PHẠM HỮU KHANG *(Chủ biên)*

VB.NET

VÍ DỤ & BÀI TẬP Visual Basic.NET

- Ví dụ dễ hiểu
- Diễn giải rõ ràng
- Tính thực tiễn cao
- Bài tập đa dạng
- Bài giải chi tiết

Lập trình Cơ sở dữ liệu & Report

CD-ROM Kèm theo sách

NHÀ XUẤT BẢN LAO ĐỘNG XÃ HỘI

PHẠM HỮU KHANG *(Chủ biên)*
HOÀNG ĐỨC HẢI
PHƯƠNG LAN *(Hiệu đính)*

VÍ DỤ & BÀI TẬP Visual Basic.NET

LẬP TRÌNH

Cơ sở dữ liệu & Report

NHÀ XUẤT BẢN LAO ĐỘNG XÃ HỘI

GIỚI THIỆU

Visual Basic .NET cung cấp đối tượng ADO.NET bao gồm hai thành phần chính là .NET Data Provider và đối tượng DataSet, cho phép bạn kết nối với mọi loại cơ sở dữ liệu, xử lý và trình bày chúng bằng nhiều hình thức khác nhau.

Bằng cách sử dụng các đối tượng của ADO.NET, bạn có thể xây dựng các lớp dữ liệu cho từng thực thể hoặc xây dựng lớp tương tác với cơ sở dữ liệu dùng cho mọi Project trên nền .NET.

Ngoài ra, cuốn sách này còn cung cấp cho bạn những kiến thức liên quan đến việc trình bày dữ liệu trên các điều khiển, kết hợp các điều khiển, thao tác dữ liệu, nhằm tạo ra ứng dụng mang tính thực tiễn cao.

Các bước xây dựng và tương tác với Report được thiết kế bằng Crystal Report, trình bày chi tiết từ bước thiết kế cho đến cách truyền các tham số vào Report.

Cuốn sách bao gồm 8 chuyên đề chính, xuyên suốt từ đối tượng Connection, Command, DataReader, DataAdapter, DataSet, DataTable, DataView, DataRow, DataColumn, DataRelation đến việc trình bày dữ liệu trên các điều khiển và xây dựng các lớp dùng chung.

Giáo trình cung cấp cho bạn một khối lượng ví dụ hữu ích, đa dạng, trên 300 bài tập tham khảo có lời giải rõ ràng liên quan đến nhiều vấn đề thực tiễn.

HƯỚNG DẪN SỬ DỤNG VÍ DỤ VÀ BÀI TẬP CHUYÊN ĐỀ VISUAL BASIC.NET

Để sử dụng các ví dụ và bài tập đính kèm theo sách, trước tiên bạn cần quan tâm đến đường dẫn của ổ đĩa có tồn tại trên máy của bạn hay chưa nếu ví dụ hay bài tập tham chiếu đến ổ đĩa.

Những bài tập liên quan đến tài nguyên hay dịch vụ của hệ điều hành chỉ thực thi nếu hệ điều hành trên máy bạn là Windows 2000, XP hay 2003.

Ngoài ra, với những ví dụ hay bài tập có liên quan đến cơ sở dữ liệu, bạn cần xem lại hay thay đổi mật khẩu trong chuỗi kết nối cho phù hợp (nếu là cơ sở dữ liệu SQL Server, đường dẫn tập tin cơ sở dữ liệu Northwind.mdb nếu liên quan đến cơ sở dữ liệu Access).

Ví dụ và bài tập được tổ chức theo từng Solution, bao gồm nhiều Project, bạn có thể chọn từng Project để thực thi bằng cách khai báo mặc định Project khởi động hoặc chọn tên Project | R-Click | Debug | Start new instance.

Chú ý, mọi ví dụ và bài tập đính kèm theo sách được thiết kế không giới hạn trong phần nội dung lý thuyết đang trình bày. Chính vì lẽ đó, một ví dụ liên quan đến các dịch vụ hay máy in của hệ điều hành cũng được sử dụng mặc dù về mặt lý thuyết chúng chưa được trình bày.

Bên cạnh những lớp cơ sở của .NET Framework, trong ứng dụng đính kèm còn sử dụng lớp có tên clsSystem (đính kèm tập tin clsSystem.dll) và lớp myUserControl (đính kèm tập tin myUserControl.dll), bạn có thể tham chiếu lại trong Project nếu ứng dụng phát sinh lỗi khi thực thi.

MK.PUB

LỜI NGỎ

Kính thưa quý Bạn đọc gần xa, Ban xuất bản MK.PUB trước hết xin bày tỏ lòng biết ơn và niềm vinh hạnh trước nhiệt tình của đông đảo Bạn đọc đối với tủ sách MK.PUB trong thời gian qua.

Khẩu hiệu của chúng tôi là:

* Lao động khoa học nghiêm túc.

* Chất lượng và ngày càng chất lượng hơn.

* Tất cả vì Bạn đọc.

Rất nhiều Bạn đọc đã gửi *mail* cho chúng tôi đóng góp nhiều ý kiến quý báu cho tủ sách.

Ban xuất bản MK.PUB xin được kính mời quý Bạn đọc tham gia cùng nâng cao chất lượng tủ sách của chúng ta.

Trong quá trình đọc, xin các Bạn ghi chú lại các sai sót (dù nhỏ, lớn) của cuốn sách hoặc các nhận xét của riêng Bạn. Sau đó xin gửi về địa chỉ:

E-mail: *mk.book@minhkhai.com.vn - mk.pub@minhkhai.com.vn*

Hoặc gửi về: Nhà sách Minh Khai

249 Nguyễn Thị Minh Khai, Q.I, Tp. Hồ Chí Minh

Nếu Bạn ghi chú trực tiếp lên cuốn sách, rồi gửi cuốn sách đó cho chúng tôi thì chúng tôi sẽ xin hoàn lại cước phí bưu điện và gửi lại cho Bạn cuốn sách khác.

Chúng tôi xin gửi tặng một cuốn sách của tủ sách MK.PUB tùy chọn lựa của Bạn theo một danh mục thích hợp sẽ được gửi tới Bạn.

Với mục đích ngày càng nâng cao chất lượng của tủ sách MK.PUB, chúng tôi rất mong nhận được sự hợp tác của quý Bạn đọc gần xa.

"MK.PUB và Bạn đọc cùng làm !"

MK.PUB

MỤC LỤC

PHẦN I:

LÝ THUYẾT TÓM TẮT & VÍ DỤ CHUYÊN ĐỀ

Chuyên đề 16:
LÀM VIỆC VỚI ĐỐI TƯỢNG ADO.NET

Tóm tắt chuyên đề 16

Khi làm việc với .NET, để kết nối với các cơ sở dữ liệu như: SQL Server, Access, Excel, Oracle, MySQL và những cơ sở dữ liệu khác, bạn có thể sử dụng đối tượng ADO.NET.

Bằng cách sử dụng các trình điều khiển và đối tượng ADO.NET, bạn có thể tương tác với loại dữ liệu như SQL Server, Access hay Excel.

Trong chuyên đề này, chúng ta sẽ tìm hiểu cách khai báo sử dụng các trình điều khiển cơ sở dữ liệu bằng ngôn ngữ lập trình Visual Basic.NET.

Các vấn đề chính sẽ được đề cập:

- ✓ *Giới thiệu ADO.NET.*
- ✓ *Trình điều khiển cơ sở dữ liệu.*
- ✓ *Đối tượng SqlConnection.*
- ✓ *Đối tượng OleDbConnection.*
- ✓ *Đối tượng SqlCommand.*
- ✓ *Làm việc với đối tượng SqlParameter.*

1. ĐỐI TƯỢNG ADO.NET TRONG VISUAL BASIC.NET

Thành phần của *ADO.NET* được thiết kế nhằm tăng tốc độ truy cập và thao tác dữ liệu trong môi trường đa lớp (*n-tier*). Hai thành phần chính của *ADO.NET* là đối tượng *DataSet* (thuộc lớp không kết nối) và trình điều khiển cơ sở dữ liệu *.NET* (*.NET Data Provider)* thuộc lớp kết nối.

.NET Data Provider là tập các đối tượng thuộc thành phần *ADO.NET*, bao gồm đối tượng *Connection* (kết nối cơ sở dữ liệu), đối

tượng *Command* (thực thi phát biểu *SQL*), *DataReader* (bộ đọc dữ liệu trực tiếp) và *DataAdapter* (bộ điều phối dữ liệu).

Với cơ chế ba tầng trong ứng dụng *.NET* (tầng dữ liệu *Data tier*, tầng *Business tier* và tầng hiện thực *Presentation tier*) tách biệt, cho phép người dùng truy cập dữ liệu từ tầng hiện thực thông qua tầng *Business tier*.

Trong *ADO.NET* khái niệm con trỏ (*Cursor*) không tồn tại, thay vào đó đối tượng *DataSet* được xem như một con trỏ tĩnh (*static*) và đối tượng *DataReader* là con trỏ chỉ đọc (*Readonly Curosor*).

ADO.NET cung cấp các trình điều khiển tương ứng với các loại cơ sở dữ liệu. Chẳng hạn, *SQL Server .NET Data Provider* dùng cho cơ sở dữ liệu *SQL Server 7.0/2000*.

OleDb .NET Data Provider là trình điều khiển dùng cho các loại cơ sở dữ liệu tương tác nhúng như: Cơ sở dữ liệu *Access*, *Excel* hay cơ sở dữ liệu *SQL Server* 6.5. Tuy nhiên, bạn cũng có thể sử dụng trình điều khiển này cho cơ sở dữ liệu *SQL Server 7.0/2000*.

Odbc .NET Data Provider là trình điều khiển dùng cho các loại cơ sở dữ liệu tương tác thông qua *ODBC* của hệ điều hành *Windows*.

Trong trường hợp làm việc với cơ sở dữ liệu *Oracle* hay *MySQL*, bạn cần tải trình điều khiển *Oracle .NET Data Provider* hay *MySQL .NET Data Provider*, sau đó cài đặt chúng vào hệ thống.

Trong chuyên đề này, chúng ta chỉ tập trung tìm hiểu cách làm việc với 3 loại cơ sở dữ liệu chính của *Microsoft* là *SQL Server*, *Access* và *Excel*.

2. TRÌNH ĐIỀU KHIỂN CƠ SỞ DỮ LIỆU

Khi lấy dữ liệu từ nguồn dữ liệu (dữ liệu quan hệ, tập tin văn bản, thông tin trong *email*, ...), bạn cần quản lý và sử dụng trình điều khiển (*Provider*).

Do nội dung cuốn sách chỉ tập trung tìm hiểu 3 loại cơ sở dữ liệu chính là *SQL Server*, *Access* và *Excel*. Chính vì vậy, chúng ta sử dụng hai trình điều khiển chính là *SQL Server .NET Data Provider* và *OleDb .NET Data Provider*.

Provider được xem như một cầu nối giữa ứng dụng với cơ sở dữ liệu, chúng dùng để kết nối với dữ liệu nguồn, thực thi phát biểu *SQL* và nhận dữ liệu trả về. Những dữ liệu trả về có thể được xử lý trực tiếp hay lưu trữ trên đối tượng *DataSet* (thuộc lớp không kết nối, chúng ta sẽ tìm hiểu chi tiết trong chuyên đề kế tiếp).

Nếu dữ liệu lưu trữ trên đối tượng *DataSet*, chúng có thể trình bày cho người sử dụng với nhiều hình thức khác nhau, như kết hợp dữ liệu từ nhiều bảng hay dữ liệu truy xuất từ nhiều loại cơ sở dữ liệu khác nhau.

Như trình bày ở hình trên, *.NET Provider* cho phép bạn cập nhật sự thay đổi của dữ liệu trên đối tượng *DataSet* vào dữ liệu nguồn. Khi lập trình ứng dụng *Visual Basic.NET*, khai báo và sử dụng hai trình điều khiển trên về cơ bản là tương tự như nhau dù hai đối tượng này thuộc không gian tên *System.Data.SqlClient* và *System.Data.OleDb*.

Mặc dù, hai trình điều khiển trên thuộc hai không gian tên khác nhau nhưng chúng cung cấp các chức năng tương tự nhau khi làm việc với cơ sở dữ liệu.

Chúng ta sẽ tham khảo sơ lược sự tương ứng các đối tượng thuộc hai trình diều khiển *SQL Server.NET Data Provider* và *OLE DB.NET Data Provider* như sau:

SQL Server.NET Data Provider	OLE DB.NET Data Provider
SqlConnection	OleDbConnection
SqlCommand	OleDbCommand
SqlDataAdapter	OleDbDataAdapter
SqlDataReader	OleDbDataReader
SqlParameter	OleDbParameter

Các đối tượng thuộc một trong hai trình điều khiển trình bày trong bảng trên chứa đựng các đối tượng hiện thực dữ liệu tương ứng với phần tử trong *.NET Provider* là: *Connection, Command, DataReader, DataAdapter* như sau:

Đối tượng	Diễn giải
Connection	Thiết lập kết nối cơ sở dữ liệu với nguồn dữ liệu.
Command	Thực hiện lên trên nguồn dữ liệu, đặt vào các tham số và thu nhận các chuyển tác từ kết nối *Connection*
DataReader	Đọc luồng dữ liệu từ dữ liệu nguồn theo một chiều và dữ liệu chỉ đọc.
DataAdapter	Lưu trữ đối tượng *DataSet* và giải quyết vấn đề thao tác, xử lý và cập nhật dữ liệu trở lại dữ liệu nguồn.

3. ĐỐI TƯỢNG SQLCONNECTION

Khi làm việc với cơ sở dữ liệu *SQL Server 7.0/2000*, bạn sử dụng không gian tên *System.Data.SqlClient*. Ngược lại, trường hợp bạn làm việc với cơ sở dữ liệu *Access* hay *Excel* thì khai báo và sử dụng không gian tên *System.Data.OleDb*.

Để kết nối cơ sở dữ liệu *SQL Server*, bạn sử dụng đối tượng *SqlConnection*. Sau đây, chúng ta tham khảo từng công đoạn khai báo và thực hiện của chúng.

3.1. Khai báo không gian tên

Để khai báo và sử dụng đối tượng *SqlConnection*, trước tiên bạn phải khai báo trên phần đầu mỗi *Class* phát biểu *Imports* dùng để nạp không gian tên *System.Data.SqlClient* như sau:

```
Imports System.Data.SqlClient
```

3.2. Khai báo đối tượng SqlConnection

Khi đã khai báo không gian tên, bằng cách sử dụng cú pháp như sau, bạn khai báo để sử dụng đối tượng *SqlConnection*:

```
Dim myConn As SqlConnection
```

3.3. Khởi tạo đối tượng SqlConnection

Sau khi khai báo biến đối tượng *SqlConnection*, bạn có thể khởi tạo đối tượng này với chuỗi kết nối (bao gồm các thông tin *IP* hay tên *Server*, *Database* là tên cơ sở dữ liệu).

```
myConn = New SqlConnection("chuỗi kết nối")
```

Hoặc sử dụng thuộc tính *ConnectionString* để gán chuỗi kết nối cơ sở dữ liệu:

```
myConn = New SqlConnection()
myConn.ConnectionString="Chuỗi kết nối"
```

Lưu ý, bạn cũng có thể vừa khai báo vừa khởi tạo đối tượng *SqlConnection* với biến *gsCon* (chuỗi kết nối cơ sở dữ liệu) trên cùng một câu lệnh như sau:

```
Dim gsCon As String
gsCon="Server=.;Database=Northwind"
gsCon +=";User ID=sa;Password=sa;"
Dim myConn As SqlConnection = _
            New SqlConnection(gsCon)
```

Hoặc:

```
Dim gsCon As String
gsCon="Server=.;Database=Northwind"
gsCon +=";User ID=sa;Password=sa;"
Dim myConn As New SqlConnection(gsCon)
```

Tùy thuộc vào cách sử dụng đặt quyền truy cập cơ sở dữ liệu, với tài khoản của cơ sở dữ liệu hay sử dụng tài khoản của hệ điều hành, nội dung của chuỗi kết nối cơ sở dữ liệu *SQL Server* sẽ khác nhau.

3.3.1. Sử dụng SQL Server Authentication

Nếu sử dụng *SQL Server Authentication* (đăng nhập bằng đặt quyền cơ sở dữ liệu *SQL Server*), chuỗi kết nối bao gồm các thuộc tính như: *Server, Database, User ID* và *Password*.

Ví dụ, khai báo chuỗi kết nối cơ sở dữ liệu *SQL Server* với các thuộc tính như sau:

```
Dim gsCon As String
gsCon="Server=.;Database=Northwind"
gsCon +=";User ID=sa;Password=sa;"
```

Tuy nhiên, bạn cũng có thể sử dụng *UID* và *PWD* thay vì sử dụng từ khóa *User ID* và *Password* như sau:

```
Dim gsCon As String
gsCon="Server=.;Database=Northwind"
gsCon +=";UID=sa;PWD=sa;"
```

Nếu mật khẩu là rỗng, bạn cũng khai báo thuộc tính *PWD* trong chuỗi kết nối:

```
Dim gsCon As String
gsCon="Server=.;Database=Northwind"
gsCon +=";UID=sa;PWD=;"
```

Lưu ý, khi sử dụng *SQL Server Authentication*, nghĩa là *SQL Server* được cấu hình tùy chọn *SQL Server* and *Windows* trong phần *Security* của *SQL Server*.

Để tham khảo chi tiết về điều này, bạn có thể tìm đọc cuốn sách *"Quản trị SQL Server 2000"* do nhà sách Minh Khai phát hành.

3.3.2. Sử dụng Windows Authentication

Nếu sử dụng *Windows Authentication*, chuỗi kết nối bao gồm các thuộc tính như: *Server, Database*. Khi đó, thuộc tính *User ID* và *Password* chính là *Username* và *Password* mà người sử dụng đang đăng nhập hệ điều hành.

Ví dụ, bạn khai báo chuỗi kết nối cơ sở dữ liệu *SQL Server* với các thuộc tính như sau:

```
Dim gsCon As String
gsCon="Server=.;Database=Northwind;"
```

```
gsCon +="Integrated Security=SSPI;"
```

Tuy nhiên, khi sử dụng *Windows Authentication*, tức trong *SQL Server* đã tồn tại những *User* của hệ điều hành, điều này có nghĩa là bạn đã đăng ký người sử dụng thuộc hệ điều hành trong phần *login* của *SQL Server*.

3.4. Mở kết nối cơ sở dữ liệu

Sau khi khai báo và khởi tạo đối tượng *SqlConnection*, bạn có thể mở kết nối cơ sở dữ liệu *SQL Server* bằng cách sử dụng phương thức *Open()* với cú pháp như sau:

```
myConn.Open()
```

3.5. Kiểm tra trạng thái của kết nối

Bằng cách sử dụng thuộc tính *State* của đối tượng *Connection* để so sánh với *ConnectionState* của không gian tên *System.Data*, bạn có thể kiểm tra xem kết nối đang ở trạng thái đóng hay không.

```
If myConn.State <> ConnectionState.Closed Then
   myConn.Close()
   myConn.Dispose()
End If
```

3.6. Đóng kết nối

Sau khi thực thi một số công việc trên cơ sở dữ liệu, như thực thi phát biểu *SQL* hay khai báo và sử dụng các đối tượng khác, nếu không sử dụng đối tượng *SqlConnection*, bạn khai báo đóng kết nối này bằng cách khai báo phương thức *Close()* như sau:

```
myConn.Close()
```

3.7. Giải phóng bộ nhớ

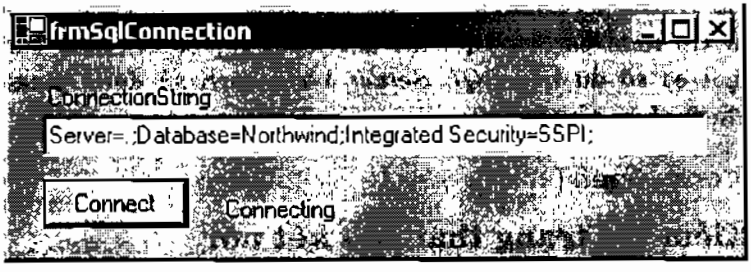
Nếu sau khi đóng đối tượng *SqlConnection*, bạn còn tiếp tục mở với cơ sở dữ liệu khác thì không cần giải phóng bộ nhớ do biến này chiếm giữ.

Trong trường hợp chương trình không tiếp tục sử dụng đối tượng, bạn nên giải phóng bộ nhớ biến này bằng cách sử dụng phương thức *Dispose* như khai báo sau:

```
myConn.Dispose()
```

Lưu ý, kể từ khi mở kết nối cơ sở dữ liệu đến khi xử lý tập lệnh (sau khi mở kết nối), bạn cần cài đặt phát biểu *Try Catch Finally End Try* để kiểm soát lỗi phát sinh nếu có.

Chẳng hạn, thêm *Form* vào *Project* và đặt tên *frmSqlConnect*, thiết kế các điều khiển trông giống như hình 16-1.



Hình 16-1: *Kết nối cơ sở dữ liệu SQL Server*

Trong biến cố *Click* của nút *Button1*, bạn khai báo kết nối cơ sở dữ liệu *SQL Server* bằng đặt quyền *Windows Authentication* như ví dụ 16-1.

Ví dụ 16-1: Kết nối cơ sở dữ liệu SQL Server

```
Private Sub Button1_Click(ByVal sender As _
System.Object, ByVal e As System.EventArgs) _
Handles Button1.Click
    Dim gsCon As String= TextBox1.Text
    Dim myConn As New SqlConnection(gsCon)
    Try
        myConn.Open()
        Label2.Text = "Connected"
    Catch ex As Exception
        Label2.Text = ex.Message
    Finally
        myConn.Close()
        myConn.Dispose()
    End Try
End Sub
```

Lưu ý, bạn cần khai báo không gian tên trên đầu *Class* như sau:

```
Imports System.Data.SqlClient
```

Để tìm hiểu thêm về sử dụng *SqlConnection*, bạn có thể thực hiện các ví dụ sau:

1. Thiết kế *Form*, cho phép người sử dụng nhập *UserName* và *Password*, sau đó khai báo đoạn chương trình để kết nối cơ sở dữ liệu. Nếu kết nối không thành công thì phun lỗi ra màn hình bằng lệnh *MsgBox*.
2. Viết ứng dụng *Console*, sử dụng chuỗi kết nối cơ sở dữ liệu ứng với *Windows Authentication*, nếu kết nối thành công bạn liệt kê danh sách các bảng có trong cơ sở dữ liệu đó ra màn hình *Command Prompt*.

4. KHAI BÁO VÀ SỬ DỤNG OLEDBCONNECTION

Tương tự như khai báo biến đối tượng *SqlConnection* dùng cho cơ sở dữ liệu *SQL Server* vừa trình bày ở trên, khi sử dụng cơ sở dữ liệu *MS Access* hay *Excel*, bạn có thể dùng đối tượng *OleDbConnection*.

4.1. Khai báo không gian tên

Để khai báo và sử dụng đối tượng *OleDbConnection*, trước tiên bạn phải thêm phát biểu như sau vào phần đầu của *Class*:

```
Imports System.Data.OleDb
```

4.2. Khai báo biến đối tượng OleDbConnection

Để khai báo đối tượng *OleDbConnection* kết nối cơ sở dữ liệu *Access* hay *Excel*, bạn sử dụng đối tượng *OleDbConnection*. Chẳng hạn, bạn khai báo để sử dụng đối tượng này với cú pháp như sau:

```
Dim myConn As OleDbConnection
```

4.3. Khởi tạo biến đối tượng OleDbConnection

Sau khi khai báo biến đối tượng *OleDbConnection*, bạn có thể khởi tạo biến đối tượng này với chuỗi kết nối, bao gồm tên và đường dẫn tập tin cơ sở dữ liệu, trình điều khiển cơ sở dữ liệu *Access* như sau:

```
myConn = New OleDbConnection("chuỗi kết nối")
```

4.4. Chuỗi kết nối cơ sở dữ liệu Access

Nếu bạn kết nối cơ sở dữ liệu *Access*, chuỗi kết nối khai báo tương tự như sau:

```
Dim gsCon As String
gsCon="Provider=Microsoft.Jet.OLEDB.4.0;"
gsCon +="Data Source=Northwind.mdb"
```

4.5. Chuỗi kết nối dữ liệu Excel

Trong trường hợp kết nối dữ liệu *Excel*, chuỗi kết nối được khai báo tương tự như sau:

```
Dim gsCon As String
gsCon="Provider=Microsoft.Jet.OLEDB.4.0;"
gsCon +="Data Source=D:\Book1.xls;Extended "
gsCon +="Properties=""Excel 8.0;HDR=YES;"""
```

Lưu ý, nếu cơ sở dữ liệu phiên bản 10.0, bạn có thể khai báo chuỗi kết nối trên thành:

```
Dim gsCon As String
gsCon="Provider=Microsoft.Jet.OLEDB.4.0;"
gsCon +="Data Source=D:\Book1.xls;Extended "
gsCon +="Properties=""Excel 10.0;HDR=YES;"""
```

Ngoài ra, trong trường hợp khai báo đối tượng *SqlConnection*, bạn có thể vừa khai báo và khởi tạo biến đối tượng *OleDbConnection* trên cùng một câu lệnh:

```
Dim gsCon As String
gsCon="Provider=Microsoft.Jet.OLEDB.4.0;"
gsCon +="Data Source=mydatabase.mdb"
Dim myConn As SqlConnection = _
            New SqlConnection(gsCon)
```

4.6. Mở kết nối cơ sở dữ liệu

Sau khi khai báo và khởi tạo đối tượng *OleDbConnection*, bạn có thể mở kết nối cơ sở dữ liệu *Excel* hoặc *Access* bằng phương thức *Open()* với cú pháp như sau:

```
myConn.Open()
```

4.7. Đóng kết nối

Sau khi thực hiện một số công việc, ví dụ như thực thi phát biểu *SQL* hay khai báo và sử dụng các đối tượng nào đó, nếu chương trình

không có nhu cầu sử dụng đối tượng *Connection*, bạn phải khai báo đóng kết nối này bằng phương thức *Close()* như sau:

```
myConn.Close()
```

4.8. Giải phóng bộ nhớ

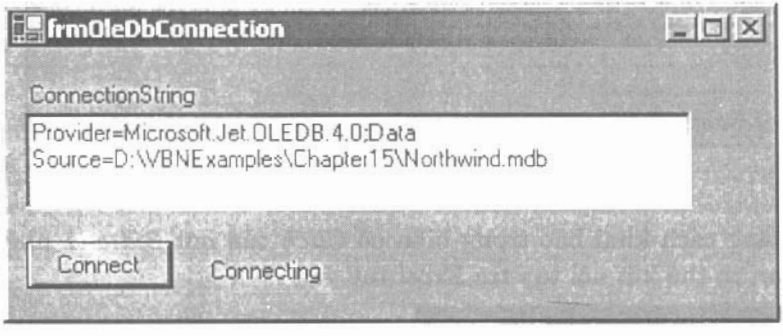
Tương tự như trường hợp làm việc với cơ sở dữ liệu *SQL Server*, khi đóng đối tượng *Connection* của cơ sở dữ liệu *Access* mà bạn tiếp tục mở với cơ sở dữ liệu khác thì không cần giải phóng biến đối tượng này.

Tuy nhiên, nếu không tiếp tục sử dụng đối tượng *Connection*, bạn nên giải phóng bộ nhớ bằng cách sử dụng phương thức *Dispose* như khai báo sau:

```
myConn.Dispose()
```

Tương tự như trường hợp *SqlConnection*, bạn cần cài đặt phát biểu *Try .. Catch .. Finally .. End Try*, để kiểm soát lỗi phát sinh khi mở cơ sở dữ liệu *Access*.

Ví dụ, thêm *Form* và đặt tên *frmOleDbConnect* vào *Project*, rồi thiết kế các điều khiển trên *Form* như hình 16-2.



Hình 16-2: *Kết nối cơ sở dữ liệu Access*

Trong biến cố *Click* của nút *Button1*, bạn khai báo kết nối cơ sở dữ liệu *Access* như ví dụ 16-2.

Ví dụ 16-2: Kết nối cơ sở dữ liệu Access

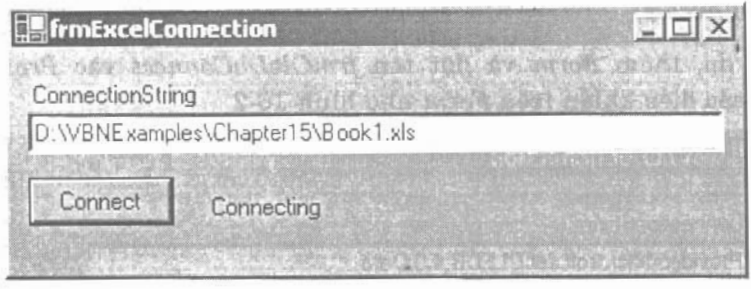
```
Private Sub Button1_Click(ByVal sender As _
System.Object, ByVal e As System.EventArgs) _
Handles Button1.Click
    Dim gsCon As String= TextBox1.Text
```

```
        Dim myConn As New OleDbConnection(gsCon)
        Try
                myConn.Open()
                Label2.Text = "Connected"
        Catch ex As Exception
                Label2.Text = ex.Message
        Finally
                myConn.Close()
                myConn.Dispose()
        End Try
End Sub
```

Chú ý, bạn cần khai báo không gian tên *System.Data.OleDb* trên phần đầu của *Class* như sau:

```
Imports System.Data.OleDb
```

Tiếp tục tương tác với tập tin *Excel*, bạn thêm *Form* vào *Project*, đặt tên là *frmExceConnection* và thiết kế các điều khiển như hình 16-3.



Hình 16-3: *Đọc dữ liệu từ Excel*

Bằng cách khai báo trong biến cố *Click* của nút *Button1* như ví dụ 16-3, bạn có thể kết nối tập tin *Excel* tùy ý.

Ví dụ 16-3: Kết nối tập tin Excel

```
Private Sub Button1_Click(ByVal sender As _
System.Object, ByVal e As System.EventArgs) _
Handles Button1.Click
    'Khai báo chuỗi kết nối tập tin Excel
    Dim gsCon As String
    gsCon = "Provider=Microsoft.Jet.OLEDB.4.0;"
    gsCon += "Data Source=" + TextBox1.Text
    gsCon += ";Extended Properties="
    gsCon += """Excel 8.0;HDR=YES;"""
```

```
    Dim myConn As New OleDbConnection(gsCon)
    Try
      myConn.Open()
      Label2.Text = "Connected"
    Catch ex As Exception
      Label2.Text = ex.Message
    Finally
      myConn.Close()
      myConn.Dispose()
    End Try
End Sub
```

Trong các chuyên đề của cuốn *"Ví dụ và bài tập Visual Basic .Net – Lập trình Windows forms và tập tin"*, bạn đã tham khảo tập tin *App.config*, tập tin bao gồm các thuộc tính như: *server*, *database*, *data source*, ..., mỗi khi có nhu cầu tương tác với cơ sở dữ liệu, chuỗi kết nối được kết hợp từ việc đọc các thuộc tính này.

Ví dụ, bạn có thể khai báo chuỗi kết nối trong tập tin *App.config* bằng định dạng ***XML*** như ví dụ 16-4.

Ví dụ 16-4: Tập tin App.config

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
 <appSettings>
     <add key="server" value="."/>
     <add key="database" value="northwind"/>
     <add key="access"
     value="Provider=Microsoft.Jet.OLEDB.4.0;"
     Data Source= "C:\Program Files\Microsoft
       Office\Office\Samples\Northwind.mdb" />
  </appSettings>
</configuration>
```

Sau đó, để đọc được các khóa khai báo trong tập tin *App.config*, bạn sử dụng không gian tên *System.Configuration*.

```
Imports x= _
System.Configuration.ConfigurationSettings
```

Tiếp tục khai báo để lấy giá trị từ các khóa vào hai biến toàn cục trong phương thức *getDBInfo* như sau:

```
Sub getDBInfo()
```

```
        gsServer = x.AppSettings("server")
        gsDatabase = x.AppSettings("database")
    End Sub
```

Lưu ý, bạn có thể lưu thông tin chuỗi kết nối cơ sở dữ liệu trong *Windows Registry* hoặc tập tin định dạng *Text*.

Để tìm hiểu thêm về sử dụng *OleDbConnection*, bạn có thể thực hiện các ví dụ sau:

1. Thiết kế *Form*, cho phép người sử dụng nhập *Username* và *Password*, sau đó khai báo đoạn chương trình để kết nối cơ sở dữ liệu và kiểm tra trong bảng *tblUsers* của cơ sở dữ liệu *Access* tùy chọn. Nếu kết nối không thành công thì phun lỗi ra màn hình bằng lệnh *MsgBox*.
2. Viết ứng dụng *Console*, sử dụng chuỗi kết nối liệu *Excel*, nếu kết nối thành công, bạn liệt kê danh sách dữ liệu trong bảng tính (*Sheet*) ra màn hình *Command Prompt*.
3. Tương tự như trên, bạn khai báo một *Class* và đặt tên *clsExcel*, bằng cách khai báo và khởi tạo đối tượng *OleDbConnection* trong *Constructor* của *Class* với chuỗi kết nối xem như tham số truyền vào *Constructor*. Sau đó, khai báo ứng dụng để sử dụng *clsExcel* này bằng cách truyền hoặc gán chuỗi kết nối vào thuộc tính của đối tượng *clsExcel*.

5. ĐỐI TƯỢNG SQLCOMMAND

Sau khi đã làm quen cách kết nối cơ sở dữ liệu *SQL Server* bằng đối tượng *SqlConnection* hay cơ sở dữ liệu *Access* và *Excel* bằng đối tượng *OleDbConnection*, bạn dễ nhận thấy tập lệnh đã khai báo tương tự nhau.

Khi làm việc với đối tượng *Command*, chúng ta chọn đối tượng *SqlConnection* và cơ sở dữ liệu *SQL Server* để tìm hiểu cách sử dụng với chúng.

Nếu muốn thực thi phát biểu *SQL* dạng hành động (*Delete, Insert, Update, Create, Drop, Alter*, các thủ tục (*Stored Procedure*), hoặc phát biểu *Select*, bạn có thể sử dụng các phương thức và thuộc tính của đối tượng *SqlCommand*.

5.1. Khai báo đối tượng SqlCommand

Để khai báo và sử dụng đối tượng *SqlCommand*, bạn thao tác từng bước tương tự như khai báo và khởi tạo đối tượng *SqlConnection*.

Chẳng hạn, bạn sử dụng cú pháp để khai báo đối tượng *SqlCommand* như sau:

```
Dim myCom As SqlCommand
```

Nếu làm việc với cơ sở dữ liệu *Access*, nghĩa là trước đó bạn khai báo và khởi tạo đối tượng *OleDbConnection* thì dùng cú pháp sau:

```
Dim myCom As OleDbCommand
```

5.2. Khởi tạo đối tượng SqlCommand

Sau khi khai báo đối tượng *SqlCommand*, để khởi tạo và sử dụng đối tượng *SqlCommand*, bạn có thể dùng cú pháp sau:

```
myCom= New SqlCommand(strSQL,myConn)
```

Ngoài ra, bạn cũng có thể khởi tạo đối tượng *SqlCommand* không có tham số, nhưng sau đó sử dụng hai thuộc tính *CommandText* và *Connection* để khai báo như sau:

```
myCom= New SqlCommand()
myCom.CommandText=strSQL
myCom.Connection =myConn
```

Ví dụ, trong trường hợp bạn kết nối cơ sở dữ liệu *SQL Server* có tên *Northwind*, sau đó lấy giá trị của cột và hàng thứ nhất trong tập trả về, bạn khai báo như sau:

```
' Khai báo chuỗi kết nối cơ sở dữ liệu SQL Server
Dim gsCon As String
gsCon="database=Northwind;UID=sa;PWD=sa;"
gsCon +="server=."
' Khai báo và khởi tạo đối tượng SqlConnection
Dim myConn As SqlConnection()
myConn= New SqlConnection(gsCon)
' Khai báo chuỗi SQL
Dim strSQL As String
strSQL="Select CustomerID from Customers"
```

```
' Khai báo và khởi tạo đối tượng SqlCommand
Dim myCom As SqlCommand()
myCom =New SqlCommand(strSQL,myConn)
strSQL= myCom.ExecuteScalar()
```

Chú ý, nếu bạn sử dụng đối tượng *OleDbConnection* để làm việc với cơ sở dữ liệu *Access* thì khai báo và sử dụng đối tượng *OleDbCommand* như sau:

```
' Khai báo chuỗi kết nối cơ sở dữ liệu Access
Dim gsCon As String
gsCon="Provider=Microsoft.Jet.OLEDB.4.0;"
gsCon +="Data Source=Northwind.mdb"
' Khai báo và khởi tạo đối tượng OleDbConnection
Dim myConn As OleDbConnection()
myConn= New OleDbConnection(gsCon)
' Khai báo chuỗi SQL
Dim strSQL as String
strSQL-"Select CustomerID from Customers"
' Khai báo và khởi tạo đối tượng OleDbCommand
Dim myCom As OleDbCommand()
myCom =New OleDbCommand(strSQL,myConn)
strSQL= myCom.ExecuteScalar()
```

5.3. Thuộc tính

Đối tượng *SqlCommand* cung cấp các thuộc tính như: *CommandText*, *CommandType*, *Connection*, *CommandTimeout* và *Transaction*.

- CommandText: Chuỗi *SQL* hay tên đối tượng.
- *CommandType*: Loại đối tượng cung cấp trong thuộc tính *CommandText*. Chẳng hạn, bạn khai báo thủ tục nội tại của *SQL Server* cho thuộc tính *CommandText* thì giá trị khai báo cho thuộc tính *CommandType* là *StoredProcedure*.
- *Connection*: Đối tượng *Connection* tương ứng. Chẳng hạn, khi bạn sử dụng *SqlCommand* thì thuộc tính này có thể gán là đối tượng *SqlConnection*.
- *CommandTimeout*: Thời gian chờ tính bằng giây trước khi kết thúc thực thi.

- *Transaction*: Cho phép bạn khai báo chuyển tác với ba thuộc tính là *Save*, *Commit* và *RollBack*.

5.4. Phương thức

Đối tượng *SqlCommand* cung cấp 3 phương thức chính để thực thi phát biểu *SQL* là *ExecuteNonQuery*, *ExecuteScalar* và *ExecuteReader*.

5.4.1. Phương thức ExecuteNonQuery

Phương thức *ExecuteNonQuery* trả về giá trị số nguyên, tương ứng với số mẩu tin đã thực thi, được sử dụng cho các phát biểu *SQL* như cập nhật dữ liệu, xóa, thêm mới mẩu tin hay các phát biểu dạng hành động và thủ tục nội tại trong *SQL Server*.

Chẳng hạn, bạn khai báo và sử dụng phương thức *ExecuteNonQuery* của đối tượng *Command* trong phương thức *doSQL* thuộc lớp có tên *clsDatabase* như ví dụ 16-5.

Ví dụ 16-5: Phương thức thực thi phát biểu SQL

```
Function doSQL(ByVal strSQL As String, ByRef
EffectedRecord As Integer) As String
    Dim strError As String = ""
  ' Khai báo và khởi tạo đối tượng SqlConnection
  Dim myConn As New SqlConnection(gsCon)
  ' Khai báo và khởi tạo đối tượng SqlCommand
  Dim myCom As New SqlCommand(strSQL, myConn)
    ' Sử dụng cấu trúc quản lý lỗi
    Try
      ' Mở kết nối cơ sở dữ liệu
      myConn.Open()
      ' Thực thi phát biểu SQL dạng hành động, trả về số mẩu
      ' tin đã thực thi
       EffectedRecord =myCom.ExecuteNonQuery()
      ' Đóng kết nối cơ sở dữ liệu
     myConn.Close()
    Catch myException As Exception
      ' Nếu lỗi phát sinh trả về chuỗi lỗi
      strError=myException.ToString()
    Finally
      ' Sau khi kết thúc xử lý lỗi phát sinh, bạn nên kiểm tra
```

```
        ' kết nối cơ sở dữ liệu còn mở hay không, nếu còn mở thì
        ' bạn khai báo đóng kết nối
        If (myConn.State = ConnectionState.Open)
            Then
          myConn.Close()
        End If
    End Try
End Function
```

Đối với trường hợp làm việc với cơ sở dữ liệu *Access* có tên *Northwind*, bạn có thể sử dụng phương thức *ExecuteNonQuery* của đối tượng *OleDbConnection* để thực thi phát biểu *SQL* như ví dụ 16-6.

Ví dụ 16-6: Phương thức thực thi phát biểu SQL ứng với Access

```
Function doSQL(ByVal strSQL As String, ByRef
EffectedRecord As Integer) As String
    Dim strError As String = ""
  ' Khai báo và khởi tạo đối tượng OleDbConnection
  Dim myConn As New OleDbConnection(gsCon)
  ' Khai báo và khởi tạo đối tượng SqlCommand
  Dim myCom As New OleDbCommand(strSQL, myConn)
    ' Sử dụng cấu trúc quản lý lỗi
    Try
      ' Mở kết nối cơ sở dữ liệu OleDb
      myConn.Open()
      ' Thực thi phát biểu SQL dạng hành động
      myCom.ExecuteNonQuery()
      ' Đóng kết nối cơ sở dữ liệu
      myConn.Close()
    Catch myException As Exception
      ' Nếu lỗi phát sinh trả về chuỗi lỗi
      strError=myException.ToString()
    Finally
      ' Sau khi kết thúc xử lý lỗi phát sinh, bạn nên kiểm tra
      ' kết nối cơ sở dữ liệu còn mở hay không, nếu còn mở thì
      ' bạn khai báo đóng kết nối
      If (myConn.State = ConnectionState.Open)
          Then
        myConn.Close()
      End If
```

```
        End Try
    End Function
```

Trong ví dụ trên, giá trị của biến *gsCon* được lấy từ tập tin *App.Config* hoặc bạn có thể lấy từ tập tin dạng *Text* với định dạng như sau:.

```
    gsCon ="Provider=Microsoft.Jet.OLEDB.4.0;"
    gsCon +="Data Source=C:\Program
Files\Microsoft"
    gsCon +=" Office\Office\Samples\Northwind.mdb"
```

Hay đối với cơ sở dữ liệu *SQL Server* thì chuỗi kết nối cơ sở dữ liệu có hình dạng như sau:

```
    gsCon="server=.;database=Northwind;"
    gsCon +="uid=sa;pwd=sa"
```

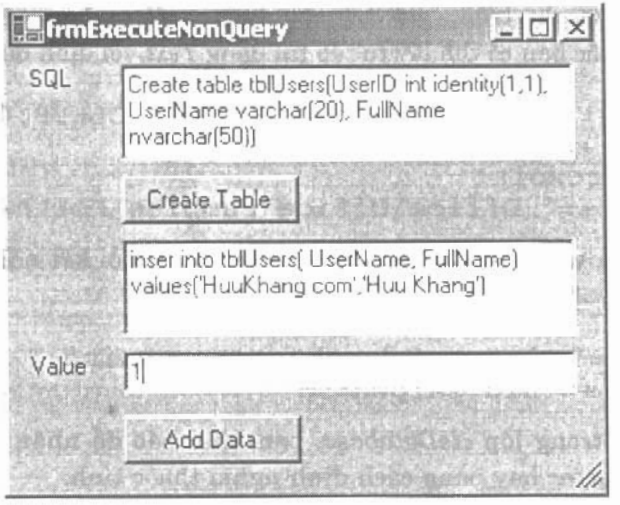
Lưu ý, trong lớp *clsDatabase*, bạn khai báo để nhận chuỗi kết nối trong *Constructor* hay bằng cách định nghĩa thuộc tính:

```
    Private gsCon As String
    ' Khai báo Constructor
    Sub New(ByVal strCon As String)
      gsCon = strCon
    End Sub
    ' Khai báo Constructor
    Sub New()

    End Sub
    ' Khai báo thuộc tính
    Property Connection() As String
      Get
        Return gsCon
      End Get
      Set(ByVal Value As String)
        gsCon = Value
      End Set
    End Property
```

Chuỗi kết nối cơ sở dữ liệu được truyền vào đối tượng *clsDatabase* dưới hình thức là tham số của *Constructor* mỗi khi chúng được khởi tạo hoặc bạn có thể gán vào thuộc tính sau khi đối tượng được khởi tạo.

Để tham khảo thêm cách làm này, bạn thêm *Form* vào *Project* với tên *frmExecuteNonQuery*, thêm điều khiển *TextBox* để nhập phát biểu

tạo bảng dữ liệu (*Create Table*) và điều khiển *TextBox* thứ hai để nhận số mẩu tin thực thi từ phát biểu *Insert*.



Hình 16-4: *Thực thi phát biểu SQL*

Sau đó, bạn khai báo trong biến cố *Click* của nút *Button1* để gọi phương thức *doSQL* của lớp *clsDatabase* như ví dụ 16-7.

Ví dụ 16-7: Gọi phương thức doSQL

```
Private Sub Button1_Click(ByVal sender As _
System.Object, ByVal e As System.EventArgs) _
Handles Button1.Click
    ' Khai báo và khởi tạo đối tượng clsDatabase
      Dim cls As New clsDatabase(gsCon)
    ' Gọi phương thức doSQL
    strError = cls.doSQL(TextBox1.Text)

End Sub
```

Khi thực thi phát biểu *Insert* dùng để thêm mẩu tin vào bảng vừa tạo ra từ ví dụ trên, bạn khai báo như ví dụ 16-8.

Ví dụ 16-8: Thực thi phát biểu SQL

```
Private Sub Button2_Click(ByVal sender As _
System.Object, ByVal e As System.EventArgs) _
Handles Button2.Click
```

```
        ' Khai báo và khởi tạo đối tượng clsDatabase
        Dim cls As New clsDatabase
        cls.Connection=gsCon
        ' Gọi phương thức doSQL
        strError = cls.doSQL(TextBox3.Text, i)
        TextBox2.Text = i.ToString
    End Sub
```

Khi thực thi chương trình, kết quả trả về sẽ trình bày tương tự như hình 16-4.

Chú ý: Trong lớp *clsDatabase*, chúng ta cài đặt *Overload* hai phương thức cùng tên *doSQL* nhưng khác nhau về số lượng tham số như ví dụ 16-8-1.

Nếu bạn có nhu cầu lấy ra số mẩu tin thực thi thì bạn sử dụng phương thức có hai tham số.

Ví dụ 16-8-1: Phương thức doSQL (Ovreloading)

```
    Function doSQL(ByVal strSQL As String) As String
        Dim strError As String = ""
        Dim myCon As New SqlConnection(gsCon)
        Try
            myCon.Open()
            Dim myCom As New SqlCommand(strSQL, myCon)
            myCom.ExecuteNonQuery()
        Catch ex As Exception
            strError = ex.Message
        Finally
            myCon.Close()
            myCon.Dispose()
        End Try
        Return strError
    End Function
```

Để tìm hiểu thêm về phương thức này, bạn thực hành các ví dụ sau:

1. Tạo ứng dụng *Windows Forms*, cho phép người sử dụng nhập tên cơ sở dữ liệu và tạo cơ sở dữ liệu đó trong *SQL Server*.
2. Thiết kế *Form*, cho phép người sử dụng chọn cơ sở dữ liệu đang tồn tại trong *SQL Server* rồi tạo mới một *Table* và thêm dữ liệu vào *Table* đó.

3. Bằng cách sử dụng điều khiển *DataGrid*, liệt kê danh sách các mẩu tin trong bảng vừa tạo trong câu 2, khi người sử dụng chọn những mẩu tin cần xóa trên các *checkbox*, lập tức các mẩu tin đó sẽ bị xóa và in ra số mẩu tin đã xóa.

5.4.2. Phương thức ExecuteScalar

Trong trường hợp phát biểu *SQL* dạng *Select*, phương thức sẽ trả về là một giá trị kiểu *object* tương ứng với giá trị của cột và hàng đầu tiên. Ví dụ, sử dụng phương thức *ExecuteScalar* bằng khai báo như ví dụ 16-9.

Ví dụ 16-9: Phương thức ExecuteScalar

```
Function doSQL(ByVal strSQL As String, _
ByRef Result As object) As String
   ' Khai báo biến chuỗi lỗi
   Dim strError As String = ""
   ' Khai báo và khởi tạo đối tượng kết nối cơ sở dữ liệu
   Dim myCon As New SqlConnection(gsCon)
   Try
      myCon.Open()
      Dim myCom As SqlCommand
      ' Khởi tạo đối tượng SqlCommand
      myCom = New SqlCommand(strSQL, myCon)
      ' Gọi phương thức ExecuteScalar
      Result = myCom.ExecuteScalar()
   Catch ex As Exception
      strError = ex.Message
   Finally
      myCon.Close()
      myCon.Dispose()
   End Try
   Return strError
End Function
```

Do kiểu dữ liệu trả về của phương thức *ExecuteScalar* là *object*, cho nên bạn có thể khai báo tham biến truyền vào phương thức có kiểu *object*, bằng cách chuyển đổi kiểu dữ liệu tương ứng cho tham biến *result* mỗi khi gọi phương thức.

Ví dụ, trong trường hợp muốn lấy tổng số lượng khách hàng đang có trong bảng *Customers*, bạn chỉ cần khai báo phát biểu *SQL* dạng *Select* với phép toán *Count* như sau:

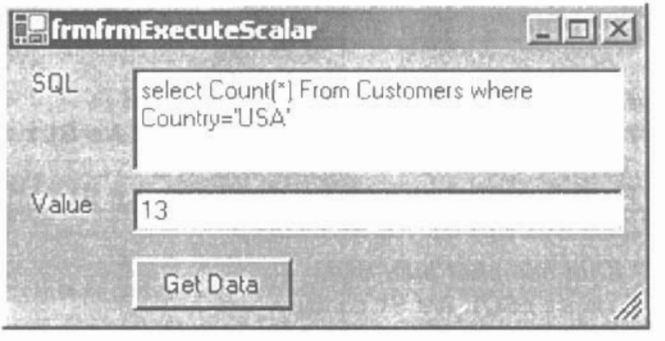
```
Select Count(*) from Customers
```

Sau đó, trong ứng dụng *Visual Basic.NET*, bạn gọi phương thức có tên *getValue* nhận tham trị là phát biểu *SQL* dạng *Select* trả về một giá trị là tổng số mẩu tin của bảng *Customers* khai báo ở trên và tham biến là biến *object* như ví dụ 16-10.

Ví dụ 16-10: Gọi phương thức getValue

```
Dim strError As String
Private Sub Button1_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles Button1.Click
    ' Khai báo và khởi tạo đối tượng clsDatabase
    Dim cls As New clsDatabase(gsCon)
    Dim obj As Object
    ' gọi phương thức getValue
    strError = cls.getValue(TextBox1.Text, obj)
    TextBox2.Text = obj.ToString
End Sub
```

Khi thực thi chương trình, kết quả thực thi trả về là tổng số khách hàng có quốc tịch là *USA* sẽ là 13 như hình 16-5.



Hình 16-5: *Sử dụng phương thức ExecuteScalar*

Để làm việc với phương thức *ExecuteScalar*, bạn có thể thực hành các ví dụ sau:

1. Thiết kế *Form*, cho phép người sử dụng chọn một sản phẩm trong điều khiển *ComboBox*, sau đó in ra số lượng đang còn trong kho của sản phẩm đó.
2. Thiết kế thủ tục nội tại để thêm mẩu tin vào bảng có cột số tự động, sau đó khai báo và sử dụng thủ tục này với mục đích mỗi khi thêm một mẩu tin bạn lấy ra được số tự động đó. Sau đó, thiết kế *Form* cho phép người sử dụng thêm mẩu tin và trình bày số tự động lấy được ra màn hình.
3. Thiết kế *Form*, liệt kê danh sách các *Country*, mỗi khi người sử dụng chọn một *Country* bạn lấy ra khách hàng có số tiền đặt hàng lớn nhất thuộc *Country* đó, đồng thời trình bày danh sách các đơn đặt hàng của khách hàng đó trên điều khiển *DataGrid*.

5.4.3. Phương thức ExecuteReader

Khi bạn muốn truy cập dữ liệu với số lượng nhỏ và không cần xử lý dữ liệu hay điều hướng trên mẩu tin, bạn có thể sử dụng đối tượng *SqlDataReader* hay *OleDbDataReader*.

Chẳng hạn, khai báo phương thức có tên *getValue* như ví dụ 16-11 nhận hai tham trị là *Username*, *Password* và tham biến là mảng dạng chuỗi. Sau đó, chương trình kiểm tra người sử dụng này có tồn tại trong bảng *tblUsers* (cơ sở dữ liệu *Northwind*) hay không. Nếu tồn tại, chương trình sẽ gán giá trị của các cột *UserID*, *FullName*, *Email* vào phần tử mảng tương ứng.

Ví dụ 16-11: Gọi phương thức getValue

```
' Truyền hai tham số là username và password
Function getValue(ByVal Username As String, _
     ByVal Password As String , _
     ByRef myValues() As String) As String
     Dim StrError As String =""
     ' Khai báo phát biểu SQL
     Dim strSQL As String
     strSQL= "Select Password, " & _
       "UserID,FullName,Email From tblUsers " & _
       "Where UserName='" + Username + "'"
     ' Khai báo và khởi tạo đối tượng SqlConnection
     Dim Conn As SqlConnection = _
       New SqlConnection(gsCon)
```

```
        ' Khai báo và khởi tạo đối tượng SqlCommand
        Dim myCom As SqlCommand = _
          New SqlCommand(strSQL, Conn)
      Try
        ' Mở kết nối cơ sở dữ liệu SQL Server
        Conn.Open()
        ' Khai báo đối tượng SqlDataReader, sau đó sử dụng phương
        ' thức ExecuteReader của SqlCommand để điền dữ liệu từ dữ
        ' liệu nguồn vào đối tượng SqlDataReader
        Dim myRD As SqlDataReader

        myRD= myCom.ExecuteReader()
        ' Nếu tồn tại mẩu tin
        If myRD.Read() Then
          ' So sánh password trong cơ sở dữ liệu và password do
          ' người sử dụng nhập thông qua tham số Password
         If Password.Equals(myRD.GetString(0))
    Then
            ' Gán fullname, email và userId vào mảng
            myValues(1) = _
              myRD.GetInt32(1).ToString
            myValues(2) = myRD.GetString(2)
            myValues(3) = myRD.GetString(3)
        End If
      Catch ex As String
        strError=ex.Mesage
      Finally
        myRD.Close()
        Conn.Close()
      End Try
      Return strError
    End Function
```

Sau đó, thêm *Form* vào *Project* và đặt tên *frmExecuteReader*, thiết kế một số điều khiển, sau đó khai báo để gọi phương thức *getValue* nếu người sử dụng cung cấp đầy đủ *Username* và *Password* như ví dụ 16-12.

Ví dụ 16-12: Kiểm tra người sử dụng

```
Private Sub Button1_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles Button1.Click
```

```
        'Yêu cầu người sử dụng nhập username
        If txtUser.Text = "" Then
          MsgBox("Please enter your Username")
          txtUser.Focus()
          Exit Sub
        End If
        'Yêu cầu người sử dụng nhập password
        If txtPwd.Text = "" Then
          MsgBox("Please enter your Password")
          txtPwd.Focus()
          Exit Sub
        End If
        'Gọi phương thức kiểm tra username và password
        Dim myValues(3) As String
        Dim cls As clsDatabase(gsCon)
        Dim strError As String = _
            clsGetValue(txtUser.Text, _
          txtPwd.Text, myValues)
        'Nếu giá trị phần tử đầu tiên khác rỗng, tức là user hợp lệ
        If Not myValues(0) Is Nothing Then
          'In ra UserID
          TextBox1.Text = myValues(0) + vbNewLine
          'In ra FullName
          TextBox1.Text += myValues(1) + vbNewLine
          'In ra Email
          TextBox1.Text += myValues(2) + vbNewLine
        End If
    End Sub
```

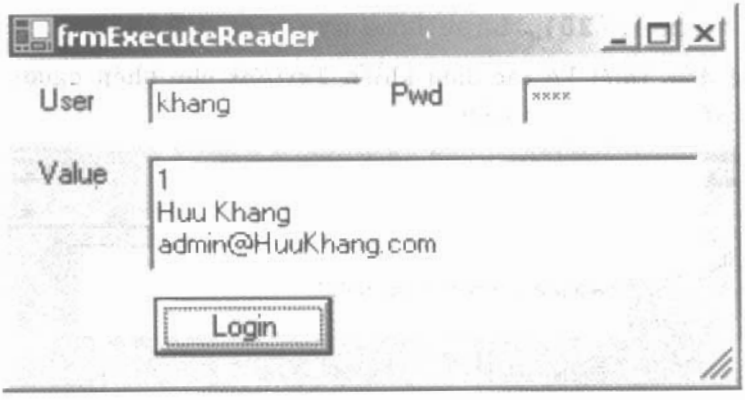
Lưu ý, cấu trúc của bảng *tblUsers* được định nghĩa bằng phát biểu *CREATE TABLE* như sau:

```
Create Table tblUsers
(
    UserID int identity(1,1) primary key,
    UserName varchar(20) not null unique,
    Password varchar(10) null,
    FullName nvarchar(20) null ,
    Email varchar(20) null ,
    JoinDate smalldatetime default getdate()
)
```

Bạn có thể thêm mẩu tin vào bảng *tblUsers* bằng cú pháp của phát biểu *Insert* như sau:

```
insert into tblUsers(UserName,
Password,FullName,Email)
values('Khang','1234','Huu Khang',
'admin@HuuKhang.com')
```

Khi thực thi chương trình, bằng cách cung cấp *username* và *password* thì kết quả trả về bao gồm mã, tên và *email* như hình 16-6.



Hình 16-6: *Kết quả đăng nhập*

Để tìm hiểu thêm về phương thức *ExecuteReader*, bạn có thể thực hành các ví dụ sau:

1. Viết ứng dụng *Console*, yêu cầu người sử dụng nhập vào tên cơ sở dữ liệu, bạn liệt kê tên của các đối tượng cơ sở dữ liệu cùng với ngày tạo ra nó.
2. Thiết kế *Form*, cho phép liệt kê danh sách *Country* trong bảng *Customers* vào điều khiển *ComboBox*.

6. LÀM VIỆC VỚI ĐỐI TƯỢNG THAM SỐ

Trong trình bày của phần khai báo và sử dụng phương thức *ExecuteNonQuery* thuộc đối tượng *SqlCommand*, bạn có thể khai báo một phát biểu *SQL* dạng hành động bằng cách kết nối những giá trị từ *Form* nhập liệu, sau đó sử dụng đối tượng *SqlCommand* để thực thi chúng.

Khi người sử dụng nhập giá trị vào điều khiển *TextBox* như hình 16-7 rồi nhấn nút *"Add Data without Parameter"*, lỗi sẽ phát sinh nếu

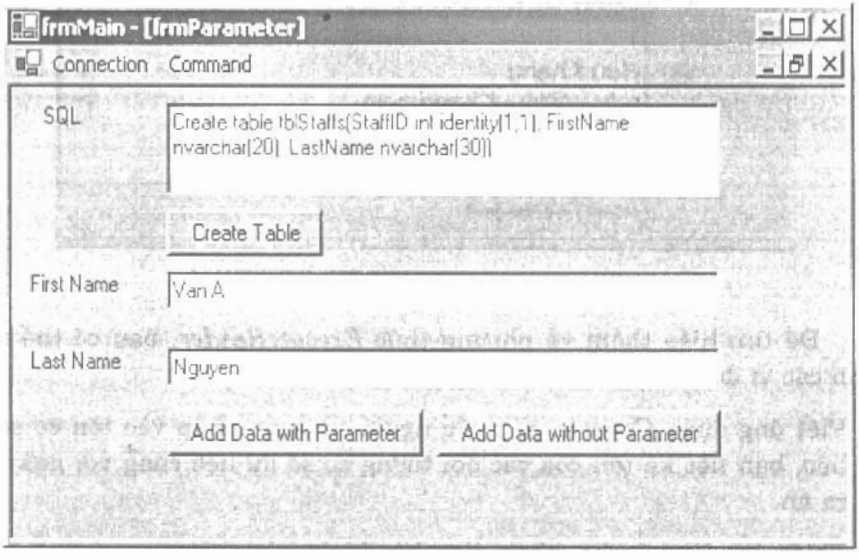
phát biểu *SQL* không xử lý dấu nháy đơn (chúng ta sẽ tìm hiểu cách sử dụng *Parameter* trong phần cuối của chuyên đề này).

Chẳng hạn, bạn tạo mới bảng dữ liệu có tên *tblStaffs* bằng cách nhấn nút *Create Table* trên *Form* có tên *frmParameter* hoặc trong cơ sở dữ liệu *SQL Server*.

Ví dụ 16-13: Tạo bảng dữ liệu

```
Create table tblStaffs
(StaffID int identity(1,1), FirstName
nvarchar(20), LastName nvarchar(30))
```

Kế đến, thiết kế các điều khiển *TextBox* cho phép người sử dụng nhập họ và tên của nhân viên.



Hình 16-7: *Nhập dữ liệu*

Bằng cách gọi phương thức *doSQL* trong lớp *clsDatabase* ứng với phát biểu *SQL*, bạn khai báo trong biến cố *Click* của nút *Add Data without Parameter* như ví dụ 16-14.

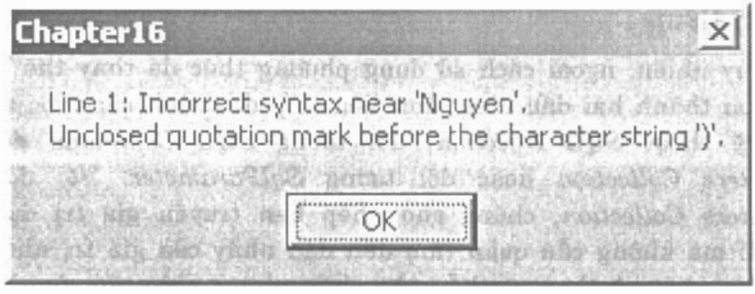
Ví dụ 16-14: Thực thi phát biểu SQL

```
Private Sub Button1_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles Button1.Click
    Dim cls As New clsDatabase
```

```
        cls.Connection = gsCon
        Dim strSQL As String
        strSQL = "insert into tblStaff values('"
        strSQL += txtFirstName.Text + "','"
        strSQL += txtLastName.Text + "')"
        strError = cls.doSQL(strSQL)
        If Not strError.Equals("") Then
           MsgBox(strError)
        End If
    End Sub
```

Lỗi sẽ phát sinh tương tự như hình 16-8 nếu bạn nhập chuỗi là Van A'. Điều này có nghĩa là tên của nhân viên thật sự có dấu nháy ('). Đây chính là một trong những nguyên nhân cho phép những người truy cập bất hợp pháp có thể mượn lỗ hổng này để tấn công cơ sở dữ liệu của bạn.

Kỹ thuật dùng tấn công bằng lỗ hổng này được gọi là *SQL Injection* (bạn có thể tham khảo kỹ thuật này trong cuốn sách cùng tác giả có tên *Quản trị SQL Server 2000* do nhà sách Minh Khai phát hành).



Hình 16-8: *Lỗi phát sinh*

Để tránh trường hợp này, bạn cần sử dụng phương thức *Replace* của ngôn ngữ lập trình *Visual Basic.NET* để thay thế một dấu nháy thành hai dấu nháy liên tiếp.

```
myString =myString.Replace("'","''")
```

Mục đích của vấn đề này là tạo ra một phát biểu *SQL* có cú pháp hợp lệ mà cơ sở dữ liệu có thể hiểu và cho phép thực thi.

Để kiểm tra điều này, bạn khai báo lại phát biểu *SQL* trong ví dụ 16-14 thành ví dụ 16-14-1.

Ví dụ 16-14-1: Thực thi phát biểu SQL

```
Private Sub Button1_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles Button1.Click
    Dim cls As New clsDatabase
    cls.Connection = gsCon
    Dim strSQL As String
    strSQL = "insert into tblStaff values('"
    strSQL += _
     txtFirstName.Text.Replace("'","''")
    strSQL += "','" + _
     txtLastName.Text.Replace("'","''")
    strSQL += "')"
    strError = cls.doSQL(strSQL)
   If Not strError.Equals("") Then
      MsgBox(strError)
    End If
End Sub
```

Tuy nhiên, ngoài cách sử dụng phương thức để thay thế một dấu nháy đơn thành hai dấu nháy đơn liên tiếp để tránh các cuộc tấn công bằng kỹ thuật *SQL Injection*, đối tượng *SQL Command* cung cấp *Parameters Collection* hoặc đối tượng *SqlParameter*. Với đối tượng *Parameters Collection*, chúng cho phép bạn truyền giá trị dưới dạng tham số mà không cần quan tâm đến dấu nháy của giá trị nhập, đồng thời bằng cách này bạn có thể ngăn chặn những cuộc tấn công cơ sở dữ liệu bằng kỹ thuật *SQL Injection*.

6.1. Parameters Collection

Parameters Collection là một phần của đối tượng *SqlCommand*, cho phép bạn truyền giá trị có kiểu là *object* hay kiểu dữ liệu cùng kích thước của chúng tương ứng với tham số.

Ví dụ, bạn thêm *Form* rồi đặt tên *frmParametersCollection* và khai báo phương thức có tên *doSQL* (*Overload*) trong lớp *clsDatabase* nhận mảng giá trị chuỗi, sau đó sử dụng phương thức *Add* của *Parameters Collection* để thêm giá trị cho từng tham số trình bày như trong ví dụ 16-15.

Ví dụ 16-15: Dùng tham số

```
Function doSQL(ByVal strParameter As String())
As String
    Dim strError As String = ""
    Dim myCon As New SqlConnection(gsCon)
    Try
      myCon.Open()
      ' Khởi tạo đối tượng SqlCommand
      Dim myCom As New SqlCommand
      ' Gán thuộc tính Connection
      myCom.Connection = myCon
      ' Định nghĩa phát biểu SQL với hai tham số
      myCom.CommandType = CommandType.Text
      myCom.CommandText = _
      "insert into tblStaffs " + _
        "values(@FirstName,@LastName)"
      ' Gán giá trị cho tham số thứ nhất
      myCom.Parameters.Add("@FirstName", _
        strParameter(0))
      ' Gán giá trị cho tham số thứ hai
      myCom.Parameters.Add("@LastName", _
        strParameter(1))
      ' Thực thi phát biểu SQL
      myCom.ExecuteNonQuery()
    Catch ex As Exception
      strError = ex.Message
    Finally
      myCon.Close()
      myCon.Dispose()
    End Try
    Return strError
End Function
```

Lưu ý: Tham số trong chuỗi *SQL* được định nghĩa kèm theo dấu @ ngay trước tên tham số. Nếu trong phát biểu *SQL* có tham số *@FirstName*, trong phương thức *Add* bạn khai báo cùng tham số *@FirstName*.

Tương tự như trên, bạn nhấn nút *"Add Data with Parameter"*, mẩu tin sẽ được thêm vào cơ sở dữ liệu.

Ngoài ra, bạn cũng có thể sử dụng đối tượng *SqlParameter* để định nghĩa từng cặp tên và giá trị, sau đó thêm đối tượng này vào đối tượng *SqlCommand* bằng cách dùng phương thức *Add* của *Parameters Collection*.

6.2. Đối tượng SqlParameter

Tương tự như cách sử dụng *Parameters Collection* thuộc đối tượng *SqlCommand*, bạn có thể sử dụng đối tượng *SqlParameter* để thêm từng cặp tham số và giá trị vào *Parameters Collection* thay vì thêm từng phần.

Chẳng hạn, khai báo phương thức *Overload* có tên *doSQL* nhận hay tham số ứng với hai mảng chứa danh sách tên tham số và giá trị tương ứng và sử dụng đối tượng *SqlParameter* như ví dụ 16-16.

Ví dụ 16-16: Sử dụng đối tượng SqlParameter

```
Function doSQL(ByVal strPara() As String, ByVal
strValue() As String) As String
    Dim strError As String = ""
    ' Khai báo và khởi tạo đối tượng SqlConnection
    Dim myCon As New SqlConnection(gsCon)
    Try
        ' Mở kết nối cơ sở dữ liệu
        myCon.Open()
        Dim myCom As New SqlCommand
        myCom.Connection = myCon
        ' Chọn định dạng chuỗi SQL
        myCom.CommandType = CommandType.Text
        ' Khai báo đối tượng SqlParameter
        Dim myPara As SqlParameter
        Dim strSQL As String
        ' Khai báo phát biểu SQL
        strSQL = "insert into tblStaffs values("
        ' Duyệt trên số tham số
        For i As Integer = 0 To strPara.Length - 1
            ' Khai báo phát biểu SQL
            strSQL += strPara(i) + ","
            ' Khởi tạo đối tượng SqlParameter
            myPara = New SqlParameter(strPara(i), _
            strValue(i))
```

```
            ' Thêm đối tượng SqlParameter vào đối tượng
            ' SqlCommand
            myCom.Parameters.Add(myPara)
        Next
        ' Loại bỏ dấu phẩy cuối cùng
        strSQL = strSQL.Substring(0, _
        strSQL.Length - 1) + ")"
        ' Thực thi phát biểu SQL
        myCom.CommandText = strSQL
        myCom.ExecuteNonQuery()
    Catch ex As Exception
        strError = ex.Message
    Finally
        myCon.Close()
        myCon.Dispose()
    End Try
    Return strError
End Function
```

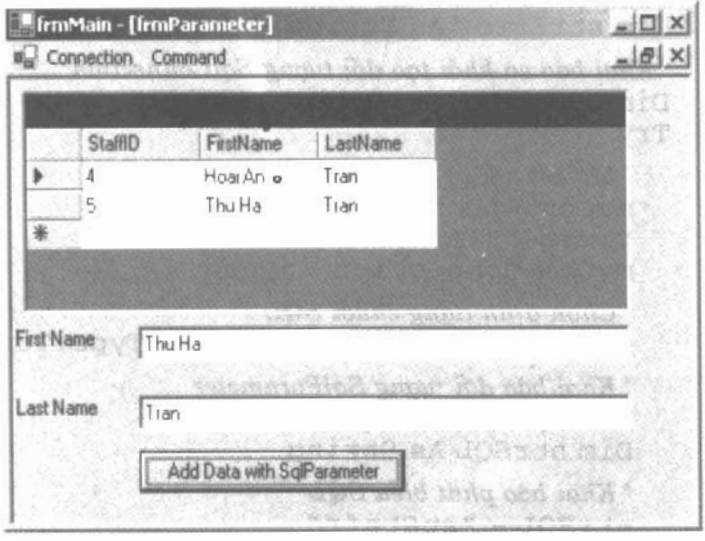
Chú ý, bạn có thể không khai báo tên tham số trong chuỗi *SQL* của ví dụ 16-16 như ví dụ 16-17, nếu sử dụng thủ tục nội tại.

Ví dụ 16-17: Sử dụng thủ tục nội tại

```
Function doSQL(ByVal strPara() As String, ByVal
strValue() As String) As String
    Dim strError As String = ""
    ' Khai báo và khởi tạo đối tượng SqlConnection
    Dim myCon As New SqlConnection(gsCon)
    Try
        ' Mở kết nối cơ sở dữ liệu
        myCon.Open()
        Dim myCom As New SqlCommand
        myCom.Connection = myCon
        ' Chọn định dạng chuỗi SQL
        myCom.CommandType = CommandType.Text
        ' Khai báo đối tượng SqlParameter
        Dim myPara As SqlParameter
        Dim strSQL As String
        ' Khai báo phát biểu SQL
        strSQL = "spStaff"
        ' Duyệt trên số tham số
```

```
        For i As Integer = 0 To strPara.Length - 1
            ' Khởi tạo đối tượng SqlParameter
            myPara = New SqlParameter(strPara(i), _
            strValue(i))
            ' Thêm đối tượng SqlParameter vào đối tượng
            ' SqlCommand
            myCom.Parameters.Add(myPara)
        Next
        ' Thực thi phát biểu SQL
        myCom.CommandText = strSQL
        myCom.ExecuteNonQuery()
    Catch ex As Exception
        strError = ex.Message
    Finally
        myCon.Close()
        myCon.Dispose()
    End Try
    Return strError
End Function
```

Bằng cách thiết kế *frmParameter* tương tự như hình 16-9, cho phép người sử dụng nhập mới họ và tên của nhân viên, sau khi nhấn nút "*Add Data with SqlParameter*", lập tức thông tin nhân viên sẽ xuất hiện trên điều khiển *DataGrid*.



Hình 16-9: *Thêm dữ liệu bằng đối tượng SqlParameter*

Để làm điều này, bạn khai báo trong biến cố *Click* của nút "*Add Data with SqlParameter*" như ví dụ 16-18.

Ví dụ 16-18: Sử dụng đối tượng SqlParameter

```
Private Sub btnSave_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles btnSave.Click
    ' Khai báo và khởi tạo đối tượng clsDatabase
    Dim cls As New clsDatabase
    Dim strError As String
    cls.Connection = gsCon
    ' Khai báo mảng tên tham số
    Dim myPara() As String = _
      {"@FirstName", "@LastName"}
    ' Khai báo mảng chứa giá trị
    Dim myValue(2) As String
    myValue(0) = txtFirstName.Text
    myValue(1) = txtLastName.Text
    ' Gọi phương thức doSQL
    strError = cls.doSQL(myPara, myValue)
    If Not strError.Equals("") Then
      MsgBox(strError)
    End If
    ' Khai báo và khởi tạo đối tượng DataTable
    Dim myDT As New DataTable
    strError = cls.getValue(myDT, _
      "select * from tblStaffs")
    ' Trình bày dữ liệu trên điều khiển DataGrid
    DataGrid1.DataSource = myDT
End Sub
```

Lưu ý: Bạn có thể áp dụng cách thực thi phát biểu *SQL* bằng kỹ thuật này với các loại cơ sở dữ liệu khác.

Để tìm hiểu thêm đối tượng *SqlParameter* và phương thức *Add* của *Parameters Collection*, bạn có thể thực hành các ví dụ sau:

1. Xây dựng lớp có phương thức sử dụng đối tượng *SqlParameter* cho phép thêm dữ liệu vào bảng trong cơ sở dữ liệu *SQL Server*.

2. Tạo *Class* và định nghĩa phương thức sử dụng đối tượng *Parameters Collection* cho phép liệt kê dữ liệu trong bảng chỉ định bằng thủ tục nội tại của *SQL Server* có truyền tham số.

7. KẾT LUẬN

Bạn vừa tìm hiểu một số kỹ thuật thực hiện phát biểu *SQL* với giá trị trả về là số mẩu tin thực thi hay giá trị của cột và hàng đầu tiên với cơ sở dữ liệu *SQL Server* và *Access*.

Ngoài ra, bạn cũng tham khảo cách sử dụng tham số truyền vào chuỗi *SQL* của đối tượng *SqlCommand*. Để tìm hiểu thêm các ví dụ và bài tập về đối tượng *SqlParameter*, bạn có thể tham khảo chuyên đề 22 trình bày cách xây dựng lớp làm việc với cơ sở dữ liệu.

Trong chuyên đề kế tiếp, chúng ta sẽ tìm hiểu cách làm việc với các thuộc tính và phương thức của đối tượng *SqlDataReader*.

Chuyên đề 17:
LÀM VIỆC VỚI ĐỐI TƯỢNG DATAREADER

Tóm tắt chuyên đề 17

Trong chuyên đề trước, bạn đã tham khảo cách sử dụng đối tượng SqlConnection, SqlCommand, cách thực thi phát biểu SQL, thủ tục nội tại cùng với các phương thức, thuộc tính của đối tượng SqlParameter, SqlCommand .

Trong chuyên đề này, chúng ta sẽ tiếp tục tìm hiểu cách khai báo sử dụng đối tượng SqlDataReader cùng với các trình điều khiển cơ sở dữ liệu bằng ngôn ngữ lập trình Visual Basic.NET.

Các vấn đề chính sẽ được đề cập:

- ✓ *Đối tượng SqlDataReader.*
- ✓ *Đọc nhiều tập mẩu tin.*
- ✓ *Đối tượng SqlDataReader và các điều khiển.*

1. ĐỐI TƯỢNG SQLDATAREADER

Trong phần trình bày phương thức *ExecuteReader* của đối tượng *SqlCommand* thuộc chuyên đề trước, bạn đã biết cách khai báo để nắm giữ tập dữ liệu từ cơ sở dữ liệu nguồn do phương thức *ExecuteReader* trả về cho đối tượng *SqlDataReader*.

Như vậy, đối tượng *SqlDataReader* là bộ đọc dùng để đọc dữ liệu trực tiếp từ cơ sở dữ liệu nguồn thông qua phương thức *ExecuteReader* của đối tượng *SqlCommand*.

Tuy nhiên, bạn chỉ nên sử dụng đối tượng *SqlDataReader* khi tập dữ liệu có số lượng mẩu tin vừa phải và không có nhu cầu xử lý trên tập dữ liệu đó.

1.1. Thuộc tính

SqlDataReader cùng cấp các thuộc tính như: *IsClosed*, *FieldCount*, *HasRows* và *Item*.

- *IsClosed*: Thuộc tính trả về giá trị luận lý chỉ định đối tượng *SqlDataReader* đã đóng hay chưa.
- *FieldCount*: Thuộc tính này trả về số nguyên ứng với số cột dữ liệu có trong tập dữ liệu mà đối tượng *SqlDataReader* đang nắm giữ.
- *HasRows*: Trả về giá trị *True* và *False* tương ứng với mẩu tin tồn tại trong đối tượng *SqlDataReader*.
- *Item(i)*: Cho phép bạn lấy giá trị của cột theo chỉ mục.
- *Item(columnname)*: Cho phép bạn lấy giá trị của cột theo tên cột.

Chẳng hạn, bạn khai báo đối tượng *SqlDataReader* để liệt kê danh sách các cột dữ liệu tương ứng với phát biểu *Select* dựa vào ba thuộc tính *FieldCount*, *HasRows* và *Item* như ví dụ 17-1:

Ví dụ 17-1: Liệt kê tập dữ liệu

```
Function getValue() As String
    ' Khai báo chuỗi lỗi trả về
    Dim strError As String = ""
    ' Khai báo và khởi tạo đối tượng Connection
    Dim myCon As New SqlConnection(gsCon)
    ' Khai báo chuỗi SQL
    Dim strSQL As String
    strSQL = "select EmployeeID, FirstName,"
    strSQL += "LastName  from Employees"
    Try
      myCon.Open()
      Dim myCom As SqlCommand
      myCom = New SqlCommand(strSQL, myCon)
      Dim myRD As SqlDataReader
      myRD = myCom.ExecuteReader
      ' Nếu tồn tại mẩu tin
      If myRD.HasRows Then
        ' Đọc từng mẩu tin
        While myRD.Read()
          ' Đọc từng cột
          For i As Integer = 0 To myRD.FieldCount - 1
          ' In giá trị ra màn hình
            Console.Write(" {0}", _
            myRD.Item(i).ToString)
```

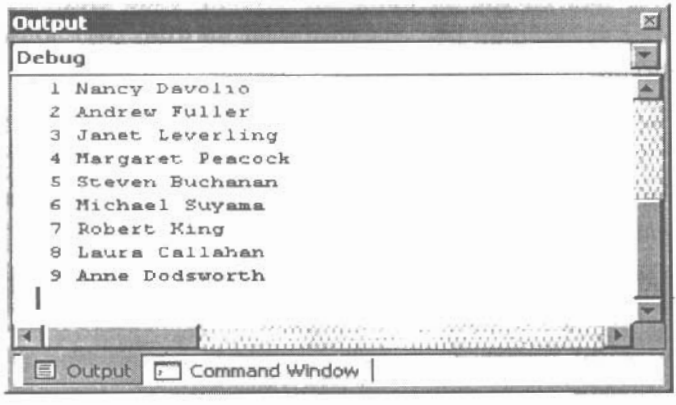
```
            Next
            Console.WriteLine()
          End While
        End If
      Catch ex As Exception
        strError = ex.Message
      Finally
        myCon.Close()
        myCon.Dispose()
      End Try
      Return strError
    End Function
```

Bằng cách thiết kế *Form* có tên *frmHasRow* rồi khai báo gọi phương thức *getValue* của lớp *clsDatabase* từ ví dụ 17-1 thành ví dụ 17-2:

Ví dụ 17-2: Gọi phương thức getValue

```
Private Sub Button1_Click(ByVal sender As _
System.Object, ByVal e As System.EventArgs) _
Handles Button1.Click
  Dim cls As New clsDatabase(gsCon)
  cls.getValue()
End Sub
```

Khi thực thi chương trình, kết quả trả về là danh sách nhân viên trong bảng *Employees* như hình 17-1.



Hình 17-1: *Kiểm tra số mẩu tin*

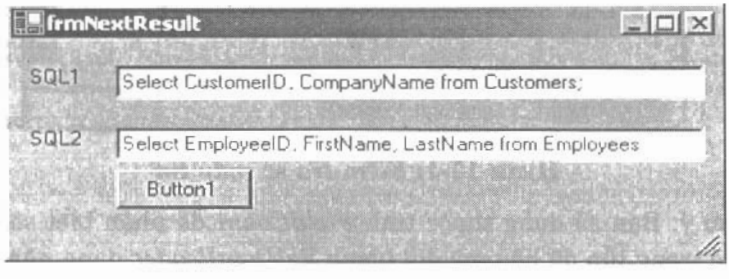
Lưu ý: Bạn sử dụng thuộc tính *FieldCount* để nhận biết số lượng cột dữ liệu có trong tập dữ liệu mà đối tượng *SqlDataReader* đang nắm giữ.

1.2. Phương thức

Ngoài phương thức *Read* dùng để đọc từng mẩu tin, đối tượng *SqlDataReader* còn cung cấp một số phương thức khác để đọc dữ liệu từng cột tương ứng với kiểu dữ liệu của chúng, ví dụ như các phương thức.

- *Close*: Phương thức dùng để đóng đối tượng *SqlDataReader*.
- *GetString(i)*: Phương thức trả về giá trị dạng chuỗi của cột dữ liệu chỉ định.
- *GetInt32(i)*: Phương thức trả về giá trị số nguyên.
- *GetName(i)*: Phương thức trả về giá trị dạng chuỗi là tên cột của cột thứ i.
- *GetOrdinal(name)*: Phương thức trả về giá trị là thứ tự của cột với tên chỉ định.
- *Item(i / name)*: Bạn có thể sử dụng thuộc tính này trả về giá trị kiểu *object* của cột thứ i hoặc tên cột.
- *GetValues(mảng đối tượng)*: Cho phép bạn truyền vào biến mảng kiểu *object*, sau đó phương thức này sẽ điền giá trị của tất cả các cột vào mảng ứng với mẩu tin hiện hành (tương tự như thuộc tính *ItemArray* của đối tượng *DataRow*).
- Ngoài ra, bạn có thể sử dụng các phương thức khác để lấy giá trị tương ứng với kiểu dữ liệu của chính cột đó như: *GetBoolean*, *GetByte*, *GetDateTime*, ...

Tuy nhiên, trong trường hợp có hai hoặc nhiều tập dữ liệu như hình 17-2, thay vì sử dụng hai đối tượng *SqlDataReader* thì bạn sử dụng phương thức *NextResult* để truy cập đến tập dữ liệu kế tiếp mà đối tượng *SqlDataReader* đang nắm giữ.



Hình 17-2: *Hai phát biểu SQL*

Chú ý, hai phát biểu *Select* được khai báo cách nhau dấu chấm phẩy (;) và có thể có số lượng cột dữ liệu của mỗi tập khác nhau.

Chẳng hạn, chúng ta có hai tập dữ liệu tương ứng với hai phát biểu *Select*, bằng cách sử dụng phương thức *NextResult* của đối tượng *SqlDataReader*, bạn có thể khai báo để liệt kê danh sách mẩu tin như ví dụ 17-3:

Ví dụ 17-3: Sử dụng phương thức NextResult

```
Function getValue(ByVal SQL1 As String, _
ByVal SQL2 As String) As String
   'Khai báo chuỗi lỗi trả về
   Dim strError As String = ""
   'Khai báo và khởi tạo đối tượng Connection
   Dim myCon As New SqlConnection(gsCon)
   Try
      myCon.Open()
      Dim myCom As SqlCommand
      myCom = New SqlCommand(SQL1+SQL2, myCon)
      Dim myRD As SqlDataReader
      myRD = myCom.ExecuteReader
      'Đọc từng mẩu tin của tập thứ nhất
        While myRD.Read()
          'Đọc từng cột
          For i As Integer = 0 To myRD.FieldCount - 1
          'In giá trị ra màn hình
            Console.Write(" {0}", _
            myRD.Item(i).ToString)
          Next
          Console.WriteLine()
       End While
      'Nhảy đến tập mẩu tin thứ hai
      myRD.NextResult
      'Đọc từng mẩu tin của tập thứ hai
        While myRD.Read()
          'Đọc từng cột
          For i As Integer = 0 To myRD.FieldCount - 1
          'In giá trị ra màn hình
            Console.Write(" {0}", _
```

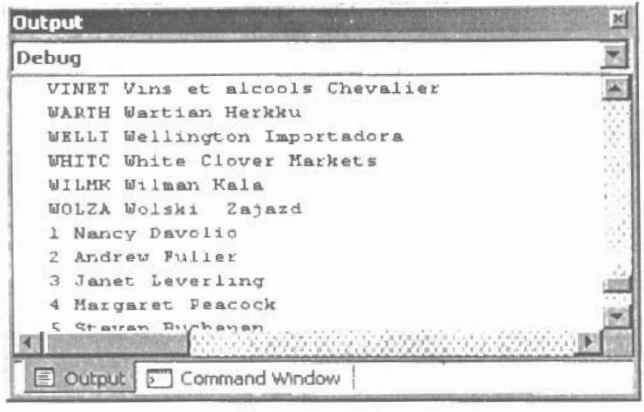
```
            myRD.Item(i).ToString)
          Next
          Console.WriteLine()
        End While
      Catch ex As Exception
        strError = ex.Message
      Finally
        myCon.Close()
        myCon.Dispose()
      End Try
      Return strError
    End Function
```

Kế đến, thêm *Form* vào *Project* và đặt tên *frmNextResult*, bằng cách kết hợp hai phát biểu *Select* tương ứng với danh sách khách hàng và nhân viên, rồi gọi phương thức *getValue* như ví dụ 17-4:

Ví dụ 17-4: Gọi phương thức getValue

```
Private Sub Button1_Click(ByVal sender As _
System.Object, ByVal e As System.EventArgs) _
Handles Button1.Click
  Dim cls As New clsDatabase(gsCon)
  cls.getValue(TextBox1.Text, TextBox2.Text)
End Sub
```

Kết quả trả về là hai danh sách khách hàng và nhân viên tương ứng với hai phát biểu *Select* như hình 17-3.



Hình 17-3: *Dùng phương thức NextResult*

Để tìm hiểu thêm đối tượng *SqlDataReader*, bạn có thể thực hành các ví dụ sau:

1. Khai báo phương thức, nhận tham số là phát biểu *Select*, trả về một đối tượng bao gồm giá trị của các cột của hàng đầu tiên, sau đó sử dụng phương thức này từ phương thức *Main* của chương trình.
2. Viết ứng dụng *Console*, liệt kê mã, tên, địa chỉ của danh sách khách hàng và nhà cung cấp trong hai bảng tương ứng có tên *Customers*, *Suppliers* của cơ sở dữ liệu *Northwind*.
3. Định nghĩa phương thức nhận tham số là phát biểu *Select*, trả về một mảng một chiều kiểu đối tượng chứa giá trị của các cột của 10 hàng đầu tiên, sau đó sử dụng phương thức này từ phương thức *Main* của chương trình.

2. ĐỐI TƯỢNG SQLDATAREADER VÀ ĐIỀU KHIỂN

Như trình bày ở trên thì đối tượng *SqlDataReader* là bộ đọc dữ liệu, đối tượng này sẽ không tồn tại nếu không kết nối cơ sở dữ liệu.

Chính vì lý do đó, khi sử dụng đối tượng này để lấy dữ liệu thì bạn chỉ có thể thực hiện tại thời điểm mà kết nối cơ sở dữ liệu đang còn hiệu lực.

Chẳng hạn, khi muốn trình bày danh sách mã và tên khách hàng vào điều khiển *ComboBox*, bạn không thể khai báo một phương thức trả về đối tượng *SqlDataReader*, thay vào đó bạn sử dụng mảng dữ liệu, rồi sau đó đọc dữ liệu từ mảng này vào điều khiển.

Bởi vì nếu bạn không sử dụng hình thức này thì không thể triển khai phương thức có sử dụng đối tượng *SqlDataReader* tại tầng giữa (*Middle tier*) mà chỉ đọc trực tiếp từ tầng giao tiếp (*Presentation*) đến tầng cơ sở dữ liệu (*Data tier*).

Ví dụ, thêm *Form* vào *Project* và đặt tên *frmComboBox*, kế đến bạn khai báo trong lớp *clsMyDatabase* một phương thức, nhận tham biến là điều khiển *ComboBox*, sau đó điền dữ liệu vào điều khiển *ComboBox* như ví dụ 17-5.

Ví dụ 17-5: Điền dữ liệu vào điều khiển ComboBox

```
Function SetControl(ByRef myControl As
ComboBox, ByVal strSQL As String) As String
    Dim strError As String = ""
    ' Khai báo đối tượng SqlDataReader
```

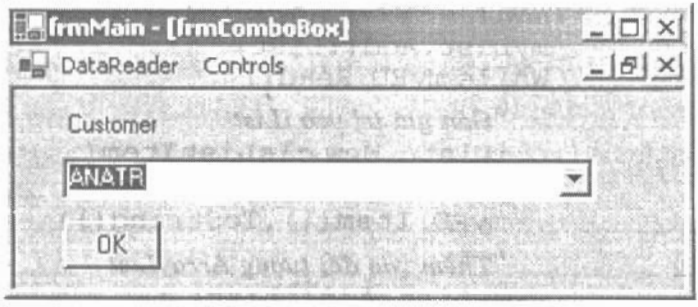
```
        Dim myRD As SqlDataReader
        Try
            'Gọi phương thức mở kết nối cơ sở dữ liệu
            OpenConnection()
            Dim myCom As SqlCommand
            'Khởi tạo đối tượng SqlCommand
            myCom = New SqlCommand(strSQL, myCon)
            'Điền dữ liệu vào đối tượng SqlDataReader
            myRD = myCom.ExecuteReader
            If myRD.HasRows Then
                'Duyệt qua các mẩu tin
                While myRD.Read()
                    'Thêm vào điều khiển
                    myControl.Items.Add( _
                    myRD.Item(0).ToString())
                End While
            End If
            myRD.Close()
        Catch ex As Exception
            strError = ex.Message
        Finally
            CloseConnection()
        End Try
        Return strError
    End Function
```

Sau đó, khai báo đoạn chương trình trong biến cố *Click* của nút *OK* để gọi phương thức *SetControl* từ *Form* như ví dụ 17-6.

Ví dụ 17-6: Điền dữ liệu vào điều khiển ComboBox

```
Private Sub Button1_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles Button1.Click
    ComboBox1.Items.Clear()
    Dim cls As New clsMyDatabase(gsCon)
    cls.SetControl(Me.ComboBox1, _
    "select CustomerID from Customers")
End Sub
```

Khi thực thi chương trình, nếu người sử dụng nhấn nút *OK*, lập tức danh sách khách hàng sẽ xuất hiện trên điều khiển *ComboBox* như hình 17-4.



Hình 17-4: *Liệt kê mã khách hàng*

Lưu ý, trong ví dụ trên chúng ta chỉ có thể gán mã khách hàng vào điều khiển *ComboBox*, khi nhấn nút *GET* thì mã khách hàng được xuất ra màn hình như hình 17-5 bằng cách sử dụng thuộc tính *Text*.

Hình 17-5: *Lấy giá trị chọn*

Trong trường hợp muốn thêm mã và tên của khách hàng vào điều khiển, bạn khai báo lại phương thức *SetControl* như ví dụ 17-7.

Ví dụ 17-7: Liệt kê mã và tên khách hàng

```
Function SetControl(ByRef myList As ArrayList,
ByVal strSQL As String) As String
    Dim strError As String = ""
    Dim myRD As SqlDataReader
    Try
      OpenConnection()
      Dim myCom As SqlCommand
      myCom = New SqlCommand(strSQL, myCon)
      myRD = myCom.ExecuteReader
      If myRD.HasRows Then
        'Khai báo và khởi tạo đối tượng clsListItem
          Dim iList As clsListItem
        'Gán giá trị đầu tiên
          iList = New clsListItem("", _
          "---Select ---")
```

```
            ' Thêm vào đối tượng ArrayList
              myList.Add(iList)
              While myRD.Read()
                ' Gán giá trị vào iList
                iList = New clsListItem( _
                myRD.Item(0).ToString(), _
                myRD.Item(1).ToString())
                ' Thêm vào đối tượng ArrayList
                myList.Add(iList)
              End While
          End If
          myRD.Close()
        Catch ex As Exception
          strError = ex.Message
        Finally

          CloseConnection()
        End Try
        Return strError
    End Function
```

Trong đó, lớp *clsListItem* có hai tham số được khai báo trong *Constructor* như ví dụ 17-8.

Ví dụ 17-8: Cấu trúc lớp clsListItem

```
Public Class clsListItem
  Private mValue As String
  Private mName As String
  ' Khai báo nhận giá trị trong Constructor
  Public Sub New(ByVal strValue As String, _
     ByVal strName As String)
    mValue = strValue
    mName = strName
  End Sub
  ' Khai báo thuộc tính nhận giá trị cho biến Name
  Property Name() As String
    Get
      Return mName
    End Get
    Set(ByVal Value As String)
      mName = Value
    End Set
```

```
    End Property
    ' Khai báo thuộc tính nhận giá trị cho biến Value
    Property Value() As String
      Get
        Return mValue
      End Get
      Set(ByVal Value As String)
        mValue = Value
      End Set
    End Property
End Class
```

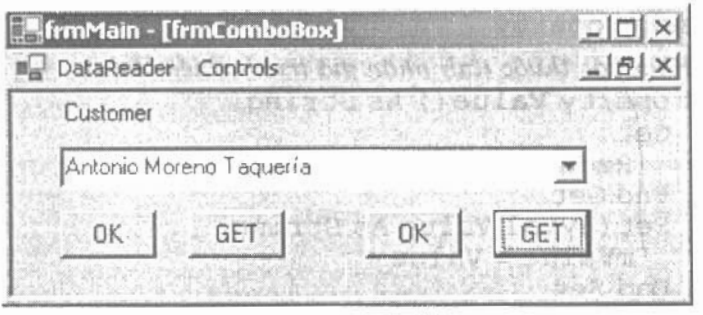
Sau đó, bạn khai báo trong biến cố *Click* của nút *OK* để gọi phương thức *SetControl* như ví dụ 17-9.

Ví dụ 17-9: Gọi phương thức SetControl

```
Private Sub Button4_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles Button4.Click
    ' Xóa dữ liệu cũ
    ComboBox1.Items.Clear()
    ' Khai báo và khởi tạo đối tượng clsMyDatabase
    Dim cls As New clsMyDatabase(gsCon)
    ' Khai báo và khởi tạo đối tượng ArrayList
    Dim myList As New ArrayList
    ' Gọi phương thức SetControl
    cls.SetControl(myList, _
    "select CustomerID,CompanyName from
      Customers")
    ' Gán đối tượng myList vào ComboBox1
    ComboBox1.DataSource = myList
    ' Định nghĩa ValueMember và DisplayMember
    Me.ComboBox1.ValueMember = "Value"
    Me.ComboBox1.DisplayMember = "Name"
    ComboBox1.EndUpdate()
End Sub
```

Khi thực thi chương trình, nếu người sử dụng nhấn nút *OK* thì danh sách tên khách hàng xuất hiện trên điều khiển *ComboBox* như hình 17-6.

Mã khách hàng sẽ được xuất ra mỗi khi người sử dụng nhấn nút *GET* như hình 17-7.



Hình 17-6: *Danh sách tên khách hàng*

Hình 17-7: *Lấy giá trị từ việc chọn tên*

Để tìm hiểu thêm đối tượng *SqlDataReader* và điều khiển, bạn có thể thực hành các ví dụ sau:

1. Viết phương thức, nhận tham số là phát biểu *Select*, sau đó kết nối cơ sở dữ liệu và liệt kê danh sách tên cột dữ liệu cùng với dữ liệu trên điều khiển *DataGrid* thông qua đối tượng *ArrayList*
2. Thiết kế *Form*, cho phép liệt kê danh sách *Country* trong bảng *Customers* vào điều khiển *ComboBox* thứ nhất và danh sách loại sản phẩm vào *ComboBox* thứ hai.
3. Bằng cách sử dụng điều khiển *ListView*, bạn có thể điền tên của bảng dữ liệu vào phần *Header*, sau đó điền dữ liệu của các cột tương ứng vào phần dưới của điều khiển *ListView*.
4. Sử dụng điều khiển *TreeView*, bạn có thể liệt kê danh sách các loại đối tượng cơ sở dữ liệu và từng loại đối tượng cơ sở dữ liệu.

3. KẾT LUẬN

Chúng ta vừa tìm hiểu chuyên đề về cách sử dụng các phương thức và thuộc tính của đối tượng *SqlDataReader*.

Trong chuyên đề kế tiếp, bạn sẽ tiếp tục tìm hiểu chi tiết đối tượng *SqlDataAdapter* và *DataSet*.

Chuyên đề 18:

ĐỐI TƯỢNG DATAADAPTER VÀ DATASET

Tóm tắt chuyên đề 18

Đối tượng DataSet thuộc lớp không kết nối, dùng để nắm giữ tập dữ liệu của mọi trình điều khiển dữ liệu trong .NET.

Ngoài ra, đối tượng này được xem như Container cho phép lưu trữ nhiều đối tượng DataTable, DataView và mối quan hệ giữa chúng.

Để sử dụng đối tượng DataSet, bạn cần dùng đến đối tượng DataAdapter như bộ điều phối và điền dữ liệu vào đối tượng DataSet.

Người sử dụng có thể xử lý, cập nhật, xóa, thêm mới hay điều hướng trên tập dữ liệu của DataSet mà không ảnh hưởng đến dữ liệu nguồn, trừ khi họ chỉ thị cập nhật trở lại dữ liệu nguồn bằng phương thức Update của đối tượng DataAdapter.

Bằng cách sử dụng cơ sở dữ liệu SQL Server, chúng ta khám phá các thuộc tính, phương thức của hai đối tượng chính trong chuyên đề này là SqlDataAdapter và DataSet.

Các vấn đề chính sẽ được đề cập:

- ✓ *Đối tượng SqlDataAdapter.*
- ✓ *Đối tượng DataSet.*
- ✓ *Cập nhật dữ liệu từ đối tượng DataSet.*
- ✓ *Ghi và đọc XML bằng đối tượng DataSet.*

1. ĐỐI TƯỢNG SQLDATAADAPTER

Hai thành phần chính của *ADO.NET* là *.NET Data Provider* và *DataSet*, *SqlDataAdapter* là một trong những *.NET Data Provider* dùng để kết nối cơ sở dữ liệu *SQL Server*.

Sau khi kết nối cơ sở dữ liệu bằng đối tượng *SqlConnection* (đối với cơ sở dữ liệu *SQL Server*), bộ điều phối *SqlDataAdapter* sẽ dùng phương thức *Fill* của chúng để điền dữ liệu từ một hay nhiều bảng vào từng đối tượng *DataTable* trong đối tượng *DataSet*.

Nếu người sử dụng thay đổi dữ liệu trên đối tượng *DataSet*, bạn có thể cập nhật dữ liệu đã được thay đổi vào dữ liệu nguồn bằng phương thức *Update*.

Sau đây, chúng ta tìm hiểu các phương thức và thuộc tính của đối tượng *SqlDataAdapter*.

1.1. Phương thức

Hai phương thức chính của đối tượng *SqlDataAdapter* thường sử dụng là *Fill* và *Update*.

1.1.1. Phương thức Fill

Phương thức *Fill* dùng để điền dữ liệu từ dữ liệu nguồn vào đối tượng *DataSet* hay đối tượng *DataTable* với cú pháp như sau:

```
myData.Fill(DataSet Object,TableName As String)

myData.Fill(DataTable Object)

myData.Fill(DataSet Object, _
StartRecord As Ineteger, _
MaxRecord As Ineteger,TableName As String)
```

Chẳng hạn, bạn khai báo để điền dữ liệu từ cơ sở dữ liệu nguồn là tập dữ liệu của cơ sở dữ liệu *SQL Server* vào đối tượng *DataSet* như sau:

```
Dim myData As New SqlDataAdapter(SQL, myCon)
Dim myDS As New DataSet
myData.Fill(myDS, SQL)
```

1.1.2. Phương thức Update

Khi cần cập nhật dữ liệu thay đổi kể từ lần cập nhật cuối cùng từ đối tượng *DataSet* (hay đối tượng *DataTable*) vào dữ liệu nguồn, bạn có thể sử dụng phương thức *Update* với cú pháp tương tự như sau:

```
myData.Update(DataSet Object)
myData.Update(DataTable Object)
```

Ví dụ, sau khi người sử dụng thay đổi dữ liệu trên đối tượng *DataSet*, bằng cách sử dụng phương thức *Update* để cập nhật những dữ liệu đã thay đổi vào dữ liệu nguồn từ *DataSet* như ví dụ 18-1:

Ví dụ 18-1: Cập nhật dữ liệu

```
Dim SQL As String
SQL = "Select EmployeeID,FirstName, LastName "
SQL += "  from Employees"
Dim myData As New SqlDataAdapter(SQL, myCon)
Dim myDS As New DataSet
' Điền dữ liệu vào DataSet
myData.Fill(myDS)
' Cập nhật dữ liệu cột FirstName của mẩu tin thứ nhất
myDS.Tables(0).Rows(0).Item("FirstName")="Lan"
' Cập nhật vào dữ liệu nguồn từ đối tượng DataSet
myData.Update(myDS)
```

1.2. Thuộc tính

Ngoài hai phương thức thường sử dụng vừa trình bày ở trên, bạn có thể sử dụng các thuộc tính chính như: *DeleteCommand*, *InsertCommand*, *SelectCommand*, *UpdateCommand* và *TableMappings*.

1.2.1. Thuộc tính DeleteCommand

Cho phép bạn gán hay lấy chuỗi *SQL* dạng *Delete* tương ứng với bảng dữ liệu khai báo để xóa dữ liệu trong dữ liệu nguồn ứng với dữ liệu trong đối tượng *DataSet* hay *DataTable*.

1.2.2. Thuộc tính InsertCommand

Thuộc tính cho phép bạn gán hay lấy chuỗi *SQL* dạng *Insert* tương ứng với bảng dữ liệu được khai báo khi đọc dữ liệu từ dữ liệu nguồn vào đối tượng *DataSet* hay *DataTable*.

1.2.3. Thuộc tính SelectCommand

Cho phép bạn gán hay lấy chuỗi *SQL* dạng *Select* tương ứng với bảng dữ liệu đã khai báo để đọc dữ liệu từ dữ liệu nguồn vào đối tượng *DataSet* hay *DataTable*.

1.2.4. Thuộc tính UpdateCommand

Tương tự như vậy, thuộc tính này cho phép bạn gán hay lấy chuỗi *SQL* dạng *Update* tương ứng với bảng dữ liệu khai báo để cập nhật dữ liệu vào dữ liệu nguồn từ đối tượng *DataSet* hay *DataTable*.

1.2.5. Thuộc tính TableMappings

Đây là thuộc tính ánh xạ tên bảng dữ liệu trong đối tượng *DataTable* hay *DataSet* muốn cập nhật trở lại dữ liệu nguồn với bảng cần cập nhật trong dữ liệu nguồn.

Lưu ý, nếu bạn sử dụng điều khiển *SqlDataAdapter* thì 4 thuộc tính này sẽ có phát biểu *SQL* tương ứng khi chọn tùy chọn trong cửa sổ *"Choose a Query Type"*. Tuy nhiên, trong trường hợp bạn khai báo đối tượng *SqlDataAdapter* thì 3 thuộc tính *InsertCommand*, *DeleteCommand* và *UpdateCommand* chưa tồn tại.

Để tạo ra 3 phát biểu ứng với 3 thuộc tính này, bạn sử dụng đối tượng *SqlCommandBuilder* để tạo ra 3 phát biểu *SQL* tương ứng cho 3 thuộc tính này cùng với việc sử dụng thuộc tính *TableMappings*.

Chẳng hạn, sau khi khai báo và khởi tạo đối tượng *SqlDataAdapter* và điền dữ liệu vào đối tượng *DataSet*, người sử dụng có thể thay đổi dữ liệu trên điều khiển *DataGrid* rồi nhấn nút *Update*.

Bằng cách khai báo và sử dụng đối tượng *SqlCommandBuilder* để tạo ra các phát biểu cập nhật dữ liệu từ đối tượng *DataSet* cho đối tượng *SqlDataAdapter* để cập nhật chúng vào dữ liệu nguồn như ví dụ 18-2:

Ví dụ 18-2: Lấy những mẩu tin có thay đổi

```
DataGrid1.Update()
If myDS.HasChanges = True Then
  Dim newDS As New DataSet
  newDS = myDS.GetChanges
  ' Khai báo và khởi tạo đối tượng SqlCommandBuilder
  Dim myBuilder As New SqlCommandBuilder(myData)
  ' Khai báo ánh xạ tên bảng dữ liệu
  myData.TableMappings.Add("Table", _
    myDS.Tables(0).TableName)
  ' Cập nhật dữ liệu nguồn
  myData.Update(newDS)
  LoadData()
End If
```

Để sử dụng *SqlDataAdapter* bạn dùng một trong hai cách, cách thứ nhất là khai báo và sử dụng đối tượng *SqlDataAdapter*, cách thứ hai dùng điều khiển *SqlDataAdapter* từ thanh công cụ (*Toolbox*).

1.3. Điều khiển SqlDataAdapter

Để khai báo và sử dụng điều khiển *SqlDataAdapter*, trước tiên bạn kéo điều khiển này vào *Form* từ ngăn *Data* của *ToolBox*, kế đến chọn khai báo mới hay sử dụng lại kết nối cơ sở dữ liệu đang có trong phần "*Choose Your Data Connection*".

Trong trường hợp chọn nút *New Connection*, bạn chọn trình điều khiển cơ sở dữ liệu SQL *Server*, ứng với cơ sở dữ liệu *SQL Server* cùng với tham số khác như: Tên *Server*, tên cơ sở dữ liệu, tài khoản và *password*.

Kế đến, chọn loại câu truy vấn trong phần "*Choose a Query Type*", đối với trường hợp này chúng ta chọn mặc định "*Use SQL Statements*". Bằng cách khai báo phát biểu *SQL* trong phần "*Generate the SQL Statements*" hay sử dụng tiện ích hỗ trợ "*Query Builder*".

Sau khi khai báo phát biểu *SQL* hợp lệ, giả sử chúng ta khai báo phát biểu *Select* để truy vấn bảng *Employees* trong cơ sở dữ liệu *Northwind* như sau:

```
SELECT
  EmployeeID,
  LastName,
  FirstName,
  Title,
  BirthDate,
  Address,
  City
FROM
  Employees
```

Ngoài ra, bạn cũng có thể khai báo phát biểu *Select*: *SELECT * FROM Employees*, có hay không sử dụng mệnh đề *Where*.

Kết thúc quá trình cấu hình điều khiển *SqlDataAdapter* trở lại giao diện *Form*, lập tức điều khiển *SqlConnection* sẽ xuất hiện trên *Form*.

Lưu ý, bạn nên khai báo chuỗi kết nối để cập nhật lại thuộc tính *ConnectionString* của điều khiển *SqlConnection* trong khi thi hành chương trình từ biến cố *Load* của *Form* như ví dụ 18-3:

Ví dụ 18-3: Khởi tạo chuỗi kết nối cơ sở dữ liệu

```
Private Sub frmDataAdapterControl_Load(ByVal _
sender As System.Object, ByVal e As _
System.EventArgs) Handles MyBase.Load
    ' Gán lại thuộc tính chuỗi kết nối
    Me.SqlConnection1.ConnectionString = gsCon
End Sub
```

Trong đó, biến sử dụng toàn cục của ứng dụng được khai báo và khởi tạo giá trị (từ tập tin *App.config*) trong phương thức *Main* của *Module* trực thuộc *Project* như ví dụ 18-3-1:

Ví dụ 18-3-1: Chuỗi kết nối cơ sở dữ liệu

```
' Khai báo sử dụng không gian tên
Imports System.Configuration
Module Module1
  Public gsCon As String
  Sub Main()
    ' Khởi tạo biến gsCon
    gsCon = _
ConfigurationSettings.AppSettings("strConn")
    ' Khởi động frmMain
    Dim myForm As New frmMain
    myForm.ShowDialog()
  End Sub
End Module
```

Kế đến, chọn điều khiển *SqlDataDapter*, *R-Click* | *Generate DataSet*, bằng cách chọn tên mới hay điền vào đối tượng *DataSet* đang tồn tại trong cửa sổ *Generate DataSet*, bạn sẽ tạo ra điều khiển *DataSet* trên *Form* với tên *DataSet11*.

Để điền dữ liệu vào đối tượng *DataSet*, trước tiên bạn sử dụng phương thức *Fill* cùng với phát biểu *SQL* hay tên bảng dữ liệu tương ứng như ví dụ 18-4:

Ví dụ 18-4: Điền dữ liệu vào đối tượng DataSet

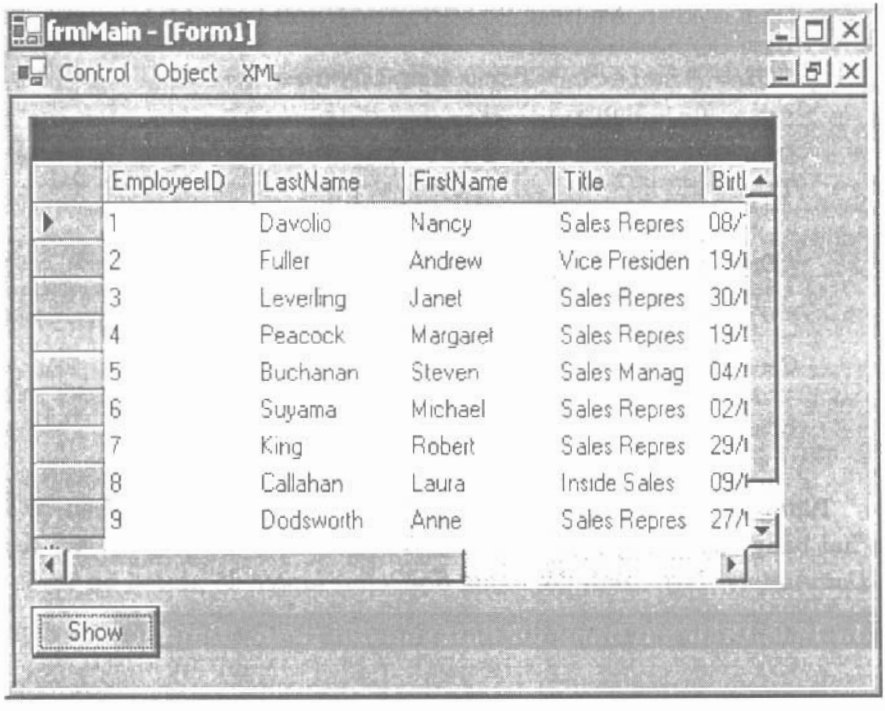
```
Private Sub Button1_Click(ByVal sender As _
System.Object, ByVal e As System.EventArgs) _
Handles Button1.Click
```

```
    ' Điền dữ liệu vào DataSet11
    SqlDataAdapter1.Fill(DataSet11)
    ' Điền dữ liệu vào DataGrid
    DataGrid1.DataSource = DataSet11.Tables(0)
End Sub
```

Trong trường hợp điền dữ liệu của hai bảng vào chung một đối tượng *DataSet*, bạn có thể sử dụng cú pháp tương tự như sau:

```
' Điền dữ liệu bảng thứ nhất vào DataSet11
SqlDataAdapter1.Fill(DataSet11)
' Điền dữ liệu bảng thứ hai vào DataSet11
SqlDataAdapter2.Fill(DataSet11)
' Điền dữ liệu vào DataGrid
DataGrid1.DataSource = DataSet11.Tables(0)
```

Khi thực thi chương trình, bạn chọn *Control | DataAdapter* trên *menu*, rồi chọn nút *Show* thì kết quả trình bày như hình 18-1.



Hình 18-1: *Sử dụng điều khiển SqlDataAdapter*

1.4. Đối tượng SqlDataAdapter

Khác với trường hợp trên, bạn có thể khai báo và sử dụng đối tượng *SqlDataAdapter* để điền dữ liệu từ dữ liệu nguồn vào đối tượng hay điều khiển *DataSet* (được tạo ra từ điều khiển *SqlDataAdapter* trong *Design View*).

Tuy nhiên, khi sử dụng điều khiển *DataSet* thì bạn cần thêm điều khiển này từ cửa sổ *ToolBox* vào *Form* hoặc tạo ra bằng chức năng *Generate DataSet* của điều khiển *SqlDataAdapter*.

Để khai báo và sử dụng đối tượng *SqlDataAdapter*, bạn thêm *Form* vào *Project* rồi đặt tên *frmDataAdapterObject*, sau đó khai báo trong biến cố *Click* của nút *Show* như ví dụ 18-5:

Ví dụ 18-5: Đối tượng SqlDataAdapter

```
Private Sub Button1_Click(ByVal sender As _
System.Object, ByVal e As System.EventArgs) _
Handles Button1.Click
   Dim myCon As New SqlConnection(gsCon)
   Dim SQL As String
   SQL = "Select * from Employees"
   ' Khai báo và khởi tạo đối tượng SqlDataAdapter
   Dim myData As New SqlDataAdapter(SQL, myCon)
   Try
      Dim myDS As New DataSet
   ' Điền dữ liệu vào đối tượng DataSet
      myData.Fill(myDS)
      DataGrid1.DataSource = myDS.Tables(0)
   Catch ex As Exception
      MsgBox(ex.Message)
   End Try
End Sub
```

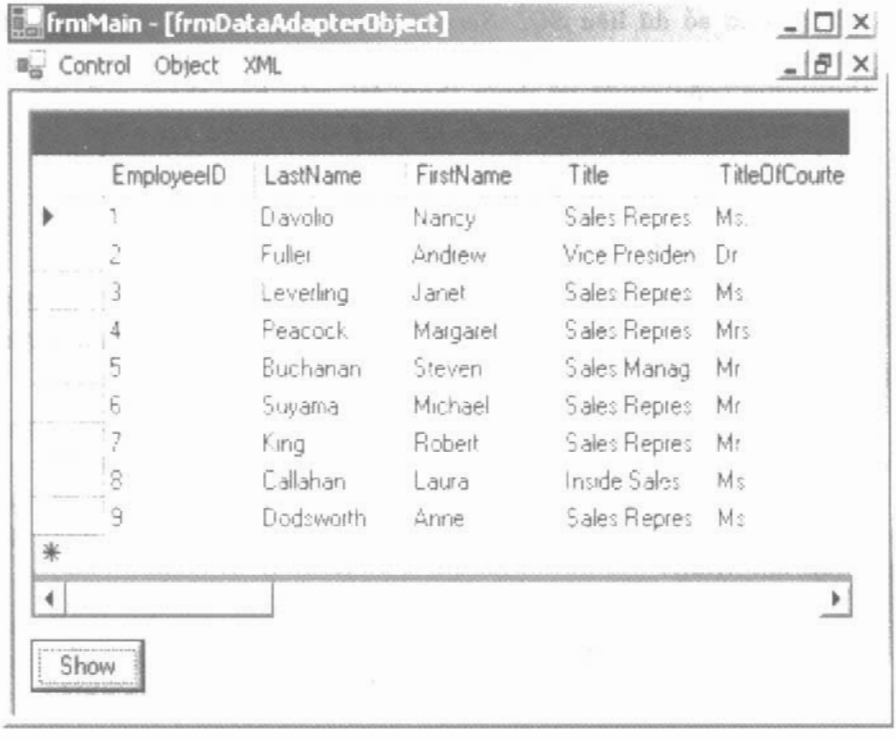
Tương tự như trường hợp điều khiển *SqlDataAdapter*, để điền dữ liệu của hai bảng vào chung một đối tượng *DataSet*, bạn sử dụng hai đối tượng *SqlDataAdapter* hay chính đối tượng *SqlDataAdapter* đó như ví dụ 18-6.

Ví dụ 18-6: Điền hai bảng dữ liệu vào đối tượng DataSet

```
Dim SQL As String
   SQL = "Select * from Employees"
   ' Khai báo và khởi tạo đối tượng SqlDataAdapter
```

```
    Dim myData As SqlDataAdapter
    Try
        Dim myDS As New DataSet
    ' Điền dữ liệu bảng thứ nhất vào đối tượng DataSet
        myData = New SqlDataAdapter(SQL, myCon)
        myData.Fill(myDS)
    ' Điền dữ liệu bảng thứ hai vào đối tượng DataSet
        SQL="Select * from Orders"
        myData= New SqlDataAdapter(SQL, myCon)
        myData.Fill(myDS)
        DataGrid1.DataSource = myDS.Tables(0)
    Catch ex As Exception
        MsgBox(ex.Message)
End Try
```

Khi thực thi chương trình, bạn chọn *menu* có tên *Object* | *DataAdapter* trên *menu*, kế đến chọn nút *Show* thì kết quả trình bày như hình 18-2.



Hình 18-2: *Khai báo sử dụng đối tượng SqlDataAdapter*

Lưu ý, *SqlDataAdapter* là một đối tượng trung gian làm bộ điều phối cơ sở dữ liệu, chúng ta sẽ tìm hiểu thêm đối tượng này trong phần trình bày đối tượng *DataTable*.

Để làm việc với điều khiển hay đối tượng *DataAdapter*, bạn có thể thực hiện các ví dụ sau:

1. Tạo ứng dụng *Console*, sau đó đọc danh sách các bảng dữ liệu từ một tên cơ sở dữ liệu nhập từ bàn phím điền vào đối tượng *DataTable* và liệt kê chúng ra màn hình.
2. Thiết kế *Form*, cho phép người sử dụng chọn tên bảng dữ liệu trên điều khiển *ComboBox* và trình bày mẩu tin của bảng được chọn trên điều khiển *DataGrid*.
3. Bằng cách chọn 3 bảng dữ liệu *Customers, Orders* và *Order Details*, bạn có thể đọc dữ liệu của ba bảng này vào đối tượng *DataSet*, sau đó trình danh sách của từng khách hàng cùng với từng đơn đặt hàng ra màn hình *Command Prompt*.
4. Sử dụng cơ sở dữ liệu *SQL Server* có tên *Northwind*, bạn thiết kế *Form* và dùng đối tượng *SqlDataAdapter* để điền dữ liệu của bảng *Categories*, nếu người sử dụng thay đổi, xóa hay thêm mẩu tin và nhấn nút *Update*, bạn cập nhật sự thay đổi đó trở lại cơ sở dữ liệu nguồn.

2. ĐỐI TƯỢNG DATASET

DataSet là một trong hai thành phần chính của *ADO.NET*, nó được thiết kế nhằm tăng tốc độ truy cập và thao tác dữ liệu trong môi trường ứng dụng đa lớp (*n-tier*).

DataSet được xem như một *Container*, dùng để lưu các đối tượng *DataTable*, *DataView* và *DataRelation* cùng với lược đồ quan hệ của chúng. Khi có tập dữ liệu lớn hoặc cần thao tác trên tập dữ liệu đó, bạn nên sử dụng đối tượng *DataSet* để nắm giữ chúng thay vì sử dụng đối tượng *SqlDataReader*.

Lưu ý, đối tượng *DataSet* sử dụng bộ nhớ truy cập nhanh (*Cache*) của máy tính để lưu trữ dữ liệu mà nó nắm giữ, chính vì vậy nếu dung lượng dữ liệu càng lớn thì đối tượng này chiếm giữ bộ nhớ càng nhiều.

2.1. Thuộc tính

Đối tượng *DataSet* cung cấp các thuộc tính như: *DataSetName, HasErrors, Tables, Relations.*

2.1.1. Thuộc tính DataSetName

Thuộc tính *DataSetName*, cho phép gán hay lấy chuỗi ứng với tên của đối tượng *DataSet*. Ví dụ, bạn khai báo như sau:

```
Dim myDS As New DataSet("myEmployees")
            myData.Fill(myDS)
Console.WriteLine(myDS.DataSetName)
```

2.1.2. Thuộc tính HasErrors

Thuộc tính *HasErrors* trả về giá trị *Boolean*, cho biết một trong những đối tượng *DataTable* bên trong *DataSet* phát sinh lỗi.

2.1.3. Thuộc tính HasChanges

Thuộc tính này trả về *True* nếu có sự thay đổi dữ liệu trong đối tượng *DataSet*. Bằng cách dựa vào thuộc tính này, bạn có thể kiểm soát người sử dụng có thay đổi dữ liệu trong đối tượng *DataSet* hay không.

2.1.4. Thuộc tính Tables

Trong đối tượng *DataSet* có thể có một hoặc nhiều đối tượng *DataTable*, thuộc tính *Tables* trả về một tập đối tượng *DataTable*. Chẳng hạn, bạn có thể truy cập từng đối tượng *DataTable* và in ra tên của chúng như sau:

```
Dim myDS As New DataSet
myData.Fill(myDS)
' Duyệt trên từng đối tượng DataTable
For Each myDT As DataTable in myDS.Tables
    Console.WriteLine(myDT.TableName)
Next
```

Ngoài ra, bạn cũng có thể tham chiếu đến *Table* chỉ định trong đối tượng *DataSet* bằng cách cung cấp chỉ mục (bảng nào tạo ra trước sẽ có số chỉ mục nhỏ hơn) hay tên của đối tượng *DataTable*.

Lưu ý, để biết được số mẩu tin của bảng trong đối tượng *DataSet*, bạn có thể truy cập đến thuộc tính *Count* của *Rows Collection* thuộc *Tables* thứ i như sau:

```
Dim j As Integer
j=myDS.Tables(0).Rows.Count
```

Ví dụ, khi điền dữ liệu vào điều khiển *DataGrid*, bạn cần chỉ định tên bảng dữ liệu của đối tượng *DataTable* như sau:

```
Dim myDS As New DataSet
myData.Fill(myDS)
' Điền dữ liệu vào điều khiển DataGrid
DataGrid1.DataSource= myDS.Tables("Employees")
```

Ngoài ra, bạn cũng có thể thêm đối tượng *DataTable* vào đối tượng *DataSet* bằng phương thức *Add* của *Tables Collection* (trình bày trong phần đối tượng *DataTable*) như sau:

```
Dim myDS As New DataSet
            myData.Fill(myDS)
Dim myDT As New DataTable
myData = New SqlDataAdapter(SQL, myCon)
' Điền dữ liệu vào đối tượng DataTable
myData.Fill(myDT)
' Thêm đối tượng DataTable vào đối tượng DataSet
myDS.Tables.Add(myDT)
```

2.1.5. Thuộc tính Relations

Tương tự thuộc tính *Table*, thuộc tính *Relations* trả về tập các quan hệ của những cặp đối tượng *DataTable* trong đối tượng *DataSet*.

Ngoài ra, bạn cũng có thể sử dụng phương thức *Add* của *Relations Collection* để thêm quan hệ giữa hai bảng với nhau (trình bày trong phần đối tượng *DataRelation*).

2.2. Phương thức

Đối tượng *DataSet* cung cấp các phương thức, cho phép bạn chấp nhận (*AcceptChanges*) hay từ chối (*RejectChanges*) dữ liệu đã thay đổi, trích lọc (*GetChanges*) dữ liệu thay đổi, ghi (*WriteXML*) và đọc (*ReadXML*) tập tin *XML*, kết hợp (*Merge*), xóa (*Clear*) hay sao chép (*Copy*) đối tượng *DataSet*.

2.2.1. Phương thức AcceptChanges

Chấp nhận sự thay đổi dữ liệu do người sử dụng tạo ra trên đối tượng *DataSet*.

2.2.2. Phương thức Clear

Khi có nhu cầu loại bỏ tất cả những mẩu tin trong các đối tượng *DataTable* thuộc đối tượng *DataSet*, bạn có thể sử dụng phương thức *Clear*.

Tuy nhiên, nếu loại bỏ đối tượng *DataTable* khỏi đối tượng *DataSet* thì bạn dùng phương thức *Clear* của *Tables Collection*.

2.2.3. Phương thức RejectChanges

Trong trường hợp bạn không chấp nhận sự thay đổi dữ liệu trên đối tượng *DataSet* thì chọn phương thức này.

2.2.4. Phương thức GetChanges

Phương thức *GetChanges* trả về một đối tượng *DataSet* chứa đựng các đối tượng *DataTable*, bao gồm những hàng dữ liệu có thay đổi, thêm mới hay xóa. Chẳng hạn, trong trường hợp bạn cập nhật sự thay đổi dữ liệu từ *DataSet* vào dữ liệu nguồn.

Trước tiên, thêm *From* vào *Project* với tên *frmUpdate* và khai báo phương thức *LoadData* để điền dữ liệu vào điều khiển *DataGrid* như 18-7:

Ví dụ 18-7: Điền dữ liệu vào điều khiển DataGrid

```
Sub LoadData()
    ' Xóa đối tượng DataTable
    myDS.Tables.Clear()
    ' Khởi tạo đối tượng DataConnection
    Dim myCon As New SqlConnection(gsCon)
    Try
        ' Mở kết nối cơ sở dữ liệu
        myCon.Open()
        strSQL = "select * from tblCustomers"
        ' Khai báo và khởi tạo đối tượng SqlDataAdapter
        myData = New SqlDataAdapter(strSQL, myCon)
        ' Điền dữ liệu vào đối tượng DataSet
        myData.Fill(myDS, strSQL)
        ' Điền dữ liệu vào điều khiển DataGrid
        Me.DataGrid1.DataSource = myDS.Tables(0)
    Catch ex As Exception
        MsgBox(ex.Message)
    End Try
End Sub
```

Lưu ý, bảng *tblCustomers* không tồn tại trong cơ sở dữ liệu *Northwind* trừ khi trước đó bạn đã tạo ra nó. Nếu bảng này chưa tồn tại, bạn có thể sử dụng phát biểu *SQL* tạo bảng như ví dụ 18-8:

Ví dụ 18-8: Tạo bảng tblCustomers

```
Create Table tblCustomers
(
     CustomerID varchar(10) Primary Key,
     CustomerName varchar(50),
     Address varchar(100),
     City varchar(50)
)
```

Sau đó, chuyển mẩu tin từ bảng *Customers* trong cơ sở dữ liệu *Northwind* vào bảng *tblCustomers* bằng cách sử dụng *Query Analyzer* (dùng làm dữ liệu mẫu) như ví dụ 18-9:

Ví dụ 18-9: Thêm mẩu tin vào bảng tblCustomers

```
Insert into tblCustomers
Select CustomerID ,CompanyName, Address,City
From Customers
```

Kế đến, bạn khai báo để gọi phương thức *LoadData* trong biến cố của *frmUpdate* như ví dụ 18-10.

Ví dụ 18-10: Gọi phương thức LoadData()

```
Private Sub frmUpdate_Load(ByVal sender As _
System.Object, ByVal e As System.EventArgs) _
Handles MyBase.Load
   LoadData()
End Sub
```

Chú ý, nhằm giới hạn số mẩu tin cần truy vấn, bạn sử dụng phương thức *Fill* của đối tượng *SqlDataAdapter* với 4 tham số như sau:

```
myData.Fill(DataSet Object, _
StartRecord As Ineteger, _
   MaxRecord As Ineteger,TableName As String)
```

Chẳng hạn, bạn khai báo phương thức *GetValue* (*Overload*) trong lớp *clsDatabase*, nhận tham biến là đối tượng *DataSet*, tham trị thứ nhất là phát biểu *Select*, tham trị thứ hai là số mẩu tin bắt đầu và tham trị cuối cùng là số mẩu tin cần lấy ra như ví dụ 18-11.

Ví dụ 18-11: Giới hạn mẩu tin

```
Public Function GetValue( _
  ByRef myDS As DataSet, _
  ByVal strSQL As String, _
  ByVal StartRecord As Integer, _
  ByVal MaxRecords As Integer) As String
    Dim strError As String = ""
    ' Khai báo và khởi tạo đối tượng SqlConnection
    Dim myCon As New SqlConnection(gsCon)
    ' Khai báo đối tượng SqlDataAdapter
    Dim adData As SqlDataAdapter
    Try
      ' Mở kết nối cơ sở dữ liệu
      myCon.Open()
      ' Khởi tạo đối tượng SqlDataAdapter
      adData = New SqlDataAdapter(strSQL, myCon)
      ' Gọi tptFill với 4 tham số
      adData.Fill(myDS, StartRecord, _
        MaxRecords, strSQL)
      strError = "OK"
    Catch ex As Exception
      strError = ex.Message
    Finally

    End Try
    Return strError
End Function
```

Kế tiếp, bạn thêm *Form* vào *Project* và đặt tên *frmLimitRecords*. Thêm điều khiển *DataGrid* và hai nút ứng với hai chức năng trình bày tất cả mẩu tin hay giới hạn mẩu tin theo hai điều khiển *TextBox*.

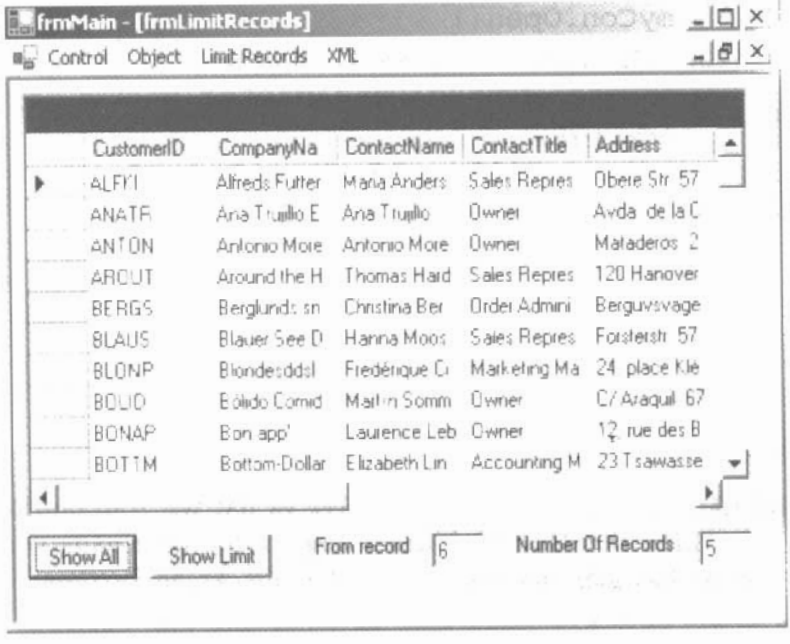
Sau đó, khai báo trong biến cố *Click* của nút *ShowAll* để gọi phương thức *getValue* với tham số như ví dụ 18-11-1.

Ví dụ 18-11-1: Gọi phương thức GetValue

```
Private Sub Button1_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles Button1.Click
    ' Khai báo phát biểu SQL
    Dim SQL As String
```

```
        SQL = "Select * from Customers"
        Try
            'Khởi tạo đối tượng DataSet
            myDS = New DataSet
            'Gọi phương thức GetValue hai tham số
            cls.getValue(myDS, SQL)
            DataGrid1.DataSource = myDS.Tables(0)
        Catch ex As Exception
            MsgBox(ex.Message)
        End Try
    End Sub
```

Khi thực thi chương trình, nếu bạn nhấn nút Show All, lập tức danh sách khách hàng sẽ trình bày như hình 18-3.



Hình 18-3: *Tất cả mẩu tin của bảng Customers*

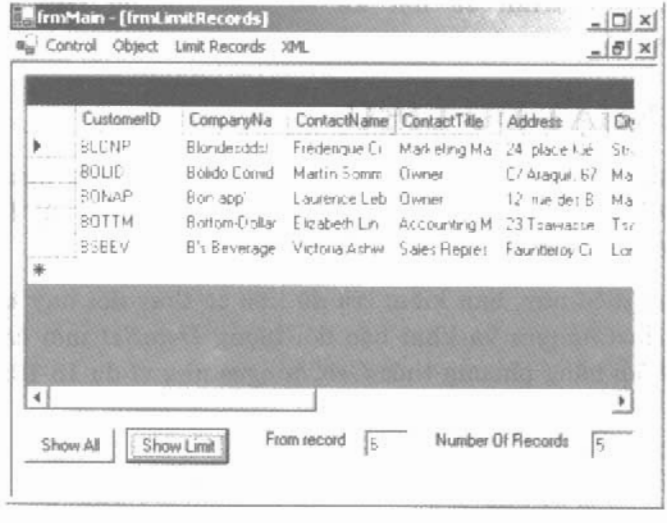
Tuy nhiên, nếu bạn muốn trình bày N mẩu tin kể từ mẩu tin thứ i trong bảng *Customers* (tùy vào phát biểu *Select* có sử dụng mệnh đề *Order By* hay *Where* hay không, tập dữ liệu sẽ có thứ tự khác nhau), kết quả sẽ trình bày tương tự như hình 18-3-1.

Để làm điều này, bạn khai báo trong biến cố Click của nút Show Limit để gọi phương thức *GetValue* với 4 tham số như ví dụ 18-11-2:

Ví dụ 18-11-2: Gọi phương thức GetValue với 4 tham số

```
Private Sub btnShowLimit_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles btnShowLimit.Click
    ' Khai báo phát biểu SQL
    Dim SQL As String
    SQL = "Select * from Customers"
    Try
        ' Khởi tạo đối tượng DataSet
        myDS = New DataSet
        ' Gọi phương thức GetValue với 4 tham số
        cls.getValue(myDS, SQL, txtFrom.Text, _
            txtTo.Text)
        ' Điền dữ liệu vào điều khiển DataGrid
        DataGrid1.DataSource = myDS.Tables(0)
    Catch ex As Exception
        MsgBox(ex.Message)
    End Try
End Sub
```

Giả sử trong trường hợp này, muốn trình bày 5 mẩu tin kể từ mẩu tin thứ 6 trong bảng *Customers*, bạn nhập số 6 vào phần *From record* và số 5 vào phần *Number of records* rồi nhấn nút *Show Limit*, kết quả trình bày như hình 18-3-1.



Hình 18-3-1: *Giới hạn mẩu tin*

Lưu ý, bạn sử dụng hình thức giới hạn này khi tập dữ liệu trong bảng quá lớn, trong trường hợp tập dữ liệu vừa phải thì bạn nên sử dụng đối tượng *DataView* để lọc dữ liệu theo tiêu chuẩn chọn lọc.

Để tìm hiểu thêm về đối tượng *DataSet*, bạn có thể thực hành các ví dụ sau:

1. Sử dụng đối tượng *DataSet* để đọc 3 bảng dữ liệu *Customers*, *Orders* và *Order Details* rồi điền lên điều khiển *DataGrid*, mỗi khi người sử dụng chọn một khách hàng thì chương trình có thể liệt kê danh sách các đơn đặt hàng cùng với tổng số tiền họ đã thanh toán.

2. Thiết kế *Form*, sử dụng điều khiển *OleDbDataAdapter* để trình bày danh sách *Employees* thuộc cơ sở dữ liệu *Northwind.mdb* trên điều khiển *DataGrid*. Sau khi người sử dụng thay đổi hay thêm mới mẩu tin, bạn cho phép người sử dụng cập nhật những thay đổi đó trở lại dữ liệu nguồn.

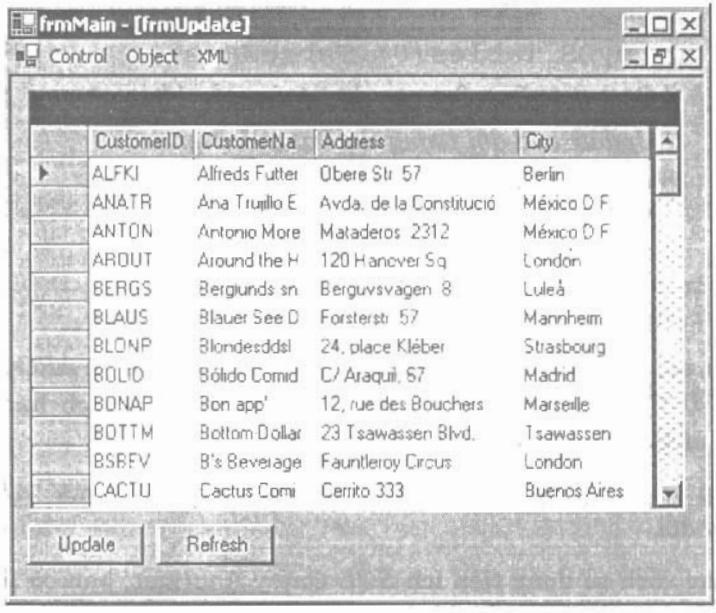
3. Sử dụng đối tượng *DataSet* để đọc dữ liệu từ bảng *Customers*, sau đó điền danh sách *Country* vào điều khiển *ComboBox*, mỗi khi người sử dụng chọn *Country* thì bạn trình bày danh sách khách hàng của *Country* đó.

4. Tạo ứng dụng *Windows Forms*, cho phép người sử dụng chọn bảng dữ liệu, nhập số mẩu tin bắt đầu và số mẩu tin cần trình bày, sau đó chương trình sẽ liệt kê số mẩu tin đó trên điều khiển *DataGrid*.

3. CẬP NHẬT DỮ LIỆU

Sau khi nạp dữ liệu lên điều khiển *DataGrid*, nếu người sử dụng thay đổi dữ liệu trên *DataGrid* như hình 18-4 và nhấn nút *Update*, lập tức dữ liệu đó sẽ được cập nhật trở lại dữ liệu nguồn.

Để làm điều này, bạn kiểm tra dữ liệu có thay đổi hay không bằng thuộc tính *HasChanges* và khai báo đối tượng *DataSet* mới để nhận tập dữ liệu thay đổi bằng phương thức *GetChanges* như ví dụ 18-12:



Hình 18-4: *Cập nhật dữ liệu*

Ví dụ 18-12: Cập nhật dữ liệu

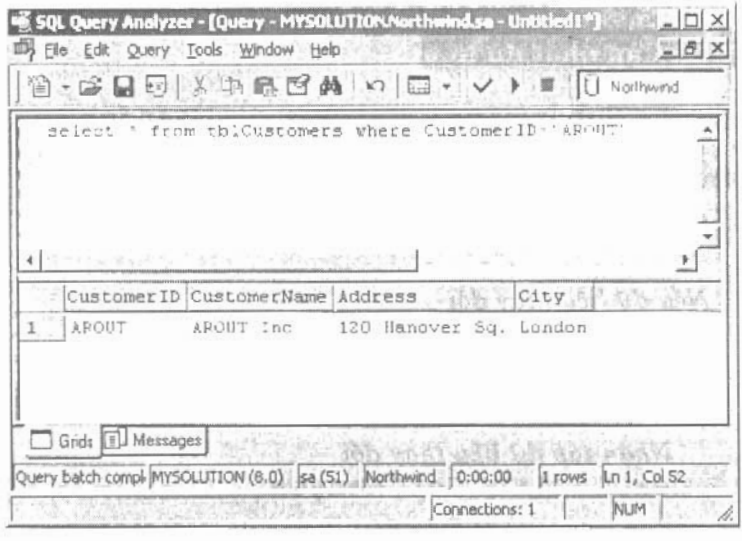
```
Private Sub Button1_Click(ByVal sender As _
System.Object, ByVal e As System.EventArgs) _
Handles Button1.Click
    ' Cập nhật DataGrid
    DataGrid1.Update()
    ' Nếu dữ liệu thay đổi
    If myDS.HasChanges = True Then
        ' Khai báo một đối tượng DataSet
        Dim newDS As New DataSet
        ' Nhận tập dữ liệu thay đổi
        newDS = myDS.GetChanges
        ' Tạo ra các phát biểu SQL tương ứng với thuộc tính
        ' UpdateCommand, DeleteCommand và InsertCommand cho
        ' đối tượng SqlDataAdapter bằng đối tượng
        ' SqlCommandBuilder
        Dim myBuilder As New _
            SqlCommandBuilder(myData)
        ' Sử dụng thuộc tính TableMappings để ánh xạ tên Table
```

```
        myData.TableMappings.Add("Table", _
          myDS.Tables(0).TableName)
        ' Cập nhật DataSet vào dữ liệu nguồn bằng phương thức
        ' Update của đối tượng SqlDataAdapter
        myData.Update(newDS)
        ' Nạp lại dữ liệu
        LoadData()
    End If
End Sub
```

Giả sử, bạn thay đổi tên của khách hàng có mã là *AROUT* thành *"AROUT INC"* và nhấn *Update*, lập tức tên của mã khách hàng này sẽ được cập nhật trong cơ sở dữ liệu *Northwind*.

Chú ý, đối tượng *newDS* chỉ nắm giữ mẫu tin có mã *AROUT* đã được thay đổi.

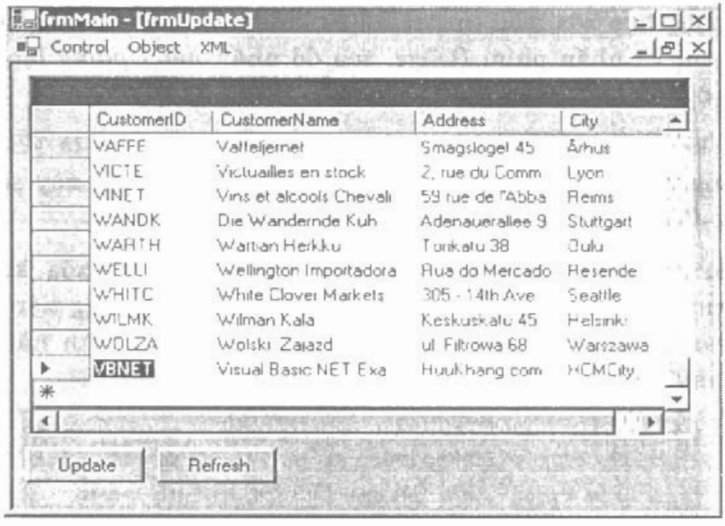
Bằng cách sử dụng tiện ích *SQL Query Analyzer*, bạn có thể liệt kê mẫu tin của khách hàng như hình 18-5.



Hình 18-5: *Kiểm tra tên các khách hàng*

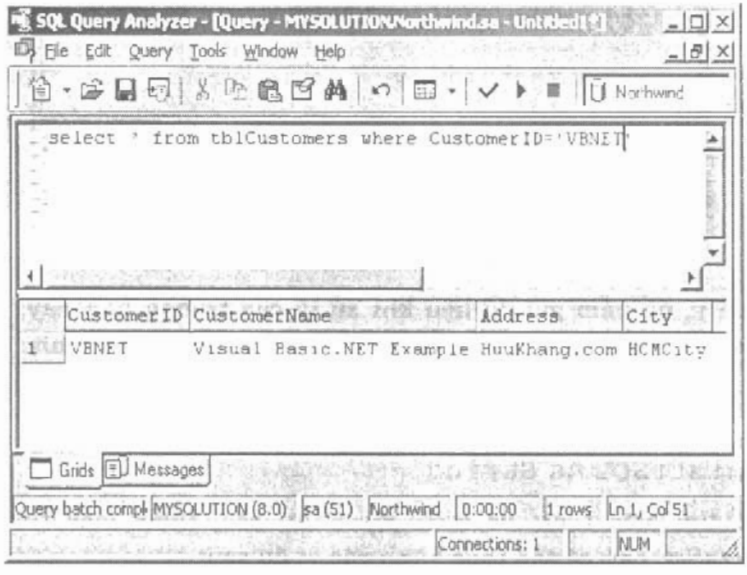
Nếu người sử dụng thêm mẫu tin trên điều khiển *DataGrid* và nhấn nút *Update*, mẫu tin này cũng được thêm vào bảng *tblCustomers*.

Giả sử, bạn thêm mẫu tin có giá trị tương ứng các cột *VBNET, Visual Basic .NET Example, HuuKhang.com, HCMCity* như hình 18-6.



Hình 18-6: *Thêm mẩu tin vào DataSet*

Trong trường hợp cập nhật, đối tượng *newDS* chỉ nắm giữ mẩu tin vừa thêm, sau đó đối tượng *SqlDataAdapter* sẽ thêm mẩu tin này vào bảng *tblCustomers*.



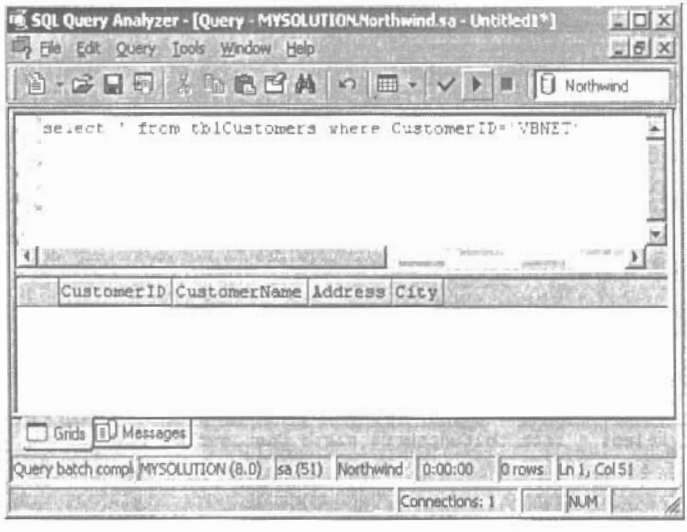
Hình 18-7: *Thêm mẩu tin*

Bằng cách nhấn nút *Update*, trở lại tiện ích *SQL Query Analyzer*, bạn có thể liệt kê mẩu tin của khách hàng có mã *VBNET* như hình 18-7.

Trường hợp xóa mẩu tin trên điều khiển *DataGrid*, bằng cách chọn vào mẩu tin và nhấn phím *Delete*, sau đó nhấn nút *Update* lập tức mẩu tin đó sẽ bị xóa khỏi bảng *tblCustomers*.

Để kiểm tra điều này, bạn chọn vào mẩu tin có mã là *VBNET* vừa thêm trong ví dụ trên, sau khi nhấn phím *Delete*, mẩu tin này sẽ xá khỏi đối tượng *DataSet* (chưa có hiệu lực trong cơ sở dữ liệu).

Nếu bạn nhấn nút *Update*, mẩu tin này sẽ bị xóa khỏi bảng *tblCustomers*, kiểm tra kết quả thực thi bằng cách sử dụng tiện ích *SQL Query Analyzer*, bạn có thể không tìm mẩu tin của khách hàng có mã *VBNET* như hình 18-8.



Hình 18-8: *Xóa mẩu tin*

Lưu ý, để nắm giữ dữ liệu khi xử lý của trường hợp này, bạn cần khai báo đối tượng *DataSet*, *SqlDataAdapter* và biến *strSQL* như sau:

```
Dim myDS As New DataSet
Dim myData As SqlDataAdapter
Dim strSQL As String
```

Để tìm hiểu thêm về cách cập nhật, xóa, thêm mẩu tin vào đối tượng *DataSet*, bạn có thể thực hành các ví dụ sau:

1. Thiết kế *Form* dùng để điền dữ liệu của bảng nào đó trong cơ sở dữ liệu *Northwind*, sau đó cho phép người sử dụng thêm mới, thay đổi hay xóa mẩu tin rồi cập nhật trở lại dữ liệu nguồn.

2. Thiết kế *Form* dùng để điền dữ liệu của bảng nào đó trong cơ sở dữ liệu *Northwind*, sau đó cho phép người sử dụng thêm mới, thay đổi hay xóa mẩu tin rồi liệt kê những mẩu tin thay đổi đó trên điều khiển *DataGrid* khác.

4. ĐỐI TƯỢNG DATASET VÀ XML

Trong cuốn *"Ví dụ và bài tập Visual Basic .Net – Lập trình Windows Forms và tập tin"*, chúng ta đã tìm hiểu về các đối tượng làm việc với dữ liệu định dạng *XML* như: *XmlTextWriter*, *XmlTextReader* và *Xml Document (DOM)*.

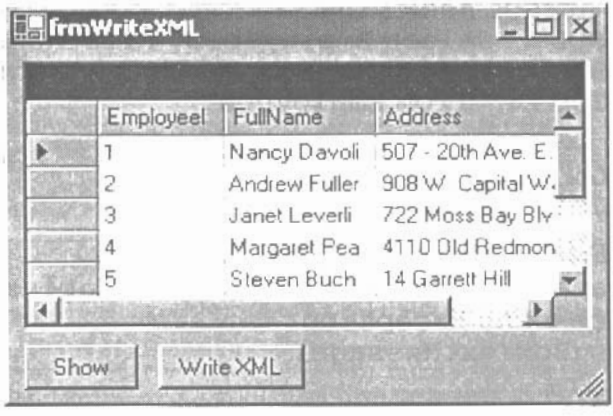
Bạn có thể đọc và ghi nội dung ra và vào tập tin định dạng *Xml*. Ngoài ra, bạn có thể sử dụng đối tượng *Xml Document* để thêm, xóa, cập nhật và điều hướng dữ liệu.

Khi làm việc với đối tượng *DataSet*, bạn có thể sử dụng hai phương thức *WriteXml* dùng để ghi dữ liệu trong đối tượng *DataSet* ra tập tin định dạng *XML* và phương thức *ReadXml* dùng để đọc nội dung tập tin định dạng *Xml* vào đối tượng *DataSet*.

4.1. Phương thức WriteXml

Phương thức *WriteXml* cho phép ghi tập dữ liệu trong đối tượng *DataSet* ra tập tin định dạng *XML*, đối tượng *XmlWriter*, *TextWriter* hay *Stream*.

Tuy nhiên, khi cần ghi tập dữ liệu trong đối tượng *DataSet* ra tập tin định dạng *Xml* thì sử dụng phương thức *WriteXml* với tham số là tên tập tin như hình 18-9.



Hình 18-9: *Ghi dữ liệu ra tập tin XML*

Bằng cách thêm *Form* với tên *frmWriteXml* như hình 18-9 vào *Project* rồi khai báo trong biến cố *Click* của nút *WriteXml* như ví dụ 18-13:

Ví dụ 18-13: Ghi dữ liệu từ DataSet ra định dạng Xml

```
Private Sub Button2_Click(ByVal sender As _
System.Object, ByVal e As System.EventArgs) _
Handles Button2.Click
  Try
    ' Ghi dữ liệu trong đối tượng DataSet ra tập tin XML
    myDS.WriteXml("C:\Employees.xml")
  Catch ex As Exception
    MsgBox(ex.Message)
  End Try
End Sub
```

Lưu ý, để điền dữ liệu lên điều khiển *DataGrid*, bạn khai báo sử dụng phương thức *getValue* của lớp *clsDatabase* trong biến cố *Click* của nút *Show* như ví dụ 18-14:

Ví dụ 18-14: Điền dữ liệu vào điều khiển DataGrid

```
Dim myDS As New DataSet
Private Sub Button1_Click(ByVal sender As _
System.Object, ByVal e As System.EventArgs) _
Handles Button1.Click
  ' Khai báo và khởi tạo đối tượng clsDatabase
  Dim myCls As New clsDatabase(gsCon)
  ' Khai báo chuỗi SQL
  Dim strSQL As String
  strSQL = "Select EmployeeID, FirstName + ' '
  strSQL += "+ LastName As FullName, Address "
  strSQL += " from Employees"
  Try
    ' Gọi phương thức getValue
    myCls.getValue(myDS, strSQL)
    ' Điền dữ liệu vào điều khiển DataGrid
    DataGrid1.DataSource = myDS.Tables(0)
  Catch ex As Exception
    MsgBox(ex.Message)
  End Try
End Sub
```

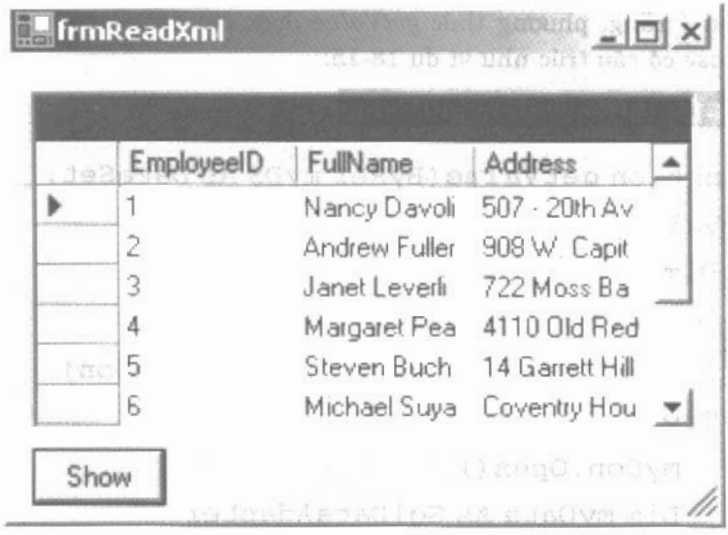
Chú ý rằng, phương thức *getValue* được cài đặt *Overload* trong lớp *clsDatabase* có cấu trúc như ví dụ 18-15:

Ví dụ 18-15: Phương thức getValue

```
Function getValue(ByRef myDS As DataSet, _
ByVal strSQL As String) As String
  Dim strError As String = ""
  ' Chuỗi kết nối được khởi tạo trong Constructor
  Dim myCon As New SqlConnection(gsCon)
  Try
    myCon.Open()
    Dim myData As SqlDataAdapter
    myData =New SqlDataAdapter( strSQL, myCon)
    myDS = New DataSet
    ' Điền dữ liệu vào đối tượng DataSet
    myData.Fill(myDS, strSQL)
  Catch ex As Exception
    strError = ex.Message
  Finally
    myCon.Close()
    myCon.Dispose()
  End Try
  ' Phương thức trả về chuỗi lỗi
  Return strError
End Function
```

4.2. Phương thức ReadXml

Ngoài phương thức *Fill* của đối tượng *SqlDataAdapter* dùng để điền dữ liệu từ dữ liệu nguồn vào đối tượng *DataSet*, bạn có thể điền dữ liệu vào đối tượng *DataSet* bằng cách sử dụng phương thức *ReadXML* để đọc dữ liệu từ tập tin *XML*, đối tượng *TextReader*, *XmlReader* hay *Stream* như hình 18-10



Hình 18-10: *Đọc dữ liệu từ tập tin XML*

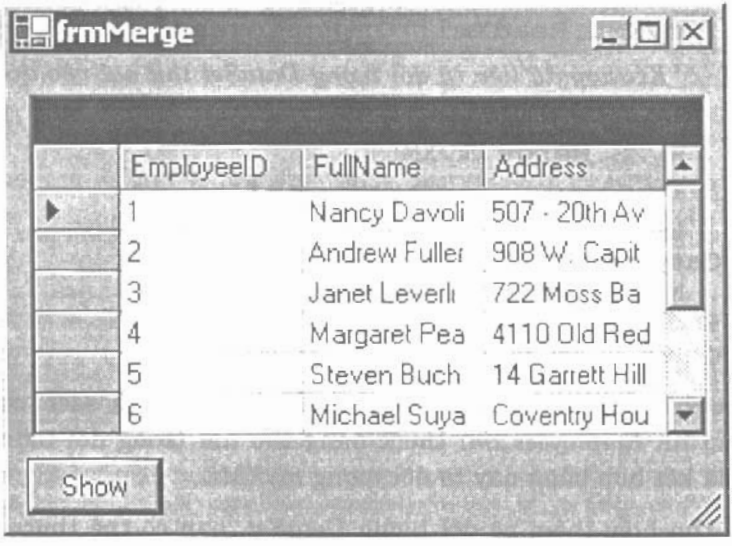
Để đọc dữ liệu từ tập tin *Xml*, bạn thêm *Form* và đặt tên *frmReadXml* như hình 18-10 vào *Project* và khai báo trong biến cố *Click* của nút *Show* như ví dụ 18-16:

Ví dụ 18-16: Đọc dữ liệu từ tập tin Xml

```
Private Sub Button1_Click(ByVal sender As _
System.Object, ByVal e As System.EventArgs) _
Handles Button1.Click
   Try
      Dim myDS As New DataSet
      'Đọc dữ liệu từ tập tin XML
      myDS.ReadXml("C:\Employees.xml")
      DataGrid1.DataSource = myDS.Tables(0)
   Catch ex As Exception
      MsgBox(ex.Message)
   End Try
End Sub
```

4.3. Phương thức Merge

Trong trường hợp cần kết hợp dữ liệu giữa nhiều đối tượng *DataSet* lại với nhau, hay kết hợp đối tượng *DataTable* vào đối tượng *DataSet*, bạn có thể sử dụng phương thức *Merge* của đối tượng *DataSet*.



Hình 18-11: *Kết hợp đối tượng DataSet*

Để làm điều này, bạn thêm *Form* vào *Project* với tên *frmMerge* như hình 18-11 và khai báo trong biến cố *Click* của nút *Merge* như ví dụ 18-17:

Ví dụ 18-17: Đọc dữ liệu từ tập tin Xml

```
Private Sub Button1_Click(ByVal sender As _
System.Object, ByVal e As System.EventArgs) _
Handles Button1.Click
    ' Khai báo và khởi tạo đối tượng clsDatabase
    Dim myCls As New clsDatabase(gsCon)
    ' Khai báo chuỗi SQL
    Dim strSQL As String
    strSQL = "Select CustomerID, CompanyName, "
    strSQL += " Address from Customers"
    Try
        ' Khai báo và khởi tạo đối tượng DataSet thứ nhất
        Dim myDS As New DataSet
        ' Điền danh sách khách hàng vào đối tượng DataSet thứ nhất
        myCls.getValue(myDS, strSQL)
        ' Khai báo và khởi tạo đối tượng DataSet thứ hai
        Dim myXML As New DataSet
        ' Điền danh sách nhân viên vào đối tượng DataSet thứ hai
```

```
        myXML.ReadXml("C:\Employees.xml")
        ' Kết hợp dữ liệu từ đối tượng DataSet thứ hai vào đối tượng
        ' DataSet thứ nhất
        myDS.Merge(myXML)
        ' Trình bày dữ liệu của bảng thứ hai
        DataGrid1.DataSource = myDS.Tables(1)
      Catch ex As Exception
        MsgBox(ex.Message)
      End Try
    End Sub
```

Lưu ý, trong ví dụ trên chúng ta trình bày danh sách nhân viên đọc từ tập tin *Employees.xml* thuộc bảng thứ hai trong đối tượng *myDS* sau khi đã kết hợp bảng này từ đối tượng *myXML*.

Để tìm hiểu thêm về đối tượng *DataSet*, bạn có thể thực hành các ví dụ sau:

1. Thiết kế *Form*, dùng để trình bày danh sách nhân viên trong bảng *Employees* từ cơ sở dữ liệu *Northwind* trên điều khiển *DataGrid*, khi người sử dụng nhấn nút *Write*, chương trình sẽ ghi tất cả mẩu tin có trong bảng *Employees* ra tập tin *XML*.
2. Tạo ứng dụng *Windows Forms*, dùng để trình bày danh sách nhân viên trong tập tin *XML* vừa tạo ra trong câu 1.

5. KẾT LUẬN

Bạn vừa tìm hiểu cách làm việc với hai đối tượng chính trong lớp không kết nối là *DataSet* và đối tượng bộ điều phối cơ sở dữ liệu *SqlDataAdapter*.

Trong chuyên đề kế tiếp, chúng ta tiếp tục tìm hiểu chi tiết hai đối tượng *DataTable, DataView*.

Chuyên đề 19:

ĐỐI TƯỢNG DATATABLE VÀ DATAVIEW

Tóm tắt chuyên đề 19

DataTable là thành phần chính của đối tượng DataSet, dùng để nắm giữ tập dữ liệu của một hay nhiều thực thể kết hợp thuộc cơ sở dữ liệu.

Bạn có thể sử dụng đối tượng DataView để trích lọc dữ liệu trên tập dữ liệu đang nắm giữ bởi đối tượng DataTable.

Sau khi người sử dụng thay đổi dữ liệu, nếu muốn cập nhật phần dữ liệu thay đổi trở lại cơ sở dữ liệu nguồn, bạn có thể sử dụng phương thức Update của đối tượng SqlDataAdapter.

Các vấn đề chính sẽ được đề cập:

- ✓ *Đối tượng DataTable.*
- ✓ *Đối tượng DataView.*
- ✓ *Cập nhật dữ liệu từ đối tượng DataTable.*

1. ĐỐI TƯỢNG DATATABLE

Đối tượng *DataTable* là phần tử chính trong đối tượng *DataSet*. Mỗi đối tượng *DataTable* sẽ nắm giữ một hay nhiều đối tượng *DataRow*. Đối tượng *DataTable* có thể có quan hệ với các đối tượng *DataTable* khác trong cùng đối tượng *DataSet*.

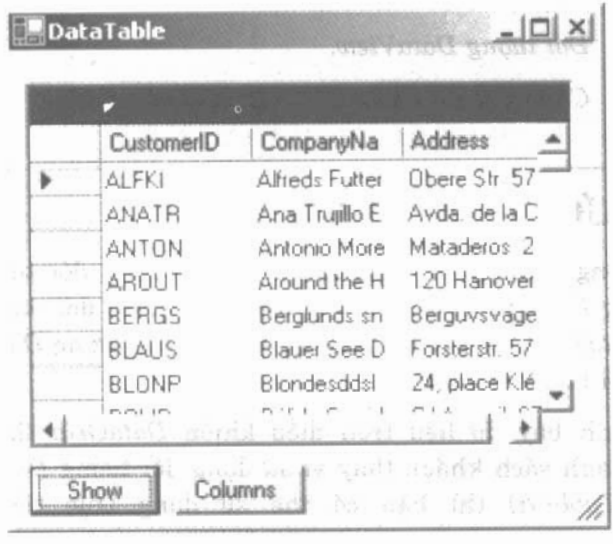
Khi trình bày dữ liệu trên điều khiển *DataGrid* (hoặc các điều khiển dạng danh sách khác), thay vì sử dụng đối tượng *DataSet* ứng với thuộc tính *Tables(i)* thì bạn có thể sử dụng trực tiếp đối tượng *DataTable*.

Ví dụ, bạn khai báo sử dụng đối tượng *DataTable* như ví dụ 19-1:

Ví dụ 19-1: Trình bày dữ liệu của đối tượng DataTable

```
Private Sub Button1_Click(ByVal sender As _
System.Object, ByVal e As System.EventArgs) _
Handles Button1.Click
   Dim myCls As New clsDatabase(gsCon)
   Dim strSQL As String
   strSQL = "Select CustomerID, CompanyName, "
      strSQL = " Address from Customers"
   Try
      Dim myDT As New DataTable
      'Gọi phương thức trong lớp clsDatabase
      myCls.getValue(myDT, strSQL)
      DataGrid1.DataSource = myDT
   Catch ex As Exception
      MsgBox(ex.Message)
   End Try
End Sub
```

Khi thực thi chương trình, nếu bạn nhấn nút *Show* thì kết quả trình bày như hình 19-1.



Hình 19-1: *Sử dụng đối tượng DataTable*

Chú ý, phương thức *getValue* sử dụng ở ví dụ trên được khai báo trong lớp *clsDatabase* đã trình bày trong chuyên đề 18.

1.1. Thuộc tính

Đối tượng *DataTable* cung cấp các thuộc tính như: *DataSet, HasErrors, TableName, ChildRelations, ParentRelations, PrimaryKey, Columns* và *Rows*.

1.1.1. Thuộc tính DataSet

Thuộc tính *DataSet* trả về đối tượng *DataSet* mà đối tượng *DataTable* trực thuộc.

1.1.2. Thuộc tính HasErrors

Tương tự như thuộc tính *HasErrors* trong đối tượng *DataSet*, thuộc tính *HasErrors* trả về *True* nếu có phát sinh lỗi trong hàng (đối tượng *DataRow)* dữ liệu.

1.1.3. Thuộc tính TableName

Thuộc tính *TableName* trả về tên của bảng dữ liệu, nếu đối tượng *DataTable* không khai báo tên thì tên của nó chính là tên bảng dữ liệu hay chuỗi *SQL*.

1.1.4. Thuộc tính ChildRelations, ParentRelations

Thuộc tính *ChildRelations* và *ParentRelations* trả về tập quan hệ cha hay con ứng với hai đối tượng *DataTable*.

1.1.5. Thuộc tính PrimaryKey

Thuộc tính *PrimaryKey* trả về một mảng bao gồm các cột khai báo như khóa chính (*Primary Key*).

1.1.6. Thuộc tính Columns

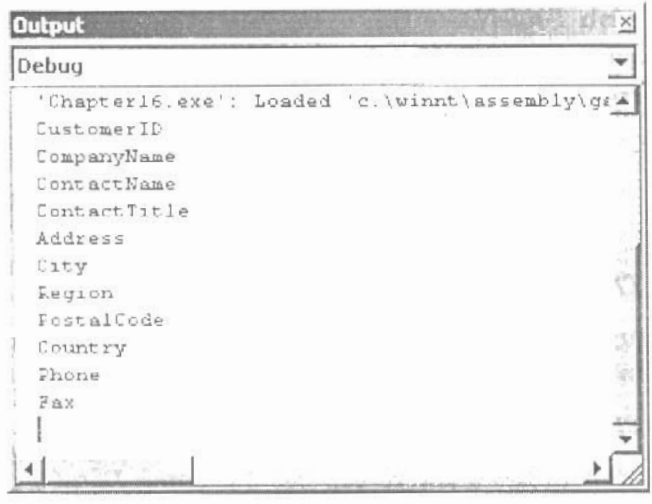
Thuộc tính *Column* trả về tập các đối tượng *DataColumn* của đối tượng *DataTable*. Chẳng hạn, bạn có thể in ra tên của các cột dữ liệu trong đối tượng *DataTable* bằng thuộc tính *Count* của *Columns Collection* như ví dụ 19-2:

Ví dụ 19-2: Tên cột dữ liệu

```
Private Sub Button2_Click(ByVal sender As _
System.Object, ByVal e As System.EventArgs) _
Handles Button2.Click
   Dim myCls As New clsDatabase(gsCon)
   Dim strSQL As String
```

```
    strSQL = "Select * from Customers"
    Try
      Dim myDT As New DataTable
      myCls.getValue(myDT, strSQL)
      ' Duyệt trên từng cột của đối tượng DataTable
      For i As Integer = 0 To myDT.Columns.Count - 1
        Console.WriteLine( _
        myDT.Columns(i).ColumnName)
      Next
    Catch ex As Exception
      MsgBox(ex.Message)
    End Try
  End Sub
```

Khi thực thi chương trình, nếu bạn nhấn nút *Columns* thì kết quả trả về danh sách tên các cột dữ liệu mà đối tượng *DataTable* đang nắm giữ như hình 19-2.



Hình 19-2: *Tên các cột dữ liệu*

1.1.7. Thuộc tính Rows

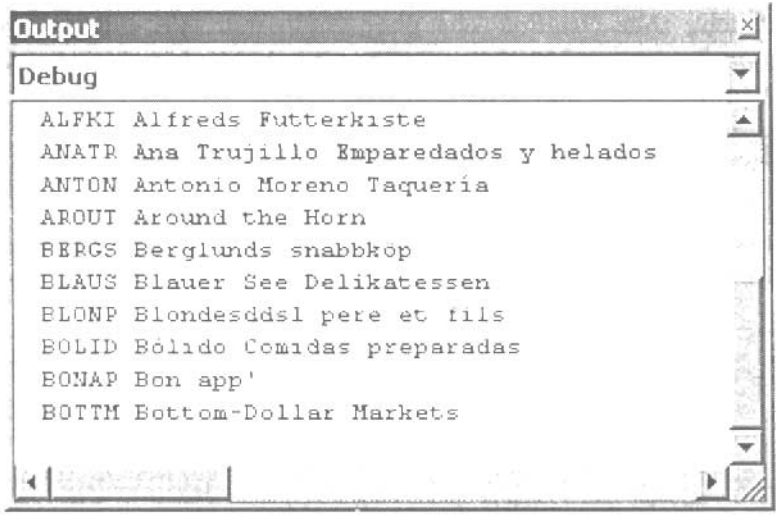
Ngược lại với thuộc tính *Columns* trả về tập đối tượng *DataColumn*, thuộc tính *Rows* trả về tập đối tượng *DataRow* của đối tượng *DataTable*.

Ví dụ, bạn có thể in ra giá trị của các cột dữ liệu của mỗi hàng trong đối tượng *DataTable* bằng cách sử dụng thuộc tính *Count* của *Rows Collection* như ví dụ 19-3:

Ví dụ 19-3: Số mẩu tin

```
Private Sub Button3_Click(ByVal sender As _
System.Object, ByVal e As System.EventArgs) _
Handles Button3.Click
    Dim myCls As New clsDatabase(gsCon)
    Dim strSQL As String
    strSQL = "Select top 10 * from Customers"
    Try
        Dim myDT As New DataTable
        myCls.getValue(myDT, strSQL)
        'Duyệt trên từng hàng
        For i As Integer = 0 To myDT.Rows.Count - 1
        'In giá trị cột thứ nhất
         Console.Write(myDT.Rows(i).Item(0) + " ")
        'In giá trị cột thứ hai
         Console.WriteLine(myDT.Rows(i).Item(1))
        Next
    Catch ex As Exception
        MsgBox(ex.Message)
    End Try
End Sub
```

Tương tự như trong trường hợp thuộc tính *Columns*, khi thực thi chương trình, nếu bạn nhấn nút *Rows* thì kết quả trả về như hình 19-3.



Hình 19-3: *Giá trị cột thứ 0*

1.2. Phương thức

Ngoài các thuộc tính vừa trình bày ở trên, đối tượng *DataTable* cung cấp các phương thức như: (*AcceptChanges*) hay từ chối (*RejectChanges*) dữ liệu thay đổi, trích lọc (*GetChanges*) dữ liệu thay đổi, nạp danh sách đối tượng *DataRow* (*ImportRow*) và thêm hàng dữ liệu (*NewRow*).

1.2.1. Phương thức AcceptChanges

Chấp nhận sự thay đổi dữ liệu trên đối tượng *DataTable*.

1.2.2. Phương thức Clear

Khi có nhu cầu loại bỏ tất cả mẩu tin trong các đối tượng *DataTable*, bạn có thể sử dụng phương thức *Clear*.

1.2.3. Phương thức RejectChanges

Trong trường hợp bạn không chấp nhận sự thay đổi dữ liệu trên đối tượng *DataTable* thì chọn phương thức *RejectChanges*.

1.2.4. Phương thức GetChanges

Phương thức *GetChanges* trả về một đối tượng *DataTable* chứa đựng các đối tượng *DataRow* mà dữ liệu có thay đổi, thêm mới hay xóa.

1.2.5. Phương thức NewRow

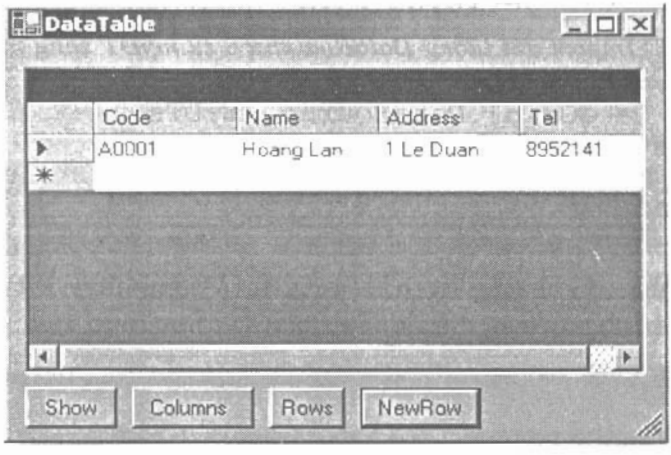
Phương thức *NewRow* cho phép thêm mới một *DataRow* vào đối tượng *DataTable*, sau khi thêm mới bạn có thể định nghĩa tên của từng cột dữ liệu tương ứng như ví dụ 19-4.

Ví dụ 19-4: Thêm mẩu tin vào đối tượng DataTable

```
Private Sub Button1_Click(ByVal sender As _
System.Object, ByVal e As System.EventArgs) _
Handles Button4.Click
  Try
    Dim myDT As New DataTable
    ' Khai báo tên cột dữ liệu
    myDT.NewRow()
    myDT.Columns.Add("Code")
    myDT.Columns.Add("Name")
    myDT.Columns.Add("Address")
    myDT.Columns.Add("Tel")
    Dim myRow As DataRow
```

```
        ' Thêm dữ liệu vào hàng
        myRow = myDT.NewRow()
        myRow.Item(0) = "A0001"
        myRow.Item(1) = "Hoang Lan"
        myRow.Item(2) = "1 Le Duan"
        myRow.Item(3) = "8952141"
        myDT.Rows.Add(myRow)
        DataGrid1.DataSource = myDT
    Catch ex As Exception
        MsgBox(ex.Message)
    End Try
End Sub
```

Khi thực thi chương trình, nếu người sử dụng nhấn nút *NewRow*, lập tức dữ liệu trình bày trên điều khiển *DataGrid* như hình 19-4.



Hình 19-4: *Thêm hàng dữ liệu*

1.2.6. Phương thức ImportRow

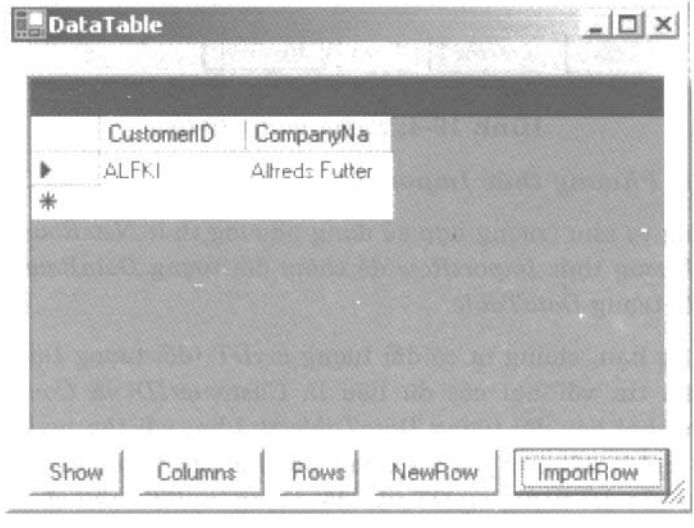
Tương tự như trường hợp sử dụng phương thức *NewRow*, bạn có thể sử dụng phương thức *ImportRow* để thêm đối tượng *DataRow* đang có dữ liệu vào đối tượng *DataTable*.

Chẳng hạn, chúng ta có đối tượng *myDT* (đối tượng *DataTable*) lưu trữ 91 mẩu tin với hai cột dữ liệu là *CustomerID* và *CompanyName*. Bằng cách khai báo đối tượng *DataTable* thứ hai với tên *myDTs*, sau đó, bạn định nghĩa tên cột và sử dụng phương thức *ImportRow* để thêm hàng thứ 0 của đối tượng *myDT* vào đối tượng *myDTs* như ví dụ 19-5:

Ví dụ 19-5: Import dữ liệu

```
Private Sub Button5_Click(ByVal sender As _
System.Object, ByVal e As System.EventArgs) _
Handles Button5.Click
   Dim myCls As New clsDatabase(gsCon)
   Dim strSQL As String
   strSQL = "Select CustomerID, CompanyName "
      strSQL += " from Customers"
   Try
      Dim myDT As New DataTable
      Dim myDTs As New DataTable
      myDTs.NewRow()
      myDTs.Columns.Add("CustomerID")
      myDTs.Columns.Add("CompanyName")
      myCls.getValue(myDT, strSQL)
      ' Import đối tượng DataRow thứ 0 từ myDT sang myDTs
      myDTs.ImportRow(myDT.Rows(0))
      DataGrid1.DataSource = myDTs
   Catch ex As Exception
      MsgBox(ex.Message)
   End Try
End Sub
```

Khi thực thi chương trình, nếu người sử dụng nhấn nút *ImportRow* thì dữ liệu trình bày trên điều khiển *DataGrid* như hình 19-5.



Hình 19-5: *Import đối tượng DataRow*

Để tìm hiểu thêm đối tượng *DataTable*, bạn có thể thực hành các ví dụ sau:

1. Khai báo phương thức, nhận một phát biểu *SQL* là câu lệnh *Select* sau đó trả về một đối tượng *DataTable*, tạo ứng dụng *Form* để sử dụng đối tượng *DataTable* này.
2. Khai báo *Class*, bao gồm phương thức nhận phát biểu *SQL* dạng *Select* trả về mảng một chiều có kiểu *ArrayList*, sau đó thiết kế *Form* truyền phát biểu *Select* tương ứng và trình bày dữ liệu trên điều khiển *ListView* từ phương thức trên.
3. Thiết kế *Form*, với các *TextBox* cho phép người sử dụng nhập mã, tên, địa chỉ, và điện thoại của nhân viên sau đó trình bày chúng trên *DataGrid* của cùng *Form*.
4. Tạo *Project*, khai báo một *Stored Procedure* để liệt kê danh sách mẩu tin trong bảng *tblCustomers* đã được tạo ra trong phần lý thuyết, sau đó cho phép người sử dụng thêm mới, xóa, thay đổi dữ liệu và cập nhật lại dữ liệu nguồn.

2. ĐỐI TƯỢNG DATAVIEW

Đối tượng *DataView* cho phép trình bày dữ liệu trong đối tượng *DataTable* với các tính năng như: kết hợp (*databindable*), sắp xếp (*sorting*), lọc (*filtering*), tìm kiếm (*searching*), kích hoạt (*editing*) và điều hướng (*navigation*).

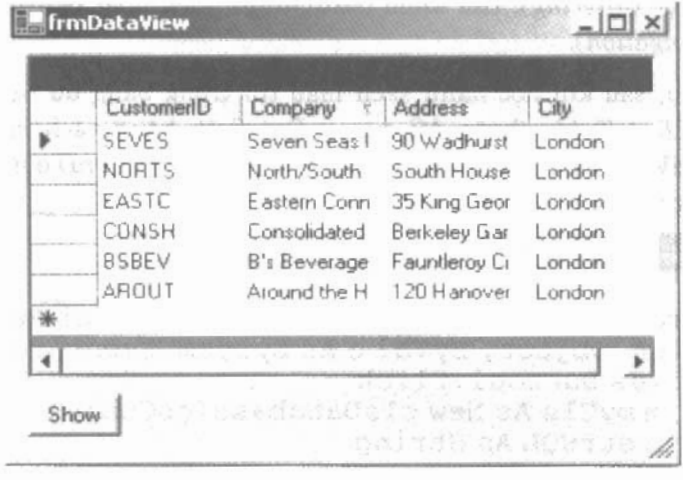
Ví dụ, sau khi đọc danh sách mẩu tin trong bảng dữ liệu điền vào đối tượng *DataTable* (hoặc đối tượng *DataSet*), bằng cách sử dụng đối tượng *DataView* bạn có thể lọc và sắp xếp dữ liệu từ *DataTable* trước khi điền chúng vào điều khiển *DataGrid* như ví dụ 19-6:

Ví dụ 19-6: Lọc dữ liệu bằng đối tượng DataView

```
Private Sub Button1_Click(ByVal sender As _
System.Object, ByVal e As System.EventArgs) _
Handles Button1.Click
   Dim myCls As New clsDatabase(gsCon)
   Dim strSQL As String
   strSQL = "Select CustomerID, CompanyName, "
      strSQL += "Address,City from Customers"
   Try
      Dim myDT As New DataTable
```

```
        ' Điền dữ liệu vào đối tượng DataTable
        myCls.getValue(myDT, strSQL)
        ' Khai báo và khởi tạo đối tượng DataView
        Dim myDW As New DataView(myDT)
        With myDW
            ' Lọc dữ liệu
            .RowFilter = "City = 'London'"
            ' Sắp xếp dữ liệu
            .Sort = "CompanyName DESC"
        End With
        ' Điền dữ liệu vào điều khiển DataGrid
        DataGrid1.DataSource = myDW
    Catch ex As Exception
        MsgBox(ex.Message)
    End Try
End Sub
```

Trong ví dụ trên, sau khi điền dữ liệu từ bảng *Customers* vào đối tượng *DataTable* thông qua phương thức *getValue* của đối tượng *clsDatabase*, bằng cách sử dụng đối tượng *DataView* bạn có thể trình bày những khách hàng có giá trị trong cột *City* là *London*, kết quả trả về như hình 19-6.



Hình 19-6: *Lọc dữ liệu theo thành phố London*

Để tìm hiểu thêm đối tượng *DataView*, bạn có thể thực hành các ví dụ sau:

1. Thiết kế *Form*, dùng đối tượng *DataView* liệt kê danh sách *Country* trên điều khiển *ComboBox*, mỗi khi người sử dụng chọn *Country* thì bạn liệt kê danh sách khách hàng có *Country* đó trên điều khiển *DataGrid*.
2. Tạo ứng dụng *Windows Forms*, dùng đối tượng *DataView* cho phép người sử dụng chọn loại sản phẩm trên điều khiển *TreeView*, lập tức bạn liệt kê danh sách sản phẩm cùng với số lượng còn trong kho, tổng số lượng bán và số lượng đặt hàng trên điều khiển *ListView* thứ nhất và chi tiết của sản phẩm đó trên điều khiển *ListView* thứ hai
3. Thiết kế *Form*, dùng đối tượng *DataView* cho phép người sử dụng chọn *Country* trên điều khiển *ComboBox* thứ nhất và nhân viên trên điều khiển *ComboBox* thứ hai, rồi liệt kê danh sách sản phẩm do nhân viên đó thực hiện thuộc *Country* đã chọn.

3. CẬP NHẬT DỮ LIỆU TỪ ĐỐI TƯỢNG DATATABLE

Để thực hiện việc cập nhật dữ liệu mà người sử dụng đã thay đổi từ đối tượng *DataTable*, trước tiên bạn khai báo đoạn chương trình để nạp danh sách mẩu tin của bảng *tblCustomers* vào điều khiển *DataGrid* như ví dụ 19-7.

Ví dụ 19-7: Điền dữ liệu vào DataGrid

```
Sub LoadData()
    myDT.Clear()
    Dim myCon As New SqlConnection(gsCon)
    Try
      myCon.Open()
      strSQL = "select * from tblCustomers"
      myData = New SqlDataAdapter(strSQL, myCon)
      myData.Fill(myDT)
      Me.DataGrid1.DataSource = myDT
    Catch ex As Exception
      MsgBox(ex.Message)
    End Try
End Sub
```

Kế đến, bạn khai báo để gọi phương thức *LoadData* trong biến cố *Load* của *Form* như ví dụ 19-8.

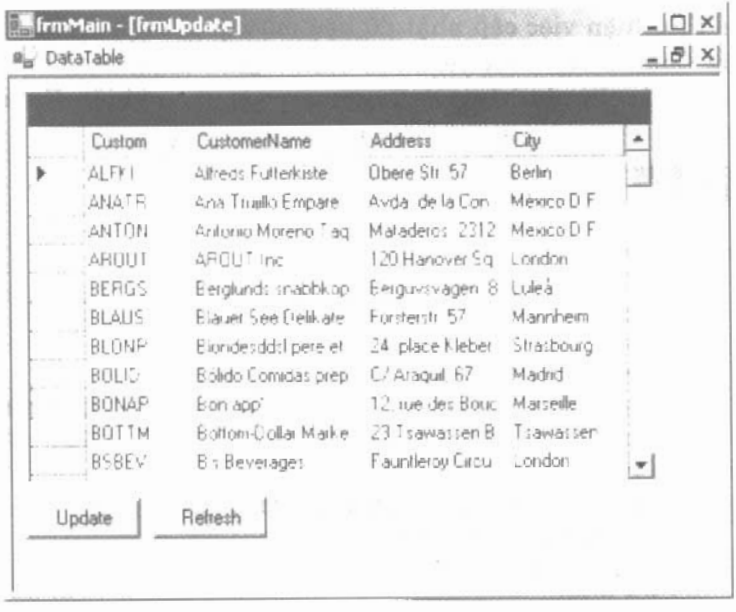
Ví dụ 19-8: Gọi phương thức Load Data

```
Private Sub frmUpdate_Load(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles MyBase.Load
    LoadData()
End Sub
```

Lưu ý, bạn cần khai báo hai biến đối tượng *SqlDataAdapter* và *DataTable* sử dụng chung cho các phương thức trong *Class* có tên *frmUpdate* như sau:

```
Dim myDT As New DataTable
Dim myData As SqlDataAdapter
Dim strSQL As String
```

Sau khi nạp dữ liệu lên điều khiển *DataGrid* từ đối tượng *DataTable* như hình 19-7, nếu người sử dụng thay đổi dữ liệu trên *DataGrid* và nhấn nút *Update* thì lập tức dữ liệu đó sẽ được cập nhật trở lại dữ liệu nguồn.



Hình 19-7: *Liệt kê dữ liệu*

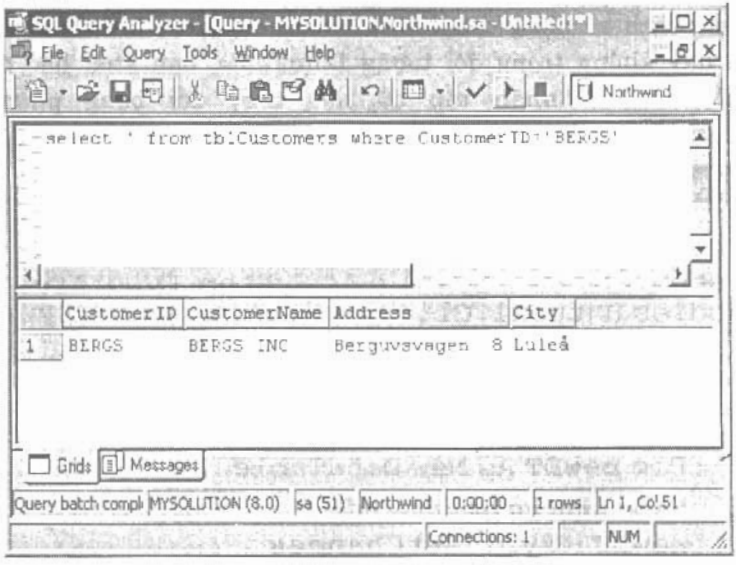
Giả sử, bạn thay đổi tên của khách hàng có mã là *BERGS* thành *BERGS INC* rồi nhấn nút *Update*.

Bằng cách sử dụng thuộc tính *HasChanges* để kiểm tra dữ liệu có thay đổi hay không trong đối tượng *DataTable*, bạn khai báo đối tượng *DataTable* mới để nhận tập dữ liệu thay đổi bằng phương thức *GetChanges* như ví dụ 19-9:

Ví dụ 19-9: Cập nhật dữ liệu

```
Private Sub Button1_Click(ByVal sender As _
System.Object, ByVal e As System.EventArgs) _
Handles Button1.Click
    ' Cập nhật DataGrid
    DataGrid1.Update()
        ' Khai báo một đối tượng DataTable
        Dim newDT As New DataTable
        ' Nhận tập dữ liệu thay đổi
        newDT = myDT.GetChanges
        ' Nếu dữ liệu thay đổi
        If Not newDT Is Nothing Then
        ' Tạo ra các phát biểu SQL tương ứng với thuộc tính
        ' UpdateCommand, DeleteCommand và InsertCommand cho
        ' đối tượng SqlDataAdapter bằng đối tượng
        ' SqlCommandBuilder
        Dim myBuilder As New _
            SqlCommandBuilder(myData)
        ' Sử dụng thuộc tính TableMappings để ánh xạ tên Table
        myData.TableMappings.Add("Table", _
            myDT.TableName)
        ' Cập nhật newDT vào dữ liệu nguồn bằng phương thức
        ' Update của đối tượng SqlDataAdapter
        myData.Updace(newDT)
        ' Nạp lại dữ liệu
        LoadData()
    End If
End Sub
```

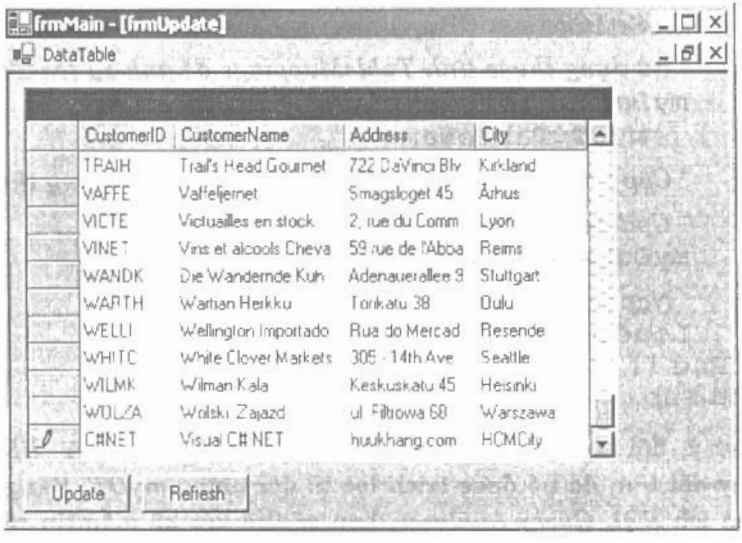
Chú ý, đối tượng *newDT* chỉ nắm giữ mẩu tin có mã *BERGS* đã được thay đổi tên, do nó được trích lọc từ đối tượng *myDT*. Bằng cách sử dụng tiện ích *SQL Query Analyzer*, bạn có thể liệt kê mẩu tin của khách hàng này như hình 19-8.



Hình 19-8: *Kiểm tra tên của khách hàng*

Nếu người sử dụng thêm mẫu tin trên điều khiển *DataGrid* và nhấn nút *Update*, những mẫu tin đã thay đổi cũng được thêm vào bảng *tblCustomers*.

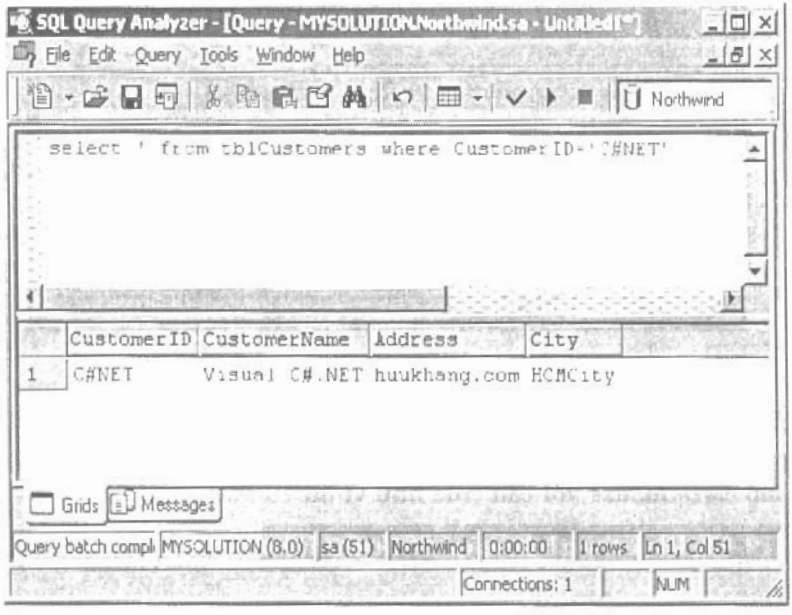
Giả sử, ban thêm mẫu tin có giá trị tương ứng các cột *C#NET*, *Visual C# .NET Example*, *HuuKhang.com*, *HCMCity* như hình 19-9.



Hình 19-9: *Thêm mẫu tin vào DataTable*

Lưu ý, trong ví dụ trên đối tượng *newDT* chỉ nắm giữ mẩu tin vừa thêm vào, sau đó đối tượng *SqlDataAdapter* sẽ thêm mẩu tin này vào bảng *tblCustomers* của cơ sở dữ liệu nguồn.

Bằng cách nhấn nút *Update*, trở lại tiện ích *SQL Query Analyzer*, bạn có thể liệt kê mẩu tin của khách hàng có mã *C#NET* như hình 19-10.



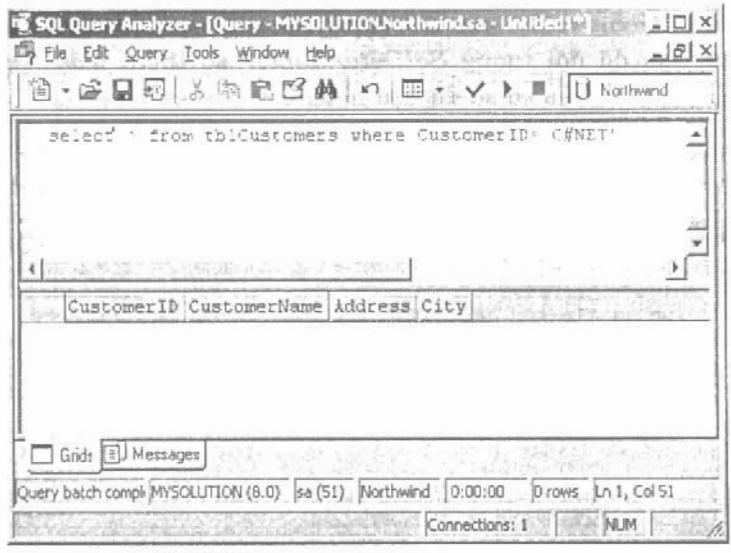
Hình 19-10: *Thêm mẩu tin*

Đối với trường hợp xóa mẩu tin trên điều khiển *DataGrid* thì thực hiện tương tự, bằng cách chọn vào mẩu tin và nhấn phim *Delete*, sau đó nhấn nút *Update* lập tức mẩu tin đó sẽ bị xóa khỏi bảng *tblCustomers*.

Để kiểm tra kết quả, bạn chọn vào mẩu tin có mã là *C#NET* vừa thêm trong ví dụ trên, sau khi nhấn phim *Delete*, mẩu tin này sẽ xóa khỏi đối tượng *DataTable* (chưa có hiệu lực trong cơ sở dữ liệu).

Nếu bạn tiếp tục nhấn nút *Update*, mẩu tin này sẽ bị xóa khỏi bảng *tblCustomers*, kiểm tra bằng cách sử dụng tiện ích *SQL Query Analyzer*, bạn có thể không tìm mẩu tin của khách hàng có mã *C#NET* như hình 19-11.

Trong phần trình bày ở trên, chúng ta khai báo đọc dữ liệu trực tiếp trong phương thức *LoadData* và cập nhật trở lại dữ liệu nguồn trực tiếp từ *Form*.



Hình 19-11: *Xóa mẩu tin*

Bây giờ, chúng ta viết lại hai đoạn chương trình dùng để trình bày và cập nhật dữ liệu từ lớp *clsDatabase*, khai báo phương thức *doUpdate* trong lớp *clsDatabase* với cấu trúc như ví dụ 19-10.

Ví dụ 19-10: Phương thức cập nhật dữ liệu

```
Function doUpdate(ByVal myDT As DataTable,
ByVal strSQL As String) As String
   Dim strError As String = ""
   'Tồn tại dữ liệu mà người sử dụng thay đổi
   If Not myDT Is Nothing Then
     Dim myCon As New SqlConnection(gsCon)
     Try
       myCon.Open()
       Dim myData As New SqlDataAdapter(strSQL, _
       myCon)
       'Sử dụng đối tượng SqlCommandBuilder
       Dim myBuilder As New _
       SqlCommandBuilder(myData)
       'Định nghĩa tên bảng dữ liệu
       myData.TableMappings.Add("Table", _
         myDT.TableName)
       'Gọi phương thức Update
```

```
            myData.Update(myDT)
        Catch ex As Exception
            strError = ex.Message
        Finally
            myCon.Close()
            myCon.Dispose()
        End Try
    End If
    Return strError
End Function
```

Kế đến, bạn thêm *Form* vào *Project* và đặt tên *frmUpdateFromClass* như hình 19-12.

Hình 19-12: *Cập nhật dữ liệu*

Bằng cách khai báo phương thức *LoadData* để nạp dữ liệu trên điều khiển *DataGrid* như ví dụ 19-11.

Ví dụ 19-11: Điền dữ liệu vào điều khiển DataGrid

```
Sub LoadData()
    myDT.Clear()
    strSQL = "select * from tblCustomers"
    ' Khai báo và khởi tạo đối tượng clsDatabase
    Dim cls As New clsDatabase(gsCon)
    ' Gọi phương thức getValue
```

```
        cls.getValue(myDT, strSQL)
        Me.DataGrid1.DataSource = myDT
    End Sub
```

Sau đó, khai báo trong biến cố *Click* của nút *Update* để gọi phương thức *doUpdate* để cập nhật dữ liệu mà người sử dụng thay đổi trên điều khiển *DataGrid* như ví dụ 19-12.

Ví dụ 19-12: Gọi phương thức doUpdate

```
Private Sub Button1_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles Button1.Click
    DataGrid1.Update()
    Dim cls As New clsDatabase(gsCon)
    ' Gọi phương thức doUpdate với dữ liệu thay đổi
    cls.doUpdate(myDT.GetChanges, strSQL)
    LoadData()
End Sub
```

Để tìm hiểu thêm về cách cập nhật, xóa. thêm mẩu tin vào đối tượng *DataTable*, bạn có thể thực hành các ví dụ sau:

1. Thiết kế *Form* để điền dữ liệu của bảng nào đó trong cơ sở dữ liệu *Northwind*, sau đó cho phép người sử dụng thêm mới, thay đổi hay xóa mẩu tin rồi cập nhật trở lại dữ liệu nguồn.
2. Tạo ứng dụng *Windows Forms*, cho phép liệt kê danh sách mẩu tin của bảng dữ liệu *tblCustomers*, sau khi người sử dụng thay đổi dữ liệu và nhấn nút *Changed*, lập tức danh sách các mẩu tin thay đổi sẽ được trình bày trên điều khiển *DataGrid* khác. Nếu người sử dụng tiếp tục nhấn nút *Update* thì tất cả mẩu tin đó sẽ cập nhật vào cơ sở dữ liệu nguồn.

4. KẾT LUẬN

Bạn vừa tìm hiểu cách làm việc với hai đối tượng chính *DataTable* và *DataView* trong đối tượng *DataSet* thuộc lớp không kết nối là *DataSet*.

Trong chuyên đề kế tiếp, chúng ta tiếp tục tìm hiểu chi tiết đối tượng *DataColumn*, *DataRow* và *DataRelation* thuộc đối tượng *DataTable*.

Chuyên đề 20:

ĐỐI TƯỢNG DATAROW, DATACOLUMN VÀ DATARELATION

Tóm tắt chuyên đề 20

Bạn đã tham khảo ví dụ và bài tập của đối tượng DataSet, DataTable và DataView trong hai chuyên đề trước, trong chuyên đề này, chúng ta tiếp tục tìm hiểu cách làm việc với các đối tượng DataRow, DataColumn là các thành phần cấu tạo nên đối tượng DataTable và đối tượng DataRelation tạo nên mối quan hệ giữa các đối tượng DataTable trong đối tượng DataSet.

Các vấn đề chính sẽ được đề cập:

- ✓ *Đối tượng DataRow.*
- ✓ *Đối tượng DataColumn.*
- ✓ *Đối tượng DataRelation.*

1. ĐỐI TƯỢNG DATAROW

DataRow là đối tượng cấu tạo nên đối tượng *DataTable*. Bằng cách sử dụng đối tượng này, bạn có thể khai báo để thêm từng mẩu tin vào đối tượng *DataTable*.

1.1. Thuộc tính

Đối tượng *DataRow* cung cấp các thuộc tính thường được sử dụng như: *Item*, *ItemArray*, *Table*.

1.1.1. Thuộc tính Item

Thuộc tính *Item* cho phép gán hay lấy giá trị của cột trong hàng dữ liệu ứng với chỉ mục cột tính từ 0 hay tên cột chỉ định.

1.1.2. Thuộc tính ItemArray

Khi bạn muốn lấy mảng một chiều kiểu *object*, mỗi phần tử trong mảng ứng với giá trị của từng cột thuộc hàng tương ứng.

1.1.3. Thuộc tính Table

Thuộc tính này trả về đối tượng *DataTable* mà đối tượng *DataRow* trực thuộc.

1.1.4. Thuộc tính IsNull

Thuộc tính này trả về *True* nếu cột chỉ định bởi chỉ mục hay tên cụ thể có giá trị *Null*.

Lưu ý, bạn có thể sử dụng thuộc tính *Count* của *Rows Collection* thuộc đối tượng *DataTable* để biết được tổng số mẩu tin có trong đối tượng *DataTable* đó.

1.2. Phương thức

Ngoài các thuộc tính vừa trình bày ở trên, đối tượng *DataRow* cung cấp một số phương thức thường sử dụng như: *AcceptChanges*, *Delete*, *RejectChanges*.

1.2.1. Phương thức AcceptChanges

Chấp nhận sự thay đổi dữ liệu trên đối tượng *DataRow*.

1.2.2. Phương thức Delete

Khi có nhu cầu loại bỏ tất cả mẩu tin trong các đối tượng *DataRow*, bạn có thể sử dụng phương thức này. Tuy nhiên, nếu loại bỏ đối tượng *DataRow* trong *DataTable* thì bạn dùng phương thức *Remove* hay *RemoveAt* của *Rows Collection* như sau:

```
myDT.Rows.RemoveAt(i)
```

1.2.3. Phương thức RejectChanges

Trong trường hợp bạn không chấp nhận sự thay đổi dữ liệu trên đối tượng *DataRow* thì chọn phương thức này.

Chẳng hạn, thêm *Form* vào *Project* với tên *frmDataRow* cùng với các điều khiển trên *Form* cho phép người sử dụng thêm mới dữ liệu vào *DataGrid*, kế đến bạn khai báo đối tượng *DataTable* như sau:

```
Dim myDT As New DataTable
```

Khởi tạo tên các cột dữ liệu cho đối tượng *DataTable* bằng cách khai báo trong biến cố *Load* của *Form* như ví dụ 20-1.

Ví dụ 20-1: Định nghĩa tên cột dữ liệu

```
Private Sub frmDataRow_Load(ByVal sender As _
System.Object, ByVal e As System.EventArgs) _
Handles MyBase.Load
```

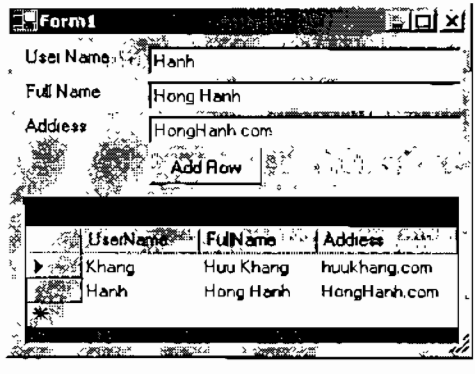
```
    myDT.Columns.Add("UserName")
    myDT.Columns.Add("FullName")
    myDT.Columns.Add("Address")
End Sub
```

Để thêm dữ liệu từ điều khiển *TextBox* vào đối tượng *DataTable*, bạn sử dụng đối tượng *DataRow* như ví dụ 20-2:

Ví dụ 20-2: Thêm dữ liệu từ điều khiển TextBox

```
Private Sub btnAdd_Click(ByVal sender As _
System.Object, ByVal e As System.EventArgs) _
Handles btnAdd.Click
    ' Khai báo sử dụng đối tượng DataRow
    Dim myDR As DataRow
    ' Tạo mới hàng trả về đối tượng DataRow
    myDR = myDT.NewRow
    ' Gán giá trị cho từng Column
    myDR.Item(0) = txtUserName.Text
    myDR.Item(1) = txtFullName.Text
    myDR.Item(2) = txtAddress.Text
    ' Thêm đối tượng DataRow vào đối tượng DataTable
    myDT.Rows.Add(myDR)
    ' Gán giá trị DataTable cho điều khiển DataGrid
    DataGrid1.DataSource = myDT
End Sub
```

Sau khi người sử dụng nhập dữ liệu vào điều khiển *TextBox* và nhấn nút *Add Row*, lập tức mẩu tin đó được thêm vào *DataGrid* như hình 20-1.



Hình 20-1: *Thêm dữ liệu vào đối tượng DataRow*

Để tìm hiểu thêm về đối tượng *DataRow*, bạn có thể thực hành các ví dụ sau:

1. Viết phương thức nhận phát biểu *Select* và trả về đối tượng ứng với mẩu tin đầu tiên, sau đó sử dụng phương thức này để điền dữ liệu lên các điều khiển tương ứng.
2. Thiết kế *Form*, cho phép người sử dụng nhập dữ liệu trên các điều khiển, sau đó trình bày dữ liệu đó trên *DataGrid*, khi người sử dụng nhấn nút *Save*, bạn lưu dữ liệu đó xuống cơ sở dữ liệu nguồn.
3. Viết ứng dụng *Windows Forms*, dùng để liệt kê tên tập tin, dung lượng, ngày tạo ra của thư mục chỉ định, sau đó điền dữ liệu đó vào điều khiển *DataGrid*.
4. Viết phương thức nhận phát biểu *Select* trả về mảng một chiều kiểu *object*, trong đó mỗi phần tử là một mảng một chiều kiểu *object* tương ứng với một hàng, rồi sử dụng phương thức này trong một *Form* để điền dữ liệu lên điều khiển *DataGrid* lấy từ mảng trên.

2. ĐỐI TƯỢNG DATACOLUMN

Đối tượng *DataRow* là thành phần của đối tượng *DataTable* thì đối tượng *DataColumn* cấu tạo nên đối tượng *DataRow*.

2.1. Thuộc tính

DataColumn cung cấp các thuộc tính cho phép bạn khai báo tên đối tượng chứa đựng dữ liệu bao gồm: Tên (*ColumnName*), kiểu dữ liệu (*DataType*), cho phép giá trị *null* (*AllowDBNull*), số tự động (*AutoIncrement*), số bắt đầu (*AutoIncrementSeed*), số nhảy (*AutoIncrementStep*), tựa đề (*Caption*), giá trị mặc định (*DefaultValue*), chiều dài (*MaxLength*) và giá trị duy nhất (*Unique*).

2.1.1. Thuộc tính ColumnName

Thuộc tính này cho phép bạn định nghĩa tên của cột dữ liệu nếu không khai báo tên trong khi khởi tạo đối tượng *DataColumn*. Chẳng hạn, bạn có thể khai báo một trong hai cách sau:

```
Dim myCol As DataColumn
          myCol = New DataColumn("UserName")
```

Hoặc sử dụng thuộc tính *ColumnName*

```
Dim myCol As DataColumn
myCol = New DataColumn()
myCol.ColumnName="UserName"
```

2.1.2. Thuộc tính AllowDBNull

Mặc định giá trị của thuộc tính này là *True* nghĩa là chấp nhận giá trị *Null*, nếu không cho phép giá trị *Null* cho cột dữ liệu thì khai báo giá trị *False* cho thuộc tính *AllowDBNull*.

2.1.3. Số tự động (AutoIncrement, AutoIncrementSeed, AutoIncrementStep)

Trong trường hợp muốn khai báo cột có số tự động tương tự như số tự động trong *SQL Server*, bạn có thể sử dụng thuộc tính này như sau:

```
Dim myCol As DataColumn
            myCol = New DataColumn("No")
' Cột No là cột số tự động
myCol.AutoIncrement=True
' Tự động bắt đầu số 1
myCol.AutoIncrementSeed=1
'Mỗi lần tăng lên 1
myCol.AutoIncrementStep=1
```

2.1.4. Thuộc tính Caption

Mặc định chuỗi trình bày trên tựa đề của mỗi cột là tên của cột đó. Tuy nhiên, nếu muốn xuất hiện với tên khác thì bạn khai báo chuỗi trong thuộc tính *Caption*. Ví dụ, tên cột dữ liệu là *OrderNumber* nhưng trình bày trên *DataGrid* là *No*.

```
Dim myCol As DataColumn
myCol = New DataColumn("OrderNumber ")
myCol.Caption="No"
```

2.1.5. Thuộc tính DefaultValue

Thuộc tính *DefaultValue* cho phép bạn gán giá trị mặc định cho cột dữ liệu khi mẩu tin thêm mới không cung cấp giá trị cho cột này.

```
myCol.DefaultValue=20
```

2.1.6. Thuộc tính MaxLength

Nếu muốn giới hạn chiều dài của chuỗi nhập vào, bạn có thể sử dụng thuộc tính *MaxLength*. Chẳng hạn, chiều dài của cột dữ liệu *Address* thường sử dụng khoảng 100 ký tự thì khai báo như sau:

```
myCol.MaxLength=100
```

2.1.7. Thuộc tính Unique

Trong trường hợp dữ liệu nhập vào cột không cho phép lặp lại, bạn sử dụng ràng buộc bằng cách khai báo thuộc tính *Unique* bằng *True*.

```
myCol.Unique=True
```

2.2. Phương thức

Đối tượng *DataColumn* cung cấp phương thức *GetType* (trả về kiểu *System.Type*) để lấy kiểu dữ liệu của cột.

```
Dim myType As System.Type =myCol.GetType()
```

Tuy nhiên, nếu muốn khai báo cột dữ liệu trong đối tượng *DataTable* bạn sử dụng phương thức *Add* của *Columns Collection* như sau:

```
myDT.Columns.Add("ColumnName")
myDT.Columns.Add("ColumnName","DataType")
```

Sau đó, sử dụng các thuộc tính khác như thuộc tính vừa trình bày ở trên cho một cột chỉ định, chẳng hạn như khai báo sau:

```
myDT.Columns.Add("Address")
myDT.Columns.Add("Age")
myDT.Columns(1).DefaultValue=20
myDT.Columns(1). DataType = _
Type.GetType("System.Int32")
```

Để sử dụng một số thuộc tính của đối tượng *DataColumn*, trước tiên bạn thêm *Form* vào *Project* với tên *frmDataColumn*, kế đến khai báo đối tượng *DataTable*:

```
Dim myDT As New DataTable
```

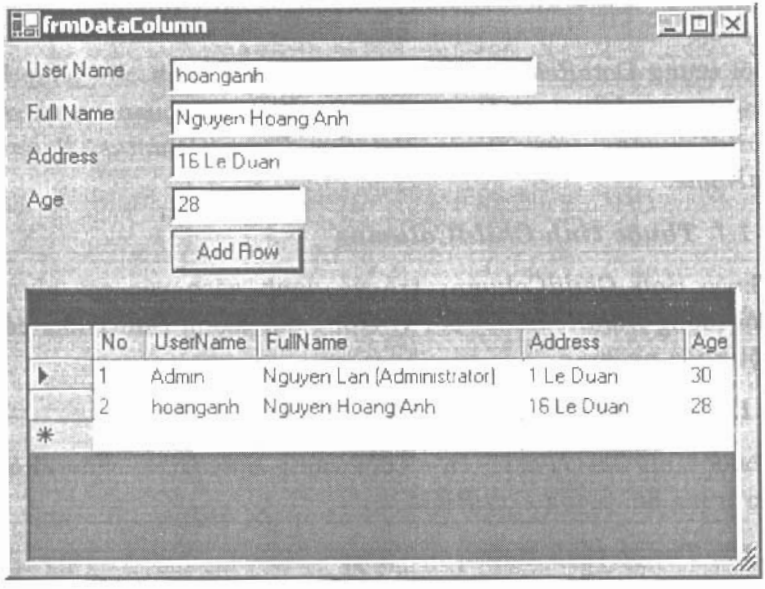
Kế đến, bạn khai báo để định nghĩa 5 cột dữ liệu tương ứng với số tự động, tài khoản, tên, địa chỉ và tuổi trong biến cố *Load* của *Form* như ví dụ 20-3

Ví dụ 20-3: Khai báo đối tượng DataColumn

```
Private Sub frmDataRow_Load(ByVal sender As _
System.Object, ByVal e As System.EventArgs) _
Handles MyBase.Load
    ' Khai báo đối tượng DataColumn
    Dim myCol As DataColumn
    ' Khai báo cột thứ nhất
    myCol = New DataColumn("OrderNumber")
    With myCol
        ' Khai báo thuộc tính Caption
        .Caption = "No"
        ' Khai báo thuộc tính kiểu dữ liệu số nguyên
        .DataType = Type.GetType("System.Int32")
        ' Khai báo thuộc tính số tự động
        .AutoIncrement = True
        .AutoIncrementSeed = 1
        .AutoIncrementStep = 1
    End With
    ' Thêm cột thứ nhất vào đối tượng DataTable
    myDT.Columns.Add(myCol)
    ' Khai báo cột thứ hai
    myCol = New DataColumn
    With myCol
        ' Khai báo thuộc tính ColumnName
        .ColumnName = "UserName"
        ' Khai báo kiểu dữ liệu chuỗi
        .DataType = Type.GetType("System.String")
        ' Dữ liệu nhập duy nhất
        .Unique = True
        ' Không cho phép Null
        .AllowDBNull = False
        ' Chiều dài lớn nhất 30 ký tự
        .MaxLength = 30
    End With
```

```
    ' Thêm cột thứ hai vào đối tượng DataTable
    myDT.Columns.Add(myCol)
    ' Khai báo cột thứ ba
    myCol = New DataColumn("FullName")
    With myCol
       ' Khai báo kiểu dữ liệu chuỗi
       .DataType = Type.GetType("System.String")
       ' Chiều dài lớn nhất 50 ký tự
       .MaxLength = 50
    End With
    ' Thêm cột thứ ba vào đối tượng DataTable
    myDT.Columns.Add(myCol)
    ' Khai báo cột thứ tư
    myCol = New DataColumn("Address")
    With myCol
       ' Khai báo kiểu dữ liệu chuỗi
       .DataType = Type.GetType("System.String")
       ' Chiều dài lớn nhất 100 ký tự
       .MaxLength = 100
    End With
    ' Thêm cột thứ tư vào đối tượng DataTable

    myDT.Columns.Add(myCol)
    ' Khai báo cột thứ năm
    myCol = New DataColumn("Age")
    With myCol
       ' Khai báo kiểu dữ liệu số nguyên
       .DataType = Type.GetType("System.Int32")
       ' Giá trị mặc định là 20
       .DefaultValue = 20
    End With
    ' Thêm cột thứ năm vào đối tượng DataTable

    myDT.Columns.Add(myCol)
End Sub
```

Sau khi người sử dụng nhập các thuộc tính vào các điều khiển trên *Form*, nếu nhấn nút *Add Row*, lập tức dữ liệu thêm vào đối tượng *DataTable* và trình bày trên điều khiển *DataGrid* như hình 20-2.



Hình 20-2: *Sử dụng đối tượng DataColumn*

Để tìm hiểu thêm đối tượng *DataColumn*, bạn thực hành các ví dụ sau:

1. Thiết kế *Form* cho phép người sử dụng nhập dữ liệu trên các điều khiển, sau đó trình bày dữ liệu đó trên *DataGrid* cùng với số tự động tăng từ số kế tiếp, khi người sử dụng nhấn nút *Save*, bạn lưu dữ liệu đó xuống cơ sở dữ liệu nguồn.

2. Tạo ứng dụng *Windows Forms*, cho phép người sử dụng nhập tên *Table*, tên cột và chọn kiểu dữ liệu cùng với các thuộc tính đơn giản khác, sau đó tạo ra đối tượng *Table* này trong cơ sở dữ liệu *Northwind*.

3. ĐỐI TƯỢNG DATARELATION

Đối tượng *DataRelation* là một thành phần quan trọng của đối tượng *DataSet* nếu bên trong chúng đang lưu trữ các đối tượng *DataTable* có quan hệ với nhau.

Tuy nhiên, khi làm việc với đối tượng này bạn phải dựa trên hai cột dữ liệu tương ứng với hai đối tượng *DataTable* để xác định khóa chính và khóa ngoại.

3.1. Thuộc tính

Đối tượng *DataRelation* cung cấp các thuộc tính cho phép bạn lấy các thông tin về đối tượng *DataTable, DataSet,* tên quan hệ và cột khóa như: *ChildColumns, ChildTable, DataSet, ParentColumns, ParentTable, RelationName.*

3.1.1. Thuộc tính ChildColumns

Thuộc tính *ChildColumns* trả về danh sách các cột khóa ngoại trong đối tượng *DataRelation*, bởi vì trong đối tượng *DataTable* có thể có nhiều cột khóa ngoại.

3.1.2. Thuộc tính ChildTable

Thuộc tính *ChildTable* trả về đối tượng *DataTable* chứa khóa ngoại khai báo trong đối tượng *DataRelation.*

3.1.3. Thuộc tính RelationName

Thuộc tính *RelationName* trả về tên của đối tượng *DataRelation.*

3.1.4. Thuộc tính DataSet

Thuộc tính *DataSet* trả về đối tượng *DataSet* chứa đối tượng *DataRelation.*

3.1.5. Thuộc tính ParentColumns

Thuộc tính *ParentColumns* trả về danh sách các cột khóa chính trong đối tượng *DataRelation.*

3.1.6. Thuộc tính ParentTable

Thuộc tính *ParentTable* trả về đối tượng *DataTable* chứa khóa chính khai báo trong đối tượng *DataRelation.*

3.2. Phương thức

Đối tượng *DataRelation* cung cấp phương thức *GetType* trả về kiểu dữ liệu của cột khóa ngoại và khóa chính trong quan hệ.

Chú ý, hai cột có quan hệ với nhau sẽ có mọi thuộc tính giống nhau. Ngoài ra, bạn có thể sử dụng *Relations Collection* của đối tượng *DataSet* để khai báo và sử dụng các thuộc tính tương tự như đối tượng *DataRelation.*

Chẳng hạn, thêm *Form* mới vào *Project* với tên *frmDataRelation* có điều khiển *DataGrid*. Kế đến bạn khai báo đối tượng *DataSet* và *DataTable* như sau:

```
Dim myDS As New DataSet
Dim myDT As New DataTable
```

Bằng cách điền danh sách khách hàng từ bảng *Customers* vào đối tượng *DataSet* ứng với đối tượng *DataTable* thứ nhất, sau đó điền tiếp danh sách sản phẩm của những đơn đặt hàng của khách hàng vào đối tượng *DataSet* tương ứng với đối tượng *DataTable* thứ hai, bạn khai báo trong biến cố *Load* của *Form* như ví dụ 20-4.

Ví dụ 20-4: Trình bày danh sách khách hàng

```
Private Sub frmDataRelation_Load(ByVal sender _
As System.Object, ByVal e As System.EventArgs) _
Handles MyBase.Load
    ' Khai báo sử dụng đối tượng clsDatabase
    Dim myCls As New clsDatabase(gsCon)
    ' Định nghĩa phát biểu SQL ứng với bảng thứ nhất
    Dim strSQL As String
    strSQL = "Select CustomerID, CompanyName, "
    strSQL+=" Address from Customers"
    Try
        ' Điền dữ liệu bảng thứ nhất vào đối tượng DataSet
        myCls.getValue(myDS, strSQL)
        ' Khai báo phát biểu SQL ứng với bảng thứ hai
        strSQL = "Select CustomerID, D.ProductID,"
        strSQL +="ProductName,Quantity,D.UnitPrice"
        strSQL += " from Orders O, [Order Details] D, "
        strSQL += " Products P where "
        strSQL += " O.OrderID=D.OrderID"
        strSQL += " and P.ProductID=D.ProductID "
        ' Điền dữ liệu bảng thứ hai vào đối tượng DataTable
        myCls.getValue(myDT, strSQL)
        ' Thêm đối tượng DataTable vào đối tượng DataSet
        myDS.Tables.Add(myDT)
    Catch ex As Exception
        MsgBox(ex.Message)
    End Try
End Sub
```

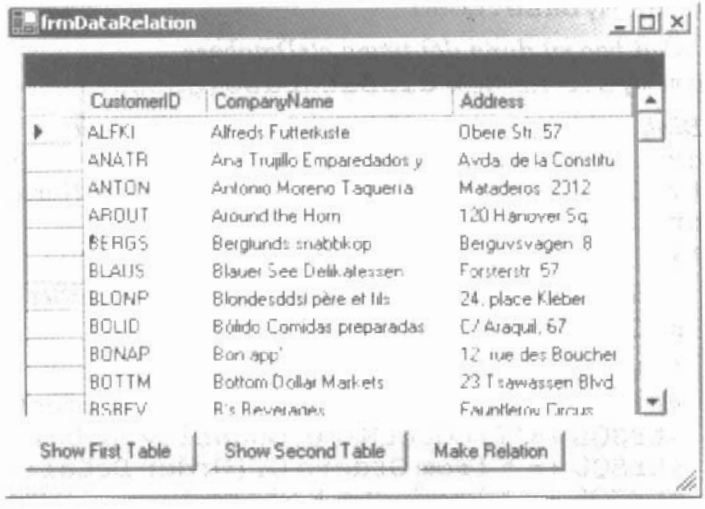
Lưu ý, trong ví dụ trên đối tượng *DataSet* đang tồn tại hai đối tượng *DataTable* tương ứng với hai tập dữ liệu.

Kế đến, khai báo để điền dữ liệu của đối tượng *DataTable* thứ nhất vào điều khiển *DataGrid* trong biến cố *Click* của nút *Show First Table* như ví dụ 20-5.

Ví dụ 20-5: Điền dữ liệu

```
Private Sub Button1_Click(ByVal sender As _
System.Object, ByVal e As System.EventArgs) _
Handles Button1.Click
  DataGrid1.DataSource = myDS.Tables(0)
End Sub
```

Như vậy, mỗi khi người sử dụng nhấn vào nút *Show First Table* thì kết quả liệt kê danh sách khách hàng như hình 20-3.



Hình 20-3: *Liệt kê danh sách khách hàng*

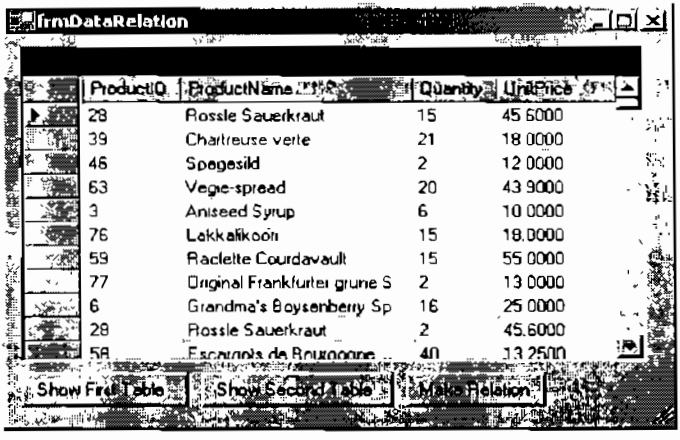
Tương tự, để điền dữ liệu trong đối tượng *DataTable* thứ hai vào điều khiển *DataGrid* thì bạn khai báo trong biến cố *Click* của nút *Show Second Table* như ví dụ 20-6:

Ví dụ 20-6: Thêm dữ liệu vào điều khiển DataGrid

```
Private Sub Button3_Click(ByVal sender As _
System.Object, ByVal e As System.EventArgs) _
Handles Button3.Click
```

```
    DataGrid1.DataSource = myDS.Tables(1)
End Sub
```

Mỗi khi người sử dụng nhấn vào nút *Show Second Table* thì kết quả liệt kê danh sách sản phẩm từ những đơn đặt hàng của tất cả khách hàng như hình 20-4.



Hình 20-4: *Danh sách sản phẩm của khách hàng*

Tuy nhiên, để liên kết danh sách sản phẩm của từng khách hàng đã đặt hàng, bạn cần tạo quan hệ giữa hai đối tượng *DataTable* này thông qua hai cột tương ứng là *CustomerID*.

Để làm điều này, bạn khai báo một phương thức nhận tham biến là đối tượng *DataSet*, sau đó khởi tạo đối tượng *DataRelation* để gán cột *CustomerID* tương ứng của hai đối tượng *DataTable* như ví dụ 20-7:

Ví dụ 20-7: Tạo quan hệ

```
Sub MakeRelation(ByRef myDS As DataSet)
    ' Khai báo cột thứ nhất
    Dim PKCol As DataColumn
    ' Khai báo cột thứ hai
    Dim FKCol As DataColumn
    ' Cột thứ nhất tương ứng với khóa chính (bảng thứ nhất với
    ' cột thứ nhất)
    PKCol = myDS.Tables(0).Columns(0)
    ' Cột thứ hai tương ứng với khóa ngoại (bảng thứ hai với
    ' cột thứ nhất)
```

```
    FKCol = myDS.Tables(1).Columns(0)
    ' Khai báo và khởi tạo đối tượng DataRelation ứng với hai cột
    Dim myRel As New DataRelation("ABC", _
      PKCol, FKCol)
    ' Thêm đối tượng DataRelation vào đối tượng DataSet
    myDS.Relations.Add(myRel)
End Sub
```

Lưu ý, phần trình bày chi tiết sản phẩm cần liên kết giữa ba bảng dữ liệu *Products*, *Orders* và *Order Details*, do quan hệ giữa bảng *Customers* và bảng *Order Details* thông qua *Orders*, chính vì vậy chúng ta có phát biểu *Select*:

```
Select
    CustomerID, D.ProductID,
    ProductName,Quantity,D.UnitPrice
From Orders O,[Order Details] D, Products P
Where O.OrderID=D.OrderID
And P.ProductID=D.ProductID
```

Trong đối tượng *DataTable* thứ hai vừa khai báo ở trên, có cột dữ liệu *CustomerID* dùng để quan hệ, khi trình bày trên điều khiển *DataGrid* bạn nên che dấu cột này.

Để gọi phương thức tạo quan hệ giữa hai đối tượng *DataTable* trong đối tượng *DataSet*, bạn khai báo trong biến cố *Click* của nút *Make Relation* như ví dụ 20-7:

Ví dụ 20-7: Gọi phương thức tạo quan hệ

```
Private Sub Button2_Click(ByVal sender As _
System.Object, ByVal e As System.EventArgs) _
Handles Button2.Click
  Try
    ' Nếu chưa tồn tại quan hệ
    If myDS.Relations.Count = 0 Then
    ' Gọi phương thức tạo quan hệ
      MakeRelation(myDS)
    End If
    ' Điền dữ liệu bảng thứ nhất lên điều khiển DataGrid
    DataGrid1.DataSource = myDS.Tables(0)
  Catch ex As Exception
    MsgBox(ex.Message)
```

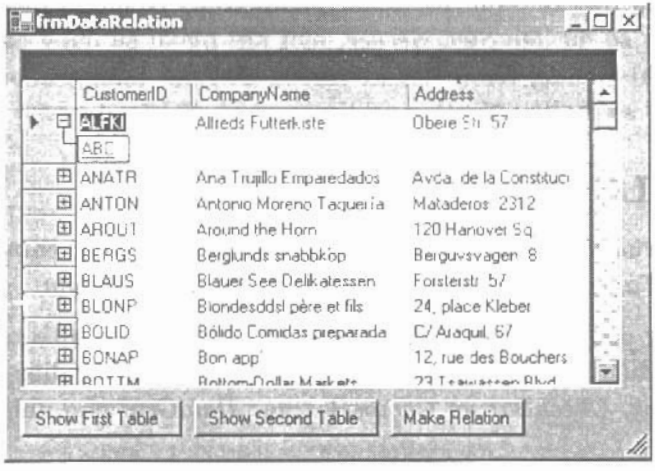
```
End Try
End Sub
```

Khi người sử dụng nhấn vào nút *Make Relation*, lập tức danh sách khách hàng xuất hiện trên điều khiển *DataGrid* ứng với từng dấu + như hình 20-5.

Hình 20-5: *Danh sách khách hàng*

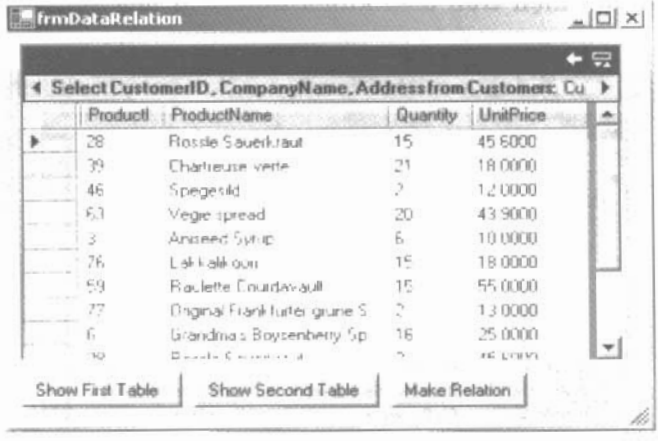
Chú ý, kích thước của các cột trong điều khiển *DataGrid* được định dạng tự động, bạn có thể tham khảo cách định dạng điều khiển này trong chuyên đề kế tiếp.

Tiếp tục như vậy, khi người sử dụng nhấn vào biểu tượng dấu +, lập tức tên quan hệ xuất hiện như hình 20-6.



Hình 20-6: *Tên quan hệ*

Mỗi khi người sử dụng chọn vào tên quan hệ *ABC* như hình 20-6, lập tức danh sách sản phẩm của chính khách hàng đó sẽ xuất hiện như hình 20-7.



Hình 20-7: *Danh sách sản phẩm*

Để tìm hiểu thêm về đối tượng *DataRelation*, bạn có thể thực hiện các ví dụ sau:

1. Thiết kế *Form*, cho phép người sử dụng liệt kê danh sách đơn đặt hàng, mỗi khi chọn vào một đơn đặt hàng thì danh sách đơn đặt hàng chi tiết xuất hiện.

2. Bằng cách sử dụng điều khiển *DataGrid*, bạn viết đoạn chương trình cho phép liệt kê danh sách sản phẩm trong cơ sở dữ liệu *Northwind* thuộc cơ sở dữ liệu *SQL Server*, mỗi khi người sử dụng chọn một sản phẩm thì chương trình liệt kê những đơn đặt hàng của sản phẩm đó trong cơ sở dữ liệu *Northwind* thuộc cơ sở dữ liệu *Access*.

4. KẾT LUẬN

Bạn vừa tìm hiểu cách làm việc với ba đối tượng chính trong đối tượng *DataTable* là *DataRow*, *DataColumn* và *DataRelation*.

Trong chuyên đề kế tiếp, chúng ta tiếp tục tìm hiểu chi tiết cách định dạng điều khiển *DataGrid* cùng với việc chèn các điều khiển khác như: *TextBox*, *CheckBox* và *ComboBox* vào *DataGrid* cho nhiều mục đích khác nhau.

Chuyên đề 21:

ĐIỀU KHIỂN DATAGRID VÀ DATABINDINGS

Tóm tắt chuyên đề 21

Trong những chuyên đề trước, chúng ta đã làm quen cách trình bày dữ liệu trên điều khiển DataGrid. Tuy nhiên, trong chuyên đề này bạn sẽ tiếp cận chi tiết hơn, phong phú hóa trình bày dữ liệu trên điều khiển này như định dạng màu, chữ, kích thước.

Bạn sẽ học cách nhúng các điều khiển TextBox, CheckBox và ComboBox vào điều khiển DataGrid, bạn cung cấp cho người sử dụng giao diện giao tiếp đa dạng.

Ngoài ra, chuyên đề này cũng hướng dẫn cách khai báo điều hướng mẩu tin trên điều khiển bằng cách sử dụng phương thức DataBindings và thuộc tính Position của đối tượng BindingManagerBase.

Các vấn đề chính sẽ được đề cập:

- ✓ *Điều khiển DataGrid.*
- ✓ *Định dạng điều khiển TextBox vào DataGrid.*
- ✓ *Định dạng điều khiển CheckBox vào DataGrid.*
- ✓ *Định dạng điều khiển ComboBox vào DataGrid.*
- ✓ *Điều hướng dữ liệu.*

1. ĐIỀU KHIỂN DATAGRID

Trong phần trình bày của các chuyên đề trước, để làm việc với cơ sở dữ liệu bạn khai báo lớp *clsDatabase* (có thể triển khai lớp này ở tầng *Bussiness Logic*) và các phương thức dùng để làm việc với các điều khiển trên *Form*.

Để trình bày dữ liệu trên điều khiển *DataGrid* với định dạng đơn giản, trước tiên chúng ta thêm *Form* vào *Project* và thêm điều khiển *DataGrid* cùng với điều khiển *ComboBox* như hình 21-1.

Kế đến, khai báo đối tượng *DataTable* và nạp danh sách *Country* vào điều khiển *ComboBox*, nếu không phát sinh lỗi thì tiếp tục điền danh sách khách hàng vào đối tượng *DataTable*.

Bằng cách gọi phương thức *getControls* cùng với đối tượng *DataTable*, bạn có thể trích lọc danh sách khách hàng của *Country* thứ nhất vào điều khiển *DataGrid* như ví dụ 21-1.

Ví dụ 21-1: Gọi phương thức getControl

```
' Khai báo đối tượng DataTable
Dim myDT As New DataTable
' Khai báo và khởi tạo đối tượng ClsControls ứng với điều khiển
' ComboBox
Dim myCls As New clsControls

Private Sub Form2_Load(ByVal sender As _
System.Object, ByVal e As System.EventArgs) _
Handles MyBase.Load
    ' Gọi phương thức getControls
   myCls.getControls(Me.cbCountry, _
      "select distinct Country from Customers", _
      "Country", "Country")
   If myCls.Erorr <> "" Then
     MsgBox(myCls.Erorr)
   Else
      ' Khai báo và khởi tạo đối tượng clsDatabase
      Dim myDB As New clsDatabase(gsCon)
      ' Gọi phương thức getValue
      myDB.getValue(myDT, _
      "select * from Customers")
      ' Gọi lại phương thức getControls ứng với điều khiển
      ' DataGrid
      Me.Label1.Text = "Records: " & _
      myCls.getControls(Me.DataGrid1, myDT, _
      cbCountry.Text)
   End If
End Sub
```

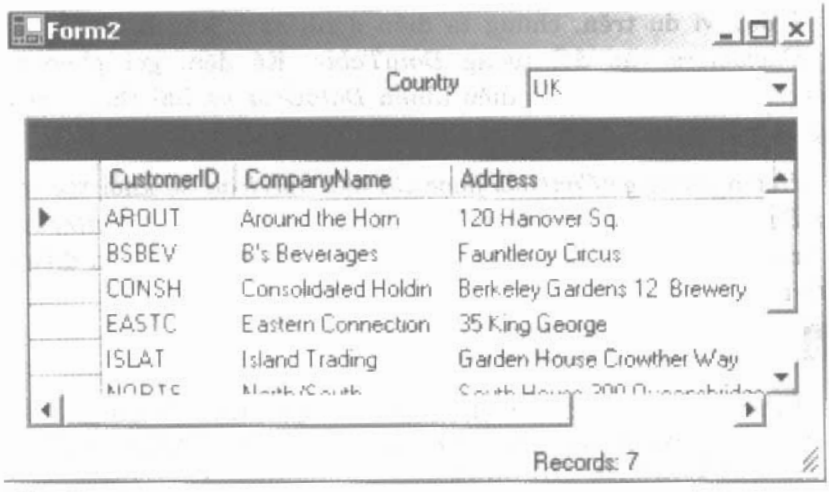
Trong ví dụ trên, chúng ta điền danh sách khách hàng có trong bảng *Customers* vào đối tượng *DataTable*. Kế đến, gọi phương thức *getControls* với tham trị là điều khiển *DataGrid* và hai tham trị là đối tượng *DataTable* với chuỗi chọn trên điều khiển *ComboBox*.

Phương thức *getControls* được cài đặt *Overload* sẽ khởi tạo một đối tượng *DataView* và lọc khách hàng có *Country* bằng với *Country* mà bạn chọn trên điều khiển *ComboBox*, sau đó trình bày chúng trên điều khiển *DataGrid* như ví dụ 21-2.

Ví dụ 21-2: Phương thức getControl

```
Function getControls(ByVal myControl As _
DataGrid, ByVal myDT As DataTable, ByVal Value _
As String) As Integer
   Dim i As Integer = 0
   Try
      ' Khai báo và khởi tạo đối tượng DataView
      Dim myDV As New DataView(myDT)
      ' Lọc dữ liệu theo Country
      myDV.RowFilter = "Country='" + Value + "'"
      Dim err As String = ""
      i = myDV.Count
      ' Điền dữ liệu từ đối tượng DataView vào điều khiển
      ' DataGrid
      myControl.DataSource = myDV
   Catch ex As Exception
      myError = ex.Message
   End Try
   Return i
End Function
```

Tương tự như vậy, mỗi khi người sử dụng thay đổi giá trị *Country* trên điều khiển *ComboBox*, danh sách khách hàng thuộc *Country* đó sẽ trình bày trên điều khiển *DataGrid* như hình 21-1.



Hình 21-1: *Liệt kê danh sách khách hàng*

Để làm điều này, bạn khai báo trong biến cố *SelectionChangeCommitted* của điều khiển *ComboBox* có tên *cbCountry* như ví dụ 21-3.

Ví dụ 21-3: Biến cố SelectionChangeCommitted

```
Sub cbCountry_SelectionChangeCommitted( _
ByVal sender As Object, ByVal e As _
System.EventArgs) Handles _
cbCountry.SelectionChangeCommitted
   Dim myCls As New clsControls
   Me.Label1.Text = "Records: " & _
      myCls.getControls(Me.DataGrid1, myDT, _
      cbCountry.Text)
End Sub
```

Lưu ý, do số lượng mẩu tin của bảng *Customers* có hạn nên bạn chỉ nạp dữ liệu một lần vào đối tượng *DataTable*, sau đó lọc dữ liệu của đối tượng này và trình bày trên điều khiển *DataGrid* mỗi khi người sử dụng thay đổi *Country*.

Trong trường hợp mẩu tin trình bày có số lượng lớn, mỗi khi người sử dụng thay đổi *Country*, bạn nên xem xét việc đọc lại cơ sở dữ liệu thay vì nạp một lần như trường hợp trên. Bởi vì đối tượng *DataTable* lẫn *DataSet* thuộc lớp không kết nối, chúng sử dụng bộ nhớ truy cập nhanh (*RAM* lẫn đĩa cứng) để lưu trữ dữ liệu cho hai đối tượng này.

Trong trường hợp muốn điều chỉnh chiều rộng của từng cột, định dạng màu lẫn chọn kiểu chữ cho dữ liệu trên điều khiển *DataGrid*, bạn có thể sử dụng đối tượng *DataGridTableStyle*.

Để làm điều này, trước tiên bạn khai báo phương thức khởi tạo đối tượng *DataGridTableStyle*. Kế đến, thay đổi định dạng của điều khiển bằng cách thay đổi thuộc tính như ví dụ 21-4.

Ví dụ 21-4: Phương thức định dạng điều khiển DataGrid

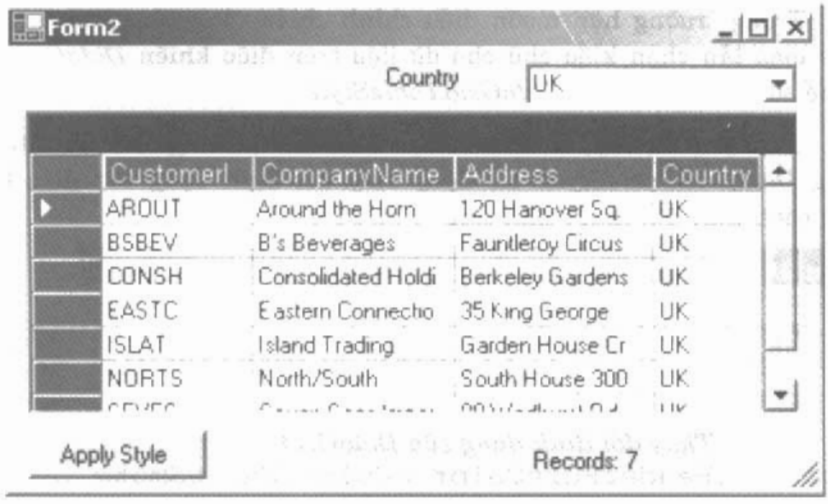
```
Function ApplyStyle() As DataGridTableStyle
  Dim myStyle As New DataGridTableStyle
  With myStyle
    ' Thay đổi định dạng của DataGrid
    .HeaderForeColor = Color.WhiteSmoke
    .HeaderBackColor = Color.Green
    .HeaderFont = New Font("Arial", 10, _
      FontStyle.Regular)
    .ReadOnly = True
    .GridLineColor = Color.HotPink
    .SelectionBackColor = Color.Gray
    .SelectionForeColor = Color.Olive
  End With
  Return myStyle
End Function
```

Sau đó, khai báo trong biến cố *Click* của nút *Apply Style*, bạn có thể áp dụng định dạng này cho điều khiển *DataGrid* bằng cách sử dụng phương thức *Add* của *TableStyles Collection* thuộc *DataGrid* như ví dụ 21-5.

Ví dụ 21-5: Gọi phương thức ApplyStyle

```
Private Sub Button1_Click(ByVal sender As _
System.Object, ByVal e As System.EventArgs) _
Handles Button1.Click
  ' Gọi phương thức ApplyStyle
  DataGrid1.TableStyles.Add(ApplyStyle)
End Sub
```

Chẳng hạn, trong trường hợp này điều khiển *DataGrid* sẽ thay đổi màu và đường viền như hình 21-2.



Hình 21-2: *Áp dụng định dạng*

Để tìm hiểu thêm cách áp dụng đối tượng *DataGridTableStyle* cho điều khiển *DataGrid*, bạn có thể thực hành các ví dụ sau:

1. Thiết kế *Form*, cho phép người sử dụng liệt kê danh sách khách hàng với định dạng tùy ý, mỗi khi chọn vào một khách hàng thì danh sách đơn đặt hàng xuất hiện với định dạng khác với định dạng danh sách khách hàng, tiếp tục chọn vào đơn đặt hàng thì danh sách các sản phẩm của đơn đặt hàng đó xuất hiện.
2. Viết ứng dụng *Windows Forms*, dùng để liệt kê tên tập tin, dung lượng, ngày tạo ra của thư mục chỉ định, sau đó điền dữ liệu đó vào điều khiển *DataGrid* với định dạng nào đó.

2. ĐỊNH DẠNG ĐIỀU KHIỂN TEXTBOX VÀO DATAGRID

Để cho phép định dạng trên từng *Cell* của điều khiển *DataGrid*, bạn phải kết hợp đối tượng *DataGridTextBoxColumn* vào đối tượng *DataGridTableStyle*.

Chẳng hạn, chúng ta che dấu cột *Country*, thay tựa đề của cột *CompanyName* thành *Customer Name* và chỉ cho phép người sử dụng thay đổi giá trị (tên) của cột *CompanyName* ứng với từng khách hàng thì sử dụng đối tượng *DataGridTextBoxColumn*.

Để làm điều này, trước tiên bạn khai báo phương thức *ApplyStyle* tương tự trường hợp trên như ví dụ 21-6.

Ví dụ 21-6: Khai báo phương thức ApplyStyle

```
Function ApplyStyle() As DataGridTableStyle
  Dim myStyle As New DataGridTableStyle
  With myStyle
    ' Thay đổi định dạng của DataGrid
    .HeaderForeColor = Color.WhiteSmoke
    .HeaderBackColor = Color.Green
    .HeaderFont = New Font("Arial", 10, _
      FontStyle.Regular)
    ' Cho phép thay đổi dữ liệu trên DataGrid
    .ReadOnly = False
    .GridLineColor = Color.HotPink
    .SelectionBackColor = Color.Gray
    .SelectionForeColor = Color.Olive
  End With
  Return myStyle
End Function
```

Kế đến, định nghĩa một phương thức có tên *ApplyColumnStyle*, sử dụng đối tượng *DataGridTextBoxColumn* để định dạng cho từng ô dữ liệu.

Sau đó, áp dụng đối tượng này bằng cách sử dụng phương thức *Add* của *GridColumnStyles Collection* thuộc đối tượng *DataGridTableStyle* như ví dụ 21-7.

Ví dụ 21-7: Khai báo phương thức ApplyColumnStyle

```
Sub ApplyColumnStyle()
  ' Khai báo và khởi tạo đối tượng DataGridTableStyle
  Dim myColStyle As New DataGridTableStyle
  ' Gán đối tượng DataGridTableStyle từ phương thức ApplyStyle
  myColStyle = ApplyStyle()
  ' Duyệt trên từng cột dữ liệu của đối tượng DataTable
  For Each dc As DataColumn In myDT.Columns
    ' Khai báo và khởi tạo đối tượng DataGridTextBoxColumn
    Dim myTxt As New DataGridTextBoxColumn
```

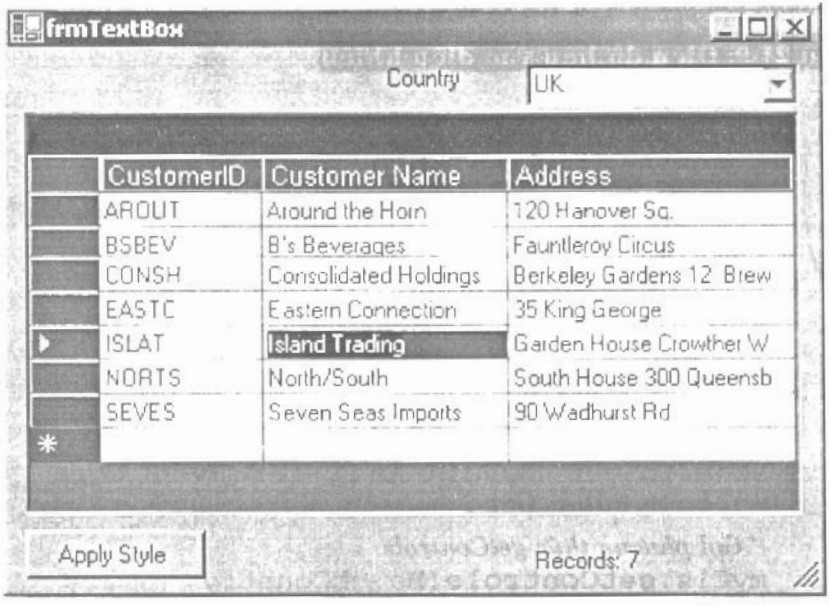
```
        'Thay đổi thuộc tính cột dữ liệu
        With myTxt
            'Ánh xạ tên cột tương ứng với đối tượng DataTable
            .MappingName = dc.ColumnName
            .HeaderText = dc.Caption
            'Nếu cột là CompanyName thì thay đổi
            Select Case dc.ColumnName
              Case "CompanyName"
                .HeaderText = "Customer Name"
                .Alignment = HorizontalAlignment.Left
                .TextBox.BackColor = Color.Red
                'Cho thay đổi dữ liệu
                .ReadOnly = False
                .Width = 200
              Case "Country"
                'Nếu cột là Country thì che dấu và không cho thay đổi
                .Width = 0
                .ReadOnly = True
              Case Else
                'Các cột khác không cho thay đổi dữ liệu
                .ReadOnly = True
            End Select
        End With
        'Áp dụng định dạng cột vào đối tượng DataGridTableStyle
        myColStyle.GridColumnStyles.Add(myTxt)
    Next
    myColStyle.MappingName = myDT.TableName
    'Áp dụng định dạng bảng vào điều khiển DataGrid
    DataGrid1.TableStyles.Add(myColStyle)
End Sub
```

Bằng cách gọi phương thức *ApplyColumnStyle* trong biến cố *Click* của nút *Apply Style* như ví dụ 21-8.

Ví dụ 21-8: Gọi phương thức ApplyColumnStyle

```
Private Sub Button1_Click(ByVal sender As _
System.Object, ByVal e As System.EventArgs) _
Handles Button1.Click
  ApplyColumnStyle()
End Sub
```

Sau khi điền dữ liệu vào điều khiển *DataGrid*, nếu bạn nhấn nút *Apply Style* thì định dạng *DataGrid* sẽ thay đổi, người sử dụng có thể thay đổi dữ liệu trên cột *Customer Name* như hình 21-3.



Hình 21-3: *Định dạng từng ô dữ liệu*

Để tìm hiểu thêm đối tượng *DataGridTextBoxColumn*, bạn thực hành các ví dụ sau:

1. Thiết kế *Form*, liệt kê danh sách các sản phẩm chi tiết của đơn đặt hàng ứng với từng khách hàng, chỉ cho phép người sử dụng thay đổi cột số lượng, sau đó cập nhật trở lại cơ sở dữ liệu nguồn.

2. Tạo một ứng dụng, cho phép người sử dụng nhập mã sản phẩm trong cột thứ nhất, lập tức tên của sản phẩm xuất hiện trong cột thứ hai và số lượng còn trong kho xuất hiện trong cột thứ 3.

3. ĐỊNH DẠNG ĐIỀU KHIỂN CHECKBOX VÀO DATAGRID

Tương tự như cách chèn điều khiển *TextBox* vào điều khiển *DataGrid*, nếu muốn cho phép người sử dụng chọn (*Check*) trên từng hàng của điều khiển này thì bạn sử dụng đối tượng *DataGridBoolColumn*.

Để làm điều này, trước tiên bạn khai báo nạp dữ liệu lên trình điều khiển *DataGrid* với cột dữ liệu có kiểu dữ liệu mang giá trị 0/1 (giả sử bạn khai báo cột *Available*) và định dạng điều khiển này như đã trình bày trong hai phần trên với ví dụ 21-9.

Ví dụ 21-9: Điền dữ liệu vào điều khiển

```
' Khai báo đối tượng DataTable
Dim myDT As New DataTable
' Khai báo và khởi tạo đối tượng ClsControls ứng với điều khiển
' ComboBox
Dim myCls As New clsControls
' Khai báo biến nắm giữ số mẩu tin mỗi khi đọc và điền vào
' điều khiển DataGrid
Dim iRecord As Integer=0
Private Sub Form2_Load(ByVal sender As _
System.Object, ByVal e As System.EventArgs) _
Handles MyBase.Load
    ' Gọi phương thức getControls
   myCls.getControls(Me.cbCountry, _
      "select distinct Country from Customers", _
      "Country", "Country")
   If myCls.Erorr <> "" Then
     MsgBox(myCls.Erorr)
   Else
      ' Khai báo và khởi tạo đối tượng clsDatabase
      Dim myDB As New clsDatabase(gsCon)
      ' Gọi phương thức getValue
      myDB.getValue(myDT, _
     "select CustomerID,CompanyName,Address," & _
      " Country,1 As Available  from Customers")
      ' Gọi lại phương thức getControls ứng với điều khiển
      ' DataGrid và lấy số mẩu tin
      iRecord= myCls.getControls(Me.DataGrid1, _
      myDT,cbCountry.Text)
      Me.Label1.Text = "Records: " & iRecord

   End If
End Sub
```

Kế đến, trong phương thức *ApplyColumnStyle* chúng ta phân ra hai trường hợp, nếu cột dữ liệu bình thường thì sử dụng định dạng bằng đối tượng *DataGridTextBoxColumn* như trình bày ở phần trên.

Tuy nhiên, trong trường hợp cột dữ liệu có kiểu dữ liệu dạng *True/False* hay 0 và 1 thì bạn sử dụng đối tượng *DataGridBoolColumn* như ví dụ 21-10.

Ví dụ 21-10: Khai báo phương thức ApplyColumnStyle

```
Sub ApplyColumnStyle()
    ' Khai báo và khởi tạo đối tượng DataGridTableStyle
  Dim myColStyle As New DataGridTableStyle
  ' Gán đối tượng DataGridTableStyle từ phương thức ApplyStyle
  myColStyle = ApplyStyle()
  ' Duyệt trên từng cột dữ liệu của đối tượng DataTable
  For Each dc As DataColumn In myDT.Columns
    ' Nếu không phải là cột Available
    If dc.ColumnName <> "Available" Then
      ' Khai báo định dạng như TextBox
      Dim myTxt As New DataGridTextBoxColumn
      With myTxt
        .MappingName = dc.ColumnName
        .HeaderText = dc.Caption
        .ReadOnly = True
        Select Case dc.ColumnName
          Case "CompanyName"
            .HeaderText = "Customer Name"
            .TextBox.BackColor = Color.Red
            .Width = 150
          Case "Country", "CustomerID"
            .Width = 0
          Case Else
            .Width = 150
        End Select
      End With
      ' Áp dụng định dạng vào đối tượng DataGridTableStyle
      myColStyle.GridColumnStyles.Add(myTxt)
    Else
      ' Nếu cột Available thì sử dụng đối tượng
      ' DataGridBoolColumn
      Dim chkCol As  New DataGridBoolColumn
```

```
            'Thay đổi thuộc tính của cột Available
            With chkCol
               .MappingName = dc.ColumnName
               .HeaderText = dc.Caption
               .Alignment = HorizontalAlignment.Center
               'Cho phép thay đổi giá trị
               .ReadOnly = False
               'Không cho phép giá trị Null
               .AllowNull = False
               'Giá trị thực khi người sử dụng không chọn
               .FalseValue = 0
               'Giá trị thực khi người sử dụng chọn
               .TrueValue = 1
               .Width = 50
            End With
            'Áp dụng định dạng vào đối tượng DataGridTableStyle
            myColStyle.GridColumnStyles.Add(chkCol)
         End If
      Next
      myColStyle.MappingName = myDT.TableName
      'Áp dụng định dạng vào điều khiển DataGrid
      DataGrid1.TableStyles.Add(myColStyle)
   End Sub
```

Lưu ý, nếu muốn chuyển sang dạng *CheckBox*, trạng thái của chúng ở dạng không chọn (*uncheck*) thì bạn khai báo như sau:

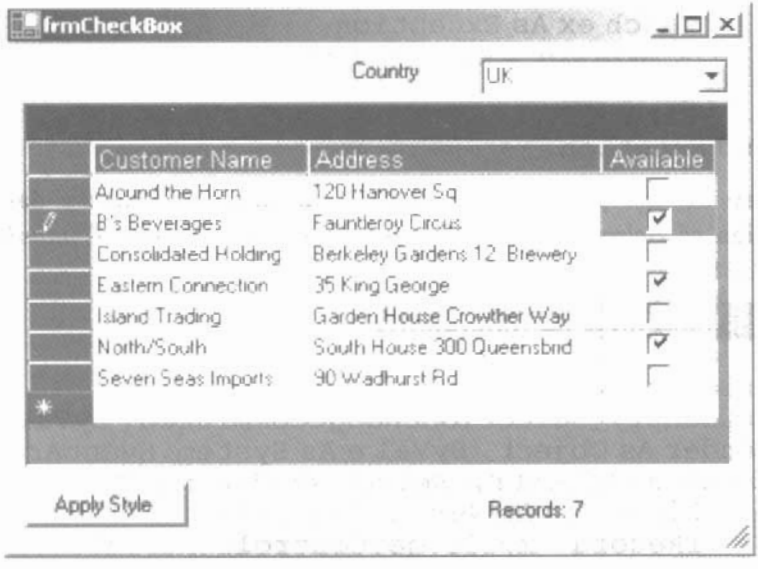
```
With chkCol
        .MappingName = dc.ColumnName
        .HeaderText = dc.Caption
        .Alignment = HorizontalAlignment.Center
        'Cho phép thay đổi giá trị
        .ReadOnly = False
        'Không cho phép giá trị Null
        .AllowNull = False
        'Giá trị thực khi người sử dụng không chọn
        .FalseValue = 1
        'Giá trị thực khi người sử dụng chọn
        .TrueValue = 0
        .Width = 50
End With
```

Kế đến, bạn gọi phương thức này trên trong biến cố của nút *Apply Style* như ví dụ 21-11.

Ví dụ 21-11: Gọi phương thức ApplyColumnStyle

```
Private Sub Button1_Click(ByVal sender As _
System.Object, ByVal e As System.EventArgs) _
Handles Button1.Click
   ApplyColumnStyle()
End Sub
```

Sau khi điền dữ liệu vào điều khiển *DataGrid*, nếu người sử dụng nhấn nút *Apply Style*, lập tức cột *Available* trở thành dạng *CheckBox*, người sử dụng có thể chọn hay không trên từng *CheckBox* này như hình 21-4.



Hình 21-4: *Sử dụng đối tượng DataGridBoolColumn*

Mỗi khi người sử dụng chọn *Country* trên *ComboBox*, lập tức danh sách khách hàng thuộc *Country* đó trình bày trên điều khiển *DataGrid* Để làm điều này, bạn cài đặt phương thức *getControl* như ví dụ 21-11-1.

Ví dụ 21-11-1: Lọc dữ liệu

```
Function getControls(ByVal myControl As
DataGrid, ByVal myDT As DataTable, ByVal Value As
String) As Integer
    Dim i As Integer = 0
```

```
        Try
            ' Sử dụng đối tượng View để lọc dữ liệu trên đối tượng
            ' DataTable đang tồn tại
            Dim myDV As New DataView(myDT)
            If Value <> "" Then
              myDV.RowFilter = _
              "Country='" + Value + "'"
            End If
            Dim err As String = ""
            i = myDV.Count
            ' Điền dữ liệu vào điều khiển DataGrid từ
            ' đối tượng DataView
            myControl.DataSource = myDV
        Catch ex As Exception
            myError = ex.Message
        End Try
        Return i
    End Function
```

Sau đó, gọi phương thức *getControls* ứng với *Country* đang chọn trong biến cố *SelectionChangeCommitted* của điều khiển *ComboBox* như ví dụ 21-11-2.

Ví dụ 21-11-2: Liệt kê khách hàng

```
Private Sub
cbCountry_SelectionChangeCommitted( ByVal
sender As Object, ByVal e As System.EventArgs)
Handles cbCountry.SelectionChangeCommitted
    Dim myCls As New clsControls
    iRecord = myCls.getControls(Me.DataGrid1, _
    myDT, cbCountry.Text)
    Me.Label1.Text = "Records: " & iRecord
End Sub
```

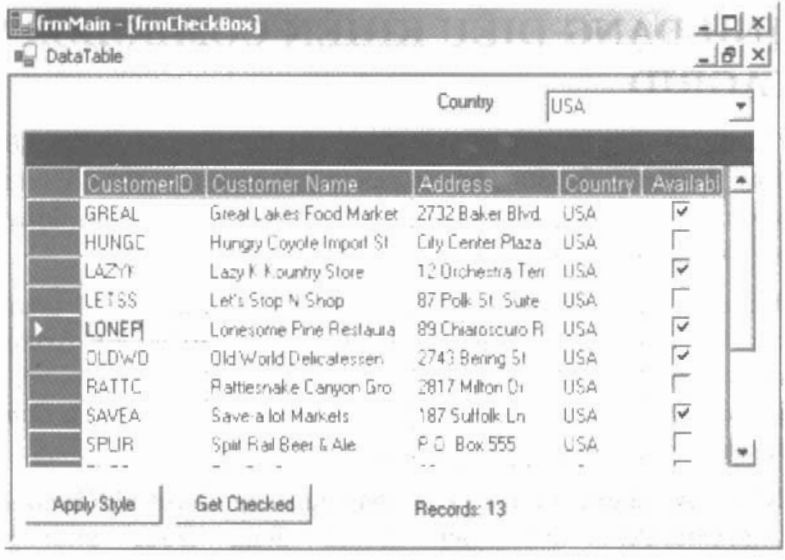
Để lấy giá trị mã khách hàng của những hàng dữ liệu mà người sử dụng chọn vào *CheckBox* trong điều khiển *DataGrid*, bạn thêm nút vào *frmCheckBox* và khai báo trong biến cố *Click* như ví dụ 21-11-3.

Ví dụ 21-11-3: Lấy giá trị chọn

```
Private Sub Button2_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles Button2.Click
```

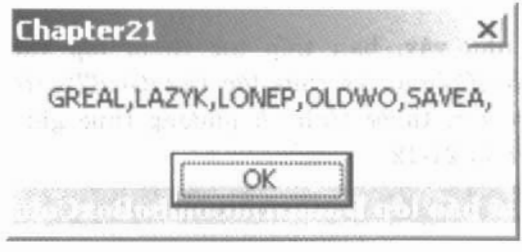
```
        Dim strSelect As String = ""
        For i As Integer = 0 To iRecord - 1
          If CType(DataGrid1(i, 4), _
            Boolean) = True Then
            strSelect += DataGrid1(i, 0) + ","
          End If
        Next
        MsgBox(strSelect)
    End Sub
```

Chẳng hạn, sau khi người sử dụng chọn vào 3 khách hàng thuộc *Country* có giá trị là *USA* như hình 21-4-1.



Hình 21-4-1: *Chọn khách hàng*

Nếu nhấn nút *GetChecked*, lập tức danh sách mã khách hàng xuất ra màn hình theo định dạng cách nhau dấu phẩy như hình 21-4-2.



Hình 21-4-2: *Giá trị chọn*

Để tìm hiểu thêm cách định dạng *CheckBox* trên điều khiển *DataGrid*, bạn thực hành các ví dụ sau:

1. Liệt kê danh sách sản phẩm trên điều khiển *DataGrid*, nếu người sử dụng chọn những sản phẩm trên *checkbox* và nhấn nút *Update* thì bạn tăng giá của sản phẩm đó lên 10 đồng.
2. Thiết kế *Form*, cho phép người sử dụng chọn một hay nhiều đơn đặt hàng, sau đó nhấn nút *Delete* thì cửa sổ xác nhận hành động xóa hay không, nếu người sử dụng tiếp tục chọn *Yes* thì bạn xóa các đơn đặt hàng đó trong cơ sở dữ liệu.

4. ĐỊNH DẠNG ĐIỀU KHIỂN COMBOBOX VÀO DATAGRID

Tương tự như hai cách định dạng cột với *TextBox*, *CheckBox* vừa trình bày ở trên, nếu cho phép người sử dụng chọn giá trị trong một danh sách dạng xổ xuống (*dropdownlist*) thì bạn có thể định dạng cột đó như một điều khiển *ComboBox*.

Không giống như hai cách định dạng vừa trình bày ở trên, *.NET* cung cấp hai đối tượng chính là *DataGridTextBoxColumn* (trường hợp *TextBox*) và *DataGridBoolColumn* (trường hợp *CheckBox*) , đối với trường hợp định dạng *ComboBox* thì bạn phải sử dụng đến đối tượng *ComboBox* và khai báo ghi đè lên các phương thức của nó.

Như vậy, chúng ta cần khai báo hai lớp *DataGridComboBox* và *DataGridComboBoxColumn* sau đó sử dụng trong phương thức *ApplyColumnStyle*.

Để làm điều này, trước tiên bạn thêm lớp vào *Project* và đặt tên *DataGridComboBox* kế thừa *System.Windows.Forms.ComboBox*, rồi khai báo hai phương thức *FindValue* và *FindDisplay*.

Tương tự như vậy, bạn tiếp tục thêm lớp vào *Project* với tên *DataGridComboBoxColumn* kế thừa lớp *DataGridTextBoxColumn*, sau đó khai báo *Constructor*, thuộc tính và phương thức ghi đè của đối tượng *ComboBox* như ví dụ 21-12.

Ví dụ 21-12: Khai báo lớp DataGridComboBoxColumn

```
Public Sub New(ByVal DataSource As DataView, _
  ByVal displayString As String, _
```

```
    ByVal valueString As String)
    myCurrencyManager = Nothing
    myEditing = False
    myCombo = New DataGridComboBox
    myCombo.DropDownStyle = _
        ComboBoxStyle.DropDownList
    myCombo.Visible = False
    ' Gán thuộc tính DataSource
    myCombo.DataSource = DataSource
    ' Gán thuộc tính DisplayMember
    myCombo.DisplayMember = displayString
    ' Gán thuộc tính ValueMember
    myCombo.ValueMember = valueString
    AddHandler myCombo.Leave, _
       AddressOf myCombo_Leave
    AddHandler myCombo.SelectionChangeCommitted, _
    AddressOf myCombo_SelectionChangeCommitted
End Sub
```

Tuy nhiên, trong trường hợp này chúng ta sử dụng một *User Control* có tên *myUserControls* bao gồm hai lớp vừa khai báo ở trên *DataGridComboBox* và *DataGridComboBoxColumn*.

Bằng cách tham chiếu đến tập tin *myUserControls.dll* vào *Project* và khai báo sử dụng không gian tên này như sau:

```
Imports myUserControls
Public Class frmCombo
   Inherits System.Windows.Forms.Form
...
```

Trong phương thức *ApplyColumnStyle*, nếu cột dữ liệu là *Country* thì bạn khai báo và khởi tạo đối tượng *DataGridComboBoxColumn*, các cột khác sử dụng đối tượng *DataGridTextBoxColumn* như ví dụ 21-13.

Ví dụ 21-13: Phương thức ApplyColumnStyle

```
Sub ApplyColumnStyle()
    ' Khai báo và khởi tạo đối tượng DataGridTableStyle
    Dim myColStyle As New DataGridTableStyle
```

```
    ' Gọi phương thức ApplyStyle để áp dụng định dạng cho
DataGrid
    myColStyle = ApplyStyle()
    ' Duyệt trên từng cột dữ liệu
    For Each dc As DataColumn In myDT.Columns
        ' Nếu cột dữ liệu khác
        If dc.ColumnName <> "Country" Then
            ' Khai báo và khởi tạo đối tượng DataGridTextBoxColumn
            Dim myTxt As New DataGridTextBoxColumn
            With myTxt
                .MappingName = dc.ColumnName
                .HeaderText = dc.Caption
                .ReadOnly = True
                .Width = 150
                Select Case dc.ColumnName
                    Case "CompanyName"
                        .HeaderText = "Customer Name"
                        .TextBox.BackColor = Color.Red
                        .Width = 150
                    Case "CustomerID"
                        .Width = 0
                End Select
            End With
            ' Thêm đối tượng DataGridTextBoxColumn vào đối tượng
            ' DataGridTableStyle
            myColStyle.GridColumnStyles.Add(myTxt)
        Else
            ' Nếu cột dữ liệu là Country (dùng ComboBox)
            ' Khai báo và khởi tạo đối tượng
            ' DataGridComboBoxColumn
            Dim myCb As DataGridComboBoxColumn
            ' Khai báo và khởi tạo đối tượng DataTable
            Dim myCountry As New DataTable
            ' Khai báo và khởi tạo đối tượng clsDatabase
            Dim myDB As New clsDatabase(gsCon)
            ' Gọi phương thức getValue để điền dữ liệu cột Country
            ' trong bảng Customers vào đối tượng DataTable
            myDB.getValue(myCountry, _
            "select Distinct Country from Customers")
```

```
            'Khai báo và khởi tạo đối tượng
            'DataGridComboBoxColumn
            myCb = New DataGridComboBoxColumn( _
          myCountry.DefaultView, _
             "Country", "Country")
            myCb.MappingName = dc.ColumnName
            myCb.HeaderText = dc.Caption
            'Thêm đối tượng DataGridComboBoxColumn vào đối
            'tượng DataGridTableStyle
            myColStyle.GridColumnStyles.Add(myCb)
          End If
        Next
        myColStyle.MappingName = myDT.TableName
        DataGrid1.TableStyles.Add(myColStyle)
    End Sub
```

Lưu ý, khi khai báo và khởi tạo đối tượng *DataGridComboBoxColumn*, bạn cần truyền 3 tham số vào *Constructor* là đối tượng *DataView* ứng với thuộc tính *DataSource*, hai chuỗi tương ứng với thuộc tính *DisplayMember* và *ValueMember* của đối tượng *ComboBox*.

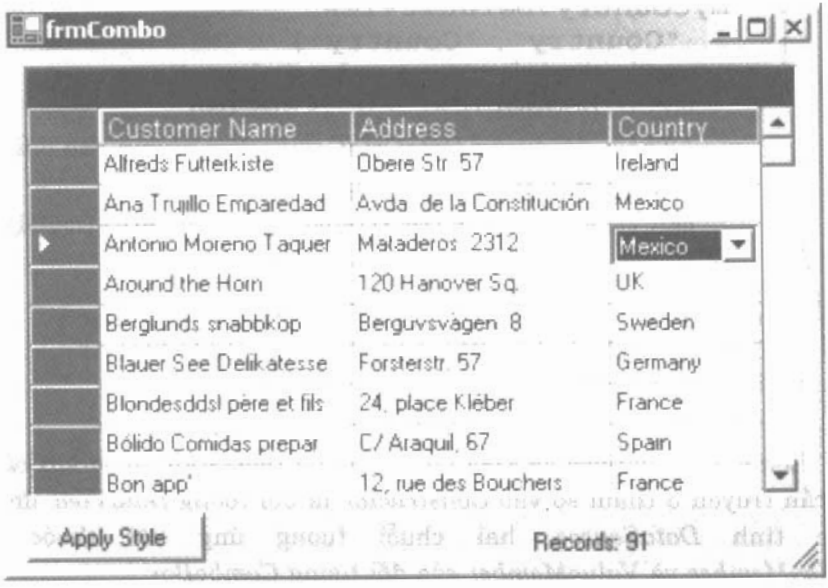
```
myCb = New DataGridComboBoxColumn( _
myCountry.DefaultView, "Country", "Country")
     myCb.MappingName = dc.ColumnName
     myCb.HeaderText = dc.Caption

myColStyle.GridColumnStyles.Add(myCb)
```

Sau đó, khai báo để gọi phương thức *ApplyColumnStyle* trong biến cố *Click* của nút *Apply Style* như ví dụ 21-14.

Ví dụ 21-14: Gọi phương thức ApplyColumnStyle

```
Private Sub Button1_Click(ByVal sender As _
System.Object, ByVal e As System.EventArgs) _
Handles Button1.Click
  ApplyColumnStyle()
End Sub
```

Sau khi nạp *Form*, dữ liệu của bảng *Customers* trình bày trên điều khiển *DataGrid*, nếu người sử dụng nhấn nút *Apply Style* thì danh sách *ComboBox* xuất hiện trong cột *Country* như hình 21-5.



Hình 21-5: *Định dạng ComboBox trong DataGrid*

Để tìm hiểu thêm các định dạng điều khiển *ComboBox* trong điều khiển *DataGrid* bằng cách sử dụng thư viện *Assembly* có tên *myUserControls* đính kèm theo đĩa, bạn có thể thực hành các ví dụ sau:

1. Thiết kế *Form*, trình bày danh sách sản phẩm cùng với cột cho phép người sử dụng chọn một trong hai giá trị *Yes* hay *No* bằng điều khiển *ComboBox*.

2. Tạo *Project*, trong đó có sử dụng điều khiển *DataGrid* liệt kê danh sách khách hàng trong bảng *Customers* với định dạng *ComboBox* trên cột *Country*, nếu người sử dụng nhấn nút *Update* thì bạn cập nhật lại những khách hàng đã thay đổi *Country*.

5. ĐIỀU HƯỚNG DỮ LIỆU

Nếu trình bày dữ liệu từng mẩu tin trên các điều khiển, cho phép người sử dụng điều hướng (*navigation*) thì bạn sử dụng đối tượng *BindingManagerBase*.

```
Dim myBMB As BindingManagerBase
```

Sau khi điền dữ liệu vào đối tượng *DataTable* hay *DataSet*, để hiển thị dữ liệu tương ứng của từng cột thuộc hàng hiện hành trên điều khiển, bạn sử dụng phương thức *Add* của *DataBindings Collection* thuộc điều khiển.

```
txtCity.DataBindings.Add("Text", myDT, "City")
```

Trong đó *Text* chính là thuộc tính *Text* của điều khiển *txtCity*, *myDT* là đối tượng *DataTable* cần lấy dữ liệu, *City* là tên cột dữ liệu.

Bằng cách sử dụng thuộc tính *BindingContext* của *Form* để điền đối tượng *DataTable* vào đối tượng *BindingManagerBase*.

```
myBMB = Me.BindingContext(myDT)
```

Mỗi khi người sử dụng di chuyển đến mẩu tin kế tiếp, mẩu tin trước đó, mẩu tin đầu tiên hay mẩu tin cuối cùng, bạn có thể dựa vào thuộc tính *Position*, hoặc có thể trỏ đến mẩu tin cần trình bày.

Để thực hiện ý tưởng này, trước tiên bạn thêm *Form* vào *Project* với tên *frmNavigation*. Kế đến, khai báo đối tượng *BindingManagerBase* và *DataTable* như sau:

```
Public Class frmNavigation
  Inherits System.Windows.Forms.Form
  Dim myDT As New DataTable
  Dim myBMB As BindingManagerBase
```

Sau đó, thêm 4 điều khiển *TextBox* tương ứng với 4 cột dữ liệu là *CompanyName*, *Address*, *City* và *Country*. Bằng cách sử dụng phương thức *getValue* của đối tượng *clsDatabase*, bạn điền dữ liệu vào đối tượng *DataTable* và sử dụng phương thức *Add* của *DataBindings Collection* thuộc điều khiển để trình bày dữ liệu trên điều khiển như ví dụ 21-15.

Ví dụ 21-15: Khai báo lớp trong biến Load của Form

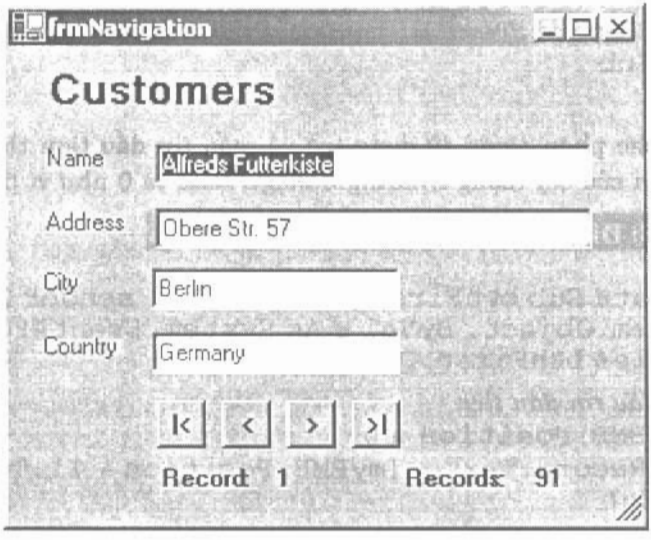
```
Private Sub frmNavigation_Load(ByVal sender _
 As System.Object, ByVal e As System.EventArgs) _
Handles MyBase.Load
   ' Khai báo và khởi tạo đối tượng clsDatabase
   Dim myCls As New clsDatabase(gsCon)
```

```
    ' Khai báo biến chuỗi lỗi
    Dim strError As String
    ' Gọi phương thức getValue
    strError = myCls.getValue(myDT, _
      "select * from Customers")
    ' Nếu có dữ liệu
    If Not strError Is Nothing Then
      ' Điền dữ liệu lên từng điều khiển
      txtName.DataBindings.Add("Text", _
        myDT, "CustomerName")
      txtAddress.DataBindings.Add("Text", _
        myDT, "Address")
      txtCity.DataBindings.Add("Text", _
        myDT, "City")
      txtCountry.DataBindings.Add("Text", _
        myDT, "Country")
      ' Gán đối tượng DataTable vào đối tượng
      ' BindingManagerBase
      myBMB = Me.BindingContext(myDT)
      ' Liệt kê tổng số mẩu tin
      lbRecords.Text = _
        myDT.Rows.Count.ToString
    Else
      MsgBox("Can not load data")
      btnPre.Enabled = False
      btnNext.Enabled = False
      btnLast.Enabled = False
      btnFirst.Enabled = False
    End If
  End Sub
```

Lưu ý, nếu phương thức *getValue* phát sinh lỗi thì biến *strError* sẽ khác rỗng, lập tức các nút *btnNext*, *btnPre*, *btnLast* và *btnFirst* sẽ vô hiệu hóa bằng cách khai báo như sau:

```
btnPre.Enabled = False
btnNext.Enabled = False
btnLast.Enabled = False
btnFirst.Enabled = False
```

Khi người sử dụng gọi *frmNavigation* từ *Menu*, lập tức dữ dữ liệu của mẩu tin đầu tiên xuất hiện trên *Form* như hình 21-6.



Hình 21-6: *Sử dụng BindingManagerBase*

Nếu người sử dụng chọn nút > (*Next*) thì mẩu tin kế tiếp trình bày trên các điều khiển. Để làm điều này, bạn khai báo trong biến cố *Click* của nút *btnNext* như ví dụ 21-16.

Ví dụ 21-16: Điều hướng đến mẩu tin kế tiếp

```
Private Sub btnNext_Click(ByVal sender As _
System.Object, ByVal e As System.EventArgs) _
Handles btnNext.Click
   ' Mẩu tin kế tiếp
   myBMB.Position += 1
   lbRecord.Text = (myBMB.Position + 1).ToString
End Sub
```

Khai báo trong biến cố *Click* của nút *btnPre* để cho phép người sử dụng di chuyển về mẩu tin trước đó so với mẩu tin hiện hành như ví dụ 21-17.

Ví dụ 21-17: Điều hướng đến mẩu tin trước đó

```
Private Sub btnPre_Click(ByVal sender As _
System.Object, ByVal e As System.EventArgs) _
```

```
Handles btnPre.Click
    ' Mẩu tin trước đó
    myBMB.Position -= 1
    lbRecord.Text = (myBMB.Position + 1).ToString
End Sub
```

Nếu cho phép người sử dụng trở về mẩu tin đầu tiên thì gán thuộc tính *Position* của đối tượng *BindingManagerBase* là 0 như ví dụ 21-18.

Ví dụ 21-18: Diều hướng đến mẩu tin đầu tiên

```
Private Sub btnFirst_Click(ByVal sender As _
System.Object, ByVal e As System.EventArgs) _
Handles btnFirst.Click
    ' Mẩu tin đầu tiên
    myBMB.Position = 0
    lbRecord.Text = (myBMB.Position + 1).ToString
End Sub
```

Tương tự như vậy, trong trường hợp cho phép người sử dụng di chuyển đến mẩu tin cuối cùng thì bạn gán thuộc tính *Position* bằng số mẩu tin đang có trong đối tượng *DataTable* như ví dụ 21-19.

Ví dụ 21-19: Diều hướng đến mẩu tin cuối cùng

```
Private Sub btnLast_Click(ByVal sender As _
System.Object, ByVal e As System.EventArgs) _
Handles btnLast.Click
    ' Mẩu tin cuối cùng
    myBMB.Position = myDT.Rows.Count - 1
    lbRecord.Text = (myBMB.Position + 1).ToString
End Sub
```

Lưu ý, để trình bày mẩu tin thứ i trên điều khiển *Label*, bạn sử dụng thuộc tính *Position* như sau:

```
lbRecord.Text = (myBMB.Position + 1).ToString
```

Chẳng hạn, bạn nhấn nút >| (*Last*), lập tức *Form* sẽ trình bày mẩu tin cuối cùng như hình 21-7.

Hình 21-7: *Mẩu tin cuối cùng*

Để tìm hiểu thêm về đối tượng *BindingManagerBase* và *DataBindings*, bạn có thể hoàn thành các ví dụ sau:

1. Liệt kê danh sách khách hàng, trình bày từng mẩu tin và điều hướng bằng cách dùng đối tượng *BindingManagerBase*, nếu cột dữ liệu là khóa ngoại, bạn sử dụng điều khiển dạng danh sách.
2. Thiết kế *Form* tương tự như ví dụ ở trên, mỗi khi người sử dụng chọn khách hàng, bạn liệt kê danh sách đơn đặt hàng trên điều khiển *DataGrid*.
3. Tạo *Project*, liệt kê danh sách đơn đặt hàng từng trang, mỗi trang là 200 mẩu tin.

6. KẾT LUẬN

Bạn vừa tìm hiểu cách làm việc với điều khiển *DataGrid* đơn giản, định dạng *DataGrid* với các điều khiển như *TextBox*, *CheckBox*, *ComboBox* bằng cách sử dụng phương thức và thuộc tính của đối tượng *DataGridTableStyle*, *DataGridTextBoxColumn*, *DataGridBoolColumn* và đối tượng *DataGridComboBoxColumn*.

Ngoài ra, trong chuyên đề này chúng ta cũng tìm hiểu cách trình bày mẩu tin trên điều khiển *TextBox* và cho người sử dụng di chuyển trên từng mẩu tin bằng đối tượng *BindingManagerBase*.

Mặc dù đã để *ByVal* nhưng *VB.NET* luôn xem các điều khiển là tham biến, chính vì vậy khi bạn khai báo phương thức để điền dữ liệu vào điều khiển điều khiển này được xem như tham biến cho dù bạn sử dụng từ khóa *ByVal*.

Giả sử, nếu bạn truyền điều khiển *TextBox* từ bên ngoài vào khi gọi phương thức thì mọi phép gán thay đổi giá trị của điều khiển đều tác động làm thay đổi đến điều khiển *TextBox* bên ngoài. Điều đó cho biết rằng tất cả các *Control* đều truyền theo tham biến (*ByRef*) bất chấp bạn có khai báo từ khóa là tham trị *ByVal* đi chăng nữa.

Trong chuyên đề kế tiếp, chúng ta sẽ tìm hiểu cách cài đặt lớp làm việc với cơ sở dữ liệu cho từng thực thể.

Chuyên đề 22:
LỚP DỮ LIỆU CHO TỪNG THỰC THỂ

Tóm tắt chuyên đề 22

Bạn vừa làm quen cách tương tác với cơ sở dữ liệu bằng nhiều hình thức như đọc trực tiếp bằng đối tượng SqlDataReader, SqlDataAdaper và trình bày dữ liệu bằng đối tượng DataSet hay DataTable từ những chuyên đề trước.

Trong chuyên đề này, chúng ta tiếp tục tìm hiểu cách xây dựng lớp tương tác với cơ sở dữ liệu.

Các vấn đề chính sẽ được đề cập:

- ✓ *Thảo luận giải pháp tương tác cơ sở dữ liệu.*
- ✓ *Xây dựng lớp bao gồm các thuộc tính và phương thức của từng bảng dữ liệu.*

1. THẢO LUẬN GIẢI PHÁP TƯƠNG TÁC CƠ SỞ DỮ LIỆU

Trong những chuyên đề trước, chúng ta thường khai báo và sử dụng một lớp có tên *clsDatabase*, bao gồm các phương thức và thuộc tính cho phép truy cập cơ sở dữ liệu với các mục đích truy vấn, cập nhật hay thực thi phát biểu *SQL* khác.

Hầu hết các phương thức trong lớp *clsDatabase* nhận tham trị là phát biểu *SQL*, sau đó thực thi phát biểu *SQL* và trả về kết quả là tập mẩu tin hay một mẩu tin đã thực thi.

Ngoài ra, trong lớp *clsDatabase* còn có một số phương tnức tương tác với cơ sở dữ liệu bằng cách truyền các giá trị cùng với tên của cột dữ liệu với *Parameters Collection* của đối tượng *SqlCommand* trong chuyên đề 16.

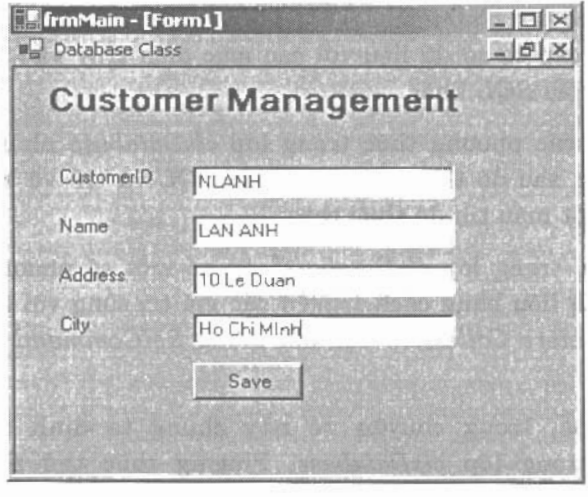
Tuy nhiên, trong chuyên đề này chúng ta định nghĩa lại các phương thức trong lớp *clsDatabase*. Phương thức thứ nhất chính là *ExecuteSQL* có cấu trúc như ví dụ 22-1.

Ví dụ 22-1: Phương thức thêm mẩu tin

```
Function ExecuteSQL(ByVal strSQL As String) As
String
    Dim strError As String = ""
    Dim myCon As New SqlConnection(gsCon)
    Try
      myCon.Open()
      Dim myCom As New SqlCommand(strSQL, myCon)
      myCom.ExecuteNonQuery()
    Catch ex As Exception
      strError = ex.Message
    Finally
      myCon.Close()
      myCon.Dispose()
    End Try
    Return strError
End Function
```

Như vậy, phương thức nhận tham số là phát biểu *SQL* dạng hành động dạng tổng quát. Chẳng hạn, bạn thiết kế *Form* có tên *frmCustomers* dùng để thêm mẩu tin vào bảng *tblCustomers* (được tạo ra trong chuyên đề trước) có giao diện như hình 22-1.

Mỗi khi người sử dụng nhấn nút *Save*, lập tức thông tin trên màn hình sẽ được thêm vào bảng *tblCustomers* nếu các thông tin đó đã được kiểm tra hợp lệ.



Hình 22-1: *Thêm danh sách khách hàng*

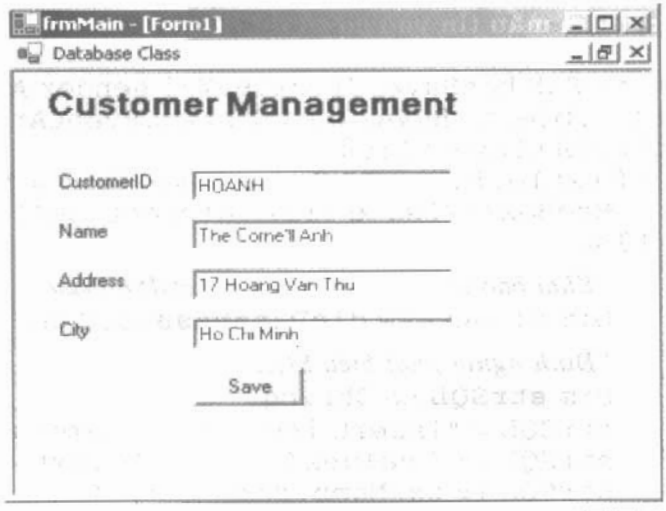
Tương tự như cách tương tác cơ sở dữ liệu đã trình bày trong các chuyên đề trước, bạn khai báo gọi phương thức *ExecuteSQL* vừa trình bày trong ví dụ 22-1 của lớp *clsDatabase* như ví dụ 22-2.

Ví dụ 22-2: Thêm mẩu tin vào cơ sở dữ liệu

```
Private Sub btnSave_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles btnSave.Click
    If txtID.Text = "" Or txtName.Text = "" Then
      MsgBox("Please fill information")
    Else
      ' Khai báo và khởi tạo đối tượng clsDatabase
      Dim cls As New clsDatabase(gsCon)
      ' Định nghĩa phát biểu SQL
      Dim strSQL As String
      strSQL = "Insert into tblCustomers "
      strSQL += " values('"+ txtID.Text + "','"
      strSQL += txtName.Text + "','"
      strSQL += txtAddress.Text + "','"
      strSQL += txtCity.Text + "')"
      ' Khai báo biến nhận chuỗi lỗi nếu phát sinh
      Dim strError As String
      ' Gọi phương thức ExecuteSQL
      strError = cls.ExecuteSQL(strSQL)
      If strError <> "" Then
        MsgBox(strError)
      Else
      ' Thêm mẩu tin thành công
        For Each txt As Control In Me.Controls
          If TypeOf (txt) Is TextBox Then
            txt.Text = ""
          End If
        Next
      End If
    End If
End Sub
```

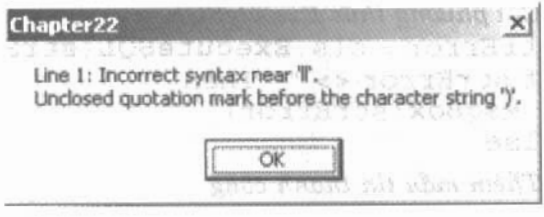
Trong ví dụ trên, chúng ta sử dụng phương thức *ExecuteNonQuery* của đối tượng *SqlCommand* để thực thi phát biểu *Insert* thêm mẩu tin vào bảng *tblCustomers*.

Lỗi sẽ phát sinh nếu bạn nhập giá trị có dấu nháy trong bất kỳ thông tin nào của khách hàng, do trong phát biểu *Insert* chưa xử lý dấu nháy. Ví dụ, bạn nhập tên của khách hàng có dấu nháy như hình 22-2.



Hình 22-2: *Nhập dữ liệu có dấu nháy*

Khi nhấn nút *Save*, lỗi sẽ phát sinh ngay sau dấu nháy trong phát biểu *SQL* như hình 22-3.



Hình 22-3: *Lỗi phát sinh*

Nguyên nhân phát sinh lỗi là do bạn chưa xử lý dấu nháy của phát biểu *SQL* khai báo trong ví dụ 22-2.

```
' Định nghĩa phát biểu SQL
Dim strSQL As String
strSQL = "Insert into tblCustomers values('"
strSQL += txtID.Text + "','"
strSQL += txtName.Text + "','"
strSQL += txtAddress.Text + "','"
strSQL += txtCity.Text + "')"
```

Vấn đề quan tâm là tên của khách hàng thật sự có dấu nháy chứ không phải là dấu của tiếng Việt. Đối với trường hợp này, bạn phải xử lý dấu nháy trước khi thực thi chúng bằng cách sử dụng phương thức *Replace* của đối tượng *Text* hoặc phương thức *Replace* của *Visual Basic.NET* như sau:

```
' Định nghĩa phát biểu SQL
Dim strSQL As String
strSQL = "Insert into tblCustomers values('"
strSQL += txtID.Text.Replace ("'","''") + "','"
strSQL += txtName.Text.Replace ("'","''")+ "','"
strSQL += txtAddress.Text.Replace ("'","''")
strSQL += "','"
strSQL += txtCity.Text.Replace ("'","''") + "')"
```

Lưu ý, bằng cách này bạn định nghĩa phát biểu *SQL* từ phần giao diện (*Form*) rồi khởi tạo đối tượng và truyền chúng vào phương thức để thực thi.

Như trình bày trong chuyên đề 16, bằng cách sử dụng đối tượng *SqlParameter* hoặc phương thức *Add* của *Parameters Collection* thuộc đối tượng *SqlCommand*, bạn cũng có thể thêm mẩu tin mà không cần quan tâm đến dấu nháy của giá trị nhập. Để làm điều này, bạn cài đặt phương thức *ExecuteSQL* như ví dụ 22-3.

Ví dụ 22-3: Thêm mẩu tin bằng đối tượng tham số

```
Function ExecuteSQL(ByVal strSQL As String,
ByVal arrParas As String()) As String
    Dim strError As String = ""
    Dim myCon As New SqlConnection(gsCon)
    Try
        ' Mở kết nối cơ sở dữ liệu
        myCon.Open()
        Dim myCom As New SqlCommand
        myCom.Connection = myCon
        myCom.CommandType = CommandType.Text
        ' Gán phát biểu SQL
        myCom.CommandText=strSQL
        ' Gán giá trị tương ứng với các tham số
        myCom.Parameters.Add("@ID", _
          arrParas(0))
```

```
            myCom.Parameters.Add("@Name", _
            arrParas (1))
            myCom.Parameters.Add("@Address", _
            arrParas (2))
            myCom.Parameters.Add("@City", _
            arrParas (3))
            'Gọi phương thức thực thi phát biểu SQL
            myCom.ExecuteNonQuery()
        Catch ex As Exception
            strError = ex.Message
        Finally
            myCon.Close()
            myCon.Dispose()
        End Try
        Return strError
    End Function
```

Sau đó, thêm nút thứ hai vào *frmCustomers* và khai báo để gọi phương thức này như ví dụ 22-4.

Ví dụ 22-4: Sử dụng Parameters Collection

```
Private Sub btnPara_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles btnPara.Click
    If txtID.Text = "" Or txtName.Text = "" Then
        MsgBox(" Please fill information ")
    Else
        'Khai báo và khởi tạo đối tượng clsDatabase
        Dim cls As New clsDatabase(gsCon)
        'Định nghĩa phát biểu SQL
        Dim strSQL As String
        strSQL = "insert into tblCustomers values "
        strSQL +=" (@ID,@Name,@Address,@City)"
        'Khai báo mảng chứa giá trị
        Dim arrPara(4) As String
        arrPara(0) = txtID.Text
        arrPara(1) = txtName.Text
        arrPara(2) = txtAddress.Text
        arrPara(3) = txtCity.Text
        Dim strError As String
        'Gọi phương thức ExecuteSQL
        StrError=cls.ExecuteSQL(strSQL,arrPara)
```

```
        If strError <> "" Then
          MsgBox(strError)
        Else
          ' Thêm mẩu tin thành công
          For Each txt As Control In Me.Controls
            If TypeOf (txt) Is TextBox Then
            txt.Text = ""
            End If
          Next
        End If
      End If
End Sub
```

Mặc dù trong ví dụ 22-4 đã trình bày cách sử dụng phương thức *Add* của *Parameters Collection* của đối tượng *SqlCommand* nhằm tránh việc xử lý dấu nháy, nhưng vẫn còn mang tính đặc thù cho từng bảng dữ liệu.

2. XÂY DỰNG LỚP BAO GỒM CÁC THUỘC TÍNH VÀ PHƯƠNG THỨC CỦA TỪNG BẢNG DỮ LIỆU

Theo như cách trình bày ở phần 1, chúng ta không nên sử dụng các phát biểu *SQL* để truy cập cơ sở dữ liệu, thay vào đó bạn nên sử dụng *Parameters* để truyền giá trị từ bên ngoài vào các phương thức của lớp giao tiếp cơ sở dữ liệu.

Song song với việc bảo vệ cơ sở dữ liệu bằng việc sử dụng đối tượng *SqlParameter*, chúng ta thử xem xét một giải pháp mang tính tham khảo là xây dựng một lớp cho từng thực thể dữ liệu, bao gồm những thuộc tính và phương thức dùng để xử lý dữ liệu cho từng thực thể.

Ngoài ra, khi làm việc với cơ sở dữ liệu, bạn nên khai báo và sử dụng các thủ tục nội tại trong cơ sở dữ liệu *SQL*, nhằm sử dụng hiệu quả các tham số của thủ tục.

Chẳng hạn, trong những chương trước chúng ta đã tạo ra bảng dữ liệu có tên *tblCustomers*. Để thêm dữ liệu vào bảng này, bạn có thể sử dụng phát biểu *Insert* trực tiếp hay khai báo thủ tục như ví dụ 22-5.

Ví dụ 22-5: Khai báo thủ tục thêm mẩu tin

```
CREATE PROC spCustomers
  @flag tinyint,
```

```
    @CustomerID varchar(10),
    @CustomerName varchar(50),
    @Address varchar(100),
    @City varchar(50)
as
    set nocount on
    declare @IsExit int
    set @IsExit =1
    -- Trường hợp thêm mẩu tin
    if @flag=0
      if not exists(select * from tblCustomers
        where CustomerID=@CustomerID)
        insert into tblCustomers
        values(@CustomerID,@CustomerName,
        @Address,@City)
      else
        set @IsExit =0
    -- Trường hợp cập nhật mẩu tin
    if @flag=1
      update tblCustomers
      set CustomerName=@CustomerName,
      Address=@Address,
      City=@City
      where CustomerID=@CustomerID
    -- Trường hợp xóa mẩu tin
    if @flag=2
      Delete from tblCustomers
      where CustomerID=@CustomerID
    set nocount off
    select @IsExit
```

Lưu ý, mỗi khi thêm hay cập nhật dữ liệu của bảng *tblCustomers*, thay vì sử dụng phát biểu *Insert* hay *Update* thì bạn dùng thủ tục trên với giá trị truyền vào *flag* để nhận biết trạng thái thêm hay cập nhật.

Thủ tục trả về giá trị bằng cách lấy giá trị của biến *IsExit*, nếu giá trị là 1 thì cập nhật hay thêm thành công, trong trường hợp tồn tại mẩu tin thì giá trị trả về là 0.

Trở lại vấn đề khai báo lớp, bao gồm các thuộc tính và phương thức xử lý dữ liệu của từng thực thể, bạn có thể tạo ra từng lớp tương ứng với từng thực thể trong cơ sở dữ liệu. Ví dụ, trường hợp này chúng ta có hai bảng dữ liệu *tblStaffs* và *tblCustomers* có cấu trúc như sau:

```
Create Table tblCustomers
```

```
(
    CustomerID varchar(10) Primary Key,
    CustomerName varchar(50),
    Address varchar(100),
    City varchar(50)
)

Create Table tblStaffs
(
    StaffID int identity(1,1) Primary Key,
    FirstName nvarchar(20),
    LastName nvarchar(30)
)
```

Để xây dựng lớp tương ứng với hai thực thể *tblCustomers*, bạn tạo mới *Class* và đặt tên tương ứng là *clsCustomers*.

Trong đó, cấu trúc lớp *clsCustomers* bao gồm các thuộc tính có kiểu dữ liệu tương thích với kiểu dữ liệu của các cột trong *SQL Server* như ví dụ 22-6.

Ví dụ 22-6: Cấu trúc lớp clsCustomers

```
Public Class clsCustomers
  Public CustomerID As String
  Public CustomerName As String
  Public Address As String
  Public City As String
End Class
```

Tương tự như vậy, bạn khai báo lớp *clsStaffs* có cấu trúc như ví dụ 22-7.

Ví dụ 22-7: Cấu trúc lớp clsStaffs

```
Public Class clsStaffs
  Public StaffID As Integer
  Public FirstName As String
  Public LastName As String
End Class
```

Kế đến, tuần tự khai báo các phương thức để thêm mẩu tin vào cơ sở dữ liệu, cập nhật hay xóa, truy vấn dữ liệu thông qua các thuộc tính này.

Tuy nhiên, do phần gán dữ liệu từ *Form* giao diện người sử dụng vào các thuộc tính của lớp đều giống nhau cho cả ba trường hợp thêm,

xóa, cập nhật. Chính vì vậy, chúng ta khai báo hai biến mảng chứa tên tham số và giá trị tương ứng như sau:

```
Dim strSQL As String = ""
' Khai báo mảng tham số
Dim myPara() As String = {"@flag", _
    "@CustomerID", "@CustomerName", _
    "@Address", "@City"}
' Khai báo mảng giá trị tương ứng với tham số
Dim myValue(5) As String
```

Tương tự như lớp *clsCustomers*, bạn cũng khai báo hai biến mảng chứa tên tham số và giá trị tương ứng để sử dụng cho các phương thức thêm mới, cập nhật xóa trong lớp *clsStaffs* như sau:

```
Dim strSQL As String = ""
' Khai báo mảng tham số
Dim myPara() As String = {"@flag", "@StaffID", _
    "@FirstName", "@LastName"}
' Khai báo mảng giá trị tương ứng với tham số
Dim myValue(4) As String
```

Để thêm mẩu tin vào bảng *tblCustomers* từ các thuộc tính trong lớp *clsCustomers*, bạn khai báo phương thức có tên *AddRecord* trong lớp *clsCustomers* như ví dụ 22-8.

Ví dụ 22-8: Thêm mẩu tin khách hàng

```
Public Function AddRecord() As String
    ' Khai báo tên phương thức thêm, cập nhật, xóa dữ liệu bảng
    ' tblCustomers trong SQL Server
    strSQL = "spCustomers "
    ' Trường hợp thêm mẩu tin
    myValue(0) = 0
    ' Gọi phương thức này để điền dữ liệu từ thuộc tính vào
    ' mảng myValue
    FillData()
    ' Gọi phương thức doSQL của đối tượng clsDatabase
    strError = cls.doSQL(strSQL, myPara, _
     myValue)
```

```
    Return strError
End Function
```

Trong đó, đối tượng *clsDatabase* đã được khai báo và khởi tạo trong *Constructor* của *Class* như ví dụ 22-9.

Ví dụ 22-9: Khai báo Constructor

```
' Khai báo biến chuỗi lỗi
Dim strError As String = ""
' Khai báo đối tượng clsDatabase
Dim cls As clsDatabase
Sub New()
    ' Khởi tạo đối tượng clsDatabase
    cls = New clsDatabase(gsCon)
End Sub
```

Kế đến, bạn khai báo phương thức có tên *FillData* cho phép điền dữ liệu từ các thuộc tính vào mảng giá trị có tên *myValue* như ví dụ 22-10.

Ví dụ 22-10: Phương thức điền giá trị

```
Private Sub FillData()
    myValue(1) = CustomerID
    myValue(2) = CustomerName
    myValue(3) = Address
    myValue(4) = City
End Sub
```

Tiếp tục, bạn khai báo phương thức cập nhật mẩu tin trong lớp *clsCustomers* có tên *UpdateRecord* như ví dụ 22-11.

Ví dụ 22-11: Phương thức cập nhật mẩu tin khách hàng

```
Public Function UpdateRecord() As String
    ' Phương thức thêm, cập nhật, xóa dữ liệu bảng
    ' tblCustomers trong SQL Server
    strSQL = "spCustomers "
    ' Trường hợp cập nhật mẩu tin
    myValue(0) = 1
    ' Gọi phương thức này để điền dữ liệu từ thuộc tính vào
    ' mảng myValue
    FillData()
```

```
    ' Gọi phương thức doSQL của đối tượng clsDatabase
    strError = cls.doSQL(strSQL, myPara, _
     myValue)
    Return strError
End Function
```

Từ ví dụ 22-8 và 22-11, bạn thêm phương thức có tên *DeleteRecord* để xóa mẩu tin của bảng *tblCustomers* trong lớp *clsCustomers* như ví dụ 22-11.

Ví dụ 22-11: Phương thức xóa mẩu tin khách hàng

```
Public Function DeleteRecord() As String
    ' Phương thức thêm, cập nhật, xóa dữ liệu bảng
    ' tblCustomers trong SQL Server
    strSQL = "spCustomers "
    ' Trường hợp xóa mẩu tin
    myValue(0) = 2
    ' Gọi phương thức này để điền dữ liệu từ thuộc tính vào
    ' mảng myValue
    FillData()
    ' Gọi phương thức doSQL của đối tượng clsDatabase
    strError = cls.doSQL(strSQL, myPara, _
     myValue)
    Return strError
End Function
```

Đối với trường hợp truy vấn dữ liệu, bạn có thể khai báo chung trong thủ tục *spCustomers*. Tuy nhiên, trong trường hợp này, chúng ta khai báo thủ tục *spGetCustomers* dùng để truy vấn bảng *Customers* bằng hai cách trích lọc.

Cách thứ nhất truy vấn mẩu tin khách hàng có mã bằng với giá trị mã khách hàng của thuộc tính *CustomerID*, cách thứ hai là liệt kê danh sách khách hàng có *City* bằng với giá trị của thuộc tính *City*.

Để làm điều này, bạn khai báo thủ tục có tên *spGetCustomers* có cấu trúc như ví dụ 22-12.

Ví dụ 22-12: Truy vấn bảng tblCustomers

```
Create proc spGetCustomers
  @flag tinyint,
  @CustomerID varchar(10),
```

```
    @CustomerName varchar(50),
    @Address varchar(100),
    @City varchar(50)
As
    -- Nếu flag=0 thì truy vấn một mẩu tin
    if @flag=0
      select * from tblCustomers
      where CustomerID=@CustomerID
    else
      if @flag=1
        -- Nếu flag=1 thì truy vấn nhiều mẩu tin
        select * from tblCustomers
        where City=@City
      else
        -- Truy vấn tất cả mẩu tin
        select * from tblCustomers
```

Lưu ý, trong thủ tục *spGetCustomers* chúng ta không sử dụng hai tham số *@CustomerName* và *@Address*.

Sau đó, khai báo phương thức *SelectRecord* nhận tham biến có tên *myDT* là đối tượng *DataTable* và tham trị có tên *iQuery* là số *Short*, 0 ứng với trường hợp truy vấn mẩu tin có mã bằng giá trị của thuộc tính *CustomerID*, 1 nếu truy vấn những mẩu tin có *City* bằng giá trị của thuộc tính *City*, ngược lại thì truy vấn tất cả khách hàng như ví dụ 22-13.

Ví dụ 22-13: Truy vấn dữ liệu

```
Public Function SelectRecord(ByRef myDT As
DataTable, ByVal iQuery As Short) As String
    ' Thủ tục truy vấn dữ liệu bảng tblCustomers trong
    ' SQL Server
    strSQL = "spGetCustomers "
    ' Trường hợp xóa mẩu tin
    myValue(0) = iQuery
    ' Gọi phương thức này để điền dữ liệu từ thuộc tính vào
    ' mảng myValue
    FillData()
    ' Gọi phương thức getValue của đối tượng clsDatabase
    strError = cls.getValue(myDT, strSQL, _
      myPara, myValue)
```

```
    Return strError
End Function
```

Đối với trường hợp lấy ra một mẩu tin để cập nhật (*Edit*), bạn có thể cài đặt *Overload* phương thức *SelectRecord* để đọc từng cột giá trị và gán chúng vào các thuộc tính của lớp *clsCustomers* tương ứng như ví dụ 22-14.

Ví dụ 22-14: Lấy ra một mẩu tin

```
Public Function SelectRecord() As String
    ' Khai báo và khởi tạo đối tượng DataTable
    Dim myDT As New DataTable
    ' Khai báo phát biểu SQL
    strSQL = "spGetCustomers "
    ' Truy vấn một mẩu tin ứng với mã khách hàng chỉ định
    myValue(0) = 0
    ' Điền dữ liệu vào mảng giá trị và tham số
    FillData()
    ' Gọi phương thức getValue của lớp clsDatabase
    strError = cls.getValue(myDT, strSQL, _
      myPara, myValue)
    ' Nếu tồn tại mẩu tin
    If myDT.Rows.Count > 0 Then
      ' Lấy giá trị của cột tương ứng gán vào thuộc tính
      CustomerName = _
       Convert.ToString(myDT.Rows(0).Item(1))
      Address = _
      Convert.ToString(myDT.Rows(0).Item(2))
      City = _
      Convert.ToString(myDT.Rows(0).Item(3))
    End If
    Return strError
End Function
```

Trong đó, phương thức *getValue* được cài đặt *Overload* trong lớp *clsDatabase* như ví dụ 22-15.

Ví dụ 22-15: Cấu trúc phương thức getValue

```
Function getValue(ByRef myDT As DataTable,
ByVal strSP As String, ByVal strPara() As
String, ByVal strValue() As String) As String
```

```
        Dim strError As String = ""
        Dim myCon As New SqlConnection
        myCon.ConnectionString = gsCon
        Try
          myCon.Open()
          ' Khai báo và khởi tạo đối tượng SqlCommand
          Dim myCom As New SqlCommand
          myCom.Connection = myCon
          ' Định nghĩa loại phát biểu dùng cho đối tượng
          ' SqlCommand
          myCom.CommandType = _
            CommandType.StoredProcedure
          ' Khai báo tên thủ tục nội tại
          myCom.CommandText = strSP
          ' Khai báo đối tượng SqlParameter
          Dim myPara As SqlParameter
          ' Duyệt trên từng phần tử tham số
          For i As Integer = 0 To strPara.Length - 1
            ' Khởi tạo đối tượng SqlParameter
            myPara = New SqlParameter(strPara(i), _
            strValue(i))
            ' Thêm đối tượng SqlParameter vào đối tượng
            ' SqlCommand
            myCom.Parameters.Add(myPara)
          Next
          ' Khai báo và khởi tạo đối tượng SqlDataAdapter
          Dim myData As New SqlDataAdapter(myCom)
          ' Điền dữ liệu vào đối tượng DataTable
          myData.Fill(myDT)
        Catch ex As Exception
          strError = ex.Message
        Finally
          myCon.Close()
          myCon.Dispose()
        End Try
        Return strError
    End Function
```

Lưu ý, trong các ví dụ trình bày phương thức thêm, xóa, cập nhật dữ liệu bảng *tblCustomers* như: *AddRecord*, *UpdateRecord*, *DeleteRecrod*,

chúng ta có sử dụng phương thức *doSQL* (cài đặt *Overloading*) của lớp *clsDatabase* có cấu trúc như ví dụ 22-16.

Ví dụ 22-16: Phương thức doSQL

```
Function doSQL(ByVal strSP As String, ByVal
strPara() As String, ByVal strValue() As String)
As String
    Dim strError As String = ""
    Dim myCon As New SqlConnection(gsCon)
    Try
      myCon.Open()
      ' Khai báo và khởi tạo đối tượng SqlCommand
      Dim myCom As New SqlCommand
      myCom.Connection = myCon
      ' Định nghĩa loại phát biểu dùng cho đối tượng
      ' SqlCommand
      myCom.CommandType = _
        CommandType.StoredProcedure
      ' Khai báo tên thủ tục nội tại
      myCom.CommandText = strSP

      ' Khai báo đối tượng SqlParameter
      Dim myPara As SqlParameter
      ' Duyệt trên từng phần tử giá trị
      For i As Integer = 0 To strValue.Length - 1
        ' Khởi tạo đối tượng SqlParameter
        myPara = New SqlParameter(strPara(i), _
        strValue(i))
        ' Thêm đối tượng SqlParameter vào đối tượng
        ' SqlCommand
        myCom.Parameters.Add(myPara)
      Next
      ' Thực thi thủ tục nội tại
      myCom.ExecuteNonQuery()
    Catch ex As Exception
      strError = ex.Message
    Finally
      myCon.Close()
      myCon.Dispose()
    End Try
```

```
    Return strError
End Function
```

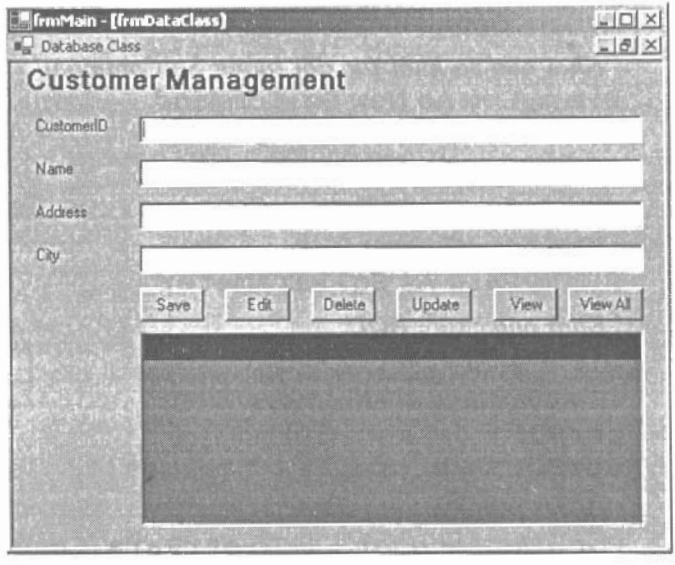
Ngoài ra, bạn có thể viết lại phương thức *doSQL* ở trên thành phương thức *ExecuteSQL* sử dụng chung cho mọi bảng dữ liệu ứng với tác vụ thêm mẩu tin bằng cách khai báo thêm tham trị là tên bảng dữ liệu như ví dụ 22-17.

Ví dụ 22-17: Thêm mẩu tin vào bảng dữ liệu

```
Function ExecuteSQL(ByVal strTable As String,
ByVal strPara() As String, ByVal strValue() As
String) As String
    Dim strError As String = ""
    Dim myCon As New SqlConnection(gsCon)
    Try
      myCon.Open()
      ' Khai báo và khởi tạo đối tượng SqlCommand
      Dim myCom As New SqlCommand
      myCom.Connection = myCon
      ' Chọn loại phát biểu
      myCom.CommandType = CommandType.Text
      ' Khai báo đối tượng SqlParameter
      Dim myPara As SqlParameter
      ' Khai báo chuỗi SQL
      Dim strSQL As String
      ' Định nghĩa phát biểu Insert
      strSQL = "insert into "
      strSQL + =strTable + " values("
      ' Duyệt trên từng tham số
      For i As Integer = 0 To strPara.Length - 1
        ' Thêm tham số vào phát biểu Insert
        strSQL += strPara(i) + ","
        ' Khởi tạo đối tượng SqlParameter
        myPara = New SqlParameter(strPara(i), _
        strValue(i))
      ' Thêm đối tượng SqlParameter vào đối tượng
      ' SqlCommand
        myCom.Parameters.Add(myPara)
      Next
      strSQL = strSQL.Substring(0, _
```

```
            strSQL.Length - 1) + ")"
            myCom.CommandText = strSQL
            ' Thực thi phát biểu SQL
            myCom.ExecuteNonQuery()
        Catch ex As Exception
            strError = ex.Message
        Finally
            myCon.Close()
            myCon.Dispose()
        End Try
        Return strError
    End Function
```

Sau khi kết thúc khai báo lớp *clsCustomers*, bạn thêm *Form* vào *Project* với tên *frmDataClass* và thiết kế giao diện như hình 22-4.



Hình 22-4: *Sử dụng lớp dữ liệu*

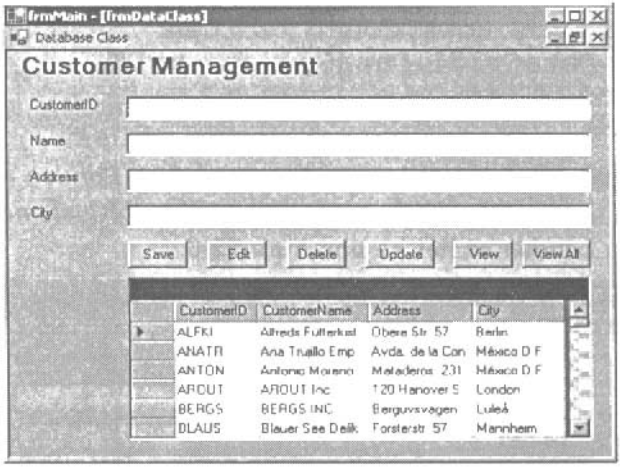
Để sử dụng lớp *clsCustomers*, trước tiên bạn khai báo trong biến cố *Click* của nút *btnViewAll* như ví dụ 22-18, cho phép người sử dụng liệt kê tất cả mẩu tin trong bảng *tblCustomers*.

Ví dụ 22-18: Liệt kê tất cả mẩu tin

```
Private Sub btnView_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles btnView.Click
```

```
        'Xóa dữ liệu trong đối tượng DataTable
        myDT.Rows.Clear()
        'Phương thức điền dữ liệu từ TextBox vào các thuộc tính
        SetData()
        cls.SelectRecord(myDT, 2)
        'Trình bày dữ liệu trên điều khiển DataGrid
        If myDT.Rows.Count > 0 Then
            DataGrid1.DataSource = myDT
        End If
    End Sub
```

Khi người sử dụng nhấn nút *View All*, lập tức danh sách mẩu tin có trong bảng *tblCustomers* trình bày như hình 22-5.



Hình 22-5: *Liệt kê danh sách khách hàng*

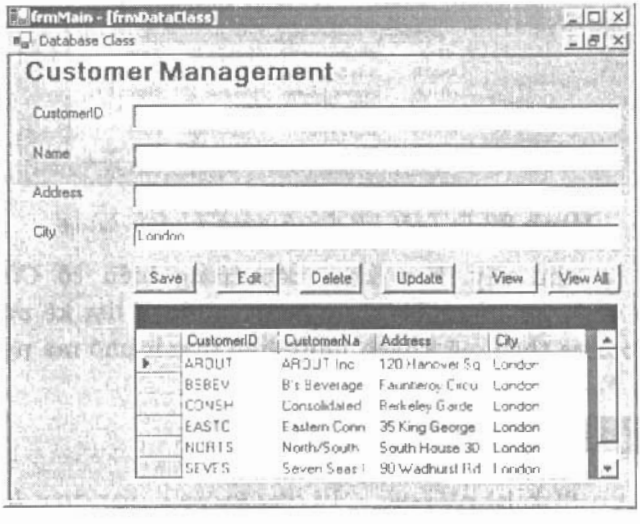
Tương tự như vậy, bạn khai báo trong biến cố *Click* của nút *btnView* như ví dụ 22-18-1, cho phép người sử dụng liệt kê mẩu tin trong bảng *tblCustomers* theo mã khách hàng hay thành phố mà người sử dụng đã nhập trên *Form*.

Ví dụ 22-18-1: Liệt kê mẩu tin có chọn lọc

```
Private Sub btnView_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles btnView.Click
        'Xóa dữ liệu trong đối tượng DataTable
        myDT.Rows.Clear()
```

```
    ' Phương thức điền dữ liệu từ TextBox vào các thuộc tính
    SetData()
    ' Nếu người sử dụng không nhập mã hay thành phố, chương
    ' trình liệt kê tất cả khách hàng trong bảng tblCustomers
    If txtID.Text = "" And txtCity.Text = "" Then
      cls.SelectRecord(myDT, 2)
    ElseIf txtID.Text <> "" Then
      ' Nếu lọc theo mã khách hàng
      cls.SelectRecord(myDT, 0)
    ElseIf txtCity.Text <> "" Then
      ' Nếu lọc theo thành phố
      cls.SelectRecord(myDT, 1)
    End If
    If myDT.Rows.Count > 0 Then
      DataGrid1.DataSource = myDT
    End If
End Sub
```

Nếu người sử dụng nhập vào thành phố là *London* và nhấn nút *View*, lập tức danh sách khách hàng trong bảng *tblCustomers* có giá trị của cột *City* là *London* sẽ liệt kê như hình 22-6.



Hình 22-6: *Liệt kê khách hàng có chọn lọc*

Trong đó, phương thức *SetData* đọc giá trị trên các *TextBox* điền vào các thuộc tính tương ứng đã khai báo trong lớp *clsCustomers* như ví dụ 22-19.

Ví dụ 22-19: Gán giá trị vào thuộc tính

```
' Khai báo và khởi tạo đối tượng clsCustomers
Dim cls As New clsCustomers
' Khai báo và khởi tạo đối tượng DataTable
Dim myDT As New DataTable
Sub SetData()
    cls.CustomerID = txtID.Text
    cls.CustomerName = txtName.Text
    cls.Address = txtAddress.Text
    cls.City = txtCity.Text
End Sub
```

Để cho phép người sử dụng kích hoạt mẩu tin của khách hàng được chỉ định trong điều khiển *CustomerID*, bạn khai báo trong biến cố của nút *btnEdit* như ví dụ 22-20.

Ví dụ 22-20: Kích hoạt mẩu tin

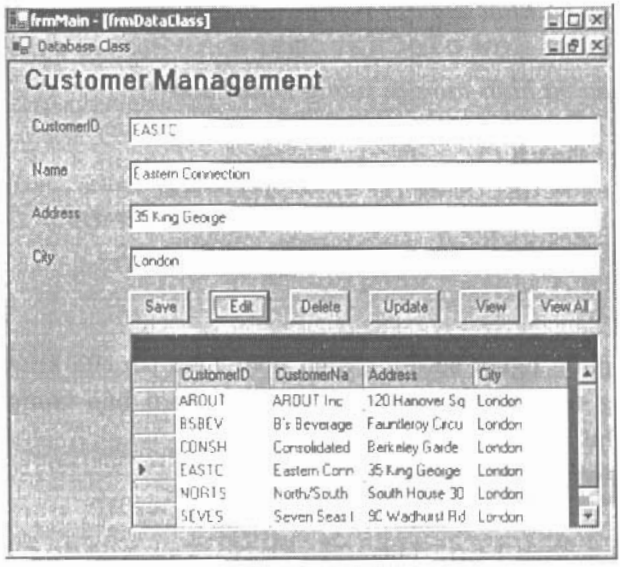
```
Private Sub btnEdit_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles btnEdit.Click
    If txtID.Text = "" Then
      MsgBox("Please enter CustomerID")
    Else
      SetData()
      cls.SelectRecord()
      GetData()
      ' Vô hiệu hóa điều khiển mã khách hàng
      txtID.Enabled = False
    End If
End Sub
```

Đoạn chương trình trên sẽ gọi phương thức có tên *SetData* để điền giá trị trên các *TextBox* vào các thuộc tính của lớp *clsCustomers*, kế đến gọi phương thức *SelectRecord* để lấy ra mẩu tin và gán giá trị của từng cột vào các thuộc tính của lớp *clsCustomers* như ví dụ 22-14.

Sau đó, tiếp tục gọi phương thức *GetData* để lấy giá trị từ các thuộc tính của lớp *clsCustomers* để gán vào các *TextBox* tương ứng trên *Form*.

Như vậy, mỗi khi người sử dụng muốn kích hoạt một khách hàng nào đó, họ có thể nhập mã khách hàng trong phần *CustomerID* hay chọn

mã khách hàng từ điều khiển *DataGrid* bên dưới, rồi nhấn nút *Edit* thì thông tin của khách hàng sẽ trình bày như hình 22-7.



Hình 22-7: *Kích hoạt thông tin khách hàng*

Lưu ý, để cho phép người sử dụng lấy giá trị mã khách hàng từ điều khiển *DataGrid* mỗi khi họ chọn vào một mẩu tin, bạn khai báo trong biến cố *DoubleClick* như ví dụ 22-21.

Ví dụ 22-21: Chọn và kích hoạt khách hàng

```
Private Sub DataGrid1_DoubleClick(ByVal sender
As Object, ByVal e As System.EventArgs) Handles
DataGrid1.DoubleClick
    txtID.Text = _
  DataGrid1.Item(DataGrid1.CurrentRowIndex, 0)
    Call btnEdit_Click(sender, e)
End Sub
```

Sau khi *Edit* một mẩu tin, để cho phép người sử dụng cập nhật thông tin khách hàng, bạn có thể khai báo trong biến cố *Click* của nút *btnUpdate* như ví dụ 22-22.

Ví dụ 22-22: Cập nhật mẩu tin

```
Private Sub btnUpdate_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles btnUpdate.Click
```

```
    If txtID.Text = "" Then
       Exit Sub
    Else
      ' Gọi phương thức điền thông tin vào thuộc tính
      SetData()
      ' Gọi phương thức UpdateRecord trong lớp clsCustomers
       strError = cls.UpdateRecord()
       If strError <> "" Then
         MsgBox(strError)
       Else
         MsgBox("Update Successfull")
       End If
    End If
End Sub
```

Tương tự như trường hợp cập nhật mẩu tin, nếu cho phép người sử dụng xóa mẩu tin hiện hành, bạn khai báo trong biến cố *Click* của nút *btnDelete* như ví dụ 22-23.

Ví dụ 22-23: Xóa mẩu tin

```
Private Sub btnDelete_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles btnDelete.Click
    If txtID.Text = "" Then
       Exit Sub
    Else
      ' Gọi phương thức điền thông tin vào thuộc tính
      SetData()
      ' Gọi phương thức DeleteRecord trong lớp clsCustomers
       strError = cls.DeleteRecord()
       If strError <> "" Then
         MsgBox(strError)
       Else
         MsgBox("Delete Successfull")
       End If
    End If
End Sub
```

Tuy nhiên, khi xóa mẩu tin bạn luôn yêu cầu khách hàng xác nhận hành động của họ để tránh trường hợp nhầm lẫn. Để làm điều này bạn sử dụng hàm *MsgBox* như ví dụ 22-23-1.

Ví dụ 22-23-1: Xóa mẩu tin có xác nhận

```
Private Sub btnDelete_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles btnDelete.Click
    If txtID.Text = "" Then
       Exit Sub
     Else
       ' Gọi phương thức điền thông tin vào thuộc tính
       SetData()
       If MsgBox("Sure?", MsgBoxStyle.OKCancel, _
        "") = MsgBoxResult.OK Then
       ' Gọi phương thức DeleteRecord trong lớp clsCustomers
        strError = cls.DeleteRecord()
        If strError <> "" Then
          MsgBox(strError)
        Else
          MsgBox("Delete Successfull")
        End If
      End If
    End If
End Sub
```

Đối với trường hợp thêm mẩu tin, bạn cũng khai báo trong biến cố *Click* của nút *btnSave* như ví dụ 22-24.

Ví dụ 22-24: Thêm mẩu tin

```
Private Sub btnSave_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles btnSave.Click
    If txtID.Text = "" Or txtName.Text = "" Then
       MsgBox("Please enter valid data")
     Else
       ' Gọi phương thức điền thông tin vào thuộc tính
       SetData()
       ' Gọi phương thức AddRecord trong lớp clsCustomers
        strError = cls.AddRecord()
        If strError <> "" Then
          MsgBox(strError)
        Else
          MsgBox("Insert Successfull")
        End If
```

```
    End If
End Sub
```

Ngoài ra, bạn thêm nút *btnNew* cho phép người sử dụng xóa thông tin đang tồn tại trên các điều khiển và cho phép người sử dụng nhập mã khách hàng như ví dụ 22-25.

Ví dụ 22-25: Xóa thuộc tính đang tồn tại

```
Private Sub btnNew_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles btnNew.Click
    txtID.Enabled = True
    txtName.Text = ""
    txtAddress.Text = ""
    txtCity.Text = ""
End Sub
```

Lưu ý, sau khi thêm nút *btnNew* vào *Form*, bạn có thể kết hợp hai nút *btnSave* và *btnUpdate* thành một, khi đó dựa vào thuộc tính *Enabled* của điều khiển *txtID* để nhận biết khi nào thì gọi phương thức *UpdateRecord* hay *AddRecord* trong lớp *clsCustomers*.

Chẳng hạn, bạn khai báo lại đoạn chương trình trong biến cố *Click* của nút *btnSave* như ví dụ 22-26.

Ví dụ 22-26: Cập nhật và thêm mới mẩu tin

```
Private Sub btnSave_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles btnSave.Click
    If txtID.Text = "" Or txtName.Text = "" Then
       MsgBox("Please enter valid data")
    Else
      ' Gọi phương thức điền thông tin vào thuộc tính
       SetData()
       If txtID.Enabled=True Then
         ' Gọi phương thức AddRecord trong lớp clsCustomers
         strError = cls.AddRecord()
       Else
         ' Gọi phương thức UpdateRecord trong lớp clsCustomers
         strError = cls.UpdateRecord()
```

```
            End If
            If strError <> "" Then
              MsgBox(strError)
            Else
              MsgBox("Insert/Update Successfull")
            End If
          End If
    End Sub
```

Lưu ý, chuyên đề cuối cùng của tập sách này sẽ trình bày cho bạn cách xây dựng lớp dùng chung cho dự án phần mềm viết bằng *Visual Basic.NET*.

Để tìm hiểu thêm cách xây dựng lớp làm việc với cơ sở dữ liệu cho từng thực thể, bạn có thể tạo lớp làm cho bảng dữ liệu có tên *tblStaffs* như sau:

1. Xây dựng *Shared Assembly* có tên *SQLDatabase* bao gồm các phương thức cho phép thực hiện các tác vụ thêm, cập nhật, xóa và truy vấn dữ liệu *SQL Server*.
2. Tạo *Class* và đặt tên *clsStaffs*, kế đến bạn khai báo thuộc tính, phương thức để tương tác với cơ sở dữ liệu.
3. Viết ứng dụng *Form*, cho phép người sử dụng tương tác với cơ sở dữ liệu thông qua lớp *clsStaffs* cùng với lớp *clsDatabase* trong *SQLDatabase*.
4. Xây dựng lớp dùng chung cho mọi trường hợp tương tác với cơ sở dữ liệu.

3. KẾT LUẬN

Chúng ta vừa tìm hiểu cách xây dựng lớp làm việc với cơ sở dữ liệu ứng với từng thực thể. Bạn có thể tham khảo cách xây dựng lớp dùng cho mọi trường hợp tương tác với cơ sở dữ liệu trong phần bài tập.

Trong chuyên đề kế tiếp, chúng ta sẽ tiếp tục tìm hiểu cách thiết kế, tương tác, đọc dữ liệu, lọc và xuất dữ liệu với *Crystal Report*.

Chuyên đề 23:

KHÁM PHÁ CRYSTAL REPORT

Tóm tắt chuyên đề 23

Chuyên đề 22 cung cấp cho bạn giải pháp kết nối và xử lý dữ liệu, ngoài ra bạn cũng đã làm quen lớp chứa thuộc tính và phương thức tương tác với cơ sở dữ liệu.

Trong chuyên đề này, chúng ta tiếp tục tìm hiểu cách trình bày dữ liệu bằng Crystal Report.

Các vấn đề chính sẽ được đề cập:

- ✓ *Thiết kế Report bằng Crystal Report.*
- ✓ *Tương tác Report từ Visual Basic.NET.*
- ✓ *Xuất dữ liệu ra định dạng khác.*

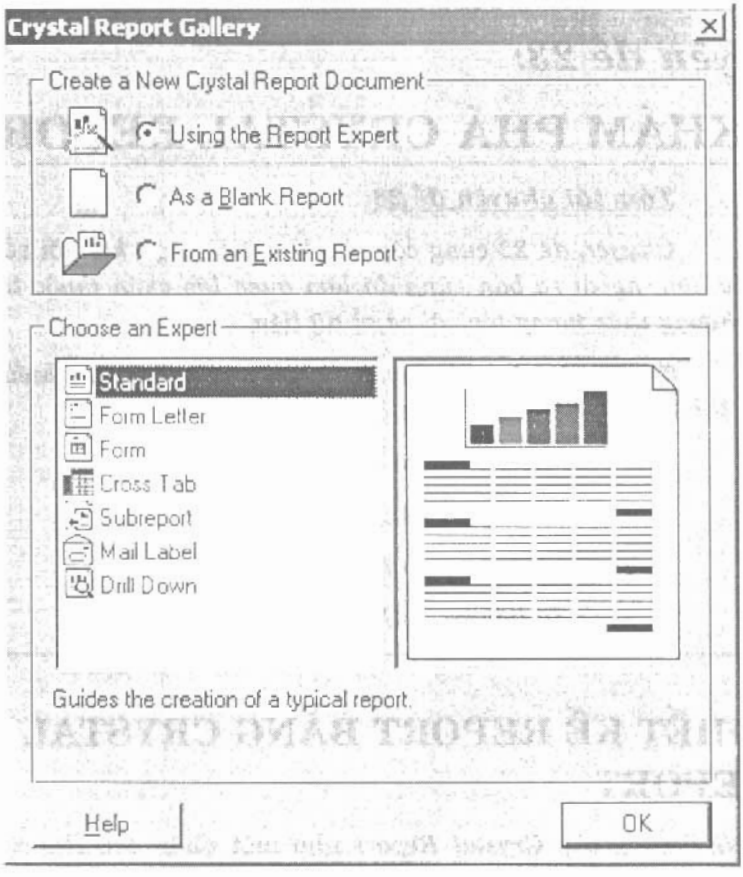
1. THIẾT KẾ REPORT BẰNG CRYSTAL REPORT

.NET cung cấp *Crystal Report* như một phần của *Project*, bạn có thể thêm báo cáo (*Report*) bằng cách chọn *Project* | *Add New Item* | *Crystal Report*.

Mỗi đối tượng *Crystal Report* được tạo ra đều xuất hiện trong danh sách thành phần của dự án, tên mở rộng của chúng là *.rpt*. Tuy nhiên, bạn cũng có thể thiết kế *Report* từ ứng dụng *Crystal Report*, sau đó thêm chúng vào *Project*.

Crystal Report cung cấp nhiều hình dạng, cho phép bạn thiết kế *Report* theo các mẫu như: *Standard*, *Letter*, *Cross-Tab*, *Mail Label* và *Subreport*,... Trong trường hợp bạn chọn vào tùy chọn *"Using the Report Expert"* thì cửa sổ xuất hiện như hình 23-1.

Nếu bạn tự quyết định thiết kế *Report* mà không cần sự trợ giúp của *Crystal Report* thì chọn vào tùy chọn *"As a Blank Report"*. Tuy nhiên, nếu thiết kế *Report* từ một *Report* đang tồn tại thì chọn vào tùy chọn *"From an Existing Report"*.



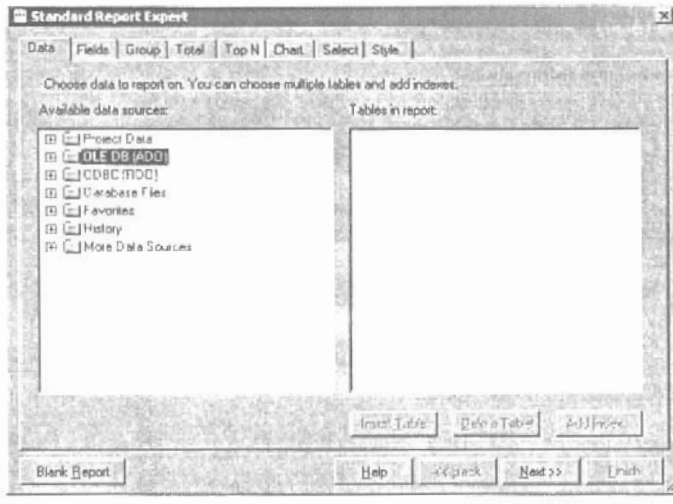
Hình 23-1: *Các loại Report*

Bây giờ chúng ta tìm hiểu cách thiết kế *Report* dạng chuẩn bằng cách chọn vào *Project* | *Add New Item* | *Crystal Report*. Kế đến, bạn chọn vào biểu tượng *Crystal Report* | đặt tên *Standard.rpt*.

Chọn tùy chọn mặc định là "*Using the Report Expert*" rồi chọn vào *Standard*, một cửa sổ hỗ trợ thiết kế xuất hiện như hình 23-2.

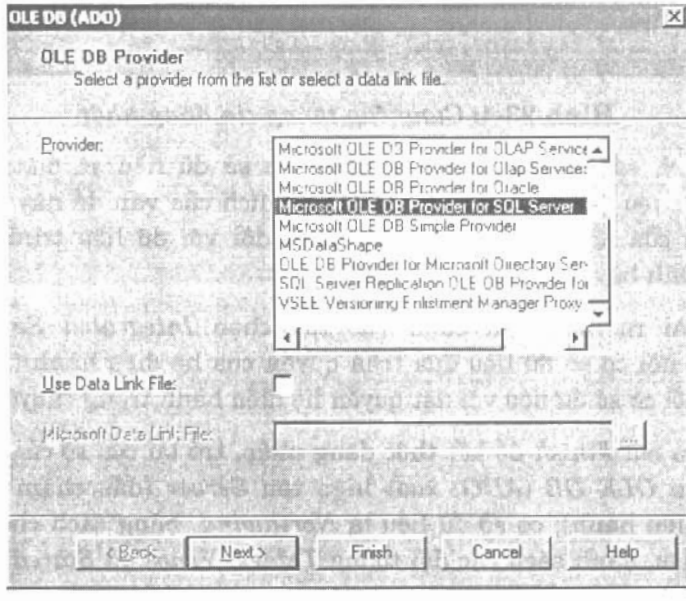
Từ ngăn *Data*, bạn chọn vào một trong những tùy chọn như *Project Data* (nếu trước đó đã tạo kết nối cơ sở dữ liệu), *OLE DB (ADO)* để tạo mới kết nối cơ sở dữ liệu: *Database Files* hay *More Data Sources* chọn tập tin *XML*,

Giả sử, trong trường hợp hợp tạo *Crystal Report* lần đầu tiên, bạn chọn vào tùy chọn *OLE DB (ADO)*, một cửa sổ chọn trình điều khiển cơ sở dữ liệu xuất hiện.



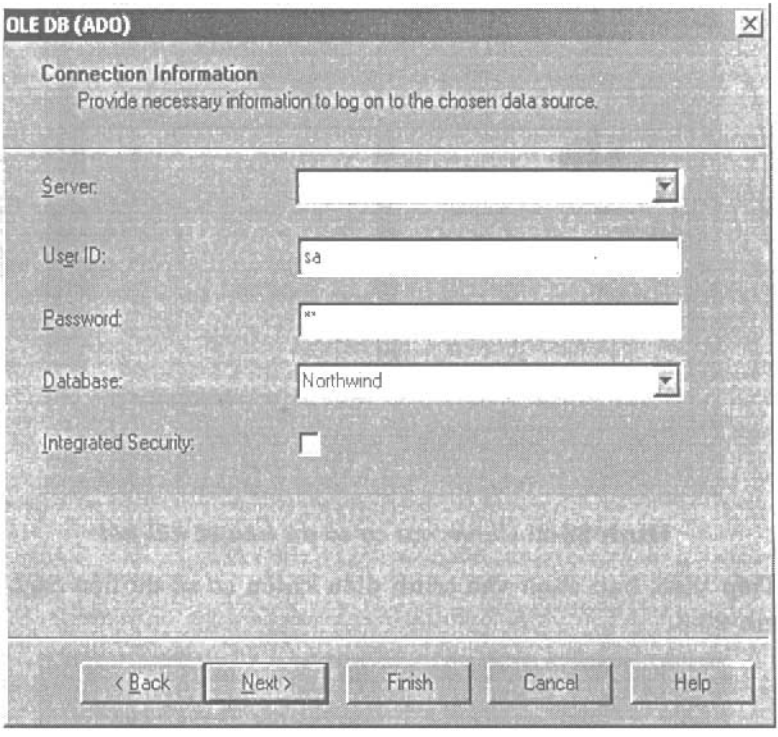
Hình 23-2: *Chọn loại cơ sở dữ liệu sẽ kết nối*

Tiếp theo, bạn chọn vào trình điều khiển cơ sở dữ liệu *SQL Server* như hình 23-3.



Hình 23-3: *Chọn trình điều khiển cơ sở dữ liệu*

Sau khi chọn nút *Next*, cửa sổ yêu cầu đăng nhập cơ sở dữ liệu *SQL Server* xuất hiện như hình 23-4.

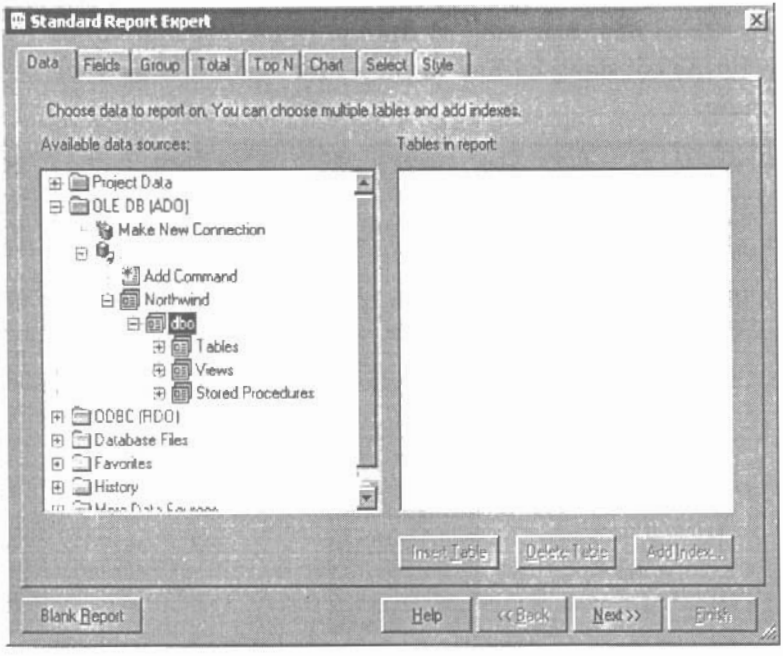


Hình 23-4: *Cung cấp thông tin đăng nhập*

Lưu ý, các thông tin đăng nhập cơ sở dữ liệu sẽ được cập nhật trong khi triệu gọi đối tượng *Report*, mục đích của vấn đề này là sử dụng đặt quyền của người sử dụng hiện hành đối với dữ liệu trình bày trên *Report* (trình bày trong phần kế tiếp).

Ngoài ra, nếu bạn chọn vào tùy chọn *Integrated Security*, đặt quyền kết nối cơ sở dữ liệu dựa trên quyền của hệ điều hành (tham khảo cách kết nối cơ sở dữ liệu với đặt quyền hệ điều hành trong chuyên đề 16).

Chọn nút *Finish* để kết thúc đăng nhập, trở lại cửa sổ của hình 23-2, trong phần *OLE DB (ADO)* xuất hiện tên *Server* (dấu chấm tương ứng với máy hiện hành), cơ sở dữ liệu là *Northwind*, bằng cách chọn vào tên cơ sở dữ liệu, danh sách các đối tượng *Tables, Views* và *Stored Procedure* xuất hiện như hình 23-5.



Hình 23-5: *Danh sách thực thể cơ sở dữ liệu*

Tiếp tục chọn vào ngăn *Views* và chọn *View* có tên "*Product Sales for 1997*", rồi nhấn nút *Insert Table* để thêm vào ngăn "*Tables in Report*". Chọn tên *Table* hay *View* và nhấn nút *Insert Table* trong trường hợp *Report* của bạn cần trình bày dữ liệu trên nhiều *Tables*.

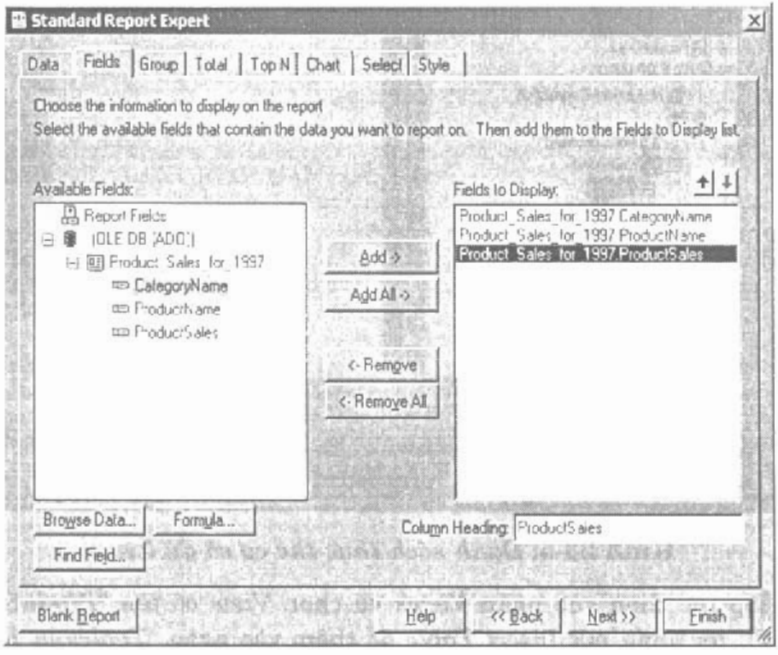
Lưu ý, nếu bạn tạo *Report* khác thì phần kết nối cơ sở dữ liệu đã tồn tại trong ngăn *Project Data | Current Connections* thay vì tạo mới như trường hợp đầu tiên.

Sau khi đã chọn danh sách các bảng hay đối tượng *View* vào danh sách *Tables in Report*, chọn *Next* để tiếp tục thì cửa sổ mới sẽ xuất hiện như hình 23-6.

Bằng cách chọn những cột dữ liệu cần trình bày trên *Report* từ ngăn *Available Fields*, bạn nhấn nút Add hay *Add All* vào ngăn *Fields to Display*.

Trong trường hợp muốn loại bỏ cột dữ liệu đang có trong ngăn *Fields to Display* thì chọn *Remove* hay *Remove All*.

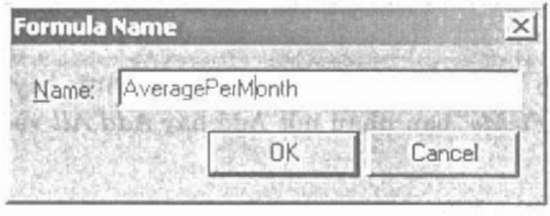
Từ cửa sổ này, bạn cũng có thể tạo ra các cột mới với dữ liệu được kết hợp từ các cột đang có hay là biểu thức nào đó bằng cách chọn vào nút *Formula*.



Hình 23-6: *Danh sách cột dữ liệu*

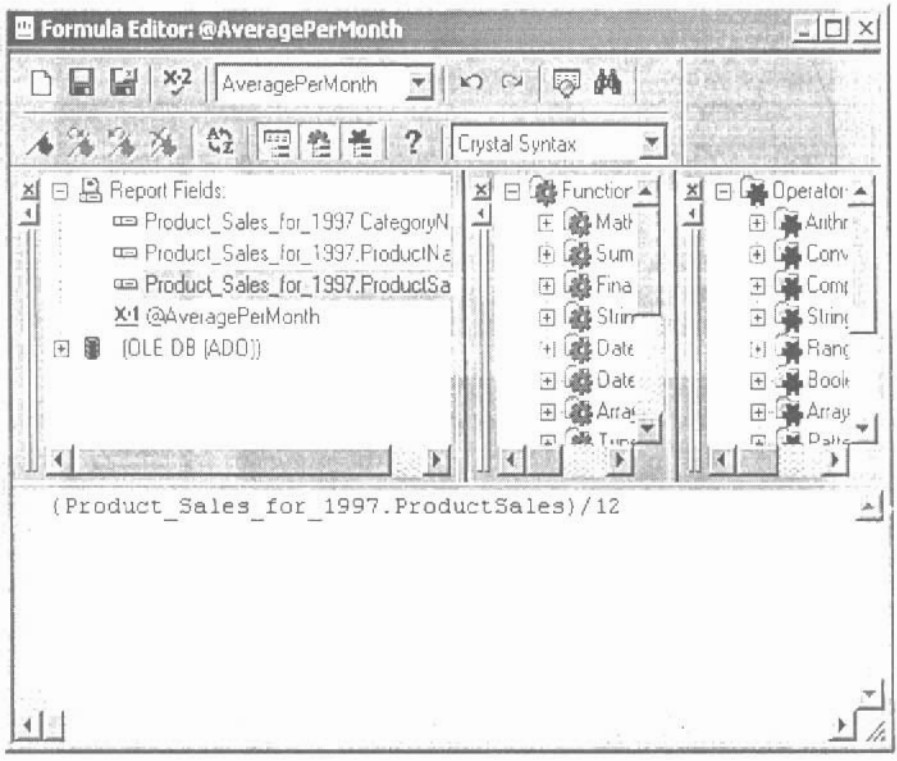
Chẳng hạn, chúng ta muốn tạo ra cột để trình bày số liệu bán hàng trung bình cho mỗi tháng bằng cách lấy số liệu từ cột *ProductSales* chia cho 12

Để làm điều này, bạn chọn vào nút *Formula*, cửa sổ xuất hiện như hình 23-7, đặt tên *AveragePerMonth*.



Hình 23-7: *Tạo Formula*

Sau khi nhấn nút *OK*, lập tức cửa sổ dùng để định nghĩa biểu thức xuất hiện như hình 23-8.



Hình 23-8: *Khai báo biểu thức*

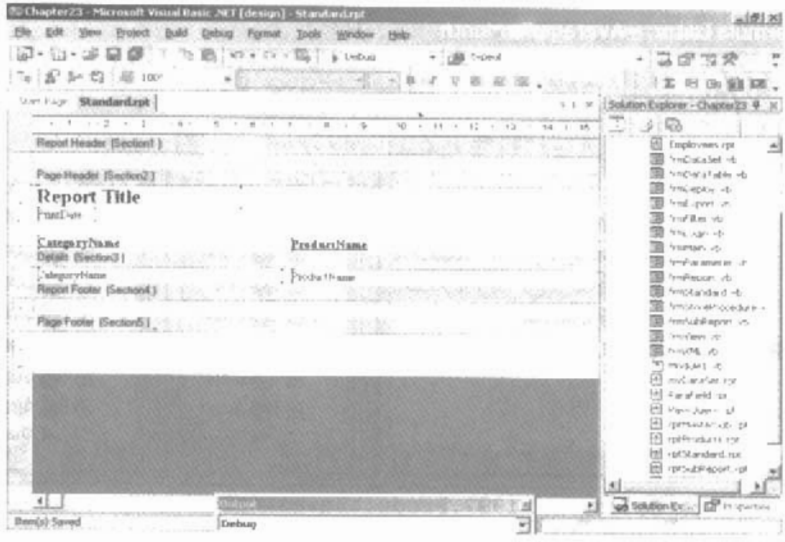
Nhấn biểu tượng *Save* trên thanh công cụ của cửa sổ hình 23-8 để lưu lại biểu thức, sau đó chọn nút *Close* để đóng cửa sổ và kết thúc khai báo biểu tượng cho *Formula*.

Trở lại cửa sổ hình 23-6, bạn sẽ tìm thấy tên *AveragePerMonth* của *Formula* vừa tạo xuất hiện, chọn *AveragePerMonth* và nhấn nút *Add* để thêm cột này vào danh sách *Fields to Display*.

Tiếp tục chọn *Next*, bạn có thể khai báo nhóm dữ liệu theo một cột nào đó, đối với trường hợp này chúng ta không khai báo nhóm dữ liệu.

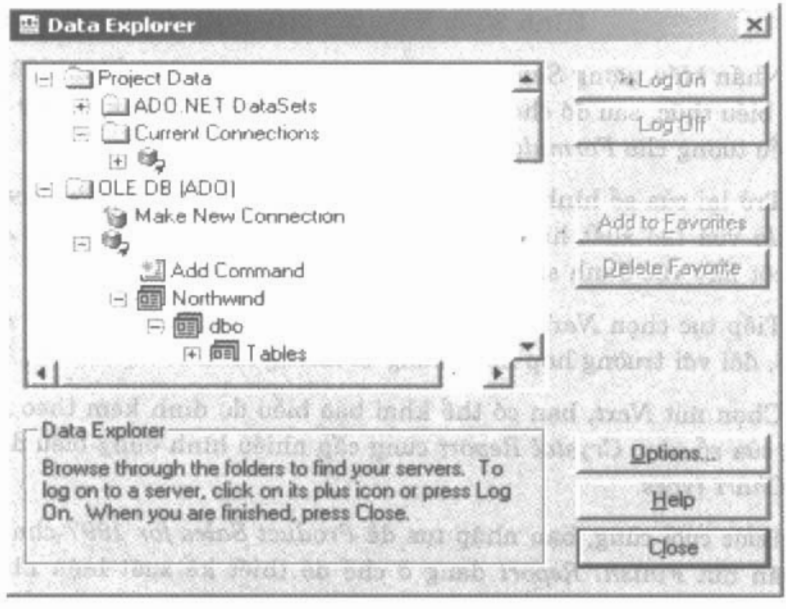
Chọn nút *Next*, bạn có thể khai báo biểu đồ đính kèm theo *Report*. Trong cửa sổ này, *Crystal Report* cung cấp nhiều hình dạng biểu đồ thuộc ngăn *Chart types*.

Bước cuối cùng, bạn nhập tựa đề *Product Sales for 1997* cho *Report* và nhấn nút *Finish*. *Report* đang ở chế độ thiết kế xuất hiện như hình 23-9.



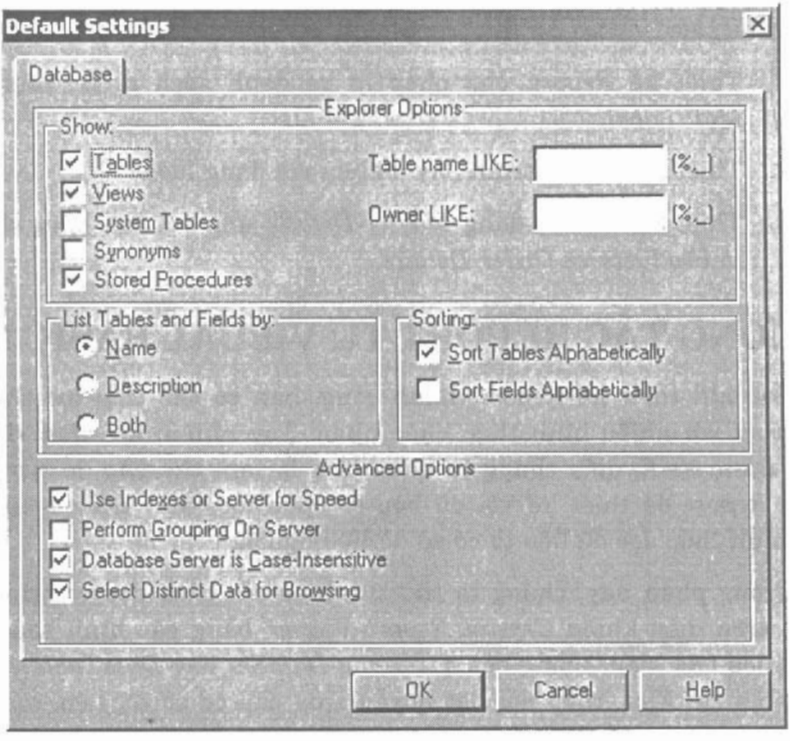
Hình 23-9: *Thiết kế Report*

Lưu ý, nếu bạn không tìm thấy danh sách các thủ tục nội tại của cơ sở dữ liệu *SQL Server* trong hình 23-5, bạn có thể *R-Click* trên *Report* của cửa sổ hình 23-9 và chọn vào *Database | Log On/Off Server*, cửa sổ xuất hiện như hình 23-10.



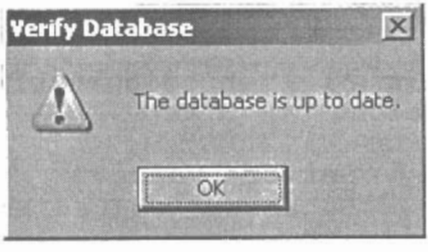
Hình 23-10: *Chọn tùy chọn cho Report*

Nhấn nút *Options*, cửa sổ cho phép chọn các tùy chọn xuất hiện như hình 23-11, bạn chọn vào tùy chọn *Stored Procedures* của ngăn *Show*.



Hình 23-11: *Cho phép xuất hiện thủ tục nội tại*

Ngoài ra, để thêm mới đối tượng *Table, View* hay *Stored Procedure* vào *Report* đang tồn tại, bạn thực hiện tương tự như trên bằng cách *R-Click* trên *Report* và chọn vào *Database* | *Add/Remove Database*.



Hình 23-12: *Cập nhật cấu trúc*

Tương tự như vậy, nếu cấu trúc của *Table, View* hay *Stored Procedure* có thay đổi sau khi kết thúc thiết kế *Report*, bạn có thể cập

nhật bằng cách *R-Click* trên *Report* và chọn vào *Database* | *Verify Database*, cửa sổ cập nhật thành công xuất hiện như hình 23-12.

Để tìm hiểu thêm về *Crystal Report*, bạn có thể thiết kế các loại *Report* như sau:

1. Thiết kế *Report*, cho phép in ra danh sách nhân viên dạng *Mail Label*.
2. Thiết kế *Report* trình bày dữ liệu với dạng lưới.
3. Thiết kế *Report* dạng *Master-Details* ứng với hai bảng dữ liệu là *Products* và *Order Details*.

2. TƯƠNG TÁC REPORT TỪ VISUAL BASIC.NET

Sau khi thiết kế *Report* thành công, bạn có thể tiếp tục thiết kế các *Report* với nhiều hình thức khác nhau. Tuy nhiên, khi làm việc với *Visual Basic.NET*, điều chúng ta quan tâm là làm thế nào để tương tác với các *Report* đã thiết kế với dữ liệu có chọn lọc, thay đổi quyền đăng nhập, hình thức đọc dữ liệu từ cơ sở dữ liệu nguồn.

Trong phần này, chúng ta thử tìm hiểu các cách đọc và trình bày dữ liệu trên điều khiển *Crystal Report Viewer* bằng các hình thức như: Đọc dữ liệu trực tiếp, thông qua đối tượng *DataSet* hay *DataTable*, đọc dữ liệu từ tập tin *Xml*, thay đổi đặt quyền truy cập cơ sở dữ liệu, làm việc với thủ tục nội tại và *Report* dạng *Master-Detail*.

2.1. Khai báo dùng chung

Để chuẩn bị làm việc với *Crystal Report*, trước tiên khai báo tập tin *App.config* như ví dụ 23-1.

Ví dụ 23-1: Cấu trúc tập tin App.config

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
 <appSettings>
    <add key="Server" value="."/>
    <add key="Database" value="Northwind"/>
 </appSettings>
</configuration>
```

Kế đến, khai báo các biến sử dụng chung cho dự án tương ứng với *Server, Database, UserName, Password* trong *Module* như ví dụ 23-2.

Ví dụ 23-2: Khai báo biến dùng chung

```
Imports System.Configuration
Module Module1
    Public gsServer As String
    Public gsDatabase As String
    Public gsUser As String
    Public gsPwd As String

    ' Khai báo phương thức Main để khởi động chương trình
    Sub Main()

        ' Đọc giá trị gán vào biến gsServer, gsDatabase
        gsDatabase = _
ConfigurationSettings.AppSettings("Database")
        gsServer = _
ConfigurationSettings.AppSettings("Server")

        ' Khởi tạo đối tượng Form chính
        Dim myForm As New frmMain
        myForm.ShowDialog()
    End Sub
End Module
```

Sau đó, khai báo *frmLogin* để yêu cầu người sử dụng nhập *UserName, Password* để đăng nhập cơ sở dữ liệu *SQL Server* như hình 23-13.

Hình 23-13: *Đăng nhập hệ thống*

Để kiểm tra quyền sử dụng cơ sở dữ liệu của người sử dụng, bạn khai báo trong biến cố *Click* của nút *btnLogin* như ví dụ 23-2-1.

Ví dụ 23-2-1: Đăng nhập

```
Private Sub btnLogin_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles btnLogin.Click
    Dim cls As clsDatabase
    Try
        'Mở kết nối cơ sở dữ liệu thành công
        cls = New clsDatabase("server=" + _
        gsServer + ";database=" + gsDatabase + _
        ";uid=" + txtUser.Text + ";pwd=" + _
        txtPwd.Text)
        cls.CloseConnection()
        gsUser = txtUser.Text
        gsPwd = txtPwd.Text
        Me.Close()
    Catch ex As Exception
        MsgBox(ex.Message)
    End Try
End Sub
```

Trong mỗi *Form* trình bày báo cáo, bạn cần khai báo để sử dụng hai không gian tên chính như sau:

```
Imports CrystalDecisions.CrystalReports.Engine
Imports CrystalDecisions.Shared
```

Đối tượng để nắm giữ *Crystal Report* là *ReportDocument* cùng với việc sử dụng điều khiển *Crystal Report Viewer*.

2.2. Mở Report trực tiếp

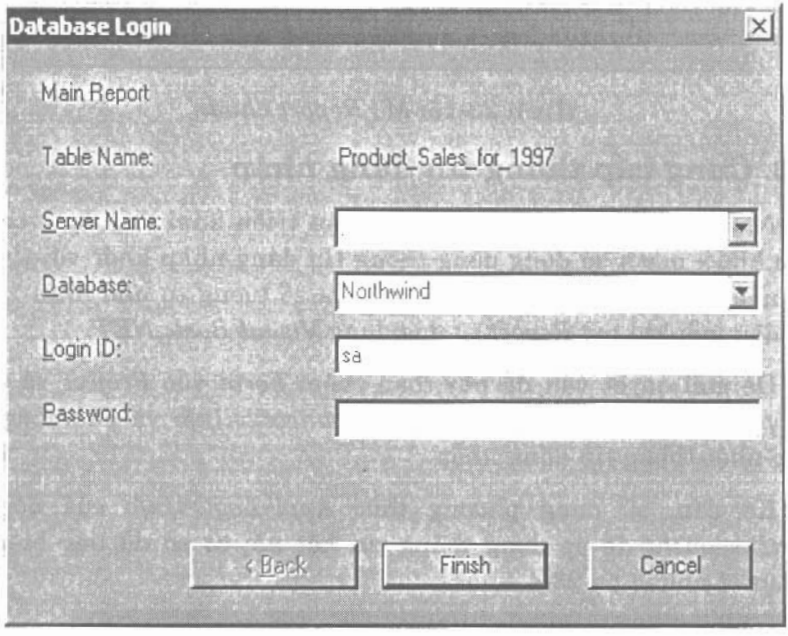
Sau khi thiết kế *Report* có tên *Standard.rpt* trong phần 1, bằng cách thêm *Form* vào *project* và đặt tên *frmStandard*. Kế đến, thêm điều khiển *Crystal Report Viewer* và nút *Preview*.

Bằng cách khai báo đoạn chương trình để khởi tạo đối tượng *ReportDocument* ứng với *Report* có tên *Standard*, sau đó gán đối tượng này vào thuộc tính *ReportSource* của điều khiển *Crystal Report Viewer* như ví dụ 23-3.

Ví dụ 23-3: Mở Crystal Report chuẩn

```
Private Sub Button1_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles Button1.Click
    Dim objRep As New ReportDocument
    objRep = New Standard
    RV.ReportSource = objRep
    RV.Refresh()
    objRep.Dispose()
End Sub
```

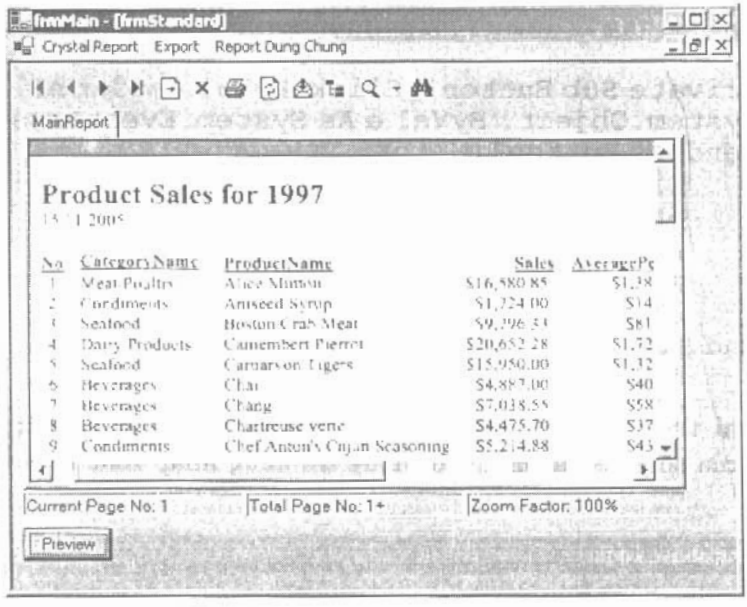
Khi thực thi chương trình, nếu người sử dụng nhấn nút *Preview*, lập tức cửa sổ yêu cầu cung cấp thông tin đăng nhập xuất hiện như hình 23-14.



Hình 23-14: *Yêu cầu đăng nhập*

Lưu ý: Các thông tin đăng nhập trong cửa sổ trên mặc định là thông tin khi bạn tạo kết nối cơ sở dữ liệu trong khi thiết kế *Report*.

Sau khi cung cấp *Password* hợp lệ, lập tức *Report* trình bày như hình 23-15.



Hình 23-15: *Mở Report chuẩn*

2.3. Cung cấp thông tin đăng nhập

Như trình bày ở phần trên, khi bạn triển khai ứng dụng trên một hỗ trợ khác, người sử dụng dùng thông tin đăng nhập khác với thông tin đăng nhập trong khi thiết kế *Report*, cửa sổ tương tự như hình 23-14 sẽ xuất hiện mỗi khi mở *Report* từ ứng dụng *Visual Basic.NET*.

Để giải quyết vấn đề này, bạn thêm *Form* vào *Project* và đặt tên *frmDeploy*, sau đó sử dụng đối tượng *ConnectionInfo* và *TableLogOnInfo* để cập nhật thông tin đăng nhập.

Kế đến, sử dụng phương thức *ApplyLogOnInfo* của đối tượng *ReportDocument* để áp dụng thông tin kết nối cơ sở dữ liệu hiện hành cho *Report* như ví dụ 23-4.

Ví dụ 23-4: Cập nhật thông tin đăng nhập cơ sở dữ liệu

```
Private Sub Button1_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles Button1.Click

    ' Khai báo và khởi tạo đối tượng ReportDocument
    Dim objRep As ReportDocument
    objRep = New Standard
```

```
    ' Khai báo và khởi tạo đối tượng ConnectionInfo
    Dim myCon As New ConnectionInfo
    ' Khai báo và khởi tạo đối tượng TableLogOnInfo
    Dim myInfo As New TableLogOnInfo
    ' Khai báo lại thông tin đăng nhập
    With myCon
       .ServerName = gsServer
       .DatabaseName = gsDatabase
       .UserID = gsUser
       .Password = gsPwd
    End With
    ' Gán đối tượng ConnectionInfo vào thuộc tính
    ' ConnectionInfo của đối tượng TableLogOnInfo
    myInfo.ConnectionInfo = myCon
    ' Định nghĩa tên bảng dữ liệu hay View
    Dim myTable As String = _
       " Product Sales for 1997 "
    ' Áp dụng đối tượng TableLogOnInfo cho bảng dữ liệu thứ
    ' 0 trong Report
    objRep.Database.Tables( _
       0).ApplyLogOnInfo(myInfo)
    ' Gán đối tượng ReportDocument vào thuộc tính
  ReportSource
    RV.ReportSource = objRep
    RV.Refresh()
    objRep.Dispose()
End Sub
```

Lưu ý: Nếu tên bảng hay *View* trong cơ sở dữ liệu tồn tại khoảng trắng, bạn có thể sử dụng số thứ tự để chỉ định đối tượng *Table* hay *View* bên trong *Report*.

```
objRep.Database.Tables(0 _
).ApplyLogOnInfo(myInfo)
```

Hoặc

```
objRep.Database.Tables(myTable _
).ApplyLogOnInfo(myInfo)
```

Khi thực thi chương trình, nếu người sử dụng nhấn nút *Preview*, lập tức *Report* trình bày như hình 23-16 với thông tin định nghĩa hiện hành

Hình 23-16: *Đăng nhập cơ sở dữ liệu*

2.4. Lọc dữ liệu từ điều khiển Crystal Report Viewer

Mặc dù *Report* đang nắm giữ một tập dữ liệu, bạn cũng có thể lọc dữ liệu để trình bày theo một tiêu chí nào đó. Chẳng hạn, bạn thêm mới *Form* và đặt tên *frmFilter*. Sau đó, khai báo chuỗi lọc cho thuộc tính *SelectionFormula* của điều khiển *Crystal Report Viewer* như ví dụ 23-5.

Ví dụ 23-5: Lọc dữ liệu

```
Private Sub Button1_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles Button1.Click
    ' Khai báo và khởi tạo đối tượng ReportDocument
```

```
    Dim objRep As ReportDocument
    objRep = New Filter
    'Khai báo và khởi tạo đối tượng ConnectionInfo
    Dim myCon As New ConnectionInfo
    'Khai báo và khởi tạo đối tượng TableLogOnInfo
    Dim myInfo As New TableLogOnInfo
    'Khai báo lại thông tin đăng nhập
    With myCon
       .ServerName = gsServer
       .DatabaseName = gsDatabase
       .UserID = gsUser
       .Password = gsPwd
    End With
    'Gán đối tượng ConnectionInfo vào thuộc tính
    'ConnectionInfo của đối tượng TableLogOnInfo
    myInfo.ConnectionInfo = myCon
    'Định nghĩa tên bảng dữ liệu hay View
    Dim myTable As String = _
       " Customers "
    'Áp dụng đối tượng TableLogOnInfo cho bảng dữ liệu thứ
    '0 trong Report
    objRep.Database.Tables( _
      myTable).ApplyLogOnInfo(myInfo)
    'Khai báo lọc dữ liệu
    RV.SelectionFormula = _
    "{Customers.City}<>'London'"
    'Gán đối tượng ReportDocument vào thuộc tính
  ReportSource
    RV.ReportSource = objRep

    RV.Refresh()
    objRep.Dispose()
End Sub
```

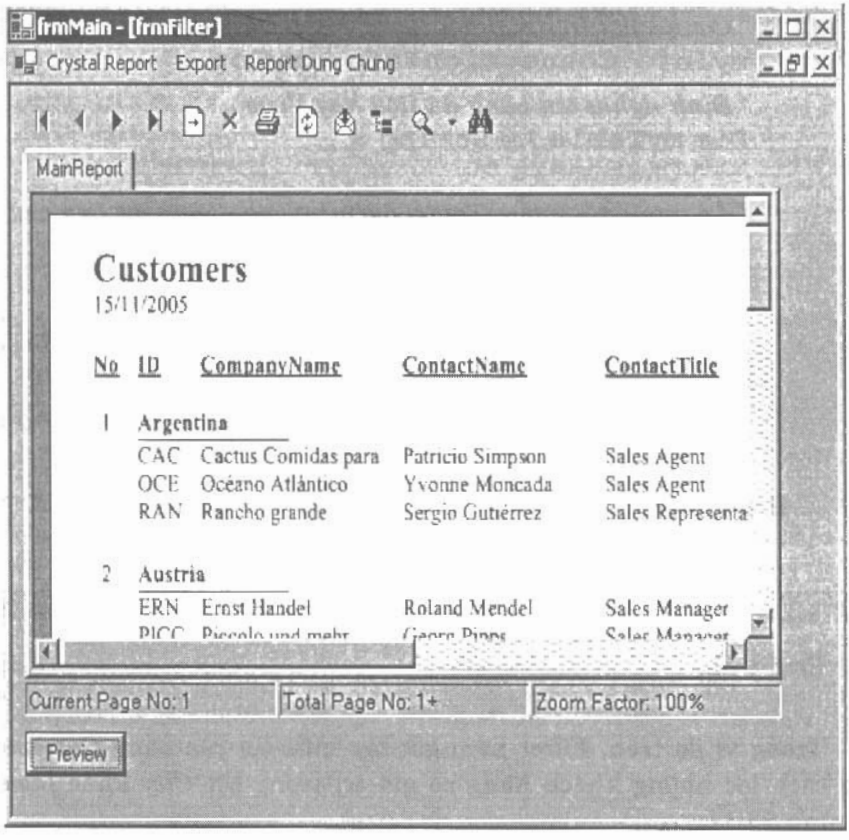
Trong ví dụ trên, *Filter* nắm giữ tập mẩu tin của bảng *Customers*, bằng cách lọc những khách hàng có giá trị trong cột *City* khác *London* như hình 23-17.

Chú ý, chúng ta có nhiều cách để lọc dữ liệu khi trình bày trên *Report* (trình bày trong những phần kế tiếp). Tuy nhiên, đối với trường hợp này, bạn sử dụng thuộc tính *SelectionFormula* của điều khiển *Crystal Report Viewer*.

2.5. Điền dữ liệu vào Crystal Report từ đối tượng DataSet

Trong những trường hợp trên, các *Report* đều đọc dữ liệu bằng cách kết nối theo hình thức *OLE DB (ADO)*, đồng thời bạn sử dụng hai đối tượng *ConnectionInfo* và *TableLogOnInfo* để cập nhật thông tin đăng nhập.

Tuy nhiên, một trong những điểm mạnh của *Visual Basic.NET* là lớp không kết nối, phải kể đến hai đối tượng *DataSet* và *DataTable*.



Hình 23-17: *Lọc dữ liệu*

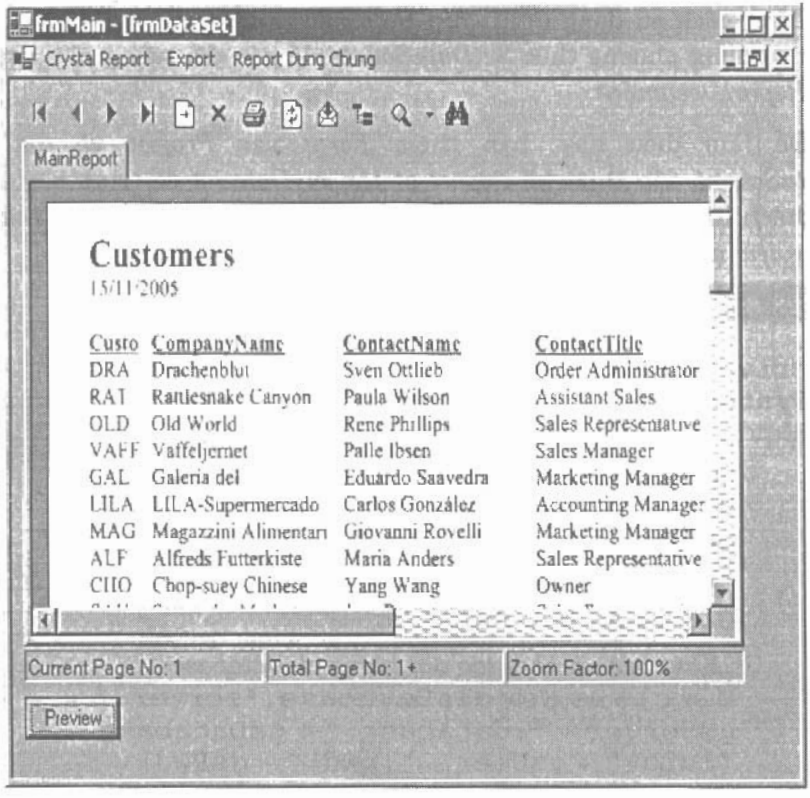
Bằng cách sử dụng đối tượng *DataSet* để nắm giữ tập dữ liệu, sau đó bạn sử dụng phương thức *SetDataSource* để gán đối tượng này vào đối tượng *ReportDocument*.

Để làm điều này, bạn thêm *Form* vào *Project* và đặt tên *frmDataSet*, kế đến thiết kế *Report* có tên *myDataSet* để trình bày danh sách khách hàng từ bảng *Customers*, rồi khai báo trong biến cố *Click* của nút *Preview* như ví dụ 23-6.

Ví dụ 23-6: Trình bày dữ liệu từ đối tượng DataSet

```
Private Sub Button1_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles Button1.Click
    ' Khai báo và khởi tạo đối tượng ReportDocument
    Dim objRep As ReportDocument
    objRep = New myDataSet
    ' Khai báo và khởi tạo đối tượng DataSet
    Dim ds As New DataSet
    ' Khai báo và khởi tạo đối tượng clsDatabase
    Dim cls As New clsDatabase("server=" + _
    gsServer + ";database=" + gsDatabase + _
    ";uid=" + gsUser + ";pwd=" + gsPwd)
    ' Khai báo phát biểu SQL
    Dim strSQL As String = _
    "select * from Customers"
    ' Gọi phương thức getValue
    cls.getValue(ds, strSQL)
    ' Gán đối tượng DataTable vào đối tượng ReportDocument
    objRep.SetDataSource(ds.Tables(0))
    RV.ReportSource = objRep
    RV.Refresh()
    objRep.Dispose()
End Sub
```

Khi thực thi chương trình, thay vì *Report* đọc dữ liệu trực tiếp từ cơ sở dữ liệu *SQL Server*, bạn đọc dữ liệu bằng đối tượng *ADO.NET*, sau đó điền chúng vào đối tượng *ReportDocument*, kết quả trình bày như hình 23-18.



Hình 23-18: *Điền dữ liệu từ đối tượng DataSet*

Trong đó, lớp *clsDatabase* được định nghĩa với 2 phương thức chính, phương thức *getValue* (điền mẫu tin vào đối tượng *DataSet*) và *CloseConnection* (đóng kết nối cơ sở dữ liệu) cùng với *Constructor* (mở kết nối cơ sở dữ liệu) như ví dụ 23-7.

Ví dụ 23-7: Cấu trúc lớp clsDatabase

```
Imports System.Data.SqlClient
Public Class clsDatabase
  Private myCon As SqlConnection
  'Khai báo Constructor với tham số
  Sub New(ByVal strCon As String)
    myCon = New SqlConnection(strCon)
    If myCon.State = ConnectionState.Closed
Then
      myCon.Open()
    End If
```

```
    End Sub
    ' Khai báo Constructor
    Sub New()
    End Sub
    ' Khai báo phương thức đóng kết nối cơ sở dữ liệu
    Public Sub CloseConnection()
      If myCon.State <> _
         ConnectionState.Closed Then
        myCon.Close()
        myCon.Dispose()
      End If
    End Sub
    ' Khai báo hàm điền mẩu tin vào đối tượng DataSet
    Function getValue(ByRef myDS As DataSet, _
      ByVal strSQL As String) As String
      Dim strError As String = ""
      Try
        Dim myData As New SqlDataAdapter( _
        strSQL, myCon)
        myDS = New DataSet
        myData.Fill(myDS, strSQL)
      Catch ex As Exception
        strError = ex.Message
      End Try
      Return strError
    End Function
  End Class
```

Lưu ý, không đăng nhập vào cơ sở dữ liệu bằng *OLE DB* của *Crystal Report* mà sử dụng đối tượng *ADO.NET*. Ngoài ra, bằng cách này, bạn có thể lọc dữ liệu từ phát biểu *Select*, thủ tục nội tại hay sử dụng đối tượng *DataView* để lọc dữ liệu.

2.6. Điền dữ liệu vào Crystal Report từ đối tượng DataTable

Tương tự như trường hợp trên, bạn khai báo thêm phương thức *getValue* (*Overload*) trong lớp *clsDatabase* để điền mẩu tin vào đối tượng *DataTable* như ví dụ 23-8.

Ví dụ 23-8: Cấu trúc lớp clsDatabase

```
    ' Khai báo hàm điền mẩu tin vào đối tượng DataTable
```

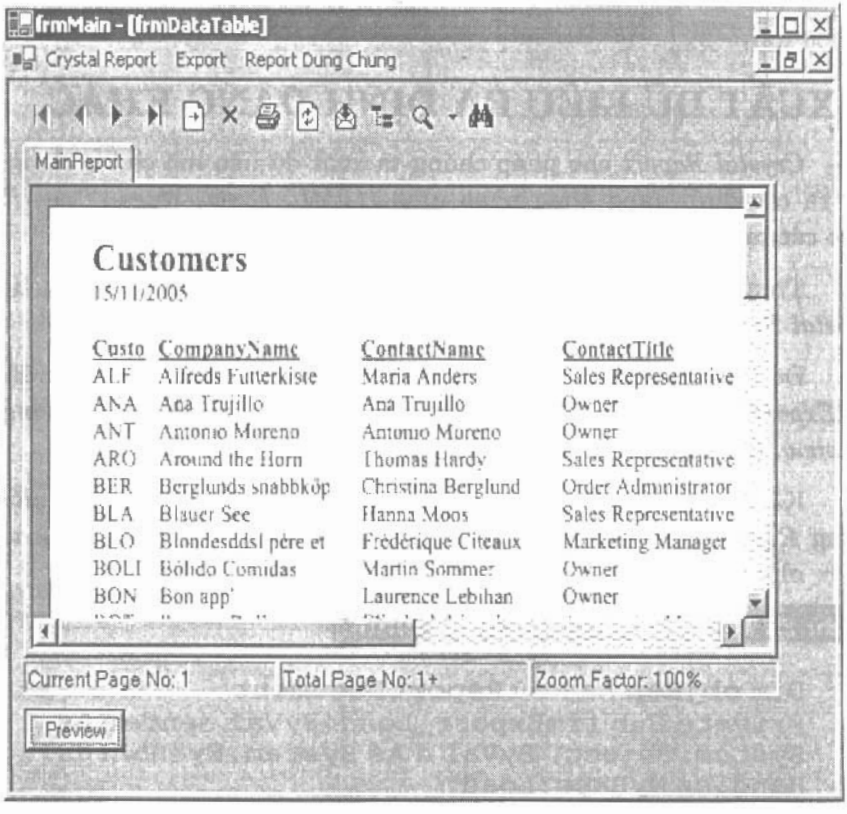
```
Function getValue(ByRef myDT As DataTable, _
  ByVal strSQL As String) As String
  Dim strError As String = ""
  Try
    Dim myData As New SqlDataAdapter(strSQL, _
    myCon)
    myDT = New DataTable
    myData.Fill(myDT)
  Catch ex As Exception
    strError = ex.Message
  End Try
  Return strError
End Function
```

Kế đến, thêm *Form* vào *Project* và đặt tên *frmDataTable*, khai báo trong biến cố *Click* của nút *Preview* như ví dụ 23-9.

Ví dụ 23-9: Trình bày dữ liệu từ đối tượng DataTable

```
Private Sub Button1_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles Button1.Click
    ' Khai báo và khởi tạo đối tượng ReportDocument
    Dim objRep As ReportDocument
    objRep = New myDataSet
    ' Khai báo và khởi tạo đối tượng DataTable
    Dim dt As New DataTable
    ' Khai báo và khởi tạo đối tượng clsDatabase
    Dim cls As New clsDatabase("server=" + _
    gsServer + ";database=" + gsDatabase + _
    ";uid=" + gsUser + ";pwd=" + gsPwd)
    ' Khai báo phát biểu SQL
    Dim strSQL As String = _
    "select * from Customers"
    ' Gọi phương thức getValue
    cls.getValue(dt)
    ' Gán đối tượng DataTable vào đối tượng ReportDocument
    objRep.SetDataSource(dt)
    RV.ReportSource = objRep
    RV.Refresh()
    objRep.Dispose()
End Sub
```

Tương tự như trường hợp đối tượng *DataSet*, khi thực thi chương trình, thay vì *Report* đọc dữ liệu trực tiếp từ cơ sở dữ liệu *SQL Server*, bạn đọc dữ liệu bằng đối tượng *ADO.NET*, sau đó điền chúng vào đối tượng *ReportDocument*, kết quả trình bày như hình 23-19.



Hình 23-19: *Trình bày dữ liệu bằng đối tượng DataTable*

Để tìm hiểu thêm cách trình bày dữ liệu bằng *Crystal Report* từ *Visual Basic.NET*, bạn có thể thực hành các ví dụ sau:

1. Thiết kế *Form* và *Report*, cho phép trình bày dữ liệu của hai đối tượng *Table* có quan hệ cha con trong cơ sở dữ liệu *Northwind*.

2. Tạo một thủ tục nội tại nhận tham số là *Country*, kế đến thiết kế *Form* và *Report*, cho phép trình bày dữ liệu của đối tượng *Store Procedure* này trong cơ sở dữ liệu *Northwind* với tham số truyền vào từ bên ngoài.

3. Thiết kế *Form* và *Report*, cho phép trình bày dữ liệu của đối tượng *Table* trong cơ sở dữ liệu *Northwind*. Đồng thời, bạn trình bày thông tin của công ty trên *Report* từ *Visual Basic.NET*.
4. Xây dựng *Form*, trình bày dữ liệu của tập tin *XML* bằng *Crystal Report*.

3. XUẤT DỮ LIỆU RA ĐỊNH DẠNG KHÁC

Crystal Report cho phép chúng ta xuất dữ liệu mà chúng đang nắm giữ ra các định dạng khác nhau như: *HTML*, *Excel*, *Work*, *Text*, *PDF*,... theo các hình thức như ra đĩa hay *mail*.

Trong trường hợp này, chúng ta tìm hiểu cách xuất dữ liệu từ *Crystal Report* ra tập tin định dạng *PDF* và *Excel*.

Để làm điều này, trước tiên bạn thêm *Form* vào *Project* và đặt tên *frmExport*. Thêm điều khiển *Crystal Report Viewer* và 3 nút tương ứng *Preview*, *Export PDF* và *Export Excel*.

Kế đến, thiết kế *Report* với tên *rptProducts* và khai báo biến đối tượng *ReportDocument* dùng chung và khởi tạo trong biến cố *Load* của *Form* như ví dụ 23-10.

Ví dụ 23-10: Khai báo biến dùng chung

```
Dim objRep As New ReportDocument
Private Sub frmExport_Load(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles MyBase.Load
    objRep = New myDataSet
End Sub
```

Khai báo đoạn chương trình trong biến cố *Click* của nút *Preview* để điền dữ liệu từ đối tượng *DataTable* vào đối tượng *ReportDocument* như ví dụ 23-11.

Ví dụ 23-11: Trình bày dữ liệu

```
Private Sub Button1_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles Button1.Click
    Try
      Dim dt As New DataTable
```

```
        Dim cls As New clsDatabase("server=" + _
        gsServer + ";database=" + gsDatabase + _
        ";uid=" + gsUser + ";pwd=" + gsPwd)
        Dim strSQL As String = _
        "select * from Customers"
        cls.getValue(dt, strSQL)
        objRep.SetDataSource(dt)
        RV.ReportSource = objRep
        RV.Refresh()
        'Cho phép sử dụng hai nút Export
        Button2.Enabled = True
        Button3.Enabled = True
    Catch ex As Exception

    End Try
End Sub
```

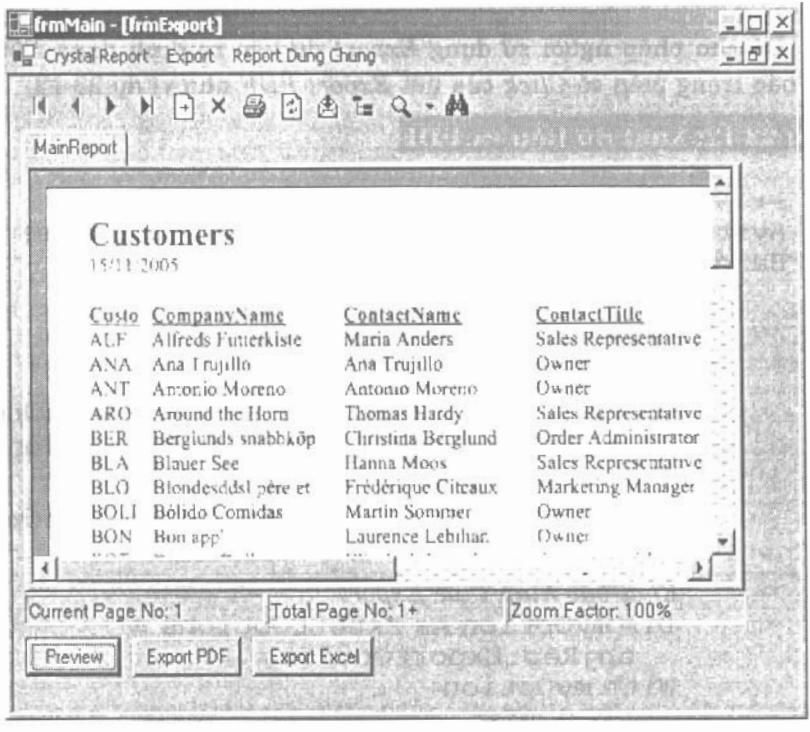
Để cho phép người sử dụng *Export* dữ liệu ra định dạng *PDF*, bạn khai báo trong biến cố *Click* của nút *Export PDF* như ví dụ 23-12.

Ví dụ 23-12: Xuất dữ liệu ra PDF

```
Private Sub Button2_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles Button2.Click
   Try
      'Khai báo tên và đường dẫn tập tin
      Dim Filename As String = "C:\test.pdf"
      'Khai báo và khởi tạo đối tượng DiskFileDestinationOptions
      Dim myDisk As DiskFileDestinationOptions= _
        New DiskFileDestinationOptions
      'Gán tên tập tin cho đối tượng DiskFileDestinationOptions
        myDisk.DiskFileName = Filename
      'Khai báo hình thức Export
        Dim myOption As ExportOptions = _
          objRep.ExportOptions
        With myOption
          'Xuất ra đĩa
          .DestinationOptions = myDisk
          'Dạng tập tin
          .ExportDestinationType = _
            ExportDestinationType.DiskFile
```

```
            'Định dạng PDF
            .ExportFormatType = _
            ExportFormatType.PortableDocFormat
          End With
          'Xuất dữ liệu
          objRep.Export()
          'Kích hoạt môi trường của ứng dụng tương ứng
          Process.Start(Filename)
        Catch ex As Exception
          MsgBox(ex.Message)
        End Try
    End Sub
```

Khi thực thi chương trình, người sử dụng nhấn nút *Preview* thì dữ liệu trình bày trên điều khiển *Crystal Report Viewer* như hình 23-20.

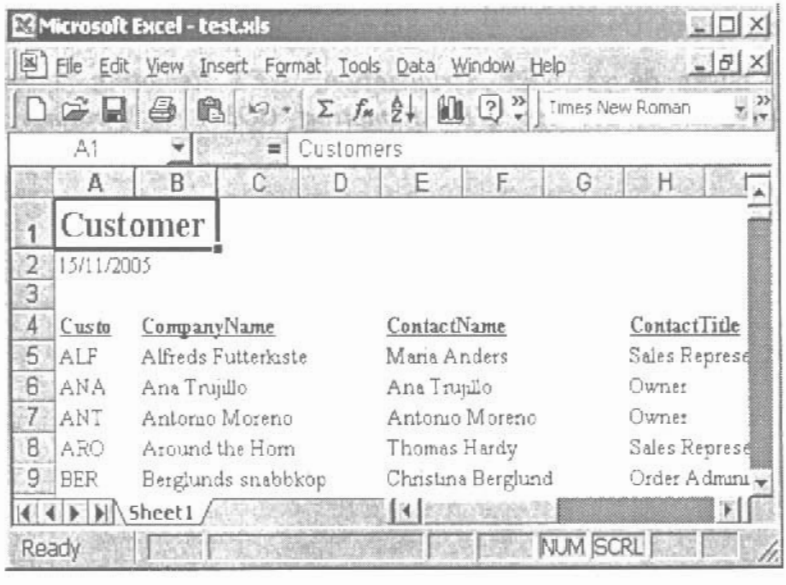


Hình 23-20: *Trình bày dữ liệu để xuất*

Trong trường hợp xuất dữ liệu ra tập tin *PDF* thì nhấn nút *Export PDF*, cửa sổ *Adobe Reader* xuất hiện như hình 23-21.

Hình 23-21: *Dữ liệu PDF*

Trong trường hợp nhấn nút *Export Excel*, lập tức cửa sổ *MS Excel* kích hoạt như hình 23-22.



Hình 23-22: *Dữ liệu Excel*

Lưu ý, khai báo *Process.Start(Filename)* cho phép bạn kích hoạt môi trường của ứng dụng tương ứng, ví dụ tập tin *Excel* thì *MS Excel* sẽ kích hoạt và *Adobe Reader* sẽ kích hoạt nếu tập tin định dạng *PDF*.

Để tìm hiểu thêm cách *Export* dữ liệu từ *Crystal Report* ra định dạng khác, bạn thực hiện các ví dụ sau:

1. Thiết kế *Form*, trình bày dữ liệu từ thủ tục nội tại và xuất ra định dạng *PDF*.
2. Thiết kế *Form*, trình bày danh sách bảng dữ liệu của cơ sở dữ liệu *Northwind* và xuất ra định dạng *Word*.
3. Thiết kế *Form*, trình bày danh sách dịch vụ của hệ điều hành và xuất ra định dạng *Excel*.

4. KẾT LUẬN

Bạn vừa tìm hiểu cách thiết kế *Report* bằng *Crystal Report* và tương tác với các *Report* này bằng nhiều hình thức khác nhau như: *OLE DB, ADO.NET*.

Trong chuyên đề kế tiếp, chúng ta tiếp tục tìm hiểu về *Windows API, Windows Registry*, Thuộc tính *(Attribute), Windows Services, EventLog, Windows Sequence Language, Windows Management Instrumentation, Multithreading, Unmanage Environment*.

Chuyên đề về *.NET Serialization, .NET Remoting, Transaction, COM+* và *MTS*, kết hợp *.NET Remoting, COM+* và *Windows Service, .NET Security*.

PHẦN II:

BÀI GIẢI CỦA BÀI TẬP

Chuyên đề 16:
LÀM VIỆC VỚI ĐỐI TƯỢNG ADO.NET

1. ĐỐI TƯỢNG ADO.NET TRONG VISUAL BASIC.NET

Không có bài tập.

2. TRÌNH ĐIỀU KHIỂN CƠ SỞ DỮ LIỆU

Không có bài tập.

3. ĐỐI TƯỢNG SQLCONNECTION

1. Thiết kế *Form*, cho phép người sử dụng nhập *UserName* và *Password*, sau đó khai báo đoạn chương trình để kết nối cơ sở dữ liệu. Nếu kết nối không thành công thì phun lỗi ra màn hình bằng lệnh *MsgBox*.

Thêm Module và khai báo các biến dùng chung:

```
Module Module1
   Public gsServer As String = "."
   Public gsDatabase As String = "northwind"
   Public gsUserID As String
   Public gsUserName As String
   Public gsFullName As String
   Public gsWelcome As String = "huukhang.com"
End Module
```

Kế đến, thêm Class vào Project và đặt tên clsDatabase, bạn khai báo các biến dùng trong Class:

```
Private psCon As String
Private strSQL As String
Private strError As String = ""
Dim myCon As SqlConnection
```

Tiếp theo, khai báo phương thức có tên Authenticate nhận hai tham trị strUserName, strPwd là hai chuỗi chứa UserName và Password

từ frmLogin, tham biến myValues là mảng kiểu chuỗi ứng với những giá trị sẽ lấy ra nếu đăng nhập thành công.

Bằng cách định nghĩa phát biểu Select để đọc bảng tblUsers để lấy giá trị cột FullName và UserID điền vào phần tử số 0 và 1 của mảng myValues.

```
Private Function Authenticate( _
  ByVal strUserName As String, _
  ByRef myValues As String()) As String
    ' Định nghĩa phát biểu SQL
    strSQL = "select UserID, FullName "
    strSQL += " from tblUsers where UserName "
    strSQL += " ='" + strUserName + "'"
    ' Phần tử thứ 0 là UserName truyền từ bên ngoài vào
    Dim myCom As SqlCommand
    Dim myRD As SqlDataReader
    Try
      myCom = New SqlCommand(strSQL, myCon)
      myRD = myCom.ExecuteReader
      ' Nếu tồn tại mẩu tin
      If myRD.Read() Then
      ' Lấy UserID và FullName
        myValues(0) = _
        Convert.ToString(myRD.GetValue(0))
        myValues(1) = _
        Convert.ToString(myRD.GetValue(1))
        ' Đăng nhập thành công
        strError = "OK"
      Else
        ' Không tồn tại UserName trong bảng tblUsers
        strError = "NOT"
      End If
      myRD.Close()
    Catch ex As Exception
      strError = ex.Message
    Finally
      If Not myRD.IsClosed Then
        myRD.Close()
      End If
    End Try
    Return strError
End Function
```

Tiếp tục khai báo phương thức OpenConnection, nhận tham trị là chuỗi UserName và tham biến là mảng chứa giá trị lấy ra như sau:

```
Public Function OpenConnection(ByVal
strUserName As String, ByVal strPwd As String,
ByRef myValues As String())As String
    'Định nghĩa chuỗi kết nối
    psCon = "server=" + gsServer
    psCon += ";database=" + gsDatabase
    psCon += ";uid=" + strUserName
    psCon += ";pwd=" + strPwd
    Try
    'Mở kết nối
      myCon = New SqlConnection(psCon)
      myCon.Open()
      'Gọi phương thức Authenticate
      strError = Authenticate( _
        strUserName, myValues)
    Catch eq As SqlException
      strError = eq.Message
    Catch ex As Exception
      strError = ex.Message
    End Try
    Return strError
End Function
```

Ngoài ra, bạn cũng khai báo phương thức đóng kết nối cơ sở dữ liệu như sau:

```
Public Sub CloseConnection()
    If myCon.State <> _
      ConnectionState.Closed Then
      myCon.Close()
      myCon.Dispose()
    End If
End Sub
```

Sau đó, bạn khai báo trong biến cố Click của nút btnLogin như sau:

```
Private Sub btnLogin_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles btnLogin.Click
```

```
    'Yêu cầu nhập UserName
      If txtUser.Text = "" Then
        MsgBox("Please enter user name", _
        MsgBoxStyle.OKOnly, gsWelcome)
        Exit Sub
      End If
      'Yêu cầu nhập Password
      If txtPwd.Text = "" Then
        MsgBox("Please enter password", _
        MsgBoxStyle.OKOnly, gsWelcome)
        Exit Sub
      End If
      'Khai báo mảng để nhận giá trị
      Dim arrString(2) As String
      'Khai báo và khởi tạo đối tượng clsDatabase
      Dim cls As New clsDatabase
      Dim strResult As String
      'Gọi phương thức OpenConnection
      strResult = cls.OpenConnection( _
        txtUser.Text, txtPwd.Text, arrString)
      Select Case strResult
        Case "OK"
          GsUserName = txtUser.Text
          MsgBox("Valid User")
          Me.Close()
        Case "NOT"
          MsgBox("Invalid User")
        Case Else
          MsgBox(strResult)
      End Select
    cls.CloseConnection()
  End Sub
```

Mỗi khi thực thi chương trình, người sử dụng nhập username và password rồi nhấn nút Login, kết quả trả về chuỗi tương ứng như trình bày ở hình 24-1.

Hình 24-1: *Đăng nhập*

Lưu ý, nếu bạn chưa có tên bảng tblUsers trong cơ sở dữ liệu Northwind thì sử dụng phát biểu SQL sau để tạo chúng.

```
create table tblUsers
(
  UserID int identity(1,1) primary key,
  UserName varchar(20) not null,
  FullName varchar(50) ,
  Email varchar(50),
  activate bit default 1,
  JoinDate smalldatetime default getdate()
)
```

2. Viết ứng dụng *Console*, sử dụng chuỗi kết nối cơ sở dữ liệu ứng với *Windows Authentication*, nếu kết nối thành công bạn liệt kê danh sách các bảng có trong cơ sở dữ liệu đó ra màn hình *Command Prompt*.

Thêm Module với tên mặc định là Module2 và khai báo các biến sử dụng chung:

```
Module Module2
  Public gsServer As String = "."
  Public gsDatabase As String = "northwind"
  Public gsUserID As String
  Public gsWelcome As String = "huukhang.com"
End Module
```

Kế đến, bạn khai báo lớp clsDatabase bao gồm các biến dùng chung trong Class.

```
Imports System.Data.SqlClient
Public Class clsDatabase
  Private psCon As String
  Private strSQL As String
  Private strError As String = ""
  Dim myCon As SqlConnection
```

Để liệt kê danh sách bảng dữ liệu trong cơ sở dữ liệu sử dụng trong chuỗi kết nối cơ sở dữ liệu bạn khai báo như sau:

```
Public Function ListOfObject(ByRef myObject As
ArrayList) As String
    ' Khai báo phát biểu liệt kê danh sách bảng dữ liệu
    strSQL = "select name from "
    strSQL += "sysobjects where type='U'"
    Dim myCom As SqlCommand
    Dim myRD As SqlDataReader
    Try
      myCom = New SqlCommand(strSQL, myCon)
      myRD = myCom.ExecuteReader
      ' Duyệt danh sách bảng dữ liệu
      While myRD.Read()
      ' Điền vào mảng
        myObject.Add(myRD.GetString(0))
      End While
      myRD.Close()
    Catch ex As Exception
      strError = ex.Message
    Finally
      If Not myRD.IsClosed Then
        myRD.Close()
      End If
    End Try
    Return strError
End Function
```

Tiếp tục khai báo phương thức mở kết nối cơ sở dữ liệu dựa vào đặt quyền hệ điều hành với cấu trúc như sau:

```
Public Function OpenConnection()
    'Định nghĩa chuỗi kết nối
    psCon = "server=" + gsServer
    psCon += ";database=" + gsDatabase
    psCon += ";Integrated Security=SSPI"
    Try
    'Mở kết nối
      myCon = New SqlConnection(psCon)
      myCon.Open()
      strError = "OK"
    Catch eq As SqlException
      strError = eq.Message
    Catch ex As Exception
      strError = ex.Message
    End Try
    Return strError
End Function
```

Tiếp theo, bạn cũng khai báo phương thức đóng kết nối cơ sở dữ liệu như sau:

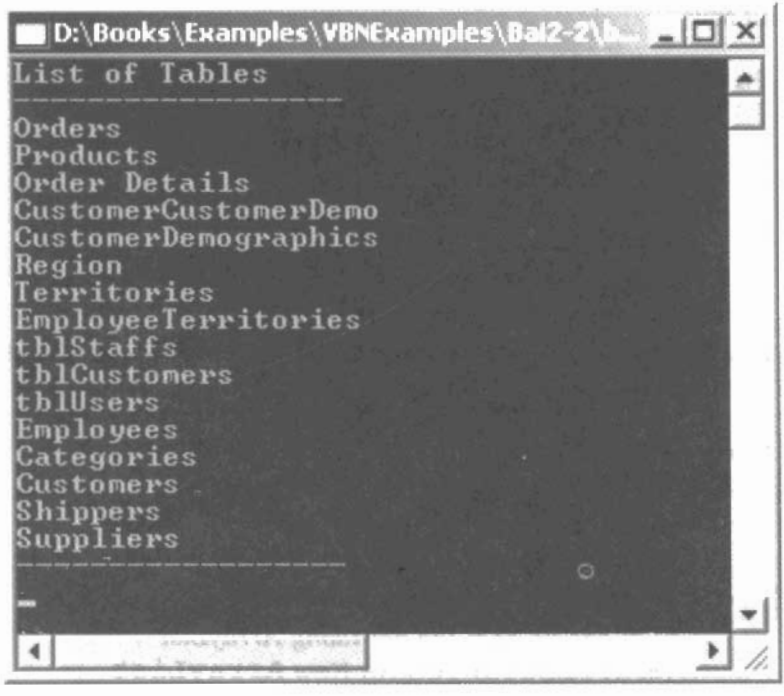
```
Public Sub CloseConnection()
    If myCon.State <> _
    ConnectionState.Closed Then
      myCon.Close()
      myCon.Dispose()
    End If
End Sub
```

Sau đó, bạn khai báo trong phương thức Main của Module1 như sau:

```
Module Module1
  Sub Main()
    Console.WriteLine("List of Tables")
    'Khai báo và khởi tạo đối tượng clsDatabase
    Dim cls As New clsDatabase
    'Khai báo và khởi tạo đối tượng ArrayList
    Dim arrLstTables As New ArrayList
    'Gọi phương thức OpenConnection
```

```
        If cls.OpenConnection() = "OK" Then
            cls.ListOfObject(arrLstTables)
            Console.WriteLine("------------------")
            'In ra danh sách Table
            For Each strTable As String _
            In arrLstTables
                Console.WriteLine(strTable)
            Next
            Console.WriteLine("------------------")
            'Gọi phương thức CloseConnection
            cls.CloseConnection()
        End If
        Console.ReadLine()
    End Sub

End Module
```

Khi thực thi chương trình, nếu kết nối thành công thì danh sách tên bảng dữ liệu của cơ sở dữ liệu Northwind sẽ xuất hiện như hình 24-2.



Hình 24-2: *Danh sách Table*

4. KHAI BÁO VÀ SỬ DỤNG OLEDBCONNECTION

1. Thiết kế *Form*, cho phép người sử dụng nhập *User name* và *Password*, sau đó khai báo đoạn chương trình để kết nối cơ sở dữ liệu và kiểm tra trong bảng *tblUsers* của cơ sở dữ liệu *Access* tùy chọn. Nếu kết nối không thành công thì phun lỗi ra màn hình bằng lệnh *MsgBox*.

Thêm Module và khai báo các biến dùng chung:

```
Module Module1
   Public gsFile As String = "C:\Northwind.mdb"
   Public gsUserID As String
   Public gsUserName As String
   Public gsPassword As String
   Public gsFullName As String
   Public gsWelcome As String = "huukhang.com"
End Module
```

Kế đến, thêm Class vào Project và đặt tên clsDatabase, bạn khai báo các biến dùng trong Class ứng với cơ sở dữ liệu dạng OleDb:

```
Imports System.Data.OleDb
Public Class clsDatabase
   Private psCon As String
   Private strSQL As String
   Private strError As String = ""
   Dim myCon As OleDbConnection
```

Tiếp theo, khai báo phương thức có tên Authenticate nhận hai tham trị strUserName, strPwd là hai chuỗi chứa UserName và Password từ frmLogin, tham biến myValues là mảng kiểu chuỗi ứng với những giá trị sẽ lấy ra nếu đăng nhập thành công.

Bằng cách định nghĩa phát biểu Select để đọc bảng tblUsers lấy giá trị cột FullName và UserID điền vào phần tử số 0 và 1 của mảng myValues.

```
Private Function Authenticate( _
ByVal strUserName As String, _
   ByVal strPwd As String _
   ByRef myValues As String()) As String
      ' Định nghĩa phát biểu SQL
      strSQL = "select Password,UserID,FullName"
```

```
        strSQL += " from tblUsers where UserName "
        strSQL += " ='" + strUserName + "'"
        'Phần tử thứ 0 là UserName truyền từ bên ngoài vào
        Dim myCom As OleDbCommand
        Dim myRD As OleDbDataReader
        Try
          myCom = New OleDbCommand(strSQL, myCon)
          myRD = myCom.ExecuteReader
          'Nếu tồn tại mẩu tin
          If myRD.Read() Then
          'Nếu đúng mật khẩu
          If myRD.GetString(0) = strPwd Then
          'Lấy UserID và FullName
            myValues(0) = _
            Convert.ToString(myRD.GetValue(0))
            myValues(1) = _
            Convert.ToString(myRD.GetValue(1))
            'Đăng nhập thành công
            strError = "OK"
           Else
              'Nếu sai mật khẩu
              strError = "Wrong Password"
            End If
          Else
            'Không tồn tại UserName trong bảng tblUsers
            strError = "Invalid UserName"
          End If
          myRD.Close()
        Catch ex As Exception
          strError = ex.Message
        Finally
          If Not myRD.IsClosed Then
            myRD.Close()
          End If
        End Try
        Return strError
    End Function
```

Tiếp tục khai báo phương thức OpenConnection, nhận hai tham trị là chuỗi UserName, Password và tham biến là mảng chứa giá trị lấy ra như sau:

```
Public Function OpenConnection(ByVal
strUserName As String, ByVal strPwd As String,
ByRef myValues As String())As String
    'Định nghĩa chuỗi kết nối cơ sở dữ liệu Access
    psCon =
"Provider=Microsoft.Jet.OLEDB.4.0;"
    psCon += "Data Source=" + gsFile
    Try
    'Mở kết nối
      myCon = New OleDbConnection(psCon)
      myCon.Open()
      'Gọi phương thức Authenticate
      strError = Authenticate( _
        strUserName, strPwd, myValues)
    Catch eq As OleDbException
      strError = eq.Message
    Catch ex As Exception
      strError = ex.Message
    End Try
    Return strError
End Function
```

Ngoài ra, bạn cũng khai báo phương thức đóng kết nối cơ sở dữ liệu như sau:

```
Public Sub CloseConnection()
    If myCon.State <> _
      ConnectionState.Closed Then
      myCon.Close()
      myCon.Dispose()
    End If
End Sub
```

Sau đó, bạn khai báo trong biến cố Click của nút btnLogin như sau:

```
Private Sub btnLogin_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles btnLogin.Click
  'Yêu cầu nhập UserName
    If txtUser.Text = "" Then
      MsgBox("Please enter user name", _
      MsgBoxStyle.OKOnly, gsWelcome)
      Exit Sub
    End If
```

```
    'Yêu cầu nhập Password
    If txtPwd.Text = "" Then
      MsgBox("Please enter password", _
      MsgBoxStyle.OKOnly, gsWelcome)
      Exit Sub
    End If
    'Khai báo mảng để nhận giá trị
    Dim arrString(2) As String
    'Khai báo và khởi tạo đối tượng clsDatabase
    Dim cls As New clsDatabase
    Dim strResult As String
    'Gọi phương thức OpenConnection
    strResult = cls.OpenConnection( _
      txtUser.Text, txtPwd.Text, arrString)
    Select Case strResult
      Case "OK"
        MsgBox("Valid User")
        Me.Close()
      Case "NOT"
        MsgBox("Invalid User")
      Case Else
        MsgBox(strResult)
    End Select
  cls.CloseConnection()
End Sub
```

Mỗi khi thực thi chương trình, người sử dụng nhập username và password rồi nhấn nút Login, kết quả trả về chuỗi tương ứng trình bày ở trên như hình 24-3.

Hình 24-3: *Đăng nhập cơ sở dữ liệu Access*

Lưu ý, nếu bạn chưa có tên bảng tblUsers trong cơ sở dữ liệu Access có tên Northwind thì sử dụng phát biểu SQL sau để tạo chúng.

```
CREATE TABLE tblUsers
(
UserID Autonumber primary key,
UserName text(20) not null,
FullName text(50) ,
Email text(50),
activate true/false default true,
JoinDate datetime default date()
)
```

2. Viết ứng dụng *Console*, sử dụng chuỗi kết nối liệu *Excel*, nếu kết nối thành công, bạn liệt kê danh sách dữ liệu trong bảng tính (*Sheet*) ra màn hình *Command Prompt*.

Để thực hiện ví dụ này, trước tiên bạn khai báo trong Module các biến tên và đường dẫn tập tin Excel cho Project.

```
Module Module2
   Public gsFile As String = "Book1.xls"
   Public gsWelcome As String = "huukhang.com"
End Module
```

Kế đến, tạo Class có tên clsExcel và khai báo cấu trúc như sau:

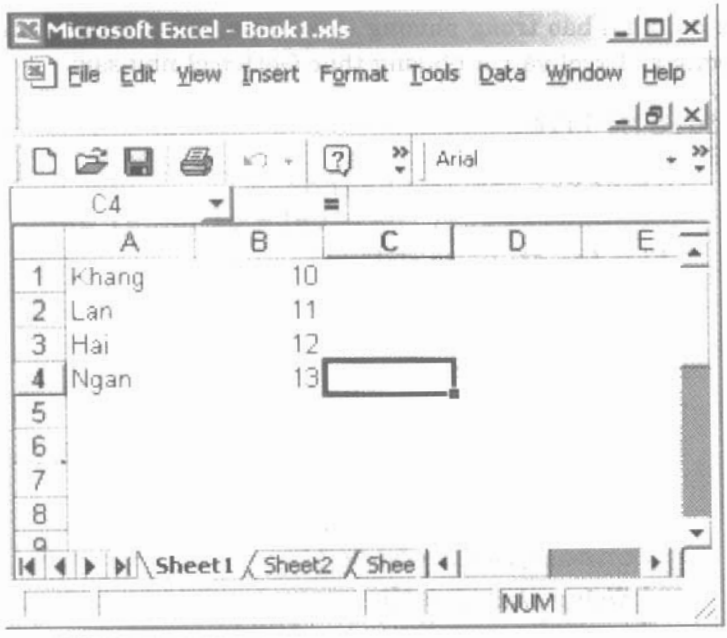
```
' Khai báo sử dụng không gian tên
Imports System.Data.OleDb
' Khai báo tên Class
Public Class clsExcel
   ' Khai báo biến dùng chung
   Private psCon As String
   Private strSQL As String
   Private strError As String = ""
   Dim myCon As OleDbConnection
   ' Khai báo phương thức kết nối tập tin Excel
   Public Function OpenConnection() As String

psCon="Provider=Microsoft.Jet.OLEDB.4.0;"
      psCon += "Data Source=" + gsFile
      psCon += ";Extended Properties="
```

```
        psCon += """Excel 8.0;HDR=YES;"""
        Try
          myCon = New OleDbConnection(psCon)
          myCon.Open()
          strError = "OK"
        Catch eq As OleDbException
          strError = eq.Message
        Catch ex As Exception
          strError = ex.Message
        End Try
        Return strError
      End Function
      ' Khai báo phương thức đóng kết nối
      Public Sub CloseConnection()
        If myCon.State <> ConnectionState.Closed _
          Then
          myCon.Close()
          myCon.Dispose()
        End If
      End Sub

      ' Khai báo phương thức đọc bảng tính điền vào đối tượng
DataTable
      Public Function GetExcel( _
        ByRef myTable As DataTable) As String
        Try
          Dim strSQL As String
          strSQL = "Select * from [Sheet1$]"
          Dim myData As New OleDbDataAdapter( _
          strSQL, myCon)
          myData.Fill(myTable)
        ' Điền dữ liệu thành công
          strError = "OK"
        Catch ex As Exception
          strError = ex.Message
        Finally

        End Try
        Return strError
      End Function
End Class
```

Sau đó, khai báo trong phương thức Main của chương trình để khởi tạo đối tượng clsExcel và gọi phương thức GetExcel như sau:

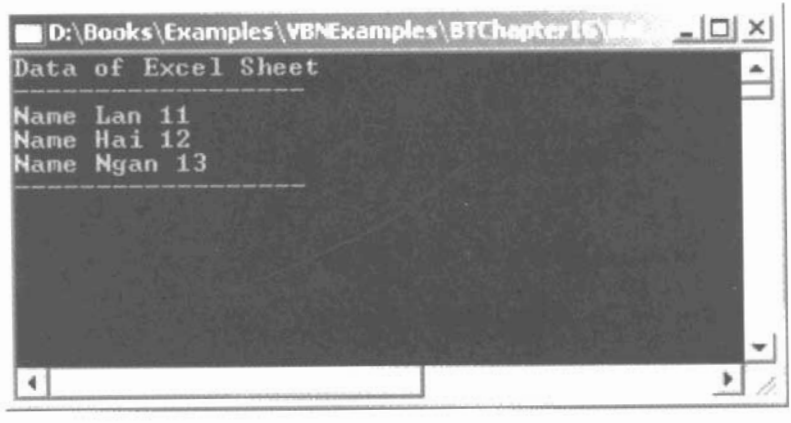
```
Module Module1
  Sub Main()
    Dim strError As String
    Console.WriteLine("Data of Excel Sheet")
    ' Khai báo đối tượng clsExcel
    Dim cls As New clsExcel
    ' Khai báo đối tượng DataTable
    Dim dtExcel As New DataTable
    ' Nếu kết nối tập tin Excel thành công
    If cls.OpenConnection() = "OK" Then
      ' Gọi phương thức GetExcel
      strError = cls.GetExcel(dtExcel)
      Console.WriteLine("----------------")
      ' Không phát sinh lỗi
      If strError.Equals("OK") Then
        ' Duyệt trên từng hàng dữ liệu
        For Each drExcel As DataRow In _
          dtExcel.Rows
          ' In tên và giá trị
          Console.WriteLine("Name {0} {1}", _
          drExcel.Item(0), drExcel.Item(1))
        Next
        Console.WriteLine("----------------")
        cls.CloseConnection()
      End If
    Else
      Console.WriteLine("Error: {0}", strError)
    End If
    Console.ReadLine()
  End Sub
End Module
```

Giả sử, chúng ta có tập tin Excel với tên là Book1.xls nằm trong thư mục *Bin* của ứng dụng với nội dung của Sheet1 như hình 24-4.



Hình 24-4: *Cấu trúc Sheet của Excel*

Bằng cách thực thi chương trình, lập tức dữ liệu trong Sheet1 sẽ trình bày trên màn hình Command Prompt như hình 24-4-1.



Hình 24-4-1: *Dữ liệu trong tập tin Excel*

3. Tương tự như trên, bạn khai báo một *Class* và đặt tên *clsExcel*, bằng cách khai báo và khởi tạo đối tượng *OleDbConnection* trong *Constructor* của *Class* với chuỗi kết nối xem như tham số truyền vào *Constructor*.

Sau đó, khai báo ứng dụng để sử dụng *clsExcel* này bằng cách truyền hoặc gán chuỗi kết nối vào thuộc tính của đối tượng *clsExcel*.

Tương tự như ví dụ vừa trình bày ở trên, trước tiên bạn khai báo biến nắm giữ tên và đường dẫn tập tin Excel trong Module.

```
Module Module2
   Public gsFile As String = "Book1.xls"
   Public gsWelcome As String = "huukhang.com"
End Module
```

Kế đến, thêm lớp vào Project và đặt tên clsExcel rồi khai báo sử dụng không gian tên.

```
Imports System.Data.OleDb
Public Class clsExcel
    ...
```

Khai báo Constructor để nhận tham trị là chuỗi kết nối tập tin Excel và mở kết nối với tập tin này như sau:

```
   Private strSQL As String
   ' Khai báo biến lỗi dùng chung
   Public strError As String = ""
   ' Khai báo biến dữ liệu OleDbConnection
   Dim myCon As OleDbConnection
   Sub New(ByVal psCon As String)
      Try
         myCon = New OleDbConnection(psCon)
         myCon.Open()
         ' Nếu kết nối thành công
         strError = "OK"
      Catch eq As OleDbException
         strError = eq.Message
      Catch ex As Exception
         strError = ex.Message
      End Try
End Sub
```

Kế đến, khai báo phương thức đóng kết nối tập tin Excel đang mở với cấu trúc:

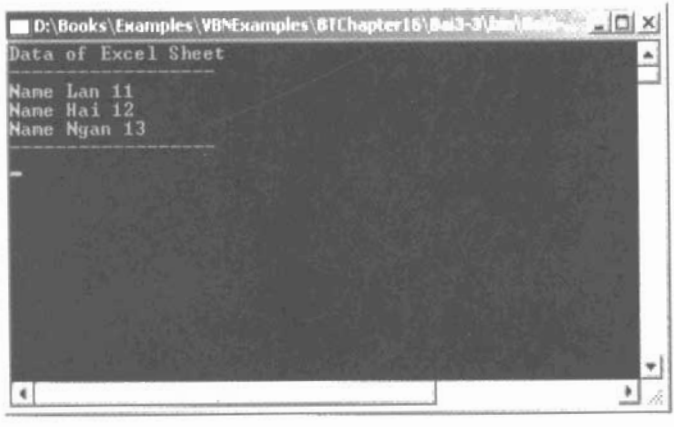
```
Public Sub CloseConnection()
    If myCon.State <> ConnectionState.Closed _
```

```
            Then
            myCon.Close()
            myCon.Dispose()
        End If
    End Sub
```

Sau đó, khai báo phương thức điền dữ liệu có trong bảng tính Sheet1 của tập tin Excel vào đối tượng DataTable như sau:

```
    Public Sub GetExcel(ByRef myTable As DataTable)
        Try
            Dim strSQL As String
            'Khai báo phát biểu SQL
            strSQL = "Select * from [Sheet1$]"
            Dim myData As New OleDbDataAdapter( _
            strSQL, myCon)
            'Điền dữ liệu vào đối tượng DataTable
            myData.Fill(myTable)
            strError = "OK"
        Catch ex As Exception
            strError = ex.Message
        Finally

        End Try
    End Sub
```

Cuối cùng, bạn khai báo để khởi tạo đối tượng clsExcel và gọi phương thức GetExcel để trình bày dữ liệu ra màn hình Command Prompt như hình 24-5.



Hình 24-5: *Trình bày dữ liệu Excel*

5. ĐỐI TƯỢNG SQLCOMMAND

5.1. Khai báo đối tượng SqlCommand

5.1.1. Phương thức ExecuteNonQuery

1. Tạo ứng dụng *Windows Forms*, cho phép người sử dụng nhập tên cơ sở dữ liệu và tạo cơ sở dữ liệu đó trong *SQL Server*.

Để thực hiện ví dụ này, trước tiên bạn thêm Class vào Project và đặt tên clsDatabase. Kế đến, khai báo các biến sử dụng chung:

```
Imports System.Data.SqlClient
Public Class clsDatabase
  Private psCon As String
  Private strSQL As String
  Private strError As String = ""
  Dim myCon As SqlConnection
```

Định nghĩa phương thức mở kết nối cơ sở dữ liệu SQL Server bằng đặt quyền hệ điều hành hoặc SQL Server tương tự như sau:

```
Public Function OpenConnection() As String
    ' Khai báo chuỗi kết nối cơ sở dữ liệu
    psCon = "server=" + gsServer
    psCon += ";database=" + gsDatabase
    psCon += ";Integrated Security=SSPI"
    Try
      ' Mở kết nối cơ sở dữ liệu
      myCon = New SqlConnection(psCon)
      myCon.Open()
      strError = "OK"
    Catch eq As SqlException
      strError = eq.Message
    Catch ex As Exception
      strError = ex.Message
    End Try
    Return strError
End Function
```

Tương như những phần trước, bạn cũng khai báo phương thức đóng kết nối cơ sở dữ liệu:

```
Public Sub CloseConnection()
    If myCon.State <> _
```

```
            ConnectionState.Closed Then
          myCon.Close()
          myCon.Dispose()
        End If
    End Sub
```

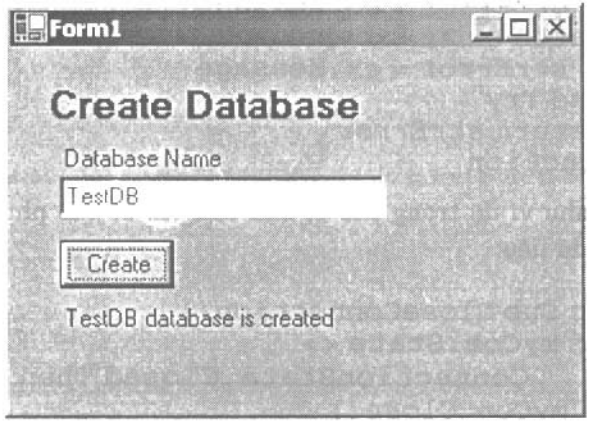
Tiếp tục, khai báo phương thức ExecuteSQL nhận tên cơ sở dữ liệu, định nghĩa phát biểu CREATE DATABASE và sử dụng phương thức ExecuteNonQuery để thực thi phát biểu này như sau:

```
Public Function ExecuteSQL(ByVal DatabaseName
As String) As String
    ' Khai báo chuỗi SQL để kiểm tra cơ sở dữ liệu tồn tại?
    strSQL = "select name from "
    strSQL += "master.dbo.sysdatabases"
    strSQL += " where name='" + DatabaseName + "'"
    Dim myCom As SqlCommand
    Try
      ' Khai báo và khởi tạo đối tượng SqlCommand
      myCom = New SqlCommand(strSQL, myCon)
      ' Nếu cơ sở dữ liệu chưa tồn tại
      If myCom.ExecuteScalar Is Nothing Then
      ' Khai báo phát biểu SQL
        strSQL = "create database "
        strSQL += DatabaseName
        myCom = New SqlCommand(strSQL, myCon)
        ' Thực thi phát biểu SQL để tạo cơ sở dữ liệu
        myCom.ExecuteNonQuery()
        strError = DatabaseName
        strError += " database is created"
      Else
        ' Cơ sở dữ liệu đã tồn tại
        strError = DatabaseName
        strError += " is available in system"
      End If
    Catch ex As Exception
      strError = ex.Message
    Finally
      myCom.Dispose()
    End Try
    Return strError
End Function
```

Để tạo Database, bạn khai báo gọi phương thức ExecuteSQL của lớp clsDatabase trong biến cố Click của nút Create như sau:

```
Private Sub Button1_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles Button1.Click
    If txtName.Text <> "" Then
        ' Khai báo và khởi tạo đối tượng clsDatabase
        Dim cls As New clsDatabase
        ' Gọi phương thức mở kết nối cơ sở dữ liệu
        lbResult.Text = cls.OpenConnection()
        ' Gọi phương thức tạo cơ sở dữ liệu
        lbResult.Text = _
          cls.ExecuteSQL(txtName.Text)
        cls.CloseConnection()
    End If
End Sub
```

Khi thực thi chương trình, giao diện Form trông giống như hình 24-6, nếu người sử dụng nhập tên cơ sở dữ liệu và nhấn nút Create, lập tức cơ sở dữ liệu đó sẽ tạo ra trong SQL Server.



Hình 24-6: *Tạo cơ sở dữ liệu*

2. Thiết kế *Form*, cho phép người sử dụng chọn cơ sở dữ liệu đang tồi tại trong *SQL Server* rồi tạo mới một *Table* và thêm dữ liệu vào *Table* đó.

Để thực hiện ví du này, trước tiên bạn thêm Class vào Project và đặt tên clsDatabase. Kế đến, khai báo các biến sử dụng chung:

```
Imports System.Data.SqlClient
Public Class clsDatabase
  Private psCon As String
  Private strSQL As String
  Private strError As String = ""
  Dim myCon As SqlConnection
```

Định nghĩa phương thức mở kết nối cơ sở dữ liệu SQL Server bằng đặt quyền hệ điều hành hoặc SQL Server tương tự như sau:

```
Public Function OpenConnection() As String
    ' Khai báo chuỗi kết nối cơ sở dữ liệu
    psCon = "server=" + gsServer
    psCon += ";database=" + gsDatabase
    psCon += ";Integrated Security=SSPI"
    Try
      ' Mở kết nối cơ sở dữ liệu
      myCon = New SqlConnection(psCon)
      myCon.Open()
      strError = "OK"
    Catch eq As SqlException
      strError = eq.Message
    Catch ex As Exception
      strError = ex.Message
    End Try
    Return strError
End Function
```

Tương như ví dụ trong câu 1, bạn cũng khai báo phương thức đóng kết nối cơ sở dữ liệu:

```
Public Sub CloseConnection()
    If myCon.State <> _
        ConnectionState.Closed Then
      myCon.Close()
      myCon.Dispose()
    End If
End Sub
```

Tiếp tục, khai báo phương thức ExecuteSQL nhận tên cơ sở dữ liệu, phát biểu CREATE TABLE và sử dụng phương thức ExecuteNonQuery để thực thi phát biểu này như sau:

```
Public Function ExecuteSQL(ByVal DatabaseName
As String, ByVal TableName As String) As String
    ' Khai báo chuỗi SQL để kiểm tra bảng dữ liệu đã tồn tại
    ' trong cơ sở dữ liệu?
    strSQL = "select name from " + DatabaseName
    strSQL += ".dbo.sysobjects"
    strSQL += " where name='" + TableName + "'"
    Dim myCom As SqlCommand
    Try
      ' Khai báo và khởi tạo đối tượng SqlCommand
      myCom = New SqlCommand(strSQL, myCon)
      ' Nếu bảng chưa tồn tại trong cơ sở dữ liệu
      If myCom.ExecuteScalar Is Nothing Then
        myCom = New SqlCommand(TableName, myCon)
        ' Thực thi phát biểu SQL để tạo bảng dữ liệu
        myCom.ExecuteNonQuery()
        strError = " Table is created"
      Else
        ' Cơ sở dữ liệu đã tồn tại
        strError = "Table is available in "
        strError += DatabaseName
      End If
    Catch ex As Exception
      strError = ex.Message
    Finally
      myCom.Dispose()
    End Try
    Return strError
End Function
```

Để liệt kê danh sách tên cơ sở dữ liệu cho phép người sử dụng chọn, bạn khai báo gọi phương thức GetDatabase trong lớp clsDatabase như sau:

```
Public Function GetDatabases(ByRef dtDatabases
As DataTable) As String
    ' Khai báo phát biểu Select để liệt kê danh sách cơ sở dữ liệu
    strSQL = "select name from "
    strSQL += "master.dbo.sysdatabases"
    Dim myData As SqlDataAdapter
    Try
```

```
            myData = New SqlDataAdapter(strSQL, myCon)
            myData.Fill(dtDatabases)
            strError = "OK"
        Catch ex As Exception
            strError = ex.Message
        Finally
            myData.Dispose()
        End Try
        Return strError
    End Function
```

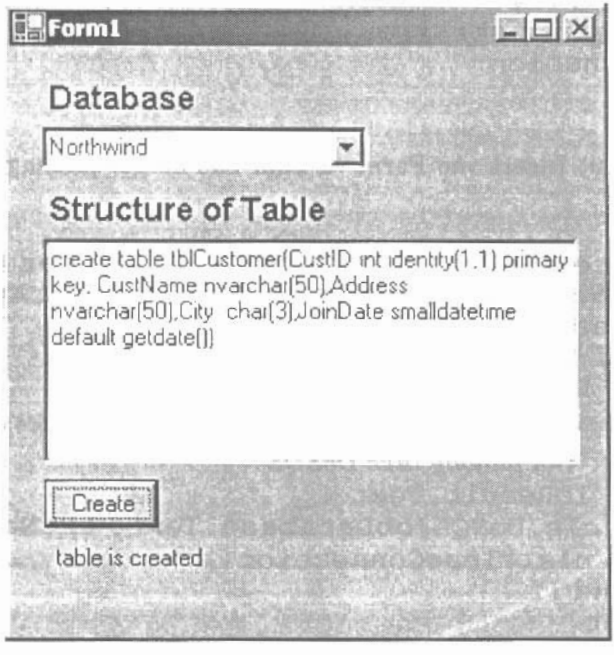
Sau đó, khai báo trong biến cố Load của Form để điền danh sách cơ sở dữ liệu vào điều khiển ComboBox như sau:

```
Private Sub Form1_Load(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles MyBase.Load
    'Khai báo và khởi tạo đối tượng DataTable
    Dim dtDB As New DataTable
    'Khai báo và khởi tạo đối tượng clsDatabase
    Dim cls As New clsDatabase
    'Nếu mở kết nối cơ sở dữ liệu thành công
    If cls.OpenConnection = "OK" Then
        'Gọi phương thức GetDatabase
        cls.GetDatabases(dtDB)
        'Tồn tại cơ sở dữ liệu
        If dtDB.Rows.Count > 0 Then
            cbDatabase.DataSource = dtDB
            cbDatabase.DisplayMember = "name"
            cbDatabase.ValueMember = "name"
        End If
        'Đóng kết nối
    cls.CloseConnection()
    End If
End Sub
```

Để cho phép người sử dụng tạo bảng dữ liệu từ cấu trúc nhập trên điều khiển txtName trong cơ sở dữ liệu đang chọn của điều khiển ComboBox, bạn khai báo trong biến cố Click của nút Create như sau:

```
Private Sub Button1_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles Button1.Click
    If txtName.Text <> "" Then
        ' Khai báo và khởi tạo đối tượng clsDatabase
        Dim cls As New clsDatabase
        ' Gọi phương thức mở kết nối cơ sở dữ liệu
        lbResult.Text = cls.OpenConnection()
        ' Gọi phương thức tạo bảng dữ liệu trong cơ sở dữ liệu
        lbResult.Text = _
          cls.ExecuteSQL(cbDatabase.Text, _
          txtName.Text)
        cls.CloseConnection()
    End If
End Sub
```

Khi thực thi chương trình, nếu người sử dụng chọn tên cơ sở dữ liệu trên ComboBox và nhập phát biểu Create Table như hình 26-7 rồi nhấn nút Create, lập tức bảng dữ liệu sẽ tạo ra trong cơ sở dữ liệu đang chọn.



Hình 24-7: *Tạo bảng dữ liệu*

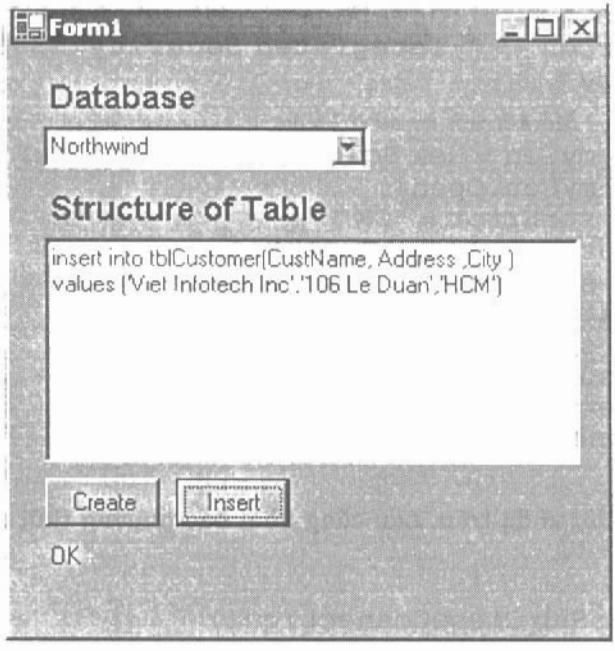
Sau khi tạo bảng dữ liệu thành công, bạn có thể thêm mẩu tin vào bảng này bằng cách định nghĩa phát biểu Insert như hình 24-7-1.

Ngoài ra, bạn phải khai báo phương thức trong lớp clsDatabase cho phép nhận phát biểu SQL dạng Insert như sau:

```
Public Function DoSQL(ByVal DatabaseName As
String, ByVal strSQL As String) As String
    ' Định nghĩa lại phát biểu SQL
    strSQL = "use " + DatabaseName + ";" + strSQL
    Dim myCom As SqlCommand
    Try
      myCom = New SqlCommand(strSQL, myCon)
      ' Thực thi phát biểu SQL
      myCom.ExecuteNonQuery()
      strError = "OK"
    Catch ex As Exception
      strError = ex.Message
    Finally
      myCom.Dispose()
    End Try
    Return strError
End Function
```

Thêm nút Insert vào Form và khai báo để gọi phương thức DoSQL như sau:

```
Private Sub Button2_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles Button2.Click
    If txtName.Text <> "" Then
      Dim cls As New clsDatabase
      lbResult.Text = cls.OpenConnection()
      ' Gọi phương thức DoSQL
      lbResult.Text = _
      cls.DoSQL(cbDatabase.Text, txtName.Text)
      cls.CloseConnection()
    End If
End Sub
```



Hình 24-7-1: *Thêm mẩu tin*

3. Bằng cách sử dụng điều khiển *DataGrid*, liệt kê danh sách các mẩu tin trong bảng vừa tạo trong câu 2, khi người sử dụng chọn những mẩu tin cần xóa trên các *checkbox*, lập tức các mẩu tin đó sẽ bị xóa và in ra số mẩu tin đã xóa.

Để thực hiện ví dụ này, trước tiên bạn thêm Class vào Project và khai báo các biến dùng chung như sau:

```
Imports System.Data.SqlClient
Public Class clsDatabase
   Private psCon As String
   Private strSQL As String
   Private strError As String = ""
   Dim myCon As SqlConnection
```

Kế đến, bạn khai báo phương thức mở kết nối cơ sở dữ liệu SQL Server bằng đặt quyền hệ điều hành hoặc SQL Server tương tự như sau:

```
Public Function OpenConnection() As String
    ' Khai báo chuỗi kết nối cơ sở dữ liệu
    psCon = "server=" + gsServer
```

```
        psCon += ";database=" + gsDatabase
        psCon += ";Integrated Security=SSPI"
        Try
            ' Mở kết nối cơ sở dữ liệu
            myCon = New SqlConnection(psCon)
            myCon.Open()
            strError = "OK"
        Catch eq As SqlException
            strError = eq.Message
        Catch ex As Exception
            strError = ex.Message
        End Try
        Return strError
    End Function
```

Tương như ví dụ trên, bạn cũng khai báo phương thức đóng kết nối cơ sở dữ liệu:

```
    Public Sub CloseConnection()
        If myCon.State <> _
            ConnectionState.Closed Then
            myCon.Close()
            myCon.Dispose()
        End If
    End Sub
```

Tiếp tục, khai báo phương thức ExecuteSQL nhận mảng giá trị chọn từ điều khiển DataGrid và sử dụng phương thức ExecuteNonQuery để thực thi thủ tục nội tại trong SQL Server này như sau:

```
    Public Function ExecuteSQL(ByVal strValue As
    String) As String
        Dim myCom As SqlCommand
        StrSQL="spDelCustomer "
        Try
            ' Khai báo và khởi tạo đối tượng SqlCommand
            myCom = New SqlCommand(strSQL, myCon)
            ' Khai báo tham số
            myCom.Parameters.Add("@myValue", _
            strValue)
            ' Thực thi thủ tục
```

```
            myCom.ExecuteNonQuery()
            strError = "OK"
        Catch ex As Exception
            strError = ex.Message
        Finally
            myCom.Dispose()
        End Try
        Return strError
    End Function
```

Trong đó, thủ tục spCustomer dùng để xóa những mẩu tin trong bảng tblCustomer có giá trị của cột CustID xuất hiện trong mảng giá trị truyền vào.

```
CREATE PROC spDelCustomer
  @myValue varchar(100)
AS
  declare @SQL nvarchar(500)
  declare @ParmDefinition nVARCHAR(100)
  set @SQL=N'Delete from tblCustomer where
    cast(CustID as varchar(10)) in(@myValues)'
  set @ParmDefinition=N'@myValues
varchar(100)'  EXEC sp_executeSQL @SQL,
    @ParmDefinition,@myValues=@myValue
```

Lưu ý, thủ tục hệ thống có tên *sp_executeSQL* cho phép bạn thực thi một phát biểu SQL dạng chuỗi khai báo trong SQL Server:

```
sp_executesql [@stmt =] stmt
[
  { , [@params =] N'@parameter_name data_type
[,...n]' }
  { , [@param1 =] 'value1' [,...n] }
]
```

Do trong bảng dữ liệu tblCustomer, cột CustID có kiểu số nguyên, mảng giá trị chọn truyền vào là dạng chuỗi, chính vì vậy trong phát biểu Delete bạn khai báo chuyển kiểu Int sang Varchar:

```
'Delete from tblCustomer where
    cast(CustID as varchar(10)) in(@myValues)
```

Chính vì vậy, chúng ta khai báo lại thủ tục spDelCustomer với cấu trúc như sau:

```
CREATE PROC spDelCustomer
  @myValue varchar(100)
As
  declare @SQL nvarchar(500)
  -- Dùng phương thức char với tham số là 39 để định nghĩa dấu
nháy
  set @myValue=char(39)+
  replace(@myValue,',',char(39)+','+char(39))+
  char(39)
  set @SQL=N'Delete from tblCustomer where
    cast(CustID as varchar(10)) in (' +
      @myValue+')'
  exec sp_executeSQL @SQL
```

Ngoài ra, thủ tục *sp_executeSQL* yêu cầu bạn định nghĩa kiểu dữ liệu cho tham số trong phát biểu Delete là @myValues như sau:

```
set @ParmDefinition=N'@myValues varchar(100)'
```

Bằng cách khai báo phương thức nhận tham biến là đối tượng DataTable, bạn có thể đọc dữ liệu trong bảng tblCustomer và điền vào đối tượng DataTable như sau:

```
Public Function GetValue(ByRef dtDatabases As
DataTable, ByVal strTable As String) As String
    ' Định nghĩa phát biểu SQL với cột Available
    strSQL = "select 0 As Available,* from "
    strSQL += strTable
    Dim myData As SqlDataAdapter
    Try
      myData = New SqlDataAdapter(strSQL, myCon)
      myData.Fill(dtDatabases)
      strError = "OK"
    Catch ex As Exception
      strError = ex.Message
    Finally
      myData.Dispose()
    End Try
    Return strError
End Function
```

Tiếp tục, khai báo để gọi phương thức GetValue điền danh sách mẩu tin trong bảng tblCustomer vào điều khiển DataGrid như sau:

```
Private Sub btnLoad_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles btnLoad.Click
    'Khởi tạo đối tượng DataTable
    dtView = New DataTable
    'Khởi tạo đối tượng clsDatabase
    cls = New clsDatabase
    If cls.OpenConnection = "OK" Then
        'Gọi phương thức GetValue
        cls.GetValue(dtView, "tblCustomer")
        'Nếu tồn tại mẩu tin
        If dtView.Rows.Count > 0 Then
            'Điền dữ liệu
            Me.DataGrid1.DataSource = dtView
            'Định nghĩa Style
            ApplyColumnStyle()
            'Cho phép sử dụng nút Delete
            btnDelete.Enabled = True
        End If
    End If
End Sub
```

Lưu ý, bạn khai báo hai biến sử dụng chung cho lớp Form1 là dtView và clsDatabase.

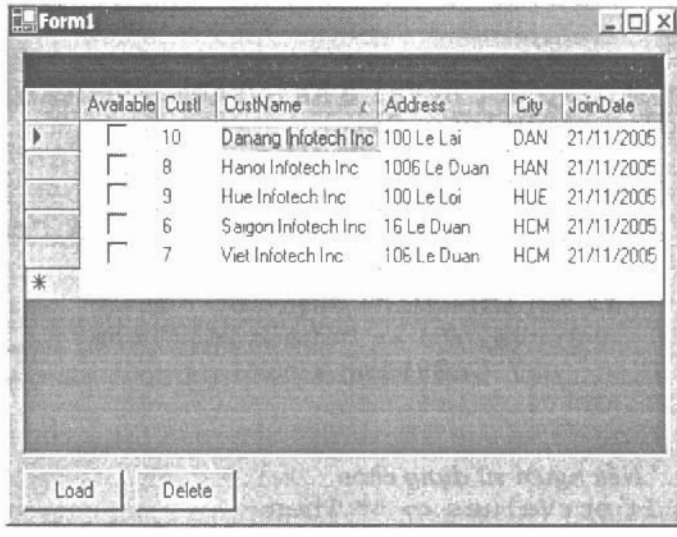
```
Dim dtView As DataTable
Dim cls As clsDatabase
```

Để xuất hiện cột dạng CheckBox, bạn cần khai báo phương thức ApplyColumnStyle có cấu trúc như sau:

```
Sub ApplyColumnStyle()
    Dim myColStyle As New DataGridTableStyle
    'Duyệt trên tất cả các cột
    For Each dc As DataColumn In dtView.Columns
        'Trường hợp cột khác
        If dc.ColumnName <> "Available" Then
```

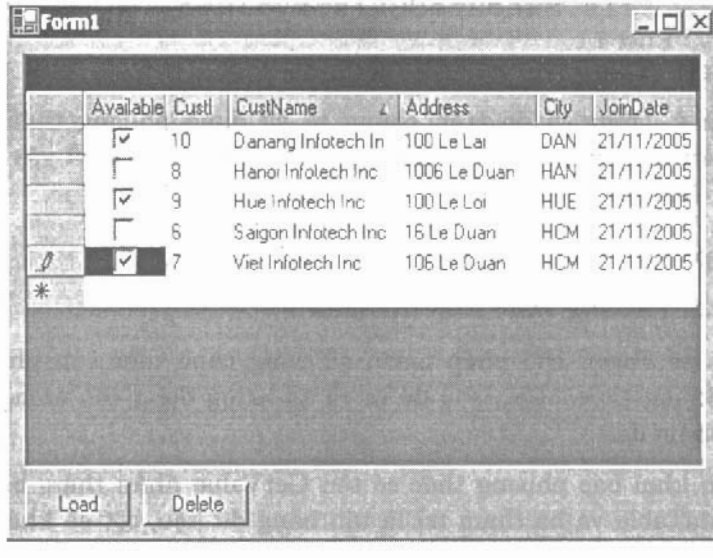
```
            Dim myTxt As New DataGridTextBoxColumn
            With myTxt
               .MappingName = dc.ColumnName
               .HeaderText = dc.Caption
               Select Case dc.ColumnName
                  Case "CustID"
                     .ReadOnly = True
                     .Width = 50
                  Case "CustName"
                     .ReadOnly = True
                     .Width = 100
                  Case "Address"
                     .ReadOnly = True
                     .Width = 100
                  Case "City"
                     .ReadOnly = True
                     .Width = 50
               End Select

            End With
            myColStyle.GridColumnStyles.Add(myTxt)
         Else
         ' Trường hợp cột CheckBox
            Dim chkCol As DataGridBoolColumn = _
            New DataGridBoolColumn
            With chkCol
               .MappingName = dc.ColumnName
               .HeaderText = dc.Caption
               .ReadOnly = False
               .AllowNull = False
               .FalseValue = 0
               .TrueValue = 1
               .Width = 50
            End With
            myColStyle.GridColumnStyles.Add( _
         chkCol)
         End If
      Next
      myColStyle.MappingName = dtView.TableName
      DataGrid1.TableStyles.Add(myColStyle)
End Sub
```

Giả sử, bạn đã thêm 5 mẩu tin vào bảng tblCustomer, sau khi người sử dụng nhấn nút Load, lập tức 5 mẩu tin xuất hiện trên điều khiển DataGrid như hình 24-8.



Hình 24-8: *Danh sách mẩu tin*

Nếu người sử dụng chọn vào một số mẩu tin như hình 24-8-1 và nhấn nút Delete, lập tức những mẩu tin này sẽ bị xóa.



Hình 24-8-1: *Chọn mẩu tin để xóa*

Để làm điều này, bạn khai báo để lấy ra danh sách mã khách hàng từ cột CustID ứng với những mẩu tin được chọn rồi gọi phương thức ExecuteSQL của đối tượng clsDatabase trong biến cố Click của nút Delete như sau:

```
Private Sub btnDelete_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles btnDelete.Click
    Dim strValues As String = ""
    'Duyệt trên từng mẩu tin
    For i As Integer = 0 To dtView.Rows.Count - 1
      'Nếu có chọn
      If DataGrid1.Item(i, 0) = True Then
        strValues += DataGrid1.Item(i, _
          1).ToString + ","
      End If
    Next
    'Nếu người sử dụng chọn
    If strValues <> "" Then
      'Loại bỏ dấu phẩy cuối cùng
      strValues = x.Left(strValues, _
        strValues.Length - 1)
      'Gọi phương thức ExecuteSQL
      cls.ExecuteSQL(strValues)
    End If
End Sub
```

Lưu ý, trong ví dụ trên chúng ta sử dụng phương thức Left của không gian tên Microsoft.VisualBasic bằng định danh x, vì vậy bạn khai báo định danh này như sau:

```
Imports x = Microsoft.VisualBasic
```

5.1.2. Phương thức ExecuteScalar

1. Thiết kế *Form*, cho phép người sử dụng chọn một sản phẩm trong điều khiển *ComboBox*, sau đó in ra số lượng đang còn trong kho của sản phẩm đó.

Bạn khai báo phương thức có tên GetValue nhận tham biến là đối tượng DataTable và ba tham trị là tên bảng dữ liệu, cột sẽ khai báo cho thuộc tính ValueMember và cột khai báo cho thuộc tính DisplayMember.

```
Public Function GetValue(ByRef dtView As
DataTable, ByVal strTable As String, ByVal
strName As String, ByVal strValue As String) As
String
    ' Định nghĩa phát biểu SQL
    strSQL = "select " + strValue + ","
    strSQL += strName + " from " + strTable
    Dim myData As SqlDataAdapter
    Try
      myData = New SqlDataAdapter(strSQL, myCon)
      myData.Fill(dtView)
      strError = "OK"
    Catch ex As Exception
      strError = ex.Message
    Finally
      myData.Dispose()
    End Try
    Return strError
End Function
```

Kế đến, cài đặt phương thức Overload là GetValue nhận tham trị là chuỗi và trả về giá trị của cột thuộc hàng thứ nhất từ việc gọi phương thức ExecuteScalar.

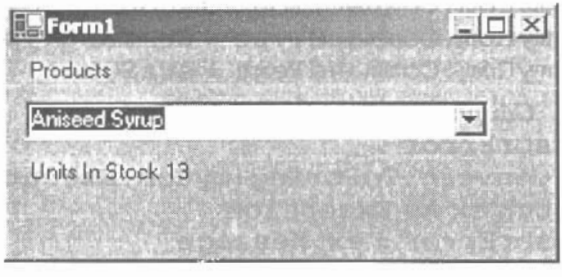
```
Public Function GetValue(ByVal strID As String)
As String
    Dim myCom As SqlCommand
    ' Khai báo phát biểu SQL
    strSQL = "select UnitsInStock from "
    strSQL += "products where Productid='"
    strSQL += strID + "'"
    Try
      myCom = New SqlCommand
      myCom.Connection = myCon
      myCom.CommandType = CommandType.Text
      myCom.CommandText = strSQL
      ' Gọi phương thức ExecuteScalar
      strError = _
      Convert.ToString(myCom.ExecuteScalar)
    Catch ex As Exception
      strError = ex.Message
    Finally
      myCom.Dispose()
    End Try
```

```
        Return strError
    End Function
```

Để liệt kê danh sách tên và mã sản phẩm vào điều khiển ComboBox, bạn gọi phương thức GetValue trong biến cố Load của Form.

```
Private Sub Form1_Load(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles MyBase.Load
    Dim dtView As New DataTable
    Dim cls As New clsDatabase
    'Nếu mở kết nối thành công
    If cls.OpenConnection = "OK" Then
      'Gọi phương thức GetValue để điền dữ liệu vào điều
      'khiển ComboBox
      cls.GetValue(dtView, "Products", _
        "productid", "productname")
      If dtView.Rows.Count > 0 Then
        cbID.DataSource = dtView
        cbID.DisplayMember = "productname"
        cbID.ValueMember = "productid"
      End If
      'Gọi phương thức GetValue để lấy giá trị trong cột
      'UnitsInStock
      lbQtty.Text = "Units In Stock " + _
        cls.GetValue(cbID.SelectedValue)
      cls.CloseConnection()
    End If
End Sub
```

Mỗi khi người sử dụng thay đổi tên sản phẩm, lập tức giá trị trong cột UnitsInStock sẽ xuất hiện như hình 24-9.



Hình 24-9: *Lấy giá trị cột UnitsInStock*

2. Thiết kế thủ tục nội tại để thêm mẩu tin vào bảng có cột số tự động, sau đó khai báo và sử dụng thủ tục này với mục đích mỗi khi thêm một mẩu tin bạn lấy ra được số tự động đó. Sau đó, thiết kế *Form* cho phép người sử dụng thêm mẩu tin và trình bày số tự động lấy được ra màn hình.

Giả sử, chúng ta đã có bảng tblCustomer đã được tạo ra trong ví dụ trước, bằng cách khai báo phương thức nhận 3 tham trị tương ứng với Name, Address và City, rồi sử dụng phương thức ExecuteScalar để thực thi thủ tục như sau:

```
Public Function ExecuteSQL(ByVal Name As String,
ByVal Address As String, ByVal City As String) As
String
    strSQL = "spInsCustomer"
    Dim myCom As SqlCommand
    Try
      ' Khai báo và khởi tạo đối tượng SqlCommand
      myCom = New SqlCommand(strSQL, myCon)
      ' Khai báo tham số
      myCom.Parameters.Add("@Name", Name)
      myCom.Parameters.Add("@Address", _
        Address)
      myCom.Parameters.Add("@City", City)
      ' Thực thi phát biểu SQL
      strError = _
      Convert.ToString(myCom.ExecuteScalar)
    Catch ex As Exception
      strError = ex.Message
    Finally
      myCom.Dispose()
    End Try
    Return strError
End Function
```

Trong đó, thủ tục nội tại dùng để thêm mẩu tin vào bảng tbCustomer có cấu trúc như sau:

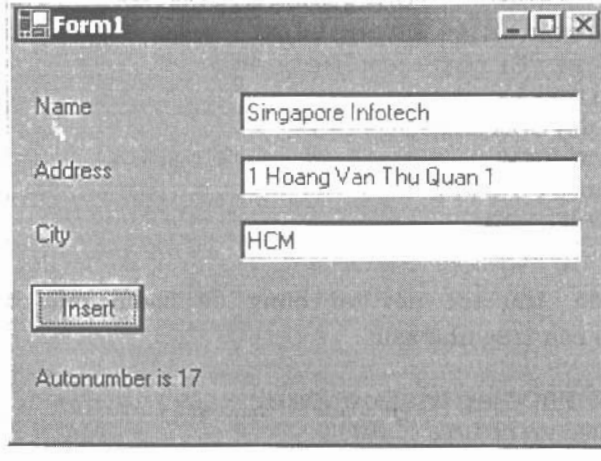
```
CREATE PROC spInsCustomer
  @Name varchar(100),
  @Address varchar(100),
  @City char(3)
```

```
AS
   Insert into tblCustomer(
     CustName,Address,City)
   values(@Name,@Address,@City)
   select @@identity
```

Sau đó, bạn khai báo để gọi phương thức ExecuteSQL trong biến cố Click của nút Insert như sau:

```
Private Sub Button2_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles Button2.Click
    If txtName.Text <> "" Then
      Dim cls As New clsDatabase
      ' Nếu mở kết nối cơ sở dữ liệu thành công
      If cls.OpenConnection() = "OK" Then
        ' Thực thi thủ tục nội tại
        lbResult.Text = "Autonumber is " + _
          cls.ExecuteSQL(txtName.Text, _
        txtAdd.Text, txtCity.Text)
        cls.CloseConnection()
      End If
    End If
End Sub
```

Khi thực thi chương trình, nếu người sử dụng nhập giá trị vào các điều khiển TextBox và nhấn nút Insert, kết quả số tự động trình bày như hình 24-10.



Hình 24-10: *Lấy số tự động*

3. Thiết kế *Form*, liệt kê danh sách các *Country*, mỗi khi người sử dụng chọn một *Country* bạn lấy ra khách hàng có số tiền đặt hàng lớn nhất thuộc *Country* đó, đồng thời trình bày danh sách các đơn đặt hàng của khách hàng đó trên điều khiển *DataGrid*.

Để thực hiện ví dụ này, trước tiên bạn khai báo phương thức trong lớp clsDatabase, nhận tham biến là ArrayList, sau đó đọc cột Country của bảng Customers và điền vào đối tượng này.

```
Public Function GetValue(ByRef arrValue As
ArrayList, ByVal strTable As String, ByVal
strName As String, ByVal strValue As String) As
String
    'Khai báo phát biểu SQL
    strSQL = "select distinct " + strValue + ","
    strSQL += strName + " from " + strTable
    Dim myCom As SqlCommand
    Dim RD As SqlDataReader
    Try
      myCom = New SqlCommand(strSQL, myCon)
      RD = myCom.ExecuteReader
    'Đọc từng mẩu tin
      While RD.Read
        'Điền vào ArrayList
        arrValue.Add(RD.GetString(0))
      End While
      strError = "OK"
    Catch ex As Exception
      strError = ex.Message
    Finally
      RD.Close()
    End Try
    Return strError
End Function
```

Trở lại Form1, bạn khai báo để gọi phương thức GetValue trong biến cố Load của Form để liệt kê danh sách các Country trên điều khiển ComboBox như sau:

```
Private Sub Form1_Load(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles MyBase.Load
```

```
    Dim cls As New clsDatabase
    'Nếu kết nối cơ sở dữ liệu thành công
    If cls.OpenConnection = "OK" Then
      'Khai báo và khởi tạo đối tượng ArrayList
      Dim arrValue As New ArrayList
      'Gọi phương thức GetValue
      cls.GetValue(arrValue, _
        "Customers", "Country", "Country")
      'Điền Country vào ComboBox từ đối tượng ArrayList
      For Each strValue As String In arrValue
        cbID.Items.Add(strValue)
      Next
      cls.CloseConnection()
    End If
End Sub
```

Kế đến, bạn khai báo phương thức thứ hai cùng tên, nhận tham biến thứ nhất là đối tượng DataTable và tham trị thứ hai là giá trị Country.

Bằng cách sử dụng mệnh đề Where để kết nối các bảng Customers, Orders, Order details và Products lại với nhau và điền dữ liệu vào đối tượng DataTable như sau:

```
Public Function GetValue(ByRef dtView As
DataTable, ByVal strID As String) As String
    Dim myCom As SqlCommand
    strSQL = "select D.ProductID,ProductName,"
    strSQL += " Sum(Quantity*D.Unitprice)  "
    strSQL += "As Amount from Customers C, "
    strSQL += "Orders O,[Order details] D ,"
    strSQL += " Products P where "
    strSQL += " C.Customerid = O.CustomerID "
    strSQL += " and O.orderid = D.OrderID "
    strSQL += " and D.Productid = D.Productid "
    strSQL += " and Country='" + strID + "' Group"
    strSQL += " by D.ProductID,ProductName"
    Dim myData As SqlDataAdapter
    Try
      myData = New SqlDataAdapter(strSQL, myCon)
      myData.Fill(dtView)
      strError = "OK"
    Catch ex As Exception
```

```
            strError = ex.Message
        Finally
            myData.Dispose()
        End Try
        Return strError
    End Function
```

Lưu ý, để liệt kê danh sách sản phẩm cùng với tổng giá trị đặt hàng của mỗi sản phẩm, bạn sử dụng phát biểu SQL như sau:

```
select D.ProductID, ProductName,
Sum(Quantity*D.Unitprice) from Customers C,
Orders O, [Order details] D , Products P where
C.Customerid = O.CustomerID and O.orderid =
D.OrderID and D.Productid = D.Productid
Group by D.ProductID, ProductName
```

Để cho phép lấy tổng giá trị đặt hàng của mỗi Country chọn từ điều khiển ComboBox, bạn khai báo phương thức GetValue trong clsDatabase như sau:

```
Public Function GetValue(ByVal strID As String)
As String
    Dim myCom As SqlCommand
    ' Khai báo phát biểu SQL với mệnh đề GROUP BY
    strSQL = "select Sum(Quantity*Unitprice) "
    strSQL += "from Customers C, Orders O,"
    strSQL += "[Order details] D where "
    strSQL += " C.Customerid = O.CustomerID "
    strSQL += " and O.orderid = D.OrderID "
    strSQL += " and Country='" + strID + "'"
    Try
        myCom = New SqlCommand
        myCom.Connection = myCon
        myCom.CommandType = CommandType.Text
        myCom.CommandText = strSQL
        strError = _
        Convert.ToString(myCom.ExecuteScalar)
    Catch ex As Exception
        strError = ex.Message
    Finally
        myCom.Dispose()
    End Try
```

```
    Return strError
End Function
```

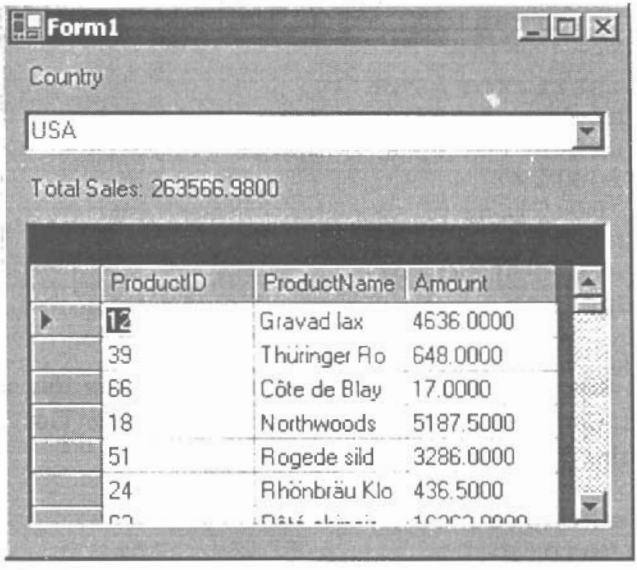
Tương tự như trình bày liệt kê danh sách sản phẩm, bạn có thể sử dụng phát biểu SQL như sau để tính tổng giá trị đặt hàng của tất cả sản phẩm thuộc một Country đã chọn như sau:

```
Select Sum(Quantity*D.Unitprice)
from Customers C, Orders O,
[Order details] D , Products P
where C.Customerid = O.CustomerID
and O.orderid = D.OrderID
and D.Productid = D.Productid
and Country='USA'
```

Mỗi khi người sử dụng chọn Country trên điều khiển ComboBox, bạn khai báo để gọi phương thức GetValue của lớp clsDatabase trong biến cố SelectedIndexChanged như sau:

```
Private Sub cbID_SelectedIndexChanged(ByVal
sender As System.Object, ByVal e As
System.EventArgs) Handles
cbID.SelectedIndexChanged
    Dim cls As New clsDatabase
    Dim dtView As New DataTable
    'Nếu kết nối cơ sở dữ liệu thành công
    If cls.OpenConnection = "OK" Then
      'Điền danh sách sản phẩm vào điều khiển DataGrid
      cls.GetValue(dtView, cbID.Text)
      DataGrid1.DataSource = dtView
      'Tính tổng giá trị đặt hàng của Country đó
      lbQtty.Text = "Total Sales: " + _
        cls.GetValue(cbID.Text)
      cls.CloseConnection()
    End If
End Sub
```

Khi thực thi chương trình, nếu người sử dụng chọn một trong những Country trên ComboBox, lập tức danh sách sản phẩm cùng với tổng giá trị đặt hàng của Country đó trình bày như hình 24-11.



Hình 24-11: *Tổng giá trị đặt hàng*

5.1.3. Phương thức ExecuteReader

1. Viết ứng dụng *Console*, yêu cầu người sử dụng nhập vào tên cơ sở dữ liệu, bạn liệt kê tên của các đối tượng cơ sở dữ liệu cùng với ngày tạo ra nó.

Để thực hiện ví dụ này, bạn khai báo phương thức GetValue trong lớp clsDatabase như sau:

```
Public Function GetValue(ByRef arrValue As
ArrayList, ByVal strDatabase As String) As
String
    ' Khai báo phát biểu SQL
    strSQL = "select name from " + strDatabase
    strSQL += ".dbo.sysobjects "
    strSQL += "where type='U'"
    Dim myCom As SqlCommand
    Dim RD As SqlDataReader
    Try
      myCom = New SqlCommand(strSQL, myCon)
      RD = myCom.ExecuteReader
      ' Duyệt trên từng Table
      While RD.Read
        ' Thêm vào mảng
```

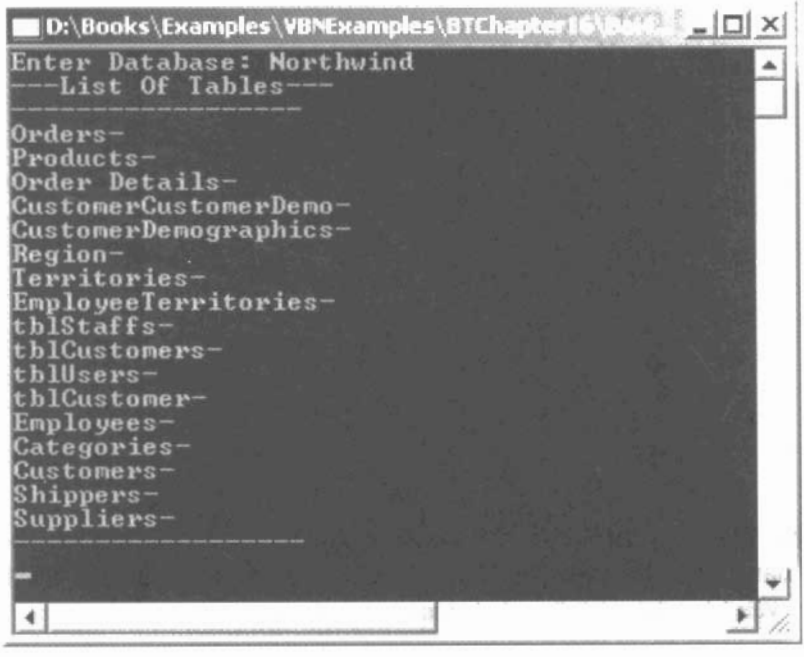
```
            arrValue.Add(RD.GetString(0))
          End While
          strError = "OK"
        Catch ex As Exception
          strError = ex.Message
        Finally
          RD.Close()
        End Try
        Return strError
    End Function
```

Sau đó, khai báo để yêu cầu người sử dụng nhập tên cơ sở dữ liệu từ màn hình Command Prompt và gọi phương thức GetValue từ lớp clsDatabase như sau:

```
Module Module1
  Sub Main()
    Dim strError As String
    Dim strDB As String
    'Yêu cầu người sử dụng nhập tên cơ sở dữ liệu
    Console.Write("Enter Database: ")
    strDB = Console.ReadLine
    'Khai báo và khởi tạo đối tượng clsDatabase
    Dim cls As New clsDatabase
    'Nếu mở kết nối cơ sở dữ liệu thành công
    If cls.OpenConnection() = "OK" Then
      If strDB = "" Then strDB = "Northwind"
      'Khai báo và khởi tạo đối tượng ArrayList
      Dim arrList As New ArrayList
      'Gọi phương thức GetValue
      strError = cls.GetValue(arrList, strDB)
      Console.WriteLine("---List Of Tables---")
      Console.WriteLine("--------------------")
      If strError.Equals("OK") Then
        'In ra màn hình Command Prompt danh sách Table
        For Each strName As String In arrList
          Console.WriteLine("{0}-", strName)
        Next
        Console.WriteLine("----------------")
        cls.CloseConnection()
      End If
    Else
```

```
            Console.WriteLine("Error: {0}", strError)
        End If
        Console.ReadLine()
    End Sub
End Module
```

Khi thực thi chương trình, nếu người sử dụng nhập vào tên cơ sở dữ liệu hợp lệ, lập tức danh sách bảng dữ liệu sẽ xuất hiện như hình 24-12.



Hình 24-12: *Liệt kê danh sách Table*

2. Thiết kế *Form*, cho phép liệt kê danh sách *Country* trong bảng *Customers* vào điều khiển *ComboBox*.

Để thực hiện ví dụ này, trước tiên bạn khai báo lớp có tên là clsItem bao gồm hai thuộc tính Value và Name tương ứng với giá trị trình bày trên điều khiển ComboBox.

```
Public Class clsItem
    ' Khai báo hai biến cục bộ
    Private mValue As String
    Private mName As String
    ' Khai báo Constructor
```

```
    Public Sub New(ByVal strValue As String, _
        ByVal strName As String)
      mValue = strValue
      mName = strName
    End Sub
    ' Khai báo hai thuộc tính
    Property Name() As String
      Get
        Return mName
      End Get
      Set(ByVal Value As String)
        mName = Value
      End Set
    End Property

    Property Value() As String
      Get
        Return mValue
      End Get
      Set(ByVal Value As String)
        mValue = Value
      End Set
    End Property
  End Class
```

Kế đến, bạn khai báo phương thức GetValue để đọc giá trị của hai cột từ bảng dữ liệu điền vào đối tượng ArrayList thông qua đối tượng clsItem vừa khai báo ở trên như sau:

```
Public Function GetValue(ByRef arrValue As
ArrayList, ByVal strTable As String, ByVal
strName As String, ByVal strValue As String) As
String
    strSQL = "select distinct " + strValue + ","
    strSQL += strName + " from " + strTable
    Dim myCom As SqlCommand
    Dim RD As SqlDataReader
    Try
      myCom = New SqlCommand(strSQL, myCon)
      RD = myCom.ExecuteReader
      ' Khai báo đối tượng clsItem
      Dim Item As clsItem
      While RD.Read
```

```
            'Khởi tạo đối tượng clsItem ứng với hai giá trị
            Item = New clsItem(RD.GetString(0), _
            RD.GetString(1))
            'Thêm đối tượng clsItem vào đối tượng ArrayList
            arrValue.Add(Item)
          End While
          strError = "OK"
        Catch ex As Exception
          strError = ex.Message
        Finally
          RD.Close()
        End Try
        Return strError
    End Function
```

Sau đó, trong biến cố Load của Form, bạn khai báo để điền danh sách Country vào điều khiển ComboBox như sau:

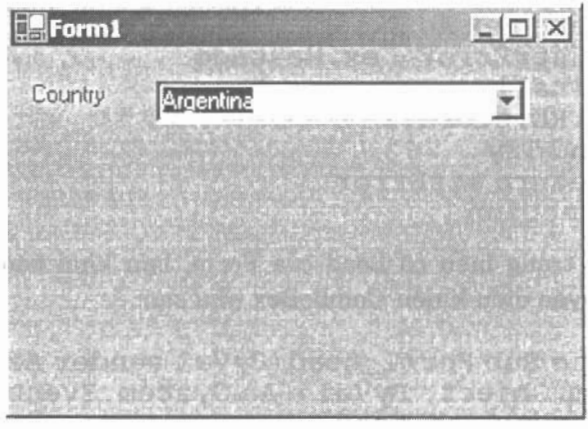
```
Private Sub Form1_Load(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles MyBase.Load
    'Khai báo và khởi tạo đối tượng clsDatabase
    Dim cls As New clsDatabase
    'Nếu kết nối cơ sở dữ liệu thành công
    If cls.OpenConnection = "OK" Then
      'Khai báo và khởi tạo đối tượng ArrayList
      Dim arrValue As New ArrayList
      'Gọi phương thức GetValue
      cls.GetValue(arrValue, _
         "Customers", "Country", "Country")
      'Định nghĩa các thuộc tính của ComboBox
      cbID.ValueMember = "Value"
      cbID.DisplayMember = "Name"
      cbID.DataSource = arrValue
      cls.CloseConnection()
    End If
End Sub
```

Lưu ý, bạn cần thêm Module vào Project và khai báo các biến dùng cho kết nối cơ sở dữ liệu.

```
Module Module1
  Public gsServer As String = "."
```

```
    Public gsDatabase As String = "Northwind"
    Public gsUserID As String
    Public gsWelcome As String = "huukhang.com"
End Module
```

Khi thực thi chương trình, nếu không có lỗi phát sinh thì danh sách Country sẽ xuất hiện trên điều khiển ComboBox như hình 24-13.



Hình 24-13: *Danh sách Country*

6. LÀM VIỆC VỚI ĐỐI TƯỢNG THAM SỐ

1. Xây dựng lớp có phương thức sử dụng đối tượng *SqlParameter* cho phép thêm dữ liệu vào bảng trong cơ sở dữ liệu *SQL Server*.

Bằng cách khai báo một phương thức ExecuteSQL nhận tham trị là tên bảng dữ liệu, mảng các tham số, mảng chứa giá trị tương ứng với tham số, sau đó duyệt trên từng phần tử tham số, bạn định nghĩa phát biểu Insert rồi thực thi chúng như sau:

```
Function ExecuteSQL(ByVal strTable As String,
ByVal arrPara() As String, ByVal arrValue() As
String) As String
    Dim strError As String = ""
    Try
        ' Khai báo và khởi tạo đối tượng SqlCommand
        Dim myCom As New SqlCommand
        ' Định nghĩa thuộc tính
        myCom.Connection = myCon
        myCom.CommandType = CommandType.Text
```

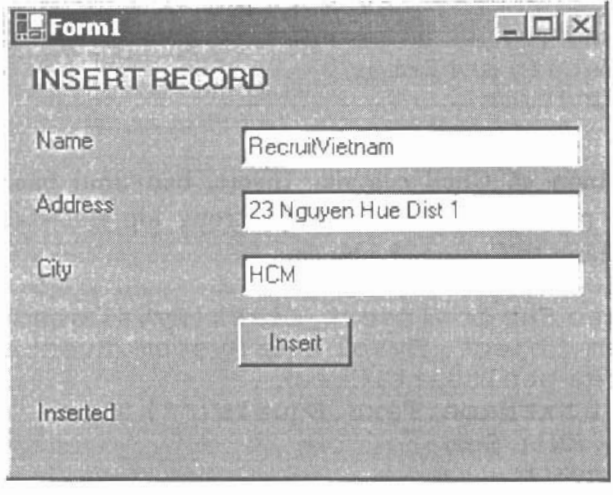
```
        ' Khai báo đối tượng SqlParameter
        Dim myPara As SqlParameter
        ' Định nghĩa phát biểu SQL
        Dim strSQL As String = "insert into "
        Dim strValue As String = " values("
        strSQL += strTable + "("
        ' Khai báo và khởi tạo đối tượng SqlCommand
        For i As Integer = 0 To arrPara.Length - 1
          strSQL += arrPara(i) + ","
          strValue += "@" + arrPara(i) + ","
          myPara = New SqlParameter( _
            "@" + arrPara(i), arrValue(i))
        ' Thêm tham số vào đối tượng SqlCommand
          myCom.Parameters.Add(myPara)
        Next
        ' Loại bỏ dấu phẩy cuối cùng
        strValue = strValue.Substring(0, _
          strValue.Length - 1) + ")"
        strSQL = strSQL.Substring(0, _
          strSQL.Length - 1) + ")"
        myCom.CommandText = strSQL + strValue
        ' Thực thi phát biểu SQL
        myCom.ExecuteNonQuery()
        strError = "OK"
      Catch ex As Exception
        strError = ex.Message
      End Try
      Return strError
  End Function
```

Trong biến cố Click của nút Insert, bạn khai báo đoạn chương trình để gọi phương thức ExecuteSQL trong lớp clsDatabase để thêm mẩu tin vào bảng tblCustomer như sau:

```
Private Sub btnInsert_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles btnInsert.Click
    If txtName.Text.Equals("") Then
      Exit Sub
    End If
    ' Khai báo và khởi tạo đối tượng clsDatabase
```

```
        Dim cls As New clsDatabase
        'Định nghĩa mảng các tham số
        Dim arrPara() As String = _
        {"CustName", "Address", "City"}
        'Định nghĩa mảng các giá trị
        Dim arrValue() As String = _
        {txtName.Text, txtAdd.Text, txtCity.Text}
        ' Nếu kết nối cơ sở dữ liệu thành công
        If cls.OpenConnection().Equals("OK") Then
            ' Nếu thêm mẩu tin thành công
            If cls.ExecuteSQL("tblCustomer", _
                arrPara, arrValue).Equals("OK") Then
                ' Xóa dữ liệu trên màn hình
                lbResult.Text = "Inserted"
                txtName.Text = ""
                txtAdd.Text = ""
                txtCity.Text = ""
            End If
            cls.CloseConnection()
        End If
    End Sub
```

Khi thực thi chương trình, nếu người sử dụng nhập tên, địa chỉ và thành phố vào các điều khiển TextBox tương ứng rồi nhấn nút Insert, lập tức mẩu tin thêm vào bảng và kết quả trả về như hình 24-14:



Hình 24-14: *Thêm mẩu tin*

2. Tạo *Class* và định nghĩa phương thức sử dụng đối tượng *Parameters Collection* cho phép liệt kê dữ liệu trong bảng chỉ định bằng thủ tục nội tại của *SQL Server* có truyền tham số.

Giả sử, chúng ta khai báo thủ tục để liệt kê danh sách khách hàng trong bảng Customers theo Country như sau:

```
CREATE PROC spViewCustomer
  @Country varchar(20)
AS
  SELECT CustomerID,CompanyName,Address,City
  FROM Customers
  WHERE Country=@Country
```

Kế đến, bạn khai báo phương thức có tên GetValue nhận tham biến là đối tượng ArrayList, ba tham trị tương ứng với tên bảng dữ liệu và hai cột ứng với DisplayMember và ValueMember của điều khiển ComboBox.

```
Public Function GetValue(ByRef arrValue As
ArrayList, ByVal strTable As String, ByVal
strName As String, ByVal strValue As String) As
String
    ' Định nghĩa phát biểu SQL
    strSQL = "select distinct " + strValue + ","
    strSQL += strName + " from " + strTable
    Dim myCom As SqlCommand
    Dim RD As SqlDataReader
    Try
      ' Khởi tạo đối tượng SqlCommand
      myCom = New SqlCommand(strSQL, myCon)
      ' Gọi phương thức ExecuteReader
      RD = myCom.ExecuteReader
      While RD.Read
      ' Điền đối tượng vào đối tượng ArrayList
        arrValue.Add(RD.GetString(0))
      End While
      strError = "OK"
    Catch ex As Exception
      strError = ex.Message
    Finally
      RD.Close()
```

```
        End Try
        Return strError
    End Function
```

Tiếp tục, bạn khai báo phương thức GetValue (Overload), nhận tham biến là đối tượng DataTable và tham trị là giá trị của tham số Country như sau:

```
    Public Function GetValue(ByRef dtView As
    DataTable, ByVal strValue As String) As String
        strSQL = "spViewCustomer "
        Dim myData As SqlDataAdapter
        Try
          Dim myCom As New SqlCommand(strSQL)
          myCom.Connection = myCon
          ' Khai báo thuộc tính CommandType là thủ tục
          myCom.CommandType = _
            CommandType.StoredProcedure
          Dim myPara As SqlParameter
          ' Gán giá trị cho tham số Country
          myPara = New SqlParameter("@Country", _
            strValue)
          ' Thêm đối tượng Sqlparameter vào đối tượng
          ' SqlCommand
          myCom.Parameters.Add(myPara)
          myData = New SqlDataAdapter(myCom)
          ' Điền dữ liệu vào đối tượng DataTable
          myData.Fill(dtView)
          strError = "OK"
        Catch ex As Exception
          strError = ex.Message
        Finally
          myData.Dispose()
        End Try
        Return strError
    End Function
```

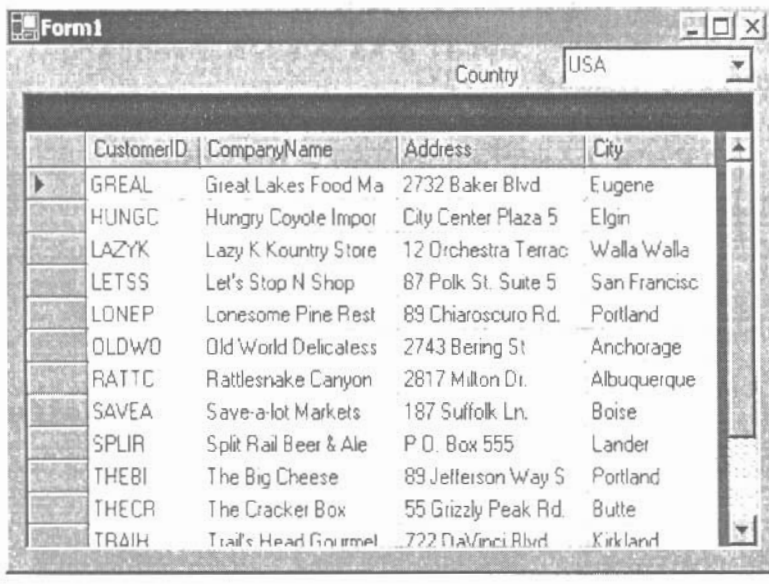
Sau đó, bạn khai báo để nạp danh sách Country vào điều khiển ComboBox trong biến cố Load của Form như sau:

```
Private Sub Form1_Load(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles MyBase.Load
    ' Nếu kết nối cơ sở dữ liệu thành công
    If cls.OpenConnection = "OK" Then
      ' Khai báo và khởi tạo đối tượng ArrayList
      Dim arrValue As New ArrayList
      ' Gọi phương thức GetValue
      cls.GetValue(arrValue, "Customers", _
        "Country", "Country")
      For Each strValue As String In arrValue
        cbID.Items.Add(strValue)
      Next
    End If
End Sub
```

Mỗi khi người sử dụng chọn Country, bạn gọi phương thức LoadData trong biến cố SelectedIndexChanged với tham trị là giá trị của thuộc tính Text từ ComboBox như sau:

```
Private Sub LoadData(ByVal Country As String)
    Dim dtView As DataTable
    dtView = New DataTable
    cls = New clsDatabase
    ' Nếu kết nối cơ sở dữ liệu thành công
    If cls.OpenConnection = "OK" Then
      ' Gọi phương thức GetValue
      cls.GetValue(dtView, Country)
      ' Nếu tồn tại mẩu tin thì điền vào điều khiển DataGrid
      If dtView.Rows.Count > 0 Then
        Me.DataGrid1.DataSource = dtView
      End If
    End If
End Sub
```

Kết quả trả về như hình 24-15.



Hình 24-15: *Liệt kê danh sách khách hàng*

Chuyên đề 17:
LÀM VIỆC VỚI ĐỐI TƯỢNG DATAREADER

1. ĐỐI TƯỢNG SQLDATAREADER

Để tìm hiểu thêm đối tượng *SqlDataReader*, bạn có thể thực hành các ví dụ sau:

1. Khai báo phương thức, nhận tham số là phát biểu *Select*, trả về một đối tượng bao gồm giá trị của các cột của hàng đầu tiên, sau đó sử dụng phương thức này từ phương thức *Main* của chương trình.

Để thực hiện ví dụ này, trước tiên bạn khai báo phương thức GetValue nhận tham biến là mảng hai chiều và tham trị là phát biểu SQL trong lớp clsDatabase như sau:

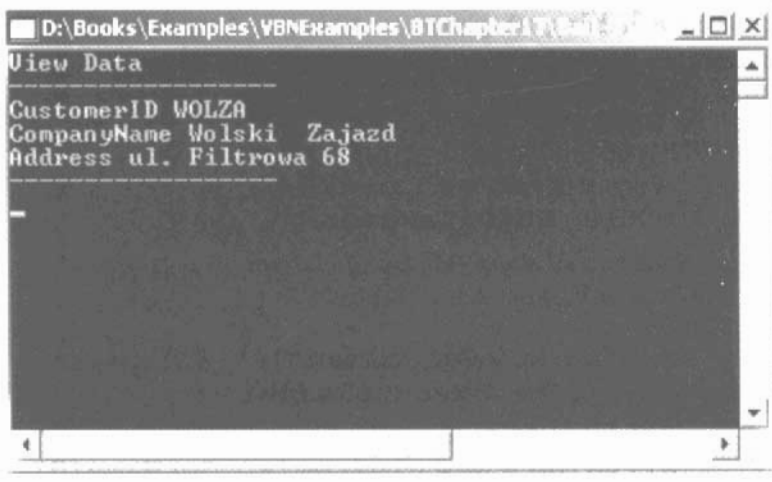
```
Public Function GetValue(ByRef arrValue(,) As
String, ByVal strSQL As String) As String
    Dim myCom As SqlCommand
    Dim RD As SqlDataReader
    Try
      myCom = New SqlCommand(strSQL, myCon)
      RD = myCom.ExecuteReader
      ' Khai báo lại kích thước mảng hai chiều
      ReDim arrValue(RD.FieldCount - 1, 1)
      ' Điền tên của cột dữ liệu vào mảng
      For i As Integer = 0 To RD.FieldCount - 1
        arrValue(i, 0) = RD.GetName(i)
      Next
      ' Điền giá trị của cột dữ liệu vào mảng
      While RD.Read
        For i As Integer = 0 To RD.FieldCount - 1
          arrValue(i, 1) = RD.GetString(i)
        Next
      End While
      RD.Close()
      strError = "OK"
    Catch ex As Exception
      strError = ex.Message
```

```
        Finally

        End Try
        Return strError
    End Function
```

Sau đó, bạn khai báo và khởi tạo đối tượng clsDatabase trong phương thức Main, rồi gọi phương thức GetValue với phát biểu SQL như sau:

```
Module Module1
  Sub Main()
    Console.WriteLine("View Data")
    Dim cls As New clsDatabase
    'Khai báo mảng hai chiều
    Dim arrValue(,) As String
    'Nếu kết nối cơ sở dữ liệu thành công
    If cls.OpenConnection() = "OK" Then
      'Gọi phương thức GetValue
      cls.GetValue(arrValue, _
      "Select CustomerID,CompanyName," & _
      "Address from Customers")
      Console.WriteLine("--- ----------")
      'Duyệt trên từng phần tử của mảng
      For i As Integer = 0 To _
          arrValue.GetLength(0) - 1
        For j As Integer = 0 To _
          arrValue.GetLength(1) - 1
          'In giá trị ra màn hình
          Console.Write("{0} ", arrValue(i, j))
        Next
        Console.WriteLine()
      Next
      Console.WriteLine("----------")
      cls.CloseConnection()
    End If
    Console.ReadLine()
  End Sub
End Module
```

Khi thực thi chương trình, lập tức mẩu tin đầu tiên của bảng Customers xuất hiện như hình 25-1.



Hình 25-1: *Mẫu tin đầu tiên*

2. Viết ứng dụng *Console*, liệt kê mã, tên, địa chỉ của danh sách khách hàng và nhà cung cấp trong hai bảng tương ứng có tên *Customers*, *Suppliers* của cơ sở dữ liệu *Northwind*.

Để liệt kê danh sách khách hàng trong bảng *Customers* và nhà cung cấp thuộc bảng *Suppliers*, bạn khai báo lớp clsItem với 3 biến cục bộ như sau:

```
Public Class clsItem
    Private mID As String
    Private mName As String
    Private mAddress As String

    ' Khai báo Constructor
    Public Sub New(ByVal strID As String, _
        ByVal strName As String, _
            ByVal strAddress As String)
        mID = strID
        mName = strName
        mAddress = strAddress
    End Sub
End Class
```

Kế đến, khai báo phương thức GetValue nhận tham biến là đối tượng ArrayList và tham trị là phát biểu SQL, bằng cách sử dụng phương thức NextResult của đối tượng SqlDataReader, bạn có thể đọc dữ

liệu 3 cột của mỗi bảng và ghi vào đối tượng ArrayList thông qua đối tượng clsItem.

```
Public Function GetValue(ByRef arrValues As
ArrayList, ByVal strSQL As String) As String
    Dim myCom As SqlCommand
    Dim RD As SqlDataReader
    'Khai báo sử dụng đối tượng clsItem
    Dim arrValue As clsItem
    Try
      myCom = New SqlCommand(strSQL, myCon)
      RD = myCom.ExecuteReader
      'Khởi tạo đối tượng clsItem với tên cột
      arrValue = New clsItem(RD.GetName(0), _
      RD.GetName(1), RD.GetName(2))
      'Thêm đối tượng clsItem vào đối tượng ArrayList
      arrValues.Add(arrValue)
      While RD.Read
        'Khởi tạo đối tượng clsItem với giá trị từng cột ứng
        'với từng hàng dữ liệu khách hàng
        arrValue=New clsItem(RD.GetString(0), _
        RD.GetString(1), RD.GetString(2))
      'Thêm đối tượng clsItem vào đối tượng ArrayList
        arrValues.Add(arrValue)
      End While
      'Nhảy đến tập mẩu tin kế tiếp
      RD.NextResult()
      'Khởi tạo đối tượng clsItem với tên cột
      arrValue = New clsItem(RD.GetName(0), _
      RD.GetName(1), RD.GetName(2))
      'Thêm đối tượng clsItem vào đối tượng ArrayList
      arrValues.Add(arrValue)
      While RD.Read
        'Khởi tạo đối tượng clsItem với giá trị từng cột ứng
        'với từng hàng dữ liệu nhà cung cấp
        arrValue = New clsItem( _
        Convert.ToString(RD.GetValue(0)), _
        RD.GetString(1), RD.GetString(2))
        'Thêm đối tượng clsItem vào đối tượng ArrayList
        arrValues.Add(arrValue)
```

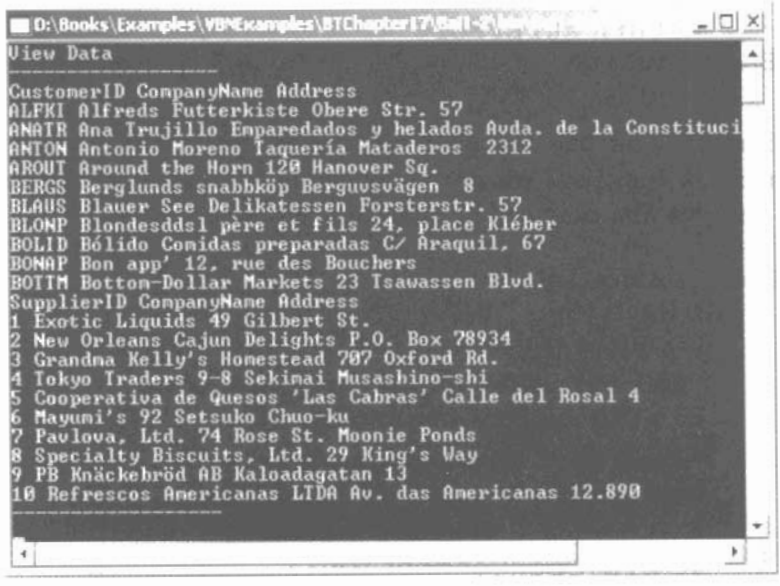
```
            End While
            RD.Close()
            strError = "OK"
        Catch ex As Exception
            strError = ex.Message
        Finally

        End Try
        Return strError
    End Function
```

Sau đó, khai báo hai phát biểu Select ứng với hai bảng dữ liệu Customers và Suppliers rồi gọi phương thức GetValue để in danh sách khách hàng và nhà cung cấp ra màn hình như sau:

```
Module Module1
  Sub Main()
    Console.WriteLine("View Data")
    'Khai báo và khởi tạo đối tượng clsDatabase
    Dim cls As New clsDatabase
    'Khai báo và khởi tạo đối tượng ArrayList
    Dim arrValues As New ArrayList
    'Nếu kết nối cơ sở dữ liệu thành công
    If cls.OpenConnection() = "OK" Then
      'Gọi phương thức GetValue
      cls.GetValue(arrValues, _
      "Select top 10 CustomerID,CompanyName" & _
      ",Address from Customers;" & _
      "Select top 10 SupplierID,CompanyName" & _
      ",Address from Suppliers;")
      Console.WriteLine("--------------------")
      'Duyệt qua từng phần tử
      For Each arrValue As clsItem In arrValues
        Console.Write("{0} ", arrValue.ID)
        Console.Write("{0} ", arrValue.Name)
        Console.Write("{0} ", arrValue.Address)
        Console.WriteLine()
      Next
      Console.WriteLine("--------------------")
      cls.CloseConnection()
    End If
    Console.ReadLine()
```

```
    End Sub
End Module
```

Khi thực thi chương trình, lập tức danh sách 10 khách hàng đầu tiên trong bảng Customers và 10 nhà cung cấp đầu tiên trong bảng Suppliers xuất hiện như hình 25-2.



Hình 25-2: *Danh sách khách hàng và nhà cung cấp*

3. Định nghĩa phương thức nhận tham số là phát biểu *Select*, trả về một mảng một chiều kiểu đối tượng chứa giá trị của các cột của 10 hàng đầu tiên, sau đó sử dụng phương thức này từ phương thức *Main* của chương trình.

Để thực hiện ví dụ này, trước tiên ban khai báo phương thức GetValue nhận tham biến là mảng hai chiều kiểu object và hai tham trị là phát biểu SQL và số mẩu tin cần lấy ra.

```
Public Function GetValue(ByRef arrObj(,) As
Object, ByVal strSQL As String, ByVal row As
Integer) As String
    Dim myCom As SqlCommand
    Dim RD As SqlDataReader
    Try
        Dim i As Integer
        myCom = New SqlCommand(strSQL, myCon)
```

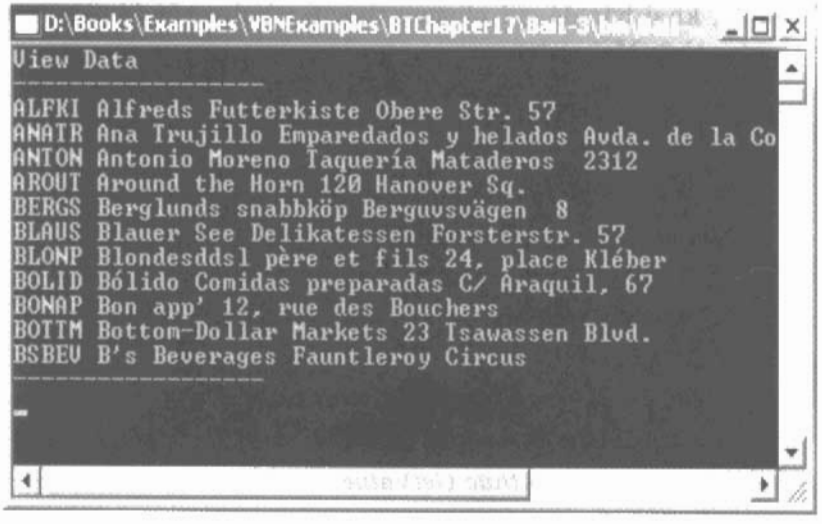
```
            RD = myCom.ExecuteReader
            ' Số cột dữ liệu
            j = RD.FieldCount - 1
            ' Khai báo lại kích thước mảng
            ReDim arrObj(row, j)
            Dim i As Integer = 0
            ' Điền dữ liệu vào mảng
            While RD.Read
               ' Duyệt trên từng cột dữ liệu
               For col As Integer = 0 To j
                  arrObj(i, col) = _
                  Convert.ToString(RD.GetValue(col))
               Next
               i += 1
            End While

            RD.Close()
            strError = "OK"
         Catch ex As Exception
            strError = ex.Message
         Finally

         End Try
         Return strError
      End Function
```

Sau đó, khai báo để gọi phương thức GetValue của lớp clsDatabase trong phương thức Main như sau:

```
Module Module1
   Sub Main()
      Console.WriteLine("View Data")
      ' Khai báo và khởi tạo đối tượng clsDatabase
      Dim cls As New clsDatabase
      ' Khai báo mảng hai chiều kiểu đối tượng
      Dim arrObj(,) As Object
      ' Nếu kết nối cơ sở dữ liệu thành công
      If cls.OpenConnection() = "OK" Then
         ' Gọi phương thức GetValue
         cls.GetValue(arrObj, _
```

```
            "Select CustomerID,CompanyName," & _
            "Address from Customers", 10)
            Console.WriteLine("--------------------")
            ' Duyệt trên từng chiều của mảng
            For i As Integer = 0 To _
              arrObj.GetLength(0) - 1
              For j As Integer = 0 To _
                arrObj.GetLength(1) - 1
                ' In kết quả ra màn hình
                Console.Write("{0} ", arrObj(i, j))
              Next
              Console.WriteLine()
            Next
            Console.WriteLine("---                    ")
            cls.CloseConnection()
        End If
        Console.ReadLine()
    End Sub
End Module
```

Khi thực thi chương trình, 10 mẩu tin đầu tiên xuất hiện tương tự như hình 25-3.



Hình 25-3: *Trình bày 10 mẩu tin*

2. ĐỐI TƯỢNG SQLDATAREADER VÀ ĐIỀU KHIỂN

Để tìm hiểu thêm đối tượng *SqlDataReader* và điều khiển, bạn có thể thực hành các ví dụ sau:

1. Viết phương thức, nhận tham số là phát biểu *Select*, sau đó kết nối cơ sở dữ liệu và liệt kê danh sách tên cột dữ liệu cùng với dữ liệu trên điều khiển *DataGrid* thông qua đối tượng *ArrayList*.

Bằng cách khai báo lớp tương ứng với bảng dữ liệu (tham khảo chuyên đề 22) như sau:

```
Public Class clsItem
  Private mID As String
  Private mName As String
  Private mAddress As String
  ' Khai báo Constructor
  Public Sub New(ByVal strID As String, _
  ByVal strName As String, _
  ByVal strAddress As String)
    mID = strID
    mName = strName
    mAddress = strAddress
  End Sub
  ' Khai báo thuộc tính Name
  Property Name() As String
    Get
      Return mName
    End Get
    Set(ByVal Value As String)
      mName = Value
    End Set
  End Property
  ' Khai báo thuộc tính ID
  Property ID() As String
    Get
      Return mID
    End Get
    Set(ByVal Value As String)
      mID = Value
    End Set
  End Property
```

```
    'Khai báo thuộc tính Address
    Property Address() As String
      Get
        Return mAddress
      End Get
      Set(ByVal Value As String)
        mAddress = Value
      End Set
    End Property
End Class
```

Kế đến, bạn khai báo phương thức GetValue trong lớp clsDatabas nhận tham biến là đối tượng ArrayList, tham trị phát biểu Select, s dụng đối tượng SqlDataReader đọc dữ liệu từ bảng và điền vào đối tượn ArrayList.

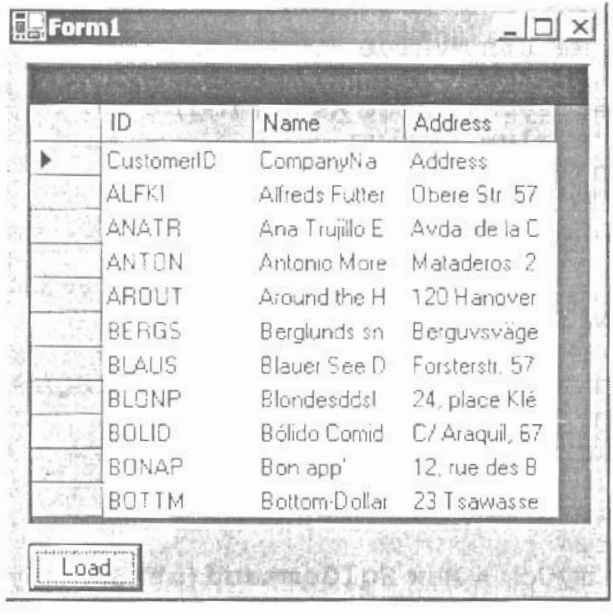
```
Public Function GetValue(ByRef arrObj As
ArrayList, ByVal Rows As Integer, ByVal strSQL As
String) As String
    Dim myCom As SqlCommand
    Dim RD As SqlDataReader
    Try
      myCom = New SqlCommand(strSQL, myCon)
      RD = myCom.ExecuteReader
      'Khai báo đối tượng clsItem
      Dim Item As clsItem
      Dim i As Integer
      'Khai báo tên cột
      Item = New clsItem(RD.GetName(0), _
        RD.GetName(1), RD.GetName(2))
      arrObj.Add(Item)
      'Khai báo giá trị của từng cột
      While RD.Read
        Item = New clsItem(RD.GetString(0), _
          RD.GetString(1), RD.GetString(2))
        arrObj.Add(Item)
        i += 1
      End While
      RD.Close()
      strError = "OK"
    Catch ex As Exception
      strError = ex.Message
    Finally
```

```
        End Try
        Return strError
    End Function
```

Sau đó, khai báo để gọi phương thức GetValue của lớp clsDatabase và điền dữ liệu từ đối tượng ArrayList vào điều khiển DataGrid như sau:

```
Private Sub btnLoad_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles btnLoad.Click
    Dim cls As New clsDatabase
    Dim list As ArrayList = New ArrayList
    If cls.OpenConnection() = "OK" Then
      cls.GetValue(list, _
      "Select CustomerID,CompanyName," & _
      "Address from Customers")
      DataGrid1.DataSource = list
      cls.CloseConnection()
    End If
End Sub
```

Khi thực thi chương trình, nếu người sử dụng nhấn nút Load, lập tức danh sách khách hàng xuất hiện như hình 25-4.



Hình 25-4: *Danh sách khách hàng*

2. Thiết kế *Form*, cho phép liệt kê danh sách *Country* trong bảng *Customers* vào điều khiển *ComboBox* thứ nhất và danh sách loại sản phẩm vào *ComboBox* thứ hai.

Tương tự như trên, bạn khai báo lớp clsItem gồm hai thuộc tính Value và Name như sau:

```
Public Class clsItem
  Private mValue As String
  Private mName As String
  Public Sub New(ByVal strID As String, _
    ByVal strName As String)
    mValue = strID
    mName = strName
  End Sub
  Property Name() As String
    Get
      Return mName
    End Get
    Set(ByVal Value As String)
      mName = Value
    End Set
  End Property
  Property Value() As String
    Get
      Return mValue
    End Get
    Set(ByVal Value As String)
      mValue = Value
    End Set
  End Property
End Class
```

Kế đến, bạn khai báo phương thức GetValue nhận tham trị là phát biểu SQL trả về đối tượng ArrayList.

```
Public Function GetValue(ByVal strSQL As
String) As ArrayList
    Dim myCom As SqlCommand
    Dim arrList As New ArrayList
    Dim RD As SqlDataReader
    Try
      myCom = New SqlCommand(strSQL, myCon)
      RD = myCom.ExecuteReader
      ' Khai báo đối tượng clsItem
```

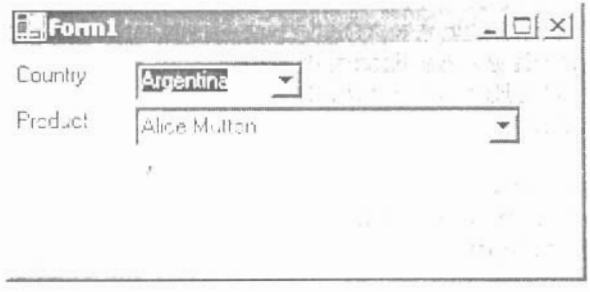
```
        Dim Item As clsItem
        Dim i As Integer
        'Duyệt qua từng mẩu tin
        While RD.Read
          Item = New clsItem( _
            Convert.ToString(RD.GetValue(0)), _
            RD.GetString(1))
          'Thêm vào đối tượng ArrayList
          arrList.Add(Item)
          i += 1
        End While
        RD.Close()
        strError = "OK"
      Catch ex As Exception
        strError = ex.Message
      Finally

      End Try
      Return arrList
  End Function
```

Sau đó, khai báo để gọi phương thức GetValue trong lớp clsDatabase trong biến cố Load của Form, để điền danh sách Country và Product vào điều khiển ComboBox như sau:

```
Private Sub Form1_Load(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles MyBase.Load
    'Khai báo và khởi tạo đối tượng clsDatabase
    Dim cls As New clsDatabase
    'Khai bái đối tượng ArrayList
    Dim list As ArrayList
    'Nếu kết nối cơ sở dữ liệu thành công
    If cls.OpenConnection() = "OK" Then
      'Điền dữ liệu vào cbCountry
      list = cls.GetValue( _
      "Select distinct Country" & _
      " ,Country from Customers")
      cbCountry.DataSource = list
      cbCountry.ValueMember = "Value"
      cbCountry.DisplayMember = "Name"
      'Điền dữ liệu vào cbProduct
```

```
            list = cls.GetValue( _
            "Select productid " & _
            " ,productname from Products")
            cbProduct.DataSource = list
            cbProduct.ValueMember = "Value"
            cbProduct.DisplayMember = "Name"
            cls.CloseConnection()
        End If
    End Sub
```

Khi thực thi chương trình, danh sách Country và Product trình bày trên hai điều khiển ComboBox như hình 25-5.



Hình 25-5: *Trình bày dữ liệu trên điều khiển ComboBox*

3 Bằng cách sử dụng điều khiển *ListView*, bạn có thể điền tên của bảng dữ liệu vào phần *Header*, sau đó điền dữ liệu của các cột tương ứng vào phần dưới của điều khiển *ListView*.

Để cho phép sử dụng đối tượng ArrayList với bảng dữ liệu không giới hạn số cột, trước tiên bạn khai báo lớp clsItem có cấu trúc như sau:

```
Public Class clsItem
    Private mName() As String

    ' Khai báo Constructor
    Public Sub New(ByVal strName() As String)
        mName = strName
    End Sub

    ' Khai báo thuộc tính GET và SET
    Property Name() As String()
        Get
            Return mName
        End Get
        Set(ByVal Value() As String)
            mName = Value
        End Set
```

```
    End Property
End Class
```

Lưu ý, thay vì sử dụng từng thuộc tính ứng với từng cột dữ liệu trong bảng, bạn có thể sử dụng mảng kiểu String.

Kế đến, bạn khai báo phương thức GetValue nhận tham trị là phát biểu SQL và trả về đối tượng ArrayList. Bằng cách sử dụng đối tượng clsItem, bạn có thể định nghĩa tên cột dữ liệu và giá trị của chúng như sau:

```
Public Function GetValue(ByVal strSQL As String)
As ArrayList
    Dim myCom As SqlCommand
    Dim arrList As New ArrayList
    Dim RD As SqlDataReader
    Try
      myCom = New SqlCommand(strSQL, myCon)
      RD = myCom.ExecuteReader
      ' Khai báo đối tượng clsItem
      Dim Item As clsItem
      ' Khai báo số cột dữ liệu
      Dim col As Integer = RD.FieldCount - 1
      ' Khai báo mảng giá trị
      Dim arrValue(col) As String
      ' Gán tên cột dữ liệu vào mảng
      For j As Integer = 0 To col
        arrValue(j) = RD.GetName(j)
      Next
      ' Khởi tạo đối tượng clsItem
      Item = New clsItem(arrValue)
      ' Thêm đối tượng clsItem vào đối tượng ArrayList
      arrList.Add(Item)
      Dim i As Integer
      ' Duyệt trên từng mẩu tin
      While RD.Read
        ReDim arrValue(col)
        ' Sử dụng phương thức GetValues với tham trị là mảng
        RD.GetValues(arrValue)
        ' Khởi tạo đối tượng clsItem
        Item = New clsItem(arrValue)
        ' Thêm đối tượng clsItem vao đối tượng ArrayList
```

```
            arrList.Add(Item)
            i += 1
          End While
          RD.Close()
          strError = "OK"
        Catch ex As Exception
          strError = ex.Message
        Finally

        End Try
        Return arrList
    End Function
```

Sau đó, bạn khai báo phương thức có tên FillData, nhận tham trị là đối tượng ArrayList, rồi đọc dữ liệu từ đối tượng này điền vào điều khiển ListView.

```
    Sub FillData(ByVal arrList As ArrayList)
        lvData.Clear()
        With lvData
          Dim i As Integer
          .Items.Clear()
          .CheckBoxes = False
          .View = View.Details
          ' Khai báo đối tượng ListViewItem
          Dim item1 As ListViewItem
          ' Khai báo đối tượng clsItem
          Dim Item As clsItem
          ' Duyệt trên từng phần tử của ArrayList
          For Each obj As clsItem In arrList
            ' Phần tử đầu tiên là tên cột
            If i = 0 Then
              For i = 0 To obj.Name().Length - 1
                .Columns.Add(obj.Name(i), 100, 0)
              Next
            Else
            ' Phần tử kế tiếp là giá trị
              item1 = New ListViewItem(obj.Name(0))
              For i = 1 To obj.Name().Length - 1
                item1.SubItems.Add(obj.Name(i))
              Next
              .Items.Add(item1)
            End If
```

```
        Next
    End With
End Sub
```

Bằng cách gọi phương thức GetValue của lớp clsDatabase trong biến cố Click của nút btnLoad như sau.

```
Private Sub btnLoad_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles btnLoad.Click
    'Khai báo và khởi tạo đối tượng clsDatabase
    Dim cls As New clsDatabase
    'Khai báo đối tượng ArrayList
    Dim list As ArrayList
    'Nếu kết nối cơ sở dữ liệu thành công
    If cls.OpenConnection() = "OK" Then
        'Gọi phương thức GetValue của đối tượng clsDatabase
        list = cls.GetValue( _
        "Select CustomerID, CompanyName" & _
        " ,Address,City from Customers")
        'Gọi phương thức FillData
        FillData(list)
        cls.CloseConnection()
    End If
End Sub
```

Khi thực thi chương trình, nếu người sử dụng nhấn nút Load, lập tức danh sách dữ liệu trình bày tương tự như hình 25-6.

Hình 25-6: *Điền dữ liệu trên điều khiển ListView*

4. Sử dụng điều khiển *TreeView*, bạn có thể liệt kê danh sách các loại đối tượng cơ sở dữ liệu và từng loại đối tượng cơ sở dữ liệu.

Tương tự như ví dụ trên, bạn khai báo phương thức GetValue nhận tham trị là phát biểu SQL và trả về đối tượng ArrayList. Bằng cách sử dụng đối tượng clsItem, ban có thể định nghĩa tên cột dữ liệu và giá trị của chúng như sau:

```
Public Function GetValue(ByVal strSQL As String)
As ArrayList
    Dim myCom As SqlCommand
    Dim arrList As New ArrayList
    Dim RD As SqlDataReader
    Try
      myCom = New SqlCommand(strSQL, myCon)
      RD = myCom.ExecuteReader
      ' Khai báo đối tượng clsItem
      Dim Item As clsItem
      ' Khai báo số cột dữ liệu
      Dim col As Integer = RD.FieldCount - 1
      ' Khai báo mảng giá trị
      Dim arrValue(col) As String
      ' Gán tên cột dữ liệu vào mảng
      For j As Integer = 0 To col
        arrValue(j) = RD.GetName(j)
      Next
      ' Khởi tạo đối tượng clsItem
      Item = New clsItem(arrValue)
      ' Thêm đối tượng clsItem vào đối tượng ArrayList
      arrList.Add(Item)
      Dim i As Integer
      ' Duyệt trên từng mẩu tin
      While RD.Read
        ReDim arrValue(col)
        ' Sử dụng phương thức GetValues với tham trị là mảng
        RD.GetValues(arrValue)
      ' Khởi tạo đối tượng clsItem
        Item = New clsItem(arrValue)
      ' Thêm đối tượng clsItem vào đối tượng ArrayList
        arrList.Add(Item)
```

```
            arrList.Add(Item)
            i += 1
          End While
          RD.Close()
          strError = "OK"
        Catch ex As Exception
          strError = ex.Message
        Finally

        End Try
        Return arrList
    End Function
```

Sau đó, bạn khai báo phương thức có tên FillData, nhận tham trị là đối tượng ArrayList, rồi đọc dữ liệu từ đối tượng này điền vào điều khiển ListView.

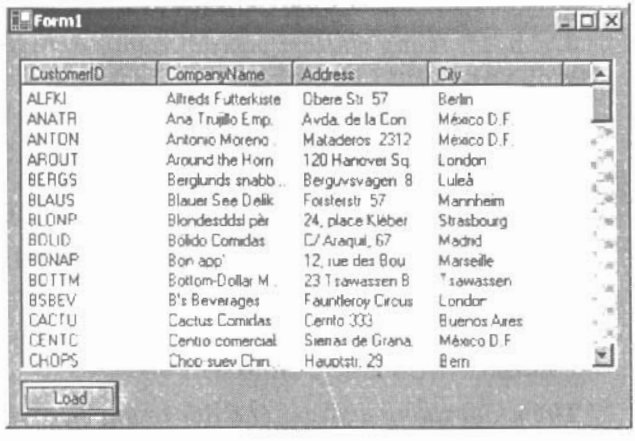
```
    Sub FillData(ByVal arrList As ArrayList)
        lvData.Clear()
        With lvData
          Dim i As Integer
          .Items.Clear()
          .CheckBoxes = False
          .View = View.Details
          ' Khai báo đối tượng ListViewItem
          Dim item1 As ListViewItem
          ' Khai báo đối tượng clsItem
          Dim Item As clsItem
          ' Duyệt trên từng phần tử của ArrayList
          For Each obj As clsItem In arrList
            ' Phần tử đầu tiên là tên cột
            If i = 0 Then
              For i = 0 To obj.Name().Length - 1
                .Columns.Add(obj.Name(i), 100, 0)
              Next
            Else
            ' Phần tử kế tiếp là giá trị
              item1 = New ListViewItem(obj.Name(0))
              For i = 1 To obj.Name().Length - 1
                item1.SubItems.Add(obj.Name(i))
              Next
              .Items.Add(item1)
            End If
```

```
        Next
    End With
End Sub
```

Bằng cách gọi phương thức GetValue của lớp clsDatabase trong biến cố Click của nút btnLoad như sau.

```
Private Sub btnLoad_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles btnLoad.Click
    ' Khai báo và khởi tạo đối tượng clsDatabase
    Dim cls As New clsDatabase
    ' Khai báo đối tượng ArrayList
    Dim list As ArrayList
    ' Nếu kết nối cơ sở dữ liệu thành công
    If cls.OpenConnection() = "OK" Then
        ' Gọi phương thức GetValue của đối tượng clsDatabase
        list = cls.GetValue( _
        "Select CustomerID, CompanyName" & _
        " ,Address,City from Customers")
        ' Gọi phương thức FillData
        FillData(list)
        cls.CloseConnection()
    End If
End Sub
```

Khi thực thi chương trình, nếu người sử dụng nhấn nút Load, lập tức danh sách dữ liệu trình bày tương tự như hình 25-6.



Hình 25-6: *Điền dữ liệu trên điều khiển ListView*

4. Sử dụng điều khiển *TreeView*, bạn có thể liệt kê danh sách các loại đối tượng cơ sở dữ liệu và từng loại đối tượng cơ sở dữ liệu.

Tương tự như ví dụ trên, bạn khai báo phương thức GetValue nhận tham trị là phát biểu SQL và trả về đối tượng ArrayList. Bằng cách sử dụng đối tượng clsItem, bạn có thể định nghĩa tên cột dữ liệu và giá trị của chúng như sau:

```
Public Function GetValue(ByVal strSQL As String)
As ArrayList
    Dim myCom As SqlCommand
    Dim arrList As New ArrayList
    Dim RD As SqlDataReader
    Try
      myCom = New SqlCommand(strSQL, myCon)
      RD = myCom.ExecuteReader
      ' Khai báo đối tượng clsItem
      Dim Item As clsItem
      ' Khai báo số cột dữ liệu
      Dim col As Integer = RD.FieldCount - 1
      ' Khai báo mảng giá trị
      Dim arrValue(col) As String
      ' Gán tên cột dữ liệu vào mảng
      For j As Integer = 0 To col
        arrValue(j) = RD.GetName(j)
      Next
      ' Khởi tạo đối tượng clsItem
      Item = New clsItem(arrValue)
      ' Thêm đối tượng clsItem vào đối tượng ArrayList
      arrList.Add(Item)
      Dim i As Integer
      ' Duyệt trên từng mẩu tin
      While RD.Read
        ReDim arrValue(col)
        ' Sử dụng phương thức GetValues với tham trị là mảng
        RD.GetValues(arrValue)
      ' Khởi tạo đối tượng clsItem
        Item = New clsItem(arrValue)
      ' Thêm đối tượng clsItem vào đối tượng ArrayList
        arrList.Add(Item)
```

```
            i += 1
        End While
        RD.Close()
        strError = "OK"
    Catch ex As Exception
        strError = ex.Message
    Finally

    End Try
    Return arrList
End Function
```

Kế đến, khai báo phương thức có tên FillData, nhận tham trị là đối tượng ArrayList, rồi đọc dữ liệu từ đối tượng này điền vào điều khiển TreeView.

```
Sub FillData(ByVal arrList As ArrayList)
    'Xóa các Node đang tồn tại
    tvData.Nodes.Clear()
    With tvData
        Dim i As Integer = 0
        .CheckBoxes = False
        'Khai báo đối tượng Node
        Dim myNode As TreeNode
        'Khai báo đối tượng clsItem
        Dim Item As clsItem
        'Duyệt trên từng phần tử của đối tượng ArrayList
        For Each obj As clsItem In arrList
            'Tạo ra các Node cha với mã khách hàng
            .Nodes.Add(obj.Name(0))
            'Tạo từng Node con là tên, địa chỉ, ...
            For j As Integer = 1 To _
                obj.Name().Length - 1
                myNode = New TreeNode(obj.Name(j))
                .Nodes(i).Nodes.Add(myNode)
            Next
            i += 1
        Next
    End With
End Sub
```

Sau đó, gọi phương thức GetValue của lớp clsDatabase từ biến cố Click của nút btnLoad như sau.

```
Private Sub btnLoad_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles btnLoad.Click
    ' Khai báo và khởi tạo đối tượng clsDatabase
    Dim cls As New clsDatabase
    ' Khai báo đối tượng ArrayList
    Dim list As ArrayList
    ' Nếu kết nối cơ sở dữ liệu thành công
    If cls.OpenConnection() = "OK" Then
        ' Gọi phương thức GetValue của đối tượng clsDatabase
        list = cls.GetValue( _
        "Select CustomerID, CompanyName" & _
        " ,Address,City from Customers")
        ' Gọi phương thức FillData
        FillData(list)
        cls.CloseConnection()
    End If
End Sub
```

Khi thực thi chương trình, nếu người sử dụng nhấn nút Load, lập tức danh sách dữ liệu trình bày với dạng cây tương tự như hình 25-7.

Hình 25-7: *Trình bày dữ liệu trên điều khiển TreeView*

Chuyên đề 18:
ĐỐI TƯỢNG DATAADAPTER VÀ DATASET

1. ĐỐI TƯỢNG SQLDATAADAPTER

1. Tạo ứng dụng *Console*, sau đó đọc danh sách các bảng dữ liệu từ một tên cơ sở dữ liệu nhập từ bàn phím điền vào đối tượng *DataTable* và liệt kê chúng ra màn hình.

Để thực hiện ví dụ này, trước tiên bạn khai báo phương thức GetValue nhận tham biến là đối tượng DataTable và tham trị là phát biểu SQL, sau đó sử dụng phương thức Fill của đối tượng SqlDataAdapter để điền tập dữ liệu vào đối tượng DataTable như sau:

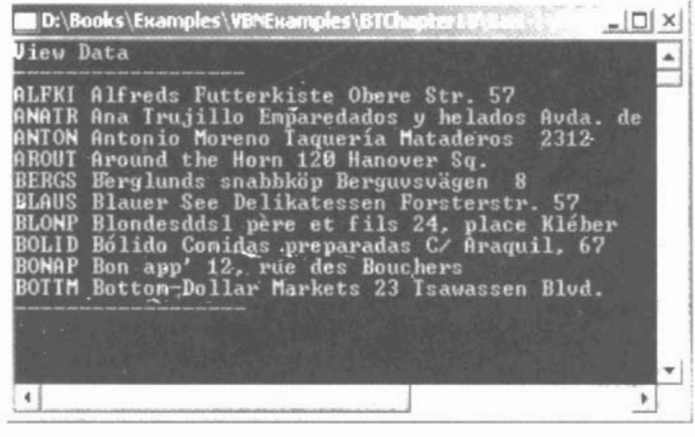
```
Public Function GetValue( _
  ByRef myDT As DataTable, _
  ByVal strSQL As String) As String
    Dim myCom As SqlCommand
    Dim adData As SqlDataAdapter
    Try
      adData = New SqlDataAdapter(strSQL, myCon)
      adData.Fill(myDT)
      strError = "OK"
    Catch ex As Exception
      strError = ex.Message
    Finally

    End Try
    Return strError
End Function
```

Sau đó, bạn có thể khai báo để khởi tạo đối tượng clsDatabase và gọi phương thức GetValue trong phương thức Main như sau:

```
Module Module1
  Sub Main()
    Console.WriteLine("View Data")
    'Khai báo và khởi tạo đối tượng clsDatabase
    Dim cls As New clsDatabase
    'Khai báo và khởi tạo đối tượng DataTable
```

```
        Dim myDT As New DataTable
        ' Nếu kết nối cơ sở dữ liệu
        If cls.OpenConnection() = "OK" Then
            ' Gọi phương thức GetValue
            cls.GetValue(myDT, _
            "Select top 10 CustomerID," & _
            "CompanyName,Address from Customers")
            Console.WriteLine("-----                    ")
            ' Duyệt trên từng hàng dữ liệu
            For i As Integer = 0 To _
            myDT.Rows.Count - 1
                ' Duyệt trên từng cột dữ liệu
                For j As Integer = 0 To _
                myDT.Columns.Count - 1
                    Console.Write("{0} ", _
                    myDT.Rows(i).Item(j))
                Next
                Console.WriteLine()
            Next
            Console.WriteLine("--------------------");
            cls.CloseConnection()
        End If
        Console.ReadLine()
    End Sub
End Module
```

Khi thực thi chương trình, lập tức danh sách các mẩu tin trình bày tương tự như hình 26-1



Hình 26-1: *Danh sách mẩu tin*

2. Thiết kế *Form*, cho phép người sử dụng chọn tên bảng dữ liệu trên điều khiển *ComboBox* và trình bày mẩu tin của bảng được chọn trên điều khiển *DataGrid*.

Tương tự như bài tập 1 vừa trình bày ở trên, bạn khai báo phương thức GetValue nhận tham biến là đối tượng DataTable và tham trị là phát biểu SQL.

```
Public Function GetValue( _
  ByRef myDT As DataTable, _
  ByVal strSQL As String) As String

    Dim adData As SqlDataAdapter
    Try
      adData = New SqlDataAdapter(strSQL, myCon)
      adData.Fill(myDT)
      strError = "OK"
    Catch ex As Exception
      strError = ex.Message
    Finally

    End Try
    Return strError
End Function
```

Kế đến, khai báo để gọi phương thức GetValue của lớp clsDatabase với đối tượng DataTable rồi điền danh sách tên bảng dữ liệu vào điều khiển ComboBox trong biến cố Load như sau:

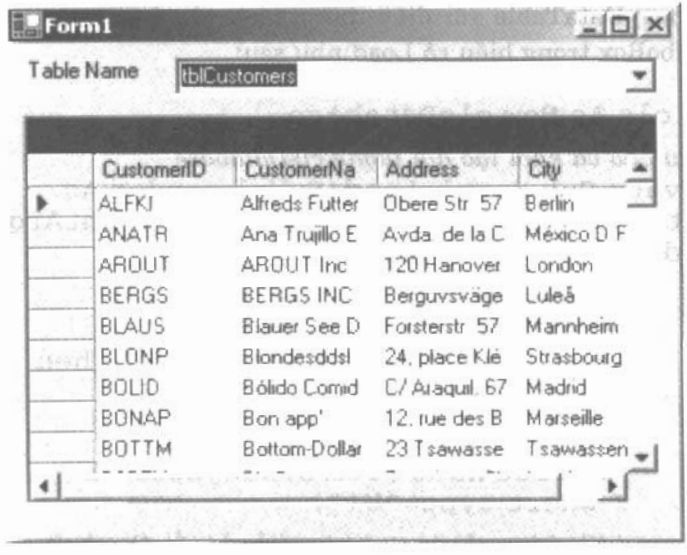
```
Dim cls As New clsDatabase
' Khai báo và khởi tạo đối tượng clsDatabase
Private Sub Form1_Load(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles MyBase.Load
    ' Khai báo và khởi tạo đối tượng DataTable
    Dim myDT As New DataTable
    If cls.OpenConnection() = "OK" Then
      ' Gọi phương thức GetValue
      cls.GetValue(myDT, _
      "Select name from sysobjects" & _
      " where type='U'")
      ' Khai báo thuộc tính cho điều khiển ComboBox
      Me.cbTable.DataSource = myDT
```

```
            Me.cbTable.DisplayMember = "name"
            Me.cbTable.ValueMember = "name"
        End If
    End Sub
```

Mỗi khi người sử dụng chọn tên bảng dữ liệu trên điều khiển ComboBox, lập tức danh sách dữ liệu của bảng đó xuất hiện trên điều khiển DataGrid, bạn có thể khai báo trong biến cố SelectionChangeCommitted như sau:

```
Private Sub
cbTable_SelectionChangeCommitted(ByVal sender
As Object, ByVal e As System.EventArgs) Handles
cbTable.SelectionChangeCommitted
    ' Khai báo và khởi tạo đối tượng DataTable
    Dim myDT As New DataTable
    ' Gọi phương thức GetValue
    cls.GetValue(myDT, _
    "Select * from [" + cbTable.Text + "]")
    Me.dgData.DataSource = myDT
End Sub
```

Khi thực thi chương trình, lập tức danh sách bảng dữ liệu xuất hiện trên điều khiển ComboBox và dữ liệu của bảng đó xuất hiện như hình 26-2.



Hình 26-2: *Trình bày dữ liệu*

3. Bằng cách chọn 3 bảng dữ liệu *Customers, Orders* và *Order Details*, bạn có thể đọc dữ liệu của ba bảng này vào đối tượng *DataSet*, sau đó trình danh sách của từng khách hàng cùng với từng đơn đặt hàng ra màn hình *Command Prompt*.

Đối với trường hợp này, bạn sử dụng đối tượng DataSet thay vì dùng đối tượng DataTable trong phương thức GetValue như sau:

```
Public Function GetValue( _
  ByRef myDS As DataSet, _
  ByVal strSQL As String) As String
    Dim adData As SqlDataAdapter
    Try
      adData = New SqlDataAdapter(strSQL, myCon)
      adData.Fill(myDS)
      strError = "OK"
    Catch ex As Exception
      strError = ex.Message
    Finally

    End Try
    Return strError
End Function
```

Kế đến, khai báo để gọi phương thức GetValue của lớp clsDatabase trong phương thức Main như sau:

```
Module Module1
  Sub Main()
    Console.WriteLine("View Data")
' Khai báo và khởi tạo đối tượng clsDatabase
    Dim cls As New clsDatabase
' Khai báo và khởi tạo đối tượng DataSet
    Dim myDS As New DataSet
' Khai báo biến lưu trữ phát biểu SQL
    Dim strSQL As String
' Nếu kết nối cơ sở dữ liệu
```

```
        If cls.OpenConnection() = "OK" Then
            strSQL = "Select top 10 CustomerID," & _
            "CompanyName,Address from Customers;" & _
            "select top 5 * from Orders;" & _
            "select top 5* from [Order Details];"
' Gọi phương thức GetValue
            cls.GetValue(myDS, strSQL)
' Duyệt trên từng bảng dữ liệu đang có trong
' đối tượng DataSet
          For t As Integer = 0 To _
          myDS.Tables.Count - 1
              Console.WriteLine( _
    myDS.Tables(t).TableName)
          Console.WriteLine("--------------------")
' Duyệt trên từng hàng dữ liệu của bảng thứ t
          For i As Integer = 0 To _
          myDS.Tables(t).Rows.Count - 1
' Duyệt trên từng cột dữ liệu của bảng thứ t
            For j As Integer = 0 To _
            myDS.Tables(t).Columns.Count - 1
              Console.Write("{0} ", _
              myDS.Tables(t).Rows(i).Item(j))
            Next
            Console.WriteLine()
          Next
          Console.WriteLine("--- ----------")
        Next
          cls.CloseConnection()
        End If
        Console.ReadLine()
    End Sub
End Module
```

Khi thực thi chương trình, lập tức danh sách các mẩu tin của ba bảng tuần tự trình bày tương tự như hình 26-3

Hình 26-3: *Danh sách dữ liệu 3 bảng*

4. Sử dụng cơ sở dữ liệu *SQL Server* có tên *Northwind*, bạn thiết kế *Form* và dùng đối tượng *SqlDataAdapter* để điền dữ liệu của bảng *Categories*, nếu người sử dụng thay đổi, xóa hay thêm mẩu tin và nhấn nút *Update*, bạn cập nhật sự thay đổi đó trở lại cơ sở dữ liệu nguồn.

Để thực hiện ví dụ này, trước tiên bạn khai báo phương thức có tên GetValue trong lớp clsDatabase, nhận tham biến là đối tượng DataTable như sau:

```
Public Function GetValue( _
  ByRef myDT As DataTable, _
  ByVal strSQL As String) As String

    Dim adData As SqlDataAdapter
```

```
        Try
          adData = New SqlDataAdapter(strSQL, myCon)
          adData.Fill(myDT)
          strError = "OK"
        Catch ex As Exception
          strError = ex.Message
        Finally

        End Try
        Return strError
    End Function
```

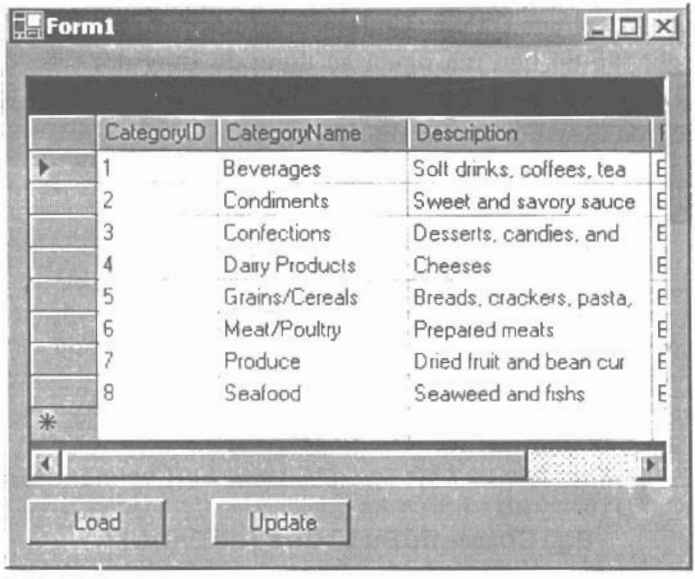
Kế đến, khai báo 3 biến sử dụng chung cho lớp Form1.

```
Dim cls As New clsDatabase
Dim myData As DataTable
Dim strSQL As String
```

Bằng cách khai báo để gọi phương thức GetValue, rồi điền dữ liệu lấy ra được vào điều khiển DataGrid trong biến cố Click của nút Load như sau:

```
Private Sub btnLoad_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles btnLoad.Click
    'Nếu kết nối cơ sở dữ liệu thành công
    If cls.OpenConnection() = "OK" Then
    'Khởi tạo đối tượng DataTable
      myData = New DataTable
      'Khai báo phát biểu SQL
      strSQL = "Select * from Categories"
      'Gọi phương thức GetValue
      cls.GetValue(myData, strSQL)
      'Điền dữ liệu vào điều khiển DataGrid
      Me.dgData.DataSource = myData
    End If
End Sub
```

Khi thực thi chương trình, nếu người sử dụng nhấn nút Load, lập tức danh sách mẩu tin của bảng Categories xuất hiện như hình 26-4.



Hình 26-4: *Danh sách mẩu tin*

Để cho phép người sử dụng cập nhật dữ liệu thay đổi trên điều khiển DataGrid, bạn khai báo trong biến cố Click của nút Update như sau:

```
Private Sub btnUpdate_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles btnUpdate.Click
    ' Khai báo và khởi tạo đối tượng DataTable
    Dim myNewData As New DataTable
    ' Lấy ra tập dữ liệu thay đổi
    myNewData = myData.GetChanges
    ' Nếu dữ liệu thay đổi
    If Not myNewData Is Nothing Then
        ' Gọi phương thức cập nhật dữ liệu thay đổi
        If cls.UpdateData(myNewData, _
        strSQL) = "OK" Then
            MsgBox("Updated")
            ' Nạp lại dữ liệu lên điều khiển DataGrid
            Call btnLoad_Click(sender, e)
        End If
    End If
End Sub
```

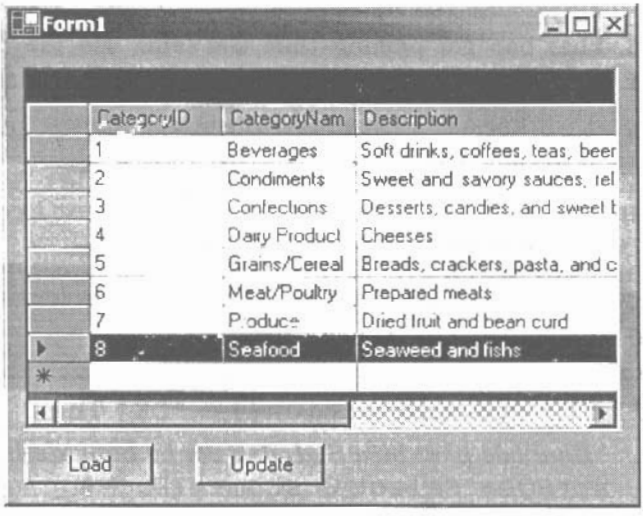
Trong đó, phương thức UpdateData nhận đối tượng DataTable đang nắm giữ tập dữ liệu mà người sử dụng đã thay đổi trên điều khiển DataGrid. Bằng cách sử dụng đối tượng SqlCommandBuilder, bạn có thể cập nhật dữ liệu thay đổi đó vào dữ liệu nguồn.

```
Public Function UpdateData( _
    ByVal myNewDT As DataTable, _
    ByVal strSQL As String) As String
    Dim myCom As SqlCommand
    Dim adData As SqlDataAdapter
    Try
        ' Khởi tạo đối tượng SqlDataAdapter
        adData = New SqlDataAdapter(strSQL, myCon)
        ' Khai báo và khởi tạo đối tượng SqlCommandBuilder
        Dim myBuilder As New _
            SqlCommandBuilder(adData)
        ' Sử dụng thuộc tính TableMappings
        adData.TableMappings.Add("Table", _
            myNewDT.TableName)
        ' Gọi phương thức Update
        adData.Update(myNewDT)
        strError = "OK"
    Catch ex As Exception
        strError = ex.Message
    Finally

    End Try
    Return strError
End Function
```

Chẳng hạn, người sử dụng thay đổi chuỗi "Seaweed and fish" thành "Seaweed and fishs" của mẩu tin thứ 8 và nhấn nút Update, dữ liệu sẽ cập nhật và thông báo như hình 26-4-1.

Hình 26-4-1: *Cập nhật thành công*

Bằng cách gọi lại phương thức btnLoad_Click, bạn có thể nạp lại dữ liệu đã thay đổi, khi đó mẩu tin thứ 8 trình bày như hình 26-4-2.



Hình 26-4-2: *Dữ liệu đã cập nhật*

2. ĐỐI TƯỢNG DATASET

1. Sử dụng đối tượng *DataSet* để đọc 3 bảng dữ liệu *Customers*, *Orders* và *Order Details* rồi điền lên điều khiển *DataGrid*, mỗi khi người sử dụng chọn một khách hàng thì chương trình có thể liệt kê danh sách các đơn đặt hàng cùng với tổng số tiền họ đã thanh toán.

Đối với trường hợp nhiều bảng dữ liệu có quan hệ, bạn sử dụng đối tượng DataSet thay vì dùng đối tượng DataTable trong phương thức GetValue như sau:

```
Public Function GetValue( _
  ByRef myDS As DataSet, _
  ByVal strSQL As String) As String

    Dim adData As SqlDataAdapter
    Try
      adData = New SqlDataAdapter(strSQL, myCon)
      adData.Fill(myDS)
      strError = "OK"
    Catch ex As Exception
      strError = ex.Message
    Finally
```

```
        End Try
        Return strError
    End Function
```

Kế đến, khai báo gọi phương thức GetValue của lớp clsDatabase trong biến cố Click của nút Load để điền dữ liệu vào điều khiển DataGrid như sau:

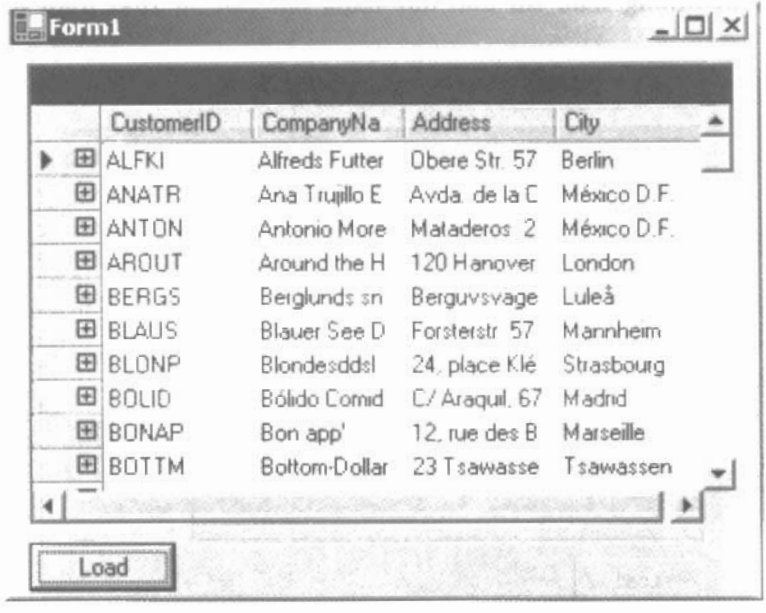
```
Private Sub btnLoad_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles btnLoad.Click
    Dim strSQL As String
    Dim myDS As New DataSet
    Dim cls As New clsDatabase
    'Nếu kết nối cơ sở dữ liệu thành công
    If cls.OpenConnection() = "OK" Then
        'Khai báo phát biểu SQL ứng với ba bảng dữ liệu
        strSQL="Select CustomerID, " & _
        " CompanyName, Address,City" & _
        " from Customers;" & _
        "select * from Orders;" & _
        "select * from [Order Details];"
        'Gọi phương thức GetValue
        cls.GetValue(myDS, strSQL)
        'Tạo quan hệ giữa ba bảng dữ liệu
        makeRelation(myDS)
        'Điền dữ liệu bảng thứ nhất vào điều khiển DataGrid
        Me.DataGrid1.DataSource = myDS.Tables(0)
        cls.CloseConnection()
    End If
End Sub
```

Trong đó, phương thức tạo quan hệ sử dụng đối tượng DataColumn và DataRelation được khai báo như sau:

```
Sub makeRelation(ByRef myDS As DataSet)
    'Khai báo cột làm khóa chính
    Dim PK As DataColumn
    'Khai báo cột làm khóa phụ
    Dim FK As DataColumn
    'Khóa chính của bảng thứ nhất
```

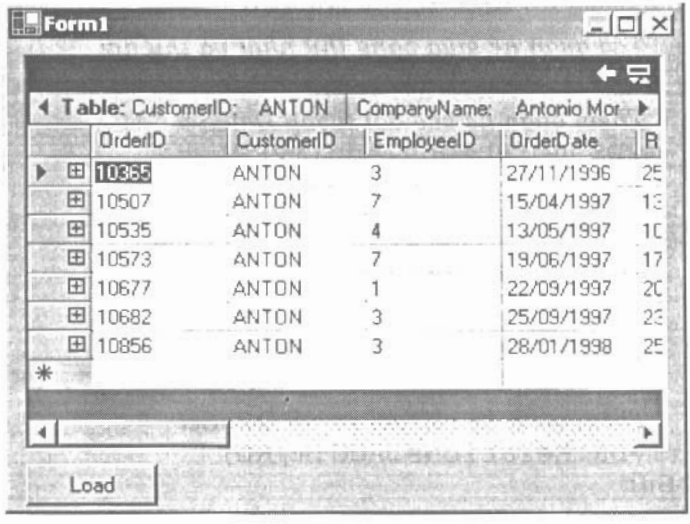
```
        PK = myDS.Tables(0).Columns(0)
        ' Khóa phụ của bảng thứ hai
        FK = myDS.Tables(1).Columns(1)
        Dim myRD As DataRelation
        ' Tạo quan hệ giữa bảng thứ nhất và thứ hai
        myRD = New DataRelation("Orders", PK, FK)
        ' Thêm quan hệ vào đối tượng DataSet
        myDS.Relations.Add(myRD)
        ' Khóa chính của bảng thứ hai
        PK = myDS.Tables(1).Columns(0)
        ' Khóa phụ của bảng thứ ba
        FK = myDS.Tables(2).Columns(0)
        ' Tạo quan hệ giữa bảng thứ hai và thứ ba
        myRD = New DataRelation("Details", PK, FK)
        ' Thêm quan hệ vào đối tượng DataSet
        myDS.Relations.Add(myRD)
    End Sub
```

Khi thực thi chương trình, nếu người sử dụng nhấn nút Load, lập tức danh sách khách hàng sẽ xuất hiện như hình 26-5.



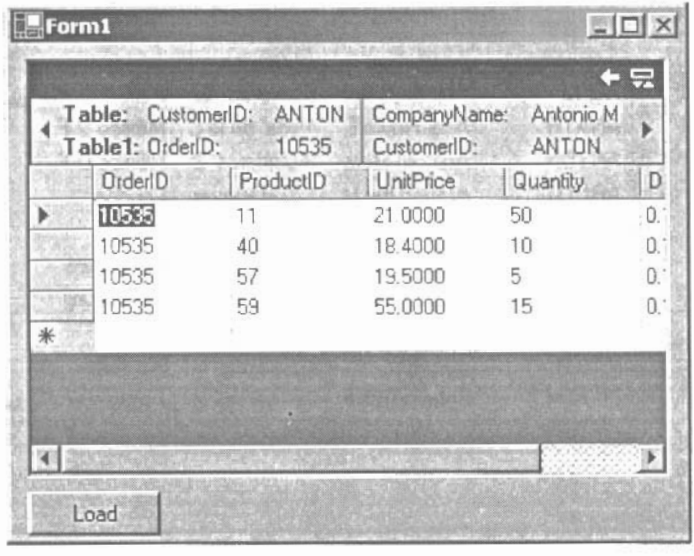
Hình 26-5: *Danh sách khách hàng*

Nếu người sử dụng nhấn vào dấu +, lập tức xuất hiện tên Orders, nếu muốn liệt kê những mẩu tin trong bảng Orders thuộc khách hàng đó thì nhấn vào liên kết Orders, dữ liệu trình bày như hình 26-5-1.



Hình 26-5-1: *Danh sách đơn đặt hàng*

Tương tự như vậy, bạn có thể chọn dấu + trên mỗi đơn đặt hàng, danh sách những mẩu tin đơn đặt hàng chi tiết sẽ xuất hiện như hình 26-5-2.



Hình 26-5-2: *Đơn đặt hàng chi tiết*

2. Thiết kế *Form*, sử dụng điều khiển *OleDbDataAdapter* để trình bày danh sách *Employees* thuộc cơ sở dữ liệu *Northwind.mdb* trên điều khiển *DataGrid*. Sau khi người sử dụng thay đổi hay thêm mới mẩu tin, bạn cho phép người sử dụng cập nhật những thay đổi đó trở lại dữ liệu nguồn.

Để thực hiện điều này, trước tiên bạn khai báo Class và đặt tên clsDatabase, khai báo sử dụng không gian tên OleDb bằng lệnh Imports và biến myCon như sau:

```
Imports System.Data.OleDb
Public Class clsDatabase
  Private psCon As String

  Private strError As String = ""
  Dim myCon As OleDbConnection
```

Kế đến, bạn khai báo phương thức kết nối cơ sở dữ liệu tương tự như các ví dụ trên. Tuy nhiên, chuỗi kết nối trong trường hợp này sử dụng cơ sở dữ liệu Access.

```
Public Function OpenConnection() As String
    ' Khai báo chuỗi kết nối cơ sở dữ liệu Access
    psCon =
"Provider=Microsoft.Jet.OLEDB.4.0;"
    psCon += "Data Source=D:\Books\Examples\"
    psCon += "VBNExamples\BTChapter18\"
    psCon += "\Bai2-2\" + gsDatabase
    Try
      ' Khởi tạo đối tượng OleDbConnection
      myCon = New OleDbConnection(psCon)
      ' Mở kết nối cơ sở dữ liệu Access
      myCon.Open()
      strError = "OK"
    Catch eq As OleDbException
      strError = eq.Message
    Catch ex As Exception
      strError = ex.Message
    End Try
    Return strError
End Function
```

Tương tự như vậy, bạn cần khai báo phương thức đóng kết nối cơ sở dữ liệu Access như sau:

```
Public Sub CloseConnection()
    If myCon.State <> ConnectionState.Closed
Then
      myCon.Close()
      myCon.Dispose()
    End If
End Sub
```

Để cho phép liệt kê danh sách trên điều khiển DataGrid, bạn khai báo phương thức GetValue nhận tham biến là đối tượng DataSet và tham trị là phát biểu SQL như sau:

```
Public Function GetValue( _
  ByRef myDS As DataSet, _
  ByVal strSQL As String) As String
    ' Khai báo đối tượng OleDbDataAdapter
    Dim adData As OleDbDataAdapter
    Try
      ' Khởi tạo đối tượng OleDbDataAdapter
      adData = New OleDbDataAdapter( _
      strSQL, myCon)
      ' Điền dữ liệu vào đối tượng DataTable
      adData.Fill(myDS, strSQL)
      strError = "OK"
    Catch ex As Exception
      strError = ex.Message
    Finally

    End Try
    Return strError
End Function
```

Trở lại lớp Form1, bạn khai báo để gọi phương thức GetValue của đối tượng clsDatabase trong biến cố Click của nút Load như sau:

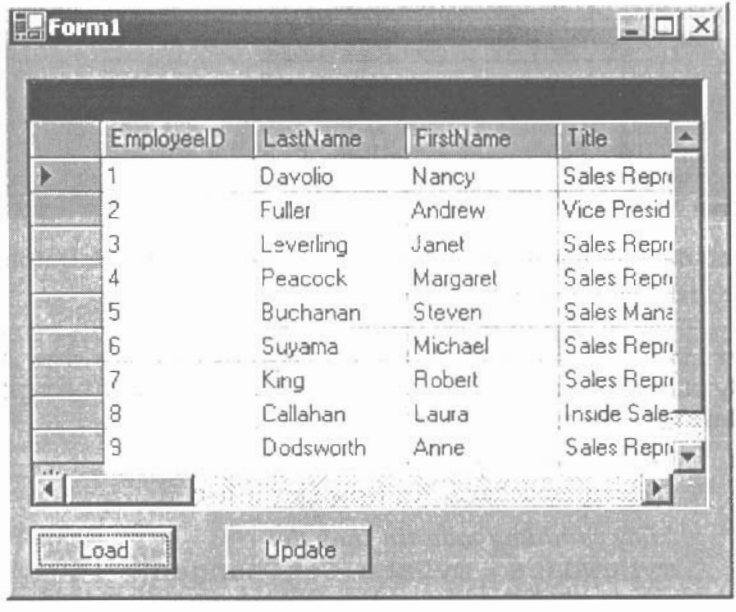
```
Private Sub btnLoad_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles btnLoad.Click
    ' Nếu kết nối cơ sở dữ liệu thành công
```

```
        If cls.OpenConnection() = "OK" Then
            ' Khởi tạo đối tượng DataSet
            myData = New DataSet
            ' Định nghĩa phát biểu SQL
            strSQL = "Select * from Employees"
            ' Gọi phương thức GetValue
            cls.GetValue(myData, strSQL)
            Me.dgData.DataSource = myData.Tables(0)
        End If
    End Sub
```

Lưu ý, bạn khai báo biến sử dụng chung trong lớp Form1, bao gồm đối tượng DataTable, clsDatabase và phát biểu SQL như sau:

```
Dim cls As New clsDatabase
Dim myData As DataSet
Dim strSQL As String
```

Khi thực thi chương trình, nếu người sử dụng nhấn nút Load, lập tức danh sách nhân viên trong bảng Employess của cơ sở dữ liệu Access xuất hiện như hình 26-6.



Hình 26-6: *Danh sách nhân viên*

Để cho phép người sử dụng cập nhật những dữ liệu đã thay đổi trên điều khiển DataGrid, bạn khai báo phương thức UpdateData trong lớp clsDatabase như sau:

```
Public Function UpdateData( _
    ByVal myNewDS As DataSet, _
    ByVal strSQL As String) As String
    ' Khai báo đối tượng OleDbDataAdapter
    Dim adData As OleDbDataAdapter
    Try
        ' Khởi tạo đối tượng OleDbDataAdapter
        adData = New OleDbDataAdapter(strSQL, _
            myCon)
        ' Khai báo và khởi tạo đối tượng OleDbCommandBuilder
        Dim myBuilder As New _
            OleDbCommandBuilder(adData)
        ' Khai báo ánh xạ tên bảng dữ liệu
        adData.TableMappings.Add("Table", _
            myNewDS.Tabels(0).TableName)
        ' Gọi phương thức Update để cập nhật dữ liệu
        adData.Update(myNewDS)
        strError = "OK"
    Catch ex As Exception
        strError = ex.Message
    Finally

    End Try
    Return strError
End Function
```

Sau đó, bạn gọi phương thức UpdateData của lớp clsDatabase trong biến cố Click của nút Update như sau:

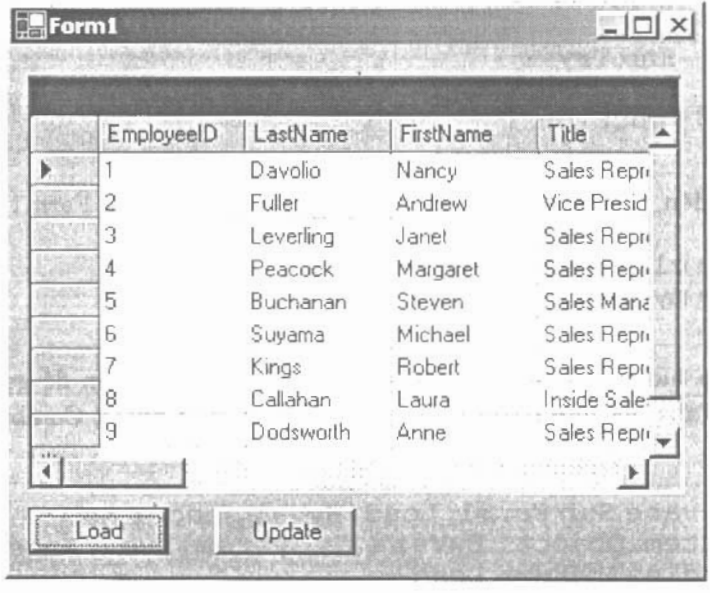
```
Private Sub btnUpdate_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles btnUpdate.Click
    ' Khai báo và khởi tạo đối tượng DataSet
    Dim myNewData As New DataSet
    ' Lấy ra những mẩu tin thay đổi
    myNewData = myData.GetChanges
    ' Nếu dữ liệu có thay đổi
```

```
        If Not myNewData Is Nothing Then
            'Gọi phương thức UpdateData
            If cls.UpdateData(myNewData, strSQL) = _
               "OK" Then
               MsgBox("Updated")
               Call btnLoad_Click(sender, e)
            End If
        End If
    End Sub
```

Giả sử sau khi thay đổi tên King thành Kings của nhân viên có mã thứ 7, nếu người sử dụng nhấn nút Update, dữ liệu đó sẽ được cập nhật vào dữ liệu nguồn, thông báo xuất hiện như hình 26-6-1.

Hình 26-6-1: *Cập nhật thành công*

Sau đó, dữ liệu trình bày với tên Kings như hình 26-6-2, bạn cũng có thể kiểm tra bằng Query Analyzer.



Hình 26-6-2: *Dữ liệu đã thay đổi*

3. Sử dụng đối tượng *DataSet* để đọc dữ liệu từ bảng *Customers*, sau đó điền danh sách *Country* vào điều khiển *ComboBox*, mỗi khi người sử dụng chọn *Country* thì bạn trình bày danh sách khách hàng của *Country* đó.

Đối với ví dụ này, chúng ta sử dụng đối tượng DataSet để lưu danh sách Country và danh sách khách hàng. Mỗi khi thay đổi Country từ điều khiển ComboBox thì lọc những khách hàng trong DataTable đã nạp để trình bày trên điều khiển DataGrid.

Để thực hiện ví dụ này, trước tiên bạn khai báo phương thức nhận tham biến là đối tượng DataSet như sau:

```
Public Function GetValue( _
  ByRef myDS As DataSet, _
  ByVal strSQL As String) As String
    Dim myCom As SqlCommand
    Dim adData As SqlDataAdapter
    Try
      adData = New SqlDataAdapter(strSQL, myCon)
      adData.Fill(myDS)
      strError = "OK"
    Catch ex As Exception
      strError = ex.Message
    Finally

    End Try
    Return strError
End Function
```

Kế đến, bạn khai báo biến sử dụng chung trong lớp Form1.

```
Dim cls As New clsDatabase
Dim myData As New DataSet
```

Tiếp tục, bạn khai báo trong biến cố Load của Form để gọi phương thức GetValue và nạp danh sách Country vào điều khiển ComboBox như sau:

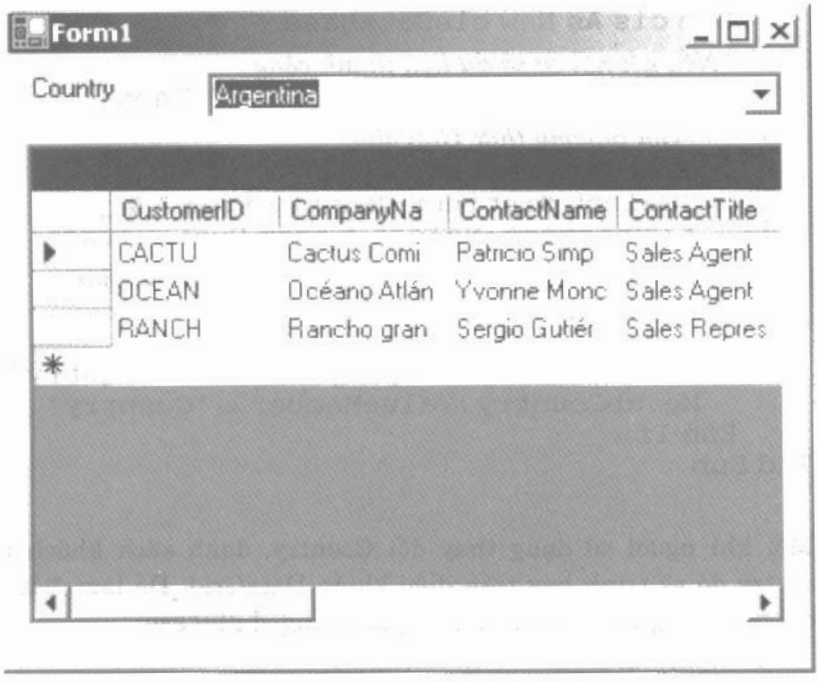
```
Private Sub Form1_Load(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles MyBase.Load
    ' Khai báo và khởi tạo đối tượng clsDatabase
```

```
    Dim cls As New clsDatabase
    'Nếu kết nối cơ sở dữ liệu thành công
    If cls.OpenConnection() = "OK" Then
      'Gọi phương thức GetValue
      cls.GetValue(myData, _
      "Select distinct Country from " & _
      " Customers;Select * from Customers")
    'Điền danh sách Country vào điều khiển ComboBox
      Me.cbCountry.DataSource = _
        myData.Tables(0)
      Me.cbCountry.DisplayMember = "Country"
      Me.cbCountry.ValueMember = "Country"
    End If
End Sub
```

Mỗi khi người sử dụng thay đổi Country, danh sách khách hàng của Country đó sẽ trình bày trên điều khiển DataGrid. Để làm điều này, bạn khai báo trong biến cố SelectedIndexChanged như sau:

```
Private Sub
cbCountry_SelectedIndexChanged(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles cbCountry.SelectedIndexChanged
    'Khai báo đối tượng DataView
    Dim myView As New
DataView(myData.Tables(1))
    'Khai báo lọc mẩu tin theo Country
    myView.RowFilter = "Country='" + _
      cbCountry.Text + "'"
    'Điền dữ liệu vào điều khiển DataGrid từ đối tượng DataView
    Me.dgData.DataSource = myView
End Sub
```

Khi thực thi chương trình, danh sách khách hàng của Country mặc định trình bày như hình 26-7.



Hình 26-7: *Danh sách khách hàng*

4 Tạo ứng dụng *Windows Forms*, cho phép người sử dụng chọn bảng dữ liệu, nhập số mẩu tin bắt đầu và số mẩu tin cần trình bày, sau đó chương trình sẽ liệt kê số mẩu tin đó trên điều khiển *DataGrid*.

Để thực hiện ví dụ này, trước tiên bạn khai báo phương thức GetValue trong lớp clsDatabase, nhận 4 tham số tuần tự là tham biến dạng đối tượng DataSet, tham trị thứ nhất là tên bảng dữ liệu, hai tham trị kế tiếp là số mẩu tin bắt đầu và số mẩu tin cần lấy ra.

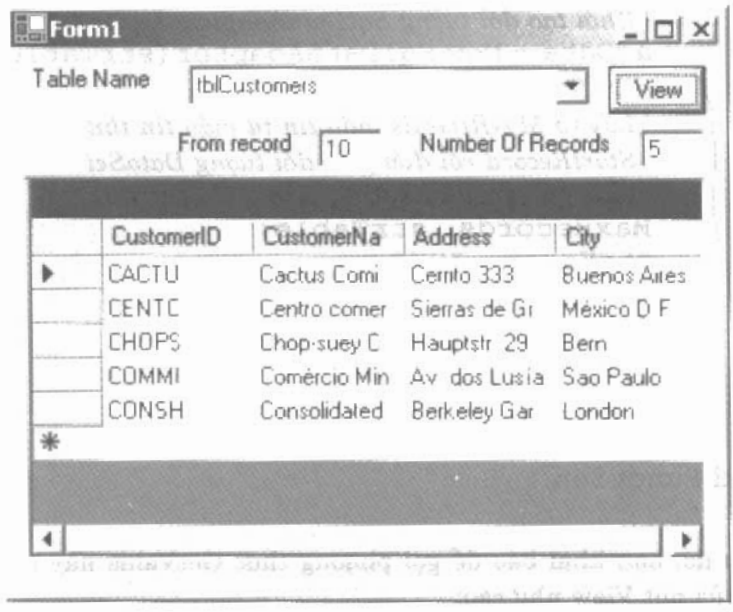
```
Public Function GetValue( _
    ByRef myDS As DataSet, _
    ByVal strTable As String, _
    ByVal StartRecord As Integer, _
    ByVal MaxRecords As Integer) As String
    Dim strError As String = ""
    ' Khai báo đối tượng SqlDataAdapter
    Dim adData As SqlDataAdapter
    ' Khai báo phát biểu SQL
    strTable = "select * from " + strTable
    Try
```

```
        ' Khởi tạo đối tượng SqlDataAdapter
        adData = New SqlDataAdapter(strTable, _
         myCon)
        ' Lấy ra MaxRecords mẩu tin từ mẩu tin thứ
        ' StartRecord rồi điền vào đối tượng DataSet
        adData.Fill(myDS, StartRecord, _
        MaxRecords, strTable)
        strError = "OK"
      Catch ex As Exception
        strError = ex.Message
      Finally
        adData.Dispose()
      End Try
      Return strError
  End Function
```

Sau đó, bạn khai báo để gọi phương thức GetValue này trong biến cố Click của nút View như sau:

```
Private Sub btnView_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles btnView.Click
    ' Khởi tạo đối tượng DataSet
    myData = New DataSet
    ' Nếu người sử dụng chọn tên bảng dữ liệu
    If Not cbTable.SelectedValue Is Nothing Then
      ' Gọi phương thức GetValue với 4 tham số
      cls.GetValue(myData, _
      "[" + cbTable.Text + "]", _
        txtFrom.Text, txtTo.Text)
    End If
    Me.dgData.DataSource = myData.Tables(0)
End Sub
```

Khi thực thi chương trình, bằng cách chọn tên bảng dữ liệu từ điều khiển ComboBox và nhập hai số vào phần From record (mẩu tin bắt đầu lấy) và Number of records (số mẩu tin sẽ lấy ra) , rồi nhấn nút View, kết quả trình bày như hình 26-8.



Hình 26-8: *Lấy ra một số mẩu tin*

Lưu ý, bạn cần khai báo hai biến sử dụng chung trong lớp Form1 như sau:

```
Dim myData As DataSet
Dim cls As New clsDatabase
```

Ngoài ra, để liệt kê danh sách tên bảng dữ liệu của cơ sở dữ liệu Northwind trên điều khiển ComboBox, bạn gọi phương thức GetValue với hai tham số trong biến cố Load của Form như sau:

```
Private Sub Form1_Load(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles MyBase.Load
    'Nếu kết nối cơ sở dữ liệu thành công
    If cls.OpenConnection() = "OK" Then
        'Khởi tạo đối tượng DataSet
        myData = New DataSet
        'Gọi phương thức GetValue
        cls.GetValue(myData, _
        "Select name from " & _
        " sysobjects where type='U'")
        Me.cbTable.DataSource = myData.Tables(0)
```

```
            Me.cbTable.DisplayMember = "name"
            Me.cbTable.ValueMember = "name"
        End If
    End Sub
```

3. CẬP NHẬT DỮ LIỆU

1. Thiết kế *Form*, dùng để điền dữ liệu của bảng nào đó trong cơ sở dữ liệu *Northwind*, sau đó cho phép người sử dụng thêm mới, thay đổi hay xóa mẩu tin rồi cập nhật trở lại dữ liệu nguồn.

Để thực hiện ví dụ này, trước tiên bạn khai báo phương thức có tên GetValue trong lớp clsDatabase, nhận tham biến là đối tượng DataSet như sau:

```
Public Function GetValue( _
  ByRef myDS As DataSet, _
  ByVal strSQL As String) As String
    Dim adData As SqlDataAdapter
    Try
      adData = New SqlDataAdapter(strSQL, myCon)
      adData.Fill(myDS, strSQL)
      strError = "OK"
    Catch ex As Exception
      strError = ex.Message
    Finally

    End Try
    Return strError
End Function
```

Tiếp theo, bạn khai báo 3 biến sử dụng chung cho lớp Form1.

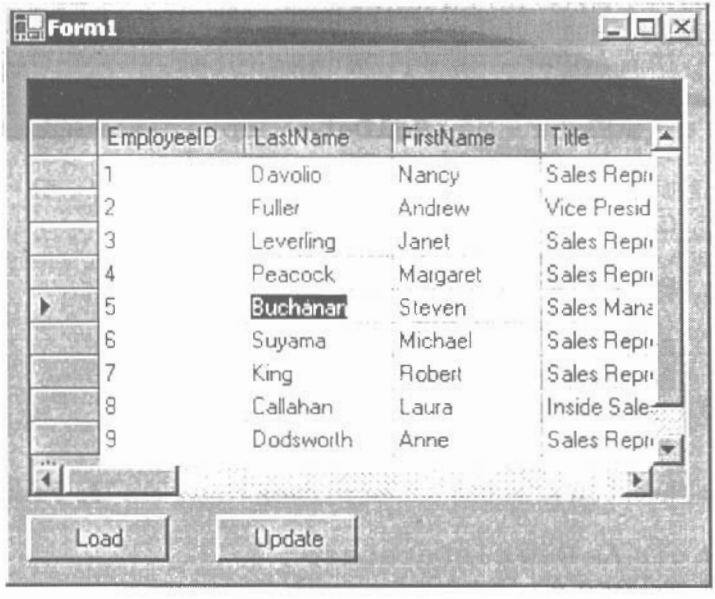
```
Dim cls As New clsDatabase
Dim myData As DataSet
Dim strSQL As String
```

Bằng cách khai báo để gọi phương thức GetValue, rồi điền dữ liệu lấy ra được vào điều khiển DataGrid trong biến cố Click của nút Load như sau:

```
Private Sub btnLoad_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles btnLoad.Click
```

```
        ' Nếu kết nối cơ sở dữ liệu thành công
        If cls.OpenConnection() = "OK" Then
        ' Khởi tạo đối tượng DataSet
          myData = New DataSet
          ' Khai báo phát biểu SQL
          strSQL = "Select * from Employees"
          ' Gọi phương thức GetValue
          cls.GetValue(myData, strSQL)
          ' Điền dữ liệu vào điều khiển DataGrid
          Me.dgData.DataSource = myData.Tables(0)
        End If
    End Sub
```

Khi thực thi chương trình, nếu người sử dụng nhấn nút Load, lập tức danh sách mẩu tin của bảng Employees xuất hiện như hình 26-9.



Hình 26-9: *Danh sách nhân viên*

Để cho phép người sử dụng cập nhật dữ liệu thay đổi trên điều khiển DataGrid, bạn khai báo trong biến cố Click của nút Update như sau:

```
Private Sub btnUpdate_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles btnUpdate.Click
```

```
        ' Khai báo và khởi tạo đối tượng DataSet
        Dim myNewData As New DataSet
        ' Lấy ra tập dữ liệu thay đổi
        myNewData = myData.GetChanges
        ' Nếu dữ liệu thay đổi
        If Not myNewData Is Nothing Then
            ' Gọi phương thức cập nhật dữ liệu thay đổi
            If cls.UpdateData(myNewData, _
            strSQL) = "OK" Then
                MsgBox("Updated")
                ' Nạp lại dữ liệu lên điều khiển DataGrid
                Call btnLoad_Click(sender, e)
                ' Chấp nhận sự thay đổi
                myData.AcceptChanges()
            End If
        End If
    End Sub
```

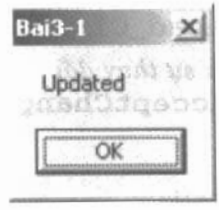
Lưu ý, bằng cách sử dụng đối tượng SqlCommandBuilder, bạn có thể cập nhật dữ liệu thay đổi trên điều khiển DataGrid vào dữ liệu nguồn.

Để làm điều này, bạn khai báo phương thức UpdateData nhận đối tượng DataSet đang nắm giữ tập dữ liệu mà người sử dụng đã thay đổi trên điều khiển DataGrid.

```
    Public Function UpdateData( _
        ByVal myNewDS As DataSet, _
        ByVal strSQL As String) As String

        Dim adData As SqlDataAdapter
        Try
            ' Khởi tạo đối tượng SqlDataAdapter
            adData = New SqlDataAdapter(strSQL, myCon)
            ' Khai báo và khởi tạo đối tượng SqlCommandBuilder
            Dim myBuilder As New _
                SqlCommandBuilder(adData)
            ' Sử dụng thuộc tính TableMappings
            adData.TableMappings.Add("Table", _
                myNewDS.Tables(0).TableName)
            ' Gọi phương thức Update
            adData.Update(myNewDS)
```

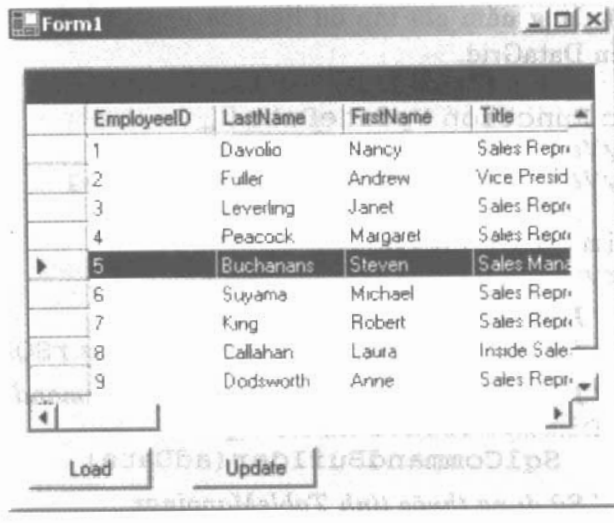
```
            strError = "OK"
        Catch ex As Exception
            strError = ex.Message
        Finally

        End Try
        Return strError
    End Function
```

Chẳng hạn, người sử dụng thay đổi chuỗi "**Buchanan**" trong cột Lastname thành "**Buchanans**" của mẩu tin thứ 5 và nhấn nút Update, dữ liệu sẽ cập nhật và thông báo như hình 26-9-1.



Hình 26-9-1: *Cập nhật thành công*

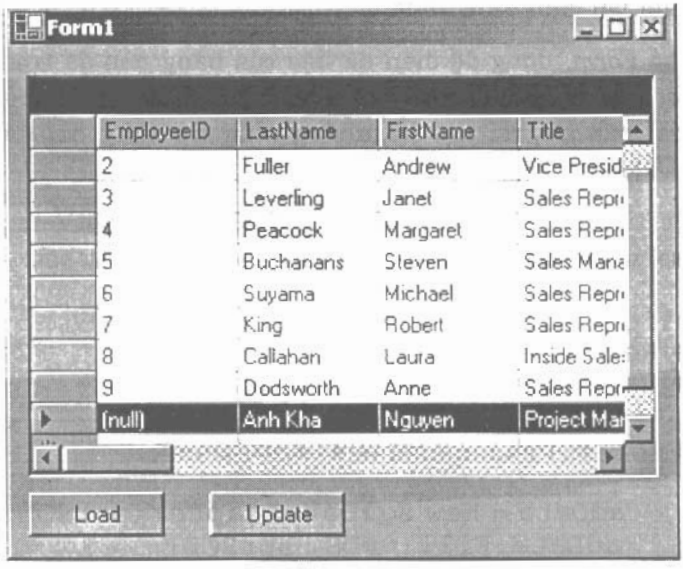
Bằng cách gọi lại phương thức btnLoad_Click, bạn có thể nạp lại dữ liệu đã thay đổi, khi đó mẩu tin thứ 5 trình bày như hình 26-9-2.



Hình 26-9-2: *Dữ liệu đã cập nhật*

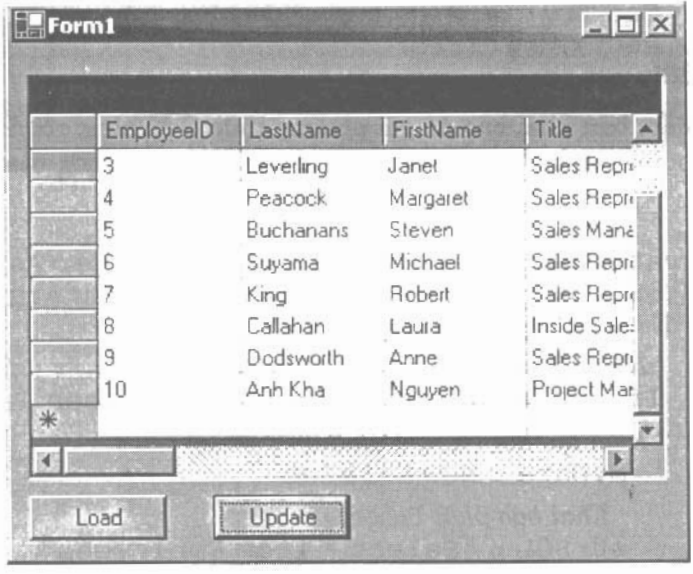
Giả sử, sau khi thay đổi tên của nhân viên thứ 5, bạn tiếp tục thêm mới nhân viên thứ 10 với các thông tin như hình 269-3.

Lưu ý, cột EmployeeID có kiểu số tự động, chính vì vậy bạn không cần nhập số vào cột này.



Hình 26-9-3: *Thêm mới mẩu tin*

Sau khi nhấn nút Update, dữ liệu sẽ được thêm vào bảng Employees, lập tức số 10 xuất hiện trong cột EmployeeID như hình 26-9-4.



Hình 26-9-4: *Dữ liệu đã được thêm vào*

Tương tự như vậy, bằng cách chọn vào mẩu tin cuối cùng (có mã là 10), nhấn phím Delete, mẩu tin này sẽ bị xóa khỏi cơ sở dữ liệu nguồn, kết quả trình bày như hình 26-9.

2. Thiết kế *Form*, dùng để điền dữ liệu của bảng nào đó trong cơ sở dữ liệu *Northwind*, sau đó cho phép người sử dụng thêm mới, thay đổi hay xóa mẩu tin rồi liệt kê những mẩu tin thay đổi đó trên điều khiển *DataGrid* khác.

Trước tiên bạn khai báo phương thức có tên GetValue trong lớp clsDatabase, nhận tham biến là đối tượng DataTable như sau:

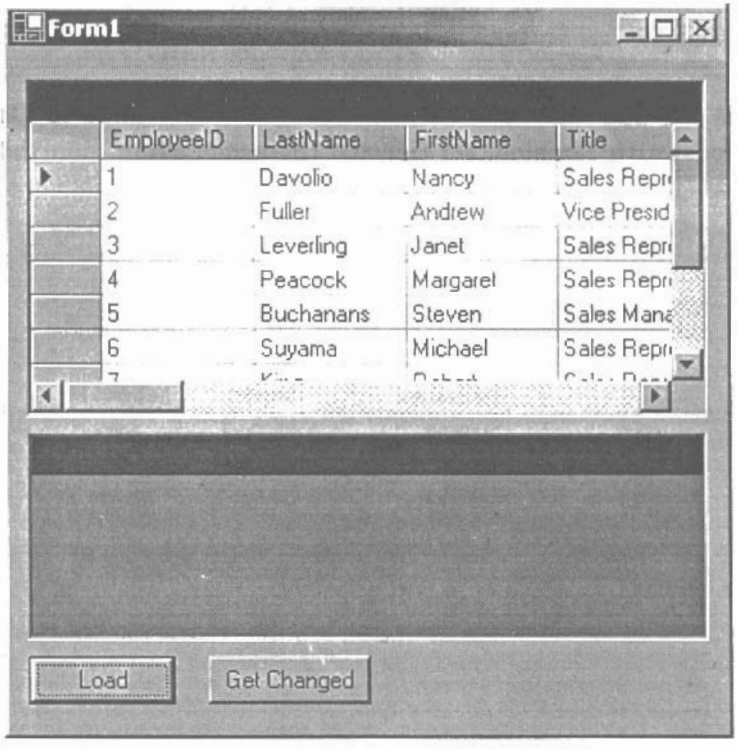
```
Public Function GetValue( _
  ByRef myDS As DataSet, _
  ByVal strSQL As String) As String

    Dim adData As SqlDataAdapter
    Try
      adData = New SqlDataAdapter(strSQL, myCon)
      adData.Fill(myDS, strSQL)
      strError = "OK"
    Catch ex As Exception
      strError = ex.Message
    Finally

    End Try
    Return strError
End Function
```

Kế đến, bạn khai báo để gọi phương thức GetValue, rồi điền dữ liệu lấy ra được vào điều khiển DataGrid trong biến cố Click của nút Load như sau:

```
Private Sub btnLoad_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles btnLoad.Click
    'Nếu kết nối cơ sở dữ liệu thành công
    If cls.OpenConnection() = "OK" Then
    'Khởi tạo đối tượng DataSet
      myData = New DataSet
      'Khai báo phát biểu SQL
      strSQL = "Select * from Employees"
      'Gọi phương thức GetValue
```

```
            cls.GetValue(myData, strSQL)
            ' Điền dữ liệu vào điều khiển DataGrid
            Me.dgData.DataSource = myData.Tables(0)
        End If
    End Sub
```

Khi thực thi chương trình, nếu người sử dụng nhấn nút Load, lập tức danh sách mẩu tin của bảng Employees xuất hiện như hình 26-10.



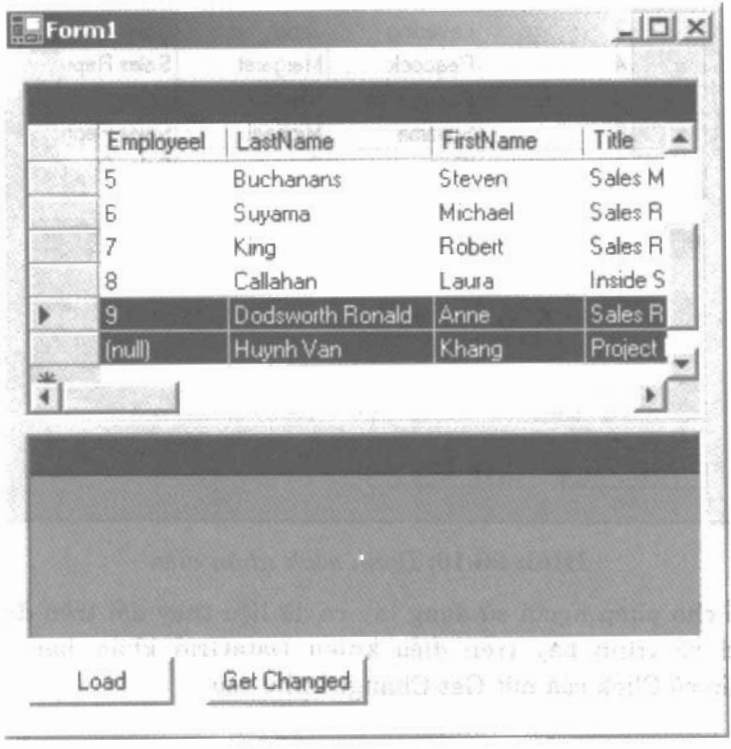
Hình 26-10: *Danh sách nhân viên*

Để cho phép người sử dụng lấy ra dữ liệu thay đổi trên điều khiển DataGrid và trình bày trên điều khiển DataGrid khác, bạn khai báo trong biến cố Click của nút Get Changed như sau:

```
Private Sub btnGetChange_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles btnGetChange.Click
    ' Khai báo và khởi tạo đối tượng DataSet
    Dim myNewData As New DataSet
```

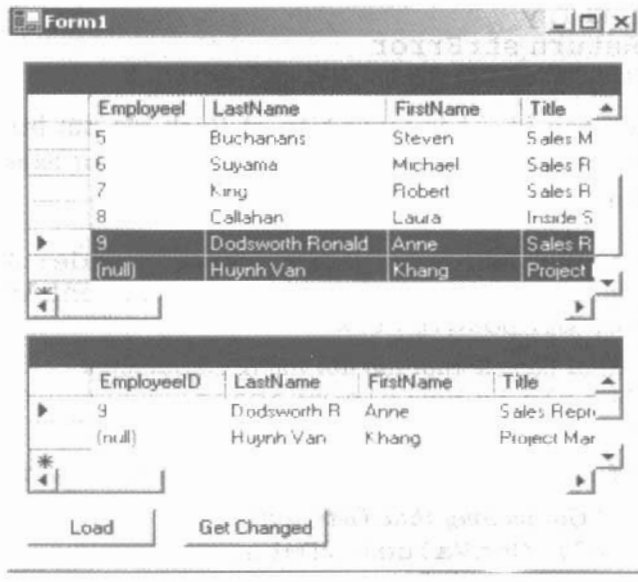
```
        ' Lấy ra tập dữ liệu thay đổi
        myNewData = myData.GetChanges
        ' Nếu dữ liệu thay đổi
        If Not myNewData Is Nothing Then
            ' Gán tập dữ liệu thay đổi vào điều khiển
            ' DataGrid thứ hai
            dgChange.DataSource= myNewData.Tables(0)
        End If
    End Sub
```

Giả sử, bạn thay đổi tên của nhân viên thứ nhất từ Dodsworth thành Dodsworth Ronaldo, rồi tiếp tục thêm mới nhân viên thứ 10 như hình 26-10-1.



Hình 26-10-1: *Đổi tên và thêm mới nhân viên*

Sau đó, bạn nhấn nút Get Changed, lập tức hai mẩu tin (thay đổi tên và mẩu tin mới), xuất hiện trên điều khiển DataGrid thứ hai như hình 26-10-2.



Hình 26-10-2: *Danh sách mẩu tin thay đổi*

4. ĐỐI TƯỢNG DATASET VÀ XML

1. Thiết kế *Form*, dùng để trình bày danh sách nhân viên trong bảng *Employees* từ cơ sở dữ liệu *Northwind* trên điều khiển *DataGrid*, khi người sử dụng nhấn nút *Write*, chương trình sẽ ghi tất cả mẩu tin có trong bảng *Employees* ra tập tin *XML*.

Để thực hiện ví dụ này, trước tiên bạn khai báo phương thức nhận tham biến là đối tượng DataSet như sau:

```
Public Function GetValue( _
  ByRef myDS As DataSet, _
  ByVal strSQL As String) As String
    Dim myCom As SqlCommand
    Dim adData As SqlDataAdapter
    Try
      adData = New SqlDataAdapter(strSQL, myCon)
      adData.Fill(myDS)
      strError = "OK"
    Catch ex As Exception
      strError = ex.Message
    Finally
```

```
        End Try
        Return strError
    End Function
```

Tiếp tục, bạn khai báo trong biến cố Click của nút btnLoad để gọi phương thức GetValue và nạp danh sách nhân viên từ bảng Employees vào điều khiển DataGrid như sau:

```
Private Sub btnLoad_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles btnLoad.Click
    'Khai báo và khởi tạo đối tượng clsDatabase
    Dim cls As New clsDatabase
    'Nếu kết nối cơ sở dữ liệu thành công
    If cls.OpenConnection() = "OK" Then
        'Gọi phương thức GetValue
        cls.GetValue(myData, _
        "Select * from EmployeeID " & _
        ",LastName,FirstName, " & _
        " Address from Employees")
    'Điền danh sách nhân viên vào điều khiển DataGrid
        Me. dgData.DataSource = _
            myData.Tables(0)
    End If
End Sub
```

Lưu ý, biến myData được khai báo biến sử dụng chung trong lớp Form1.

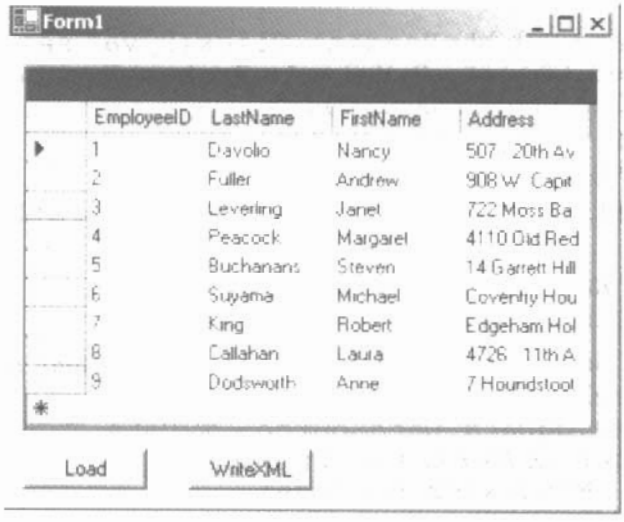
```
Dim myData As New DataSet
```

Mỗi khi người sử dụng nhấn nút WriteXML, để ghi danh sách nhân viên ra tập tin Xml, bạn có thể khai báo trong biến cố Click của nút btnWrite như sau:

```
Private Sub btnUpdate_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles btnWrite.Click
    Dim strPath As String
    'Định nghĩa đường dẫn tập tin
    strPath = Application.StartupPath + _
        "\Employees.xml"
```

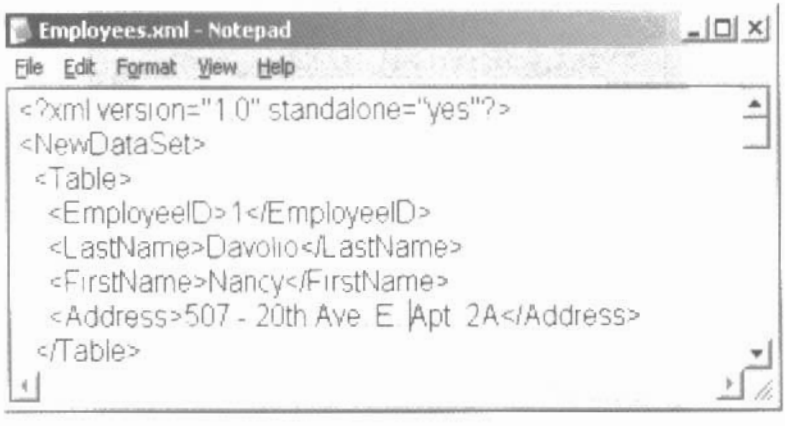
```
    ' Sử dụng phương thức WriteXml của đối tượng DataSet
    myData.WriteXml(strPath)
End Sub
```

Khi thực thi chương trình, danh sách nhân viên trong bảng Employees trình bày như hình 26-11.



Hình 26-11: *Danh sách nhân viên*

Nếu bạn nhấn nút WriteXml, lập tức xuất hiện tập tin Employees.xml trong thư mục chứa tập tin .exe với nội dung tương tự như hình 26-11-1.



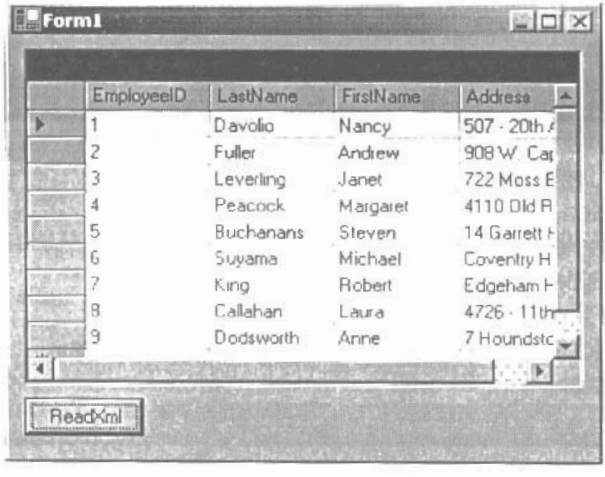
Hình 26-11-1: *Nội dung tập tin Xml*

2. Tạo ứng dụng *Windows Forms*, dùng để trình bày danh sách nhân viên trong tập tin *XML* vừa tạo ra trong câu 1.

Tiếp theo bài tập trên, bạn chép tập tin Employees.xml của bài tập trên vào thư mục *bin*, rồi khai báo đoạn chương trình trong biến cố Click của nút ReadXml như sau:

```
Private Sub btnReadXml_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles btnReadXml.Click
  Try
    ' Định nghĩa đường dẫn tập tin
    strPath = Application.StartupPath + _
      "\Employees.xml"
    ' Khai báo và khởi tạo đối tượng DataSet
    Dim myData As New DataSet
    ' Sử dụng phương thức ReadXml của đối tượng DataSet
    myData.ReadXml(strPath)
    Me.dgData.DataSource = myData.Tables(0)
  Catch ex As Exception
      MsgBox(ex.Message)
  End Try
End Sub
```

Khi thực thi chương trình, nếu người sử dụng nhấn nút ReadXml, lập tức danh sách dữ liệu trình bày trên điều khiển DataGrid như hình 26-12.



Hình 26-12: *Đọc dữ liệu từ tập tin Xml*

Chuyên đề 19:

ĐỐI TƯỢNG DATATABLE VÀ DATAVIEW

1. ĐỐI TƯỢNG DATATABLE

1. Khai báo phương thức, nhận một phát biểu *SQL* là câu lệnh *Select* sau đó trả về một đối tượng *DataTable*. Tạo ứng dụng *Form* để sử dụng đối tượng *DataTable* này.

Thêm Class vào Project và khai báo phương thức GetValue nhận phát biểu SQL và trả về đối tượng DataTable như sau:

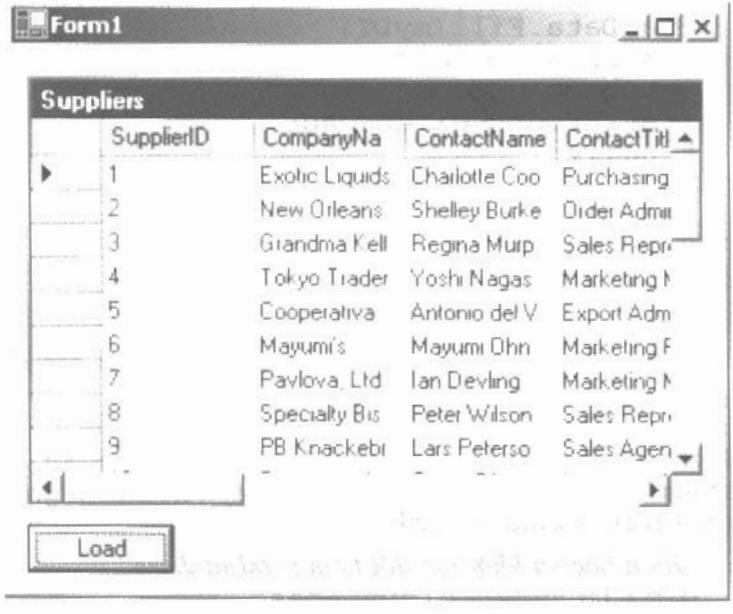
```
Public Function GetValue( _
  ByVal strSQL As String) As DataTable
    Dim myDT As New DataTable
    Dim adData As SqlDataAdapter
    Try
      adData = New SqlDataAdapter(strSQL, myCon)
      adData.Fill(myDT)
      strError = "OK"
    Catch ex As Exception
      strError = ex.Message
    Finally

    End Try
    Return myDT
End Function
```

Kế đến, bạn khai báo để gọi phương thức GetValue của đối tượng clsDatabase trong biến cố Click của nút btnLoad như sau:

```
Private Sub btnLoad_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles btnLoad.Click
    ' Khai báo và khởi tạo đối tượng clsDatabase
    Dim cls As New clsDatabase
    ' Nếu kết nối cơ sở dữ liệu thành công
    If cls.OpenConnection() = "OK" Then
```

```
        ' Khai báo và khởi tạo đối tượng DataTable
        Dim myData As DataTable
        myData = New DataTable
        ' Khai báo phát biểu SQL
        Dim strSQL As String
        strSQL = "Select * from Suppliers"
        ' Gọi phương thức GetValue
        myData = cls.GetValue(strSQL)
        ' Khai báo thuộc tính Caption của điều khiển DataGrid
        dgData.CaptionText = "Suppliers"
        ' Nếu có mẩu tin trong đối tượng DataTable
        If myData.Rows.Count > 0 Then
          Me.dgData.DataSource = myData
        End If
      End If
End Sub
```

Khi thực thi chương trình, nếu người sử dụng nhấn nút Load thì lập tức danh sách nhà cung cấp trong bảng Suppliers sẽ xuất hiện tương tự như hình 27-1.



Hình 27-1: *Danh sách nhà cung cấp*

2. Khai báo *Class*, bao gồm phương thức nhận phát biểu *SQL* dạng *Select* trả về mảng một chiều có kiểu *ArrayList*, sau đó thiết kế *Form* truyền phát biểu *Select* tương ứng và trình bày dữ liệu trên điều khiển *ListView* từ phương thức trên.

Để làm bài tập này, trước tiên bạn khai báo lớp clsItem, bao gồm thuộc tính Values và Constructor như sau:

```
Public Class clsItem
  ' Khai báo mảng đối tượng
  Private mValues() As Object
  ' Khai báo Constructor
  Public Sub New(ByVal strValues() As Object)
    mValues = strValues
  End Sub
  ' Định nghĩa thuộc tính GET và SET
  Property Values() As Object()
    Get
      Return mValues
    End Get
    Set(ByVal Value() As Object)
      mValues = Value
    End Set
  End Property
End Class
```

Kế đến, bạn thêm Class vào Project và đặt tên clsDatabase, bằng cách khai báo phương thức GetValue nhận tham trị là phát biểu SQL và trả về đối tượng ArrayList tương tự như các ví dụ trước.

Tuy nhiên, trong trường hợp này chúng ta sử dụng thuộc tính ItemArray của đối tượng DataRow thay vì sử dụng phương thức GetValue của đối tượng SqlDataReader.

```
Public Function GetValue(ByVal strSQL As String)
As ArrayList
    Dim arrList As New ArrayList
    Dim myDT As New DataTable
    Try
      ' Khai báo điền dữ liệu vào đối tượng DataTable
      Dim myData As New SqlDataAdapter( _
        strSQL, myCon)
      myData.Fill(myDT)
```

```
            'Khai báo đối tượng clsItem
            Dim Item As clsItem
            'Biến tính số cột dữ liệu
            Dim col As Integer = myDT.Columns.Count - 1
            'Lấy tên cột dữ liệu đưa vào mảng
            Dim arrValue(col) As Object
            'Duyệt trên từng cột dữ liệu
            For j As Integer = 0 To col
              arrValue(j) = _
                myDT.Columns(j).ColumnName
            Next
            'Khởi tạo đối tượng clsItem
            Item = New clsItem(arrValue)
            'Thêm đối tượng clsItem vào ArrayList
            arrList.Add(Item)
            'Duyệt trên từng hàng
            For i As Integer = 0 To myDT.Rows.Count - 1
              'Sử dụng thuộc tính ItemArray để lấy ra mảng dữ
              'liệu của hàng thứ i
              arrValue = myDT.Rows(i).ItemArray()
            'Khởi tạo đối tượng clsItem
              Item = New clsItem(arrValue)
            'Thêm đối tượng clsItem vào ArrayList
              arrList.Add(Item)
            Next
            strError = "OK"
          Catch ex As Exception
            strError = ex.Message
          Finally

          End Try
        Return arrList
```

Sau đó, bạn khai báo để gọi phương thức GetValue trong biến cố Click của nút Load như sau:

```
Private Sub btnLoad_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles btnLoad.Click
    Dim cls As New clsDatabase
    Dim list As ArrayList
```

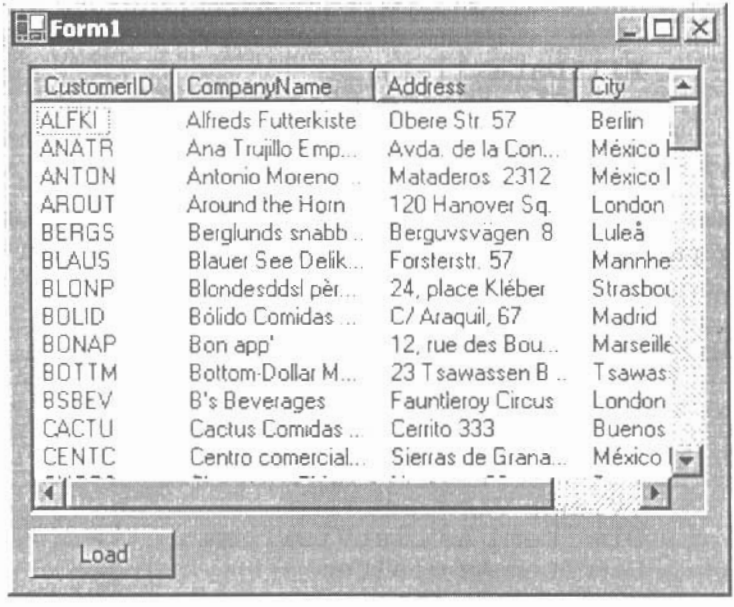
```
        'Nếu kết nối cơ sở dữ liệu thành công
        If cls.OpenConnection() = "OK" Then
            'Gọi phương thức GetValue
            list = cls.GetValue( _
            "Select CustomerID, CompanyName" & _
            " ,Address,City from Customers")
            'Điền dữ liệu vào điều khiển ListView
            FillData(list)
            cls.CloseConnection()
        End If
    End Sub
```

Trong đó, phương thức FillData nhận đối tượng ArrayList, duyệt qua từng phần tử, khởi tạo đối tượng ListViewItem và điền vào điều khiển Listview.

```
    Sub FillData(ByVal arrList As ArrayList)
        lvData.Clear()
        With lvData
            Dim i As Integer
            .Items.Clear():.CheckBoxes = False
            .View = View.Details
            Dim item1 As ListViewItem
            Dim Item As clsItem
            'Duyệt qua từng phần tử của đối tượng ArrayList
            For Each obj As clsItem In arrList
                If i = 0 Then
                    'Khai báo Header
                    For i = 0 To obj.Values().Length - 1
                        .Columns.Add( _
                    obj.Values(i).ToString, 100, 0)
                    Next
                Else
                    'Trình bày dữ liệu
                    item1 = New ListViewItem( _
                    obj.Values(0).ToString)
                    'Duyệt qua từng cột dữ liệu
                    For i = 1 To obj.Values().Length - 1
                        item1.SubItems.Add(obj.Values(i))
                    Next
                    .Items.Add(item1)
                End If
            Next
```

```
        End With
    End Sub
```

Khi thực thi chương trình, nếu người sử dụng nhấn nút Load, lập tức danh sách khách hàng trình bày trên điều khiển ListView như hình 27-2.



Hình 27-2: *Trình bày dữ liệu trên điều khiển ListView*

3. Thiết kế *Form*, với các *TextBox* cho phép người sử dụng nhập mã, tên, địa chỉ, và điện thoại của nhân viên, sau đó trình bày chúng trên *DataGrid* của cùng *Form*.

Để thực hiện ví dụ này, trước tiên bạn khai báo biến đối tượng DataTable dùng chung trong lớp Form1.

```
Dim myDT As DataTable
```

Kế đến, khởi tạo đối tượng DataTable và khai báo cột dữ liệu trong biến cố Load của Form như sau:

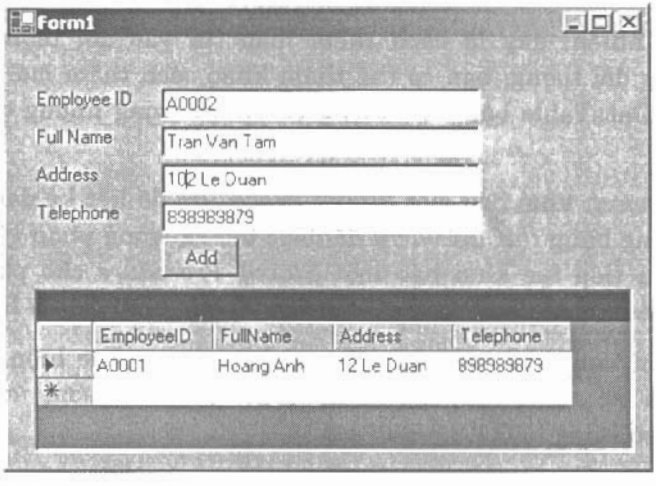
```
Private Sub Form1_Load(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles MyBase.Load
    myDT = New DataTable
    myDT.Columns.Add("EmployeeID")
```

```
        myDT.Columns.Add("FullName")
        myDT.Columns.Add("Address")
        myDT.Columns.Add("Telephone")
    End Sub
```

Sau khi nhập thông tin vào các điều khiển TextBox, nếu người sử dụng nhấn nút Add, bạn khai báo mảng đối tượng để lưu trữ các giá trị đó, rồi thêm chúng vào đối tượng DataTable bằng phương thức Add của Rows Collection như sau:

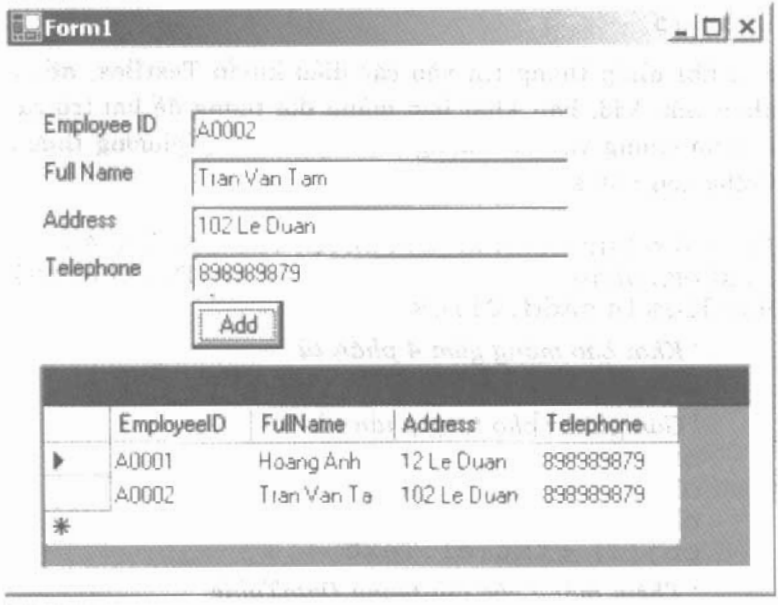
```
Private Sub btnAdd_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles btnAdd.Click
    ' Khai báo mảng gồm 4 phần tử
    Dim obj(3) As String
    ' Gán giá trị cho từng phần tử
    obj(0) = txtID.Text
    obj(1) = txtFullName.Text
    obj(2) = txtAddress.Text
    obj(3) = txtTel.Text
    ' Thêm mảng vào đối tượng DataTable
    myDT.Rows.Add(obj)
    Me.DataGrid1.DataSource = myDT
End Sub
```

Khi thực thi chương trình, bằng cách nhập một số thông tin như hình 27-3.



Hình 27-3: *Nhập mới mẩu tin*

Nếu nhấn nút Add, lập tức thông tin trên các điều khiển TextBox sẽ được thêm vào điều khiển DataGrid như hình 27-3-1



Hình 27-3-1: *Thêm mẩu tin*

Lưu ý, bạn cần khai báo để xóa thông tin trên các điều khiển TextBox sau khi thêm vào đối tượng DataTable thành công. Ngoài ra, bạn cũng kiểm tra thông tin nhân viên đã tồn tại trong đối tượng DataTable hay chưa trước khi thêm vào.

Tuy nhiên, đây là cách thêm mẩu tin vào đối tượng DataTable bằng mảng đối tượng, bạn có thể tham khảo cách thêm mới mẩu tin vào đối tượng DataTable bằng đối tượng DataRow trong những chuyên đề kế tiếp.

4. Tạo *Project*, khai báo một *Stored Procedure* để liệt kê danh sách mẩu tin trong bảng *tblCustomers* đã được tạo ra trong phần lý thuyết. Sau đó, bạn tiếp tục khai báo một *Stored Procedure* cho phép người sử dụng thêm mới, xóa, thay đổi dữ liệu và cập nhật lại dữ liệu nguồn.

Trước tiên, bạn khai báo Stored Procedure thực hiện cả bốn chức năng thêm mới, truy vấn, cập nhật, và xóa mẩu tin tương ứng với giá trị của tham số @flag là 0,1,2,3 như sau:

```
CREATE PROC spViewCustomers
```

```
    @flag tinyint,
    @CustomerID varchar(10),
    @CustomerName varchar(50),
    @Address varchar(100),
    @City varchar(50)
AS
    -- Trường hợp truy vấn dữ liệu
    if @flag=0
    begin
        select * from tblCustomers
        where 1=1
        -- Nếu có cung cấp giá trị theo mã khách hàng
        and (case @CustomerID when '' then
            @CustomerID else CustomerID end)
            =@CustomerID
        -- Nếu có cung cấp giá trị theo tên khách hàng
        and (case @CustomerName when '' then
            @CustomerName else CustomerName end)
            like @CustomerName
        -- Nếu có cung cấp giá trị theo địa chỉ khách hàng
        and (case @Address when '' then @Address
            else Address end) like @Address
        -- Nếu có cung cấp giá trị theo thành phố khách hàng
        and (case @City when '' then @City
            else City end) =@City
    end
    else
    begin
     set nocount on
     declare @IsExit int
     set @IsExit =1
     -- Trường hợp thêm mới mẩu tin
     if @flag=1
        -- Nếu mã khách hàng chưa tồn tại
        if not exists(select * from tblCustomers
            where CustomerID=@CustomerID)
                begin
                    insert into tblCustomers
                    values(@CustomerID,@CustomerName,
                    @Address,@City)
                    set @IsExit =@@identity
                end
        else
```

```
        -- Trường hợp thêm mới mẩu tin nhưng mã khách
        -- hàng đã tồn tại
        set @IsExit =0
    if @flag=2
      -- Trường hợp cập nhật mẩu tin
        update tblCustomers
        set CustomerName=@CustomerName,
        Address=@Address,
        City=@City
        where CustomerID=@CustomerID
    if @flag=3
      -- Trường hợp xóa mới mẩu tin
        delete from tblCustomers
        where CustomerID=@CustomerID
    set nocount off
    select @IsExit
  end
```

Nếu sử dụng tiện ích Query Analyzer, bạn có thể gọi thủ tục để liệt kê danh sách mẩu tin tương tự như sau:

```
spViewCustomers 0,'','','',''
GO
spViewCustomers 0,'ANTON','','',''
GO
spViewCustomers 0,'','','','London'
GO
spViewCustomers 0,'','','%ki%',''
GO
spViewCustomers 0,'','%ki%','',''
```

Trong trường hợp bạn muốn xóa một khách hàng, gọi thủ tục trên như sau:

```
spViewCustomers 3,'ANTON','','',''
```

Đối với trường hợp thêm mẩu tin, bạn gọi thủ tục như sau:

```
spViewCustomers 2,'ANTON','NEW NAME','NEW
ADDRESS','NEW CITY'
```

Tương tự như vậy, trường hợp thêm mới mẩu tin, bạn gọi thủ tục với cú pháp:

```
spViewCustomers 1,'KHANG','NEW NAME','NEW
ADDRESS','NEW CITY'
```

Lưu ý, nếu cột CustomerName, Address, City dùng để lưu trữ kiểu chuỗi Unicode (sử dụng kiểu nvarchar), bạn có thể gọi thủ tục như sau:

```
spViewCustomers 1,'KHANG',
N'NEW NAME',N'NEW ADDRESS',N'NEW CITY'
```

Kế đến, bạn khai báo phương thức GetValue nhận tham biến là đối tượng DataTable, tham trị thứ nhất là tên thủ tục nội tại và hai tham trị còn lại là hai mảng chuỗi ứng với tham số và giá trị như sau:

```
Public Function GetValue( _
  ByRef myDT As DataTable, _
  ByVal strSQL As String, _
  ByVal arrPara() As String, _
  ByVal arrValue() As String) As String
    ' Khai báo và khởi tạo đối tượng SqlCommand
    Dim myCom As New SqlCommand(strSQL, myCon)
    ' Khai báo đối tượng SqlCommand
    Dim adData As SqlDataAdapter
    Try
      ' Chọn loại lệnh
      myCom.CommandType = _
      CommandType.StoredProcedure
      ' Gán tham số và giá trị
      For i As Integer = 0 To arrPara.Length - 1
        myCom.Parameters.Add(arrPara(i), _
        arrValue(i))
      Next
      ' Khởi tạo đối tượng SqlDataAdapter
      adData = New SqlDataAdapter(myCom)
      ' Điền dữ liệu vào đối tượng DataTable
      adData.Fill(myDT)
      strError = "OK"
    Catch ex As Exception
      strError = ex.Message
```

```
        Finally

        End Try
        Return strError
    End Function
```

Sau đó, khai báo trong biến cố Click của nút Load để gọi phương thức GetValue của đối tượng clsDatabase như sau:

```
Private Sub btnLoad_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles btnLoad.Click
    'Khai báo và khởi tạo đối tượng clsDatabase
    Dim cls As New clsDatabase
    'Nếu kết nối cơ sở dữ liệu thành công
    If cls.OpenConnection() = "OK" Then
      'Khai báo và khởi tạo đối tượng DataTable
      Dim myData As New DataTable
      'Gán giá trị cho mảng giá trị
      arrValue(0) = "0"
      arrValue(1) = txtID.Text
      arrValue(2) = "%" + txtName.Text + "%"
      arrValue(3) = "%" + txtAddress.Text + "%"
      arrValue(4) = txtCity.Text
      'Gọi phương thức GetValue
      cls.GetValue(myData, strSQL, _
        arrPara, arrValue)
      If myData.Rows.Count > 0 Then
        Me.dgData.DataSource = myData
      End If
    End If
End Sub
```

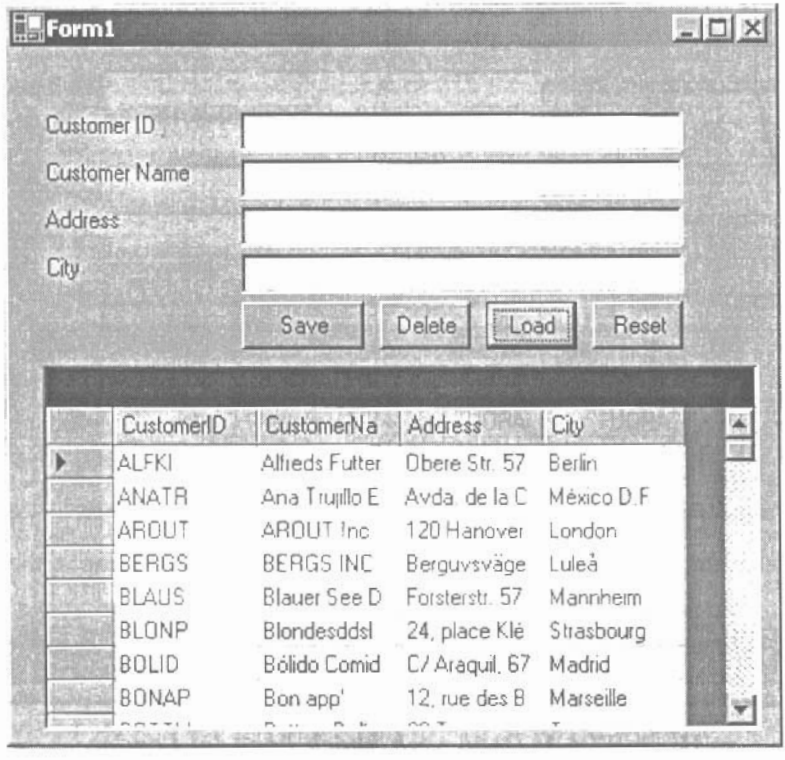
Trong đó, hai cột Address và CustomerName sử dụng phép toán like với ký hiệu %. Như vậy, đối với trường hợp này, chúng ta sẽ gán giá trị cho mảng này là:

```
arrValue(0) = "0"
arrValue(1) = txtID.Text
arrValue(2) = "%" + txtName.Text + "%"
arrValue(3) = "%" + txtAddress.Text + "%"
arrValue(4) = txtCity.Text
```

Lưu ý, phần tử thứ nhất có giá trị là 0 tương ứng với tùy chọn truy vấn dữ liệu.

Ngoài ra, chúng ta có thể thay đổi giá trị của phần tử đầu tiên tùy vào một trong 4 chức năng thêm, xóa, cập nhật hay truy vấn.

Khi thực thi chương trình, nếu bạn nhấn vào nút Load, lập tức danh sách mẫu tin trong bảng tblCustomers xuất hiện như hình 27-4.



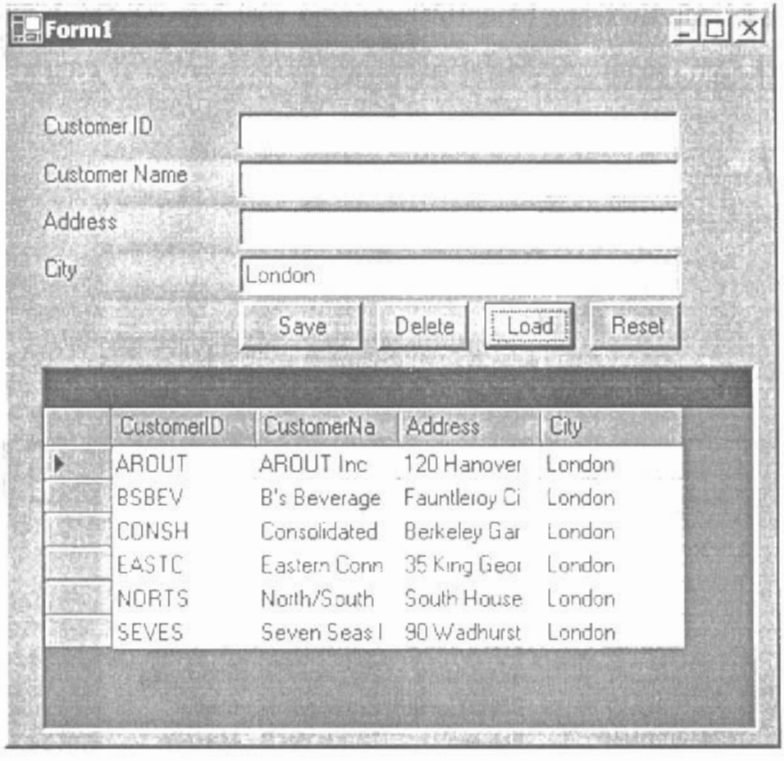
Hình 27-4: *Trình bày danh sách khách hàng*

Trong đó, mảng chứa đựng tham số, giá trị tương ứng, tên phương thức được định nghĩa dùng chung như sau:

```
' Khai báo mảng giá trị
Dim arrValue(4) As String
' Khai báo phát biểu SQL
Dim strSQL As String = "spViewCustomers "
' Khai báo mảng tham số
Dim arrPara() As String = _
```

```
{"@flag", "@CustomerID", _
"@CustomerName", _
"@Address", "@City"}
```

Bằng cách nhập London vào phần City và nhấn nút Load, bạn sẽ có danh sách khách hàng có cột City là London như hình 27-4-1.

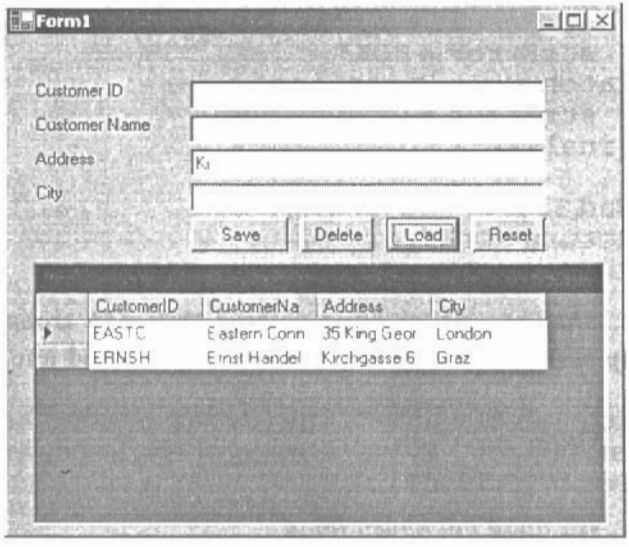


Hình 27-4-1: *Danh sách khách hàng tại London*

Lưu ý, đối với cột CustomerName và Address, thủ tục sử dụng phép toán Like với ký hiệu %.

```
and (case @CustomerName when '' then
@CustomerName else CustomerName end)
like @CustomerName
  and (case @Address when '' then
  @Address else Address end) like @Address
```

Tương tự như vậy, nếu nhập chuỗi "Ki" vào phần Address và nhấn nút Load thì chương trình sẽ liệt kê danh sách những khách hàng có giá trị trong cột Address có chứa chuỗi "Ki" như hình 27-4-2.



Hình 27-4-2: *Lọc theo Address*

Tiếp tục, bạn khai báo phương thức ExecuteSQL, nhận tham trị là tên phương thức, tham biến thứ hai là kết quả trả về của thủ tục, hai tham trị còn lại là hai mảng ứng với tham số và giá trị tương ứng.

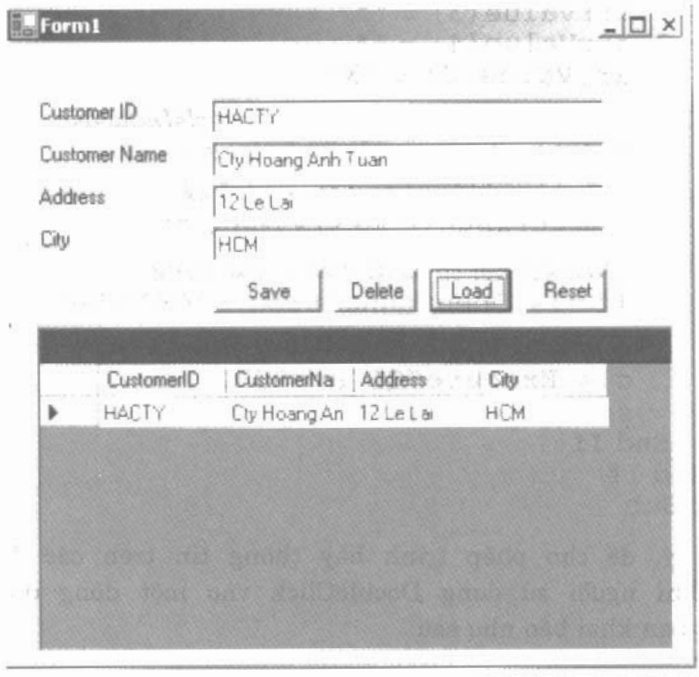
```
Public Function ExecuteSQL( _
    ByVal strSQL As String, _
    ByRef strResult As String, _
    ByVal arrPara() As String, _
    ByVal arrValue() As String) As String
    ' Khai báo đối tượng SqlCommand
    Dim myCom As SqlCommand
    Try
        ' Khởi tạo đối tượng SqlCommand
        myCom = New SqlCommand(strSQL, myCon)
        ' Chọn kiểu khai báo lệnh là thủ tục
        myCom.CommandType = _
          CommandType.StoredProcedure
        ' Khai báo tham số và giá trị
        For i As Integer = 0 To arrPara.Length - 1
          myCom.Parameters.Add(arrPara(i), _
            arrValue(i))
        Next
        ' Thực thi thủ tục
```

```
            strResult = myCom.ExecuteScalar()
            strError = "OK"
        Catch ex As Exception
            strError = ex.Message
        Finally

        End Try
        Return strError
    End Function
```

Sau đó, khai báo để sử dụng phương thức này trong trường hợp thêm mới, cập nhật tùy thuộc vào giá trị của nút btnAdd như sau:

```
Private Sub btnAdd_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles btnAdd.Click
    'Nếu nhập mã khách hàng
    If txtID.Text <> "" Then
        'Gán giá trị cho mảng
        arrValue(1) = txtID.Text
        arrValue(2) = txtName.Text
        arrValue(3) = txtAddress.Text
        arrValue(4) = txtCity.Text
        If btnAdd.Text = "Save" Then
            'Thêm mới mẩu tin
            arrValue(0) = "1"
        Else
            'Cập nhật mẩu tin
            arrValue(0) = "2"
        End If
        'Khai báo và khởi tạo đối tượng clsDatabase
        Dim cls As New clsDatabase
        'Khai báo biến nhận kết quả trả về
        Dim strResult As String = ""
        'Nếu kết nối cơ sở dữ liệu thành công
        If cls.OpenConnection = "OK" Then
    'Gọi phương thức ExecuteSQL
        cls.ExecuteSQL(strSQL, _
        strResult, arrPara, arrValue)
      End If
    End If
End Sub
```

Giả sử, bạn thêm thông tin của khách hàng như hình 27-4-3 và nhấn nút Save.



Hình 27-4-3: *Nhập mới mẩu tin*

Lưu ý, để xóa thông tin trên các điều khiển nhập, bạn khai báo trong biến cố Click của nút Reset như sau:

```
Private Sub btnReset_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles btnReset.Click
    txtID.Text = ""
    txtName.Text = ""
    txtAddress.Text = ""
    txtCity.Text = ""
    btnAdd.Text = "Save"
End Sub
```

Tương tự, để cho phép người sử dụng xóa mẩu tin đang chọn, bạn khai báo trong biến cố Click của nút Delete như sau:

```
Private Sub btnDelete_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles btnDelete.Click
```

```
        If txtID.Text <> "" Then
          arrValue(1) = txtID.Text
          arrValue(2) = ""
          arrValue(3) = ""
          arrValue(4) = ""
          arrValue(0) = "3"
          'Khai báo và khởi tạo đối tượng clsDatabase
          Dim cls As New clsDatabase
          'Khai báo biến nhận kết quả trả về
          Dim strResult As String = ""
          'Nếu kết nối cơ sở dữ liệu thành công
          If cls.OpenConnection = "OK" Then
        'Gọi phương thức ExecuteSQL
          cls.ExecuteSQL(strSQL, _
          strResult, arrPara, arrValue)
        End If
      End If
    End Sub
```

Chú ý, để cho phép trình bày thông tin trên các điều khiển TextBox khi người sử dụng DoubleClick vào một dòng dữ liệu của DataGrig, bạn khai báo như sau:

```
Private Sub dgData_DoubleClick(ByVal sender As
Object, ByVal e As System.EventArgs) Handles
dgData.DoubleClick
    With dgData
      txtID.Text = .Item(.CurrentRowIndex, 0)
      txtName.Text = .Item(.CurrentRowIndex, 1)
      txtAddress.Text = _
        .Item(.CurrentRowIndex, 2)
      txtCity.Text = .Item(.CurrentRowIndex, 3)
    End With
    btnAdd.Text = "Update"
End Sub
```

2. ĐỐI TƯỢNG DATAVIEW

1. Thiết kế *Form*, dùng đối tượng *DataView* liệt kê danh sách *Country* trên điều khiển *ComboBox*, mỗi khi người sử dụng chọn *Country* thì bạn liệt kê danh sách khách hàng có *Country* đó trên điều khiển *DataGrid*.

Để thực hiện ví dụ này, trước tiên bạn kiểm tra trong cơ sở dữ liệu Northwind đã tồn tại thủ tục có tên spFilterCustomers. Nếu chưa tồn tại thì bạn có thể thực thi phát biểu tạo thủ tục như sau:

```
CREATE proc spFilterCustomers
As
  Select CustomerID,CompanyName,
  Address,City,Country
  from Customers
```

Kế đến, bạn khai báo phương thức trong lớp clsDatabase nhận tên thủ tục và trả về đối tượng DataTable nắm giữ tập mẩu tin.

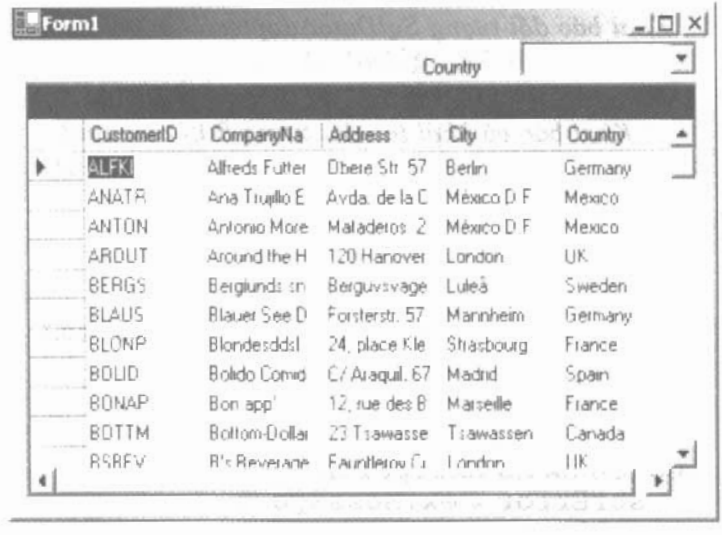
```
Public Function GetValue(ByVal strSQL As String)
As DataTable
    ' Khai báo và khởi tạo đối tượng DataTable
    Dim dtData As New DataTable
    ' Khai báo đối tượng SqlDataAdapter
    Dim myData As SqlDataAdapter
    Try
      ' Khai báo và khởi tạo đối tượng SqlCommand
      Dim myCom As New SqlCommand(strSQL)
      myCom.Connection = myCon
      myCom.CommandType = _
        CommandType.StoredProcedure
      ' Khởi tạo đối tượng SqlDataAdapter
      myData = New SqlDataAdapter(myCom)
      ' Điền dữ liệu vào đối tượng DataTable
      myData.Fill(dtData)
      strError = "OK"
    Catch ex As Exception
      strError = ex.Message
    Finally
      myData.Dispose()
    End Try
    Return dtData
End Function
```

Sau đó, khai báo phương thức LoadData trong lớp Form1 để gọi phương thức GetValue của đối tượng clsDatabase.

```
Private Sub LoadData()
```

```
        ' Khởi tạo đối tượng DataTable
        dtData = New DataTable
        ' Khởi tạo đối tượng clsDatabase
        cls = New clsDatabase
        ' Nếu kết nối cơ sở dữ liệu thành công
        If cls.OpenConnection = "OK" Then
            ' Gọi phương thức GetValue
            dtData = _
                cls.GetValue("spFilterCustomers")
            If dtData.Rows.Count > 0 Then
                Me.DataGrid1.DataSource = dtData
            End If
        End If
    End Sub
```

Sau khi Form1 nạp lên màn hình, lập tức danh sách tất cả mẩu tin trong bảng Customers trình bày trên điều khiển DataGrid như hình 27-5.



Hình 27-5: *Liệt kê danh sách khách hàng*

Như vậy, tập mẩu tin khách hàng được lưu trữ trong đối tượng DataTable, khi người sử dụng chọn Country, thay vì đọc lại cơ sở dữ liệu để lọc ra những khách hàng của Country đó thì bạn sử dụng thuộc tính RowFilter của đối tượng DataView để lọc dữ liệu từ đối tượng DataTable.

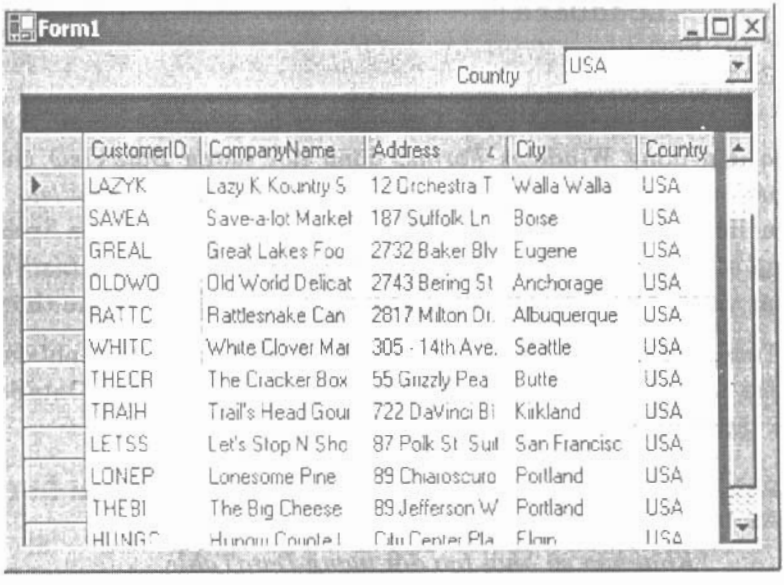
Để làm điều này, bạn khai báo phương thức FilterData, nhận giá trị Country và lọc mẩu tin trong đối tượng DataTable như sau:

```
Private Sub FilterData(ByVal Country As String)
    ' Khai báo và khởi tạo đối tượng DataView
    Dim dtView As New DataView(dtData)
    ' Khai báo lọc dữ liệu
    dtView.RowFilter = _
        "Country='" + Country + "'"
    ' Trình bày mẩu tin
    Me.DataGrid1.DataSource = dtView
End Sub
```

Sau đó, bạn khai báo để gọi phương thức này trong biến cố SelectedIndexChanged của điều khiển ComboBox.

```
Private Sub cbID_SelectedIndexChanged(ByVal
sender As System.Object, ByVal e As
System.EventArgs) Handles
cbID.SelectedIndexChanged
    FilterData(cbID.Text)
End Sub
```

Chẳng hạn, trong trường hợp này bạn chọn vào Country có tên là USA, lập tức danh sách khách hàng thuộc USA sẽ trình bày như hình 27-5-1.



Hình 27-5-1: *Lọc khách hàng*

Lưu ý, trong ví dụ trên chúng ta cần khai báo dùng chung hai biến đối tượng clsDatabase và DataTable như sau:

```
Dim cls As New clsDatabase
Dim dtData As DataTable
```

Ngoài ra, bạn khai báo trong biến cố Load của Form để gọi phương thức LoadData và phương thức GetValue của đối tượng clsDatabase để điền danh sách Country vào điều khiển ComboBox như sau:

```
Private Sub Form1_Load(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles MyBase.Load
    'Nếu kết nối cơ sở dữ liệu thành công
    If cls.OpenConnection = "OK" Then
        'Điền danh sách Country vào điều khiển ComboBox
        Dim arrValue As New ArrayList
        cls.GetValue(arrValue, _
          "Customers", "Country", "Country")
        For Each strValue As String In arrValue
          cbID.Items.Add(strValue)
        Next
        'Điền dữ liệu vào điều khiển DataGrid
        LoadData()
    End If
End Sub
```

2. Tạo ứng dụng *Windows Forms*, dùng đối tượng *DataView* cho phép người sử dụng chọn loại sản phẩm trên điều khiển *TreeView*, sau đó bạn liệt kê danh sách sản phẩm cùng với số lượng còn trong kho, tổng số lượng bán và số lượng đặt hàng trên điều khiển *ListView* thứ nhất và chi tiết của sản phẩm đó trên điều khiển *ListView* thứ hai.

Tương tự như ví dụ trên, trước tiên bạn khai báo phương thức GetValue thứ nhất, nhận tên thủ tục và trả về đối tượng DataSet nắm giữ hai tập mẩu tin.

```
Public Function GetValue(ByVal strSQL As String)
As DataTable
    'Khai báo và khởi tạo đối tượng DataTable
    Dim dtData As New DataTable
```

```
        'Khai báo đối tượng SqlDataAdapter
        Dim myData As SqlDataAdapter
        Try
            'Khai báo và khởi tạo đối tượng SqlCommand
            Dim myCom As New SqlCommand(strSQL)
            myCom.Connection = myCon
            myCom.CommandType = _
                CommandType.StoredProcedure
            'Khởi tạo đối tượng SqlDataAdapter
            myData = New SqlDataAdapter(myCom)
            'Điền dữ liệu vào đối tượng DataTable
            myData.Fill(dtData)
            strError = "OK"
        Catch ex As Exception
            strError = ex.Message
        Finally
            myData.Dispose()
        End Try
        Return dtData
    End Function
```

Kế đến, bạn khai báo phương thức GetValue thứ hai, nhận tham trị là tên bảng dữ liệu, chuỗi tương ứng, tên và giá trị (dùng để điền vào điều khiển TreeView) rồi trả về đối tượng ArrayList như sau.

```
    Public Function GetValue(ByVal strTable As
    String, ByVal strName As String, ByVal strValue
    As String) As ArrayList
        'Khai báo phát biểu SQL
        strSQL = "select distinct " + strValue
        strSQL += "," + strName + " from " + strTable
        'Khai báo và khởi tạo đối tượng ArrayList
        Dim arrValue As New ArrayList
        Dim myCom As SqlCommand
        Dim RD As SqlDataReader
        Dim cls As clsItem
        Try
            'Khởi tạo đối tượng SqlCommand
            myCom = New SqlCommand(strSQL, myCon)
            RD = myCom.ExecuteReader
            'Duyệt trên mỗi mẩu tin
```

```
        While RD.Read
            ' Khởi tạo đối tượng clsItem
            cls = New clsItem( _
            Convert.ToString(RD.GetValue(0)), _
            RD.GetString(1))
            ' Thêm vào đối tượng ArrayList
            arrValue.Add(cls)
        End While
        strError = "OK"
      Catch ex As Exception
        strError = ex.Message
      Finally
        RD.Close()
      End Try
      Return arrValue
  End Function
```

Sau đó, bạn khai báo để gọi phương thức GetValue trong biến cố Click của nút Load như sau:

```
Private Sub btnLoad_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles btnLoad.Click
    Dim list As ArrayList
    list = cls.GetValue("Products", _
      "ProductName", "ProductID")
    ' Gọi phương thức FillData để điền dữ liệu vào điều khiển
    ' TreeView
    FillData(list)
End Sub
```

Trong đó, phương thức FillData nhận đối tượng ArrayList, bạn sử dụng đối tượng TreeNode để điền từng đối tượng bao gồm thuộc tính Name và Value vào hai thuộc tính Text và Tag của đối tượng TreeNode như sau:

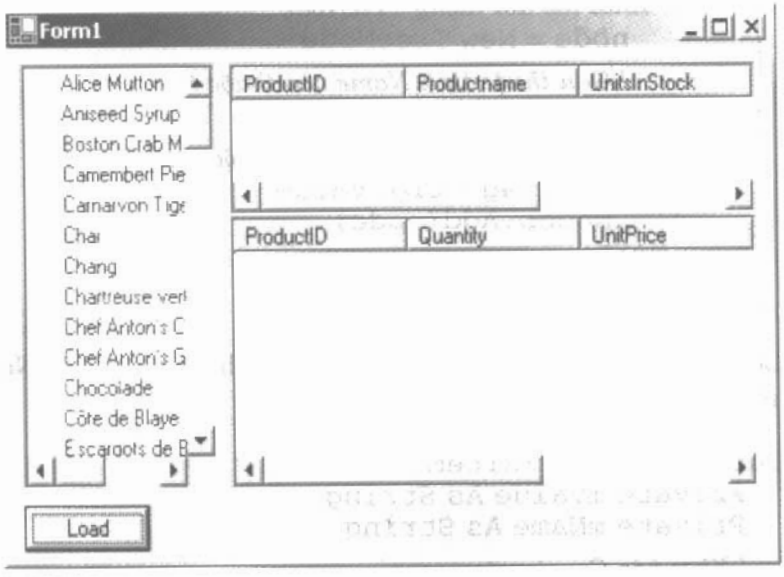
```
Sub FillData(ByVal arrList As ArrayList)
    tvData.Nodes.Clear()
    With tvData
      Dim i As Integer
      Dim node As TreeNode
      Dim Item As clsItem
```

```
            ' Duyệt trên từng phần tử của ArrayList
            For Each obj As clsItem In arrList
            ' Khởi tạo đối tượng TreeNode
              node = New TreeNode
              ' Gán thuộc tính Name vào thuộc tính Text
              node.Text = obj.Name
              ' Gán thuộc tính Value vào thuộc tính Tag
              node.Tag = obj.Value
              .Nodes.Add(node)
            Next
        End With
    End Sub
```

Lưu ý, bạn cần khai báo lớp clsItem với hai thuộc tính Name và Value như sau:

```
Public Class clsItem
  Private mValue As String
  Private mName As String
  ' Khai báo Constructor
  Public Sub New(ByVal strID As String, _
    ByVal strName As String)
    mValue = strID
    mName = strName
  End Sub
  ' Khai báo thuộc tính
  Property Name() As String
    Get
      Return mName
    End Get
    Set(ByVal Value As String)
      mName = Value
    End Set
  End Property
  Property Value() As String
    Get
      Return mValue
    End Get
    Set(ByVal Value As String)
      mValue = Value
    End Set
  End Property
End Class
```

Khi thực thi chương trình, danh sách sản phẩm trình bày trên điều khiển TreeView như hình 27-6.



Hình 27-6: *Danh sách sản phẩm*

Mỗi khi người sử dụng chọn vào Node, bằng cách lấy mã sản phẩm từ thuộc tính Tag của Node đang chọn, gọi phương thức FillData để điền danh sách sản phẩm chi tiết và số lượng đặt hàng, tồn kho và tổng số lượng bán như sau:

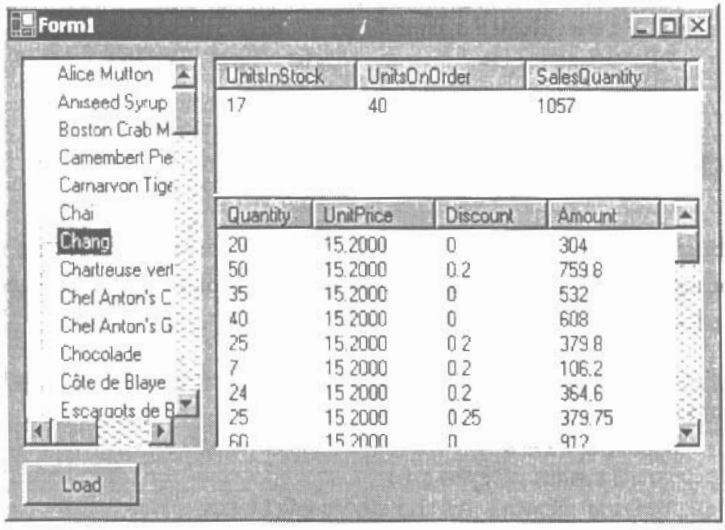
```
Private Sub tvData_AfterSelect(ByVal sender As
System.Object, ByVal e As TreeViewEventArgs)
Handles tvData.AfterSelect
    Dim dtView As DataView
    ' Lọc dữ liệu của bảng thứ nhất từ đối tượng DataSet
    dtView = New DataView(dtData.Tables(0))
    ' Khai báo lọc dữ liệu theo từng sản phẩm
    dtView.RowFilter = _
    "ProductID='" + e.Node.Tag.ToString + "'"
    ' Gọi phương thức FillData
    FillData(lvData, dtView)
    ' Lọc dữ liệu của bảng thứ hai từ đối tượng DataSet
    dtView = New DataView(dtData.Tables(1))
    ' Khai báo lọc dữ liệu theo từng sản phẩm
```

```
        dtView.RowFilter = _
        "ProductID='" + e.Node.Tag.ToString + "'"
        'Gọi phương thức FillData
        FillData(lvDetail, dtView)
    End Sub
```

Trong đó, phương thức FillData thứ hai, nhận đối tượng ListView và đối tượng DataView, sau đó điền dữ liệu từ đối tượng DataView đó vào đối tượng ListView.

```
Sub FillData(ByRef lView As ListView, ByVal
dtView As DataView)
    With lView
       .Items.Clear()
       Dim item1 As ListViewItem
       Dim Item As clsItem
       'Duyệt trên từng hàng dữ liệu
       For i As Integer = 0 To dtView.Count - 1
          'Khởi tạo đối tượng ListViewItem
          item1 = New ListViewItem( _
            dtView.Item(i).Item(0).ToString)
         'Duyệt trên từng cột dữ liệu
          For j As Integer = 1 To _
             lvData.Columns.Count - 1
             item1.SubItems.Add( _
             dtView.Item(i).Item(j))
          Next
          'Thêm đối tượng ListViewItem vào đối tượng ListView
          .Items.Add(item1)
       Next
    End With

End Sub
```

Chẳng hạn, trong trường hợp này bạn chọn vào sản phẩm có tên Change, tổng số lượng bán hàng, tồn trong kho, số lượng đặt hàng trình bày trên điều khiển ListView thứ nhất, chi tiết của sản phẩm đó trình bày trên điều khiển ListView thứ hai như hình 27-6-1.



Hình 27-6-1: *Chi tiết sản phẩm*

Chú ý, đối tượng DataSet được khởi tạo và điền dữ liệu bằng phương thức GetValue trong biến cố Load của Form.

```
Private Sub Form1_Load(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles MyBase.Load
    'Nếu kết nối cơ sở dữ liệu thành công
    If cls.OpenConnection = "OK" Then
      dtData = New DataSet
      'Gọi phương thức GetValue
      dtData = cls.GetValue("spSales")
      lvData.CheckBoxes = False
      'Chọn kiểu trình bày
      lvData.View = View.Details
      lvDetail.View = View.Details
      'Khai báo tên cột cho điều khiển ListView thứ nhất
      For Each dc As DataColumn In _
        dtData.Tables(0).Columns
        lvData.Columns.Add(dc.ColumnName, _
          100, 0)
      Next
      'Khai báo tên cột cho điều khiển ListView thứ hai
      For Each dc As DataColumn In _
        dtData.Tables(1).Columns
```

```
            lvDetail.Columns.Add( _
            dc.ColumnName, 100, 0)
        Next
    End If
End Sub
```

Trong ví dụ trên, chúng ta sử dụng thủ tục có tên spSales, bạn có thể sử dụng cú pháp sau để tạo chúng trong cơ sở dữ liệu Northwind.

```
CREATE PROC spSales
AS
   -- Trả về danh sách sản phẩm với số lượng bán, đặt hàng,
   -- tồn kho
   SELECT P.ProductID,Productname,
   UnitsInStock,UnitsOnOrder,
   sum(Quantity) As SalesQuantity
   FROM Products P, [Order Details] D
   WHERE P.ProductID=D.ProductID
   group by P.ProductID,Productname,
   UnitsInStock,UnitsOnOrder
   order by Productname
   -- Chi tiết sản phẩm đã bán
   SELECT
   ProductID,Quantity,UnitPrice,Discount,
   Quantity*UnitPrice-Discount As Amount
   FROM [Order Details]

spSales
```

3. Thiết kế *Form*, dùng đối tượng *DataView* cho phép người sử dụng chọn *Country* trên điều khiển *ComboBox* thứ nhất và nhân viên trên điều khiển *ComboBox* thứ hai thì liệt kê danh sách sản phẩm của nhân viên đó thuộc *Country* đã chọn.

Để thực hiện ví dụ này, trước tiên bạn khai báo thủ tục nội tại trong SQL Server có tên spSalesByCountryAndEmployee với cấu trúc như sau:

```
CREATE PROC spSalesByCountryAndEmployee
AS
   SELECT O.EmployeeID,C.Country,
   P.ProductID,Productname,
   UnitsInStock,UnitsOnOrder,
   sum(Quantity) As SalesQuantity
```

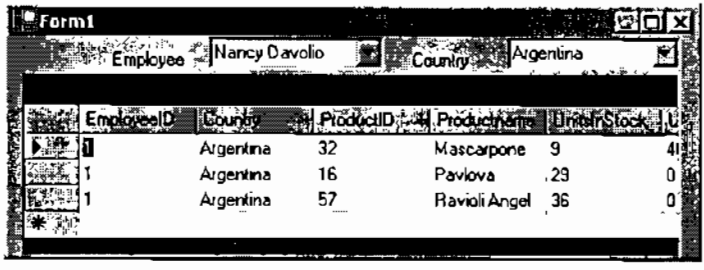
```
FROM
   Products P, [Order Details] D,
   Customers C,Employees E, Orders O
WHERE P.ProductID=D.ProductID
   AND C.CustomerID=O.CustomerID
   AND O.OrderID=D.OrderID
   AND O.EmployeeID=E.EmployeeID
GROUP BY O.EmployeeID,C.Country,
   P.ProductID,Productname,
   UnitsInStock,UnitsOnOrder
ORDER BY Productname
```

Kế đến, bạn khai báo trong biến cố Load của Form để gọi phương thức GetValue nhận đối tượng ArrayList trả về, bằng cách duyệt trên từng phần tử của đối tượng ArrayList, bạn có thể điền danh sách Country và Employee vào hai điều khiển ComboBox như sau:

```
Private Sub Form1_Load(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles MyBase.Load
    If cls.OpenConnection = "OK" Then
        ' Khai báo đối tượng ArrayList
        Dim arrValue As ArrayList
        ' Khởi tạo đối tượng ArrayList thứ nhất
        arrValue = New ArrayList
        ' Gọi phương thức GetValue để lấy danh sách Country
        arrValue = cls.GetValue( _
          "Customers", "Country", "Country")
        ' Điền vào điều khiển ComboBox
        cbCountry.DataSource = arrValue
        cbCountry.DisplayMember = "Name"
        cbCountry.ValueMember = "Value"
        ' Khởi tạo đối tượng ArrayList thứ hai
        arrValue = New ArrayList
        ' Gọi phương thức GetValue để lấy danh sách Employee
        arrValue = cls.GetValue("Employees", _
        "FirstName + ' ' + LastName As FullName", _
        "EmployeeID")
        ' Điền vào điều khiển ComboBox
        cbEmployee.DataSource = arrValue
        cbEmployee.DisplayMember = "Name"
        cbEmployee.ValueMember = "Value"
```

```
        ' Gọi phương thức trình bày dữ liệu trên điều khiển
      DataGrid
        LoadData()
      End If
  End Sub
```

Khi thực thi chương trình, danh sách Country và Employee xuất hiện trên hai điều khiển ComboBox tương ứng như hình 27-1.



Hình 27-7: *Lọc dữ liệu*

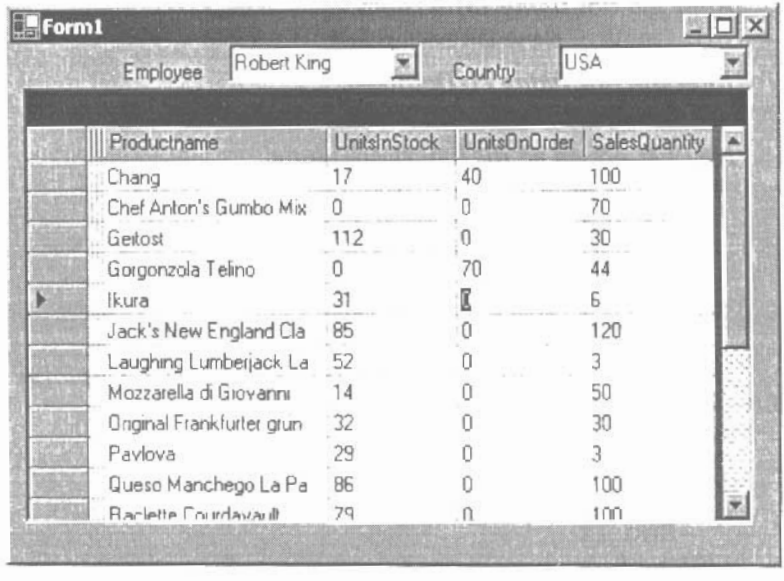
Trong đó, phương thức FillData kết nối cơ sở dữ liệu và gọi phương thức GetValue để điền danh sách chi tiết sản phẩm vào đối tượng DataTable.

```
Private Sub LoadData()
  ' Khởi tạo đối tượng DataTable
    dtData = New DataTable
  ' Khởi tạo đối tượng clsDatabase
    cls = New clsDatabase
  ' Nếu kết nối cơ sở dữ liệu thành công
    If cls.OpenConnection = "OK" Then
      ' Gọi phương thức GetValue
      dtData = cls.GetValue( _
      "spSalesByCountryAndEmployee")
      If dtData.Rows.Count > 0 Then
        ' Trình bày dữ liệu lọc theo Country và Employee
        FilterData(Convert.ToString( _
        cbCountry.SelectedValue), _
        Convert.ToString( _
        cbEmployee.SelectedValue))
      End If
    End If
End Sub
```

Lưu ý phương thức FilterData sử dụng đối tượng DataView để lọc danh sách sản phẩm chi tiết theo Country và Employee đang chọn trên màn hình được khai báo như sau:

```
Private Sub FilterData(ByVal Country As String,
ByVal Employee As String)
    Dim dtView As New DataView(dtData)
    dtView.RowFilter = " 1=1 "
    'Nếu có chọn Country
    If Country <> "" Then
      dtView.RowFilter += _
      " and Country='" + Country + "'"
    End If
    'Nếu có chọn Employee
    If Employee <> "" Then
      dtView.RowFilter += _
      " and EmployeeID=" + Employee
    End If
    Me.DataGrid1.DataSource = dtView
End Sub
```

Giả sử, bạn chọn Employee bất kỳ, lập tức danh sách sản phẩm chi tiết lọc theo Employee đó cùng với Country đang chọn. Tương tự như vậy chọn trường hợp chọn Country, kết quả trình bày như hình 27-7-1.



Hình 27-7-1: *Lọc dữ liệu*

3. CẬP NHẬT DỮ LIỆU TỪ ĐỐI TƯỢNG DATATABLE

1. Thiết kế *Form* để diễn dữ liệu của bảng nào đó trong cơ sở dữ liệu *Northwind*, sau đó cho phép người sử dụng thêm mới, thay đổi hay xóa mẩu tin rồi cập nhật trở lại dữ liệu nguồn.

Để thực hiện ví dụ này, trước tiên bạn khai báo phương thức có tên GetValue trong lớp clsDatabase, nhận tham biến là đối tượng DataTable như sau:

```
Public Function GetValue( _
  ByRef myDT As DataTable, _
  ByVal strSQL As String) As String
    Dim adData As SqlDataAdapter
    Try
      adData = New SqlDataAdapter(strSQL, myCon)
      adData.Fill(myDT)
      strError = "OK"
    Catch ex As Exception
      strError = ex.Message
    Finally

    End Try
    Return strError
End Function
```

Lưu ý, bạn khai báo 3 biến sử dụng chung cho lớp Form1.

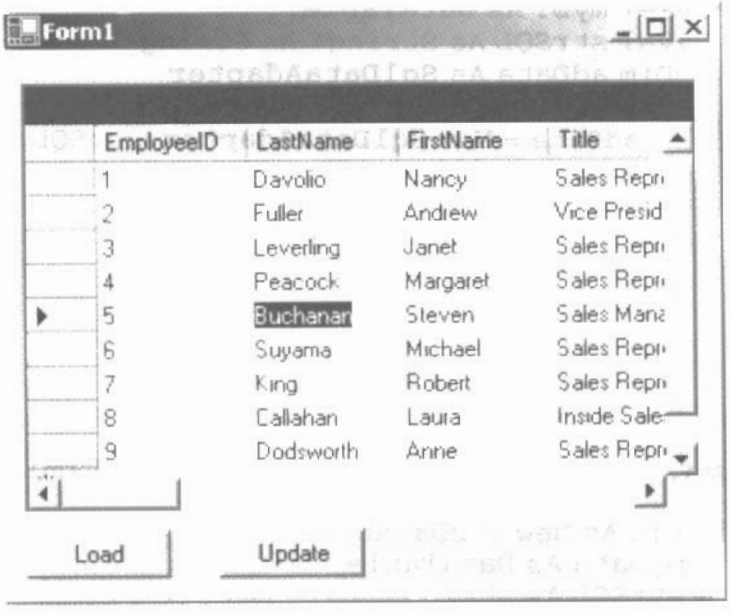
```
Dim cls As New clsDatabase
Dim myData As DataTable
Dim strSQL As String
```

Bằng cách khai báo để gọi phương thức GetValue, rồi điền dữ liệu lấy ra được vào điều khiển DataGrid trong biến cố Click của nút Load như sau:

```
Private Sub btnLoad_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles btnLoad.Click
    'Nếu kết nối cơ sở dữ liệu thành công
    If cls.OpenConnection() = "OK" Then
    'Khởi tạo đối tượng DataTable
```

```
        myData = New DataTable
        ' Khai báo phát biểu SQL
        strSQL = "Select * From Employees"
        ' Gọi phương thức GetValue
        cls.GetValue(myData, strSQL)
        ' Điền dữ liệu vào điều khiển DataGrid
        Me.dgData.DataSource = myData
    End If
End Sub
```

Khi thực thi chương trình, nếu người sử dụng nhấn nút Load, lập tức danh sách mẫu tin của bảng Employees xuất hiện như hình 27-8.



Hình 27-8: *Danh sách nhân viên*

Nếu cho phép người sử dụng cập nhật dữ liệu thay đổi trên điều khiển DataGrid, bạn khai báo trong biến cố Click của nút Update như sau:

```
Private Sub btnUpdate_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles btnUpdate.Click
    ' Khai báo và khởi tạo đối tượng DataTable
    Dim myNewData As New DataTable
    ' Lấy ra tập dữ liệu thay đổi
```

```
        myNewData = myData.GetChanges
        ' Nếu dữ liệu thay đổi
        If Not myNewData Is Nothing Then
            ' Gọi phương thức cập nhật dữ liệu thay đổi
            If cls.UpdateData(myNewData, _
            strSQL) = "OK" Then
                MsgBox("Updated")
                ' Nạp lại dữ liệu lên điều khiển DataGrid
                Call btnLoad_Click(sender, e)
                ' Chấp nhận sự thay đổi
                myData.AcceptChanges()
            End If
        End If
    End Sub
```

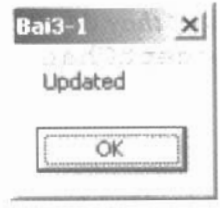
Chú ý, bằng cách sử dụng đối tượng SqlCommandBuilder, bạn có thể cập nhật dữ liệu thay đổi trên điều khiển DataGrid vào dữ liệu nguồn.

Để làm điều này, bạn khai báo phương thức UpdateData nhận đối tượng DataSet đang nắm giữ tập dữ liệu mà người sử dụng đã thay đổi trên điều khiển DataGrid.

```
    Public Function UpdateData( _
        ByVal myNewDT As DataTable, _
        ByVal strSQL As String) As String

        Dim adData As SqlDataAdapter
        Try
            ' Khởi tạo đối tượng SqlDataAdapter
            adData = New SqlDataAdapter(strSQL, myCon)
            ' Khai báo và khởi tạo đối tượng SqlCommandBuilder
            Dim myBuilder As New _
                SqlCommandBuilder(adData)
            ' Sử dụng thuộc tính TableMappings
            adData.TableMappings.Add("Table", _
                myNewDT. TableName)
            ' Gọi phương thức Update
            adData.Update(myNewDT)
            strError = "OK"
        Catch ex As Exception
```

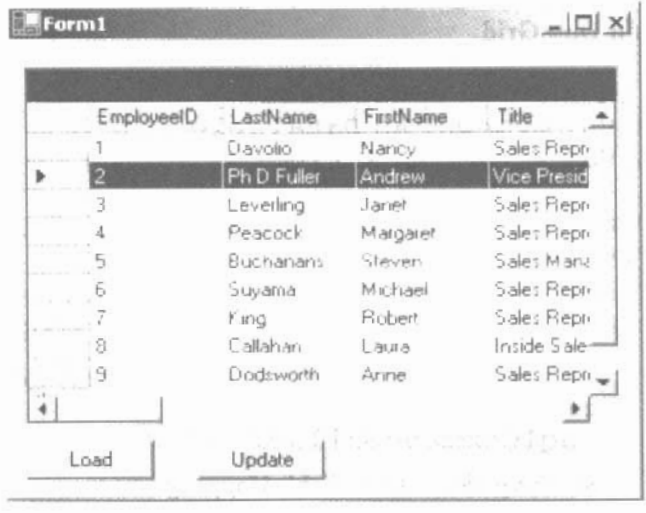
```
            strError = ex.Message
        Finally

        End Try
        Return strError
    End Function
```

Chẳng hạn, người sử dụng thay đổi chuỗi "**Fuller**" trong cột Lastname thành "**PhD Fuller**" của mẩu tin thứ 2 và nhấn nút Update, dữ liệu sẽ cập nhật và thông báo như hình 27-8-1.



Hình 27-8-1: *Cập nhật thành công*

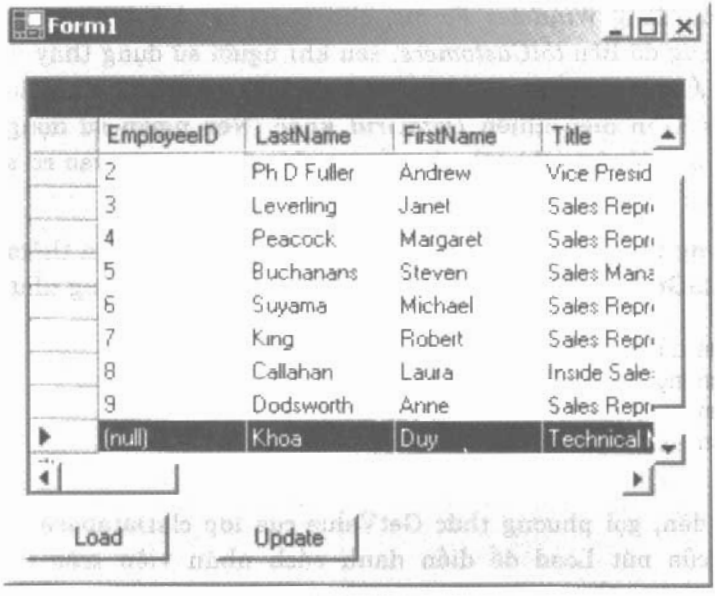
Bằng cách gọi lại phương thức btnLoad_Click, bạn có thể nạp lại dữ liệu đã thay đổi, khi đó mẩu tin thứ 2 trình bày như hình 27-8-2.



Hình 27-8-2: *Dữ liệu đã cập nhật*

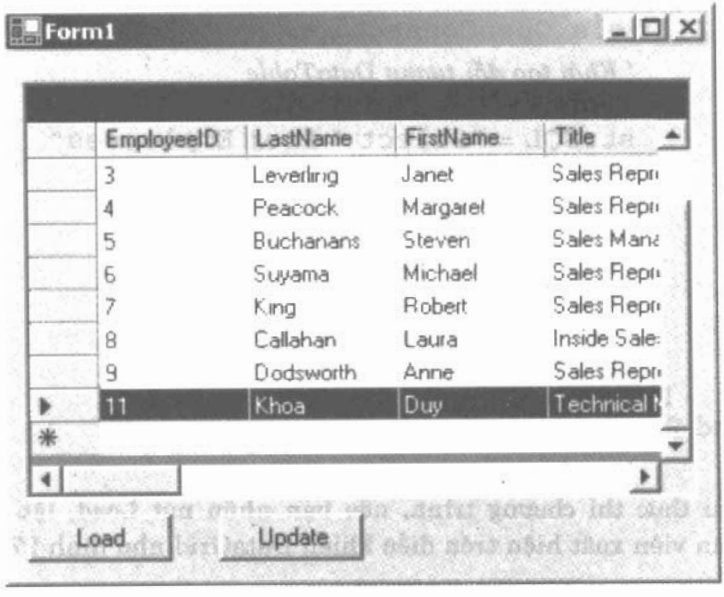
Ví dụ, sau khi thay đổi tên của nhân viên thứ 2, bạn tiếp tục thêm mới nhân viên thứ 10 với các thông tin như hình 27-8-3.

Chú ý, cột EmployeeID có kiểu số tự động, chính vì vậy bạn không cần nhập số vào cột này.



Hình 27-8-3: *Thêm mới mẩu tin*

Sau khi nhấn nút Update, dữ liệu sẽ được thêm vào bảng Employees, lập tức số 11 (số tự động phát sinh) xuất hiện trong cột EmployeeID như hình 27-8-4.



Hình 27-8-4: *Dữ liệu đã được thêm vào*

2. Tạo ứng dụng *Windows Forms*, cho phép liệt kê danh sách mẩu tin của bảng dữ liệu *tblCustomers*, sau khi người sử dụng thay đổi dữ liệu và nhấn nút *Update*, lập tức danh sách các mẩu tin thay đổi sẽ được liệt kê trên điều khiển *DataGrid* khác. Nếu người sử dụng tiếp tục nhấn nút *Update* thì tất cả mẩu tin đó sẽ cập nhật vào cơ sở dữ liệu nguồn.

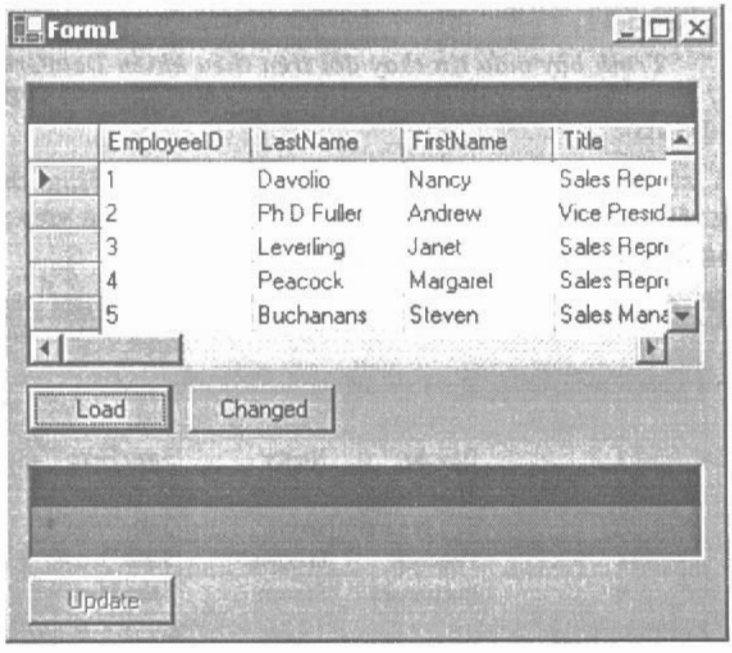
Tương tự như ví dụ trên, đối với trường hợp này bạn thêm hai điều khiển DataGrid vào Form và khai báo 4 biến sử dụng chung như sau:

```
Dim cls As New clsDatabase
Dim myData As DataTable
Dim myChangedwData As DataTable
Dim strSQL As String
```

Kế đến, gọi phương thức GetValue của lớp clsDatabase trong biến cố Click của nút Load để điền danh sách nhân viên trên điều khiển DataGrid thứ nhất.

```
Private Sub btnLoad_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles btnLoad.Click
    ' Nếu kết nối cơ sở dữ liệu thành công
    If cls.OpenConnection() = "OK" Then
        ' Khởi tạo đối tượng DataTable
        myData = New DataTable
        strSQL = "Select * from Employees"
        ' Gọi phương thức getValue
        cls.GetValue(myData, strSQL)
        ' Trình bày dữ liệu
        Me.dgData.DataSource = myData
        ' Cho phép sử dụng nút Changed
        btnChanged.Enabled = True
    End If
End Sub
```

Khi thực thi chương trình, nếu bạn nhấn nút Load, lập tức danh sách nhân viên xuất hiện trên điều khiển DataGrid như hình 27-9.



Hình 27-9: *Liệt kê danh sách nhân viên*

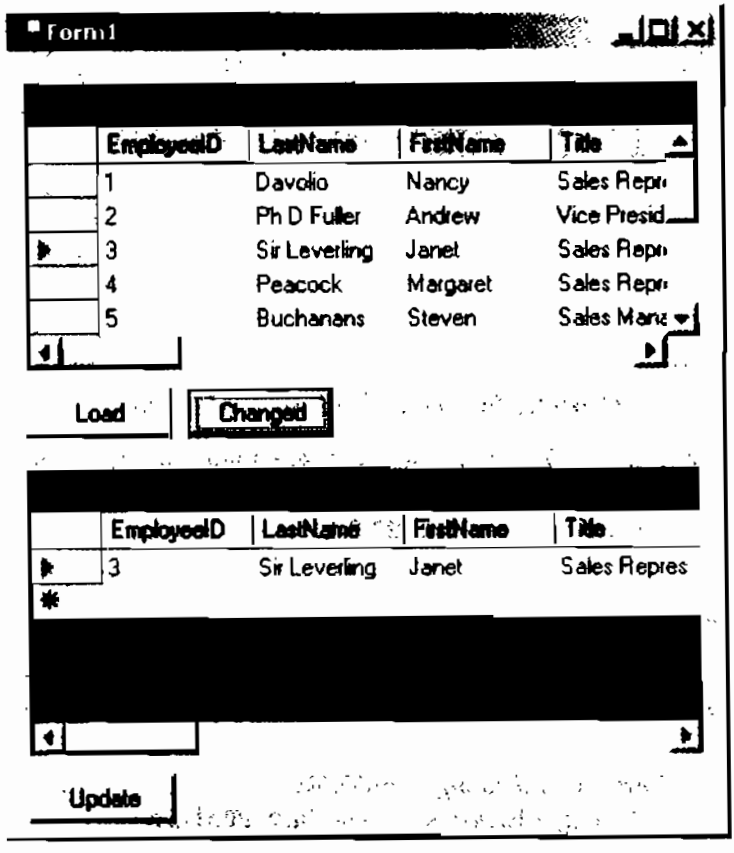
Sau khi thay đổi, thêm mới hay xóa mẩu tin, nếu bạn nhấn nút Changed, lập tức danh sách mẩu tin vừa thay đổi sẽ xuất hiện trên điều khiển DataGrid thứ hai như hình 27-9-1.

Để làm điều này, bạn khai báo trong biến cố Click của nút btnChanged như sau:

```
Private Sub btnChanged_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles btnChanged.Click
    ' Khởi tạo đối tượng DataTable
    myChangedwData = New DataTable
    ' Lấy ra mẩu tin thay đổi
    myChangedwData = myData.GetChanges
    ' Nếu có mẩu tin thay đổi
    If Not myChangedwData Is Nothing Then
        ' Cho phép sử dụng nút Update
        btnUpdate.Enabled = True
    Else
        btnUpdate.Enabled = False
```

```
        End If
        ' Trình bày mẩu tin thay đổi trên điều khiển DataGrid thứ 2
        Me.dgChanged.DataSource = myChangedwData
    End Sub
```

Giả sử, bạn thay đổi tên của nhân viên Leverling thành Sir Leverling rồi nhấn nút Changed, lập tức mẩu tin của nhân viên này xuất hiện trong điều khiển DataGrid thứ hai như hình 27-9-1.



Hình 27-9-1: *Những mẩu tin thay đổi*

Nếu người sử dụng nhấn nút Update, lập tức mẩu tin này sẽ được cập nhật vào dữ liệu nguồn.

Chú ý, bạn sử dụng phương thức AcceptChanges để chấp nhận sự thay đổi đó. Nếu người sử dụng tiếp tục nhấn nút Changed, mẩu tin vừa cập nhật sẽ không xuất hiện trong điều khiển DataGrid thứ hai.

Chuyên đề 20:

ĐỐI TƯỢNG DATAROW, DATACOLUMN VÀ DATARELATION

1. ĐỐI TƯỢNG DATAROW

1. Viết phương thức nhận phát biểu *Select* và trả về đối tượng ứng với mẩu tin đầu tiên, sau đó sử dụng phương thức này để điền dữ liệu lên các điều khiển tương ứng.

Thêm Class vào Project và đặt tên clsDatabase, khai báo phương thức GetValue nhận phát biểu Select bằng cách sử dụng đối tượng DataTable. Bạn có thể đọc dữ liệu từ cơ sở dữ liệu và điền chúng vào đối tượng DataTable này, rồi sau đó trả về đối tượng DataRow.

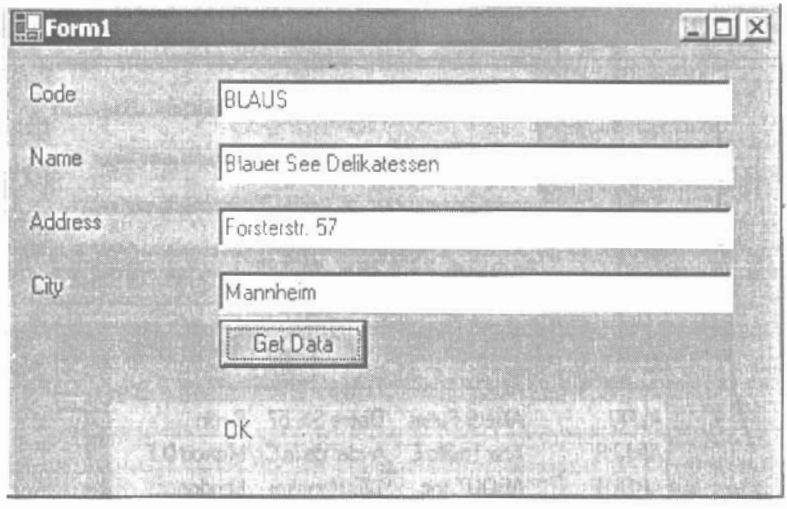
```
Public Function GetValue( _
  ByVal strSQL As String) As DataRow
    Dim adData As SqlDataAdapter
    Dim myDR As DataRow
    Try
      ' Khởi tạo đối tượng SqlDataAdapter
      adData = New SqlDataAdapter(strSQL, myCon)
      ' Khai báo và khởi tạo đối tượng DataTable
      Dim myDT As New DataTable
      ' Điền dữ liệu vào đối tượng DataTable
      adData.Fill(myDT)
      ' Nếu tồn tại mẩu tin
      If myDT.Rows.Count > 0 Then
        ' Lấy ra mẩu tin đầu tiên
        myDR = myDT.Rows(0)
      End If
      strError = "OK"
    Catch ex As Exception
      strError = ex.Message
    Finally

    End Try
    ' Trả về đối tượng DataRow
```

```
    Return myDR
End Function
```

Sau đó, khai báo trong biến cố Click của nút GetData như sau:

```
Private Sub btnGet_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles btnGet.Click
    If txtID.Text <> "" Then
      Dim cls As New clsDatabase
      'Nếu kết nối cơ sở dữ liệu thành công
      If cls.OpenConnection() = "OK" Then
        'Khai báo đối tượng DataRow
        Dim dr As DataRow
        'Gọi phương thức GetValue
        dr = cls.GetValue("select * from " & _
        "tblCustomers where CustomerID='" _
        + txtID.Text + "'")
        If Not dr Is Nothing Then
          'Điền thông tin từ đối tượng DataRow
          txtName.Text = dr.Item(1)
          txtAdd.Text = dr.Item(2)
          txtCity.Text = dr.Item(3)
        Else
          'Xóa thông tin đang tồn tại
          txtName.Text = ""
          txtAdd.Text = ""
          txtCity.Text = ""
        End If
        lbResult.Text = cls.strError
        cls.CloseConnection()
      End If
    End If
End Sub
```

Khi thực thi chương trình, bạn nhập mã khách hàng và nhấn nút Get Data. Nếu mã khách hàng tồn tại trong bảng tblCustomers, lập tức thông tin khác sẽ trình bày trên các điều khiển tương tự như hình 28-1.



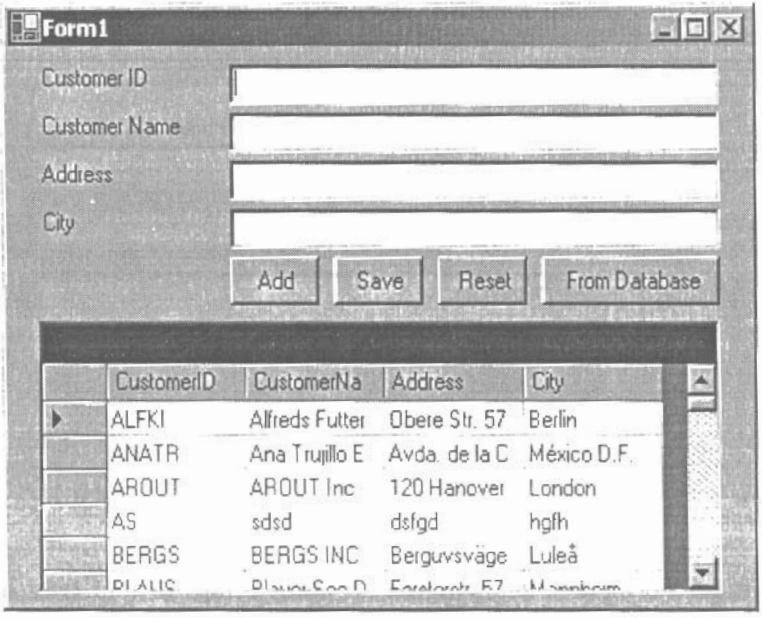
Hình 28-1: *Sử dụng đối tượng DataRow*

2. Thiết kế *Form*, cho phép người sử dụng nhập dữ liệu trên các điều khiển, sau đó trình bày dữ liệu đó trên *DataGrid*. Khi người sử dụng nhấn nút *Save*, bạn lưu dữ liệu đó xuống cơ sở dữ liệu nguồn.

Để thực hiện ví dụ này, trước tiên bạn khai báo phương thức nhận tham biến là đối tượng DataTable và tham trị là phát biểu SQL. Kế đến, bạn khai báo phương thức có tên LoadData để gọi phương thức GetValue như sau:

```
Sub LoadData()
    If cls.OpenConnection() = "OK" Then
      myData = New DataTable
      strSQL = "Select * from tblCustomers"
      cls.GetValue(myData, strSQL)
      Me.dgData.DataSource = myData
    End If
End Sub
```

Để điền dữ liệu từ bảng tblCustomers, bạn có thể gọi phương thức LoadData trong biến cố Load của Form và Click của nút Load. Mỗi khi Form1 nạp lên, mặc định danh sách khách hàng trình bày như hình 28-2.



Hình 28-2: *Liệt kê danh sách khách hàng*

Để cho phép người sử dụng thêm thông tin nhập trên điều khiể TextBox vào điều khiển DataGrid, bạn sử dụng đối tượng DataRow kế hợp với đối tượng DataTable như sau:

```
Private Sub btnAdd_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles btnAdd.Click
    'Nhập mã khách hàng
    If txtID.Text <> "" Then
    'Khai báo đối tượng DataRow
      Dim dr As DataRow
      'Xóa dữ liệu trong đối tượng DataTable
      myData.Rows.Clear()
      'Tạo mới một hàng dữ liệu dựa trên cấu trúc của
      'bảng tblCustomers
      dr = myData.NewRow
      'Gán giá trị cho từng cột
      dr.Item(0) = txtID.Text
      dr.Item(1) = txtName.Text
      dr.Item(2) = txtAddress.Text
```

```
            dr.Item(3) = txtCity.Text
            'Thêm đối tượng DataRow vào đối tượng DataTable
            myData.Rows.Add(dr)
            'Trình bày trên điều khiển DataGrid
            Me.dgData.DataSource = myData
        End If
    End Sub
```

Để cho phép người sử dụng cập nhật dữ liệu lưu trữ trong đối tượng DataTable đang trình bày trên điều khiển DataGrid, bạn khai báo phương thức UpdateData như sau:

```
Public Function UpdateData( _
    ByVal myNewDT As DataTable, _
    ByVal strSQL As String) As String
    'Khai báo đối tượng SqlDataAdapter
    Dim adData As SqlDataAdapter
    Try
        'Khởi tạo đối tượng SqlDataAdapter
        adData = New SqlDataAdapter(strSQL, myCon)
        'Khai báo và khởi tạo đối tượng SqlCommandBuilder
        Dim myBuilder As _
        New SqlCommandBuilder(adData)
        'Khai báo tên bảng dữ liệu
        adData.TableMappings.Add("Table", _
        myNewDT.TableName)
        'Gọi phương thức Update của đối tượng SqlDataAdapter
        adData.Update(myNewDT)
        strError = "OK"
    Catch ex As Exception
        strError = ex.Message
    Finally

    End Try
    Return strError
End Function
```

3. Viết ứng dụng *Windows Forms*, dùng để liệt kê tên tập tin, dung lượng, ngày tạo ra của thư mục chỉ định; sau đó điền dữ liệu đó vào điều khiển *DataGrid*.

Để cho phép dùng chọn thư mục mỗi khi nhấn vào nút Open, bạn khai báo sử dụng đối tượng FolderBrowserDialog trong biến cố Click của nút Open như sau:

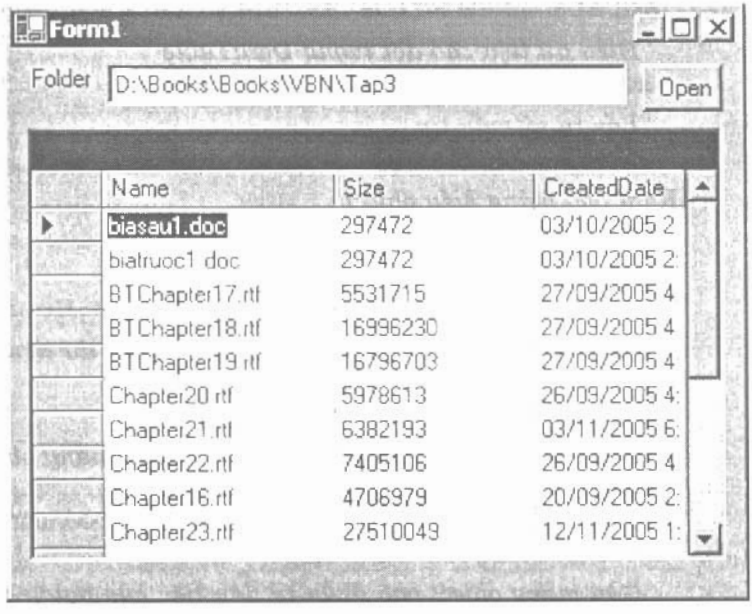
```
Private Sub btnOpen_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles btnOpen.Click
    ' Khai báo và khởi tạo đối tượng FolderBrowserDialog
    Dim fd As New FolderBrowserDialog
    ' Chọn thư mục mặc định chọn
    fd.RootFolder = _
    Environment.SpecialFolder.MyComputer
    ' Nếu người sử dụng chọn nút OK sau khi chọn thư mục
    If fd.ShowDialog = DialogResult.OK Then
        ' Lấy ra đường dẫn thư mục
        txFolder.Text = fd.SelectedPath
        ' Gọi phương thức liệt kê tập tin
        ReadFiles(txFolder.Text)
    End If
End Sub
```

Trong đó, phương thức ReadFiles nhận tham trị là tên và đường dẫn thư mục. Sử dụng đối tượng DirectoryInfo và FileInfo, bạn có thể liệt kê danh sách tập tin cùng với thuộc tính của chúng như sau:

```
Sub ReadFiles(ByVal strPath As String)
    Dim dir As New DirectoryInfo(strPath)
    ' Khai báo và khởi tạo đối tượng DataTable
    Dim dt As New DataTable("Files")
    ' Định nghĩa các cột dữ liệu
    dt.Columns.Add("Name")
    dt.Columns.Add("Size")
    dt.Columns.Add("CreatedDate")
    ' Khai báo đối tượng DataRow
    Dim dr As DataRow
    ' Duyệt trên từng tập tin
    For Each file As FileInfo In dir.GetFiles
        ' Thêm mới một hàng
        dr = dt.NewRow
```

```
            ' Gán giá trị
            dr.Item(0) = file.Name
            dr.Item(1) = file.Length.ToString
            dr.Item(2) = file.CreationTime
            ' Thêm đối tượng DataRow vào đối tượng DataTable
            dt.Rows.Add(dr)
        Next
        ' Trình bày danh sách tập tin
        Me.DataGrid1.DataSource = dt
    End Sub
```

Khi thực thi chương trình, nếu người sử dụng chọn vào nút Open, lập tức cửa sổ chọn thư mục xuất hiện. Chọn nút OK sau khi chọn thư mục, lập tức danh sách tập tin của thư mục đó trình bày như hình 28-3.



Hình 28-3: *Danh sách tập tin*

4. Viết phương thức nhận phát biểu *Select* trả về mảng một chiều kiểu *object*, trong đó mỗi phần tử là một mảng một chiều kiểu *object* tương ứng với một hàng, rồi sử dụng phương thức này trong một *Form* để điền dữ liệu lên điều khiển *DataGrid* lấy từ mảng trên.

Để thực hiện ví dụ này, trước tiên bạn khai báo phương thức GetValue nhận phát biểu SQL duyệt trên từng mẩu tin bằng cách sử

dụng đối tượng DataRow, chuyển mỗi mẩu tin thành một mảng đối tượng rồi gán vào phần tử thứ I của một mảng đối tượng khác. Sau đó, phương thức trả về mảng đối tượng.

```
Public Function GetValue( _
  ByVal strSQL As String) As Object()
    ' Khai báo và khởi tạo đối tượng DataTable
    Dim myDT As New DataTable
    ' Khai báo mảng đối tượng
    Dim myObj() As Object
    ' Khai báo SqlDataAdapter
    Dim adData As SqlDataAdapter
    Try
      ' Khởi tạo đối tượng SqlDataAdapter
      adData = New SqlDataAdapter(strSQL, myCon)
      ' Điền dữ liệu vào đối tượng DataTable
      adData.Fill(myDT)
      ' Lấy ra số mẩu tin
      Dim row As Integer = myDT.Rows.Count
    ' Khai báo mảng kiểu object
      ReDim myObj(row)
      ' Lấy số cột dữ liệu
      Dim Col As Integer= myDT.Columns.Count - 1
      ' Khai báo mảng đối tượng để lấy ra tên cột dữ liệu
      Dim myObjs(col) As Object
      Dim i As Integer = 0
      ' Phần tử đầu tiên của mảng trả về là một mảng object
      For i = 0 To myObjs.Length - 1
        myObjs(i) = myDT.Columns(i).ColumnName
      Next
      ' Gán mảng object vào phần tử đầu tiên của mảng trả về
      myObj(0) = myObjs
      i = 1
      ' Duyệt trên từng hàng dữ liệu
      For Each dr As DataRow In myDT.Rows
        ' Nạp mảng object lấy từ thuộc tính ItemArray vào
        ' từng phần tử của mảng trả về
        myObj(i) = dr.ItemArray
        i += 1
```

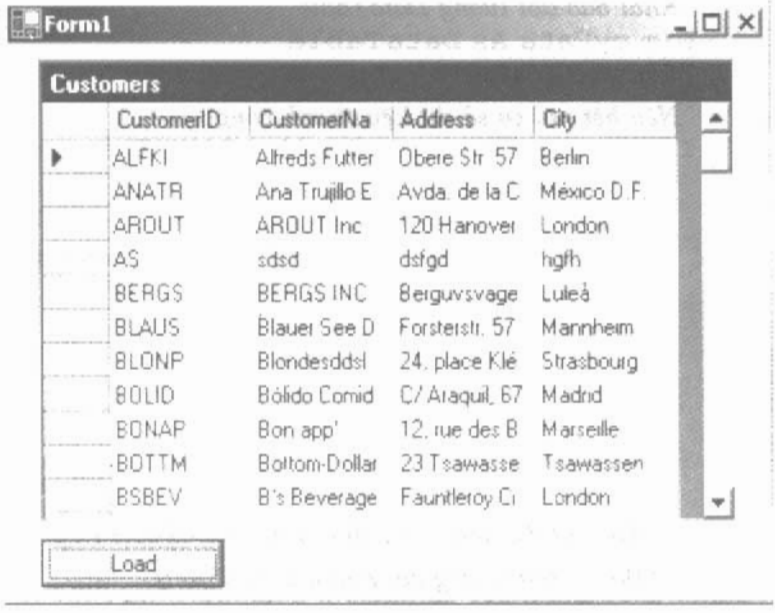
```
            Next
            strError = "OK"
        Catch ex As Exception
            strError = ex.Message
        Finally

        End Try
        Return myObj
    End Function
```

Sau đó, khai báo để gọi phương thức GetValue trong biến cố Click của nút Load.

```
Private Sub btnLoad_Click( _
  ByVal sender As System.Object, _
  ByVal e As System.EventArgs) _
  Handles btnLoad.Click
    dgData.CaptionText = "Customers"
    ' Khai báo và khởi tạo đối tượng clsDatabase
    Dim cls As New clsDatabase
    ' Khai báo đối tượng DataTable
    Dim myData As DataTable
    Dim strSQL As String
    ' Nếu kết nối cơ sở dữ liệu thành công
    If cls.OpenConnection() = "OK" Then
      ' Khởi tạo đối tượng DataTable
      myData = New DataTable
      strSQL = "Select * from tblCustomers"
      ' Khai báo mảng kiểu object
      Dim Objs() As Object
      ' Khai báo đối tượng DataRow
      Dim dr As DataRow
      ' Gọi phương thức GetValue
      Objs = cls.GetValue(strSQL)
      ' Tạo cột dữ liệu trong đối tượng DataTable từ phần tử
      ' thứ 0 (tương ứng với mảng kiểu object)
      For Each Obj As Object In Objs(0)
        myData.Columns.Add(Obj.ToString)
      Next
      ' Duyệt qua từng phần tử của mảng Objs
```

```
        Dim i As Integer
        For i = 1 To Objs.Length - 1
            ' Thêm mới một hàng
            dr = myData.NewRow
            ' Chuyển mỗi phần tử kiểu object thành mảng kiểu
            ' object, rồi gán vào đối tượng DataTow bằng thuộc
            ' tính ItemArray
            dr.ItemArray = CType(Objs(i), Object())
            ' Thêm đối tượng DataRow vào đối tượng DataTable
            myData.Rows.Add(dr)
        Next
        Me.dgData.DataSource = myData
    End If
End Sub
```

Khi thực thi chương trình, nếu người sử dụng nhấn nút Load, lập tức danh sách khách hàng trình bày trên điều khiển DataGrid như hình 28-4.



Hình 28-4: *Sử dụng mảng object*

2. ĐỐI TƯỢNG DATACOLUMN

1. Thiết kế *Form* cho phép người sử dụng nhập dữ liệu trên các điều khiển, sau đó trình bày dữ liệu đó trên điều khiển *DataGrid* cùng với số tự động tăng từ số kế tiếp. Khi người sử dụng nhấn nút *Save*, bạn lưu dữ liệu đó xuống cơ sở dữ liệu nguồn.

Bằng cách sử dụng đối tượng DataColumn, bạn khai báo phương thức DefineTable với 4 cột dữ liệu, cột SupplierID là cột số tự động, bốn cột còn lại là CompanyName, Address, City và Country như sau:

```
Sub DefineTable()
    ' Khởi tạo đối tượng DataTable
    myTable = New DataTable
    ' Khai báo đối tượng DataColumn
    Dim dc As DataColumn
    ' Khởi tạo đối tượng DataColumn với tên SupplierID
    dc = New DataColumn("SupplierID")
    ' Khai báo thuộc tính cho cột
    With dc
       .DataType = Type.GetType("System.Int32")
       .AutoIncrement = True
       ' Số tự động là số kế tiếp so với dữ liệu đang có
       .AutoIncrementSeed=myData.Rows.Count + 1
       .AutoIncrementStep = 1
    End With
    ' Thêm đối tượng DataColumn vào đối tượng DataTable
    myTable.Columns.Add(dc)
    ' Khởi tạo đối tượng DataColumn
    dc = New DataColumn
    ' Khai báo thuộc tính cho cột CompanyName
    With dc
       .ColumnName = "CompanyName"
       .DataType = Type.GetType("System.String")
       .Unique = True
       .AllowDBNull = False
       .MaxLength = 30
    End With
    ' Thêm đối tượng DataColumn vào đối tượng DataTable
    myTable.Columns.Add(dc)
```

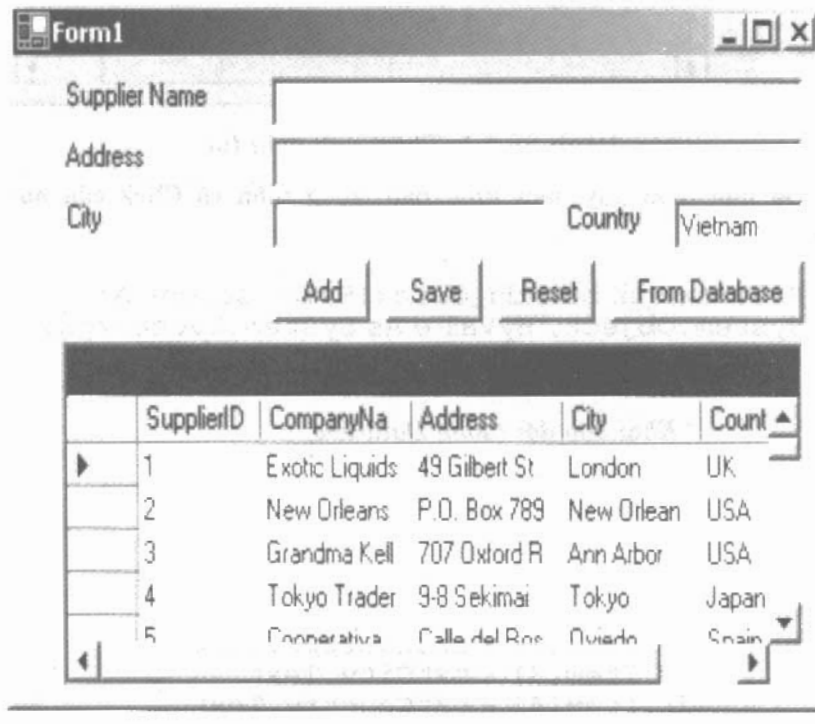
```
    'Khởi tạo đối tượng DataColumn
    dc = New DataColumn("Address")
    'Khai báo thuộc tính cho cột Address
    With dc
       .DataType = Type.GetType("System.String")
       .MaxLength = 50
    End With
    'Thêm đối tượng DataColumn vào đối tượng DataTable
    myTable.Columns.Add(dc)
    'Khởi tạo đối tượng DataColumn có tên City
    dc = New DataColumn("City")
    'Khai báo thuộc tính cho cột City
    With dc
       .DataType = Type.GetType("System.String")
       .MaxLength = 20
       .DefaultValue = "Ho Chi Minh"
    End With
    'Thêm đối tượng DataColumn vào đối tượng DataTable
    myTable.Columns.Add(dc)
    'Khởi tạo đối tượng DataColumn là cột Country
    dc = New DataColumn("Country")
    'Khai báo thuộc tính cho cột Country
    With dc
       .DataType = Type.GetType("System.String")
       .MaxLength = 20
       .DefaultValue = "Vietnam"
    End With
    'Thêm đối tượng DataColumn vào đối tượng DataTable
    myTable.Columns.Add(dc)
  End Sub
```

Trong đó, dữ liệu có trong đối tượng myData được lấy ra từ phương thức LoadData như sau:

```
Sub LoadData()
    If cls.OpenConnection() = "OK" Then
    'Sử dụng đối tượng DataTable khác
      myData = New DataTable
    'Định nghĩa phát biểu SQL
      strSQL = "Select SupplierID,CompanyName, "
```

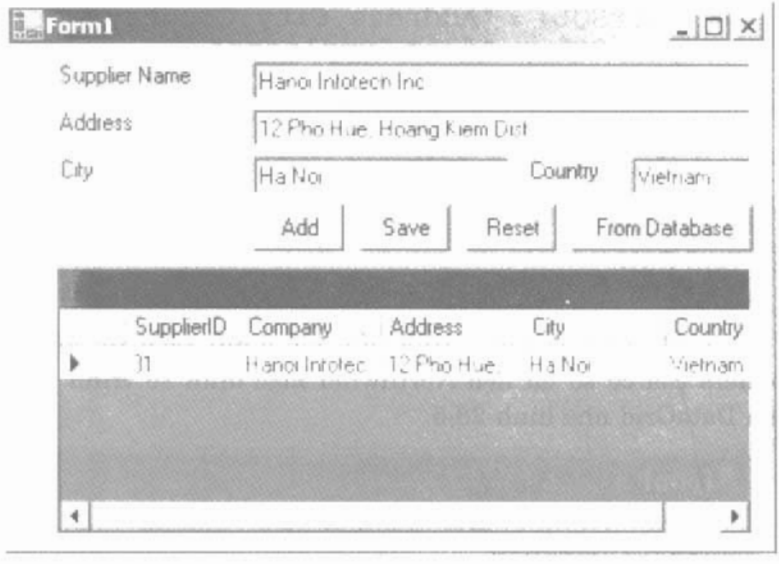
```
        strSQL+ = "Address,City,Country "
        strSQL += "from Suppliers"
        ' Gọi phương thức GetValue
          cls.GetValue(myData, strSQL)
          ' Trình bày dữ liệu
          Me.dgData.DataSource = myData
        End If
    End Sub
```

Khi thực thi chương trình, lập tức danh sách dữ liệu trong bảng Suppliers của cơ sở dữ liệu Northwind mặc định sẽ trình bày trên điều khiển DataGrid như hình 28-5.



Hình 28-5: *Danh sách nhà cung cấp*

Nếu người sử dụng nhập thông tin trên các điều khiển TextBox và nhấn nút OK, lập tức thông tin đó sẽ trình bày trên điều khiển DataGrid tương tự như hình 28-5-1.

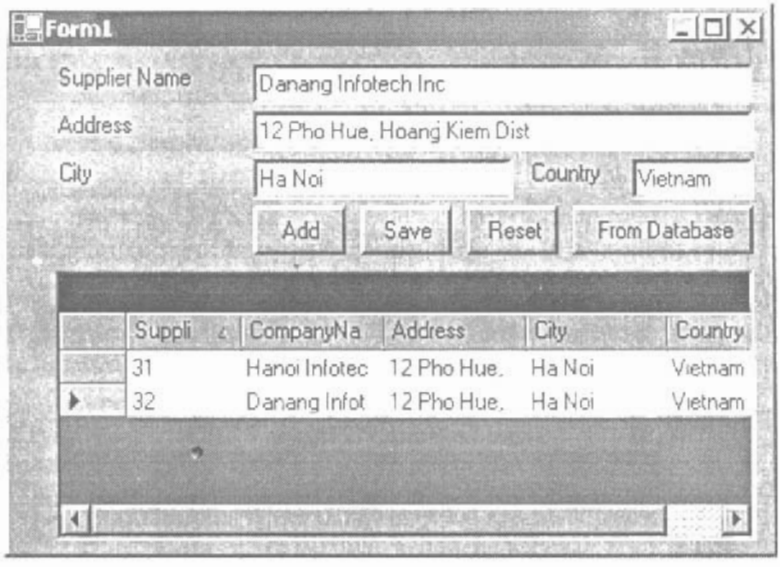


Hình 28-5-1: *Thêm mới mẩu tin*

Để làm điều này, bạn khai báo trong biến cố Click của nút Add như sau:

```
Private Sub btnAdd_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles btnAdd.Click
    If txtName.Text <> "" Then
        ' Khai báo đối tượng DataRow
        Dim dr As DataRow
        ' Thêm mới một hàng
        dr = myTable.NewRow
        ' Gán giá trị tương ứng trừ cột số tự động
        dr.Item(1) = txtName.Text
        dr.Item(2) = txtAddress.Text
        dr.Item(3) = txtCity.Text
        dr.Item(4) = txtCountry.Text
        ' Thêm đối tượng DataRow và đối tượng DataTable
        myTable.Rows.Add(dr)
        ' Trình bày dữ liệu
        Me.dgData.DataSource = myTable
    End If
End Sub
```

Lưu ý, tổng số mẩu tin trong bảng Suppliers hiện là 30, chính vì vậy khi bạn thêm mới mẩu tin lập tức cột SupplierID sẽ là 31.



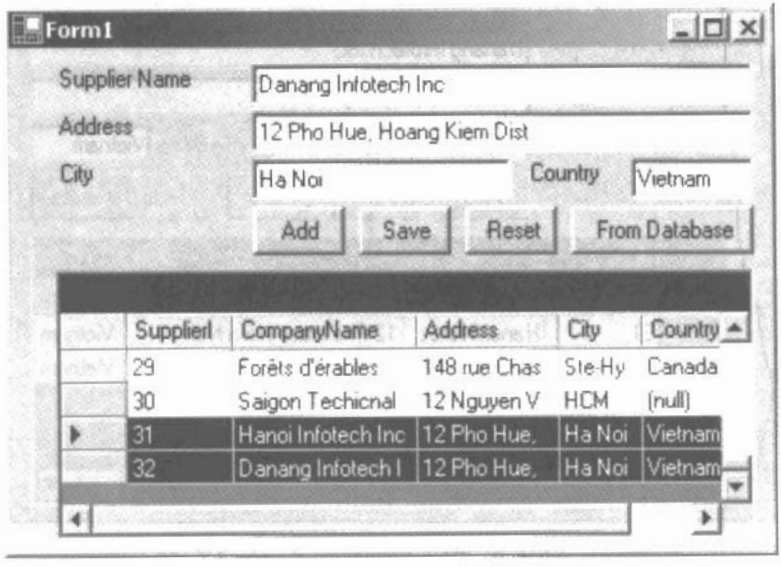
Hình 28-5-2: *Thêm mới mẩu tin kế tiếp*

Nếu tiếp tục thêm mới mẩu tin, lập tức giá trị tại cột số tự động là 32 tương tự như hình 28-5-2.

Để cho phép lưu trữ hai mẩu tin trên vào bảng Suppliers, bạn khai báo đoạn chương trình như sau trong biến cố Click của nút Save.

```
Private Sub btnSave_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles btnSave.Click
    If Not myTable Is Nothing Then
        'Gọi phương thức Update của đối tượng clsDatabase
        If cls.UpdateData(myTable, strSQL) _
            = "OK" Then
            MsgBox("Updated")
            'Đọc dữ liệu từ cơ sở dữ liệu
            Call btnLoad_Click(sender, e)
        End If
    End If
End Sub
```

Giả sử, bạn nhấn nút Save trong trường hợp này, lập tức hai mẩu tin vừa thêm được lưu và trình bày trên điều khiển DataGrid như hình 28-5-3.



Hình 28-5-3: *Lưu dữ liệu thành công*

Chú ý, để thực hiện thành công các ví dụ trên, bạn cần khai báo 4 biến sử dụng chung trên phần đầu của Class.

```
Dim cls As New clsDatabase
Dim myData As DataTable
Dim myTable As DataTable
Dim strSQL As String
```

2. Tạo ứng dụng *Windows Forms*, cho phép người sử dụng nhập tên *Table*, tên cột và chọn kiểu dữ liệu cùng với các thuộc tính đơn giản khác, sau đó tạo ra đối tượng *Table* này trong cơ sở dữ liệu *Northwind*.

Để thực hiện ví dụ này, trước tiên bạn khai báo trong biến cố Load của Form như sau:

```
Private Sub Form1_Load(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles MyBase.Load
    ' Khởi tạo đối tượng DataTable
```

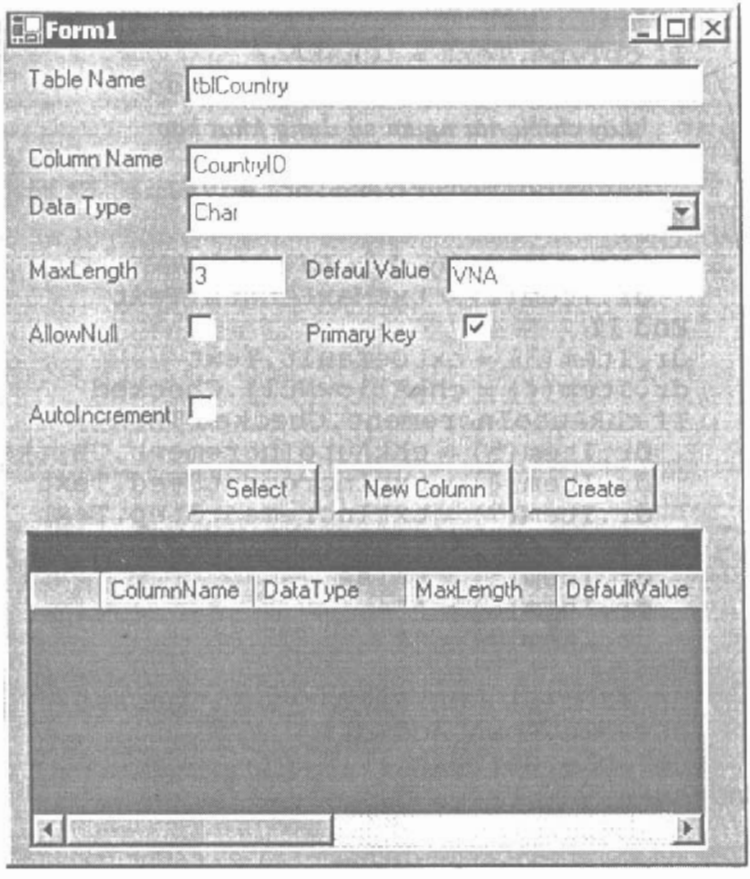
```
        dtTable = New DataTable
        'Khai báo tên cột dữ liệu
        dtTable.Columns.Add("ColumnName")
        dtTable.Columns.Add("DataType")
        dtTable.Columns.Add("MaxLength")
        dtTable.Columns.Add("DefaultValue")
        dtTable.Columns.Add("AllowNull")
        dtTable.Columns.Add("AutoIncrement")
        dtTable.Columns.Add("AutoIncrementSeed")
        dtTable.Columns.Add("AutoIncrementStep")
        dtTable.Columns.Add("PrimaryKey")
        'Trình bày dữ liệu trên điều khiển DataGrid
        dgTable.DataSource = dtTable
        'Liệt kê danh sách kiểu dữ liệu
        cbType.DataSource = DefineType()
        cbType.DisplayMember = "TypeName"
        cbType.ValueMember = "TypeValue"
    End Sub
```

Trong đó, phương thức điền danh sách kiểu dữ liệu vào điều khiển ComboBox được trình bày như sau:

```
    Function DefineType() As DataTable
        'Khai báo và khởi tạo đối tượng DataTable
        Dim dtType As New DataTable
        'Khai báo đối tượng DataColumn
        Dim Col As DataColumn
        'Khởi tạo đối tượng DataColumn
        Col = New DataColumn("TypeValue")
        'Định nghĩa kiểu dữ liệu
        Col.DataType = _
          Type.GetType("System.String")
        'Thêm đối tượng DataColumn và đối tượng DataTable
        dtType.Columns.Add(Col)
        'Khởi tạo đối tượng DataColumn
        Col = New DataColumn("TypeName")
        'Định nghĩa kiểu dữ liệu
        Col.DataType = _
          Type.GetType("System.String")
        'Thêm đối tượng DataColumn và đối tượng DataTable
```

```
        dtType.Columns.Add(Col)
        ' Khai báo đối tượng DataRow
        Dim dr As DataRow
        ' Thêm mới hàng
        dr = dtType.NewRow()
        ' Khai báo giá trị
        dr.Item(0) = ""
        dr.Item(1) = ""
        ' Thêm đối tượng DataRow và đối tượng DataTable
        dtType.Rows.Add(dr)
        dr = dtType.NewRow()
        dr.Item(0) = "4"
        dr.Item(1) = "Int"
        dtType.Rows.Add(dr)
        dr = dtType.NewRow()
        dr.Item(0) = "2"
        dr.Item(1) = "Tinyint"
        dtType.Rows.Add(dr)
        dr = dtType.NewRow()
        dr.Item(0) = "10"
        dr.Item(1) = "Char"
        dtType.Rows.Add(dr)
        dr = dtType.NewRow()
        dr.Item(0) = "50"
        dr.Item(1) = "Varchar"
        dtType.Rows.Add(dr)
        dr = dtType.NewRow()
        dr.Item(0) = "16"
        dr.Item(1) = "SmallDateTime"
        dtType.Rows.Add(dr)
        Return dtType
    End Function
```

Khi thực thi chương trình, giao diện Form cho phép người sử dụng định nghĩa cấu trúc Table như hình 28-6.



Hình 28-6: *Tạo bảng dữ liệu*

Để cho phép người sử dụng khai báo cột dữ liệu và trình bày chúng dưới điều khiển DataGrid, bạn khai báo trong biến cố Click của nút Select như sau:

```
Private Sub btnSelect_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles btnSelect.Click
    ' Khai báo và khởi tạo đối tượng DataRow
    Dim dr As DataRow
    ' Thêm mới hàng dữ liệu
    dr = dtTable.NewRow()
    ' Khai báo các thuộc tính
    dr.Item(0) = txtColumn.Text
    dr.Item(1) = Convert.ToString(cbType.Text)
```

```
        'Nếu kiểu chuỗi
        If cbType.Text = "Char" _
        Or cbType.Text = "Varchar" Then
            'Lấy chiều dài người sử dụng khai báo
            dr.Item(2) = _
              CType(cbType.SelectedValue, String)
        Else
            'Ngược lại thì lấy chiều dài mặc định
            dr.Item(2) = txtMaxLength.Text
        End If
        dr.Item(3) = txtDefault.Text
        dr.Item(4) = chkAllowNull.Checked
        If chkAutoIncrement.Checked Then
            dr.Item(5) = chkAutoIncrement.Checked
            dr.Item(6) = txtIncrementSeed.Text
            dr.Item(7) = txtIncrementStep.Text
        Else
            dr.Item(5) = False
            dr.Item(6) = ""
            dr.Item(7) = ""
        End If
        dr.Item(8) = chkPrimarykey.Checked
        dtTable.Rows.Add(dr)
        dgTable.Refresh()
    End Sub
```

Kế đến, bạn khai báo đoạn chương trình cho phép người sử dụng tạo bảng dữ liệu trong cơ sở dữ liệu SQL Server như sau:

```
    Private Sub btnCreate_Click(ByVal sender As
    System.Object, ByVal e As System.EventArgs)
    Handles btnCreate.Click
        'Khai báo biến SQL
        Dim strTable As String
        'Khai báo phát biểu CREATE TABLE
        strTable = _
          "create table " + txtTableName.Text
        strTable += "("
        'Duyệt trên bảng dữ liệu
        For Each dr As DataRow In dtTable.Rows
            strTable += dr.Item(0) + " "
            strTable += dr.Item(1) + "("
            strTable += dr.Item(2) + ") "
```

```
            If dr.Item(3) <> "" Then
              strTable += " default "
              If dr.Item(1) = "Char" Or _
              dr.Item(1) = "Varchar" Then
                strTable += "' " + dr.Item(3) + "'"
              Else
                strTable += dr.Item(3)
              End If
            End If
            If dr.Item(4) Then
              strTable += " Null "
            End If
            If dr.Item(5) Then
              strTable += " identity( " _
               + dr.Item(6) + "," + dr.Item(7) + ")"
            End If
            If dr.Item(8) Then
              strTable += " primary key"
            End If
            strTable += ","
          Next
          strTable = x.Left(strTable, _
              strTable.Length - 1)
          strTable += ")"
          ' Khai báo và khởi tạo đối tượng clsDatabase
          Dim cls As New clsDatabase
          ' Gọi phương thức ExecuteSQL để tạo bảng dữ liệu
          cls.ExecuteSQL(strTable)
      End Sub
```

Lưu ý, phương thức ExecuteSQL được định nghĩa trong lớp clsDatabase có cấu trúc:

```
      Public Function ExecuteSQL( _
        ByVal strSQL As String) As String
          Dim myCom As SqlCommand
          Try
            myCom = New SqlCommand(strSQL, myCon)
            myCom.ExecuteNonQuery()
            strError = "OK"
          Catch ex As Exception
            strError = ex.Message
          Finally
```

```
        End Try
        Return strError
    End Function
```

Ngoài ra, bạn cần khai báo để chọn giá trị mặc định mỗi khi người sử dụng nhấn vào nút New Column.

```
Private Sub btnNew_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles btnNew.Click
        txtTableName.Text = ""
        txtColumn.Text = ""
        cbType.SelectedValue = ""
        txtMaxLength.Text = ""
        txtDefault.Text = ""
        txtIncrementSeed.Text = "1"
        txtIncrementStep.Text = "1"
        chkAllowNull.Checked = True
        chkAutoIncrement.Checked = False
        chkPrimarykey.Checked = False
    End Sub
```

Chú ý, nếu người sử dụng chọn vào tùy chọn Primary key, lập tức tùy chọn Allow Null sẽ được loại bỏ. Tương tự như vậy cho trường hợp người sử dụng chọn vào tùy chọn Allow Null, lập tức tùy chọn Primary key sẽ bị loại bỏ.

Để làm điều này, bạn khai báo trong biến cố CheckedChanged của hai điều khiển CheckBox này như sau:

```
Private Sub chkAllowNull_CheckedChanged(ByVal
sender As System.Object, ByVal e As
System.EventArgs) Handles
chkAllowNull.CheckedChanged
        If chkAllowNull.Checked Then
          chkPrimarykey.Checked = False
        End If
    End Sub

Private Sub chkPrimarykey_CheckedChanged(ByVal
sender As System.Object, ByVal e As
System.EventArgs) Handles
chkPrimarykey.CheckedChanged
        If chkPrimarykey.Checked Then
```

```
            chkAllowNull.Checked = False
        End If
    End Sub
```

Ngoài ra, bạn cũng khai báo để hiển thị hai phần nhập số bắt đầu và số nhảy khi người sử dụng chọn vào tùy chọn số tự động như sau:

```
Private Sub
chkAutoIncrement_CheckedChanged(ByVal sender
As System.Object, ByVal e As System.EventArgs)
Handles chkAutoIncrement.CheckedChanged
    If chkAutoIncrement.Checked Then
      GroupBox1.Visible = True
    Else
      GroupBox1.Visible = False
    End If
End Sub
```

3. ĐỐI TƯỢNG DATARELATION

1. Thiết kế *Form* cho phép người sử dụng liệt kê danh sách đơn đặt hàng, mỗi khi chọn vào một đơn đặt hàng thì danh sách đơn đặt hàng chi tiết xuất hiện.

Bằng cách khai báo phương thức GetValue trong lớp clsDatabase với cấu trúc như sau:

```
Public Function GetValue( _
  ByRef myDS As DataSet, _
  ByVal strSQL As String) As String
    ' Khai báo đối tượng SqlDataAdapter
    Dim adData As SqlDataAdapter
    Try
      ' Khởi tạo đối tượng SqlDataAdapter
      adData = New SqlDataAdapter(strSQL, myCon)
      ' Điền dữ liệu vào đối tượng DataSet
      adData.Fill(myDS, strSQL)
      strError = "OK"
    Catch ex As Exception
      strError = ex.Message
    Finally
```

```
        End Try
        Return strError
    End Function
```

Kế đến, bạn khai báo để nạp danh sách dữ liệu trong hai bảng Orders và Order Details vào đối tượng DataSet như sau:

```
Sub LoadData()
    Dim strSQL As String
    strSQL = "select * from Orders;"
    strSQL += "select * from [Order Details]"
    Dim cls As New clsDatabase
    If cls.OpenConnection = "OK" Then
      cls.GetValue(dsData, strSQL)
      cls.CloseConnection()
    End If
End Sub
```

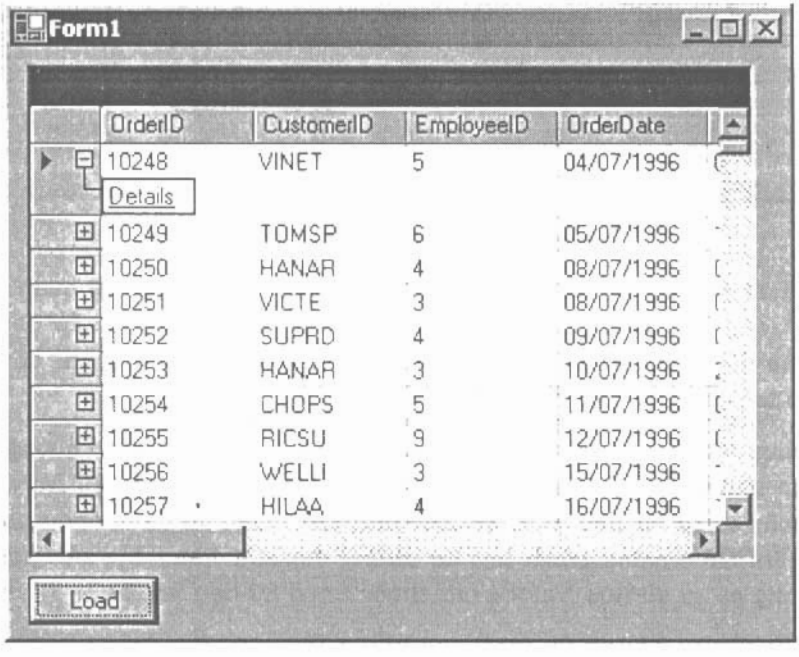
Sau khi gọi phương thức LoadData, đối tượng DataSet nắm giữ hai tập dữ liệu tương ứng của bảng Orders và Order Details. Bằng cách khai báo đối tượng DataRelation cùng với hai đối tượng DataColumn, bạn có thể tạo quan hệ giữa hai bảng này như sau:

```
Sub Relation()
    ' Khai báo đối tượng DataColumn
    Dim pkCol As DataColumn
    ' Gán đối tượng DataColumn với cột thứ nhất trong
    ' bảng Orders
    pkCol = dsData.Tables(0).Columns(0)
    ' Khai báo đối tượng DataColumn
    Dim fkCol As DataColumn
    ' Gán đối tượng DataColumn với cột thứ hai trong
    ' bảng Order Details
    fkCol = dsData.Tables(1).Columns(0)
    ' Tạo quan hệ giữa hai cột
    Dim rlData As New DataRelation( _
      "Details", pkCol, fkCol)
    ' Thêm quan hệ này vào đối tượng DataSet
    dsData.Relations.Add(rlData)
End Sub
```

Sau đó, bạn gọi hai phương thức này trong biến cố Click của nút Load.

```
Private Sub btnLoad_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles btnLoad.Click
    ' Khởi tạo đối tượng DataSet
    dsData = New DataSet
    ' Gọi phương thức LoadData
    LoadData()
    ' Gọi phương thức tạo quan hệ
    Relation()
    ' Điền dữ liệu vào điều khiển DataGrid
    DataGrid1.DataSource = dsData.Tables(0)
    ' Liệt kê chi tiết Order thứ nhất
    DataGrid1.Expand(0)
End Sub
```

Khi thực thi chương trình, nếu người sử dụng nhấn nút Load, lập tức danh sách mẩu tin trong bảng Orders xuất hiện như hình 28-7.

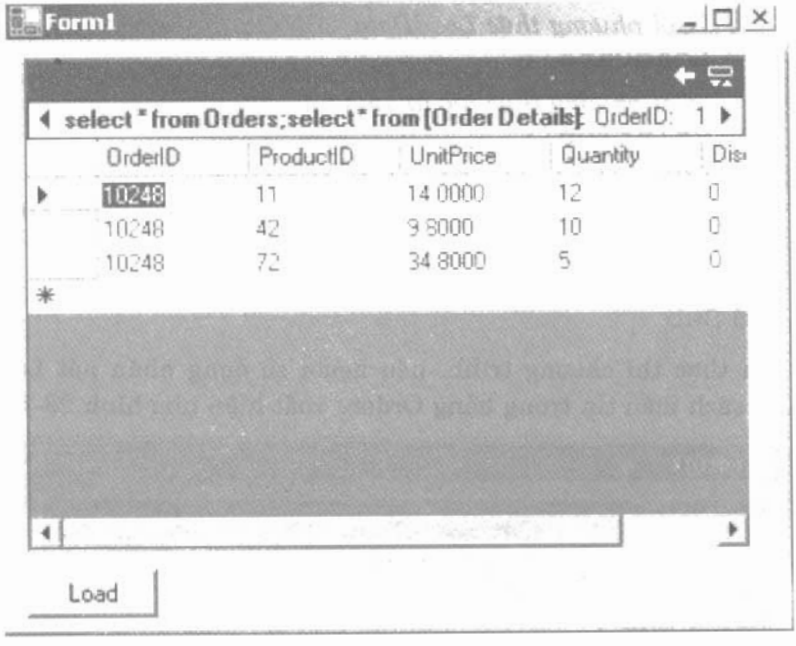


Hình 28-7: *Danh sách mẩu tin của bảng Orders*

Chú ý, trong ví dụ trên chúng ta có sử dụng đối tượng DataSet với tên dsData, bạn phải khai báo biến sử dụng chung này như sau

```
Dim dsData As DataSet
```

Khi người sử dụng chọn vào liên kết Details, lập tức danh sách chi tiết đơn đặt hàng xuất hiện tương tự như hình 28-7-1



Hình 28-7-1: *Chi tiết đơn đặt hàng*

Lưu ý, mặc dù hai bảng Orders và Order Details đã có quan hệ với nhau trong cơ sở dữ liệu Northwind, nhưng khi chúng ta nạp hai bảng này vào đối tượng DataSet thì chúng độc lập với nhau trừ khi bạn khai báo để tạo quan hệ cho chúng.

2. Bằng cách sử dụng điều khiển *DataGrid*, bạn viết đoạn chương trình cho phép liệt kê danh sách sản phẩm trong cơ sở dữ liệu *Northwind* thuộc cơ sở dữ liệu *SQL Server*, mỗi khi người sử dụng chọn một sản phẩm thì chương trình liệt kê những đơn đặt hàng của sản phẩm đó trong cơ sở dữ liệu *Northwind* thuộc cơ sở dữ liệu *Access*.

Tương tự như ví dụ trên, tuy nhiên trong trường hợp này hai bảng có quan hệ thuộc hai loại cơ sở dữ liệu khác nhau.

Để làm điều này, trước tiên bạn thêm Class vào Project và đặt tên là clsDatabase, rồi khai báo phương thức OpenConnection để mở kết nối cơ sơ dữ liệu SQL Server.

```
Public Function OpenConnection() As String
    ' Khai báo chuỗi kết nối cơ sở dữ liệu SQL Server
    psCon = "server=" + gsServer
    psCon += ";database=" + gsDatabase
    psCon += ";Integrated Security=SSPI"
    Try
    ' Khởi tạo đối tượng SqlConnection
      myCon = New SqlConnection(psCon)
      ' Mở kết nối cơ sở dữ liệu
      myCon.Open()
      strError = "OK"
    Catch eq As SqlException
      strError = eq.Message
    Catch ex As Exception
      strError = ex.Message
    End Try
    Return strError
End Function
```

Kế đến, bạn khai báo phương thức GetValue nhận tham biến là đối tượng DataTable. Bằng cách sử dụng đối tượng SqlDataAdapter, bạn có thể điền danh sách sản phẩm từ bảng Products vào đối tượng DataTable.

```
Public Function GetValue( _
  ByRef myDT As DataTable, _
  ByVal strSQL As String) As String
    Dim adData As SqlDataAdapter
    Try
      adData = New SqlDataAdapter(strSQL, myCon)
      adData.Fill(myDT)
      strError = "OK"
    Catch ex As Exception
      strError = ex.Message
    Finally

    End Try
    Return strError
End Function
```

Tương tự như vậy, bạn thêm Class vào Project và đặt tên clsAccess. Tiếp theo, bạn khai báo phương thức OpenConnection để mở kết nối cơ sở dữ liệu Access như sau:

```
Public Function OpenConnection() As String
    ' Khai báo chuỗi kết nối cơ sở dữ liệu Access
    psCon = "Provider="
    psCon += "Microsoft.Jet.OLEDB.4.0;"
    psCon += "Data Source=Northwind.mdb"
    Try
      ' Khởi tạo đối tượng OleDbConnection
      myCon = New OleDbConnection(psCon)
      ' Mở kết nối cơ sở dữ liệu Access
     myCon.Open()
      strError = "OK"
    Catch eq As OleDbException
      strError = eq.Message
    Catch ex As Exception
      strError = ex.Message
    End Try
    Return strError
End Function
```

Kế đến, bạn khai báo phương thức GetValue nhận tham biến là đối tượng DataTable. Bằng cách sử dụng đối tượng OleDbDataAdapter, bạn có thể điền danh sách sản phẩm từ bảng Order Details trong cơ sở dữ liệu Northwind.mdb vào đối tượng DataTable.

```
Public Function GetValue( _
  ByRef myDT As DataTable, _
  ByVal strSQL As String) As String
    Dim adData As OleDbDataAdapter
    Try
      adData = _
        New OleDbDataAdapter(strSQL, myCon)
      adData.Fill(myDT)
      strError = "OK"
    Catch ex As Exception
      strError = ex.Message
    Finally

    End Try
    Return strError
End Function
```

Trở lại Form1, bạn khai báo phương thức LoadData để nạp danh sách dữ liệu trong hai bảng Orders và Order Details vào đối tượng DataSet như sau:

```
Sub LoadData()
    Dim strSQL As String
    ' Khai báo và khởi tạo đối tượng DataTable
    Dim dtSQL As New DataTable
    ' Định nghĩa phát biểu SQL
    strSQL = "select ProductName, "
    strSQL +=" ProductID from Products"
    ' Khai báo và khởi tạo đối tượng clsDatabase
    Dim cls As New clsDatabase
    ' Nếu kết nối cơ sở dữ liệu SQL Server thành công
    If cls.OpenConnection = "OK" Then
       ' Điền dữ liệu bảng Products vào đối tượng
       ' DataTable thứ nhất
       cls.GetValue(dtSQL, strSQL)
       cls.CloseConnection()
    End If
    ' Định nghĩa phát biểu SQL
    strSQL = "select * from [Order Details]"
    ' Khai báo và khởi tạo đối tượng DataTable
    Dim dtAccess As New DataTable
    ' Khai báo và khởi tạo đối tượng clsAccess
    Dim clsA As New clsAccess
    ' Nếu kết nối cơ sở dữ liệu Access thành công
    If clsA.OpenConnection = "OK" Then
       ' Điền dữ liệu bảng Order Details vào đối tượng
       ' DataTable thứ hai
       clsA.GetValue(dtAccess, strSQL)
       clsA.CloseConnection()
    End If
    ' Thêm đối tượng DataTable thứ nhất vào đối tượng DataSet
    dsData.Tables.Add(dtSQL)
    ' Thêm đối tượng DataTable thứ hai vào đối tượng DataSet
    dsData.Tables.Add(dtAccess)
  End Sub
```

Sau khi gọi phương thức LoadData, đối tượng DataSet đang nắm giữ hai đối tượng DataTable ứng với hai tập dữ liệu của bảng Products (thuộc cơ sở dữ liệu SQL Server) và Order Details (thuộc cơ sở dữ liệu Access).

Bằng cách khai báo đối tượng DataRelation cùng với hai đối tượng DataColumn, bạn có thể tạo quan hệ giữa hai bảng này như sau:

```
Sub Relation()
    ' Khai báo đối tượng DataColumn
    Dim pkCol As DataColumn
    ' Gán đối tượng DataColumn với cột thứ nhất trong
    ' bảng Products
    pkCol = dsData.Tables(0).Columns(1)
    ' Khai báo đối tượng DataColumn
    Dim fkCol As DataColumn
    ' Gán đối tượng DataColumn với cột thứ hai trong
    ' bảng Order Details
    fkCol = dsData.Tables(1).Columns(1)
    ' Tạo quan hệ giữa hai cột
    Dim rlData As New DataRelation( _
       "Details", pkCol, fkCol)
    ' Thêm quan hệ này vào đối tượng DataSet
    dsData.Relations.Add(rlData)
End Sub
```

Sau đó, bạn gọi hai phương thức này trong biến cố Click của nút Load.

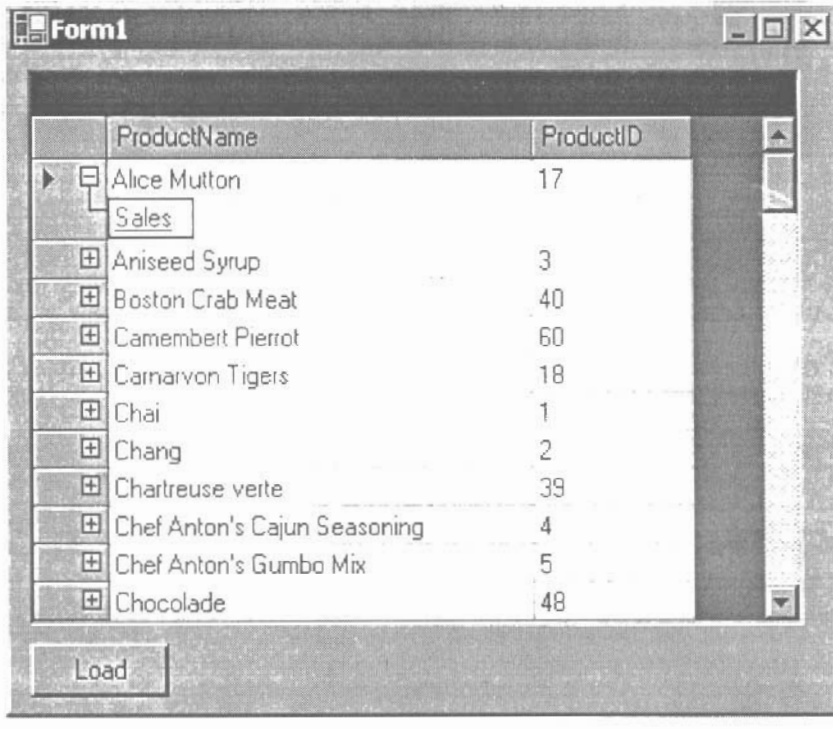
```
Private Sub btnLoad_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles btnLoad.Click
    ' Khởi tạo đối tượng DataSet
    dsData = New DataSet
    ' Gọi phương thức LoadData
    LoadData()
    ' Gọi phương thức tạo quan hệ
    Relation()
    ' Điền dữ liệu vào điều khiển DataGrid
```

```
        DataGrid1.DataSource = dsData.Tables(0)
        ' Liệt kê chi tiết Order thứ nhất
        DataGrid1.Expand(0)
    End Sub
```

Lưu ý, trong ví dụ trên chúng ta có sử dụng đối tượng DataSet với tên dsData, bạn phải khai báo biến sử dụng chung này như sau:

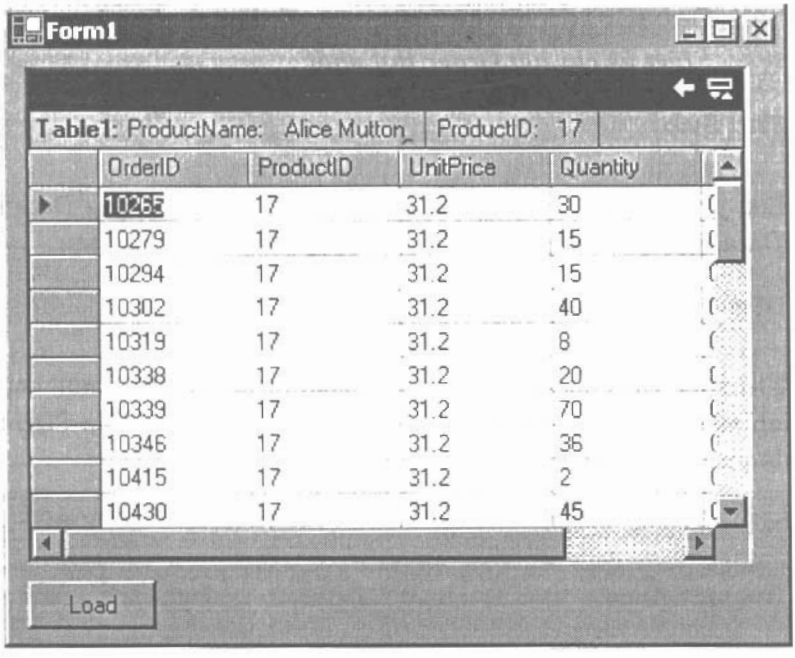
```
Dim dsData As DataSet
```

Khi thực thi chương trình, nếu người sử dụng nhấn nút Load, lập tức danh sách mẩu tin trong bảng Products của cơ sở dữ liệu SQL Server xuất hiện như hình 28-8.



Hình 28-8: *Danh sách sản phẩm*

Khi người sử dụng chọn vào liên kết Sales, lập tức danh sách chi tiết đơn đặt hàng xuất hiện tương tự như hình 28-8-1.



Hình 28-8-1: *Chi tiết đơn đặt hàng*

Chuyên đề 21:

ĐIỀU KHIỂN DATAGRID VÀ DATABINDINGS

1. ĐIỀU KHIỂN DATAGRID

1. Thiết kế *Form* cho phép người sử dụng liệt kê danh sách khách hàng với định dạng tùy ý. Mỗi khi chọn vào một khách hàng thì danh sách đơn đặt hàng xuất hiện với định dạng khác với định dạng danh sách khách hàng. Tiếp tục chọn vào đơn đặt hàng thì danh sách các sản phẩm của đơn đặt hàng đó xuất hiện.

Để thực hiện ví dụ này, trước tiên bạn khai báo phương thức GetValue trong lớp clsDatabase. Phương thức này nhận tham biến là đối tượng DataSet và biến chứa 3 phát biểu Select tương ứng với 3 bảng dữ liệu Customers, Orders và Order Details.

```
Public Function GetValue( _
  ByRef myDS As DataSet, _
  ByVal strSQL As String) As String
    Dim adData As SqlDataAdapter
    Try
      adData = New SqlDataAdapter(strSQL, myCon)
      adData.Fill(myDS, strSQL)
      strError = "OK"
    Catch ex As Exception
      strError = ex.Message
    Finally

    End Try
    Return strError
End Function
```

Kế đến, bạn khai báo phương thức trong lớp Form1 dùng để gọi phương thức GetValue với 3 phát biểu SQL như sau:

```
Sub LoadData()
    Dim strSQL As String
    ' Khai báo 3 phát biểu SQL
```

```
        strSQL = "select CustomerID,CompanyName,"
        strSQL += "Address,City from Customers;"
        strSQL += "select * from Orders;"
        strSQL += "select * from [Order Details];"
        ' Khai báo và khởi tạo đối tượng clsDatabase
        Dim cls As New clsDatabase
        ' Nếu kết nối cơ sở dữ liệu thành công
        If cls.OpenConnection = "OK" Then
            ' Gọi phương thức GetValue
            cls.GetValue(dsData, strSQL)
            cls.CloseConnection()
        End If
    End Sub
```

Để tạo quan hệ giữa các bảng này với nhau trong đối tượng DataSet, bạn khai báo phương thức Relation như sau:

```
    Sub Relation()
        ' Khai báo cột tương ứng với cột CustomerID trong bảng
        ' Customers
        Dim pkCol1 As DataColumn
        pkCol1 = dsData.Tables(0).Columns(0)
        ' Khai báo cột tương ứng với cột CustomerID trong bảng
        ' Orders
        Dim fkCol1 As DataColumn
        fkCol1 = dsData.Tables(1).Columns(1)
        ' Khai báo và khởi tạo đối tượng DataRelation
        Dim rlData As DataRelation
        ' Tạo quan hệ giữa bảng Customers và Orders
        rlData = New DataRelation("Orders", _
        pkCol1, fkCol1)

        ' Thêm quan hệ giữa bảng Customers và Orders vào đối
        ' tượng DataSet
        dsData.Relations.Add(rlData)
        ' Khai báo cột tương ứng với cột OrderID trong bảng
        ' Orders
        Dim pkCol2 As DataColumn
        pkCol2 = dsData.Tables(1).Columns(0)
```

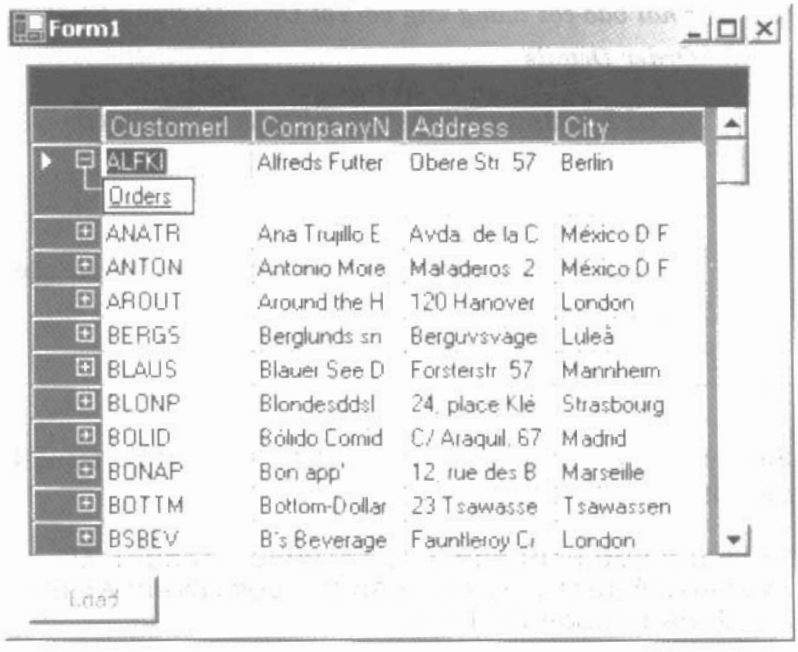
```
        ' Khai báo cột tương ứng với cột OrderID trong bảng
        ' Order Details
        Dim fkCol2 As DataColumn
        fkCol2 = dsData.Tables(2).Columns(0)
        ' Khởi tạo đối tượng DataRelation
        rlData = New DataRelation("Details", _
          pkCol2, fkCol2)
        ' Thêm quan hệ giữa bảng Orders và Order Details vào đối
        ' tượng DataSet
        dsData.Relations.Add(rlData)
    End Sub
```

Sau đó, bạn gọi phương thức LoadData và Relation trong biến cố Click của nút Load như sau:

```
    Private Sub btnLoad_Click(ByVal sender As
    System.Object, ByVal e As System.EventArgs)
    Handles btnLoad.Click
        ' Khởi tạo đối tượng DataSet
        dsData = New DataSet
        ' Điền dữ liệu vào DataGrid
        LoadData()
        ' Tạo quan hệ
        Relation()
        ' Trình bày danh sách khách hàng
        DataGrid1.DataSource = dsData.Tables(0)
        DataGrid1.Expand(0)
        ' Định dạng cho dữ liệu của bảng
        ApplyStyle(0)
        ApplyStyle(1)
        ApplyStyle(2)
        btnLoad.Enabled = False
    End Sub
```

Trong đó, phương thức ApplyStyle sẽ định dạng DataGrid tùy thuộc vào bảng dữ liệu mà bạn trình bày.

Khi thực thi chương trình, lập tức danh sách khách hàng trình bày với định dạng màu như hình 29-1.



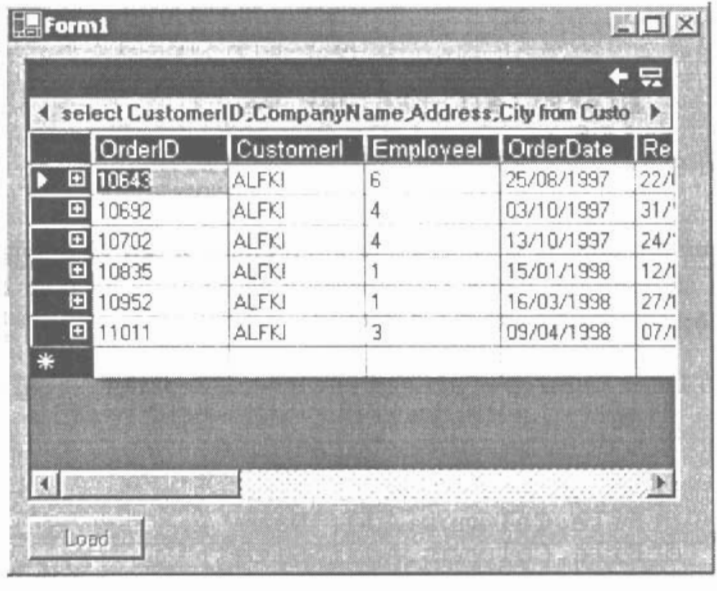
Hình 29-1: *Định dạng danh sách khách hàng*

Chẳng hạn, ứng với tham trị i bằng 0 là định dạng cho danh sách khách hàng, 1 định dạng cho danh sách đơn đặt hàng và 2 là định dạng cho danh sách chi tiết đơn đặt hàng.

```
Sub ApplyStyle(ByVal i As Integer)
    ' Khai báo và khởi tạo đối tượng DataGridTableStyle
    Dim myStyle As New DataGridTableStyle
    With myStyle
        .HeaderForeColor = Color.WhiteSmoke
        ' Tùy vào dữ liệu trình bày
        Select Case i
            Case 0
                ' Danh sách khách hàng
                .HeaderBackColor = Color.Green
                .SelectionForeColor = Color.Olive
            Case 1
                ' Danh sách đơn đặt hàng
                .HeaderBackColor = Color.DarkBlue
                .SelectionForeColor = Color.Maroon
            Case 2
```

```
            ' Danh sách đơn đặt hàng chi tiết
            .HeaderBackColor = Color.Black
            .SelectionForeColor = Color.LightBlue
          Case Else
            .HeaderBackColor = Color.Green
            .SelectionForeColor = Color.Olive
        End Select
        .HeaderFont = New Font("Arial", 10, _
          FontStyle.Regular)
        .ReadOnly = True
        .GridLineColor = Color.HotPink
        .SelectionBackColor = Color.Gray
      End With

      ' Khai báo ứng với bảng dữ liệu
      myStyle.MappingName = _
      dsData.Tables(i).TableName

    ' Gán định dạng cho điều khiển DataGrid
    DataGrid1.TableStyles.Add(myStyle)
End Sub
```

Nếu người sử dụng nhấn vào liên kết Orders của khách hàng nào đó, lập tức danh sách đơn đặt hàng của khách hàng đó liệt kê với định dạng như hình 29-1-1.



Hình 29-1-1: *Liệt kê danh sách đơn đặt hàng*

2. Viết ứng dụng *Windows Forms* dùng để liệt kê tên tập tin, dung lượng, ngày tạo ra của thư mục chỉ định. Sau đó điền dữ liệu đó vào điều khiển *DataGrid* với định dạng nào đó.

Đối với trường hợp này, bạn khai báo sử dụng không gian tên là System.IO.

```
Imports System.IO
```

Kế đến, bạn khai báo đoạn chương trình dùng đối tượng FolderBrowserDialog cho phép người sử dụng chọn thư mục như sau:

```
Private Sub btnFolder_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles btnFolder.Click
    ' Khai báo và khởi tạo đối tượng FolderBrowserDialog
    Dim fdFile As New FolderBrowserDialog
    ' Chọn thư mục mặc định
    fdFile.RootFolder = _
    Environment.SpecialFolder.Desktop
    ' Nếu người sử dụng chọn OK
    If fdFile.ShowDialog = DialogResult.OK Then
      ' Gán đường dẫn cho điều khiển TextBox
      txtFolder.Text = fdFile.SelectedPath
    ' Gọi phương thức liệt kê danh sách tập tin
      ListFile(txtFolder.Text)
    End If
End Sub
```

Trong đó, phương thức ListFile nhận chuỗi là tên và đường dẫn thư mục. Bạn sử dụng đối tượng DirectoryInfo và FileInfo để điền các thuộc tính của tập tin vào điều khiển DataGrid.

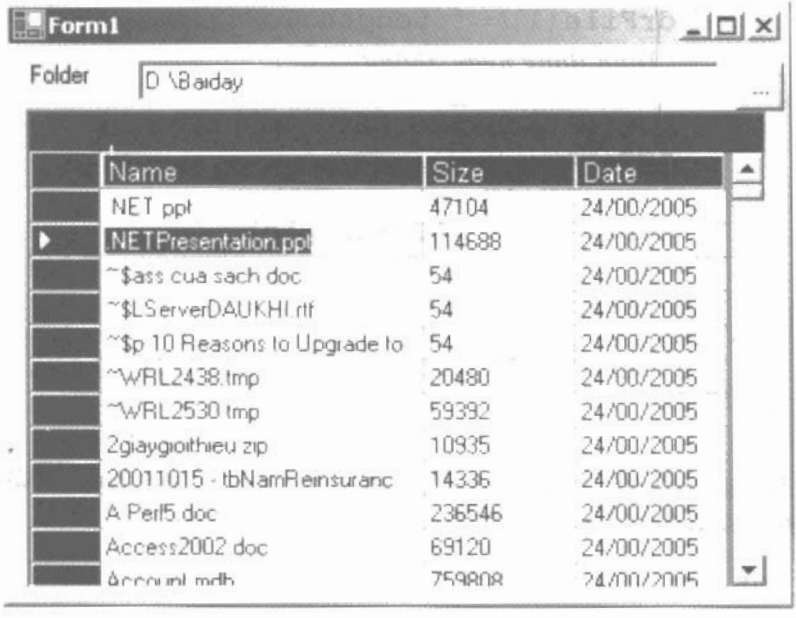
```
Sub ListFile(ByVal strPath As String)
    Dim dir As New DirectoryInfo(strPath)
    Dim dtFile As New DataTable
    ' Định nghĩa cột dữ liệu
    dtFile.Columns.Add("Name")
    dtFile.Columns.Add("Size")
    dtFile.Columns.Add("Date")
    Dim drFile As DataRow
```

```
        'Duyệt trên từng tập tin
        For Each f As FileInfo In dir.GetFiles("*.*")
            drFile = dtFile.NewRow
            drFile(0) = f.Name
            drFile(1) = f.Length.ToString
            'Định dạng ngày, tháng
            drFile(2) = _
            f.CreationTime.Date.ToString( _
            "dd/mmm/yyyy")
            'Thêm vào đối tượng DataTable
            dtFile.Rows.Add(drFile)
        Next
        'Áp dụng định dạng
        ApplyStyle(dtFile)
        Me.DataGrid1.DataSource = dtFile
    End Sub
```

Bằng cách sử dụng đối tượng DataGridTableStyle, bạn có thể thay đổi định dạng điều khiển như sau:

```
    Sub ApplyStyle(ByVal dtFile As DataTable)
        'Khai báo và khởi tạo đối tượng DataGridTableStyle
        Dim myStyle As New DataGridTableStyle
        'Với đối tượng DataGridTableStyle
        With myStyle
            'Khai báo các màu
            .HeaderForeColor = Color.White
            .HeaderBackColor = Color.Blue
            .SelectionForeColor = Color.Gray
            .HeaderFont = New Font("Arial", 10, _
            FontStyle.Regular)
            .ReadOnly = True
            .GridLineColor = Color.HotPink
            .SelectionBackColor = Color.Green
        End With
        'Khai báo thuộc tính bảng dữ liệu
        myStyle.MappingName = dtFile.TableName
        'Áp dụng định dạng cho điều khiển
        DataGrid1.TableStyles.Add(myStyle)
    End Sub
```

Khi thực thi chương trình, nếu người sử dụng chọn tên thư mục lập tức danh sách tập tin của thư mục đó xuất hiện tương tự như hình 29-2.



Hình 29-2: *Danh sách tập tin*

2. ĐỊNH DẠNG ĐIỀU KHIỂN TEXTBOX VÀO DATAGRID

1. Thiết kế *Form*, liệt kê danh sách các sản phẩm chi tiết của đơn đặ hàng ứng với từng khách hàng, cho phép người sử dụng thay đổi cộ số lượng, sau đó cập nhật trở lại cơ sở dữ liệu nguồn.

Để thực hiện ví dụ này, trước tiên bạn tạo một bảng dữ liệu có tê OrderDetails có cấu trúc như cấu trúc bảng Order Details bằng phát biể như sau:

```
CREATE TABLE OrderDetails
(
  OrderID int,
  ProductID int,
  Quantity smallint,
  UnitPrice money,
  Discount real
```

```
    primary key (OrderID, ProductID)
)
```

Lưu ý: Trong bảng Order Details không tồn tại khóa chính, do đó khi cập nhật bằng đối tượng SqlDataAdapter sẽ phát sinh lỗi.

Tiếp theo, bạn khai báo phương thức doUpdate nhận tham trị là đối tượng DataTable nắm giữ tập dữ liệu được người sử dụng thay đổi và tham trị là phát biểu SQL. Sau đó cập nhật tập dữ liệu này vào cơ sở dữ liệu nguồn.

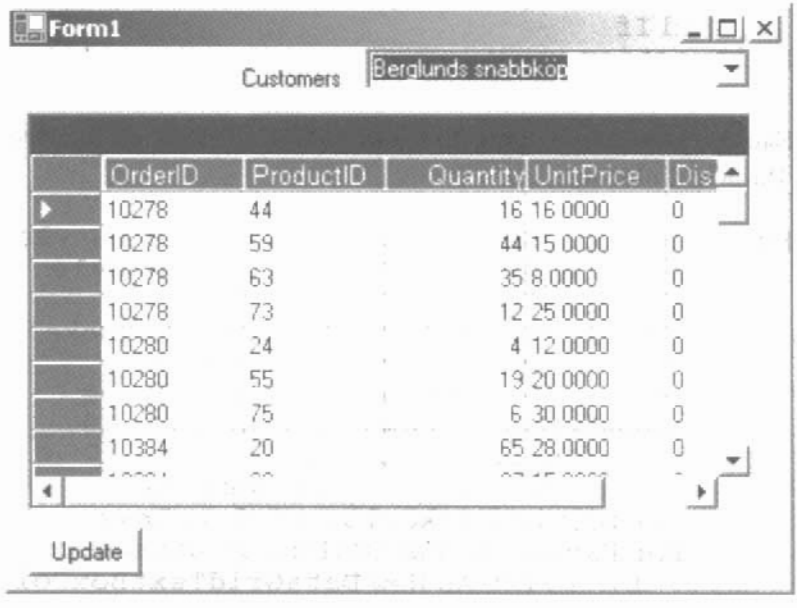
```
Function doUpdate(ByVal myDT As DataTable, ByVal
strSQL As String)
    If Not myDT Is Nothing Then
      Dim myData As New SqlDataAdapter( _
        strSQL, myCon)
      Dim myBuilder As New _
        SqlCommandBuilder(myData)
      myData.TableMappings.Add( _
      "Table", myDT.TableName)
      myData.Update(myDT)
    End If
End Function
```

Sau đó, khai báo phương thức dùng để định dạng dữ liệu trên điều khiển DataGrid như sau:

```
Function ApplyStyle() As DataGridTableStyle
    Dim myStyle As New DataGridTableStyle
    With myStyle
      .HeaderForeColor = Color.WhiteSmoke
      .HeaderBackColor = Color.Green
      .SelectionForeColor = Color.Olive
      .HeaderFont = New Font("Arial", 10, _
        FontStyle.Regular)
      .GridLineColor = Color.HotPink
      .SelectionBackColor = Color.Gray
      For Each dc As DataColumn In dtData.Columns
        Dim myTxt As New DataGridTextBoxColumn
        With myTxt
          .MappingName = dc.ColumnName
          .HeaderText = dc.Caption
          Select Case dc.ColumnName
```

```
                Case "Quantity"
                'Chỉ cho phép cập nhật cột Quantity
                   .Alignment = _
                HorizontalAlignment.Right
                   .ReadOnly = False
                   .Width = 70
                Case Else
                   .Width = 70
                   .ReadOnly = True
             End Select
          End With
          myStyle.GridColumnStyles.Add(myTxt)
       Next
    End With
    myStyle.MappingName = dtData.TableName
    Return myStyle
End Function
```

Khi thực thi chương trình, danh sách chi tiết đơn đặt hàng của khách hàng đang chọn mặc định được trình bày như hình 29-3.



Hình 29-3: *Danh sách chi tiết đơn đặt hàng*

Khi người sử dụng nhấn nút Update, bạn gọi phương thức doUpdate như sau:

```
Private Sub btnUpdate_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles btnUpdate.Click
    'Khai báo lại phát biểu SQL tương ứng với bảng
    strSQL = "select * from OrderDetails"
    cls.doUpdate(dtData.GetChanges, strSQL)
End Sub
```

Lưu ý, bạn cần khai báo hai phương thức khác như GetValue và đoạn chương trình gọi phương thức LoadData trong biến cố SelectionChangeCommitted của điều khiển ComboBox.

2. Tạo một ứng dụng cho phép người sử dụng nhập mã sản phẩm trong cột thứ nhất, lập tức tên của sản phẩm xuất hiện trong cột thứ hai và số lượng còn trong kho xuất hiện trong cột thứ 3.

Để thực hiện ví dụ này, trước tiên bạn khai báo phương thức GetValue trong lớp clsDatabase dùng để điền dữ liệu trên điều khiển DataGrid.

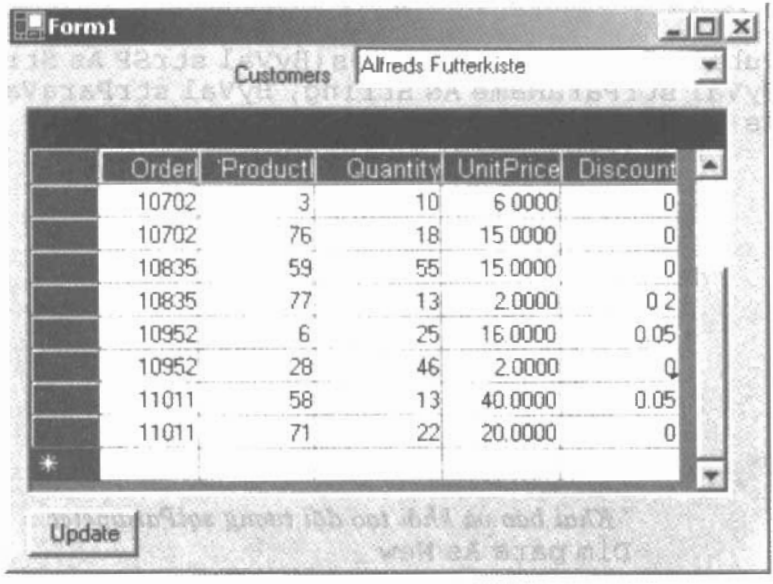
Kế đến, khai báo phương thức GetValue dùng để điền danh sách khách hàng trên điều khiển ComboBox.

Để cho phép lấy ra giá trị trong cột UnitsInStock và UnitPrice của mỗi sản phẩm, bạn khai báo phương thức GetValues như sau:

```
Public Function GetValues(ByVal strSP As String,
ByVal strParaName As String, ByVal strParaValue
As Integer) As Integer()
    Dim arrValue(2) As Integer
    Dim myCom As SqlCommand
    Dim RD As SqlDataReader
    Dim cls As clsItem
    Try
      myCom = New SqlCommand
      'Khai báo thuộc tính của đối tượng SqlCommand
      With myCom
        .Connection = myCon
        .CommandType = CommandType.Text
        .CommandText = strSP
        'Khai báo và khởi tạo đối tượng sqlParameter
        Dim para As New _
         SqlParameter(strParaName, _
        Type.GetType("System.Int32"))
```

```
            para.Value = strParaValue
            .Parameters.Add(para)
            RD = .ExecuteReader
          End With
          ' Nếu tồn tại mẩu tin
          If RD.Read Then
            arrValue(0) = _
            Convert.ToInt32(RD.GetValue(0))
            arrValue(1) = _
            Convert.ToInt32(RD.GetValue(1))
          End If
          RD.Close()
          strError = "OK"

        Catch ex As Exception
          strError = ex.Message
        Finally

        End Try
        Return arrValue
    End Function
```

Khi thực thi chương trình, danh sách chi tiết đặt hàng của mỗi khách hàng được trình bày như hình 29-4.

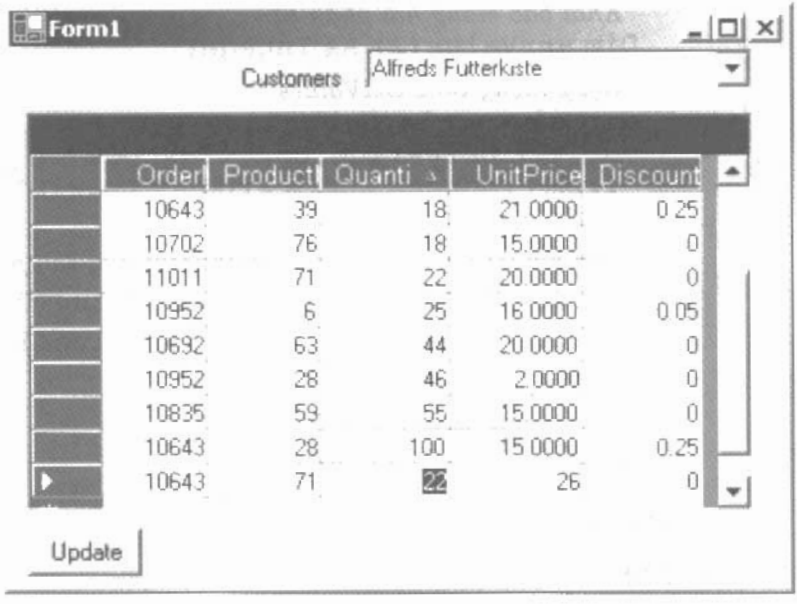


OrderI	ProductI	Quantity	UnitPrice	Discount
10702	3	10	6.0000	0
10702	76	18	15.0000	0
10835	59	55	15.0000	0
10835	77	13	2.0000	0.2
10952	6	25	16.0000	0.05
10952	28	46	2.0000	0
11011	58	13	40.0000	0.05
11011	71	22	20.0000	0

Hình 29-4: *Danh sách chi tiết đặt hàng*

Mỗi khi người sử dụng nhập mới mã sản phẩm trên cột thứ hai, lập tức số lượng còn trong kho và giá bán sẽ được trình bày trên cột thứ 2 và 3 tương ứng.

Giả sử, bạn thêm sản phẩm có mã là 71, sau khi nhấn Tab để chuyển qua cột kế tiếp, lập tức mã hợp đồng và các giá trị khác đã được điền trên hàng như hình 29-4-1.



Hình 29-4-1: *Thêm mới mẩu tin*

Lưu ý, giá trị 10643 tại cột thứ nhất là mã đơn đặt hàng trước đó, 22 là số lượng đang có trong kho, 26 là giá của sản phẩm vừa nhập.

Để làm điều này, bạn khai báo đoạn chương trình trong biến cố CurrentCellChanged của điều khiển DataGrid như sau:

```
Private Sub DataGrid1_CurrentCellChanged(ByVal
sender As Object, ByVal e As System.EventArgs)
Handles DataGrid1.CurrentCellChanged
    'Khai báo biến để lấy vị tri hàng và cột
    Dim col As Integer
    col= DataGrid1.CurrentCell.ColumnNumber
    Dim row As Integer
    row= DataGrid1.CurrentCell.RowNumber
    'Nếu đang ở cột thứ 1
```

```
        If col = 2 Or col = 0 Or col = 1 Then
          Dim Item As Integer
          ' Lấy mã sản phẩm vừa nhập
          If Convert.ToString( _
              DataGrid1.Item(row, 1)) <> "" Then
            Item = _
          Convert.ToInt32(DataGrid1.Item(row, 1))
            ' Khai báo mảng hai phần tử
            Dim arrValue(2) As Integer
            ' Gọi phương thức GetValues
            arrValue = cls.GetValues( _
            "select UnitPrice,UnitsInStock " & _
            "from Products where ProductID=@ID",_
             "@ID", Item)
            ' Gán giá trị UnitPrice và UnitsInStock của sản phẩm
            DataGrid1.Item(row, 2) = arrValue(0)
            DataGrid1.Item(row, 3) = arrValue(1)
            DataGrid1.Item(row, 4) = 0
            ' Số đơn đặt hàng trước đó
            DataGrid1.Item(row, 0) = _
              DataGrid1.Item(row - 1, 0)
          End If

        End If
    End Sub
```

3. ĐỊNH DẠNG ĐIỀU KHIỂN CHECKBOX VÀO DATAGRID

1. Liệt kê danh sách sản phẩm trên điều khiển *DataGrid*, nếu người sử dụng chọn những sản phẩm trên *checkbox* và nhấn nút *Update* thì bạn tăng giá của sản phẩm đó lên 10 đồng.

Để cho phép người sử dụng chọn vào các checkbox tương ứng với từng sản phẩm, bạn khai báo phương thức LoadData như sau:

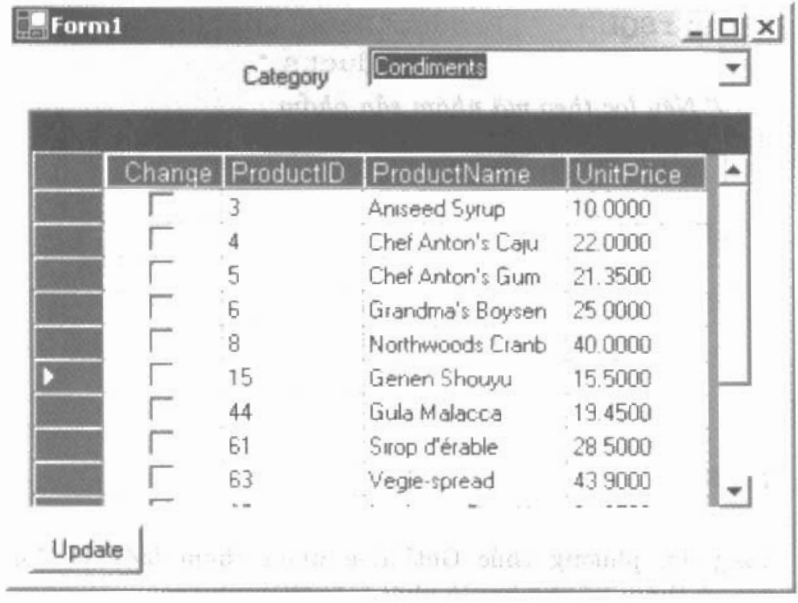
```
Sub LoadData(ByVal ID As String)
    dtData.Clear()
    ' Liệt kê danh sách sản phẩm
    strSQL = "select 0 as Changed,ProductID"
```

```
        strSQL += ",ProductName,UnitPrice "
        strSQL += " from Products "
        'Nếu lọc theo mã nhóm sản phẩm
        If ID <> "" Then
          strSQL += " where CategoryID='"
          strSQL += ID + "'"
        End If
        'Gọi phương thức GetValue
        If cls.OpenConnection = "OK" Then
          cls.GetValue(dtData, strSQL)
        End If
        'Điền dữ liệu trên điều khiển DataGrid
        DataGrid1.DataSource = dtData
    End Sub
```

Trong đó, phương thức GetValue nhận tham biến là đối tượng DataTable và tham trị thứ hai là phát biểu SQL như sau:

```
Public Function GetValue( _
  ByRef myDT As DataTable, _
  ByVal strSQL As String) As String
    Dim adData As SqlDataAdapter
    Try
      adData = New SqlDataAdapter(strSQL, myCon)
      adData.Fill(myDT)
      strError = "OK"
    Catch ex As Exception
      strError = ex.Message
    Finally

    End Try
    Return strError
End Function
```

Khi thực thi chương trình, danh sách sản phẩm của từng nhóm sản phẩm trình bày tương tự như hình 29-5.



Hình 29-5: *Danh sách sản phẩm*

Sau khi người sử dụng chọn một số sản phẩm bằng cách check vào checkbox và nhấn nút Update, lập tức các sản phẩm đó sẽ có giá tăng thêm 10.

Để thực hiện điều này, bạn khai báo trong biến cố Click của nút Update như sau:

```
Private Sub btnUpdate_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles btnUpdate.Click
    Dim strID As String
    ' Duyệt trên từng sản phẩm
    For i As Integer = 0 To dtData.Rows.Count - 1
        ' Nếu có chọn
        If DataGrid1.Item(i, 0) Then
            ' Lấy ra mã sản phẩm
            strID += _
            Convert.ToString(_
              DataGrid1.Item(i, 1)) + ","
        End If
    Next
    ' Gọi phương thức doUpdate
```

```
        cls.doUpdate(strID)
    End Sub
```

Trong đó, phương thức doUpdate nhận chuỗi bao gồm mã sản phẩm cách nhau dấu phẩy ứng với danh sách sản phẩm mà người sử dụng đang chọn, sử dụng đối tượng SqlCommand để thực thi thủ tục nội tại.

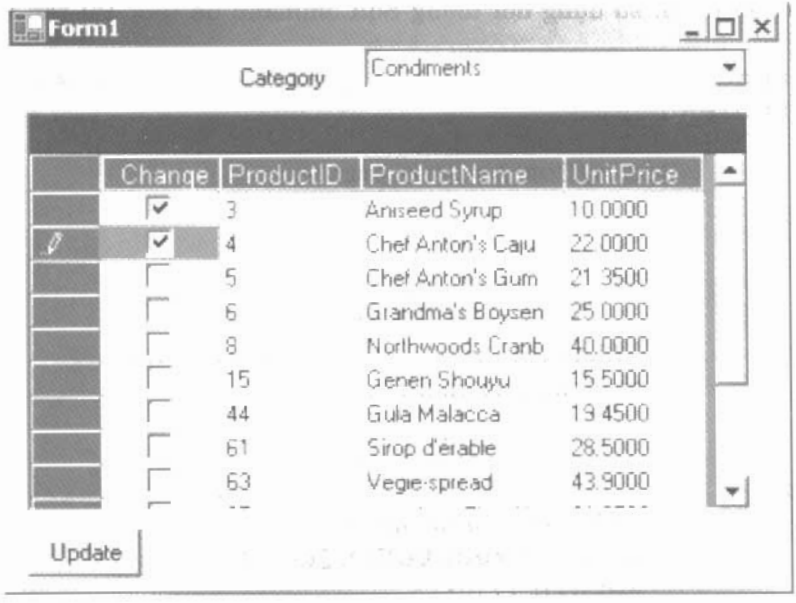
```
Function doUpdate(ByVal strID As String) As
Boolean
    If strID <> "" Then
      Dim myCom As SqlCommand
      Try
        ' Gọi thủ tục nội tại
        myCom = New SqlCommand( _
          "spUpdatePrice", myCon)
        ' Khai báo tham số và giá trị của thủ tục
        myCom.Parameters.Add( _
        "@myValue", strID)
         myCom.CommandType = _
          CommandType.StoredProcedure
        ' Thực thi thủ tục nội tại
        myCom.ExecuteNonQuery()
        Return True
      Catch ex As Exception
        strError = ex.Message
        Return False
      End Try
    End If
End Function
```

Trong đó, thủ tục nội tại spUpdatePrice sử dụng thủ tục hệ thống có tên sp_executeSQL để thực thi phát biểu SQL có cấu trúc như sau:

```
Create Proc spUpdatePrice
  @myValue varchar(100)
As
  declare @SQL nvarchar(500)
  set @myValue=char(39)+
    replace(@myValue,',',char(39)
    +','+char(39))+char(39)
  set @SQL=N'Update Products set UnitPrice=
    UnitPrice +10 where cast(ProductID as
```

```
    varchar(10)) in (' + @myValue+')'
exec sp_executeSQL @SQL
```

Chẳng hạn, bạn chọn hai sản phẩm đầu tiên và nhấn nút Update, lập tức giá của hai sản phẩm đó được tăng thêm 10 như hình 29-5-1.



Hình 29-5-1: *Tăng giá*

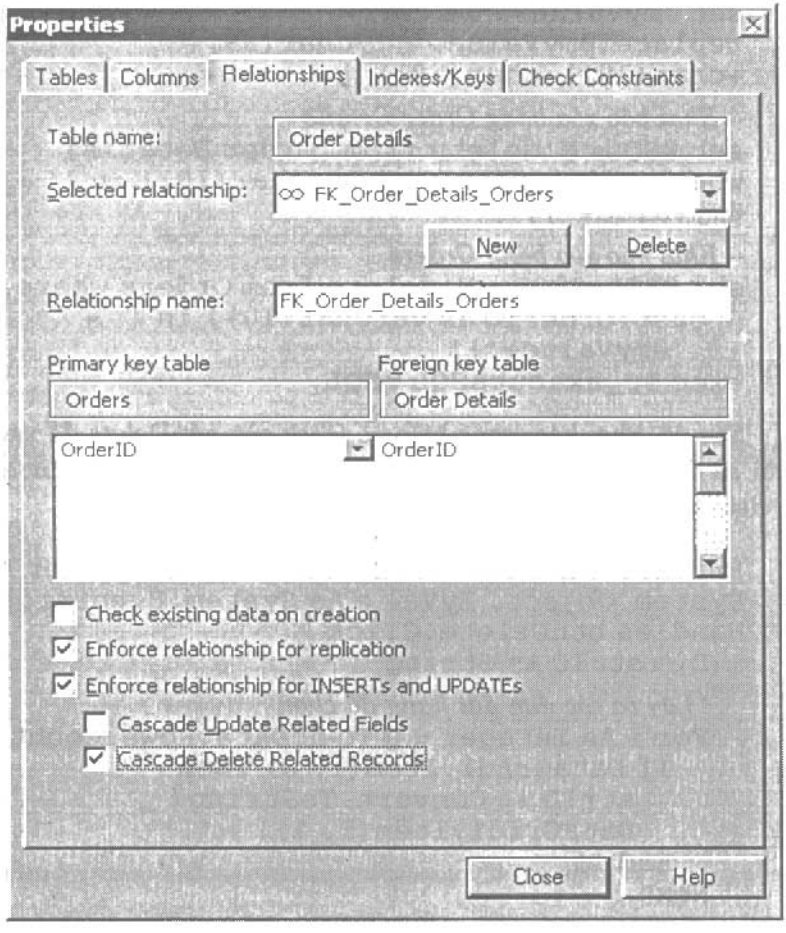
2. Thiết kế *Form* cho phép người sử dụng chọn một hay nhiều đơn đặt hàng. Sau đó nhấn nút *Delete* thì cửa sổ xác nhận hành động xóa hay không, nếu người sử dụng tiếp tục chọn *Yes* thì bạn xóa các đơn đặt hàng đó trong cơ sở dữ liệu.

Tương tự như ví dụ trên, đối với trường hợp này chúng ta khai báo thủ tục nội tại có cấu trúc như sau:

```
CREATE PROC spDeleteOrders
  @myValue varchar(100)
As
  declare @SQL nvarchar(500)
  set @myValue=char(39)+
  replace(@myValue,',',char(39)+','
  +char(39))+char(39)
  -- Khai báo xóa bảng Orders
  set @SQL=N'delete from Orders where
```

```
        cast(OrderID as varchar(10)) in (' +
          @myValue+')'
    exec sp_executeSQL @SQL
```

Lưu ý, nếu sử dụng cơ sở dữ liệu SQL Server 2000 thì khi tạo quan hệ 1-N giữa hai bảng Orders và [Order Details], bạn chọn vào tùy chọn Cascade Delete Related Records như hình 29-6.



Hình 29-6: *Tạo quan hệ*

Mỗi khi mẩu tin trong bảng Orders bị xóa, những mẩu tin trong bảng Order Details có giá trị tại cột OrderID bằng với giá trị của cột OrderID trong bảng Orders của mẩu tin đã xóa sẽ tự động xóa theo.

Trong trường hợp bạn không chọn vào tùy chọn này khi làm việc với cơ sở dữ liệu SQL Server 2000 hoặc cơ sở dữ liệu SQL Server 7.0

(không tồn tại tùy chọn trên), bạn cần xóa mẩu tin trong bảng Order Details trước khi xóa mẩu tin trong bảng Orders.

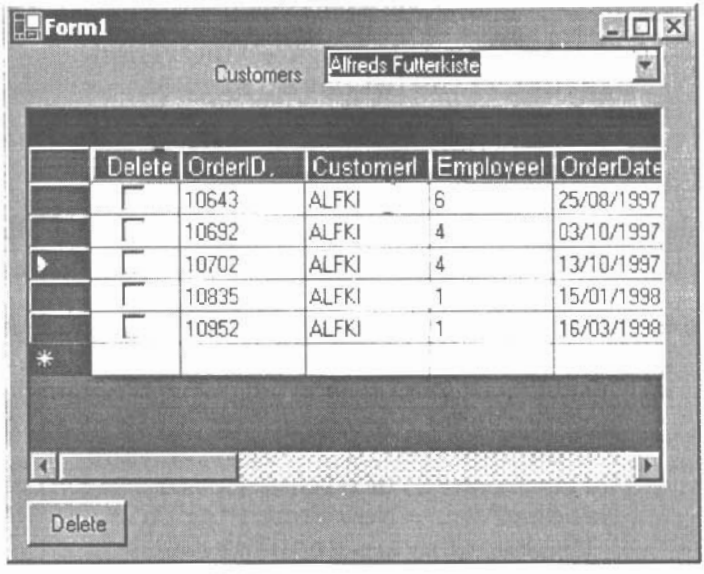
```
CREATE PROC spDeleteOrders
  @myValue varchar(100)
As
  declare @SQL nvarchar(500)
  set @myValue=char(39)+
  replace(@myValue,',',char(39)+','
  +char(39))+char(39)
  -- Khai báo xóa bảng Order Details
  set @SQL= N'delete from [Order Details]
  where cast(OrderID as varchar(10)) in (' +
  @myValue+');'
  -- Khai báo xóa bảng Orders
  set @SQL=@SQL + N'delete from Orders where
    cast(OrderID as varchar(10)) in (' +
      @myValue+')'
  exec sp_executeSQL @SQL
```

Sau đó, khai báo trong biến cố Click của nút Delete để xác nhận người sử dụng quyết định Yes hay No trước khi bạn gọi phương thức doDelete như sau:

```
Private Sub btnDelete_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles btnDelete.Click
  Dim strID As String
  ' Lấy ra các đơn đặt hàng đã chọn
  For i As Integer = 0 To dtData.Rows.Count - 1
    If DataGrid1.Item(i, 0) Then
      strID += Convert.ToString( _
      DataGrid1.Item(i, 1)) + ","
    End If
  Next
  ' Nếu người sử dụng có chọn
  If strID <> "" Then
    ' Xác nhận hành động xóa
    If MsgBox("Delete?", _
    MsgBoxStyle.YesNo, "ABC") = _
    MsgBoxResult.Yes Then
      ' Gọi phương thức doDelete
```

```
            cls.doDelete(strID)
            LoadData(Convert.ToString(_
            cbID.SelectedValue))
          End If
        End If
    End Sub
```

Khi thực thi chương trình, danh sách đơn đặt hàng sẽ xuất hiện trên điều khiển như hình 29-6-1.



Hình 29-6-1: *Danh sách đơn đặt hàng*

Nếu người sử dụng chọn vào đơn đặt hàng cuối cùng và nhấn nút Delete, lập tức đơn đặt hàng đó bị loại khỏi bảng Orders, đồng thời những mẩu tin trong bảng Order Details ứng với đơn đặt hàng này cũng bị xóa.

4. ĐỊNH DẠNG ĐIỀU KHIỂN COMBOBOX VÀO DATAGRID

1. Thiết kế *Form*, trình bày danh sách sản phẩm cùng với cột cho phép người sử dụng chọn một trong hai giá trị *Yes* hay *No* bằng điều khiển *ComboBox*.

Để thực hiện ví dụ này, trước tiên bạn khai báo phương thức LoadData để gọi phương thức GetValue của lớp clsDatabase và điền danh sách sản phẩm vào điều khiển DataGrid như sau:

```
Sub LoadData(ByVal ID As String)
    dtData.Clear()
    'Mặc định của cột Choose là No
    strSQL = "select 'No' as Choose,ProductID"
    strSQL += ",ProductName,UnitPrice "
    strSQL += " from Products "
    If ID <> "" Then
      strSQL += " where CategoryID='"
      strSQL += ID + "'"
    End If
    If cls.OpenConnection = "OK" Then
      cls.GetValue(dtData, strSQL)
    End If
    DataGrid1.DataSource = dtData
End Sub
```

Kế đến, bạn khai báo phương thức ApplyStyle để định nghĩa ComboBox trong cột thứ nhất của điều khiển DataGrid như sau:

```
Function ApplyStyle() As DataGridTableStyle
    Dim myStyle As New DataGridTableStyle
    With myStyle
      .HeaderForeColor = Color.WhiteSmoke
      .HeaderBackColor = Color.Green
      .SelectionForeColor = Color.Olive
      .HeaderFont = New Font("Arial", _
        10, FontStyle.Regular)
      .GridLineColor = Color.HotPink
      .SelectionBackColor = Color.Gray
      For Each dc As DataColumn In dtData.Columns
        Select Case dc.ColumnName
          Case "Choose"
            'Cột dạng ComboBox
            Dim myCb As DataGridComboBoxColumn
            'Khai báo đối tượng DataGridComboBoxColumn
            With myCb
              = New DataGridComboBoxColumn( _
              YesNo.DefaultView, "Choose", _
              "Choose")
              .MappingName = dc.ColumnName
              .HeaderText = dc.Caption
            End With
            myStyle.GridColumnStyles.Add(myCb)
          Case Else
```

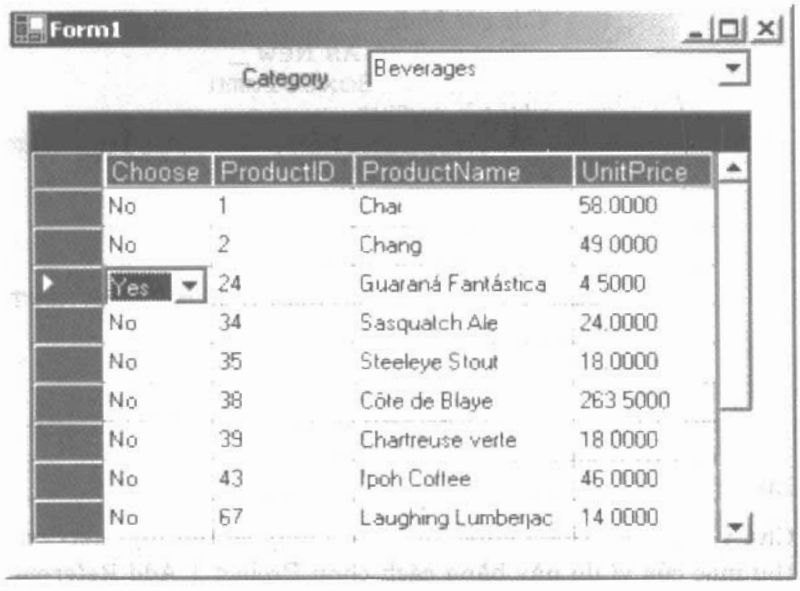
```
                ' Các cột khác
                    Dim myTxt As New _
                DataGridTextBoxColumn
                    With myTxt
                        .MappingName = dc.ColumnName
                        .HeaderText = dc.Caption
                        .Width = 70
                        .ReadOnly = True
                    End With
                myStyle.GridColumnStyles.Add(myTxt)
            End Select
        Next
    End With
    myStyle.MappingName = dtData.TableName
    Return myStyle
End Function
```

Chú ý, bạn cần tham chiếu đến Assembly có tên myUserControl.dll trong thư mục của ví dụ này bằng cách chọn Project | Add Reference.

Ngoài ra, trong ví dụ trên chúng ta có sử dụng phương thức YesNo, phương thức trả về đối tượng DataTable chứa hai mẩu tin ứng với Yes và No.

```
Function YesNo() As DataTable
    Dim dtYesNo As New DataTable
    dtYesNo.Columns.Add("Choose")
    Dim dr As DataRow
    dr = dtYesNo.NewRow
    dr.Item(0) = "No"
    dtYesNo.Rows.Add(dr)
    dr = dtYesNo.NewRow
    dr.Item(0) = "Yes"
    dtYesNo.Rows.Add(dr)
    Return dtYesNo
End Function
```

Khi thực thi chương trình, danh sách sản phẩm trình bày trên điều khiển DataGrid với cột đầu tiên là điều khiển ComboBox có hai giá trị Yes và No như hình 29-7.



Hình 29-7: *ComboBox trong điều khiển DataGrid*

2. Tạo *Project*, trong đó có sử dụng điều khiển *DataGrid* liệt kê danh sách khách hàng trong bảng *Customers* với định dạng *ComboBox* trên cột *Country*. Nếu người sử dụng nhấn nút *Update* thì bạn cập nhật lại những khách hàng đã thay đổi *Country*.

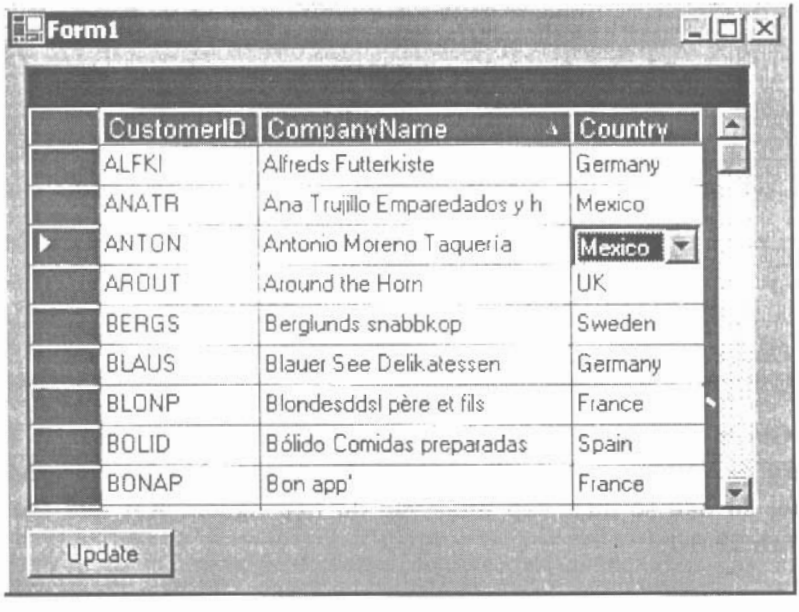
Đối với ví dụ này, bạn khai báo biến đối tượng DataTable có tên dtCountries, sau đó gọi phương thức GetValue để điền danh sách Country vào đối tượng này trong biến Load của Form.

```
Private Sub Form1_Load(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles MyBase.Load
    If cls.OpenConnection = "OK" Then
        ' Điền dữ liệu trên điều khiển DataGrid
        LoadData()
        ' Lấy ra tập các Country
        cls.GetValue(dtCountries, _
        "select distinct Country,Country " & _
          "from Customers")
        ' Áp dụng định dạng
        DataGrid1.TableStyles.Add(ApplyStyle)
    End If
End Sub
```

Trong phương thức ApplyStyle, nếu cột dữ liệu là Country, bạn khai báo đối tượng DataTable và gán bằng đối tượng dtCountries rồi sử dụng đối tượng DataGridComboBoxColumn như sau:

```
Case "Country"
   Dim myCb As DataGridComboBoxColumn
   ' Khởi tạo đối tượng dtCountry
   dtCountry = New DataTable
   ' Gán đối tượng dtCountry bằng đối tượng dtCountries
   dtCountry = dtCountries
   ' Khởi tạo đối tượng DataGridComboBoxColumn
   myCb = New DataGridComboBoxColumn( _
   dtCountry.DefaultView, "Country", "Country")
   With myCb
      .MappingName = dc.ColumnName
      .HeaderText = dc.Caption
   End With
   myStyle.GridColumnStyles.Add(myCb)
```

Khi thực thi chương trình, điều khiển ComboBox xuất hiện trên cột Country với giá trị chính là giá trị của khách hàng đó như hình 29-8.



Hình 29-8: *Danh sách Country*

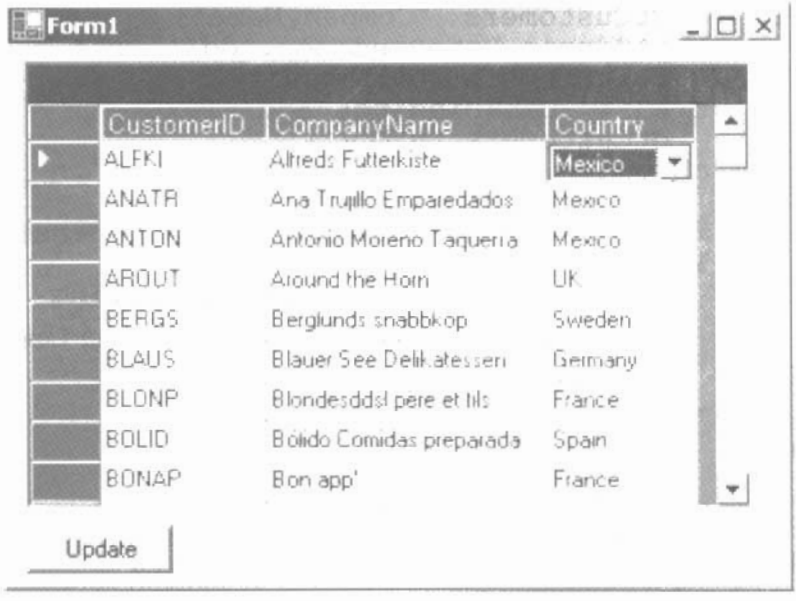
Sau khi thay đổi Country của từng khách hàng, nếu người sử dụng nhấn nút Update, lập tức các khách hàng đó được cập nhật. Để thực hiện điều này, bạn khai báo trong biến cố Click của nút Update như sau:

```
Private Sub btnUpdate_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles btnUpdate.Click
    cls.doUpdate(dtData.GetChanges, strSQL)
    LoadData()
End Sub
```

Trong đó, phương thức doUpdate được định nghĩa trong lớp clsDatabase có cấu trúc:

```
Function doUpdate(ByVal myDT As DataTable, ByVal
strSQL As String)
    If Not myDT Is Nothing Then
      Dim myData As New SqlDataAdapter( _
        strSQL, myCon)
      Dim myBuilder As New _
          SqlCommandBuilder(myData)
      myData.TableMappings.Add( _
        "Table", myDT.TableName)
      myData.Update(myDT)
    End If
End Function
```

Giả sử, khách hàng Alfreds Futterkiste đang có giá trị tại cột Country là Germany, nếu bạn thay đổi thành Mexico và nhấn nút Update, dữ liệu sẽ được cập nhật vào dữ liệu nguồn. Kết quả trình bày như hình 29-8-1.



Hình 29-8-1: *Cập nhật cột Country*

5. ĐIỀU HƯỚNG DỮ LIỆU

1. Liệt kê danh sách khách hàng. trình bày từng mẩu tin và điều hướng bằng cách dùng đối tượng *BindingManagerBase*, nếu cột dữ liệu là khóa ngoại, bạn sử dụng điều khiển dạng danh sách.

Trước tiên bạn khai báo hai biến đối tượng DataTable, đối tượng dtCustomers nắm giữ danh sách khách hàng trong bảng Customers, đối tượng dtCountries nắm giữ danh sách Country, đối tượng myBMB và đối tượng clsDatabase

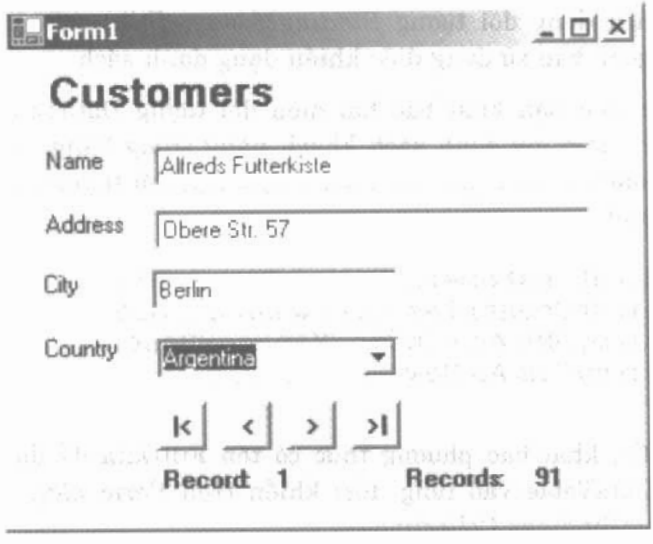
```
Dim dtCustomers As New DataTable
Dim dtCountries As New DataTable
Dim myBMB As BindingManagerBase
Dim myCls As New clsDatabase
```

Kế đến, khai báo phương thức có tên FillData để điền dữ liệu từ đối tượng DataTable vào từng điều khiển trên Form bằng phương thức Add của DataBindings Collection.

```
Sub FillData()
    txtName.DataBindings.Add("Text", _
```

```
        dtCustomers, "CompanyName")
    txtAddress.DataBindings.Add("Text", _
        dtCustomers, "Address")
    txtCity.DataBindings.Add("Text", _
        dtCustomers, "City")
    cbCountry.DataBindings.Add("Text", _
        dtCustomers, "Country")
    myBMB = Me.BindingContext(dtCustomers)
    lbRecords.Text = _
        (dtCustomers.Rows.Count).ToString
        ' Gọi phương thức GetValue để điền danh sách Country
        ' vào điều khiển cbCountry
    myCls.GetValue(dtCountries, _
        "select distinct Country, " & _
        " Country from Customers")
    cbCountry.DataSource = dtCountries
    cbCountry.DisplayMember = "Country"
    cbCountry.ValueMember = "Country"
End Sub
```

Khi thực thi chương trình, danh sách Country được điền vào điều khiển ComboBox và giá trị mặc định chọn chính là giá trị của khách hàng đang trình bày như hình 29-9.



Hình 29-9: *Sử dụng DataBindings Collection*

Lưu ý, bạn có thể sử dụng các điều khiển khác ứng với giá trị của từng cột trong bảng thay vì sử dụng TextBox, nhằm bảo đảm Form nhập liệu phù hợp.

2. Thiết kế Form tương tự như ví dụ ở trên, mỗi khi người sử dụng chọn khách hàng, bạn liệt kê danh sách đơn đặt hàng trên điều khiển DataGrid.

Để thực hiện ví dụ này, trước tiên bạn khai báo phương thức có tên FillData để điền dữ liệu từ đối tượng DataTable vào từng điều khiển trên Form bằng phương thức Add của DataBindings Collection.

```
Sub FillData()
    txtName.DataBindings.Add("Text", _
      dtCustomers, "CompanyName")
    txtAddress.DataBindings.Add("Text", _
      dtCustomers, "Address")
    txtCity.DataBindings.Add("Text", _
      dtCustomers, "City")
    cbCountry.DataBindings.Add("Text", _
      dtCustomers, "Country")
    myBMB = Me.BindingContext(dtCustomers)
    lbRecords.Text = _
      (dtCustomers.Rows.Count).ToString
      ' Gọi phương thức GetValue để điền danh sách Country
      ' vào điều khiển cbCountry
    myCls.GetValue(dtCountries, _
      "select distinct Country, " & _
      " Country from Customers")
    cbCountry.DataSource = dtCountries
    cbCountry.DisplayMember = "Country"
    cbCountry.ValueMember = "Country"
End Sub
```

Để cho phép trình bày danh sách đơn đặt hàng mỗi khi người sử dụng chọn khách hàng, bạn khai báo phương thức có tên GetData, sử dụng thuộc tính RowFilter đối tượng DataView để lọc đơn đặt hàng của từng khách hàng:

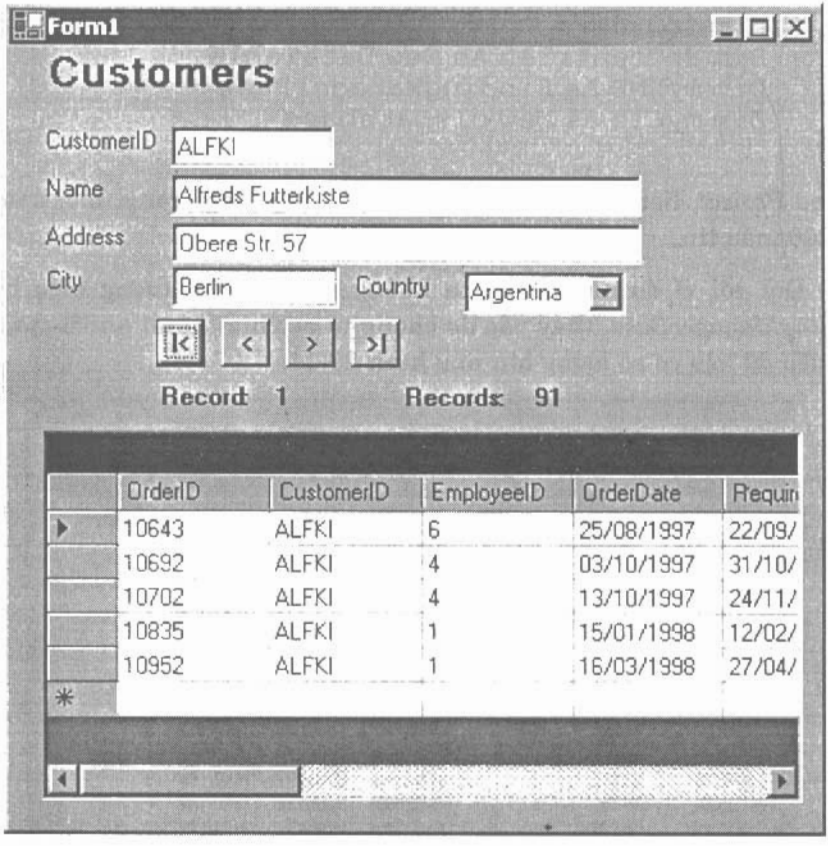
```
Sub GetData(ByVal ID As String)
    If dtOrders.Rows.Count > 0 Then
      Dim dvOrders As New DataView(dtOrders)
      dvOrders.RowFilter = _
```

```
            "CustomerID='" + ID + "'"
          DataGrid1.DataSource = dvOrders
        End If
    End Sub
```

Sau đó, gọi hai phương thức này trong biến cố Load của Form như sau:

```
    Private Sub Form1_Load(ByVal sender As
    System.Object, ByVal e As System.EventArgs)
    Handles MyBase.Load
        Dim strError As String
        'Nếu kết nối cơ sở dữ liệu thành công
        If myCls.OpenConnection = "OK" Then
          'Lấy ra danh sách khách hàng
          strError = myCls.GetValue(dtCustomers, _
          "select * from Customers")
        End If
        If strError = "OK" Then
          'Điền dữ liệu vào các điều khiển
          FillData()
          'Lấy ra danh sách đơn đặt hàng
          myCls.GetValue(dtOrders, _
            "select * from Orders")
          'Trình bày danh sách đơn đặt hàng của khách
          'hàng hiện hành
          GetData(txtID.Text)
        Else
          MsgBox("Can not load data")
          btnPre.Enabled = False
          btnNext.Enabled = False
          btnLast.Enabled = False
          btnFirst.Enabled = False
        End If
    End Sub
```

Khi thực thi chương trình, mặc định Form trình bày khách hàng đầu tiên, đồng thời danh sách đơn đặt hàng của khách hàng này cũng trình bày trên điều khiển DataGrid như hình 29-10.



Hình 29-10: *Danh sách đơn đặt hàng*

Để cho phép liệt kê đơn đặt hàng của từng khách hàng mỗi khi người sử dụng chọn vào các nút điều hướng, bạn khai báo để gọi phương thức GetData trong biến cố Click của các nút tương tự như sau:

```
Private Sub btnLast_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles btnLast.Click
    myBMB.Position = dtCustomers.Rows.Count - 1
    lbRecord.Text = _
    (myBMB.Position + 1).ToString
    GetData(txtID.Text)
End Sub
```

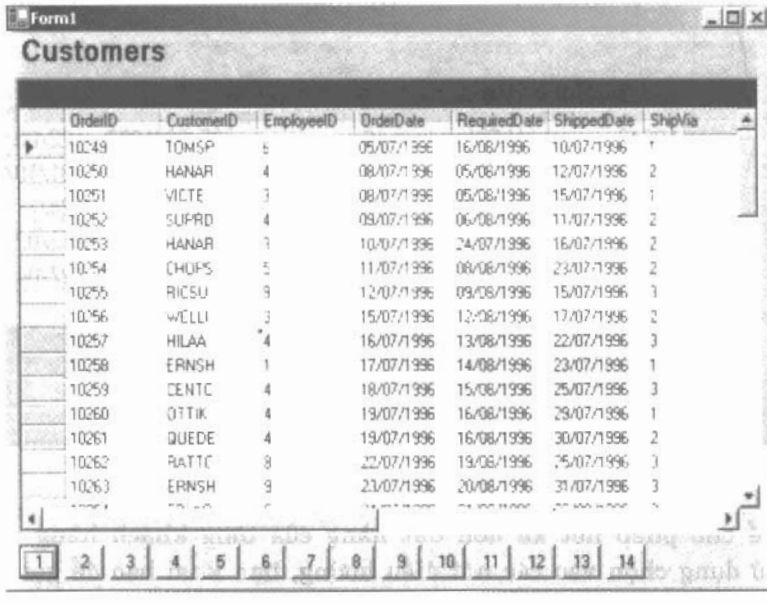
Ngoài ra, bạn cần khai báo các biến dùng chung trên đầu lớp Form1.

```
Dim dtCustomers As New DataTable
```

```
Dim dtOrders As New DataTable
Dim dtCountries As New DataTable
Dim myBMB As BindingManagerBase
Dim myCls As New clsDatabase
```

3. Tạo *Project*, liệt kê danh sách đơn đặt hàng từng trang, mỗi trang là 200 mẩu tin.

Đối với ví dụ này chúng ta không sử dụng đối tượng điều hướng *BindingManagerBase*, thay vào đó chúng ta sẽ chia thành nhiều trang dữ liệu nếu dữ liệu có số lượng lớn như hình 29-11.



Hình 29-11: *Phân trang*

Để xác định được số trang cần trình bày, bạn dựa vào thuộc tính Count của Rows Collection ứng với mỗi đối tượng DataTable. Trong trường hợp này, lần đầu tiên chúng ta gọi phương thức GetValue với phát biểu Select * from Orders, số mẩu tin trả về là 829 mẩu tin.

```
Public Function GetValue( _
    ByVal strTable As String) As DataTable
    Dim adData As SqlDataAdapter
    Dim myDT As New DataTable
    Try
      adData = New SqlDataAdapter( _
```

```
            "select * from " + strTable, myCon)
          adData.Fill(myDT)
          strError = "OK"
        Catch ex As Exception
          strError = ex.Message
        Finally

        End Try
        Return myDT
    End Function
```

Kế đến, bạn gọi lại phương thức GetValue ứng với phát biểu Select * from Orders, nhưng sử dụng phương thức Fill của đối tượng SqlDataAdapter với hai tham trị là mẩu tin bắt đầu và số mẩu tin lấy ra.

```
    Public Function GetValue( _
       ByVal StartRecord As Integer, _
      ByVal MaxRecord As Integer, _
      ByVal strTable As String) As DataTable
        Dim adData As SqlDataAdapter
        Dim myDS As New DataSet
        Try
          adData = New SqlDataAdapter( _
          "select * from " + strTable, myCon)
          ' Sử dụng phương thức Fill với số mẩu tin giới hạn
          adData.Fill(myDS, StartRecord, _
            MaxRecord, strTable)
          strError = "OK"
        Catch ex As Exception
          strError = ex.Message
        Finally

        End Try
        Return myDS.Tables(0)
    End Function
```

Sau đó, bạn gọi hai phương thức này trong biến cố Load của Form và tính ra được số trang cần trình bày.

```
    Private Sub Form1_Load(ByVal sender As
    System.Object, ByVal e As System.EventArgs)
    Handles MyBase.Load
        Dim strError As String
        If myCls.OpenConnection = "OK" Then
            ' Gọi lần nhất để lấy ra tổng số mẩu tin
```

```
            dtCustomers = myCls.GetValue("Orders")
            Dim rows As Integer = _
              dtCustomers.Rows.Count
            ' Gọi lần hai để lấy ra số mẩu tin của trang số 1
            dtCustomers = myCls.GetValue( _
              1, MaxRecord, _
            "Orders")
            DataGrid1.DataSource = dtCustomers
            Dim page As Integer = rows \ MaxRecord
            If rows Mod MaxRecord > 0 Then
              page += 1
            End If
            ' Gọi phương thức để tạo ra số nút ứng với số trang
            CreateButton(page)
        End If
    End Sub
```

Trong đó, phương thức CreateButton nhận tham trị là số trang cần trình bày rồi tạo ra các nút.

```
    Sub CreateButton(ByVal NumberPage As Integer)
        Dim i As Integer
        ' Khai báo đối tượng Button
        Dim btnPage As Button
        For i = 1 To NumberPage
            ' Khởi tạo đối tượng Button
            btnPage = New Button
            ' Khai báo các thuộc tính cho đối tượng Button
            With btnPage
                .Name = i.ToString
                .Text = i.ToString()
                .Size = New System.Drawing.Size(30, 24)
                .Location = New System.Drawing.Point( _
                  35 * (i - 1) + 8, 360)
                ' Gán phương thức biến cố cho nút
                AddHandler .Click, _
                AddressOf ButtonHandler
            End With
            ' Thêm đối tượng Button vào Form
            Me.Controls.Add(btnPage)
        Next
    End Sub
```

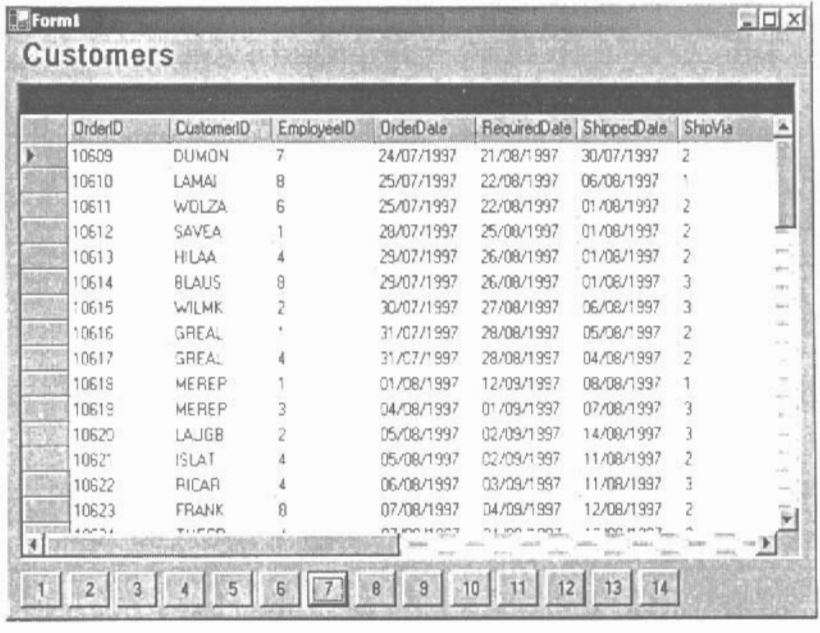
Khi tạo ra mỗi nút ứng với mỗi trang, bạn gán phương thức ButtonHandler vào biến cố Click của nút đó có nội dung như sau:

```
Public Sub ButtonHandler(ByVal sender As
System.Object, ByVal e As System.EventArgs)
' Dựa vào thuộc tính Text để biết trang thứ mấy
    Paging(CType(sender, Button).Text)
End Sub
```

Trong đó, phương thức Paging nhận chuỗi i và đổi sang số, rồi gọi phương thức GetValue ứng với mẩu tin của trang đó.

```
Sub Paging(ByVal PageNo As String)
    dtCustomers = myCls.GetValue( _
    Convert.ToInt32( _
      PageNo - 1) * MaxRecord + 1, MaxRecord, _
    "Orders")
    DataGrid1.DataSource = dtCustomers
End Sub
```

Như vậy, mỗi khi người sử dụng nhấn vào nút, chẳng hạn nút thứ 7, lập tức mẩu tin thứ 361 đến mẩu tin thứ 381 hiện ra như hình 29-12.



Hình 29-12: *Trang thứ 7*

Lưu ý, để ví dụ trên thực thi đúng, bạn khai báo 3 biến dùng chung trên phần đầu của lớp như sau:

```
Dim dtCustomers As New DataTable
Dim myCls As New clsDatabase
Dim MaxRecord As Integer = 60
```

Chuyên đề 22:
LỚP DỮ LIỆU CHO TỪNG THỰC THỂ

1. THẢO LUẬN GIẢI PHÁP TƯƠNG TÁC CƠ SỞ DỮ LIỆU

Không có bài tập.

2. XÂY DỰNG LỚP BAO GỒM CÁC THUỘC TÍNH VÀ PHƯƠNG THỨC CỦA TỪNG BẢNG DỮ LIỆU

1. Xây dựng *Shared Assembly* có tên *SQLDatabase* bao gồm các phương thức cho phép thực hiện các tác vụ thêm, cập nhật, xóa và truy vấn dữ liệu *SQL Server*.

Để xây dựng Assembly này, trước tiên bạn tạo mới Project thuộc loại ClassLibrary và đặt tên Project tùy ý. Kế đến, khai báo lớp *SQLDatabase* có cấu trúc tuần tự như sau.

Khai báo biến sử dụng chung và Constructor nhận chuỗi kết nối cơ sở dữ liệu.

```
Imports System.Data.SqlClient
Public Class SQLDatabase
  Private psCon As String
  Public strError As String = ""
  Dim myCon As SqlConnection
  'Khai báo Constructor
  Sub New(ByVal strCon As String)
    psCon = strCon
    psCon = "server=" + gsServer
    psCon += ";database=" + gsDatabase
    psCon += ";Integrated Security=SSPI"
  End Sub
  Sub New()
  End Sub
  'Khai báo thuộc tính
  Property Connection() As String
```

```
        Get
          Return psCon
        End Get
        Set(ByVal Value As String)
          psCon = Value
        End Set
    End Property
```

Tiếp theo, bạn khai báo phương thức để mở và đóng kết nối cơ sở dữ liệu SQL Server.

```
    'Mở kết nối cơ sở dữ liệu
    Public Function OpenConnection() As String
        Try
          myCon = New SqlConnection(psCon)
          myCon.Open()
          strError = "OK"
        Catch eq As SqlException
          strError = eq.Message
        Catch ex As Exception
          strError = ex.Message
        End Try
        Return strError
      End Function
      'Đóng kết nối cơ sở dữ liệu
      Public Sub CloseConnection()
        If myCon.State <> _
           ConnectionState.Closed Then
          myCon.Close()
          myCon.Dispose()
        End If
    End Sub
```

Khai báo phương thức thực thi phát biểu SQL dạng hành động bằng phương thức ExecuteNonQuery.

```
    Public Sub ExecuteSQL(ByVal strSQL As String,
    ByRef EffectedRecord As Integer)
        Try
          'Khai báo và khởi tạo đối tượng SqlCommand
          Dim myCom As New SqlCommand(strSQL, myCon)
          EffectedRecord = myCom.ExecuteNonQuery()
        Catch ex As Exception
          strError = ex.Message
        End Try
    End Sub
```

Trong trường hợp bạn muốn sử dụng thủ tục hệ thống để thực thi phát biểu SQL dạng hành động, bạn khai báo phương thức với các tham trị là mảng tham số và mảng giá trị tương ứng với tham số như sau:

```
Public Function ExecuteSQL(ByVal strTable As
String, _
  ByVal strPara() As String, _
  ByVal strValue() As String) As Integer
    ' Khai báo nhận số mẫu tin thực thi
    Dim EffectedRecord As Integer = 0
    Try
      Dim myCom As New SqlCommand
      myCom.Connection = myCon
      myCom.CommandType = CommandType.Text
      ' Khai báo đối tượng SqlParameter
      Dim myPara As SqlParameter
      ' Duyệt qua từng phần tử giá trị
      For i As Integer = 0 To strValue.Length - 1
        ' Khởi tạo đối tượng SqlParameter
        myPara = New SqlParameter(strPara(i), _
          strValue(i))
        ' Thêm đối tượng SqlParameter vào đối tượng
        ' SqlCommand
        myCom.Parameters.Add(myPara)
      Next
      ' Thực thi thủ tục hệ thống
      EffectedRecord = myCom.ExecuteNonQuery()
    Catch ex As Exception
      strError = ex.Message
    End Try
    Return EffectedRecord
End Function
```

Trong trường hợp thủ tục nội tại thực thi trả về giá trị là số nguyên tự động phát sinh ứng với cột số tự động, bạn sử dụng phương thức ExecuteScalar.

```
Public Sub ExecuteSQL(ByVal strSQL As String,
ByRef Value As Object)
    Try
      Dim myCom As New SqlCommand(strSQL, myCon)
```

```
        ' Thực thi thủ tục hệ thống, nhận giá trị trả về là chuỗi
        Value = myCom.ExecuteScalar
      Catch ex As Exception
        strError = ex.Message
      End Try
  End Sub
```

Ngoài ra, nếu sau khi thực thi thủ tục với tham số và tham trị, giá trị trả về là chuỗi thì bạn khai báo phương thức như sau:

```
Public Sub ExecuteSP(ByVal strSP As String, _
  ByVal strPara() As String, _
  ByVal strValue() As String, _
    ByRef Value As String)
    Try
      Dim myCom As New SqlCommand
      myCom.Connection = myCon
      myCom.CommandType = CommandType.Text
      ' Khai báo đối tượng SqlParameter
      Dim myPara As SqlParameter
      ' Duyệt qua từng phần tử giá trị
      For i As Integer = 0 To strValue.Length - 1
        ' Khởi tạo đối tượng SqlParameter
        myPara = New SqlParameter(strPara(i), _
          strValue(i))
        ' Thêm đối tượng SqlParameter vào đối tượng
        ' SqlCommand
        myCom.Parameters.Add(myPara)
      Next
      ' Thực thi thủ tục hệ thống
      Value = myCom.ExecuteScalar()
    Catch ex As Exception
      strError = ex.Message
    End Try
End Sub
```

Lưu ý, đây là các phương thức gợi ý, trong thực tế bạn cần xây dựng các phương thức cho phép thực thi thủ tục hệ thống với các tham số và giá trị của chúng có thể thuộc nhiều kiểu dữ liệu khác nhau.

Ngoài các phương thức thực thi phát biểu SQL dạng hành động, bạn cần khai báo một số phương thức dùng để truy vấn dữ liệu.

Đối với trường hợp này, trước tiên bạn khai báo phương thức nhận phát biểu SQL trả về đối tượng DataTable.

```
Public Sub getValue(ByRef myDT As DataTable, _
  ByVal strSQL As String)
    Dim myCon As New SqlConnection
    Try
      Dim myData As New SqlDataAdapter( _
      strSQL, myCon)
      myData.Fill(myDT)
    Catch ex As Exception
      strError = ex.Message
    End Try
End Sub
```

Trong trường hợp cần điền nhiều bảng dữ liệu vào đối tượng DataSet, bạn có thể khai báo phương thức như sau:

```
Public Sub getValue(ByRef myDS As DataSet, _
  ByVal strSQL As String)
    Dim myCon As New SqlConnection
    Try
      Dim myData As New SqlDataAdapter( _
        strSQL, myCon)
      myData.Fill(myDS, strSQL)
    Catch ex As Exception
      strError = ex.Message
    End Try
End Sub
```

Đối với trường hợp lấy ra đối tượng DataTable bằng cách thực thi thủ tục nội tại có tham số, bạn cần khai báo phương thức sử dụng đối tượng SqlParameter như sau:

```
Public Sub getValue(ByRef myDT As DataTable, _
  ByVal strSP As String, _
  ByVal strPara() As String, _
  ByVal strValue() As String)
    Try
      ' Khai báo và khởi tạo đối tượng SqlCommand
        Dim myCom As New SqlCommand
      ' Gán thuộc tính
        myCom.Connection = myCon
        myCom.CommandType = _
```

```
            CommandType.StoredProcedure
          myCom.CommandText = strSP
        ' Khai báo đối tượng SqlParameter
          Dim myPara As SqlParameter
        ' Duyệt trên từng tham số
          For i As Integer = 0 To strPara.Length - 1
          ' Khởi tạo đối tượng SqlParameter
            myPara = New SqlParameter(strPara(i), _
              strValue(i))
            myCom.Parameters.Add(myPara)
          Next
        ' Khai báo và khởi tạo đối tượng SqlDataAdapter
          Dim myData As New SqlDataAdapter(myCom)
        ' Điền dữ liệu vào đối tượng DataTable
          myData.Fill(myDT)
        Catch ex As Exception
          strError = ex.Message
        End Try
    End Sub
```

Nếu lấy ra đối tượng DataSet bằng cách thực thi thủ tục nội tại có tham số, bạn cần khai báo phương thức để sử dụng đối tượng SqlParameter như sau:

```
Public Sub getValue(ByRef myDT As DataSet, _
  ByVal strSP As String, _
  ByVal strPara() As String, _
  ByVal strValue() As String)
    Try
      ' Khai báo và khởi tạo đối tượng SqlCommand
        Dim myCom As New SqlCommand
      ' Gán thuộc tính
        myCom.Connection = myCon
        myCom.CommandType = _
          CommandType.StoredProcedure
        myCom.CommandText = strSP
      ' Khai báo đối tượng SqlParameter
        Dim myPara As SqlParameter
      ' Duyệt trên từng tham số
        For i As Integer = 0 To strPara.Length - 1
        ' Khởi tạo đối tượng SqlParameter
```

```
            myPara = New SqlParameter(strPara(i), _
              strValue(i))
            myCom.Parameters.Add(myPara)
          Next
        'Khai báo và khởi tạo đối tượng SqlDataAdapter
          Dim myData As New SqlDataAdapter(myCom)
        'Điền dữ liệu vào đối tượng DataSet
          myData.Fill(myDS)
        Catch ex As Exception
          strError = ex.Message
        End Try
    End Sub
```

Phương thức trên sử dụng đối tượng DataSet nắm giữ tập dữ liệu, nếu bạn chỉ cần lấy ra một số mẩu tin thì sử dụng phương thức Fill như ví dụ sau:

```
    Public Sub getValue(ByRef myDT As DataSet, _
      ByVal strSP As String, _
      ByVal strPara() As String, _
      ByVal strValue() As String, _
      ByVal StartRecord As Integer, _
      ByVal MaxRecord As Integer, _
      ByVal strTable As String)
        Try
        'Khai báo và khởi tạo đối tượng SqlCommand
          Dim myCom As New SqlCommand
        'Gán thuộc tính
          myCom.Connection = myCon
          myCom.CommandType = _
            CommandType.StoredProcedure
          myCom.CommandText = strSP
        'Khai báo đối tượng SqlParameter
          Dim myPara As SqlParameter
        'Duyệt trên từng tham số
          For i As Integer = 0 To strPara.Length - 1
          'Khởi tạo đối tượng SqlParameter
            myPara = New SqlParameter(strPara(i), _
              strValue(i))
            myCom.Parameters.Add(myPara)
          Next
        'Khai báo và khởi tạo đối tượng SqlDataAdapter
```

```
        Dim myData As New SqlDataAdapter(myCom)
    'Điền dữ liệu vào đối tượng DataSet
        myData.Fill(myDS, _
        StartRecord, MaxRecord, strTable)
    Catch ex As Exception
        strError = ex.Message
    End Try
End Sub
```

Trong trường hợp lấy ra một đối tượng ArrayList, bạn có thể sử dụng đối tượng SqlDataReader thay vì dùng đối tượng DataTable như sau:

```
Public Function GetArrayList( _
  ByVal strTable As String, _
  ByVal strName As String, _
  ByVal strValue As String) As ArrayList
    Dim strSQL As String
    'Định nghĩa phát biểu SQL
    strSQL = "select distinct " + strValue
    strSQL += "," + strName + " from " + strTable
    'Khai báo và khởi tạo đối tượng ArrayList
    Dim arrValue As New ArrayList
    Dim myCom As SqlCommand
    Dim RD As SqlDataReader
    Dim cls As clsItem
    Try
      'Khởi tạo đối tượng SqlCommand
      myCom = New SqlCommand(strSQL, myCon)
      RD = myCom.ExecuteReader
      'Duyệt trên từng mẩu tin
      While RD.Read
      'Khởi tạo đối tượng clsItem
        cls = New clsItem( _
        Convert.ToString(RD.GetValue(0)), _
        RD.GetString(1))
      'Thêm đối tượng clsItem vào ArrayList
        arrValue.Add(cls)
      End While
      strError = "OK"
    Catch ex As Exception
      strError = ex.Message
```

```
            RD.Close()
        Finally

        End Try
        Return arrValue
    End Function
```

Ngoài ra, bạn cũng có thể khai báo phương thức dùng để lấy ra một đối tượng ứng với một hàng dữ liệu bằng cách sử dụng phương thức GetValues của đối tượng SqlDataReader như sau:

```
Public Function GetObject(ByVal strTable As
String, _
  ByVal strName As String, _
  ByVal strValue As String) As Object()
    ' Khai báo phát biểu SQL
    Dim strSQL As String
    strSQL = "select distinct " + strValue
    strSQL += "," + strName + " from " + strTable
    Dim myCom As SqlCommand
    Dim RD As SqlDataReader
    ' Khai báo mảng đối tượng
    Dim objs As Object()
    Try
      myCom = New SqlCommand(strSQL, myCon)
      RD = myCom.ExecuteReader
      ' Nếu tồn tại mẩu tin
      If RD.Read Then
      ' Dùng phương thức GetValues để lấy ra mảng đối tượng
        RD.GetValues(objs)
        RD.Close()
      End If
    Catch ex As Exception
      strError = ex.Message
    End Try
    Return objs
End Function
```

Sau khi khai báo thành công các phương thức và thuộc tính, sử dụng tiện ích StrongName để tạo ra khóa nhận dạng duy nhất của Assembly và khai báo chúng trong tập tin AssemblyInfo.vb (tham khảo cách tạo Shared Assembly trong cuốn "*Ví dụ và bài tập Visual Basic .Net – Lập trình hướng đối tượng*").

Tiếp theo, bạn biên dịch Project ra thành tập tin DLL, nếu muốn lớp này xuất hiện trong ngăn .NET mỗi khi tham chiếu bằng cách sử dụng chức năng Add Reference thì bạn chép chúng vào thư mục C:\WINDOWS\Microsoft.NET\Framework\v1.1.4322.

Ngoài ra, khi sử dụng tiện ích GacUtil.exe để đăng ký DLL vào GAC, bạn nên tham khảo cách đăng ký Assembly vào GAC trong cuốn *"Ví dụ và bài tập Visual Basic .Net – Lập trình hướng đối tượng"*.

Chú ý, chúng tôi giới thiệu cách tổng quát để xây dựng lớp tương tác với cơ sở dữ liệu, bạn có thể tham khảo chi tiết về cách cài đặt lớp này trên tầng giữa (Middle tier) trong cuốn *"Ví dụ và bài tập Visual Basic .Net – Chuyên đề COM + và MTS"*.

2. Tạo *Class* và đặt tên *clsStaffs*, kế đến bạn khai báo thuộc tính, phương thức để tương tác với cơ sở dữ liệu.

   Giả sử, chúng ta có bảng dữ liệu *clsStaffs* với cấu trúc như sau:

```
CREATE TABLE tblStaffs
(
  Code char(5) not null primary key,
  FirstName nvarchar(30) not null,
  LastName nvarchar(30) not null,
  Address nvarchar(50) not null,
  DOB smalldatetime,
  Gender bit default 0,
  JoinDate smalldatetime default getdate()
)
```

   Bằng cách thêm Class vào Project, tham chiếu đến lớp SQLDatabase và khai báo các biến tương ứng với các cột dữ liệu như sau:

```
Imports Bai2_1
Public Class clsStaffs
  Private shFlag As Short
  Private strCode As String
  Private strFirstName As String
  Private strLastName As String
  Private strAddress As String
  Private strDOB As String
  Private bolGender As Boolean
  Private strJoinDate As String
  Private cls As SQLDatabase
End Class
```

Lưu ý, biến shFlag tương ứng với tham số trong thủ tục spStaffs, dùng để phân biệt 4 hành động chính xử lý trên dữ liệu.

Kế đến, bạn khai báo thuộc tính cho phép gán và lấy giá trị của mỗi biến cục bộ vừa khai báo ở trên. Chẳng hạn, chúng ta có các biến strCode, strFirstName, strLastName và strAddress đều là kiểu chuỗi, bằng cách kiểm tra chiều dài ứng với chiều dài của chúng trong bảng tblStaffs, bạn khai báo như sau:

```
' Khai báo thuộc tính ứng với biến strCode
Public Property Code() As String
    Get
      Return strCode
    End Get
    Set(ByVal Value As String)
      If Value.Length > 5 Then
        Value = Left(Value, 5)
      End If
      strCode = Value
    End Set
End Property

' Khai báo thuộc tính ứng với biến strFirstName
Public Property FirstName() As String
    Get
      Return strFirstName
    End Get
    Set(ByVal Value As String)
      If Value.Length > 30 Then
        Value = Left(Value, 30)
      End If
      strFirstName = Value
    End Set
  End Property
' Khai báo thuộc tính ứng với biến strLastName
Public Property LastName() As String
    Get
      Return strLastName
    End Get
    Set(ByVal Value As String)
      If Value.Length > 30 Then
        Value = Left(Value, 30)
      End If
      strLastName = Value
    End Set
```

```
End Property
'Khai báo thuộc tính ứng với biến strAddress
Public Property Address() As String
    Get
      Return strAddress
    End Get
    Set(ByVal Value As String)
      If Value.Length > 50 Then
        Value = Left(Value, 50)
      End If
      strAddress = Value
    End Set
End Property
```

Đối với hai biến DOB và JoinDate, chúng có kiểu dữ liệu là SmallDateTime trong cơ sở dữ liệu SQL Server. Tuy nhiên, bạn nên khai báo chúng bằng chuỗi để nhận và chuyển chúng sang định dạng chuỗi dd/mmm/yyyy.

```
'Khai báo thuộc tính ứng với biến strDOB
Public Property DOB() As String
    Get
      Return strDOB
    End Get
    Set(ByVal Value As String)
      If Value.Length > 11 Then
        Value = Left(Value, 11)
      End If
      strDOB = Value
    End Set
End Property
'Khai báo thuộc tính ứng với biến strJoinDate
Public Property JoinDate() As String
    Get
      Return strJoinDate
    End Get
    Set(ByVal Value As String)
      If Value.Length > 11 Then
        Value = Left(Value, 11)
      End If
      strJoinDate = Value
    End Set
End Property
```

Ví dụ, khi người sử dụng nhập giá trị thời gian là 15/12/2005, bạn chuyển chúng thành 15/Dec/2005 rồi truyền vào cơ sở dữ liệu, giá trị thời gian là 2005-12-15.

Lưu ý, bằng cách này bạn có thể làm việc với giá trị kiểu thời gian độc lập với định dạng của mọi cơ sở dữ liệu.

Riêng trường hợp biến Gender, bên trong có kiểu dữ liệu là bit (tương ứng với 0 và 1), bạn có thể khai báo thuộc tính có kiểu dữ liệu Boolean như sau:

```
Public Property Gender() As Boolean
    Get
      Return bolGender
    End Get
    Set(ByVal Value As Boolean)
      If Value <> 1 And Value <> 0 Then
        Value = True
      End If
      bolGender = Value
    End Set
End Property
```

Ngoài ra, bạn cần khai báo thuộc tính cho biến bolFlag để nhận 1 trong 4 giá trị là 0,1,2,3 tương ứng với 4 hành động xử lý dữ liệu Select, Insert, Update và Delete.

```
Public Property Flag() As Short
    Get
      Return shFlag
    End Get
    Set(ByVal Value As Short)
      If Value > 3 And Value < 0 Then
        Value = 0
      End If
      shFlag = Value
    End Set
End Property
```

Sau khi khai báo các thuộc tính, bạn tiếp tục khai báo thủ tục trong cơ sở dữ liệu để thực hiện các chức năng như Insert, Delete, Update, Select dựa vào tham số @flag.

```
CREATE proc spStaffs
  @flag tinyint,
```

```
    @Code char(5),
    @FirstName nvarchar(30) ,
    @LastName nvarchar(30),
    @Address nvarchar(50),
    @DOB char(11),
    @Gender bit
as
    set nocount on
    declare @IsExit int
    set @IsExit =1
    set nocount on
    -- Liệt kê danh sách nhân viên
    if @flag=0
      select * from tblStaffs
      where 1=1
      and (case @Code
           when '' then @Code else Code end) =@Code
      and (case @FirstName
           when '' then @FirstName
           else FirstName end) = @FirstName
      and (case @LastName
           when '' then @LastName
           else LastName end) = @LastName
      and (case @Address
           when '' then @Address
           else Address end) = @Address
      and (case @DOB when '' then @DOB else
           convert(char(11),DOB,103) end) =@DOB
    else
      begin
      -- Thêm mới nhân viên
        if @flag=1
          if not exists(select * from tblStaffs
          where Code=@Code)
          insert into tblStaffs(Code, FirstName
          ,LastName,Address,DOB,Gender)
          values(@Code,@FirstName,@LastName,
          @Address,cast(@DOB as smalldatetime),
          @Gender)
        else
          set @IsExit =0
      -- Cập nhật nhân viên
      if @flag=2
        update tblStaffs
```

```
        set FirstName=@FirstName,
        LastName=@LastName,
        Address=@Address,
        DOB=cast(@DOB as smalldatetime),
        Gender=@Gender
        where Code=@Code
    -- Xóa nhân viên theo mã
    if @flag=3
        delete from tblStaffs
        where Code=@Code
    select @IsExit
end

set nocount off
```

Kế đến, bạn có thể kiểm tra thủ tục này bằng tiện ích Query Analyzer như sau:

```
spStaffs 0,'','','','','',''
```

Sau đó, bạn khai báo phương thức cho đối tượng clsStaffs như Insert, Delete, Update bằng cách sử dụng phương thức ExecuteSP của lớp clsDatabase như sau:

```
Public Function Execute(ByRef records As
Integer) As String
    Dim strError As String
    Try
        ' Khởi tạo đối tượng clsDatabase
        cls = New SQLDatabase
        ' Nếu kết nối cơ sở dữ liệu thành công
        If cls.OpenConnection() = "OK" Then
        ' Khai báo mảng giá trị
            Dim Values() As String = _
                {shFlag, strCode, strFirstName, _
                strLastName, strAddress, _
                strDOB, IIf(bolGender, "1", "0")}
            ' Gọi phương thức ExecuteSP đối tượng clsDatabase
            cls.ExecuteSP( _
                "spStaffs", Paras, Values, records)
            strError=cls.strError
            cls.CloseConnection()
        End If
    Catch ex As Exception
```

```
            strError = ex.Message
        End Try
        Return strError
    End Function
```

Trong đó, mảng tham số tương ứng với tham số trong thủ tục nội tại của SQL Server được khai báo trong phần đầu của lớp.

```
' Khai báo mảng tham số
Dim Paras() As String = _
        {"@flag", "@Code", "@FirstName", _
        "@LastName", "@Address", _
        "@DOB", "@Gender"}
```

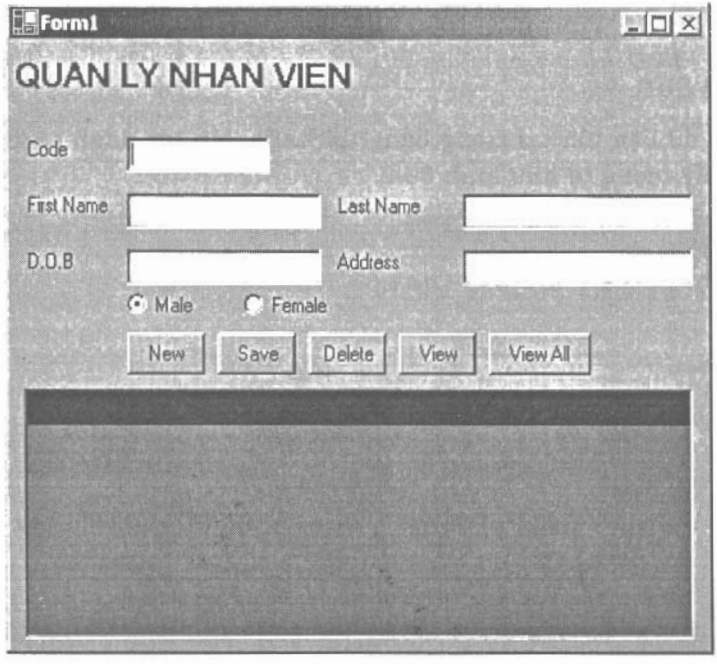
Đối với trường hợp truy vấn dữ liệu, bạn khai báo phương thức GetData với cấu trúc như sau:

```
Public Function GetData( _
  ByRef records As DataTable) As String
    Dim strError As String = ""
    Try
      ' Khởi tạo đối tượng clsDatabase
      cls = New SQLDatabase
      If cls.OpenConnection() = "OK" Then
      ' Nếu kết nối cơ sở dữ liệu thành công
        ' Khai báo mảng giá trị
        Dim Values() As String = _
        {shFlag, strCode, strFirstName, _
        strLastName, strAddress, _
        strDOB, IIf(bolGender, "1", "0")}
      ' Gọi phương thức GetValue của đối tượng clsDatabase
        cls.getValue(records, "spStaffs", _
        Paras, Values)
        cls.CloseConnection()
        strError = "OK"
      End If
    Catch ex As Exception
      strError = ex.Message
    End Try
    Return strError
End Function
```

Lưu ý, các giải pháp trên chỉ mang tính tham khảo, trong thực tế bạn có thể sử dụng nhiều cách để xây dựng lớp tương tác với cơ sở dữ liệu (Data Access).

3. Viết ứng dụng *Form*, cho phép người sử dụng tương tác với cơ sở dữ liệu thông qua lớp *clsStaffs* cùng với lớp *clsDatabase* trong *SQLDatabase*.

Để sử dụng clsStaffs, trước tiên bạn thiết kế Form bao gồm các điều khiển TextBox, RadioButton, DataGrid và Button như hình 30-1.



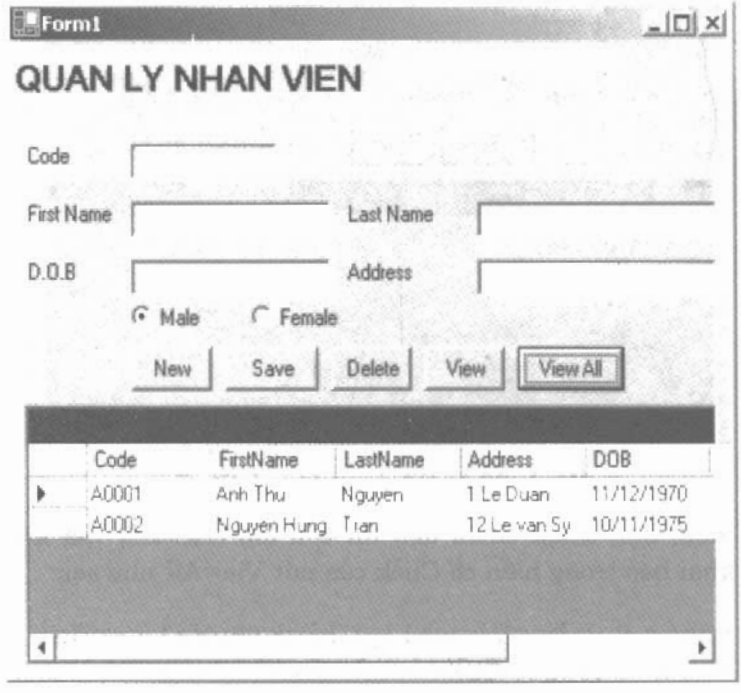
Hình 30-1: *Sử dụng lớp clsStaffs*

Kế đến, bạn tham chiếu đến dll của lớp clsStaffs vừa khai báo ở trên, rồi khai báo trong biến cố Click của nút ViewAll như sau:

```
Private Sub btnViewAll_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles btnViewAll.Click
    ' Khai báo và khởi tạo đối tượng clsStaffs
    Dim cls As New clsStaffs
    ' Gán giá trị cho các thuộc tính đối với trường hợp truy vấn
    cls.Flag = 0
    cls.Code = ""
    cls.FirstName = ""
    cls.LastName = ""
    cls.Address = ""
    cls.DOB = ""
```

```
        cls.Gender = 0
        'Khai báo và khởi tạo đối tượng DataTable
        Dim dtData As New DataTable
        'Gọi phương thức GetData để khởi tạo đối tượng clsStaffs
        If cls.GetData(dtData) = "OK" Then
          Me.DataGrid1.DataSource = dtData
        End If
    End Sub
```

Nếu dữ liệu tồn tại trong bảng tblStaffs, lập tức danh sách mẩu tin sẽ trình bày tương tự như hình 30-2.



Hình 30-2: *Truy vấn dữ liệu*

Trong trường hợp có lọc theo giá trị nhập trên các điều khiển TextBox, bạn khai báo trong biến cố Click của nút View như sau:

```
Private Sub btnView_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles btnView.Click
        'Khai báo và khởi tạo đối tượng clsStaffs
        Dim cls As New clsStaffs
```

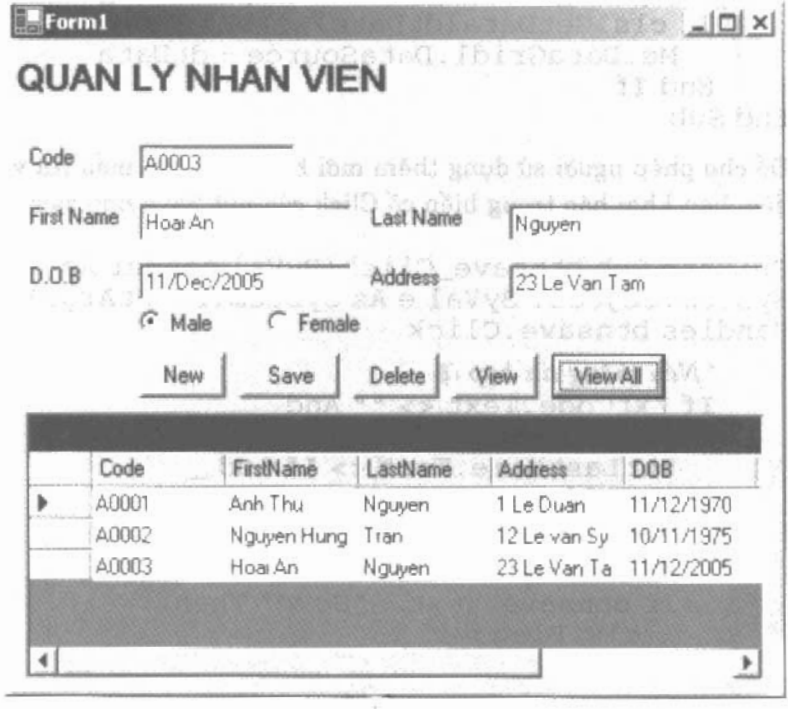
```
        ' Gán giá trị cho các thuộc tính đối với trường hợp truy vấn
        cls.Flag = 0
        cls.Code = txtCode.Text
        cls.FirstName = txtFirstName.Text
        cls.LastName = txtLastName.Text
        cls.Address = txtAddress.Text
        cls.DOB = txtDOB.Text
        cls.Gender = IIf(rdMale.Checked, 1, 0)
        ' Khai báo và khởi tạo đối tượng DataTable
        Dim dtData As New DataTable
        ' Gọi phương thức GetData để khởi tạo đối tượng clsStaffs
        If cls.GetData(dtData) = "OK" Then
          Me.DataGrid1.DataSource = dtData
        End If
    End Sub
```

Để cho phép người sử dụng thêm mới hay cập nhật mẩu tin vào cơ sở dữ liệu, bạn khai báo trong biến cố Click của nút Save như sau:

```
    Private Sub btnsave_Click(ByVal sender As
    System.Object, ByVal e As System.EventArgs)
    Handles btnsave.Click
        ' Nếu thông tin hợp lệ
        If txtCode.Text <> "" And _
          txtFirstName.Text <> "" And _
          txtLastName.Text <> "" And _
          txtAddress.Text <> "" Then
          Dim cls As New clsStaffs
          ' Trường hợp thêm mới mẩu tin
          If btnsave.Text = "Save" Then
            cls.Flag = 1
          Else
          ' Trường hợp cập nhật mẩu tin
            cls.Flag = 2
          End If
          ' Khai báo thuộc tính
          cls.Code = txtCode.Text
          cls.FirstName = txtFirstName.Text
          cls.LastName = txtLastName.Text
          cls.Address = txtAddress.Text
          cls.DOB = txtDOB.Text
          cls.Gender = IIf(rdMale.Checked, 1, 0)
          Dim i As Integer
```

```
            ' Gọi phương thức Execute
            Dim str As String = cls.Execute(i)
            If str ="OK" Then
              MsgBox("Added or Updated")
            End If
        End If
    End Sub
```

Giả sử, bạn nhập dữ liệu vào các điều khiển TextBox, nhấn nút Save, sau đó nhấn tiếp nút ViewAll, lập tức danh sách mẩu tin trình bày như hình 30-3.



Hình 30-3: *Thêm mẩu tin*

Ngoài ra, bạn khai báo trong biến cố Click của nút Delete để cho phép người sử dụng xóa mẩu tin đang kích hoạt.

```
Private Sub btnDelete_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles btnDelete.Click
    ' Nếu tồn tại mã nhân viên
    If txtCode.Text <> "" Then
```

```
        'Khai báo và khởi tạo đối tượng clsStaffs
        Dim cls As New clsStaffs
        'Xóa mẩu tin
        cls.Flag = 3
        cls.Code = txtCode.Text
        cls.FirstName = ""
        cls.LastName = ""
        cls.Address = ""
        cls.DOB = ""
        cls.Gender = 0
        Dim i As Integer
        'Nếu xóa thành công
        If cls.Execute(i) = "OK" Then
          MsgBox("Deleted")
        End If
      End If
End Sub
```

Lưu ý, nên gọi phương thức btnViewAll_Click ngay sau khi thêm hay cập nhật mẩu tin thành công, đồng thời bạn gọi phương thức btnNew_Click để xóa thông tin trên màn hình.

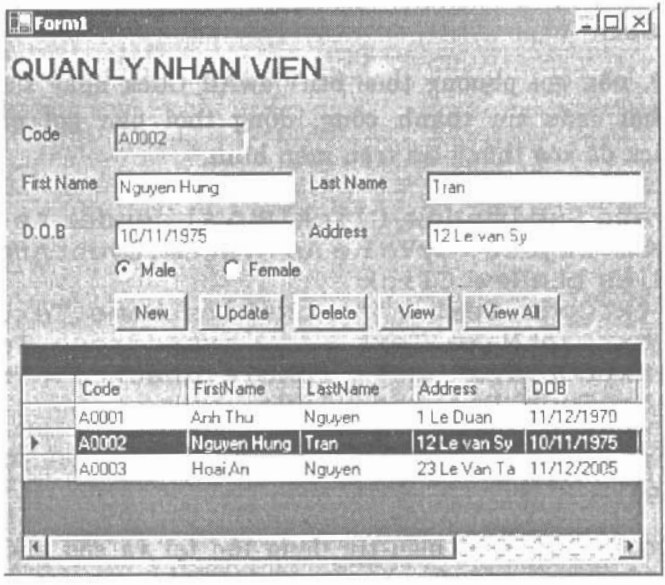
```
Private Sub btnNew_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles btnNew.Click
    txtCode.Text = "" : txtFirstName.Text = ""
    txtLastName.Text = "" : txtAddress.Text = ""
    txtDOB.Text = "" : rdMale.Checked = True
    txtCode.Enabled = True
    btnSave.Text = "Save"
    btnDelete.Enabled = False
End Sub
```

Nếu muốn kích hoạt mẩu tin đang tồn tại và cho phép người sử dụng thay đổi thông tin của nhân viên, mỗi khi chọn vào điều khiển DataGrid, bạn khai báo như sau:

```
Private Sub DataGrid1_Click(ByVal sender As
Object, ByVal e As System.EventArgs) Handles
DataGrid1.Click
    With DataGrid1
      txtCode.Text = _
      .Item(DataGrid1.CurrentRowIndex, 0)
      txtFirstName.Text = _
      .Item(DataGrid1.CurrentRowIndex, 1)
      txtLastName.Text = _
```

```
            .Item(DataGrid1.CurrentRowIndex, 2)
            txtAddress.Text = _
            .Item(DataGrid1.CurrentRowIndex, 3)
            txtDOB.Text = _
            .Item(DataGrid1.CurrentRowIndex, 4)
            rdMale.Checked = _
            .Item(DataGrid1.CurrentRowIndex, 5)
        End With
        txtCode.Enabled = False
        btnsave.Text = "Update"
        btnDelete.Enabled = True
    End Sub
```

Giả sử, bạn chọn vào mẩu tin thứ hai trên điều khiển DataGrid, lập tức thông tin của nhân viên này trình bày trên các điều khiển như hình 30-4.



Hình 30-4: *Cập nhật mẩu tin*

4. Xây dựng lớp dùng chung cho mọi trường hợp tương tác với cơ sở dữ liệu.

Để thực hiện ví dụ này, trước tiên bạn tạo Project loại Class Library đặt tên là Database, tạo Class thứ nhất đặt tên clsStaffs có cấu trúc tương tự như sau:

```
Public Class clsStaffs
```

```
    'Khai báo các biến cục bộ
    Private shFlag As Short
    Private strCode As String
    Private strFirstName As String
    Private strLastName As String
    Private strAddress As String
    Private strDOB As String
    Private bolGender As Boolean
    Private strJoinDate As String
    'Khai báo mảng các tham số trong thủ tục
    Public Paras() As String = _
          {"@flag", "@Code", "@FirstName", _
          "@LastName", "@Address", _
          "@DOB", "@Gender"}
'Khai báo phương thức trả về mảng giá trị
Public Function GetValue() As String()
    Dim Values() As String = _
    {shFlag, strCode, strFirstName, _
    strLastName, strAddress, _
    strDOB, IIf(bolGender, "1", "0")}
    Return Values
End Function
'Khai báo các thuộc tính cho từng biến cục bộ
Public Property Code() As String
    Get
      Return strCode
    End Get
    Set(ByVal Value As String)
      If Value.Length > 5 Then
        Value = Left(Value, 5)
      End If
      strCode = Value
    End Set
End Property
...
```

Tương tự như vậy, bạn cũng khai báo lớp clsCustomers bao gồm các biến thuộc tính và mảng tham số như sau:

```
Public Class clsCustomers
  Private shFlag As Short
  Public strCustomerID As String = ""
  Public strCustomerName As String = ""
  Public strAddress As String = ""
```

```
    Public strCity As String = ""
    Public myPara() As String = _
      {"@flag", "@CustomerID", _
      "@CustomerName", "@Address", "@City"}

  Public Function GetValue() As String()
      Dim Values() As String = _
      {shFlag, strCustomerID, strCustomerName, _
      strAddress, strCity}
      Return Values
  End Function

  Public Property CustomerID() As String
      Get
        Return strCustomerID
      End Get
      Set(ByVal Value As String)
        If Value.Length > 5 Then
          Value = Left(Value, 5)
        End If
        strCustomerID = Value
      End Set
  End Property
  ...
```

Lưu ý, bạn có thể khai báo danh sách các tham số @flag, @Code, @FirstName, @LastName, @Address, @DOB, @Gender bằng tập tin định dạng Xml, rồi khai báo phương thức để đọc tham số này từ tập tin Xml mỗi khi khởi tạo đối tượng clsStaffs.

Sau đó, bạn biên dịch Project này thành tập tin DLL có tên Database.

Tiếp tục, tạo mới Project loại Class Library và đặt tên Common, bạn có thể khai báo các thuộc tính và phương thức dùng để gọi các phương thức thực thi phát biểu SQL, thủ tục nội tại hay truy vấn dữ liệu trong lớp SQLDatabase (tham chiếu đến SQLDatabase.dll).

```
Private cls As SQLDatabase
' Mảng tham số
Private Values() As String
' Mảng giá trị tương ứng tham số
Private Paras() As String
' Tên thủ tục
```

```
Private strSP As String
```

Kế đến, bạn khai báo thuộc tính ứng với 3 biến cục bộ này như sau:

```
' Thuộc tính mảng giá trị tương ứng tham số
Public Property ArrValue() As String()
    Get
      Return Values
    End Get
    Set(ByVal Value As String())
      Values = Value
    End Set
End Property
' Thuộc tính ứng với tên thủ tục
Public Property StoreProcedure() As String
    Get
      Return strSP
    End Get
    Set(ByVal Value As String)
      strSP = Value
    End Set
End Property
' Mảng tham số của thủ tục nội tại
Public Property ArrPara() As String()
    Get
      Return Paras
    End Get
    Set(ByVal Value As String())
      Paras = Value
    End Set
End Property
```

Để cho phép thêm, cập nhật và xóa dữ liệu của một bảng nào đó trong cơ sở dữ liệu SQL Server, bạn khai báo phương thức Execute như sau:

```
Public Function Execute(ByRef records As
Integer) As String
    Dim strError As String
    Try
      ' Khởi tạo đối tượng clsDatabase
      cls = New SQLDatabase
      ' Nếu kết nối cơ sở dữ liệu thành công
      If cls.OpenConnection() = "OK" Then
        ' Gọi phương thức ExecuteSP đối tượng clsDatabase
```

```
            cls.ExecuteSP( _
              strSP, Paras, Values, records)
            strError=cls.strError
            cls.CloseConnection()
          End If
        Catch ex As Exception
          strError = ex.Message
        End Try
        Return strError
    End Function
```

Đối với trường hợp truy vấn dữ liệu, bạn khai báo phương thức GetData với cấu trúc như sau:

```
    Public Function GetData( _
      ByRef records As DataTable) As String
        Dim strError As String = ""
        Try
          ' Khởi tạo đối tượng clsDatabase
          cls = New SQLDatabase
          If cls.OpenConnection() = "OK" Then
          ' Nếu kết nối cơ sở dữ liệu thành công
          ' Gọi phương thức GetValue của đối tượng clsDatabase
            cls.getValue(records, strSP, _
            Paras, Values)
            cls.CloseConnection()
            strError = "OK"
          End If
        Catch ex As Exception
          strError = ex.Message
        End Try
        Return strError
    End Function
```

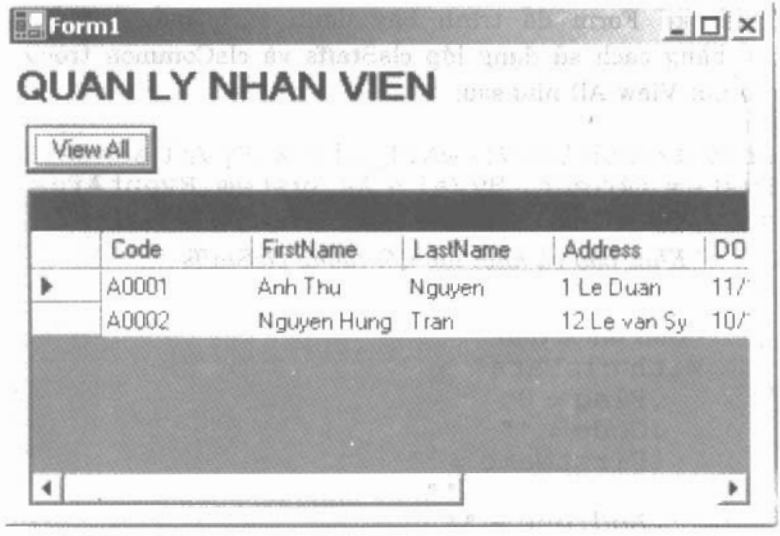
Để kiểm tra cách sử dụng không gian tên Database cùng với Common vào Project, bạn tạo mới ứng dụng Windows Forms và đặt tên Bai2-4, tham chiếu đến Common.dll và Database.dll rồi khai báo sử dụng:

```
    Imports Bai2_4
    Imports Database
```

Thiết kế Form để trình bày danh sách mẩu tin trong bảng tblStaffs, bằng cách sử dụng lớp clsStaffs và clsCommon trong biến cố Click của nút View All như sau:

```
Private Sub btnViewAll_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles btnViewAll.Click
    ' Khai báo và khởi tạo đối tượng clsStaffs
    Dim clsData As New clsStaffs
    ' Gán thuộc tính
    With clsData
       .Flag = 0
       .Code = ""
       .FirstName = ""
       .LastName = ""
       .Address = ""
       .DOB = ""
       .Gender = 0
    End With
    ' Khai báo và khởi tạo đối tượng clsCommon
    Dim cls As New clsCommon
    ' Gán thuộc tính tên thủ tục nội tại
    cls.StoreProcedure = "spStaffs"
    ' Gán thuộc tính mảng tham số và mảng giá trị
    With cls
       .ArrPara = clsData.Paras
       .ArrValue = clsData.GetValue
    End With
    ' Khai báo và khởi tạo đối tượng DataTable
    Dim dtData As New DataTable
    ' Lấy về tập dữ liệu
    If cls.GetData(dtData) = "OK" Then
      Me.DataGrid1.DataSource = dtData
    End If
End Sub
```

Khi thực thi chương trình, nếu kết nối thành công thì kết quả trả về như hình 30-5.



Hình 30-5: *Lớp sử dụng chung*

Chuyên đề 23:

KHÁM PHÁ CRYSTAL REPORT

1. THIẾT KẾ REPORT BẰNG CRYSTAL REPORT

1. Thiết kế *Report*, cho phép in ra danh sách nhân viên dạng *Mail Label*.

   Bài tập tự thiết kế.

2. Thiết kế *Report* trình bày dữ liệu với dạng lưới.

   Bài tập tự thiết kế.

3. Thiết kế *Report* dạng *Master-Details* ứng với hai bảng dữ liệu là *Products* và *Order Details*.

   Bài tập tự thiết kế.

2. TƯƠNG TÁC REPORT TỪ VISUAL BASIC.NET

1. Thiết kế *Form* và *Report*, cho phép trình bày dữ liệu của hai đối tượng *Table* có quan hệ cha con trong cơ sở dữ liệu *Northwind*.

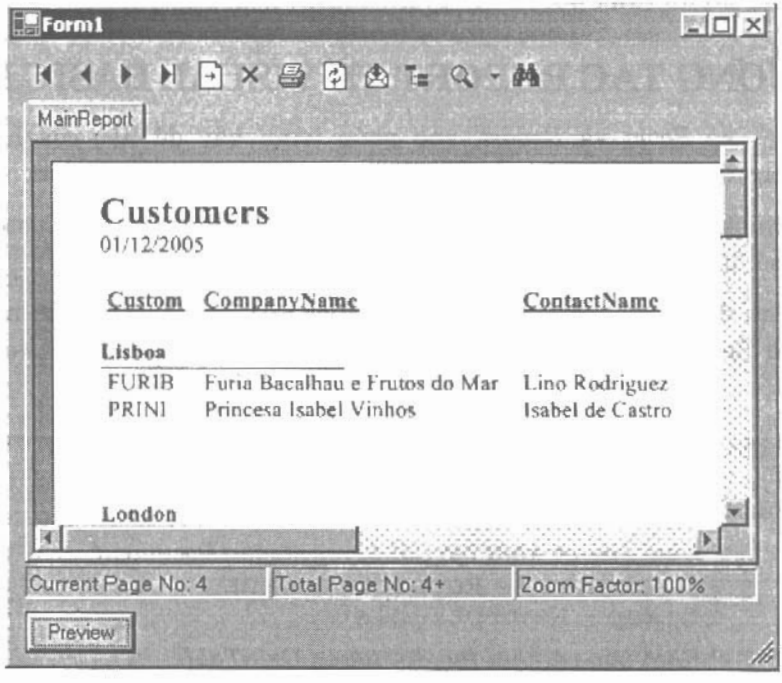
Thiết kế Report có tên rptTable dựa trên bảng dữ liệu Customers của cơ sở dữ liệu *Northwind*.

Sau đó, khai báo trong biến cố Click của nút Preview để nạp dữ liệu vào Report rồi trình bày trên điều khiển CrystalReportViewer như sau:

```
Private Sub Button1_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles Button1.Click
    ' Khai báo và khởi tạo đối tượng ReportDocument
    Dim objRep As ReportDocument
    objRep = New rptTable
    ' Khai báo và khởi tạo đối tượng DataTable
    Dim dt As New DataTable
    ' Khai báo và khởi tạo đối tượng clsDatabase
```

```
        Dim cls As New clsDatabase( _
        "server=" + gsServer + ";database=" + _
        gsDatabase + ";uid=" + gsUser + _
        ";pwd=" + gsPwd)
        ' Khai báo phát biểu SQL
        Dim strSQL As String = _
           "Select * from Customers"
        ' Gọi phương thức GetValue của đối tượng clsDatabase
        cls.getValue(dt, strSQL)
        ' Gán đối tượng DataTable cho ReportDocument
        objRep.SetDataSource(dt)
        ' Gán đối tượng ReportDocument cho điều khiển
        ' CrystalReportViewer
        CRV.ReportSource = objRep
    End Sub
```

Khi thực thi chương trình, nếu người sử dụng nhấn nút Preview, lập tức danh sách khách hàng trình bày như hình 31-1.



Hình 31-1: *Sử dụng đối tượng DataTable*

2. Tạo một thủ tục nội tại nhận tham số là *Country*, kế đến thiết kế *Form* và *Report*, cho phép trình bày dữ liệu của đối tượng *Store Procedure* này trong cơ sở dữ liệu *Northwind* với tham số truyền vào từ bên ngoài.

Khai báo thủ tục nội tại dùng để truy vấn mẩu tin trong bảng Customers thông qua cột Country như sau:

```
create proc spViewCustomer
  @Country varchar(20)
As
  select CustomerID,CompanyName,Address,City
  from Customers
  where Country=@Country
```

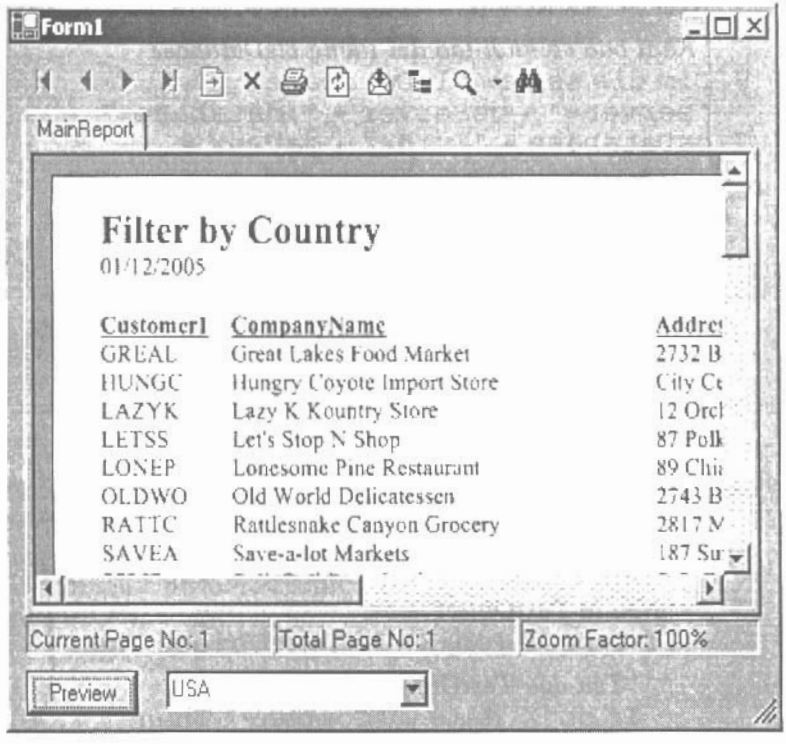
Bạn khai báo liệt kê danh sách Country trên điều khiển ComboBox trong biến cố Load của Form như sau:

```
Private Sub Form1_Load(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles MyBase.Load
    ' Khai báo và khởi tạo đối tượng clsDatabase
    Dim cls As New clsDatabase( _
        "server=" + gsServer + ";database=" + _
        gsDatabase + ";uid=" + _
      gsUser + ";pwd=" + gsPwd)
    ' Khai báo và khởi tạo đối tượng DataTable
    Dim dt As New DataTable
    ' Gọi phương thức GetArrayList
    cbCountry.DataSource = _
    cls.GetArrayList("Customers", _
      "Country", "Country")
    cbCountry.DisplayMember = "Name"
    cbCountry.ValueMember = "Value"
End Sub
```

Kế đến, bạn khai báo trong biến cố Click của nút Preview để nạp Report có tên rptStoreProcedure như sau:

```
Private Sub Button1_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles Button1.Click
    ' Khai báo đối tượng ReportDocument
    Dim objRep As ReportDocument
    ' Khởi tạo đối tượng ReportDocument
    objRep = New rpStoreProcedure
    ' Khai báo và khởi tạo đối tượng ConnectionInfo
    Dim myCon As New ConnectionInfo
    ' Khai báo và khởi tạo đối tượng TableLogOnInfo
    Dim myInfo As New TableLogOnInfo
    ' Định nghĩa phát biểu SQL
    Dim myTable As String = "spViewCustomer"
    ' Khai báo thuộc tính kết nối cơ sở dữ liệu
    With myCon
       .ServerName = gsServer
       .DatabaseName = gsDatabase
       .UserID = gsUser
       .Password = gsPwd
    End With
    ' Gán đối tượng kết nối vào thuộc tính ConnectionInfo
    myInfo.ConnectionInfo = myCon
    ' Khai báo giá trị cho tham số trong thủ tục
    objRep.SetParameterValue( _
       "@Country", cbCountry.Text)
   objRep.Database.Tables(0).ApplyLogOnInfo( _
        myInfo)
    CRV.ReportSource = objRep
    CRV.Refresh()
    objRep.Dispose()
End Sub
```

Khi thực thi chương trình, kết quả trình bày danh sách khách hàng của USA như hình 30-2.



Hình 31-2: *Sử dụng chuỗi tham số*

3. Thiết kế *Form* và *Report*, cho phép trình bày dữ liệu của đối tượng *Table* trong cơ sở dữ liệu *Northwind*. Đồng thời, bạn trình bày thông tin của công ty trên *Report* từ *Visual Basic.NET*.

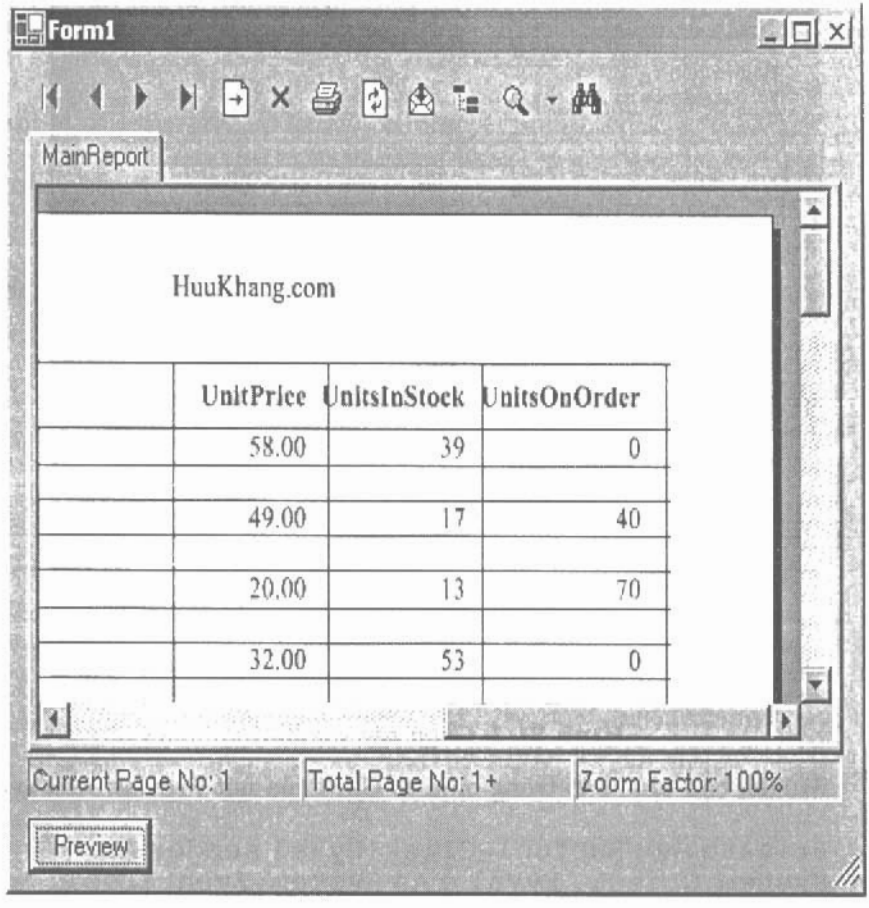
Trước tiên bạn thiết kế Report từ bảng dữ liệu có tên Products, thêm TextBox và đặt tên Company trên phần Page Header.

Bằng cách khai báo và sử dụng đối tượng DataTable, bạn có thể trình bày dữ liệu của bảng Products trên điều khiển CrystalReportViewer với tên công ty truyền từ Form.

```
Private Sub Button1_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles Button1.Click
    ' Khai báo và khởi tạo đối tượng ReportDocument
    Dim objRep As ReportDocument
```

```
    objRep = New rptTable
    'Khai báo và khởi tạo đối tượng DataTable
    Dim dt As New DataTable
    'Khai báo và khởi tạo đối tượng clsDatabase
    Dim cls As New clsDatabase( _
    "server=" + gsServer + ";database=" + _
    gsDatabase + ";uid=" + gsUser + _
     ";pwd=" + gsPwd)
    'Khai báo phát biểu SQL
    Dim strSQL As String = _
       "Select * from Customers"
    'Gọi phương thức GetValue của đối tượng clsDatabase
    cls.getValue(dt, strSQL)
    'Gán đối tượng DataTable cho ReportDocument
    objRep.SetDataSource(dt)
    'Duyệt trên từng Object của CrystalReport
    Dim objX As ReportObject
    For Each objX In _
       objRep.ReportDefinition.ReportObjects
       'Nếu là TextObject
       If TypeOf (objX) Is TextObject Then
          'Tên của TextObject là Company
          If objX.Name = "Company" Then
             Dim objText As TextObject = _
             CType(objX, xy.TextObject)
             'Gán chuỗi giá trị
             objText.Text = "HuuKhang.com"
          End If
       End If
    Next
    'Gán đối tượng ReportDocument cho điều khiển
    'CrystalReportViewer
    CRV.ReportSource = objRep
End Sub
```

Khi thực thi chương trình, nếu người sử dụng nhấn nút Preview, lập tức danh sách sản phẩm trong bảng Products trình bày trên CrystalReport cùng với tên công ty như hình 31-3.

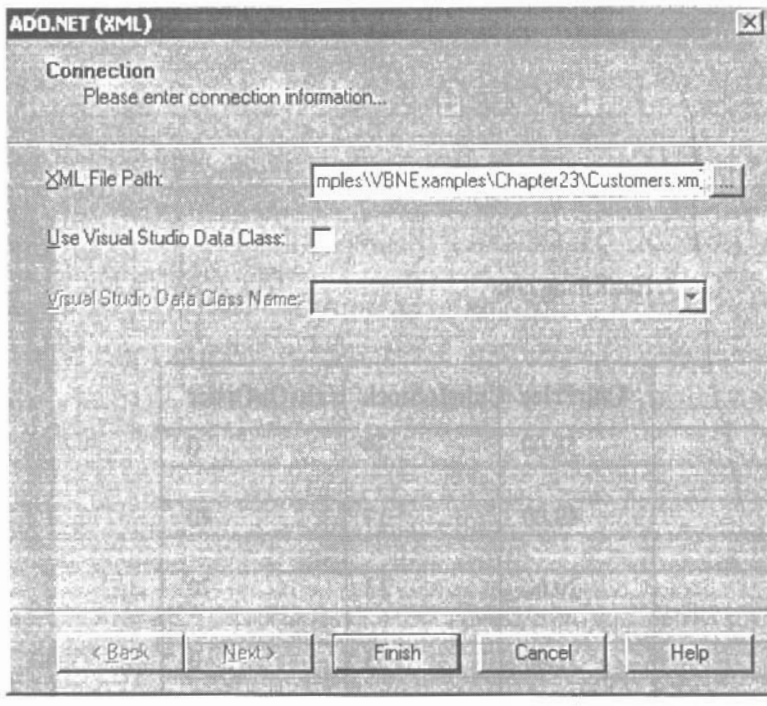


Hình 31-3: *Truyền chuỗi sang CrystalReport*

4 Xây dựng *Form*, trình bày dữ liệu của tập tin *XML* bằng *Crystal Report*.

Khi thiết kế Report, thay vì bạn chọn nguồn dữ liệu là OLE DB (ADO) thì bạn chọn vào ngăn More Data Source | ADO.NET (XML), lập tức cửa sổ xuất hiện yêu cầu bạn chọn tập tin Xml như hình 31-4.

Sau khi chọn tập tin Xml, lập tức tên của bảng dữ liệu xuất hiện trong ngăn More Data Source | ADO.NET (XML) rồi thao tác các bước kế tiếp tương tự như các ví dụ trên.



Hình 31-4: *Chọn tập tin Xml*

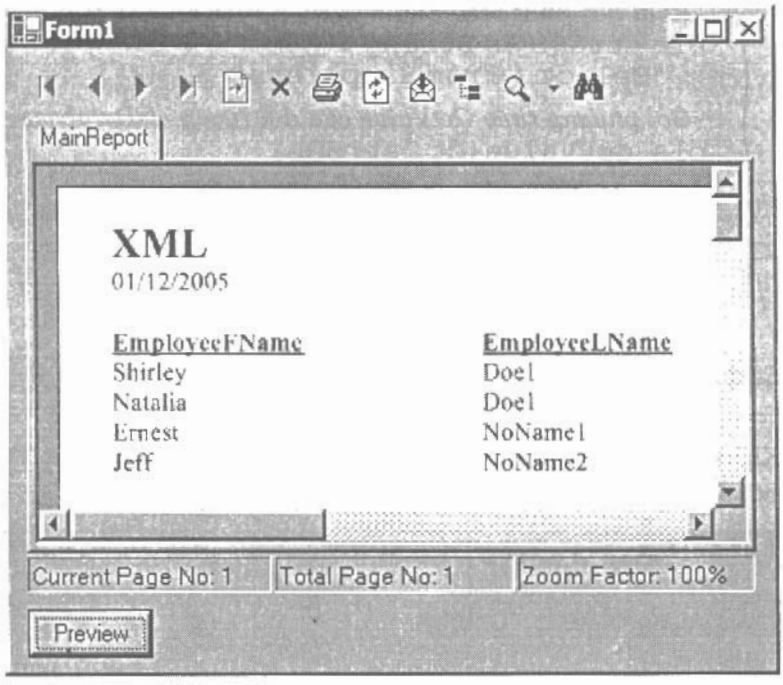
Sau đó, bạn khai báo trong biến cố Click của nút Preview như sau:

```
Private Sub Button1_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles Button1.Click
    ' Khai báo và khởi tạo đối tượng ReportDocument
    Dim objRep As ReportDocument
    objRep = New rptXML
    ' Gán đối tượng ReportDocument cho điều khiển
    ' CrystalRepoerViewer
    CRV.ReportSource = objRep
End Sub
```

Lưu ý, bạn khai báo sử dụng không gian tên của đối tượng ReportDocument trên phần đầu của Class:

```
Imports CrystalDecisions.CrystalReports.Engine
```

Khi thực thi chương trình, lập tức danh sách dữ liệu có trong tập tin Xml xuất hiện như hình 31-5.



Hình 31-5: *Dữ liệu Xml*

3. XUẤT DỮ LIỆU RA ĐỊNH DẠNG KHÁC

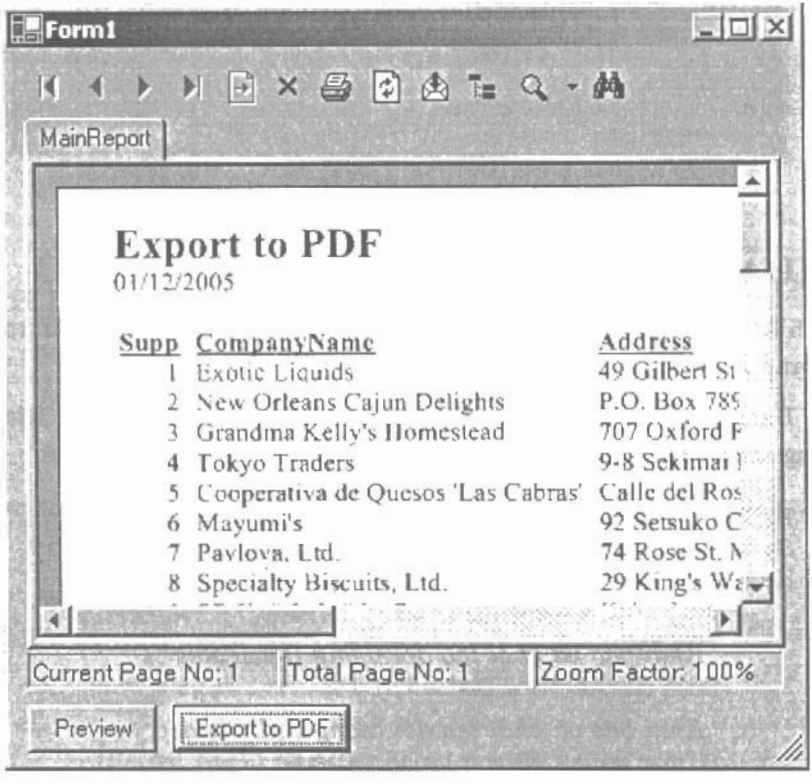
1. Thiết kế *Form*, trình bày dữ liệu từ thủ tục nội tại và xuất ra định dạng *PDF*.

Bằng cách sử dụng đối tượng DataTable, bạn có thể khai báo đoạn chương trình trong biến cố Click của nút Preview như sau:

```
Private Sub Button1_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles Button1.Click
    objRep = New rptPDF
    ' Khai báo và khởi tạo đối tượng DataTable
    Dim dt As New DataTable
    ' Khai báo và khởi tạo đối tượng clsDatabase
    Dim cls As New clsDatabase( _
    "server=" + gsServer + ";database=" + _
```

```
        gsDatabase + ";uid=" + gsUser + _
         ";pwd=" + gsPwd)
        ' Khai báo phát biểu SQL
        Dim strSQL As String = _
           "Select * from Suppliers"
        ' Gọi phương thức GetValue của đối tượng clsDatabase
        cls.getValue(dt, strSQL)
        ' Gán đối tượng DataTable cho ReportDocument
        objRep.SetDataSource(dt)
        ' Gán đối tượng ReportDocument cho điều khiển
        ' CrystalReportViewer
        CRV.ReportSource = objRep
End Sub
```

Khi thực thi chương trình, nếu người sử dụng nhấn nút Preview, lập tức danh sách nhà cung cấp trình bày như hình 31-6.



Hình 31-6: *Danh sách nhà cung cấp*

Khi người sử dụng nhấn nút Export to PDF, lập tức Adobe Reader sẽ được kích hoạt, dữ liệu của bảng Suppliers sẽ xuất hiện như hình 31-7.

Hình 31-7: *Xuất dữ liệu ra định dạng PDF*

Để làm điều này, bạn khai báo đoạn chương trình sử dụng đối tượng DiskFileDestinationOptions và ExportOptions như sau.

```
Private Sub btnExport_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles btnExport.Click
    Try
        ' Khai báo tên tập tin cần xuất ra
        Dim Filename As String = _
           "C:\Suppliers.pdf"
        ' Khai báo đối tượng DiskFileDestinationOptions
        Dim myDisk As DiskFileDestinationOptions
        ' Khởi tạo đối tượng DiskFileDestinationOptions
        myDisk = New DiskFileDestinationOptions
        ' Khai báo đối tượng ExportOptions
        myDisk.DiskFileName = Filename
```

```
        Dim myOption As ExportOptions
        myOption = objRep.ExportOptions
        'Khai báo thuộc tính cho đối tượng ExportOptions
        With myOption
           .DestinationOptions = myDisk
           .ExportDestinationType = _
             ExportDestinationType.DiskFile
           .ExportFormatType = _
             ExportFormatType.PortableDocFormat
        End With
        'Gọi phương thức Export của đối tượng ReportDocument
        objRep.Export()
        Process.Start(Filename)
      Catch ex As Exception
        MsgBox(ex.Message)
      End Try
  End Sub
```

Lưu ý, bạn cần khai báo biến kiểu đối tượng ReportDocument dùng chung như sau:

```
'Khai báo và khởi tạo đối tượng ReportDocument
Dim objRep As ReportDocument
```

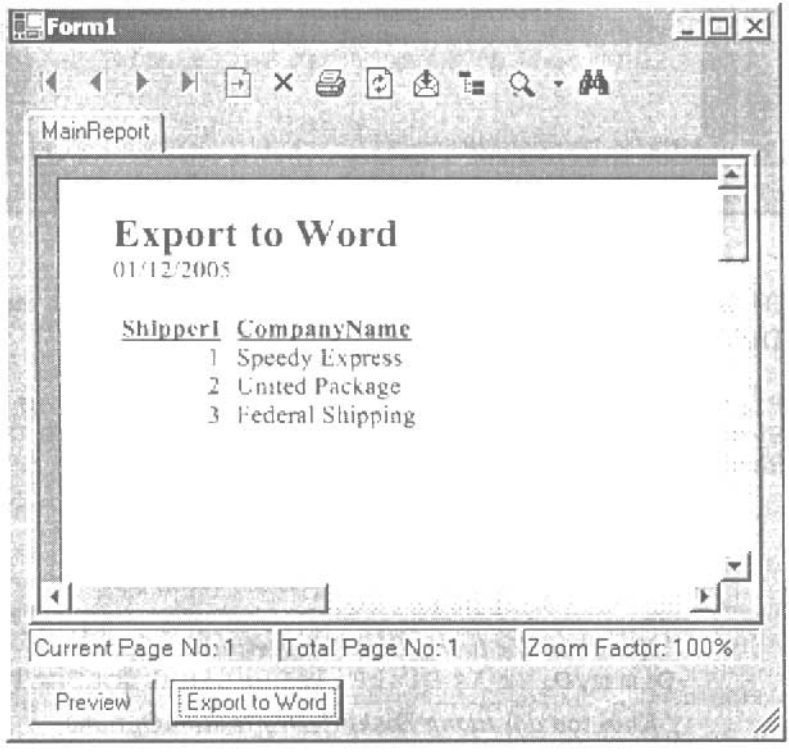
2. Thiết kế *Form*, trình bày danh sách bảng dữ liệu của cơ sở dữ liệu *Northwind* và xuất ra định dạng *Word*.

Tương tự như trên, trong trường hợp này bạn khai báo đoạn chương trình trong biến cố Click của nút Export to Word như sau:

```
Private Sub Button1_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles Button1.Click
    objRep = New rptWord
    'Khai báo và khởi tạo đối tượng DataTable
    Dim dt As New DataTable
    'Khai báo và khởi tạo đối tượng clsDatabase
    Dim cls As New clsDatabase( _
    "server=" + gsServer + ";database=" + _
    gsDatabase + ";uid=" + gsUser + _
     ";pwd=" + gsPwd)
    'Khai báo phát biểu SQL
```

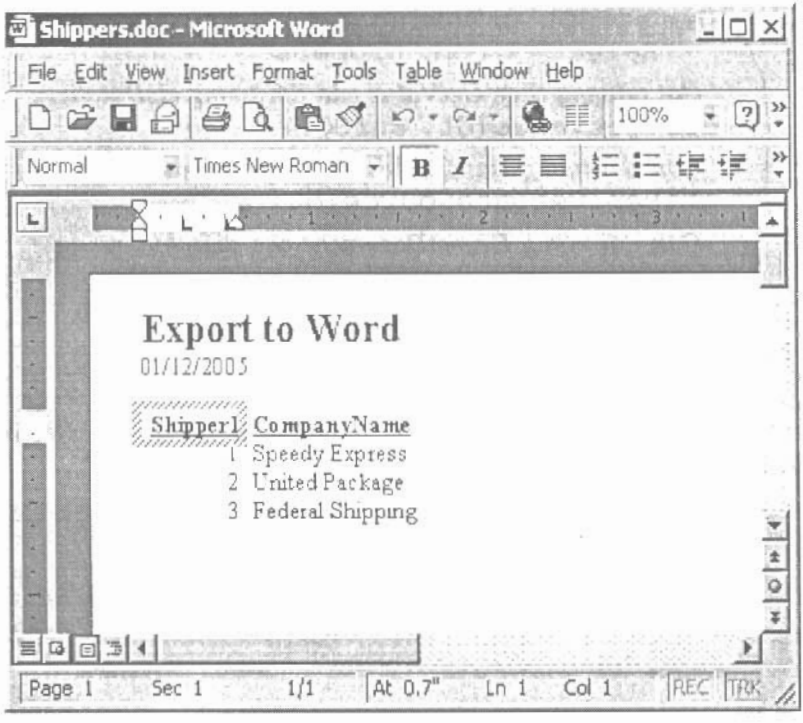
```
        Dim strSQL As String = _
            "Select * from Shippers"
        ' Gọi phương thức GetValue của đối tượng clsDatabase
        cls.getValue(dt, strSQL)
        ' Gán đối tượng DataTable cho ReportDocument
        objRep.SetDataSource(dt)
        ' Gán đối tượng ReportDocument cho điều khiển
        ' CrystalReportViewer
        CRV.ReportSource = objRep
    End Sub
```

Khi thực thi chương trình, nếu người sử dụng nhấn nút Preview, lập tức danh sách nhà vận chuyển trình bày như hình 31-8.



Hình 31-8: *Danh sách nhà vận chuyển*

Khi người sử dụng nhấn nút Export to Word, lập tức Microsoft Word sẽ được kích hoạt, dữ liệu của bảng Shippers sẽ xuất hiện như hình 31-9.



Hình 31-9: *Xuất dữ liệu ra định dạng Word*

Để làm điều này, bạn khai báo đoạn chương trình sử dụng đối tượng DiskFileDestinationOptions và ExportOptions như sau.

```
Private Sub btnExport_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles btnExport.Click
    Try
        ' Khai báo tên tập tin cần xuất ra
        Dim Filename As String = _
          "C:\Shippers.doc"
        ' Khai báo đối tượng DiskFileDestinationOptions
        Dim myDisk As DiskFileDestinationOptions
        ' Khởi tạo đối tượng DiskFileDestinationOptions
        myDisk = New DiskFileDestinationOptions
        ' Khai báo đối tượng ExportOptions
        myDisk.DiskFileName = Filename
        Dim myOption As ExportOptions
        myOption = objRep.ExportOptions
```

```
        'Khai báo thuộc tính cho đối tượng ExportOptions
        With myOption
          .DestinationOptions = myDisk
          .ExportDestinationType = _
            ExportDestinationType.DiskFile
          .ExportFormatType = _
            ExportFormatType.WordForWindows
        End With
        'Gọi phương thức Export của đối tượng ReportDocument
        objRep.Export()
        Process.Start(Filename)
      Catch ex As Exception
        MsgBox(ex.Message)
      End Try
  End Sub
```

Lưu ý, bạn cần khai báo biến kiểu đối tượng ReportDocument dùng chung như sau:

```
'Khai báo và khởi tạo đối tượng ReportDocument
Dim objRep As ReportDocument
```

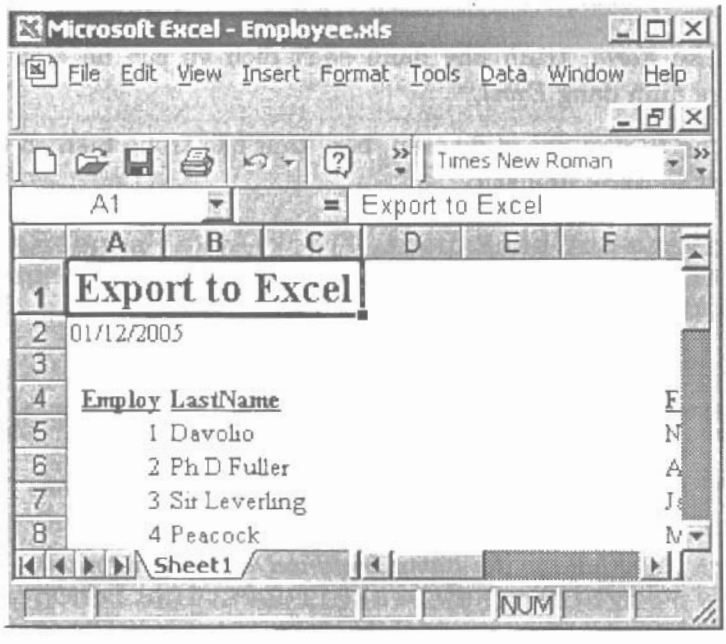
3. Thiết kế *Form*, trình bày danh sách dịch vụ của hệ điều hành và xuất ra định dạng *Excel*.

Tương tự như hai ví dụ trên, bạn khai báo trong biến cố Click của nút Export to Excel như sau:

```
Private Sub btnExport_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles btnExport.Click
    Try
      'Khai báo tên tập tin cần xuất ra
      Dim Filename As String = _
        "C:\Employees.xls"
      'Khai báo đối tượng DiskFileDestinationOptions
      Dim myDisk As DiskFileDestinationOptions
      'Khởi tạo đối tượng DiskFileDestinationOptions
      myDisk = New DiskFileDestinationOptions
      'Khai báo đối tượng ExportOptions
      myDisk.DiskFileName = Filename
      Dim myOption As ExportOptions
```

```
        myOption = objRep.ExportOptions
        'Khai báo thuộc tính cho đối tượng ExportOptions
        With myOption
          .DestinationOptions = myDisk
          .ExportDestinationType = _
            ExportDestinationType.DiskFile
          .ExportFormatType = _
            ExportFormatType.Excel
        End With
        'Gọi phương thức Export của đối tượng ReportDocument
        objRep.Export()
        Process.Start(Filename)
      Catch ex As Exception
        MsgBox(ex.Message)
      End Try
  End Sub
```

Khi thực thi chương trình, danh sách nhân viên trình bày trên điều khiển CrystalReportViewer, nếu bạn nhấn nút Export to Excel, lập tức Microsoft Excel được kích hoạt như hình 31-10.



Hình 31-10: *Xuất dữ liệu ra Excel*

VÍ DỤ & BÀI TẬP VISUAL BASIC.NET LẬP TRÌNH CƠ SỞ DỮ LIỆU & REPORT

NHÀ XUẤT BẢN LAO ĐỘNG – XÃ HỘI
41B Lý Thái Tổ – Hà Nội – Tel: 8.241706 – Fax. 9.348283

Chịu trách nhiệm xuất bản: NGUYỄN ĐÌNH THIÊM
Chịu trách nhiệm nội dung: NGUYỄN BÁ NGỌC
Biên soạn: PHẠM HỮU KHANG – HOÀNG ĐỨC HẢI
Sửa bản in: NGỌC AN
Trình bày bìa: HUỲNH THẢO

Thực hiện liên doanh: Công ty TNHH Minh Khai S.G
E-mail: mk.book@minhkhai.com.vn Website: www.minhkhai.com.vn

Tổng phát hành

- Nhà sách Minh Khai: 249 Nguyễn Thị Minh Khai - Quận 1 - TP.HCM
  ĐT: (08) 9.250.590 – 9.250.591. Fax: (08) 8.331.124
- Nhà sách Minh Châu: Nhà 30 -Ngõ 22 -Ta Quang Bửu - Bách Khoa -Hà Nội
  ĐT: (04) 8.692.785 Fax: (04) 8.683.995

Đại lý các khu vực

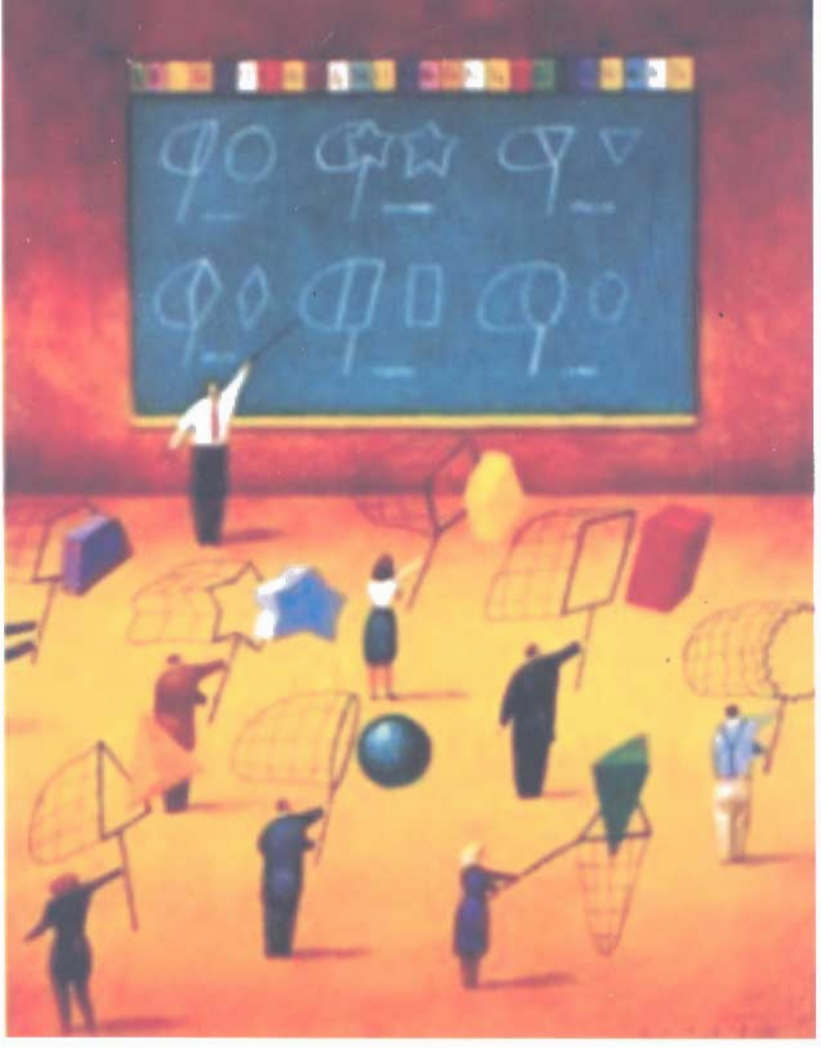
- Nhà sách Huy Hoàng: 95 Núi Trúc - Kim Mã - Ba Đình -Hà Nội
  ĐT: (04) 7.365.859
- Cty cổ phần sách thiết bị trường học Đà Nẵng: 78 Bạch Đằng - Đà Nẵng
  ĐT: 0511.837100
- Nhà sách Chánh Trí: 116A Nguyễn Chí Thanh - Đà Nẵng
  ĐT: 0511.820129
- Cty phát hành sách Khánh Hòa:
  - Nhà sách Ponagar: 73 Thống Nhất - Nha Trang -Khánh Hòa
    ĐT: 058.822636
  - Siêu Thị Sách Tân Tiến - 11 Lê Thành Phương - Nha Trang - Khánh Hòa
    ĐT: 058.827303
- Nhà sách Năm Hiển: 79/6 Xô Viết Nghệ Tĩnh – TP.Cần Thơ
  ĐT: 071. 821668

In 4.000 cuốn (kèm đĩa CD bài tập), khổ 16 x 24 cm,
tại Xí nghiệp in Machinco - Số 21 Bùi Thị Xuân, Q.1, TP.HCM

Giấy chấp nhận đăng ký kế hoạch xuất bản số 38-2006/CXB/124-194/LĐXH

Mã số $\frac{124-194}{30-12}$

In xong và nộp lưu chiểu quý 1 năm 2006.



- Giới thiệu ADO.NET.
- Đối tượng Connection, Command.
- Sử dụng đối tượng SqlParameter.
- Làm việc với đối tượng DataReader.
- Lưu trữ và thao tác dữ liệu với đối tượng DataSet.
- Ghi và đọc dữ liệu XML.
- Cập nhật dữ liệu bằng đối tượng DataAdapter.
- Đối tượng DataTable, DataView.
- Đối tượng DataRow, DataColumn, DataRelation.
- Điều khiển DataGrid và DataBindings.
- Định dạng điều khiển DataGrid.
- Kết hợp ComboBox, CheckBox, TextBox trong điều khiển DataGrid.
- Lớp dữ liệu cho từng thực thể.
- Xây dựng lớp tương tác dữ liệu.
- Khám phá Crystal Report.
- Bài giải chi tiết của gần 300 bài tập .

MK.PUB
MinhKhai