МЕТОДИЧНІ ВКАЗІВКИ ЧАСТИНА №3
ДО ВИКОНАННЯ ЛАБОРАТОРНИХ РОБІТ

З ДИСЦИПЛІНИ:

«Організація баз даних»

Для студентів 3 курсу навчання
Спеціальності– Комп’ютерні науки

Затвердженно на методичній комісії
Факультету комп’ютерних наук
Протокол№____ від _____
Декан _____Коваленко Л.Б.

Затверджено на засіданні кафедри
Інформаційних технологій
Протокол№___ від _____
Зав. Кафедрою _____Кузніченко С.Д.

Одеса 2018

Методичні вказівки частина №3 для виконання лабораторних робіт з дисципліни «Організація баз даних» для студентів III курсу денної форми навчання. Спеціальність – «Комп’ютерні науки» / Козловська В.П.. Штефан Н.З.– Одеса, ОДЕКУ, 2018 – с. 56

ЗМІСТ

ВСТУП

Методичні вказівки призначені для виконання лабораторних робіт №№7-8 з дисципліни «Організація баз даних».

Завдання основані на результатах попередніх лабораторних робіт, а саме створена база даних згідно з варіантом студента.

Оцінювання здійснюється згідно робочої програми (з урахуванням виконаної програмної частини роботи та кількість часу, за який студент реалізував поставлену перед ним задачу).

У процесі самостійного вивчення курсу студент повинен керуватися його програмою і вивчити за конспектом лекцій та літературою, що рекомендована викладачем, відповідний теоретичний матеріал.

Лабораторна робота №7 «Підключення БД MS SQL до клієнтського додатку у Visual Studio 2010»

Мета роботи:

1. Знайомство з інтерфейсом Visual Studio 2010.
2. Отримання практичних навичок підключення бази даних MS SQL за допомогою компонентів Visual Studio 2010.

Теоретичні відомості

Система Microsoft Visual Studio має в наявності потужні вбудовані засоби створення і управління базами даних. Передбачена можливість створити новий проект бази даних і імпортувати схему бази даних з існуючої бази даних, SQL-файлу скрипта.

Потім можна викликати ті ж кошти програми проектування з графічним інтерфейсом (редактор Transact-SQL, конструктор таблиць), які доступні для розробки підключеної бази даних, щоб внести зміни в проект бази даних поза мережею і опублікувати зміни у виробничій базі даних.

Для створення клієнтського додатку до бази даних по перше необхідно створити проект в Visual Studio 2010.

Проект - це основна одиниця, з якої працює програміст. Він вибирає тип проекту - а Visual Studio створює шаблон проекту відповідно до обраного типу. У Visual Studio рідко створюється порожній проект. Замість цього ми повинні вказати середовищі тип проекту, який хочемо створити, а середовище розробки генерує файли і програмний код, які служать основою для обраного типу проекту. Після цього можна працювати над проектом, додаючи ваш код до створеного шаблоном проекту, що значно спрощує роботу.

Практична частина

У нашому випадку необхідно створити проект, заснований на Windows Forms, для чого Visual Studio згенерує проект з порожньою формою. Вибраний тип проекту також говорить компілятору які зовнішні бібліотеки необхідно підключити.

Перейдемо безпосередньо до реалізації завдання лабораторної роботи. Після запуску Microsoft Visual Studio вибираємо на стартовій сторінці пункт New Project – створити новій проект (див. рисунок 1):

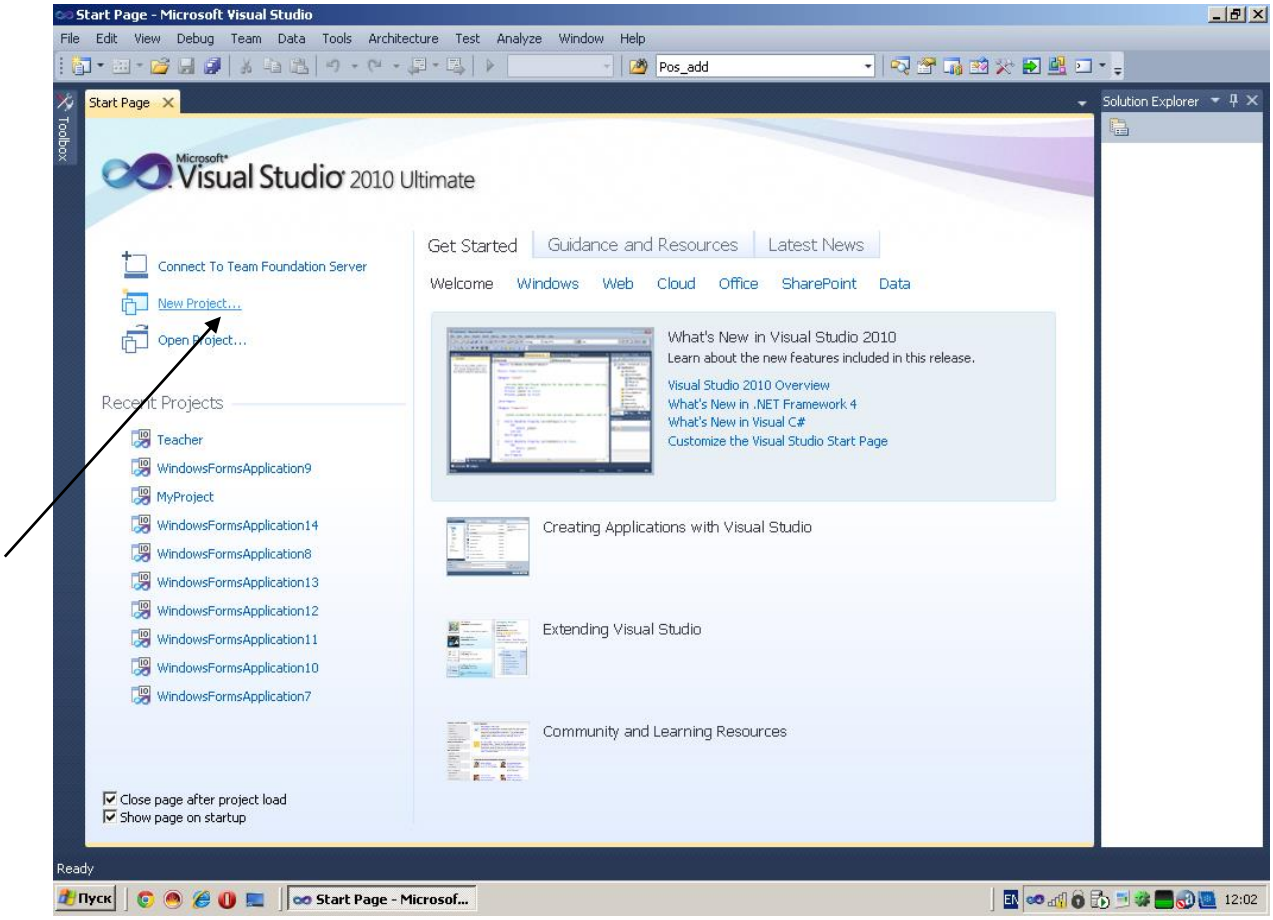


Рисунок 1 – Створення нового проекту у Visual Studio 2010

У діалоговому вікні New Project зі списку встановлених шаблонів Visual C # обраний за замовчуванням. Кількома за додатком Windows Forms Application Visual C #.

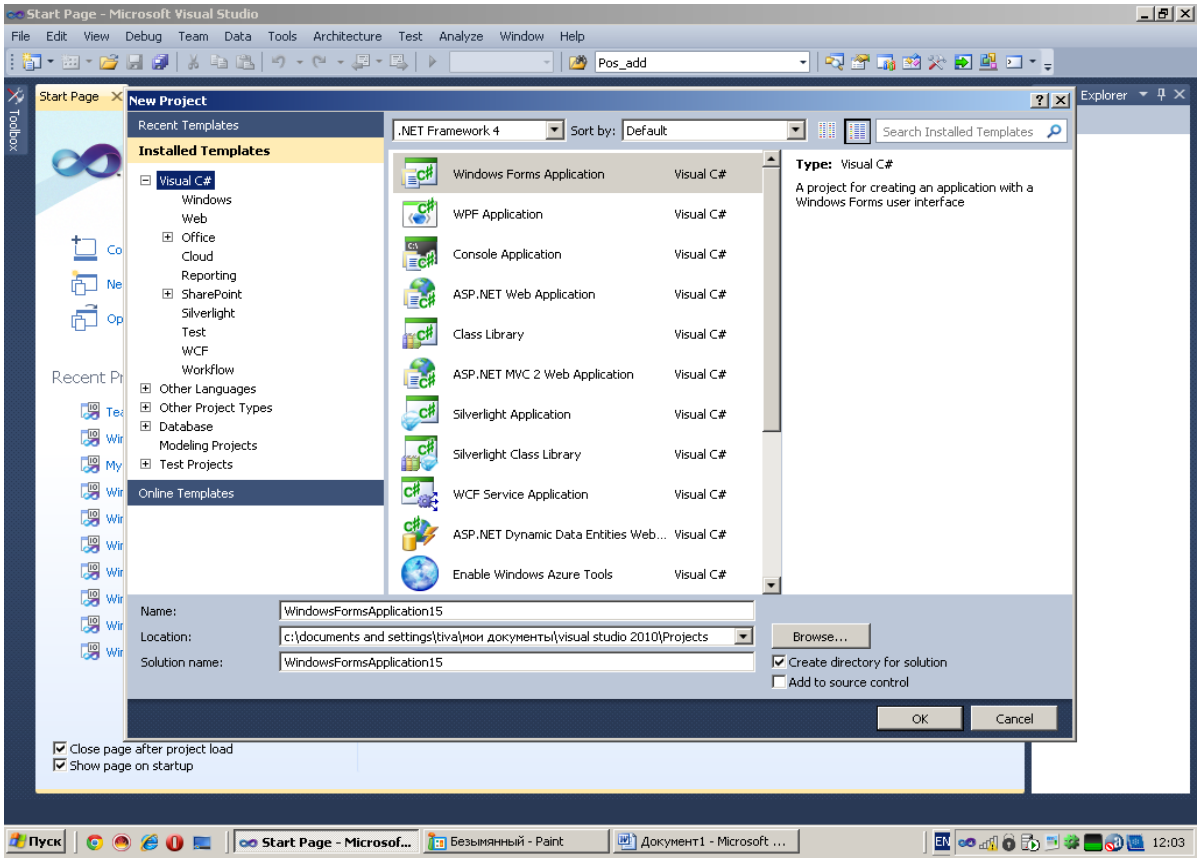


Рисунок 2 – Вибір шаблону проектування

В поле Name потрібно вказати ім'я для майбутньої програми ApplicationPersonal (згідно з завданням на лабораторну роботу, студент пише своє прізвище), в поле Location можна змінити заданий за замовчуванням шлях до проекту.

У разі роботи над прикладом простого проекту, буде зручно все його файли зберігати в одній папці, необхідно скинути прапорець Create Directory For Solution. Тим не менше, більшість додатків зазвичай складаються з декількох проектів, тому скидати даний прапорець в загальному випадку не рекомендується - ця опція допомагає систематизувати структуру папок і уникнути плутанини і протиріч.

У разі установки прапорця Add To Source Control, Visual Studio відкриє нове вікно, в якому можна настроїти репозиторій вихідного коду (source

control). Він являє собою сховище для реєстрації коду. Ця можливість особливо корисна для колективної роботи, коли кожен розробник зможе реєструвати свій код в загальному сховищі вихідних кодовпрі роботі над масштабним рішенням.

Для завершення підтверджуємо, натиснувши кнопку ОК.

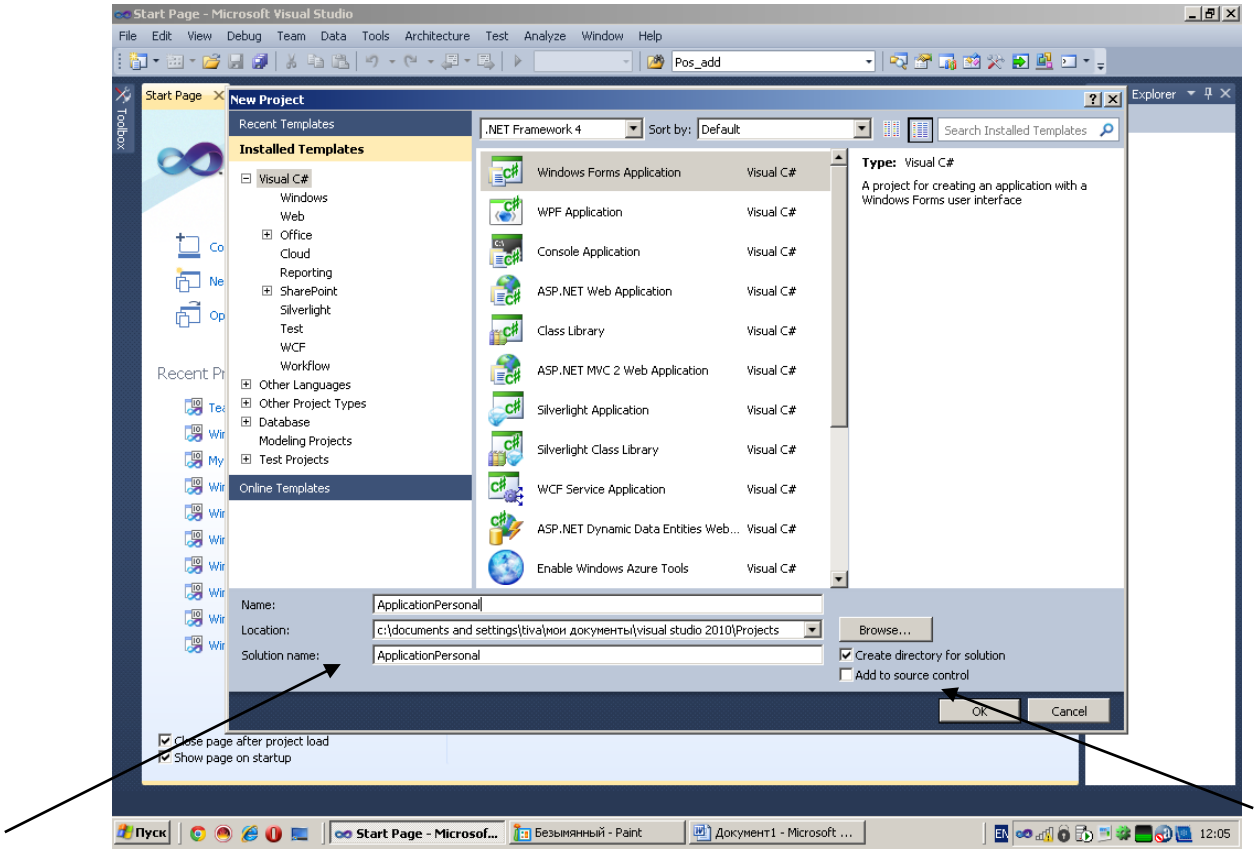


Рисунок 3 – Завдання імені та повного шляху до головної папки проекту

Після підтвердження імені та повного шляху до проекту відкривається основна форма програми, з якою і будемо працювати далі (рисунок 4).

Visual Studio надає безліч вікон, який відображають інформацію, необхідну для створення додатків. Деякий вікна, такі як Solution Explorer, відображені за замовчуванням, інші, наприклад налагоджувальні, з'являється при запуску програми в режимі налагодження. Повний список доступних

вікон розташований в головному меню Visual Studio під опцією View (рисунок 5).

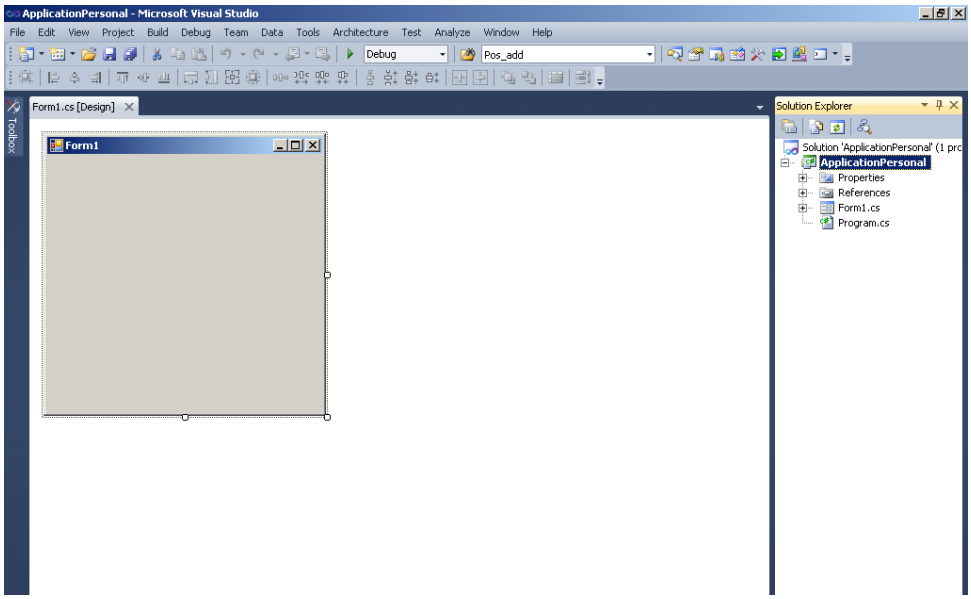


Рисунок 4 - Головна форма проекту

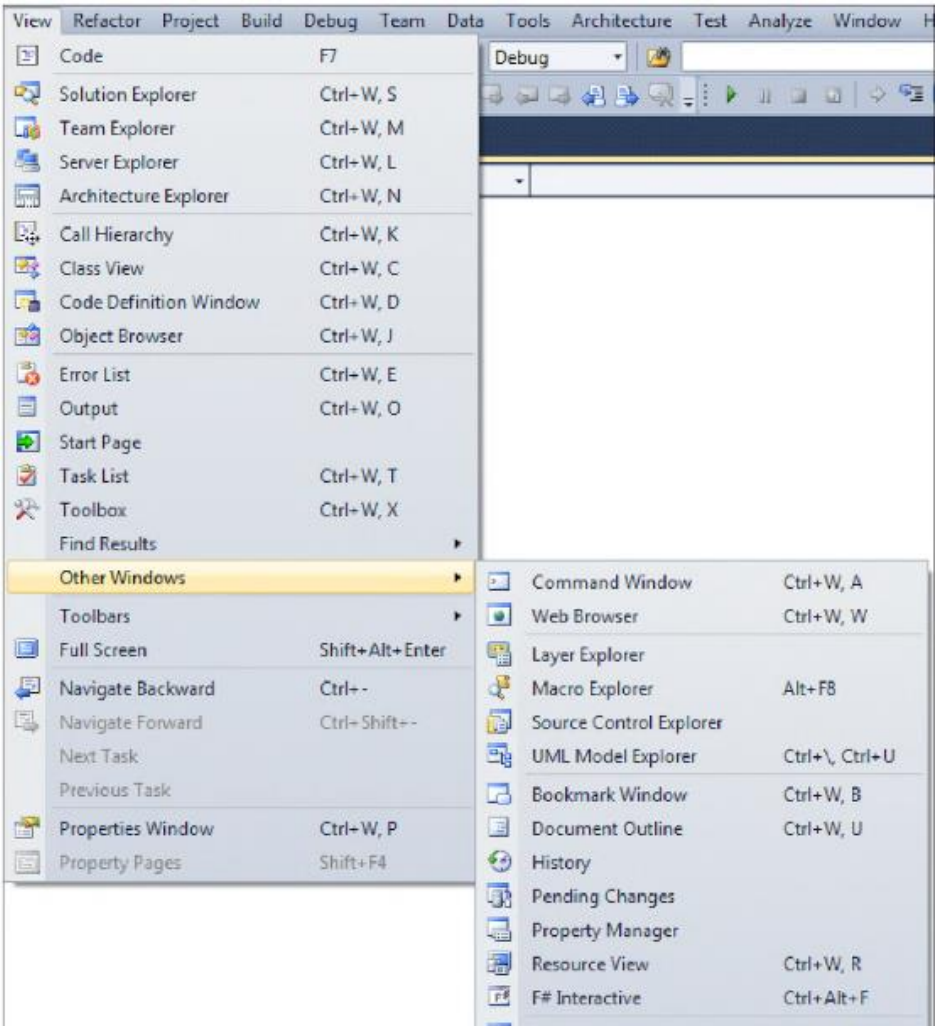
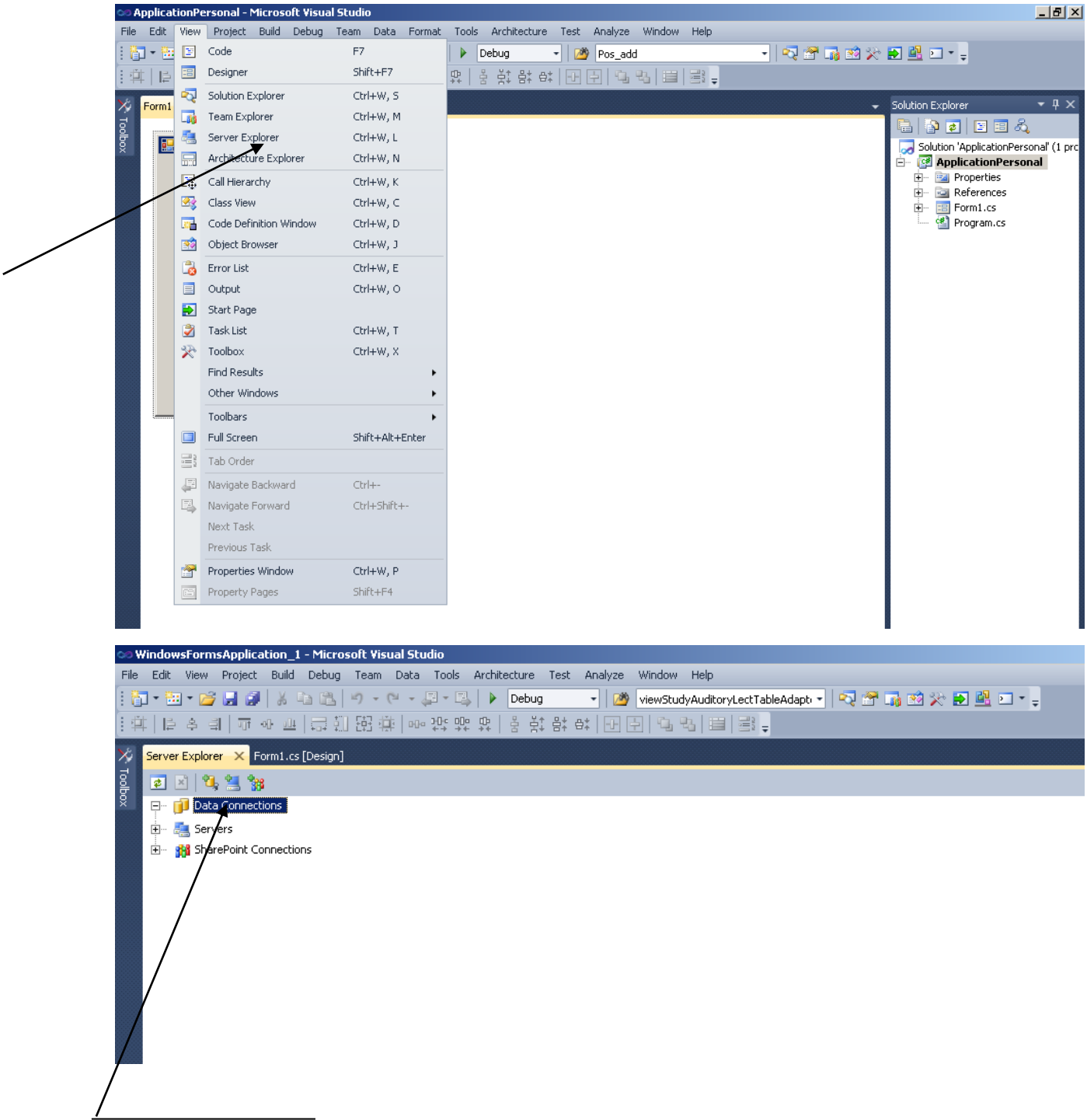


Рисунок 5 – Список доступних вікон Visual Studio 2010

Наступним кроком необхідно підключити нашу базу даних Personal. Для цього нпотрібно зайти в пункт меню Vew-> Server Explorer. У вікні клацаємо правим кліком по Data Connections та обираємо Add Connection ... (додати з'єднання).



Правой кнопкой

Рисунок 6 – З'єднання Data Connection

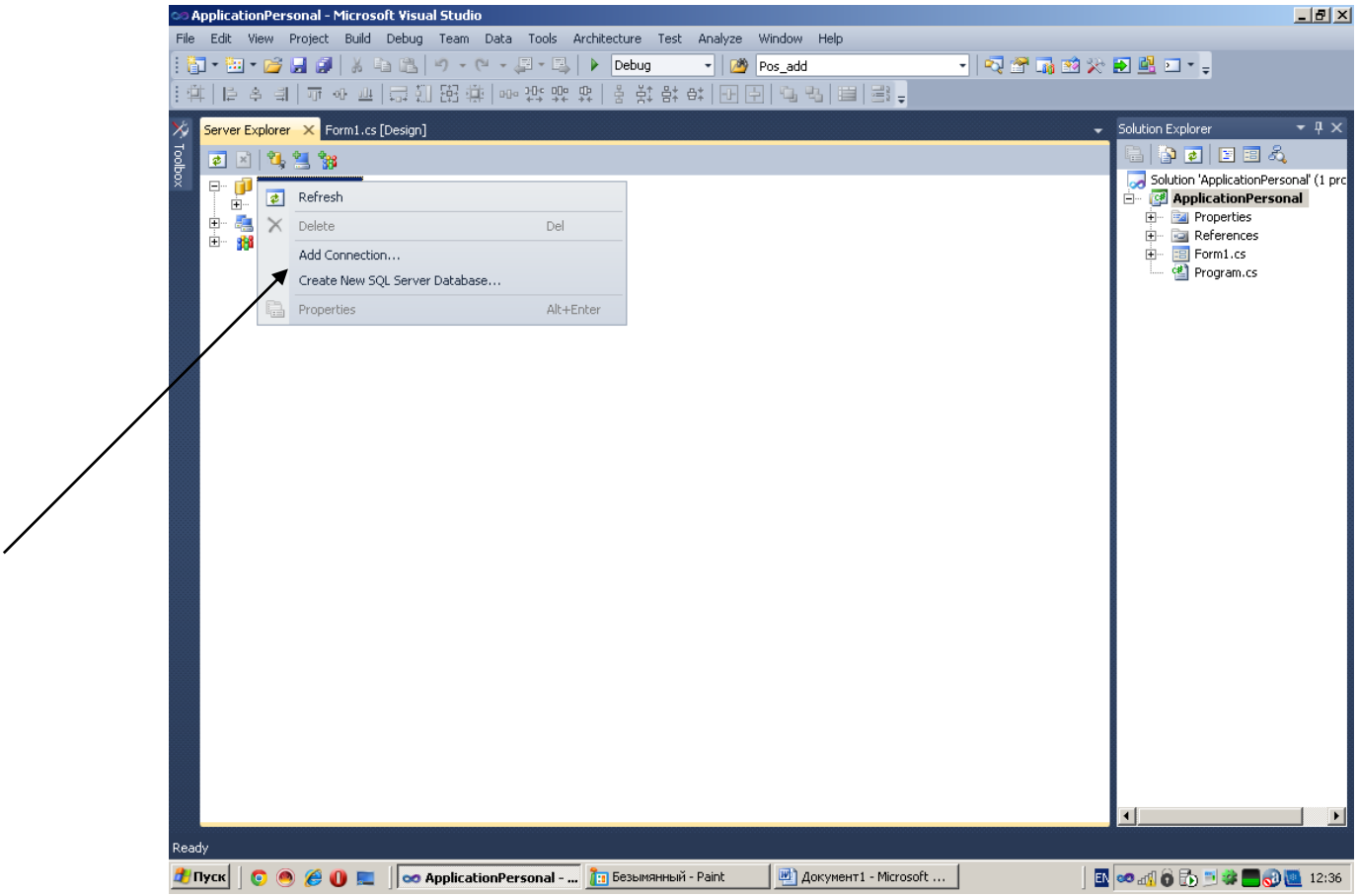


Рисунок 7 – Створення нового з'єднання

Після вибору з'єднання необхідно вибрати джерело даних Data Source. За замовчуванням джерелом є Microsoft SQL Server (SQL Client) (рисунок 8).

За допомогою кнопки Change… робимо виклик вікна Change Data Source (рисунок 9), де можно побачити список доступних джерел, у нашому випадку треба залишити Microsoft SQL Server (SqlClient), а також Data provider – вибираємо .Net Framework Data Provider for OLE DB і підтверджуємо вибір кнопкою OK.

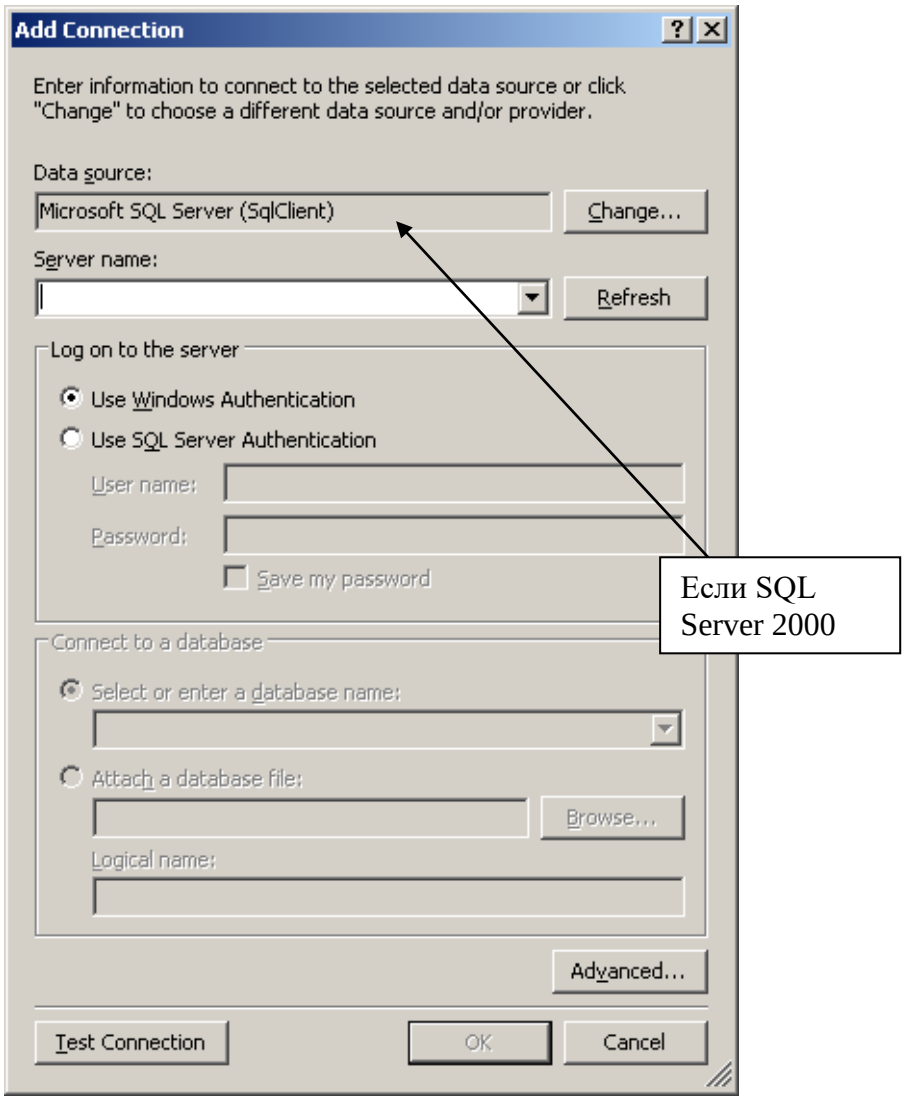


Рисунок 8 – Параметри з’єднання з сервером.

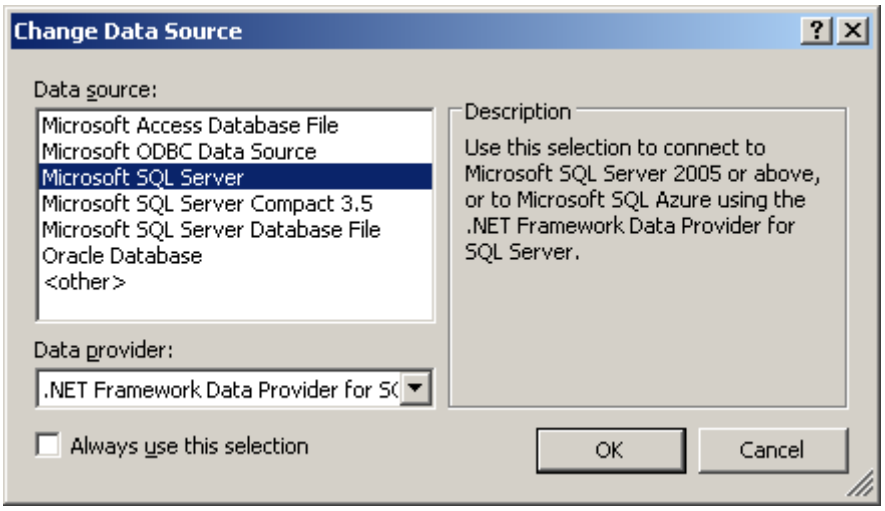


Рисунок 9 – Вікно Change Data Source.

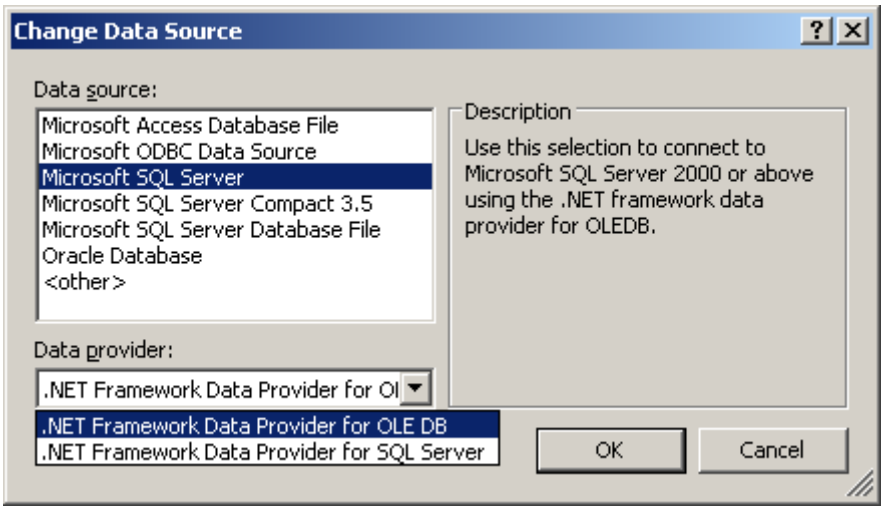


Рисунок 10 – Вікно Change Data Source (параметр Data provider).

Вікно Add Connection дозволяє вибрати ім'я доступного сервера і ім'я підключається бази даних (наприклад Personal). Для параметра Log on to the server залишаємо Use Windows Authentication. Після цього запускаємо тестування з'єднання кнопкою Test Connection.

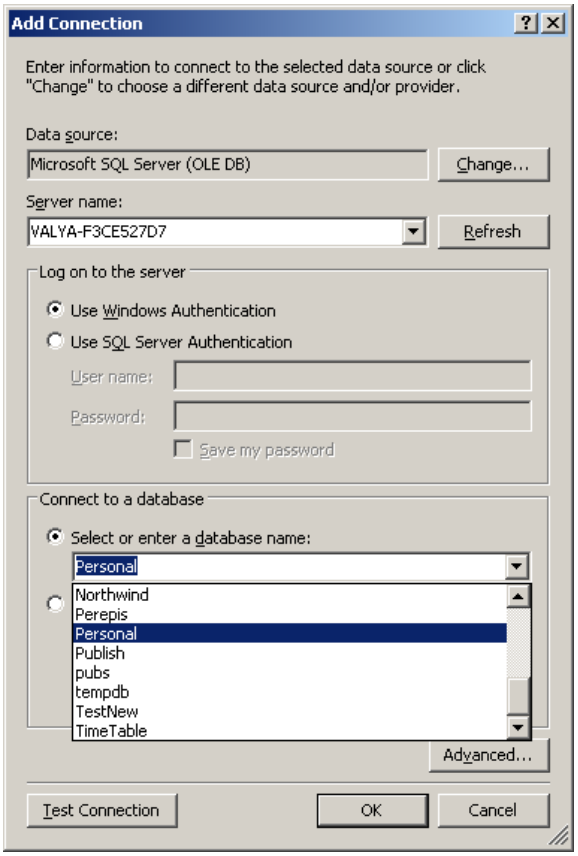


Рисунок 11– Вікно Add Connection.

При успішному з'єднанні висвічується повідомлення «Test connection succeeded».

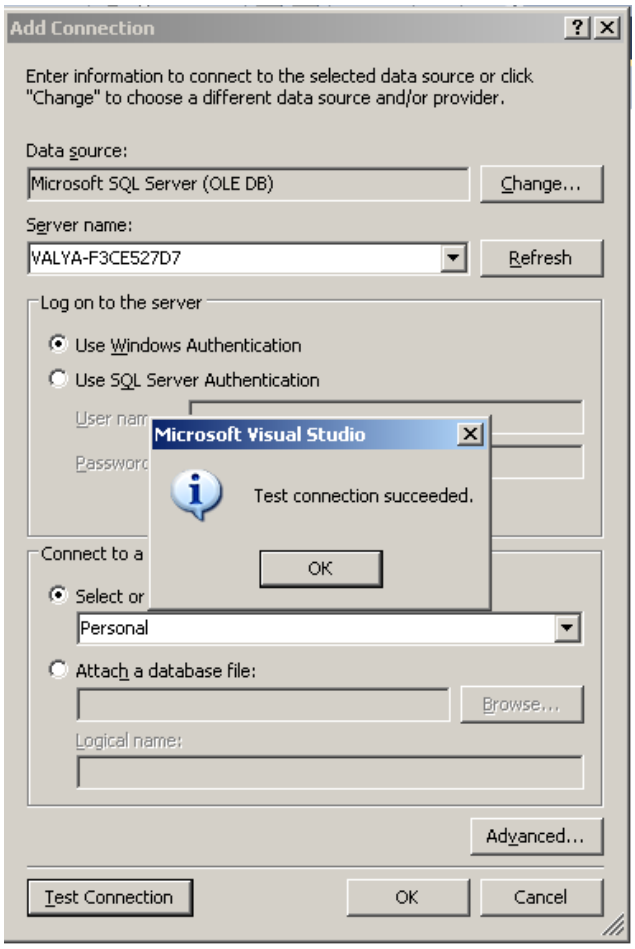


Рисунок 12 – Успішне підтвердження з’єднання.

Після чого в Server Explorer відображається ім'я джерела даних (рисунок 13).

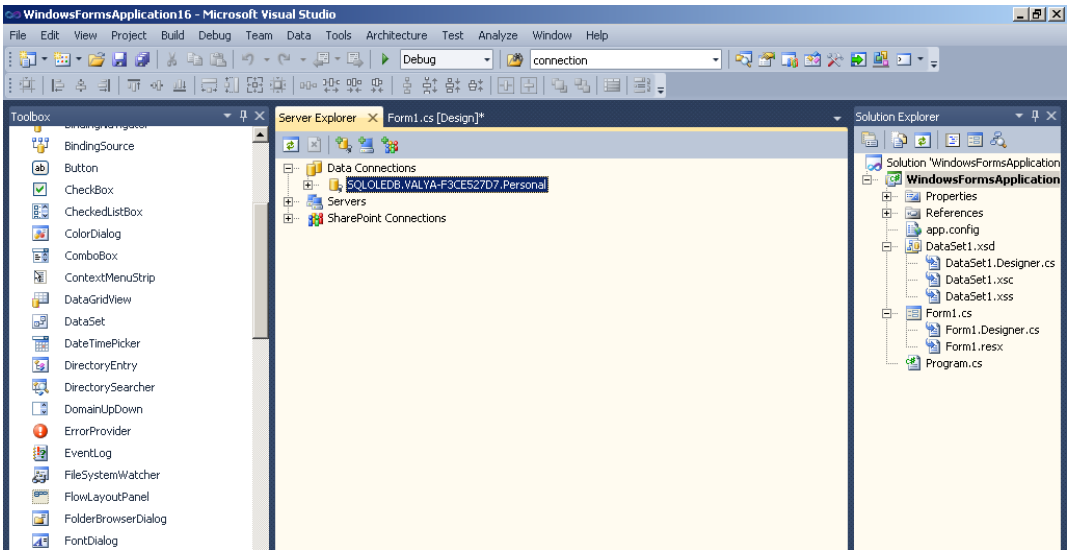


Рисунок 13 – Вкладка Server Explorer.

Переходимо до форми додатку.

Елементи управління на формі називається об'єктами. Кожен об'єкт має свій набір властивостей, подій і методів:

- Властивості об'єкта - це його характеристики (висота, ширина і т.д.);
- Події об'єкта - це події операційних систем або події ініціюються користувачем, на які може реагувати об'єкт (натискання кнопки);
- Методи об'єкта - дії, які можна виробляти з об'єктом в ході виконання програм.

В БД всі об'єкти форм діляться на два класи:

- об'єкти управління - об'єкти, які здійснюють управління БД (Наприклад: Кнопка або Список, що випадає);
- об'єкти для відображення інформації - елементи, що відображають вміст таблиць, запитів або фільтрів, що дозволяють додавати змінювати і видаляти інформацію, і проводити її аналіз.

Всі форми в клієнтському додатку діляться на три групи:

1. Форми для роботи з даними - форми, що містять як об'єкти управління, так і об'єкти перегляду даних. Такі форми призначені для відображення, зміни, видалення та аналізу даних;
2. Кнопкові форми - форми, що містять тільки об'єкти управління, призначаються для відкриття всіх інших форм.

   Зауваження: Кнопкова форма, яка з'являється першою після запуску програми, називається, головною кнопковою формою.
3. Інформаційні та службові форми - форми, що містять тільки елементи управління, призначені для відображення службової інформації (довідки), незв'язаної з таблицями, запитами і фільтрами, або для виконання службових операцій не пов'язаних з даними (Наприклад: форма з калькулятором)

Існує два види дизайну форм:

1. Стрічкові форми - форми, що виводять інформацію по одному запису;

2. Табличні форми - форми виводять інформацію у вигляді таблиці.

Найбільш часто в БД використовуються наступні об'єкти для відображення інформації:

1. Текстове поле (TextBox)

2. Напис (Label)

3. Напис з посиланням (LinkLabel)

4. Календар (DataPicker)

5. Перемикач (CheckBox)

6. Таблиця (DataGridView)

7. Список (ListBox)

8. Список, що випадає (ComboBox)

9. Текстове поле з маскою введення (MaskedTextBox)

TextBox - відображає текст і числові поля, це найбільш часто вживається об'єкт для відображення даних. Його можна створювати або перетягуванням з вікна "Data Sources", або підключити вручну. Створення цього об'єкта, перетягуванням можливо майже у полів будь-яких типів даних.

Label - повністю аналогічний об'єкту TextBox, але не дозволяє змінити дані. Цей об'єкт використовується для відображення заблокованих незмінних полів.

LinkLabel - спеціальний об'єкт для відображення посилань на адреси в Інтернеті. Його використовують для відображення текстових полів, якщо в них зберігаються адреси Інтернету або якийсь комп'ютерної мережі. Це новий об'єкт, йому не було аналога в Visual Basic 6.0.

DataPicker - спеціальний об'єкт, призначений для відображення полів типу даних "Дата / Час" у вигляді календаря.

CheckBox - об'єкт використовується для відображення логічних полів, може бути створений перетягуванням тільки для логічних полів.

DataGridView - об'єкт, що відображає джерело даних (таблицю, запит або фільтр) у вигляді таблиці.

ListBox - список відображає значення полів і дозволяє вибирати значення полів зі списку. Більш того, пункти списку можна задавати, використовуючи інше джерело даних.

ComboBox - об'єкт подібний об'єкту ListBox, проте інформація відображається не в списку, а випадаючому списку.

Для початку на головній формі проекту необхідно розмістити компонент TabControl з палітри Toolbox Visual Studio (рисунок 14).

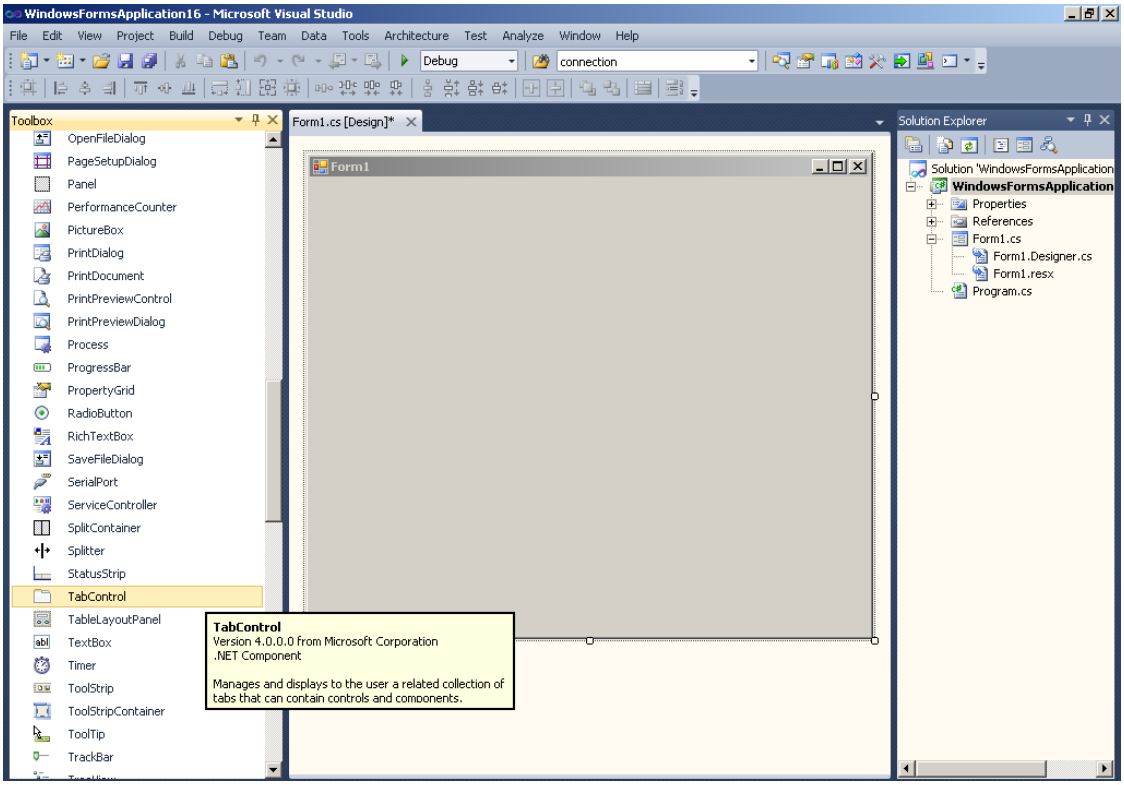


Рисунок 14 – Головна форма проекту.

TabControl: це компонент, який складається з вкладок. Кожна вкладка є контейнером, тобто - там можна розташовувати інші компоненти Visual Studio. Навіть можна ще один TabControl туди помістити. Між вкладками можна перемикатися за допомогою миші (всіх закладках задається своє ім'я).

За замовчуванням з'являється дві вкладки. Але - це кількість можна збільшити. Для цього треба виділити сам елемент TabControl (саме елемент, а не вкладку; для цього треба виділяти саму верхню частину компонента TabControl). Після цього справа зверху з'явиться трикутник. Натиснувши на нього з'явиться дві кнопки: AddTab (додавання вкладки) і Remove Tab (видалити вкладку):

Також - управляти вкладками можна за допомогою властивості TabControl під назвою TabPages. Навпаки цього властивості буде кнопка (з трьома кнопками), натиснувши на неї відкриється вікно, в якому вже можна працювати з кожною конкретною вкладкою (видаляти, додавати вкладки, і задавати різні властивості для вкладок):

Отже, правим кліком по формі на TabControl вибираємо Add Tab (додати нову закладку - для кожної таблиці з нашої бази даних буде своя закладка для відображення її даних). Таким чином маємо 5 закладок.

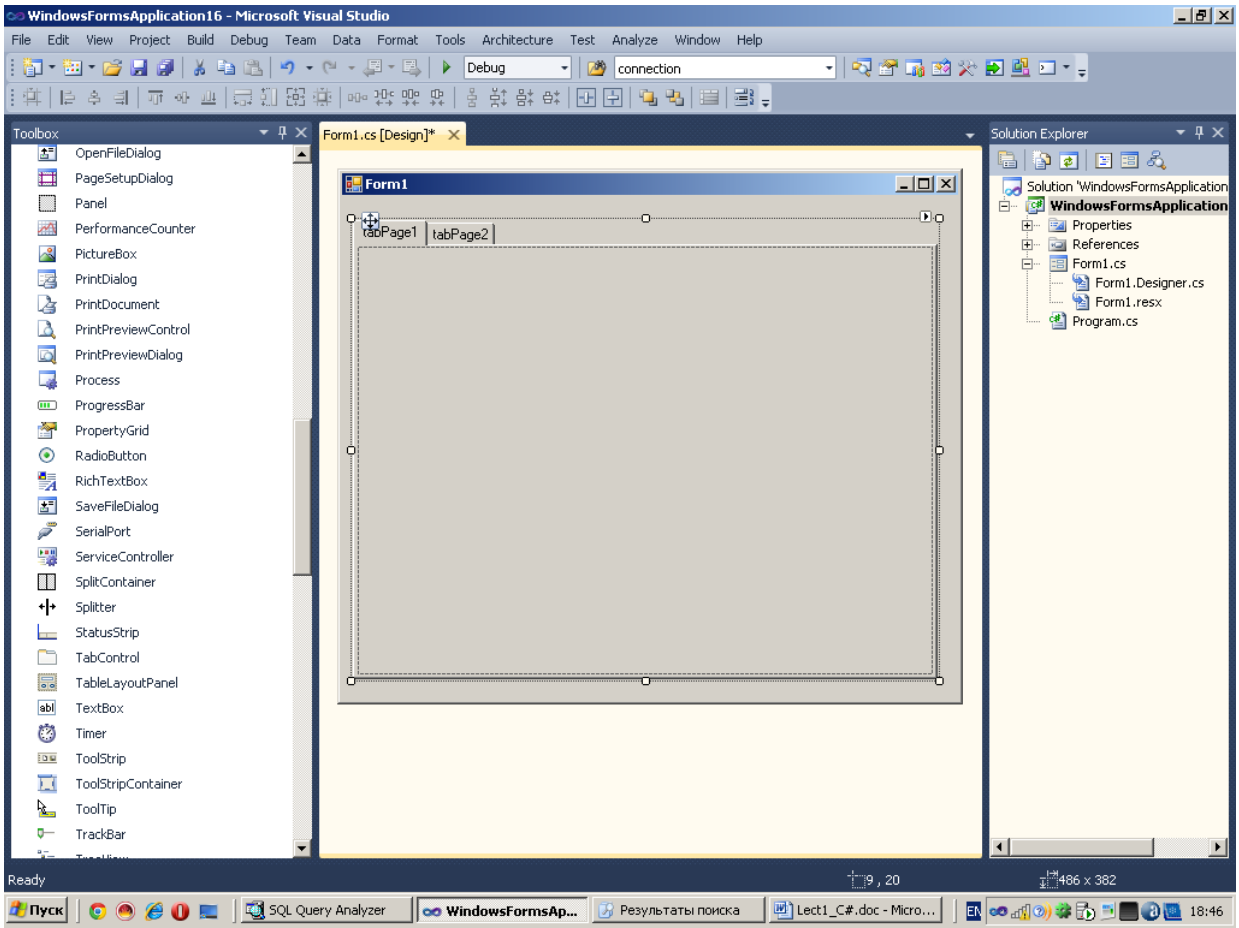


Рисунок 15 – Робота з компонентом TabControl.

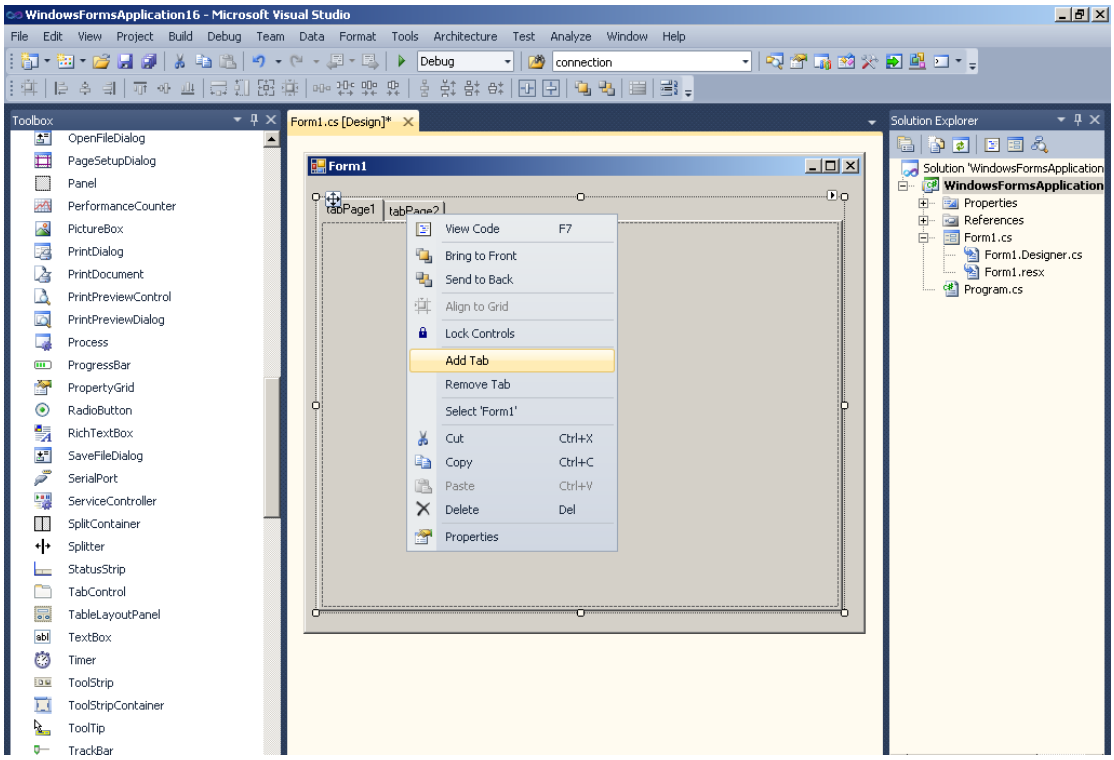


Рисунок 16 – Додавання нових закладок.

Тепер необхідно зайти до властивостей компонента TabControl на нашій формі за допомогою правого кліка. Вікно Properties дає можливість їх керуванням.

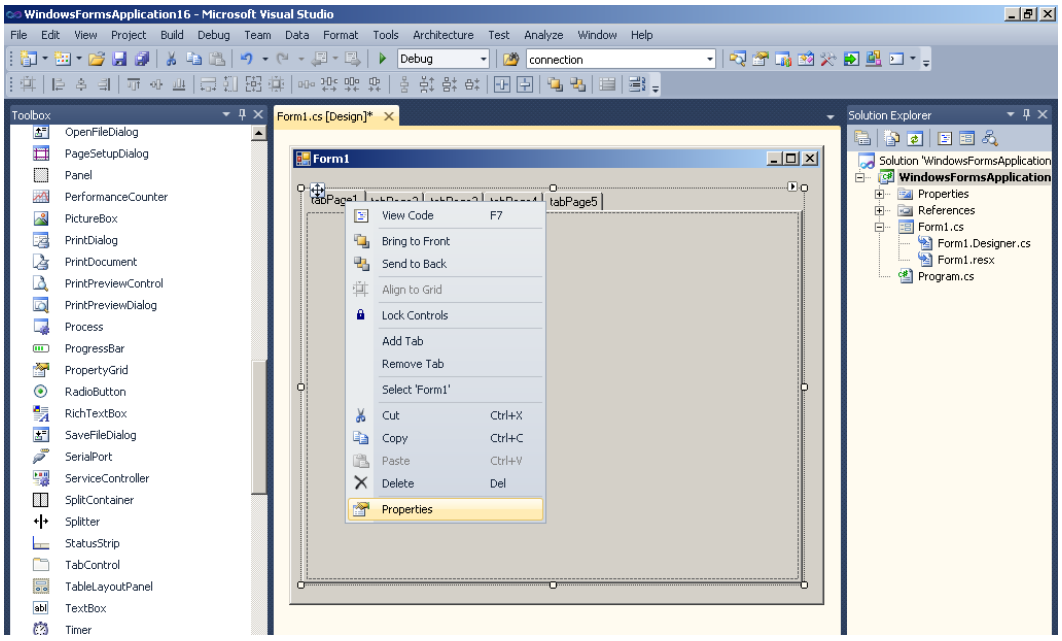


Рисунок 16- Вибір Properties компонента TabControl.

Потрібно зайти у властивість TabControl->TabPages… де для кожної сторінки змінюємо текстовий напис TabPage-> на ім'я таблиці з БД (як показано на рисунку 17).

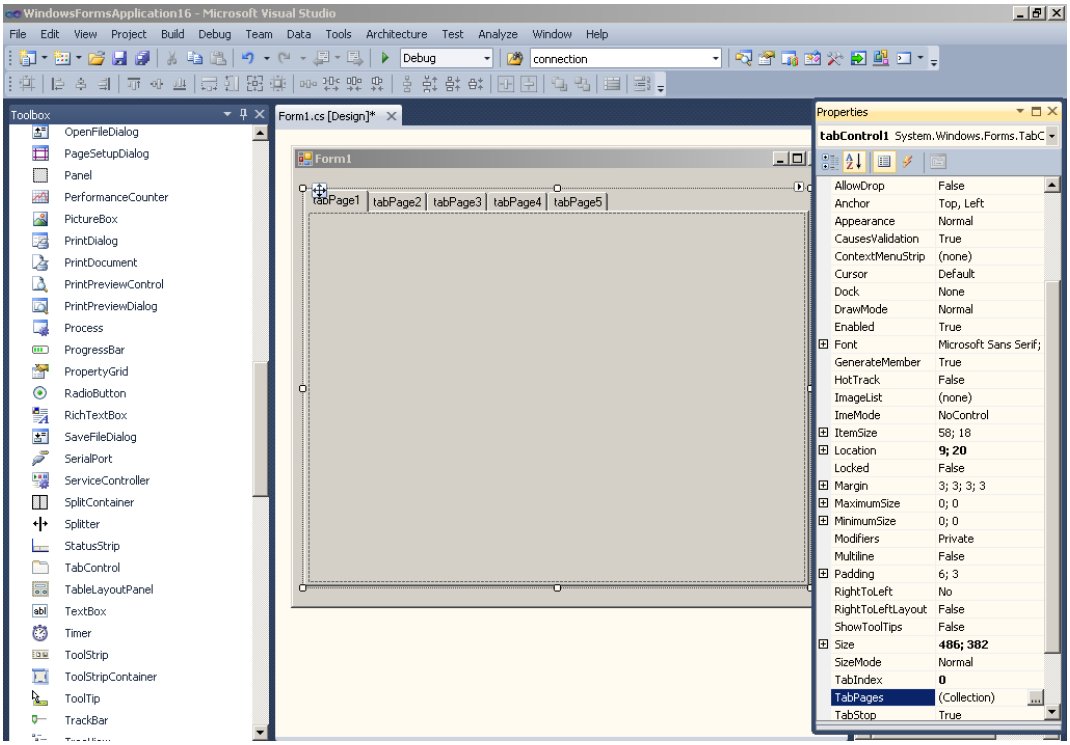


Рисунок 17 – Вибір властивості TabPages.

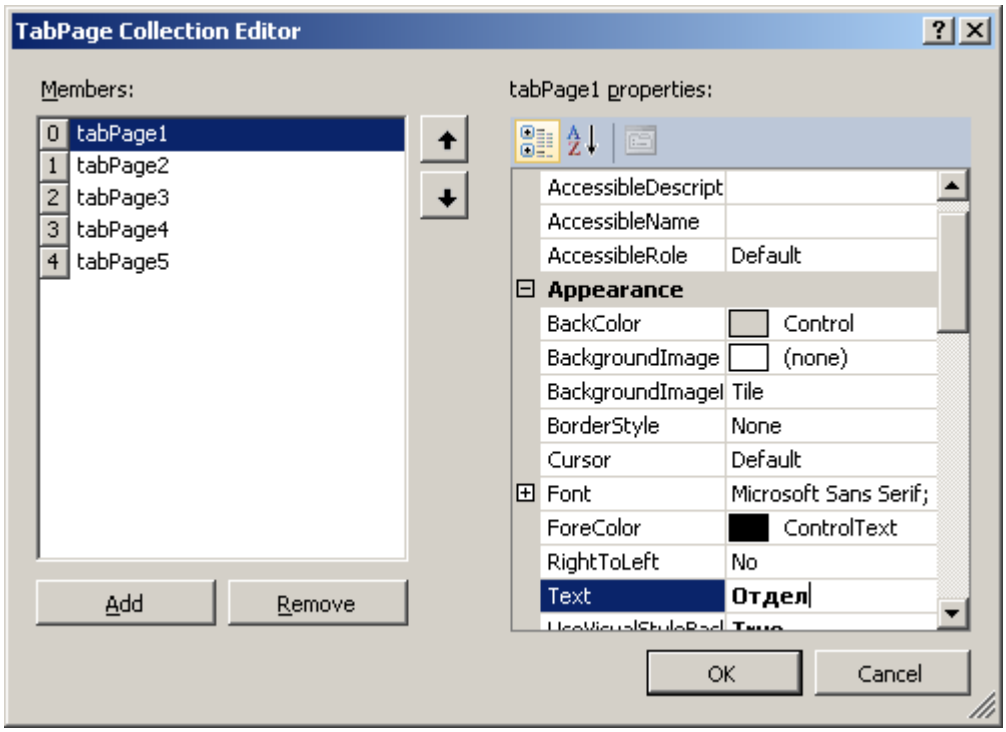


Рисунок 18 – Робота з Text TabControl.

Так для tabPage1-> висвітиться напис Відділ,

tabPage2-> висвітиться напис Посада,

tabPage3-> висвітиться напис Особовий склад,

tabPage4-> висвітиться напис Відділ,

tabPage5-> висвітиться напис Відпустка.

В результаті головна форма проекту прийме наступний вигляд:

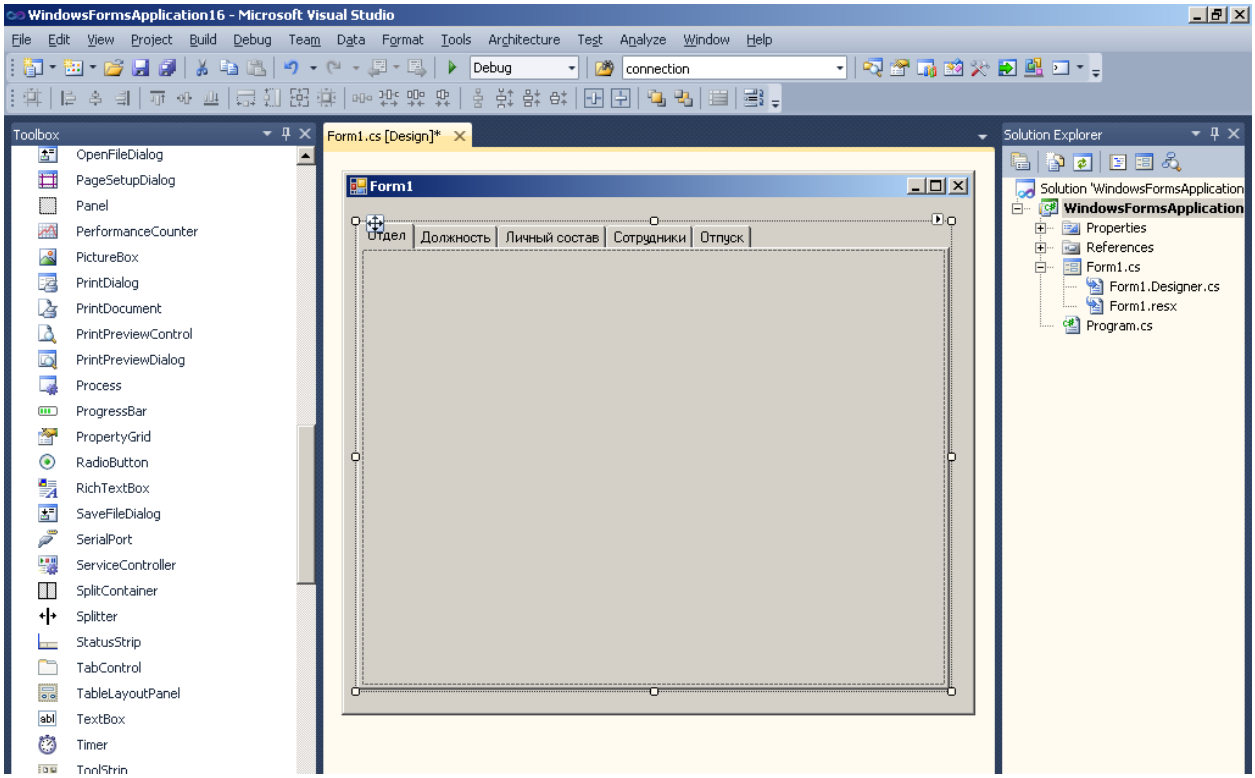


Рисунок 19 – Перейменовані закладки TabControl.

Задайте властивості форми наступним чином:

• FormBorderStyle (Стиль кордону форми): Fixed3D;

• MaximizeBox (Кнопка розгортання форми на весь екран): False;

• MinimizeBox (Кнопка згортання форми на панель задач): False;

• Text (Текст напису в заголовку форми): призвище студента.

Тепер необхідно вибрати елемент керування DataGridView з палітри Toolbox. Він дозволяє відображати і редагувати табличні дані з різних типів джерел данних. Розглянемо його детальніше.

Об'єкт DataGridView призначений для відображення всієї інформації з таблиць, запитів або фільтрів на формі у вигляді таблиці. Цей об'єкт може бути створений як вручну (з подальшим його підключенням), так і перетягуванням за все джерела даних з вікна "Data Sources".

Однак найбільш часто його створюють перетягуванням всієї таблиці, запиту або фільтра з вікна "Data Sources" на форму.

При перетягуванні цього об'єкта на форму, як і у випадку з іншими об'єктами з'являється панель навігації. Вона виконує функції: переміщення по записах, додавання, видалення і збереження записів. Після створення об'єкта DataGridView можна налаштовувати як властивості всього об'єкта, так і властивості окремих стовпців.

Почнемо з настройки властивостей всього об'єкта. Налаштування даних властивостей здійснюється в основному через меню дій.

Можливі наступні варіанти ввімкнення:

- Choose Data Source - джерело даних, що відображається в таблиці;
- Enable Adding - додавати записи;
- Enable Deleting - дозволяється користувачам видаляти записи;
- Enable Editing - дозволяється користувачам змінювати значення полів таблиці;
- Enable Column Reordering - дозволяється користувачам змінювати порядок стовпців, просто перетягуючи їх мишею.

Також в меню дій можливі наступне дії з таблицею:

- Dock in parent container - вписати об'єкт в форму;
- Preview Data - з'являється вікно з попереднім переглядом таблиці;
- Add Query - додає SQL - запит, який виконується на стороні клієнта;
- Add Column - додавання нового стовпця в таблицю;
- Edit Columns - настройка властивостей окремих стовпців таблиці.

Тепер перейдемо до налаштування окремих стовпців таблиці.

Якщо в меню дій вибрати пункт "Edit Columns", то з'являється вікно, де можна додавати, видаляти і редагувати стовпці. Для цього в списку стовпців лівій частині вікна вибираємо стовпець, а в правій - налаштовуємо його властивості. Найбільш часто настроюються наступні властивості:

1. Name - ім'я стовпця;

2. AutoSizeMode - підгонка ширини шпальти по його вмісту;

3. ColumnType - визначає зовнішній вигляд комірок стовпчика (який об'єкт для відображення інформації знаходиться в комірках стовпчика);

4. DataPropertyName - ім'я, що відображає в стовпці поля;

5. Frozen - фіксація шпальти (стовпець не пересувається при прокручуванні таблиці);

6. HeaderText - текст заголовка стовпця;

7. Width - ширина поля;

8. MaxInputLength - максимально вводиться довжина тексту;

9. MinimumWidth - мінімальна ширина стовпця;

10. ReadOnly - блокування стовпчика для редагування даних;

11. Resizable - дозволяє змінювати ширину стовпця;

12. SortMode - сортування даних в таблиці з цього стовпцю;

13. ToolTipText - підказка для стовпця;

14. Visible - робить стовпець невидимим.

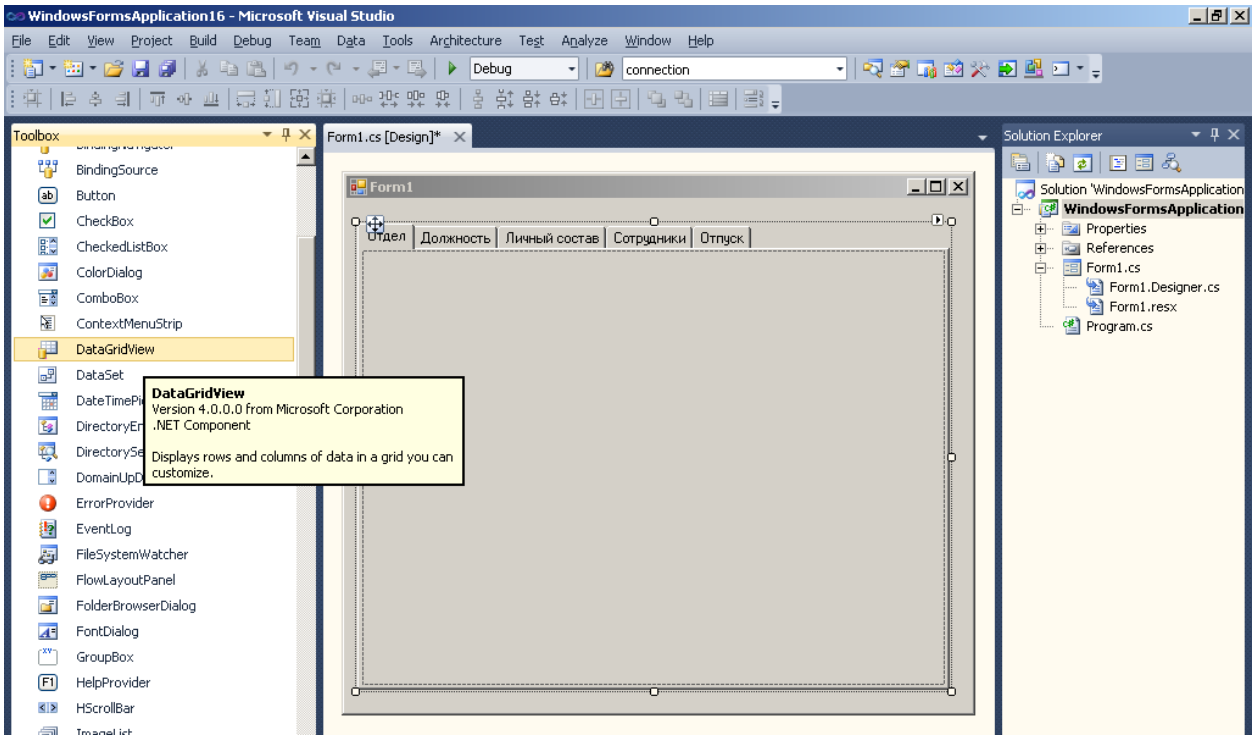


Рисунок 20 – Вибір компонента DataGridView.

У вікні завдань DataGridView Tasks знімаємо галочки з Enable Adding (можливість додавання), Enable Eding (можливість редагування), Enable Deleting (можливість видалення) (рисунок 21).

Клацаємо по Choose Data Source (вибір джерела) і заходимо в Add Prijects Data Source… (рисунок 22).

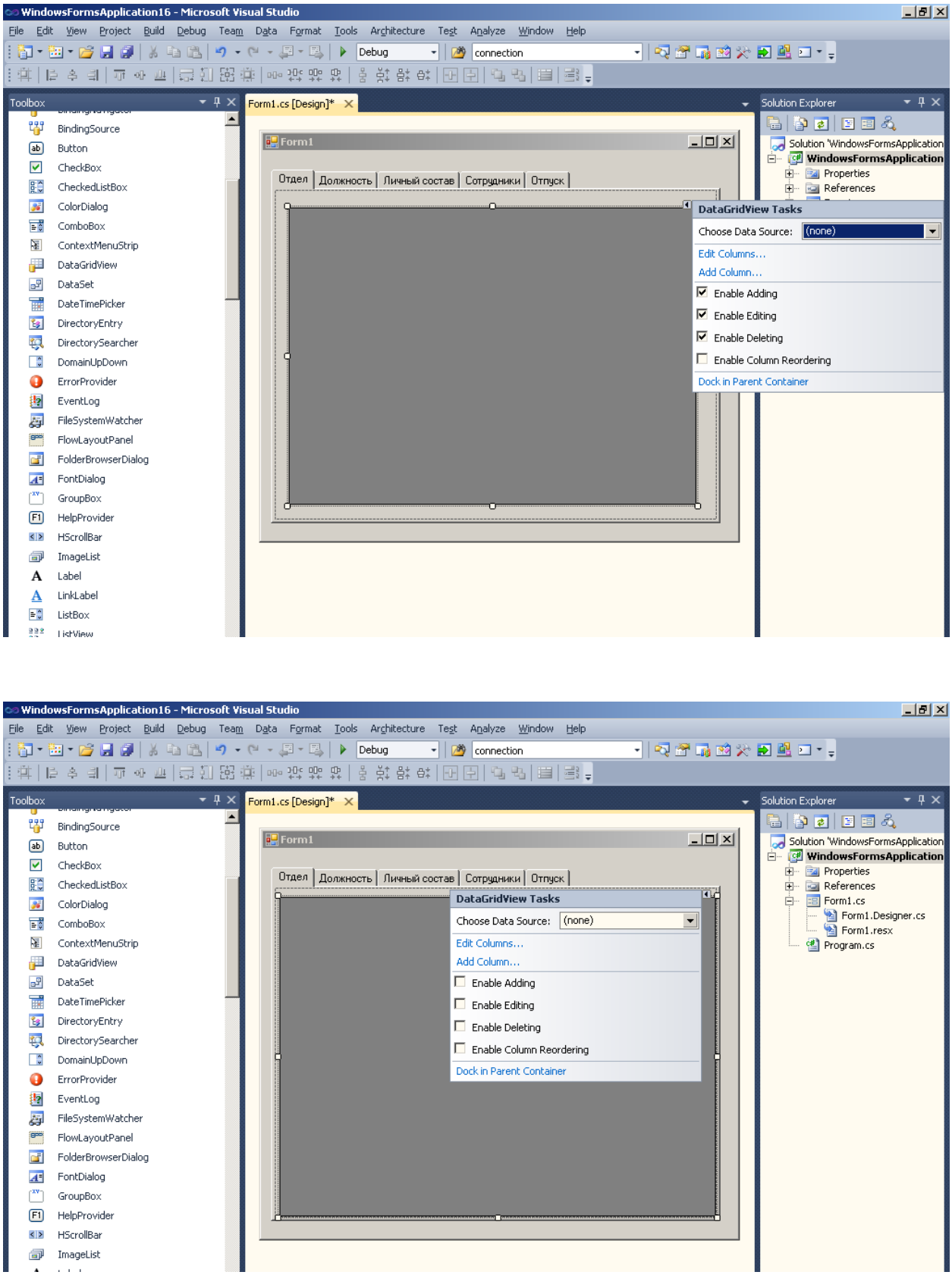


Рисунок 21 – Робота з вікном DataGridView Tasks.

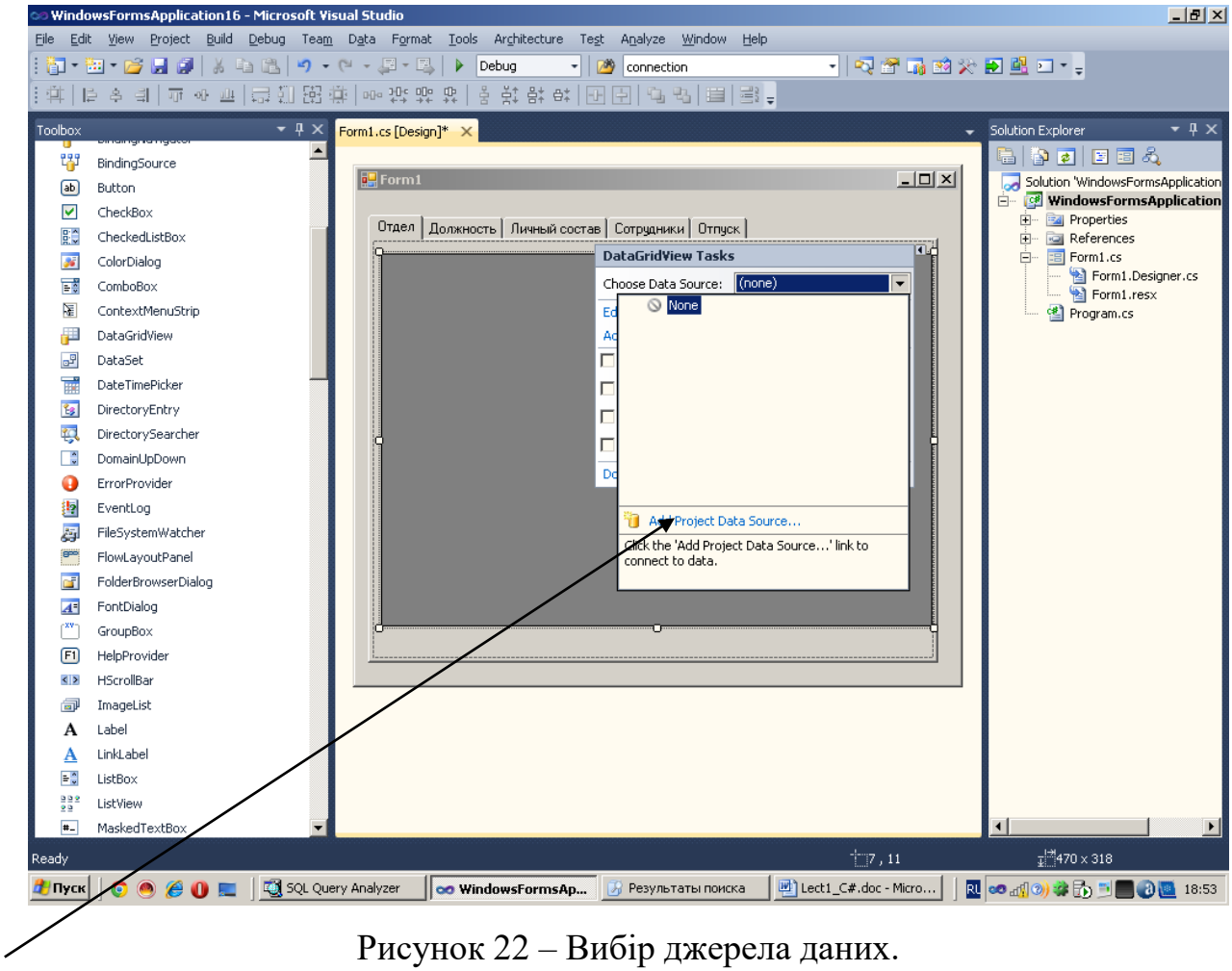


Рисунок 22 – Вибір джерела даних.

Використовуючи вікно Data Source Configuration Wizard як джерело даних вибираємо базу даних (DataBase) і переходимо далі по кнопці Next..

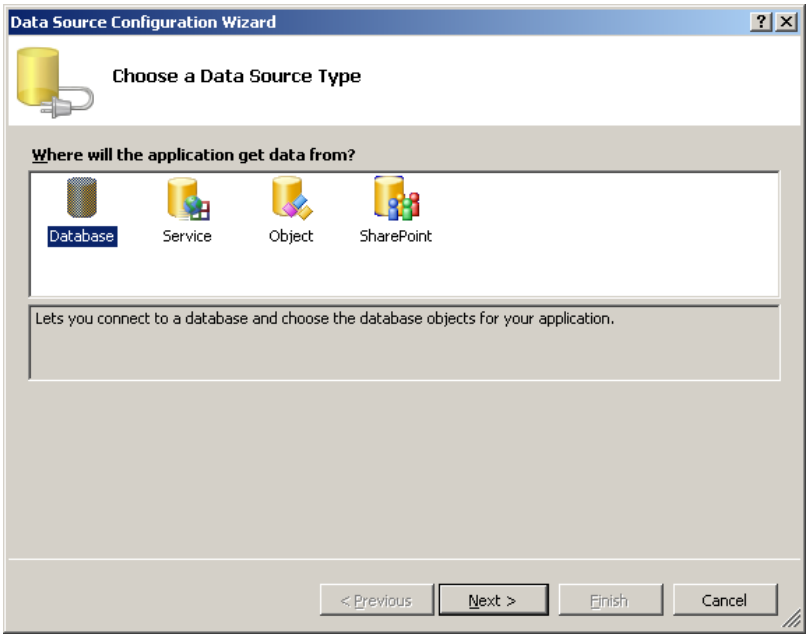


Рисунок 23 – Choose a data source type.

Як модель бази даних потрібно вказати «Набір даних» (Dataset). Натискаємо далі на кнопку Next та переходимо у наступне вікно.

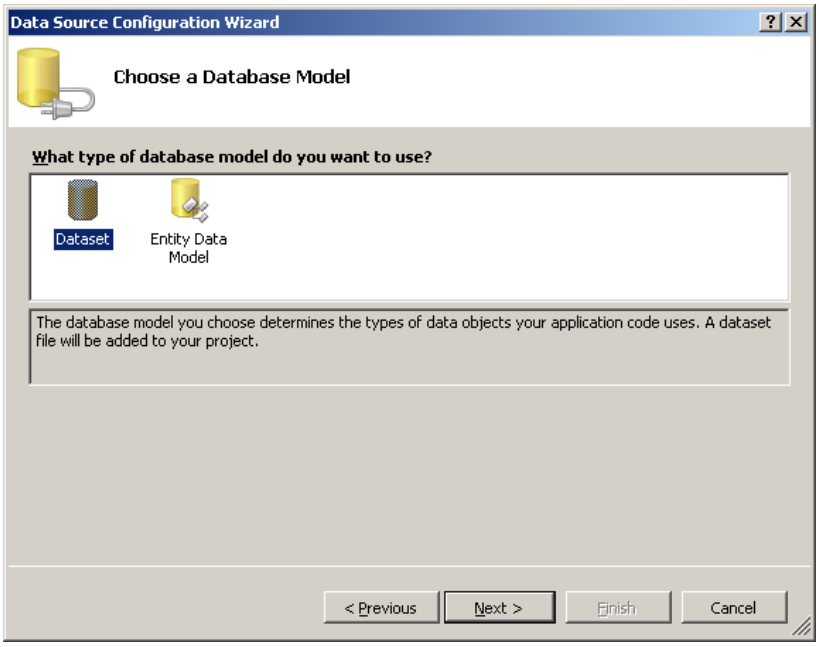


Рисунок 24 – Choose a Database Model.

Потім у вікні Вибору підключення бази даних (Data Source Configuration Wizard) необхідно ввести параметри підключення до бази MS SQL сервера. Переходимо на наступний крок кнопкою Next..

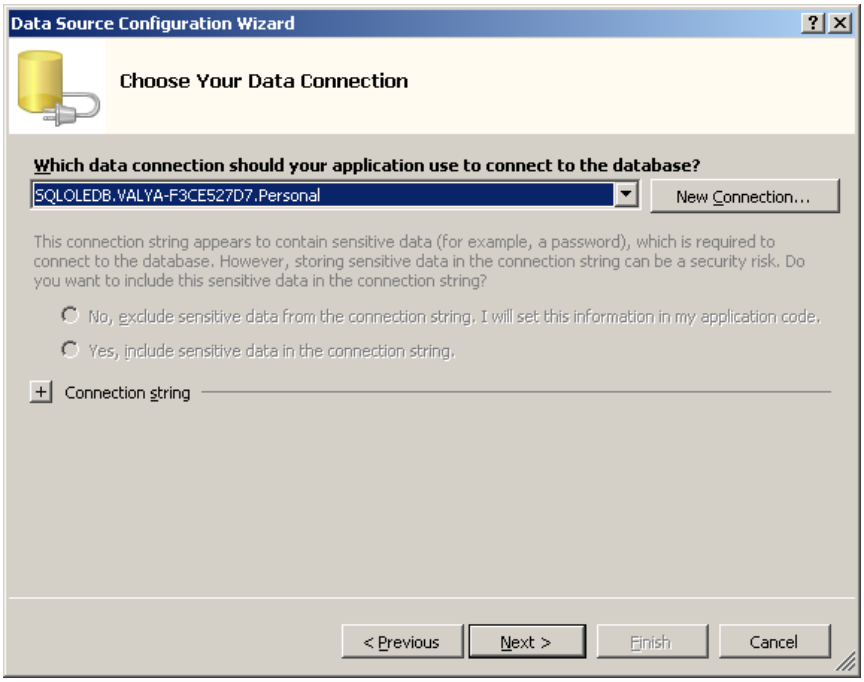


Рисунок 25 – Вибір з’єднання

Ставимо позначку на угоді збереження рядка підключення в файлі конфігурації. натискаємо Next і переходимо на наступний крок.

Клацнувши на кнопку Next, майстер Data Source Configuration відобразить діалогове вікно, в якому він поцікавиться, чи зберігати рядок підключення у файлі конфігурації програми. І хоча в даному підході є багато за і проти, одне з ключових його переваг полягає в тому, що він являє собою простий спосіб відокремити інформацію про підключення від іншої частини коду.

Якщо в діалоговому вікні встановити прапорець Yes, save the connection string as, то середовище Visual Studio збереже рядок підключення і ім'я, яке ви введете, в файлі конфігурації програми для виконуваного проекту. Крім того, вона додасть логічні оператори в код проекту, щоб отримувати і використовувати цей рядок підключення для взаємодії з БД. Файли конфігурації додатків - стандартне місце зберігання інформації про рядку підключення.

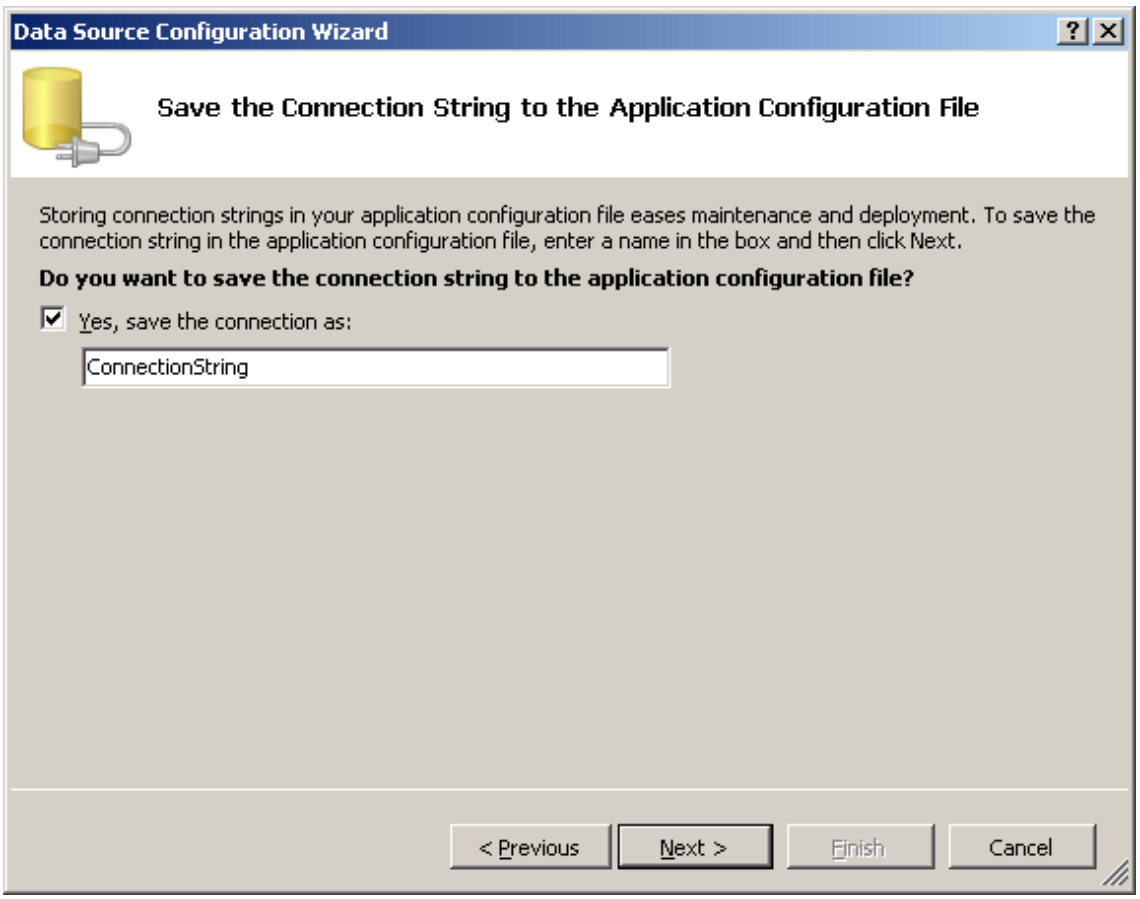
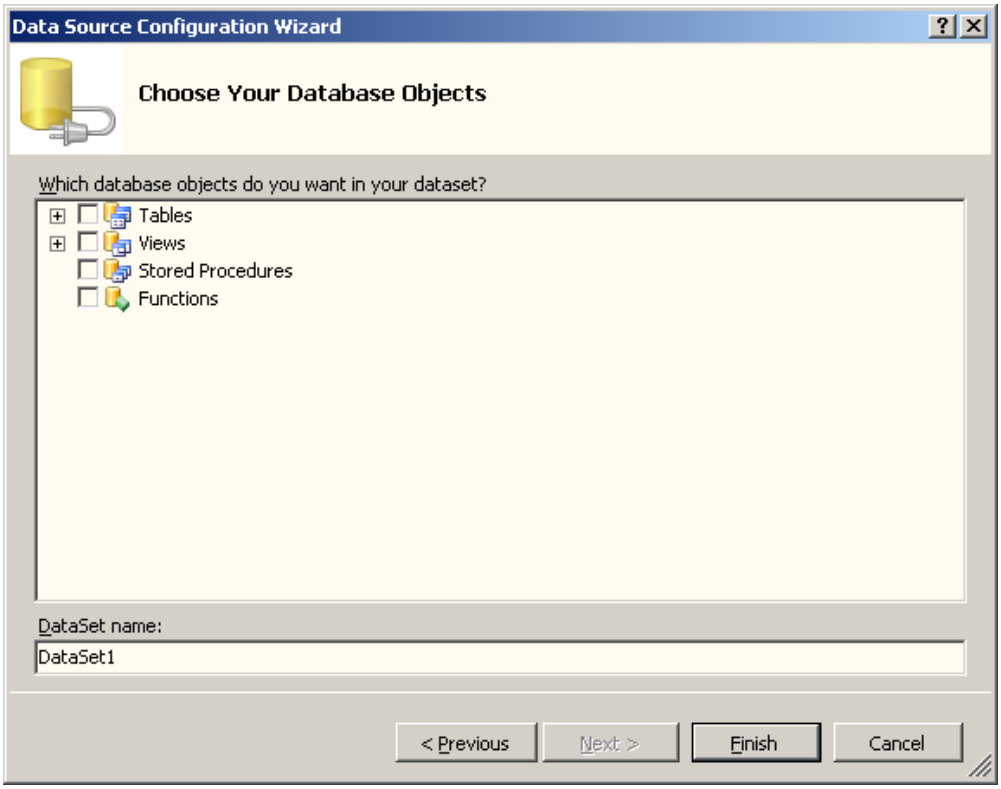


Рисунок 26 – Save the Connection String to the Application File

Тепер у вікні Data Source Configuration Wizard необхідно указати об'єкти-джерела даних (рисунок 27).

Можна вибирати також окремі стовпці в таблицях та подання. У нижній частині діалогового вікна є поле, яке дозволяє вказати ім'я для класу DataSet. За замовчуванням Visual Studio використовує ім'я підключеної в даний момент БД і додає до цього імені слово DataSet.



Після натискання на кнопку Finish робота майстра Data Source Configuration буде завершена і у вікні Solution Explorer (команда View-Solution Explorer) в проекті з'явиться новий запис. Елемент нового проекту має ім'я, вказане в діалоговому вікні та містить визначення класу, відомого як DataSet зі строгим контролем типів.

Об'єктDataSet - це набір даних, в нього входять колекції DataTables і DataRelations. Колекція Data Table містить набори рядків і стовпців з даними,

клас Data Relation дозволяє пов'язувати дані елементів Data Tables. Дані в об'єкті Data Set від'єднані від БД. Після збирання результатів запиту в об'єкті Data Set за допомогою об'єкта Sql Data Adapter з'єднання між БД і об'єктом Data Set припиняється. Об'єкт Sql Data Adapter призначений для роботи з від'єднаними даними.

Можливо, найкраще підтвердження такої його структури - метод Fill. Для виклику цього методу не потрібно «живе» підключення до БД. Коли викликаний метод Fill об'єкта Sql Data Adapter, хто ж не має відкритого підключення до БД, об'єкт відкриває з'єднання, виконує запит до БД, вибирає і заносить результати запиту в об'єкт Data Set і потім закриває з'єднання з БД.

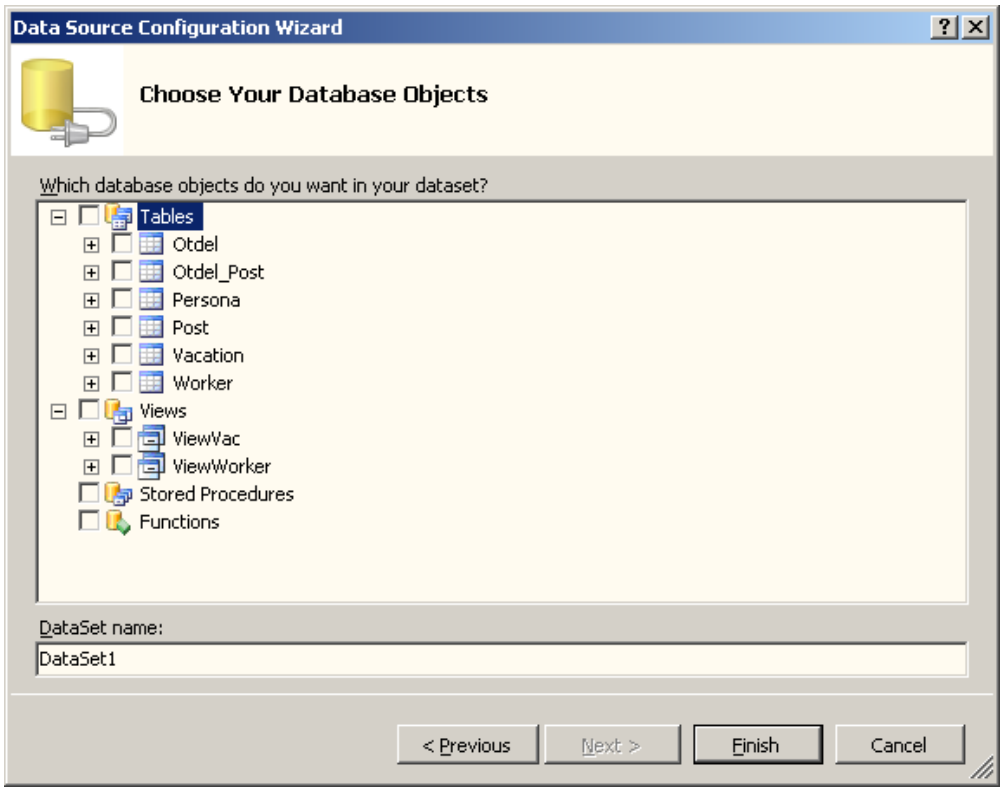


Рисунок 27 – Вибір об’єктів бази даних

Відзначаємо Tables, Views, Stored Procedures (Таблиці, Уявлення і Збережені процедури). В поле DataSet name вказуємо DataSet1.

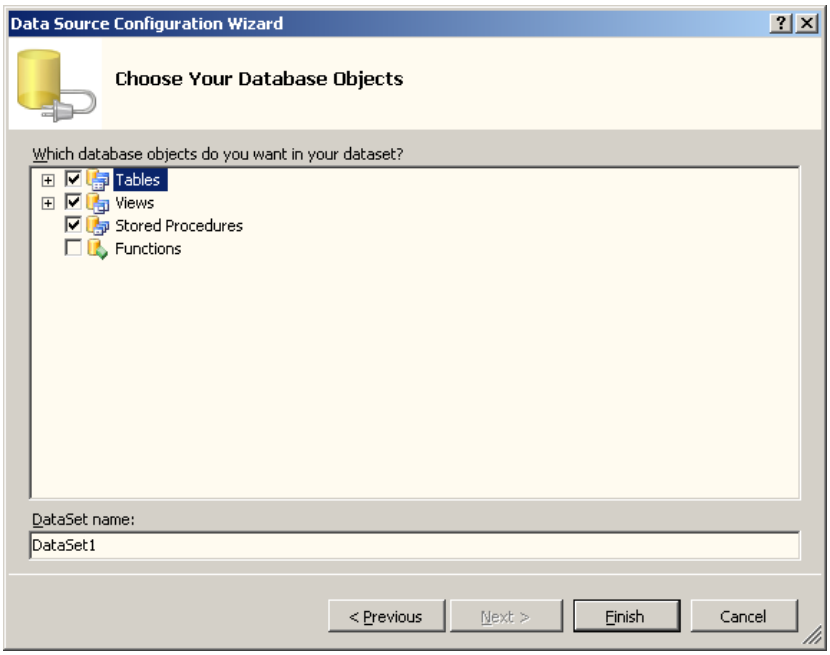


Рисунок 28 – Choose Your database Objects

Тепер в списку Project Data Source -> Data Set відображаються всі наші об'єкти бази даних Personal, як показано на рисунку 29.

Далі кожну сторінку на формі необхідно прив'язати до свого джерела для відображення даних (рисунок 30).

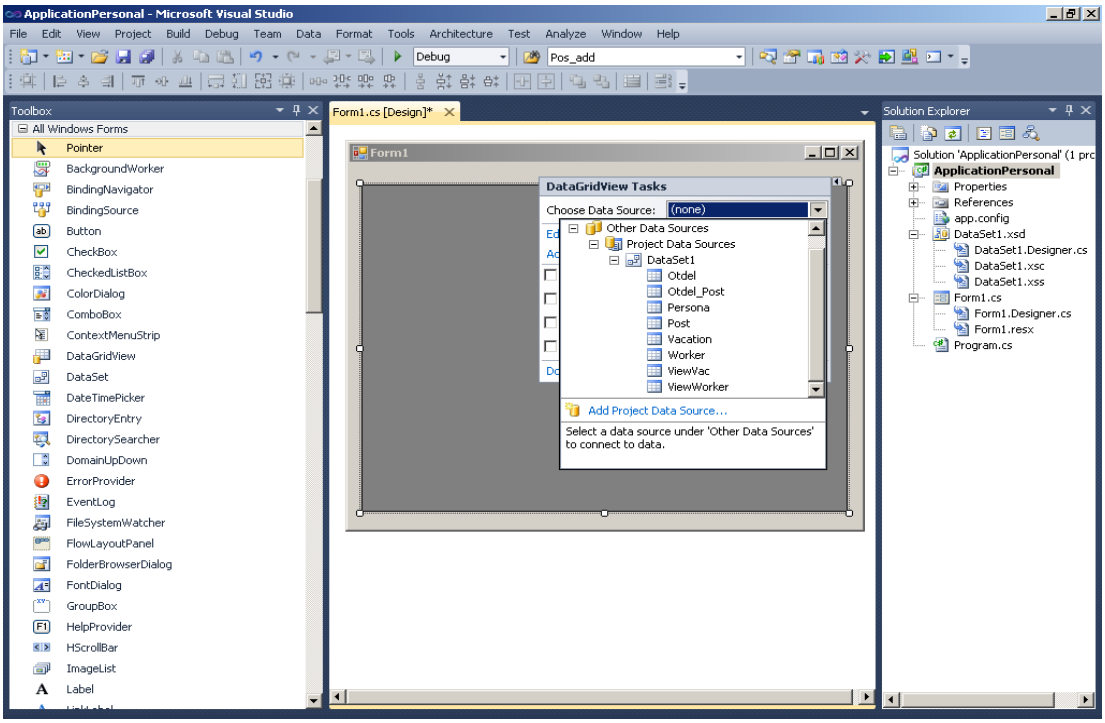


Рисунок 29 – Відображення списку Project data Sources

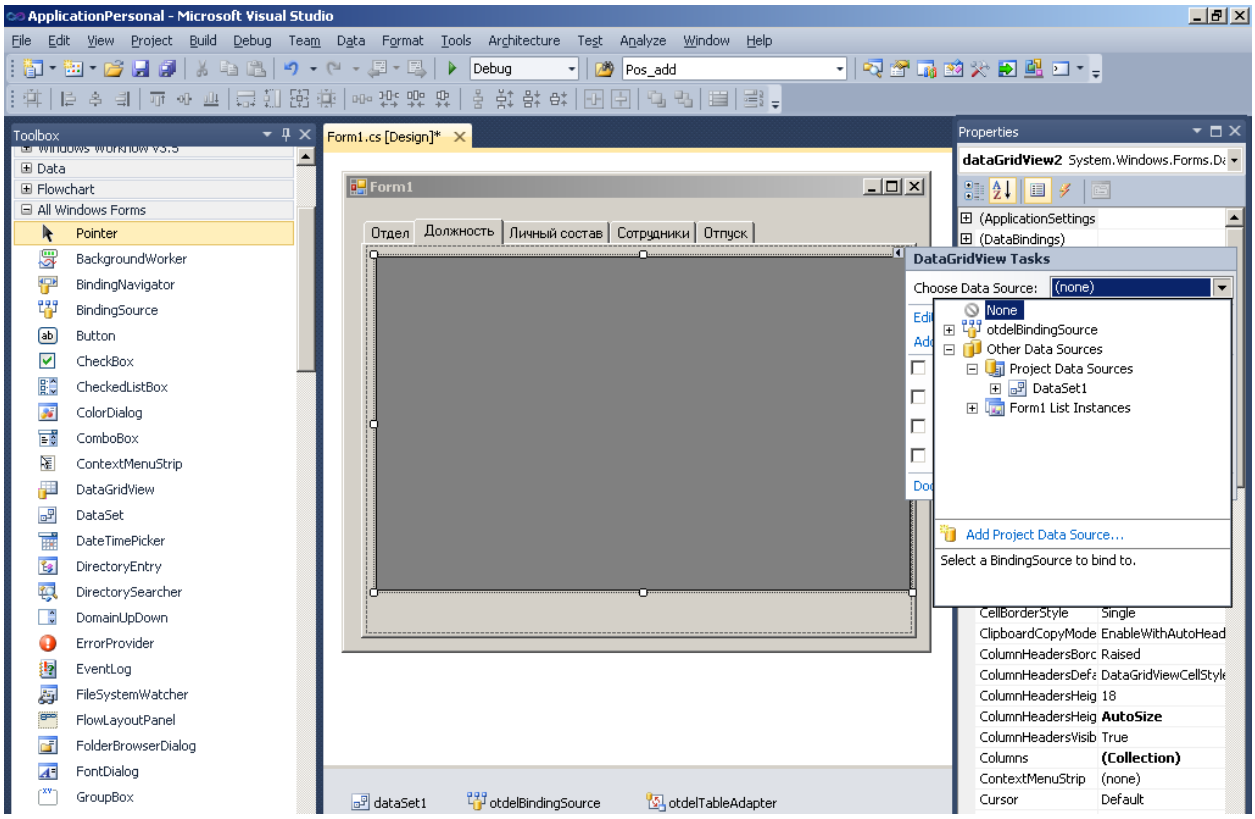


Рисунок 30 – Робота з Choose data Sources

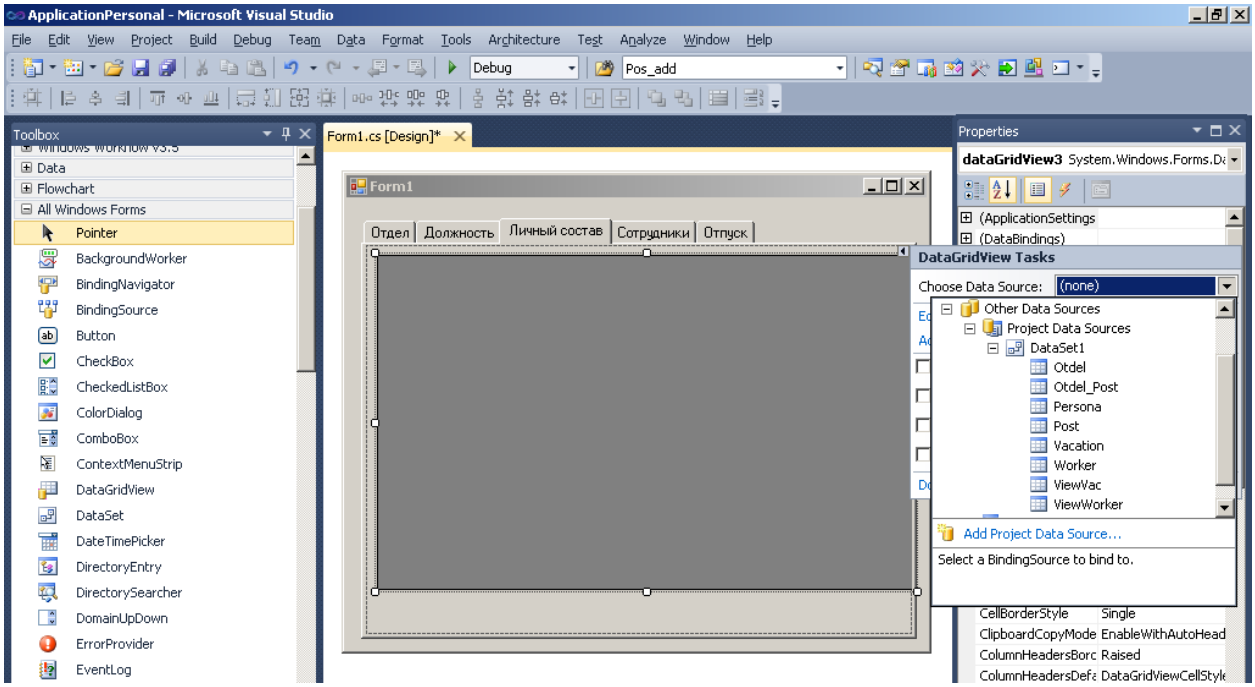


Рисунок 31

За прикладом сторінки Відділ заходимо в властивості правим кліком. У вікні Properties вибираємо властивість Columns (Collections ...).

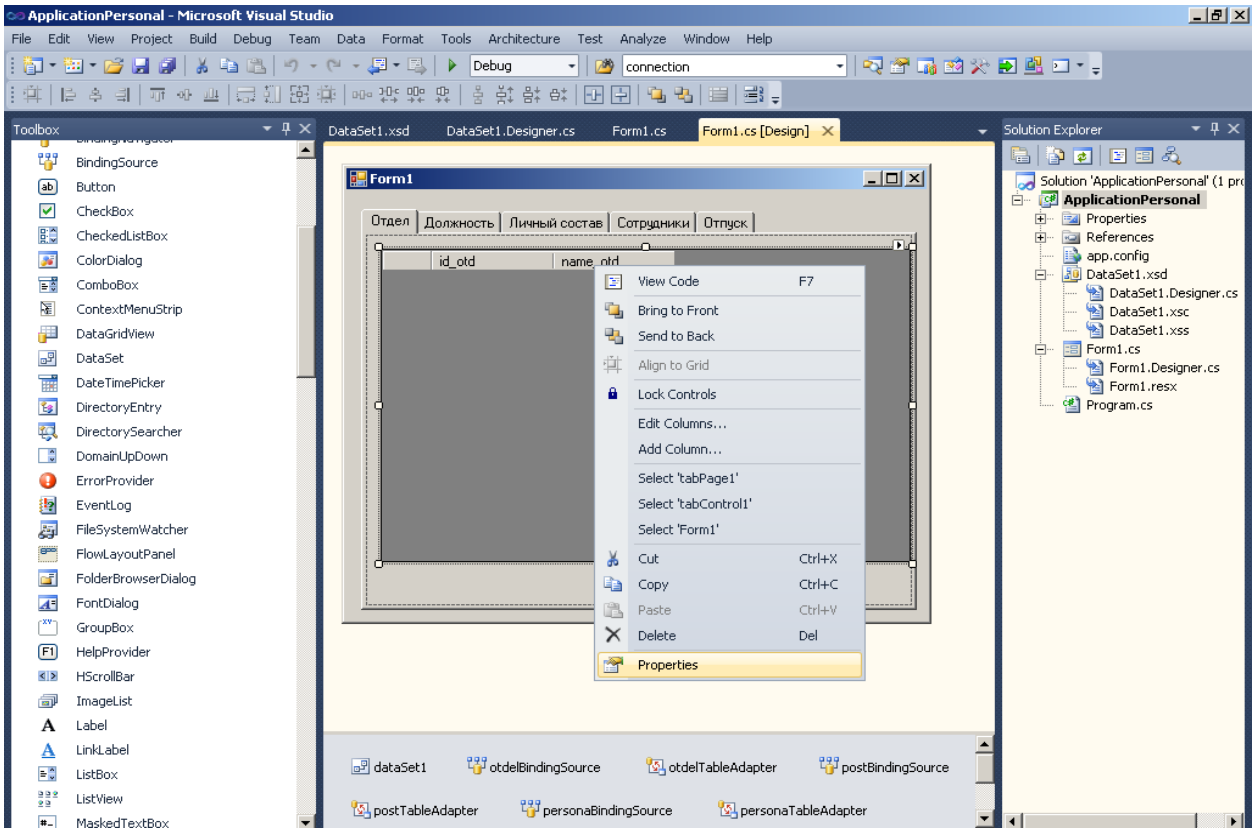


Рисунок 32

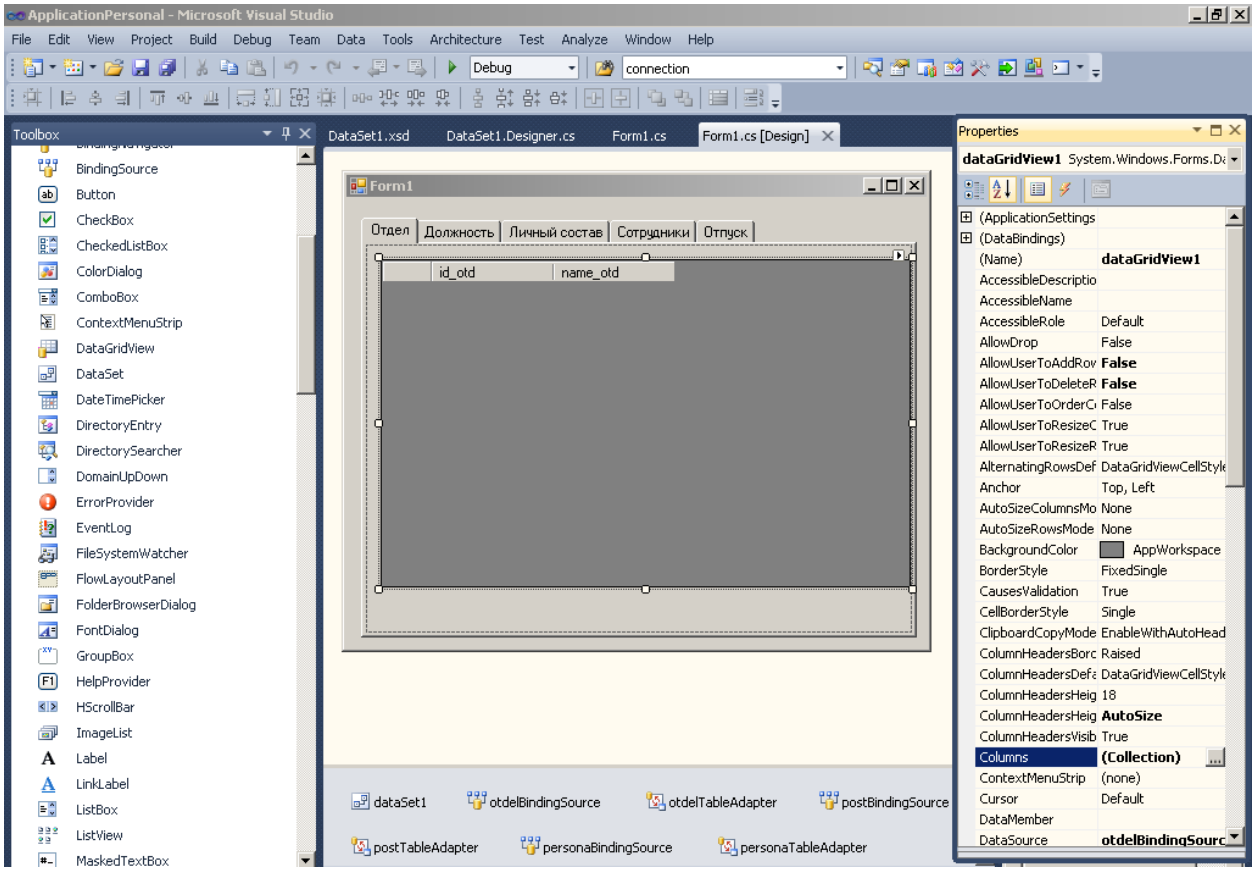


Рисунок 33 – Робота з властивостями dataGridView1

У вікні Edit Columns вибираємо поле таблиці Otdel-> id_otd. Значення поля Name - є ідентифікатором для даного об'єкта в коді програми.

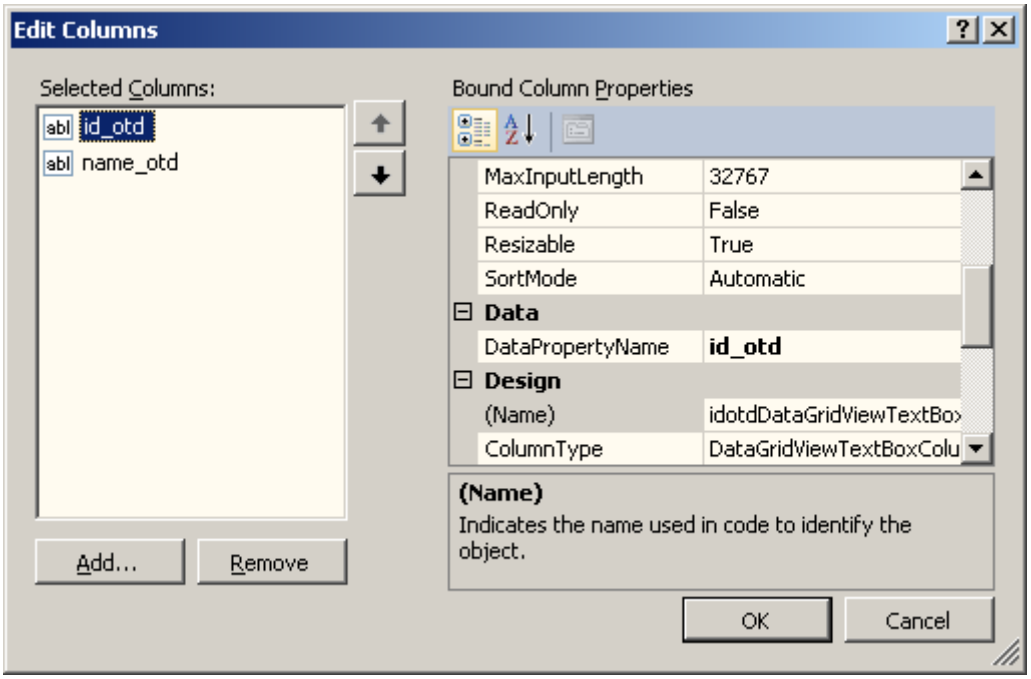


Рисунок 34 – Работа з вікном Edit Columns

У списку Bound Column Properties потрібно змінити властивість Visible->True.

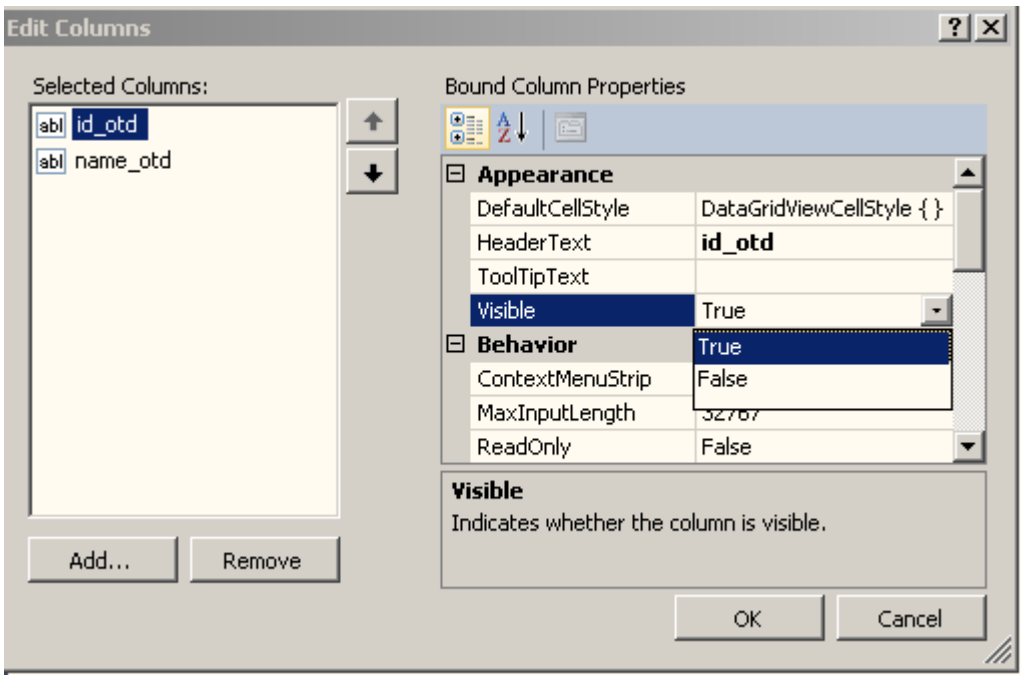


Рисунок 35 – Властивість Visible для поля id_otd

Для зручності користувача заголовок поля name_otd позначимо як «Назва відділу».

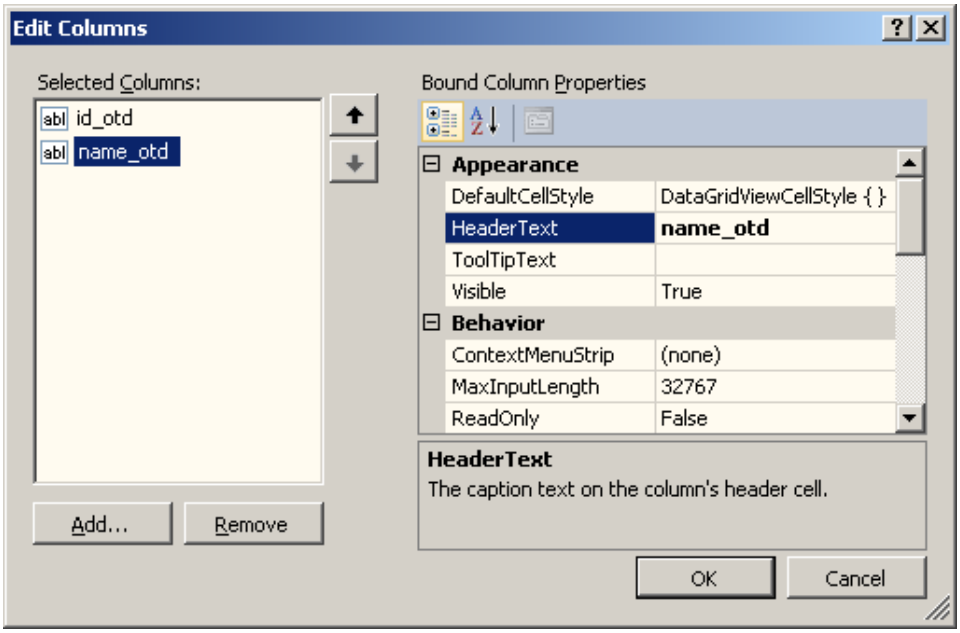


Рисунок 36 – Робота с властивістю HeaderText

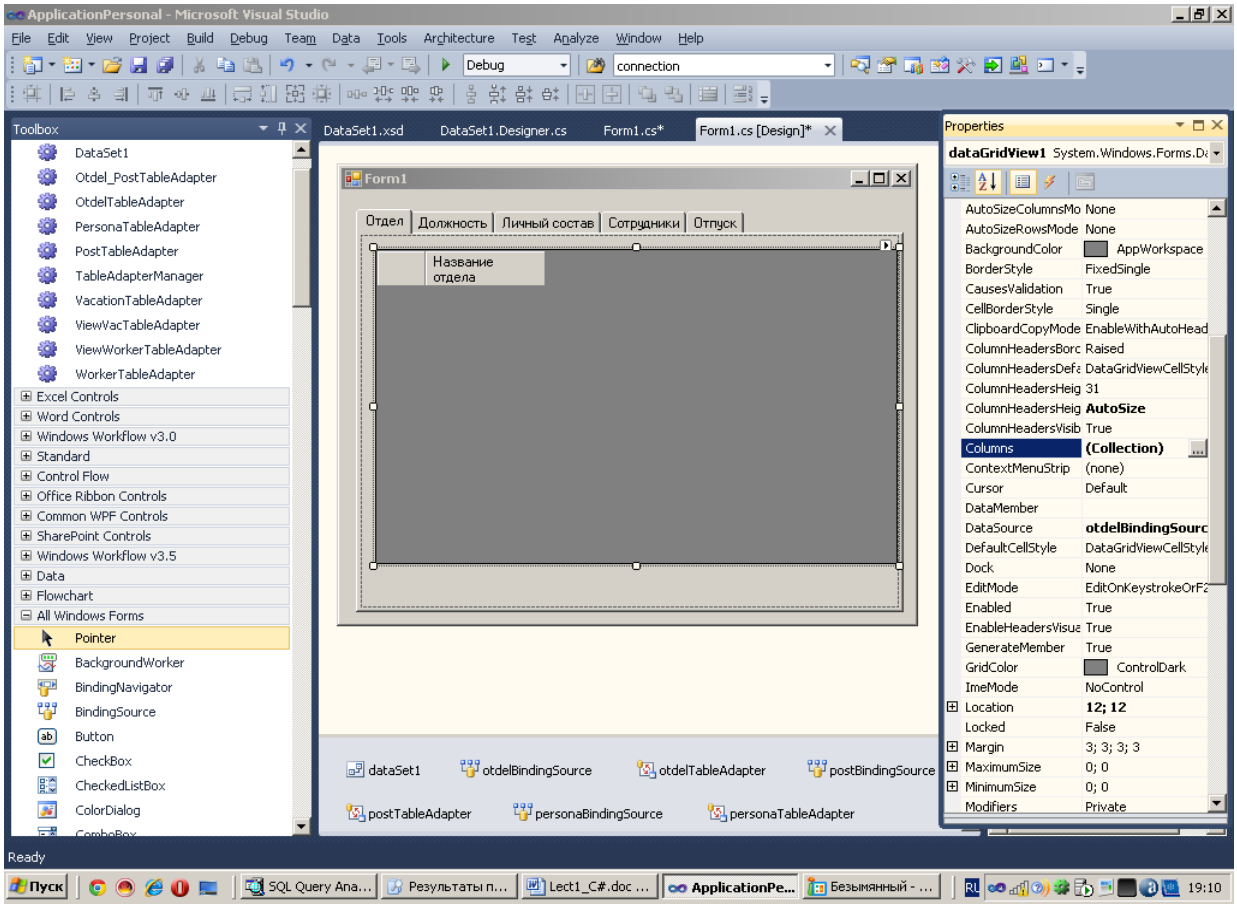


Рисунок 37 – Змінений заголовок поля «Название отдела».

В результаті вміст таблиці буде відображатися в такий спосіб:

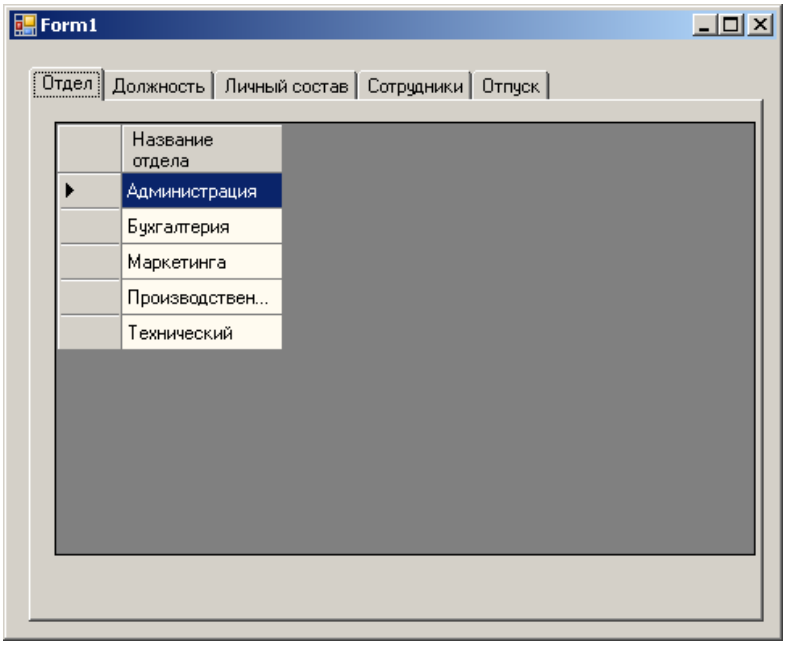


Рисунок 38 – Відображення змісту таблиці Відділ.

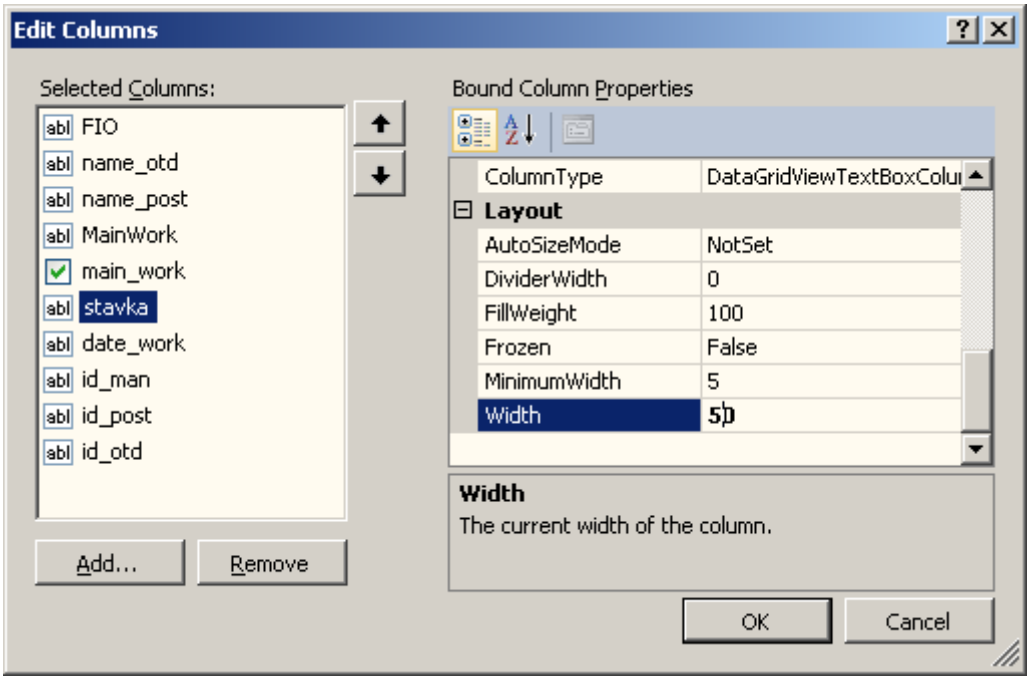


Рисунок 39 – Робота з властивістю Width для атрибуту stavka.

Для атрибуту таблиці Відділ – stavka за допомогою параметру Width вкажемо максимальну кількість введених знаків.

Аналогічно все вищесказане виконуємо щодо інших сторінок (назви стовпців міняємо на російську мову для зручного використання клієнтського застосування).

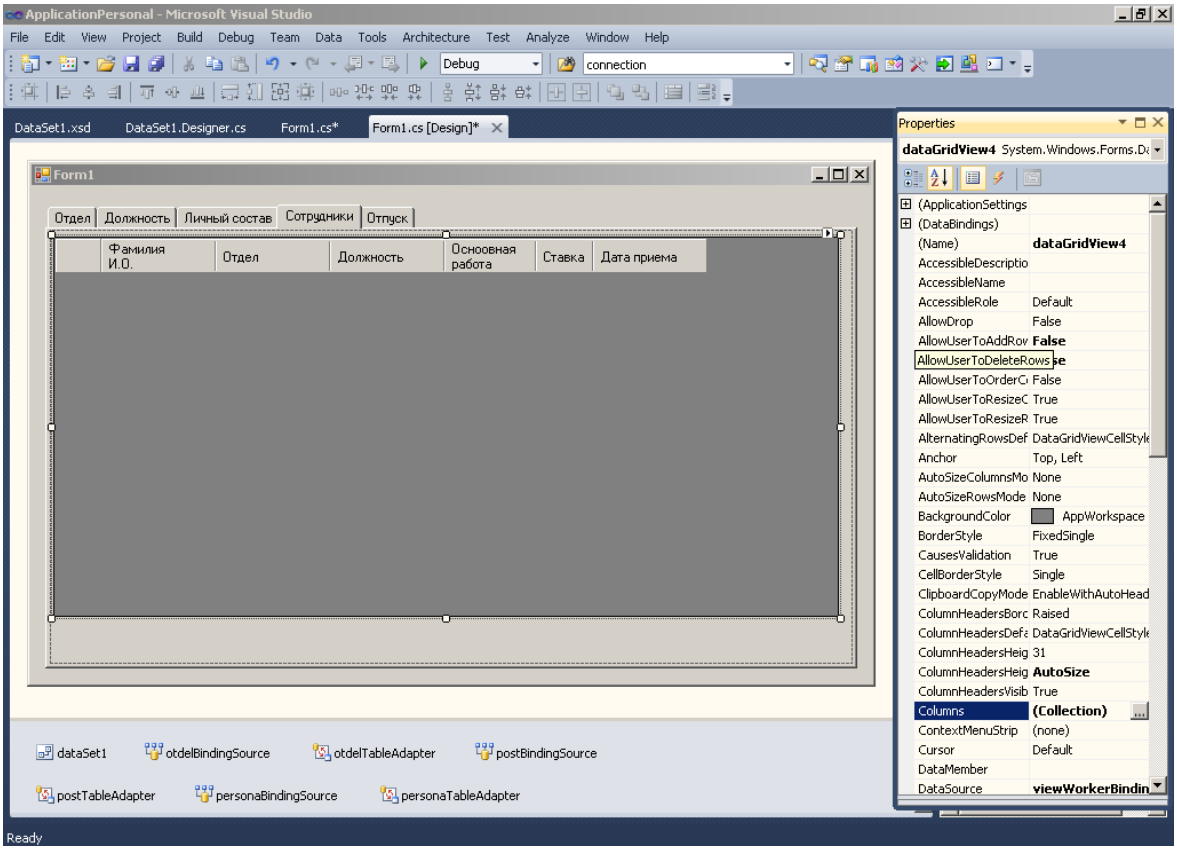
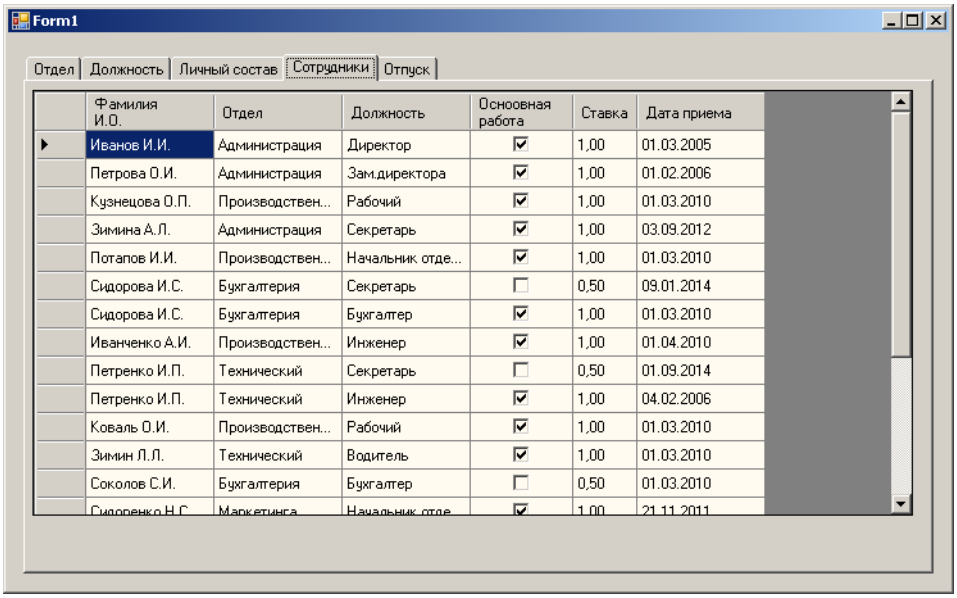


Рисунок 40 – Поля таблиці «Сотрудники».



Фамилия И.О.	Отдел	Должность	Основная работа	Ставка	Дата приема
Иванов И.И.	Администрация	Директор	☑	1,00	01.03.2005
Петрова О.И.	Администрация	Зам.директора	☑	1,00	01.02.2006
Кузнецова О.П.	Производствен...	Рабочий	☑	1,00	01.03.2010
Зимина А.Л.	Администрация	Секретарь	☑	1,00	03.09.2012
Потапов И.И.	Производствен...	Начальник отде...	☑	1,00	01.03.2010
Сидорова И.С.	Бухгалтерия	Секретарь	☐	0,50	09.01.2014
Сидорова И.С.	Бухгалтерия	Бухгалтер	☑	1,00	01.03.2010
Иванченко А.И.	Производствен...	Инженер	☑	1,00	01.04.2010
Петренко И.П.	Технический	Секретарь	☐	0,50	01.09.2014
Петренко И.П.	Технический	Инженер	☑	1,00	04.02.2006
Коваль О.И.	Производствен...	Рабочий	☑	1,00	01.03.2010
Зимин Л.Л.	Технический	Водитель	☑	1,00	01.03.2010
Соколов С.И.	Бухгалтерия	Бухгалтер	☐	0,50	01.03.2010
Сидоренко Н.С.	Маркетинга	Начальник отде...	☑	1,00	21.11.2011

Рисунок 41 – Відображення даних таблиці на формі.

Завдання

Кожен студент повинен підключить свою базу даних (з попередньої лабораторної роботи) до клієнтського додатку у Visual Studio 2010 згідно з наведеним вище послідовним алгоритмом. Результат повинен бути аналогічним рисунку 41.

Контрольні питання

1. Дайте визначення властивостям об’єкта. Наведіть приклади.
2. Що таке події та методи об’єкту?
3. На які класи поділяють об’єкти?
4. Які групи клієнтських форм ви знаєте?
5. Які види дизайну форм ви знаєте?

Лабораторная работа№8 «Створення додаткових форм для редагування даних таблиць»

Мета: навчитися поширювати клієнтський додаток додатковими формами до таблиць для редагування даних**.**

Практична частина

У 7-й лабораторній роботі розробляється додаток для перегляду вмісту таблиць бази даних. При цьому студенти виконують підключення до бази даних (Data Connection), а потім на всіх закладках TabControl встановлюють інструмент DataGridView для перегляду даних і вибирають для нього джерело даних – таблицю або уявлення.

Вибрані джерела даних можна побачити в програмному вікні - під формою розробляється, як видно на рисунку 1.

Тут:

1. dataSet1 - набір даних (у вигляді об'єкта класу DataSet), лічених безпосередньо з сервера;
2. viewPosBindingSource і viewTeachBindingSource - об'єкти класу BindingSource, призначені для зв'язку елементів форми з набором даних (джерела даних для DataGridView);
3. viewPosTableAdapter і viewTeachTableAdapter - об'єкти (адаптери таблиць), класи яких описані у файлі DataSet1.xsd: вони використовуються для завантаження даних в DataSet з сервера і маніпуляції ними (можливе використання адаптерів для поновлення даних на сервері).

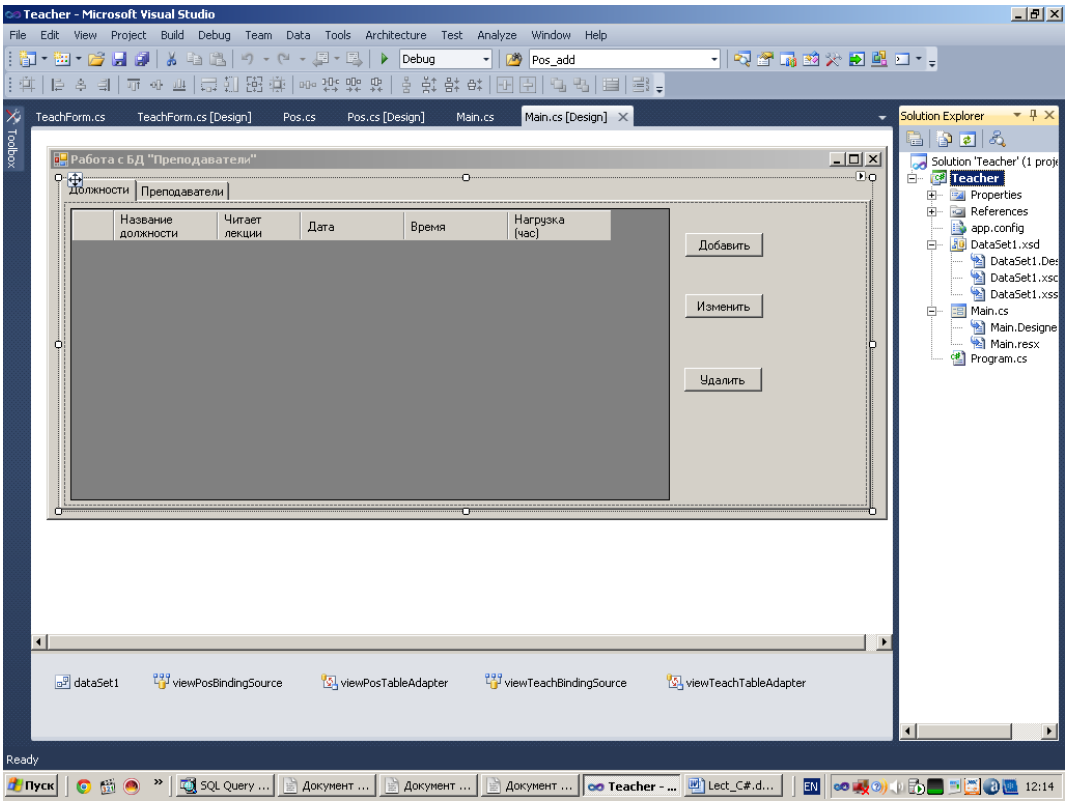


Рисунок 1 – Видиме відображення джерел даних програми

Також генерується метод, який оновлює вміст джерел даних:

```
private void Form1_Load(object sender, EventArgs e)
        {
            // TODO: This line of code loads data into the
'dataSet1.ViewTeach' table. You can move, or remove it, as needed.
            this.viewTeachTableAdapter.Fill(this.dataSet1.ViewTeach);
            // TODO: This line of code loads data into the 'dataSet1.ViewPos'
table. You can move, or remove it, as needed.
            this.viewPosTableAdapter.Fill(this.dataSet1.ViewPos);
        }
```

Окремі рядки цього методу потрібно використовувати після додавання, зміни або видалення записів у відповідних таблицях. Наприклад, команда:

```
this.viewTeachTableAdapter.Fill(this.dataSet1.ViewTeach);
```

оновлює (перечитує) дані в джерелі даних, пов'язаному з поданням ViewTeach, створеному для перегляду таблиці Teachers. Цю команду в програмі потрібно застосовувати після додавання, зміни або видалення записів в таблиці Teachers.

Під формою додатка відображаються не всі джерела даних, а тільки ті, які використані в інструментах DataGridView. Всі підключення джерела даних (які можна використовувати в програмі) можна побачити, якщо клацнути по імені файлу DataSet1.xsd (якщо його не перейменовували) в правій частині екрана в віконці "Solution Explorer" (рис.2).

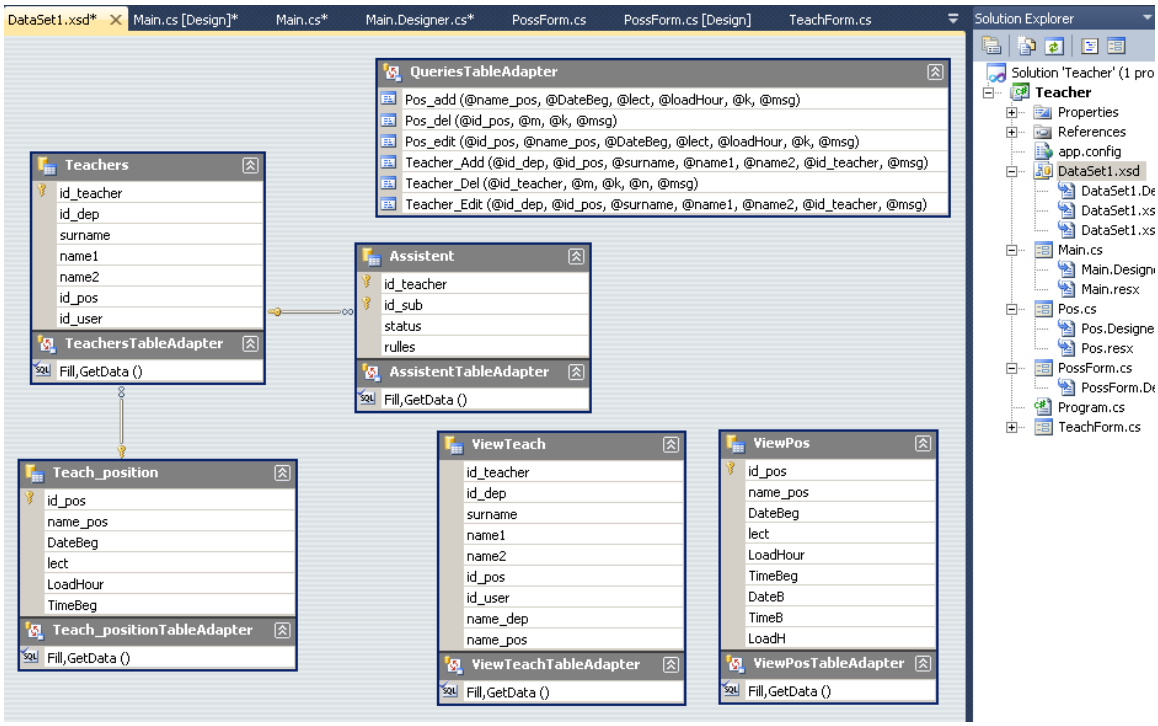


Рисунок 2 – Перегляд джерел даних додатка

На малюнку 2 видно, що в якості джерел даних вказані 3 таблиці: Teachers, Assistent, Teach_position, - 2 подання: - ViewTeach, ViewPos, - і 6 процедур: - Pos_add, Pos_del, Pos_edit, Teacher_add, Teacher_del, Teacher_edit.

При установці інструментів DataGridView потрібно було вибирати режим роботи цих інструментів без можливості додавання, зміни і видалення записів прямо в цих табличних компонентах.

Завдання

Для можливості додавання і зміни записів в таблицях потрібно використовувати додаткові форми, на яких користувач повинен писати або вибирати дані. Також користувач повинен мати можливість видалити обрану їм запис в таблиці. Зазначені дії можна виконувати або за допомогою пунктів меню, або за допомогою кнопок. Розглянемо спосіб з кнопками.

На кожну вкладку компонента TabControl необхідно встановити 3 кнопки (Button). Після установки кнопок потрібно змінити у них властивості (Properties): (Name) і Text. У властивості (Name) вказується ім'я компонента, як змінної програми, тому слід давати компонентам осмислені імена, щоб легше писати і читати програму.

Наприклад, кнопка, при натисканні якої викликається форма для додавання запису в таблицю Teachers, може називатися addTeachersButton (AddTeacherButt, AddTeachButt, AddTeachButton і т.п.).

Як властивості Text кнопок можна використовувати слова «Додати», «Змінити», «Видалити», а можна їх уточнити: «Додати викладача», «Змінити дані викладача», «Видалити викладача» (рис.3).

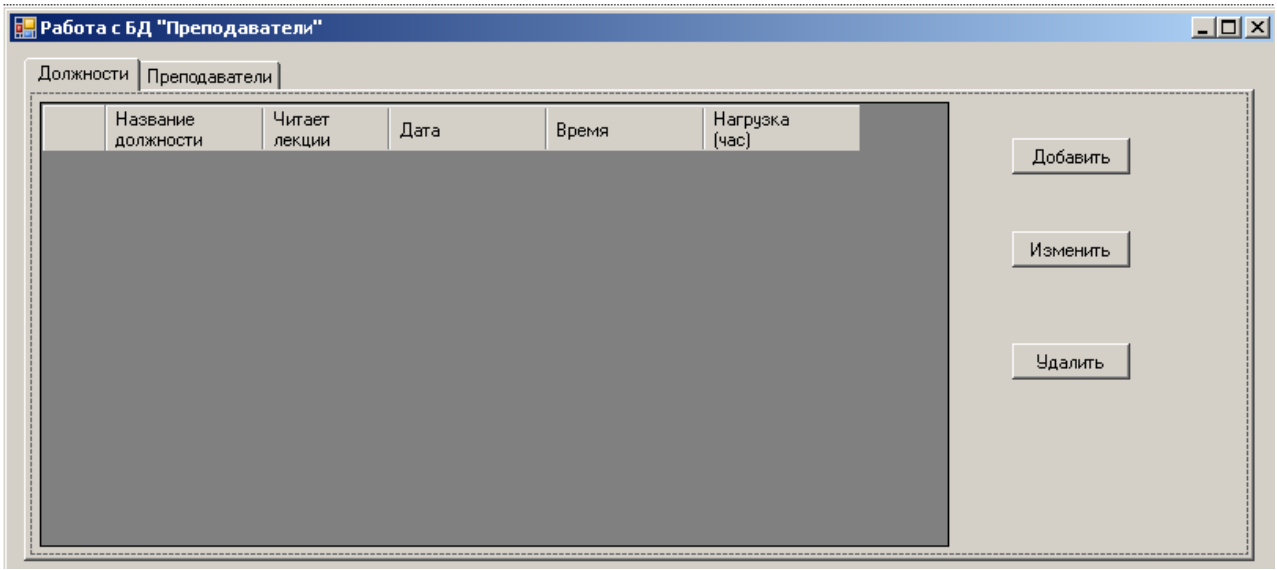


Рисунок 3 – Кнопки для маніпулювання даними в таблиці Teachers

Для маніпулювання даними потрібно використовувати створені раніше (лабораторна робота №3) збережені процедури. У середовищі Visual Studio можна не тільки переглянути перелік використовуваних процедур зі списком параметрів (рис.2), але також можна визначити їх типи, і уточнити, які параметри є вихідними.

Для цього потрібно в контекстному меню збереженої процедури вибрати пункт Properties (Властивості), серед властивостей вибрати Parameters (Параметри), і натиснути на три крапки біля слова (Collection) (рис.4).

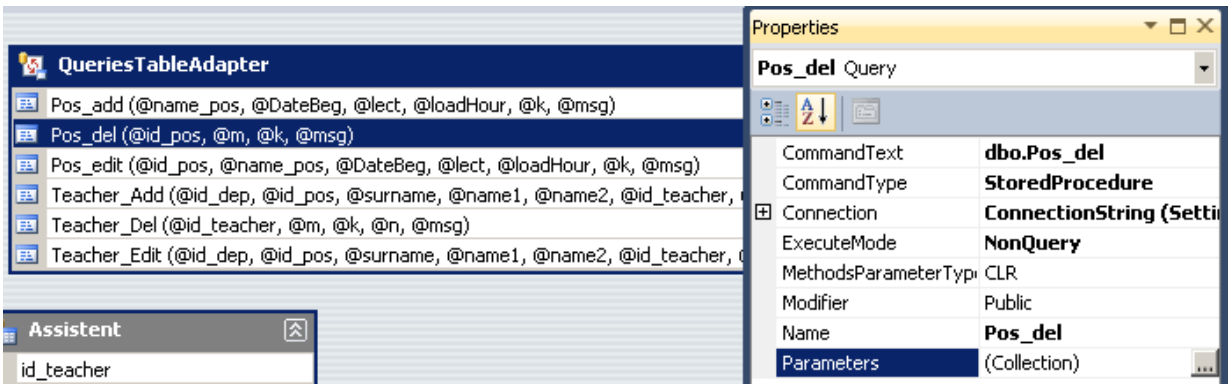


Рисунок 4 – Перегляд властивостей збереженої процедури

Відкриється вікно редактора колекції параметрів, в якому зліва відображаються всі параметри процедури, що, а праворуч - властивості виділеного параметра. Для всіх типів параметрів основною властивістю є властивість Provider Type - це тип, який вказується для параметра при ініціалізації в програмі збереженої процедури.

Для цілих типів даних важливою властивістю є також властивість DbType - це тип, до якого потрібно конвертувати значення, що привласнюються параметру, наприклад, при введенні значення за допомогою компонента TextBox. Правда, для типу SQL Server varchar чомусь вказується тип Char як Provider Type (рис.7), хоча при ініціалізації процедури, що потрібно вказувати тип VarChar (рис.10).

Для символьних типів також важливо властивість Size - довжина рядка (рис.7).

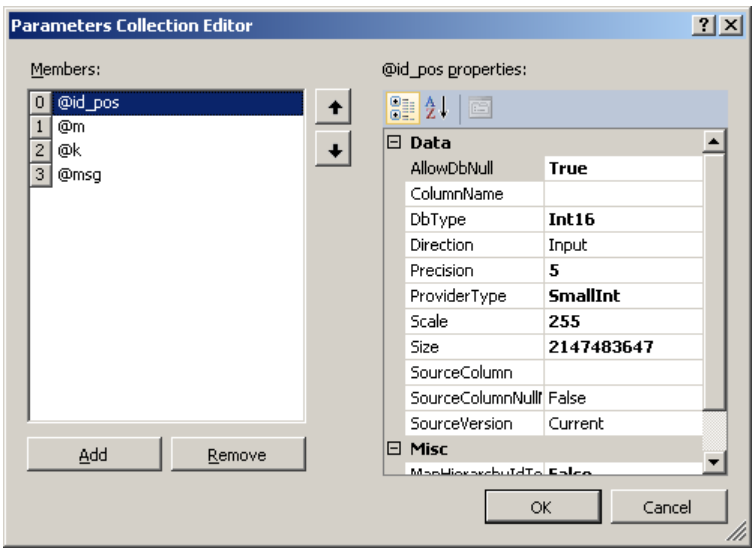


Рисунок 5 – Властивості параметра, що має тип smallint в SQL Server (вхідного)

За замовчуванням вважається, що параметри процедури є вхідними параметрами - для них властивість Direction = Input. Для вихідних параметрів властивість Direction = InputOutput (рис.6,7). Це властивість обов'язково потрібно вказувати при ініціалізації збереженої процедури.

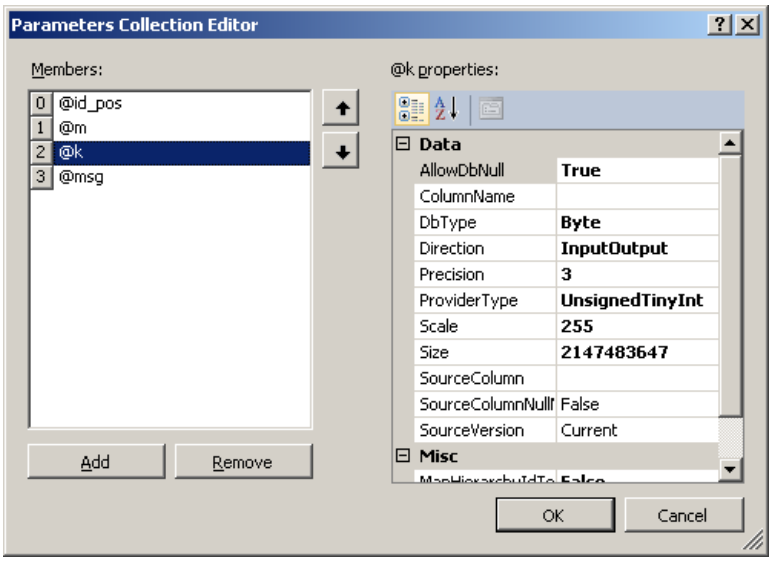


Рисунок 6 – Властивості параметра, що має тип tinylint в SQL Server (вихідного)

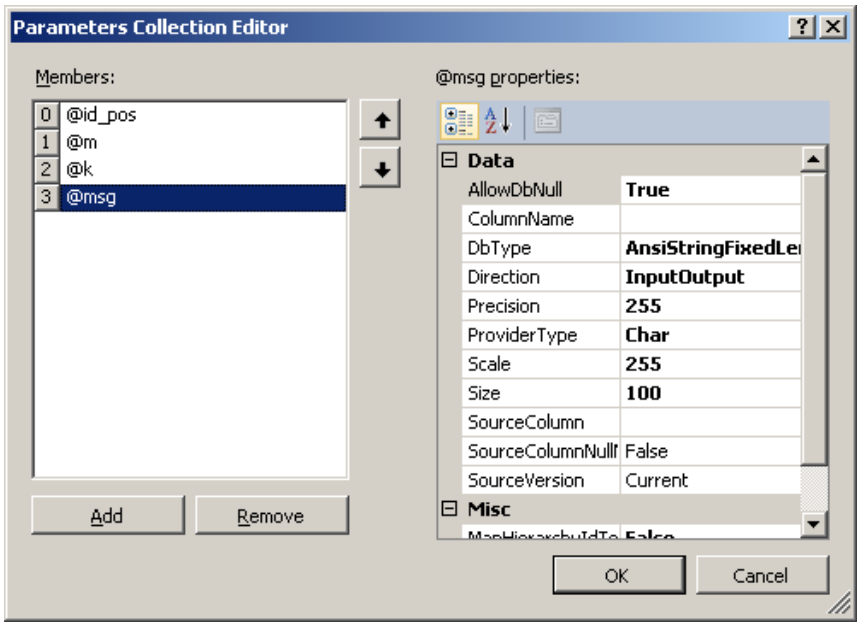


Рисунок 7 – Властивості параметра, що має тип varchar (100) в SQL Server (вихідного)

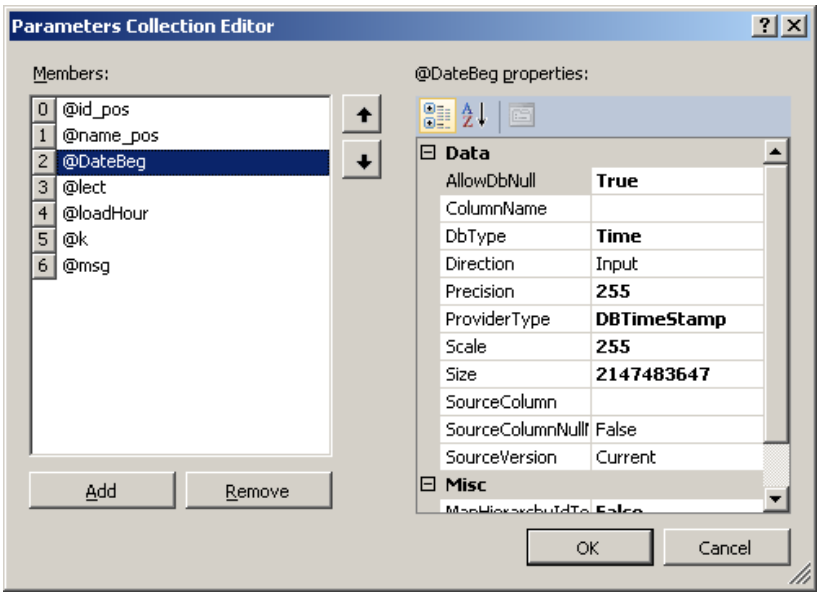


Рисунок 8 – Властивості параметра, що має тип datetime в SQL Server (вхідного)

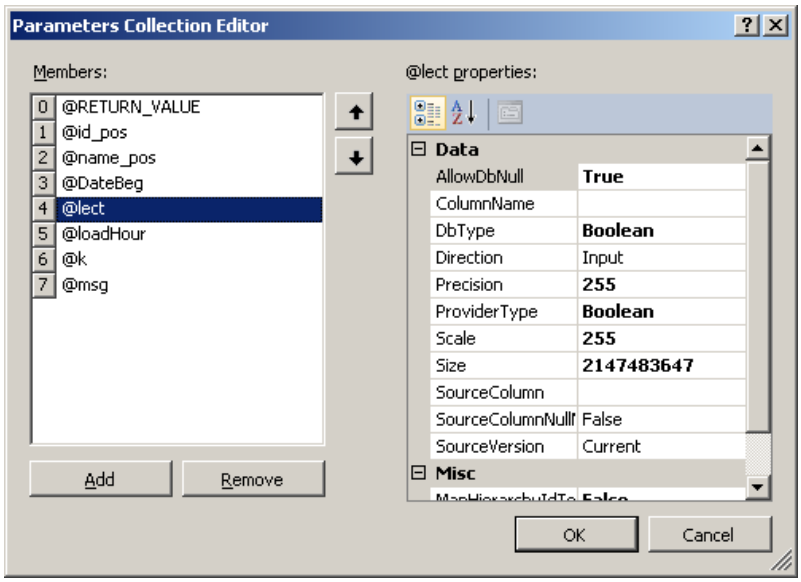


Рисунок 9 – Властивості параметра, що має тип bit в SQL Server (вхідного)

Оскільки при підключенні бази даних MS SQL Server 2000 використовується Data Provider for OLE DB, то для виклику збережених процедур використовуються об'єкти класу OleDbCommand. Типи параметрів вказуються через перерахування OleDbType (рис.10).

Об'єкти класу OleDbCommand (для простоти будемо називати їх просто командами), які використовуються для виклику збережених процедур - описуються як приватні (private) змінні класу форми перед конструктором форми (рис.10).

У конструкторі виконується ініціалізація цих команд. Можна весь код ініціалізації всіх команд помістити прямо в конструктор, але тоді він стане погано читаються. Тому краще ініціалізацію кожної команди описати як окремий метод, і в конструкторі викликати ці методи для всіх команд даного класу форми.

У прикладі на рис.10 на формі MainForm стоять дві кнопки для видалення записів з таблиць: кнопка DelPosButton і кнопка DelTeachButton; кожен з обробників події натискання на кнопку викликає свою збережену процедуру: Pos_del (команда DelPosCmd), Teacher_del (команда DelTeachCmd). Метод init_command_pos () ініціалізує команду DelPosCmd, а метод init_command_teach () - команду DelTeachCmd.

Teacher.MainForm | DelPosButton_Click(object sender, EventArgs e)

```
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace Teacher
{
    public partial class MainForm : Form
    {
        private System.Data.OleDb.OleDbCommand DelPosCmd;
        private System.Data.OleDb.OleDbCommand DelTeachCmd;
        public MainForm()
        {
            InitializeComponent();
            init_commands_pos();
            init_commands_teach();
        }

        private void init_commands_pos()
        {
            DelPosCmd = new System.Data.OleDb.OleDbCommand();
            DelPosCmd.Connection = this.viewPosTableAdapter.Connection;
            DelPosCmd.CommandText = "Pos_del";
            DelPosCmd.CommandType = System.Data.CommandType.StoredProcedure;
            DelPosCmd.Parameters.Add("@id_pos", System.Data.OleDb.OleDbType.SmallInt);
            DelPosCmd.Parameters.Add("@m", System.Data.OleDb.OleDbType.UnsignedTinyInt);
            DelPosCmd.Parameters.Add("@k", System.Data.OleDb.OleDbType.UnsignedTinyInt).Direction =
                                                    System.Data.ParameterDirection.InputOutput;
            DelPosCmd.Parameters.Add("@msg", System.Data.OleDb.OleDbType.VarChar, 100).Direction =
                                                    System.Data.ParameterDirection.InputOutput;
        }
```

Рисунок 10 – Опис і иініціалізація зберігаємої процедури

Розглянемо докладніше код програмі на малюнку 10. У перших рядках опису класу MainForm описуються його локальні змінні (приватні властивості). Такими в даному випадку є об'єкти класу OleDbCommand, який призначений для виконання команд SQL (запитів) і збережених процедур - DelPosCmd і DelTeachCmd:.

```
private System.Data.OleDb.OleDbCommand DelPosCmd;
```

У конструкторі викликається стандартний метод InitializeComponent (), а потім - методи для ініціалізації команд (використовуваних для виклику збережених процедур):

```
public MainForm()
        {
            InitializeComponent();
            init_commands_pos();
            init_commands_teach();
        }
```

Далі описуються власне методи ініціалізації команд. Наприклад, метод `init_commands_pos()`:

```
private void init_commands_pos()
        {
            DelPosCmd = new System.Data.OleDb.OleDbCommand();
            DelPosCmd.Connection =
this.viewPosTableAdapter.Connection;
            DelPosCmd.CommandText = "Pos_del";
```

```
            DelPosCmd.CommandType =
System.Data.CommandType.StoredProcedure;
            DelPosCmd.Parameters.Add("@id_pos",
System.Data.OleDb.OleDbType.SmallInt);
            DelPosCmd.Parameters.Add("@m",
System.Data.OleDb.OleDbType.UnsignedTinyInt);
            DelPosCmd.Parameters.Add("@k",
              System.Data.OleDb.OleDbType.UnsignedTinyInt).Direction =
              System.Data.ParameterDirection.InputOutput;
            DelPosCmd.Parameters.Add("@msg",
               System.Data.OleDb.OleDbType.VarChar, 100).Direction =
               System.Data.ParameterDirection.InputOutput;
        }
```

У першому рядку методу створюється об'єкт DelPosCmd типу OleDbCommand. У другому рядку инициализируется його властивість Connection - так визначається використовується цією командою підключення до сервера. Для ініціалізації використовується властивість Connection відповідного табличного адаптера (адаптера змінною таблиці).

У третьому рядку властивості CommandText присвоюється назва збереженої процедури. В 4-му рядку вказується, що команда є збереженої процедурою.

У рядках з 5-го по 8-ю додаються параметри команди (збереженої процедури). ***Параметри додаються в тому порядку, як вони перераховані при описі збереженої процедури!***

У процедурі Pos_del параметри @k і @msg є вихідними параметрами. Це зазначено в методі за допомогою властивості Direction.

Власне виклик збереженій процедурі відбувається в обробнику події натискання на кнопку DelPosButton - методі DelPosButton_Click (рис.11).

```
private void DelPosButton_Click(object sender, EventArgs e)
{
    if (MessageBox.Show("Хотите удалить должность " + (string)(dataGridView1.CurrentRow.Cells[1].Value) + "?",
          "Удаление должности", MessageBoxButtons.OKCancel) == System.Windows.Forms.DialogResult.OK)
    {
        DelPosCmd.Parameters["@id_pos"].Value = (short)(dataGridView1.CurrentRow.Cells[0].Value);
        DelPosCmd.Parameters["@m"].Value = 0;
        DelPosCmd.Parameters["@k"].Value = System.DBNull.Value;
        DelPosCmd.Parameters["@msg"].Value = System.DBNull.Value;
        System.Data.ConnectionState previousConnectionState = DelPosCmd.Connection.State;
        if (((DelPosCmd.Connection.State & System.Data.ConnectionState.Open)
                != System.Data.ConnectionState.Open))
        {
            DelPosCmd.Connection.Open();
        }
        try
        {
            DelPosCmd.ExecuteNonQuery();
            if ((byte)(DelPosCmd.Parameters["@k"].Value) > 0)
            {
                if (MessageBox.Show((string)(DelPosCmd.Parameters["@msg"].Value) + "\nВсе равно удалить?",
                    "Удаление из связанных таблиц", MessageBoxButtons.OKCancel) == System.Windows.Forms.DialogR
                {
                    DelPosCmd.Parameters["@m"].Value = 1;
                    DelPosCmd.ExecuteNonQuery();
                }
            }
        }
        finally
        {
            if ((previousConnectionState == System.Data.ConnectionState.Closed))
                DelPosCmd.Connection.Close();
        }
        this.viewPosTableAdapter.Fill(dataSet1.ViewPos);
    }
}
```

Рисунок 11 – Видалення запису з таблиці Teach_position за допомогою хранимой процедури

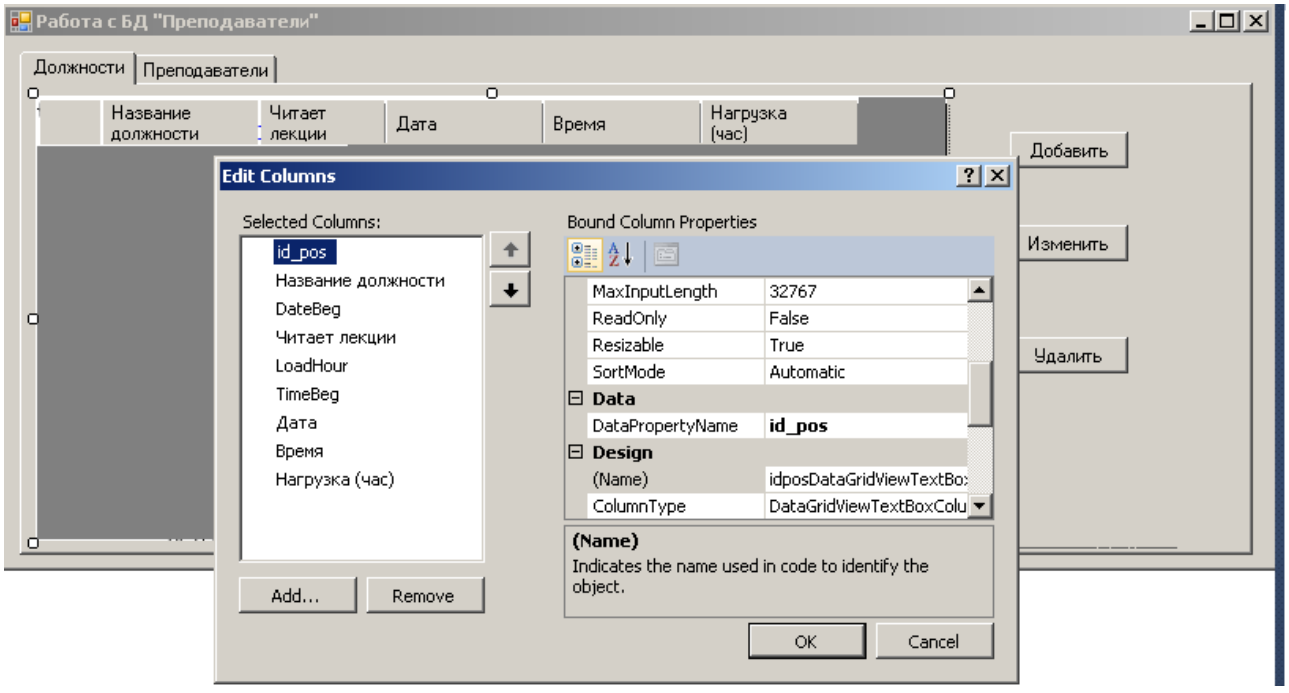


Рисунок 12 – Столбці преставлення ViewPos

У першому рядку методу DelPosButton_Click викликається діалогове вікно з заголовком «Видалення посади» і текстом всередині «Хочете видалити посаду назву». При цьому замість назву підставляється назва посади в виділеному рядку DataViewGrid. Атрибут Name_pos знаходиться в другому стовпці уявлення ViewPos (рис.12), але нумерація починається з нуля.

Якщо користувач відповів «OK» на питання в діалоговому вікні (вибрав кнопку «OK») - System.Windows.Forms.DialogResult.OK в програмі - то виконується решті код методу DelPosButton_Click.

Спочатку присвоюються значення параметрам процедури, що Pos_del; вхідних параметрів присвоюються певні значення, а вихідним параметрам - порожні (System.DBNull.Value).

Потім в змінну previousConnectionState записується поточний стан з'єднання команди DelPosCmd з базою даних. Далі перевіряється, якщо поточний стан не є станом з відкритим з'єднанням, то з'єднання відкривається.

В операторі try .. finally виконується процедура, що зберігається. Якщо параметр @k повертає значення більше нуля (код помилки - наявність записів в пов'язаних таблицях), то виводиться діалогове вікно «Видалення з пов'язаних таблиць» і вас запитають на видалення виділеної (на DataViewGrid) записи з таблиці Teach_position і всіх пов'язаних з нею записів в підлеглих таблицях.

Якщо користувач вибере «OK», то збережена процедура викликається знову з параметром @m = 1, що означає видалення записів в пов'язаних таблицях. У розділі finally закривається з'єднання команди з базою даних. В останньому рядку коду методу DelPosButton_Click оновлюється вміст DataViewGrid.

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

```
            if (MessageBox.Show("Хотите удалить должность " +
              (string)(dataGridView1.CurrentRow.Cells[1].Value) + "?",
                  "Удаление должности", MessageBoxButtons.OKCancel) ==
               System.Windows.Forms.DialogResult.OK)
            {
                DelPosCmd.Parameters["@id_pos"].Value =
                     (short)(dataGridView1.CurrentRow.Cells[0].Value);
                DelPosCmd.Parameters["@m"].Value = 0;
                DelPosCmd.Parameters["@k"].Value =
System.DBNull.Value;
                DelPosCmd.Parameters["@msg"].Value =
                   System.DBNull.Value;
                System.Data.ConnectionState previousConnectionState =
                   DelPosCmd.Connection.State;
                if (((DelPosCmd.Connection.State &
                       System.Data.ConnectionState.Open)
                        != System.Data.ConnectionState.Open))
                {
                    DelPosCmd.Connection.Open();
                }
                try
                {
                    DelPosCmd.ExecuteNonQuery();
                    if ((byte)(DelPosCmd.Parameters["@k"].Value) > 0)
                    {
                        if
                      (MessageBox.Show((string)(DelPosCmd.Parameters["@
                      msg"].Value) + "\nВсе равно удалить?",
                            "Удаление из связанных таблиц",
                           MessageBoxButtons.OKCancel) ==
                           System.Windows.Forms.DialogResult.OK)
                          {
                              DelPosCmd.Parameters["@m"].Value = 1;
                              DelPosCmd.ExecuteNonQuery();
```

```
                }
            }
        }
        finally
        {
            if ((previousConnectionState ==
                  System.Data.ConnectionState.Closed))
                DelPosCmd.Connection.Close();
        }
        this.viewPosTableAdapter.Fill(dataSet1.ViewPos);
    }
}
```

Контрольні питання

1. Як елементи форми можна зв’язати з набором даних?
2. Як переглянути джерела даних проекту у Visual Studio?
3. Як переглянути властивості збереженої процедури?

ЛІТЕРАТУРА

1. Дейт К. Дж. Введение в системы баз данных, 6-е издание: Пер. с англ. – К.; М.; СПб.: Издательский дом «Вильямс», 2000. – 848 с.: ил.
2. Дет Л.Дж. Введение в системы баз данных – 6-е изд. – К.: Диалектика, 1998.
3. Мейер С.М. Проектирование баз данных – М.: Мир, 1987.
4. Мамаев Е.В. Micrjsoft® SQL Sever 2000. — СПб.: БХВ-Петербург, 2004. — 1280 с.: ил.
5. Дэвидсон Л. Проектирование баз даннях на SQL Server 2000.: Пер. с англ. – М..: БИНОМ. Лаборатория знаний, 2003. – 680 с., ил.
6. Базы данных. Проектирование, реализация и сопровождение. Теория и практика. 3-е издание.: Пер. с англ. – М.: Издательский дом «Вильямс», 2003. – 1440 с.: ил.