Lời nói đầu.

Lập trình windows nâng cao (Lập trình windows 2) là môn học trong chương trình đào tạo đại học. Được sử dụng nhiều trong việc phát triển các phần mềm quản lý. Trong phạm vi tài liệu này, tập trung chủ yếu vào việc phát triển chương trình phần mềm theo ***mô hình 3 lớp***.

Nội dung của bài giảng được tổng hợp từ nhiều nguồn tài liệu, được tổ chức thành các chương, bao gồm:

Chương 1: Giao diện nâng cao

Chương 2: Xây dựng ứng dụng đa tầng

Chương 3: Xây dựng ứng dụng với ADO.Net

Chương 4: Quan hệ giữa các bảng

Chương 5: Truy cập dữ liệu với ADO.Net

Chương 6: Tạo báo cáo với CrystalReport

Bài giảng dùng cho sinh viên năm 3, sau khi đã học xong Lập trình hướng đối tượng, Lập trình windows 1, Hệ quản trị cơ sở dữ liệu quan hệ, xin cảm ơn Khoa CNTT – Đại học Thành Đô đã giúp tôi biên soạn, và sẽ còn nhiều thiếu sót. Mọi ý kiến đóng góp xin gửi về hòm thư hoangtn1204@gmail.com.

Hà nội, tháng 12 năm 2020.

MỤC LỤC.

Contents

DANH MỤC HÌNH

DANH MỤC BẢNG

Chương 1: GIAO DIỆN NÂNG CAO

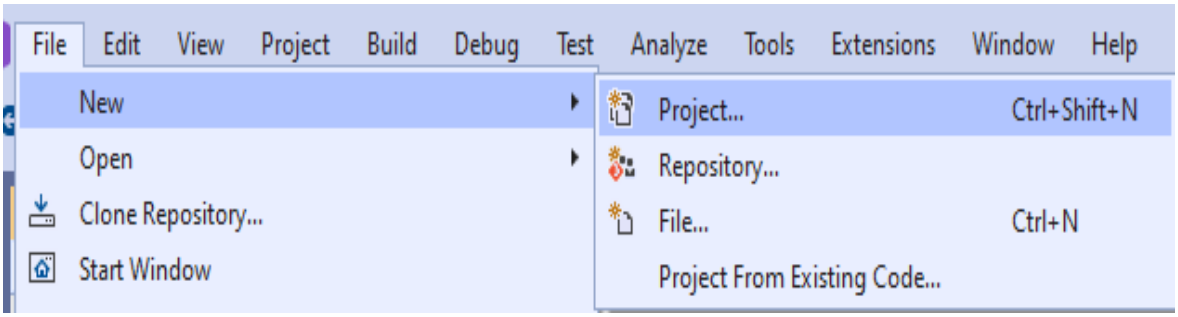
1.1. Xây dựng ứng dụng nhiều form

1.1.1. Thêm form và xóa form

Window form chính là cửa sổ của một màn hình ứng dụng. Nó chứa đựng các **dữ liệu**, **điều khiển** (control) trên đó và là cửa sổ giao tiếp giữa người sử dụng (user) và máy tính. Các bước tạo ứng dụng nhiều form trên windows như sau:

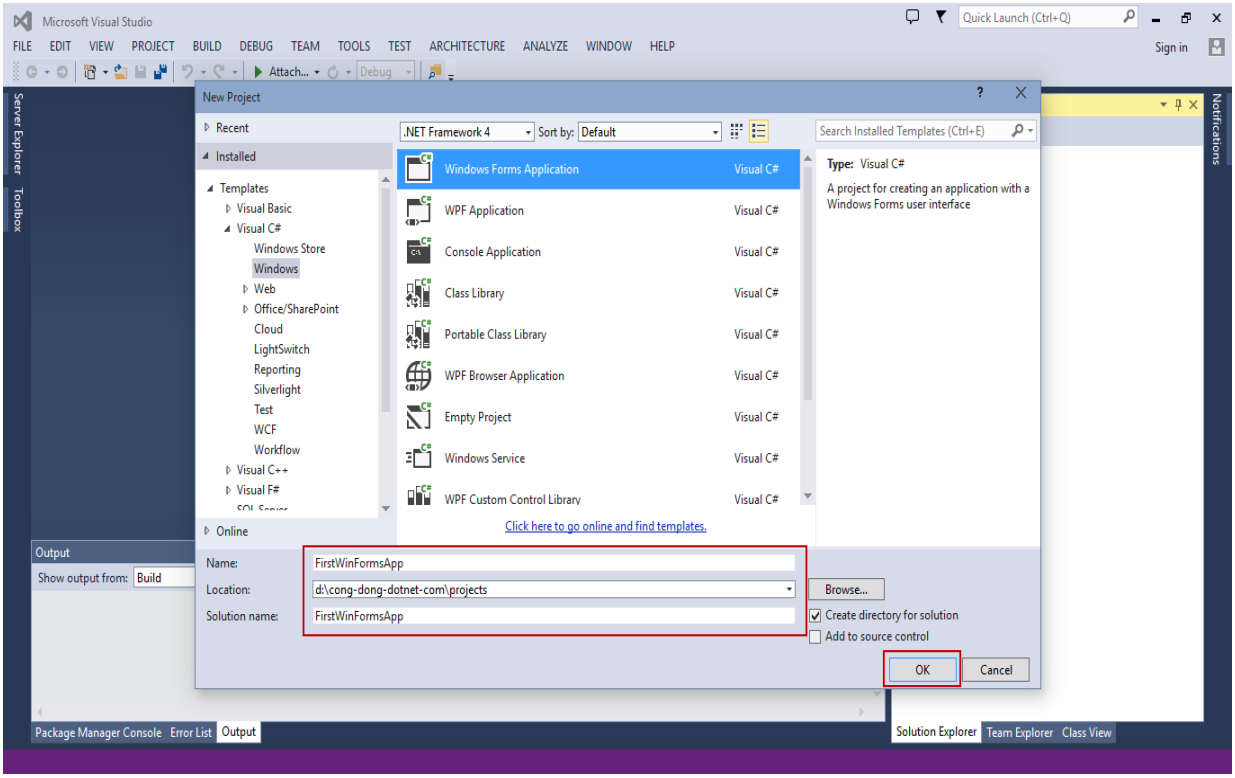
Với Visual Studio 2020:

Bước 1: Khởi chạy **Visual Studio** (VS), Tạo dự án mới trong VS bằng cách chọn menu File > New > Project... như hình bên dưới: **Create a New Project** in VS.



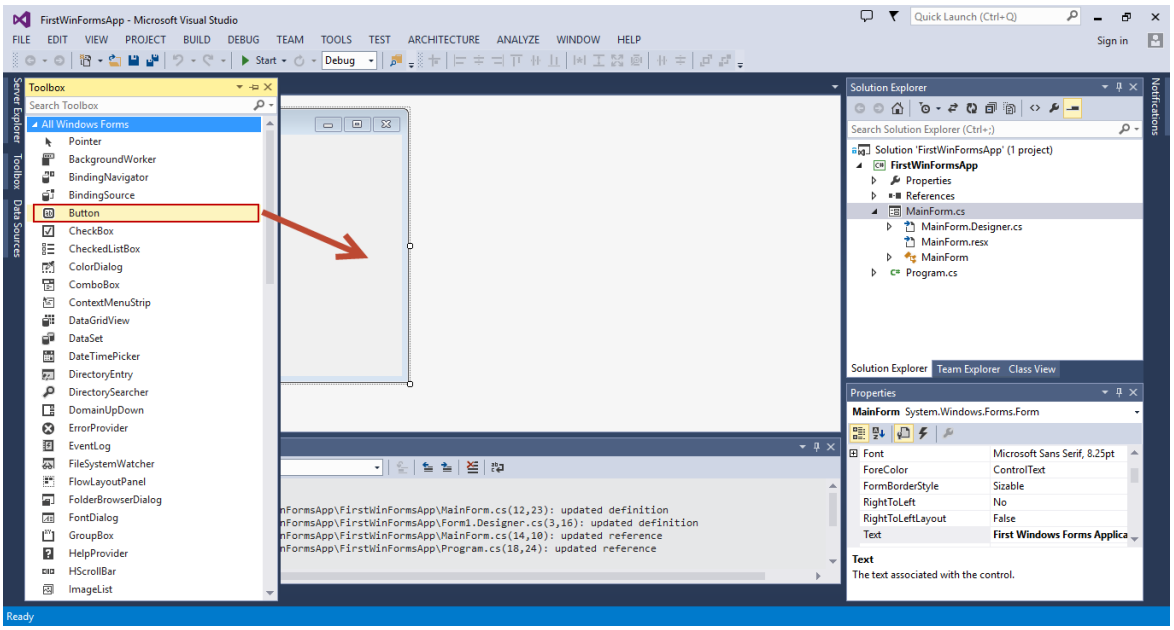
Hình 1. 1 Tạo dự án mới Windows Form

Bước 2: Một hộp thoại **New Project** hiện ra và sau đó chọn như hình bên dưới:



Hình 1. 2 Chọn kiểu dự án Window Application

Bước 3: Kéo control Button từ toolbox đến Form trong phần designer như hình dưới đây,nếu muốn thêm các thành phần khác của form cũng thêm một cách tương tự.



Hình 1. 3 Cửa sổ và công cụ làm việc

1.1.2. Khai báo và truy xuất qua lại các form

Việc gửi dữ liệu giữa các Fom trong lập trình Windows là rất quan trọng và cần thiết để tạo nên 1 ứng dụng thống nhất. Tổng quan bốn phương pháp cơ bản để gửi dữ liệu giữa các form:

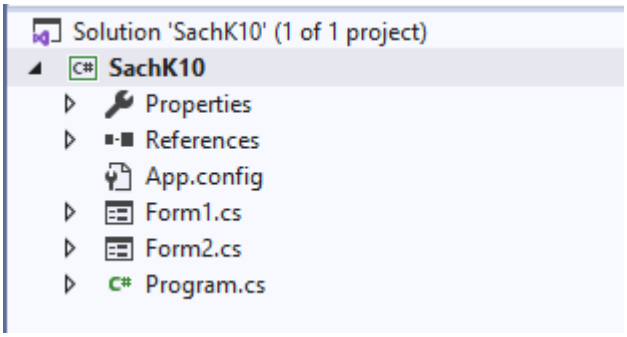
1) Sử dụng constructor

2) Sử dụng objects

3) Sử dụng properties

4) Sử dụng delegates

Trong 4 phương pháp trên, sử dụng **ủy quyền (delegates)** được đánh giá cao nhất vì nó không phụ thuộc vào quá trình **khởi tạo đối tượng**, sự **tồn tại của đối tượng**. Nó chỉ đơn giản sử dụng các **sự kiện (events)** do **người lập trình (coder)** bẫy sẵn.



Hình 1. 4 Dự án truyền dữ liệu giữa các Form

Bài toán đặt ra: Viết chương trình nhập dữ liệu trên textbox của form 1, nhấn nút trên form 1 để hiển thị form 2 và hiển thị nội dung dữ liệu đã nhập ở form 1 lên Label ở form 2.

STT	Tên điều khiển	Kiểu điều khiển	Ghi chú
1	Form1	Form	
2	textbox1	Texbox	
3	button1	Button	
4	Form2	Form	
5	label1	Label	

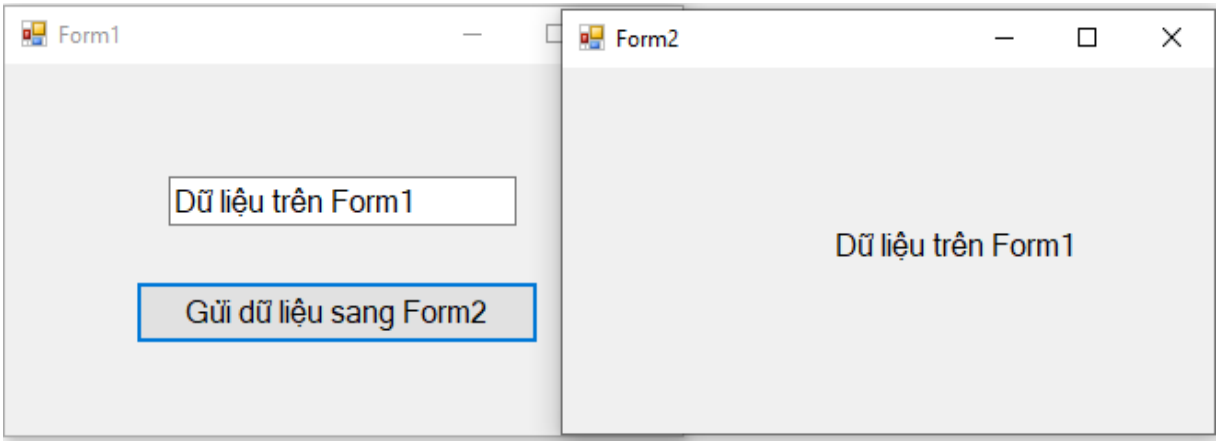
Bước 1: Tạo một **dự án (project)** có 2 forms: **Form1** và **Form2**

Bước 2: Với **Form 1**: Thêm 1 **Textbox** (textBox1) và một **Button** (button1), sử dụng để gửi dữ liệu

Bước 3: Với **Form 2**: Thêm 1 **Label** (label1) để **hiển thị dữ liệu** nhận được từ **Form1**

Phương Pháp Constructor:

Là phương pháp đơn giản nhất trong 4 phương pháp. Phương pháp này sử dụng hàm dựng (**Constructor**). Hàm dựng là hàm có đặc thù là nó được gọi đầu tiên khi ta tiến hành khởi tạo đối tượng và chỉ gọi một lần (Từ khi khởi tạo đến khi hủy đối tượng). Với phương pháp này ta chỉ cần thêm một tham biến vào hàm dựng của form2.



Hình 1. 5 Gửi dữ liệu từ Form1 sang Form2

Thêm tham biến vào **hàm khởi dựng** (**Constructor**) của **Form2**

```
public Form2(string strTextBox)
      {
          InitializeComponent();
          label1.Text=strTextBox;
          //Gán dữ liệu nhận được vào Label để thể hiện
      }
```

Tại **nút** (**button**) "Gửi dữ liệu sang form 2" ta tiến hành khởi tạo một đối tượng của **Form2** và **Form2** sẽ hiện ra mỗi khi ta click vào nút này.

```
private void button1_Click(object sender, System.EventArgs e)
      {
          Form2 frm = new Form2(textBox1.Text);
          frm.Show();
      }
```

Phương Pháp Đối tượng (Object)

Object là kiểu **tham chiếu (reference)** và được tạo trên bộ nhớ **Heap**[1] và sử dụng từ khóa **new**. Phương pháp này gồm các bước sau:

Bước 1: Tại Form1 thay đổi **quyền truy cập** (access modifier) cho **textbox** từ riêng tư (**private**) thành công khai (**public**).

```
    public class Form1 : System.Windows.Forms.Form {
        public System.Windows.Forms.TextBox textBox1;
```

Tại **Form2** khai báo một biến frm1 có kiểu **Form1**

```
    public class Form2 : System.Windows.Forms.Form1 {
        private System.Windows.Forms.Label label1;
        public Form1 frm1;
        // ..............................
```

Bước 2: Tại **Form1** viết code **hàm sự kiện Click** của nút (**button**) "Gửi dữ liệu sang form 2".

```
private void btnSend_Click(object sender, System.EventArgs e)
    {
        Form2 frm = new Form2();
        frm.frm1 = this;  //Gán form1 cho biến frm1 có kiểu Form1 của Form2
        frm.Show();     //Hiện Form2    }
```

Bước 3: Trong hàm **Load** của **Form2**, gán giá trị của **textbox1** của **form1** cho **Label** của **form2**

```
private void Form2_Load(object sender, System.EventArgs e)
      {
          label1.Text = ((Form1)frm1).textBox1.Text;
      }
```

[1] Bộ nhớ Heap dùng để cấp phát bộ nhớ cho đối tượng, biến toàn cục. Bất cứ khi nào khai báo đối tượng thì các giá trị của đối tượng sẽ được lưu trữ trong Heap.

Kết quả: Chạy chương trình và ta có được kết quả như ***Hình 1.4***.

Phương pháp sử dụng thuộc tính (Properties):

- **Thuộc tính** (**Properties**) cho phép **đối tượng** truy cập trực tiếp đến các thành viên của **lớp** (**class**). Form cũng là một **lớp** (**class**), trong phương pháp này thì thêm **thuộc tính** (**Properties**) cho mỗi **form**.:

- Ở **Form1**, thêm thuộc tính `_textBox1`, để **nhận** giá trị từ ô Textbox

```
public string _textBox1 { get { return textBox1.Text; } }
```

- Ở **Form2** viết thêm thuộc tính `_textBox`, để **gán** giá trị cho **Label**

```
public string _textBox { set { label1.Text = value; } }
```

- Ở **Form1**, trong sự kiện **button click** khởi tạo và hiển thị **Form2**. Giá trị của _textbox1 trong **Form1** sẽ được chuyền sang cho **Form2** thông qua thuộc tính `_textBox` và gán vào label1 của **Form2**.

```
private void button1_Click(object sender, System.EventArgs e)
  {
      Form2 frm = new Form2();
      frm._textBox = _textBox1; //gán giá trị của thuộc tính _textbox1 của Form1 cho
thuộc tính _textbox của Form2
      frm.Show(); // Hiển thị Form2
  }
```

Kết quả được hiển thị như ***Hình 1.5.***

Sử dụng cơ chế ủy quyền (Delegates).

- **Delegates** là một ***kiểu tham chiếu*** (reference type), được sử dụng như là một phương thức đặc biệt (**Con trỏ hàm**), có thể đăng ký bất kỳ phương thức nào với **delegates**. Và đây là một kỹ thuật rất thường được sử dụng trong **lập trình sự kiện**.

Bước 1: Ở **Form1** thêm một **delegate** `delPassData` với tham số text

```
public delegate void delPassData(TextBox text);
```

Bước 2: Trong **Form2** tạo phương thức để **delegate** (`delPassData`) sẽ trỏ tới. Trong phương thức này gán giá trị của ô **textbox** trong **Form1** (`txtForm1`) vào **Label** của **Form2** (`label1`):

```
    public void funData(TextBox txtForm1) { label1.Text = txtForm1.Text; }
```

Bước 3: Trong sự kiện **button click** (`btnSend_Click`) của **Form1** khởi tạo đối tượng **Form2** (`frm`) và **delegate** (`del`). Chỉ ra một phương thức của **Form2** (`funData`) và gọi **delegate** như sau:

```
private void btnSend_Click(object sender, System.EventArgs e)
    {
        Form2 frm = new Form2();
        delPassData del = new delPassData(frm.funData);
        del(this.textBox1);
        frm.Show();
    }
```

Kết quả được thể hiện như Hình 1.5.

Trong 4 phương pháp này, phương pháp sử dụng hàm dựng (Constructor) được đánh giá là kém nhất, vì đối tượng chỉ được khởi tạo một lần, nếu đối tượng ko được hủy(form2 không hủy hoặc tắt đi) mà ta lại muốn thay đổi dữ liệu chuyền từ form1 sang form2 thì phương pháp này ko thực hiện được. Delegate là phương pháp khắc phục rất tốt nhược điểm này.

1.1.3. Các thuộc tính và phương thức cơ bản của form

Bảng các thuộc tính giống nhau của các Control trong C#.
Có thể thay đổi thuộc tính ở khung **Properties** trong lúc thiết kế **Form**, và cũng có thể thay đổi thuộc tính bằng **code**.

Thuộc tính	Chức năng
(name)	Tên đại diện cho control đó. Nó như một tên biến vậy. Rất quan trọng
Achor	Cố định control này khi thay đổi kích thước form
BackColor	Màu nền của control đó
BackgroundImage	Hình nền của control đó
ContextMenuStrip	Menu khi ấn chuột phải lên control
Cursor	Hình con chuột khi rê lên control
Dock	Gần giống với Anchor nhưng nó sẽ chiếm toàn bộ phần được được đặt. VD chọn Dock là bottom thì toàn bộ phần dưới form sẽ không đặt được phần từ khác
Enabled	Có cho phép sử dụng nó hay không.
Font	Chứa các thuộc tính về màu, cỡ, kiểu chữ mô tả (hoặc nội dung, giá trị) control đó
Location	Vị trí của control đó trên form
Tag	Nội dung đánh dấu của control đó. Nó giúp control có hai giá trị miêu tả. Tag khồn được hiện thị
Text	Nội dung miêu tả control đó. Hoặc chính là giá trị của nó (với textbox)
TextAlign	Căn lề nội chữ miêu tả của control đó
Visible	Hiển thị nó lên form hay không.

Trong lập trình trên Console ta hay gọi một **phương thức** là một "hàm". Nhưng lập trình C# hướng đối tượng thì hàm đặt trong lớp (**class)** được gọi là **phương thức**. Mỗi **class** dù là class thường hay là **Form**, hay bất cứ **Control** nào khác đều có **phương thức** riêng của nó.

Gọi (call) đến **phương thức**, giống như là ra lệnh cho đối tượng phải làm một hành động gì đó vậy. Cũng giông như trẻ con khi còn nhỏ ai cũng có phương thức là Học(), Ăn(), QuétNhà(), RwarBat(), …

PHƯƠNG THỨC CỦA ĐỐI TƯỢNG

Phương thức	Chức năng
Active	Khởi động Form, và Focus vào nó
BringToFront	Đưa control ra ngoài cùng, phía trên theo trục Z
CenterToParent	Căn chỉnh control nằm giữa khung viền của Form cha của nó
CenterToScreen	Căn chỉnh control nằm giữa màn hình hiện tại
Close	Đóng form lại
Dispose	Giải phóng mọi tài nguyên đã sử dụng bởi component
Focus	Tập trung vào form. Giống như mình Alt-Tab để chọn Form đó
Hide	Ẩn Control khỏi người dùng
OnActive	Khởi động sự kiện Activated (Thực hiện hành động sau khi Form đc Active)
OnClick	Khởi động sự kiện Click (Thực hiện sau khi control được Click)
Refresh	Làm mới lại Form/Control, vô hiệu hóa client và lập tức vẽ lại chính nó và các control con đi cùng
Show()	Hiển thị Control ra màn hình
ShowDialog()	Hiện thị Form như một Dialog
ToString()	Hiển thị chuỗi tương ứng cho nội dung

SỰ KIỆN CỦA CÁC ĐỐI TƯỢNG

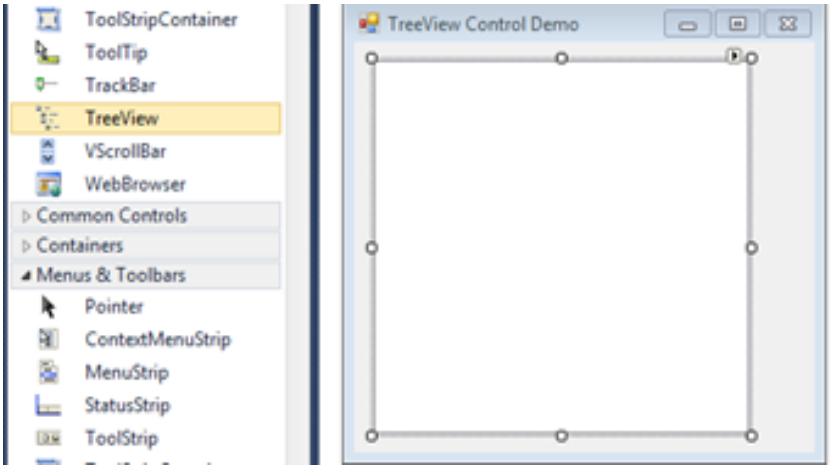
Sự kiện	Điều kiện xảy ra
Click	Ấn vào control đó
DoubleClick	Nháy kép vào control đó
KeyDown	Bắt đầu ấn phím
KeyUp	Đã ấn phím xong
KeyPress	Trong khi ấn phím
MouseDown	Ấn chuột
MouseUp	Thả chuột
MouseHover	Rê chuột qua control
MouseLeave	Rê chuột ra khỏi control

1.2. Một số đối tượng điều khiển nâng cao

1.2.1. TreeView

Với điều khiển **TreeView** trong Windows Form, có thể hiển thị một hệ thống phân cấp các **node** cho những người dùng. Giống như cửa sổ phía bên trái trong Windows Explorer trong hệ điều hành Windows. Mỗi một **node** trong **TreeView** có thể chứa những node khác, được gọi là những **node con** (child nodes). Dùng để hiển thị các node cha hay các node con giống như mở rộng (expanded) hay thu gọn (collapsed). Cũng có thể hiển thị **TreeView** với các **checkbox** bằng cách thiết lập thuộc tính cho **Checkboxs** cho node = true. Có thể thực hiện việc lựa chọn hay xóa các node bằng cách thiết lập thuộc tính **Checked** của node là true hay false.

TreeView control chứa các node ở cấp cao nhất trong Nodes Collection. Mỗi TreeNode đề có nodes collection của mình để chứa các node con. Cả 2 thuộc tính collection là của kiểu TreeNodeCollection với các hàm thành viên cho phép bạn thêm (add) hoặc xóa (remove) và sắp xếp các node ở một cấp đơn trong hệ thống phân cấp các node.



Hình 1. 6 Điều khiển Tree View

Sử dụng phương thức **Add** bằng thuộc tính **Nodes** của **TreeView**

```
// Adds new node as a child node of the currently selected node. TreeNode newNode = new
TreeNode("Text for new node");

treeView1.Nodes.Add(newNode);

TreeNode newNode1 = new TreeNode("Text for new node 1");
```

```
treeView1.Nodes.Add(newNode1);
TreeNode newNode2 = new TreeNode("Welcome to TreeView Control Tutorial");
treeView1.Nodes.Add(newNode2);
TreeNode node1 = new TreeNode("C#");
TreeNode node2 = new TreeNode("VB.NET");
TreeNode node3 = new TreeNode("C++");
TreeNode[] array = new TreeNode[] { node1, node2, node3 };
TreeNode programmingLanguage = new TreeNode("Programming Language", array);
treeView1.Nodes.Add(programmingLanguage);
```

Kết quả của ví dụ trên:

Hình 1. 7 Kết quả chương trình sử dụng Tree View

– Sử dụng phương thức **Remove** để xóa một node đơn, **Clear** để xóa tất cả các node.

Add thông tin tùy chỉnh vào TreeView

Tạo một lớp node mới dẫn xuất từ lớp TreeNode, trong đó có một trường tùy biến để ghi lại một đường dẫn tập tin.

```
class myTreeNode : TreeNode {
    public string FilePath;
    public myTreeNode(string fp)
    {
        FilePath = fp;
        this.Text = fp.Substring(fp.LastIndexOf("\\"));}}
```

– Hoặc sử dụng node mới được dẫn xuất như một tham số trong các lời gọi hàm.

Trong ví dụ dưới đây, đường dẫn đặt ra cho các vị trí của trường **Text** là thư mục **My Document**. Điều này có thể được thực hiện bởi ta có thể giả định rằng hầu hết các máy chạy hệ điều hành windows sẽ có thư mục này, cũng cho phép người dùng với các mức độ truy cập hệ thống tối thiểu một cách an toàn khi chạy ứng dụng.

```
// Lưu ý: sử dụng ký tự @ khi sử dụng đường dẫn xác định.
treeView1.Nodes.Add(new myTreeNode
(System.Environment.GetFolderPath
    (System.Environment.SpecialFolder.Personal)+ @"\TextFile.txt")
);
```

Sử dụng **Casting** để chuyển đổi kiểu của các đối tượng.

```
private void treeView1_NodeMouseClick(object sender, TreeNodeMouseClickEventArgs e)
{
   myTreeNode myNode = (myTreeNode)e.Node;
   MessageBox.Show("Node được chọn là:  " + myNode.FilePath);
}
```

Xác định node nào trong **TreeView** đã được click

– Sử dụng **EventArgs object** để trả về một tham chiếu đến đối tượng node được click.

– Xác định được **node** được click bằng cách kiểm tra lớp **TreeViewEventArgs** có chứa dữ liệu liên quan đến sự kiện này.

```
private void treeView1_AfterSelect(object sender,
TreeViewEventArgs e)
{
    // Determine by checking the Text property.
MessageBox.Show(e.Node.Text);
}
```

Duyệt qua tất cả các node của TreeView

– Tạo một *thủ tục đệ quy* (recursive) để kiểm tra mỗi node.

– Gọi thủ tục.

Ví dụ dưới đây minh họa việc làm thế nào để in các thuộc tính **Text** của các đối tượng **TreeNode**.

```
private void PrintRecursive(TreeNode treeNode)
{
    // in ra các node. listBox1.Items.Add(treeNode.Text);
    MessageBox.Show(treeNode.Text);
    // Sử dụng đệ quy để in ra từng node. foreach (TreeNode tn in
treeNode.Nodes)
    {
        PrintRecursive(tn);
    }
}
// Gọi thủ tục sử dụng TreeView
private void CallRecursive(TreeView treeView)
{
    // IN ra từng node sử dụng đệ quy.
TreeNodeCollection nodes = treeView.Nodes;
```

```
    foreach (TreeNode n in nodes)
    {
        PrintRecursive(n);
    }
}
```

Gọi lại phương thức trên trong sự kiện **button click**.

```
private void btnPrintNode_Click(object sender, EventArgs e)
{
    PrintRecursive(treeView1.SelectedNode);}
```

Thiết lập icon cho các node trong **TreeView**

– Để tạo icon cho mỗi node của TreeView ta phải có một imageList chứa các hình cần sử dụng Thiết lập thuộc tính imageList của TreeView = imageList mà ta muốn sử dụng

```
treeView1.ImageList = imageList1;
```

Có thể được thực hiện bằng cách khác như tùy chỉnh thuộc tính Image List của TreeView trong giao diện thiết kế.

Thiết lập thuộc tính **ImageIndex** và **SelectedImageIndex** của **node**.

Việc thiết lập này có thể thực hiện theo 2 cách:

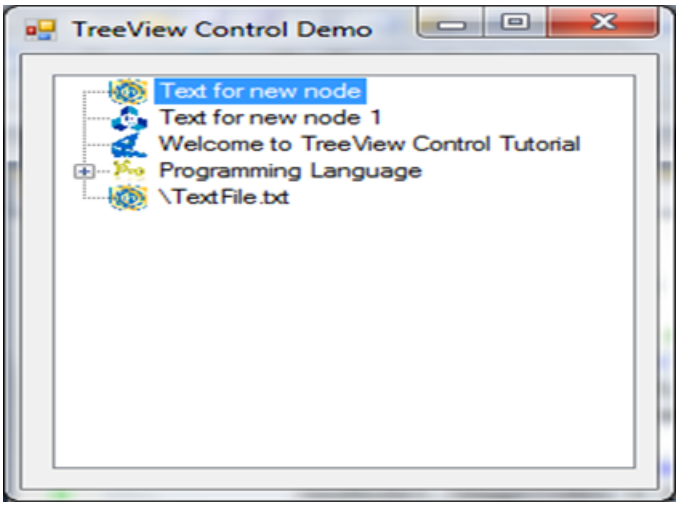
– Click vào button của mục ImageIndex trong thanh properties từ giao diện thiết kế của TreeView trên Form.

– Sử dụng code:

```
TreeNode newNode = new TreeNode("Text for new node");
newNode.ImageIndex = 0;
treeView1.Nodes.Add(newNode);
TreeNode newNode1 = new TreeNode("Text for new node 1");
newNode1.ImageIndex = 1;
```

```
treeView1.Nodes.Add(newNode1);
TreeNode newNode2 = new TreeNode("Welcome to TreeView Control Tutorial");
newNode2.ImageIndex = 2;
treeView1.Nodes.Add(newNode2);
TreeNode node1 = new TreeNode("C#");
TreeNode node2 = new TreeNode("VB.NET");
TreeNode node3 = new TreeNode("C++");
TreeNode[] array = new TreeNode[] { node1, node2, node3 };
TreeNode programmingLanguage = new TreeNode("Programming Language", array);
programmingLanguage.ImageIndex = 6;
treeView1.Nodes.Add(programmingLanguage);
```

Kết quả:



Hình 1. 8 Kết quả khác sử dụng tree view

Đính kèm một ShortCut Menu cho một TreeView Node

– Khởi tạo một **TreeView control** với các thiết lập thuộc tính thích hợp, tạo ra một node gốc, và sau đó thêm các node con.

– Tạo một **ContexMenuStrip** và sau đó thêm một **ToolScriptMenuItem** cho mỗi thao tác mà bạn muốn thực hiện trong thời gian chương trình thực thi.

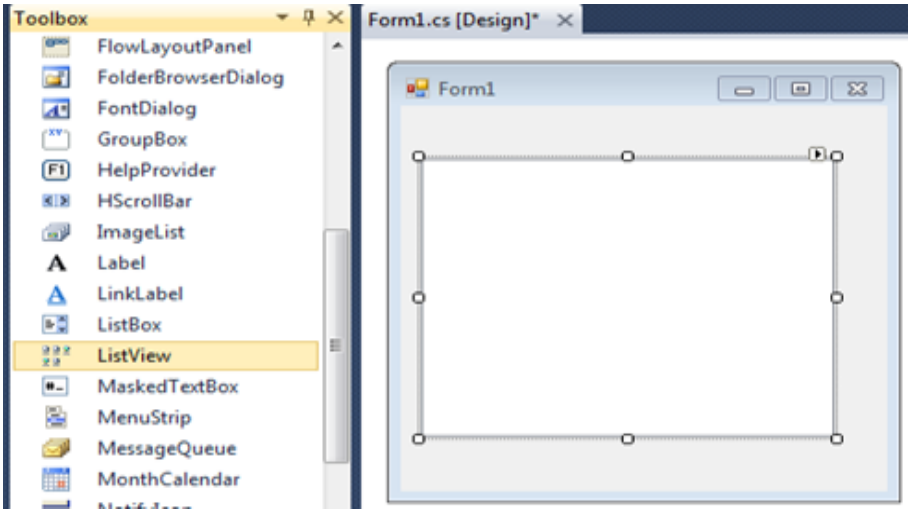
– Thiết lập thuộc tính **ContextMenuStrip** của **Tree Node** thích hợp vào menu shortcut bạn tạo ra.

– Khi thuộc tính này được thiết lập, menu chuột phải sẽ tự hiển thị khi bạn bấm chuột phải vào node.

```
// Declare the TreeView and ContextMenuStrip private TreeView
menuTreeView;

private ContextMenuStrip docMenu;

public void InitializeMenuTreeView()

{

    // Create the TreeView. menuTreeView = new TreeView();

    menuTreeView.Size = new Size(200, 200);

    // Create the root node. TreeNode docNode = new
TreeNode("Documents");

    // Add some additional nodes.
docNode.Nodes.Add("phoneList.doc");

    docNode.Nodes.Add("resume.doc");

    // Add the root nodes to the TreeView.
menuTreeView.Nodes.Add(docNode);

    // Create the ContextMenuStrip. docMenu = new
ContextMenuStrip();

    //Create some menu items. ToolStripMenuItem openLabel = new
ToolStripMenuItem();

    openLabel.Text = "Open";

    ToolStripMenuItem deleteLabel = new ToolStripMenuItem();

    deleteLabel.Text = "Delete";
```

```
    ToolStripMenuItem renameLabel = new ToolStripMenuItem();
    renameLabel.Text = "Rename";
    //Add the menu items to the menu. docMenu.Items.AddRange(new
ToolStripMenuItem[]{openLabel,
    deleteLabel, renameLabel});
    // Set the ContextMenuStrip property to the ContextMenuStrip.
docNode.ContextMenuStrip = docMenu;
    // Add the TreeView to the form.
this.Controls.Add(menuTreeView);}
```

1.2.2. ListView



Hình 1. 9 Add ListView

Có thể viết code thuần để add listview vào form.

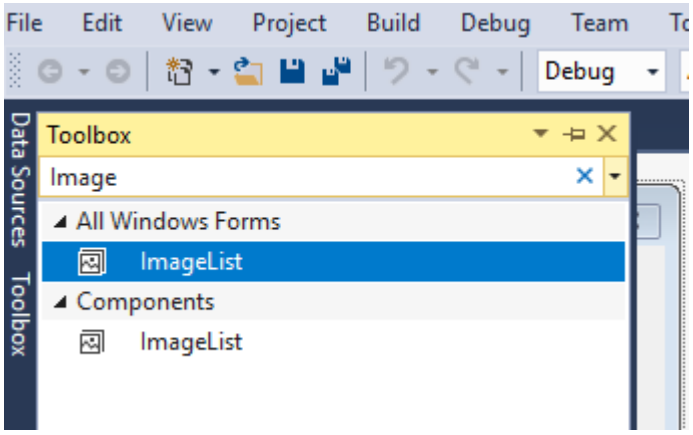
```
ListView myListView = new ListView();    // Khai báo một ListView
control.
myListView.Size = new System.Drawing.Size(390, 100);    // Kích
thước hiển thị
this.Controls.Add(myListView);           // Add ListView control
vừa khai báo vào Form
```

Thêm các Items vào ListView

```
myListView.Items.Add("Công Nghệ Thông Tin");
myListView.Items.Add("Khoa Kinh Tế");
myListView.Items.Add ("Khoa Cơ Bản");
myListView.Items.Add("Khoa Du lịch ngoại ngữ");
myListView.Items.Add("Khoa Công nghệ kỹ thuật Ô tô");
myListView.Items.Add("Khoa Y Dược");
```

1.2.3. ImageList

Giả sử có các ảnh Garden.jpg, Tree.jpg, Waterfall.jpg trong thư mục `C:\Images`



Hình 1. 10 Thêm ImageList vào dự án

```
Graphics g = Graphics.FromHwnd(this.Handle);
ImageList photoList = new ImageList();
photoList.TransparentColor = Color.Blue;
photoList.ColorDepth = ColorDepth.Depth32Bit;
photoList.ImageSize = new Size(200, 200);
photoList.Images.Add(Image.FromFile(@"C:\Images\Garden.jpg"));
photoList.Images.Add(Image.FromFile(@"C:\Images\Tree.jpg"));
```

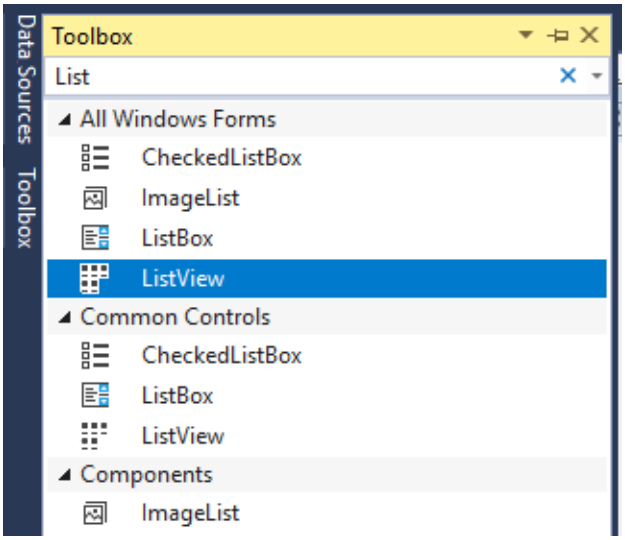
```
photoList.Images.Add(Image.FromFile(@"C:\Images\Waterfall.jpg"));
for (int count = 0; count < photoList.Images.Count; count++)
{
    photoList.Draw(g, new Point(20, 20), count);
    // Paint the form and wait to load the image
    Application.DoEvents();
    System.Threading.Thread.Sleep(1000);
}
```

1.2.4. GridView

Grid view thường được dùng để load dữ liệu từ một bảng, sẽ tìm hiểu sâu hơn ở các chương tiếp theo.

Hình 1. 11 Load dữ liệu bằng GridView

1.2.6. ListView



Hình 1. 12 Add List view vào dự án

```
using System;
using System.Drawing;
using System.Windows.Forms;
namespace KhoaCNTTViduListview
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            listView1.View = View.Details;
            listView1.GridLines = true;
            listView1.FullRowSelect = true;

            //Add column header
            listView1.Columns.Add("ProductName", 100);
            listView1.Columns.Add("Price", 70);
            listView1.Columns.Add("Quantity", 70);

            //Add items in the listview
            string[] arr = new string[4];
            ListViewItem itm;

            //Add first item
            arr[0] = "product_1";
            arr[1] = "100";
```

```
            arr[2] = "10";
            itm = new ListViewItem(arr);
            listView1.Items.Add(itm);

            //Add second item
            arr[0] = "product_2";
            arr[1] = "200";
            arr[2] = "20";
            itm = new ListViewItem(arr);
            listView1.Items.Add(itm);
        }
        private void button1_Click(object sender, EventArgs e)
        {
            string productName = null;
            string price = null;
            string quantity = null;

            productName = listView1.SelectedItems[0].SubItems[0].Text;
            price = listView1.SelectedItems[0].SubItems[1].Text;
            quantity = listView1.SelectedItems[0].SubItems[2].Text;

            MessageBox.Show(productName + " , " + price + " , " + quantity);
        }
    }
}
```

Bài tập cuối chương 1.

Bài tập 1.

Hãy giải phương trình bậc 2 như sau: các hệ số a, b, c nhập trên form1, nhưng kết quả hiển thị trên form 2.

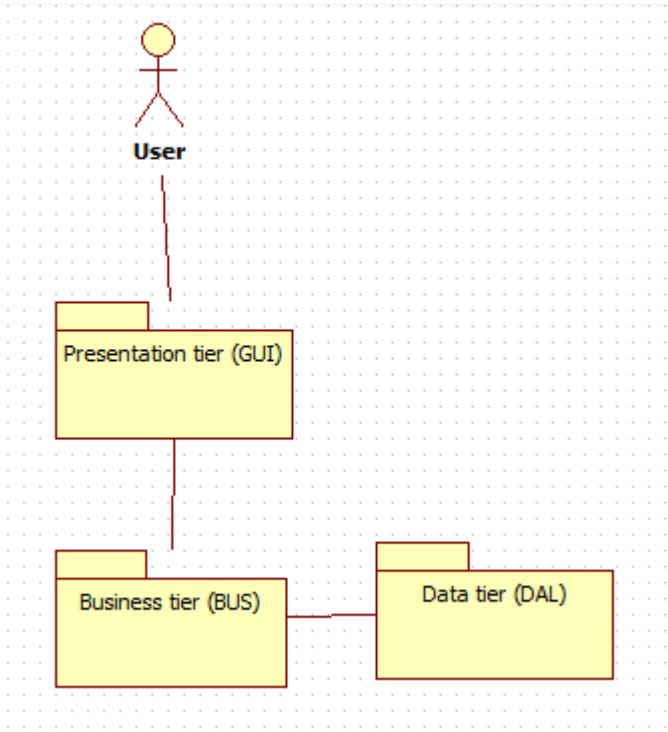
Hãy viết chương trình nhập vào 1 số nguyên trên form1, trên form 2 hãy hiển thị xem số đó là chẵn hay lẻ, hoàn hảo hay không…

Bài tập 2.

- Viết lớp số có thuộc tính giá trị, phương thức kiểm tra là số nguyên tố, âm, dương, và phương thức đọc số.
- Tạo formNhap, cho phép nhập một số nguyên bất kỳ, tạo nút Đọc, chuyển số đó sang formXuLy, trên form xử lý hiển thị việc đọc số.(756) bảy trăm năm sáu.

Chương 2: XÂY DỰNG ỨNG DỤNG ĐA TẦNG

2.1. Mô hình UML



Hình 2. 1 Mô hình 3 tầng khái quát

Phân biệt tầng (Tier) và lớp (Layer)

Tầng cho chúng ta thấy sự tách biệt vật lý với nhau, những tầng này có thể nằm cùng một nơi hay các nơi khác nhau trên thực tế các ứng dụng lớn thì Database sẽ nằm ở một Server, các API hay Web Service nằm một Server khác và ứng dụng thì chạy ở Client.

Khác với tầng, lớp không thể hiện rõ sự tách biệt về mặt vật lý chúng thường nằm chung một nơi nhưng ở các namspace khác nhau.

Theo wiki thì **mô hình 3 tầng**:

> *3-tiers là một kiến trúc kiểu client/server mà trong đó giao diện người dùng (UI-user interface), các quy tắc xử lý (BR-business rule hay BL-business logic), và việc lưu trữ dữ liệu được phát triển như những module độc lập, và hầu hết là được duy*

trì trên các nền tảng độc lập, và mô hình 3 tầng (3-tiers) được coi là một kiến trúc phần mềm và là một mẫu thiết kế.

3-Tiers có tính vật lý (physical): là mô hình **client-server** (mỗi tier có thể đặt chung 1 nơi hoặc nhiều nơi, kết nối với nhau qua Web services, WCF, Remoting...).

+ **Presentation tier** bao gồm các thành phần phần xử lý giao diện **Graphic User Interface** (GUI)

Business tier gồm các thành phần **Business Logic Layer** (BLL), **Data Access Layer** (DAL) và **Data Tranfer Object** (DTO).

Data tier lưu trữ dữ liệu, là các hệ quản trị CSDL như MS SQL Server, Oracle, SQLite, MS Access, XML files, text files...

Ưu điểm:

- Dễ dàng mở rộng, thay đổi quy mô của hệ thống: Khi cần tải lớn, người quản trị có thể dễ dàng thêm các máy chủ vào nhóm, hoặc lấy bớt ra trong trường hợp ngược lại.

Nhược điểm:

- Việc truyền dữ liệu giữa các tầng sẽ chậm hơn vì phải truyền giữa các tiến trình khác nhau dữ liệu cần phải được đóng gói -> truyền đi -> mở gói trước khi có thể dùng được.

Mô hình 3 lớp (3-layer) gồm có 3 phần chính:

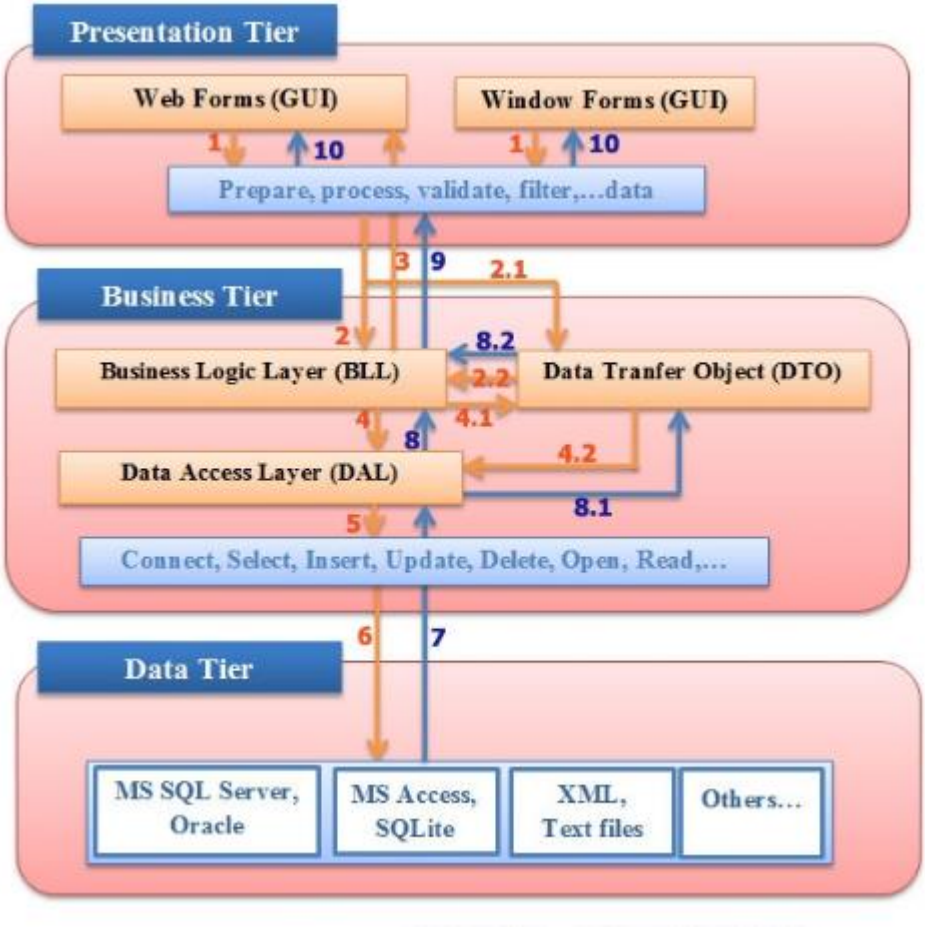
– **Presentation Layer (GUI)**: Lớp này có nhiệm vụ chính giao tiếp với người dùng. Nó gồm các thành phần giao diện (win form, web form…) và thực hiện các công việc như nhập liệu, hiển thị dữ liêu, kiểm tra tính đúng đắn dữ liệu trước khi gọi lớp **Business Logic Layer** (BLL).

– **Business Logic Layer (BLL cũng có thể đặt là BUS):** Layer này phân ra 2 thành nhiệm vụ:

Đây là nơi đáp ứng các yêu cầu thao tác dữ liệu của **GUI layer**, xử lý chính nguồn dữ liệu từ Presentation Layer trước khi truyền xuống **Data Access Layer** và lưu xuống hệ quản trị CSDL.

Đây còn là nơi kiểm tra các ràng buộc, tính toàn vẹn và hợp lệ dữ liệu, thực hiện tính toán và xử lý các yêu cầu nghiệp vụ, trước khi trả kết quả về **Presentation Layer**.

– **Data Access Layer (DAL)**: Lớp này có chức năng giao tiếp với hệ quản trị CSDL như thực hiện các công việc liên quan đến lưu trữ và truy vấn dữ liệu (tìm kiếm, thêm, xóa, sửa…).



Hình 2. 2 Mô hình 3 tầng thể hiện chi tiết

Mô hình 3-layer gồm **2 tiến trình sau**:

Tiến trình thứ nhất:

- Người sử dụng tác động lên **GUI** yêu cầu hiển thị thông tin lên màn hình. Tại đây GUI sẽ kiểm tra yêu cầu của người dùng nhập có hợp lệ hay không, nếu không hợp lệ sẽ thông báo cho người dùng.

- Ngược lại yêu cầu sẽ được gửi trực tiếp đến **BLL** (2) hoặc thông qua lớp chuyển đổi dữ liệu **DTO** hỗ trợ luân chuyển (2.1 & 2.2), tại đây **BLL** sẽ **xử lý nghiệp vụ** về yêu cầu của người dùng, nếu yêu cầu không hợp lệ hoặc tự xử lý yêu cầu không cần phải truy vấn thì BLL sẽ gửi thông tin về **GUI** (3) và **GUI** sẽ hiển thị kết quả cho người dùng.

- Trong trường hợp **BLL** cần **thao tác trên dữ liệu** từ **CSDL** thì **BLL** sẽ gửi yêu cầu đến trực tiếp đến **DAL** (4) hoặc thông qua **DTO** (4.1 & 4.2), nhờ **DAL** giao tiếp với hệ quản trị **CSDL** (5) lấy hoặc thêm, xóa, sửa dữ liệu.

- **DAL** sẽ giao tiếp hệ quản trị **CSDL** (5) với các truy vấn (**sử dụng công nghệ ADO**, LINQ to SQL, NHibernate, Entity Framework)

Tiến trình thứ 2:

- Sau khi **DAL** thực hiện giao tiếp, hệ quản trị **CSDL** sẽ trả kết quả truy vấn về **DAL** (7), **DAL** sẽ gởi thông tin về dữ liệu vừa lấy trực tiếp sang **BLL** (8) hoặc thông qua **DTO** (8.1 & 8.2) xử lý tiếp nghiệp vụ với yêu cầu đã gửi từ trước, sau khi xử lý xong nghiệp vụ, **BLL** sẽ gởi thông tin đến **GUI** (10), **GUI** sẽ hiển thị thông báo và kết quả yêu cầu lên màn hình. **DTO Layer** lớp này chỉ là phụ dùng để chuyển đổi dữ liệu để dễ xử lý, có thể có khoặc không tùy từng dự án.

Ưu điểm

Việc phân chia thành từng lớp giúp cho **code tường minh** hơn. Nhờ vào việc chia ra từng lớp đảm nhận các chức năng khác nhau và riêng biệt như giao diện, xử lý, truy vấn thay vì để tất cả lại một chỗ. Nhằm giảm sự kết dính.

Dễ bảo trì khi được phân chia, thì một thành phần của hệ thống sẽ dễ thay đổi. Việc thay đổi này có thể được cô lập trong 1 lớp, hoặc ảnh hưởng đến lớp gần nhất mà không ảnh hưởng đến cả chương trình.

Dễ phát triển, **tái sử dụng**: khi chúng ta muốn thêm một chức năng nào đó thì việc lập trình theo một mô hình sẽ dễ dàng hơn vì chúng ta đã có chuẩn để tuân theo. Và việc **sử dụng lại** khi có sự thay đổi giữa hai môi trường (Winform sang Web form …) thì chỉ việc thay đổi lại lớp GUI.

Dễ bàn giao. Nếu mọi người đều theo một **quy chuẩn** đã được định sẵn, thì công việc bàn giao, tương tác với nhau sẽ dễ dàng hơn và tiết kiệm được nhiều thời gian.

Dễ phân phối khối lượng công việc. Mỗi một nhóm, một bộ phận sẽ nhận một nhiệm vụ trong mô hình 3 lớp. Việc phân chia rõ ràng như thế sẽ giúp các lập trình viên kiểm soát được khối lượng công việc của mình.

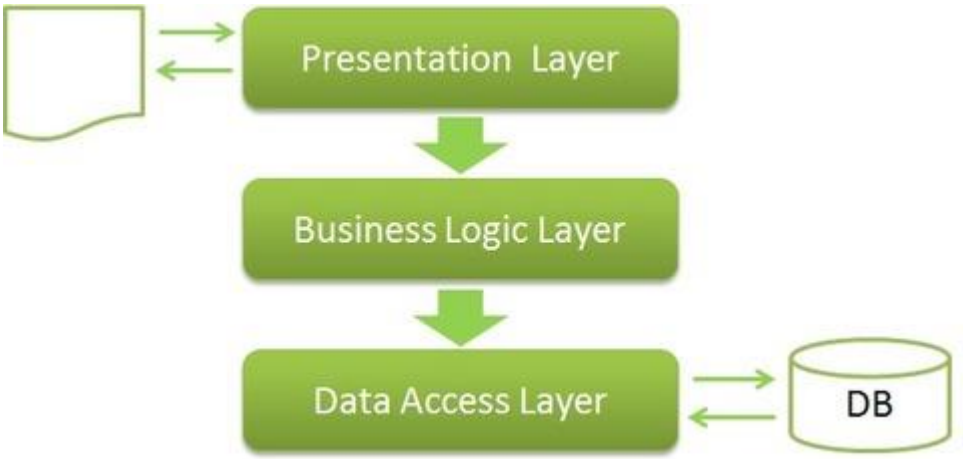
2.2. Xây dựng ứng dụng theo mô hình 3 tầng (3 Tier)

Có rất nhiều cách đặt tên cho các thành phần của 3 lớp như:

Cách 1: **GUI, BUS, DAL**

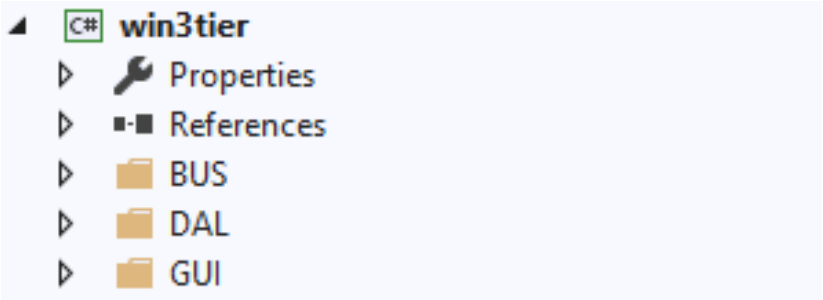
Cách 2: **GUI, BLL, DAO, DTO**

Cách 3: **Presentation, BLL, DAL**



Hình 2. 3 Mô hình 3 lớp

Trong chương này, chúng tôi sẽ trình bày xây dựng mô hình 3 lớp theo cách 1, trình bày 3 lớp theo tính Logic, mà không trình bày mỗi lớp thành 1 dự án riêng như ***Hình 2.4***



Hình 2. 4 Tạo mô hình 3 lớp Logic trên một dự án

Cách triển khai ***mô hình 3 lớp*** thường là tạo ra ***3 dự án***, một dự án là BUS loại dự án Library , một dự án là DAL cũng có loại dự án Library, một dự án là GUI. Tuy nhiên, để cho đơn giản, chúng tôi chỉ tạo ra dự án dễ theo dõi và chúng tôi sẽ giải 2 bài toán theo mô hình 3 lớp theo cách sau:

- Bài toán nhập vào 2 số nguyên, tính tổng của hai số.
- Bài toán giải phương trình bậc nhất 1 ẩn.

Phân tích tính tổng 2 số theo mô hình:

- **Data**: nhập từ bàn phím
- **GUI**: Form, 3 Textbox: 2 là nhập, 1 xuất kết quả, có nút ra hiệu xuất kết quả.
- **Data Access**: chuyển đổi dữ liệu từ chuỗi (string) sang số.
- **Bussiness Layer**: 1 lớp tính tổng 2 số, phương thức khởi dựng (2 textbox,2 số nguyên, không tham số.), phương thức tính tổng 2 số.

Phân tích giải phương trình bậc nhất theo mô hình:

- **Data**: nhập từ bàn phím
- **Data Access Layer**: Chuyển đổi dữ liệu từ chuỗi sang số thực.
- **Business Logic Layer**: Có phương thức khởi dựng không tham số, có tham số (2 texbox, 2 số thực), phương thức tìm nghiệm của pt.
- **GUI**: form có 3 textbox, 2 text nhập, 1 nghiệm hoặc thông báo vô nghiệm.

2.2.1. Tầng giao diện (Presentation tier) (GUI)

Presentation Layer là lớp tương tác với người sử dụng, lớp này được sử dụng chủ yếu để **nhận dữ liệu** của người sử dụng và truyền sang **Business Logic Layer** để thực hiện những thao tác xử lý dữ liệu sau trả về từ **Business Logic Layer** được trình bày sao cho người dùng có thể hiểu được.

Lớp này trong lập trình trên môi trường Windows chính là các Form, để giao tiếp với người dùng.

Trong lớp này cần phải mô tả cấu trúc các form có trong dự án theo mẫu sau:

Bảng 2. 1 Mẫu mô tả giao diện

STT	Tên điều khiển	Kiểu điều khiển	Hiển thị	Ghi chú

Đối với 2 ví dụ ta lần lượt có bảng mô tả trong ***Bảng 2.2*** và ***Bảng 2.3***.

Mô tả cho bài toán tính tổng hai số:

Bảng 2. 2 Mô tả giao diện bài toán tính tổng 2 số.

STT	Tên điều khiển	Kiểu điều khiển	Hiển thị	Ghi chú
1	frmNhap2so	Form		
2	label1	Label	Số thứ nhất:	
3	label2	Label	Số thứ hai:	
4	label3	Label	Nhập 2 số	
5	label4	Label	Kết quả:	
6	txtSothu1	TextBox		
7	txtSothu2	TextBox		
8	txtKetqua	TextBox		
9	btnXoatextbox	Button	Xóa TextBox	
10	btnOK	Button	Tính tổng	

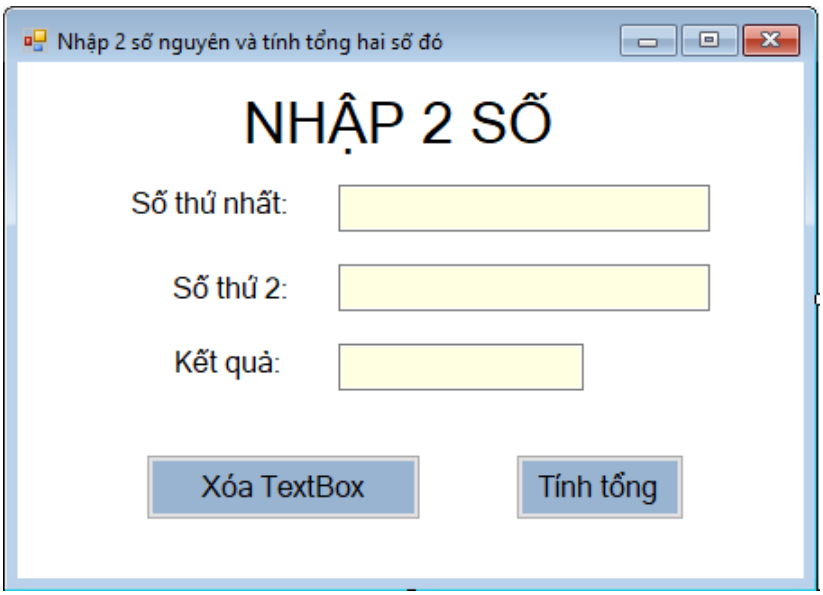
Trong ***Bảng 2.2*** có mô tả 1 **Form**, 3 **Label**, 3 **TextBox** và 2 **Button**, mỗi **Button** sẽ thực hiện một công việc riêng: Xóa nội dung trong **TextBox** và thực hiện tính tổng 2 số. Với các điều khiển **Label** và điều khiển **Button** có các **giá trị hiển thị** tương ứng, những giá trị hiển thị này cần phải được cố định trong suốt quá trình thực thi dự án. Với mô tả giao diện này, khi thiết kế giao diện trên Visual Studio sẽ có được Form như ***Hình 2.5***.

Viết code cho sự kiện Click của nút btnXoatextbox:

```
private void btnXoatextbox_Click(object sender, EventArgs e)
    {
        TextBox[] txt = new TextBox[] {txtKetQua,txtSothu1,txtSothu2 };
        DAL.XoaTextBox.Xoa(txt);       }
```

Viết code cho sự kiện Click của nút btnOK:

```
private void btnOK_Click(object sender, EventArgs e)
    {
        // Tao doi tuong conghaiso ten la obj
        BUS.CongHaiSo objCongHaiSo = new BUS.CongHaiSo(txtSothu1,txtSothu2);
        txtKetQua.Text = objCongHaiSo.Tinhtong().ToString();
    }
```



Hình 2. 5 Form nhập 2 số nguyên và tính tổng hai số.

Mô tả cho bài toán giải phương trình bậc nhất 1 ẩn:

Bảng 2. 3 Mô tả giao diện bài toán giải phương trình bậc nhất một ẩn

STT	Tên điều khiển	Kiểu điều khiển	Hiển thị	Ghi chú
1	Ptbacnhat	Form	Ptbacnhat	
2	label1	Label	X +	
3	label2	Label	= 0	
4	label3	Label	KẾT QUẢ:	

5	label4	Label	GIẢI PHƯƠNG TRÌNH BẬC NHẤT 1 ẨN	
6	txtA	TextBox		
7	txtB	TextBox		
8	txtKetqua	TextBox		
9	btnXoa	Button	Xóa dữ liệu	
10	btnGiaiPT	Button	Giải PT	

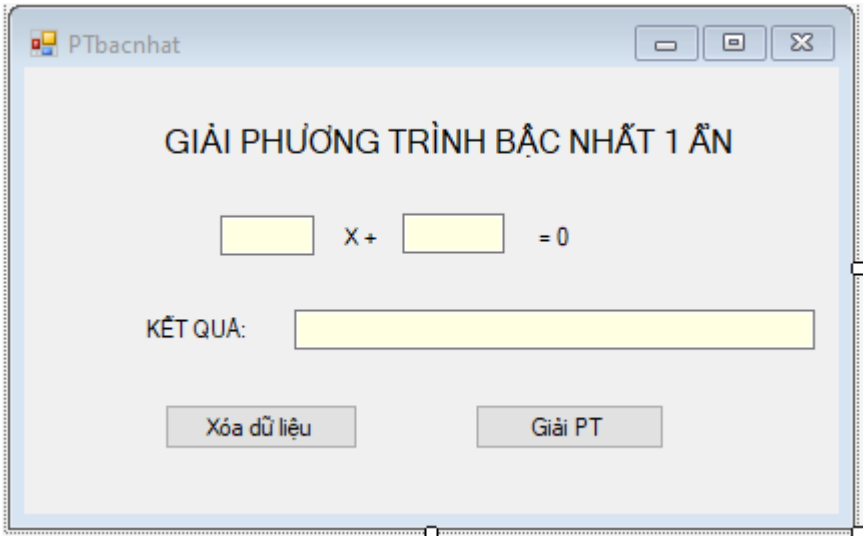
Viết code cho sự kiện Click của nút btnXoa:

```
private void btnXoa_Click(object sender, EventArgs e)
{
    TextBox[] txt = new TextBox[] { txtA, txtB, txtKetqua };
    DAL.XoaTextBox.Xoa(txt);
}
```

Viết code cho sự kiện Click của nút btnGiaiPT:

```
private void btnGiaiPT_Click(object sender, EventArgs e)
{
    BUS.GiaiPTbac1 obj = new BUS.GiaiPTbac1(txtA, txtB);
    txtKetqua.Text = obj.TimNghiemPTbac1();
}
```

Trong ***Bảng 2.3*** ta thấy rằng, bài toán giải phương trình bậc nhất 1 ẩn có 10 điều khiển, bao gồm 1 **Form**, 4 **Label**, 3 **TextBox** và 2 **Button**. Với mô tả giao diện này, khi thiết kế giao diện trên Visual Studio sẽ có được Form như ***Hình 2.6***



Hình 2. 6 Form giải phương trình bậc nhất 1 ẩn.

Trên thực tế, việc viết Code cho các nút trên Form thường được triển khai khi đã xây dựng xong các lớp dưới (lớp **BUS** và lớp **DAL**), do vậy đôi khi người ta cũng gọi lớp GUI là **lớp ứng dụng** vì người dùng sẽ thao tác trên lớp này.

Với các bài toán có nhiều hơn 1 Form thì để mô tả giao diện, mỗi Form tương ứng sẽ được mô tả bằng một bảng giống như bảng 2.1

2.3.2. Tầng xử lý (Business tier): BUS

Lớp (tầng) xử lý nghiệp vụ (Business Logic Layer)

Business Logic Layer (BUS) hoạt động như một cầu nối giữa **Presentation Layer** và lớp dữ liệu **Data Layer (DAL)**. Tất cả thông tin mà người dùng nhập vào được truyền đến **Business Logic Layer**.

Với bài toán **tính tổng 2 số** ta xây dựng lớp nghiệp vụ là việc cộng tổng hai số đó lại như sau:

```
using System.Windows.Forms;
namespace win3tier.BUS
{
    public class CongHaiSo
```

```
    {
        public int So1 { get; set; }
        public int So2 { get; set; }
        //phuong thuc khoi dung khong tham so (contructor no parameter)
        public CongHaiSo()
        {}
        //phuong thuc khoi dung co tham so
        public CongHaiSo(TextBox txtso1, TextBox txtso2)
        {
            So1 = DAL.Chuyendoichuoithanhsonguyen.Chuyendoi(txtso1.Text);
            So2 = DAL.Chuyendoichuoithanhsonguyen.Chuyendoi(txtso2.Text);
        }
        public CongHaiSo(int so1, int so2)
        {
            So1 = so1;So2 = so2;
        }
        public int Tinhtong()
        {
            return So1 + So2;
        }
    }
}
```

Thấy rằng trong **lớp nghiệp vụ** của bài toán tính tổng hai số chỉ có 1 lớp (class) CongHaiSo, có hai thuộc tính So1, So2, **phương thức khởi dựng** (contructor) không tham số, hai phương thức khởi dựng với 2 tham số, một phương thức tính tổng 2 số.

Bài toán giải phương trình bậc nhất 1 ẩn được xây dựng lớp nghiệp vụ như sau: Nghiệp vụ của bài toán này chính là việc giải phương trình bậc nhất 1 ẩn.

```
using System.Windows.Forms;
```

```
namespace win3tier.BUS
{
   public class GiaiPTbac1
   {
      public double A { get; set; }
      public double B { get; set; }
      public GiaiPTbac1() { }
      public GiaiPTbac1(TextBox txtA,TextBox txtB)
      {
         A = DAL.ChuyenStringsangThuc.Chuyendoi(txtA.Text.ToString());
         B = DAL.ChuyenStringsangThuc.Chuyendoi(txtB.Text.ToString());
      }
      public GiaiPTbac1(double a, double b)
      {
         A = a;B = b;
      }
      public string TimNghiemPTbac1()
      {
         string ketqua;
         if (A==0)
         {
            ketqua = "Phương trình vô nghiệm.";
         }
         else
         {
            ketqua = " Phương trình có nghiệm là: " + (-B / A).ToString();
         }
         return ketqua;
```

```
    }
  }
}
```

Ta thấy rằng, bài toán này được xây dựng với lớp nghiệp vụ chỉ có 1 lớp (class) GiaiPTbac1 gồm 2 thuộc tính, 1 phương thức khởi dựng không tham số, 2 phương thức khởi dựng có 2 tham số, một phương thức tìm nghiệm phương trình bậc nhất trả về kiểu chuỗi.

2.2.3. Tầng dữ liệu (Data Access Layer DAL)

Lớp dữ liệu dùng để tạo kết nối cơ sở dữ liệu, với bài toán kết nối cơ sở dữ liệu trên một hệ quản trị cơ sở dữ liệu thì đây là lớp tạo ra các thực thể dữ liệu, hoặc chuẩn hóa dữ liệu sẽ được làm ở đây vì ta sẽ bỏ qua lớp **DTO**.

Với 2 bài toán ở ví dụ trong chương này, dữ liệu được nhập từ bàn phím dạng chuỗi, do đó lớp dữ liệu của hai bài toán này lần lượt như sau:

Lớp dữ liệu của bài toán tính tổng hai số chúng tôi sẽ xây dựng 2 lớp (class).

Lớp dùng để xử lý dữ liệu, **xóa trắng các Textbox**:

```
using System.Windows.Forms;
namespace win3tier.DAL
{
   public static class   XoaTextBox
   {
      public static void Xoa(TextBox[] txt)
      {
         for (int i = 0; i < txt.Length; i++)
         {
            txt[i].Text = "";
         }
      }
```

```
    }
}
```

Và lớp chuyển đổi chuỗi thành số nguyên, dùng để chuyển đổi dữ liệu do người dùng nhập từ bàn phím vào thành số nguyên:

```
namespace win3tier.DAL
{
    public static class Chuyendoichuoithanhsonguyen
    {
        public static int Chuyendoi(string str)
        {
            int ketqua;
            ketqua = int.Parse(str);
            return ketqua;
        }
    }
}
```

Bài toán giải phương trình bậc nhất 1 ẩn, lớp dữ liệu bao gồm: **xóa trắng các Textbox** và lớp chuyển đổi **chuỗi thành số thực**.

```
namespace win3tier.DAL
{
    public static class ChuyenStringsangThuc
    {
        public static double Chuyendoi(string str)
        {
            double ketqua;
            ketqua = double.Parse(str);
            return ketqua;       }    }}
```

Trên thực tế trong các dự án, người ta thường viết ***khai báo giao diện (interface[2])*** cho mỗi lớp trước khi thực thi.

Bài tập cuối chương 2.

Hãy viết chương trình giải tam giác, giải phương trình bậc 2, sắp xếp, tìm kiếm sử dụng phương pháp lập trình 3 lớp chia dự án thành 3 phần

- Phần dữ liệu, dùng để tạo, tải dữ liệu,
- Phần xử lý dùng để viết các thuật toán giải quyết bài toán theo cấu trúc lớp
- Phần 3 giao diện người dùng, hiển thị kết quả trên form.

Thực hành mô hình 3 lớp như sau, hãy viết thêm interface cho mỗi class:

STT	DAL	BUS	GUI
1	Nhập vào 2 số {interface,class}	Tính tổng, hiệu, tích, thương của 2 số {class,interface}	- Hiển thị giao diện nhập. - Hiển thị kết quả tổng, hiệu, tích thương {Form}
2	Nhập vào 3 số {interface, class}	3 số đó là 3 cạnh tam giác? Tam giác gì? Tính S, C tam giác đó	-Hiển thị giao diện nhập. -Hiển thị kết quả tính toán 3 số là tam giác? Tam giác gì, …
3	Nhập vào một dãy số nguyên {interface, class}	- Tính tổng các số nguyên. - Tìm số nguyên bé thứ nhì.	- Giao diện nhập - Giao diện tìm số nguyên bé thứ nhì. - Giao diện hiển thị các số nguyên tố trong mảng

[2] Một interface được hiểu như là 1 khuôn mẫu mà mọi **lớp thực thi** nó đều phải tuân theo. Interface sẽ định nghĩa phần "làm gì" (khai báo) và những lớp thực thi interface này sẽ định nghĩa phần "làm như thế nào" (định nghĩa nội dung) tương ứng.

		- Tìm các số nguyên tố	

Chương 3: XÂY DỰNG ỨNG DỤNG VỚI ADO.NET

3.1. Giới thiệu ADO.Net

Trong thực tế, có rất nhiều ứng dụng cần **tương tác với cơ sở dữ liệu**. **.NET Framework** cung cấp một **tập các đối tượng** cho phép **truy cập vào cơ sở dữ liệu**, tập các đối tượng này được gọi chung là ADO.NET.

ADO.NET tương tự với ADO, điểm khác biệt chính ở chỗ ADO.NET là một **kiến trúc dữ liệu rời rạc**, không kết nối (**Disconnected Data Architecture**). Với kiến trúc này, dữ liệu được nhận về từ cơ sở dữ liệu và được lưu trên vùng nhớ cache của máy người dùng. Người dùng có thể **thao tác trên dữ liệu** họ nhận về và chỉ **kết nối đến cơ sở dữ liệu** khi họ cần **thay đổi các dòng dữ liệu** hay **yêu cầu dữ liệu mới**.

Việc ***kết nối không liên tục*** đến cơ sở dữ liệu đã đem lại nhiều thuận lợi, trong đó điểm lợi nhất là việc giảm đi một lưu lượng lớn truy cập vào cơ sở dữ liệu cùng một lúc, tiết kiệm đáng kể tài nguyên bộ nhớ. Giảm thiểu đáng kể vấn đề hàng trăm ngàn kết nối cùng truy cập vào cơ sở dữ liệu cùng một lúc.

ADO.NET kết nối vào **cơ sở dữ liệu** để **lấy dữ liệu** và **kết nối trở lại** để **cập nhật dữ liệu** khi người dùng thay đổi chúng. Hầu hết mọi ứng dụng đều sử dụng nhiều thời gian cho việc **đọc và hiển thị dữ liệu**, vì thế **ADO.NET** đã cung cấp một **tập hợp con các đối tượng dữ liệu** không kết nối cho các ứng dụng để người dùng có thể đọc và hiển thị chúng mà **không cần kết nối vào cơ sở dữ liệu**.

Các **đối tượng ngắt kết nối** này làm việc tương tự đối với các ứng dụng Web.

Để có thể hiểu rõ được cách làm việc của ADO.NET, chúng ta cần phải nắm được một số khái niệm cơ bản về **cơ sở dữ liệu quan hệ** và **ngôn ngữ truy vấn dữ liệu**, như: khái niệm về **dòng, cột, bảng, quan hệ giữa các bảng, khóa chính, khóa ngoại** và cách **truy vấn dữ liệu trên các bảng** bằng **ngôn ngữ truy vấn SQL** : **SELECT, UPDATE, DELETE** … hay cách **viết các thủ tục** (**Store Procedure**) …. .

Một số loại kết nối hiện đang sử dụng:

1982 ra đời ***ODBC driver (Open Database Connectivity)*** của **Microsoft**. Chỉ truy xuất được **thông tin quan hệ**, không truy xuất được dữ liệu không quan hệ như : tập tin văn bản, email …Ta phải truy cập **ODBC** thông qua **DSN**.

Để truy cập được tất cả **Datastore**, dùng **OLEDB provider** thông qua **ODBC**. Là vỏ bọc của **ODBC** hoặc không. **OLEDB** dễ sử dụng hơn **ODBC**, nhưng chỉ có 1 số ít ngôn ngữ có thể hiểu được (**C++**), vì thế ra đời **ADO**. **OLEDB** là giao diện ở mức **lập trình hệ thống** để **quản lý dữ liệu**. **OLEDB** đơn giản chỉ là một tập các **giao diện COM** đóng gói thành các **system service** để quản trị các CSDL khác nhau. Gồm 4 đối tượng chính : **Datasource, Session, Command, Rowset.**

ADO là một **COM**, do đó được dùng với bất kỳ ngôn ngữ nào tương thích với **COM**. **ADO** không độc lập **OS**, nhưng độc lập ngôn ngữ : **C++,VB, JavaScript, VBScript** …Là vỏ bọc của **OLEDB** và **ADO** gồm 3 đối tượng chính : **Connection, Command, Recordset**.

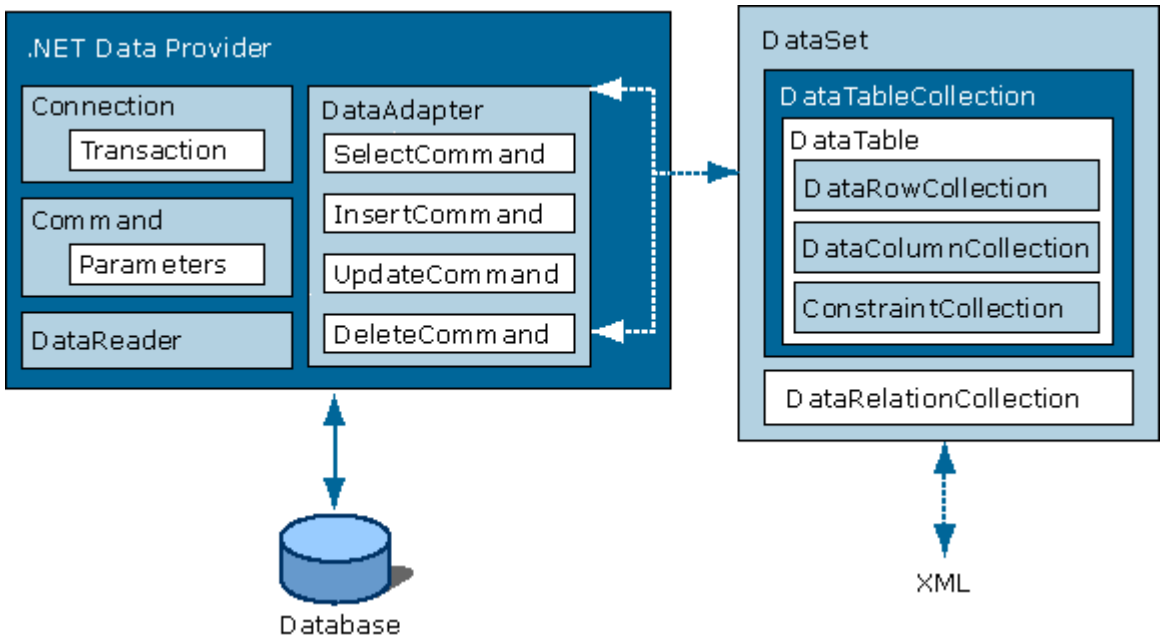
Remote Data Services (**RDS**) của **Microsoft** cho phép dùng **ADO** thông qua các **giao thức HTTP**, **HTTPS** và **DCOM** để **truy cập dữ liệu** qua **Web**.

Microsoft Data Access Components (MDAC) là tổ hợp của **ODBC**, **OLEDB, ADO** và cả **RDS**.

Ta có thể **kết nối dữ liệu** bằng một trong các cách: dùng **ODBC driver** (**DSN**), dùng **OLEDB** thông qua **ODBC** hoặc **OLEDB** không thông qua **ODBC**.

Kiến trúc ADO.NET

ADO.NET được chia ra làm hai phần chính rõ rệt, được thể hiện qua hình 3.1.



Hình 3. 1 Các thành phần của ADO.NET

Thành phần chính thứ nhất của **ADO.NET** là **DataSet** là thành phần chính cho đặc trưng ***kết nối không liên tục*** của kiến trúc **ADO.NET**. **DataSet** được thiết kế để có thể thích ứng với bất kỳ nguồn dữ liệu nào. **DataSet** chứa một hay nhiều đối tượng **DataTable** mà nó được tạo từ tập các **dòng và cột dữ liệu**, cùng với **khoá chính**, **khóa ngoại, ràng buộc** và các thông tin liên quan đến đối tượng **DataTable** này. Bản thân DataSet được dạng như một tập tin **XML**.

Thành phần chính thứ hai của **ADO.NET** chính là ***NET Provider Data***, nó chứa các đối tượng phục vụ cho việc **thao tác trên cơ sở dữ liệu** được hiệu quả và nhanh chóng, nó bao gồm một tập các đối tượng **Connection**, **Command**, **DataReader** và **DataAdapter**. Đối tượng **Connection** cung cấp **một kết nối** đến **cơ sở dữ liệu**, **Command** cung cấp **một thao tác** đến **cơ sở dữ liệu**, **DataReader** cho phép chỉ đọc dữ liệu và **DataAdapter** là cấu nối trung gian giữa **DataSet** và **nguồn dữ liệu**.

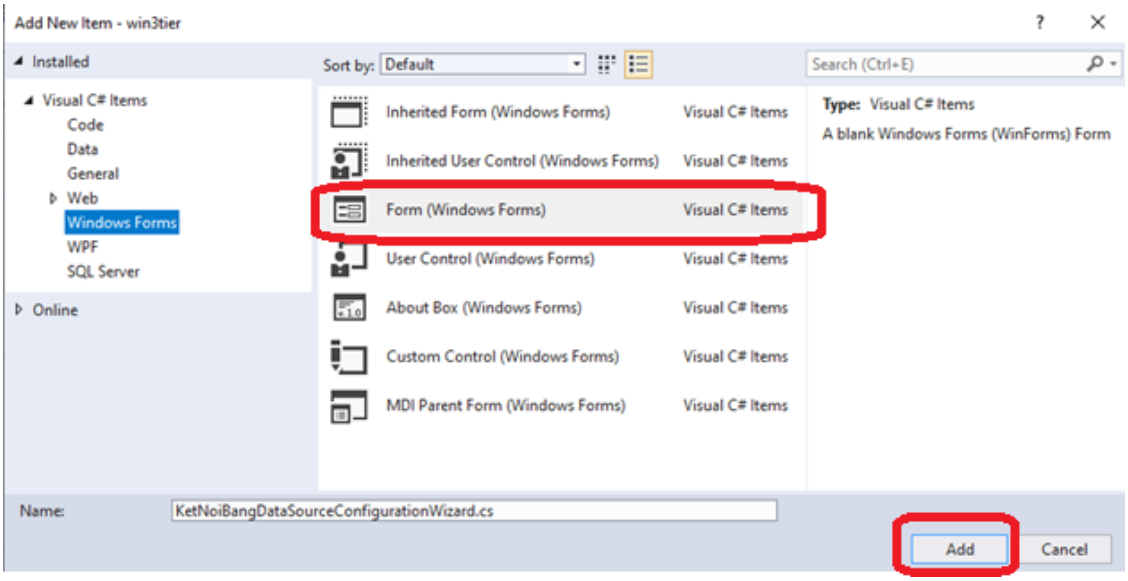
Mô hình đối tượng của **DataSet** bao gồm một tập các đối tượng **DataRelation** cũng như tập các đối tượng **DataTable**. Các đối tượng này đóng vai trò như các **thuộc tính** của **DataSet** như hình 3.2

3.2. Tạo kết nối sử dụng thẻ Data Source Configuration Wizard

Kết nối cơ sở dữ liệu bằng **Data Source Configuration Wizard** được thực hiện bằng **kéo thả chuột** mà **không cần Code** cũng có thể kết nối được với cơ sở dữ liệu.

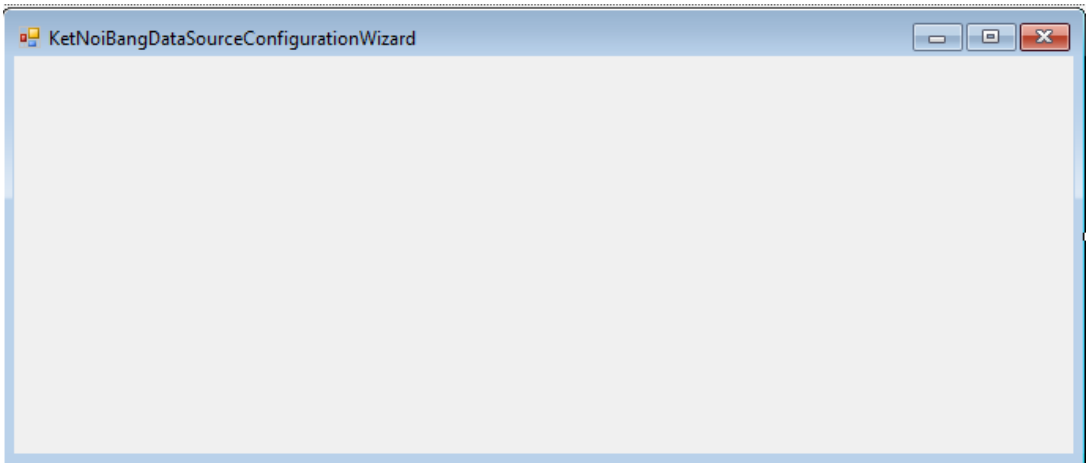
Để tạo form thông tin sinh viên, nội dung được lấy từ bảng SinhVien trong cơ sở dữ liệu, thực hiện các bước sau:

Bước 1. Tạo form có tên: **KetNoiBangDataSourceConfigurationWizard.cs**



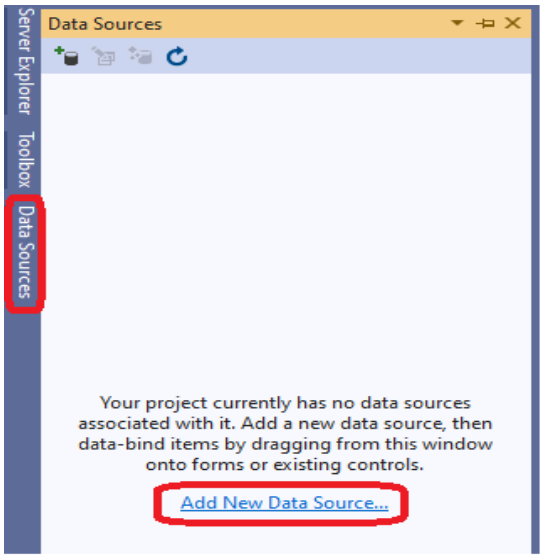
Hình 3. 2 Tạo form KetNoiBangDataSourceConfigurationWizard

Kết quả ta được form như ***Hình 3.8***



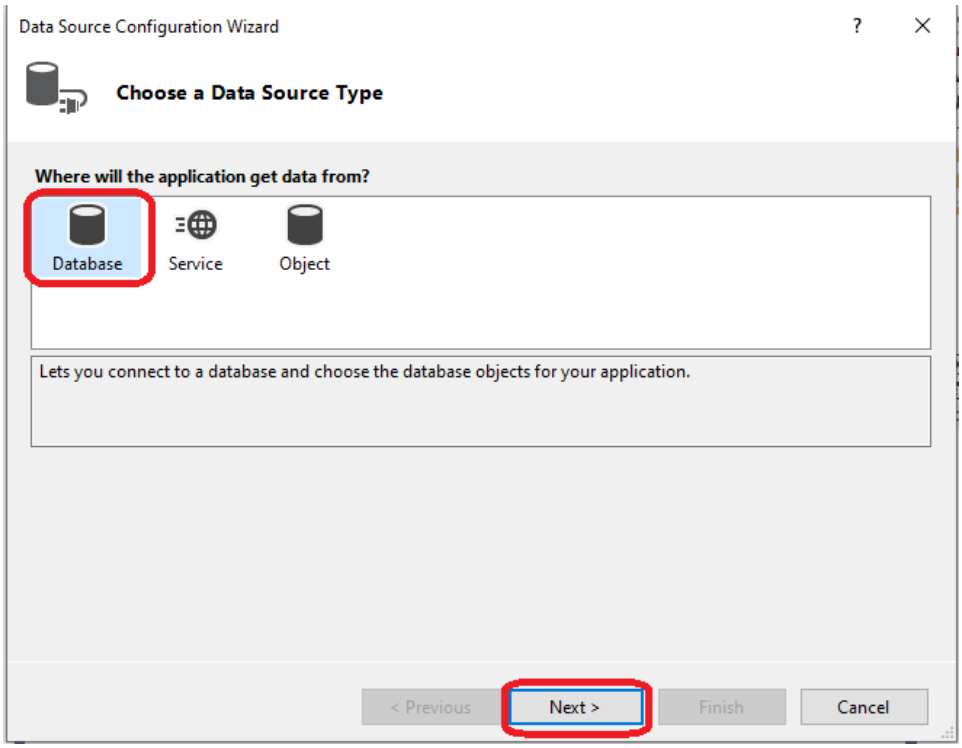
Hình 3. 3 Form KetNoiBangDataSourceConfigurationWizard

Bước 2. Chọn **Data Sources/ Add New Data Source …**



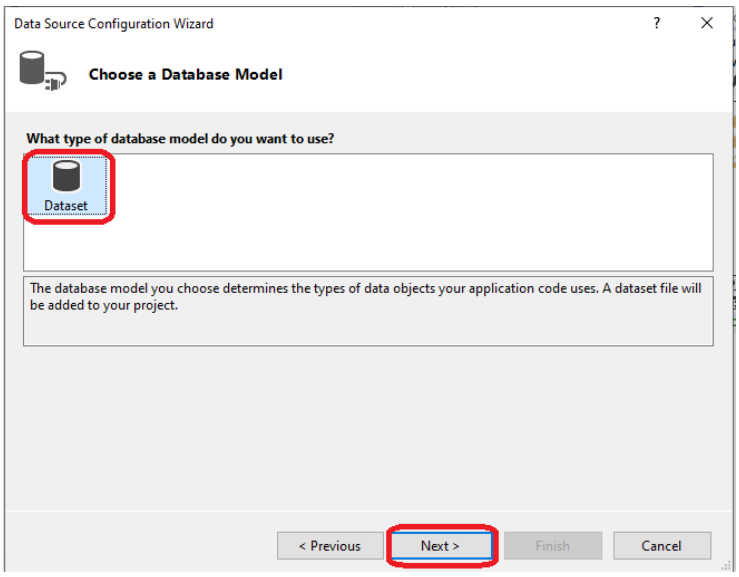
Hình 3. 4 Cửa sổ chọn nguồn dữ liệu

Bước 3: Cửa sổ **Data Source Configuration** hiện ra, chọn **Database ->Next**



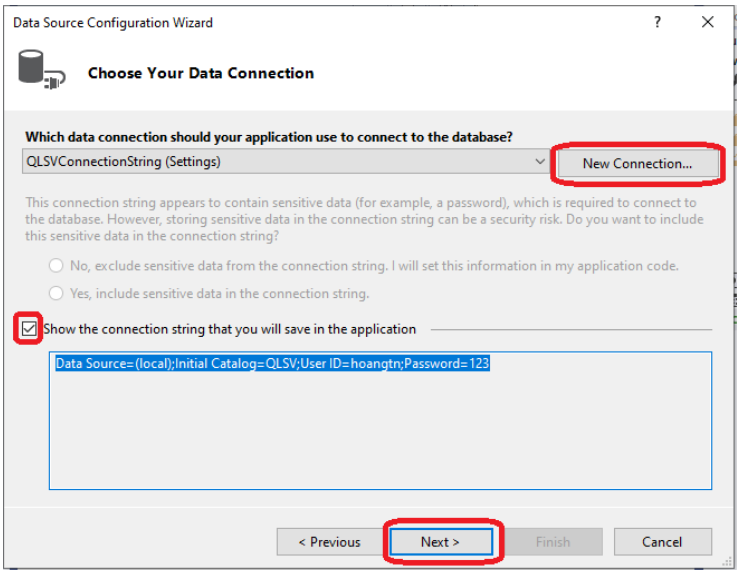
Hình 3. 5 Chọn kiểu dữ liệu nguồn

Bước 4: Chọn mô hình cơ sở dữ liệu là **Dataset-> Next**



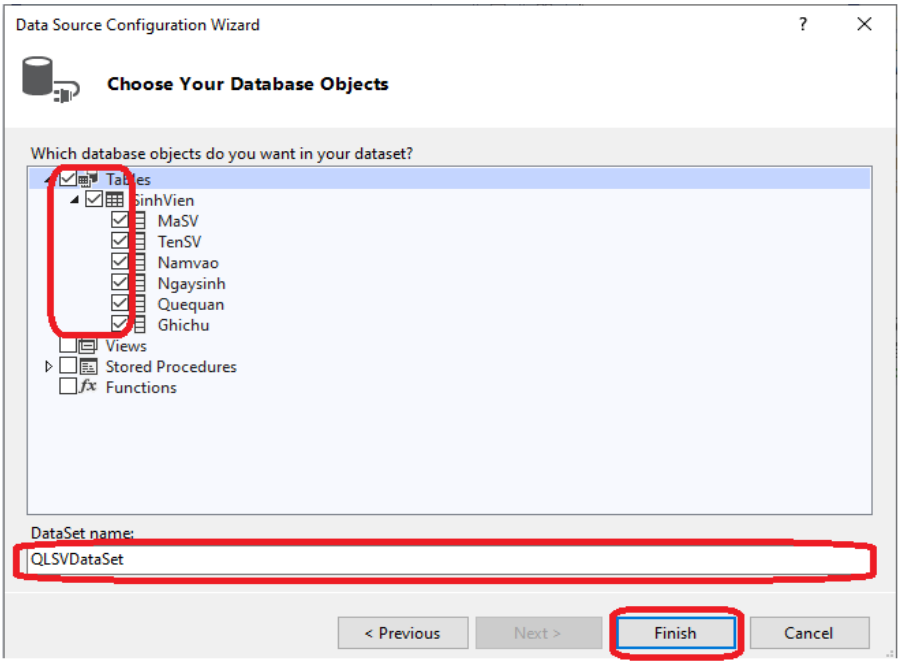
Hình 3. 6 Chọn kiểu mô hình cơ sở dữ liệu Dataset

Bước 5. Chọn **New Connection**, chọn **Next**, tích vào hộp kiểm nếu muốn xem chuỗi kết nối.



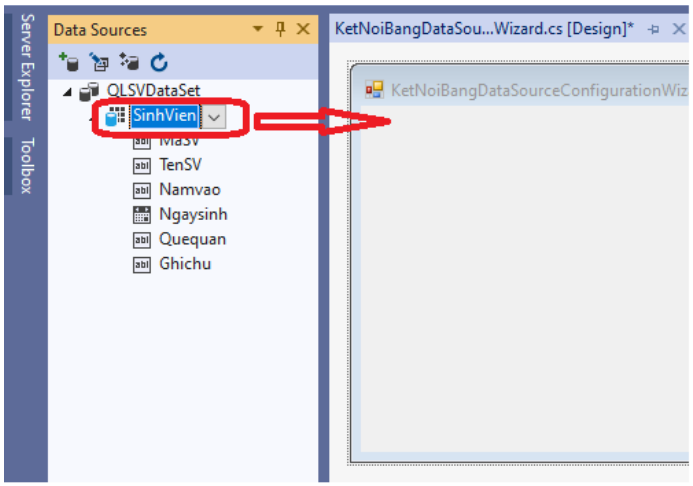
Hình 3. 7 Chọn/tạo chuỗi kết nối cơ sở dữ liệu

Bước 6. Chọn đối tượng cơ sở dữ liệu sẽ dùng: Bảng SinhVien



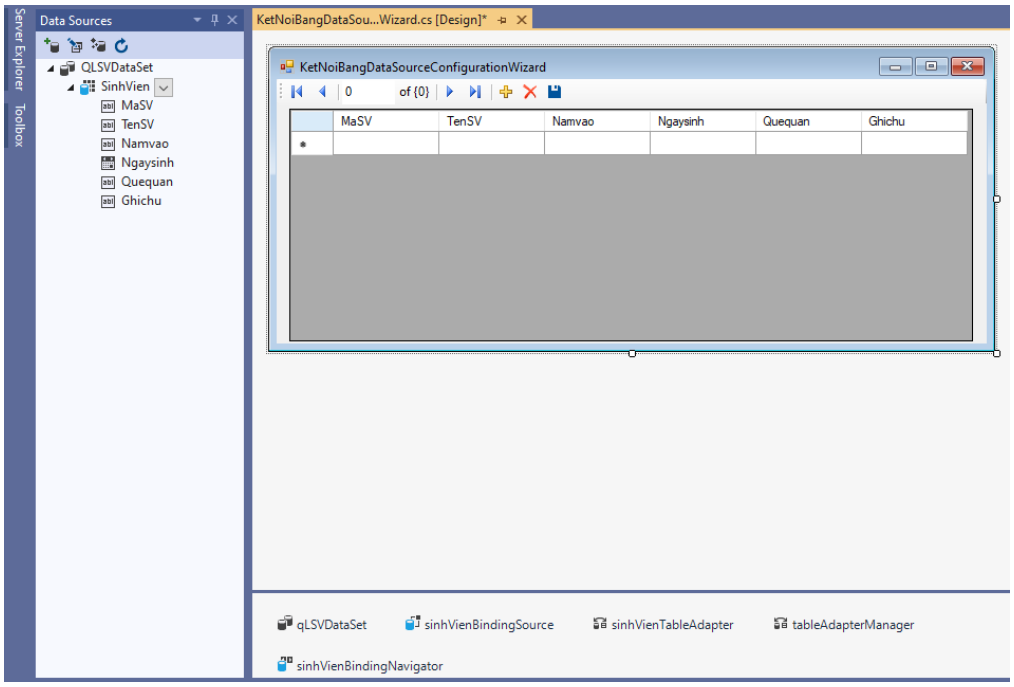
Hình 3. 8 Chọn kết nối tới bảng sinh viên

Bước 7, Kéo thả đối tượng Sinhvien sang form KetNoiBangDataSourceConfigurationWizard



Hình 3. 9 Kéo thả đối tượng dữ liệu sang form

Bước 8, Chỉnh lại Datagridview cho đẹp và ta được kết quả:



Hình 3. 10 Kết quả của việc tạo kết nối cơ sở dữ liệu tự động.

Như vậy, bằng cách tạo kết nối cơ sở dữ liệu bằng **Data Source Configuration Wizard** thì chương trình sẽ tự động tạo ra các đối tượng **Dataset, BidingSource, Adapter** …

3.3. Tạo kết nối đến cơ sở dữ liệu (CSDL) MS Access và SQL Server

3.3.1 Kết nối CSDL MS Access.

Để kết nối đến **CSDL MS Access**, chúng ta sử dụng không gian tên **System.Data.OleDb**. Tạo đối tượng **OleDbConnection** và gán chuỗi kết nối cho đối tượng kèm với đường dẫn chỉ đến file **CSDL Access**.

```
using System.Data.OleDb;

namespace win3tier.DAL
{
   public class KetNoiCSDLMSAccess
   {
      /// <summary>
      /// Kết nối đến cơ sở dữ liệu MS Access có đường dẫn là DuongDanFilemdb
      /// </summary>
      /// <param name="conn">Đối tượng kết nối MS Access</param>
      /// <param name="DuongDanFilemdb">Đường dẫn file cơ sở dữ liệu *.mdb</param>
      public void Open_DataAccess(ref OleDbConnection conn, string DuongDanFilemdb)

      {
         conn = new OleDbConnection();
         string s = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" + DuongDanFilemdb;

         conn.ConnectionString = s;

         conn.Open();
      }

   }
}
```

Hứng dữ liệu truy vấn vào trong Dataset

```
      /// <summary>
      /// Truy vấn dữ liệu, kết quả gửi vào Dataset
      /// </summary>
      /// <param name="sqlSelect">Câu lệnh truy vấn Sql</param>
      /// <param name="conn">Đối tượng kết nối cơ sở dữ liệu Access</param>
      /// <returns>Dataset chứa dữ liệu truy vấn</returns>
      public DataSet GetData(string sqlSelect, OleDbConnection conn)

      {
         DataSet ds = new DataSet();

         OleDbDataAdapter daShowData = new OleDbDataAdapter(sqlSelect, conn);
```

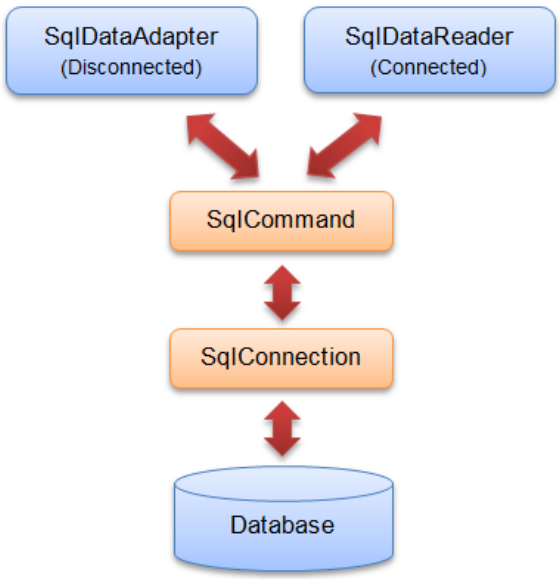
```
        daShowData.Fill(ds);
        return ds;
    }
```

Như vậy, các hệ cơ sở dữ liệu khác nhau đều cùng có các **DataAdapter, Connection, Command**… chỉ khác nhau tiền tố như ở **SQL Server là Sql, Microsoft Access là Oledb**.

3.3.2 Kết nối CSDL MS SQL Server.

Đối tượng SqlConnection

Đối tượng này dùng để kết nối chương trình ứng dụng với cơ sở dữ liệu trong Hệ quản trị cơ sở dữ liệu. Ví dụ



Hình 3. 11 Vị trí của đối tượng SqlConection

Đối tượng **SqlConnection** dùng để **kết nối cơ sở dữ liệu** có chuỗi kết nối cơ sở dữ liệu.

Có hai cách xác nhận kết nối client đến server

- *Xác nhận bằng quyền windows (Windows authentication)*

- **Integrated Security** = **SSPI** trong chuỗi kết nối

- *Xác nhận bằng quyền đăng nhập SQL Server (SQL Server authentication)*
  - **User ID/uid** và **Password/pwd** trong chuỗi kết nối

Bảng 3. 1 Các thuộc tính của SqlConnection

Tên thuộc tính	Kiểu
ConnectionString	string
ConnectionTimeout	int
Database	string
DataSource	string
ServerVersion	string
State	ConnectionState

Bảng 3. 2 Ý nghĩa của từng thuộc tính trong chuỗi kết nối (ConnectionString)

Data Source	Xác định máy chủ chứa cơ sở dữ liệu. Có thể là một máy chủ cục bộ, tên miền, hoặc một địa chỉ IP.
Initial Catalog	Tên cơ sở dữ liệu
Integrated Security	Đặt là SSPI nếu muốn truy cập cơ sở dữ liệu ủy quyền windows
User ID	Tên người dùng trong SQL Server.
Password	Mật khẩu của SQL Server User ID.

Chuỗi có dạng như sau:

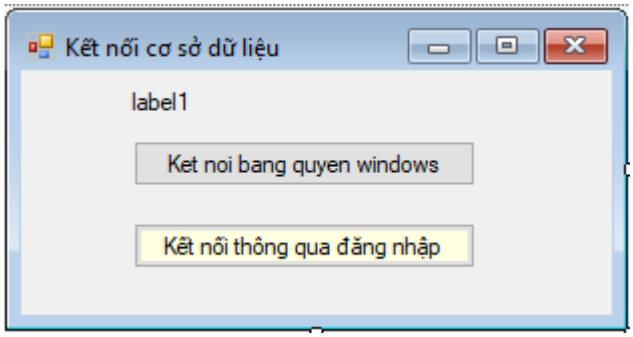
```
"Server=ServerAddress;
Database=DataBaseName;
```

```
User Id=Username;
Password = myPassword;"
```

Hoặc

```
“Data Source = ServerAddress;
Database=DataBaseName;
User  Id=Username;
Password = myPassword;”
```

Tạo kết nối đến CSDL QLSV trên máy tính có tên là DESKTOP-H68FGFI\SQLSERVER**:**



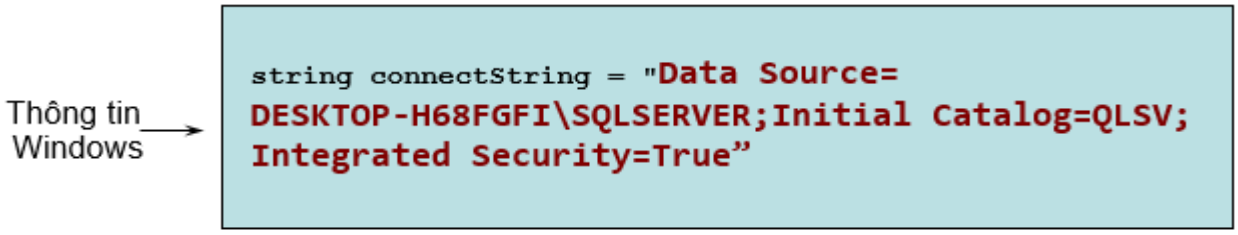
Hình 3. 12 Form kết nối CSDL.

Viết code cho nút kết nối, nhấn đúp chuột vào nút “Ket noi bang quyen windows”, sau đó chép đoạn code dưới đây, chú ý nhớ thêm không gian tên `using System.Data.SqlClient;`

Viết code cho nút **btnKetnoibangquyenwindows**.

```
private void btnKetnoibangquyenwindows_Click(object sender, EventArgs e)
      {
         SqlConnection Conn = new SqlConnection();
         Conn.ConnectionString =
            @"Data Source=DESKTOP-H68FGFI\SQLSERVER;Initial Catalog=QLSV;
Integrated Security=True";
         try
         {
            Conn.Open();
            lblTB.Text = "Ket noi bang windows thanh cong!";
         }
         catch (Exception ex)
```

```
        {
            lblTB.Text = "Ket noi that bai" + ex.ToString();
        }
    }
```

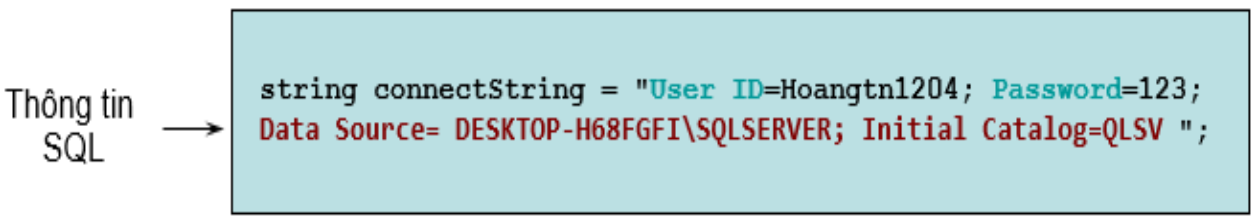


Hình 3. 13 Chuỗi kết nối (ConnectionString) sử dụng quyền Windows truy cấp

Ta thấy rằng, nút **btnKetnoibangquyenwindows** dùng để kết nối đến Sql Server với máy chủ có tên là "DESKTOP-H68FGFI\SQLSERVER" và sử dụng cơ sở dữ liệu có tên là "QLSV".

Viết code cho nút **btnKetNoiThongQuaDangNhap**

```
    private void btnKetNoiThongQuaDangNhap_Click(object sender, EventArgs e)
    {
        SqlConnection Conn = new SqlConnection();
        Conn.ConnectionString =
            @"Data Source=DESKTOP-H68FGFI\SQLSERVER;Initial Catalog=QLSV;User
ID=hoangtn1204;Password =123";
        try
        {
            Conn.Open();
            lblTB.Text = "Kết nối thông qua tài khoản hoangtn1204, mật khẩu 123 thành công!";
        }
        catch (Exception ex)
        {
            lblTB.Text = "Kết nối thất bại!!!" + ex.ToString();
        }
    }
```

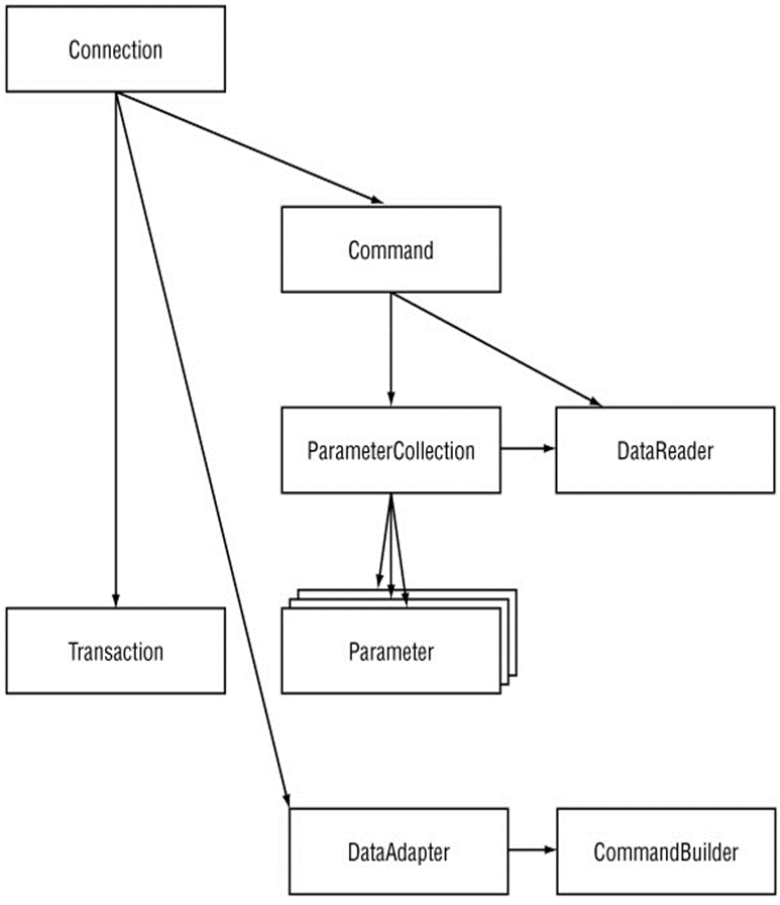


Hình 3. 14 Chuỗi kết nối (ConnectionString) sử dụng quyền đăng nhập SQL Server.

Ta thấy rằng, nút **btnKetnoiThongQuaDangNhap** dùng để kết nối đến Sql Server với máy chủ có tên là "DESKTOP-H68FGFI\SQLSERVER" và sử dụng cơ sở dữ liệu có tên là "QLSV" thông qua tài khoản đăng nhập của SQL Server, tên tài khoản: "hoangtn1204" và mật khẩu "123".

Sử dụng SqlConnection

Mục đích của việc tạo một **đối tượng SqlConnection** là để các mã lệnh **ADO.NET** khác có thể làm việc được với **database**. Các đối tượng **ADO.NET** khác, như **SqlCommand** và **SqlDataAdapter** dùng một **connection** như một tham số.



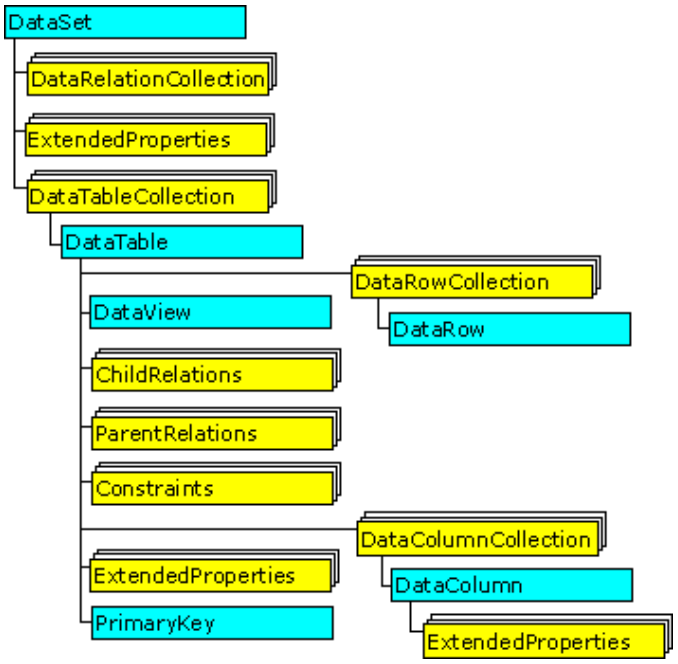
Hình 3. 15 Mô hình quản lý các lớp kết nối

Quá trình sử dụng **SqlConnection** gồm các bước sau:

- **Bước 1:** Tạo một SqlConnection.
- **Bước 2:** Mở connection.
- **Bước 3: Truyền connection** cho các đối tượng ADO.NET khác.
- **Bước 4**: Thực hiện các thao tác database với các đối tượng ADO.NET này.
- **Bước 5:** Đóng connection.

Hai bước tạo một **SqlConnection** và mở **connection** đã được thực hiện bằng 2 cách trong phần **Đối tượng SqlConnection,** ở bước thứ 3 có thể được thực hiện truyền biến connection từ form này sang form khác bằng cách áp dụng 1 trong 4 cách đã được trình bày ở đầu chương 1, mục 1.1.2 (độc giả có thể xem lại). **Bước 4** thao tác với **Database**, thường được dùng cho **SqlCommand**.

3.4. Đối tượng DataSet



Hình 3. 16 Mô hình đối tượng DataSet

Đối tượng DataTable và DataColumn

Ta có thể viết mã C# để tạo ra đối tượng **DataTable** hay nhận về từ kết quả của câu truy vấn đến **cơ sở dữ liệu**. **DataTable** có một số thuộc tính dùng chung như thuộc tính **Columns**, từ thuộc tính này ta có thể truy cập đến đối tượng **DataColumnsCollection** thông qua chỉ mục hay tên của cột để nhận về các đối tượng **DataColumn** thích hợp, mỗi **DataColumn** tương ứng với một cột trong một bảng dữ liệu.

3.5. Tạo và hiển thị dữ liệu từ DataSet

3.5.1 Khởi tạo đối tượng DataSet

```
DataSet dataset1 = new DataSet();
DataSet dataset2= new DataSet("Mydataset");
```

Trong phần khởi tạo này, đối tượng **dataset1** được tạo ra mà không đặt tên, đối tượng **dataset2** được tạo ra với tên là "Mydataset".

Một đối tượng **Dataset** chứa nhiều bảng **Table** gọi là tập **Tables**. Các Table này có thể được gọi ra theo tên (**Name**) hoặc chỉ số (**Index.**).

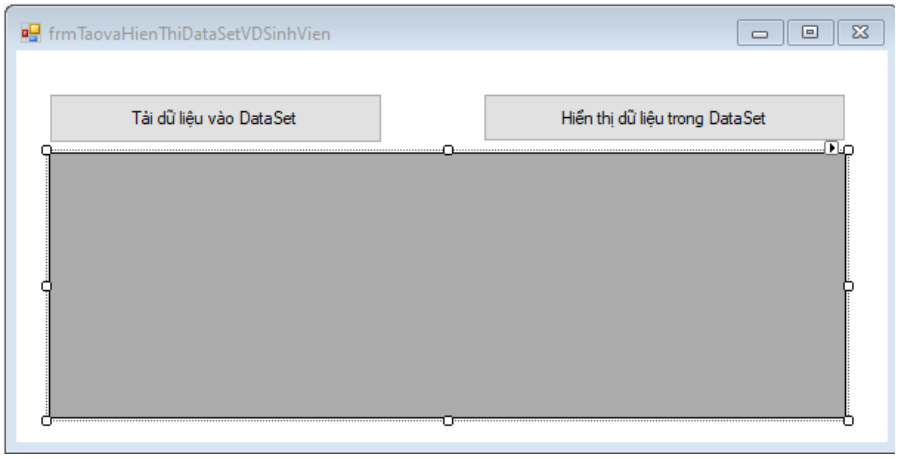
3.5.2 Chứa dữ liệu vào DataSet

Ví dụ, tạo form có cấu trúc như sau:

Bảng 3. 3 Cấu trúc Form SinhVien

STT	Tên điều khiển	Kiểu điều khiển	Hiển thị	Ghi chú
1	frmTaovaHienThiDataSetVDSinhVien	Form		
2	button1	Button	Tải dữ liệu vào DataSet	
3	button2	Button	Hiển thị dữ liệu trong DataSet	
4	dataGridView1	DataGridView		

Kết quả sẽ có form như hình



Hình 3. 17 Form dữ liệu sinh viên được hiển thị trên DataGridView

Viết hàm Nạp dữ liệu vào DataSet từ cơ sở dữ liệu.

```
DataSet ds;
DataSet LoadData()
      {
         //Khai báo biến kết nối
         var conn = new SqlConnection("Data Source =.; Initial Catalog = QLSV; Integrated Security =
SSPI");
         //Mở kết nối
         conn.Open();
         //Khai báo câu lệnh select
         var cmd = "Select * from SinhVien";
         //Khai báo đối tượng DataAdapter
         var dataAdapter = new SqlDataAdapter(cmd, conn);
         //Khai báo đối tượng DataSet
         var dataSet = new DataSet();
         //Điền dữ liệu có được vào DataSet
         dataAdapter.Fill(dataSet, "SinhVien");
         //Đóng kết nối
         conn.Close();
         // Trả về giá trị dataSet
         return dataSet;       }
```

Trong phần nạp dữ liệu này, ta thấy rằng, một đối tượng thuộc lớp **SqlConection** được tạo ra kết nối với Sql Server trên máy chủ hiện tại (dấu chấm (.) là để chỉ localhost), cơ sở dữ liệu QLSV, kết nối thông qua quyền windows.

Sau khi mở kết nối, có một câu lệnh sql truy vấn toàn bộ dữ liệu của bảng SinhVien. Một đối tượng **DataAdapter** được tạo ra để nhận câu lệnh truy vấn, sử dụng biến kết nối cơ sở dữ liệu đã được tạo ra ở trên. Một đối tượng **DataSet** được tạo ra *dataSet* để ***hứng dữ liệu***

truy vấn được truy vấn, bảng trực tiếp nhận dữ liệu, nằm trong DataSet đó được đặt tên là ***SinhVien***.

Viết code cho nút button1 (Tải dữ liệu vào Dataset):

```
/// <summary>
/// Load dữ liệu sinh viên vào DataSet
/// </summary>
/// <param name="sender"></param>
/// <param name="e">Sự kiện click chuột</param>
private void button1_Click(object sender, EventArgs e)
{
    ds = LoadData();
}
```

Hiển thị dữ liệu đã nạp trong **Dataset** lên **DataGridView**. Viết code cho nút button2 (Hiển thị dữ liệu trong DataSet)

```
/// <summary>
/// Hiển thị dữ liệu từ DataSet lên DataGridView
/// </summary>
/// <param name="sender"></param>
/// <param name="e">Sự kiện click chuột</param>
private void button2_Click(object sender, EventArgs e)
{
    dataGridView1.DataSource = ds.Tables["SinhVien"];
}
```

Để đếm số dòng trên bảng ta có thể thực hiện:

```
int sodong = dataset.Tables[0].Rows.Count;
```

3.5.3 Tạo dữ liệu cho DataSet

Ngoài cách **nạp dữ liệu từ datatable**, cũng có thể tạo dữ liệu động cho **DataTable** thông qua các **collection Columns** và **Rows**.

Kiểu dữ liệu **DataColumn** chứa đầy đủ các thuộc tính cần thiết để tạo ra một mô hình dữ liệu hoàn chỉnh cho **DataTable**. Ta có thể tạo một column dùng làm ID với chỉ số tự động tăng bắt đầu từ 1, không cho phép null và là duy nhất như sau:

```
DataColumn col;
public DataColumn CreateDataColumnID()
{
    col = new DataColumn("ID", typeof(int));
    col.AllowDBNull = false;
    col.AutoIncrement = true;
    col.AutoIncrementSeed = 1;
```

```
        col.Unique = true;
        return col;
    }
```

Hoặc tạo ra những Column khác như sau:

```
DataColumn col;
public DataColumn CreateDataColumnString(string name)
    {
        col.ColumnName = name;
        col.DataType = typeof(string);
        return col;       }
```

Ví dụ tạo một bảng **NhanVien** có các trường **MaNV, TenNV, NgaySinh, QueQuan** lần lượt có các kiểu dữ liệu tương ứng là **int, string** và **DateTime, string** như sau:

```
    public DataTable AutoCreatDataNhanVien()
    {
        DataTable table = new DataTable("NhanVien");
        DataColumn col = new DataColumn("MaNV", typeof(int));

        col.AllowDBNull = false;
        col.AutoIncrement = true;
        col.AutoIncrementSeed = 1;
        col.Unique = true;

        table.Columns.Add(col);
        table.Columns.Add("TenNV", typeof(string));
        table.Columns.Add("NgaySinh", typeof(DateTime));
        table.Columns.Add("QueQuan", typeof(string));
        return table;
    }
```

Gán **dữ liệu** cho bảng **NhanVien**:

DataTable NhanVien ta vẫn là rỗng vì chưa có dữ liệu (chỉ có mô hình dữ liệu). Để tạo một **DataRow** gọi phương thức **DataTable.NewRow().** Phương thức này trả về một **DataRow** với các ô chứa dữ liệu tương ứng với các cột của **DataTable**:

```
    public void AutoSignDataNhanVien(ref DataTable dtNhanvien)
    {
        DataRow newRow =dtNhanvien.NewRow();
        newRow["ID"] = 1; // remove this line
        newRow["Name"] = "Nguyễn Thành Nam";
        newRow["Birthday"] = new DateTime(1992, 3, 4);
        dtNhanvien.Rows.Add(newRow);
        newRow["Name"] = "Nguyễn Thành Nam";
    }
```

Thêm **DataTable** vào **DataSet** như sau:

```
    public void AddDataTable2DataSet(ref DataSet ds,DataTable dt)
{
    ds.Tables.Add(dt);
}
```

3.6. Đọc dữ liệu vào các điều khiển cơ bản

Từ dữ liệu đã có, gọi ra từ **Cơ sở dữ liệu**, sau đó truyền giá trị cho thuộc tính giá trị của các điều khiển.

3.6.1 Đưa dữ liệu vào DataGridView từ DataSet

Dữ liệu được đọc vào DataGridView, gán đối tượng DataTable vào thuộc tính **DataSource** của **DataGridView**.

```
public static void AssignValueDataGridView(ref
    DataGridView dataGridView,DataTable dt)
{
    dataGridView.DataSource = dt;
}
```

3.6.2 Dữ liệu đọc vào TextBox:

Gán giá trị vào thuộc tính Text của TextBox

```
    public static void AssignValueTextBox(
    ref TextBox txt,string value)
{
    txt.Text = value;
}
```

Bài tập cuối chương 3.

Áp dụng mô hình 3 tầng ở chương 2 và kiến thức đã học ở chương 3 để làm các bài tập tổng hợp sau:

Bài tập 1. Tạo cơ sở dữ liệu trong SQL Server có tên là QLBH, tạo bảng Hàng, Khách hàng, Nhân viên, Đặt hàng, Hóa đơn, Hóa đơn chi tiết.

Bài tập 2. Viết các thủ tục:

- Viết các **thủ tục** thêm mới, sửa, xóa, tìm kiếm theo mã hàng cho bảng Hàng

- Viết các **thủ tục** thêm mới, sửa, xóa, tìm kiếm theo mã khách hàng cho bảng Khách hàng

- Viết các **thủ tục** thêm mới, sửa, xóa, tìm kiếm theo mã nhân viên cho bảng Nhân viên

- Viết các **thủ tục** thêm mới, sửa, xóa, tìm kiếm theo mã hóa đơn cho bảng Đặt hàng.

- Viết các **thủ tục** thêm mới, sửa, xóa, tìm kiếm theo mã khách hàng cho bảng hóa đơn

- Viết các **thủ tục** thêm mới, sửa, xóa, tìm kiếm theo mã khách hàng cho bảng Hóa đơn chi tiết

- Viết thủ tục thống kê nhân viên trên 50 tuổi.

- Viết thủ tục thống kê những hàng mà khách hàng (biết mã) đã mua theo thời gian.

- Viết thủ tục thống kê những mặt hàng có giá trị lớn hơn một giá trị nhập vào.

-Viết thủ tục thống kê những đơn hàng đã mua theo ngày.

Bài tập 3. Xây dựng tầng GUI với các form:

- Form Hàng để **cập nhật** thông tin cho hàng hóa.

- Form Khách hàng để **cập nhật** thông tin cho khách hàng.

- Form Nhân viên để **cập nhật** thông tin cho nhân viên.

- Form Đặt hàng để **cập nhật** thông tin đặt hàng của khách hàng.

- Form Hóa đơn để **cập nhật** thông tin cho Hóa đơn.

- Form Hóa đơn chi tiết để **cập nhật** thông tin cho hóa đơn chi tiết.

Bài tập 4 Xây dựng tầng **BUS** với các lớp

- Tìm kiếm thông tin Hàng,
- Tìm kiếm thông tin Khách hàng,
- Tìm kiếm thông tin Nhân viên,
- Tìm kiếm thông tin Đặt hàng,
- Tìm kiếm thông tin Hóa đơn,
- Tìm kiếm thông tin hóa đơn chi tiết.
- Thống kê nhân viên trên 50 tuổi.
- Xuất hóa đơn theo ngày.
- Báo cáo danh sách mặt hàng khách hàng (đã biết mã) đã mua theo thời gian.

Bài tập 5. Xây dựng tầng **DAL** với các lớp

- Lớp kết nối cơ sở dữ liệu
- Lớp Hàng cập nhật thông tin Hàng
- Lớp Nhân viên cập nhật thông tin Nhân viên
- Lớp Đặt hàng cập nhật thông tin đặt hàng
- Lớp Hóa đơn cập nhật thông tin hóa đơn
- Lớp Hóa đơn chi tiết cập nhật thông tin hóa đơn chi tiết

Chương 4: QUAN HỆ GIỮA CÁC BẢNG

4.1. Các kiểu quan hệ

Sinh viên được tìm hiểu về các kiểu quan hệ trong cơ sở dữ liệu quan hệ, ở đây chủ yếu sử d ụng quan hệ 1 –n.

Nhằm đảm bảo CSDL không phát sinh ra những dữ liệu rác, những dữ liệu trùng lặp (tính toàn vẹn dữ liệu), thì Database thường tạo ra nhiều bảng dữ liệu - chia thông tin ra nhiều bảng - mỗi bảng này hướng quản lý một loại thông tin nào đó. Sau đó những bảng này kết hợp lại với nhau để có thông tin đầy đủ về đối tượng nào đó. Để làm điều này, bạn sẽ thiết lập những trường dữ liệu chung của các bảng từ đó hình thành mối liên hệ giữa các bảng. Phần này tìm hiểu về các mỗi liên hệ (relationship) giữa các bảng và cách sử dụng chúng trong CSDL

Khi tạo được mối liên hệ giữa các bảng thì bạn có thể tạo ra các truy vấn, các biểu mẫu và báo cáo hiện thị thông tin các nhiều bảng một lúc.

4.2. Sử dụng GridView để hiển thị quan hệ giữa các bảng

Đây coi như một bài tập trên lớp của sinh viên sau khi đã học xong chương 3, sinh viên gọi dữ liệu từ cơ sở dữ liệu và hiển thị dữ liệu có được lên **GridView**.

Có thể tùy biến tên cột trong **GridView**.

4.3. Tạo các quan hệ master/detail

Demo dưới sử dụng 2 bảng table: **Customers** và **Orders**. Hiển thị 1 **Gridview** là danh sách khách hàng, khi nhấn vào khách hàng nào sẽ liệt kê tất cả các đơn hàng của khách hàng đó. Hiển thị thông tin chi tiết của đơn đặt hàng: sản phẩm, địa chỉ, thời gian giao hàng, điện thoại...

Cần sử dụng các thư viện sau:

```
using System.ComponentModel
using System.Text
using System.Data.SqlClient
```

Tạo các kết nối đến database

```
SqlConnection con =new SqlConnection()
  public void taoketnoi()
{
      string strKetnoi = "Data Source=.;Initial Catalog=NORTHWND;Integrated Security=True";
      con.ConnectionString = strKetnoi;
      con.Open()
  }
  public void dongketnoi(){
      con.Close();}
   public Datatable _layDuLieu(string query){
      taoketnoi()
```

```
        Datatable datatable =new Datatable();
        SqlDataAdapter da =new SqlDataAdapter()
        da.SelectCommand = new SqlCommand(query, con)
        da.Fill(datatable)
        dongketnoi()
        return datatable
    }
```

Hàm chạy

```
void chay(){
Dataset dataset= New DataSet()
        DataTable customers = _layDuLieu("select CustomerID, CompanyName, ContactName, ContactTitle, Address, City, Country, Phone, Fax from Customers")
       DataTable  orders = _layDuLieu("select OrderID, CustomerID, OrderDate, ShipName, ShipAddress, ShipCity, ShipCountry from Orders")
        customers.TableName = "Customers"
        orders.TableName = "Orders"
        ' add table to dataset
        dataset.Tables.Add(customers)
        dataset.Tables.Add(orders)
        ' tao relation
        dataset.Relations.Add("Đơn hàng", customers.Columns("CustomerID"), orders.Columns("CustomerID"))
        GridControl1.DataSource = dataset
        GridControl1.DataMember = "Customers"
}
```

4.4. Tạo các biểu mẫu kiểu master/detail

Trong phần này sinh viên cần download dữ liệu mẫu `NORTHWND` và import vào hệ quản trị cơ sở dữ liệu SQL server.

Bài tập cuối chương 4

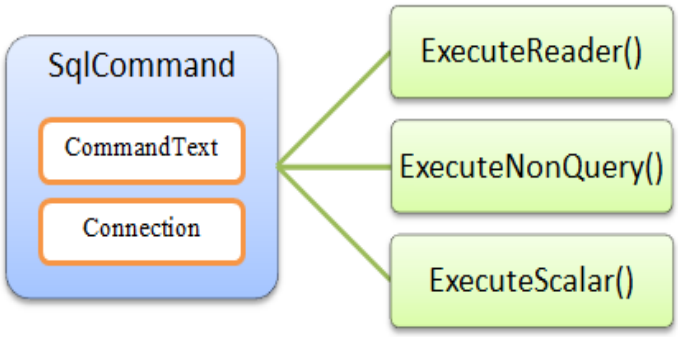
Hãy tạo ra các form để hiển thị dữ liệu trong cơ sở dữ liệu mẫu `NORTHWND`, mỗi bảng được hiện trên 1 grid view và trên môt form.

Chương 5: TRUY CẬP DỮ LIỆU VỚI ADO.NET

5.1. Đối tượng DataSet và DataAdapter

5.1.1 Đối tượng SqlCommand

Để sử dụng đối tương **SqlCommand**, cần phải có biến kết nối đến cơ sở dữ liệu, sử dụng đối tượng **SqlConnection**



Hình 3. 18 Mô hình đối tượng SqlComand.

Giả sử dự án cần kết nối đến cơ sở dữ liệu **QLSV** trên hệ quản trị cơ sở dữ liệu **SQL Server**, biết rằng **tài khoản truy cập** vào cơ sở dữ liêu **QLSV** là *hoangtn*, mật khẩu truy cập là **123**. Như đã trình bày trong chương 3, có hai cách kết nối vào cơ sở dữ liệu **QLSV**:

Cách 1: Sử dụng tên truy cập hoangtn, mật khẩu 123.

```
/// <summary>
/// Kết nối cơ sở dữ liệu QLSV bằng user hoangtn, mật khẩu 123
/// </summary>
/// <returns>Trả về True nếu kết nối thành công, trả về False nếu kết nối thất bại</returns>
public static bool KetNoiCoSoDuLieuQLSV(ref SqlConnection conn)
{
    bool kt = true;
    conn = new SqlConnection();
```

```
        conn.ConnectionString = "Data Source =.; Initial Catalog = QLSV; User ID = hoangtn;
Password=123";
        try
        {
            conn.Open();
        }
        catch
        {
            kt = false;
        }
        return kt;       }
```

Cách 2: Đăng nhập thông qua quyền của Windows.

```
    /// <summary>
    /// Kết nối cơ sở dữ liệu QLSV bằng quyền windows
    /// </summary>
    /// <returns>Trả về True nếu kết nối thành công, trả về False nếu kết nối thất bại</returns>
    public static bool KetNoiCoSoDuLieuQaQuyenWindowsQLSV(ref SqlConnection conn)
    {
        bool kt = true;
        conn = new SqlConnection();
        conn.ConnectionString = "Data Source =.; Initial Catalog = QLSV; Integrated Security = SSPI";
        try
        {
            conn.Open();
        }
        catch
        {
            kt = false;
```

```
        }
        return kt;
    }
```

Đối tượng **SqlCommand** cho phép chọn **kiểu tương tác** thực hiện với **database**. Ví dụ, thực hiện các lệnh **select, insert, modify** và **delete** các dòng trong một **table** của **database**. Đối tượng này có thể được dùng để hỗ trợ **mô hình quản lý dữ liệu ngắt kết nối** (**disconnected**).

Các thuộc tính của **Command**

- *Thuộc tính CommandText* là câu lệnh SQL hoặc tên thủ tục nội tại.
- *Thuộc tính CommandType* mặc định là **Text**, tương ứng **là câu lệnh** SQL. Là **StoredProcedure** khi giá trị của thuộc tính CommandText **là tên thủ tục nội tại**
- Thuộc tính **Connection** Đối tượng kết nối **CSDL**

Các **thuộc tính của tham số** của **Command**.

- **Parameter:** là các tham số.
- **Name**: Tên tham số. Ví dụ "@tenSV"
- **DbType**: Kiểu tham số. Ví dụ "Nvarchar"
- **Size**: Kích thước tham số .Ví dụ 50.
- **Value**: Giá trị của tham số. Ví dụ "Nguyễn Văn A"

Hình 4. 1 Mô hình hoạt động trao đổi dữ liệu từ ứng dụng với cơ sở dữ liệu.

5.1.2 Đối tượng DataAdapter

DataAdapter là một bộ gồm 4 đối tượng:

- **SelectCommand**: Cho phép lấy thông tin từ nguồn dữ liệu về.
- **InsertCommand**: Cho phép thêm dữ liệu vào bảng trong nguồn dữ liệu.

- **UpdateCommand**: Cho phép điều chỉnh dữ liệu của bảng trong nguồn dữ liệu.
- **DeleteCommand**: Cho phép xóa dữ liệu của bảng trong nguồn dữ liệu.

Các phương thức của **DataAdapter**

- Lấy dữ liệu từ nguồn: Sử dụng **DataAdapter** để lấy dữ liệu về cho các đối tượng
- **DataTable**: **Fill*(<DataTable>)***
- **DataSet**: **Fill*(<DataSet>)*** -->Dữ liệu lấy về **DataSet** dưới dạng các **dataTable** với tên mặc định là: **Table,Table1, Table2**. . .:
- Đổ dữ liệu vào **Datset** cho bảng **DataTable** nếu chưa có sẽ tạo mới:
- **Fill**(<DataSet>,<Tên dataTable>): Đẩy dữ liệu trong DataAdapter vào bảng có tên là "*Tên dataTable*" trong DataSet có tên là "*DataSet*".
- **Phương thức** trả về mẫu tin lấy về được.

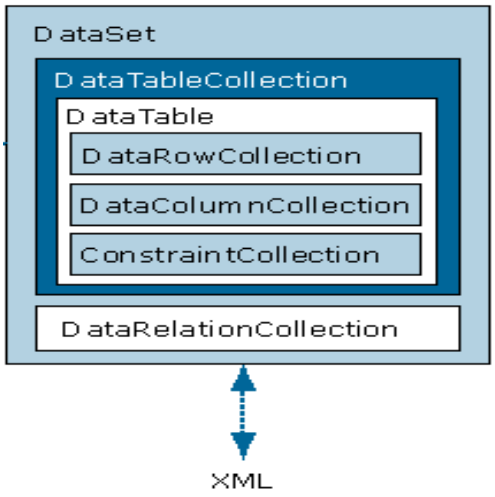
```
DataSet DS = new Dataset();
int so= DA.Fill(DS,"Sinhvien") ;
```

- Để **cập nhật dữ liệu** về nguồn dữ liệu.
  - **Update**(<mảng dòng>): **Cập nhật các dòng** (Các đối tượng DataRow) vào **nguồn dữ liệu**.
  - **Update**(<Dataset>): **Cập nhật** các thay đổi trên tất cả các bảng của **Dataset** vào nguồn dữ liệu.
  - **Update**(<DataTable>): **Cập nhật** tất cả các thay đổi trên **DataTable** vào nguồn dữ liệu.
  - **Update**(<Dataset>,<Tên bảng>) **Cập nhật** các thay đổi trên bảng trong **Dataset** vào nguồn dữ liệu.

Đây cũng là một trong những đối tượng được sử dụng nhiều trong lập trình **ADO.NET**

5.1.3 Đối tượng Dataset

Là thành phần chính của **kiến trúc không kết nối cơ sở dữ liệu**, được dùng để nắm giữ dữ liệu của cơ sở dữ liệu và **cho phép thay đổi dữ liệu** bên trong đối tượng này để sau đó **cập nhật trở lại cơ sở dữ liệu nguồn** bằng **phương thức Update** của đối tượng **DataAdapter.**



Hình 3. 19 Đối tượng DataSet

DataAdapter là bộ đọc dữ liệu từ nguồn dữ liệu, và **DataTable** là đối tượng lưu trữ dữ liệu không kết nối, nó như một bảng tạm để chứa dữ liệu và nó không bị ảnh hưởng bởi dữ liệu đó từ nguồn nào.

Các **đối tượng không kết nối** cho phép

– Lưu trữ một **bản sao thông tin** lấy từ cơ sở dữ liệu.

– Khi đã **ngắt kết nối** tới cơ sở dữ liệu.

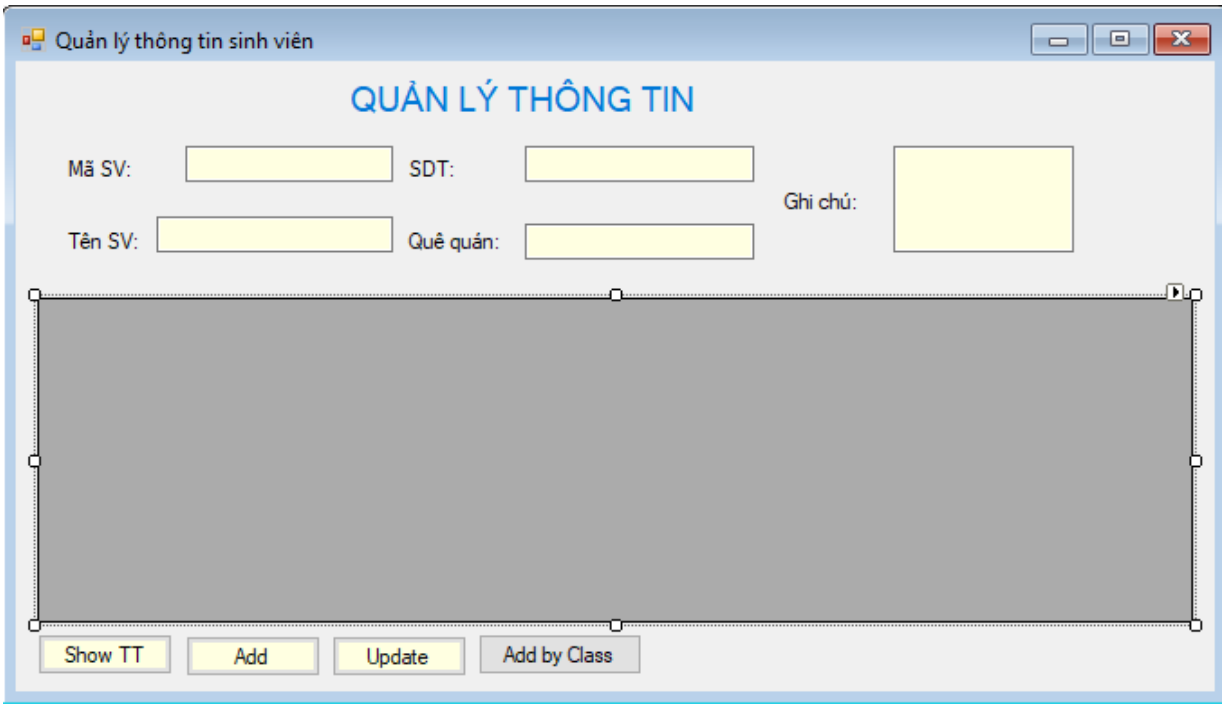
– Đọc các dòng theo thứ tự bất kỳ

– **Tìm kiếm, sắp xếp** hay **trích lọc** các **dòng** một cách linh hoạt.

– Tạo ra các **thay đổi trên dữ liệu**, sau đó **đồng bộ (cập nhật)** các thay đổi này vào **cơ sở dữ liệu**.

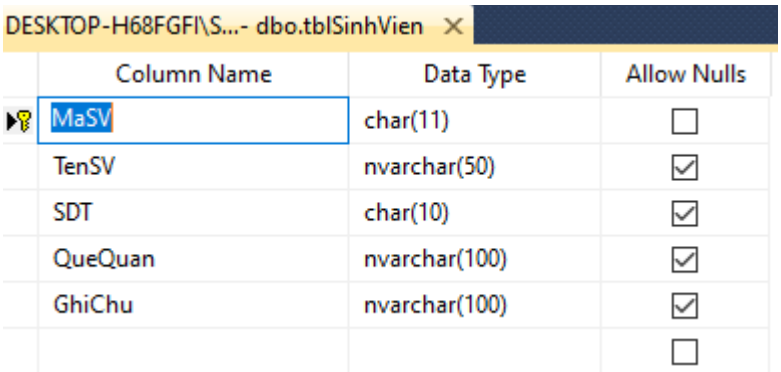
5.1.4 Ví dụ về việc cập nhật dữ liệu vào CSDL

Tạo form cập nhật thông tin sinh viên như ***Hình 5.1***



Hình 5. 1 Form Quản lý thông tin sinh viên.

Tạo CSDL *QLSV* trong ***Sql Server***, tạo bảng tblSinhVien như ***Hình 5.2***:



Hình 5. 2 Thiết kế bảng tblSinhVien.

Cập nhật thông tin sinh viên, sử dụng câu lệnh SQL trên C#.

Các thư viện đã sử dụng:

```
using System;
using System.Data;
using System.Data.SqlClient;
```

```
using System.Windows.Forms;
```

Viết code cho nút "*Show TT*"

```
private void btnShowTT_Click(object sender, EventArgs e)
    {
        Display();

    }
```

Trong đó hàm *Display* được viết như sau:

```
    void Display()
    {
        dt.Clear();
        SqlConnection connection = new SqlConnection(StrConectionString);
        string text = "select * from tblSinhVien";
        SqlCommand command = new SqlCommand(text, connection);
        SqlDataAdapter da = new SqlDataAdapter(command);
        da.Fill(dt);
        dgvTTSV.DataSource = dt;
        DataBinding(dt);

    }
```

Hàm DataBinding được viết như sau:

```
    void DataBinding(DataTable dt)
    {
        txtMaSV.DataBindings.Add("Text", dt, "MaSV",true);
        string t = txtMaSV.Text;
        txtTenSV.DataBindings.Add("Text", dt, "TenSV",true);
        txtSDT.DataBindings.Add("Text", dt, "SDT",true);
        txtQueQuan.DataBindings.Add("Text",dt,"QueQuan",true);
        txtGhiChu.DataBindings.Add("Text", dt, "Ghichu",true);

    }
```

Viết code cho nút "Add"

```
    /// <summary>
    /// Tạo thêm biến chứa chuỗi kết nối
    /// </summary>
    string StrConectionString;
    /// <summary>
    /// Phương thức khởi dựng mặc định với tham số là chuỗi kết nối
    /// </summary>
    /// <param name="_StrConnectionString">Chuỗi kết nối CSDL</param>
    public frmSinhVien(string _StrConnectionString)
    {
        InitializeComponent();
```

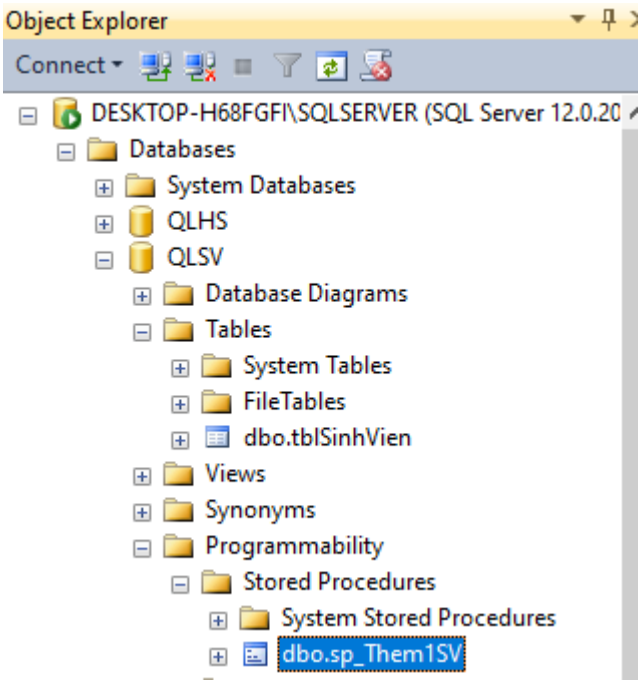
```
        StrConectionString = _StrConnectionString;
    }

        private void btnAdd_Click(object sender, EventArgs e)
        {
            dt.Clear();
            SqlConnection connection = new SqlConnection(StrConectionString);
            string text = "Insert into tblSinhvien(MaSV,TenSV,SDT,QueQuan,Ghichu)"
                + "values(@MaSV,@TenSV,@SDT,@QueQuan,@GhiChu)";
            if (connection.State == ConnectionState.Closed) connection.Open();
            SqlCommand command = new SqlCommand(text, connection);
            command.Parameters.AddWithValue("@MaSV", txtMaSV.Text.ToString().Trim());
            command.Parameters.AddWithValue("@TenSV", txtTenSV.Text.ToString().Trim());
            command.Parameters.AddWithValue("@SDT", txtSDT.Text.ToString().Trim());
            command.Parameters.AddWithValue("@QueQuan",
txtQueQuan.Text.ToString().Trim());
            command.Parameters.AddWithValue("@GhiChu",
txtGhiChu.Text.ToString().Trim());
            command.ExecuteNonQuery();
            command.Dispose();
            connection.Dispose();
            Display();

        }
```

Viết code cho nút "Update"

```
    private void btnUpdate_Click(object sender, EventArgs e)
    {
        //dt.Clear();
        SqlConnection connection = new SqlConnection(StrConectionString);
        string strSqlCommand = "Update tblSinhVien " +
            "set TenSV ='" + txtTenSV.Text + "', SDT='" + txtSDT.Text + "'," +
            "QueQuan='" + txtQueQuan.Text + "',GhiChu='" + txtGhiChu.Text +
            "'where MaSV='" + txtMaSV.Text + "'";
        if (connection.State == ConnectionState.Closed) connection.Open();
        SqlCommand command = new SqlCommand(strSqlCommand, connection);
        command.ExecuteNonQuery();
        command.Dispose();
        connection.Dispose();

    }
```

Cập nhật thông tin sinh viên, sử dụng thủ tục trên Sql Server.

Trên SQL Server tạo thủ tục sp_Them1SV như ***Hình 5.3***



Hình 5. 3 Tạo thủ tục sp_Them1SV

Nội dung thủ tục sp_Them1SV như sau:

```
-- =============================================
-- Author:                  <hoangtn>
-- Create date: <17-12-2020>
-- Description:    <Thêm một sinh viên vào bảng sinh viên>
-- =============================================
CREATE PROCEDURE [dbo].[sp_Them1SV]
 (
  @MaSV char(11),
@TenSV nvarchar(50),
@SDT char(10),
@QueQuan nvarchar(100),
 @GhiChu nvarchar(100)
 )
AS
BEGIN
        -- SET NOCOUNT ON added to prevent extra result sets from
        -- interfering with SELECT statements.
        SET NOCOUNT ON;
    -- Insert statements for procedure here
insert into tblSinhVien(MaSV,TenSV,SDT,QueQuan,GhiChu)
values (@MaSV,@TenSV,@SDT,@QueQuan,@GhiChu)
END
```

Bên C# viết interface "*ISinhVien*" như sau:

```
interface ISinhVien
{
    string MaSV { get; set; }
    string TenSV { get; set; }
    string SDT { get; set; }
    string QueQuan { get; set; }
    string GhiChu { get; set; }
    int Them1SV(string strSqlConnection);
    int Sua1SV(string strSqlConnection);
    int Xoa1SV(string strSqlConnection);
}
```

Sau đó viết lớp "*clsSinhVien*" thực thi interface "*ISinhVien*"

```
class clsSinhVien : ISinhVien
{
    public string MaSV { get; set; }
    public string TenSV { get; set; }
    public string SDT { get; set; }
    public string QueQuan { get; set; }
    public string GhiChu { get; set; }
    public clsSinhVien(string _masv, string _tensv,
        string _sdt, string _quequan,string _ghichu)
    {
        MaSV = _masv; TenSV = _tensv; SDT = _sdt; QueQuan = _quequan;GhiChu = _ghichu;
    }
    public int Them1SV(string strConection)
    {
        SqlConnection connection = new SqlConnection(strConection);
        if (connection.State == ConnectionState.Closed) connection.Open();
        SqlCommand command = new SqlCommand();
        command.Connection = connection;
        command.CommandText = "sp_Them1SV";
        command.CommandType = CommandType.StoredProcedure;
        command.Parameters.AddWithValue("@MaSV",MaSV );
        command.Parameters.AddWithValue("@TenSV", TenSV);
        command.Parameters.AddWithValue("@SDT", SDT);
        command.Parameters.AddWithValue("@QueQuan", QueQuan);
        command.Parameters.AddWithValue("@GhiChu", GhiChu);
        command.ExecuteNonQuery();
        command.Dispose();
        connection.Dispose();
        return 0;
    }
    public int Sua1SV(string strConection) { return 0; }
    public int Xoa1SV(string strConection) { return 0; }
}
```

Viết code cho nút "*Add by Class*" để thêm mới một bản ghi vào bảng "*tblSinhVien*" như sau:

```
private void btnAddbyClass_Click(object sender, EventArgs e)
{
    DAL.clsSinhVien sv = new DAL.clsSinhVien(txtMaSV.Text, txtTenSV.Text
        , txtSDT.Text, txtQueQuan.Text, txtGhiChu.Text);
    sv.Them1SV(StrConectionString);

}
```

Trong ví dụ tiếp theo sẽ sử dụng câu lệnh **Sql** trực tiếp trong C#, tạo cơ sở dữ liệu sau:

Dự án cần thêm mới thông tin cho mỗi sinh viên khi tiến hành nhập học. Giả sử bảng SinhVien trong cơ sở dữ liệu như sau:

DESKTOP-5LG1IQR.QLSV - dbo.SinhVien

Column Name	Data Type	Allow Nulls
MaSV	char(10)	☐
TenSV	nvarchar(50)	☑
Namvao	int	☑
Ngaysinh	date	☑
Quequan	nvarchar(50)	☑
Ghichu	nvarchar(50)	☑

Hình 3. 20 Cấu trúc bảng SinhVien

Tạo một thủ tục trong SQL Server để thêm dữ liệu vào bảng SinhVien như sau:

```
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- =============================================
-- Author:                  hoangtn
-- Create date: <Create Date,,>
-- Description:    Thêm mới một dòng dữ liệu vào bảng SinhVien
-- =============================================
CREATE PROCEDURE sp_Insert1record
        -- Add the parameters for the stored procedure here
```

```
        (
        @MaSV char(10),
        @TenSV nvarchar(50),
        @Namvao int,
        @Ngaysinh date,
        @Quequan nvarchar(50),
        @Ghichu nvarchar(50)
        )
AS
BEGIN
        -- SET NOCOUNT ON added to prevent extra result sets from
        -- interfering with SELECT statements.
        SET NOCOUNT ON;
   -- Insert statements for procedure here
        Insert into SinhVien(MaSV, TenSV,Namvao,Ngaysinh,Quequan,Ghichu)
        Values(@MaSV, @TenSV,@Namvao,@Ngaysinh,@Quequan,@Ghichu)
END
GO
```

Sử dụng đối tượng **SqlCommand** để thêm 1 bộ thông tin sinh viên vào bảng **SinhVien** trong **cơ sở dữ liệu**.

```
        /// <summary>
      /// Thêm mới một thông tin sinh viên vào cơ sở dữ liệu
      /// Giả sử thêm mới sinh viên có
      /// Mã SV:1900585001, Tên SV: Nguyễn Văn A
      /// Năm vào: 2019, ngày sinh: 12/09/2001
      /// Quê quán: Hưng yên, Ghi chú: SV xét tuyển đạt
      /// </summary>
      /// <param name="Conn">Biến Connection đã được kết nối đến CSDL QLSV</param>
```

```
public void Them1banghiSV(SqlConnection Conn)
{
    SqlCommand sqlComm = new SqlCommand();
    sqlComm.Connection = Conn;
    sqlComm.CommandType = System.Data.CommandType.StoredProcedure;
    sqlComm.CommandText = "sp_Insert1record";
    //tham so truyen vao cho comm
    //tham so @MaSV
    SqlParameter sqlParameter = new SqlParameter("@MaSV", System.Data.SqlDbType.Char,
10);
    sqlParameter.Value = "1900585001";
    sqlComm.Parameters.Add(sqlParameter);
    //tham so @TenSV
    sqlParameter = new SqlParameter("@TenSV", System.Data.SqlDbType.NVarChar, 50);
    sqlParameter.Value = "Nguyễn Văn A";
    sqlComm.Parameters.Add(sqlParameter);
    //tham so @Namvao
    sqlParameter = new SqlParameter("@Namvao", System.Data.SqlDbType.Int);
    sqlParameter.Value = 2019;
    sqlComm.Parameters.Add(sqlParameter);
    // tham so @Ngaysinh
    sqlParameter = new SqlParameter("@Ngaysinh", System.Data.SqlDbType.Date);
    sqlParameter.Value = new DateTime(2001, 9, 12);
    sqlComm.Parameters.Add(sqlParameter);
    //Tham so @Quequan
    sqlParameter = new SqlParameter("@Quequan", System.Data.SqlDbType.NVarChar, 50);
    sqlParameter.Value = "Hưng yên";
    sqlComm.Parameters.Add(sqlParameter);
    //Tham so @Ghichu
```

```
        sqlParameter = new SqlParameter("@GhiChu", System.Data.SqlDbType.NVarChar, 50);
        sqlParameter.Value = "SV xét tuyển đạt";
        sqlComm.Parameters.Add(sqlParameter);
        try
        {
            sqlComm.ExecuteNonQuery();
        }
        catch (Exception ex)
        {
            string e = ex.ToString();
        }
        finally
        {
            sqlComm.Dispose();
        }
    }
```

5.2. Sử dụng GridView để cập nhật dữ liệu trực tiếp trên DataSet

Ta cũng có thể sử dụng GridView để cập nhật trực tiếp dữ liệu.

Tạo một bảng như sau: (bảng tbl_students)

```
USE [test]
GO
/****** Object:  Table [dbo].[tbl_students]    Script Date: 11/29/2015 15:45:49 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[tbl_students](
    [id] [nchar](10) NOT NULL,
    [name] [nvarchar](50) NULL,
```

```
    [address] [nvarchar](150) NULL,
    [phone] [nchar](10) NULL,
    [email] [nvarchar](50) NULL,
 CONSTRAINT [PK_tbl_students] PRIMARY KEY CLUSTERED
(
    [id] ASC
)WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS  = ON, ALLOW_PAGE_LOCKS  = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
```

Đầu tiên, chúng ta cần import thư viện sql client vào

```
using System.Data.SqlClient;
```

- Khai báo cáo biến kết nối với Microsoft Sql Server, biến này khai báo Global

```
SqlConnection sqlCon = new SqlConnection("Server=(local); Database=test; Integrated
Security=TRUE");
SqlCommandBuilder sqlCommand = null;
SqlDataAdapter sqlAdapter = null;
DataSet dataset = null;
```

- Tiếp đến, chúng ta viết thêm một **void** để **load database** từ cơ sở dữ liệu vào **datagridview**, ở đây chúng ta select thêm biến delete tạo một link hiển thị vào datagridview.

```
private void LoadData()
 {
        try
        {
            sqlAdapter = new SqlDataAdapter("SELECT *, 'Delete' AS [Delete] FROM tbl_students",
sqlCon);
            sqlCommand = new SqlCommandBuilder(sqlAdapter);
            sqlAdapter.InsertCommand = sqlCommand.GetInsertCommand();
            sqlAdapter.UpdateCommand = sqlCommand.GetUpdateCommand();
            sqlAdapter.DeleteCommand = sqlCommand.GetDeleteCommand();
            dataset = new DataSet();
            sqlAdapter.Fill(dataset, "tbl_students");
```

```
            dataGridView1.DataSource = null;
            dataGridView1.DataSource = dataset.Tables["tbl_students"];
            for (int i = 0; i < dataGridView1.Rows.Count; i++)
            {
                DataGridViewLinkCell linkCell = new DataGridViewLinkCell();
                dataGridView1[<span style="color:#FF0000"><strong>5</strong></span>, i] = linkCell;
            }
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
        }
}
```

Tiếp đến, chúng ta sẽ viết sự kiện formload để load database vào datagridview.

```
private void Form1_Load(object sender, EventArgs e)
 {
        try
        {
            sqlCon.Open();
            LoadData();
        }
         catch (Exception ex) { MessageBox.Show(ex.Message); }
 }
```

Tiếp đến, chúng ta viết sự kiện CellContentClick cho datagridview. khi nhấn vào nút insert, update hay delete

```
private void dataGridView1_CellContentClick(object sender, DataGridViewCellEventArgs e)
 {
       try
        {
            if (e.ColumnIndex == 5)
            {
                string Task = dataGridView1.Rows[e.RowIndex].Cells[5].Value.ToString();
```

```
            if ( Task == "Delete")
            {
                if (MessageBox.Show("Bạn có chắc chắm muốn xóa không?", "Đang xóa...",
MessageBoxButtons.YesNo, MessageBoxIcon.Question) == DialogResult.Yes)
                {
                    int rowIndex = e.RowIndex;
                    dataGridView1.Rows.RemoveAt(rowIndex);
                    dataset.Tables["tbl_students"].Rows[rowIndex].Delete();
                    sqlAdapter.Update(dataset, "tbl_students");
                }
            }
            else if(Task == "Insert")
            {
                int row = dataGridView1.Rows.Count - 2;
                DataRow dr = dataset.Tables["tbl_students"].NewRow();
                dr["id"] = dataGridView1.Rows[row].Cells["id"].Value;
                dr["name"] = dataGridView1.Rows[row].Cells["name"].Value;
                dr["address"] = dataGridView1.Rows[row].Cells["address"].Value;
                dr["phone"] = dataGridView1.Rows[row].Cells["phone"].Value;
                dr["email"] = dataGridView1.Rows[row].Cells["email"].Value;
                dataset.Tables["tbl_students"].Rows.Add(dr);

dataset.Tables["tbl_students"].Rows.RemoveAt(dataset.Tables["tbl_students"].Rows.Count - 1);
                dataGridView1.Rows.RemoveAt(dataGridView1.Rows.Count - 2);
                dataGridView1.Rows[e.RowIndex].Cells[5].Value = "Delete";
                sqlAdapter.Update(dataset, "tbl_students");
            }
            else if (Task == "Update")
            {
                int r = e.RowIndex;
                dataset.Tables["tbl_students"].Rows[r]["id"] = dataGridView1.Rows[r].Cells["id"].Value;
                dataset.Tables["tbl_students"].Rows[r]["name"] =
dataGridView1.Rows[r].Cells["name"].Value;
                dataset.Tables["tbl_students"].Rows[r]["address"] =
dataGridView1.Rows[r].Cells["address"].Value;
```

```
                    dataset.Tables["tbl_students"].Rows[r]["phone"] =
dataGridView1.Rows[r].Cells["phone"].Value;
                    dataset.Tables["tbl_students"].Rows[r]["email"] =
dataGridView1.Rows[r].Cells["email"].Value;
                    sqlAdapter.Update(dataset, "tbl_students");
                    dataGridView1.Rows[e.RowIndex].Cells[5].Value = "Delete";
                }
            }
        }
        catch (Exception ex) {
            MessageBox.Show(ex.Message);
        }
}
```

- Tiếp đến, ta viết sự kiện userAddRow để khi thêm một dòng mới vào cơ sở dữ liệu.

```
private void dataGridView1_UserAddedRow_1(object sender, DataGridViewRowEventArgs e)
{
        try
        {
            int lastRow = dataGridView1.Rows.Count - 2;
            DataGridViewRow nRow = dataGridView1.Rows[lastRow];
            DataGridViewLinkCell linkCell = new DataGridViewLinkCell();
            dataGridView1[5, lastRow] = linkCell;
            nRow.Cells["Delete"].Value = "Insert";
        }
        catch (Exception ex) { MessageBox.Show(ex.Message); }
}
```

- Và cuối cùng chúng ta viết sự kiện khi double click vào datagridview để update một dòng dữ liệu.

```
private void dataGridView1_CellContentDoubleClick(object sender, DataGridViewCellEventArgs e)
{
        try
        {
            int lastRow = e.RowIndex  ;
            DataGridViewRow nRow = dataGridView1.Rows[lastRow];
```

```
            DataGridViewLinkCell linkCell = new DataGridViewLinkCell();
            dataGridView1[5, lastRow] = linkCell;
            nRow.Cells["Delete"].Value = "Update";
        }
        catch (Exception ex) { MessageBox.Show(ex.Message);
}
```

5.3. Cập nhật dữ liệu từ Text Box vào CSDL

```
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Data.SqlClient;
using System.Data;
namespace Luong_BHXH
{
   class AccessData
   {
      public SqlConnection TaoKetNoi()
      {
         return new SqlConnection("Data Source=CNTT-HOA;Initial Catalog=BHXH_LUONG;Integrated
Security=True");
      }
      //Ham tra ve Datatable
      public DataTable TaoBang(string sql)
      {
         SqlConnection con = TaoKetNoi();
         SqlDataAdapter ad = new SqlDataAdapter(sql, con);
         DataTable dt = new DataTable();
         ad.Fill(dt);
         return dt;
      }
      //Ham thuc hien ExecuteNonQuery
      public void ExecuteNonQuery(string sql)
      {
         SqlConnection con = TaoKetNoi();
```

```
            SqlCommand cmd = new SqlCommand(sql, con);
            con.Open();
            cmd.ExecuteNonQuery();
            con.Close();
            cmd.Dispose();
        }
        public SqlDataReader ExecuteReader(string sql)
        {
            SqlConnection conn = TaoKetNoi();
            conn.Open();
            SqlCommand cmd = new SqlCommand(sql, conn);
            SqlDataReader reader = cmd.ExecuteReader();
            return reader;
        }
    }
}
```

Bài tập chương 5

Hãy tạo ra các form cập nhật cho tất cả các bảng trong CSDL NORTWind.

Chương 6: TẠO BÁO CÁO VỚI CRYSTAL REPORT

6.1. Thiết kế Report

Visual Studio đi kèm một công cụ báo cáo rất hữu ích – **Crystal Report** mà có sẵn cho bạo tạo các báo cáo để hiển thị dữ liệu của bạn lấy được từ **CSDL**.Trong bài thực hành này chúng ta sẽ thảo luận về cách lấy dữ liệu được lưu trữ trong các bảng của hệ quản trị cơ sở dữ liệu **SQL Server**, rồi hiển thị trên **Form projec**.

Giả sử bạn muốn tạo một báo cáo để hiển thị dữ liệu từ bảng dữ liệu Products của CSDL NorthWinds. Đây là CSDL mẫu của Microsoft cung cấp, bạn có thể tải về và cài đặt theo link sau***:*** ***https://www.tektutorialshub.com/crystal-reports/download-crystal-reports-for-visual-studio-2019/***.

Crystal Reports Version	Supported IDE	tải bản này cho win 64 bit	Developer Edition Download	Runtime Download
Service Pack 29	Visual Studio 2019, 2017,2015,2013,2012,2010		SP 29	32 Bit 64 Bit
Service Pack 28	Visual Studio 2019, 2017,2015,2013,2012,2010		SP 28	32 Bit 64 Bit
Service Pack 27	Visual Studio 2019, 2017,2015,2013,2012,2010		SP 27	32 Bit 64 Bit
Service Pack 26	Visual Studio 2019, 2017,2015,2013,2012,2010		SP 26	32 Bit 64 Bit
Service Pack 25	Visual Studio 2019, 2017,2015,2013,2012,2010		SP 25	32 Bit 64 Bit

Người sử dụng có quyền để đăng nhập tới hệ quản trị cở sở dữ liệu mà chứa CSDL này, giả sử trên máy (**User Id** = sa, **Password** = admin123). Thông tin đăng nhập sẽ hữu ích cho bạn khi chúng ta kết nối cơ sở dữ liệu dùng mã nguồn C#. Các bước được tiến hành như sau:

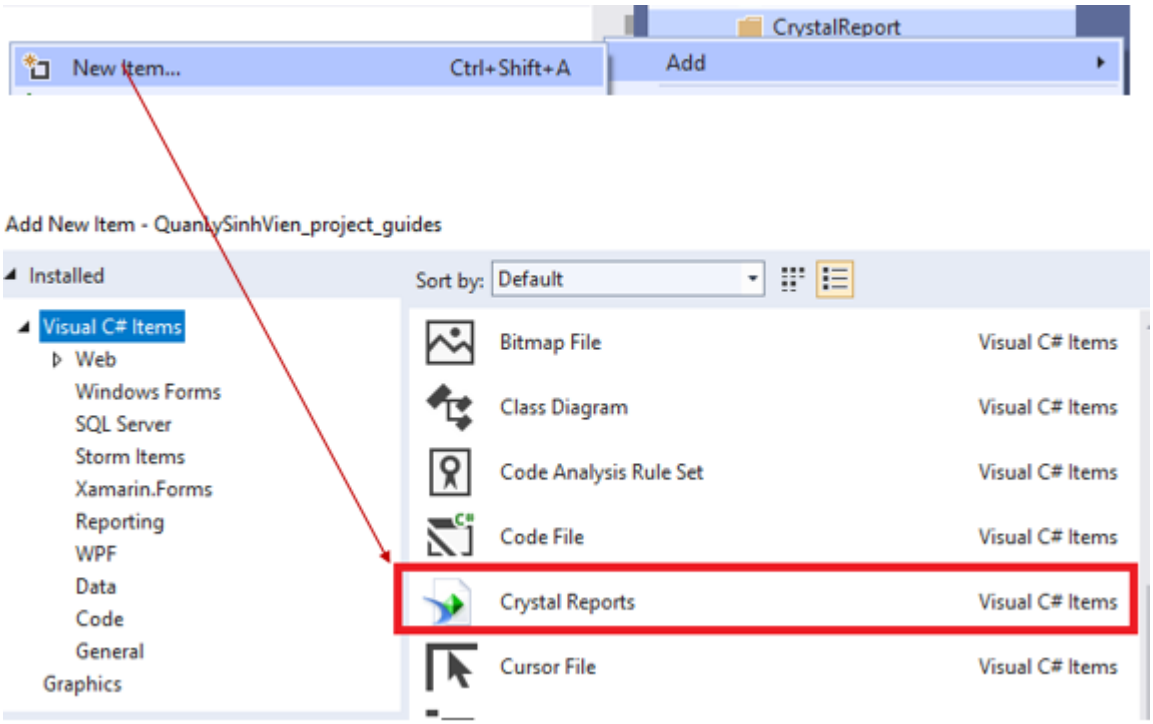
Add một DataSet tới Project

. Kích chuột phải vào tên của ***Project -> Add -> New Item***. Dưới **Categories**, chọn **Data** và dưới **Templates** chọn **DataSet**. Đặt **Name** là **dsProducts**.

Trước khi bạn học về tất cả những cái cơ bản của Crystal Report thì tốt hơn là bây giờ ta sẽ tạo thử một **Crystal** nhỏ một cách nhanh chóng để bạn hình dung về **Crystal Report** ...

Bây giờ bạn hãy mở **Visual Studio** và tạo cho bạn một project mới . Nó có thể tạo trong cả hai ngôn ngữ là **VB.NET** hoặc **C#** và nó có thể là một **Window Application** hoặc ASP.NET. Thiết kế báo cáo với Crystal Report là một kiểu chương trình độc lập .Những bước để làm một Crystal thì giống nhau cho cả hai kiểu chương trình ứng dụng này .

Trong Project được mở bạn chọn **Project > Add New Item** . Điều này sẽ làm hiện ra những dánh sách các kiểu mẫu chương trình có thể thấy.

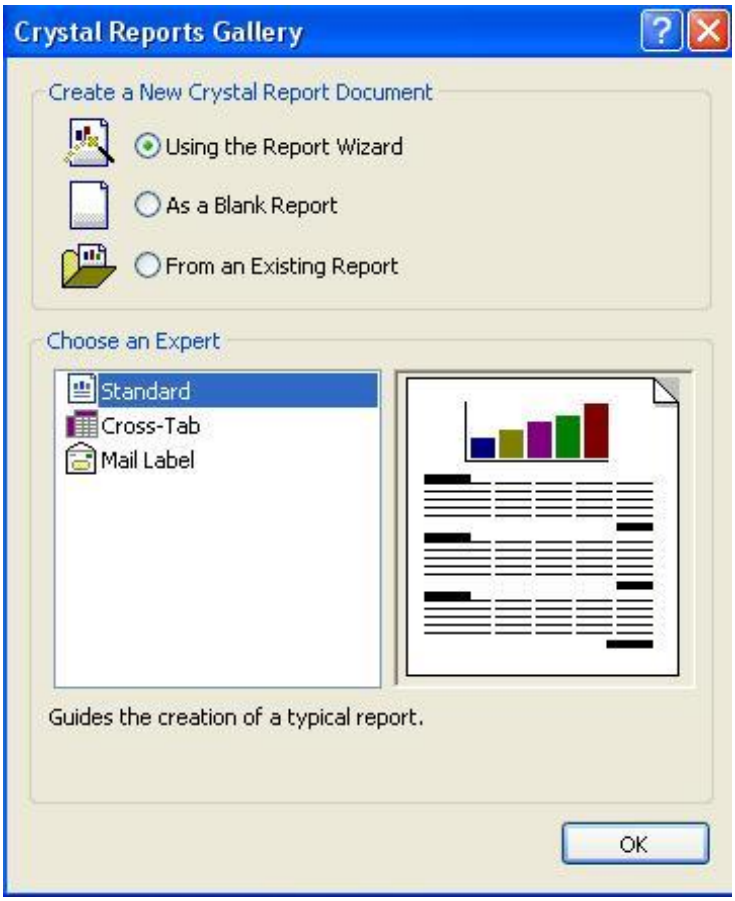


Hình 6. 1 Tạo mới 1 Crystal Report

6.1.1 Thiết kế Crystal Report sử dụng Report Wizard.

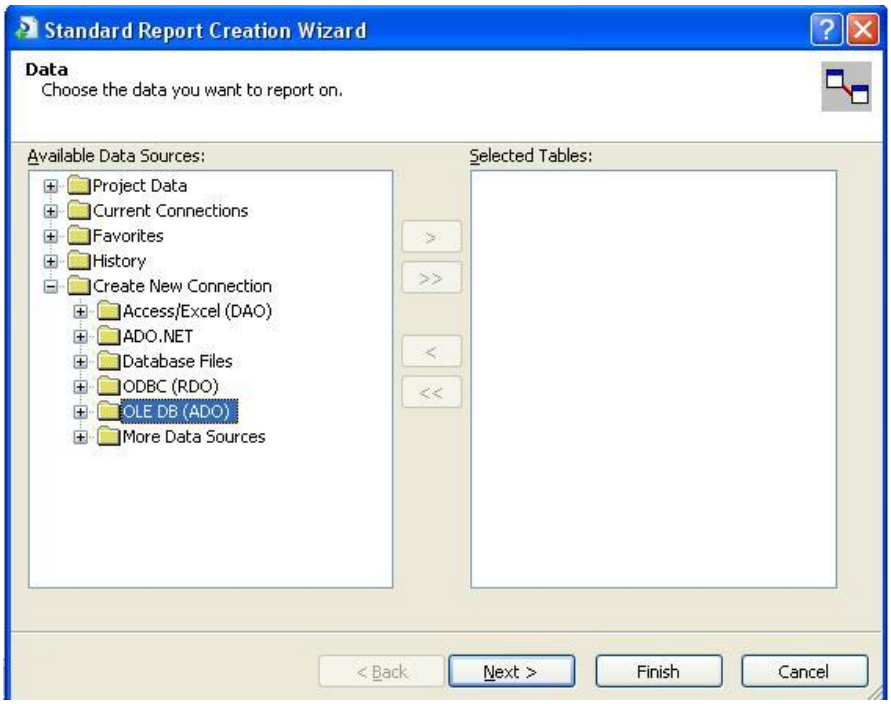
Bạn chọn kiểu trong này là **Crystal Report** . Nhập tên cho nó là Employee List . Hình dưới sẽ cho bạn thấy hộp hội thoại này cho một Window Application của bạn . Giờ bạn hãy click vào create để tạo một report .

Khi đó hộp thoại **Crystal Report Gallery** sẽ xuất hiện ... Bạn hãy chấp nhận hai giá trị mặc định của nó là **Using The Report Wizard** và Standard và nhấn OK (***Hình 6.2***)



Hình 6. 2 Chọn định dạng tài liệu Crystal Report mặc định.

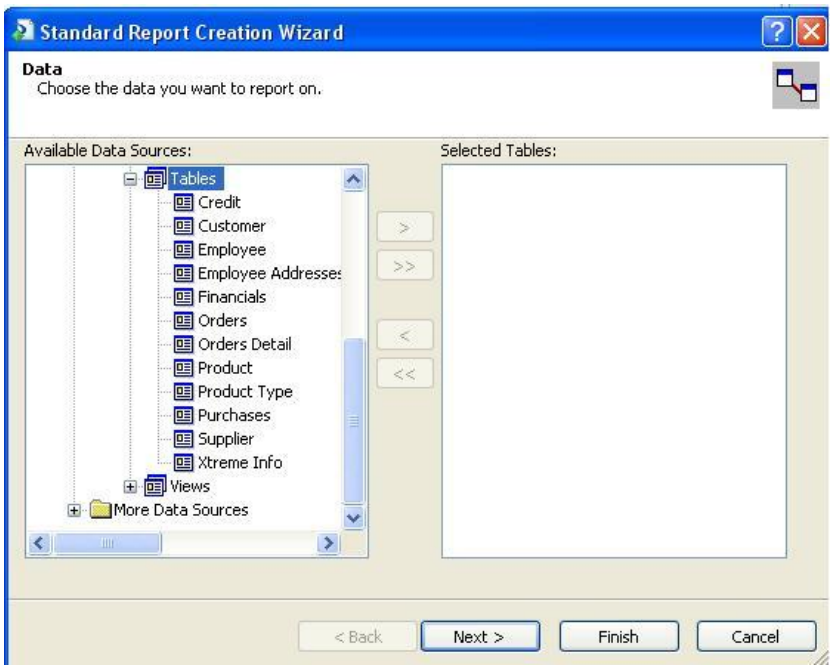
Bây giờ điều tiếp theo là ta phải chọn **cơ sở dữ liệu** bằng việc nhấn chuột vào tuỳ chọn **OLE DB(ADO)** (hình dưới) .



Hình 6. 3 Chọn CSDL nguồn từ OLEDB (ADO)

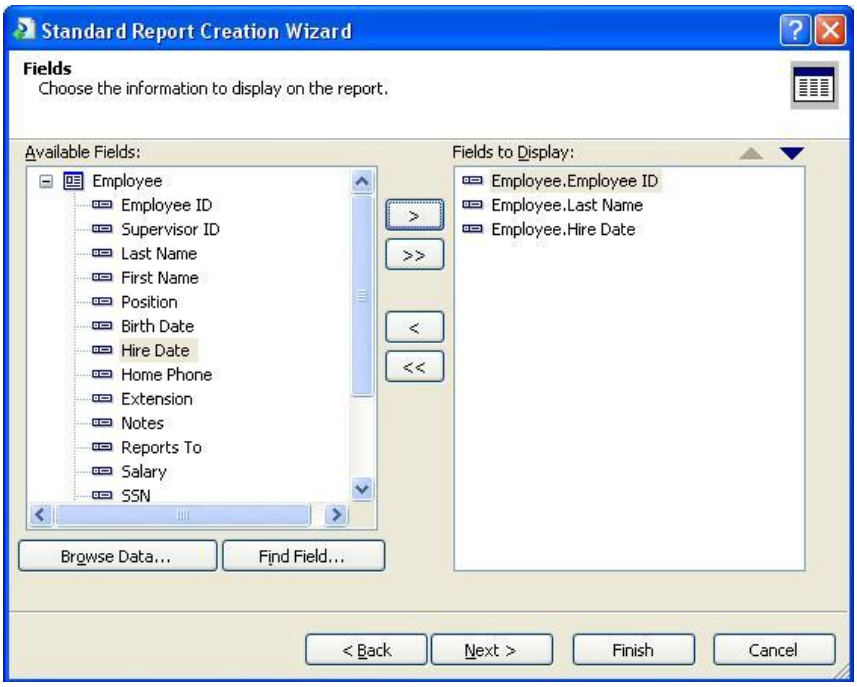
Điều này sẽ làm xuất hiện một hộp thoại mới và bạn chọn trong này là **Microsoft Jet 4.0 OLE DB Provider** . Trong hộp thoại tiếp theo bạn chọn đường dẫn đến căn cứ dữ liệu Xtreme (database name phải là tên đường dẫn đầy đủ trong máy bạn ví dụ D:\Microsoft Visual Studio .NET 2005\Crystal Reports\Samples\en\Databases) . Sau đó bạn Click vào Finish .

Giờ bạn sẽ chở lại **tab Data** . Bạn hãy Click vào mắt **Tables** để bung nó ra và kích đúp vào bảng **Employee** (hoặc bạn có thể kéo và thả nó) để có thể di chuyển nó qua cửa sổ mới tên là **Tables In Report** . Click vào **Next**



Hình 6. 4 Chọn các cột sẽ xuất hiện trong Crystal Report.

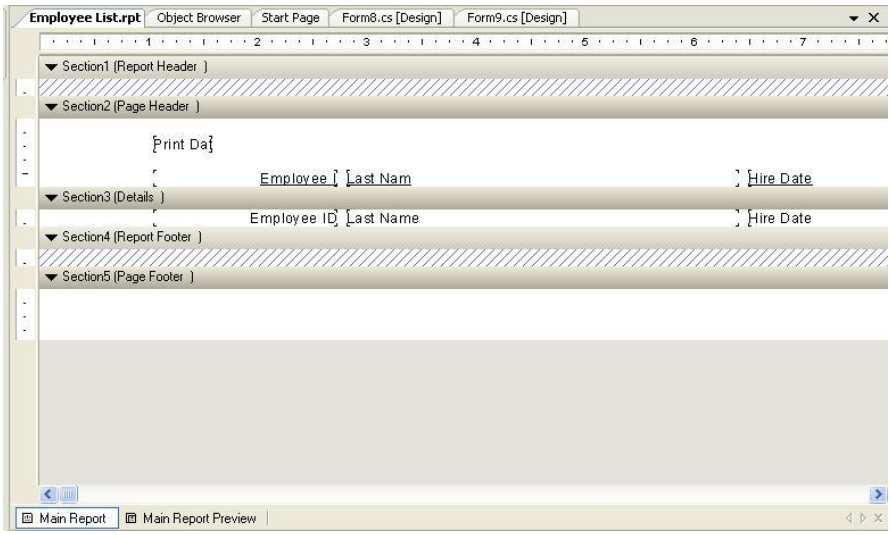
Để hiển thị những vùng cần có trên báo cáo , bạn có thể chọn chúng tại **tab Fields** . Nhấp đúp chuột vào những vùng bạn chọn để chèn chúng vào cửa sổ bên phải : Trong ví dụ này Tôi sẽ chọn các vùng **Employee ID** , **Last Name** , **Hire Data** . Bạn sẽ thấy nó như hình dưới .



Hình 6. 5 Chọn các trường trong bảng Emloyee

Đến đây bạn có thể tiếp tục theo tác với báo cáo của bạn như nhóm lại và chọn dữ liệu để in . Next để đến tab Style rồi nhấn Finish .

Giờ bạn sẽ thấy một báo cáo của bạn sẽ thấy như hình dưới . Nếu báo cáo của bạn sử dụng trong một chương trình ứng dụng thực tế thì bước tiếp theo của bạn là bạn phải sửa đổi Form để có thể xem và in báo cáo này .

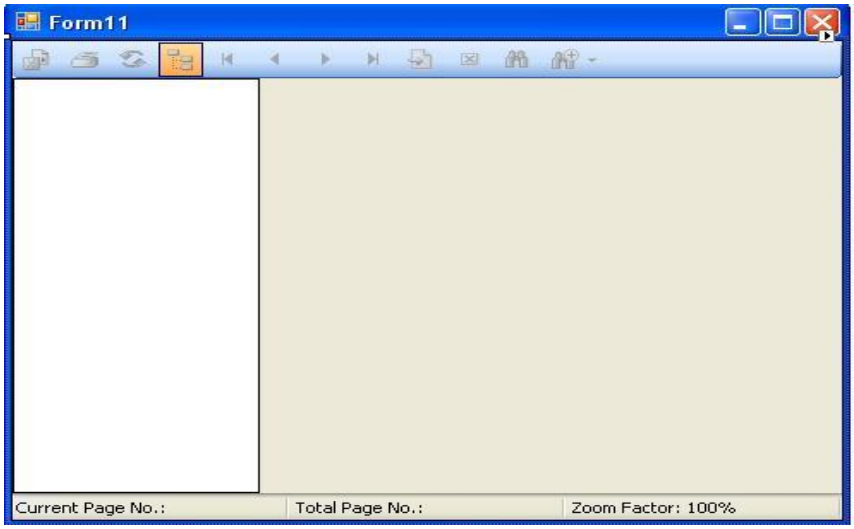


Hình 6. 6 Thiết kế báo cáo nhân viên.

Xem trước một báo cáo với Window Form

Sau khi đã tạo một bản báo cáo trong bài trước, giờ bạn có thể xem nó sử dụng cả **Window Form** hoặc **ASP.NET**. Trong bài này chỉ hướng dẫn để xem nó trong một **Window Form**...

Giờ ta sẽ sửa đổi 1 Form để có thể xem được báo cáo mà chúng ta đã tạo . Khi bạn tạo một project mới thì một Form tên là Form1 luôn được gán đến project của bạn . Giờ bạn hãy mở design của Form này lên và thêm vào đó điều khiển **CrystalReportViewer** .

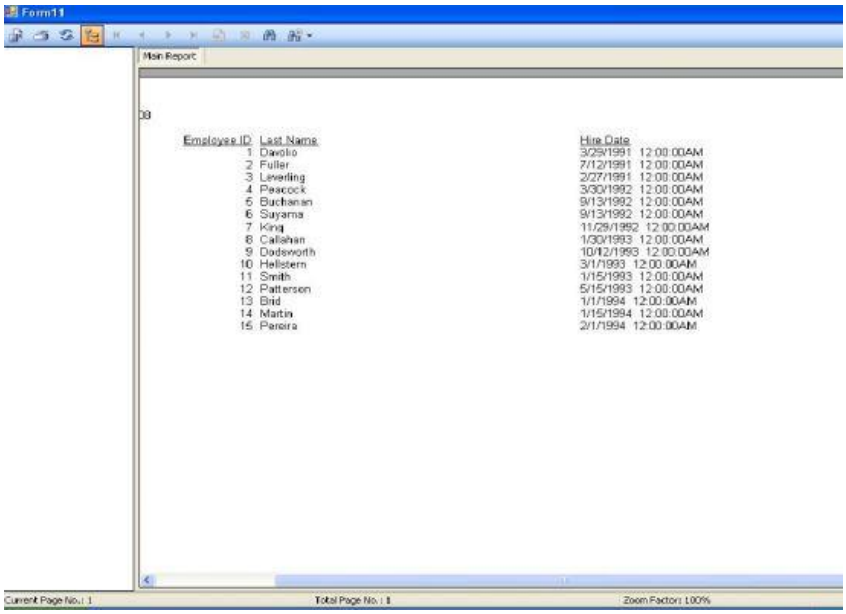


Hình 6. 7 Form đã được thêm đối tượng CrystalReportViewer

Với control này bạn có nhiều cách để xem trước một báo cáo. Trong ví dụ này ta sẽ sử dụng thành phần (component) **ReportDocument** bởi vì nó dễ dàng để lựa chọn . Bạn thêm thành phần component **ReportDocument** đến form bằng cách double - clicking vào nó từ Toolbox . Sau đó bạn click phải chuột vào Form (click vào control **CrystalReportViewer** mà ta đã kéo vào) chọn **Choose a Crystal Report** . Một cửa sổ mới sẽ hiện ra trong hình

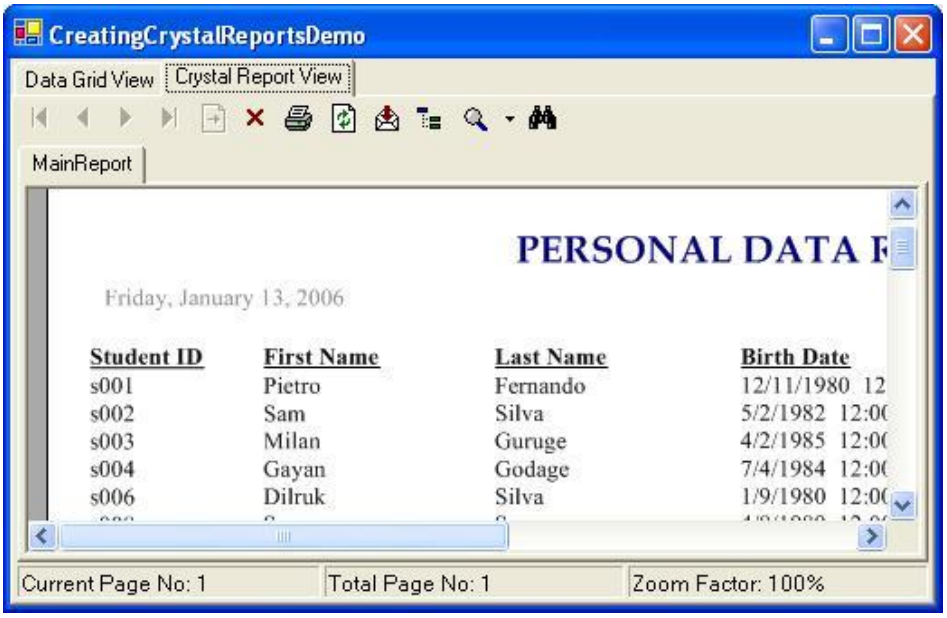
Hình 6. 8 Chọn file Crystal Report nguồn cho CrystalReportViewer

Trong cửa sổ này khi bạn sẽ chọn báo cáo cho Form của bạn .
Xem kết quả của mình bằng cách chạy thử Form vừa tạo ... và đây là kết quả như hình .



Hình 6. 9 Kết quả chạy thử báo cáo.

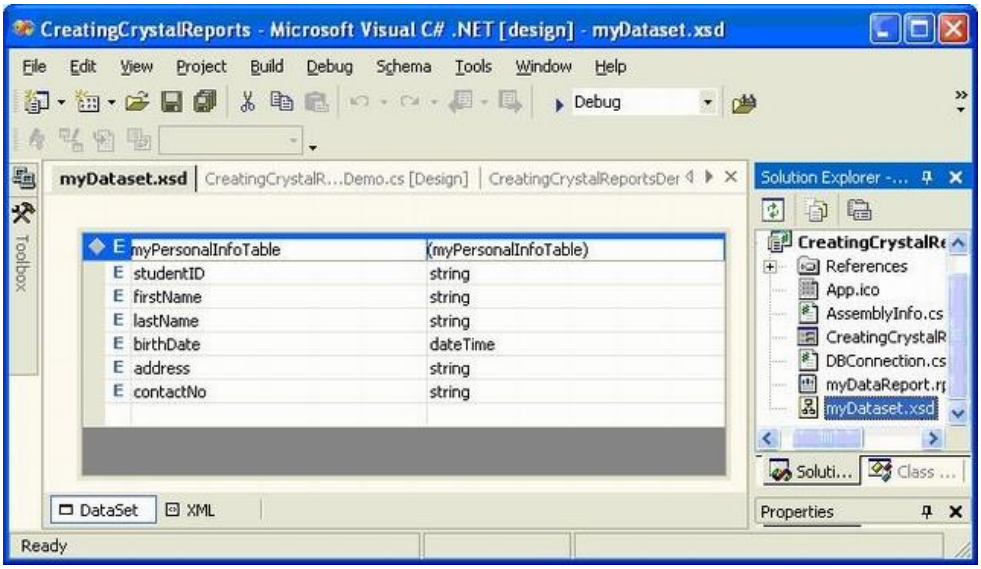
6.1.2 Tạo Crystal Reports bằng Dataset



Hình 6. 10 Hiển thị báo cáo bằng CrystalReport

Tạo một **Crystal Reports** trong C#.Net với **CSDL Access**. SQL server làm tương tự.

Tạo dataset "**Add New Item**" trong "**Project**" menu : **add a Dataset**



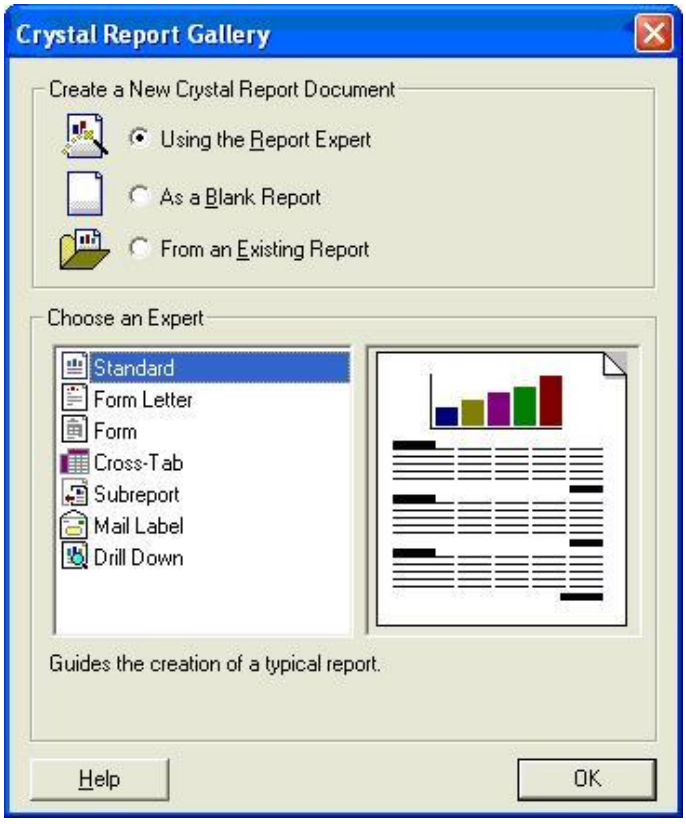
Hình 6. 11 Tạo Dataset và bảng dữ liệu có các thuộc tính nhân viên.

Sau khi tao xong dataset bạn add a Crystal Report vào project bằng cách “Add New Item” trong “Project” menu. Sau đó chọn tên xuất hiện rồi nhấn nút ok.

Tạo kết nối đến CSDL

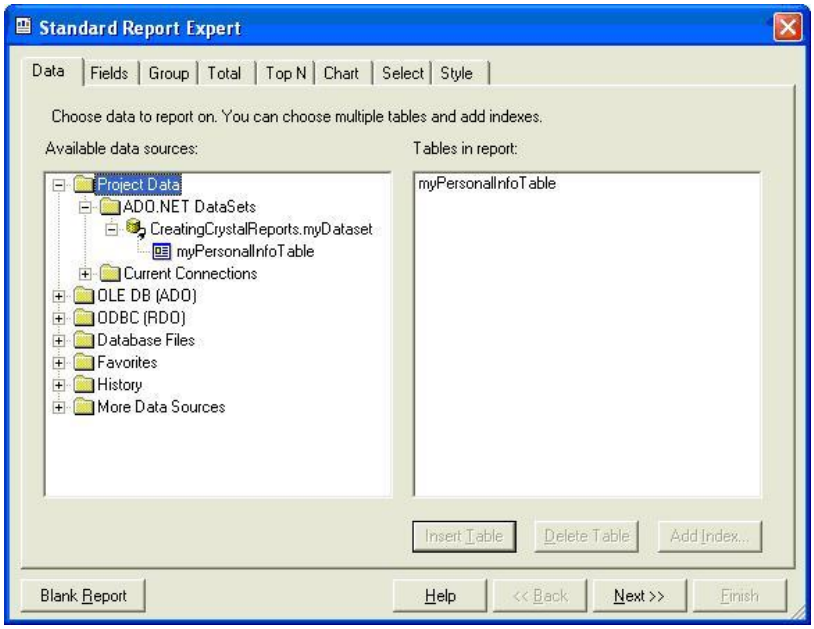
```
String connString = @"Provider=Microsoft.Jet.OLEDB.4.0;_
    Data Source=..\\..\\myDB.mdb;User ID=Admin;Password=";
OleDbConnection conn = new OleDbConnection(connString);
conn.Open();
string query = "SELECT studentID, firstName, lastName, birthDate, _
            address, contactNo FROM studentInfo";
OleDbDataAdapter oleDA = new OleDbDataAdapter(query,conn);
```

Bước 1. Thêm mới 1 **Crystal Report**, lựa chọn ***Using the Report Expert***



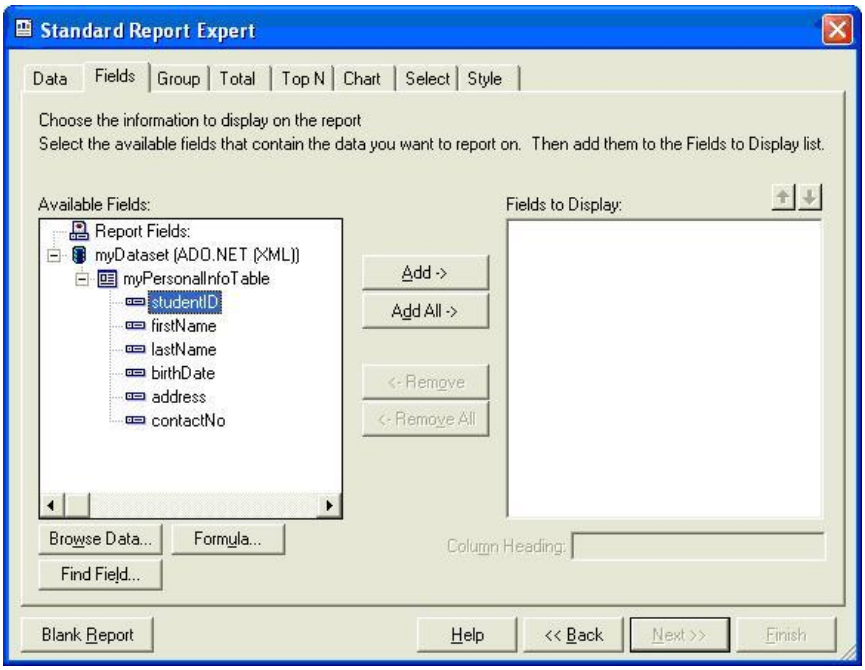
Hình 6. 12 Lựa chọn Using the Reoport Expert

Bước 2. Chọn nguồn dữ liệu cho **CrystalReport** là ***ADO.NET DataSets***



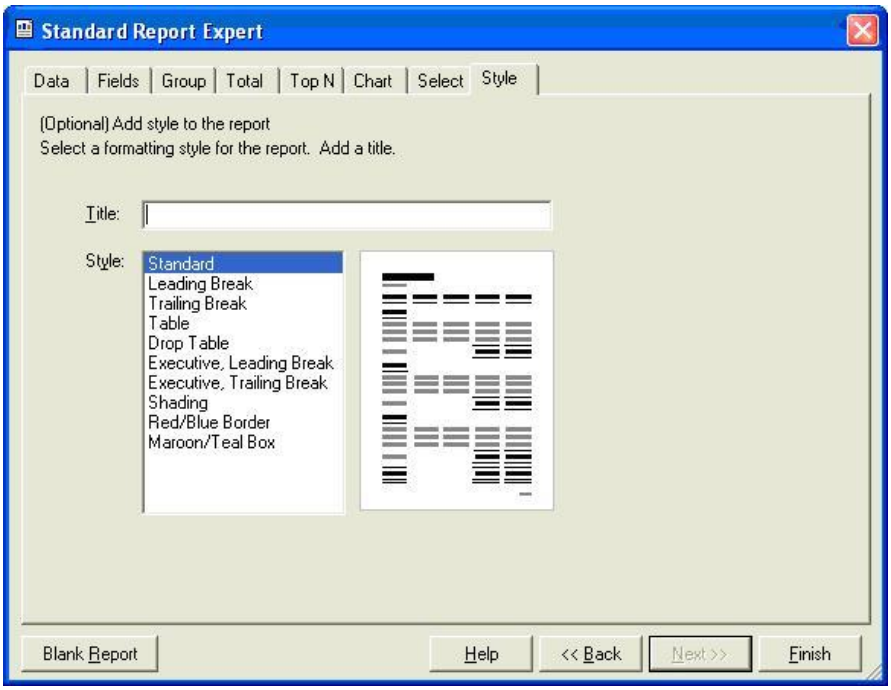
Hình 6. 13 Chọn nguồn dữ liệu cho CrystalReport là ADO.NET DataSets

Bước 3. Chọn các trường cho báo cáo:



Hình 6. 14 Chọn các fields hiển thị trong báo cáo

Bước 4. Chọn style cho report hiển thị



Hình 6. 15 Chọn Style cho Report

Kế tiếp bạn sẽ cho report hiển thị trên form Bạn tao một form và add crystalReportViewer viết code trong thuộc tính

Code:

```
// Tạo biến kết nối, tạo 1 đối tượng DataAdapter với biến kết nối vừa tạo.
DBConnection DBConn = new DBConnection();
OleDbDataAdapter myDataAdapter = DBConn.getDataFromDB();
// Tạo DataSet và đưa dữ liệu vào DatataSet thông tqua DataAdapter.
DataSet dataReport = new DataSet();
myDataAdapter.Fill(dataReport,"myPersonalInfoTable ");
// Tạo 1 báo cáo từ lớp myDataReport
myDataReport myDataReport = new myDataReport();
// gán dữ liệu nguồn cho báo cáo.
myDataReport.SetDataSource(dataReport);
```

```
// thiết lập dữ liệu nguồn cho “crystalReportViewer”
crystalReportViewer1.ReportSource = myDataReport;
```

Có thể thay đổi dường dẫn mặt định bằng:

Code:

```
crystalReportViewer1.ReportSource = @“..\Reports\salesReport.rpt”;
```

6.2. Các thuộc tính và phương thức của CrytalReportViewer thường dùng

Bảng 6. 1 Bảng phương thức của CrystalReportViewer

Phương thức	Ý nghĩa
CloseView	Đóng thẻ xem báo cáo
DrillDownOnGroup	Xem từng nhóm
ExportReport	Xuất báo cáo
GetCurrentPageNumber	Lấy số trang hiện tại
PrintReport	In báo cáo đang được xem
RefreshReport	Làm tươi dữ liệu báo cáo đang đang xem
SearchForText	Tìm kiếm báo cáo nhập vào từ hộp văn bản
ShowFirstPage	Hiển thị trang đầu tiên của báo cáo
ShowGroupTree	Hiển thị báo cáo dạng cây
ShowLastPage	Hiển thị trang cuối cùng của báo cáo
ShowNextPage	Hiển thị trang báo cáo tiếp theo
ShowNthPage	Hiển thị một trang báo cáo cụ thể
ShowPreviousPage	Hiển thị một trang báo cáo trước đó so với trang đang xem
Zoom	Phóng đại/ thu nhỏ báo cáo đang xem

6.3. Thuộc tính của Report

6.4. Field Explorer

Phần lựa chọn xem chọn hiển thị cột nào trên báo cáo

6.5. Tham khảo các hàm và phép toán trong Crytal Report

Các hàm thống kê như tính tổng, tính trung bình, …

Bài tập cuối chương 6.

Hãy tạo ra các báo cáo bất kỳ về thống kê từ CSDL NorthWind

BÀI TẬP LỚN MÔN LẬP TRÌNH WINDOW 2

(Yêu cầu cơ sở dữ liệu phải nhập > 100 bản ghi mỗi bảng, xây dựng phần mềm theo mô hình 3 lớp)

Đề tài 1. Quản lý Khoa Công nghệ thông tin, Trường Đại ABC.

Khoa CNTT trường ĐH Thành Đô cần xây dựng phần mềm quản lý Khoa gồm các quản lý sau:

- Quản lý giảng viên trong khoa:
    - Quản lý thông tin hồ sơ của giảng viên.
    - Quản lý số môn dạy của giảng viên, thuộc bộ môn nào?
- Quản lý sinh viên trong khoa.
    - Biết rằng trong khoa có 4 hệ: cao đẳng nghề, cao đẳng, đại học chính quy, đại học chính quy liên thông.

Sinh viên trong Khoa có một mã sinh viên và có thể đăng ký môn học của mình theo kỳ, Đầu mỗi kỳ, giáo vụ khoa sẽ sắp xếp các môn học của kỳ đó, mỗi giảng viên thuộc bộ môn nào sẽ được phân công môn học của bộ môn mình giả sử Khoa có 3 bộ môn: Khoa học máy tính, công nghệ phần mềm, mạng và truyền thông.

Đề xuất cơ sở dữ liệu như sau:

GiangVien(**MaGV,MaBM**,TenGV,NamSinh,TrinhDo,QueQuan,GhiChu);

BoMon(**MaBM**,TenBM,GhiChu);

MonHoc(**MaMH,MaBM**, SoTC,Ky);

He(**MaHe,TenHe**, thoigiandaotaochuan,thoigiandaotaotoida);

KhungChuongTrinh(**MaHe,MaMH**,GhiChu);

LopHoc(**MaLop,**TenLop,NamVaoTruong);

SinhVien(**MaSV, MaLop, MaHe**, HoTen, NgaySinh, QueQuan ,HoTenBo ,HoTenMe ,SoDienThoaiCaNhan ,SoDienThoaiGiaDinh, NoiOHienTai, GhiChu);

ThoiKhoaBieu(MaTKB, **MaHe, MaLop, MaMH, MaGV**, HocKy, ThoiGianBatDau, ThoiGianKetThu, NgayThuMay, GhiChu).

DangKyHoc(MaSV,MaTKB, KieuDangKy);

Diem(MaSV,MaMH,Diem,GhiChu);

Yêu cầu: Xây dựng phần mềm với CSDL trên và:

- Cho phép cập nhật thông tin giảng viên, sinh viên. Tìm kiếm sinh viên, giảng viên theo tên, mã
- Cho phép giáo vụ khoa sắp xếp các môn học chưa học, và chỉ những môn học chưa học mới được cập nhật.
- Cho phép sinh viên đăng ký những môn học mà mình chưa học hoặc đăng ký học lại những môn học không đạt.
- Sinh viên có thể tra cứu (crystalreport) kết quả học tập theo học kỳ, năm học, khóa học theo từng môn và toàn bộ kết quả.

Mỗi sinh viên chỉ đăng ký được môn học của hệ và lớp mình và không đăng ký được môn học của hệ và lớp khác.(Không xét sinh viên học nhiều hệ cùng lúc)

----------Hết----------

ĐỀ TÀI 2. QUẢN LÝ BÁN HÀNG.

Quản lý bán hang của một công ty.

Một công ty có nhu cầu quản lý các hoạt động kinh doanh của mình bằng phần mềm:

Đề xuất cơ sở dữ liệu sau:

ChungLoai(**MaLoai**,TenChungLoai,GhiChu);

HangHoa (**MaLoai, MaHang**, MaNCC, TenHang, thoigianSX, thoigianBH, GhiChu);

NCC (**MaNCC**, TenNCC, DiaChi, SoDienThoai,Email, GhiChu);

KhachHang(**MaKH**, TenKH,DiaChi,SoDienThoai,KieuKH, Email,GhiChu);

NhanVien (**MaNV**,TenNV,SoDienThoai,Email, QueQuan, DiaChiHienTai, GhiChu) ;

Luong(**MaNV**,Thoigian,Luong) ;

HoaDonNhap(**SoHDNhap**,NgayHD,MaNV,MaNCC);

HoaDonNhapCT(**SoHDNHap**,MaHang,SoLuong,DonGia,GhiChu) ;

HoaDonBan(**SoHDXuat**,NgayHD,MaNV,MaKH);

HoaDonBanCT(**SoHDXuat**, MaHang, SoLuong,DonGia,GhiChu);

Yêu cầu:

- Cập nhật thông tin cho các bảng.
- Báo cáo hàng hóa bán/ mua hàng tháng.
- Báo cáo hàng hóa bán chạy nhất trong tháng/quý/khoảng thời gian.
- Báo cáo hàng hóa bán được ít nhất trong tháng/quý/khoảng thời gian.
- Tính doanh thu/ lợi nhuận hàng tháng cho doanh nghiệp
- Báo cáo khách hàng đến mua nhiều nhất. Địa chỉ có số lượng khách mua hàng nhiều nhất.

---------- Hết ----------

ĐỀ TÀI 3 QUẢN LÝ LỖI TIẾN TRÌNH CỦA MỘT CÔNG TY SẢN XUẤT

Hệ thống xây dựng cần quản lý được thời gian sửa, thời gian chờ, và thời gian hủy trong nhà máy của các máy tính của công nhân. Các thông tin cần quản lý sẽ gồm các thời gian trên và mã số công nhân gọi người đến sửa và mã số công nhân người đến sửa.

Chương trình viết theo mô hình 3 lớp, gồm các lớp: Giao diện, Xử lý và CSDL. Giao diện gồm các form, giao diện và báo cáo (có yêu cầu sau). Xử lý gồm các lớp tính toán, nghiệp vụ được xây dựng gồm :interface và class. Lớp CSDL gồm các thủ tục trên SQL (cập nhật, sửa, xóa) và các lớp thao tác với cơ sở dữ liệu (thao tác chung như: kết nối cơ sở dữ liệu, đổ dữ liệu vào datatable, dataset, các tiện ích cơ sở dữ liệu…).

CƠ SỞ DỮ LIỆU

1. Công nhân

STT	Tên trường	Mô tả	Kiểu	Ghi chú

1	**MaCN**	Mã CN	Tối thiểu 5, đa 11	
2	HoTen	Họ tên		
3	Nhom	Nhóm công nhân		
4	MatKhau	Mật khẩu		

2. Ca làm việc

STT	Tên trường	Mô tả	Kiểu	Ghi chú
1	**MaCa**	Mã ca		
2	**TenCa**	Tên ca		
3				

3. Thongtin_Sua.

STT	Tên trường	Mô tả	Kiểu	Ghi chú
1	Mã CN gọi	Mã CN		
2	Mã CN sửa	Mã CN		
3	MaLoi	Mã lỗi		
4	MaLoiCT	Lỗi chi tiết		
5	NgaySua	Ngày sửa		
6	ThoiGian			

4. Lỗi chính

STT	Tên trường	Mô tả	Kiểu	Ghi chú
1	MaLoi	Mã lỗi		
2	TenLoi	Tên lỗi		

5. Lỗi chi tiết

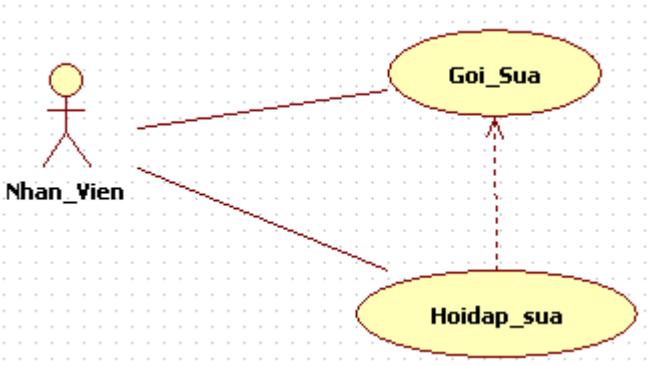
STT	Tên trường	Mô tả	Kiểu	Ghi chú
1	MaLoi	Mã lỗi		
2	MaLoiCT	Lỗi chi tiết		
3	TenLoiCT	Tên lỗi chi tiết		

6. Máy

STT	Tên trường	Mô tả	Kiểu	Ghi chú
1	IP	Địa chỉ máy		Lấy giờ theo server
2	TenMay	Tên máy		
3	MaCa			

7. Quản lý thời gian

STT	Tên trường	Mô tả	Kiểu	Ghi chú
1	SBD	Số báo danh		
2	IP	Địa chỉ máy		
3	**Thoigiancho**	Thời gian chờ		Thời gian cn bắt đầu gọi cho đến lúc người sửa đến
4	ThoiGianKetThuc			Thời điểm kết thúc sửa.
5	ThoiGianchoHuy			Thời điểm hủy (chờ và hủy)
6	**ThoiGianSua**			



ĐỀ TÀI 4. QUẢN LÝ DỰ ÁN

1. ***<u>Chương trình quản lý dự án trên WINDOWS có các chức năng sau:</u>***

- Quản lý số người tham gia dự án.

- Quản lý được số lượng tiến trình, khối lượng công việc của mỗi tiến trình, đơn vị khối lượng công việc.
- Quản lý được kế hoạch công việc.
- Báo cáo được kết quả hiện thời của dự án, báo cáo dưới dạng bảng, biểu đồ.

2. Dự án có các bảng sau:

Nhân sự:

Tên	Kiểu	Mô tả	Ghi chú
MaNS	Char(10)	Mã nhân sự	
TenNS	Nvarchar(50)	Tên nhân sự	
Ngaysinh	Date	Ngày sinh	
Email	Char(30)	Thư điện tử	
Quequan	Char(15)	Số điện thoại	
Ghichu	Nvarchar(50)	Ghi chú	

Tiến trình

Tên	Kiểu	Mô tả	Ghi chú
MaTT	Char(10)	Mã tiến trình công việc	
TenTT	Nvarchar(50)	Tên tiến trình công việc	
Donvi	Nchar(20)	Đơn vị công việc	
Ghichu	Nvarchar(50)	Ghichú	

Kế hoạch

Tên	Kiểu	Mô tả	Ghi chú
Makehoach	Char(10)	Mã kế hoạch	
TenKehoach	Nvarchar(30)	Tên kế hoạch	
Ghichu	Nvarchar(50)	Ghi chú	

Kế hoạch chi tiết

Tên	Kiểu	Mô tả	Ghi chú
Makehoach	Char(10)	Mã kế hoạch	
MaKHCT	Char(10)	Mã kế hoạch chi tiết	
Batdau	Date	Thời gian bắt đầu	
Ketthuc	Date	Thời gian kết thúc	
Hoanthanh	Nvarchar(50)	Hoàn thành	
Ghichu	Nvarchar(50)	Ghi chú	

Thực hiện

Tên	Kiểu	Mô tả	Ghi chú
Makehoach	Char(10)	Mã kế hoạch	
MaTT	Char (10)	Mã thực thi	
Muctieu	Text	Mục tiêu	
Thoigianhientai	datetime	Thời gian hiện tại	

Khoiluongdalam	int	Khối lượng đã làm	
KhoiluongDutoan	int	Khối lượng dự toán	

3. Các thủ tục SQL

- Thêm mới các bảng.
- Tìm kiếm theo mã, theo tên.
- Sửa thông tin.

4. Các lớp ADO.

- INhanSu, ITientrinh,IKehoach,IKehoachchitiet là các interface
- BNhansu,BTientrinh,BKehoach,BKehoachchitiet là các lớp
- Ireport, Breport

5. Giao diện

Thiết kế các giao diện nhập, báo cáo.

ĐỀ TÀI 5. QUẢN LÝ GIẢNG DẠY.

Giảng viên Khoa công nghệ thông tin trường Đại học Thành Đô cần quản lý việc giảng dạy của mình một cách chi tiết và yêu cầu như sau.

- Xây dựng các form để cập nhật dữ liệu.
- Đầu mỗi kỳ báo cáo về chi tiết kế hoạch lịch giảng dạy của mỗi giảng viên, các thông tin như sau.

TRƯỜNG ĐẠI HỌC THANHĐO
KHOA:

LỊCH GIẢNG DẠY

Lớp:.....K.....
Năm học:.....
Môn học:.....
Giáo viên:.....
Học kỳ:.....
Số giờ của môn học:.....
Số giờ đã giảng dạy ở học kỳ trước.....
Số giờ đã giảng trong học kỳ:.....
Số giờ còn lại:.....

Thứ tự bài giảng	Tên bài giảng	Số giờ	Thời gian thực hiện			Thiết bị, phương tiện và đồ dùng dạy học	Ghi chú
			Lớp:.....	Lớp:.....	Lớp:.....		

Hà Nội, ngày.....tháng .....năm.....

PHO HIỆU TRƯỞNG **TRƯỞNG BỘ MON/ TRƯỞNG BAN** **GIAO VIEN**

- Ngoài ra còn có chi tiết bài giảng.
- Thời khóa biểu cho mỗi giảng viên từ ngày đến ngày

Đề xuất các bảng sau:

Môn học

Tên	Kiểu	Mô tả	Ghi chú

MaMH	Char(10)	Mã môn học	
TenMH	Nvarchar(30)	Tên môn học	
Sotietlythuyet	Int	Số tiết lý thuyết	
Sotietthuchanh	Int	số tiết thực hành	
Ghichu	Nvarchar(50)	Ghi chú	

Môn học chi tiết:

Tên	Kiểu	Mô tả	Ghi chú
MaMH	Char(10)	Mã môn học	
TenBG	nvarchar(30)	Tên bài giảng	
SotietBG	Int	Số tiết của bài giảng	
PTDH	Nvarchar(30)	Phương tiện dạy học	
GhiChu	Nvarchar(50)	Ghi chú	

Giảng viên

Tên	Kiểu	Mô tả	Ghi chú
MaGV	Char(10)	Mã giảng viên	
TenGV	Nvarchar(30)	Tên giảng viên	
SĐT	Char(20)	Số điện thoại	

Email	Char(50)	Thư điện tử	
Diachi	Nvarchar(50)	Địa chỉ	
GhiChu	Nvarchar(50)	Ghi chú	

Lớp học

Tên	Kiểu	Mô tả	Ghi chú
MaLop	Char(10)	Mã lớp học	
TenLop	Nvarchar(30)	Tên lớp học	
GhiChu	Nvarchar(50)	Ghi chú	

Lịchgiang

Tên	Kiểu	Mô tả	Ghi chú
MaGV	Char(10)	Mã giảng viên	
MaLop	Char(10)	Mã lớp	
MaMH	Char(10)	Mã môn học	
Thu	Nvarchar(10)	Thứ trong tuần	
NgayBatDau	Datetime	Ngày bắt đầu	
NgayKetThuc	Datetime	Ngày kết thúc.	

- …

Đề tài 6. Quản lý đề tài thực tập tốt nghiệp và đề tài tốt nghiệp của Khoa Công nghệ thông tin.

Khoa Công nghệ thông tin trường Đại học Thành Đô cần phải quản lý đề tài tốt nghiệp của sinh viên trong khoa.

Yêu cầu:

- Vào mỗi cuối năm, sinh viên phải đăng ký đề tài thực tập tốt nghiệp, với những sinh viên có điểm trung bình > 7 theo hệ 10 và > 2.5 theo hệ tín chỉ thì có thể đăng ký đề tài tốt nghiệp.
- Báo cáo các đề tài tốt nghiệp do giáo viên đề xuất trong năm học.
- Xem được nội dung từng chương và toàn văn đề tài tốt nghiệp/ thực tập tốt nghiệp của sinh viên.
- Báo cáo số đề tài trùng với mấy năm (chọn số năm) gần đây.

Đề xuất cơ sở dữ liệu như sau:

Giáo Viên

Tên	Kiểu	Mô tả	Ghi chú
MaGV	Char(10)	Mã giáo viên	
TenGV	Nvarchar(30)	Tên giáo viên	
Dienthoai	Char(20)	Điện thoại	
Quequan	Nvarchar(50)	Quê quán	
Ghichu	Nvarchar(50)	Ghi chú	

Sinh viên

Tên	Kiểu	Mô tả	Ghi chú
MaSV	Char(10)	Mã sinh viên	
TenSV	Nvarchar(30)	Tên sinh viên	
Dienthoai	Char(20)	Điện thoại	
Quequan	Nvarchar(50)	Quê quán	
Ghichu	Nvarchar(50)	Ghi chú	

Đề tài

Tên	Kiểu	Mô tả	Ghi chú
MaDT	Char(10)	Mã đề tài	
TenDT	Nvarchar(30)	Tên đề tài	
MaGV	Char(10)	Mã giảng viên	
HuongDT	Nvarchar(50)	Định hướng đề tài	
Ghichu	Nvarchar(50)	Ghi chú	

Chi tiết đề tài

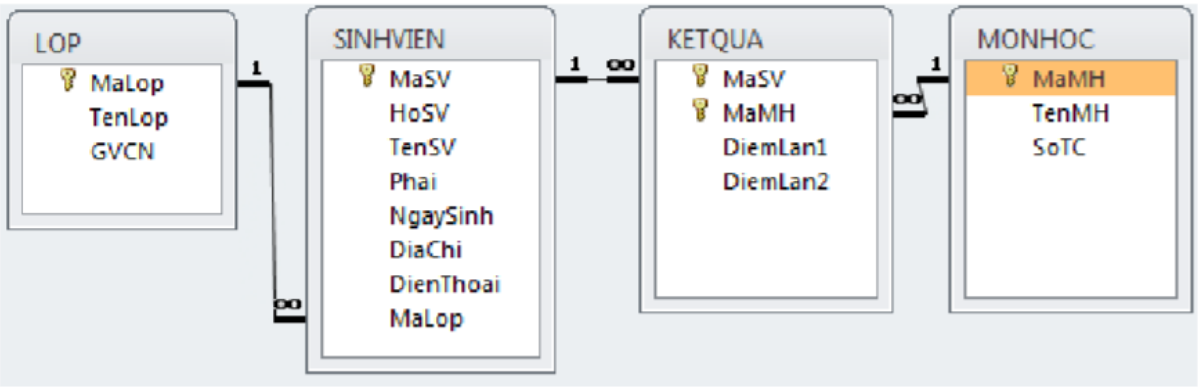
Tên	**Kiểu**	**Mô tả**	**Ghi chú**
MaDT	Char(10)	Mã đề tài	
Mact	Char(10)	Ma chi tiet	

chuong	Nvarchar(30)	Ten chuong	
Muc	Nvarchar(30)	Muc	
Noidung	Text	Noi dung muc	
Ghichu	Nvarchar(30)	Noi dung	

Dangky

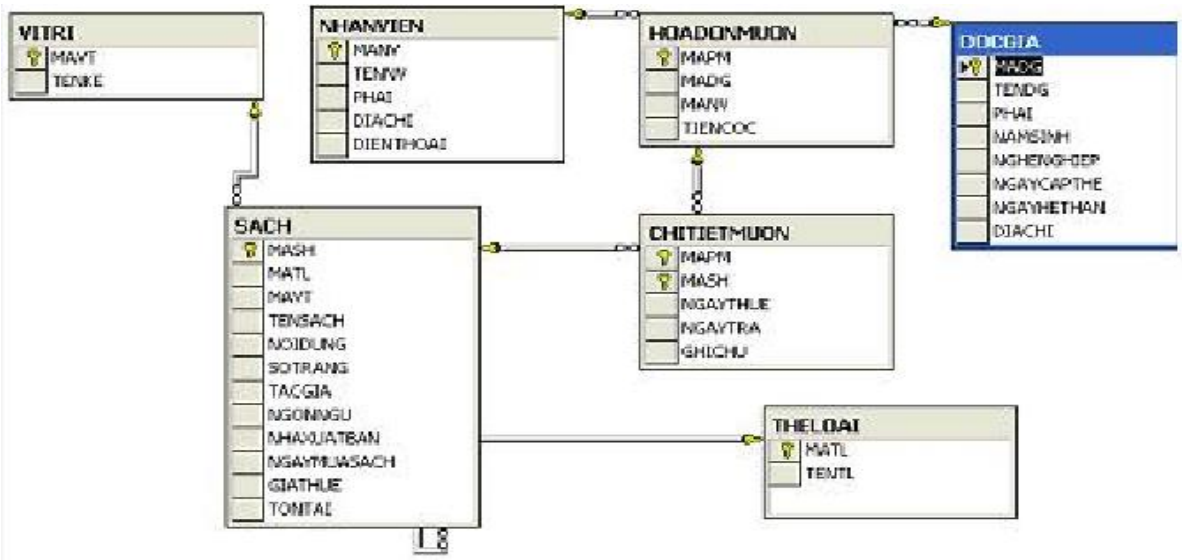
Tên	**Kiểu**	**Mô tả**	**Ghi chú**
MaGV	Char(10)	Mã giảng viên	
Masv	Char(10)	Mã sinh viên	
maDT	Char(10)	Mã đề tài	
Ghichu	Nvarchar(50)	Ghi chú	

Đề tài 7. Quản lý Sinh Viên,



Hãy thực hiện việc cập nhật cho sinh viên và báo cáo tổng kết của sinh viên theo năm, theo kỳ, theo khóa học….

Đề tài 8.1. Quản lý thư viện – Quản lý sách



- Cập nhật từng bảng.
- Báo cáo số lượng sách
- Báo cáo sách theo loại
- Tìm 1 quyển sách theo mã, cho biết vị trí, tên tác giả
- Đưa ra danh sách từng thể loại sách

Đề tài 8.2: Quản lý mượn trả- quản lý thư viện

- Cập nhật từng bảng
- Báo cáo số người mượn chưa trả/ đã trả.
- Báo cáo số nhân viên cho mượn sách từ ngày đến ngày
- Tìm độc giả mượn số lượng nhiều nhất.
- Tìm độc giả chưa từng mượn sách.

Đề tài 10.Hãy xây dựng chương trình quản lý học tập cho môn học

Trong chương trình cần quản lý được tiến trình học tập, kế hoạch học tập, bài tập của sinh viên cho môn lập trình windows nâng cao, chương trình quản lý tất cả các nhóm bài tập lớn gồm tên đề tài, thành viên, công việc của nhóm, công việc của từng người tham gia vào nhóm.

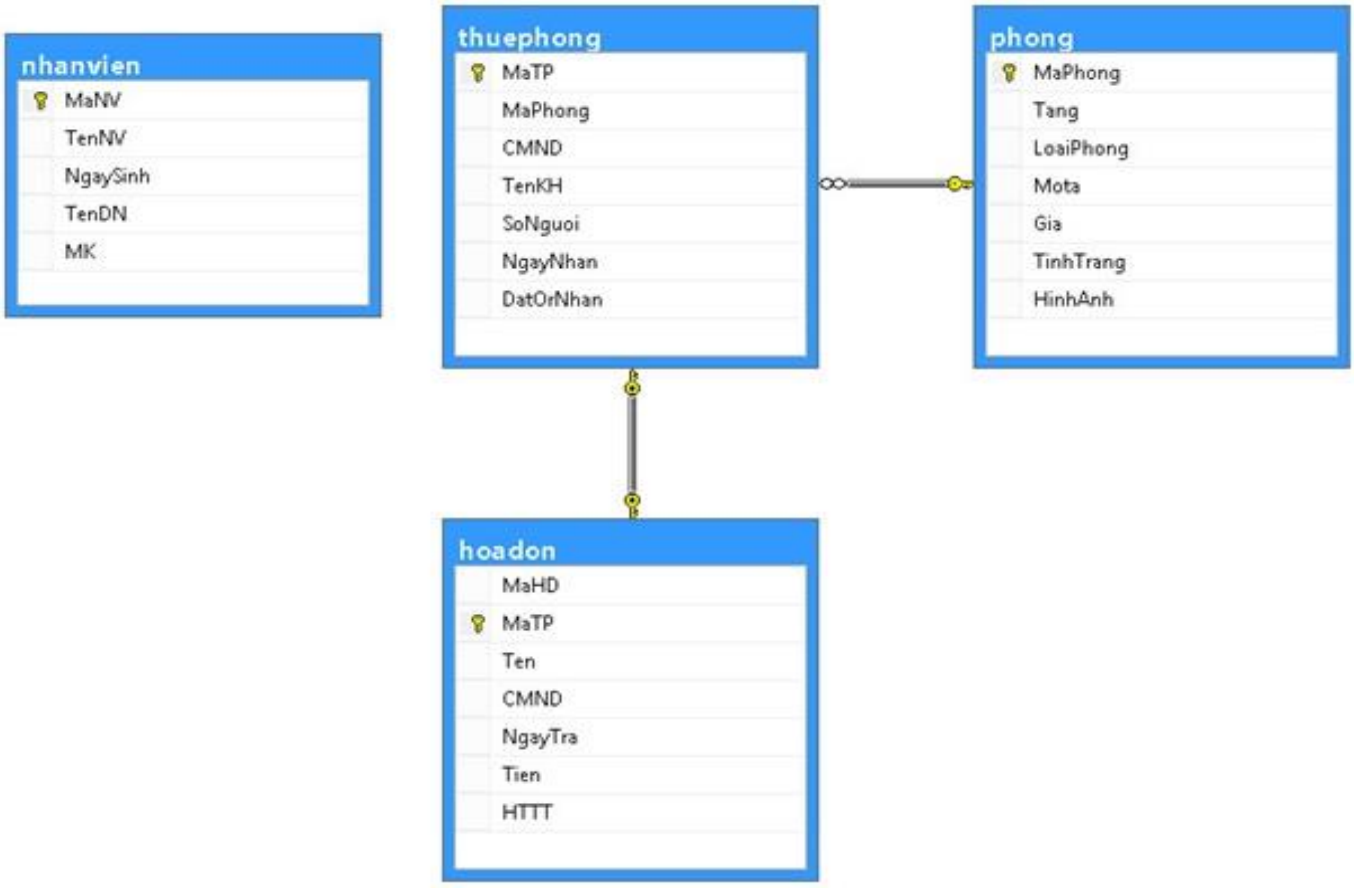
Đề tài 11: Quản lý kinh doanh

Bạn em muốn phát triển mô hình kinh doanh bán hàng trà hoa sữa, em hãy giúp bạn em phát triển một phần mềm để quản lý hoạt động kinh doanh của bạn mình, bao gồm quản lý hàng, quản lý nhân viên, khách hàng thường xuyên, khuyến mại …

Đề tài 12: Quản lý khách sạn:

Nhóm hãy phát triển phần mềm quản lý khách sạn, cho phép khách hàng đặt phòng giữ chỗ, hủy phòng, đặt bàn ăn…

- Cập nhật dữ liệu cho các bảng.
- Tìm phòng trống từ ngày đến ngày.
- Phòng được đặt nhiều nhất.
- Khách thường xuyên thuê.
- …



Đề tài 13: Quản lý hoạt động đoàn.

Nhóm hãy phát triển phần mềm để quản lý hoạt động đoàn trong trường Đại học Thành Đô. Cho phép quản lý hồ sơ đoàn, cán bộ đoàn, tạo sự kiện tình nguyện, các thông báo của đoàn được gửi email đến các thành viên. Báo cáo kết quả của các hoạt động ….

Đề tài 14. Quản lý đặt vé máy bay

Một hãng hàng không có nhu cầu quản lý đặt vé máy bay online. Nhóm hãy giúp họ phát triển phần mềm cho phép đặt vé máy bay online, chế độ ưu đãi khi đặt trước, đặt sau, hủy … Biết rằng mỗi máy bay sẽ bay một tuyến, có giờ đi, giờ đến… và số chỗ của mỗi máy bay là cố định, có các hạng…

Đề tài 15. Quản lý bến xe.

Bến xe Gia Lâm cần xây dựng phần mềm quản lý hoạt động của mình. Nhóm hãy phát triển phần mềm, cho phép quản lý các hoạt động trên, như mua vé online, xe về các tỉnh, tuyến,…

Đề tài 16. Tuyển dụng nhân sự

Trong một công ty săn đầu người (người tài cho một lĩnh vực) có nhu cầu xây dựng phần mềm quản lý nhân sự. Phần mềm cho phép các ứng viên nhập thông tin cá nhân, lĩnh vực hoạt động, đơn vị công tác, và các vị trí nhân sự của các công ty tuyển dụng… Phần mềm cho phép đưa ra báo cáo một ứng viên có thể làm việc ở một danh sách các công ty từ đó tư vấn cho các ứng viên.

TÀI LIỆU THAM KHẢO

1. Nguyễn Văn Lân (2009), *Lập trình cơ sở dữ liệu với C# - Mô hình nhiều tầng*, Nxb Lao động - Xã hội.

2. Trịnh Thế Tiến, Nguyễn Minh (2009), *Các cơ sở dữ liệu Microsoft Visual C# 2008 - Lập trình căn bản và nâng cao*, Nxb Hồng Đức.

3. Tim Patrick (2010), *Microsoft ADO.NET 4 Step by Step*, Microsoft Press.