4 модель буде мати такий вигляд (рис. 24).

5. Щоб було можливим створення інтерфейсних класів для сутностей моделі, яку було спроектовано, необхідно задати режим побудуви за допомогою контекстного меню моделі (рис. 25) та обрати тип моделі, що буде побудована (рис. 26), та її властивості (рис. 27).

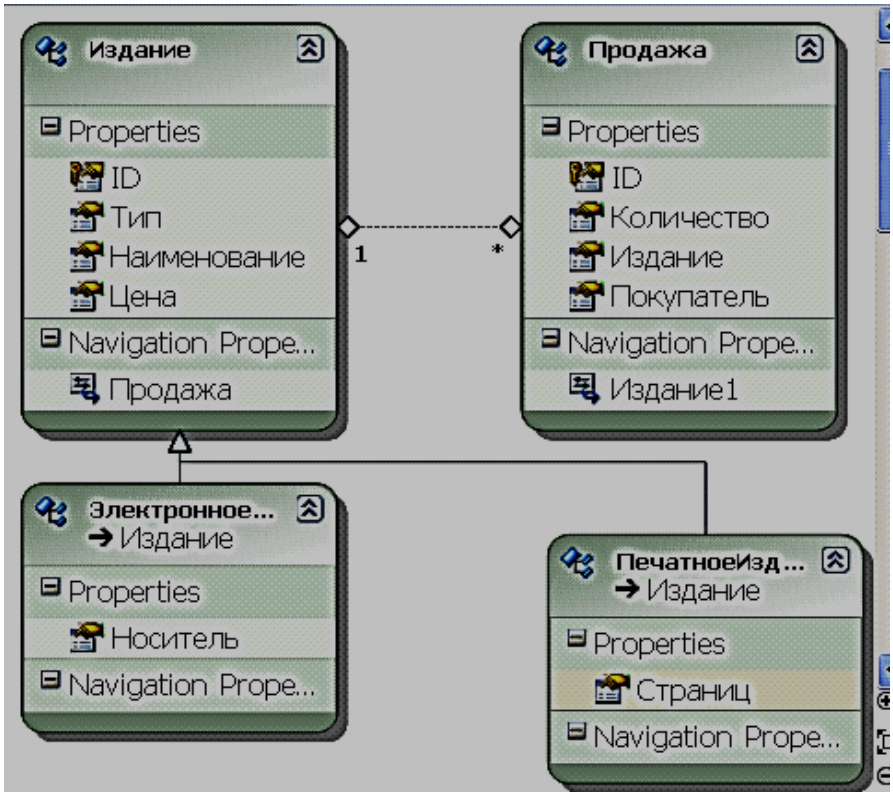

Рис. 24. **Вигляд моделі після додавання сутностей-спадкоємців та відповідних властивостей**

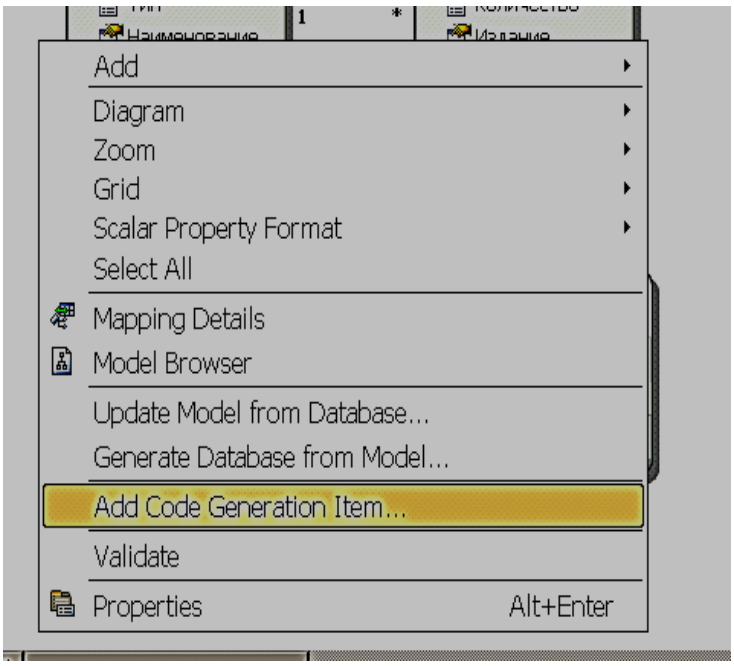

Рис. 25. **Контекстне меню операцій над моделлю**

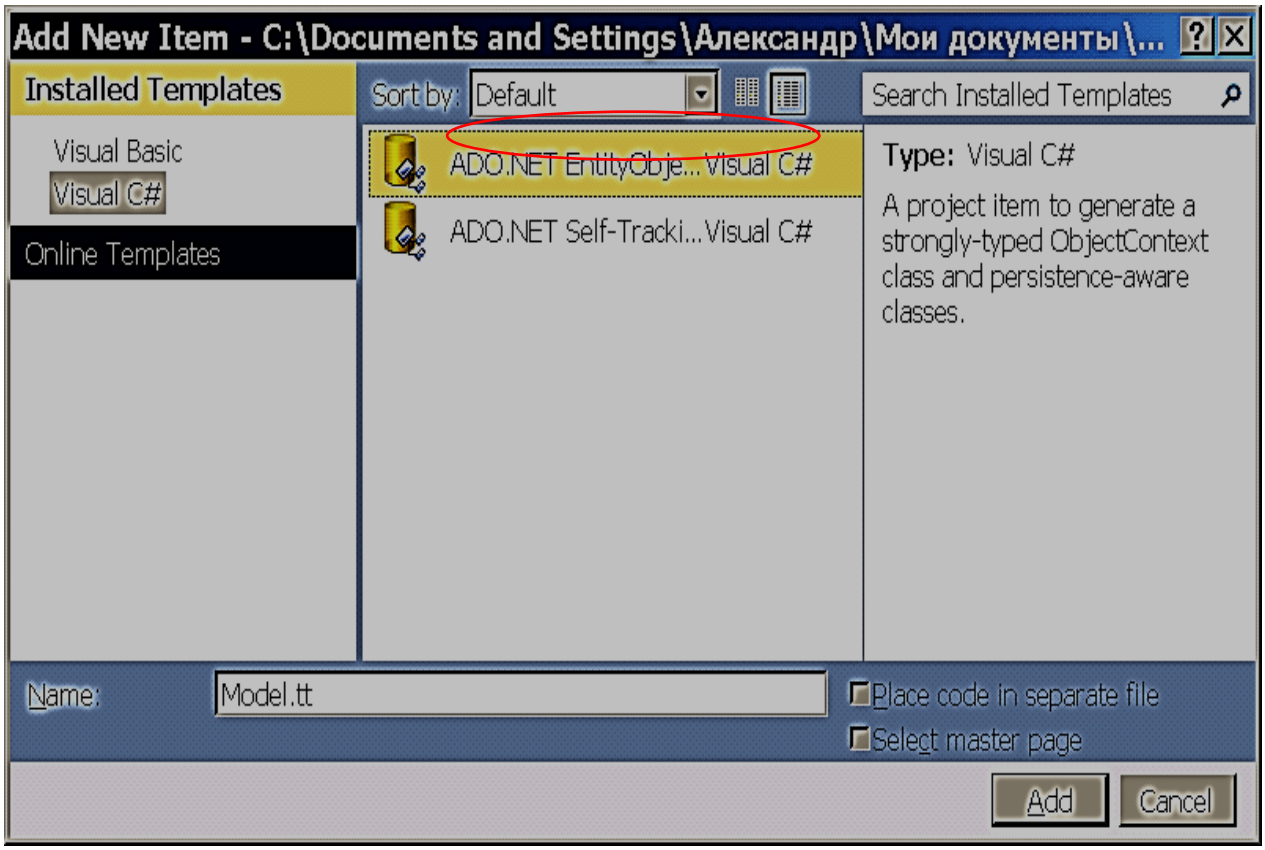

Рис. 26. **Вікно вибору типу моделі**

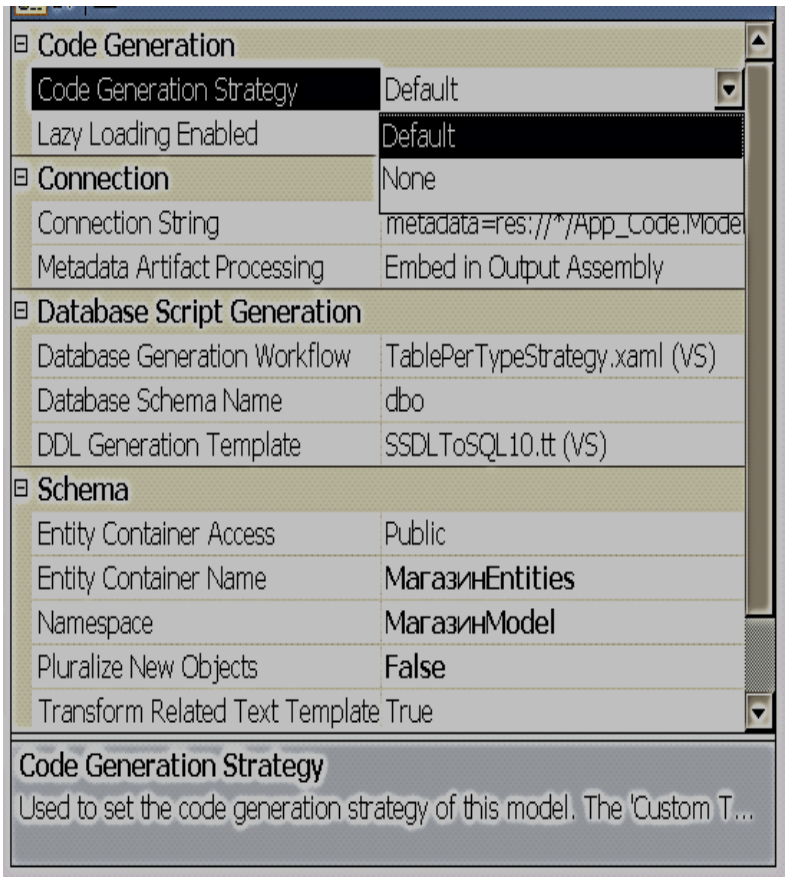

Рис. 27. **Властивості моделі**

6. Для створення інтерфейсних класів для сутностей моделі, яку було спроектовано, необхідно побудувати рішення (рис. 28).

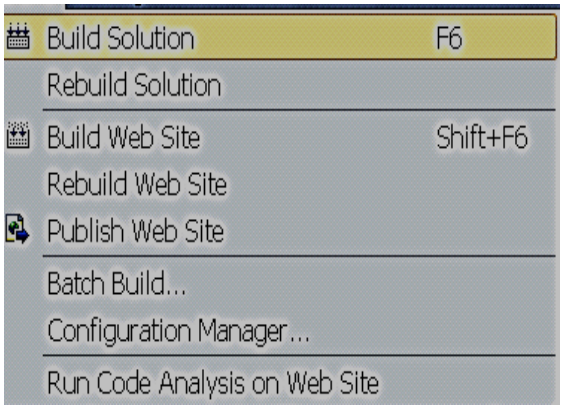

Рис. 28. **Меню операцій над сайтом та рішенням**

У згенерованому файлі з інтерфейсними класами на мові C#, до простору імен namespace МагазинModel додані класи **`ЭлектронноеИздание`** і **`ПечатноеИздание`** для сутностей, які додані у спадкоємцях

```
public partial class ЭлектронноеИздание : Издание
public partial class ПечатноеИздание : Издание
```

з відповідним описом сутностей.

7. Проведемо *валідацію моделі* – перевірку відповідності моделі сутність-зв`язок структурі відношень у базі даних (рис. 29).

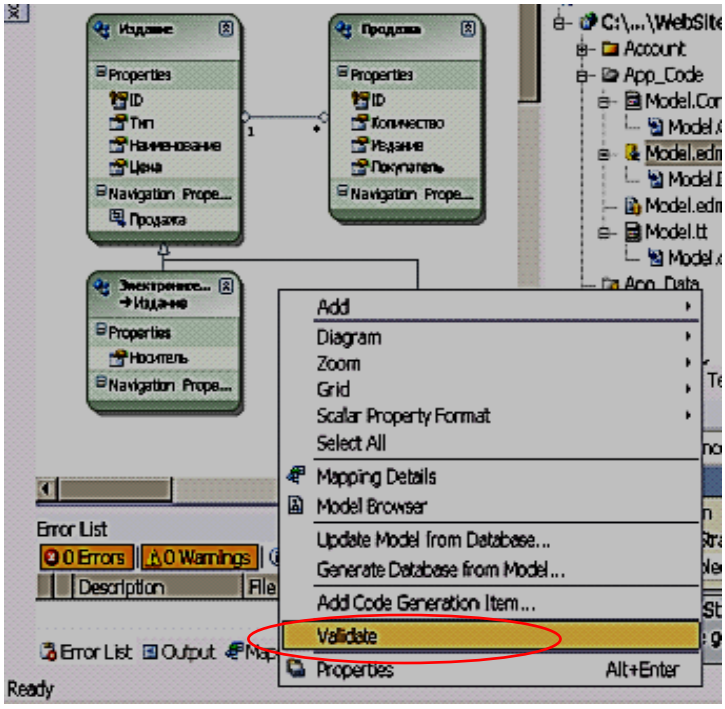

Рис. 29. **Меню операцій над моделлю**

Зрозуміло, що потрібно відображення змін моделі на схему бази даних. Дії, які для цього потрібно виконати, є темою наступної лабораторної роботи (рис. 30).

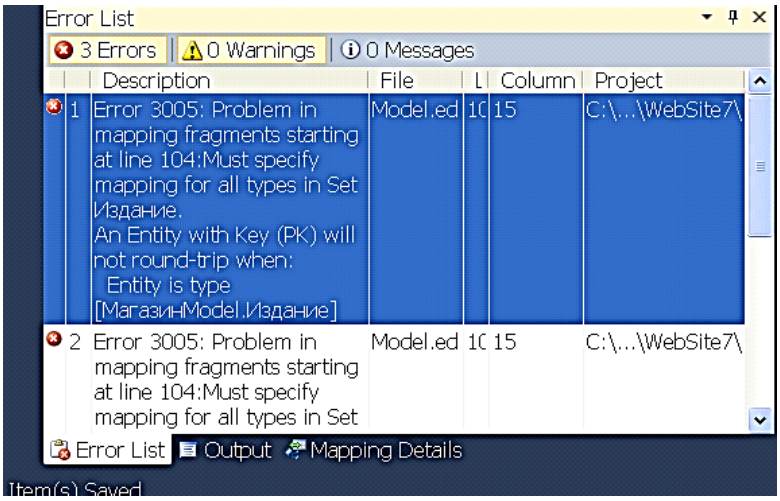

Рис. 30. **Вікно результату валідації**

Контрольні питання

1. Навести основні терміни й визначення графічних мовних засобів ER моделі.
2. Навести основні терміни й визначення текстових мовних засобів моделі даних.
3. Навести основні й похідні від них операції над даними у вигляді зв’язаних об`єктів.
4. Пояснити використання операцій, які найчастіше використовуються у процесі проектування, ведення й розвитку ІС. Навести приклади.
5. Навести переваги та недоліки об`єктної моделі даних.
6. Розкрити зміст поняття "постреляційний підхід до проектування БД".

Лабораторна робота 3. Відображення змін концептуальної моделі на фізичну

Мета роботи. Ознайомити студентів з основними прийомами і технологічними інструментами для організації відображення об`єктної моделі на схему бази даних.

По-перше, потрібно визначити базу даних.

База даних може бути визначена з використанням одного з альтернативних шляхів.

1. Угодою з використанням імені БД за замовчуванням (за іменем класу спадкоємця DbContext).
2. Явною вказівкою імені БД.
3. Явною вказівкою імені рядка підключення.
4. Явною вказівкою безпосередньо рядка підключення.
5. Явною вказівкою існуючого підключення DbConnection.

У даному проекті вже існує підключення, тому раціонально обрати останній спосіб.

1. Завантажити модель Entity Framework (рис. 31).

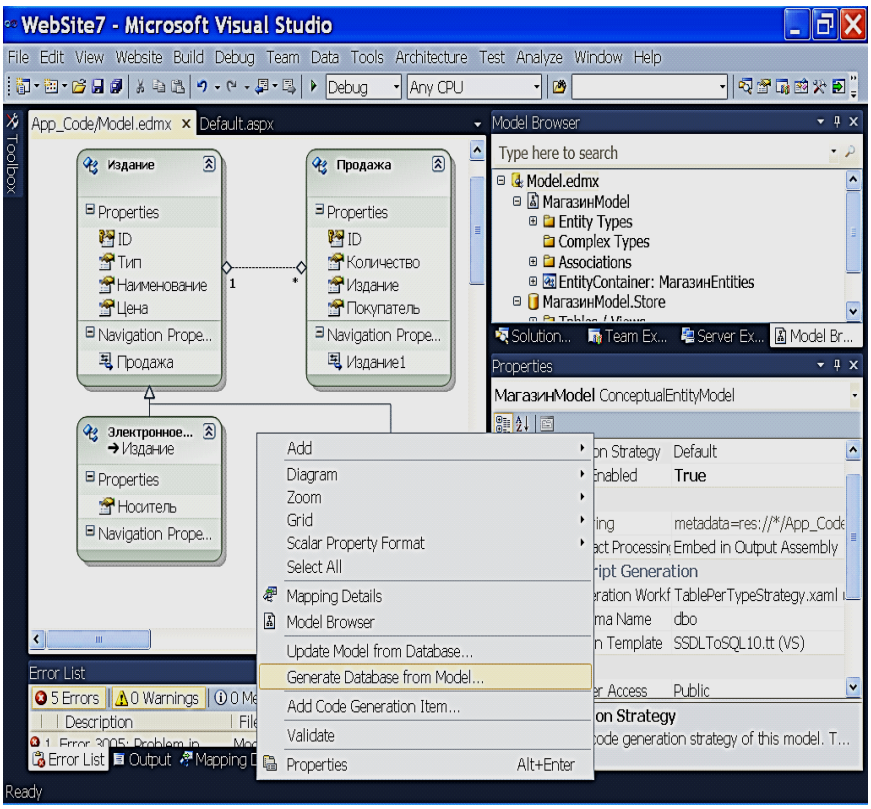

Рис. 31. **Контекстне меню вікна моделі**

2. Згенерувати скрипт – послідовність операторів мови опису даних – створення логічної (рис. 32).

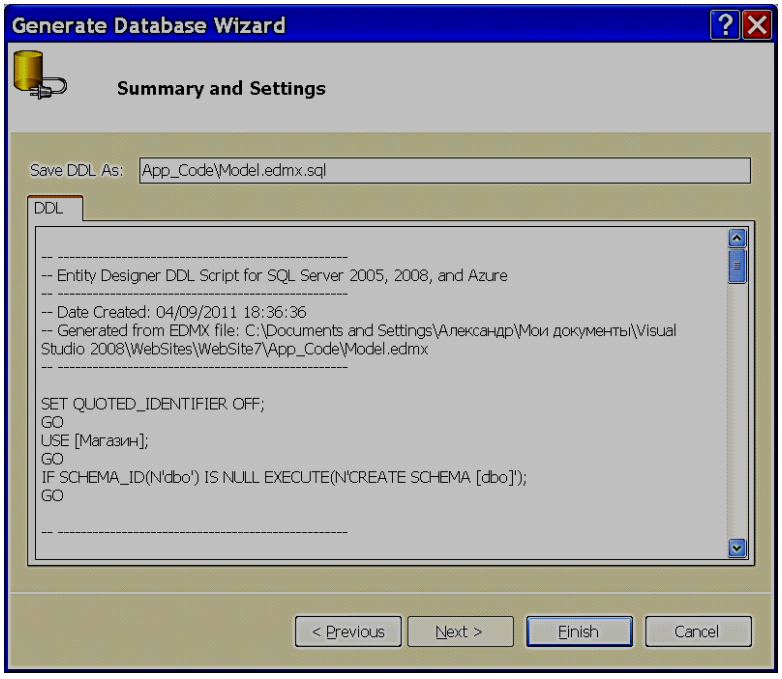

Рис. 32. **Перше вікно майстра валідації схеми бази даних за моделлю**

Розглянемо склад скрипта, що згенерував майстер.

По-перше, скрипт створює базу даних **Магазин**, якщо такої ще не існує.

```
SET QUOTED_IDENTIFIER OFF;
GO
USE [Магазин];
GO
IF SCHEMA_ID(N'dbo') IS NULL EXECUTE(N'CREATE SCHEMA
[dbo]');
GO
```

По-друге, вилучає обмеження зовнішнього ключа, якщо такі існуютть.

```
IF OBJECT_ID(N'[dbo].[FK_Продажа_Издание]', 'F') IS NOT
NULL
    ALTER TABLE [dbo].[Продажа] DROP CONSTRAINT
[FK_Продажа_Издание];
GO
```

Вилучає описи відношень-таблиць бази даних, якщо такі існуютть.

```
IF OBJECT_ID(N'[dbo].[Издание]', 'U') IS NOT NULL
    DROP TABLE [dbo].[Издание];
GO
IF OBJECT_ID(N'[dbo].[Продажа]', 'U') IS NOT NULL
    DROP TABLE [dbo].[Продажа];
GO
```

Заново створює всі таблиці, як ті, що відповідають батьківським сутностям, так і ті, що відповідають дочірнім.

```
CREATE TABLE [dbo].[Издание] (
    [ID] bigint IDENTITY(1,1) NOT NULL,
    [Тип] nvarchar(50)  NULL,
    [Наименование] nvarchar(50)  NULL,
    [Цена] decimal(18,0)  NULL
);
GO
CREATE TABLE [dbo].[Продажа] (
    [ID] bigint IDENTITY(1,1) NOT NULL,
    [Количество] int  NULL,
    [Издание] bigint  NOT NULL,
```

```
    [Покупатель] nvarchar(50)  NULL
);
GO
CREATE TABLE [dbo].[Издание_ЭлектронноеИздание] (
    [Носитель] nvarchar(20)  NOT NULL,
    [ID] bigint  NOT NULL
);
GO
CREATE TABLE [dbo].[Издание_ПечатноеИздание] (
    [Страниц] uniqueidentifier  NOT NULL,
    [ID] bigint  NOT NULL
);
GO
```

Заново створює всі обмеження первинного ключа, як ті, що відповідають батьківським сутностям, так і ті, що відповідають дочірнім сутностям.

```
ALTER TABLE [dbo].[Издание]
ADD CONSTRAINT [PK_Издание]
    PRIMARY KEY CLUSTERED ([ID] ASC);
GO
ALTER TABLE [dbo].[Продажа]
ADD CONSTRAINT [PK_Продажа]
    PRIMARY KEY CLUSTERED ([ID] ASC);
GO
ALTER TABLE [dbo].[Издание_ЭлектронноеИздание]
ADD CONSTRAINT [PK_Издание_ЭлектронноеИздание]
    PRIMARY KEY CLUSTERED ([ID] ASC);
GO
ALTER TABLE [dbo].[Издание_ПечатноеИздание]
ADD CONSTRAINT [PK_Издание_ПечатноеИздание]
    PRIMARY KEY CLUSTERED ([ID] ASC);
GO
```

Заново створює всі обмеження зовнішнього ключа, як ті, що відповідають батьківським сутностям, так і ті, що відповідають дочірнім та потрібні для цього індекси.

```
ALTER TABLE [dbo].[Продажа]
ADD CONSTRAINT [FK_Продажа_Издание]
    FOREIGN KEY ([Издание])
    REFERENCES [dbo].[Издание]
        ([ID])
    ON DELETE NO ACTION ON UPDATE NO ACTION;
CREATE INDEX [IX_FK_Продажа_Издание]
ON [dbo].[Продажа]
    ([Издание]);
GO
ALTER TABLE [dbo].[Издание_ЭлектронноеИздание]
ADD CONSTRAINT [FK_ЭлектронноеИздание_inherits_Издание]
    FOREIGN KEY ([ID])
    REFERENCES [dbo].[Издание]
        ([ID])
    ON DELETE NO ACTION ON UPDATE NO ACTION;
GO
ALTER TABLE [dbo].[Издание_ПечатноеИздание]
ADD CONSTRAINT [FK_ПечатноеИздание_inherits_Издание]
    FOREIGN KEY ([ID])
    REFERENCES [dbo].[Издание]
        ([ID])
    ON DELETE NO ACTION ON UPDATE NO ACTION;
GO
```

3. На запитання, чи потрібно оновити існуючий опис бази даних та відображення, потрібно відповісти ствердно (рис. 33).

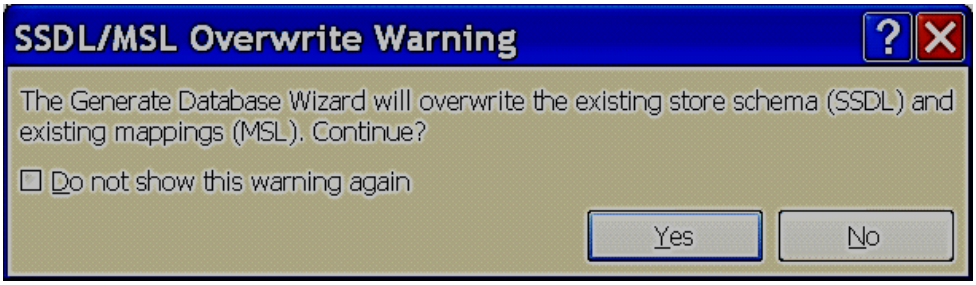

Рис. 33. **Діалог відновлення відображення**

4. Для внесення змін до схеми бази даних необхідно виконати скрипт, який було згененеровано за допомогою контекстного меню вікна SQL скрипта моделі – команда Виконати SQL (рис. 34).

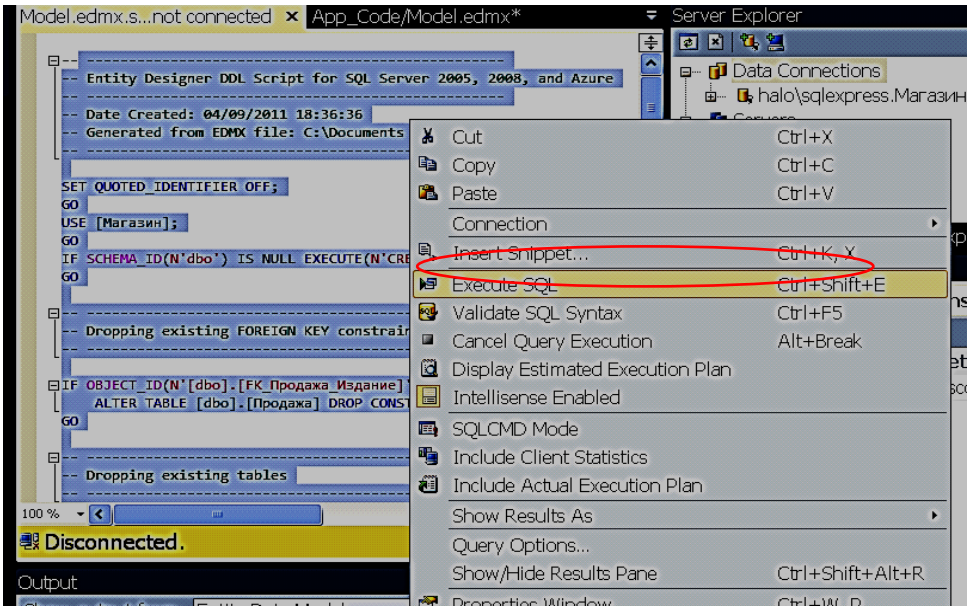

Рис. 34.**Контекстне меню вікна SQL скрипта моделі – команда Виконати SQL**

5. Для перевірки відповідності концептуальної моделі до логічної після внесення змін до схеми бази даних необхідно відновити модель за побудованим відображенням за допомогою контекстного меню тексту моделі на мові XML-команда Відновити (рис. 35).

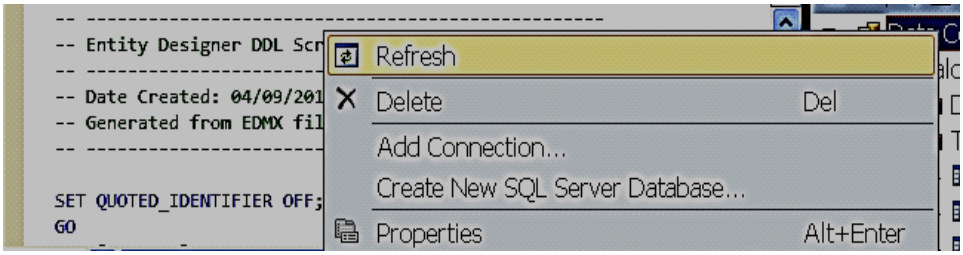

Рис. 35. **Контекстне меню вікна текстового подання моделі**

6. Для створення додатка з можливістю відображення елементів створеної моделі за допомогою Інтернету, додамо до середовища Visual Studio новий елемент (рис. 36) – веб-сайт (рис. 37, 38).

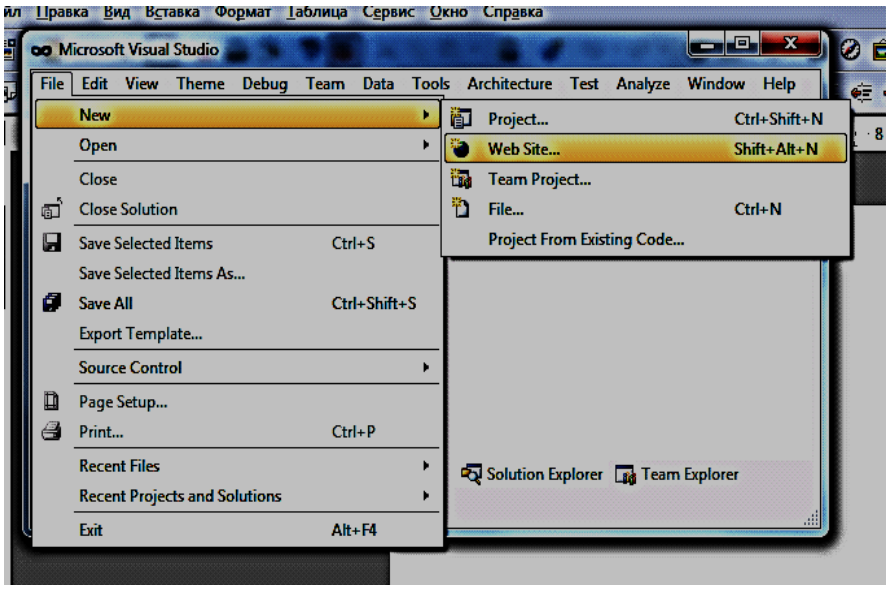

Рис. 36. **Меню додавання сайта**

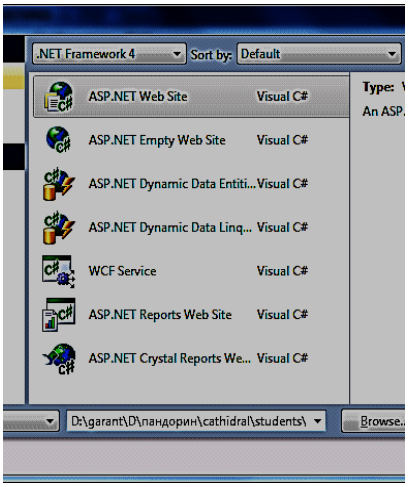

Рис. 37. **Вікно вибору типу веб-сайта**

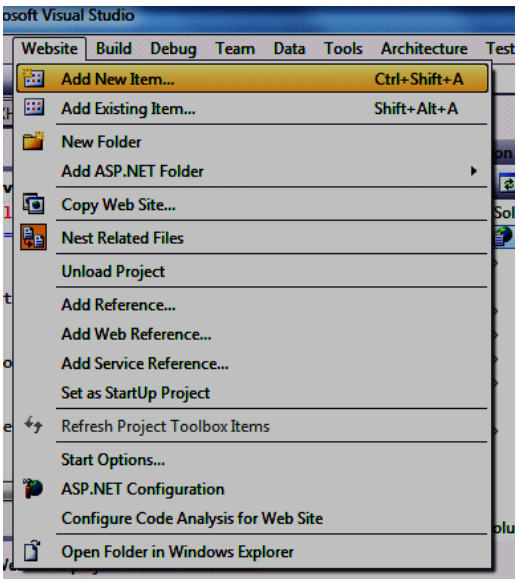

Рис. 38. **Контекстне меню створення веб-сайта**

7. Для використання моделі сутність-зв'язок додамо до веб-сайта модель Entity Data Model (рис. 39) та згодимося, відповідаючи на питання про можливість додання файла до папки App_Code (рис. 40).

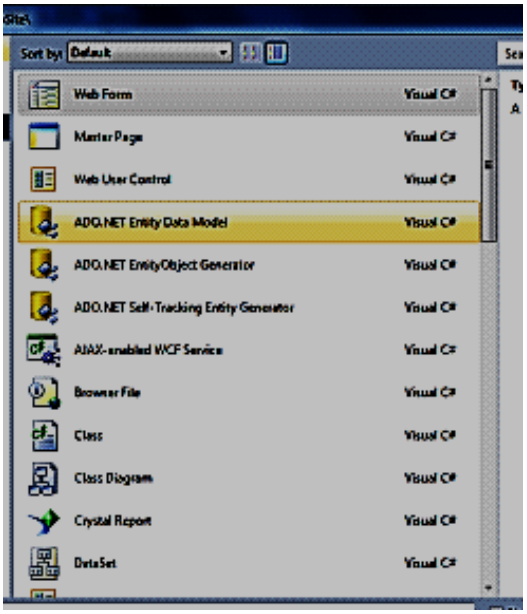

Рис. 39. **Діалог додавання моделі**

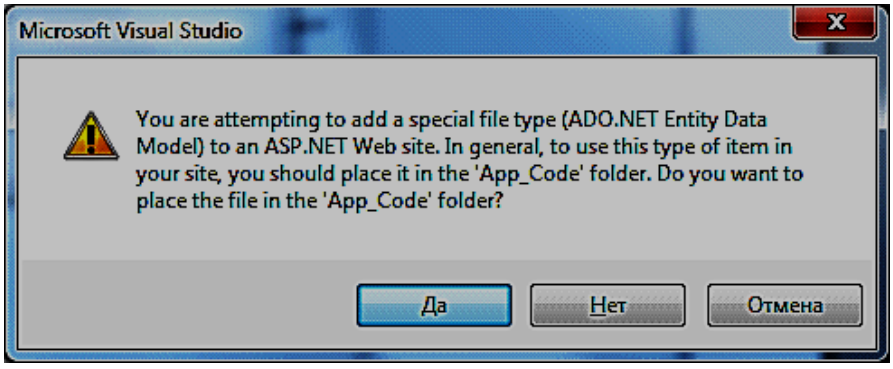

Рис. 40. **Запит про можливість додання файла до папки App_Code**

8. Побудуємо модель за схемою бази даних (рис. 41). Для цього у майстрі створення моделі визначимо параметри з'єднання з сервером (рис. 42) та оберемо таблиці бази даних, до яких потрібно побудувати модель (рис. 43 – 45).

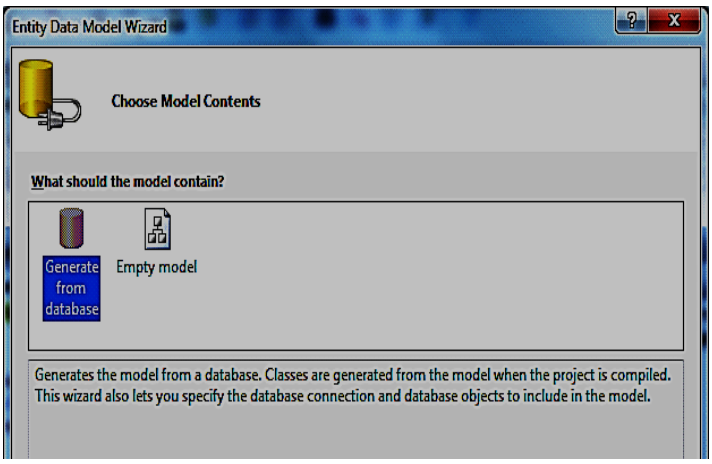

Рис. 41. **Майстер створення моделі – вибір типу**

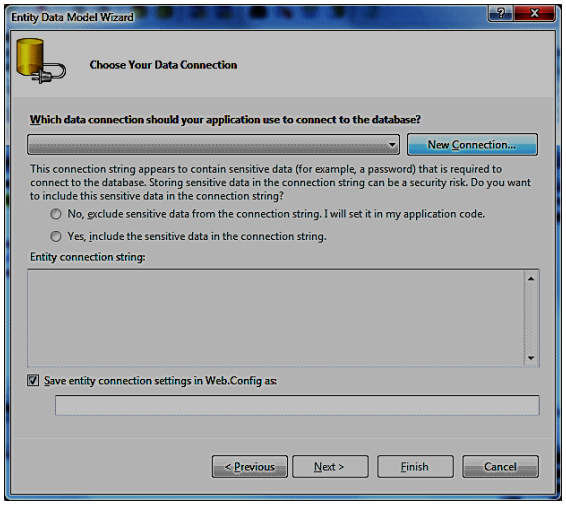

Рис. 42. **Майстер створення моделі – вибір джерела даних**

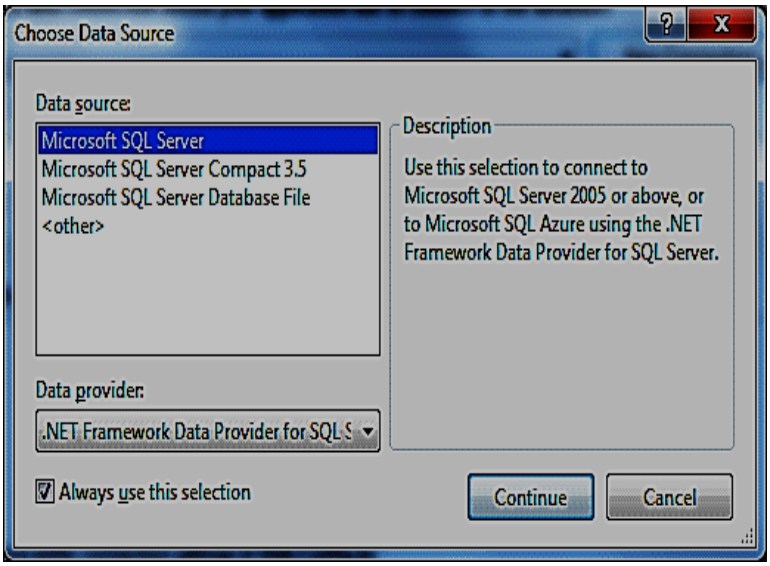

Рис. 43. **Майстер створення моделі – вибір строки з'єднання**

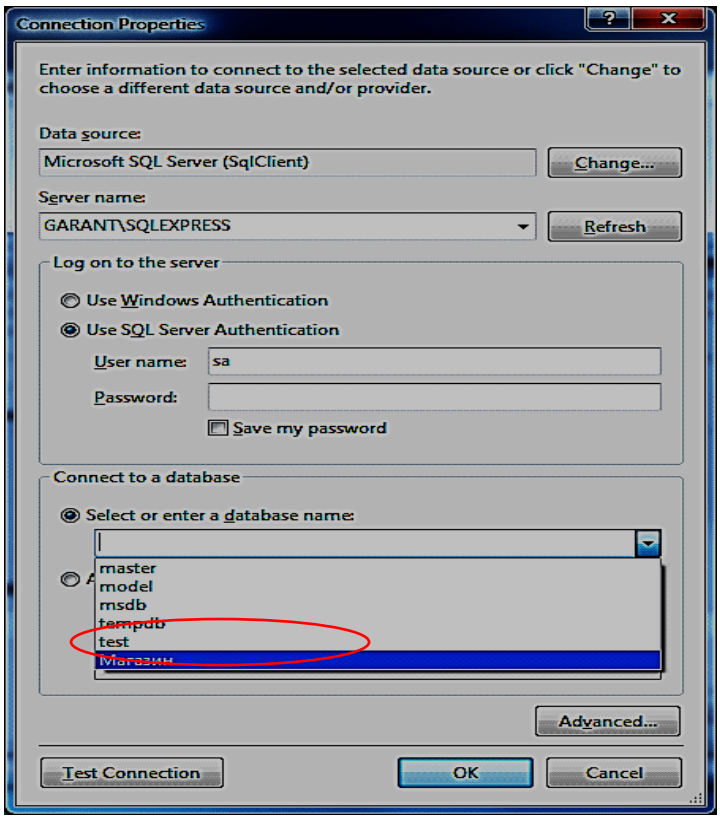

Рис. 44. **Майстер створення моделі – вибір бази даних**

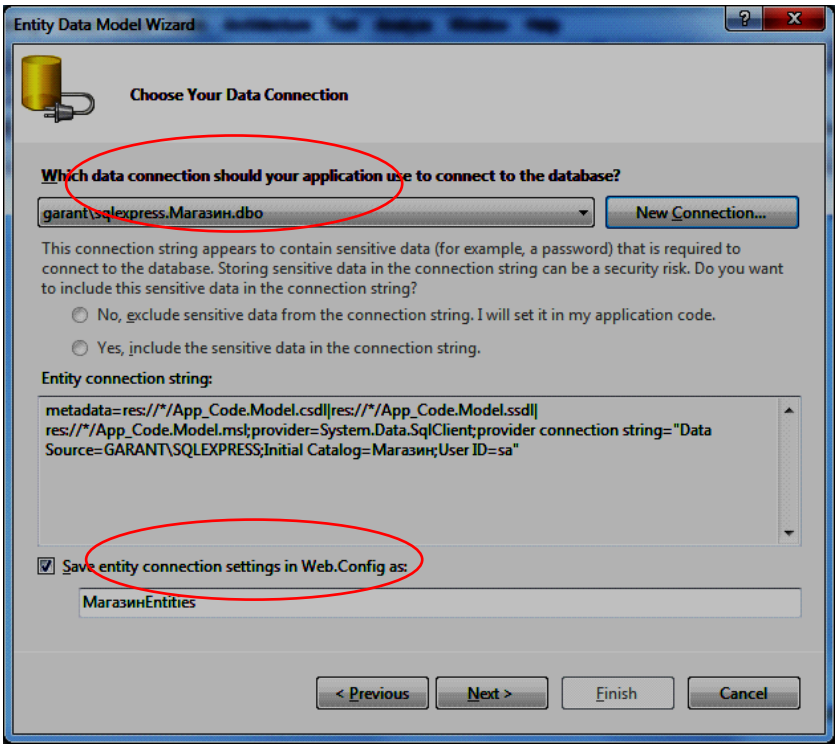

Рис. 45. **Майстер створення моделі – підтвердження строки з'єднання**

9. Чекаємо побудови переліку елементів бази даних (рис. 46) та обираємо елементи, які потрібні для побудови моделі (рис. 47).

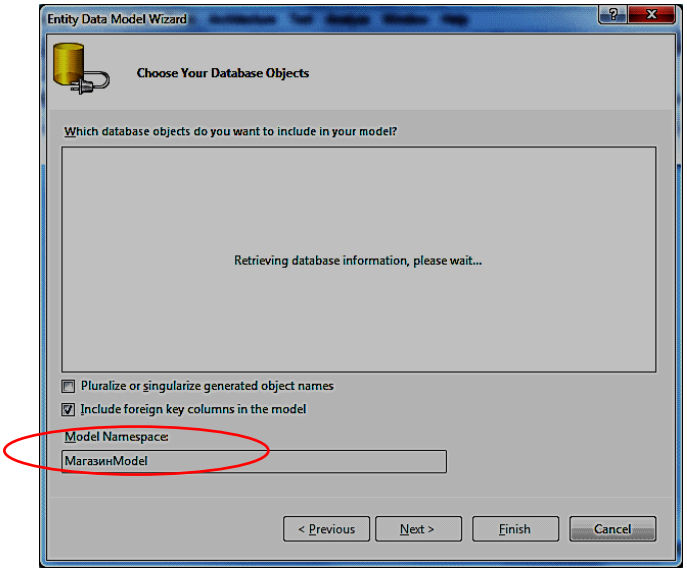

Рис. 46. **Майстер створення моделі – побудова переліку об’єктів**

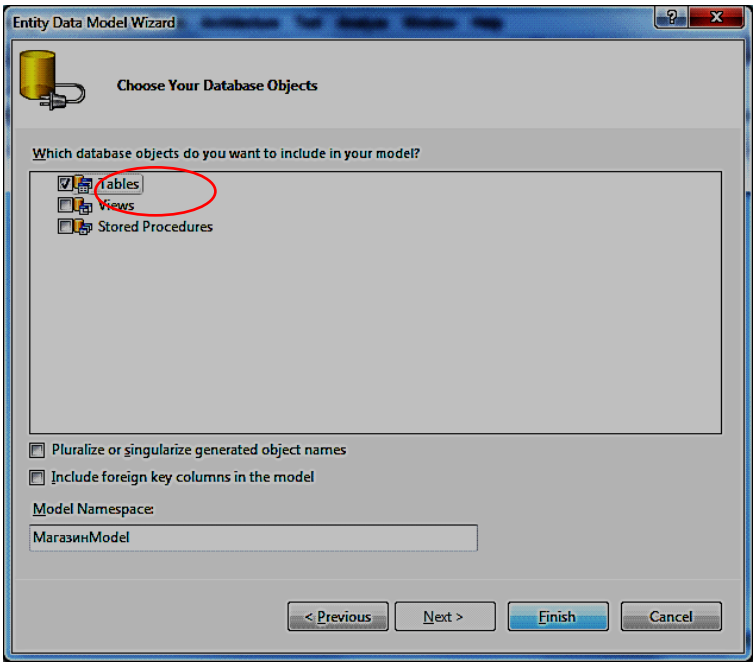

Рис. 47. **Майстер створення моделі – вікно вибору об’єктів**

10. Модель побудовано. У вікні "Подробиці відображення" – "Mapping Details" можливо переглянути, як саме відобразилися один на одного елементи бази даних і моделі та відредагувати відображення таблиці (рис. 48) та зв’язки (рис. 49).

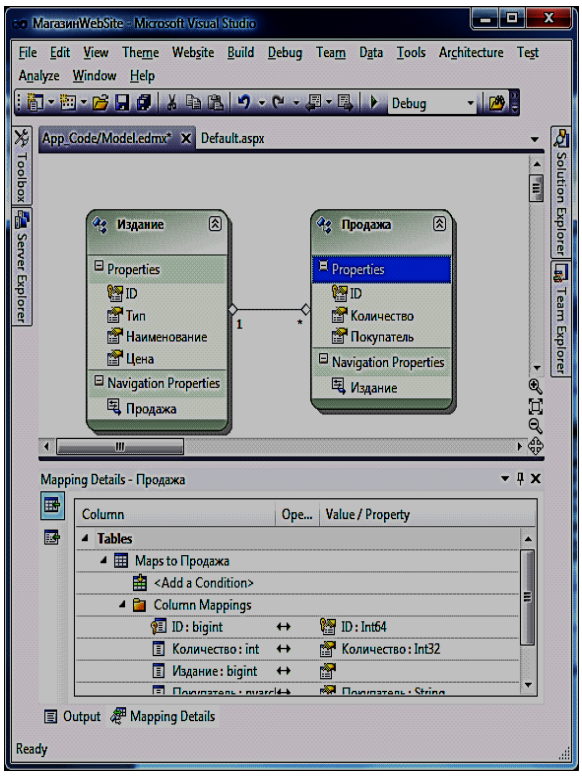

Рис. 48. **Модель та подробиці відображення атрибутів сутностей на опис стовбців таблиці**

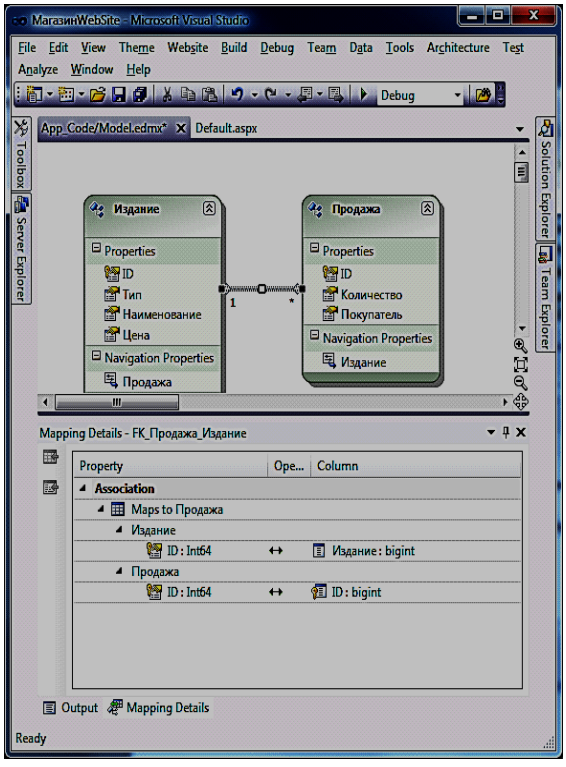

Рис. 49. **Модель та подробиці відображення зв’язків таблиць та зв’язків сутностей**

Контрольні питання

1. Навести основні терміни й визначення текстові мовні засоби моделі даних.

2. Навести основні й похідні від них операції над даними у вигляді зв’язаних об`єктів.

3. Пояснити використання операцій, які найчастіше використовуються в процесі проектування, ведення й розвитку ІС. Навести приклади.

Лабораторна робота 4. Створення об’єкта поповнення БД контентом з папки Outlook

Мета роботи. Ознайомити студентів з основними прийомами і технологічними інструментами для організації відображення стану об`єктів операційної системи як об’єктів бази даних.

Відобразимо засобами запитів LINQ до папки контактів Outlook стан даних контактів (рис. 50) на поверхні форми.

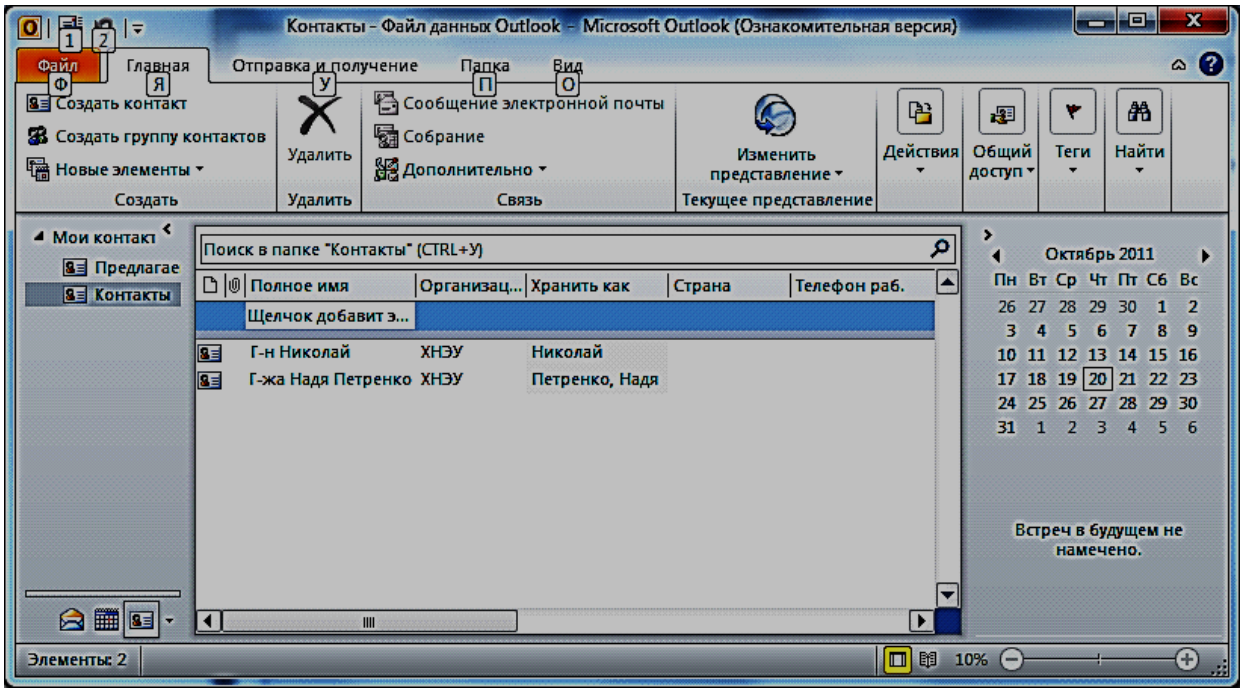

Рис. 50. **Стан папки контактів**

При побудові відображення використаємо провайдер, що описано Stefan Cruysberghs на сторінці http://scipbe. wordpress. com/2007/12/27/ articles-about-linq-and-outlook-and-onenote/.

1. Побудувати додаток (рис. 51).

2. За допомогою контекстного меню "Посилання" – "References" (рис. 52) додати посилання на COM сервер Outlook (рис. 53).

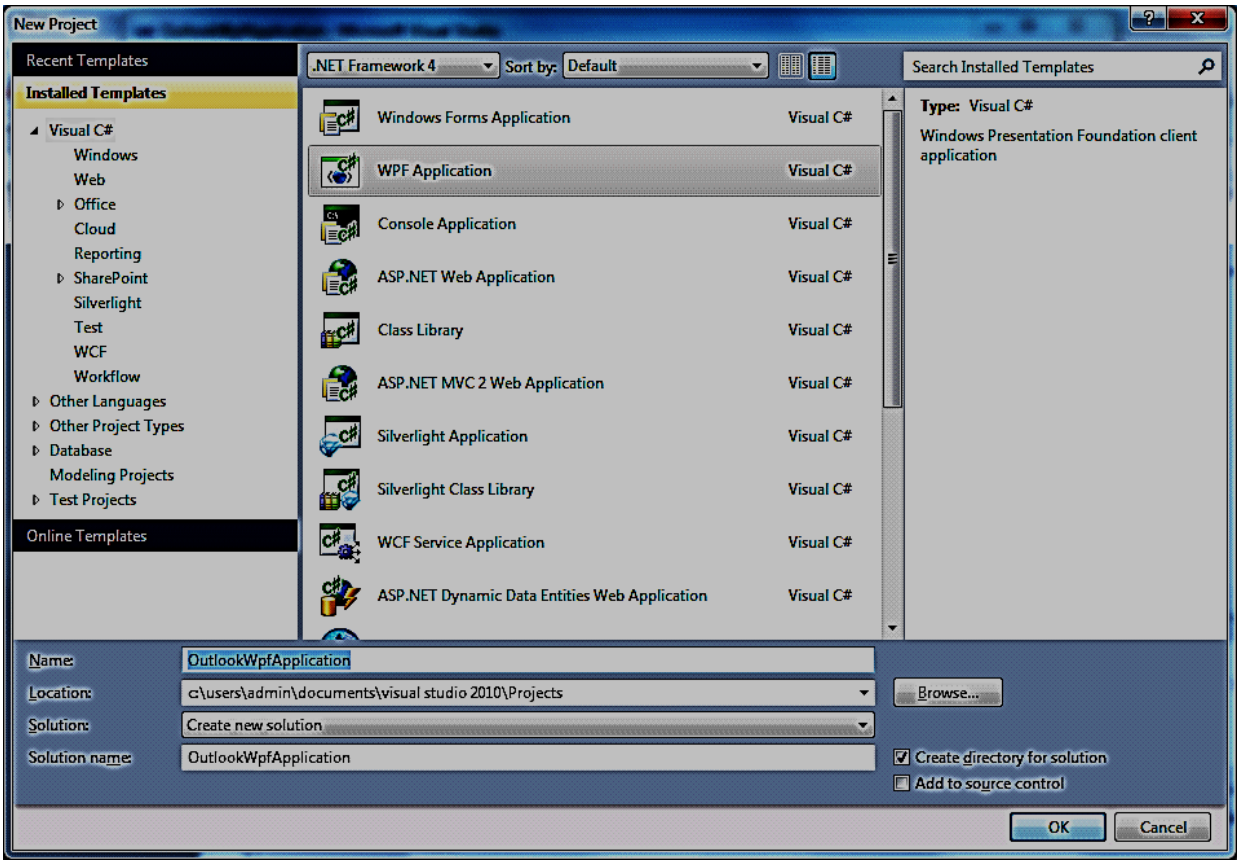

Рис. 51. **Вікно додання проекту**

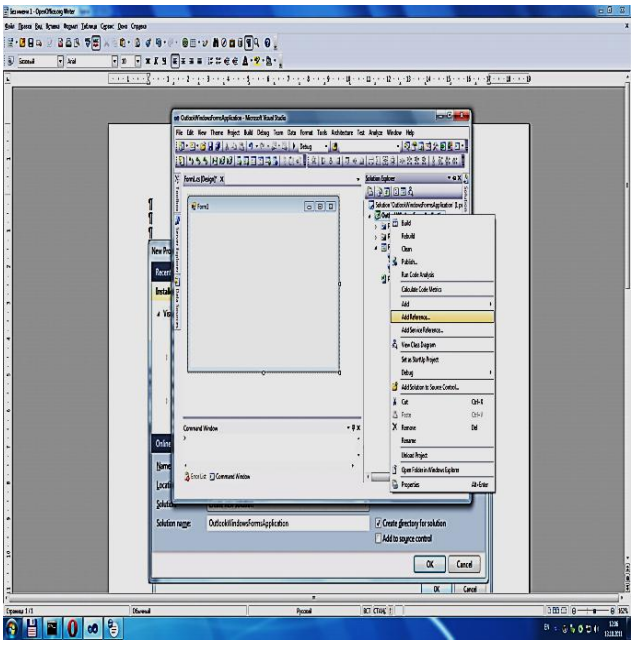

Рис. 52. **Контекстне меню "Посилання"**

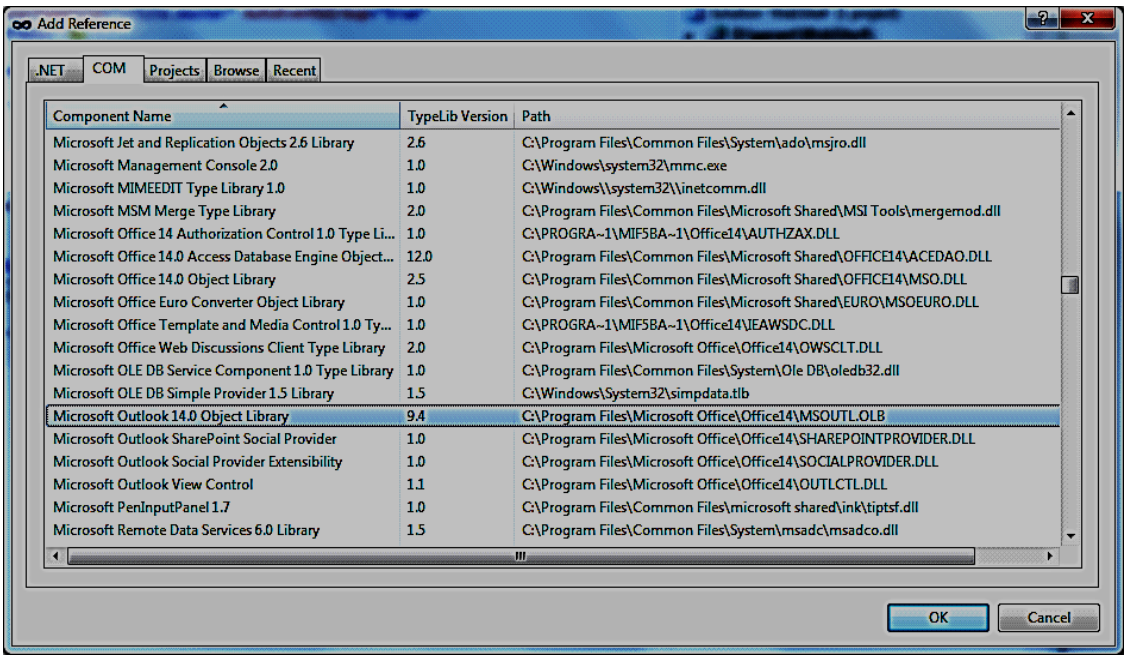

Рис. 53. **Вікно "References" закладка COM**

3. У вікні коду головного вікна додатка додати посилання на потрібні простори імен (що виділено на рис. 54).

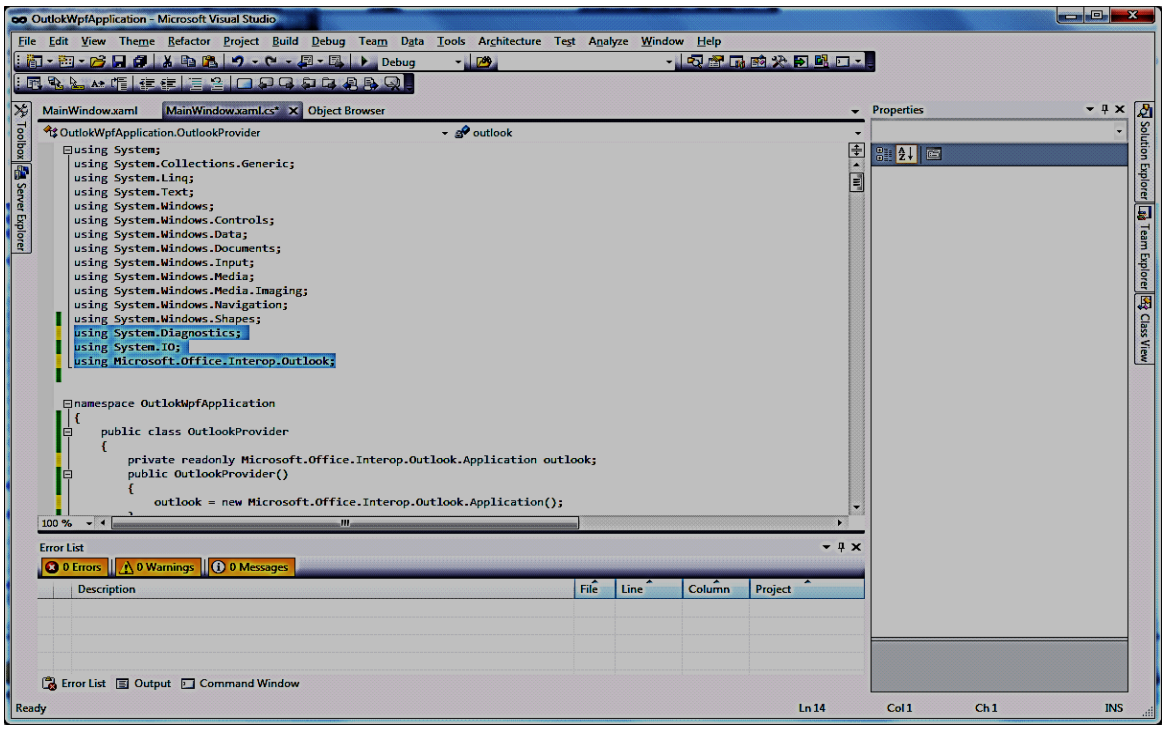

Рис. 54. **Вікно коду головного вікна додатка**

4. У вікні коду головного вікна додатка додати посилання на потрібні простори імен (що виділено на рис. 54).

5. Додати компонент DataGrid за допомогою перетягування з панелі інструментів (рис. 55) до форми головного вікна (рис. 56).

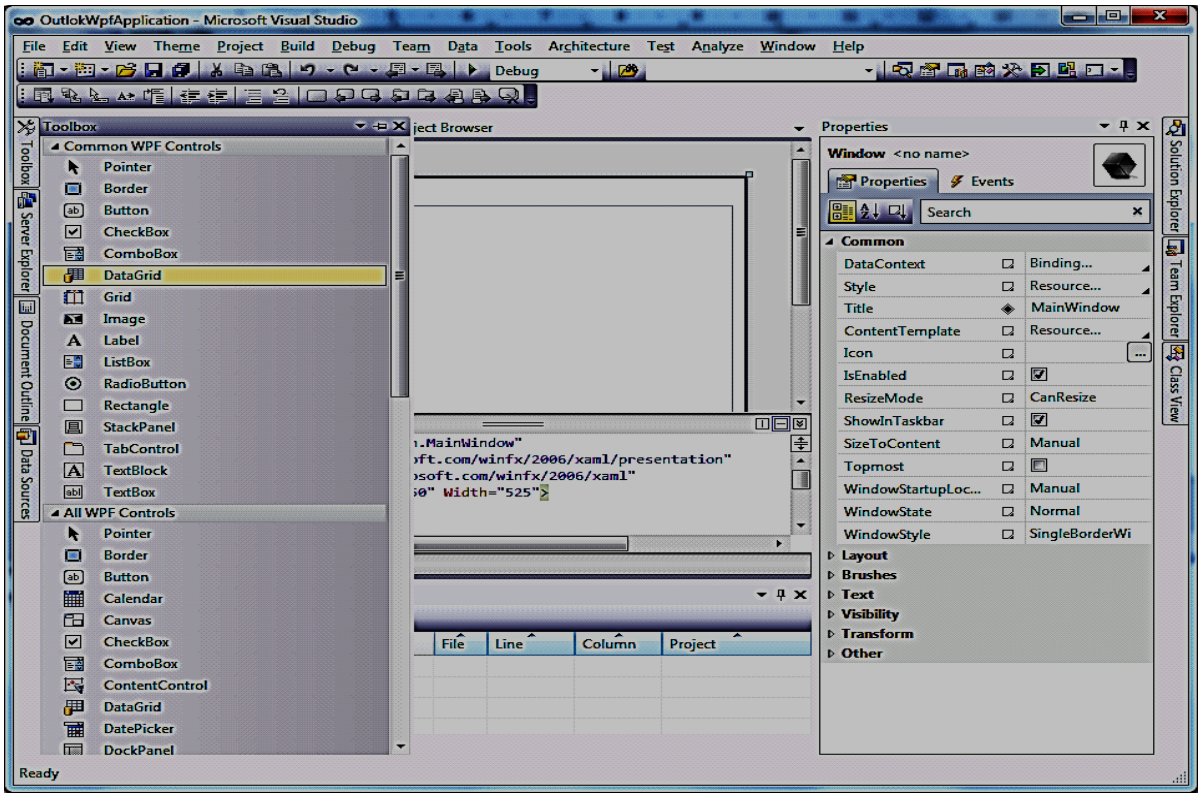

Рис. 55. **Панель інструментів та головна форма**

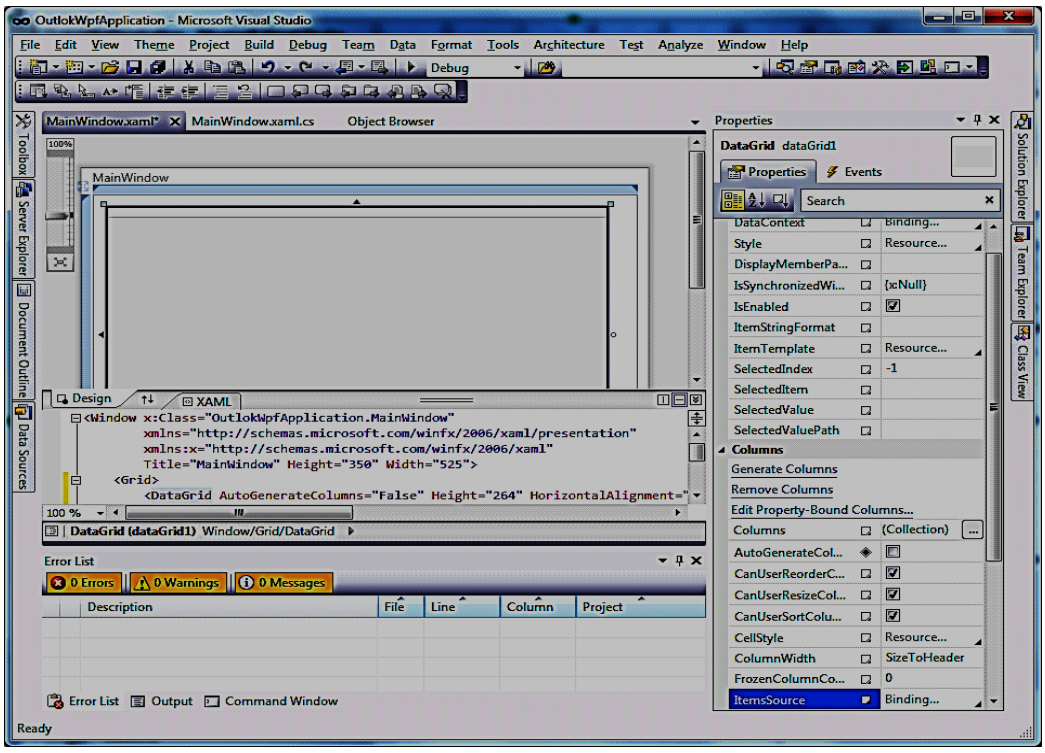

Рис. 56. **Вікно головної форми після додання компоненту**

6. Перейменувати компонент, що перетягнуто, у **`contactsDataGrid`**.

7. Виключити значення властивості **`AutoGenerateColumns`** компоненту **`contactsDataGrid`** (рис. 57).

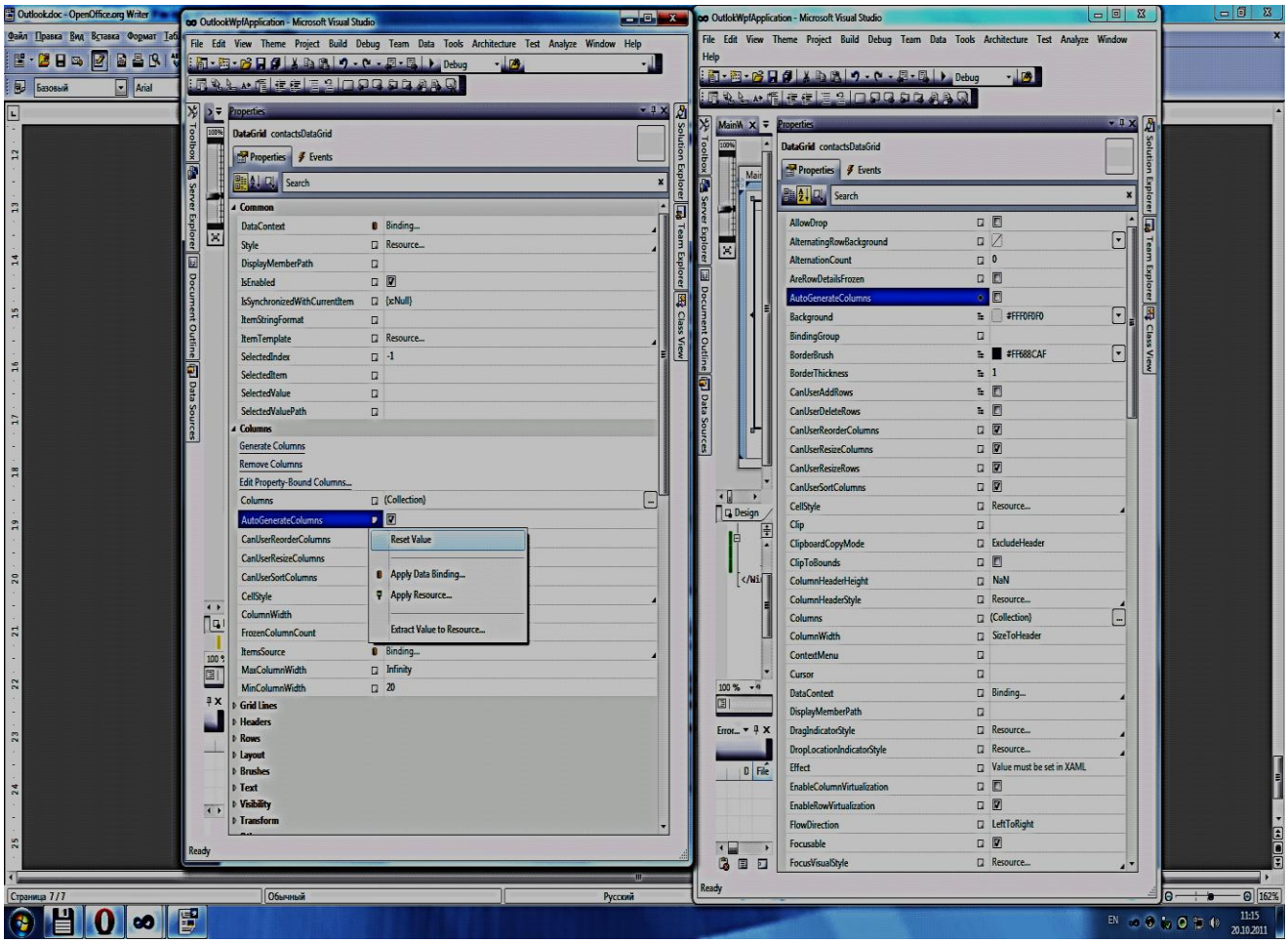

Рис. 57. **Вікно головної форми після додання компоненту**

8. У вікні коду головного вікна додатка додати текст провайдеру – клас **`OutlookProvider`**.

```
namespace OutlokWpfApplication
{
    public class OutlookProvider
    {
        private readonly
Microsoft.Office.Interop.Outlook.Application outlook;
        public OutlookProvider()
        { outlook = new
Microsoft.Office.Interop.Outlook.Application(); }
        public
Microsoft.Office.Interop.Outlook.Application Outlook
```

```
            { get { return outlook; } }
            public IEnumerable<Folder> Folders
            { get { return
GetAllFolders(outlook.ActiveExplorer().Session.Folders.Of
Type<Folder>()); } }
            private static IEnumerable<Folder>
GetAllFolders(IEnumerable<Folder> folders)
            {
                foreach (var folder in folders)
                {
                    foreach (var subfolder in
GetAllFolders(folder.Folders.OfType<Folder>()))
                    { yield return subfolder; }
                    yield return folder;
                }
            }
            public IEnumerable<T> GetItems<T>(string
path)
            {
                if (path.Substring(0, 2) != @"\\")
                { path = @"\\" + path; }
                var folder = Folders.FirstOrDefault(f =>
f.FolderPath == path);
                if (folder == null)
                { throw new ArgumentNullException("path",
"Path to Outlook folder does not exists"); }
                if ((folder.DefaultItemType ==
OlItemType.olAppointmentItem) && (typeof(T) ==
typeof(AppointmentItem)))
                    return folder.Items.OfType<T>();
                if ((folder.DefaultItemType ==
OlItemType.olContactItem) && (typeof(T) ==
typeof(ContactItem)))
                    return folder.Items.OfType<T>();
                if ((folder.DefaultItemType ==
OlItemType.olJournalItem) && (typeof(T) ==
typeof(JournalItem)))
                    return folder.Items.OfType<T>();
                if ((folder.DefaultItemType ==
OlItemType.olMailItem) && (typeof(T) ==
typeof(MailItem)))
                    return folder.Items.OfType<T>();
                if ((folder.DefaultItemType ==
OlItemType.olNoteItem) && (typeof(T) ==
typeof(NoteItem)))
```

```
                    return folder.Items.OfType<T>();
                if ((folder.DefaultItemType ==
OlItemType.olPostItem) && (typeof(T) ==
typeof(PostItem)))
                    return folder.Items.OfType<T>();
                if ((folder.DefaultItemType ==
OlItemType.olTaskItem) && (typeof(T) ==
typeof(TaskItem)))
                    return folder.Items.OfType<T>();
                return null;
            }
            public IEnumerable<ContactItem> ContactItems
            {
                get
                {

Microsoft.Office.Interop.Outlook.MAPIFolder folder =

outlook.GetNamespace("MAPI").GetDefaultFolder(OlDefaultFo
lders.olFolderContacts);
                    return
folder.Items.OfType<ContactItem>();
                }
            }
            public IEnumerable<AppointmentItem>
CalendarItems
            {
                get
                {

Microsoft.Office.Interop.Outlook.MAPIFolder folder =

outlook.GetNamespace("MAPI").GetDefaultFolder(OlDefaultFo
lders.olFolderCalendar);
                    return
folder.Items.OfType<AppointmentItem>();
                }
            }
            public IEnumerable<MailItem> InboxItems
            {
                get
                {
    Microsoft.Office.Interop.Outlook.MAPIFolder folder =
```

```
    outlook.GetNamespace("MAPI").GetDefaultFolder(OlDefau
ltFolders.olFolderInbox);
                    return
folder.Items.OfType<MailItem>();
                }
            }
            public IEnumerable<MailItem> SentMailItems
            {
                get
                {

Microsoft.Office.Interop.Outlook.MAPIFolder folder =

outlook.GetNamespace("MAPI").GetDefaultFolder(OlDefaultFo
lders.olFolderSentMail);
                    return
folder.Items.OfType<MailItem>();
                }
            }
            public IEnumerable<NoteItem> NoteItems
            {
                get
                {

Microsoft.Office.Interop.Outlook.MAPIFolder folder =

outlook.GetNamespace("MAPI").GetDefaultFolder(OlDefaultFo
lders.olFolderNotes);
                    return
folder.Items.OfType<NoteItem>();
                }
            }
            public IEnumerable<TaskItem> TaskItems
            {
                get
                {

Microsoft.Office.Interop.Outlook.MAPIFolder folder =

outlook.GetNamespace("MAPI").GetDefaultFolder(OlDefaultFo
lders.olFolderTasks);
                    return
folder.Items.OfType<TaskItem>();
                }
            }
```

```
        }
        /// <summary>

    public MainWindow()
            {
                InitializeComponent();
                OutlookProvider outlookProvider;
                outlookProvider = new OutlookProvider();
                contactsDataGrid.ItemsSource (from
contact in outlookProvider.ContactItems
                                                select new
{ contact.CompanyName, contact.FullName }).ToList();
            }
```

9. Запустити додаток на виконання (рис. 58).

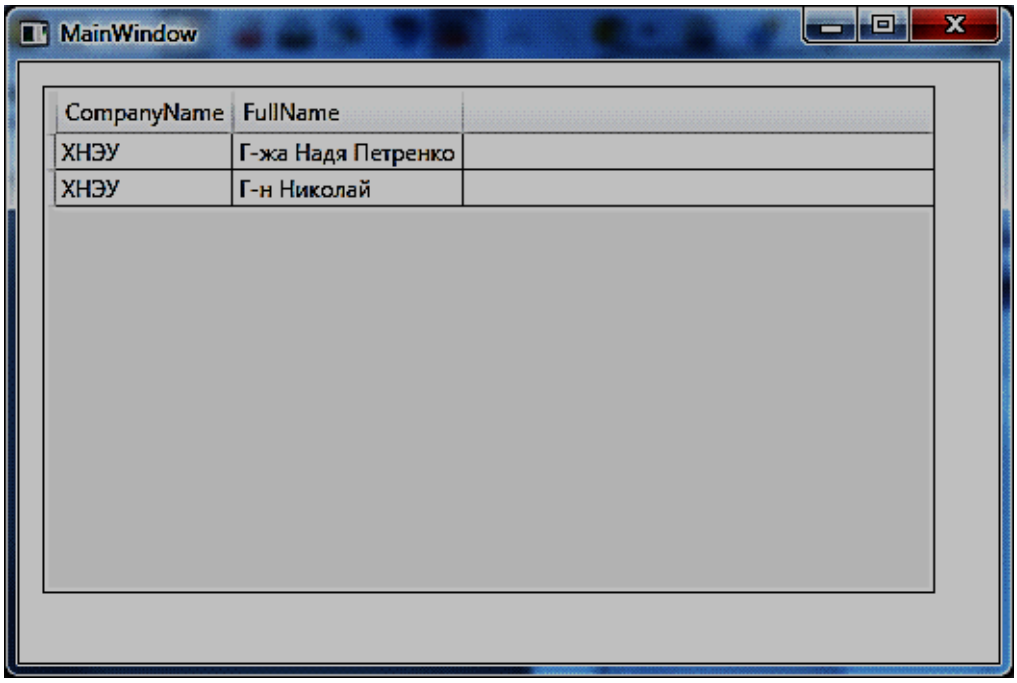

Рис. 58. **Тестування створенного додатка**

Контрольні питання

1. Навести приклади засобів представлення запрошуваних даних у формі XML.
2. Навести приклади засобів представлення запрошуваних даних у формі баз даних ADO.NET.
3. Навести приклади засобів представлення запрошуваних даних у формі баз даних об'єктів.

4. Навести приклади використання лямбда-виразів.
5. Навести приклади використання методів розширення.
6. Навести приклади використання анонімних типів.
7. Навести приклади використання локальних змінних.
8. Навести приклади використання ініціалізації об'єктів.
9. Навести приклади використання вирази запитів, що неявно типізуються.

Лабораторна робота 5. Створення об’єкта поповнення БД контентом з папки робочої станції

Мета роботи. Ознайомити студентів з основними прийомами для оперування мультимедійною інформацію за допомогою бази даних.

- Створюємо веб-додаток.
- Створюємо базу даних Album MS SQL Server .
- Створюємо таблицю у базі даних з ім’ям Album такої структури (рис. 59).

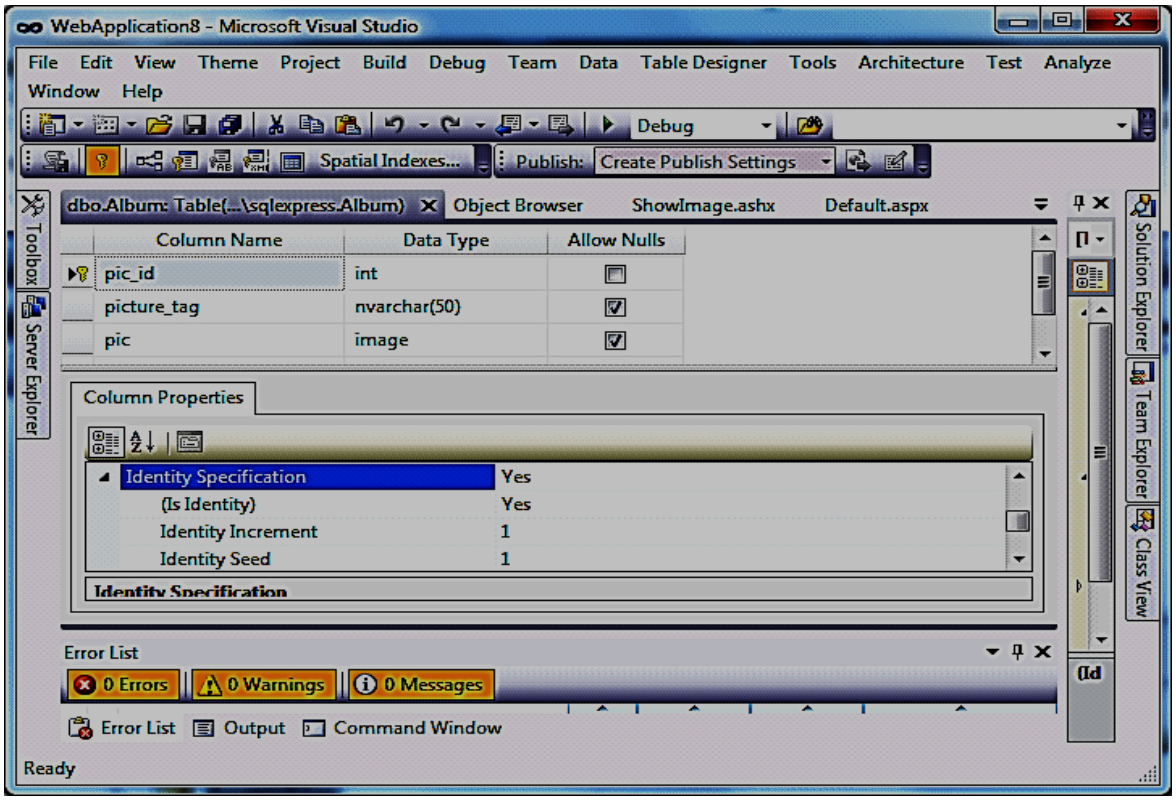

Рис. 59. **Опис таблиці Album бази даних Album**

- До головної форми додадемо наступні компоненти з панелі інструментів (рис. 60).

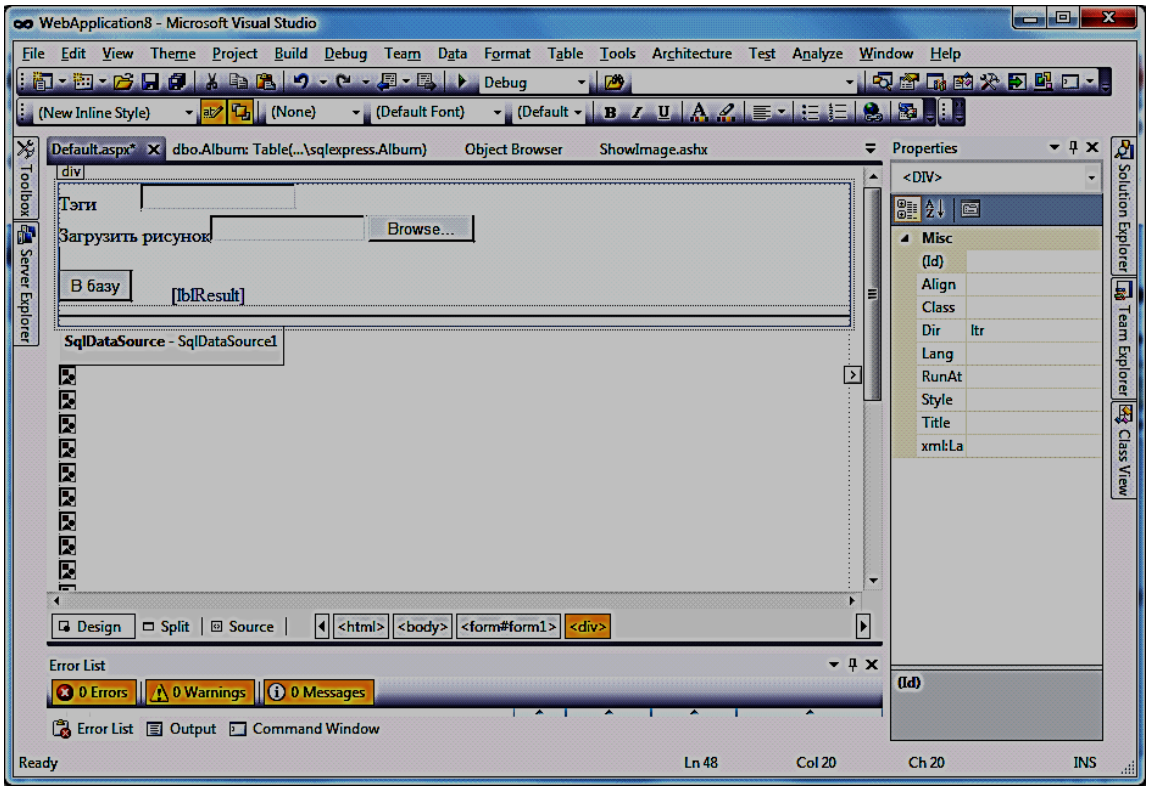

Рис. 60. **Головна сторінка додатка**

Значення основних властивостей зрозуміло з такого опису сторінки.

```
<%@ Page Language="C#" AutoEventWireup="true"
CodeFile="Default.aspx.cs" Inherits="_Default" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>Untitled Page</title>
</head>
<body>
    <form id="form1" runat="server">
    <div>
```

```
Тэги        
        <asp:TextBox                              ID="txtTags"
runat="server"></asp:TextBox>
        <br />
        <asp:Label      ID="lblImage"      runat="server"
Text="Загрузить рисунок"></asp:Label>
        <asp:FileUpload ID="imgUpload" runat="server" />
        <br />
        <br />
        <asp:Button     ID="btnSubmit"     runat="server"
onclick="btnSubmit_Click"
            Text="В базу" />

        <asp:Label
ID="lblResult"                            runat="server"
ForeColor="#0066FF"></asp:Label>
    <br />
    <hr />
    </div>
    <asp:SqlDataSource ID="SqlDataSource1" runat="server"
        ConnectionString="Data
Source=GARANT\SQLEXPRESS;Initial       Catalog=Album;User
ID=sa"
        SelectCommand="SELECT    [pic],    [picture_tag],
[pic_id] FROM [Album]"

ProviderName="System.Data.SqlClient"></asp:SqlDataSource>
    <asp:ListView      ID="ListView1"      runat="server"
DataSourceID="SqlDataSource1">
        <LayoutTemplate>
            <asp:Placeholder
            id="groupPlaceholder"
            runat="server" />
        </LayoutTemplate>
        <GroupTemplate>
            <div>
```

```
            <asp:Placeholder
            id="itemPlaceholder"
            runat="server" />
            </div>
        </GroupTemplate>
        <ItemTemplate>
             <asp:Image
            id="picAlbum"                    runat="server"
AlternateText='<% #Eval("picture_tag") %>'
           ImageUrl='<%#      "ShowImage.ashx?id="      +
Eval("pic_id") %>' />
        </ItemTemplate>

    </asp:ListView>
    </form>
</body>
</html>
```

• У коді сторінки за допомогою подвійного клацання по поверхні кнопки створюємо обробник події **btnSubmit_Click** та заповнюємо його таким текстом.

```
protected void btnSubmit_Click(object sender, EventArgs
e)
    {
        SqlConnection connection = null;
        try
        {
            FileUpload img = (FileUpload)imgUpload;
            Byte[] imgByte = null;
            if (img.HasFile && img.PostedFile != null)
            {
                //To create a PostedFile
                HttpPostedFile File =
imgUpload.PostedFile;
                //Create byte Array with file len
                imgByte = new Byte[File.ContentLength];
```

```
                //force the control to load data in array
                File.InputStream.Read(imgByte,          0,
File.ContentLength);
            }
            // Insert the picture tag and image into db
            string                  conn                  =
ConfigurationManager.ConnectionStrings["albumConnString"]
.ConnectionString;
            connection = new SqlConnection(conn);

            connection.Open();
            string      sql      =      "INSERT      INTO
Album(picture_tag,pic)      VALUES(@tag,@pic)      SELECT
@@IDENTITY AS 'Identity'";
            SqlCommand    cmd    =    new    SqlCommand(sql,
connection);
            cmd.Parameters.AddWithValue("@tag",
txtTags.Text);
            cmd.Parameters.AddWithValue("@pic", imgByte);
            int                     id                    =
Convert.ToInt32(cmd.ExecuteScalar());
            lblResult.Text = String.Format("Picture ID is
{0}", id);
            ListView1.DataBind();
        }
        catch
        {
            lblResult.Text = "There was an error";
        }
        finally
        {
            connection.Close();
        }
    }
```

- Створюємо хендлер класу ShowImage (рис. 61).

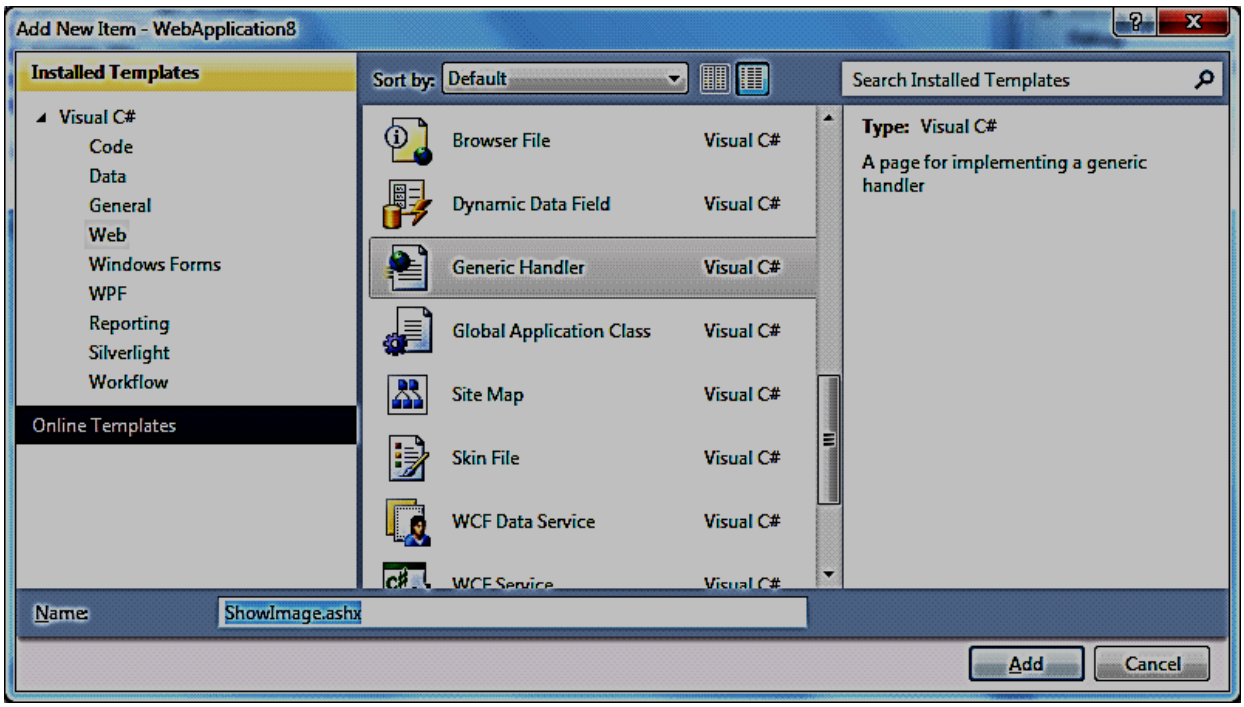

Рис. 61. **Створення хендлера**

- Заповнюємо хендлер таким кодом.

```
<%@ WebHandler Language="C#" Class="ShowImage" %>

using System;
using System.Configuration;
using System.Web;
using System.IO;
using System.Data;
using System.Data.SqlClient;

public class ShowImage : IHttpHandler
{
    public void ProcessRequest(HttpContext context)
    {
        Int32 picid;
        if (context.Request.QueryString["id"] != null)
            picid                                          =
Convert.ToInt32(context.Request.QueryString["id"]);
        else
            throw   new   ArgumentException("No   parameter
specified");
```

```
        context.Response.ContentType = "image/jpeg";
        Stream strm = ShowAlbumImage(picid);
        byte[] buffer = new byte[4096];
        int byteSeq = strm.Read(buffer, 0, 4096);

        while (byteSeq > 0)
        {
            context.Response.OutputStream.Write(buffer,
0, byteSeq);
            byteSeq = strm.Read(buffer, 0, 4096);
        }
       }
     public Stream ShowAlbumImage(int picid)
    {
        string                    conn                    =
ConfigurationManager.ConnectionStrings["albumConnString"]
.ConnectionString;
        SqlConnection       connection       =       new
SqlConnection(conn);
        string  sql  =  "SELECT  pic  FROM  Album  WHERE
[pic_id] = @ID";
        SqlCommand cmd = new SqlCommand(sql, connection);
        cmd.CommandType = CommandType.Text;
        cmd.Parameters.AddWithValue("@ID", picid);
        connection.Open();
        object img = cmd.ExecuteScalar();
        try
        {
            return new MemoryStream((byte[])img);
        }
        catch
        {
            return null;
        }
        finally
        {
            connection.Close();
        }
```

```
    }

    public bool IsReusable
    {
        get
        {
            return false;
        }
    }
}
```

- Запускаємо додаток на виконання.

Контрольні питання

1. Яке існує ПЗ для розповсюдження та опрацювання вмісту та їх призначення?
2. Які основні сфери застосування технології серверних сторінок даних?
3. Наведіть функціональну схему реалізації технології серверних сторінок даних.
4. Як треба проводити аналіз вимог обробки серверної та клієнтської частини з точки зору навантаження та комунікаційних можливостей мережного середовища?
5. Як використовуються інструменти технології серверних сторінок даних при описі обробки інформації в рамках предметної сфери?

Рекомендована література

Основна

Автоматизовані системи. Терміни та визначення. ДСТУ 2226-93. – К. : Держстандарт України, 1994. – 32 с.

Бази даних. Терміни та визначення. ДСТУ 2874-94. – К. : Держстандарт України, 1995. – 32 с.

Дейт К. Дж. Введение в системы баз данных / К. Дж. Дейт. – 8-е изд. – К. : Диалектика, 2005. – 1328 с.

Ергономічне проектування робочих систем. Основні принципи. ДСТУ EN ISO 6385:2005. – К. : Держстандарт України, 2006. – 32 с.

Інформаційні системи та технології в економіці : посібник для студентів вищих навчальних закладів / за ред. В. С. Пономаренка. – К. : Вид. центр "Академія", 2002. – 544 с.

Інформаційні технології. Основні напрямки оцінювання та відбору CASE-інструментів. ДСТУ 3919-99 (ISO/IEC 14102:1995). – К. : Держстандарт України, 200. – 32 с.

Інформаційні технології. Процеси життєвого циклу програмного забезпечення. ДСТУ 3918-99 (ISO/IEC 12207:1995). – К.: Держстандарт України, 1999. – 32 с.

Інформаційні технології. Супровід програмного забезпечення. ДСТУ ISO/IEC 14764-2002. – К. : Держстандарт України, 2002. – 32 с.

Інформаційні технології. Словник термінів. ДСТУ ISO/IEC 2382-17:2005. Частина 17. Бази даних. – К. : Держстандарт України, 2006. – 32 с.

Інформаційні технології. Словник термінів. ДСТУ ISO/IEC 2382-18:2005. Частина 18. Розподілене оброблення даних. – К. : Держстандарт України, 2006. – 32 с.

Інформаційні технології. Система стандартів з баз даних. Еталонна модель керування даними. ДСТУ 3330-96 (ГОСТ 34.321-96). – К. : Держстандарт України, 1997. – 32 с.

Інформаційні технології. Система стандартів з баз даних. Концепція та термінологія для концептуальної схеми й інформаційної бази. ДСТУ 3329-96 (ГОСТ 34.320-96). – К. : Держстандарт України, 1997. – 32 с.

Кулямин В. В. Технологии программирования. Компонентный подход / В. В. Кулямин. – М. : МГУ, 2003. – 312 с.

Купцевич Ю. Microsoft ASP.NET, Web-сервисы, Web-приложения / Ю. Купцевич. – К. : Диадектика, 2003. – 400 с.

Нортон А. C# 3.0_ Эволюция LINQ и его влияние на проект C# – MSDN Magazine, June 200.

Оньон Ф. Основы ASP.NET с примерами на C# / Ф. Оньон. – К. : Диадектика, 2004. – 304 с.

Папа Дж. Приложения для привязки данных с помощью ADO.NET и настраиваемые объекты / Папа Дж. // MSDN Magazine. – February, 2007.

Пономаренко В. С. Проектування баз даних : навч. посібн. / В. С. Пономаренко, Л. А. Павленко, І. О. Максименко. – К. : ІЗМН, 1997. – 172 с.

Системи оброблення інформації. Основні поняття. Терміни та визначення. ДСТУ 2938-94. – К. : Держстандарт України, 1995. – 32 с.

Системи стандартів з баз даних. Структура системи словників інформаційних ресурсів. ДСТУ 3302-96. – К. : Держстандарт України, 1997. – 32 с.

Системи оброблення інформації, Керування процесами оброблення даних. Терміни та визначення. ДСТУ 2940-94. – К. : Держстандарт України, 1995. – 28 с.

Системи оброблення інформації. Розроблення систем. Терміни та визначення. ДСТУ 2941-94. – К. : Держстандарт України, 1995. – 20 с.

Системи оброблення інформації. Керування процесами. Оброблення даних. Терміни та визначення. ДСТУ 2940-94. – К. : Держстандарт України, 1995. – 32 с.

Система стандартів з баз даних. Мова баз даних SQL з розширенням цілісності. ДСТУ 3149-95. – К. : Держстандарт України, 1995. – 32 с.

Системи управління якістю. Вимоги. ДСТУ ISO 9001-2001. – К. : Держстандарт України, 2001. – 32 с.

Системи управління якістю. Настанови щодо поліпшення діяльності. ДСТУ ISO 9004-2001. – К. : Держстандарт України, 2001. – 32 с.

Системи управління якістю. Основні положення та словник. ДСТУ ISO 9000-2001. – К. : Держстандарт України, 2001. – 32 с.

Додаткова

Microsoft Corporation. Разработка Web-приложений на Microsoft Visual Basic .NET и Microsoft Visual C# .NET : учебный курс MCAD/MCSD. – М. : Издательско-торговый дом "Русская Редакция", 2003. – 704 с.

Microsoft Corporation. Разработка Web-сервисов XML и серверных компонентов на Microsoft Visual Basic .NET и Microsoft Visual C# .NET : учебный курс MCAD/MCSD. – М. : Издательско-торговый дом "Русская Редакция", 2004. – 576 с.

Ресурси мережі Інтернет

Биндим грид обьектами [Электронный ресурс]. – Режим доступа : www.developeru.info/PermaLink,guid2c41e9f1-7492-4794-9aed-317634f1b8-e1.aspx.

Вендров А. М. Современные методы и средства проектирования информационных систем [Электронный ресурс]. – Режим доступа : www.citforum.ru/database/case/index.shtml CASE-технологии.

Керування даними [Электронный ресурс]. – Режим доступа : uk.wikipedia.org/wiki/Категорія:Керування_даними.

Разработка XNA [Электронный ресурс]. – Режим доступа : www.xnadev.ru.

Разработка web-приложений с ASP.NET, C#, AJAX в примерах. [Электронный ресурс]. – Режим доступа : www.developeru.info/Perma-Link,guid,35f3117e-3ad3-41ba-be95-ac02f54cd0f3.aspx.

Садоян Р. ASP на блюдечке [Электронный ресурс]. – Режим доступа : www.fububy.na.by/index.php?elif=asp/aaa07d7ebeac7309678fbca-4b7b69a78.htm.

Стандарт онтологического исследования IDEF5 [Электронный ресурс]. – Режим доступа : www.citforum.ru/cfin/idef/idef5.shtml.

Статьи по ASP [Электронный ресурс]. – Режим доступа : www.fububy.na.by/index.php?tsrid=asp.

Хоффман Кевин LINQ to Entities vs. LINQ to SQL – что когда использовать? [Электронный ресурс]. – Режим доступа : www.optim.su/issues.asp.

CRa.SH, Основы LINQ T [Электронный ресурс]. – Режим доступа : www.gotdotnet.ru/LearnDotNet/DotNet30/default.aspx.

ORM в одном классе за полчаса [Электронный ресурс]. – Режим доступа : www.developeru.info/PermaLink,guid,35f3117e-3ad3-41ba-be95-ac02f54cd0f3.aspx.

Vista для разработчика [Электронный ресурс]. – Режим доступа : www.thevista.ru/.

Зміст

НАВЧАЛЬНЕ ВИДАННЯ

Методичні рекомендації до виконання лабораторних робіт з навчальної дисципліни "ВИДАВНИЧІ БАЗИ ДАНИХ"

для студентів галузі знань 0515 "Видавничо-поліграфічна справа" всіх форм навчання

Укладач **Пандорін** Олександр Костянтинович

Відповідальний за випуск **Пушкар О. І.**

Редактор **Бутенко В. О.**

Коректор **Бриль В. О.**

План 2012 р. Поз. № 364.

Підп. до друку Формат 60 х 90 1/16. Папір MultiCopy. Друк Riso.

Ум.-друк. арк. 3,5. Обл.-вид. арк. 4,38. Тираж прим. Зам. №

Видавець і виготівник — видавництво ХНЕУ, 61166, м. Харків, пр. Леніна, 9а

Свідоцтво про внесення до Державного реєстру суб'єктів видавничої справи ***Дк № 481 від 13.06.2001 р.***